Tạp chí

VNOI

Xuân Giáp Thìn

Quý độc giả thân mến,

Tiếp nối thành công của Tạp chí VNOI – Xuân Quý Mão, Câu lạc bộ Olympic Tin học Việt Nam cho ra mắt Tạp chí VNOI – Xuân Giáp Thìn như một món quà đầu xuân dành tặng cho cộng đồng Tin học Việt Nam nhân dịp Tết Giáp Thìn 2024.

Tạp chí VNOI ở lần thứ hai xuất bản được dựa trên nguồn cảm hứng từ tạp chí Pi của hội Toán học Việt Nam. Với mục đích truyền tải kiến thức, kinh nghiệm qua nhiều chủ đề, số tạp chí đặc biệt kì này sẽ mang đến cho các bạn những bài viết thú vị với nội dung cô đọng, phù hợp với nhiều đối tượng độc giả. Tạp chí bao gồm 3 bài viết học thuật xoay quanh những chủ đề trong Tin học và Lập trình thi đấu đến từ 3 tác giả, được chọn lọc từ rất nhiều bài viết được gửi về trên khắp mọi miền tổ quốc. Ngoài ra, tạp chí kì này sẽ có sự góp mặt của 4 vị khách mời đặc biệt trong 4 bài phỏng vấn độc quyền, hứa hẹn sẽ đem đến cho độc giả những góc nhìn đặc biệt đến từ cả ba thế hệ trong lịch sử hơn 35 năm phát triển của Tin học Việt Nam - "sơ khai – kế thừa – phát triển".

Trong sự nô nức, hân hoan chào đón xuân năm mới, ban biên tập nói riêng và tập thể VNOI nói chung xin chân thành gửi lời tri ân đến các khách mời, các tác giả, quý độc giả đã luôn dành sự quan tâm, theo dõi và ủng hộ tạp chí trong suốt thời gian qua.

Nhân dịp xuân Giáp Thìn 2024, tập thể VNOI thân chúc các bạn đọc một năm mới thật nhiều sức khỏe, hạnh phúc và đạt được thành công như mong đợi!

BAN BIÊN TẬP

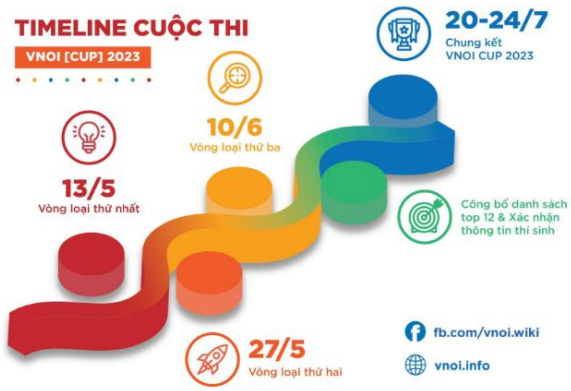
Mục lục

VNOI – Nhìn lại 2023

Được thành lập từ năm 2008, Câu lạc bộ Olympic Tin học Việt Nam - VNOI là cộng đồng của các học sinh, sinh viên có niềm đam mê với bộ môn lập trình thi đấu nói riêng và lĩnh vực công nghệ thông tin nói chung. Trải qua hơn 15 năm hoạt động, VNOI đã có nhiều đóng góp nổi bật cho nền tin học nước nhà. Đặc biệt, năm 2023 vừa qua là một năm đã để lại nhiều dấu ấn khó quên đối với Câu lạc bộ. Hãy cùng chúng mình điểm lại những hoạt động và thành tựu của VNOI trong năm vừa qua thông qua bài viết này nhé!

VNOI Cup 2023

Năm 2023 là năm thứ hai VNOI Cup được tổ chức. VNOI Cup 2023 đã quy tụ đông đảo các bạn thí sinh trên khắp cả nước tham gia, trong số đó có những thí sinh có profile rất "khủng". Chặng đua năm nay bao gồm 03 vòng loại online diễn ra trên nền tảng VNOJ; thông qua đó đã chọn ra 12 thí sinh xuất sắc nhất tham gia thi đấu Chung kết tại trường THPT Chuyên Hạ Long.



Lộ trình tổ chức VNOI CUP 2023

Ngay từ vòng loại đầu tiên, cuộc thi đã diễn ra vô cùng gay cấn và sôi động với 1369 thí sinh, là số lượng thí sinh lớn nhất trong tất cả các kỳ thi đã từng tổ chức trên hệ thống VNOJ. Vòng loại thứ hai và vòng loại thứ ba cũng đầy kịch tính khi thứ hạng trên bảng xếp hạng được thay đổi liên tục, những màn rượt đuổi điểm số diễn ra vô cùng căng thẳng và quyết liệt, không khí rất gay cấn và nghẹt thở. Sau ba vòng thi, 10 thí sinh xuất sắc xuất hiện tại vòng Chung kết bao gồm Trần Xuân Bách, Phạm Quốc Hùng, Lê Ngọc Bảo Anh, Nguyễn Tấn Sỹ Nguyên, Trương Văn Quốc Bảo, Nguyễn Vũ Đăng Huy, Vũ Hoàng Kiên, Lê Hoàng Nam, Nguyễn Đức Thắng và Đỗ Đình Đắc. Dựa trên đề cử từ cộng đồng, hai thí sinh còn lại góp mặt tại vòng Chung kết là Hoàng Ngọc Bảo Khuê và Nguyễn Chí Thanh.

Chung kết VNOI Cup 2023 tại Trường THPT Chuyên Hạ Long

Đồng hành với ICPC[1] và Olympic Tin học Sinh viên Việt Nam 2023

Đây là năm thứ ba VNOI được tin tưởng, nền tảng VNOJ được sử dụng làm nền tảng tổ chức ICPC Asia Regional (site thi tại Việt Nam). Đặc biệt, kỳ thi ICPC Asia Hue City 2023 đánh dấu lần đầu tiên máy ảo do VNOI phát triển được đưa vào sử dụng tại ICPC Vietnam. Đây là hệ thống máy thi được thiết kế bởi đội ngũ kỹ thuật của VNOI dựa trên hệ thống máy thi chính thức của ICPC World Finals[2] 2023, với một vài tính năng mới được phát triển riêng dành cho các kỳ thi ICPC tại Việt Nam, trong đó có thể kể đến:

Máy ảo sử dụng VPN, các máy thí sinh đều sẽ được kết nối tới hệ thống thi và các dịch vụ khác thuộc ICPC thông qua VPN, bao gồm trang thi chính thức, máy in, ...

Màn hình các thí sinh sẽ được ghi lại và phát trực tiếp khi cần thiết, phục vụ cho quá trình livestream, tái hiện khoảnh khắc của thí sinh, cũng

[1]ICPC: International Collegiate Programming Contest - Cuộc thi Lập trình Quốc tế lâu đời và danh giá nhất dành cho sinh viên các trường đại học và cao đẳng trên toàn cầu.

[2]World Finals: Cuộc thi ICPC cuối cùng trong một mùa giải bao gồm những đội xuất sắc nhất của các trường đại học trên thế giới vượt qua vòng loại vùng (Regional contest)

như hỗ trợ cho ban tổ chức, coach và khán giả theo dõi được từng bước đi của thí sinh trong giờ thi.

Đội ngũ VNOI và ông Nguyễn Long tại ICPC Asia Hue City 2023

VNOI rất vui và tự hào khi hệ thống máy thi đã hoạt động vô cùng trơn tru hiệu quả trong suốt thời gian diễn ra kỳ thi, từ đó mang lại trải nghiệm tốt đến với các bạn thí sinh trong và ngoài nước.

VNOI và các dự án cộng đồng 2023

Bedao Contest

Bedao Contest 2023

Bedao Contest là một dự án được thành lập vào đầu năm 2020 nhằm mục đích tổ chức các kỳ thi, tạo ra sân chơi cho những bạn trẻ yêu thuật toán và coding. Bedao và VNOI chính thức bắt tay vào tháng 8/2021 và sáp nhập như hiện nay.

Trong năm 2023, Bedao Contest tiếp tục phát huy tinh thần của dự án, các bài tập của Bedao ngày càng được đầu tư về cả chất lượng lẫn số lượng. Các contest của Bedao đa dạng chủ đề và độ khó, từ Bedao Mini Contest đến Bedao Regular Contest và Bedao Grand Contest. Đặc biệt nhất, Bedao Contest ra mắt series Bedao OI nhằm cho các thí sinh có trải nghiệm thi giống với các kỳ thi chính thức khác.

Dự án sinh test các kỳ thi chính thức

Dự án sinh test các kỳ thi chính thức được khởi động từ đầu năm 2022, tiếp nối Thư viện đề thi VNOI. Dự án này được triển khai với mục đích giúp các bạn ôn luyện và giải các đề thi từ các kỳ thi Tin học chính thức như Học sinh giỏi Quốc gia, Chọn đội tuyển Olympic, Olympic Truyền thống 30/4, Duyên hải Bắc bộ, Học sinh giỏi Thành phố/Tỉnh trên cả nước, . . .

Dự án sinh test các kỳ thi chính thức

Sau một thời gian dài vắng bóng, trong năm vừa qua, dự án đã quay trở lại với những đề thi chất lượng. Đặc biệt trong đó có thể kể đến đề thi Chọn đội tuyển Olympic năm 2018 và đề thi Chọn đội tuyển Olympic năm 2023. Một điều chưa có tiền lệ, đó là các bộ đề Chọn đội tuyển Olympic Tin học - VNTST[3] chưa bao giờ được đăng công khai, cũng như có bộ test hoàn chỉnh trên bất cứ nền tảng nào. Đây chính là điểm nổi bật nhất trong dự án này của VNOI.

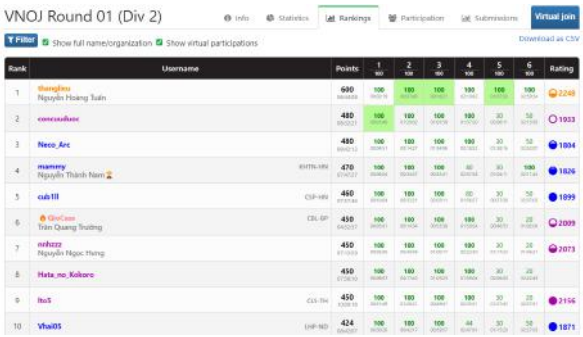
External Setters

VNOJ Round 01 - kỳ thi đầu tiên được tổ chức bởi các thành viên trong cộng đồng

[3]TST: Kỳ thi tuyển chọn đội tuyển dự thi Olympic Tin học Quốc tế, hay còn được biết đến là Vòng 2 thi chọn Học sinh giỏi Quốc gia.

Đây là một dự án tổ chức các kỳ thi chính thức (tính rating) trên VNOJ bên cạnh Bedao Contest, trong đó các bài tập sẽ do cộng đồng đóng góp hoặc Ban ra đề sẽ là các thành viên trong cộng đồng VNOI và có tài khoản trên hệ thống VNOJ.

Ngày 28/01/2024, VNOJ Round 01 đã được tổ chức; đánh dấu bước phát triển của dự án External Setters.



Bảng xếp hạng kỳ thi VNOJ Round 01

Có thể nói sự hưởng ứng, tham gia của hơn 600 thí sinh chính là "quả ngọt" của Ban ra đề, cũng là thành công bước đầu của dự án. Đây cũng chính là động lực để dự án External Setter tiếp tục được thực hiện, hứa hẹn sẽ có nhiều VNOJ Round hơn nữa.

Phát triển VNOJ

Trong năm 2023, để phục vụ cho nhu cầu luyện tập cho các kỳ thi HSGQG[4] , ICPC và Olympic Tin học Sinh Viên, VNOI đã nâng cấp công suất của máy chấm lên gấp đôi.

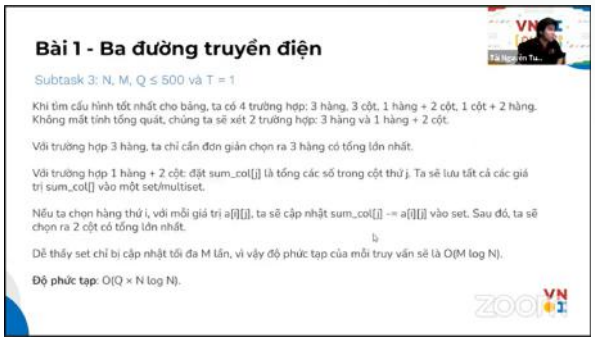
VNOI đã có đề nghị và được admin vjudge cho phép, hỗ trợ tích hợp VNOJ với Virtual Judge. Điều này cho phép các thầy cô cũng như các bạn học sinh, sinh viên có thể sử dụng các tính năng tuyệt vời, một trong số đó là việc mashup bài tập cho các training contests và luyện tập tích hợp các OJ nổi tiếng chỉ với một tài khoản.

VNOJ chính thức được tích hợp với VJudge nhằm hỗ trợ các bạn học sinh, sinh viên làm bài trên đa nền tảng với chỉ một tài khoản

[4]HSGQG: là viết tắt của kỳ thi Học sinh giỏi Quốc gia.

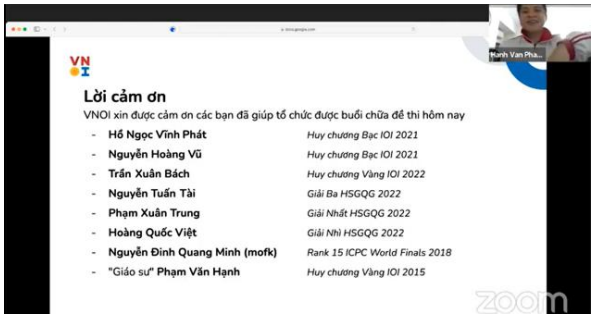
Chữa đề HSGQG 2023 – 2024

Trong ba năm trở lại đây, VNOI luôn duy trì việc chữa đề thi HSGQG môn Tin học, lời giải cho các bài tập được trình bày trong buổi chữa đề được dày công chuẩn bị bởi đội ngũ chuyên môn của VNOI.



Phần chữa bài 1 đề thi HSGQG 2024

Đặc biệt, các buổi chữa bài có sự góp mặt của dàn khách mời khủng với nhiều thành tích đáng nể trong giới lập trình thi đấu.



Đội ngũ khách mời giàu thành tích trong các buổi chữa bài HSGQG

VNOI Wiki

Giao diện mới của trang VNOI Wiki

Trong năm qua VNOI vẫn duy trì dự án VNOI Wiki với 18 bài viết chất lượng về các chủ đề trong Tin học đã được xuất bản bởi đội ngũ VNOI Wiki, nhiều kỹ thuật, thuật toán được đưa tới cộng đồng. Nổi bật nhất là việc nâng cấp giao diện VNOI Wiki giúp truyền tải hiệu quả hơn nội dung các bài viết cho cộng đồng.

VNOI Educational Contests

Bên cạnh đó là dự án Educational Contests nhằm tạo nên môi trường học tập lành mạnh cho các bạn trẻ đam mê Tin học ở Việt Nam, cho phép cộng đồng tiếp cận đa dạng các chủ đề với những bài tập vô cùng chất lượng giúp các bạn chuẩn bị thật tốt trước các kỳ thi

Các kỳ thi thuộc dự án Educational Contests

Tình nguyện viên tại VNOI

Để duy trì các dự án phục vụ cộng đồng, VNOI rất cần sự giúp đỡ của các bạn Tình nguyện viên trong mọi khía cạnh: Chuyên môn, kỹ thuật, truyền thông, ...Trong năm 2023, VNOI đã kết thúc và cấp giấy chứng nhận cho các bạn Tình nguyện viên thế hệ thứ 2, đồng thời chiêu mộ các bạn Tình nguyện viên thế hệ thứ 3.

Nhằm gắn kết tình cảm giữa Tình nguyện viên với nhau và với Câu lạc bộ, các hoạt động giao lưu trực tuyến được VNOI tổ chức thường xuyên. Hoạt động giao lưu trực tiếp cũng được tổ chức vào một số dịp đặc biệt tại hai thành phố lớn là Hà Nội và Thành phố Hồ Chí Minh.

VNOI tổ chức giao lưu Tình nguyện viên tại Hà Nội

Bên cạnh đó, VNOI thường xuyên tổ chức các buổi talkshow cho các bạn tình nguyện viên tham gia và tương tác các khách mời có tầm ảnh hưởng lớn trong giới lập trình nói riêng và ngành CNTT nói chung như Lê Yên Thanh (CEO Phenikaa MaaS), Nguyễn Thành Trung (tác giả của các bài viết Code cùng RR), Lê Xuân Mạnh (CTO Kyber Network), Lê Đôn Khuê (Technical Lead tại Be), ... Ngoài ra, các bạn tình nguyện viên còn có cơ hội kết nối và trao đổi riêng với các khách mời đặc biệt này.

VNOI tổ chức giao lưu Tình nguyện viên tại Thành phố Hồ Chí Minh

VNOI cũng chú trọng đến những phần thưởng cho những bạn tình nguyện viên đã có một kỳ làm việc đầy nỗ lực. Chính vì thế, trong năm vừa qua VNOI đã phát hành nhiều merchandise độc quyền như dây đeo, áo, móc khóa, sticker, bao lì xì, ... được trao tận tay đến các bạn tình nguyện viên nhân dịp trước Tết.

Những hoạt động của VNOI đều có dấu ấn đậm nét đến từ các bạn tình nguyện viên đều đã và đang âm thầm đóng góp chung cho nền Tin học nước nhà. Những dự án của VNOI sẽ không thành công nếu không có sự giúp sức và ủng hộ đến từ các bạn. VNOI hy vọng rằng tinh thần của Tình nguyện viên VNOI vẫn sẽ duy trì trong những năm tiếp theo.

Những món quà đặc biệt đến từ VNOI dành cho những bạn tình nguyện viên xuất sắc

VNOJ – Hệ thống chấm bài do người Việt, vì người Việt

Trong hai thập kỷ trở lại đây, phong trào Tin học đã và đang diễn ra sôi nổi trên khắp cả nước nhờ vào sự phổ cập Internet và sự phát triển của các hệ thống chấm bài trực tuyến, cho phép các bạn học sinh, sinh viên và những người có đam mê có thể luyện tập mọi lúc, mọi nơi.

Lịch sử phát triển

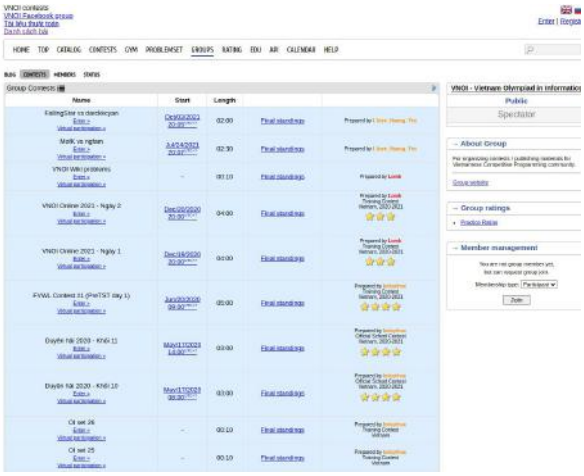
Quay trở lại năm 2007, đây là thời kỳ Internet phát triển mạnh tại Việt Nam, cùng khoảng thời gian đó trên thế giới xuất hiện nhiều Online Judge như SPOJ, UVa, ... Ấp ủ ý tưởng về một trình chấm trực tuyến cho các bạn học sinh, sinh viên Việt Nam, nhóm các sinh viên bao gồm Ngô Minh Đức, Khúc Anh Tuấn và Nguyễn Minh Hiếu đã lập ra VOJ với sự hỗ trợ của ban quản trị SPOJ và nằm dưới sự quản lý của Diễn đàn tin học Việt Nam (VNOI), hiện nay hệ thống VOJ vẫn còn hoạt động tại tên miền https://vn.spoj.com/

Các lượt nộp bài trên trang vn.spoj.com

Sau khi ra đời, VOJ là nơi lưu trữ dữ liệu về bài tập của nhiều cuộc thi quan trọng như ACM ICPC[5] , HSGQG[6] , ... và nhiều bài tập hay, bổ ích từ đóng góp của các thầy cô, các bạn học sinh, sinh viên Chuyên Tin.

Tuy nhiên, sau nhiều năm hoạt động, hệ thống VOJ dần thể hiện những điểm yếu so với các hệ thống mới, bao gồm việc khó sử dụng hơn, xuất hiện nhiều lỗi với các bài tập cũ, dẫn tới bài bị ẩn và không thể chấm được. Vì vậy vào ngày 01/04/2017, ban quản trị VNOI quyết định chuyển các bài tập từ VOJ sang hệ thống Codeforces[7] , một hệ thống chấm bài được phát triển và duy trì bởi Mikhail Mirzayanov cùng cộng sự.

Tới ngày 03/04/2020 việc tải các bài tập lên hệ thống Codeforces đã chính thức hoàn tất. Ngoài những bài đã có trên hệ thống VOJ, ban quản trị còn bổ sung thêm dữ liệu cho một số các kỳ thi như VOI 2017, VOI 2018, VNOI Online, ... Hiện nay nhóm Codeforces VNOI [8] vẫn còn trực tuyến, tuy nhiên các bài tập tại đây không còn được cập nhật, bổ sung.



Nhóm Codeforces VNOI vào thời điểm năm 2020

Ngày 28/02/2021, Đại hội thành lập Câu lạc bộ Olympic Tin học Việt Nam (VNOI) đã được diễn ra dưới sự tham dự và chỉ đạo của đại diện hội Tin học Việt Nam (VAIP), đánh dấu thời khắc VNOI trở thành một nhánh của Hội. Kế thừa nhiệm vụ của Diễn đàn Tin học, BCH Câu lạc bộ nhận thấy những bất cập khi sử dụng hệ thống chấm của nước ngoài: Khó sử dụng với người Việt (do giao diện tiếng Anh), khó quảng bá tới tất cả các bạn học sinh, ..., và quan trọng nhất là dữ liệu về các bài tập đều thuộc về các kỳ thi có quy mô tại Việt Nam. Mong muốn phát triển một hệ thống chấm bài trực tuyến của người Việt, cho người Việt và do người Việt, đồng thời cũng vì mục đích giữ dữ liệu quan trọng ở lại trong nước, BCH Câu lạc bộ đội

[5]ICPC: International Collegiate Programming Contest - Cuộc thi Lập trình Quốc tế lâu đời và danh giá nhất dành cho sinh viên các trường đại học và cao đẳng trên toàn cầu.

[6]HSGQG: là viết tắt của kỳ thi Học sinh giỏi Quốc gia.

[7]Codeforces: một trang web tổ chức các cuộc thi lập trình thi đấu với các dạng đề, bài tập đa dạng.

[8]https://codeforces.com/group/FLVn1Sc504/

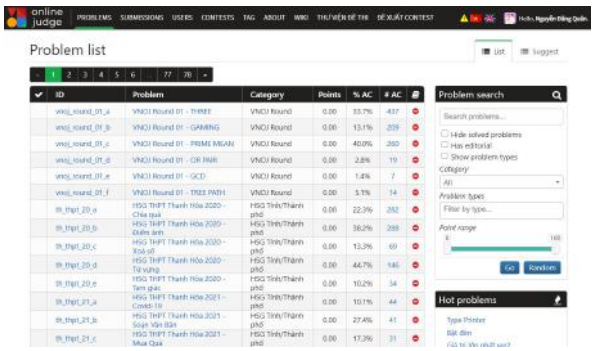
ngũ kỹ thuật của VNOI đã tìm hiểu, phát triển và cho ra mắt hệ thống VNOJ – VNOI Online Judge.

VNOJ

Hệ thống VNOJ – VNOI Online Judge được phát triển dựa trên DMOJ – một hệ thống chấm bài mã nguồn mở. Hiện nay, VNOJ vẫn đang hoạt động tại tên miền https://oj.vnoi.info, các bài tập và kỳ thi trên hệ thống vẫn được cập nhật thường xuyên, liên tục. Trong số đó, nhiều kỳ thi mang thương hiệu của VNOI đã được tổ chức thành công như Bedao Contest, VNOI CUP, ...

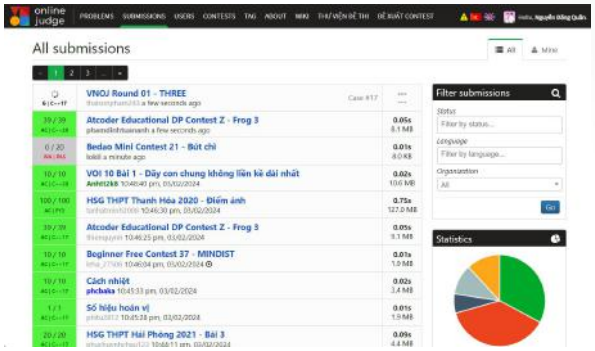
Các tính năng nổi bật của VNOJ

Được thiết kế theo "phong cách" Việt Nam dựa trên nền tảng DMOJ, hệ thống VNOJ có giao diện rất thân thiện và dễ sử dụng, người dùng có thể chuyển đổi qua lại giữa giao diện Tiếng Việt và Tiếng Anh một cách nhanh chóng. Hệ thống điểm số cũng được thiết kế lại để tạo thuận lợi cho người dùng trong việc tự theo dõi, tự đánh giá thông qua quá trình luyện tập.



VNOJ với giao diện Tiếng Anh

Với đội ngũ kỹ thuật có năng lực và tận tâm, VNOJ – VNOI Online Judge hoạt động vô cùng ổn định và bảo mật để phục vụ lượng lớn lượt sử dụng của cộng đồng. Đặc biệt trong năm 2023, VNOI đã nâng cấp công suất của máy chấm lên gấp đôi nhằm đáp ứng nhu cầu của đông đảo các bạn trẻ yêu lập trình trong quá trình ôn tập trước thềm các kỳ thi lớn như VOI, ICPC, và Olympic Tin học Sinh viên. Tính đến ngày 28/01/2024, VNOJ đã tiếp nhận và xử lí 4744496 lượt nộp bài, tương ứng khoảng 4688 lượt nộp bài trong một ngày.



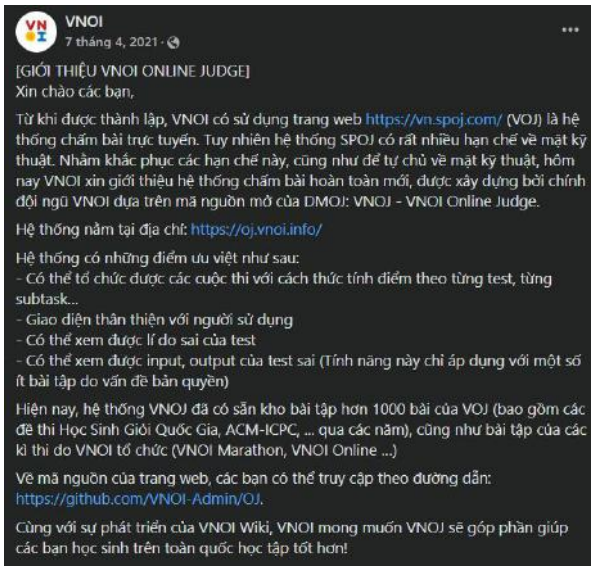
VNOJ xử lí lượng lớn lượt nộp bài trong một ngày

Vào tháng 9 năm 2021, khi dịch Covid 19 bùng phát mạnh mẽ tại Việt Nam, nhiều trường phải chuyển qua hình thức dạy online; VNOI đã phát triển và cho ra mắt tính năng Tổ chức. Tính năng này cho phép thầy cô các trường tạo bài tập và kì thi dành riêng cho học sinh của trường mình, đồng thời chức năng Tổ chức cũng cung cấp một bảng điểm nội bộ giúp các thầy cô có thể theo dõi quá trình luyện tập của học sinh. Ngoài ra, để tạo thuận lợi cho các thầy cô, VNOJ hỗ trợ tạo kỳ thi theo nhiều chuẩn, trong đó có định dạng của phần mềm Themis, phần mềm chấm thi chính thức của Bộ Giáo dục và Đào tạo.

Tính năng tổ chức của VNOJ

Một số dấu mốc quan trọng

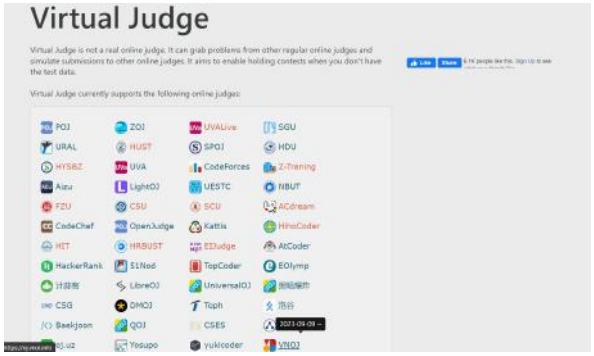
Tháng 02/2021, VNOJ – VNOI Online Judge chính thức ra mắt cộng đồng Tin học Việt Nam. VNOI công bố, giới thiệu VNOJ vào ngày 07/04/2021 trên fanpage của Câu lạc bộ.



Công bố VNOJ

Năm tháng sau, vào tháng 07/2021, VNOI và Free Contest chính thức hợp tác, theo đó tài nguyên của các kỳ Free Contest sẽ được tải lên VNOJ để phục vụ nhu cầu luyện tập sau khi thi của các bạn thí sinh. Cùng khoảng thời gian ấy, vào tháng 08/2021, VNOI và Bedao chính thức hợp tác, các kỳ thi của Bedao sẽ được tổ chức trên nền tảng VNOJ – VNOI Online Judge dưới sự hỗ trợ của đội ngũ kỹ thuật và đội ngũ chuyên môn của VNOI.

Trong năm vừa qua, với sự hỗ trợ của Xu Han (admin Virtual Judge), VNOI đã tích hợp VNOJ với Virtual Judge. Điều này nhằm hướng đến các tính năng tuyệt vời mà các thầy cô cũng như các bạn học sinh, sinh viên có thể sử dụng, bao gồm việc tạo các tập bài tập đến từ các OJ nổi tiếng chỉ với một tài khoản. Bên cạnh đó, điều này cũng chứng tỏ khát vọng đưa phong trào Tin học của Việt Nam ra quốc tế của VNOI.



VNOJ tích hợp vào VJudge ngày 09/09/2023

Được hội Tin học Việt Nam tin tưởng giao nhiệm vụ, trong ba năm trở lại đây, VNOJ luôn được lựa chọn là hệ thống chấm bài chính thức cho các kỳ thi ICPC và Olympic Tin học Sinh viên, đặc biệt là ICPC Quốc gia và ICPC Khu vực. Là đơn vị duy nhất cung cấp phần mềm hệ thống thi, Câu lạc bộ VNOI tự hào đã hoàn thành tốt nhiệm vụ, chưa từng xảy ra sự cố kỹ thuật trong thời gian thi, từ đó góp phần vào sự thành công của ICPC Khu vực và khẳng định được năng lực tổ chức của nước chủ nhà đăng cai.

VNOI CUP 2023 - Thử thách bản lĩnh, khám phá tài năng

Năm 2023 là năm thứ hai VNOI Cup được tổ chức. Tiếp nối sự thành công của mùa đầu tiên, VNOI Cup 2023 khởi động với mục đích tìm kiếm những tài năng lập trình trẻ Việt Nam. Trải qua ba vòng loại đầy cam go và vòng Chung kết vô cùng gay cấn, chủ nhân của chiếc cup vô địch VNOI Cup 2023 danh giá đã lộ diện. Hãy cùng điểm lại chặng đường tìm ra những gương mặt xuất sắc trong cuộc thi này nhé!

Đôi nét về VNOI Cup 2023

VNOI Cup 2023 là cuộc thi dành cho các cá nhân đang là học sinh, sinh viên, hoặc đã tốt nghiệp và đang theo học hoặc làm việc ở Việt Nam cũng như mọi nơi trên toàn thế giới. Đặc biệt, VNOI Cup 2023 không giới hạn độ tuổi của thí sinh tham dự.

Chặng đua bao gồm 03 vòng loại online diễn ra trên nền tảng VNOJ vào các ngày 13/05, 27/05 và 10/06; thông qua đó chọn ra 12 thí sinh xuất sắc nhất thi đấu Chung kết vào ngày 21, 22/07 tại trường THPT Chuyên Hạ Long (Quảng Ninh).

Bên cạnh những giải thưởng lên tới hàng chục triệu đồng tại vòng Chung kết, các bạn đạt thành tích xuất sắc ở tại các vòng loại còn có cơ hội nhận được 150 áo VNOI Cup 2023 được thiết kế độc quyền cho kì thi.

VNOI CUP 2023

Với tiêu chí tìm kiếm các coders tài năng trên khắp cả nước, VNOI CUP 2023 chờ đón những cá nhân có đam mê và năng lực để cùng nhau tạo nên một cuộc thi lập trình đầy sôi động và ý nghĩa. Cuộc thi đề cao tính cạnh tranh đến từ bộ đề, chính vì vậy để chuẩn bị những vòng thi chất lượng và đầy tính cạnh tranh, rất cần đến sự chuẩn bị miệt mài và tỉ mỉ của đội ngũ ra đề có trình độ chuyên môn cao.

Ban ra đề VNOI Cup 2023

Thống kê thành tích ban ra đề VNOI CUP 2023

- Đặng Đoàn Đức Trung: Trưởng ban ra đề VNOI CUP 2023 - ICPC[9] World Finalist 2023, 2022, 2021, 2020, Max 2871 Codeforces[10] ;
- Trần Quang Lộc: trưởng ban ra đề ICPC Asia HCMC 2022, Coordinator[11] Codeforces, Max 2546 Codeforces;
- Nguyễn Đinh Quang Minh: ICPC World Finalist 2018, 2021, Max 2726 Codeforces;
- Hồ Ngọc Vĩnh Phát: ICPC World Finalist 2023, 2022, HCB IOI[12] 2021, Max 2345 Codeforces;
- Phạm Tuấn Nghĩa: ICPC World Finalist 2022, 2020, Max 2412 Codeforces;
- Nguyễn Diệp Xuân Quang: ICPC World Finalist 2020, 2021, HCB APIO[13] 2017, Max 2446 Codeforces;
- Nguyễn Tuấn Tài: Max 2221 Codeforces, giải Ba HSGQG[14] 2022;

[9]ICPC: International Collegiate Programming Contest - Cuộc thi Lập trình Quốc tế lâu đời và danh giá nhất dành cho sinh viên các trường đại học và cao đẳng trên toàn cầu.

[10]Codeforces: một trang web tổ chức các cuộc thi lập trình thi đấu với các dạng đề, bài tập đa dạng.

[11]Coordinator: người tổ chức các contest cho OJ.

[12]IOI: Kỳ thi Olympic Tin học Quốc tế.

[13]APIO: Kỳ thi Olympic Tin học Châu Á -– Thái Bình Dương.

[14]HSGQG: là viết tắt của kỳ thi Học sinh giỏi Quốc gia.

- Nguyễn Hoàng Vũ: HCB IOI 2021, Max 2204 Codeforces;
- Phạm Xuân Trung: VNOI CUP 2022 Finalist, giải Nhất HSGQG 2022, Max 2259 Codeforces.

Ba vòng loại VNOI Cup 2023

Ngay từ vòng loại đầu tiên, cuộc thi đã diễn ra vô cùng gay cấn và sôi động với 1369 thí sinh, con số kỷ lục trong tất cả các kỳ thi đã từng tổ chức trên hệ thống VNOJ trước đó. Với kinh nghiệm chinh chiến từ nhiều cuộc thi lớn trong và ngoài nước, Trần Xuân Bách (fextivity) đã dành chiến thắng lội ngược dòng ngoạn mục qua việc AC toàn bộ 06 bài ở phút thứ 143, chỉ 07 phút trước khi vòng thi kết thúc. Sau vòng loại thứ nhất này, bên cạnh Trần Xuân Bách, gương mặt xuất sắc góp mặt tại vòng Chung kết chính là Phạm Quốc Hùng (hollwo_pelw). Đáng nói, Trần Xuân Bách và Phạm Quốc Hùng cũng là hai thí sinh từng góp mặt trong Chung kết VNOI Cup 2022.

Vòng loại thứ hai cũng đầy kịch tính khi thứ hạng trên bảng xếp hạng được thay đổi liên tục. Tại vòng này, hai chủ nhân xuất sắc của tấm vé tham dự vòng Chung kết đã gọi tên Lê Ngọc Bảo Anh (BaoJiaoPisu) và Nguyễn Tấn Sỹ Nguyên (flashmt). Cả hai đã có những màn rượt đuổi điểm số vô cùng căng thẳng và quyết liệt và cùng kết thúc với đồng số điểm 9500/12000.

Đến với vòng loại cuối cùng, vòng loại quyết định chủ nhân của 6/8 tấm vé còn lại tiến đến vòng Chung kết VNOI Cup 2023 tại Thành phố Hạ Long, không khí càng trở nên gay cấn và nghẹt thở hơn bao giờ hết. Sau những giờ phút thi đấu căng thẳng và quyết liệt, những tấm vé tiếp theo đã thuộc về 03 thí sinh dẫn đầu bảng xếp hạng vòng loại thứ ba - Trương Văn Quốc Bảo (BJoozz), Nguyễn Vũ Đăng Huy (Hollowed), Vũ Hoàng Kiên (BROnekton); cùng với đó là 03 thí sinh có xếp hạng cao nhất tại bảng tổng điểm ba vòng loại - Lê Hoàng Nam (noobcoder), Nguyễn Đức Thắng (marvinthang), Đỗ Đình Đắc (6aren).

Vòng Chung kết VNOI Cup 2023

Bên cạnh 10 thí sinh được chọn từ các vòng loại của cuộc thi, dựa trên đề cử từ cộng đồng, Ban tổ chức đã chọn ra 02 gương mặt xuất sắc cuối cùng tham dự vòng chung kết VNOI Cup 2023, chính là Hoàng Ngọc Bảo Khuê (Khuepr123) và Nguyễn Chí Thanh (flashhh).

Không khí phòng thi trước giờ bắt đầu kỳ thi

Đặc biệt, trong vòng Chung kết VNOI Cup 2023, Nguyễn Tấn Sỹ Nguyên là thí sinh duy nhất ở độ tuổi 30+ xuất sắc dành được tấm vé tham dự. Bên cạnh đó, vòng thi mang tính quyết định này cũng quy tụ 2/4 thành viên của đội tuyển IOI 2023, chính là Trần Xuân Bách và Nguyễn Đức Thắng; cũng chính là hai gương mặt đã mang về 02 Huy chương Bạc cho đoàn Việt Nam trong kỳ thi IOI 2023.

Ngoài ra, Vòng Chung kết còn có sự góp mặt của các thí sinh đến từ Bảng mở rộng - bảng đấu dành cho mọi thí sinh mong muốn có cơ hội cọ sát trực tiếp tại địa điểm thi của các thí sinh chính thức

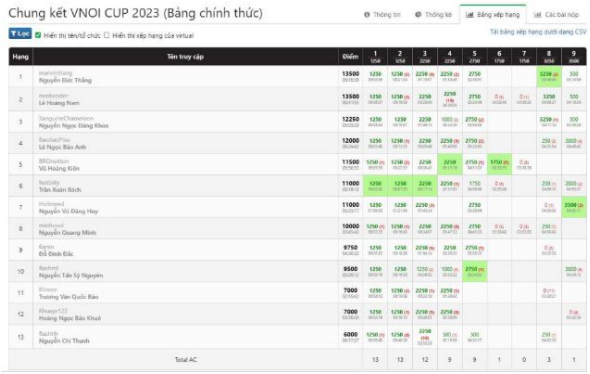
Các thí sinh Bảng mở rộng VNOI CUP

Vòng Chung kết đã diễn ra cực kỳ gay cấn và cực kỳ quyết liệt, khi cục diện bảng xếp hạng bắt đầu thay đổi vào những khoảng thời gian cuối của vòng thi. Chung cuộc, ba vị trí đầu tiên của vòng Chung kết VNOI Cup 2023 đã thuộc về ba cái tên mới:

- Nguyễn Đức Thắng (Trường THPT Chuyên Hùng Vương, Phú Thọ) - là chủ nhân của chiếc cup vô địch VNOI Cup mùa thứ hai, cùng với phần thưởng trị giá 20 triệu VNĐ.
- Lê Hoàng Nam (Trường THPT Chuyên Sư Phạm, Hà Nội) - giành cup Bạc VNOI Cup

2023, cùng với phần thưởng trị giá 15 triệu VNĐ.

- Lê Ngọc Bảo Anh (Trường THPT Chuyên Lê Quý Đôn, Đà Nẵng) - giành cup Đồng VNOI Cup 2023, cùng với phần thưởng trị giá 10 triệu VNĐ.



Bảng xếp hạng chính thức của chung kết VNOI CUP 2023

Có thể nói, VNOI Cup 2023 quy tụ đông đảo các thí sinh với profile rất "khủng" ở nhiều thế hệ khác nhau, từ những đàn anh kỳ cựu nhưng vẫn còn rất nhiều nhiệt huyết đối với lập trình thi đấu, đến những bạn trẻ là những trụ cột tương lai của nền tin học nước nhà.

Ban tổ chức và thí sinh tham dự Chung kết VNOI CUP 2023

Đây chính là dịp để thế hệ coder trẻ có thể giao lưu, cọ xát và học hỏi kinh nghiệm quý báu từ những tiền bối đi trước, những coder đã có kinh nghiệm dày dặn trong mảng lập trình thi đấu lẫn lĩnh vực công nghệ. Như vậy, VNOI Cup 2023 đã trôi qua và để lại nhiều dấu ấn khó quên, cùng nhau hẹn gặp lại tại VNOI Cup 2024 thật sôi động, bùng cháy và kịch tính hơn nhé!

OLP'23 & ICPC ASIA HUE CITY 2023 - Hành trình đáng nhớ

Vào mỗi dịp cuối năm, cũng là lúc mùa thi ICPC-OLP chính thức khởi động. Đặc biệt, năm 2023 là năm đánh dấu nhiều điểm mới trong kỳ thi. Hãy cùng chúng mình điểm lại những dấu ấn của kỳ thi năm nay cũng như chặng đường đồng hành của VNOI trong suốt kỳ thi nhé!

Về kỳ thi OLP'23 & ICPC[15] Asia Hue City 2023

Kỳ thi Olympic Tin học Sinh viên Việt Nam lần thứ 32, kỳ thi PROCON Việt Nam và kỳ thi Lập trình sinh viên quốc tế ICPC Khu vực Châu Á - TP. Huế năm 2023 được tổ chức tại trường Đại học Khoa học - Đại học Huế vào ngày 05 - 08/12/2023 vừa qua. Kỳ thi năm nay quy tụ gần 700 coders xuất sắc nhất đến từ 78 trường đại học, cao đẳng, học viện trên khắp cả nước; cùng với đó là 20 đội tuyển ICPC Quốc tế tài năng đến từ Singapore, Hàn Quốc, Indonesia, Thái Lan, Philippines và Đài Loan.

Không khí phòng thi tại trường Đại học Khoa học Huế

Đặc biệt, đối với ICPC, kỳ thi ICPC Khu vực Châu Á - TP. Huế năm 2023 sẽ chọn ra top 10 đội xuất sắc nhất tranh tài tại kỳ thi ICPC Asia Pacific Champion lần đầu tiên được tổ chức tại Hà Nội vào tháng 03/2024, hướng đến kỳ thi ICPC World Finals[16] 2024 được tổ chức tại Kazakhstan.

Đề thi năm nay được đông đảo các bạn thí sinh và các thầy cô huấn luyện viên đánh giá cao, ở việc chất lượng đề thi được đầu tư vô cùng tỉ mỉ và độ khó được phân bố hợp lý. Cùng với đó, hệ thống thi cũng nhận được những phản hồi tích cực từ các bạn thí sinh, kể cả những bạn thí sinh trong nước và những đội tuyển quốc tế.

Bên cạnh những phút giây thi đấu căng thẳng, các bạn thí sinh cùng huấn luyện viên đã có cơ hội tham gia buổi giao lưu giới thiệu công nghệ Teck Trek với những chủ đề hot như BlockChain, Web 3, Phygital, hay AI-ChatGPT. Tại đây, các bạn đã được trực tiếp gặp gỡ và giao lưu với các doanh nghiệp cùng với các tiền bối đi trước, là những "cựu binh" OLP-ICPC (ExIO) với nhiều năm kinh nghiệm trong lĩnh vực lập trình thi đấu cũng như lĩnh vực công nghệ. Đồng thời, các bạn coders còn có những phút giây đáng nhớ trong chuyến tham quan Đại Nội Huế được diễn ra sau kỳ thi.

Các kiosk phát quà từ các nhà tài trợ kỳ thi

VNOI tại kỳ thi OLP'23 & ICPC Asia Hue City 2023

VNOI rất vinh dự và tự hào khi được đồng hành hỗ trợ xuyên suốt kỳ thi. Cùng với đó, đây là năm thứ ba nền tảng VNOJ của VNOI được tin tưởng và sử dụng làm nền tảng tổ chức ICPC Khu vực Châu Á (site thi tại Việt Nam) và Olympic Tin học Sinh viên Việt Nam.

Đặc biệt, kỳ thi ICPC Asia Hue City 2023 là năm đầu tiên máy ảo được ban tổ chức ICPC Việt Nam đưa vào sử dụng. Đây là hệ thống máy thi được thiết kế bởi VNOI dựa trên hệ thống máy thi chính thức của ICPC World Finals 2023, với một

[15]ICPC: International Collegiate Programming Contest - Cuộc thi Lập trình Quốc tế lâu đời và danh giá nhất dành cho sinh viên các trường đại học và cao đẳng trên toàn cầu.

[16]World Finals: Cuộc thi ICPC cuối cùng trong một mùa giải bao gồm những đội xuất sắc nhất của các trường đại học trên thế giới vượt qua vòng loại vùng (Regional contest)

vài tính năng mới được phát triển riêng dành cho các kỳ thi ICPC tại Việt Nam như:

- VPN: Các máy thí sinh đều sẽ được kết nối tới hệ thống thi cũng như các dịch vụ khác thuộc ICPC thông qua VPN, bao gồm trang thi chính thức, máy in, etc.
- Livestream: Màn hình các thí sinh sẽ được ghi lại và phát trực tiếp khi cần thiết, phục vụ cho quá trình livestream, tái hiện khoảnh khắc của thí sinh, cũng như hỗ trợ cho ban tổ chức, coach và khán giả theo dõi được từng bước đi của thí sinh trong giờ thi.
- Client máy in: Thay vì sử dụng bảng lệnh cũ, việc in bài nay đã được tích hợp vào client của VNOI.

Phòng thi OLP-ICPC sử dụng hệ thống máy thi được thiết kế bởi đội ngũ kỹ thuật của VNOI, trường Đại học Khoa học Huế

VNOI rất vui và tự hào khi nhìn chung hệ thống máy thi đã hoạt động vô cùng trơn tru và hiệu quả trong suốt thời gian diễn ra kỳ thi, cũng như mang lại trải nghiệm tốt đến với các bạn thí sinh.

Kết quả kỳ thi OLP'23 & ICPC Asia Hue City 2023

Trải qua những ngày thi vô cùng cam go và căng thẳng, chủ nhân cho những ngôi vị danh giá đã lộ diện.

Đối với ICPC, chủ nhân của chức Vô địch đã thuộc về đội NewTrend, một đội tuyển vô cùng xuất sắc đến từ Đại học Quốc gia Seoul (Hàn Quốc). Đặc biệt, kỳ thi đã diễn ra vô cùng gay cấn và căng thẳng, khi những vị trí đầu trong bảng xếp hạng liên tục được thay đổi chủ nhân. Hơn thế nữa, sau khi đóng băng, bảng xếp hạng càng trở nên nhộn nhịp hơn bao giờ hết. Đáng nói, đội Vô địch đã có chiến thắng cực kỳ ngoạn mục và ngoài sức tưởng tượng khi AC bài cuối cùng ở phút thứ 299/300, chỉ vài tích tắc trước khi kỳ thi kết thúc.

Còn đối với phần thi Olympic Tin học Sinh viên, chủ nhân của ngôi vị dẫn đầu các bảng thi như sau:

- Vô địch bảng Siêu Cup: Trần Xuân Bách - Trường Đại học Công nghệ, Đại học Quốc gia Hà Nội; đặc biệt, Xuân Bách cũng chính là chủ nhân của tấm Huy chương Vàng IOI[17] 2022 và Huy chương Bạc IOI 2023.
- Vô địch bảng Chuyên Tin: Song Đồng Gia Phúc - Trường Đại học Khoa học Tự nhiên, Đại học Quốc gia Thành phố Hồ Chí Minh. Nói thêm, trường Đại học Khoa học Tự nhiên - Đại học Quốc gia Thành phố Hồ Chí Minh cũng là trường giành được ngôi vị Nhất Đồng đội bảng Chuyên Tin trong cuộc thi.
- Vô địch bảng Không Chuyên & Cao Đẳng: Trịnh Nguyễn Hoàng Vũ - Trường Đại học Công nghiệp Thành phố Hồ Chí Minh. Trường Đại học Công nghiệp Thành phố Hồ Chí Minh là một đại diện mới trong kỳ thi Olympic Tin học Sinh viên cũng như ICPC những năm gần đây, đặc biệt trường đã rất xuất sắc giành được giải Nhất Đồng đội bảng Không Chuyên trong kỳ thi năm nay.

Đội tuyển Seoul National University - Vô định ICPC Asia Hue City 2023

Ngoài những giải thưởng kể trên, ban tổ chức kỳ thi còn trao thêm những giải thưởng dành cho các bạn nữ sinh xuất sắc ở các khối thi. Cụ thể, bạn Phạm Thị Hoài Thu (Đại học Bách khoa Hà Nội) ở khối Chuyên Tin và bạn Phạm Thị Hà Thư (Trường Đại học Kinh tế Quốc dân) ở khối Không Chuyên đã xuất sắc giành được giải thưởng này khi giành được giải Nhất ở khối thi của mình.

[17] IOI: Kỳ thi Olympic Tin học Quốc tế.

Cảm xúc của thí sinh sau kỳ thi

NewTrend - Đại học Quốc Gia Seoul (Hàn Quốc)

Đội thi NewTrend, thuộc Đại học Quốc Gia Seoul (Hàn Quốc)

Q: Trước tiên, cảm xúc của các bạn như thế nào sau 5 tiếng đồng hồ thi đấu?

A: Tụi mình rất phấn khích. Tụi mình đã giải được thêm 2 bài nữa sau đóng băng. Vì giải được thêm bài sau đóng băng nên tụi mình phấn khích.

Q: Mọi người có lẽ cũng cảm nhận được khi thấy các bạn ở trong phòng thi.

A: Ừm, tụi mình tự tin rằng mình có thể vô địch cuộc thi lần này.

Q: Thậm chí lời giải cuối cùng của các bạn được chấm chỉ vài chục giây trước khi cuộc thi kết thúc. Các bạn có cảm nghĩ gì về đề thi năm nay?

A: Mình nghĩ đề thi rất tuyệt vời. Các bài toán cũng khá khó để giải, và tụi mình cũng học được nhiều từ bộ đề này. Tụi mình bất ngờ vì một số bài năm nay khá lạ so với các bài của site Việt Nam trong 5 năm trở lại đây.

Q: Các bạn đánh giá bộ đề này như thế nào so với các Regional khác?

A: Các bài lần này cũng đều thú vị như các Regional khác. Những bài thú vị nhất là những bài khó nhất. Dù tụi mình sau cùng không giải được nhưng tụi mình thấy khá hay.

Q: Thế còn VNOJ thì sao?

A: Mình thấy rất tiện vì mình có thể copy-paste code để nộp (thay vì chọn file như DOMjudge). Mình cũng thích tốc độ chấm của VNOJ. Với DOMjudge, nhiều lúc tụi mình phải chờ vài phút để có kết quả. VNOJ thì rất nhanh nên tụi mình có thể nhanh chóng dự đoán sai sót trong bài làm. Nên là tụi mình thích VNOJ.

Q: Vậy còn môi trường máy thi của các bạn thì sao?

A: Môi trường máy thi lần này khá giống ở Seoul Regional. Lúc in code ra giấy thì có hơi chậm hơn so với ở Seoul Regional, nhưng vậy là đủ nhanh rồi.

Q: Các bạn có gợi ý gì để chúng mình cải thiện không?

A: Phòng thi hơi nhỏ. Tụi mình phải ngồi khá sát nhau.

Q: Trên thang 1-10, các bạn đánh giá cuộc thi lần này bao nhiêu điểm?

A: 10/10 vì tụi mình đã có màn thể hiện tốt và đương nhiên cuộc thi vẫn được tổ chức rất tốt nha.

too_soft & 1RZck - Đại học Quốc Gia Đài Loan

Đội thi toosoft và đội thi 1RZck, thuộc Đại học Quốc Gia Đài Loan

Q: Sau 5 tiếng thi đấu thì các bạn cảm thấy như thế nào?

A: Cơ thể tụi mình mệt mỏi và kiệt sức vì nghĩ bài trong thời gian dài.

Q: Các bạn nghĩ sao về đề thi năm nay?

A: Tụi mình nghĩ đề thi rõ ràng và dễ đọc. Các bài có độ khó cân bằng, trái ngược với Regional tại Đài Loan. Chủ đề của và độ khó của các bài được thiết kế tốt.

Q: Các bạn có tham gia site Việt Nam các năm trước không?

A: Không. Nhưng tụi mình có đọc đề năm ngoái của site Việt Nam. Tất nhiên thì đề năm nay khó hơn rồi.

Q: Các bạn thấy sao về môi trường máy thi?

A: Tốt. Code được in khá nhanh.

Q: Vậy còn VNOJ?

A: Tụi mình thấy kết quả chấm đủ nhanh.

Q: Trên thang 1-10, các bạn đánh giá cuộc thi lần này như thế nào?

A: (nhiều thí sinh đưa ra đánh giá khác nhau, hầu hết 8, một vài 7, một 9)

Q: Các bạn có gợi ý gì để chúng mình cải thiện không ạ?

A: Chắc là socola hoặc các loại bánh tương tự. Tụi mình không có nhiều thời gian để ăn uống, nên tụi mình muốn đồ ăn khối lượng nhỏ nhưng nhiều calo.

Song Đồng Gia Phúc - Trường Đại học Khoa học Tự nhiên, Đại học Quốc gia Thành phố Hồ Chí Minh

Thí sinh Song Đồng Gia Phúc, khối thi Chuyên Tin

"Mình đánh giá rất cao về chất lượng lẫn độ khó của đề thi năm nay, việc chia subtask của các bài toán là chìa khóa giúp mình có những hướng đi đúng đắn. Mình biết rằng cuộc thi năm nay sẽ quy tụ rất nhiều thí sinh xuất sắc nên mình chỉ có một mục tiêu nho nhỏ là đạt được giải số".

Lê Ngọc Bảo Anh - Trường Đại học Bách khoa, Đại học Đà Nẵng

Thí sinh Lê Ngọc Bảo Anh, khối thi Siêu Cup

"Đến với cuộc thi lần này mình không đặt nặng kết quả, chỉ mong đạt được kết quả cao nhất. Đối với mình, máy ảo của kỳ thi rất xịn và tốt, không gặp bất kỳ trục trặc gì; theo thang điểm 10 thì mình đánh giá 8 điểm vì có phần khác một chút so với máy ảo đã được đăng lên trước đó."

Đặng Quốc Cường - Trường Đại học Công nghệ Thông tin, Đại học Quốc gia Thành phố Hồ Chí Minh

Thí sinh Đặng Quốc Cường, khối thi ICPC

"Mình là sinh viên năm nhất nên không đặt nặng vấn đề thành tích, tham gia để trải nghiệm và học hỏi kinh nghiệm là chính. Mình thấy rằng kỳ thi được đầu tư rất là công phu và mọi thứ được chuẩn bị rất kỹ lưỡng."

Phan Khắc Duy Long - Trường Đại học Công nghệ Thành phố Hồ Chí Minh

Thí sinh Phan Khắc Duy Long, khối thi ICPC

"Đây là lần đầu tiên mình được thử sức ở kỳ thi này nên mình cảm thấy đề tương đối khó và có phần khó hơn xíu so với các năm. Đây cũng là lần đầu mình được tiếp xúc với máy ảo thi nhưng cũng dễ làm quen. Mình thấy rằng máy chấm tương đối nhanh và ổn định."

Lê Minh Hoàng - Trường Đại học Khoa học Tự nhiên, Đại học Quốc gia Thành phố Hồ Chí Minh

Thí sinh Lê Minh Hoàng, khối thi ICPC

"Mình thấy rằng đề thi và các test không có bất kỳ vấn đề gì, máy thi hoạt động tốt. Về máy ảo, năm ngoái mình lần đầu tiếp xúc nên có phần bỡ ngỡ, còn năm nay mình đã làm quen và dùng tốt hơn; lúc đầu mình hơi gặp khó khăn trong việc kết nối Internet nhưng đã được các anh chị hỗ trợ."

Đào Thụy Tuyết Nhung - Trường Đại học Sài Gòn

Thí sinh Đào Thụy Tuyết Nhung, khối thi Không Chuyên

"Đến với kỳ thi lần này, mình hy vọng sẽ học hỏi được thêm nhiều kiến thức và có thêm nhiều trải nghiệm. Mình không đặt nặng thành tích vì bản thân khác chuyên ngành Công nghệ thông tin. Mình thấy rằng máy ảo rất tiện và không gặp bất kỳ trục trặc nào, mình đánh giá 10/10 điểm."

Hướng đến The 2023-2024 ICPC Asia Pacific Championship

VNOI hân hạnh và tự hào khi trở thành đơn vị tin cậy cung cấp hệ thống thi cho kỳ thi lần này với việc không xảy ra sự cố kỹ thuật trong giờ thi. Nhận được những phản hồi tích cực đến từ thí sinh và ban tổ chức là động lực để VNOI phát triển hệ thống thi ngày càng hoàn thiện nhằm đem đến những trải nghiệm thi tốt nhất cho thí sinh qua từng năm. Đây chính là bước đệm vững chắc để VNOI hướng tới The 2023-2024 ICPC Asia Pacific Championship.

The 2023-2024 ICPC Asia Pacific Championship được tổ chức tại Trường Đại học Công nghệ, ĐHQG Hà Nội

Nối tiếp những thành công của ICPC Asia Hue City 2023 vừa diễn ra, nhằm chọn ra những đội thi xuất sắc đại diện Asia Pacific tham dự Chung kết ICPC toàn cầu năm 2024 tại Kazakhstan, các thi sinh phải cạnh tranh Ngôi vị Vô địch và 16 suất tham dự ICPC World Finals qua Chung kết ICPC Asia Pacific diễn ra vào tháng 3/2024.

Chung kết ICPC Asia Pacific diễn ra vào tháng 3/2024

Với thể thức mới được áp dụng từ kỳ ICPC năm nay, Hà Nội chính thức trở thành địa điểm đầu tiên được đăng cai tổ chức Chung kết ICPC Asia Pacific 2024 phối hợp với ICPC Việt Nam và ICPC Asia Pacific. Kỳ thi sẽ được tổ chức trực tiếp thi đấu trong 5 tiếng từ 9h00-14h00 tại Khu thi ICPC tại Trường Đại học Công nghệ, ĐHQG Hà Nội trên Hệ thống chuẩn Quốc tế ICPC được ICPC Toàn cầu và Asia Pacific chỉ định và vận hành, phần mềm giám sát thi đấu của Việt Nam do VNOI vận hành.

Từ câu chuyện 'Rùa và Thỏ' đến thuật toán phân tích số nguyên

Nguyễn Minh Hiển
Sinh viên năm 2, Trường Đại học Công Nghệ, Đại học Quốc gia Hà Nội

Phân tích một số nguyên ra thừa số nguyên tố từ lâu đã là một vấn đề quan trọng trong ngành Khoa học máy tính. Đã có rất nhiều thuật toán được đưa ra nhằm tối ưu việc phân tích. Nhưng đôi khi, những thuật toán rất phức tạp lại đến từ những điều rất đơn giản trong đời sống. Vậy chúng ta cùng tìm hiểu xem, làm sao chỉ từ câu chuyện "Rùa và Thỏ", các nhà toán học lại tìm được một thuật toán phân tích số nguyên hiệu quả nha.

Câu chuyện "Rùa và Thỏ"

Bài toán dưới đây này thực tế bắt nguồn từ truyện ngụ ngôn *"Rùa và Thỏ"* của *Aesop*:

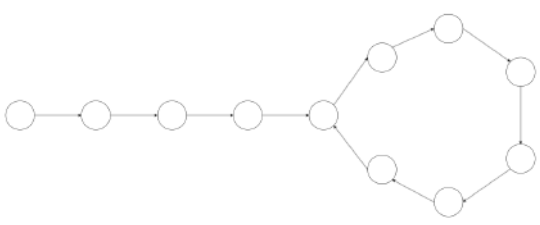
> Rùa và Thỏ cùng thi chạy quanh một vòng sân. Thỏ chạy nhanh gấp 2 lần rùa. Vậy nếu cùng xuất phát và chạy liên tục, Thỏ có gặp Rùa không? Nếu có thì gặp nhau ở đâu?

Câu trả lời vô cùng đơn giản: Khi Rùa chạy được 1 vòng, Thỏ đã chạy được tròn 2 vòng và chúng sẽ gặp nhau tại chính vị trí đã cùng xuất phát.

Từ bài toán trên, người ta lật lại vấn đề rằng: Nếu cho Rùa và Thỏ chạy đường thẳng, thì dù cho chạy nhanh gấp đôi Rùa nhưng Thỏ chẳng bao giờ Thỏ gặp được Rùa cả. Nói cách khác, chỉ khi đi vào đường vòng thì Rùa và Thỏ chạm mặt nhau được!

Từ đó, năm 1967, Floyd đã sử dụng một thuật toán tìm chu trình (đường vòng) trên đồ thị như sau (tạm gọi là **Thuật toán Floyd** [18]):

> Cho một đồ thị đơn, hữu hạn đỉnh. Mỗi đỉnh u có tối đa một đường đi có hướng đến đỉnh $f(u)$ (nếu có). Sử dụng hai con trỏ "Rùa" và "Thỏ" cùng xuất phát. Mỗi bước, Rùa đi 1 đỉnh, còn Thỏ đi 2 đỉnh. Nếu Thỏ gặp Rùa thì đường chúng đã đi có chu trình. Nếu không tồn tại chu trình, Thỏ sẽ đến "đích" trước.



Ta có mã giả của thuật toán như sau:

```
function floyd(f, x0):
    tortoise = f(x0)
    hare = f(f(x0))
    while tortoise != hare:
        tortoise = f(tortoise)
        hare = f(f(hare))
    return (tortoise, hare)
```

Bài toán tìm chu kỳ của dãy

Bài toán: Cho đa thức đa thức hệ số nguyên dương khác hằng $f(x)$ và số nguyên P. Tìm một chu kỳ của dãy số $(x_n \bmod P)$ với (x_n) được xác định như sau:

$$\begin{cases} x_0 > 0 \\ x_n = f(x_{n-1}), n \geq 1 \end{cases}$$

Lời giải:

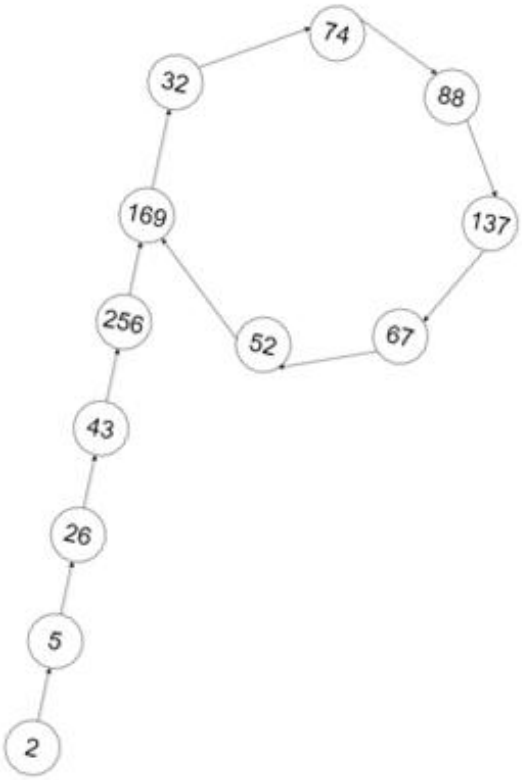
- Xét dãy (u_n) với $u_n = x_n \bmod P$. Suy ra $u_{n+1} = f(u_n) \bmod P$
- Do dãy (u_n) vô hạn nhưng chỉ nhận P giá trị: $0, 1, \ldots, P-1$ nên theo *Nguyên lý Dirichlet* tồn tại hai số nguyên $0 \leq s, t \leq P$ phân biệt sao cho:

$$u_s = u_t$$

  Suy ra $x_s \equiv x_t \pmod P \implies f(x_s) \equiv f(x_t) \pmod P \implies x_{s+1} \equiv x_{t+1} \pmod P$ hay $u_{s+1} = u_{t+1}$. Tiếp tục suy diễn, ta được $u_{s+2} = u_{t+2}, \ldots$ Hay nói cách khác dãy (u_n) tuần hoàn.
- Nếu coi $u_0, u_1, \ldots$ là một dãy cái đỉnh với các cạnh đến hướng từ u_i đến $u_{i+1} = f(u_n) \bmod P$, sử dụng thuật toán Floyd ở trên, ta luôn tìm s, t thỏa $u_s = u_t$ chính là vị trí của Rùa và Thỏ khi gặp nhau.

Minh họa cho trường hợp $p = 317$, $x_0 = 2$ và $f(x) = (x^2 + 1) \bmod 2206637$:

[18] https://en.wikipedia.org/wiki/Cycle_detection#Floyd's_tortoise_and_hare



Bảng dưới đây là minh họa cho thuật toán Floyd trong trường hợp trên:

i	*Rua*	*Tho*	*Rua* mod 317	*Tho* mod 317
0	2	2	2	2
1	5	26	5	26
2	26	458330	26	265
3	677	1671573	43	32
4	458330	641379	265	88
5	1166412	351937	169	67
6	1671573	1264682	32	169
7	2193080	2088470	74	74

Bài toán phân tích số nguyên

Từ đầu bài viết đến giờ, chúng ta vẫn chỉ quanh quẩn với các chu trình, vậy chúng có ứng dụng thế nào vào việc phân tích một số nguyên ra thừa số nguyên tố?

Năm 1975, *John Pollard* đã đề xuất một thuật toán phân tích thừa số nguyên tố như sau:

Xét số nguyên dương N. Chúng ta luôn cố gắng tách $N = P \cdot Q$ với $1 < P \leq Q < N$ rồi từ đó lặp lại việc tách với P và Q đến khi số cần tách là số nguyên tố.

Giả sử trong k số ngẫu nhiên $x_1, x_2, \cdots, x_k$ lấy từ nửa khoảng $[0; N-1)$, tồn tại hai số s, t mà $x_s \equiv x_t \pmod P$, ta suy ra:

$$P \mid \gcd(|x_s - x_t|, N)$$

- *Nhưng mục đích của chúng ta là tìm P cơ mà?* Chúng ta tìm x_s, x_t rồi suy ra P chính là $\gcd(|x_s - x_t|, N)$ đấy! Tất nhiên là phải tìm đến khi $P > 1$ rồi.
- *Liệu có luôn tồn tại x_s, x_t?* Theo nguyên lý Dirichlet thì ta lấy $k = N + 1$ thì luôn tìm được x_s, x_t để $P > 1$.
- *Vậy x_s, x_t lấy đâu ra?* Chính là từ bài toán tìm chu kỳ của dãy số bên trên.
- *Làm sao sinh được dãy (x_n) mà x_i nằm trong $[0; N-1)$? Lấy đa thức f thế nào?* Đây chính là điểm đột phá của thuật toán khi *John Pollard* đề nghị sử dụng các hàm có dạng $f(x) = (x^2 + c) \bmod N$ với c là số nguyên dương bất kỳ để sinh dãy (x_n) ngẫu nhiên với giá trị nằm trong $[0; N-1)$.
- *Nhỡ $x_s = x_t$ thì sao?* Đơn giản thôi, chọn hằng số c khác cho $f(x)$.

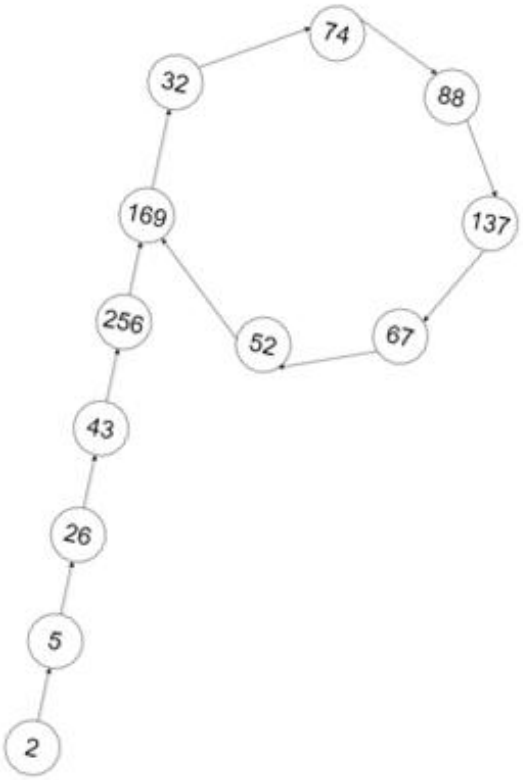
Ta mô tả **Thuật toán Rho** [19] như sau:

Sử dụng hàm $f(x) = (x^2 + c) \bmod P$ để sinh dãy (x_n) đến khi tìm được $x_s \equiv x_t \pmod P$ thỏa mãn $P = \gcd(|x_s - x_t|, N) > 1$.

Khi này P là một ước của N. Tiếp tục phân tích P và $\dfrac{N}{P}$ đến khi không phân tích được nữa.

Một điều thú vị là nhìn hình dưới đây chắc các bạn cùng hiểu tại sao *John Pollard* lại đặt tên *"Thuật toán ρ (Rho)"*:

[19] https://en.wikipedia.org/wiki/Pollard%27s_rho_algorithm



Minh họa cho $N = 2206637$, $x_0 = 2$ và $f(x) = (x^2 + 1) \bmod N$ (tương tự ví dụ của thuật toán Floyd)

Bảng minh họa cho **Thuật toán Rho** trong trường hợp trên:

i	Rua	Tho	$\gcd(\lvert Tho - Rua\rvert, N)$
0	2	2	-
1	5	26	1
2	26	458330	1
3	677	1671573	1
4	458330	641379	1
5	1166412	351937	1
6	1671573	1264682	1
7	2193080	2088470	317

Cài đặt tìm một ước của N bằng C++:

```
// CP-Algorithms
long long
mult(long long a, long long b, long long mod) {
  return (__int128)a * b % mod;
}

long long f(long long x, long long c, long long mod) {
  return (mult(x, x, mod) + c) % mod;
}

long long rho(long long n, long long x0, long long c) {
  long long tortoise = x0, hare = x0, g = 1;
  while (g == 1) {
    tortoise = f(tortoise, c, n);
    hare = f(hare, c, n);
    hare = f(hare, c, n);
    g = gcd(abs(tortoise - hare), n);
  }
  return g;
}
```

[20] https://en.wikipedia.org/wiki/Birthday_problem

Độ phức tạp thuật toán

Thuật toán Floyd

Gọi λ là độ dài chu trình và μ là vị trí đầu tiên xuất hiện chu trình. Khi đó *thuật toán FLoyd* có độ phức tạp thời gian $O(\lambda + \mu)$ và không gian $O(1)$.

Tuy nhiên làm sao ước chừng được $O(\lambda+\mu)$ so với N? Hay nói cách khác, chọn ít nhất bao nhiêu số để xuất hiện chu kỳ?

Nếu để ý, chúng ta có một bài toán hoàn toàn tương tự gọi là **Nghịch lý ngày sinh nhật (The birthday paradox):** [20]

> Trong một năm có d ngày, xác suất để trong k người ngẫu nhiên có hai người trùng ngày sinh nhật là bao nhiêu?

- Nếu $k > d$ thì xác suất là 1 theo nguyên lý Dirichlet.
- Nếu $k \leq d$, sử dụng bao hàm và loại trừ, ta được kết quả là:

$$P(k,d) = 1 - 1 \times \left(1 - \frac{1}{d}\right) \times \left(1 - \frac{2}{d}\right) \times \ldots \times \left(1 - \frac{k-1}{d}\right) = \frac{k! \times C_d^k}{d^k}$$

Trong bài toán hiện tại, k số cần chọn chính là k người, còn ước số P muốn tìm chính là d ngày trong năm.

Giả sử k nhỏ hơn nhiều so với P đang tìm. Khi này theo phép khai triển Taylor, $1 - \frac{i}{P} \sim e^{-\frac{i}{P}}$ hay:

$$P(k,P) \sim 1 - \prod_{i=0}^{k-1} e^{-\frac{i}{P}} \sim 1 - e^{-\frac{k^2}{2P}}$$

$\implies$ Nếu lấy khoảng $k = 4\sqrt{P}$ phần tử thì từ (x_i) sẽ lấy được hai số có cùng số dư khi chia cho P với xác suất 99.97%. Như vậy, nếu xét ước P của N mà nhỏ hơn $\sqrt{N}$ thì số phần tử cần lấy khoảng $4\sqrt[4]{N}$.

Nói tóm lại, $(\lambda + \mu) \sim 4\sqrt[4]{N}$ hay $O(\lambda + \mu) \sim O\left(\sqrt[4]{N}\right)$.

Tìm một ước của N

Mỗi bước tìm chu kỳ của dãy số ta sinh ra, ta tính $\gcd(|x_s - x_t|, N)$ một lần. Tổng cộng ta gọi hàm $\gcd$ trên khoảng $O\left(\sqrt[4]{N}\right)$ lần nên độ phức tạp để tìm một ước của N là $O(\sqrt[4]{N}\log N)$.

Để đảm bảo xác suất trong thuật toán tìm ước, ta cần bỏ các ước nhỏ (Ví dụ $2, 3, 5, 7, 11, 13, 17, 19$) của N trước khi phân tích. Cũng để đảm bảo tìm được một ước $1 < P < N$ thì ta cần kiểm tra tính nguyên tố của N với **kiểm tra Rabin-Miller** [21] trong độ phức tạp $O(\log^2 N)$.

Độ phức tạp toàn bộ thuật toán khi phân tích đến nay vẫn là bài toán mở (do tính ngẫu nhiên khi chọn hàm $f(x) = (x^2+c) \bmod N$), nhưng người ta vẫn cho rằng: **Thuật toán phân tích N có cùng tiệm cận độ phức tạp thời gian với thuật toán Rho.** Trong trường hợp này, độ phức tạp thời gian của thuật toán phân tích là $O(\sqrt[4]{N}\log N)$. Còn độ phức tạp không gian là $O(\log N)$ (sử dụng để lưu kết quả phân tích).

Cải tiến của Brent

Năm 1980, Brent đã cải tiến **Thuật toán Rho** tìm ước sử dụng một cách tìm chu trình khác (tạm gọi là **Thuật toán Brent** [22]).

Gọi λ là độ dài chu trình và μ là vị trí đầu tiên xuất hiện chu trình. Vẫn ý tưởng sử dụng hai con trỏ, nhưng ta sẽ tìm power là lũy thừa nhỏ nhất của 2 mà lớn hơn λ và μ. Sau đó, ta sẽ tìm x_s, x_t bằng cách xuất phát từ $x_s =$ power tìm x_t.

Mô tả: Duyệt i từ 1:

- Cho Rùa chạy đến x_{2^i-1}
- Cho Thỏ chạy từ x_{2^i} đến $x_{2^{i+1}-1}$ xem có tồn tại giá trị bằng Rùa không. Nếu tồn tại thì kết thúc thuật toán.

Mã giả bước tìm chu trình:

```
// CP- Algorithms
function brent(f, x0):
    tortoise = x0
    hare = f(x0)
    power = 1
    while tortoise != hare:
        tortoise = hare
        repeat power times:
            hare = f(hare)
            if tortoise == hare:
                break
        power *= 2
    return (tortoise, hare)
```

Để tăng tốc thuật toán, chúng ta có thể thấy rằng:

- Không cần xét $x_s - x_t$ nếu $s < t < \dfrac{3s}{2}$ (vòng lặp trước không tìm được nên chu kỳ phải lớn hơn power/2). $\implies$ *Loại được khoảng 50% số trường hợp*
- Nếu $\gcd(u_{t+1}u_{t+2}\cdots, N) > 1$ thì chắc chắn một trong các số $u_{t+1}, u_{t+2}, \ldots$ sẽ thỏa mãn $\gcd(u_j, N) > 1$ (dễ dàng chứng minh bằng phản chứng). $\implies$ *Ta không cần gọi* $\gcd(u, N)$ *mỗi bước Thỏ chạy mà gọi theo mỗi bước Rùa chạy hay khoảng* $\log(\lambda + \mu)$ lần.
- Sau cùng, ta vẫn cần một lần duyệt λ bước để tìm vị trí Thỏ gặp Rùa và sử dụng hàm $\gcd(u, N)$ mỗi bước. $\implies$ *Cả thuật toán gọi* $\gcd(u, N)$ *khoảng* $\lambda + \log(\lambda + \mu) \sim \sqrt[4]{N}$ lần.

Độ phức tạp của **Thuật toán Brent** vẫn là $O\left(\sqrt[4]{N}\log N\right)$. Tuy nhiên **Thuật toán Brent** lại có số lần sử dụng $f(x)$ ít hơn **Thuật toán Floyd** rất nhiều nên đã tăng tốc thuật toán tìm chu trình khoảng 36% và thuật toán phân tích số nguyên khoảng 24% - theo Brent [23] .

Bảng dưới đây là minh họa cho **Thuật toán Brent** với $N = 143$, $x_0 = 2$ và $c = 1$. Dãy số: $2, 5, 26, 105, 15, 83, 26, 105, \ldots$ Ở bước $i = 2$, ta không xét 15 do $s < t < \dfrac{3s}{2}$

i	0	1	2
$s = 2^i - 1$	0	1	3
$x_s \bmod N$	2	5	105
$x_{2^i} \to x_{2^{i+1}-1} \bmod N$	5	26, 105	~~15~~, 83, 26, 105
$\lvert x_t - x_s\rvert$	3	21, 100	22, 79, 0
$\gcd(\lvert x_t - x_s\rvert, N)$	-	1	11

Code C++ minh họa:

```
// CP-Algorithms
long long
brent(long long n, long long x0, long long c) {
  long long tortoise = x0, g = 1, q = 1;
  long long hare, xs;
  int m = 128, l = 1;
  while (g == 1) {
    hare = tortoise;
    for (int i = 1; i < l; i++)
      tortoise = f(tortoise, c, n);
    int k = 0;
    while (k < l && g == 1) {
      xs = tortoise;
      for (int i = 0; i < m && i < l - k; i++) {
        tortoise = f(tortoise, c, n);
        q = mult(q, abs(hare - tortoise), n);
      }
      g = gcd(q, n);
      k += m;
    }
    l *= 2;
  }
  if (g == n) do {
```

[21] https://vnoi.info/wiki/algo/algebra/primality_check.md#3-thu%E1%BA%ADt-to%C3%A1n-rabin-miller
[22] https://en.wikipedia.org/wiki/Cycle_detection#Brent's_algorithm
[23] https://en.wikipedia.org/wiki/Cycle_detection#Brent's_algorithm

```
      xs = f(xs, c, n);
      g = gcd(abs(xs - hare), n);
    } while (g == 1);
  return g;
}
```

Dưới đây là code được tham khảo từ Code RR [24] . Code đã xét trước các nhân tử $2, 3, 5, 7, 11, 13, 17, 19$ trước khi sử dụng **Thuật toán Rho Pollard** với **cải tiến của Brent**. Như đã nói ở trên, các nhà nghiên cứu ước lượng thuật toán có độ phức tạp:

- Thời gian: $O\left(\sqrt[4]{N}\log N\right)$ (chưa có chứng minh).
- Không gian: $O(\log N)$ (để lưu kết quả phân tích).

Code đã vượt qua 100 test với $N \leq 10^{18}$ trong khoảng 100ms tại Library Checker - Factorize [25] .

```
#include <bits/stdc++.h>
using namespace std;
using ll = long long;
using ull = unsigned long long;
using ld = long double;
ll mult(ll x, ll y, ll md) {
  ull q = (ld)x * y / md;
  ll res = ((ull)x * y - q * md);
  if (res >= md) res -= md;
  if (res < 0) res += md;
  return res;
}

ll powMod(ll x, ll p, ll md) {
  if (p == 0) return 1;
  if (p & 1) return mult(x, powMod(x, p - 1, md), md);
  return powMod(mult(x, x, md), p / 2, md);
}

bool checkMillerRabin(ll x, ll md, ll s, int k) {
  x = powMod(x, s, md);
  if (x == 1) return true;
  while (k--) {
    if (x == md - 1) return true;
    x = mult(x, x, md);
    if (x == 1) return false;
  }
  return false;
}

bool isPrime(ll x) {
  if (x == 2 || x == 3 || x == 5 || x == 7)
    return true;
  if (x % 2 == 0 || x % 3 == 0 || x % 5 == 0 || x % 7 ==
↪  0)
    return false;
  if (x < 121) return x > 1;
  ll s = x - 1;
  int k = 0;
  while (s % 2 == 0) {
    s >>= 1;
    k++;
  }
  if (x < 1LL << 32) {
    for (ll z : {2, 7, 61})
      if (!checkMillerRabin(z, x, s, k)) return false;
  } else {
    for (ll z :
         {2,
          325,
          9375,
          28178,
          450775,
          9780504,
          1795265022})
      if (!checkMillerRabin(z, x, s, k)) return false;
  }
  return true;
}

ll gcd(ll x, ll y) {
  return y == 0 ? x : gcd(y, x % y);
}

mt19937_64 rng(chrono::steady_clock::now()
                 .time_since_epoch()
                 .count());
long long get_rand(long long r) {
  return uniform_int_distribution<long long>(0, r - 1)(
    rng
  );
}

void pollard(ll x, vector<ll> &ans) {
  if (isPrime(x)) {
    ans.push_back(x);
    return;
  }
  ll c = 1;
  while (true) {
    c = 1 + get_rand(x - 1);
    auto f = [&](ll y) {
      ll res = mult(y, y, x) + c;
      if (res >= x) res -= x;
      return res;
    };
    ll y = 2;
    int B = 100;
    int len = 1;
    ll g = 1;
    while (g == 1) {
      ll z = y;
      for (int i = 0; i < len; i++) z = f(z);
      ll zs = -1;
      int lft = len;
      while (g == 1 && lft > 0) {
        zs = z;
        ll p = 1;
        for (int i = 0; i < B && i < lft; i++) {
          p = mult(p, abs(z - y), x);
          z = f(z);
        }
        g = gcd(p, x);
        lft -= B;
      }
      if (g == 1) {
        y = z;
        len <<= 1;
        continue;
      }
      if (g == x) {
        g = 1;
        z = zs;
        while (g == 1) {
          g = gcd(abs(z - y), x);
          z = f(z);
        }
      }
      if (g == x) break;
      assert(g != 1);
      pollard(g, ans);
      pollard(x / g, ans);
      return;
    }
```

[24] https://github.com/ngthanhtrung23/ACM_Notebook_new/blob/master/Math/NumberTheory/Pollard_factorize.h
[25] https://judge.yosupo.jp/problem/factorize

```
  }
}
// return list of all prime factors of x (can have
// duplicates)
vector<ll> factorize(ll x) {
  vector<ll> ans;
  for (ll p : {2, 3, 5, 7, 11, 13, 17, 19}) {
    while (x % p == 0) {
      x /= p;
      ans.push_back(p);
    }
  }
  if (x != 1) pollard(x, ans);
  sort(ans.begin(), ans.end());
  return ans;
}
// return pairs of (p, k) where x = product(p^k)
vector<pair<ll, int>> factorize_pk(ll x) {
  auto ps = factorize(x);
  ll last = -1, cnt = 0;
  vector<pair<ll, int>> res;
  for (auto p : ps) {
    if (p == last) ++cnt;
    else {
      if (last > 0) res.emplace_back(last, cnt);
      last = p;
      cnt = 1;
    }
  }
  if (cnt > 0) res.emplace_back(last, cnt);
  return res;
}
```

- SPOJ - FACT0 [26]
- Library Checker - Factorize [27]
- SPOJ - FACT1 [28]
- SPOJ - FACT2 [29]
- Wikipedia [30]
- CP[31] -Algorithms - Integer factorization [32]
- Giải thuật Pollard Rho [33]

[26] https://www.spoj.com/problems/FACT0/
[27] https://judge.yosupo.jp/problem/factorize
[28] https://www.spoj.com/problems/FACT1/
[29] https://www.spoj.com/problems/FACT2/
[30] https://en.wikipedia.org/wiki/Main_Page
[31] CP: Competitive Programming - Lập trình thi đấu
[32] https://cp-algorithms.com/algebra/factorization.html
[33] https://www.giaithuatlaptrinh.com/?p=393

Cây 'ảo'

Nguyễn Minh Thiện
Sinh viên năm 3, Trường Công Nghệ Thông Tin và Truyền Thông - Đại Học Cần Thơ

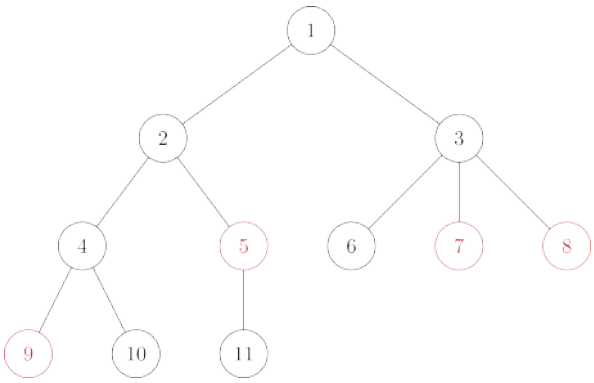
Mở đầu

Giới thiệu về Auxiliary Tree

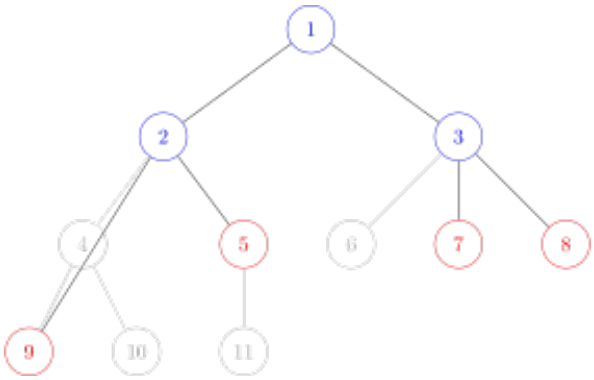
Auxiliary tree hay cây phụ (còn được biết đến với cái tên virtual tree - cây ảo) là một kĩ thuật để giải quyết một lớp các bài toán liên quan đến truy vấn trên một tập các đỉnh của một cây cho trước.

Đối với các bài toán truy vấn trên cây thông thường, có thể coi như một truy vấn và truy vấn trên toàn bộ cây thì thường sẽ dùng các kĩ thuật như centroid decomposition, quy hoạch động trên cây, DSU on tree để giải quyết. Nếu vẫn giữ nguyên cách hỏi của bài toán nhưng lúc này chia ra làm Q truy vấn, mỗi truy vấn chỉ yêu cầu tính toán trên một tập các đỉnh cho trước thì đây là kĩ thuật giúp bạn xây dựng một cây phụ được tạo ra từ tập các đỉnh trong truy vấn cùng với một số đỉnh bổ sung khác (đồng nghĩa với việc bỏ đi các đỉnh không cần thiết trong cây ban đầu), sau đó chỉ cần giải bình thường như trong trường hợp truy vấn trên toàn bộ cây.

Để hình dung rõ hơn thì sau đây là hình minh hoạ cây phụ được sinh ra từ cây ban đầu:



Cây ban đầu



Cây phụ được sinh ra

Trong đó các đỉnh màu đỏ là các đỉnh cần truy vấn, các đỉnh màu xanh là các đỉnh phụ được thêm. Nếu bạn cảm thấy mơ hồ thì cũng đừng lo lắng, ở phần còn lại của bài viết, mình sẽ tập trung vào việc giải thích cũng như cách xây dựng cây phụ, sau đó là cài đặt mẫu và các bài tập ví dụ.

Các kiến thức cần biết

Để hiểu rõ bài viết này, bạn đọc nên có kiến thức về những nội dung sau:

- Cây, các phép duyệt cơ bản (DFS, BFS).
- Lowest common ancestor (LCA).

Dạng bài toán

Hãy cùng xem qua bài toán đơn giản sau:

Bài toán: cho một cây có N đỉnh $(1 \leq N \leq 2 \cdot 10^5)$. Hãy tính tổng khoảng cách của tất cả các cặp đỉnh trong cây. Cụ thể hơn, hãy tính:

$$\sum_{u=1}^{n} \sum_{v=u+1}^{n} \text{dist}(u, v)$$

trong đó $\text{dist}(u, v)$ là khoảng cách giữa hai đỉnh u, v.

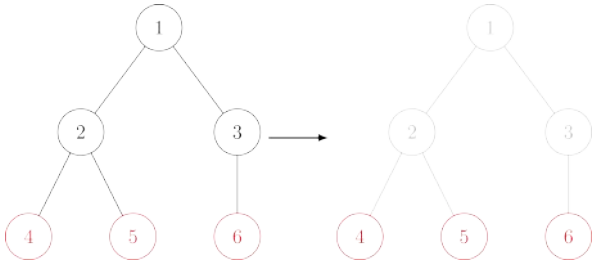
Đây là một bài toán quen thuộc và có thể giải quyết dễ dàng bằng DFS. Sau đây là phiên bản khó hơn một chút:

Phiên bản khó hơn: có tổng cộng Q truy vấn, truy vấn thứ i bao gồm k_i đỉnh (tổng các k_i không vượt quá $2 \cdot 10^5$), yêu cầu hãy tính tổng khoảng cách của tất cả các cặp đỉnh trong số k_i đỉnh này.

Với phiên bản có nhiều truy vấn, ta không thể DFS trên toàn bộ cây cho mỗi truy vấn. Tuy nhiên có một ràng buộc quan trọng là tổng $k_i \leq 2 \cdot 10^5$. Do đó nếu với mỗi truy vấn, nếu tìm được cách loại bỏ các đỉnh không cần thiết để giảm số lượng đỉnh của cây thì bài toán đã được giải quyết.

Cách xây dựng cây phụ

Cây phụ được tạo thành từ các đỉnh truy vấn, cùng với LCA của tất cả các cặp đỉnh này, các đỉnh còn lại sẽ không ảnh hưởng gì sẽ được bỏ qua. Lí do có thêm các đỉnh LCA là vì để đảm bảo các đỉnh kết nối với nhau, ngoài ra số lượng đỉnh LCA phân biệt sẽ không vượt quá số lượng đỉnh truy vấn và mối quan hệ giữa các đỉnh (quan hệ tiền bối - hậu duệ) sẽ được giữ nguyên so với cây ban đầu.



Minh hoạ việc không thể kết nối các đỉnh

Quá trình xây dựng cây phụ sẽ bao gồm hai công đoạn chính:

1. Thêm các đỉnh bổ sung (các đỉnh LCA).
2. Xây dựng các cạnh của cây.

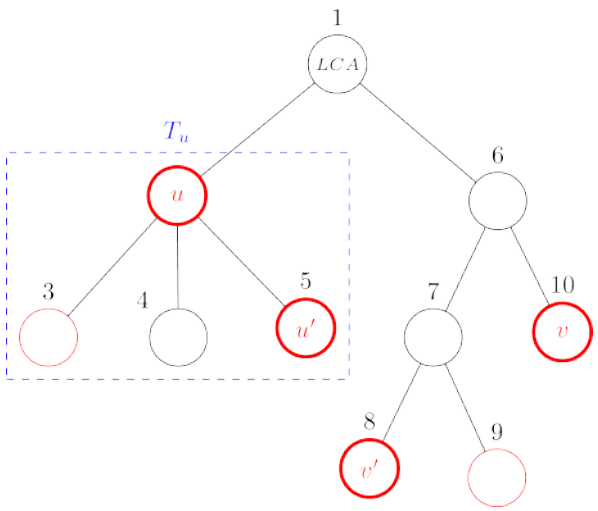
Thêm các đỉnh bổ sung

Một tính chất quan trọng làm tiền đề giúp ta xây dựng được cây phụ đó là mặc dù số lượng cặp đỉnh là rất lớn nhưng số lượng đỉnh LCA phân biệt là không nhiều.

Định lí 1: Giả sử số lượng đỉnh truy vấn là k, khi đó số lượng các đỉnh LCA được thêm không vượt quá $k-1$.

Chứng minh: sắp xếp các đỉnh truy vấn tăng dần theo thứ tự thăm trong phép duyệt DFS (danh sách A). Xét hai đỉnh u, v. Có ba trường hợp xảy ra:

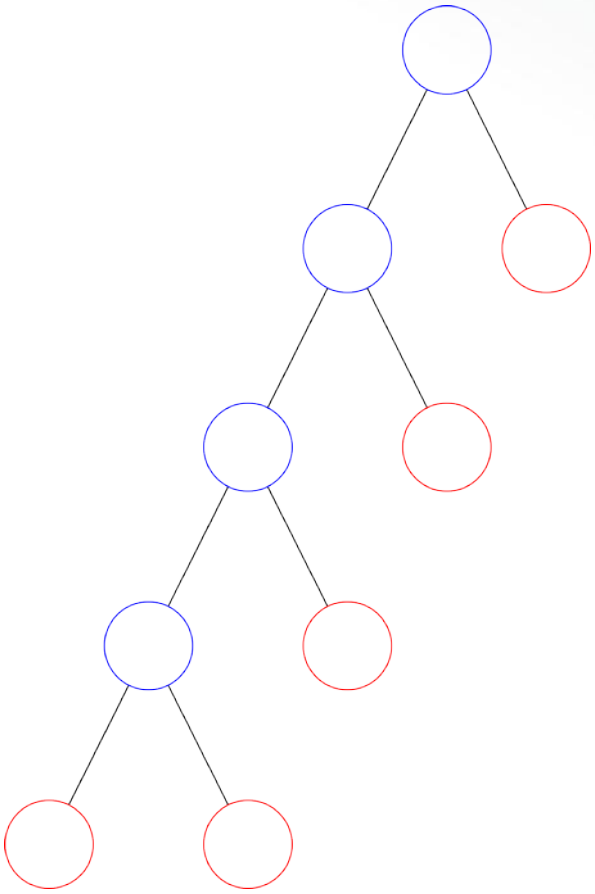
1. u hoặc v là tiền bối của đỉnh còn lại, khi đó số lượng đỉnh LCA không đổi.
2. u và v đứng kề nhau trong danh sách A: thêm LCA của u và v vào cây, số lượng đỉnh LCA tăng 1. Vì chỉ có tối đa $k-1$ cặp (u, v) như vậy nên số lượng đỉnh LCA tăng tối đa là $k-1$.
3. u và v không đứng kề nhau: giả sử u đứng trước v trong danh sách A. Kí hiệu T_u là cây con gốc u trong cây ban đầu, xét đỉnh $u' \in A$ là đỉnh cuối cùng trong T_u và đỉnh $v' \in A$ là đỉnh đầu tiên không nằm trong T_u nhưng nằm trong LCA của u và v, đỉnh v' luôn tồn tại vì v là đỉnh không thuộc T_u. Dễ thấy thứ tự nằm trong A của các đỉnh là $u \le u' < v' \le v$ và LCA của u' và v' cũng chính là LCA của u và v. Trong trường hợp này, số lượng đỉnh LCA không thay đổi. Hình minh hoạ (số trên mỗi đỉnh là thứ tự duyệt DFS):



Ví dụ minh hoạ các đỉnh u, u', v', v

Kết luận: số lượng đỉnh LCA được thêm sẽ không vượt quá $k-1$ và để tìm tất cả các đỉnh LCA, ta chỉ đơn giản lấy LCA của mọi cặp đỉnh u, v kề nhau trong A. Như vậy kích thước của cây phụ là $\mathcal{O}(2k-1) = \mathcal{O}(k)$.

Một ví dụ về trường hợp số đỉnh LCA bằng $k-1$:

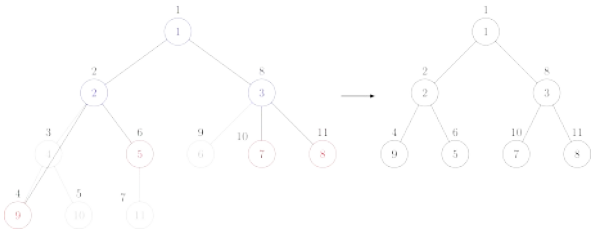


Trường hợp số đỉnh LCA đạt tối đa

Xây dựng các cạnh

Bước tiếp theo sau khi đã thêm các đỉnh bổ sung (các đỉnh LCA) là xây dựng các cạnh của cây.

Sau đây là minh hoạ thuật toán cho cây phụ ở đầu bài viết:



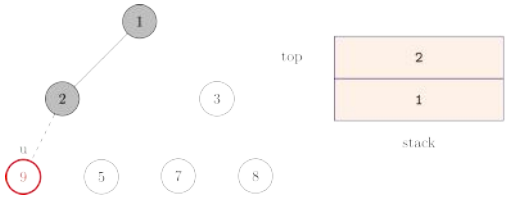
Cây phụ minh hoạ

Chú thích:

- Đỉnh màu đỏ: các đỉnh truy vấn.
- Đỉnh màu xanh: các đỉnh LCA.
- Giá trị trên đầu mỗi đỉnh biểu thị thứ tự thăm trong phép duyệt DFS.

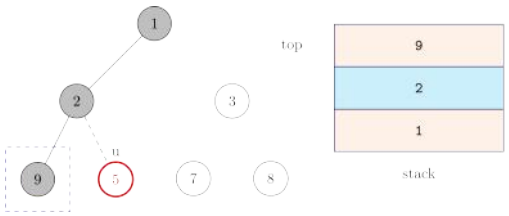
Thuật toán bao gồm các bước sau:

- Sắp xếp các đỉnh cần truy vấn cùng với các đỉnh LCA tăng dần theo thứ tự thăm trong phép duyệt DFS.
- Thêm đỉnh có thứ tự thăm nhỏ nhất vào stack (gốc của cây phụ), trong trường hợp này là đỉnh 1 trên hình.
- Lần lượt xét qua các đỉnh còn lại (giả sử đỉnh đang xét là u):
  - Nếu đỉnh ở đầu stack là tiền bối của u thì đỉnh này sẽ là cha trực tiếp của u trong cây phụ (lí do là vì các đỉnh được duyệt theo thứ tự tăng dần trong phép duyệt DFS) do đó ta thêm cạnh nối giữa u và đỉnh ở đầu stack, thêm u vào stack. Ở hình minh hoạ bên dưới, do đỉnh 2 là tiền bối của đỉnh $u = 9$ nên ta thêm cạnh nối giữa 2 và 9.



Hình minh hoạ trường hợp 1

  - Ngược lại, ta biết rằng một tiền bối gần với u nhất sẽ nằm trong stack (đỉnh này luôn tồn tại do LCA của u luôn được xét trước u) nên ta sẽ lấy phần tử ra khỏi stack cho đến khi phần tử ở đỉnh stack là tiền bối của u. Hình minh hoạ:

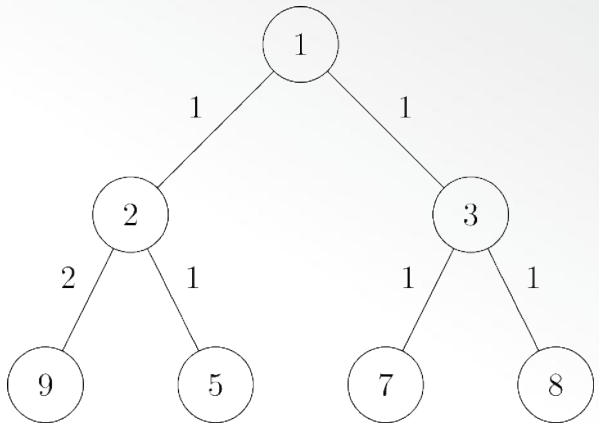


Hình minh hoạ trường hợp 2

trong trường hợp này, tiền bối gần nhất của đỉnh $u = 5$ là đỉnh 2, các đỉnh được lấy ra khỏi stack gồm $\{9\}$, ta thêm cạnh nối giữa 2, 5 và thêm đỉnh 5 vào stack.

Trọng số của các cạnh trong cây phụ bằng khoảng cách của hai đỉnh trong cây ban đầu.

Cuối cùng ta được cây phụ như sau:



Hình minh hoạ trường hợp 2

Cài đặt mẫu

Tiếp sau đây sẽ là cài đặt mẫu cho bài toán ở đầu bài viết, được chia thành các bước như sau.

Tiền xử lí

Đầu tiên ta sẽ thực hiện DFS trên cây ban đầu để xác định thứ tự thăm cũng như tìm cha của mỗi đỉnh.

Chú thích:

- `tin[u]`: thứ tự thăm của đỉnh u trong phép duyệt DFS.
- `anc[j][i]`: bảng thưa lưu cha thứ 2^j của đỉnh i.
- `adj[u]`: danh sách các đỉnh kề với u trong cây ban đầu.
- `tout[u]`: thứ tự thăm của đỉnh con cuối cùng trong cây con gốc u.
- `depth[u]`: độ sâu của đỉnh u.

```
void dfs(int u, int prev = -1) {
  tin[u] = timer++;
  anc[0][u] = prev;
  for (int v : adj[u]) {
    if (v != prev) {
      depth[v] = depth[u] + 1;
      dfs(v, u);
    }
  }
  tout[u] = timer - 1;
}

void preprocess() {
  dfs(0);
  for (int j = 1; j < LOG; ++j) {
    for (int i = 0; i < n; ++i) {
      anc[j][i] =
        (anc[j - 1][i] != -1
           ? anc[j - 1][anc[j - 1][i]]
           : -1);
    }
  }
}
```

Định nghĩa các hàm trợ giúp

Tiếp theo là một số hàm để tìm LCA và hỗ trợ cho việc xây dựng cây phụ ở bước tiếp theo.

Chú thích:

- Hàm `is_ancestor`: kiểm tra xem u có phải là tổ tiên của v hay không.
- Hàm `lca`: tìm LCA của hai đỉnh u, v.
- Hàm `comp`: so sánh u, v theo thứ tự thăm trong phép duyệt DFS.

```
bool is_ancestor(int u, int v) {
  return tin[u] <= tin[v] && tin[v] <= tout[u];
}

int lca(int u, int v) {
  if (is_ancestor(u, v)) return u;
  if (is_ancestor(v, u)) return v;
  for (int i = LOG - 1; i >= 0; --i) {
    int pu = anc[i][u];
    if (pu != -1 && !is_ancestor(pu, v)) u = pu;
  }
  u = anc[0][u];
  assert(is_ancestor(u, v));
  return u;
}

bool comp(int u, int v) { return tin[u] < tin[v]; }
```

Xây dựng cây phụ

Ở bước xây dựng này sẽ bao gồm ba công đoạn chính:

- Sắp xếp các đỉnh tăng dần theo thứ tự thăm trong phép duyệt DFS.
- Thêm các đỉnh LCA vào tập đỉnh.
- Tiếp tục sắp xếp các đỉnh theo thứ tự tăng dần trong phép duyệt DFS, thêm đỉnh gốc vào stack (đỉnh đầu tiên), sau đó duyệt qua từng đỉnh còn lại và thêm các cạnh tương ứng.

Chú thích:

- `aux_adj[u]`: danh sách các đỉnh kề với u trong cây phụ.

```
int build_auxiliary_tree(vector<int> &vers) {
  int sz = (int)vers.size();

  // thêm các đỉnh LCA.
  sort(vers.begin(), vers.end(), comp);
  for (int i = 0; i < sz - 1; ++i) {
    int new_vertex = lca(vers[i], vers[i + 1]);
    vers.push_back(new_vertex);
  }

  sort(vers.begin(), vers.end(), comp);

  // loại bỏ các đỉnh bị trùng.
  vers.erase(
    unique(vers.begin(), vers.end()), vers.end()
  );

  // thêm các cạnh.
  vector<int> stack;
  int aux_root = vers[0];
  stack.push_back(aux_root);
  for (int i = 1; i < (int)vers.size(); ++i) {
    int u = vers[i];
    while (!stack.empty() &&
           !is_ancestor(stack.back(), u))
      stack.pop_back();
    assert(!stack.empty());
    int last = stack.back();
    aux_adj[last].push_back(u);

    stack.push_back(u);
  }
  return aux_root;
}
```

Hàm này sẽ trả về đỉnh gốc của cây phụ.

Trả lời truy vấn

Phần còn lại đó là trả lời truy vấn trên mỗi cây phụ xây dựng được. Sau đây là lời giải tham khảo.

Lời giải: trên cây phụ xây dựng được, để tính tổng khoảng cách của các đỉnh thì với mỗi cạnh ta sẽ đếm lượng đóng góp của cạnh đó vào đáp án (hay số lượng đường đi mà đi qua cạnh này). Đáp án là tổng đóng góp của các cạnh.

Chú thích:

- `cnt_important[u]`: lưu số lượng đỉnh có nằm trong các đỉnh truy vấn của cây con gốc u.

```
long long dfs2(int u) {
  long long ans = 0;
  for (int v : aux_adj[u]) {
    ans += dfs2(v);
    cnt_important[u] += cnt_important[v];
  }
  for (int v : aux_adj[u]) {
    int dist = depth[v] - depth[u];
    ans += 1LL * dist * cnt_important[v] *
           (k - cnt_important[v]);
  }
  return ans;
}

void solve() {
  cin >> q;
  for (int qc = 0; qc < q; ++qc) {
    cin >> k;
    vector<int> vers(k);
    for (int i = 0; i < k; ++i) {
      cin >> vers[i];
      --vers[i];
      cnt_important[vers[i]] = 1;
    }
    int aux_root = build_auxiliary_tree(vers);
    long long sum = dfs2(aux_root);
    cout << sum << '\n';

    for (int ver : vers) {
      aux_adj[ver].clear();
      cnt_important[ver] = 0;
    }
  }
}
```

Code hoàn chỉnh

Dưới đây là code hoàn chỉnh cho bài toán ở đầu bài viết, bao gồm các hàm đã được giải thích ở trên theo trình tự.

```cpp
#include <bits/stdc++.h>
using namespace std;

const int N = (int)2e5;
const int LOG = 21;
vector<int> adj[N], aux_adj[N];
int depth[N], tin[N], tout[N];
int cnt_important[N];
int anc[LOG][N];
int timer;
int n, q, k;

void dfs(int u, int prev = -1) {
  tin[u] = timer++;
  anc[0][u] = prev;
  for (int v : adj[u]) {
    if (v != prev) {
      depth[v] = depth[u] + 1;
      dfs(v, u);
    }
  }
  tout[u] = timer - 1;
}

void preprocess() {
  dfs(0);
  for (int j = 1; j < LOG; ++j) {
    for (int i = 0; i < n; ++i) {
      anc[j][i] =
        (anc[j - 1][i] != -1
           ? anc[j - 1][anc[j - 1][i]]
           : -1);
    }
  }
}

bool is_ancestor(int u, int v) {
  return tin[u] <= tin[v] && tin[v] <= tout[u];
}

int lca(int u, int v) {
  if (is_ancestor(u, v)) return u;
  if (is_ancestor(v, u)) return v;
  for (int i = LOG - 1; i >= 0; --i) {
    int pu = anc[i][u];
    if (pu != -1 && !is_ancestor(pu, v)) u = pu;
  }
  u = anc[0][u];
  assert(is_ancestor(u, v));
  return u;
}

bool comp(int u, int v) { return tin[u] < tin[v]; }

void read_input() {
  cin >> n;
  for (int i = 0; i < n - 1; ++i) {
    int u, v;
    cin >> u >> v;
    --u;
    --v;
    adj[u].push_back(v);
    adj[v].push_back(u);
  }
}

int build_auxiliary_tree(vector<int> &vers) {
  int sz = (int)vers.size();

  sort(vers.begin(), vers.end(), comp);
  for (int i = 0; i < sz - 1; ++i) {
    int new_vertex = lca(vers[i], vers[i + 1]);
    vers.push_back(new_vertex);
  }

  sort(vers.begin(), vers.end(), comp);
  vers.erase(
    unique(vers.begin(), vers.end()), vers.end()
  );
  vector<int> stack;
  int aux_root = vers[0];
  stack.push_back(aux_root);
  for (int i = 1; i < (int)vers.size(); ++i) {
    int u = vers[i];
    while (!stack.empty() &&
           !is_ancestor(stack.back(), u))
      stack.pop_back();
    assert(!stack.empty());
    int last = stack.back();
    aux_adj[last].push_back(u);

    stack.push_back(u);
  }
  return aux_root;
}

long long dfs2(int u) {
  long long ans = 0;
  for (int v : aux_adj[u]) {
    ans += dfs2(v);
    cnt_important[u] += cnt_important[v];
  }
  for (int v : aux_adj[u]) {
    int dist = depth[v] - depth[u];
    ans += 1LL * dist * cnt_important[v] *
           (k - cnt_important[v]);
  }
  return ans;
}

void solve() {
  cin >> q;
  for (int qc = 0; qc < q; ++qc) {
    cin >> k;
    vector<int> vers(k);
    for (int i = 0; i < k; ++i) {
      cin >> vers[i];
      --vers[i];
      cnt_important[vers[i]] = 1;
    }
    int aux_root = build_auxiliary_tree(vers);
    long long sum = dfs2(aux_root);
    cout << sum << '\n';

    for (int ver : vers) {
      aux_adj[ver].clear();
      cnt_important[ver] = 0;
    }
  }
}

int main() {
  cin.tie(nullptr)->sync_with_stdio(false);
  read_input();
  preprocess();
  solve();

  return 0;
}
```

Độ phức tạp: $\mathcal{O}(N \log N)$ cho bước tiền xử lí và $\mathcal{O}(k_i \log k_i)$ cho mỗi truy vấn. Vì tổng các k_i không vượt quá N nên độ phức tạp cuối cùng là $\mathcal{O}(N \log N)$.

Các bài tập ví dụ

Codeforces[34] - 1702G2 [35]

Tóm tắt đề bài

Cho một cây có n đỉnh. Một tập các đỉnh được gọi là *passable* nếu tồn tại một đường đi đi qua tất cả các đỉnh, không có cạnh nào đi qua hai lần và đường đi này có thể đi qua các đỉnh khác (không nằm trong tập đỉnh).

Có q truy vấn. Mỗi truy vấn được cho bởi một tập các đỉnh, yêu cầu hãy xác định xem tập đỉnh này có là *passable* hay không.

Giới hạn

- $1 \leq n \leq 2 \cdot 10^5$.
- $1 \leq q \leq 10^5$.
- Tổng số lượng đỉnh trong các truy vấn không vượt quá $2 \cdot 10^5$.

Lời giải

Nhận xét rằng một tập các đỉnh là *passable* khi mỗi đỉnh chỉ có tối đa một con hoặc tối đa hai con trong trường hợp đó là đỉnh gốc.

Với mỗi truy vấn, ta sẽ xây dựng cây phụ và kiểm tra xem cây phụ được tạo thành có là *passable* hay không.

Chứng minh tính đúng đắn:

- Nếu một tập đỉnh là *passable* thì việc thêm các đỉnh LCA sẽ không thay đổi đáp án do các đỉnh LCA được thêm cũng sẽ nằm trên đường đi.
- Nếu một tập đỉnh không phải là *passable* thì việc thêm các đỉnh LCA cũng không thể khiến các đỉnh trở nên *passable*.

Cài đặt mẫu

Thay đổi lại hàm `solve` như sau:

```
void solve() {
  cin >> q;
  for (int qc = 0; qc < q; ++qc) {
    cin >> k;
    vector<int> vers(k);
    for (int i = 0; i < k; ++i) {
      cin >> vers[i];
      --vers[i];
    }
    int aux_root = build_auxiliary_tree(vers);
    bool passable = (int)aux_adj[aux_root].size() <= 2;
    for (int i = 1; i < (int)vers.size(); ++i) {
      if ((int)aux_adj[vers[i]].size() > 1) {
        passable = false;
        break;
      }
    }
    cout << (passable ? "YES" : "NO") << '\n';

    // reset lại giá trị cho các truy vấn tiếp theo
    for (int ver : vers) aux_adj[ver].clear();
  }
}
```

Độ phức tạp: do chỉ duyệt qua các đỉnh trong cây phụ nên độ phức tạp vẫn là $\mathcal{O}(N \log N)$.

Codechef - YATP [36]

Tiếp theo ta sẽ đến với bài mà mỗi cạnh trong cây đều có trọng số.

Tóm tắt đề bài

Cho một cây có N đỉnh, mỗi cạnh đều có một trọng số, với đỉnh gốc là đỉnh 1.

Định nghĩa $g(u, v)$ là trọng số của cạnh có trọng số lớn nhất trên đường đi từ u đến v.

Có Q truy vấn, mỗi truy vấn được cho bởi K đỉnh phân biệt $v_1, v_2, \ldots, v_K$, yêu cầu hãy tính tổng giá trị hàm g đối với mọi cặp đỉnh trong tập các đỉnh đã cho.

Giới hạn

- $10 \leq N \leq 10^5$.
- $1 \leq Q \leq 3 \cdot 10^4$.
- Tổng các giá trị K_i không vượt quá $1.5 \cdot 10^6$.

Lời giải

Đầu tiên xây dựng cây phụ với trọng số của cạnh u, v là $g(u, v)$, có thể dễ dàng tính thông qua bảng thưa và nhảy nhị phân (sẽ không được trình bày cụ thể tại đây).

Trên cây phụ, tính toán đóng góp của mỗi cạnh vào đáp án như sau: sắp xếp các cạnh tăng dần theo trọng số, dùng DSU để lần lượt thêm các cạnh, khi một cạnh nối hai thành phần liên thông khác nhau thì luôn đảm bảo trọng số ở các thành phần liên thông không vượt quá trọng số của cạnh hiện tại, do đó chỉ cần đếm số đường đi đi qua cạnh này.

[34] Codeforces: một trang web tổ chức các cuộc thi lập trình thi đấu với các dạng đề, bài tập đa dạng.
[35] https://codeforces.com/contest/1702/problem/G2
[36] https://www.codechef.com/DEC21A/problems/YATP

Cài đặt mẫu

Định nghĩa struct DSU:

```
struct Dsu {
  int n;
  vector<int> par, sz, cnt_important;
  Dsu() {}
  Dsu(int _n)
    : n(_n)
    , par(n)
    , sz(n)
    , cnt_important(n) {
    for (int i = 0; i < n; ++i) {
      par[i] = i;
      sz[i] = 1;
    }
  }
  void reset(int u) {
    par[u] = u;
    sz[u] = 1;
  }
  int find(int v) {
    while (v != par[v]) v = par[v] = par[par[v]];
    return v;
  }
  long long unite(int u, int v, int w) {
    u = find(u);
    v = find(v);
    assert(u != v);
    if (sz[u] < sz[v]) swap(u, v);
    sz[u] += sz[v];
    par[v] = u;

    // tính lượng đóng góp của cạnh (u, v).
    long long res =
      1LL * w * cnt_important[u] * cnt_important[v];
    cnt_important[u] += cnt_important[v];
    return res;
  }
} dsu;
```

Thay đổi hàm `solve` như sau:

```
void solve() {
  dsu = Dsu(n);
  for (int qc = 0; qc < q; ++qc) {
    cin >> k;
    vector<int> vers(k);
    for (int i = 0; i < k; ++i) {
      cin >> vers[i];
      --vers[i];
      dsu.cnt_important[vers[i]] = 1;
    }

    build_auxiliary_tree(vers);

    vector<array<int, 3>> edges;
    for (int u : vers)
      for (auto [v, w] : aux_adj[u])
        edges.push_back({u, v, w});

    // sắp xếp các đỉnh tăng dần theo trọng số
    sort(
      edges.begin(),
      edges.end(),
      [](auto a, auto b) { return a[2] < b[2]; }
    );
    long long res = 0;
    for (auto [u, v, w] : edges)
      res += dsu.unite(u, v, w);
    cout << res << '\n';

    // reset lại giá trị cho các truy vấn tiếp theo
    for (int ver : vers) {
      aux_adj[ver].clear();
      dsu.cnt_important[ver] = 0;
      dsu.reset(ver);
    }
  }
}
```

Độ phức tạp: độ phức tạp là $\mathcal{O}(N \log N)$.

Lời kết

Trong bài viết này, chúng ta đã tìm hiểu về kĩ thuật Auxiliary Tree, được áp dụng để giải quyết các bài toán liên quan đến truy vấn trên một tập đỉnh của cây. Hi vọng rằng bài viết này giúp bạn có cái nhìn tổng quan về Auxiliary Tree và vận dụng để giải quyết các bài toán tương tự.

Bài tập vận dụng

- Codechef - Chef and Pairs [37]
- Hackerrank HourRank 15 - Kitty's Calculations on a Tree [38]
- Codechef - Adjacent Leaves [39]
- Codeforces round 614 - D [40]
- Codeforces round 339 - D [41]

Tham khảo

- https://codeforces.com/blog/entry/76955
- https://blog.sengxian.com/algorithms/virtual-tree
- https://oi-wiki.org/graph/virtual-tree/

[37] https://www.codechef.com/problems/PAIRCNT
[38] https://www.hackerrank.com/contests/hourrank-15/challenges/kittys-calculations-on-a-tree/problem
[39] https://www.codechef.com/problems/ADJLEAF2
[40] https://codeforces.com/contest/1292/problem/D
[41] https://codeforces.com/contest/613/problem/D

Quy hoạch động 'hồ sơ gãy'

Dương Hoàng Long
Lớp 11 Tin, Trường Phổ Thông Năng Khiếu - DHQG-HCM

Lời nói đầu

Nếu bạn đã học CP[42] *lâu năm, bạn có thể đã từng thấy vài bài sử dụng DP Broken Profile nhưng khi lên mạng thì lại gần như không có các thông tin về nó. Vì thế, bài viết này sẽ mang đến cho các độc giả về các thông tin cơ bản về DP Broken Profile để có thể làm và hiểu được về dạng bài này.*

Kiến thức cần thiết: Quy Hoạch Động, Quy hoạch động trạng thái (bitmask).

DP Broken Profile là một dạng đặc biệt của dạng toán DP Bitmask nhằm xử lý một cách tối ưu những bài toán có một hoặc nhiều đặc điểm như:

- Bài toán thường xoay quanh lấp đầy một bảng $2D$
- Một chiều nhỏ hơn rất nhiều so với chiều còn lại.
- **Khi lấp bảng, một ô chỉ phụ thuộc vào ô kề nó.**
- Số lượng trạng thái của mỗi ô khá it (thường là 2)

Bài tập sử dụng DP Broken Profile thường rất khó nên việc hiểu biết những hướng làm, những ưu và nhược điểm của từng hướng sẽ là chìa khóa giữa việc AC bài đó hay không.

CSES - Counting Tilings [43]

- Cho một bảng $N \times M$, hãy tìm số lượng cách lấp đầy bảng bằng cách đặt các viên domino có kích thước 1×2 hoặc 2×1 không giao nhau.
- $N \leq 10, M \leq 1000$

Phân tích:

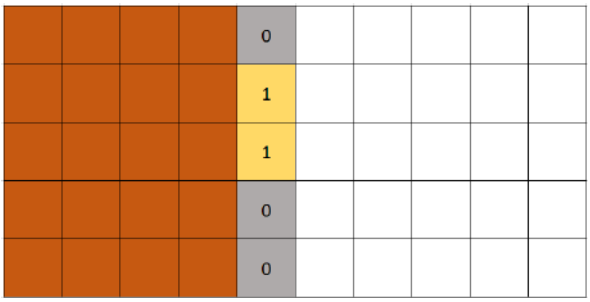
Với N bé như 1, 2 hay 3 thì công thức truy hồi rất dễ để tìm ra:

- $N = 1$: Ta thấy chỉ có duy nhất 1 cách điền (Nếu như M chẵn).
- $N = 2$: $dp_m = dp_{m-1} + dp_{m-2}$, hay hàm dp của ta sẽ là số $Fibonacci$ thứ M.
- $N = 3$: Công thức truy hồi khó hơn chút nhưng ta có thể chứng minh được công thức truy hồi của ta sẽ là $dp_m = 4 \cdot dp_{m-1} - dp_{m-2}$. Công thức đó được chứng minh ở đây [44] .

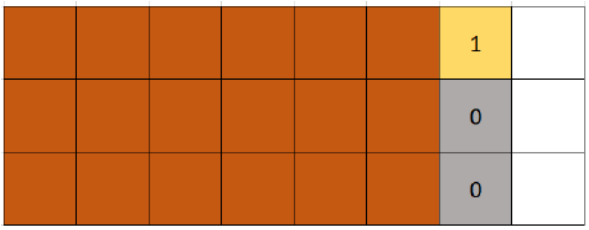
Khi ta xét trường hợp tổng quát, việc tìm công thức truy hồi chỉ có thể khi dùng Berlekamp-Massey. Nhưng với N nhỏ (nhỏ hơn 10 như trong bài này), thì ta có thể làm cách nào đơn giản hơn không?

Hướng tiếp cận "trâu"

Để tiếp cận bài này, ta sẽ xếp các miêng domino dần dần từ trái sang phải. Gọi $dp_{i,mask}$ là số lượng cách để sắp xếp domino để đầy $i-1$ cột đầu tiên và cột thứ i sẽ có hình dạng như $mask$. Hay rõ hơn, nếu ô (i, j) đã được lấp thì bit thứ j sẽ bật và nếu ô (i, j) trống thì bit thứ j sẽ được tắt. Xem hình bên dưới để hiểu rõ thêm vè trạng thái quy hoạch động:



Trạng thái $dp_{5,01100}$ *cho bảng* 5×10



Trạng thái $dp_{7,100}$ *cho bảng* 3×8

Khi có $i, mask$ thì ta phải đặt một số domino để đầy dòng i và được $nextmask$ cho cột $i+1$ để có thể cập nhật lên.

Ta thấy rằng chỉ có nhiều nhất một cách để tạo $nextmask$ từ $mask$ mà thôi.

Chứng minh:

- Nếu bit thứ k của $nextmask$ là 1 thì chắc chắn bit thứ k của $mask$ chỉ có thể là 0 do ta phải đặt domino theo chiều ngang ở ô (i, k) để biến bit thứ k của $nextmask$ là 1.
- Nếu bit thứ k của $nextmask$ là 0 thì là bit thứ k của $mask$ vừa có thể là 0 hoặc 1.
  - Nếu bit thứ k của $mask$ là 0 thì ta bắt buộc phải đặt ở đây một domino theo chiều dọc do ta phải làm đầy cột i thì mới lên xét được cột $i+1$

[42] CP: Competitive Programming - Lập trình thi đấu
[43] https://cses.fi/problemset/task/2181
[44] https://goo.gl/ixTyA5

  - Nếu bit thứ k của $mask$ là 1 thì ta không thể làm gì do ô (i, k) đã được lấp đầy.
- Như thế, trong mọi trường hợp thì ta cũng chỉ có một cách để biến $mask$ thành $nextmask$.

Nếu ta xây dựng $nextmask$ bằng cách xét tất cả các $nextmask$ rồi kiểm tra là $mask$ hiện tại và $nextmask$ có đồng khớp với nhau không thì nó sẽ mất khá lâu. Ta có for M ,for $mask$ và for $nextmask$ nên độ phức tạp sẽ là $O(M \cdot 2^N \cdot 2^N)$ hay $O(M \cdot 4^N)$, không đủ nhanh cho bài này. Việc kiểm tra $mask$ và $nextmask$ có khớp được với nhau hay không trong $O(1)$ các bạn có thể tham khảo ở đây [45] tại phút thứ 40.

Tham khảo code mẫu thuật $O(M \cdot N^4)$:

```
#include <bits/stdc++.h>
using namespace std;
int dp[1009][1024];
const int mod = 1e9 + 7;
int n, m;
bool good(int mask, int nextmask) {
  if ((mask & nextmask) != 0) return 0;
  int x = ((~mask) & (~nextmask)) & ((1 << n) - 1);
  if (x % 3 != 0) return 0;
  int y = x / 3;
  if ((y & (y << 1)) != 0) return false;
  return true;
}
int main() {
  ios_base::sync_with_stdio(false);
  cin.tie(0);
  cout.tie(0);
  cin >> n >> m;
  dp[1][0] = 1;
  for (int i = 1; i <= m; i++)
    for (int mask = 0; mask < (1 << n); mask++)
      for (int nextmask = 0; nextmask < (1 << n);
           nextmask++)
        if (good(mask, nextmask))
          (dp[i + 1][nextmask] += dp[i][mask]) %= mod;

  cout << dp[m + 1][0];
}
```

Có một tối ưu là thay vì xét tất cả các $nextmask$ thì ta chỉ xét các mask con của phần bù của $mask$ hiện tại. Nó sẽ tối ưu độ phức tạp từ $O(M \cdot 4^N)$ thành $O(M \cdot 3^N)$. Nếu ta chỉ xét các submask bằng đệ quy thì ta sẽ gặp một trở ngại là hằng số rất cao nên khó có thể áp dụng cho bài khác. Vì thế, ta có thể sử dụng một trick biến đổi bit để xét các submask bằng vòng for ở đây [46] . Như thế, code của ta vừa được tối ưu hơn và đủ nhanh để ac bài này.

Tham khảo code mẫu:

```
#include <bits/stdc++.h>
using namespace std;
int dp[1009][1024];
const int mod = 1e9 + 7;
int n, m;
bool good(int mask, int nextmask) {
  int x = ((~mask) & (~nextmask)) & ((1 << n) - 1);
  if (x % 3 != 0) return 0;
  int y = x / 3;
  if ((y & (y << 1)) != 0) return false;
  return true;
}
int main() {
  ios_base::sync_with_stdio(false);
  cin.tie(0);
  cout.tie(0);
  cin >> n >> m;
  dp[1][0] = 1;
  for (int i = 1; i <= m; i++) {
    for (int mask = 0; mask < (1 << n); mask++) {
      if (good(mask, 0)) {
        dp[i + 1][0] += dp[i][mask];
        dp[i + 1][0] %= mod;
      }

      // phan bu cua mask
      int nmask = ((1 << n) - 1) ^ mask;
      for (int nextmask = nmask; nextmask > 0;
           nextmask = (nextmask - 1) & nmask) {
        if (good(mask, nextmask)) {
          dp[i + 1][nextmask] += dp[i][mask];
          dp[i + 1][nextmask] %= mod;
        }
      }
    }
  }
  cout << dp[m + 1][0];
}
```

Tối ưu thứ nhất

Với bài này, ta sẽ dựng các $nextmask$ có thể bằng cách điền tuần tự vào các ô trống của $mask$. Hay cụ thể hơn, ta sẽ đi từng dồng của cột thứ j đang được biểu diễn bởi $mask$. Nếu như dòng(bit) thứ j đã được điền thì ta bỏ qua và xét dòng tiếp theo. Nếu như dòng(bit) thứ j đang trống thì ta xét hai trường hợp sau:

- Đặt domino 1×2 ở đây (tất nhiên là ta chỉ đặt được nếu như $i+1$ không vượt quá M. Việc này tương đương với lấp đầy ô $(i+1, j)$, hay bặt bit thứ j của $nextmask$.
- Đặt domino 2×1 ở đây nếu dòng(bit) thứ $j+1$ đang trống. Việc này không ảnh hưởng đến dòng tiếp theo nên ta sẽ không biến đổi $nextmask$.

Khi đi hết N dòng thì ta sẽ cập nhật cho $dp_{i+1,nextmask}$ tương ứng với các trường hợp ta xét.

Vì hai trường hợp ta đều cần phải xét nên khi cài đặt, ta sẽ sử dụng một hàm đệ quy để thử hết tất cả trường hợp có thể của $nextmask$.

Vì ta muốn lấp đầy hết tất cả các ô trên bảng và không lấn ra ngoài bảng nên là đáp án của ta sẽ là $dp_{m+1,0}$. Trường hợp cơ sở của ta sẽ là $dp_{1,0} = 1$.

Tham khảo code mẫu:

```
#include <bits/stdc++.h>
using namespace std;
```

[45] https://www.youtube.com/watch?v=0bnMHIFUM_o

[46] https://cp-algorithms.com/algebra/all-submasks.html#iterating-through-all-masks-with-their-submasks-complexity-o3n

```
int dp[1009][1024];
const int mod = 1e9 + 7;
int n, m;
bool on(int mask, int i) { return mask & (1 << i); }
void up(int j, int i, int mask, int nextmask) {
  if (j >= n) {
    dp[i + 1][nextmask] += dp[i][mask];
    dp[i + 1][nextmask] %= mod;
    return;
  }
  if (on(mask, j)) {
    // Neu bit thu j da duoc bat
    up(j + 1, i, mask, nextmask);
  } else {
    if (j + 1 < n && !on(mask, j + 1)) {
      // TH2
      up(j + 2, i, mask, nextmask);
    }
    // TH1
    up(j + 1, i, mask, (nextmask | (1 << j)));
  }
}
int main() {
  ios_base::sync_with_stdio(false);
  cin.tie(0);
  cout.tie(0);
  cin >> n >> m;
  dp[1][0] = 1;
  for (int i = 1; i <= m; i++)
    for (int j = 0; j < (1 << n); j++) up(0, i, j, 0);
  cout << dp[m + 1][0];
}
```

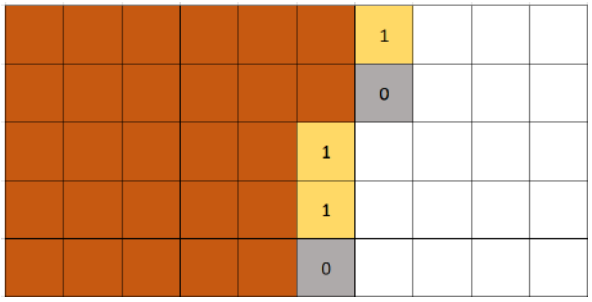
Ta nhận thấy, việc xét các $mask$ và $nextmask$ đã tốn ít nhất thời gian chạy là $O(N \cdot M \cdot 2^N)$. Nhưng code ở trên còn phần đệ quy nữa, vậy thời gian chạy của việc đệ quy là gì?

Gọi $totalcall(i)$ là số lần gọi đệ quy với một $mask$ có i số 0 liên tiếp. Ta thấy là $totalcall(1) = totalcall(2) = 1$. Còn với $n > 2$ thì $totalcall(n) = totalcall(n-1)+totalcall(n-2)$. Ta dễ dàng thấy là hàm $totalcall$ là hàm fibonacci. Vì thế, với một $mask$ có i số 0 thì ta sẽ có cận trên là F_i với F_i là số fibonacci thứ i.

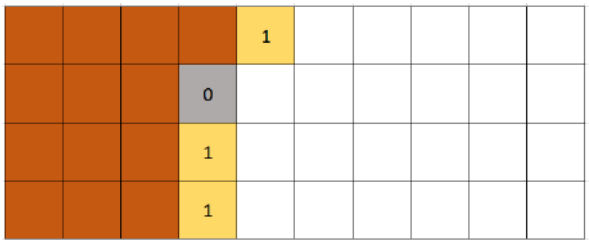
Ta thấy là có tất cả $\binom{n}{k}$ $mask$ có k số 0 với $mask$ n bit. Vì thế, cận trên của số lần gọi đệ quy của ta sẽ là $m \cdot \sum_{k=0}^{n} \binom{n}{k} \cdot F_k$ mà hàm $Fibonacci$ có tính chất là có mối quan hệ chặt chẽ với tỉ lệ vàng, hay, $F_k \approx \phi^k$. Sử dụng nhị thức Newton, ta tính được cận trên của số lần gọi đệ quy sẽ là $m \cdot \sum_{k=0}^{n} \binom{n}{k} \cdot \phi^k = m \cdot (1+\phi)^n$. Hay độ phức tạp cuối cùng của chương trình chạy là $O(N \cdot M \cdot 2^N + M \cdot (1+\phi)^N)$.

Tối ưu thứ hai

Ở đây, ta nghĩ đến việc cố gắng khử đệ quy và loại bỏ đi vế $M \cdot (1+\phi)^N$ trong độ phức tạp. Nhưng việc này không hề dễ dàng như ta nghĩ do với mỗi $mask$ để mà tìm $nextmask$ không dùng đệ quy thì không phải việc dễ dàng. Vì thế, thay vì với mỗi $mask$ ta tìm mỗi $nextmask$ để cập nhật thì ta sẽ lấp đầy bảng từ dòng một và từng cột một luôn thay vì phải đề quy để lấp đầy cả cột. Nhưng, nếu ta muốn làm thế thì phải định nghĩa lại công thức quy hoạch động của chúng ta. Gọi $dp_{i,j,mask}$ là số cách lấp đầy $i-1$ cột đầu của bảng, cột thứ i được lấp đầy j dòng và trang thái của $j-1$ dòng đầu cột $i+1$ và $N-j+1$ dòng cuối của cột i khi ghép lại là $mask$. Xem hình bên dưới để hiểu rõ hơn.



Trạng thái của $dp_{6,3,10110}$ trên bảng 5×10



Trạng thái của $dp_{4,2,1011}$ trên bảng 4×10

Khi ta biến đổi định nghĩa công thức dp của chúng ta như thế, ta có thể dễ dàng tìm ra công thức quy hoạch động hơn.

Nếu ô $(i, j+1)$ đã được lấp thì tương đương với việc là ta không thể làm gì nên $dp_{i,j+1,mask1} += dp_{i,j,mask}$ với $mask1$ là $mask$ đã tắt bit thứ $j+1$.

Nếu ô $(i, j+1)$ trống thì ta làm hai việc sau:

- Ta có thể đặt một domino 1×2 ở đây và cập nhật là $dp_{i,j+1,mask2} += dp_{i,j,mask}$ với $mask2$ là $mask$ đã bật bit thứ $j+1$.
- Nếu như ô $(i, j+2)$ trống thì ta có thể đặt một domino 2×1 ở đây và cập nhật là $dp_{i,j+2,mask} += dp_{i,j,mask}$.

Sau khi điền hết cột i thì ta sẽ cập nhật các mask cho cột $i+1$. Hay, $dp_{i+1,0,mask} = dp_{i,n,mask}$ $\forall$ $mask < 2^n$.

Đáp án của ta sẽ là $dp_{m+1,0,0}$ và trường hợp cơ sở của ta sẽ là $dp_{1,0,0} = 1$.

Tham khảo code mẫu:

```cpp
#include <bits/stdc++.h>
using namespace std;
int dp[1009][11][1024];
const int mod = 1e9 + 7;
int n, m;
bool on(int mask, int i) { return mask & (1 << i); }
int main() {
  ios_base::sync_with_stdio(false);
  cin.tie(0);
  cout.tie(0);
  cin >> n >> m;
  dp[1][0][0] = 1;
  for (int i = 1; i <= m; i++) {
    for (int j = 0; j < n; j++) {
      for (int mask = 0; mask < (1 << n); mask++) {
        if (on(mask, j)) {
          dp[i][j + 1][mask ^ (1 << j)] +=
            dp[i][j][mask];

          // bit thu j bat
          dp[i][j + 1][mask ^ (1 << j)] %= mod;
        } else {
          if (j + 1 < n && !on(mask, j + 1)) {
            dp[i][j + 2][mask] += dp[i][j][mask];

            // dat domino doc
            dp[i][j + 2][mask] %= mod;
          }
          dp[i][j + 1][mask ^ (1 << j)] +=
            dp[i][j][mask];

          // dat domino ngang
          dp[i][j + 1][mask ^ (1 << j)] %= mod;
        }
      }
    }
    for (int mask = 0; mask < (1 << n); mask++)
      dp[i + 1][0][mask] = dp[i][n][mask];
  }
  cout << dp[m + 1][0][0];
}
```

Ta dễ dàng nhận thấy là code trên có độ phức tạp là $O(N \cdot M \cdot 2^N)$.

Tối ưu bộ nhớ:

USACO [47] có một cách cài đặt có độ phức tạp ngang ngửa ta đang xét và tối ưu bộ nhớ hơn rất nhiều nhưng cách viết hơi khó hiểu nên đọc giả có thể tham khảo thêm. Thay vì DP lên như ta thì USACO DP ngược từ đó có thể tối ưu bộ nhớ. Định nghĩa hàm quy hoạch động của ta và của USACO y chang nhau.

So Sánh:

Nếu ta nhìn về độ phức tạp trên giấy của cả 4 thuật toán thì thời gian chạy của thuật toán thứ ba phải chậm hơn thuật toán thứ hai. Nhưng tốc độ chạy của thuật toán thứ ba luôn luôn nhanh hơn thuật toán thứ nhất và thứ hai. Vì sao điều này lại xảy ra? Điều này xảy ra do ta bỏ qua một thứ rất lớn là khi ta tính độ phức tạp: cận trên của số lần gọi đệ quy là F_k thì rất có ít $mask$ đạt được đến cận đó. Vì thế, khi áp dụng thì thuật toán thứ ba sẽ nhanh rất nhiều so với thuật toán thứ 2 do sự tối ưu về việc xét các $nextmask$ đủ tốt để vượt qua hằng số cao.

Nếu ta xét thời gian chạy của cả 4 thuật toán cho bài trên thì ta có thể thấy các ưu và nhược điểm của cả 4 thuật toán:

Thuật toán	Thời gian chạy	Nhận xét
$O(M \cdot 4^N)$	TLE	Khá chậm nhưng tốn khá ít bộ nhớ
$O(M \cdot 3^N)$	$0.17s$ nếu dùng for, 0.83 khi đệ quy tìm $nextmask$	Ta thấy ở đây là khoảng cách thời gian giữa việc sử dụng đệ quy và vòng for rất lớn. Đó là vì đệ quy sẽ gây ra hằng số khá cao mặc dù đệ quy trong C++ đã được tối ưu rất nhiều.
$O(N \cdot M \cdot 2^N + M \cdot (1+\phi)^N)$	$0.14s$	Sử dụng đệ quy nhưng vẫn nhanh hơn hướng làm $O(M * 3^N)$. Nó nhanh hơn là do số lượng trường hợp phải xét giảm thiểu đáng kể với việc nhận lại hằng số từ đệ quy. Bộ nhớ sử dụng cũng kha khá.
$O(N \cdot M \cdot 2^N)$	$0.09s$	Nhanh nhất và bộ nhớ dùng tương đối ít cũng như có hằng số khá thấp. Có thể tối ưu bộ nhớ và từ đó thời gian chạy hơn nữa những việc đó không cần thiết.

VNOJ - Domino [48]

- Cho một bảng A có kích thước $N \times M$, mỗi ô có một giá trị nguyên. Hãy tìm cách đặt đúng k domino 2×1 không chồng nhau để tổng các giá trị trên các ô được phủ là đạt giá trị cực đại.
  - $N \leq 4, M \leq 1000, K \leq \dfrac{N \cdot M}{2}$
  - Subtask:
    - 20%: $M \leq 5$
    - 40%: $N \leq 3$
    - 40%: $N = 4$.

Phân tích:

Subtask 1:

Ta thấy là ở đây có nhiều nhất là 20 ô nên ta có thể dễ dàng xét tất cả các trạng thái của cả 20 ô và kiểm tra xem trạng thái như thế có hợp lệ hay không rồi lấy trường hợp tốt nhất. Độ phức tạp của thuật toán trên là $O(2^{N \cdot M} \cdot N \cdot M)$.

[47] https://usaco.guide/adv/dp-more?lang=cpp#dp-on-broken-profile
[48] https://oj.vnoi.info/problem/bananabread_domino

Subtask 2:

Lúc này, bài toán của ta đã thỏa được cả 4 điểm nhận diện của DP Broken Profile. Nhưng, khác với bài toán 1, ở đây ta chỉ cần điền đủ k domino chứ không cần điểm hết cả bảng. Vì thế, ta phải nhìn lại 4 hướng làm của ta. Nếu ta làm theo hướng thứ nhất thì ta sẽ gọi $dp_{take,i,mask}$ là đã đặt $take$ viên domino, xét đến hết $i-1$ cột và cột thứ i có dạng là $mask$. Ta xét hai trường hợp:

- Nếu ta đặt domino dọc thì $mask$ sẽ không ảnh hưởng đến $nextmask$ nên ta sẽ đặt tùy ý và chỉ cập nhật có $mask$. Công thức quy hoạch động của ta sẽ là $dp_{take+1,i,mask1} = max(dp_{take+1,i,mask1}, dp_{take,i,mask} + A_{i,j} + A_{i+1,j}$ Nếu ta đặt domino dọc ở ô (i,j) và ô $(i+1,j)$.
- Nếu ta đặt domino ngang thì $mask$ và $nextmask$ chắc chắn không được trùng nhau bit nào hay ta có viết là $mask \& nextmask = 0$. Hơn nữa, thay vì đặt mỗi lần một viên domino thì ta có thể đặt nhiều viên cùng lúc. Vì thế, công thức quy hoạch động của ta sẽ là $dp_{take+placed,i+1,nextmask} = max(dp_{take+placed,i+1,nextmask}, dp_{take,i,mask} + S)$ Với $placed$ là số viên domino đã được đặt và S là tổng các ô được đặt domino.

Nếu ta tính sơ độ phức tạp thì nó sẽ là $O(K \cdot M \cdot 4^N \cdot N)$ và sẽ là $O(K \cdot M \cdot 4^N + 2^N \cdot N)$ nếu ta tính trước S. Nếu ta thử các giới hạn thì nó đủ nhanh để vượt qua subtask này.

Subtask 3:

Ở subtask này, nếu ta làm theo hướng thứ ba thì không đủ nhanh. Ta thấy là sự chênh lệch giữa số $nextmask$ có thể đồng bộ với $mask$ và số $nextmask$ có thể là không đủ lớn để ta đánh đổi và sử dụng đệ quy. Thêm với việc là N ở đây tương đối bé nên số lần gọi đệ quy và hằng số của nó có khi sẽ tệ hơn hướng làm vừa được nói. Vì thế, hướng làm thứ 3 có thể đủ nhanh để có thể ăn hết sub 3 nhưng chắc chắn không đủ nhanh để có thể AC bài.

Vì thế, ta phải lam theo hướng thứ tư. Ta sẽ gọi công thức quy hoạch động như nãy chỉ thêm chiều K là số viên domino đã đặt thôi. Hay, gọi hàm quy hoạch động là $dp_{take,i,mask,j}$ là tổng lớn nhất khi đã đặt $take$ viên domino, xét đến cột thứ i và dòng thứ j và $mask$ được định nghĩa như ở trên. Công thức quy hoạch động của ta sẽ không khác mấy so với công thức quy hoạch động ở trên. Chỉ thêm trường hợp không đặt domino và biến bài toán đếm thành toán toán tìm tổng cực đại là xong.

Lưu ý, nếu ta lưu cả 4 chiều thành một mảng toàn cục thì sẽ không đủ bộ nhớ để lưu. Vì thế, ta phải tối ưu một chiều. Ta nhận thấy là tùy theo cách cài đặt thì ta có thể tối ưu chiều K hoặc chiều M do ta có thể cài đặt sao mà ta chỉ xử lý $take$ và $take+1$ tại một thời điểm hoặc j và $j+1$ tại cùng một thời điểm. Nên ta có thể tối ưu một chiều thành 2 và tối ưu bộ nhớ. Cài đặt bên dưới sẽ tối ưu chiều M.

Tham khảo code mẫu $O(K \cdot M \cdot N \cdot 2^N)$:

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;
ll dp[2001][5][16][2];
ll arr[4][1009];
ll n, m, k;
const ll INF = -1e18;
bool on(ll mask, ll i) { return (mask & (1 << i)); }
int main() {
  ios_base::sync_with_stdio(false);
  cin.tie(0);
  cout.tie(0);
  cin >> n >> m >> k;
  for (int i = 0; i < n; i++)
    for (int j = 1; j <= m; j++) cin >> arr[i][j];
  ll ans = -1e18;
  memset(dp, -0x3f, sizeof(dp));
  dp[0][0][0][1] = 0;
  for (int j = 1; j <= m; j++) {
    for (int take = 0; take <= k; take++) {
      for (int i = 0; i < n; i++) {
        for (int mask = 0; mask < (1 << n); mask++) {
          if (dp[take][i][mask][j & 1] <= INF)
            continue;
          if (!on(mask, i)) {
            if (take + 1 <= k) {
              if (j + 1 <= m) {
                // dat domino ngang
                dp[take + 1][i + 1][mask ^ (1 << i)]
                  [j & 1] = max(
                    dp[take + 1][i + 1]
                      [mask ^ (1 << i)][j & 1],
                    dp[take][i][mask][j & 1] +
                      arr[i][j] + arr[i][j + 1]
                  );
              }
              if (!on(mask, i + 1) && i + 1 < n) {
                // dat domino doc
                dp[take + 1][i + 2][mask][j & 1] = max(
                  dp[take + 1][i + 2][mask][j & 1],
                  dp[take][i][mask][j & 1] +
                    arr[i][j] + arr[i + 1][j]
                );
              }
            }

            // khong dat domino
            dp[take][i + 1][mask][j & 1] = max(
              dp[take][i + 1][mask][j & 1],
              dp[take][i][mask][j & 1]
            );
          } else {
            // khong dat domino
            dp[take][i + 1][mask ^ (1 << i)][j & 1] =
              max(
                dp[take][i + 1][mask ^ (1 << i)]
                  [j & 1],
                dp[take][i][mask][j & 1]
              );
          }
```

```
        }
      }

      for (int mask = 0; mask < (1 << n); mask++) {
        dp[take][0][mask][(j + 1) & 1] =
          dp[take][n][mask][j & 1];
      }
    }

    for (int take = 0; take <= k; take++)
      for (int mask = 0; mask < (1 << n); mask++)
        for (int i = 0; i < n; i++) {
          // toi uu bo nho
          dp[take][i][mask][j & 1] = INF;
        }
  }
  cout << dp[k][0][0][(m + 1) & 1];
}
```

Lời Kết

Vừa rồi ta đã đi nghiên cứu và áp dụng những hướng tiếp cận khi làm bài toán DP Broken Profile:

	Hướng tiếp cận "trâu"	Hướng tiếp cận đệ quy	Hướng tiếp cận tối ưu nhất
Thời gian	$O(N \cdot M \cdot 4^N)$ có thể tối ưu thành $O(M \cdot 4^N)$ hoặc $O(M \cdot 3^N)$ tùy theo bài	$O(N \cdot M \cdot 2^N + C(N) \cdot M)$. $C(N)$ là số lần gọi đệ quy và sẽ tùy vào mỗi bài nên rất khó để tính chính xác độ phức tạp.	$O(N \cdot M \cdot 2^N)$
Bộ Nhớ	Rất ít	Tương đối nhiều	Ít
Nhận xét	Thường rất dễ suy nghĩ ra và làm bàn đạp để làm hai hướng kia.	Tùy theo bài cụ thể, hướng này có thể tốt hơn hoặc tệ hơn hướng làm trâu. Ta thường suy nghĩ tối ưu trâu như này.	Thường sẽ là cách tối ưu nhất và là cách giải của người ra đề. Thời gian chạy nhanh nhất và chỉ hi sinh rất ít bộ nhớ.

Ta thấy rằng mặc dù bài toán DP Broken Profile khá khó nhưng nó cũng chỉ xoay quanh một vài hướng tiếp cận cụ thể. Hơn nữa, tuỳ theo bài và dạng bài thì mỗi hướng sẽ có ưu điểm riêng như hướng tiệp cận thứ ba rất nhanh trong bài toán mở đầu bài nhưng lại không tốt trong bài toán áp dụng do điểm yếu là hằng số khá cao. Ngược lại, hướng tiếp cận đầu tiên trên lý thuyết khá chậm nhưng bài toán áp dụng cho thấy là sự đánh đổi thời gian và không gian của hướng tiếp cận này phù hợp cho bài thứ hai. Vì thế, mỗi người cần cân nhắc về các ưu và nhược điểm của mỗi tiếp cận và áp dụng hợp lý cho mỗi bài.

Xin chúc các bạn luyện tập tốt!

Bài tập vận dụng:

- UVA 10359 - Tiling [49]
- UVA 10918 - Tri Tiling [50]
- SPOJ - GNY07H [51]
- SPOJ - M5TILE [52]
- SPOJ - DOJ1 [53]
- SPOJ - DOJ2 [54]
- VJUDGE - MOSAIC [55]
- TIMUS - FORMULA1 [56]
- CF[57] 845F - Guards In The Storehouse [58]

Tham Khảo:

- EVILBUGGY's Blog [59]
- CSES DP Section Editorial (Part 2) [60] - Mục Counting Tillings
- USACO - DP on Broken Profile [61]

[49] https://onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&page=show_problem&problem=1300
[50] https://onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&page=show_problem&problem=1859
[51] https://www.spoj.com/problems/GNY07H/
[52] https://www.spoj.com/problems/M5TILE/
[53] https://www.spoj.com/problems/DOJ1/
[54] https://www.spoj.com/problems/DOJ1/
[55] https://vjudge.net/problem/UVALive-4608
[56] https://acm.timus.ru/problem.aspx?space=1&num=1519
[57] CF: Codeforces – một trang web tổ chức các cuộc thi lập trình thi đấu với các dạng đề, bài tập đa dạng.
[58] https://codeforces.com/contest/845/problem/F
[59] https://coderevilbuggy.blogspot.com/2018/05/broken-profile-dynamic-programming.html
[60] https://codeforces.com/blog/entry/111675
[61] https://usaco.guide/adv/dp-more?lang=cpp#dp-on-broken-profile

Phỏng vấn Hoàng Quốc Việt

Interviewer: Vương Hoàng Long – ICPC World Finalist 2021

Chào tất cả các bạn, hôm nay mình rất là vinh dự khi được phỏng vấn bạn Hoàng Quốc Việt, 'trùm' của một team nhiều thành viên nhất trong VNOI - team Bedao. Bạn có thể giới thiệu sơ lược về bản thân được không?

Xin chào mọi người, em là Hoàng Quốc Việt, nếu mọi người để ý thì tên của em trùng với tên đường ở ngoài Hà Nội. Hiện em đang học ở Trường Đại học Công nghệ, ĐHQGHN (UET) và cũng đã gắn bó với VNOI từ kỳ TNV gen 2. Sở thích của em thì ngoài chơi game như nhiều bạn khác thì em còn rất thích đá bóng nữa ạ. Ngoài ra, em cũng chơi nhiều game ạ, ai rủ em chơi gì thì em chơi nấy. Do em đang ở cùng bạn Vũ (Nguyễn Hoàng Vũ - IOI[62] 2021) nên em hay chơi Liên Minh với bạn ý, trước đây khi ở cùng với các bạn khác thì em chơi Valorant.

Vậy còn về sở thích đá bóng thì bạn có hay đi đá bóng thường xuyên không?

Em đi đá bóng khá thường xuyên ạ, thường thì 1-2 tuần em đá 1 lần. Em đá thì hay 'lên công về thở' nên đá tiền đạo. Tuy làm tiền đạo nhưng em cũng không hay ghi bàn lắm, cơ mà em cũng thường chọn vị trí để mở ra nhiều cơ hội cho đồng đội có những đường chuyền thuận lợi ạ.

Bạn là một trong số những bạn không xuất phát từ những thành phố hay tỉnh lớn nhưng lại có hoạt động trong CP[63] vô cùng ấn tượng như giải Nhì HSGQG[64] . Bạn có thể chia sẻ hành trình thi HSGQG của bạn nhân dịp Bộ Giáo Dục vừa công bố kết quả kỳ thi HSGQG năm học 2023 - 2024 được không?

Năm nay thì tỉnh Hưng Yên cũng tự hào khi có một giải Nhì ạ, về hành trình thi HSGQG thì em có xuất phát khá chậm so với các bạn khủng là phải đến khi lớp 10 em mới được tiếp cận CP. Sau đó lớp 11 thì em có đậu đội tuyển tỉnh để thi QG và được giải Khuyến khích do hồi đấy code thuật trâu chưa tốt nên em cũng bị bug và mất nhiều điểm. Khi em lớp 12 thì đề thi năm đó 'may mắn' là có bài hình nên nhiều bạn toang nhưng em lại làm tốt nên được giải Nhì. Cụ thể là ở bài hình năm đấy em trâu được cỡ 60% bài. Từ đợt đấy nên em rút ra kinh nghiệm khi đi thi HSGQG đấy là đi thi không nhất thiết phải là người giỏi nhất, chỉ cần cẩn thận thôi thì ắt sẽ giải cao.

Việc từ giải Khuyến khích lên giải Nhì là vô cùng ấn tượng, bạn có thể chia sẻ quá trình luyện tập của bạn để có bước nhảy vọt như thế được không?

Đợt em thi năm lớp 12 thì ảnh hưởng bởi dịch Covid khá là nặng nên khá ít hoạt động trên trường. Hầu hết thời gian chủ yếu là em học online nên em chủ động nhắn tin và hỏi bài các bạn khác, do có sự giúp đỡ của các bạn nên em khi làm bài cũng có được nhiều kiến thức hơn. Điển hình là em cũng hỏi bạn Trung (Phạm Xuân Trung - giải Nhất HSGQG năm 2021 - 2022) khá nhiều về bài luồng hay các bài khó để từ đó em học hỏi từ bạn nhiều hướng giải và suy nghĩ kĩ hơn, code cũng cẩn thận hơn nữa. Theo em thấy trong một kỳ thi có nhiều người khủng như thi HSGQG thì không thể vào thi mà dành thời gian nghĩ cả giờ được mà phải cố gắng cắn trâu để dành được nhiều điểm nhất có thể.

Trong hành trình đến với VOI như thế, bạn thường luyện tập trên những nền tảng nào để nâng cao khả năng code trâu của mình?

Hồi đấy trừ làm những bài dễ ra thì em làm cũng khá nhiều bài Codeforces[65] , nếu tính cả bài dễ thì từ trước đến giờ em làm cũng phải khoảng gần 2000 bài. Sau khi VNOJ có thêm nhiều nội dung mới thì

[62] IOI: Kỳ thi Olympic Tin học Quốc tế.
[63] CP: Competitive Programming - Lập trình thi đấu
[64] HSGQG: là viết tắt của kỳ thi Học sinh giỏi Quốc gia.
[65] Codeforces: một trang web tổ chức các cuộc thi lập trình thi đấu với các dạng đề, bài tập đa dạng.

em có chuyển sang làm, ví dụ như Bedao Contest, các dự án sinh test kì thi chính thức.

Năm lớp 12 bạn được giải Nhì như thế thì có phải chuẩn bị để thi tốt nghiệp không hay được vào thẳng Vòng 2 (TST[66])?

Đợt đấy lấy top 32 thì em khoảng top 40-50 nên em không vào được Vòng 2. Sau khi thi xong thì em quay sang thi tốt nghiệp, để cho dễ thì em học khối KHXH nên từ lớp Toán em được chuyển sang lớp Văn để học cùng các bạn nên là lúc đấy em còn muốn thi tốt nghiệp hơn cả TST. Ngoài ra, do thời gian học đội tuyển thì em tiếp xúc hầu hết là con trai, nên khi chuyển xuống lớp Văn học thì cũng bù đắp cho quãng thời gian đó thôi ạ. Thật ra có một số câu chuyện mà khi học đội tuyển em cũng không biết được, nên khi về lại lớp học thì em cũng có cơ hội kết nối lại với các bạn cùng lớp và làm quen thêm được nhiều bạn mới, em nghĩ đấy cũng là trải nghiệm khá là khác so với thi TST.

Bạn có thể chia sẻ quá trình nhập học đại học của bạn được không? Gia đình bạn có ủng hộ những quyết định của bạn không?

Sau thi tốt nghiệp thì em cũng được tuyển thẳng vào UET do có giải QG nên sau đó em có chọn ngành CNTT (CN1) ạ. Theo em được nghe thì ngành này được ít người biết đến so với ngành Khoa học máy tính của UET, do nghe qua học CNTT thì người ta tưởng là đi cài win dạo, nhưng mà theo em quan sát được thì có rất nhiều người khủng vào ngành này nên em cũng vào luôn. Về trường thì nhiều người khi hỏi UET thì họ không biết là trường nào, họ còn hỏi lại có phải là Học viện Bưu chính Viễn thông (PTIT) không nên em thấy trường rất ít được nhiều người biết tới. Nhưng em may mắn là có mẹ hoàn toàn ủng hộ em, nên mọi quyết định em đều tự tìm hiểu còn gia đình chỉ đứng sau ủng hộ em thôi ạ. Việc em chọn UET mà không chọn những trường khác như Bách Khoa Hà Nội (HUST) thì chắc chắn do sợ môn đại cương rồi. Nhưng mà đấy cũng không phải lý do chính mà là em vào UET để thi ICPC[67] do em được biết trường rất đầu tư vào ICPC, nên nhiều bạn mạnh vào trường cũng giúp em có thể thoải mái lập team hơn, ngay trước khi vào UET là em đã có team rồi ạ.

Kỳ thi ở Huế là năm thứ 2 bạn thi ICPC, bạn có thể kể về hành trình luyện tập và thi ICPC của bạn trong 2 năm qua được không?

Năm nhất thì bọn em làm bài nhiều hơn thì lại không được thi ICPC Regional ở TPHCM, qua năm hai thì bọn em cũng làm nhiều bài nhưng không làm nhiều như kỳ vọng, thế thì năm đấy lại được thi ICPC Regional ở Huế. Thật ra cũng do thiên thời, địa lợi, nhân hòa nữa ạ, do năm đó UET lấy số đội tham dự cũng nhiều nên đội được tham dự. Trong lúc train thì bọn em có xoay vai trò của nhau, nhất là vị trí leader khi mà mọi người sẽ thay phiên 'chỉ đạo' trong từng buổi luyện tập. Mỗi buổi như thế team sẽ cùng nhau làm 5 tiếng và sau đó thảo luận về bài và cùng nhau giải lại những bài chưa làm được. Việc luyện tập thì làm một mình sẽ chủ động thời gian hơn nhưng theo em thấy thì sẽ không có tinh thần như khi làm chung với team vì nhiều lúc có một số chỗ mình cần hỏi thì không có ngay đồng đội ngay cạnh để giải đáp được. Cường độ luyện tập thì bọn em sẽ làm theo team 1 tuần/1 lần, còn làm cá nhân thì tùy thời gian biểu của mỗi người mà bọn em làm bài trên Codeforces thôi ạ.

[66]TST: Kỳ thi tuyển chọn đội tuyển dự thi Olympic Tin học Quốc tế, hay còn được biết đến là Vòng 2 thi chọn Học sinh giỏi Quốc gia.

[67]ICPC: International Collegiate Programming Contest - Cuộc thi Lập trình Quốc tế lâu đời và danh giá nhất dành cho sinh viên các trường đại học và cao đẳng trên toàn cầu.

Bạn có kỉ niệm gì ở lần tham gia thi ICPC Regional ở Huế không?

Kỉ niệm em nhớ nhất chắc là món chè heo quay, em không biết tại sao có món đấy tồn tại nhưng em thấy ăn cũng được. Để tưởng tượng thì món đấy như lấy bột canh trộn với đường xong cho thêm bánh như há cảo vào trong chè để ăn cùng. Ngoài ra ở Huế thì em thấy nhiều cảnh đẹp, con người cũng thân thiện, hơi xui một chút mấy hôm thi thì trúng mấy hôm trời mưa thôi ạ.

Bạn có thể chia sẻ bí quyết cân bằng giữa việc ăn chơi, việc học trên trường và thi ICPC được không? Trong quá trình cân bằng bạn có gặp trở ngại nào không?

Theo em thì mình không cố cân bằng những việc này thì mình không cần cân bằng, em chỉ đơn giản xem những việc này là hoạt động thường ngày diễn ra tự nhiên thôi ạ. Như em chọn tuần nào train ICPC thì tuần đấy sẽ ít việc để làm trên lớp hơn, những ngày đi chơi thì em quyết định sẽ không học trong ngày hôm đấy để hôm sau học bù nhiều hơn. Công việc trong VNOI thì em sẽ sắp xếp từ trước nên em thấy mọi thứ nó là một phần của cuộc sống em luôn, lúc đấy em thấy mọi việc cân bằng sẽ dễ hơn.

Hiện tại thì Việt đang quản lý team Bedao và xuất bản rất nhiều chuỗi contest vô cùng chất lượng trên VNOJ, không biết quá trình từ một TNV lên 'trùm' của team Bedao thì như thế nào?

Đầu tiên khi tham gia vào VNOI em có nguyện vọng là đóng góp cho cộng đồng từ chính những bài tập mình đề xuất lên trang. Sau đấy thì team Bedao có nhu cầu tuyển những Coordinator[68] cho các contest sắp tới nên em trở thành ứng cử viên cho vị trí này, để được đề xuất thì từ trước đến giờ em luôn thấy mình hoàn thành các công việc đúng hạn, tích cực đóng góp cho team nên được mọi người ưu ái. Khi đảm nhiệm vị trí này thì em vẫn duy trì trách nhiệm của một TNV và giám sát các công việc để đảm bảo đúng hạn. Ngoài ra có một số contest toàn bộ bài đều do em tự đề xuất nên phần nào cũng thấy tự hào khi mình đóng góp một phần không nhỏ trong cộng đồng. Theo em thấy dù ở bất cứ vị trí nào thì mình vẫn phải luôn giữ vững tinh thần và trách nhiệm với công việc, trong mọi tình huống thì luôn phải chủ động tại cơ hội sẽ luôn đến với mình, chỉ là có thể mình chưa nhận ra để nắm bắt thôi.

Mình nghĩ việc duy trì trách nhiệm và luôn chủ động trong công việc là khá khó, theo bạn làm cách nào để duy trì tinh thần này?

Trước hết thì việc tham gia vào CLB phải thực sự là đam mê của mình chứ không chỉ tham gia và nhận việc. Khi có tinh thần đó thì mình sẽ xem những công việc và hoạt động đấy trở nên thân quen hơn, nhiều thứ trong đó trở thành một phần của cuộc sống và mọi người trong CLB như gia đình của mình vậy. Từ đó em có thể đóng góp hết tài năng cũng như trí lực của mình nên em sẽ có thể chủ động hơn để đảm nhận những vai trò khác nhau trong công việc.

Là một người vô cùng nhiệt huyết và tận tâm với công việc, bạn có thể giới thiệu qua những công việc trong team Bedao được không? Có bao giờ bạn gặp áp lực về chất lượng contest bị mọi người phản ánh không?

Em thấy có một quan điểm khá đúng là khi càng đảm nhận những vai trò cao hơn thì việc của mình sẽ càng ít đi nhưng theo với đó thì trách nhiệm sẽ càng nhiều. Hàng tuần thì việc làm của em cũng không

[68] Coordinator: người tổ chức các contest cho OJ.

nhiều làm, thường thì trước mỗi tuần em sẽ sắp xếp trước công việc và thời hạn cho các bạn hoàn thành, sau đó đôn đốc các bạn coordinator hoàn thiện những contest đang làm, các bạn coordinator sẽ đốc thúc các bạn chuẩn bị bài tập đã được giao. Chẳng hạn như contest VNOJ Round 1 vừa qua thì em cũng là người đứng ra quản lý đầu việc cho các bạn, thường thì các việc này dành cho coordinator nhưng do contest đầu tiên nên em muốn đảm bảo chất lượng tốt. Ngoài ra thì em còn duy trì thói quen nghĩ ra những ý tưởng mới cho hoạt động của VNOI và team Bedao.

Với quãng thời gian đóng góp nhiều như vậy, theo bạn điều gì xứng đáng để bạn nhận lại sau khi đóng góp rất nhiều cho cộng đồng?

Điều em thấy xứng đáng nhất là số lượng thí sinh quan tâm đến contest trên VNOJ ngày càng tăng, hiện tại các contest duy trì ổn định ở mức 500-700 người, trước đó thì con số này đôi khi chỉ là khoảng 100-200 thôi. Ngoài ra thì các bạn coordinator ngày càng độc lập và trưởng thành hơn trong các quyết định của mình, theo em thấy thì việc đóng góp cho team Bedao là một mối quan hệ win-win, mình cho đi cộng đồng những contest chất lượng và đổi lại là những kỹ năng trước đó mình chưa bao giờ có. Sau khi cân nhắc như vậy thì em thấy công sức của mình rất là xứng đáng khi có thể đưa bản thân và mọi người cùng phát triển.

Bạn có trải nghiệm thú vị hay đáng nhớ nào khi làm ở vị trí này không?

Trải nghiệm đáng nhớ chắc là em 'dí deadline' các bạn nhiều contest liên tục, đến nỗi mà mỗi ngày một deadline. Theo em thấy 'dí deadline' cũng là một nghệ thuật khi mình nhiều lúc phải nhắc khéo các bạn, thay vì báo sắp đến deadline thì em sẽ bảo em tạo xong contest thông báo trên VNOJ rồi. Khi đó các bạn sẽ lên VNOJ kiểm tra và so sánh lại tiến độ công việc của mình với tiến độ chung của mọi người. Nhiều trường hợp em sẽ nhắn tin riêng để chia sẻ công việc với bạn ý, vì thường team Bedao sẽ phân bố các đầu việc hợp lý cho các bạn, không để một bạn đảm nhận nhiều việc quá và một bạn khác thì ít việc quá. Em nghĩ nếu xây dựng môi trường lành mạnh như thế thì các bạn cũng được tạo điều kiện để hoàn thành sớm các công việc.

Bạn cảm thấy trong quãng thời gian dẫn dắt team Bedao thì bạn học được những gì?

Em cũng học được cách giao tiếp cũng như diễn đạt cho mọi người, mình không thể giao tiếp mà chỉ mình hiểu được mà cần diễn đạt rõ ràng để các bạn cùng tiếp nhận. Trước đó thì em bị đánh giá là diễn đạt hơi kém nhưng sau quá trình làm ở Bedao một thời gian dài thì em thấy mình tự tin trong giao tiếp hơn rất nhiều. Ngoài ra thì em còn học một kỹ năng nữa, người xưa hay gọi là 'Binh pháp Tôn Tử', đấy là kỹ năng nhận xét và đánh giá công việc của người khác. Ta sẽ phải quan sát điểm mạnh và điểm yếu của mỗi người để đặt vào vị trí bạn có thể phát huy được tối đa điểm mạnh của mình, như bạn nào vốn giỏi ra đề thì mình sẽ khuyến khích bạn ra đề nhiều hơn, bạn nào giỏi làm tester thì sẽ không cho bạn ý ra đề nhiều mà cho các bạn ý kiểm tra bài tập nhiều hơn, cũng từ đấy mình phát hiện ra những bạn làm việc trách nhiệm để đề cử lên coordinator.

Về tương lai của team Bedao thì bạn có dự tính gì để duy trì Bedao Contest là chuỗi contest chất lượng và uy tín nhất?

Em thấy ở bất cứ môi trường thi cử nào thì sự công bằng nên luôn đặt lên hàng đầu, team Bedao sẽ luôn giữ vững sự công bằng đấy bằng cách mạnh tay loại bỏ những trường hợp chép code hay cố tình gian lận. Sau khi giữ vững sự công bằng thì Bedao sẽ có uy tín nhất định, lúc này em sẽ chú trọng vào

nâng cao chất lượng bộ đề. Bộ đề hiện tại thì vẫn có những điểm yếu nhất định, vẫn tồn tại những bài test yếu, đề sai, ... nhưng trong tương lai thì team Bedao sẽ cố loại bỏ những điều ấy để hướng tới các contest hoàn hảo hơn. Ngoài ra em cũng mong muốn VNOJ duy trì tốc độ chấm bài nhanh trong các kỳ thi để đảm bảo các thí sinh có trải nghiệm thi tốt nhất.

Bạn có những dự định nào dành cho Bedao mà ngay cả các bạn trong VNOI cũng không biết không?

Em có thể tiết lộ là em đang ấp ủ chuyện xem xét các bạn tình nguyện viên nhiệt huyết lên làm Coordinator một vài contest để các bạn thể hiện khả năng quản lý của mình. Điều này nó hơi chuyên môn một chút nên em cũng không chia sẻ nhiều.

Đến với phần cuối, bạn có thể chia sẻ về những dự định cho tương lai, về ngành học và nghề nghiệp bạn muốn theo đuổi được không? Có công ty nào bạn mơ ước vào làm không? Theo bạn, một công ty lý tưởng theo bạn cần hội tụ những yếu tố nào?

Em hiện đang tham gia khá nhiều các workshop liên quan đến định hướng nghề nghiệp như Blockchain[69] và Software Engineering[70] . Em cũng muốn thử sức ở lĩnh vực nào liên quan nhiều đến tài chính một chút tại em thấy tư duy tài chính cũng khá hay, giúp bản thân suy nghĩ mạch lạc hơn. Ước mơ xa hơn của em là được làm startup, nhưng công ty lý tưởng để vào làm thì em cũng có một số cái tên như Pendle Finance, Dytechlab. Em cũng chưa đi làm nhưng dưới góc nhìn của em thì một công việc lý tưởng phải là nơi em thật sự đam mê, luôn sẵn sàng cống hiến hay thức đêm nếu có nhiệm vụ quan trọng. Đấy sẽ là nơi em ở được là một phần của công ty chứ không phải làm cho xong việc rồi nghỉ.

Cảm ơn bạn đã nhận lời và tham gia phỏng vấn ngày hôm nay. Chúc bạn một năm mới nhiều sức khỏe, ngày càng phát triển Bedao và sớm kiếm được thực tập ở các công ty lớn!

[69]Blockchain: công nghệ chuỗi – khối, cho phép truyền tải dữ liệu một cách an toàn dựa trên hệ thống mã hóa vô cùng phức tạp.

[70]Software Engineering: Kỹ thuật/Công nghệ phần mềm.

Phỏng vấn Nguyễn Đức Thắng

Interviewer: Vương Hoàng Long – ICPC World Finalist 2021

Chào mọi người, hôm nay VNOI rất vinh dự khi được phỏng vấn bạn Thắng, một hiện tượng trong làng CP[71] của năm 2023 khi các cuộc thi bạn tham gia đều đạt thành tích rất cao. Bạn có thể giới thiệu về bản thân được không?

Em là Nguyễn Đức Thắng, hiện đang là học sinh lớp 12 chuyên Toán, trường Trung học Phổ thông Chuyên Hùng Vương, tỉnh Phú Thọ. Vài năm trước, trường em cũng có một anh tên giống hệt em, cũng học chuyên Toán, và cũng nhất Quốc Gia Tin! Em thấy cái này trùng hợp quá mức luôn. Sở thích của em là xem bóng đá, thi thoảng cũng có ra sân đá cùng bạn bè cũng như chơi các game bóng đá như FC Online. Dạo này em cũng có một sở thích khác là chơi sudoku. Ngoài ra, em cũng thích xem phim (đặc biệt là những phim hành động như Mission Impossible, Điệp Viên 007 và Jason Bourne). Giải khối Rubik cũng là một phần đam mê của em ạ.

Được biết bạn là một fan cuồng của Liverpool, bạn thích điều gì ở Liverpool?

Em là fan Liverpool từ năm 2018 - năm mà Liverpool thua chung kết Champions League với Real Madrid. Thông thường thì mọi người sẽ thích các đội vô địch, nhưng em lại ấn tượng với lối chơi Gegenpressing, điệu Rock rực lửa của Klopp rồi sau đó trở thành fan lúc nào không hay. Một câu nói em rất thích của Klopp đó là "I'm the normal one". Ngoài ra tinh thần máu lửa mà các cầu thủ, ban huấn luyện cũng như cổ động viên của họ mang đến luôn đem đến cho em rất nhiều cảm xúc. Đỉnh cao là năm 2019, nơi mà Liverpool đã có cuộc lội ngược dòng không tưởng trước Barca trên sân Anfield tại bán kết cúp C1. Em đã không thể tin vào mắt mình khi xem bàn thắng thứ tư "corner taken quickly" của Liverpool ở trận đấu hôm đó (lời bình luận nhắc tới việc Trent Alexander-Arnold đá quả phạt góc nhanh, đánh lừa hàng thủ Barca, kiến tạo cho Origi ghi bàn). Em khá buồn vì đây là mùa cuối của Klopp ở đây. Dù hiện tại đội vẫn dẫn đầu trên bảng xếp hạng, nhưng Liverpool vẫn chưa thể chắc chắn về một chức vô địch, bởi Man City vừa có Kevin De Bruyne và Erling Haaland trở lại sau chấn thương nên tình hình cũng khó lường.

Vốn xuất phát học cấp 3 chuyên Toán trường Hùng Vương – trường chuyên duy nhất và tốt nhất của tỉnh Phú Thọ, với chuyên Toán là chuyên khó vào nhất, bạn đã có những bước đệm nào từ cấp 2 để lên cấp 3 có thể tỏa sáng được như thế không? Bạn có từng tham gia các kỳ thi như Violympic, Toán Quốc Tế, Toán Úc, ... dành cho học sinh từ cấp 2 trở xuống không?

Việc học toán nâng cao của em đã bắt đầu từ những năm cấp 1. Thời đó thì em vào được vòng huyện những năm cấp 1 và vòng tỉnh trong những năm cấp 2. Vòng tỉnh cũng là cấp độ cao nhất dành cho trung học cơ sở, và em may mắn đạt được giải nhì. Về trình độ toán của em, em tự nhận thấy bản thân không giỏi ở phần hình và bất đẳng thức lắm. Khi thi vào chuyên Toán thì em đứng thứ 12/30 - một vị trí khá an toàn nhưng cũng không quá xuất sắc, nên em không hài lòng lắm với kết quả này.

Nhắc đến những cuộc thi, em có tham gia Violympic khi nó vẫn còn là một cuộc thi toán nhận được nhiều sự ủng hộ của học sinh và các thầy cô, và em đều đạt được thành tích tương đối ổn. Ngoài ra, em cũng có thi các cuộc thi "Quốc Tế" đôi lần vào những năm cấp 2 nhưng thành tích đạt được cũng không mấy ấn tượng.

Em nghĩ rằng một nền móng toán học ở mức "khá" như vậy là đủ để giúp em hiểu được những kiến thức toán học được áp dụng trong các chuyên đề tin.

[71] CP: Competitive Programming - Lập trình thi đấu

Tại sao bạn thi vào lớp chuyên Toán mà không phải lớp chuyên Tin? Tại sao bây giờ bạn lại học Tin mà lại không học toán?

Em xuất phát vốn là dân học Toán, nên thi vào chuyên Toán cũng là dễ hiểu. Thực ra bản thân em cũng chán việc học Toán từ lớp 8 do em rất kém hình, và em cũng thấy mình không còn hứng thú với việc biến đổi quá nhiều. Lên cấp 3, em mới bắt đầu thử học Tin. Thật vậy, giả sử em học Tin kém mà vào lớp chuyên Tin thì ... không ổn lắm. Nếu em có cơ hội được chọn lại giữa việc học chuyên Toán hay chuyên Tin, thì em vẫn sẽ chọn Toán bởi tính cách của em vẫn hợp với các bạn lớp em bây giờ hơn. Em nghĩ đây cũng là background của khá nhiều người học cũng như thi Tin ạ.

Ở cấp 3 thì bạn tiếp xúc với môn Tin từ lúc nào?

Khoảng hồi hè trước khi vào cấp 3, cô Thái dạy trường em có nghĩ ra ý tưởng và mở một lớp dạy C++ miễn phí cho các bạn học sinh mới vào trường, chủ yếu là để đào tạo vào ĐTQG sau này. Ban đầu cũng không ai biết và để ý em bởi em không có gì nổi trội. Nhưng sau khi học được một thời gian thì có vẻ em bứt lên. Thực ra thì Tết năm đó, em cũng có tìm hiểu một chút qua về C++ nên tiếp thu nhanh hơn. Vả lại, em lúc ấy cũng rất đam mê với việc code, em code suốt cả hè luôn. Đến giờ em vẫn không hiểu sao hồi đó mình code nhiều thế.

Cảm giác của bạn khi lần đầu tiếp xúc với Tin học?

Lần đầu thì có lẽ là năm lớp 8, khi em được học Pascal trên trường. Lúc đấy được tiếp xúc với các câu lệnh, vòng lặp và một bài vài toán đơn giản làm em thấy việc lập trình khá thú vị. Tin học trong em lúc đó là cả một khoảng trời của những kiến thức hoàn toàn mới lạ và hay ho. Thậm chí, em từng có ý định học song song cả Toán và Tin bằng cách mượn đề các đứa bạn đội tuyển Tin của em để làm cùng, nhưng hồi đó em quá lười nên cuối cùng lại không làm được.

Với việc đạt giải Nhì Quốc Gia từ năm lớp 10 như thế đã cho bạn bàn đạp rất lớn khi được dành thời gian trọn vẹn 3 năm cấp 3 với môn Tin học. Bạn có thường xuyên phải học trên trường nữa không?

Trong khoảng thời gian em học đội tuyển để đi thi HSGQG[72] , trường cũng đã tạo điều kiện miễn việc học trên lớp cho em. Sau khi thi HSGQG xong, cô Thái đã xin phép trường cho em được tiếp tục việc học cùng với các anh chị ạ. Sự thật là hồi đấy chỉ có 7 anh chị lớp 11-12 thi mà trường có 8 suất nên em được vé vớt (cười). Tuy nhiên ở vòng sau em lại khá đen đủi khi đứng đồng hạng 33 và trượt Vòng 2.

Phương pháp nào để bạn từ giải Nhì Quốc gia năm lớp 10 lên Á khoa của kỳ thi chỉ sau một năm như vậy? Bạn có thay đổi gì về chiến lược học trong suốt một năm hay không?

Em nghĩ sự khác biệt lớn nhất trong 2 năm là việc em đã có ý thức stress test trong phòng thi hơn. Hồi lớp 10 thì em có phần tập trung vào việc luyện code trên VNOJ hơn mà quên mất phần chiến thuật

[72]HSGQG: là viết tắt của kỳ thi Học sinh giỏi Quốc gia.

trong phòng thi. Kết quả là em đã bị "sập" mất bài 2 năm đó và trượt - một kết quả làm em khá nuối tiếc. Sau đó em chơi nguyên hè, cho tới khi "được" khuyến khích kì thi Duyên Hải Đồng bằng Bắc Bộ. Đến lớp 11 thì em cẩn thận hơn, cộng thêm trình độ code đã tiến bộ một chút, nên kết quả tốt hơn cũng là dễ hiểu. Vả lại, em quan niệm rằng Vòng 1 chỉ cần code giỏi là vào được vòng sau rồi. Kết quả Á khoa có thể mang chút may mắn, nhưng việc có được một kết quả tốt đã nằm trong dự liệu của em khi đó rồi.

Bạn tập trung rất nhiều vào khả năng code, vậy còn khả năng nghĩ thì sao? Bạn phân bố thời gian học các thuật toán như thế nào?

Lúc mới vào thì em chỉ học các thuật toán cơ bản như Segment Tree, DSU, ... thôi ạ. Ngoài ra khi làm bài mà gặp một tính chất, một thuật toán mà em chưa biết thì em sẽ tìm đọc và làm những bài tương tự. Ngoài ra khi em lên lớp 11, với việc biết sinh test, tỉ lệ code bug của em đã giảm xuống đáng kể. Khả năng nghĩ của em thì cũng ở mức tạm ổn, chưa có gì quá khác biệt; Em hơn các bạn ở khả năng làm được hầu hết những phần có thể làm được trong các kì thi.

Bạn học TST[73] trong vòng bao lâu? Trình độ của bạn thay đổi như thế nào?

Em nhớ là trong vòng vài tháng ạ. Em lên trình khá nhiều, vì các bài được tiếp xúc khác hẳn hồi ôn thi VOI, với nhiều kiến thức, dạng bài mới. Có nhiều bài thiên về suy nghĩ, sau khi đã ra ý tưởng thì việc cài đặt lại khá đơn giản. Mặc dù thế mạnh của em là code những bài nặng về cài đặt, nhưng em vẫn rất hứng thú với các dạng bài nghĩ nhiều. Hồi em học thì anh Thái, thầy Đông thường cho các bài code ngắn, ngược lại thì có anh Hoàng luôn cho những bài cần "tay to" thực sự.

Có một số người học Vòng 2 rất quyết liệt, bài nào các thầy đưa ra cũng code để AC bằng được. Tiếp cận của bạn cho Vòng 2 là gì?

Thật ra thì cô Thái cho em đi ôn riêng với các thầy, các anh khá nhiều nên em dành phần lớn thời gian để giải bài từ những buổi đó ạ, còn bài các thầy cho ở Vòng 2 thì em có cố gắng làm nhưng những bài em không làm được thì thôi ạ. Vào vòng 2, em chỉ cố gắng để có được nhiều điểm nhất có thể và không đi quá sâu vào bài nào. Việc có được một thứ hạng cao như vậy thật sự cũng là một bất ngờ với em vì mỗi thí sinh đều có ít nhất 1 bài xanh, em thì bài nào cũng màu đỏ thôi..

Trước khi thi Vòng 2, bạn có cảm thấy hồi hộp không? Sau đó cả hai ngày thi của bạn diễn ra như thế nào?

Em cảm thấy khá là bình tĩnh, vì mục tiêu của em ban đầu chỉ là vào APIO[74] để miễn tốt nghiệp, chứ không thiết tha gì IOI[75] cho lắm. Mà nếu lớp 11 có lỡ không may trượt vòng 2, thì lớp 12 vẫn thi lại VOI được ạ. Em thấy bản thân làm hai ngày ở mức khá bình thường. Ngoài ra, điểm của em cũng khá đều, còn mọi người thì hay có 1-2 bài điểm có để gánh các bài còn lại.

[73] TST: Kỳ thi tuyển chọn đội tuyển dự thi Olympic Tin học Quốc tế, hay còn được biết đến là Vòng 2 thi chọn Học sinh giỏi Quốc gia.

[74] APIO: Kỳ thi Olympic Tin học Châu Á -– Thái Bình Dương.

[75] IOI: Kỳ thi Olympic Tin học Quốc tế.

Vậy chiến thuật của bạn có phải là tập trung cắn tất cả các subtask thay vì chỉ tập trung là một bài không?

Đối với em, đề năm vừa rồi khó hơn bình thường khá nhiều nên việc tập trung vào một bài có phần bất khả thi, trừ khi đó là anh Khuê (Hoàng Ngọc Bảo Khuê - hai lần tham gia APIO 2022, 2023). Khi ngày 1 kết thúc và em chỉ được chưa tới 100/300, em cũng đã nghĩ về một thứ hạng thấp. Tuy nhiên em lại có được vị trí thứ 7, điều này giúp em làm bài ngày 2 với một tâm thế khá thoải mái.

Sau khi qua Vòng 2 thì các bạn ở các tỉnh sẽ được ra Hà Nội học một tháng và ở kí túc xá kèm tiền hỗ trợ 250 nghìn mỗi ngày, bạn có kỉ niệm nào với việc học APIO không?

Em nhớ nhất kỉ niệm... trốn học ạ. Phòng em có 3 người: em, anh Bảo Anh và anh Triệu. Anh Triệu thì chăm chỉ, hôm nào cũng đi học đầy đủ. Còn anh Bảo Anh và em thì hay trốn, sau đó cũng có vài lần bị thầy Phương gọi hỏi thăm. Lúc trốn, em thường chỉ ở nhà ngủ thôi nên nhiều lúc thầy gọi cho cả cô Thái để kêu dậy đi học. Với tiền phụ cấp, em cũng chỉ đi ăn uống bình thường, không chi tiêu gì nhiều. Có một hôm ba bọn em đi xem phim cho đỡ chán, còn lại thì không khác gì sinh hoạt hằng ngày cả.

Ôn thi xong APIO là đến ôn thi IOI, trải nghiệm của bạn lúc đấy thế nào?

Đợt em ôn thi IOI thì ba bạn kia ở Hà Nội nên ở nhà riêng hết, chỉ có mình em ở kí túc xá thôi, mà cũng chỉ vì có 4 người học nên em cũng không trốn được. Thật ra nếu em thật sự muốn trốn thì có lẽ vẫn có thể, bởi vì các thầy chỉ cho bài thôi, đa phần các thầy sẽ không lên. Mọi người lên đầy đủ chủ yếu chỉ để học cùng nhau để có không khí cũng như trao đổi một chút thôi.

Kỳ thi IOI 2023 được tổ chức ở Hungary, là một năm hiếm hoi trong những năm gần đây các thí sinh đi thi có thể giao lưu và gặp mặt nhau tại một nước. Cảm giác của bạn khi lần đầu tiên được đi Hungary là như thế nào?

Ở Hungary, các toà nhà có kết cấu từ ngày xưa, có nhiều hoa văn khá độc lạ. Em thấy Hungary đẹp, nhưng cũng chưa phát triển lắm, vì em để ý thấy vẫn còn nhiều khu đất trống. Cảm giác hệt như ở vùng nông thôn vậy. Đợt tụi em đi thi IOI, hội du học sinh Việt Nam ở Hungary hỗ trợ rất nhiều. Ngoài ra còn có hội người Việt Nam ở Hungary tài trợ ở một số mặt nữa ạ. Em vui vì họ rất nhiệt tình, nhưng cũng hơi thắc mắc vì bay gần nửa vòng Trái Đất cuối cùng lại thưởng thức gà luộc, phở bò..., chả khác gì ở nhà cả! Em cũng rất thích Hungary vì đó là quê nhà của số 8 mới ở Anfield (sân nhà đội Liverpool) - Szoboszlai.

Khi bạn bước vào thi IOI, bạn chuẩn bị tâm lý như thế nào?

Đối với em, một khi đã đủ khả năng trở thành một trong bốn đại diện của Việt Nam tham dự IOI, thì làm bài thế nào đi nữa cũng đủ khả năng có giải, trừ khi hôm đó phong độ tệ lắm thôi. Nếu thế thật thì em cũng hi vọng sẽ được huy chương Bạc. Ngày thi đầu tiên, em làm bài suýt nữa thì xuống Đồng. Ngày thi thứ hai thì khá xanh. Đề IOI năm ngoái khi làm lại thì thật sự rất hay, nhưng các subtask thì không được thuận lợi lắm. Ví dụ như bài 1 năm ngoái, em nghĩ được subtask đường thẳng ra được thuật tham lam khá giống sol chuẩn. Tuy nhiên đến subtask $n \leq 3000$, tương ứng với 83 điểm, code chuẩn chỉ vài dòng, nhưng phải dùng Quy hoạch động, khác hẳn với hướng trước đó nên không nghĩ ra thế nào. Em đành an phận với 52 điểm. Bài 3, em rút ra được nhận xét như mọi người, nhưng khi code trâu và kiểm

tra lại thì code bug, làm em nghĩ nhận xét đó sai. Em tưởng bài đấy khó, vậy là chỉ trâu đúng subtask 1 rồi bỏ. Khi ra khỏi phòng thi, ai cũng được ít nhất 50 60 điểm, chỉ em được 14 điểm. Nếu bài đó không bug, kèm thêm bài 1 nữa thì em có thể sẽ lên được Vàng.

Sau khi thi thì cảm giác của bạn như thế nào? Vui vì được huy chương bạc hay tiếc vì không thể đạt được vàng?

Được Huy chương Bạc cũng gọi là hoàn thành mục tiêu của mình rồi, nên em cũng không tiếc nuối nhiều. Khi về lại quê nhà, bố mẹ tổ chức tiệc mừng HCB của em với hàng xóm, có cả các thầy cô cùng tới chia vui nữa ạ. Đạt được Huy chương Bạc cũng đồng nghĩa với việc được huy chương lao động hạng 3. Nếu không có gì thay đổi so với những năm trước, em sẽ được gặp Chủ tịch nước, nhưng năm nay chỉ được gặp Phó chủ tịch nên em cũng hơi tiếc nuối đôi chút.

Trường và tỉnh Phú Thọ có tổ chức tuyên dương, khen thưởng bạn không? Cảm xúc của bạn lúc đó như thế nào?

Em khá là vui, hạnh phúc, và tự hào. Nếu tính tới hiện tại, thì tổng tiền thưởng đã hơn một trăm triệu đồng. Em có dùng một khoản tiền đấy mua một cái laptop mới thôi, chứ cũng chưa có dự định gì nhiều.

Với tâm thế là một người đã có huy chương rồi, thì bạn có kì vọng gì cho Vòng 2 năm nay không?

Chắc chắn là khác năm ngoái rồi ạ. Mọi người sẽ kì vọng năm nay em có vàng IOI. Em nghĩ là năm nay được đại diện Việt Nam đi thi IOI đã là thành công rồi. Năm nay có khá nhiều bạn giỏi nên em không biết mình có thể vào được top 6 (top để được tính huy chương APIO) hay không nữa. Từ khi thi IOI năm ngoái về, em đi chơi khá là nhiều, chỉ từ sau vòng 1 năm nay thì em mới tập trung hoàn toàn lại vào code ạ.

Mình thấy hầu hết các bạn khi chuẩn bị cho IOI đều luyện tập trên Codeforces[76] rất nhiều nhưng bạn thì ngược lại, không cày Codeforces nhiều lắm. Lí do tại sao bạn lại làm thế? Khi không tham gia các contest trên Codeforces thì bạn có hay làm lại các bài trên đó không?

Em khá lười làm Codeforces vì contests Codeforces luôn diễn ra vào buổi tối muộn. Nhiều khi có những bài em có khả năng giải nhưng lại không giải được. Những bài ấy làm em khá cay cú và khó ngủ! Ngoài

[76] Codeforces: một trang web tổ chức các cuộc thi lập trình thi đấu với các dạng đề, bài tập đa dạng.

ra em cũng không thường xuyên làm lại bài lắm. Em sử dụng Codeforces nhiều hồi em mới học, bởi các bài div 2 giúp khả năng tư duy của em được nâng lên khá nhiều .

Nếu không làm Codeforces thì nguồn bài chính của bạn là ở trang nào?

Em cày cuốc khá nhiều trên VNOJ với các bài OI trên oj.uz. Các bài trên oj.uz thì em đã giải tương đối nhiều. Đề thi các năm gần đây thì em làm gần hết rồi.

Khi luyện tập thì trung bình bạn dành bao nhiêu thời gian cho một đề?

Việc dành thời gian bao lâu phụ thuộc vào tâm trạng của em. Nếu em thấy có khả năng nghĩ tiếp được thì sẽ cố gắng tới khi nào ra thì thôi. Còn ngược lại, nếu em thấy chán, hoặc là bài khó quá, thì em sẽ đọc sol và các subtask ở cuối đề ạ. Thường thì em sẽ mở từng bài và cố gắng giải hơn là giải nguyên một đề.

Vậy động lực nào để bạn có thể tiếp tục giải hết các đề IOI, APIO? Làm thế nào để bạn cảm thấy không bị chán khi giải các đề, khi mà các bài rất khó và yêu cầu cả giờ đồng hồ suy nghĩ?

Chủ yếu vì bài rất hay, cần vận dụng trí tuệ nhiều hơn, và không có quá nhiều bài yêu cầu mình phải code nhiều. Vả lại, nếu gặp các bài code nhiều, em vẫn rất thích vì sở trường của em là code cơ mà. Tuy vậy, em thấy những bài nghĩ nhiều thú vị hơn nhiều ạ. Sau khi thi APIO, dù đã có kết quả nhưng em vẫn chưa tập trung học được. Thấy vậy, anh minhcool (Nguyễn Quang Minh - IOI 2023) khuyên em bắt đầu giải các đề IOI theo từng năm. Em bắt đầu làm theo và từ đó tìm ra nhiều động lực giải đề hơn ạ.

Làm thế nào để bạn biến những đam mê tin học thành sức mạnh của bạn, mà không biến nó thành áp lực? Bạn nhìn nhận việc thi như thế nào?

Những kì thi đối với em không phải là áp lực lớn, có lẽ vì mục tiêu em đặt ra cũng không quan trọng. Lần thi áp lực nhất của em là VOI năm lớp 11, khi em phải vào được Vòng 2. Còn các cuộc thi sau đấy thì em không đặt nặng thành tích, rằng TST và APIO mình đứng rank bao nhiêu, phải có huy chương gì. Nếu em có cảm giác áp lực thì chỉ áp lực trước khi thi thôi, còn một khi đã vào phòng thi thì sẽ tập trung hoàn toàn vào làm bài và không nghĩ gì về áp lực nữa. Nhờ những kinh nghiệm đi thi từ nhỏ, nên không khí phòng thi và tâm lý đi thi của em cũng đã được rèn luyện khá nhiều.

Bạn có dự định đi du học sau khi tốt nghiệp cấp 3 không? Nếu ở Việt Nam, bạn sẽ học UET hay Đại học Bách Khoa?

Có lẽ là không. Đơn giản là vì em không muốn ra nước ngoài. Bây giờ em vẫn đang phân vân, vì UET giờ phải học ở cơ sở Hoà Lạc, mà em chẳng muốn tí nào. Tuy nhiên ở đó cũng có nhiều người em quen hơn nên UET vẫn là sự lựa chọn số một.

Bạn có phải là một người giỏi môn Anh không? Bạn có dự định học IELTS nghiêm túc không?

Em không giỏi tiếng Anh lắm. Khi đi thi IOI, em cũng không giao tiếp được gì nhiều, và điều đó khiến em cảm thấy khá đáng tiếc. Em đang có học bổng IELTS, em cũng đang học nhưng chỉ để giao tiếp được thôi, còn để thi thì em vẫn chưa nghĩ đến nhiều.

Bạn có muốn tiếp tục thi đấu ở ICPC[77] không?

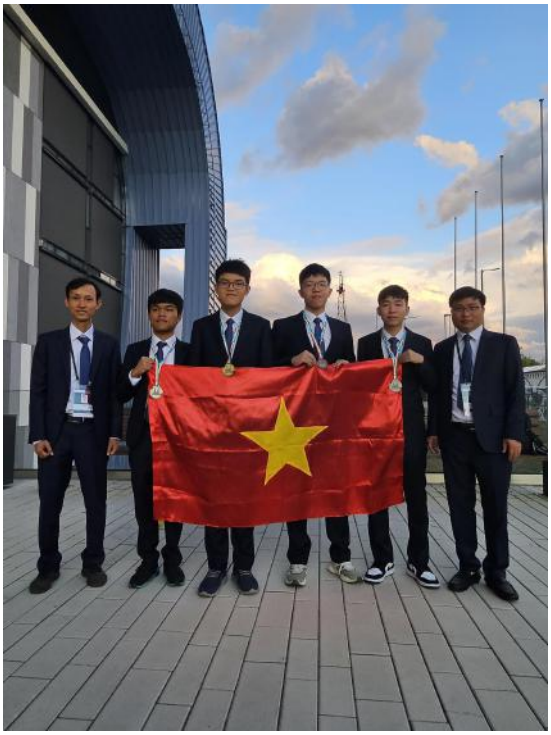
Em nghĩ nếu kiếm được đội thì em sẽ tham gia ICPC. Hơn nữa, nếu học ở UET, em sẽ nhờ thầy Phương (Hồ Đắc Phương - huấn luyện viên ICPC của trường Đại học Công nghệ, ĐHQG-HN) kiếm giúp em. Mong muốn của em là cùng đội với những người giỏi toán và hình, vì đó là hai lĩnh vực em còn chưa giỏi lắm.

Sau này, bạn có dự định sẽ làm gì? Bạn thích đi nghiên cứu nhiều hơn, hay đi làm nhiều hơn?

Em sẽ thiên về đi làm nhiều hơn, vì em không giỏi nghiên cứu nhiều lắm. Tuy nhiên trong tương lai, nếu có một chủ đề mà em đủ đam mê thì em có thể sẽ suy nghĩ lại.

Với tình hình công việc hiện tại, thì bạn có thích một lĩnh vực nhất định nào chưa? (AI, ứng dụng, web, điện thoại, ...)

Em vẫn chưa tìm hiểu nên chưa rõ lắm. Nhưng anh Bách (Trần Xuân Bách - IOI 2023) có rủ em học AI xem thử thế nào, nên em có ý định thử. Còn những lĩnh vực khác, em sẽ thử nghiệm nhiều hơn ở Đại học.



Một phương châm mà bạn luôn hướng đến?

"It's not the destination, It's the journey."

Mỗi lần đi thi chỉ cần cố gắng hết sức mình, còn kết quả đến đâu thì không quan trọng. Quan trọng là quá trình mình ôn luyện, học tập cùng nhau.

Cảm ơn marvinthang đã nhận lời tham gia phỏng vấn cho tạp chí VNOI! Tết đã sắp đến rồi, và sau Tết thì cũng là Vòng 2, nên chúc bạn năm nay đổi màu được huy chương! Chúc bạn được vàng IOI năm nay!

[77] ICPC: International Collegiate Programming Contest - Cuộc thi Lập trình Quốc tế lâu đời và danh giá nhất dành cho sinh viên các trường đại học và cao đẳng trên toàn cầu.

Phỏng vấn thầy Lê Minh Hoàng

Interviewer: Vương Hoàng Long – ICPC World Finalist 2021

Đầu tiên thì rất cảm ơn thầy hôm nay đã nhận lời phỏng vấn cho tạp chí VNOI số Tết! Thầy là người mà tất cả các bạn rất mong muốn được phỏng vấn. Còn gì tuyệt vời hơn khi được phỏng vấn một "huyền thoại", với không chỉ một, mà là hai nghiên cứu đã thay đổi nền Tin học Việt Nam – Sách "Giải thuật và Lập trình" và phần mềm chấm thi Themis mà các bạn bây giờ vẫn hay dùng. Kì thi Học sinh giỏi Quốc gia vẫn còn dùng Themis, và có thể sẽ còn được sử dụng dài dài.

Thầy là một trong những giáo viên dạy Tin thuộc những thế hệ đầu, vậy thì cơ duyên nào đưa thầy đến với môn Tin học?

Câu này nếu trả lời thì sẽ dài lắm! Từ khi mình học cấp 2, mình cũng biết Tin học là gì rồi. Hồi ấy, mình học Tin trong trong tiết Kĩ thuật. Tiết học chán lắm, mình chả có tí cảm hứng nào cả! Lên cấp 3, khi mình gặp người thầy đúng chuyên môn thì khác biệt thật, thầy không những dạy kiến thức, mà còn dạy những kĩ năng nghề nghiệp, và truyền cho mình cái đam mê nghề nghiệp nữa. Lúc đấy thì mình mới cảm thấy thích.

Nói thật thì thời xưa, bọn mình học 100% là đam mê, chứ không nhận được nhiều ưu đãi về lớp chuyên, giải Quốc Gia,... như các bạn bây giờ. Hồi mình thi Quốc Gia, không hề có tiêu chuẩn tuyển thẳng đại học! 100% là vô tư, thích thì học, không thích thì bỏ thôi.

Trong môi trường ấy, cũng may là các thầy cô hay khuyến khích, rằng "nếu em thích thì em cứ học giỏi môn này cái đã, các môn khác tính sau!". Thật vậy, nếu như cứ gây sức ép học đều đều những môn còn lại, môn Tin chỉ học vài tiết một tuần thì chả thể nên cơm cháo gì được.

Mình tới với môn Tin học, đơn giản vì Tin học nó hấp dẫn! Nó không phải như môn Toán, các bài toán khó nhăn nhở trong các kì thi đâu. Mình thích những lúc các anh sinh viên đi làm các đồ án tốt nghiệp, các phần mềm đồ hoạ "chạy nhảy", và những lúc mình làm ra được một phần mềm nào đó, được người khác sử dụng. Có thể niềm vui ấy rất là đơn giản, nhưng mình cảm thấy nó thật sự có giá trị thực tiễn, khác với giải một bài toán, khi mà mình chỉ sướng được một lúc thôi thì đã phải chuyển sang bài khác rồi.

Thầy có đề cập đến việc cảm thấy hứng thú khi viết ra phần mềm và được người khác dùng. Liệu nó có phải là lí do để thầy viết ra phần mềm chấm thi Themis – phần mềm đã thay đổi cách chúng ta dạy học ở Việt Nam hay không?

Trải qua quá trình làm nhiều phần mềm, có cái làm vì mong muốn cá nhân, cũng có những cái làm để bán nữa, mình có kinh nghiệm lập trình khá đáng kể.

Vào năm 2001, cộng đồng Tin học có hệ thống chấm tự động AMM2, nhưng khi chấm C++ phải dịch file C++ ra .EXE mới chấm được. Thế là mình bảo với cả Đông (thầy Đỗ Đức Đông), hay là "viết luôn một trình chấm đi!".

Đầu tiên thì mình cũng định viết cho vui thôi, nhưng mà sau đấy thì thầy Đông bảo: Nếu mà dùng được ổn định thế này, thì xin luôn dự án của Bộ đặt hàng luôn một thể!

Mình nghĩ Themis là một phần mềm khá đơn giản, nhưng yêu cầu lớn nhất của phần mềm đó thì phải chạy không được lỗi! Chạy chậm hay ít chức năng cũng được, nhưng một khi đã chấm thì phải chạy từ đầu tới cuối, không được ngừng giữa chừng hoặc có lỗi. Ngoài ra nó phải gọn nhẹ, dễ cài đặt trên máy xách tay để các thầy mang lên lớp trong điều kiện không có server và internet.

Nhiều bạn thắc mắc thi bằng C++ bây giờ sẽ được chấm bằng bản C++14, không biết thông tin này có chính xác không ạ?

Thật ra giả sử có một thầy ra đề thi đòi hỏi một cách chấm riêng thì thầy có quyền được chấm theo kiểu của thầy, nên sẽ không có quy ước cố định. Khi viết chương trình tương tác với code của thí sinh, nếu lệch phiên bản C++, code hoàn toàn có thể bị dịch lỗi. Tuy vậy với những bài đặc thù như thế thường sẽ có đặc tả rõ ràng trong đề.

Thầy mất khoảng bao lâu kể từ khi Themis ra bản đầu tiên để cho ra phần mềm có khả năng chạy ổn định, được đón nhận rộng rãi?

Phải mất khoảng một năm. Thực ra để thành hình sản phẩm thì mất có vài tuần thôi, nhưng mình với thầy Đông mang đi dạy khoảng một năm, thấy ổn định mới dám cho nghiệm thu và chấm ở VOI năm sau.

Một trong những đặc điểm hay ho của môn Tin học so với môn khác đó là chấm tự động. Chuyện mình đọc code rồi chấm gây ra nhiều sai sót, làm cho việc ra đề thiếu chặt chẽ. Mình đọc rất nhiều đề thi trong các kì thi cấp thấp, thì đề toàn đưa ra những dữ liệu đầu vào (input), dữ liệu đầu ra (output) không có ràng buộc input nào, nhìn có vẻ như là những người chấm có ý tưởng sẽ chạy chương trình bằng tay rồi chấm. Nhưng làm vậy thì trái với quy ước của một đề thi Tin học thuần tuý – có ràng buộc dữ liệu. Khi mà hội đồng chấm thi đòi hỏi phải nộp luôn bộ test khi ra đề, thì chắc chắn đề thi phải có những ràng buộc cơ bản như thế.

Từ khi Themis trở thành trình chấm chính thức cho các cuộc thi (như kì thi HSGQG[78]), thì thầy đã có những kỉ niệm khó quên nào chưa ạ?

May là chưa có! Mình cũng gặp một số thắc mắc, nhưng cuối cùng lần ra thì cũng là do lỗi của thí sinh thôi. Chẳng hạn như, đôi lúc có những hành vi không xác định (undefined behaviour), lúc đó code chạy trên máy thì ổn, nhưng lúc chạy trên trình chấm, rồi thử chạy trên các trang như ideone, Codeforces[79] thì nó cũng bị tình trạng giống như thế.

Thắc mắc nhiều nhất là khi Themis được đưa vào chấm chính thức bằng C++11. Lúc đấy thì nhiều code C++98 bị dịch lỗi. Thực ra thì trước khi tham gia bất kì cuộc thi nào, thí sinh phải biết quy tắc, "luật chơi" của kì thi đấy. Nhưng mà ở VOI, thì quy chế của Bộ chỉ nói là C++ thôi, không đề cập phiên bản nào, thế hệ bao nhiêu! Cho nên đôi khi, cần có một thứ phổ biến để giúp thí sinh hiểu rằng "nếu anh chấm ở nhà bằng phần mềm ấy không lỗi, thì tức là đi thi có thể dùng được các lệnh đó!". Và Themis đã làm được điều ấy. Mình nghĩ Themis đã góp phần hạn chế bớt lỗi trong quá trình thi thật, khi Themis đã được phát hành rộng rãi cho tất cả mọi người đều có thể tiếp cận.

Chuyện về những ngày đầu ra đấu trường quốc tế

Tin học là môn học hay, được dự đoán là một phần rất quan trọng trong nền giáo dục tương lai, do đó, năm 1987, Hội nghị UNESCO về giáo dục nêu đề xuất tổ chức kỳ thi tin học quốc tế (IOI[80]) bên cạnh các môn Toán, Lý, Hóa, Sinh. Bulgaria đứng ra đăng cai IOI đầu tiên vào năm 1989. Hồi đấy, Tin

[78] HSGQG: là viết tắt của kỳ thi Học sinh giỏi Quốc gia.

[79] Codeforces: một trang web tổ chức các cuộc thi lập trình thi đấu với các dạng đề, bài tập đa dạng.

[80] IOI: Kỳ thi Olympic Tin học Quốc tế.

học đã được tiếp nhận tại Việt Nam, nhưng chỉ dạy cho một số sinh viên ở các khoa đặc biệt của trường đại học mà thôi. Ông Trần Hồng Quân (Bộ trưởng Bộ Giáo Dục từ năm 1987 đến năm 1997), gọi thầy Đàm (Nhà giáo nhân dân Hồ Sĩ Đàm - đồng tác giả bộ sách giáo khoa Chuyên Tin) lên và bảo: "Tôi cũng chả biết môn này là môn gì, nhưng mà thấy bảo môn này quan trọng lắm! Về nhóm cái đội đi thi Quốc tế đi!". Thi thì, có thể giải ở những năm đầu không quan trọng đâu, chủ yếu để biết người ta học thế nào để mình học theo.

Thầy Đàm về kêu mãi 4 đứa theo chỉ tiêu tối đa đi thi quốc tế, nhưng gọi được có 3 người thôi. Ba người ấy học cấp tốc trong khoảng độ từ 3-4 tháng, về thì có huy chương Đồng. Xong rồi ngay từ những năm sau thì Việt Nam mình đã có Vàng, có Bạc. Đỉnh cao đến năm 1999 Việt Nam đứng #1 toàn đoàn. Bây giờ Việt Nam còn thiếu đúng chức vô địch IOI. Sau IOI lần đầu, ông Quân bảo là: "Chưa học gì mà đã được huy chương Đồng, vậy thì triển khai học là ngon ngay". Do đó, ông yêu cầu thầy Đàm triển khai chương trình dạy tin học phổ thông.

Các thầy của Việt Nam có thể là những chuyên gia rất giỏi, nhưng viết chương trình lại là thách thức lớn, khi phải biết dạy cái gì trước, cái gì sau. Hơn nữa, các thầy không thể áp cái cách học của mình xuống học sinh phổ thông vì hầu hết các thầy học tin đều trong trạng thái đã là Tiến sĩ, Phó tiến sĩ, Thạc sĩ, Cử nhân các ngành Toán, Lý, Tối ưu hóa, Tự động hóa rồi. Hồi đấy thì chưa bỏ cấm vận, mình chỉ giao lưu với các nước thuộc khối Vác-sa-va (Warsaw) - các nước Xã hội Chủ nghĩa thôi. Nên các thầy phải đi xin tài liệu của Ba Lan, Nga, v.v, rồi sau đó thì trộn vào thành một cái chương trình, dạy ở một vài trường chuyên: trường chuyên Khoa học Tự nhiên, trường mình (Đại học Sư phạm), Đại học Bách khoa, trường chuyên Hà Nội – Amsterdam, thành phố Hồ chí Minh với trường Phổ thông Năng Khiếu, v.v.

Học sinh của mình thì có cái hay: "nếu không thi, thì không học; còn một khi đã thi thì nó sẽ học và ông thầy cũng sẽ phải cố dạy!". Từ đó, nội dung học ngày càng được làm phong phú thêm và chuẩn hóa dần. Do vậy, chỉ có huy chương Đồng trong lần đầu tiên tham dự thì mang lại sự bất ngờ thôi, còn những năm sau, tính cạnh tranh được nâng cao, đề thi bắt đầu khó rồi.

Những năm đầu tiên, IOI vẫn có format chấm như các môn khác: học sinh trình bày thuật toán ra giấy; Chương trình có là được, không có thì cũng chả đến nỗi bị 0 điểm. Sau đó, mình phải dịch bài ra tiếng Anh, rồi từng nhóm chia nhau ra chấm. Trong ngày cuối cùng, trưởng đoàn phải đứng ra bảo vệ cho học sinh mình - tranh luận ở những điểm nào đang mấp mé giữa cho điểm hay không. Nói chung, việc chấm bài rất mệt mỏi, và hội đồng làm việc rất căng thẳng trong một khoảng thời gian dài.

Đến năm 1994, người ta cãi nhau "mệt rồi". Thôi thì, anh làm sai cũng được, nhưng tôi có một bộ test, làm sai là một phần của cuộc chơi, anh cứ đúng test đó là được. Từ năm ấy, người ta bắt đầu chấm theo bộ test. Từ khi bắt đầu chấm bộ test, Việt Nam mình kết quả cũng có chiều hướng đi xuống; Có năm chỉ được 1 tới 2 huy chương Đồng. Nhưng mà ngay sau đấy, từ năm 1995 tới 1999, Việt Nam mình đứng nhất thế giới! Dù chấm bằng bộ test nhưng tính chất bài thì vẫn giữ nguyên, là các bài toán tối ưu. Các bài đó được chấm theo độ tốt, thí sinh không cần đưa ra đáp án chính xác tuyệt đối. Hồi đấy, đa số các bài đưa ra thì sẽ

theo kiểu NP đầy đủ (NP-complete). Sau đỉnh cao 1999, không chỉ Việt Nam, rất nhiều nước khác bắt đầu dùng chiến thuật: không tập trung nghĩ thuật toán hoàn hảo cho một bài nào cả (subtask), bài nào cũng phải có code một phần nào đó. Đôi khi code ấy lại được điểm rất là cao, nhưng nói thẳng là thuật toán sai ấy! Bởi vì phải đưa ra đáp án gần đúng chứ không phải đáp án chỉ có một số, đúng chính xác như bây giờ.

Đến năm 2003 hay 2004, người ta mới bắt đầu chấm tuyệt đối – lệch một đơn vị cũng chết. Đến năm 2010, thì việc chấm đúng test nào được điểm test nấy cũng không công bằng – yếu tố may mắn còn nhiều quá. Vậy là IOI bắt đầu áp dụng format chấm theo bộ test cho từng subtask (batch) – phải đúng hết tất cả các test trong subtask đó thì mới ăn điểm.

Giáo trình phổ thông môn Tin học những ngày đầu tiên

Ở các môn khác, người ta nhặt ra rồi nâng cao một vài kiến thức từ chương trình phổ thông đại chúng; Các kiến thức ấy trở thành chương trình chuyên. Còn với môn Tin thì ngược lại, chương trình chuyên có trước, sau đó được giảm tải đi, trở thành chương trình đại trà. Cho nên người ta hay phàn nàn về việc phần lập trình của chương trình phổ thông cũ khá nặng. Thật ra thì phần lập trình đâu có giảm tải. Nó chỉ có hai dạng: biết lập trình hoặc không biết lập trình thôi! Câu lệnh thì vẫn phải học đủ chứ giảm tải được lệnh gì!

Không dạy for với while thì lại không được!

Ngày xưa, thì trong sách lớp 11 dạy thế đấy! Trong Pascal có 3 vòng lặp: repeat, while với for. Tương tự cho C, có do..while, while với for. Hàm while là hàm tổng quát nhất, thì người ta dạy mỗi while còn hai cái kia thì lại bỏ! Tước bớt công cụ thì lập trình lại càng khó hơn chứ đâu có dễ hơn gì đâu! Đôi khi bệnh thành tích hay học để thi, thì cũng có mặt tích cực: Khi mình đã đại diện cho một đơn vị nào đấy đi thi, thì mình phải có một sự nỗ lực cố gắng, và trong quá trình nỗ lực cố gắng ấy, thì mình mới thấy thích! Không có yêu cầu thành tích thì người ta chả học đâu, hoặc học mà không có định hướng, không có mục đích rõ ràng. Ông thầy cũng lười, vì môn của mình đâu có thi đâu mà phải ép các bạn học sinh nhiều vậy! Trong thể thao, một người không tập luyện bài bản gì vẫn có thể chạy, đánh bóng bàn, cầu lông, đá bóng, ... nhưng vẫn phải có các động viên thành tích cao, thì người ta mới biết chơi thế nào là đúng kỹ thuật, tránh chấn thương, rồi ăn uống, ngủ nghỉ như thế nào để nâng cao thành tích... Trong Tin học cũng tương tự thế, chỉ những thí sinh CP[81] mới biết kỷ luật nghiêm ngặt khi chấm bằng test, biết cách hạn chế các lỗi tràn hay hành vi không xác định...

Còn bệnh thành tích nó cũng có mặt tiêu cực, ngoài những thứ mà báo đài hay đề cập thì trong môn Tin học, có xu hướng không học/không dạy những kiến thức không xuất hiện hoặc ít xuất hiện trong các bài thi cho dù đó là những giá trị cốt lõi. Chẳng hạn nhiều bạn "ngại" học con trỏ, "từ chối" học lập trình hướng đối tượng, sẵn sàng code bẩn, code bắt test để che giấu lỗi...

Qua đấy mới thấy việc ra đề và làm test trách nhiệm cao đến thế nào: Không chỉ là để phân loại trao giải mà còn để định hướng việc học cho các thế hệ sau nữa. Làm thiếu trách nhiệm sẽ đẩy việc học xa rời các giá trị cốt lõi mà tập trung vào các kỹ năng vô bổ.

[81] CP: Competitive Programming - Lập trình thi đấu

Quan sát tin học Việt Nam một thời gian, trình độ các bạn từ khi Codeforces được phổ cập thì bắt đầu tăng rất nhiều. Các kĩ thuật vào những năm ngày xưa được liệt vào dạng cao cấp như Segment Tree, Chia căn, ... thì bây giờ được xem là thuật phổ thông!

Ngày xưa mấy thuật này phải tự chế ra! Thường thì các thuật ấy không có trong sách vở chính thống hoặc có nhưng khó tìm sách mà đọc lắm. Đa số các trường hợp là nhờ một người nào đấy làm bài rồi tự chế ra, sau đó truyền lại thế thôi chứ nó không phổ biến rộng.

Thầy thấy rằng trong 5 năm trở lại đây thì Tin học Việt Nam đang ở mức nào? Trình độ chung các bạn như thế nào?

Mình nghĩ rằng gần đây các bạn cũng có mặt giỏi hơn, nhưng cũng có mặt dở hơn. Tất nhiên thì ở đỉnh cao, sóng sau sẽ luôn lớn hơn sóng trước. Nhưng so với ngày xưa, thì bây giờ khi các bạn học cơ sở, các bạn phụ thuộc vào Internet và ChatGPT nhiều quá! Học những cái ABC mà đã bị phụ thuộc vào những thứ đấy, thì các bạn sẽ không học được những thứ DEF rồi XYZ. Những viên gạch, những bước chân đầu tiên, thì mình phải tự bước đi! Phải tự bước đi, để tự vấp ngã, tự đứng dậy. Những bước đi có người dắt chỉ là học động tác, bạn phải tự đi thì mới biết cách lấy thăng bằng, và cái chi tiết đó tuy là nhỏ nhưng không ai dạy được cả.

Mình thấy rất nhiều bạn như thế, kể cả là sinh viên. Tất nhiên thì những bài ở mức độ nhập môn ở đâu cũng có, ai cũng dạy như thế cả nên chắc chắn là tìm trên mạng sẽ có được code, và học thuộc rồi đi thi thì sẽ làm được thôi! Nhưng cái quan trọng là khi mình làm những bài đấy, thì mình phải tự làm, phải trải nghiệm những sai lầm, để biết cách sửa lỗi sai ấy và trưởng thành hơn. Nhiều bạn hiện tại hoàn toàn không có khả năng tự sửa chữa lỗi sai ấy, vì code đúng hết rồi thì lấy đâu mà sửa nữa! Thậm chí là chuyện chép code của nhau cũng thế, thay vì tự mình tư duy như hồi xưa, việc đầu tiên các bạn làm là search và chép code. Điều này làm cho tư duy của các bạn bị kém độ nhạy đi. Kiến thức học sinh bây giờ có thể được học nhiều hơn trước, nhưng độ nhạy bén trong suy nghĩ để vận dụng các kiến thức ấy không bằng các thế hệ xưa.

Ngay cả trong việc học C++, các bạn tập sử dụng các hàm trong thư viện sớm quá! Nếu ngay từ đầu mà đã nhảy vào dùng set, map, v.v, thì người ta sẽ không hiểu cái nguyên lý của các cấu trúc dữ liệu đó. Mà đôi khi, đích đến của việc học không phải là để cài bài tập, học là để hiểu nguyên lý! Đấy mới là cái bổ ích, là giá trị cốt lõi của khoa học máy tính. Thật ra để làm những thứ mà chỉ đơn thuần dùng set, map thì chả có phần mềm nào đòi hỏi bạn tự code bằng tay đâu!

Tỉ lệ thuận với sự phát triển của các phương tiện truyền thông, việc giao lưu hỏi bài trên các diễn đàn trở nên thuận tiện hơn. Nhưng nó cũng là cái thứ dở đấy! Ngày xưa, có những bài phải nghĩ vài tháng mới ra. Thậm chí, có những bài mà cho đến bây giờ vẫn nghĩ chưa ra. Nó ở cái mức IMPOSSIBLE ấy, bản thân người ra đề còn chưa tin là mình có được giải pháp tốt đâu. Thế nhưng mà qua quá trình xoay sở ấy, thì kĩ thuật và tư duy của mình sẽ lên khá đáng kể.

Nói về chuyện chấm tự động, cho học sinh chấm tự động ngay từ đầu, mình nghĩ không phải là điều hay. Không cần nộp bài chấm tự động, đọc code, hack được code của thằng bạn hay bị hacked rồi phân tích xem vì sao test này lại hỏng... Cách học như thế sẽ mang lại năng lực học tập tích cực hơn nhiều, giúp việc học đỡ khô khan hơn. Mình nghĩ môn này học rất hay, nhưng mà do cách học nên một số bạn bị thụ động, dẫn đến nhanh chán. Tóm lại nếu như ngay từ đầu mà các bạn không chịu khó trải nghiệm khó khăn và vượt qua chúng, thì không thể đi xa được.

ChatGPT thì gần như nó sẽ giải được hết các bài đơn giản hoặc cơ bản. Vấn đề là nếu chúng ta không nghĩ bài đấy thì sẽ không thể giải được các bài khó, sẽ chỉ mãi dậm chân tại chỗ.

Nó không phải là chỉ có các bạn học chương trình Tin học phổ thông đâu. Ngay cả trong lớp chuyên Tin, nếu như ngay từ đầu mà các bạn không chịu khó trải nghiệm khó khăn và vượt qua chúng, thì cũng chả đi được đến đâu. Sử dụng ChatGPT cho các bài đơn giản, nó sẽ đọc cả đề và solution chuẩn 100%! Nhưng nếu chúng ta không nghĩ bài đấy thì sẽ không thể giải được các bài D, E, F trong một cuộc thi trên Codeforces được, ta sẽ chỉ mãi giậm châm tại chỗ. ChatGPT nó làm mọi người lười suy nghĩ hơn.

Vấn đề của cách học này còn dính dáng tới chuyện về học toán nữa cơ. Cách học toán của các bạn hiện tại vẫn theo kiểu gần như tương tự ngay cái lúc mà bạn học toán ở cấp cơ sở, thì các bạn đã phụ thuộc vào sách bài tập, lớp học thêm, các hệ thống AI, v.v, làm tư duy toán của các bạn khá là nông. Thậm chí là, nếu mình đọc thử đề thi toán các trường chuyên, hầu như là toàn các bài rất khó, thì các bạn giải tốt, nhưng đến những cái toán vô cùng cơ bản, thì chưa chắc các bạn đã nhớ và vận dụng tốt. Phải nói rằng người thầy dạy phải khéo lắm thì mới cảm thấy các bạn đang làm bài tốt, nhưng rất có thể đó là giá trị ảo dẫn tới đánh giá sai tưởng các bạn giỏi vì làm được bài khó nên liên tục nâng cao mà không biết rằng các bạn đang mất cơ bản.

Như đã nói, ChatGPT nó làm mọi người lười suy nghĩ hơn, và điều này rất nguy hiểm. Mình dạy những bài toán cơ bản: tìm kiếm nhị phân, sàng số nguyên tố,... Mình không phải chỉ dạy đơn thuần thuật toán đấy – nó chỉ là một phần rất nhỏ thôi, vì cái tư tưởng thuật toán và cách thức triển khai chương trình mới quan trọng, và qua việc mô phỏng đúng một thuật toán thì người học mới thấy được cái hay và phát triển tư duy giải quyết vấn đề. Ví dụ với sàng số nguyên tố, ngày xưa, ông tác giả dùng cái que, chọc một vài lỗ lên lá để sàng. Ta phải mô phỏng lại quá trình đấy, để các bạn thấy được cái hay của thuật toán, và tư duy của các bạn sẽ phát triển khác đi.

Những bài toán "lớp 1, 2" thì chắc bây giờ ChatGPT làm được hết rồi! Nhưng nếu anh không chịu học lớp 1, 2 thì chẳng thể lên được 3, 4; càng chẳng thể học tiếp lên cấp 3, Đại học, Thạc sĩ, Tiến sĩ rồi đi làm được. Những thứ cơ bản ấy, sớm muộn gì cũng phải học, mọi người cũng sẽ thuộc bảng cửu chương, và biết cộng trừ nhân chia. Có thể lớp 1 học không giỏi, nhưng mình nghĩ rằng một khi đã đến lớp 12 thì chắc ai cũng phải biết đọc, viết, cộng trừ hết rồi. Tóm tắt lại, phải biết cộng biết trừ rồi mới học cái khác. Mình nghĩ là học sinh phổ thông, khi bắt đầu học một môn mới, bạn có thể nhận thức hơi chậm và kỹ năng giải bài kém - nó chẳng nói lên điều gì. Chỉ cần bạn kiên trì, khắc phục được những điểm yếu của mình, thì bạn còn có thể tiến xa hơn những bạn "phát triển sớm" hoặc được học trước, nếu như bạn không bỏ qua những năng lực cần có của từng bài học.

Codeforces theo thầy đã thay đổi nền tin học Việt Nam như thế nào, khi trong 5 tới 7 năm gần đây là nền tảng lớn nhất, phổ biến nhất cho CP-er ở Việt Nam?

Các bài của Codeforces, đặc biệt là Div 2 và Div 3, nó tập cho các bạn đang nhập môn lập trình khả năng code nhanh và code chính xác. Ta cần một giải pháp đơn giản cho các bài đơn giản. Tất nhiên để làm được bài đấy thì dễ thôi, nhưng cần phải làm vừa nhanh, vừa chính xác.

Có rất nhiều bài toán khó trên Codeforces là lắp ghép của những bài cơ bản. Với từng đoạn cơ bản, nếu bạn code vừa ngắn vừa đúng đắn, thì chương trình lớn sẽ dễ kiểm soát hơn. Ngược lại, ngay đoạn ngắn mà đã bug rồi, thì sẽ không thể viết được những chương trình phức tạp hơn. Cái đấy thầy nghĩ là cái lớn nhất mà Codeforces đóng góp được.

Còn Div 1, thì mình nghĩ nó chỉ dành cho những ông đi thi thôi, còn để học thì mình nghĩ rằng cứ cài Div 3, Div 2 lên khoảng độ Expert[82] , Candidate Master[83] gì đấy. Tuy nhiên thì Codeforces không dành cho những kiểu người già như mình, chân tay chậm rồi, không làm nhanh được. Mình làm bài chỉ để có vốn bài, xem xét xem có kĩ thuật gì hay để dạy nên có kỳ thi mình làm mỗi một bài, mà có khi còn sai.

Mình chỉ nói thêm chút là nhiều bạn quan trọng hóa hạng của các kỳ thi quá, kể cả Codeforces. Các kỳ thi vô thưởng vô phạt các bạn nên đặt việc học tập và rèn luyện chiến thuật thi lên trên hết. Không cần phải code quá vội hay ăn bớt quy trình kiểm thử. Hoặc chẳng hạn bạn dậm chân tại chỗ ở một mức lâu quá, kỳ thi nào cũng chỉ làm ABC, thế thì bạn thử một kỳ thi nào đó làm A, D hoặc chỉ D thôi đi, dành thời gian cho một bài nhiều hơn để dù không làm được, lúc đọc lời giải mình cũng ở trạng thái chủ động tiếp thu kiến thức. Thế mới đúng là thi để học và rút kinh nghiệm, nếu thi triền miên mà ở từng cuộc thi mình không thu được gì ngoài số điểm và ranking thì cuộc thi đó vô nghĩa.

Có một sự thật khá nổi tiếng trong cộng đồng là thầy có tài khoản Codeforces. Là một trong rất ít giáo viên tham gia thi Codeforces, thầy có suy nghĩ như thế nào?

Khi mình làm việc với các bạn mới học đội tuyển thì mình cũng thi Codeforces. Mình có trao giải thưởng cho ông nào thi hạng cao hơn mình. Nói thế chứ thi mà hạng cao hơn thầy thì cũng không phải khó lắm! Chủ yếu vui là chính thôi!

Có một đoạn nữa, khi mà chuyển sang dạy C++, mình biết C++, nhưng để biết ở mức độ đi dạy được, thì phải thi bằng cái đấy xong mới biết mà dạy được. Nếu mà chỉ chuyển từ ngôn ngữ này sang ngôn ngữ kia theo kiểu ánh xạ 1:1 ấy, thì nó không phải là tư duy của một người đi thi – không thể làm nhanh được. Đợt đấy, rank của mình tụt thê thảm, cứ nhập nhằng mãi. Quãng thời gian đấy, hầu như kỳ thi nào mình cũng làm, đến mức độ học sinh tưởng mình cho nick đứa nào đó để tập cơ. Mình tập bằng một ngôn ngữ mới, thì nó thay đổi thói quen viết cũ, ngay cả trong cách trình bày chương trình. Phải tập đến độ Expert thì mới tự tin đi dạy được người khác.

Theo thầy thì so C++ với Python, Python có dễ học hơn C++ nhiều không?

Nói về ngôn ngữ lập trình, mình thấy ngôn ngữ nào cũng phải bỏ công học tất cả mọi cấu trúc điều khiển và đặc tả của ngôn ngữ đó. Mình không nghĩ là có ngôn ngữ nào dễ hơn hẳn cái kia. Còn nếu mà nói khó làm chủ, thì mình nghĩ rằng C++ là một trong những ngôn ngữ khó. C++ có quá nhiều kí hiệu, và càng về sau thì ngày càng mở rộng ngôn ngữ ra làm cho code rối rắm hơn, nổi bật là lambda function và các cách viết tắt thông dụng khác nữa. Lẽ ra thì C++ nên làm theo hướng thêm đặc tả nếu cần thôi, chứ ngôn ngữ đang chạy ổn thì cứ giữ nguyên một khoảng thời gian. C++ cứ 5 năm lại lên một đời, làm cho người ta chạy theo lâu lâu cũng thấy mệt.

Còn riêng về Python, thì mình không nghĩ đây là một ngôn ngữ dễ. Nó dễ theo cái nghĩa: nó viết những chương trình đầu tiên rất dễ. Ví dụ, nhập vào một số rồi in ra cái gì đó, nhập hai số in ra tổng, ... rất dễ, chỉ cần gõ lệnh trực tiếp rồi chạy từng lệnh một, thay vì gõ cả chương trình rồi biên dịch, sửa lỗi cú pháp, v.v. Và người ta có cảm giác rằng nó dễ vì sống chết gì nó cũng chạy, chỉ là tới dòng mà mình viết sai thì nó dừng lại thế thôi. Vậy nên mình cũng chẳng hiểu căn cứ vào đâu để người ta nói Python dễ học. Có lẽ người ta cho rằng nó dễ học vì chép code trên mạng có nhiều.

[82] Expert: mức điểm hơn 1600 trên Codeforces.

[83] Candidate Master: mức điểm hơn 1900 trên Codeforces.

Nói về chuyện thuật toán và cấu trúc dữ liệu, một trong những cái mà mình thấy rất là quan trọng là đánh giá độ phức tạp tính toán. Python không đánh giá được vì phép cộng, phép nhân của nó không phải là hằng số. Với những cái đấy, thì khi mình đánh giá độ phức tạp, lúc chạy thì nó sẽ 'rùa' hơn so với cái mình đã đánh giá.

Khi mình đi dạy, mình chỉ ra rằng một thuật toán $O(n^2)$ sẽ chậm hơn $O(n \log_2 n)$, nhưng $O(n \log_2 n)$ thì phải dùng phép nhân ma trận, cần tính toán số lớn chẳng hạn. Khi đem hai code đi chạy thì code $O(n \log_2 n)$ lại chậm hơn code $O(n^2)$ đáng kể. Tự nhiên làm thế, người ta cũng mất lòng tin với lý thuyết được học. Nói đi cũng phải nói lại, ngôn ngữ biên dịch sẽ khá sát với độ phức tạp tính toán.

Nhưng nói chung, mình không nghĩ rằng học một ngôn ngữ lập trình là chuyện dễ. Vì ở ngoài kia, người ta có những lớp học cấp tốc, có những cuốn sách học ngôn ngữ nào đó trong vài tuần. Mình không nghĩ là nó sẽ được việc đâu! Nó phải có trải nghiệm, có sự luyện tập, phải làm từ từ thì nó mới có độ ngấm! Học có độ 3 tuần thì chỉ ở mức độ biết nó là gì thôi.

Là một người code C++ thì thầy là người code theo phong cách những phiên bản mới nhất, hay theo kiểu truyền thống là C++11?

Mình code theo kiểu truyền thống. Vì thật sự, ngay cả những đặc tả trong C++17, C++20 thì mình cũng chưa biết hết đâu. Mình code theo kiểu: trông nó quen thuộc thì viết thôi. Thỉnh thoảng nổi hứng lên viết những lệnh trông ngầu ngầu một tí, chứ nó không phải là chuyện thường xuyên. Một khi mà mình viết một lệnh mà bản thân mình thấy đã khó nhìn, thì khi debug (sửa lỗi) nó sẽ mệt lắm.

Còn về IDE, thì bây giờ thầy vẫn dùng gì? Thầy thấy các bạn trẻ bây giờ hay dùng VS Code, Sublime Text thì thầy cảm thấy thế nào?

Với C++ thì thầy vẫn dùng Codeblocks. Mình nghĩ là Codeblocks nó dở thật, hệ soạn thảo tích hợp nó hỗ trợ kém quá! Nếu có điều kiện thì nên dùng những phần mềm chuyên nghiệp, kết hợp debugger[84] , unit test[85] , version manager[86] , v.v.

Tuy nhiên mình lại thấy có rất nhiều bạn code thẳng bằng giao diện trên trang web nào đấy: Mở web, code thẳng vào đấy rồi nộp bài luôn. Mà mình lại chẳng thấy cách làm này được lợi ích gì cả. Nếu có lỗi thì nó gỡ kiểu gì?

Người ta đã nghĩ ra IDE – một môi trường tích hợp cả soạn thảo, biên dịch và gỡ lỗi, thì nó phải có lí do của nó. Nếu bạn code trên ideone thì bạn đã bỏ qua những lợi ích mà nó cung cấp. Ngay cả chuyện đơn giản như việc nhập xuất từ file, tuy đề không cần nhập xuất từ file, các bạn lại có thói quen cứ chạy thử chương trình rồi lọc cọc gõ theo input mẫu. Mà input mẫu thì có phải lúc nào cũng ngắn đâu, cũng có những bài rất dài! Rồi khi chạy sai, rồi lại gõ input mẫu vào. Mình nghĩ rằng công đấy tốn sức hơn cả việc soạn một sẵn một file input. Nhiều khi mỗi lần chỉ gõ hết 5-10s thôi, nhưng mà phải gõ cả chục, cả trăm lần đó là thời gian đáng kể và gây ức chế.

Cách làm đấy chỉ nhanh hơn ở những bài đơn giản có thể code ăn luôn. Chứ còn nó không hề nhanh hơn ở những bài sau này, khi mà chắc chắn ai cũng phải xoay sở một tí, phải debug rồi mới chạy được. Nhưng mình thấy rất nhiều bạn chẳng bao giờ làm file input. Như mình, khi thi Codeforces, nó bảo làm standard input/output (nhập xuất chuẩn qua màn hình console) nhưng chạy trên máy mình, mình vẫn làm sẵn file input ra đấy. Mình chẳng bao giờ chơi bài cứ chạy thử, rồi copy input mẫu đó vào. Mặc dù copy nó sẽ nhanh hơn gõ, nhưng nếu sang đề in giấy, đề Quốc gia là thôi chết rồi. Nhiều lúc ngồi gõ còn gõ sai nữa!

[84] debugger: trình sửa lỗi.

[85] unit test: kiểm thử cục bộ.

[86] version manager: trình quản lý phiên bản.

Mình chẳng hiểu vì sao nữa, bạn chỉ cần bỏ vài giây, có khi chưa tới một phút, trong việc làm sẵn một file input như thế nhưng các bạn vẫn không làm.

Chuyện chọn IDE cũng như vậy, quả thực rất là khó để dạy các bạn debug, bởi vì những bài đầu tiên thì nó không đáng để dạy những cái đấy. Khi mà bài khó lên, sẽ cần dùng debug nhưng lúc đó thì các bạn không có thói quen đặt break, while, step over, trace into, v.v, gì cả. Khi ấy tất cả mọi kỹ năng debug phụ thuộc vào console để in ra vài giá trị trung gian, thậm chí có những bạn đi thi VOI rồi mà vẫn debug theo kiểu chống mắt lên soi từng dòng code mà không hề chạy thử để khoanh vùng và cô lập lỗi. Dùng debugger kết hợp với in số liệu trong tiến trình cộng thêm assert, exception, v.v, mới là cách đúng và nếu các bạn theo nghề lập trình thì sớm muộn gì cũng phải học.

Nhắc đến việc debug, em thấy rằng ở Việt Nam rất ít người dùng debugger, tất cả đều chọn phương án in hết. Thầy có dạy học sinh dùng debugger không hay cũng in ra hết màn hình để tìm lỗi?

Thật ra thì cái đấy có dạy được đâu. Mình nghĩ hồi xưa, khi mình học debugger thì trong tư tưởng của mình: nếu người làm ra phần mềm đã nghĩ ra cái này thì đây không phải là cái thứ vứt đi, nó đã tích hợp rồi thì chắc chắn phải là cái quan trọng!

Cái thứ hai, là mình nhận ra hạn chế của trò soi code hay in màn hình nên mong muốn có một thứ giúp cho mình khoanh vùng lỗi dễ hơn. Mình đợi tới khi nào mình sinh bug, thì mình dùng thử nó xem sao! Lúc bấy giờ mới tra lại phím tắt các thứ, dần dần thì mình quen. Trò này kết hợp với việc in ra màn hình thì nó rất là lợi! Ví dụ, trong một vòng for từ 1 tới 1000000, i chạy đến 2, 3, 4 thì mới lỗi. Mình chạy đến i $=$ 4 thì nó sinh lỗi, mình break, rồi đặt điều kiện ở i $=$ 4, thế là gỡ thêm được! Chứ cứ để nó in ra màn hình thì nó sẽ rất là rối. Nếu in ra 1, 2 số thì được; hoặc trong đa số các bài thi Quốc gia thì vẫn có thể làm được, nhưng phải quen, thao tác nhanh.

Cũng phải thừa nhận có những người năng khiếu lập trình bẩm sinh. Họ code rất bẩn, rối rắm nhưng vẫn kiểm soát được và debug bằng mắt rất nhanh. Những thứ thực sự làm khó họ phải ở tầm các phần mềm hàng vạn dòng code trở lên và làm trong thời gian dài chứ không phải giới hạn vài trăm phút. Các kỳ thi không làm khó được những người này và khi họ livestream cũng phổ biến luôn cái thói quen debug bằng mắt ấy. Mình thì cho rằng nếu có năng khiếu bẩm sinh như vậy thì không cần học theo ai, còn nếu không thì cứ học theo đúng quy chuẩn phần mềm mà làm. Học làm giàu theo kiểu tỉ phú là cách phá sản nhanh nhất mà!

Những năm gần đây, thành tích của Chuyên Sư Phạm bắt đầu đi lên. Thầy có cảm nhận gì về phẩm chất chung giữa các bạn đạt thành tích cao hay không?

Mỗi người một vẻ, không giống nhau tí nào. Nhưng có thể phân loại những bạn học toán tốt từ bé. Các bạn có thể hơi loạng choạng về mặt kĩ thuật lập trình, nhưng khi mà bạn đã khắc phục được, thì tư duy quản lí chương trình và triển khai thuật toán sẽ rất sáng sủa và tiến bộ nhanh. Loại thứ hai là những bạn ngược lại – kĩ thuật lập trình tương đối tốt và chịu khó tư duy sâu. Thứ 3 là loại mà giỏi bẩm sinh ở tầm... khủng, trước giờ không được học nhiều nhưng khi đam mê rồi thì tự học 100%, vai trò của thầy rất là ít.

Mình nghĩ là ở mức độ thi Quốc gia, cứ chăm chỉ luyện tập thì chắc chắn sẽ được giải. Đôi khi, chỉ cần chăm chỉ luyện tập, kiên trì, tiến bộ, tự nhận dạng lỗi sai của mình, tránh lặp đi lặp lại một lỗi nhiều lần thôi là đủ, không cần tư chất gì đặc biệt.

Còn ví dụ nếu muốn vào vòng 2, thì phải có năng khiếu, có tư chất đặc biệt hơn đúng không ạ?

Vòng 2 thì, mình nghĩ các bạn vào đội tuyển Quốc gia đi thi châu Á hay Quốc tế, thì phải có một số năng khiếu đặc biệt, ít nhất là trong việc thi cử. Cũng có những bạn học và thi xong rồi, có thể sở thích

của bạn dẫn bạn đi làm dự án, chứ không dừng lại ở việc thi học sinh giỏi, làm những bài khó nhằn như vậy. Nói thật chứ, kể cả Quốc tế cũng thế, các bài ở tầm đỉnh cao, đôi khi nó lại xa rời thực tế. Nó không đề cao giá trị cốt lõi của môn học nữa, mà chỉ là những mẹo giải bài. Nhiều lúc thì có thể học đến đấy, các bạn không thích nữa, thì các bạn sẽ không vào Vòng 2.

Còn để đạt giải Quốc gia, thì kĩ thuật là cái quan trọng nhất, không cần phải đầu óc đâu. Nghĩ ra cái gì, làm được cái đấy là sẽ có giải. Đa số các bạn đi thi Quốc gia không may mắn mà mất giải, thì thường sẽ có lỗi to đùng như sai tên file, ăn quả 0 điểm, v.v. Đó là rủi ro chứ không phải giỏi hay kém gì ở đây. Thứ hai, mình thấy đề thi Quốc tế, nó đã xa rời thực tế thì chớ, lại còn dài dòng, khó hiểu. Nó khác hẳn Codeforces, đề nó vẫn ở độ dài vừa phải, không dài quá. Còn đề Quốc gia thì ngay cái việc mình phân tích xem người ra đề yêu cầu mình cái gì, nó cũng đã lâu rồi.

Các bạn từ đoạn chưa biết gì, để dạy lên trình độ ổn thi Quốc gia, thì mình phải dạy đủ kiến thức thì sẽ thi được. Còn với thầy, làm thế nào để thầy phát huy được các bạn đã ngang tầm Quốc gia và muốn học lên, khi kiến thức không còn là vấn đề quá lớn nữa?

Đến lúc thi Quốc gia, những tuần cuối cùng, chủ yếu mình cho các bạn rút kinh nghiệm chiến thuật. Tất nhiên thì kiến thức thì muốn học bao nhiêu cũng được, nhưng việc chuẩn bị trước cách phản ứng trong phòng thi với từng trường hợp cụ thể thì sẽ tốt hơn. Ví dụ như lúc mình đọc đề và chả thấy bài nào mình làm được, thì chắc hẳn người khác cũng sẽ trong tình trạng như thế thôi! Mình muốn điểm số bằng bạn bằng bè thì phải đưa ra chiến thuật như thế nào, chứ nếu ngồi ngắm đề thì thôi chả làm gì được. Hoặc là đi thi, mình đọc bài nào cũng có phần làm được, thì mình sẽ nghĩ bao nhiêu lâu và triển khai code trong khoảng thời gian nào. Nếu mình có sự lường trước, thì mình sẽ chủ động hơn về mặt thời gian.

Đến đoạn thi Quốc tế, lúc đấy thì chỉ có thể điểm danh kiến thức cho các bạn: cái này bạn biết chưa, cái kia nên đọc ở đâu,... rồi động viên giữ gìn sức khoẻ thôi. Ở tầm Đội tuyển quốc gia trở đi, những gì thầy làm được thì đã làm hết rồi, không chỉ riêng thầy, mà còn là nhiều người thầy khác, và cả các anh chị thế hệ trước nữa. Nói về kiến thức, nếu mình nhồi thêm nữa thì cũng chả quan trọng lắm đâu.

Trong suốt những năm thầy đi dạy đội Quốc gia, thầy có ấn tượng nào về một năm, hay một học sinh nào đó hay không?

Thật ra các bạn đi thi Quốc tế, ai cũng có cái giỏi riêng của họ cả. Kết quả có thể có những bạn được huy chương Vàng, Bạc, Đồng khác nhau do đề thi có thể không khai thác vào điểm mạnh của họ, nên kết quả khác nhau cũng dễ hiểu.

Như ngày xưa, vào năm 2011, mình đi sang Thái Lan với Linh (Nguyễn Vương Linh - IOI 2011). Linh có khả năng nghĩ rất nhanh nhưng nói thật thì Linh code không nhanh lắm đâu. Tuy vậy, Linh luôn ở trong trạng thái tự tin rằng mình sẽ đạt được kết quả khả quan. Năm đấy mình đi với cả Linh, Tuệ (Lê Khắc Minh Tuệ - IOI 2011), Nguyên (Nguyễn Tấn Sỹ Nguyên - IOI 2011) và Yến (Nguyễn Hoàng Yến - IOI 2011). Yến năm đấy đi thi, sợ nhất là vấn đề "run". Trong lúc ôn luyện đội tuyển, Yến đã đi qua một thời gian căng thẳng dài. Đi thi thì Yến vẫn run, nhưng may là được huy chương.

Kể cả một số bạn không đi thi Quốc tế, như Trung (Nguyễn Thành Trung - RR) mình cũng rất ấn tượng về kiến thức và lòng đam mê, v.v. Vòng TST[87] thật ra nó cũng chỉ là một kì thi thôi, nên khó mà

[87]TST: Kỳ thi tuyển chọn đội tuyển dự thi Olympic Tin học Quốc tế, hay còn được biết đến là Vòng 2 thi chọn Học

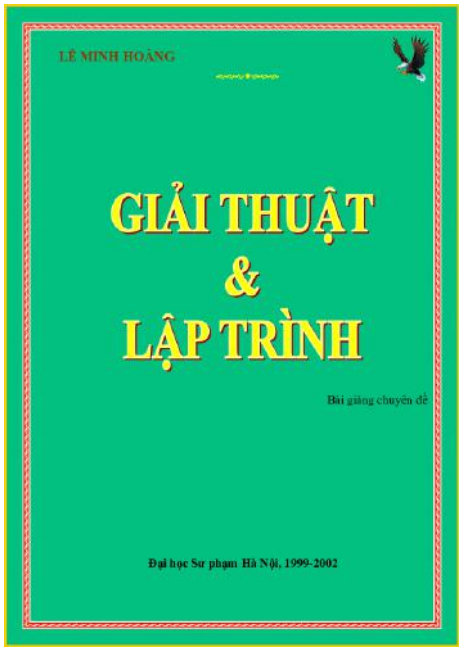
đánh giá được năng lực thực sự. Hồi xưa mình có một bạn mà mình rất là ưng – bạn Vũ (Phạm Quang Vũ), cũng đâu có đi thi Quốc tế đâu!

Nói về thành công, nếu tính thống kê ra thì giải Quốc gia, Quốc tế nó chẳng nói lên điều gì. Tất nhiên về kiến thức kỹ năng thì các bạn thi CP sẽ khác hẳn so với các bạn còn lại. Nhưng thành công nó còn phụ thuộc môi trường và lĩnh vực làm việc sau này. Nhiều bạn còn đạt được thành công ở lĩnh vực khác, chứ không dừng lại ở Tin học nữa đâu. Đi thi là để học cái tư duy, cái tính tự học là nhiều hơn kiến thức như là cách cài Segment Tree!

Cái tinh thần không bỏ cuộc, sự quyết tâm cho những gì mà mình đã bỏ thời gian ra, chắc chắn là điều cần phải cổ vũ. Xã hội nói rằng trường Chuyên luyện gà chọi chỉ để đi thi là hoàn toàn không đúng! Nhìn vào ưu điểm của gà chọi, nó khoẻ hơn, sức bền cao hơn! Nếu không đem đi chọi, thì nó vẫn sẽ chạy nhanh hơn, nhảy xa hơn! Cũng như các bạn thi CP, dù không thi nữa thì niềm đam mê và tinh thần tự học, các kiến thức được trang bị luôn có ích với các bạn.

Vì sao thầy quyết định tạo ra quyển sách đã đặt nền móng cho Lập trình Thi đấu?

Lúc đầu làm gì có sách đâu! Mỗi thầy dịch một ít sách của Ba Lan với Nga rồi tiện đâu dạy đấy. Đến năm 1994, mình tốt nghiệp cấp 3, vào khoa Toán của Đại học Sư phạm. Suốt 4 năm trời, mình có đả động gì tới cuộc thi học sinh giỏi đâu. Sau đấy, mình có hai lựa chọn: xin việc làm giáo viên Toán, hoặc tiếp tục làm công ty Tin học. Sau đấy, khối Chuyên gọi mình ở lại dạy, nên mình vẫn làm ở công ty bình thường và đi dạy học. Mình cảm thấy việc đi dạy khá hay, cũng như bản thân thật sự có ích nên mình tập trung đi dạy từ khoảng năm 2000.



Nói tới chuyện sách, sau 4 năm không đả động gì tới Học sinh giỏi Tin, mình đâu biết kì thi đang bao gồm nội dung gì đâu. Sau hôm đó thì mình phải về học lại. Nếu có sưu tập đề, thì đa số các bài hồi mình thi sẽ theo dạng một trò chơi giữa hai người với nhau, hoặc là những bài NP-complete – đưa ra các giải pháp gần đúng, v.v. Lúc đó là năm 1998, đề bài đã khó lắm rồi. Các thầy cho các bạn chấm test ầm ầm. Lúc ấy, mình nhận ra rằng trình độ mình còn non hơn các bạn lớp 12 trong đội tuyển. Điều đó buộc mình phải học thôi! Cách học tốt nhất là cứ ghi ra các kiến thức đọc được, rồi dạy cho khoá mới.

Sau đấy thì còn lộn xộn lắm. Đến năm 2002, mình đi làm nghiên cứu sinh ở Nhật Bản (JAIST - Viện Khoa học và Công nghệ Tiên tiến Nhật Bản), mình cũng không chắc là mình có về được không nên mình để free trên web dưới dạng file pdf. Mạng Internet lúc đấy thì vẫn chưa phổ biến. Nếu về sau này, bạn nào tiếp thu được kiến thức ấy, thì đỡ phải cày lại từ đầu, có thể bước tiếp.

Hiện tại, mình cũng dự định viết lại vài chuyên đề, nhưng mà nói thật thì mình cũng lười rồi, ngồi lâu thì cũng mỏi lưng! Hồi ấy, khi mình viết những chuyên đề đó, mình phải chuẩn bị cả tuần thì mới dạy được các bạn đội tuyển một buổi. Tức là khi định dạy một chuyên đề nào đó, mình cần một tuần nghĩ bài với sự giúp đỡ của rất nhiều thầy khác, kể cả thầy cũ của mình: thầy Nghĩa, thầy Tùng, thầy Thành, v.v để cho đội tuyển "ăn đủ" kiến thức. Nếu chỉ có một người dạy đội tuyển, mỗi tuần cho ra học một buổi thì thở ra bằng tai mất thôi.

Cũng từ chuyện đấy, mình viết sách. Các chuyên đề đó ban đầu chỉ để dạy thôi, nhưng sau đó thì mình thấy trong nước dùng khá phổ biến. Mình nhận ra rằng môn Tin cần nhất là tài liệu – một học sinh sẽ không cần thầy, tự học vẫn được nếu như tài liệu đủ tốt, đưa ra một lộ trình rõ ràng để tự học. Về sau này, có các bài viết trên VNOI và các nguồn trong và ngoài nước, nhưng mình vẫn cảm thấy thiếu một cuốn sách để hệ thống lại đầy đủ. Từng topic một, từng chuyên đề thì sẽ có bài riêng; nhưng lắp ghép

sinh giỏi Quốc gia.

nó vào trong một bối cảnh chung – dạy cái này vào lúc nào, thì vẫn chưa có tài liệu nào như vậy. Mình cũng đang kì vọng nhưng mà vẫn chưa xong được.

Vậy là thầy dự định sắp viết phần Giải thuật và Lập trình mới?

Thật ra quyển cũ có rất nhiều điểm mình không ưng ý, vì có những thứ mình đặt vào vị trí không hợp lí lắm trong chương trình. Ngoài ra, sách còn thiếu rất nhiều thứ so với tiêu chuẩn hiện tại cho kì thi Học sinh giỏi Quốc gia. Đúng là mình chưa hoàn toàn ưng ý vì mình đang viết giữa chừng ở bên này mà. Bây giờ thì cũng phải cố. Khi mình cố viết, thì coi như là mình có việc để làm. Chứ nếu đi dạy suốt, quanh đi quẩn lại các bài đã chuẩn bị trước thì cảm hứng nó không được tươi mới như lần dạy đầu tiên.

Code minh hoạ chắc sẽ được viết bằng C++ đúng không ạ?

Chắc là phải viết bằng C++. Nói chung sẽ rất khó để tổng hợp mọi thứ thuộc mạch cốt lõi của cấu trúc dữ liệu và giải thuật. Những kiến thức trong kì thi, đôi khi là những kiến thức rời rạc, không gắn vào đâu trong cái mạch chính này. Ngay cả câu chuyện về Segment Tree hay Fenwick Tree, khi thi cử, nó là những thứ rất phổ biến, nhưng để nói về vai trò trong mạch chính của khoa học máy tính, thì những thứ ấy là vớ vẩn. Cảm giác của mình, nó chỉ là mẹo cài đặt mà thôi.

Segment Tree thì ít nhất còn có tư tưởng chia để trị, tách tầng, lưu dữ liệu, v.v, còn Fenwick Tree thì đúng là mẹo cài đặt 100%, công nhận là học xong thì chỉ biết là: OK, cài như thế thôi!

Những sách về thuật toán, đôi khi code cài rất dở, cài theo mô hình rất kém. Nếu mình sao chép chang mô hình của nó ra như vậy, nó sẽ thành một thuật toán chậm. Đúng là code những sách đó rất là vụn.

Thầy dự định sẽ xuất bản quyển sách vào năm 2024 hay năm 2025?

Chắc phải là ít nhất 1 năm, nên có lẽ vào năm 2025. Cũng phải tuỳ theo năm mới mình phải làm gì khác nữa. Nó cũng phụ thuộc vào công việc chính, thì mình phải đảm bảo cái đã. Một giáo trình khi ra đời, mình phải đem đi dạy thử đã, để biết chỗ nào giải thích loằng ngoằng, tìm cách chỉnh sửa. Như hồi xưa, mình rất ngại việc đọc chủ đề thành phần liên thông mạnh. Khi cài xong, mình vẫn chưa hiểu được bản chất tại sao Tarjan lại như thế! Lúc mình chưa làm chủ hoàn toàn, mình đi dạy cũng sẽ lởm khởm, càng giải thích lại càng rối.

Thầy có biết đánh cờ vua không ạ?

Không, mình chỉ biết chơi cờ tướng

Cờ tướng thì bình thường có máy tính hỗ trợ không ạ?

Ngày xưa thì mình đánh được với máy vì máy đánh khá rập khuôn. Còn từ khi mà nó được học qua các thế cờ, thì ngay cả một kiện tướng cũng khó thắng lắm rồi. Bây giờ thì nó lại biết tự học, thì không có ai thắng nổi nó đâu! Thậm chí là trong ván của mình, thì máy còn chấp người đi trước 3 nước (không ăn quân) cũng thắng. Phải gọi là quá mạnh rồi!

Nghe nói thầy có đam mê với các game, đặc biệt là game mô phỏng máy bay, xe tăng. Liệu điều đó có đúng hay không?

Máy bay thì mình thích từ bé. Ngày xưa, nhà mình ở gần bảo tàng không quân, chả có gì chơi, mẹ cứ hay dẫn ra đấy rồi nghe các chú hướng dẫn viên chém gió. Thế là thích máy bay! Cứ đi ra các quán,

trên điện thoại mà có kiểu máy bay bắn bùm chíu, thì cũng thử bắn một cái.

Mình hơi khác với các bạn, vì mình nhanh chán lắm! Phá đảo xong thì mình chán, đánh mãi không qua mình cũng chán. Chứ mình không theo kiểu chơi được game lâu như thằng em đồng hào của mình vẫn hay làm, chơi cả đêm í... Với những tựa game kiểu đấy, chắc là mình không có hứng thú rồi. Ngay cả việc tìm hiểu về các khái niệm trong game, chắc mình chả đủ khả năng hiểu.

Phim thì thầy yêu thích thể loại nào nhất?

Phim hình sự phá án gì đấy. Vừa xem vừa đoán là cái đoạn mình thích nhất. Phim hài cũng hay nhưng đa số mình xem hết rồi! Còn mấy loại vũ trụ Marvel hay DC thì mình không tham. Mình thấy hình ảnh đẹp thôi chứ nội dung thì chả có gì.

Thầy có gợi ý danh sách phim hình sự phá án cho các bạn cày dịp Tết này không?

Mình thấy cái nào có vụ án mà khó đoán thì nó sẽ hay phết!

Thầy tâm đắc với bộ nào nhất?

Phải nhớ một cái tên thôi. Xem nhiều quá thì giờ lại quên béng mất tiêu đề phim ... Nói chung thì mấy phim của Hàn Quốc, Trung Quốc, Mỹ, v.v, các phim cần mình phải đoán một chút, bí hiểm một tí. Chứ phim kinh dị thì mình cũng chả sợ. Mình có thể xem được ý nghĩa bên trong của phim kinh dị, chứ chỉ doạ ma thôi thì mình không thích.

Mấy thể loại sướt mướt, tấu hài lê thê thì mình rút kinh nghiệm rồi. Ngày xưa thì người ta xem vì chẳng có lựa chọn gì khác, và mỗi tuần cũng chỉ có một tập đấy thôi. Chứ bây giờ thì mình có thể đợi hết cả 100 tập rồi xem một mạch, vừa xem vừa ăn tô phở thì tuyệt vời(!).

Thể loại nghe nhạc yêu thích của thầy?

Lúc rảnh thì mình có nghe nhạc nhưng chỉ nghe nhạc không lời thôi. Bởi vì khi làm việc đêm, mình thường có hai màn hình. Màn hình đầu thì để phim, màn hình hai để hiện code. Thật ra khi xem phim, tình tiết của phim thường sẽ chậm, nên không cần tập trung xem phim. Chủ yếu là để có một cái tiếng nói bên cạnh giúp mình không buồn ngủ. Làm đêm thì kị nhất việc đi ra uống café sau khi ngáp một cái dài, và mãi chẳng thể tập trung được, lại lăn ra ngủ tiếp.

Nghe nhạc thì mình nghe tuỳ lúc, thường là trước khi đi ngủ. Chứ nghe một thể loại nhạc trong khoảng thời gian dài vài tiếng đồng hồ, nó rất là nhức đầu. Khi làm việc, nhất là làm việc một mình trong thời gian dài, nên để trong phòng có một ít tiếng động. Nếu trong phòng có TV, cứ bật lên nghe thời sự, nghe tới đâu cũng được. Phải có một tí tiếng ồn thì mới làm được việc, vì yên tĩnh tuyệt đối thì chỉ muốn ngủ thôi!

Làm trong ngành lập trình, thầy có đam mê bộ môn phím cơ không?

Có chứ. Mình chỉ dùng Red Switch hoặc là Brown Switch thôi. Bàn phím cơ thì mình nghĩ rằng nó phải nhạy và ít tiếng ồn. Blue Switch thì tiếng ồn lớn quá, cũng làm mình mất tập trung và còn ảnh hưởng đến người bên cạnh. Giả sử trong phòng máy thi đội tuyển chẳng hạn, mỗi người mang một cái bàn phím rất ồn, thì tiếng gõ như mưa rào, nó ảnh hưởng đến phong độ mình rất nhiều. Mình có thể dùng một cái Red Switch, hay dùng bàn phím của máy xách tay – hành trình phím ngắn và âm thanh nhỏ hơn, cảm giác cái bàn phím đó gõ đỡ mệt và dễ nghe nhất.

Cảm ơn thầy đã nhận lời phỏng vấn cùng VNOI! Chúc thầy và gia đình năm mới mạnh khoẻ, hạnh phúc, để tiếp tục sự nghiệp trồng người của mình!

Phỏng vấn Đặng Đoàn Đức Trung

Interviewer: Vương Hoàng Long – ICPC World Finalist 2021

Xin chào Kuroni, cảm ơn bạn đã nhận lời tham gia phỏng vấn tạp chí VNOI 2024, được biết bạn là người rất nổi tiếng và sâu sắc, VNOI mong qua buổi phỏng vấn này có thể đem đến cho độc giả sâu hơn cái nhìn về bạn và những trải nghiệm của bạn trong hành trình đến với Lập trình thi đấu. Kuroni là một trong những người hiếm hoi đạt được thành công trên nhiều lĩnh vực. Hiện tại bạn là người giữ mức rating cao nhất Việt Nam và là một người làm đề vô cùng nổi tiếng trên Codeforces[88] , bảo chứng cho những bộ đề hay trên trang web này. Ngoài ra bạn cũng đạt được học bổng toàn phần bậc Tiến sĩ về Toán kinh tế ở trường đại học bên Mỹ. Bên cạnh việc học, bạn là một gamer rất khủng với việc tham dự OSU World Cup nữa. Qua buổi phỏng vấn lần này, bọn mình hi vọng bạn có thể chia sẻ đến cho độc giả những kinh nghiệm cũng như bí quyết để bạn có thể toàn diện trong nhiều lĩnh vực như thế.

Bạn có xuất phát điểm cấp 2 là học sinh chuyên Toán, sau đó lên cấp 3 bạn học ở trường Phổ Thông Năng Khiếu. Nổi bật nhất bạn đã từng tham gia Entropy (kì thi 'Đường lên đỉnh Olympia' của trường PTNK[89]), lý do gì đã chuyển bạn từ Toán sang Tin, mà lại không chuyển từ Toán sang một con đường thú vị không kém - thi Olympia?

Lý do mình chuyển từ Toán sang Tin với mình không thi Olympia là hai cái...hoàn toàn khác nhau. Việc mình quyết định rẽ hướng sang Tin là do sau cấp 2 mình dần mất hứng thú với việc làm Toán Chuyên, cũng vì hồi đó mình kém hình lắm. Đến bây giờ khi sang Mỹ mình cũng vẫn kém hình (cười). Còn việc mình không thi Olympia thì do lúc mình thi Entropy cũng là trong giai đoạn thi vào đội tuyển Tin của trường, thế nên sau khi thi xong Entropy thầy Hùng (Nguyễn Thanh Hùng - đồng tác giả bộ sách giáo khoa Chuyên Tin) có đùa mình là may rằng mình không thắng Entropy, chứ không thi Olympia mất rồi', cũng từ đó mình không thi Olympia luôn. Tuy có tiếc nuối khi ước mơ từ nhỏ của mình là được đứng ở cầu truyền hình một lần, nhưng nhờ đó mình mới có trải nghiệm như thi HSGQG[90] , TST[91] . Từ một góc nhìn khác thì nó vẫn tốt.

Được biết thì bạn bắt đầu thi VOI từ lớp 11 và được thi TST. Vậy lúc đó kết quả của bạn như thế nào?

Đợt đó mình thi thì được rank 4 chung cuộc, ở thời điểm đó thì mình là thí sinh lớp 11 có thứ hạng cao nhất ở TST. Ngoài ra, đấy cũng là thứ hạng của mình năm lớp 12 luôn. Điểm đặc biệt là ở cả hai năm, ngày thi đầu tiên mình đều nằm ngoài top 15 và ở ngày 2 mình bứt lên được rank 4.

Có thể các bạn độc giả chưa biết thì có một 'lời nguyền' cứ thí sinh nào được rank 4 TST thì sẽ đều không đậu vào đội tuyển IOI[92] và từ đó đến giờ thì vẫn chưa ai phá được.

Lớp 11 đã được rank 4 TST thì cũng là một kết quả rất tốt, bạn có bí quyết nào để đạt được rank cao như vậy không?

Hồi đấy mình có một kỹ năng mình vô cùng tự hào là code các bài cấu trúc dữ liệu khó mà không bị lỗi. Mình nhớ năm ấy thi TST ngày 2 thì có một bài thuần cấu trúc dữ liệu yêu cầu thí sinh phải code cấu

[88] Codeforces: một trang web tổ chức các cuộc thi lập trình thi đấu với các dạng đề, bài tập đa dạng.

[89] PTNK: là tên viết tắt của trường Phổ Thông Năng Khiếu, một trường THPT trực thuộc ĐHQG-HCM.

[90] HSGQG: là viết tắt của kỳ thi Học sinh giỏi Quốc gia.

[91] TST: Kỳ thi tuyển chọn đội tuyển dự thi Olympic Tin học Quốc tế, hay còn được biết đến là Vòng 2 thi chọn Học sinh giỏi Quốc gia.

[92] IOI: Kỳ thi Olympic Tin học Quốc tế.

trúc dữ liệu dạng persistent[93] . Mình có nghĩ ra và code được bài đấy trong giờ, nhờ bài này nên mình bứt lên được hạng 4. Hôm đấy chỉ có 2 thí sinh AC được bài này là mình với anh Nhật (Hoàng Xuân Nhật - IOI 2018). Điều này cũng bởi vì hầu hết năm lớp 10, 11 mình đều cày các bài cấu trúc dữ liệu nên tạo cho mình lợi thế rất nhiều.

Với vòng TST bạn được rank 4, vậy khi thi APIO[94] đã có khó khăn nào khiến bạn không đủ điểm để đậu đội tuyển IOI?

Năm đấy mình không đậu cũng là một việc hiển nhiên, một phần vì năm ấy đề cực kỳ khó, anh Thắng (Phạm Đức Thắng - IOI 2018) thi APIO cao nhất đoàn Việt Nam cũng chưa được đến 200 điểm. Đề năm ấy mình đánh giá là khó nghĩ, mà năm mình lớp 11 thì khả năng nghĩ thuật của mình cũng chưa chín muồi nên việc thất bại cũng không phải không ngờ tới được.

Bạn đã có thay đổi gì về chiến thuật để tiếp tục đạt được rank 4 TST ở năm thi tiếp theo không?

Sau kì thi TST lớp 11 thì mình bắt đầu luyện tập nhiều hơn đề OI. Trước đấy thì mình chỉ có làm đề trên Codeforces thôi mà thời đó Codeforces bài tập đòi hỏi phải code khá nhiều nên mình cũng code khỏe. Sau đó, năm lớp 12 mình cố gắng phát triển khả năng nghĩ bài bằng cách luyện tập các đề OI ở các nước khác. Trong thời gian này mình cố gắng học nhiều thuật toán hơn thay vì như trước chỉ làm những bài cấu trúc dữ liệu. Lúc này cũng là khởi điểm mình bắt đầu đi làm Problemsetter[95] trên các trang OJ, khi tạo đề thì cũng đòi hỏi mình phải biết nhiều thuật toán và nhiều hướng nghĩ khác nhau, chính vì thế giai đoạn này mình tập trung để nâng cao tư duy nhiều.

Vậy sau đó với kỳ thi APIO năm lớp 12, bạn đã có trải nghiệm như thế nào?

Trước khi thi TST thì thầy Hùng cũng nói với mình đừng để rank 4. Lúc đấy nghĩ là nếu mình thi tốt thì sẽ phá được lời nguyền "rank 4" này, không thì cả thế giới sẽ biết mình là "Chú Tư". Thi xong TST khi biết mình vẫn giữ rank như năm ngoái thì mình có khóc với anh Hạnh, anh ý cũng cười cười không biết ám chỉ gì (cười).

Đa số các CP[96] -er nổi tiếng trong miền Nam đều ít đi du học, bạn đi du học ngay khi tốt nghiệp cấp 3 với một trường rất tốt ở Mỹ - trường Purdue. Bạn có thể chia sẻ về cách bạn chuẩn bị hồ sơ đi du học được không?

Gia đình mình có điều kiện hơn một tí nên từ sớm mình đã được tư vấn du học và ôn SAT[97] , mình đã học SAT từ đầu năm lớp 11. Điều này cũng vì gia đình đã định hướng mình đi du học từ đầu cấp 3 nên ngay từ trước khi vào lớp 12 là mình đã thi xong SAT và chuẩn bị viết luận rồi. Mình cứ thế chuẩn bị hồ sơ liên tục từ lúc mình học đội tuyển ở trường đến lúc mình thi TST năm lớp 12. Ngay khi tốt nghiệp

[93] persistent: trong ngữ cảnh nói về cấu trúc dữ liệu thì là dạng cấu trúc dữ liệu có thể quay trở lại một trạng thái bất kỳ. Tiêu biểu là Persistent Segment Tree.

[94] APIO: Kỳ thi Olympic Tin học Châu Á -– Thái Bình Dương.

[95] Problemsetter: người chuẩn bị bài tập cho kỳ thi.

[96] CP: Competitive Programming - Lập trình thi đấu

[97] SAT: Scholastic Assessment Test là bài thi chuẩn hóa nhằm đánh giá năng lực của học sinh được phát triển và sở hữu bởi College Board – tổ chức giáo dục nổi tiếng phi lợi nhuận tại Mỹ. Nhiều trường đại học trên thế giới hoặc ở Mỹ yêu cầu bài thi này để xét duyệt đầu vào.

là mình đã sẵn sàng để đi du học rồi. Chính vì thế nên mình cũng không cần mất 1 năm làm hồ sơ như các bạn khác.

Thành tích học thuật của bạn rất khủng nhưng vẫn có thời gian để học SAT và chuẩn bị hồ sơ du học. Bí quyết nào giúp bạn có thể duy trì những việc này?

Mình học đội tuyển thì trên trường mình cày nhiều chứ về nhà cũng không nhiều lắm, chắc chắn là mình không cày nhiều bằng một số bạn mình biết được như bạn Khoa (Nguyễn Ngọc Đăng Khoa, IOI - 2023), mình nhìn profile Codeforces bạn này là biết mình không thể cày nhiều như bạn này ở hồi đó được (cười).

Hồi cấp 3 mình cũng như bao học sinh bình thường khác, mình không thể dành hết thời gian cho CP được mà sẽ có nhiều hoạt động khác nữa. Ngoài CP mình có học thêm SAT và chơi game, nhưng mình khá...lười học SAT, hồi đó mình chỉ đơn giản là lên làm bài của trung tâm giao và về nhà làm thêm các dạng là hết rồi. Mình cảm thấy cách mình phân bố thời gian không hiệu quả bằng các bạn khác nhưng đối với mình thì vừa đủ để cân bằng việc học CP và chuẩn bị hồ sơ du học cùng một lúc.

Lúc bạn chưa qua Mỹ bạn đã vô cùng nổi tiếng trong cộng đồng CP thế giới. Theo bạn điều gì đã làm bạn nổi tiếng như vậy?

Trước mình có tham gia server Discord 'AC', một server nước ngoài nổi tiếng trong giới CP. Mình đã tham gia và giao lưu trên đấy khá thường xuyên, nhưng mình nghĩ việc mình đột nhiên nổi tiếng như vậy cũng là từ tham gia tổ chức các contest trên Codeforces. Hồi lớp 11 mình đã tổ chức contest rồi nhưng mình thấy nó cũng không tốt lắm, mãi đến năm lớp 12 mình tổ chức một contest nữa thì được đánh giá cao. Từ lúc đó mình bắt đầu kết nối đến các bạn nước ngoài trên Codeforces có rating cao và cùng chí hướng để ra lò một số series contest khá là hay, mọi người biết đến tên mình nhiều hơn và đánh giá tốt chất lượng các contest mình làm. Có một kỉ niệm mình rất nhớ và tự hào đấy là đợt ICPC[98] World Finals[99] ở Dhaka, ông Mike[100] (founder Codeforces) đã chào mình và bảo mình làm thêm contest đi (cười). Đầu đuôi câu chuyện lúc đấy là mình có bắt gặp Mike và xin chụp hình cùng, ông ý có hỏi tên mình và mình trả lời, sau đó mình không ngờ tới được là ông ý nói như này: 'Oh, I remember you, you made very good contests, please make more contests'.

Lúc đấy có hai điểm mình chú ý, mình rất hãnh diện vì Mike nhớ đến mình là một người làm contest hay, thứ hai là Mike nhớ mình không làm contest từ rất lâu rồi, khoảng chừng là 2 năm. Thế nên lúc đó mình cảm thấy rất là vui vì có trùm của một trang như thế nhớ đến.

[98]ICPC: International Collegiate Programming Contest - Cuộc thi Lập trình Quốc tế lâu đời và danh giá nhất dành cho sinh viên các trường đại học và cao đẳng trên toàn cầu.

[99]World Finals: Cuộc thi ICPC cuối cùng trong một mùa giải bao gồm những đội xuất sắc nhất của các trường đại học trên thế giới vượt qua vòng loại vùng (Regional contest)

[100]Mike: (Mike Mirzayanov) một lập trình viên người Nga. Thường được biết đến là người tạo ra nền tảng chấm bài trực tuyến Codeforces.

Lý do bạn chuyển việc làm contest trên Codeforces sang những contest cho Việt Nam không?

Lý do mình ngưng tổ chức contest có lẽ là do bị burnout[101] thôi! Trong quãng thời gian đó mình tổ chức các contest liên tục nên đến một lúc bị bí ý tưởng. Việc mình chuyển hướng sang làm contest cho cộng đồng Việt Nam vì phần lớn mình được mời làm Coordinator[102] cho VNOI CUP 2023. Khoảnh khắc mình được mời làm vai trò đó, mình đã bí ý tưởng được 2 năm rồi nên mình cũng khá tự ti, nhưng sau đó mình ngạc nhiên là khi đảm nhận một vị trí quan trọng như vậy thì mình có cảm hứng và ý tưởng tạo đề lại. Mình cảm ơn VNOI nhiều vì đã tạo động lực cho mình quay lại việc tổ chức các contest. Tiết lộ một chút thì mình đang cùng bạn mofk (Nguyễn Đinh Quang Minh - max rating CF 2726) làm một contest Div 1 trên Codeforces. Nếu điều này thành hiện thực thì Kuroni x Mofk Cup sẽ được làm trên Codeforces chứ không phải là VNOJ (cười).

Dành cho các bạn chưa biết thì đề VNOI CUP 2023 được đánh giá là một bộ đề chất lượng và cực kỳ hay, rất nhiều bạn thích và tín nhiệm với bạn Kuroni là trưởng ban ra đề.

Sau khi học cấp 3 và chuyển sang Mỹ, thời gian đầu bạn có gặp trở ngại gì không?

Mình cảm thấy may mắn khi hòa nhập được văn hóa ở Mỹ. Lúc đấy mình đang trong giai đoạn nổi loạn nữa nên suy nghĩ của mình là luôn thích được đi khám phá và không muốn ở nhà. Một ấn tượng của mình trong thời gian đầu ở đây là mình được thầy Ninghui Li - thầy coach ICPC ở Purdue để ý, do là vì trên Codeforces mình có đổi thông tin trường học của mình thành Purdue. Vì vậy ngay từ tuần thứ hai của năm nhất mình đã được thầy dẫn dắt và làm trợ giảng cho một số lớp học của thầy. Từ đó mình cũng được kết nối với rất nhiều bạn đam mê CP ở Purdue. Trong số đó có một cái tên cũng khá nổi tiếng với cộng đồng Codeforces là bạn Monogon (max rating CF 2705).

Bạn đã từng đủ điều kiện tham dự 2 ICPC World Finals năm 2020 và 2021 nhưng phải tới năm 2022 thì mới có giá trị do là 2 kỳ thi kia bị ảnh hưởng bởi Covid 19. Bạn có thể chia sẻ về điều này được không?

Đúng là như vậy, mình có được tham dự kỳ thi ICPC World Finals tổ chức ở Nga nhưng mình không đi được do bị ảnh hưởng từ Covid. Lý do trực tiếp mình không đi được là vì Visa mình hết hạn nên không thể quay lại Mỹ. Vì thế khả năng mình về Việt Nam để gia hạn Visa và quay lại Mỹ là khá thấp do lệnh hạn chế đi lại ở các nước nên mình quyết định không đi.

Có một sự thật thú vị rằng bạn Kuroni là người đầu tiên trên thế giới đủ điều kiện tham dự ICPC World Finals tận 4 lần. Lần gần nhất bạn Kuroni thực sự tham gia là ICPC World Finals 2022 ở Dhaka, Bangladesh

[101] burnout: mất động lực.

[102] Coordinator: người tổ chức các contest cho OJ.

Kuroni đạt được rank 17 ICPC World Finals 2022, một rank cao ở một kỳ thi đẳng cấp. Vậy quá trình luyện tập của bạn cho kỳ thi này là như thế nào?

Năm đầu tiên ở trên đại học của mình là khoảng cỡ 2020 thì mình đang hướng tới ICPC World Finals ở Nga, mình đã luyện tập rất nhiều, hơn cả lúc mình học cấp 3. Một phần vì mình khá cay do trượt đội IOI, có một khoảng thời gian cỡ 3-4 tháng liên tiếp mà mỗi ngày mình sẽ chọn ra 1 bài độ khó 2700 bất kỳ trên Codeforces và cố gắng giải nó trong 3-4 tiếng, có một số ngày đỉnh điểm mình giải đến 2-3 bài 2700 cơ. Ngoài ra mình còn luyện riêng với đội ICPC nữa, thường thì bọn mình sẽ chọn ra một bộ đề Regional ngẫu nhiên và mỗi một tuần lại làm với nhau 1 lần, mỗi lần như thế sẽ dành ra 5 tiếng để làm và 2-3 tiếng để thảo luận về lời giải cũng như chiến thuật sau khi làm xong. Không biết may hay rủi nhưng Monogon là một số ít trong những bạn mình biết như Bùi Hồng Đức (IOI 2019, 2020) rất thích code những bài hình, nên là đợt đó mình gần như không quan tâm đến việc code những bài hình mà cứ quăng hết cho Monogon.

Sang quãng thời gian 2020 - 2021 và 2021 - 2022 thì là một khoảng trầm không chỉ trong hành trình CP của mình mà trong cuộc đời mình luôn. Hai năm đó cũng là hai năm cao điểm nhất của dịch Covid 19, tinh thần mình khá bất ổn và nếu các bạn để ý trên Codeforces thì mình có một khoảng nghỉ hơn 1 năm rưỡi đến 2 năm mà không có hoạt động nào nhiều trên Codeforces như trước. Mình nhớ một bạn đăng blog trên Codeforces hỏi tại sao mình lại ngưng hoạt động trên Codeforces lâu như thế, blog viết cũng dài và lượt upvote[103] cũng cao, nhờ đợt này mới biết mình nổi tiếng trong cộng đồng đến thế.

Đến giữa năm 2022 thì có vẻ Covid tốt hơn dần nên mình ra ngoài và giao lưu với bạn bè nhiều hơn, đợt đấy mình có gặp bạn ToMo ở Purdue (Nguyễn Thành Minh - hai lần tham gia APIO 2018, 2019) - một người ôn TST chung với mình. Sau này mình với bạn ý ở chung ký túc xá luôn, trong quãng thời gian này mình đã tìm lại cảm hứng với CP, lâu lâu ăn tối thì bạn ý lôi bài trên Codeforces ra và thảo luận với mình, các chủ đề bọn mình thảo luận nhiều khi không chỉ CP mà còn là toán nữa, ToMo cứ nói ra 1 tràng toán xong mình ngồi nghe.

Kể từ đó mình tập trung cày CP để hướng đến ICPC World Finals ở Dhaka, mình phải làm những bài khó hơn tầm rating 3000 và những bài trên trang AtCoder. Trùng với khoảng thời gian này thì mình theo đuổi hướng nghiên cứu nên khai thác những góc nhìn rộng hơn và bao quát chứ không chỉ ở phạm vi CP, may mắn là mình học được một kỹ năng quan trọng liên quan đến chủ đề 'Tối ưu hàm lồi', mình cũng đóng góp một bài trên VNOI Wiki liên quan đến chủ đề này.

Do Covid nên ICPC World Finals ở Dhaka bị dời lại khá là sâu, lúc đó đồng đội của bạn là Monogon cũng đã đi làm rồi nên không dành nhiều thời gian cho CP được nữa, vậy cách làm việc của cả đội cũng như chiến thuật có bị thay đổi không?

Có một điều mình hối hận là không tái hợp với team sớm hơn để luyện tập cho ICPC World Finals. Vì thế nên lúc thi bọn mình cũng gặp một số trở ngại, Monogon với mình không còn kết hợp nhuần nhuyễn như hồi năm 2020 nữa, nhưng về cơ bản thì cái luồng làm việc của team vẫn được giữ nguyên như đọc

[103] upvote: tính năng 'yêu thích' bài viết trên trang Codeforces.

bài, nghĩ bài, debug, ...Trong đó thì mình vẫn đảm nhiệm hầu hết các bài, quan trọng nhất vẫn là các bài cấu trúc dữ liệu, quy hoạch động hay các bài thiên hướng như đề Codeforces. Phần còn lại như các bài hình này nọ thì Monogon sẽ làm, các bài toán thì sẽ đưa cho bạn Richard Li - đồng đội còn lại của mình. Bạn là một người có khả năng làm toán khá là tốt nhưng buồn là bạn code bug kinh khủng (cười). Thế nên team cho bạn ý một vai trò nghĩ bài và sẽ đọc lời giải cho mình hoặc Monogon code. Đợt bọn mình thi thì cả team code một bài và mình bug xong debug không ra nên đưa lại cho các bạn debug, team cũng không ai debug ra nên bài đầu tiên bọn mình AC là phải sau 60 phút đầu tiên, một khởi đầu chậm so với các team khác.

Khi kỳ thi kết thúc thì mình thấy ở bạn một sự thất vọng nhẹ mặc dù rank tương đối cao. Vậy kết quả năm ấy có ảnh hưởng đến bạn như nào cho việc chuẩn bị WF sắp tới?

Mình sẽ ví World Finals năm ấy như APIO thứ hai của mình. Mình cảm thấy đủ thực lực và thời điểm cũng chín muồi nhưng mình lại để vụt mất trong tầm tay, từ sau WF đấy mình cũng làm bài nhiều hơn, tuy không thường xuyên như hồi năm nhất đại học. Nếu các bạn để ý thì mình làm CF thường xuyên lại rồi, cứ 2-3 tháng mình lại thi một contest. Ngoài ra mình tập làm nhiều bài hình trở lại vì team mình đã thiếu đi bạn Monogon, mình đã quen với việc tiếp cận bài hình và nghĩ ra code.

Theo mình thấy kĩ năng làm những bài hình là một trong những cái quan trọng nhất mình đã học kể từ sau WF ở Dhaka. Bởi vì thiếu Monogon rồi như thiếu một 'nhân tố X' ấy, bạn ấy code hình full-time luôn nên là một sự mất mát rất lớn đối với mình .

Như mình đã chia sẻ thì mình bắt đầu tạo bài trở lại và kết nối nhiều hơn với cộng đồng Tin học Việt Nam. Mình có nhiều bạn để bàn bài cùng và thấy việc thảo luận về các bài với các bạn giỏi là một cách rất nhanh để phát triển. Những năm đầu là với Monogon và các bạn trong server 'AC', những năm gần đây thì là bạn Bách (Trần Xuân Bách - IOI 2022, 2023), bạn Kiên (Vũ Hoàng Kiên - IOI 2019, 2020), bạn Vũ (Nguyễn Hoàng Vũ - IOI 2021), anh Quang (Nguyễn Diệp Xuân Quang - IOI 2017), bạn Phát (Hồ Ngọc Vĩnh Phát - IOI 2021), mofk, ... cùng rất nhiều bạn nữa trong VNOI. Mình khá ngạc nhiên là qua ICPC WF mình không làm thường xuyên CF nữa nhưng từ việc này mình vẫn nâng cao khả năng của mình ở CP lên được.

Bạn có mục tiêu và kỳ vọng gì cho ICPC World Finals 2023 lần này?

Như có một lần mình chia sẻ, là mình sẽ luôn đặt mục tiêu cao hơn khả năng hiện tại của mình, thì mục tiêu lần này của mình là Gold Medal, đồng nghĩa cũng là trong top 4 (cười).

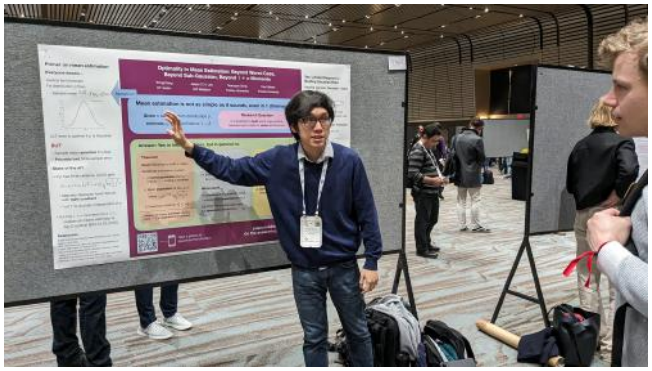
Có thể đối với nhiều người thì việc WF 2023 bị dời là một tin không hay, nhưng với mình là một điều may mắn. Đấy là sau WF 2022 thì mình cũng bắt đầu luyện tập lại rễ nên nghĩ là sẽ không đủ thời gian để luyện tập đạt được mục tiêu. Tới khi nghe tin dời lại thì mình nghĩ đây là ý trời bảo mình giành huy chương, thế nên mục tiêu của mình tiếp tục sẽ là 'Chú Tư' cơ mà không ở Việt Nam nữa mà là 'Chú Tư' thế giới luôn!

Mình cũng thấy bạn đã rục rịch luyện tập lại với team, vậy bạn có mục tiêu đạt LGM[104] (Legendary Grandmaster - rating CF 3000) trước ICPC WF không?

Đây cũng là mục tiêu của mình trong năm 2023 nhưng nó đã không thực hiện được dù hai contest cuối của năm 2023 được cộng điểm. Hiện tại, mình kết hợp làm bài trên nhiều OJ khác, điển hình là mình đang làm lại các dạng bài OI để nâng cao khả năng nghĩ của mình ở trong giờ thi tiếp, mình có mong muốn là được LGM, nhưng trước hay sau WF thì mình không dám chắc. Mình có thể tiết lộ là ở NAC (kỳ thi ICPC Regional ở Bắc Mỹ) mình đã gặp tourist[105] (max rating CF 3979). Mình nói chuyện với tourist thì tourist chia sẻ rằng nếu mình đặt mục tiêu LGM thì sau khi đạt được sẽ không có ý chí để luyện tập nữa. Mình đang phân vân là không biết nên đặt mục tiêu LGM trước hay sau World Finals, nhưng mà mình sẽ nhắm có kỹ năng của LGM trước World Finals.

World Finals được dời tầm khoảng tháng 4, tức bạn còn khoảng 3 tháng để chuẩn bị. Tất cả mọi người đề rất mong được chứng kiến Kuroni tỏa sáng ở WF lần này. Từ câu chuyện của Kuroni ta có thể thấy rất nhiều sự luyện tập, tính toán chiến thuật và sự đam mê để đạt được đến đẳng cấp của bạn Kuroni. Nếu xét ở khía cạnh CP thì bạn Kuroni rất khủng, tuy nhiên, cái điểm khác biệt giữa bạn Kuroni so với nhiều người khủng CP mình biết là bạn có thể khủng về những thứ khác, cụ thể là nghiên cứu. Có rất nhiều người ngại bước ra vùng an toàn và họ chỉ tập trung vào CP. Tuy nhiên CP giỏi thì tốt nhưng không thể xem đấy là công việc được trừ khi bạn là tourist.

Bạn có thể chia sẻ quá trình học đại học của khi đến cuối lại quyết định đi nghiên cứu, dù trước đó đã có thực tập ở Meta không?



Một phần do duyên, một phần do giai đoạn nổi loạn của mình đã đề cập trước đó, nhưng không liên quan đến vấn đề ở với gia đình mà ở đây liên quan đến Software Engineering[106] . Khi mình học năm 2 thì mình có học một lớp 'Thiết kế thuật toán' ở trường Purdue, mình có gặp thầy Alex Psomas do đang là năm Covid nên mình cũng không đến lớp hay nghe giảng online và mình chỉ nhớ lớp đó mình qua rất dễ, mình chỉ cần vào làm bài tập về nhà thi thôi là được A+. Lớp đấy cũng không có quá nhiều ký ức với mình, nhưng cái kí ức mà mình ấn tượng là sau khi được A+ lớp đấy thì thầy đã trực tiếp liên lạc với mình và hỏi mời mình học lớp của thầy là lớp 'Thuật toán trong kinh tế'. Tất nhiên lúc đấy mình năm 2, việc được một giáo sư của 1 trường rất to mời mình học lớp của thầy thì tất nhiên mình không thể nào từ chối được. Và sau khi mình học lớp đấy, mình phát hiện ra có rất nhiều thuật toán mà ngoài CP cực kì thú vị. Trước đây mình chưa từng nghĩ là kinh tế là một thứ mình sẽ đam mê, nhưng sau khi học lớp đấy thì mình thấy sự giao thoa giữa thuật toán và kinh tế rất là hay, nó bao hàm rất nhiều yếu tố khác nhau, không chỉ liên quan đến kinh tế vi mô, kinh tế vĩ mô mà còn là các chủ đề rất là tổng quát trong Computer Science[107] bao gồm độ phức tạp của thuật toán, thiết kế các thuật toán khác nhau.

[104] LGM: Legendary Grandmaster – mức điểm hơn 3000 trên Codeforces.

[105] tourist: tên thật là Gennady Korotkevich - một lập trình viên người Belarus. Anh có 6 Huy chương vàng IOI (2007 -– 2012) và 2 lần vô địch thế giới tại kỳ thi ICPC (năm 2013 và 2015).

[106] Software Engineering: Kỹ thuật/Công nghệ phần mềm.

[107] Computer Science: Khoa học máy tính.

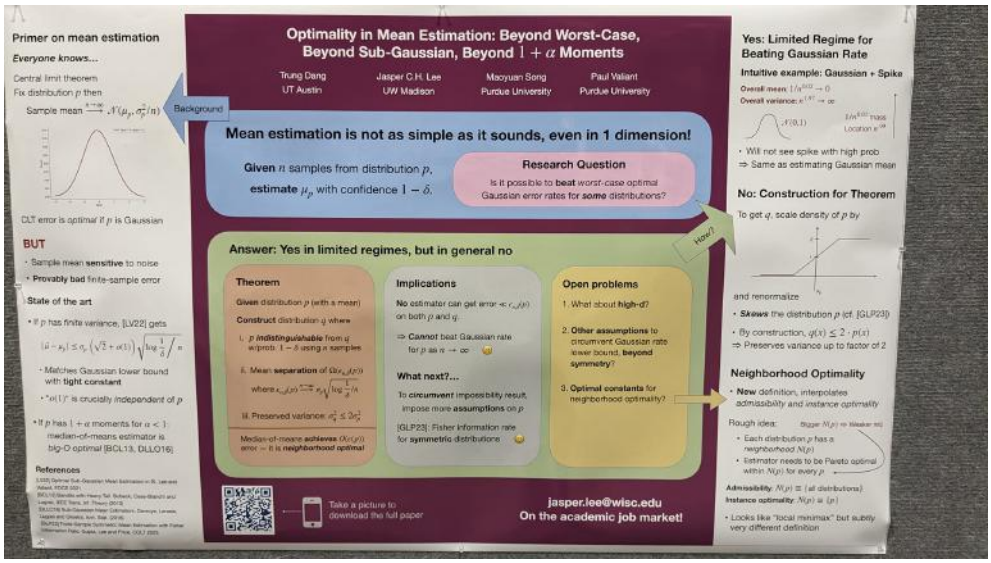
Trong đó cũng có 'Convex Optimization' cũng là 1 nhánh của Computer Science và nghiên cứu rất nhiều về việc tối ưu những hàm lồi, khi biết được ý tưởng của cái này thì mình mới nhận ra đã được áp dụng rất là nhiều.

Trong CP thì các bạn có thể biết 'Convex Hull Trick' hay 'Alien Trick', là một trong những áp dụng rất cao của 'Convex Optimization' trong CP. Mình nhớ có dịp mình đọc một thuật toán luồng cực kì hay mà được áp dụng cho 1 bài toán kinh tế và mình cảm thấy là 'Oh não to quá!', chính khoảnh khắc đấy mình quyết định là mình phải suy xét xem bản thân xem có hợp nghiên cứu hay không, tại vì trước đây điểm mạnh của mình là code khỏe nhưng thực sự mình lại thích nghĩ bài hơn. Mình nhận ra hướng nghiên cứu là một nghề cho phép mình nghĩ bài mà có thể kiếm ra tiền và cho phép bản thân có thể thử. Sau đó mình cũng liên lạc nhiều hơn với thầy và cùng thầy nghiên cứu, rồi mọi thứ diễn ra rất suôn sẻ và sau khoảng 1 năm thì thầy với mình có chung bài nghiên cứu. Hiện tại thì mình cũng đang làm với 1 thầy khác, tuy nhiên câu chuyện đó lại rất buồn cười nên mình sẽ kể sau.

Còn về lý do mình lại không theo hướng đi làm công ty dù mình có intern Meta thì mình cảm thấy trong lúc intern Facebook cũng khá là hay nhưng lại không hay bằng việc mình theo nghiên cứu, như mình nói mình thích ngồi nghĩ bài hơn là ngồi code. Ở Meta thì việc intern của mình không phải là dễ, hầu hết thời gian là mình phải ngồi thiết kế các hệ thống khác nhau và nó không phải cái mình thích nghĩ, như mình nói là mình thích ngồi nghĩ các vấn đề phức tạp hơn mà không nhất thiết phải code nó ra. Tất nhiên mình biết đi con đường học cao học có 1 cái trở ngại rất là lớn đó là mình sẽ cực kì nghèo trong vòng 6 năm tới nhưng sau khi cân đo các điểm mạnh và điểm yếu thì mình quyết định là thà vừa đủ sống trong vòng 6 năm tới nhưng được theo đuổi con đường mình đam mê hơn là code và bị burnout sau khoảng 4 năm nữa.

Mình cũng cảm thấy đúng, trong trường hợp này thì đâu đó chọn công việc mình thích nhưng ít tiền hơn là chọn công việc mình không thích nhưng nhiều tiền. Mặc dù bạn Kuroni có thể đi làm ở các công ty Quant Trading để kiếm thật nhiều tiền. Việc từ chối làm một công việc rất nhiều tiền để đi học cao học là việc không phải ai cũng làm được, rất hiếm có.

Bạn có thể trình bày hướng nghiên cứu của bạn và tại sao nó quan trọng?



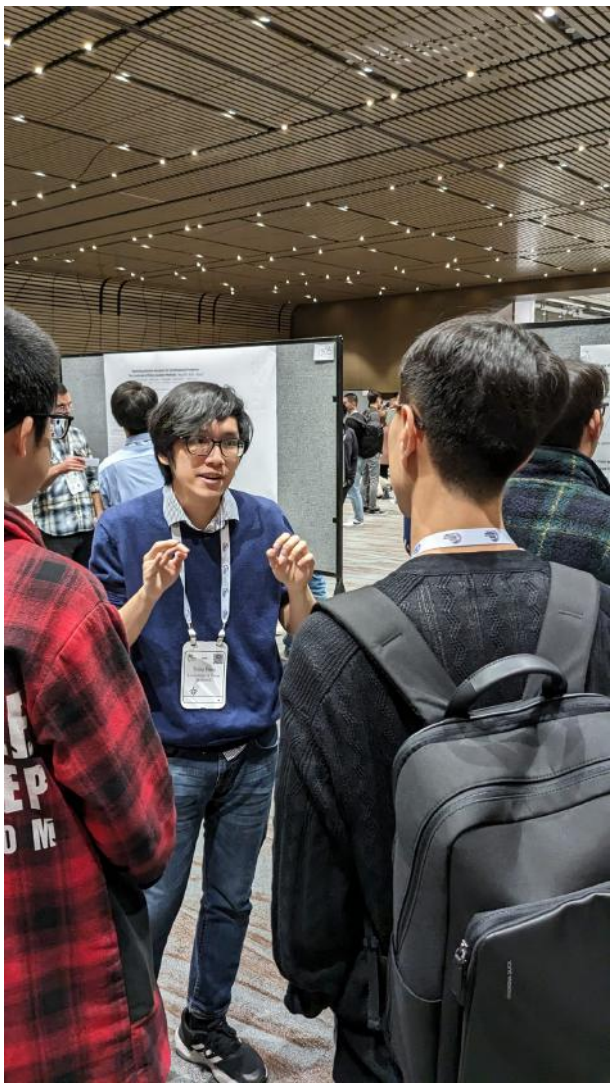
Mình sẽ nói về nghiên cứu trong quá trình mình học Tiến sĩ. Thật ra nó cũng là quá trình nghiên cứu liên tục từ đại học thôi. Mình đang nghiên cứu về giao thoa giữa kinh tế và thuật toán. Với một bài toán cơ bản như này: giả sử bạn muốn đấu giá 1 món đồ, nhưng cách đấu giá ảnh hưởng rất nhiều đến hành động của người tham gia, tức là nếu bạn đấu giá và chọn người thắng cuối cùng hoặc cái giá cuối cùng không hợp lý thì bắt đầu sẽ có những hành vi gian lận như người tham gia đấu giá bắt đầu đi tìm hiểu, cơ cấu và điều khiển giá thị trường, tất nhiên là người đấu giá không muốn như thế.

Điều này liên quan đến chủ đề nghiên cứu của mình, mình sẽ tiếp cận thị trường và quan sát những người đấu giá theo các cách và những món đồ khác nhau. Việc của mình sẽ thiết kế thể thức đấu giá sao cho người tham gia không thao túng được thị trường hay thay đổi chiến thuật của người đấu giá (incentive compatibility) và cho phép thu lợi nhuận nhiều nhất cho người bán đấu giá từ người tham gia (revenue optimization).

Về cơ bản trong phạm vi bài toán này chính là sự giao thoa giữa thuật toán và kinh tế, gọi là 'mechanism

design'. Thật ra nó chính là 'algorithm design' nhưng kết hợp thêm yếu tố 'kích thích' (incentive) vào trong đó, để có thể thiết kế một thuật toán làm cho tất cả đối tượng đều cảm thấy hài lòng. Nghiên cứu này đã đi theo mình từ năm 3 đại học đến bây giờ.

Hẳn các bạn độc giả cũng rất tò mò về cuộc sống của một người theo học Tiến sĩ ở Mỹ, vậy Kuroni có thể chia sẻ về cách bạn viết nghiên cứu và xuất hiện ở các hội nghị được không?



Mình cảm thấy cuộc sống của mình khá thoải mái. Buổi sáng mình sẽ đọc các bài nghiên cứu với 2 mục đích chính. Thứ nhất là để cập nhật kiến thức của mình về những nghiên cứu khác. Thật sự thì những nghiên cứu của mình khá là thuần toán, kiểu áp dụng và mình cũng không code nhiều. Thứ hai là để mình có thể học những cách chứng minh của người ta, xem người ta dùng những kĩ thuật nào để chứng minh. Ví dụ như vừa rồi mình vừa đọc một kỹ thuật rất hay, và đang thử áp dụng vào bài hiện tại của mình. Tất nhiên là mình vừa học hai học kỳ thôi nên cuộc sống của mình cũng chưa có nhiều điều mới, mình cũng tranh thủ để mở rộng các chủ đề có thể theo đuổi qua việc đọc thêm các bài nghiên cứu và trao đổi với những đồng nghiệp trong lúc nghiên cứu để có thể cùng hợp tác chung trong tương lai.

Đương nhiên quá trình này đòi hỏi mình sẽ phải nghĩ rất nhiều, hầu hết quá trình nghiên cứu của mình sẽ cứ chầm chậm và đột nhiên khám phá ra một hướng giải có thể ra kết quả và tiếp tục theo đuổi hướng đi đó để ra hướng giải cuối cùng. Và ít nhất 2 bài nghiên cứu mình đã viết xong đều theo xu hướng này cả, cụ thể hơn thì mình hầu hết sẽ tốn tầm vài tháng nhưng chỉ tốn 2 tuần gần cuối để ra hết ý tưởng cuối cùng của bài nghiên cứu.

Việc mình viết paper[108] thì mình tự nhận không phải người viết paper tốt nhất, tuy nhiên thầy Alex Psomas của mình viết cực kỳ hay luôn nên mình học cách viết paper của thầy. Khi viết mà nới ra vài chục trang thật ra cũng khá tự nhiên, những cái paper của mình hầu hết nội dung nhiều, mỗi cái chứng minh đã tốn nửa trang đến 1 trang rồi, và có một số chứng minh hoặc những định lý dài mình phải dùng đến những bổ đề khác thì phải kéo dài 5-6 trang. Tất nhiên rằng một bài nghiên cứu thì không chỉ giải một bài toán duy nhất mà mình phải giải rất nhiều khía cạnh khác nhau của bài toán. Chính vì thế có rất nhiều phần để xây dựng nên bài toán tổng quát. Việc viết được khoảng 20-30 trang liên tiếp thì nó cũng không quá lạ và đó không phải do phong cách viết của mình mà chỉ là hệ quả tự nhiên do nội dung mình thôi.

Bạn có thể chia sẻ qua 2 paper mà bạn đề cập trước đó không?

Một paper đã được xuất bản và một paper thì mình đang gửi đi, mình mong paper sẽ được đăng trên một hội nghị hay diễn đàn nào đó trước khi tạp chí này ra mắt. Còn paper kia thì đã được đăng ở hội nghị NeurIPS - một hội nghị về AI rất là lớn. Lý do paper mình được đăng tại hội nghị đó là do mình

[108] paper: bài nghiên cứu khoa học.

nghiên cứu về lý thuyết học máy, cụ thể là bài toán 'Mean Estimation': từ nhiều mẫu từ một phân phối ngẫu nhiên nào đấy, bạn sẽ tính toán lại giá trị trung bình hay giá trị kỳ vọng của phân phối ban đầu là bao nhiêu, bài toán này nghe rất là đơn giản đúng không! Một bài áp dụng lý thuyết xác suất thống kê cơ bản nhưng câu chuyện phức tạp hơn rất là nhiều, tính trung bình cộng không phải lúc nào cũng là cách tốt nhất, và có một thầy ở Purdue là thầy Paul Valiant có một chuỗi các bài nghiên cứu rất là nổi tiếng về 'Mean Estimation'. Về cơ bản thì thầy giải được cho cái trường hợp mà phân phối của bạn chỉ tạo ra 1 số duy nhất, trong trường hợp có 2 chiều thì nó sẽ khác. Cùng với thầy, mình đã chứng minh được bài toán này khó nhất là ở phân phối chuẩn. Điều này khá ngạc nhiên vì mọi người thường nghĩ phân phối chuẩn là một cái phân phối cực kỳ đẹp, nó thường là trường hợp tốt nhất (best case) cho các bài toán nghiên cứu. Để cho dễ tưởng tượng thì bài nghiên cứu của mình như chứng minh một bài toán có một thuật toán $O(n^2)$ và không thể làm bài toán này tốt hơn $O(n^2)$.

Cái paper thứ 2 của mình với thầy Alex Psomas là một bài toán khá là dễ thương. Khi đi mua điện thoại, cách để biết nó tốt hay không là nhìn giá, giá thì thường được định do nhà sản xuất hoặc thị trường đặt, nên về cơ bản thì giá chỉ là 'noisy sample'. Kiểu như một chiếc điện thoại có hiệu suất được đánh giá là 2 triệu đồng nhưng lại được bán là 2 triệu 500 nghìn đồng, tức nhiều tiền hơn so với hiệu năng thật của điện thoại. Lúc này, bạn sẽ tham khảo thị trường có nhiều điện thoại với những cái giá khác nhau, tất nhiên bạn sẽ muốn mua điện thoại có cái hiệu năng tốt nhất so với túi tiền hiện tại. Theo xu hướng sẽ mua điện thoại đắt nhất mà có thể trả được, nhưng giả sử, trước mắt bạn có 2 cái điện thoại với giá 15 triệu đồng và 14 triệu 999 nghìn đồng, lúc này bạn không thể mua điện thoại 15 triệu một cách ngẫu nhiên mà phải so sánh thêm. Vậy nên bài toán của mình sẽ cho giá tiền mà nhà sản xuất rao bán của những chiếc điện thoại này và kèm với những 'noisy sample' so với giá trị thật mà điện thoại đem lại và mình cần phải tìm một thuật toán mà cho ra chiếc điện thoại có giá tiền niêm yết sát với giá trị thật nhất mà đem lại hiệu năng cao nhất.

Mình cảm thấy bài toán này rất là hay mà vẫn chưa có paper nào nghiên cứu thật là kĩ về vấn đề này nên mình với thầy đã nghiên cứu. Trong bài toán này, mình cũng tìm được một thuật toán rất hay để giải quyết, mình có thể biểu diễn 'noisy value' bằng phương sai của số tiền niêm yết so với giá trị thật, lúc này mình sẽ không quan tâm các điện thoại có phương sai to và ở những ở điện thoại còn lại mình sẽ chỉ chọn những cái có giá tiền cao nhất vì đương nhiên những điện thoại có phương sai to sẽ không liên quan quá nhiều đến giá trị thật của chiếc điện thoại đem lại, chính vì thế những điện thoại có phương sai nhỏ khả năng sẽ phản ảnh chất lượng tốt nhất. Thuật toán này nghe có vẻ đúng nhưng mình không chỉ đề xuất ra nó mà còn chứng minh nó tốt luôn.

Con đường chứng minh thuật toán 'nghe nó đúng' đến 'đúng hẳn' là một con đường rất là dài.

Đây là nghiên cứu mình cảm thấy hãnh diện nhất bởi vì mình phải tốn một năm để ra được cái paper như thế. Hướng chứng minh bài toán này thật sự rất lằng nhằng tuy lúc miêu tả thuật toán thì lại rất tự nhiên.

Bạn có thể chia sẻ cách bạn duy trì sức khỏe, tinh thần cũng như sức bền để làm việc ở cường độ cao như vậy trong thời gian dài?

Đầu tiên để có sức bền thì mình nghĩ phải có đam mê trước đã. Điều quan trọng để duy trì công việc ở cường độ cao là phải có đam mê, giữ được đam mê đó bằng việc trải nghiệm nhiều khía cạnh khác nhau trong cuộc sống. Mình không chỉ dành cả ngày của mình với việc nghiên cứu mà mình cũng có một lối

sống sinh hoạt lành mạnh, như ngay hôm qua mình đăng trên Twitter là mình vừa đi một concert xong (cười).

Việc chuyển giữa các hoạt động khác nhau giúp cho mình không chán bất cứ hoạt động nào mà có phép mình thích tất cả mọi thứ cùng lúc được. Tất nhiên mình cũng không phải người đi concert quá nhiều, nhưng nếu mình đi 1-2 lần trong nhiều tháng thì mình cảm thấy tinh thần của mình tốt hơn. Bên cạnh đó mình còn duy trì tinh thần của mình qua việc đi chơi với bạn bè, rủ các bạn nấu ăn, có thể các bạn không biết chứ mình nấu ăn rất là ngon, đăng Twitter là lúc nào cũng nhiều like cực kì luôn (cười).

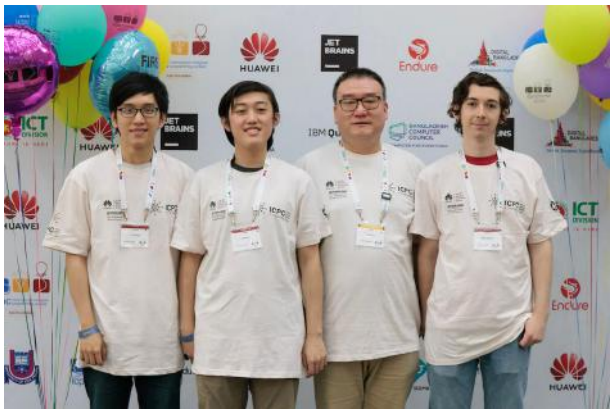
Nếu các bạn không biết thì Kuroni là người thường xuyên đăng ảnh đồ ăn lúc đêm muộn trên kênh chat Discord của VNOI. Đúng là một tội ác

Đúng rồi, thế nên việc mình làm nhiều thứ song song với việc nghiên cứu khiến mình không bao giờ thấy chán cả, mình không phải lúc nào cũng chỉ làm một việc duy nhất trong một thời điểm và mình liên tục thay đổi các hoạt động, thế nên mình thấy đấy cũng là bí kíp để mình không bị burnout trong việc nghiên cứu.

Bên cạnh đó mình còn duy trì việc tập thể dục, mình cảm thấy việc đi tập thể dục khá là quan trọng. Hồi xưa ba mẹ mình nói thì mình chả bao giờ nghe đâu, mãi khi mình chứng kiến trong quãng thời gian dịch Covid mình thê thảm như thế nào thì mình mới bắt đầu đi bộ, đánh cầu lông tuy mình cũng chả phải là người đánh cầu lồng hay lắm, nhưng sau khi chơi thì mình cảm thấy yêu đời hơn rất là nhiều.

Những phần Kuroni vừa chia sẻ là những điều khá là quan trọng, đã có rất nhiều người mất một thời gian khá dài để nhận ra những điều đấy. Theo mình thấy sau Covid thì khá nhiều bạn bị burnout nên mình mong sau những chia sẻ của bạn Kuroni thì các độc giả có thể thấy tầm quan trọng của việc nuôi dưỡng những sở thích khác nhau để dễ dàng hoán đổi, không bị burnout khi stress và đặc biệt hơn là thói quen duy trì tập thể dục

Để kết thúc phần chia sẻ, Kuroni có thể chia sẻ những dự định của mình trong năm 2024 được không?



Bên cạnh việc đạt HCV, làm 'Chú Tư' của thế giới một lần ở kỳ ICPC World Finals tới thì mình sẽ tiếp tục tổ chức các contest Codeforces tiếp và đương nhiên là lên LGM Codeforces. Mình không rõ sẽ cố gắng lên trước hay sau WF nhưng chắc chắn là trong khoảng nửa đầu của năm 2024. Về nghiên cứu thì mình mong muốn project hiện tại của mình sẽ thuận lợi và có thêm một paper nữa ở UT Austin. Thật ra bản thân mình cũng là một Swifty, một fan của Taylor Swift cũng đã lâu, phải cỡ 13-14 năm rồi nên mình cũng mong sẽ giật được một vé tour thì cũng khá là hay. Đợt rồi thì mình không may mắn có đủ tiền để đi Eras Tour nhưng mình mong muốn là sẽ được đi Taylor Swift tour một lần trong đời trước khi Taylor Swift giải nghệ. Ngoài ra thì mình muốn chơi hay hơn ở các môn thể thao, gặp nhiều bạn tốt, mở rộng mối quan hệ với không chỉ ở cộng đồng CP mà cả ở những cộng đồng khác.

Cảm ơn Kuroni đã có phần chia sẻ vô cùng tâm huyết, chúc bạn sớm đạt được những dự định của bản thân trong năm 2024!

Thách thức lập trình

...Thành công không phải là một đặc quyền và đặc lợi của riêng bất kì một ai. Tất cả chúng ta đều có thể thành công, nhưng không phải tất cả mọi người đều sẽ thành công, vì chỉ có những người thật sự nỗ lực và biết cách đầu tư vào bản thân mình mới có thể thật sự "hóa Rồng"...

Truyền thuyết kể rằng, khi trời đất mới hình thành, Ngọc Hoàng đã tạo ra mưa, gió, bão, sấm. Sau đó, do bận rộn với việc tạo ra con người, Ngọc Hoàng đã giao cho loài rồng - sinh vật quyền lực có nhiệm vụ bay lượn trên trời và phun nước để tạo ra mưa. Tuy nhiên, do số lượng rồng không đủ để tạo ra mưa cho mọi nơi, Ngọc Hoàng đã tổ chức cuộc "thi rồng" để chọn lựa các sinh vật trở thành rồng. Khi lệnh của Ngọc Hoàng được ban xuống, vua Thủy Tề đã thông báo cho tất cả các loài dưới nước tham gia. Những sinh vật nào đủ sức mạnh và tài năng, vượt qua thử thách sẽ biến thành Rồng.

Các sinh vật phải vượt qua một bài vô cùng hóc búa từ Ngọc Hoàng, bài toán được phát biểu như sau:

Thử thách Vũ Môn [109]

Cho một dãy số n số nguyên dương $a_1, a_2, \ldots, a_n$. Với mỗi số nguyên dương a_i, thí sinh cần đếm xem có bao nhiêu nghiệm nguyên của phương trình $x + y + \gcd(x, y) = a_i$

Các con vật khác đều bó tay chịu trận. Khi đến lượt Cá Chép tham gia, gió thổi mạnh và mây đen kéo đến. Cá Chép đã vượt qua bài thi và đi vào cửa Vũ Môn. Sau khi biến thành Rồng, Cá Chép phun nước tạo ra gió và mưa, giúp muôn loài thoát khỏi hạn hán và làm cho sự sống trở lại.

Kể từ ngày Cá Chép biến thành Rồng, rất nhiều cá chép khác trên sông đã tụ tập và bơi ngược dòng đến dưới Vũ Môn. Dù dòng nước có hung dữ đến mấy, luôn có vô số con cá chép dũng cảm nhảy lên, và dù có rơi xuống đến mức trầy da tróc vảy, chúng cũng không từ bỏ.

Tuy nhiên, sau nhiều năm trôi qua, không một con cá chép nào có thể chạm đến Vũ Môn. Chúng rất thất vọng và đã kéo nhau đến gặp vua Thủy Tề, xin vua hãy hạ Vũ Môn xuống thấp hơn, bởi nếu không, sẽ không có con cá nào có thể trở thành rồng.

[109] https://oj.vnoi.info/problem/dovui_2024_a

Sau một cuộc tranh luận làm rung chuyển cung điện dưới nước, cuối cùng vua Thủy Tề cũng đồng ý hạ Vũ Môn xuống thấp hơn để tất cả cá chép đều có thể dễ dàng nhảy qua. Tuy nhiên, vua Thuỷ Tề đặt điều kiện là tất cả con cá chép đều phải đưa ra đáp án chính xác của bài toán sau, thì mới hạ thấp Vũ Môn:

Lật sỏi [110]

Trò chơi ô ăn quan truyền thống là một trò chơi truyền thống vô cùng quen thuộc với trẻ em Việt Nam. Tuy nhiên chúng ta sẽ đi đến phiên bản khác của trò chơi nhưng cũng cực trí tuệ: trò chơi lật sỏi.

Sân chơi gồm n vị trí từ vị trí 0 đến $n-1$. Mỗi hiệu lệnh được đưa ra, bạn sẽ nhận được ba số nguyên t, A, B. Với mọi $A \leq i \leq B$, nếu $t = 0$ thì ta đổi trạng thái của vị trí thứ i. Nếu $t = 1$, ta cần đếm xem có bao nhiêu vị trí i đang có sỏi. Hãy nhanh tay trả lời các hiệu lệnh loại 1 nhé!

Bài toán đã không làm khó được sự quyết tâm của các con cá chép, tất cả sau đó đều hóa thành những con rồng. Ban đầu, đàn cá chép rất hân hoan và vui mừng vì cuối cùng chúng đã đạt được ước nguyện - đó là một kỳ tích mà chỉ có thể xuất hiện một lần sau hàng trăm nghìn năm.

Nhưng sau một thời gian, chúng nhìn nhau và tự hỏi: Rốt cuộc thì làm rồng và làm cá chép có gì khác nhau? Tất nhiên không con nào có thể trả lời được vì tất cả chúng đều giống hệt nhau. Thế là đàn cá rồng lại cùng nhau gặp vua Thủy Tề để than phiền, rằng chúng phải vất vả vượt qua Vũ Môn để biến thành rồng, nhưng khi đã trở thành rồng, lại không có gì thú vị hơn so với khi làm cá chép.

Vua Thủy Tề cười lớn và nói rằng: "Thực ra, trong số các ngươi, chưa có ai trở thành rồng cả. Vũ Môn mà các ngươi dễ dàng nhảy qua thực ra là giả. Ta thấy các ngươi đáng lẽ phải nỗ lực một phen gian khó, nâng cao tiêu chuẩn bản thân... nhưng các ngươi không chỉ không cố gắng mà còn đến gặp ta để than phiền. Vì vậy, ta đã che đi Vũ Môn thật và dựng lên Vũ Môn giả để các ngươi thỏa nguyện.."

Tiếp lời, vua Thủy Tề nói: "Nếu tất cả cá chép đều dễ dàng trở thành rồng thì 'rồng' cuối cùng cũng chỉ là một tên gọi khác của loài cá chép mà thôi. Nếu các ngươi muốn biết rồng thật sự khác cá chép thế nào, thì chỉ có một cách duy nhất là bằng mọi giá vượt qua Long Môn thật. Khi đó, ai chỉ là cá và ai là Rồng thì các ngươi sẽ phân biệt ra ngay. Nào hãy cùng quyết tâm, luyện tập thật nhiều để chinh phục Vũ Môn thật. Ta có một thử thách thú vị dành cho các ngươi.."

Xông nhà [111]

Mỗi dịp Tết đến xuân về, ai cũng náo nức nô đón chờ những điều may mắn, những điều tốt đẹp sẽ đến với cuộc sống. Tập tục xông nhà là một phong tục nhằm đón những điều tốt lành vào đầu năm mới, từ lâu đây đã là một truyền thống văn hoá lâu đời của người Việt. Để đón nhận những điều tốt đẹp trong năm mới, ta cùng xem qua dãy số như sau:

Với một dãy k số nguyên $[B_0, \ldots, B_{k-1}]$ được coi là 'đẹp' nếu như tồn tại hoán vị $[C_0, \ldots, C_{k-1}]$ sao cho prefix xor của dãy C tăng dần. Nói cách khác, gọi $P_i = P_{i-1} \oplus C_i$. Ta cần có $P_0 < P_1 < \ldots < P_{k-1}$.

Mỗi ngôi nhà sẽ có một dãy số A gồm n số nguyên đại diện. Năm mới bạn muốn xông nhà cho người thân quen của mình, với mỗi ngôi nhà của gia chủ, hãy tìm xem mọi prefix của A có phải dãy đẹp không nhé. Nếu mọi prefix A đều là dãy đẹp, bạn vô cùng hợp tuổi với gia chủ đấy.

Vì là một thử thách vô cùng khó nên cuối cùng chỉ có số ít cá chép vượt qua và hóa thành những chú rồng thực thụ và đầy quyền năng. Từ câu chuyện, ta thấy tất cả cá chép đều có thể biến thành rồng nhưng không phải con nào cũng sẽ vượt qua được Vũ Môn. Tương tự như vậy, thành công dành cho tất cả mọi người. Thành công không phải là một đặc quyền và đặc lợi của riêng bất kì một ai. Tất cả chúng ta đều có thể thành công, nhưng không phải tất cả mọi người đều sẽ thành công, vì chỉ có những người thật sự nỗ lực và biết cách đầu tư vào bản thân mình mới có thể thật sự "hóa Rồng".

[110] https://oj.vnoi.info/problem/dovui_2024_b
[111] https://oj.vnoi.info/problem/dovui_2024_c

Thể lệ

- Trong năm vừa qua, ắt hẳn một số bạn độc giả đã quen với dự án **Problem of the Week** của team Bedao. Dự án đã để lại nhiều dấu ấn cho cộng đồng khi hàng tuần đem đến những bài toán hóc búa, thách thức độc giả gửi về những lời giải xuất sắc về cho VNOI. Giờ đây dự án một lần nữa quay trở lại với tên 'Thách thức lập trình' trong số tạp chí VNOI Xuân Giáp Thìn.
- Ở số Thách thức lập trình [112] này sẽ có 3 bài toán được đặt ra, với mỗi bài giải được các bạn hãy viết lời giải (editorial) và bài giải (code), gửi về cho chúng mình thông qua Thách thức lập trình - Tạp chí VNOI Giáp Thìn 2024 [113] . Vào cuối thử thách, ban biên tập sẽ chọn và công bố 3 lời giải xuất sắc nhất của 3 bài toán trên VNOI [114] .
- **Bài giải hợp lệ:** các bài giải khi gửi về cho VNOI phải là những bài đã qua hết bộ test đã được soạn sẵn và không có hành vi chép bài giải, lời giải để nộp. Mọi bài giải gửi về vi phạm sẽ không được xét giải thưởng.
- **Thời gian nhận bài:** từ 10/2 đến 20/2, năm 2024.
- **Giải thưởng:** Ứng với mỗi lời giải được chọn, các bạn sẽ được một bộ merchandise đặc biệt gồm: *10 bao lì xì + móc khóa + dây đeo + sticker VNOI*. Chính vì thế hãy thật nhanh tay để giành lấy cho mình bộ quà có 1-0-2 dành riêng cho dịp Xuân Giáp Thìn này nhé!

[112] https://oj.vnoi.info/contest/thachthuc_laptrinh_giapthin2024
[113] https://forms.gle/tXfDemgzZFiHjR3k6
[114] https://www.facebook.com/vnoi.wiki

Ban biên tập

Nguyễn Trung Quân (Tổng biên tập)

Nguyễn Tấn Minh (Phó Tổng biên tập)

Nguyễn Đăng Quân

Võ Nguyễn Phương Quỳnh

Phan Vĩnh Tiến

Thiết kế

Đoàn Viết Tiến Đạt

Trần Thị Thanh Vân

Cố vấn biên tập

Phạm Thị Diễm Quỳnh

Vương Hoàng Long

Phạm Xuân Trung

Trần Quang Lộc

Hoàng Quốc Việt

Cộng tác viên

Nguyễn Quang Trường (hỗ trợ phỏng vấn OLP-ICPC)

Nguyễn Minh Hiển (tác giả bài viết)

Nguyễn Minh Thiện (tác giả bài viết)

Dương Hoàng Long (tác giả bài viết)

HỘI TIN HỌC VIỆT NAM - VAIP
CÂU LẠC BỘ OLYMPIC TIN HỌC VIỆT NAM - VNOI

TẠP CHÍ VNOI XUÂN GIÁP THÌN 2024

XUẤT BẢN NĂM 2024