DỰ ÁN THE GIFTED BATTLEFIELD
BAN CHUYÊN MÔN TIN HỌC

KÌ THI THỬ TUYỂN SINH LỚP 10 THPT CHUYÊN
ĐỢT 2 – NĂM HỌC 2023 - 2024

HƯỚNG DẪN CHẤM THI
Đề thi thử đợt 2

Môn thi: **TIN HỌC (chuyên)**
Ngày thi: **09/04/2023 – 16/04/2023**
Thời gian làm bài: **150 phút** (không kể thời gian phát đề)
Hướng dẫn chấm thi gồm 05 trang

Tổng quan đề thi

	Tên bài	File chương trình	File dữ liệu	File kết quả
Bài 1	Kí tự gần nhất	`NEARCHAR.*`	`NEARCHAR.INP`	`NEARCHAR.OUT`
Bài 2	Dãy ước	`DAYUOC.*`	`DAYUOC.INP`	`DAYUOC.OUT`
Bài 3	Đếm dãy	`DEMDAY.*`	`DEMDAY.INP`	`DEMDAY.OUT`
Bài 4	Waifu	`WAIFU.*`	`WAIFU.INP`	`WAIFU.OUT`

Dấu `*` được thay thế bởi `PAS` hoặc `CPP` của ngôn ngữ lập trình sử dụng tương ứng là `Pascal` hoặc `C++`.

I. Hướng dẫn chung

1. Bài thi của thí sinh được chấm thi bằng phần mềm chấm thi trực tuyến (online judge) trên website https://oj.giftedbat.edu.vn/, sử dụng bộ test của Dự án The Gifted Battlefield – Ban Tin học, đúng với đáp án và biểu điểm được nêu trong hướng dẫn chấm thi.
2. Điểm bài thi được xuất từ phần mềm chấm thi; không quy tròn điểm thành phần của từng câu và điểm của bài thi.

II. Đáp án và biểu điểm

Bài 1. Kí tự gần nhất (3,0 điểm)

Tổng quan

- Subtask 1: 50% số test có $n, q \leq 2000$.
- Subtask 2: 50% số test còn lại có $n, q \leq 10^5$.

Subtask 1

Chạy về trái hoặc phải theo yêu cầu của truy vấn, tìm chỉ số gần nhất thỏa mãn.

Subtask 2

Tạo mảng **prefix** và **suffix** với:

- $\text{pre}[i][j]$ là chỉ số của chữ cái j gần nhất về bên trái tính từ i. Tại mỗi vị trí i:
  - Nếu $S_{i-1} = j$ (với j là một kí tự từ a đến z) thì $\text{pre}[i][j] = i - 1$.
  - Nếu không thì $\text{pre}[i][j] = \text{pre}[i-1][j]$.
- $\text{suf}[i][j]$ là chỉ số của chữ cái j gần nhất về bên phải tính từ i. Tại mỗi vị trí i:
  - Nếu $S_{i+1} = j$ (với j là một kí tự từ a đến z) thì $\text{suf}[i][j] = i + 1$.
  - Nếu không thì $\text{suf}[i][j] = \text{suf}[i+1][j]$.

Bài 2. Dãy ước (3,0 điểm)

Tổng quan

- Subtask 1: 20% số test có $n \leq 10$ và $k \leq 7$.
- Subtask 2: 30% số test có $k \leq 2$.
- Subtask 3: 50% số test còn lại không có ràng buộc gì thêm.

Subtask 1

Quay lui sinh mọi cấu hình dãy tốt thỏa điều kiện đề bài và đếm.

Độ phức tạp: $O(n^k)$.

Subtask 2

Sử dụng 2 vòng `for` lồng nhau duyệt qua các giá trị bé hơn hoặc bằng n để lần lượt đặt vào vị trí 1 và 2 rồi đếm số lượng cấu hình thỏa yêu cầu đề bài.

Độ phức tạp: $O(n^2)$.

Subtask 3

Gọi dp$[i][j]$ là số dãy tốt khi điền số j vào vị trí i. Ta có công thức truy hồi:

$$\text{dp}[i][j] = \left(\sum_{m|j} \text{dp}[i-1][m] \right) \% \mod$$

(Chú thích: $\sum_{m|j} \text{dp}[i-1][m]$ là tổng của tất cả các số hạng dp$[i-1][m]$ với m là ước của j).

Để duyệt qua tất cả ước của x một cách hiệu quả, với mỗi x, ta có thể sàng trước tất cả ước của x rồi lưu vào mảng vector.

Độ phức tạp: $O(nk\sqrt[3]{n})$

(Độ phức tạp gần đúng: **trung bình** mỗi số sẽ có số lượng ước bằng căn bậc 3 của số đó)

Bài 3. Đếm dãy (2,0 điểm)

Tổng quan

- Subtask 1: 25% số test có $n \leq 5000$.
- Subtask 2: 25% số test có $l = r$.
- Subtask 3: 25% số test có $l = 0$.
- Subtask 4: 25% số test còn lại không có ràng buộc gì thêm.

Subtask 1

Sử dụng kiến thức prefix sum để tính nhanh tổng đoạn $[l, r]$ trong $O(1)$. Sử dụng 2 vòng `for` lồng nhau duyệt qua mọi cặp (l, r) sao cho $l \leq r$ và đếm số lượng đoạn $[l, r]$ thỏa yêu cầu bài toán.

Độ phức tạp: $O(n^2)$.

Subtask 2

Tạo một `map<long long, long long>` (tạm gọi là cnt) để lưu số lần xuất hiện của mỗi prefix sum (lưu ý khi khởi tạo cnt thì cnt[0] = 1 vì pre[0] = 0, tức là giá trị 0 đã xuất hiện 1 lần trong prefix sum).

Gọi pre[i] là prefix sum tại vị trí i, ta cần đếm số lượng vị trí j thỏa $1 \leq j \leq i$ sao cho $\text{sum}(j, i) = L \Rightarrow \text{pre}[i] - \text{pre}[j-1] = L \Rightarrow \text{pre}[j-1] = \text{pre}[i] - L$. Như vậy, ta cần đếm số lần giá trị $\text{pre}[j-1] = \text{pre}[i] - L$ xuất hiện trong cnt. Với mỗi i, ta cộng cnt[pre[i] $- L$] vào đáp án cuối cùng và sau đó cập nhật cnt[pre[i]] $++$.

Độ phức tạp: $O(n \log n)$.

Subtask 3

Ta phát biểu lại bài toán thành đếm số lượng đoạn con liên tiếp có tổng bé hơn hoặc bằng R.

Ý tưởng: với mỗi vị trí i, ta cần tìm vị trí $1 \leq j \leq i$ và j bé nhất sao cho $\text{sum}(j, i) \leq R$. Số lượng đoạn con liên tiếp kết thúc tại vị trí i và có tổng nhỏ hơn hoặc bằng R sẽ là $i - j + 1$, vì các số hạng trong dãy A không âm, mọi vị trí k thỏa $j \leq k \leq i$ đều có $\text{sum}(k, i) \leq R$. Như vậy, duyệt i từ 1 đến n, mỗi vị trí i sẽ đóng góp $i - j + 1$ giá trị vào đáp án cuối cùng.

Như vậy với mỗi vị trí i, ta cần tìm một vị trí j $(1 \leq j \leq i)$ sao cho j bé nhất và $\text{sum}(j, i) \leq R$. Ta có các nhận xét sau:

- $\text{sum}(j, i) = \text{pre}[i] - \text{pre}[j-1]$ và pre[i] cố định.
- Mảng pre[] luôn không giảm (vì các số hạng của dãy A có giá trị không âm).

Từ hai nhận xét trên, ta suy ra:

- Khi j giảm, pre[$j-1$] giảm, $\text{sum}(j, i)$ tăng;
- Khi j tăng, pre[$j-1$] tăng, $\text{sum}(j, i)$ giảm.

Từ đó, ta có hai phương pháp để tìm vị trí j cho mỗi i:

- Chặt nhị phân: $O(n \log n)$
- 2 con trỏ: $O(n)$

Độ phức tạp: $O(n)$.

Subtask 4

Gọi số lượng đoạn con liên tiếp trong mảng của dãy A có tổng bé hơn hoặc bằng x là hàm calc(x). Số lượng đoạn con liên tiếp $[l, r]$ có $L \leq \text{sum}(l, r) \leq R$ sẽ là $\text{calc}(R) - \text{calc}(L-1)$. Phương pháp giải ở **Subtask 3** chính là ý tưởng để cài đặt hàm calc(x).

Độ phức tạp: $O(n)$.

Bài 4. Waifu (2,0 điểm)

Tổng quan

- Subtask 1: 25% số test có $n \leq 100$.
- Subtask 2: 25% số test có $n \leq 2000$.
- Subtask 3: 25% số test có $n \leq 5000$.
- Subtask 4: 25% số test có $n \leq 10^5$.

Subtask 1

Sinh nhị phân toàn bộ trường hợp.

Độ phức tạp: $O(2^{n \log n})$.

Subtask 2

Gọi:

- $\text{dp}(i, j)$ là số cách tối thiểu để đi từ vị trí i đến vị trí j. Khi đó, kết quả cần tìm là $\text{dp}(1, n)$.
- $\min(i, j) = \min(\text{a}[i], \text{a}[i+1], \ldots, \text{a}[j])$.
- $\max(i, j) = \max(\text{a}[i], \text{a}[i+1], \ldots, \text{a}[j])$.

Ta có các nhận xét sau:

- Nếu $\min(i+1, j-1) > \max(\text{a}[i], \text{a}[j])$ thì $\text{dp}(i, j) = 1$.
- Nếu $\max(i+1, j-1) < \min(\text{a}[i], \text{a}[j])$ thì $\text{dp}(i, j) = 1$.
- $\text{dp}(i, i+1) = 1$.

Từ đó, ta tính được $\text{dp}(i, j)$ như sau:

$$\text{dp}(i, j) = \min\Big(\min\Big(\text{dp}(i, j), \text{dp}(i, j-1) + 1, \text{dp}(i+1, j) + 1\Big), \text{dp}(i, k) + \text{dp}(k, j)\Big),$$

với mọi k từ $i+1$ đến $j-1$.

Độ phức tạp: $O(n^3 \log n)$.

Subtask 3

Ta định nghĩa lại $\text{dp}(i)$ là số cách tối thiểu để đi từ vị trí 1 đến vị trí i. Khi đó, kết quả cần tìm là $\text{dp}(n)$.

Đồng thời, thay vì tính min hoặc max của đoạn ở giữa, ta duyệt tuần tự ngược về từng số một rồi cập nhật min max, như vậy vẫn sẽ thỏa điều kiện của đề bài. Từ đó, ta tính được $\text{dp}(i)$ theo các công thức sau:

- $\text{dp}(i) = \text{dp}(i-1) + 1$ (2 vị trí kề nhau)
- $\text{dp}(i) = \min\Big(\text{dp}(i), \text{dp}(j) + 1\Big)$ (j là một trong hai vị trí thỏa mãn điều kiện của đề bài).

Độ phức tạp: $O(n^2)$.

Subtask 4

Chú ý rằng ta có thể tối ưu **Subtask 3** từ $O(n^2)$ thành $O(n)$ vì ta không nhất thiết phải duyệt tất cả các vị trí để tìm vị trí thỏa mãn điều kiện. Thay vào đó, chúng ta chỉ quan tâm tới việc từ một vị trí bất kì, ta có thể nhảy tới các vị trí nào và vị trí nào là tối ưu nhất.

Do đó, với mỗi vị trí bất kì ta có thể lưu lại vị trí có thể tới bằng cách sử dụng stack. Ở điều kiện max, mỗi một vị trí duyệt tới ta tìm vị trí lớn hơn rồi nối nó với vị trí kia như hai đỉnh. Điều này cũng tương tự với điều kiện min. Vậy, ta tính $\text{dp}(i)$ như sau:

$$\text{dp}(i) = \min\Big(\text{dp}(i), \text{dp}(x) + 1\Big),$$

với x là những vị trí mà từ i ta có thể đi tới.

Độ phức tạp: $O(n)$.

III. Code mẫu

Bài 1. Kí tự gần nhất (3,0 điểm)

https://ideone.com/F3jVlv

Bài 2. Dãy ước (3,0 điểm)

https://ideone.com/H4B52s

Bài 3. Đếm dãy (2,0 điểm)

https://ideone.com/qRTXbL

Bài 4. Waifu (2,0 điểm)

- **Subtask 2:** https://ideone.com/tmw28u
- **Subtask 3:** https://ideone.com/AGJFMQ
- **Subtask 4:** https://ideone.com/mbRrx6

— **HẾT** —