Đại học Quốc gia Thành phố Hồ Chí Minh
Trường Đại học Công nghệ Thông tin

BÁO CÁO LẬP TRÌNH THI ĐẤU

Cấu trúc dữ liệu và giải thuật - IT003.M21.KHTN

Họ và tên: Huỳnh Đặng Vĩnh Hiền
Lớp: KHTN2021
MSSV: 21520029

Ngày 1 tháng 6 năm 2022

1 Tổng quan

Báo cáo này được thực hiện như là một đồ án cuối kỳ cho môn Cấu trúc dữ liệu và giải thuật, thuộc chương trình đào tạo Cử nhân tài năng ngành Khoa học Máy tính, Trường Đại học Công nghệ Thông tin, ĐHQG-HCM.

Bài báo này tập trung vào lĩnh vực Lập trình thi đấu, một lĩnh vực khá phổ biến và đặc sắc trong Khoa học Máy tính nói riêng và Toán - Tin học nói chung.

Báo cáo nhằm đưa ra phương án tiếp nhận và xử lý các bài toán lập trình một cách rõ nét, chi tiết thông qua các ví dụ cụ thể từ các cuộc thi lập trình thường niên, nổi tiếng trên thế giới.

Bài báo cáo gồm hai phần, trình bày chính về hai cuộc thi lập trình thuật toán bao gồm:

- Google Code Jam 2022, tập trung vào hai vòng là Round 1C và Round 2.
- Codeforces Round #795 (CodeCraft-22).

Để tham khảo báo cáo này, người đọc cần các kiến thức cơ bản về cấu trúc dữ liệu và giải thuật, đồng thời cần một số kiến thức khác như độ phức tạp tính toán và lập trình C++ nâng cao. Báo cáo này có thể dùng như một tài liệu tham khảo trong rèn luyện kỹ năng lập trình thi đấu cho các bạn ở mức độ trung bình - khá.

Các mã nguồn của các bài tập được nhắc đến trong báo cáo và các thông tin có liên quan có thể được truy cập tại `https://github.com/vinhhien323/`.

2 Google Code Jam 2022

2.1 Giới thiệu

Google Code Jam là một kỳ thi lập trình thường niên do Google tổ chức, bắt đầu từ năm 2008. Hầu hết các vòng thi đều được tổ chức trực tuyến, trừ vòng cuối cùng thi tập trung tại địa điểm do Ban tổ chức sắp đặt. Sau đây là cấu trúc của kỳ thi phiên bản năm 2022.

Google Code Jam năm 2022 bao gồm năm vòng thi chính:

1. Qualification Round, vòng loại của cuộc thi và không giới hạn số lượng thí sinh tham gia. Ở vòng này, luật của kỳ thi được nới lỏng, các thí sinh có quyền trao đổi với các thí sinh khác. Để vượt qua được vòng này, mỗi thí sinh cần đạt ít nhất 30/100 điểm.

2. Round 1, vòng chính thức đầu tiên của cuộc thi, được chia làm ba vòng nhỏ mang tên Round 1A, Round 1B và Round 1C. Ở mỗi vòng sẽ lấy 1500 thí sinh có thành tích tốt nhất để tham dự vòng tiếp theo.

3. Round 2, gồm 4500 thí sinh tham dự nhằm chọn ra 1000 thí sinh xuất sắc nhất vào vòng kế tiếp. Ngoài ra, các thí sinh này sẽ được tặng một chiếc áo thun đặc biệt từ Google.

4. Round 3, gồm 1000 thí sinh tham dự, chọn ra 25 thí sinh đặc biệt xuất sắc tham dự vòng thi trực tiếp cuối cùng.

5. World Finals, vòng thi danh giá nhất của cuộc thi. Vòng thi này được tổ chức trực tiếp, thường là ở văn phòng của Google tại các nước phát triển, toàn bộ chi phí tham gia cuộc thi do ban tổ chức đài thọ.

Cấu trúc của mỗi vòng thi bao gồm 3 hoặc 4 câu với tổng số điểm là 100, diễn ra trong vòng 150 phút (trừ Qualification Round diễn ra trong vòng 27 giờ). Mỗi bài được chia ra thành nhiều subtask, thí sinh phải giải trọn một subtask thì mới có điểm subtask đó. Đối với các thí sinh có tổng điểm bằng nhau, thời gian nộp bài sẽ được dùng để phân định thứ hạng.

2.2 Thành tích của tác giả

Username tham dự kỳ thi: **hdvhien**
Thành tích các vòng thi:

1. Qualification Round: 44 điểm, xếp hạng 11977/32702.

2. Round 1C: 34 điểm, xếp hạng 1030/6450.

3. Round 2: 12 điểm, xếp hạng 2472/3400.

2.3 Google Code Jam 2022, Round 1C

2.3.1 Câu A: Letter Blocks

Số lần nộp: 1
Số điểm đạt được: 25/25
Tóm tắt bài toán: Bạn được cho n chuỗi ký tự s_i, mỗi chuỗi chỉ bao gồm các ký tự tiếng Anh in hoa. Ta cần tìm một hoán vị bậc n, $[p_1, p_2, \ldots, p_n]$, đặt $s = s_{p_1} + s_{p_2} + \cdots + s_{p_n}$ sao cho điều kiện sau được thỏa mãn: Với mọi chữ cái nằm trong bảng chữ cái tiếng Anh in hoa, các lần xuất hiện của chúng trong s là liên tiếp nhau (1).
Ví dụ, các xâu $abbbc$, $abcd$ là thỏa mãn (1), trong khi aba, $abab$ là không thỏa mãn (1).
Đề bài yêu cầu xuất ra một xâu s bất kỳ thỏa yêu cầu trên.
Giới hạn:
$1 \leq |s_i| \leq 10$

- (10 điểm): $2 \leq n \leq 6$
- (15 điểm): $2 \leq n \leq 100$

Ví dụ:

Sample Input	Sample Output
5 CODE JAM MIC EEL ZZZZZ	ZZZZZJAMMICCODEEEL

Lời giải:
Ta có nhận xét sau:
Xét một chữ cái tiếng Anh in hoa c bất kỳ:
Nếu tồn tại hai xâu s_i và s_j với $i \neq j$ sao cho s_i và s_j đều chỉ chứa các ký tự c, s_i sẽ nằm kế cạnh s_j trong s. Do đó, ta sẽ tiến hành gộp s_i và s_j thành một xâu mới có độ dài $|s_i| + |s_j|$ chứa toàn ký tự c.
$\Rightarrow$ **Bước 1:** Với mọi c, ta tiến hành gộp các xâu chỉ chứa ký tự c thành một xâu duy nhất.
Sau khi tiến hành bước 1, với mỗi ký tự c, ta xét các xâu thuộc những loại sau đây:

1. Xâu chỉ chứa toàn ký tự c.
2. Xâu kết thúc bằng ký tự c nhưng không thuộc loại 1.
3. Xâu bắt đầu bằng ký tự c nhưng không thuộc loại 1.

Ta nhận xét nếu số lượng xâu loại 2 hoặc số lượng xâu loại 3 lớn hơn 1, bài toán sẽ trở nên vô nghiệm, do không tồn tại cách nối những xâu này lại thỏa mãn yêu cầu đề bài, do đó ta tiến hành kết thúc bài toán. Mặc khác, sau bước 1, số lượng xâu loại 1 sẽ không vượt quá 1, do đó ta có cách nối sau:

$t' = s_i + s_j + s_k$ (2)

Trong đó s_i là một xâu loại 1, s_j là một xâu loại 2 và s_k là một xâu loại 3.

Nếu ở mỗi loại không có xâu nào, ta bỏ qua xâu đó và vẫn cộng theo thứ tự (2).

⇒ **Bước 2:** Với mỗi ký tự c, ta tiến hành kiểm tra điều kiện để tạo xâu (2) có khả thi hay không. Nếu có, ta tiến hành tìm các xâu s_i, s_j, s_k, loại bỏ chúng khỏi danh sách xâu và thêm vào danh sách xâu t'. Nếu không, ta ngừng thuật toán tại đây.

Sau khi thực hiện bước 2, danh sách các xâu lúc này đang rời nhau, tuy nhiên, do đã hết điều kiện ràng buộc, ta nối các xâu này lại theo thứ tự tùy ý, tạo thành xâu s.

Tuy nhiên, ta cần kiểm tra xem các ký tự giống nhau trong s có tạo thành một dãy liên tiếp nhau hay không, điều này có thể được thực hiện dễ dàng thông qua một vòng lặp cơ bản và đánh dấu mảng.

⇒ **Bước 3:** Nối các chuỗi rời rạc trong danh sách lại thành xâu s, sau đó hậu kiểm điều kiện ràng buộc của đề bài đối với xâu s.

Vậy sau 3 bước, ta đã kiểm tra có tồn tại một cấu hình thứ tự thỏa yêu cầu đề bài hay không, và thu được xâu s nếu tồn tại một cấu hình hợp lệ.

Độ phức tạp thời gian của thuật toán: $\mathcal{O}(n^2|s_i|)$

2.3.2 Câu B: Squary

Số lần nộp: 1

Số điểm đạt được: 9/31

Tóm tắt bài toán: Cho dãy số nguyên a_i gồm n phần tử và số nguyên dương k. Hãy tìm dãy số nguyên b_j gồm m phần tử $(1 \leq m \leq k)$ sao cho:

$$\left(\sum_{i=1}^{n} a_i + \sum_{j=1}^{m} b_j\right)^2 = \sum_{i=1}^{n} a_i^2 + \sum_{j=1}^{m} b_j^2$$

Đề bài yêu cầu chỉ ra một dãy b_j thỏa điều kiện của bài toán, đồng thời yêu cầu $-10^{18} \leq b_j \leq 10^{18}$

Giới hạn:

$1 \leq n \leq 10^3$

$-10^3 \leq a_i \leq 10^3$

- (9 điểm): $k = 1$
- (22 điểm): $2 \leq k \leq 10^3$

Ví dụ:

Sample Input	Sample Output
2 1 -2 6	3

Lời giải:
Xét trường hợp $k = 1$:
Đặt $s = \sum_{i=1}^{n} a_i, S = \sum_{i=1}^{n} a_i^2, p = \sum_{1 \le i < j \le n} a_i a_j$.
Gọi số nguyên duy nhất ta sẽ thêm vào là b, ta có:
$(s + b)^2 = S + b^2$
$\Rightarrow s^2 + 2sb + b^2 = S + b^2$
$\Rightarrow sb = \dfrac{S - s^2}{2}$
Bài toán phức tạp trở thành bài toán giải phương trình bậc nhất một ẩn với các trường hợp sau:

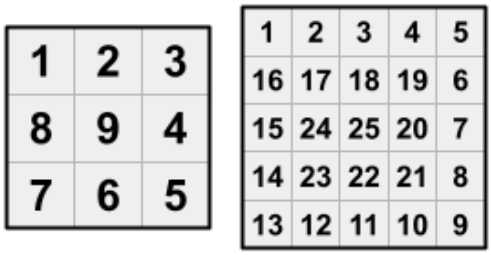
1. Khi $s = 0$ và $\dfrac{S - s^2}{2} \neq 0$, bài toán vô nghiệm.
2. Khi $s = 0$ và $\dfrac{S - s^2}{2} = 0$, bài toán có vô số nghiệm, ta chọn b tùy ý.
3. Khi $s \neq 0$, bài toán có nghiệm duy nhất là $b = \dfrac{S - s^2}{2s}$.

Độ phức tạp thời gian của thuật toán: $\mathcal{O}(n^2)$, tuy nhiên, nếu để ý kỹ, ta sẽ thấy $S + 2p = s^2$, do đó có thể tính p trong $\mathcal{O}(n)$, từ đó giảm độ phức tạp thời gian còn $\mathcal{O}(n)$.
Xét trường hợp $k \ge 2$:
Bài toán đã được chứng minh với trường hợp $k \ge 2$, bài toán luôn có nghiệm và khi đó chỉ cần tạo dãy b có đúng 2 phần tử.
Chi tiết về cách chứng minh và lời giải có thể được tham khảo thêm tại `https://codeforces.com/blog/entry/102385#comment-907850`.

2.4 Google Code Jam 2022, Round 2

2.4.1 Câu A: Spiraling Into Control

Số lần nộp: 1
Số điểm đạt được: 7/20
Tóm tắt bài toán: Cho số nguyên dương n lẻ và số nguyên không âm k, ta dựng một ma trận hình vuông kích thước $n \times n$, bao gồm các số từ 1 đến n^2 theo hình xoắn ốc.
Ví dụ với $n = 3$ hay $n = 5$, ta có ma trận như sau:



1	2	3
8	9	4
7	6	5

1	2	3	4	5
16	17	18	19	6
15	24	25	20	7
14	23	22	21	8
13	12	11	10	9

Ở ô mang giá trị x, ta chỉ có một cách đi đến ô kề cạnh mà mang giá trị $x+1$. Ví dụ: nếu ta đang đứng ở ô mang giá trị 5, bước kế ta phải di chuyển đến ô mang giá trị 6. Dễ thấy ta sẽ tốn n^2-1 bước để di chuyển từ ô mang giá trị 1 đến ô mang giá trị n^2.

Tuy nhiên, bài toán cho phép chúng ta đi theo các đường tắt. Đường tắt được định nghĩa là đường ta di chuyển từ ô mang giá trị x đến ô kề cạnh mang giá trị y sao cho $y > x+1$. Ví dụ, xét $n=5$ và ta đang ở ô mang giá trị 14, đường tắt sẽ dẫn ta đến ô mang giá trị 23.

Bài toán yêu cầu liệt kê các đường tắt cần được sử dụng để đảm bảo ta có thể di chuyển từ ô mang giá trị 1 đến ô mang giá trị n^2 trong đúng chính xác k bước.

Giới hạn:

$1 \leq k < n^2 - 1$

$n \equiv 1 \mod 2$

- (3 điểm): $n \leq 9$
- (4 điểm): $n \leq 39$
- (13 điểm): $n \leq 9999$

Ví dụ:

Sample Input	Sample Output
5 4	2 2 17 18 25

Lời giải:

Xét trường hợp $n \leq 39$

Ta sẽ sử dụng các vòng for để xây dựng ma trận bảng theo thiết kế của đề bài, đồng thời ta xây dựng mảng $nxt[i]$ cho biết vị trí kế tiếp nếu ta sử dụng đường tắt ở ô mang giá trị i.

Ta có số trạng thái là $n^2.k$, trong trường hợp tệ nhất là xấp xỉ $n^4 \Rightarrow$ Số lượng trạng thái đủ nhỏ để ta có thể quét hết tất cả các trạng thái.

Gọi $dp[i][k]$ là biến true/false cho biết từ ô mang giá trị 1, ta có thể di duyển đến ô mang giá trị i sau đúng k bước hay không. Bài toán có nghiệm khi $dp[n^2][k]$ mang giá trị true. Khi đó ta tiến hành truy vét ngược để xuất ra cấu hình kết quả.

Độ phức tạp thời gian của thuật toán: $\mathcal{O}(n^4)$.

2.4.2 Câu B: Pixelated Circle

Số lần nộp: 1
Số điểm đạt được: 5/21
Tóm tắt bài toán: Bạn được cho một số nguyên dương R, đi kèm với ma trận vuông có kích thước mỗi cạnh là $2R - 1$. Các hàng và cột được đánh số thứ tự từ $-R$ đến R, chứa các giá trị boolean. Bài toán yêu cầu hiện thực hóa mã giả của thuật toán vẽ hình/đường tròn được cung cấp sẵn trong đề bài. Biết rằng các pixel ban đầu đều có màu trắng, nếu tồn tại ít nhất một nét vẽ đi qua điểm (x, y), pixel ở ô đó sẽ mang màu đen,

Bài toán yêu cầu tính số lượng ô đen chênh lệch trong 2 cách vẽ khác nhau.

Để biết rõ hơn về mã giả cũng như bộ kiểm tra mẫu của bài toán, vui lòng xem thêm tại `https://codingcompetitions.withgoogle.com/codejam/round/00000000008778ec/0000000000b158f7`

Giới hạn:

- (5 điểm): $R \leq 100$
- (16 điểm): $R \leq 10^5$

Lời giải:
Xét $R \leq 100$:

Ta hiện thực 2 thuật toán vẽ hình được đề cập đến trong bài toán rồi đi so sánh chênh lệch số phần tử được tô màu giữa 2 thuật toán đó.

Tuy nhiên, do tọa độ khi vẽ có thể mang giá trị âm, gây khó khăn trong khâu xử lý, do đó, phép tịnh tiến được sử dụng để đơn giản hóa bài toán, chuyển toàn bộ tọa độ sang hệ tọa độ không âm.

Độ phức tạp thời gian của thuật toán: $\mathcal{O}(R^3)$.

3 Codeforces Round #795 (CodeCraft-22)

3.1 Tổng quan

Codeforces là một trang nổi tiếng chuyên về rèn luyện và tranh tài ở bộ môn lập trình thi đấu. Trang có xuất xứ từ Liên bang Nga, được sử dụng đông đảo bởi nhiều lập trình viên trên khắp thế giới. Codeforces nổi tiếng với nội dung đa dạng, nguồn đề có chất lượng cao cũng như tính thường xuyên trong việc tổ chức các cuộc thi.

Codeforces Round #795 là một vòng thi thường niên trên Codeforces, thường được tổ chức hàng tuần. Vòng thi này được đánh giá có độ phức tạp thuộc Division 2, đồng thời được chọn làm CodeCraft-22, một cuộc thi lập trình thường niên do Ấn Độ tổ chức.

Đề thi lần này gồm 6 câu với thời gian 120 phút.

Thang điểm cho các bài lần lượt là 500 - 750 - 1250 - 1750 - 2250 - 2750. Điểm của mỗi bài sẽ giảm dần theo thời gian. Với mỗi lần nộp bài không thành công hoặc nộp lại bài, người thi bị trừ thêm 50 điểm.

3.2 Thành tích của tác giả

Tài khoản dự thi trên Codeforces: **Hien**

Tổng số bài làm được: 4

Tổng số điểm nhận được: 4250 (không penalties) và 3205 (có penalties).

3.3 Thông tin về các bài

3.3.1 Bài A: Beat The Odds

Số lần nộp: 1

Số điểm đạt được: 498/500

Tóm tắt bài toán: Cho dãy số nguyên dương a có n phần tử, phần tử thứ i mang giá trị a_i. Ở mỗi thao tác, ta có thể chọn một số trong dãy và xóa nó đi, các số còn lại sẽ được nối lại với nhau. Hỏi cần sử dụng ít nhất bao nhiêu thao tác để trong dãy số còn lại, tổng giữa hai số liên tiếp bất kỳ luôn là một số chẵn.

Giới hạn:

$3 \leq n \leq 10^5$

$1 \leq a_i \leq 10^9$

Ví dụ:

Sample Input	Sample Output
5 2 4 3 6 8	1

Lời giải:

Dễ thấy để tổng giữa hai số bất kỳ trong dãy là một số chẵn, thì toàn bộ các số trong dãy phải có cùng tính chẵn lẻ. Gọi cnt là số lượng số lẻ trong dãy. Ta có đáp án cho bài toán là $\min(cnt, n-cnt)$.

Độ phức tạp thời gian của thuật toán: $\mathcal{O}(n^2)$.

3.3.2 Bài B: Shoe Shuffling

Số lần nộp: 1

Số điểm đạt được: 723/750

Tóm tắt bài toán: Trong lớp học có n bạn học sinh, bạn học sinh thứ i có kích thước giày là s_i. Ban đầu, mỗi bạn sẽ mang đôi giày có kích thước đúng với cỡ giày của bản thân.

Các bạn tiến hành trao đổi giày cho nhau, bạn thứ u có thể mang vừa giày của bạn v khi và chỉ khi $s_u \leq s_v$. Hỏi có tồn tại một cấu hình trao đổi giày sao cho không bạn nào mang lại đôi giày của chính mình và các bạn đều có một đôi giày để mang vừa hay không.

Phát biểu lại bài toán về mặt toán học, ta cần tìm một hoán vị bậc n $[p_1, p_2, \ldots, p_n]$ sao cho $i \neq p_i$ và $s_i \leq s_{p_i}$.

Giới hạn:

$1 \leq n \leq 10^5$

$1 \leq s_i \leq 10^9$

Ví dụ:

Sample Input	Sample Output
5 1 1 1 1 1	5 1 2 3 4

Lời giải:

Giả sử ta có một cấu hình hoán đổi giày liên hoàn $[a_1, a_2, \ldots, a_k]$, tức a_i mang giày của a_{i-1} với $2 \leq i \leq k$, a_1 mang giày của a_k thì điều kiện sau cần được thực hiện:

$$\begin{cases} s_{a_2} \geq s_{a_1} \\ s_{a_3} \geq s_{a_2} \\ s_{a_k} \geq s_{a_{k-1}} \\ s_{a_1} \geq s_{a_k} \end{cases}$$

Ta dễ thấy điều kiện này được thực thi khi và chỉ khi $s_{a_1} = s_{a_2} = \cdots = s_{a_k}$.

$\Rightarrow$ Ta tiến hành gộp các vị trí i có chung giá trị s_i về một nhóm. Nếu nhóm này có đúng 1 phần tử, bài toán vô nghiệm. Ngược lại, ta tiến hành hoán đổi liên hoàn như ví dụ nêu trên để xây dựng cấu hình hoán đổi giày.

Độ phức tạp thời gian của thuật toán: $\mathcal{O}(n \log(n))$ do sử dụng hàm sort.

3.3.3 Bài C: Sum of Substrings

Số lần nộp: 1
Số điểm đạt được: 1075/1250
Tóm tắt bài toán: Bạn được cung cấp một xâu nhị phân s có độ dài n. Ta gọi $f(i)$ là giá trị thập phân của số i (trong điều kiện i tạo thành một số nguyên không âm hợp lệ). Đặt $s = \sum_{i=2}^{n} f(s_{i-1}s_i)$

Bạn được cung cấp k lượt đi, ở mỗi lượt đi bạn có thể tráo đổi vị trí giữa hai phần tử liên tiếp bất kỳ trong s. Hỏi dãy sử dụng k lượt này một cách khôn ngoan sao cho s sau khi thực hiện hoạt động này là nhỏ nhất.
Giới hạn:
$1 \leq n \leq 10^5$
$0 \leq k \leq 10^9$
$s_i \in [0, 1]$
Ví dụ:

Sample Input	Sample Output
7 1 0010100	22

Lời giải:
Xét các bit mang giá trị 1 trong s, ta có:

1. s_i mang trọng số là 10 nếu $i = 1$.
2. s_i mang trọng số là 1 nếu $i = n$.
3. s_i mang trọng số là 11 cho các trường hợp còn lại.

Giải thuật tham lam: Nếu có thể, thực hiện các thủ tục sau đây:

1. Di chuyển phần tử ở vị trí (3) về vị trí (2).
2. Di chuyển phần tử ở vị trí (1) về vị trí (2).
3. Di chuyển phần tử ở vị trí (3) về vị trí (1).

Độ phức tạp thời gian của thuật toán: $\mathcal{O}(n)$

3.3.4 Câu D: Max GEQ Sum

Số lần nộp: 2
Số điểm đạt được: 909/1750
Tóm tắt bài toán: Cho dãy số nguyên a có n phần tử. Hãy kiểm tra xem bất đẳng thức:

$$\max(a_i, a_{i+1}, \ldots, a_j) \geq a_i + a_{i+1} + \cdots + a_j$$

có được thỏa mãn với mọi $1 \leq i \leq j \leq n$ hay không?
Giới hạn:
$1 \leq n \leq 2 \times 10^5$
$-10^9 \leq k \leq 10^9$
Ví dụ:

Sample Input	Sample Output
4 -1 1 -1 2	YES

Lời giải:
Ta xét 3 vị trí l, r, i, sao cho $1 \leq l \leq i \leq r \leq n$ và $\max(a_l, a_{l+1}, \ldots, a_r) = a_i$.
Từ bất đẳng thức đề bài cho, ta có:
$\max(a_l, a_{l+1}, \ldots, a_r) \geq a_l + a_{l+1} + \cdots + a_r$
$\Rightarrow a_i \geq a_l + a_{l+1} + \cdots + a_r$
$\Rightarrow sum[l:r] - a_i \leq 0$
Để bất đẳng thức sai thì ta cần tìm l và r sao cho $sum[l:r] - a[i] > 0$,
Hay ta cần tìm $\max(sum[l:r]) - a_i > 0$

Thuật toán: Ta sẽ dùng kỹ thuật chuỗi đơn điệu (hoặc tìm kiếm nhị phân + cấu trúc dữ liệu đặc biệt) để với mỗi i, ta xác định được giá trị $L[i]$ nhỏ nhất, giá trị $R[i]$ lớn nhất sao cho $\max(a_{L[i]}, a_{L[i]+1}, \ldots, a_{R[i]}) = a[i]$

Từ đó, ta có thể sử dụng các cấu trúc dữ liệu đặc biệt để với mỗi i, ta tìm $\max(sum[l:r])$ với $L[i] \leq l \leq i \leq r \leq R[i]$.

Cấu trúc dữ liệu đặc biệt ở đây bao gồm segment tree, spare table và square root division.

Độ phức tạp thời gian của thuật toán: $\mathcal{O}(n \log(n))$, $\mathcal{O}(n \log^2(n))$, $\mathcal{O}(n\sqrt{n})$ tùy thuộc vào cách cài đặt.

4 Kết luận

Các bài tập trích từ các kỳ thi được nhắc đến trải dài từ dễ đến khó, phù hợp với nhiều đối tượng và mục đích khác nhau.

Việc quy đổi độ khó giữa các kỳ thi là rất khó thực hiện và chỉ mang tính chất tương đối, do đó chỉ nên được dùng để tham khảo.

Ngoài việc chú trọng để tư duy toán và lập trình, cần có tâm lý tốt cộng với sự cẩn thận, tỷ mỹ để có được hiệu quả cao.