SỞ GIÁO DỤC VÀ ĐÀO TẠO
HÀ NỘI

ĐÁP ÁN
ĐỀ CHÍNH THỨC

KÌ THI CHỌN ĐỘI TUYỂN THÀNH PHỐ
DỰ THI HỌC SINH GIỎI QUỐC GIA
NĂM HỌC 2022 - 2023

Môn thi: **TIN HỌC**
Ngày thi: 22/10/2022
Thời gian làm bài: 180 phút

Bài I. Xóa xâu *(5 điểm)*

Bài gồm 20 test.

- **Subtask 1 (5 test)**: Duyệt nhị phân các vị trí cần xoá. ĐPT $O(2^n)$
- **Subtask 2 (5 test)**: Chọn 2 vị trí i, j cố định đoạn cần giữ lại sau khi xoá ở hai đầu của xâu S.

  Duyệt các ký tự từ i đến j của xâu s, có 2 trường hợp xảy ra:

  1. Nếu có một loại ký tự nào đó có số lượng < K thì cách chọn các vị trí từ i đến j là không thoả mãn
  2. Ngược lại, kết quả của cách chọn này sẽ là (j-i+1)- m*k

  ĐPT $O(n^3)$
- **Subtask 3 (5 test)**: Giống subtask 2 nhưng cải tiến phần kiểm tra bằng cách dùng prefix sum để kiểm tra xem có ký tự nào xuất hiện ít hơn K lần hay không. ĐPT $O(n^2 * m)$
- **Subtask 4 (5 test)**: Cải tiến từ subtask 3 bằng cách cố định một đầu của xâu sau khi xoá i (coi như xoá các ký tự ở cuối từ vị trí i+1 đến hết) chặt nhị phân vị trí j <=i sao cho đoạn từ j đến i các chữ cái vẫn có ít nhất K ký tự. ĐPT O(n*log(n)*m)

Bài II. Tứ giác *(5 điểm)*

Bài gồm 20 test.

- **Subtask 1 (5 test):** Duyệt 4 đỉnh sau đó tính diện tích tứ giác tạo bởi 4 đỉnh đó

  ĐPT $O(n^4)$
- **Subtask 2 (5 test):** Với mọi cặp đỉnh thứ (i,j) ta tìm cặp đỉnh thứ (ii,jj) sao cho diện tích của tứ giác tạo bởi 4 đỉnh (i, ii, j, jj) là lớn nhất.

  Diện tích tứ giác (i, ii, j, jj) = Diện tích tam giác (i, j, ii) + Diện tích tam giác (i, j, jj)

  Vậy (ii, jj) là hai đỉnh đối diện cắt đoạn thẳng nối hai đỉnh (i,j) sao cho khoảng cách ii, jj tới đường thẳng (i,j) là lớn nhất.

  Cố định hai đỉnh (i,j). Vì các đỉnh đã được liệt kê theo chiều kim đồng hồ. Đỉnh ii là đỉnh nằm trong khoảng từ i+1 đến j-1 sao cho khoảng cách từ đỉnh ii đến đoạn thẳng (i,j) là lớn nhất. Đỉnh jj là đỉnh nằm trong khoảng từ j+1 đến i-1 (theo vòng tròn) có khoảng cách từ đỉnh jj đến đoạn thẳng (i,j) là lớn nhất.

  ĐPT $O(n^3)$
- **Subtask 3 (5 test):**

  Cố định hai đỉnh (i,j). Sử dụng kỹ thuật chặt tam phân để tìm hai đỉnh (ii,jj)

  ĐPT $O(n^2 * \log n)$
- **Subtask 4 (5 test):**

  Nhận xét: Khi cố định điểm i và điểm j chạy theo chiều kim đồng hồ thì hai điểm ii, jj cũng chạy theo chiều tương tự. Vậy, ban đầu với mỗi điểm i cố định, ta tìm 3 đỉnh j, ii, jj sao cho tứ giác

tạo bởi 4 đỉnh (i, ii, j, jj) có diện tích lớn nhất. Sau đó, khi j dịch chuyển thì ii, jj cũng dịch chuyển cùng chiều, sử dụng kỹ thuật two-pointer để duy trì ii, jj

ĐPT $O(n^2)$

Bài III. Chơi golf *(5 điểm)*

Bài gồm 20 test.

- **Subtask 1 (4 test):**

Vì không có vật cản nên khoảng cách gần nhất là khoảng các Mahatan

Kết hợp với mỗi lần di chuyển chỉ được đi tối đa K bước ta có công thức tính:

Gọi (x,y) là toạ độ ô xuất phát, (u,v) là toạ độ ô đích

Kq = abs(x-u)/k + (abs(x-u)%k!=0) + abs(y-v)/k + (abs(y-v)%k!=0)

- **Subtask 2,3: (4 test + 4 test)**

Sử dụng thuật toán BFS trên bảng. Với mỗi ô (x,y) ta có thể đi tới:

các ô bên phải: (x+1,y), (x+2,y),…(x+k,y)

các ô bên trái: (x-1,y), (x-2,y),…(x-k,y)

các ô bên trên: (x,y-1), (x,y-2)…(x,y-k)

các ô bên dưới: (x, y+1), (x,y+2),…(x,y+k

ĐPT O(R*C*K)

- **Subtask 4: (4 test)**

Vì K >= max(R,C) nên mỗi lần di chuyển ta có thể di chuyển tới bất kì ô nào trong bảng ở hướng đó.

Sử dụng thuật toán BFS trên bảng với trạng thái (x,y,h) đang đứng ở ô (x,y), h là hướng hiện tại. Mỗi một bước nếu hướng vẫn giữ nguyên, ta coi trọng số cạnh là 0, nếu hướng bị thay đổi ta coi trọng số cạnh là 1. Vì trọng số của đồ thị là 0/1, sử dụng kỹ thuật queue hai đầu để xử lý

ĐPT O(R*C)

- **Subtask 5: (4 test)**

Sử dụng thuật toán BFS, để tránh qúa thời gian do độ phức tạp R*C*(R+C) ta có hai cách sau:

- Sử dụng thuật toán DSU để tìm nhanh được những ô chưa bị đánh dấu

- Nhận xét thấy nếu loang từ ô (x,y) đến ô (xx,yy). Nếu ô (xx,yy) đã được đánh dấu rồi thì có hai trường hợp sau:

- Hoặc ô (x,y) được loang đến sau ô (xx,yy), như vậy ta không cần loang theo hướng từ ô (x,y) đến (xx,yy) nữa vì ô (xx,yy) đã loang tới rồi

- Hoặc ô (xx,yy) chưa được lấy trong queue ra thì các ô theo hướng ô (x,y) đến (xx,yy) chưa được thăm, vậy ta sẽ thăm các ô theo hướng này.

ĐPT O(R*C)

Bài IV. Xếp lịch*(5 điểm)*

Bài gồm 20 test.

- **Subtask 1 (5 test):**

Nhận xét đầu tiên ta có là: nếu công việc y thực hiện sau công việc x, nhưng công việc y lại chuẩn bị trước công việc x, thì rõ ràng ta có thể đổi thứ tự thực hiện của hai công việc này mà thời gian sẽ không lớn hơn thời gian với thứ tự trước.

Thật vậy, nếu thời điểm chuẩn bị xong của công việc x là tc_x, sau thời gian này công việc x mới thực hiện, như vậy nếu y mà thực hiện sau x thì thời gian tối thiểu để kết thúc của cả 2 công việc này là $tc_x + b_x + b_y$, nếu ta cho công việc y thực hiện trước thì thời gian kết thúc của 2 công việc này sẽ là $\max(tc_y + b_y + b_x, tc_x + b_x)$, rõ ràng sẽ không thể lớn hơn khi y thực hiện sau x được vì y chuẩn bị trước x nên $tc_y < tc_x$.

Như vậy, thứ tự thực hiện của các công việc sẽ chính là thứ tự chuẩn bị.

Vậy cách làm là: duyệt hoán vị thứ tự thực hiện các công việc.

ĐPT O(n!)

- **Subtask 2 (5 test):** Quy hoạch động bitmask.

  F(mask) lưu thời gian sớm nhất để tất cả những công việc được hoàn thành xong (với mask là trạng thái đã hoàn thành/chưa chuẩn bị của từng công việc)

  Tại mỗi bước, chọn công việc bất kì chưa được chuẩn bị và trong lượt hiện tại sẽ chuẩn bị và thực hiện công việc đó (gọi công việc đã chọn là u).

  F(mask) = max (F(mask & (1<<u)), sum(mask)+a[u]) + b[u]

  Sum(mask) = tổng các a[i] nếu công việc thứ i đã làm xong (bit i trong mask = 1)

  ĐPT $O(n*2^n)$

- **Subtask 3 (5 test):** Bài toán này thực chất chính là bài toán lập lịch trên 2 máy. Sử dụng thuật toán Johnson như sau:
  1. Chia các khách thành hai tập, tập thứ nhất có $a_i < b_i$, tập thứ hai có $a_i > b_i$
  2. Tập thứ nhất sắp xếp theo chiều tăng của a_i , tập thứ hai sắp xếp theo chiều giảm của b_i
  3. Ghép thứ tự của tập 2 sau khi sắp xếp vào tiếp tục thứ tự của tập 1 sau khi sắp xếp ta có thứ tự tối ưu.

- **Subtask 4 (5 test):**
  Tiếp theo ta có nhận xét sau: nếu thứ tự chuẩn bị và thực hiện của N công việc là $p_1, p_2, ..., p_N$ thì thời gian kết thúc sẽ là:

  $T = \max_{i=1 \to N} \left(\sum_{j=1}^{i} a_i + \sum_{j=i}^{N} b_{p_j} \right)$

  Như vậy, bài toán trở về tìm thứ tự $p_1, p_2, ..., p_N$ để biểu thức 2 đạt giá trị nhỏ nhất hay giá trị $c_i = a_{p_i} + \sum a_{p_i} - b_{p_j}$ là nhỏ nhất.

  Để giải quyết bài toán này, ta xử lý offline, sắp xếp thứ tự của tất cả các công việc, sau đó xây dựng cây IT xử lý công thức (1) để tính max.

…….. **Hết** ……..

SỞ GIÁO DỤC VÀ ĐÀO TẠO
HÀ NỘI

ĐÁP ÁN
ĐỀ CHÍNH THỨC

KÌ THI CHỌN ĐỘI TUYỂN THÀNH PHỐ
DỰ THI HỌC SINH GIỎI QUỐC GIA
NĂM HỌC 2022 - 2023

Môn thi: **TIN HỌC**
Ngày thi: 23/10/2022
Thời gian làm bài: 180 phút

Bài V. Trò chơi *(7 điểm)*

Bài gồm 20 test.
Phát triển từ quy hoạch động cái túi

- **Subtask 1:** Có 30% số test ứng với 30% số điểm có $N \leq 2; W \leq 20$; **(6 test)**
  Duyệt $O(N^W)$
- **Subtask 2:** 20% số test khác ứng với 20% số điểm có $N, W \leq 300$; **(4 test)**
  DP $O(N \times W^2)$)
- **Subtask 3:** 20% số test khác ứng với 20% số điểm có $N \leq 1000$; **(4 test)**
  Với mỗi giá trị m, chỉ cần quan tâm tới tối đa W/m giá trị, nên sinh ra tất cả các khả năng của mỗi w và chọn W/m số lớn nhất xong dp, $O(NWlog)$
- **Subtask 4:** 20% số test khác ứng với 20% số điểm có năng lượng của màn chơi là như nhau; **(4 test)**
  Chỉ cần quan tâm tới W/m số lớn nhất, dùng set để sinh ra nhanh $O(NlogN)$
- **Subtask 5:** 10% số test còn lại ứng với 10% số điểm không có ràng buộc gì thêm. **(2 test)**
  Kết hợp **Subtask 3** và **Subtask 4**

Bài VI. Xóa số *(7 điểm)*

Bài gồm 20 test.

- **Subtask 1:** Có 20% số test ứng với 20% số điểm có $N \leq 10^3$; **(4 test)**
  Duyệt toàn bộ, làm theo mô tả của đề bài.
- **Subtask 2:** 20% số test khác ứng với 20% số điểm có $k = 2$; **(4 test)**
  Duy trì một cây ST, mỗi node quản lý 2 giá trị: max vị trí chẵn và max vị trí lẻ. Giá trị cần xóa mỗi bước là giá trị lớn nhất ở vị trí 1 trong node gốc. Khi tìm được giá trị cần xóa, tìm node bên trái nhất có giá trị đó có vị trị x, update node đó giá trị 0, sau đó lazy update các node từ vị trí $x + 1$ đến n.
- **Subtask 3:** 20% số test khác ứng với 20% số điểm có $k \leq 10$; **(4 test)**
  Tương tự Subtask 2 nhưng cải tiến, mỗi node lưu k giá trị tương ứng với giá trị max ở các vị trí khi chia lấy dư cho k.
- **Subtask 4:** 20% số test khác ứng với 20% số điểm có $100 \leq k \leq N$; **(4 test)**
  Nhận xét thấy khi $k \geq 100$ thì mỗi lần tìm max đpt tối đa là 10^4. Vậy duyệt toàn bộ theo mô tả đề bài đpt sẽ là $\frac{N^2}{k} \sim 10^8$.
- **Subtask 5:** 20% số test còn lại ứng với 20% số điểm không có ràng buộc gì thêm. **(4 test)**
  Chia dãy thành $\sqrt{n}$ đoạn, với mỗi đoạn x ta lưu trữ các thông tin sau:
  + c[x] số phần tử còn lại chưa bị xóa
  + mx[x][t] = (val, pos) là số lớn nhất và vị trí của số đó trong các vị trí p%k = t, lưu ý rằng trong mỗi đoạn chỉ có tối đa $\sqrt{n}$ phần tử và các vị trí này sẽ có số dư cho k là liên tiếp nhau, do đó ta có thể ánh xạ các số dư này thành từ 0 đến min(k, $\sqrt{n}$) − 1. Với mỗi bước xóa ta sẽ thực hiện như sau:
  + Duyệt lần lượt từng đoạn, ta sẽ kiểm tra xem trong đoạn thứ x có chứa phần tử nào có chỉ số chia hết cho k bằng cách duy trì biến cnt là số lượng phần tử trước đoạn x (khi duyệt

qua x ta cập nhật cnt+ = c[x]), ta sẽ kiểm tra [cnt, cnt + c[x]) có tồn tại f nguyên thỏa mãn cnt ≤ f × k < cnt + c[x]. Sau đó từ f ta tính được số dư sau khi được ánh xạ của đoạn x của các vị trí chia hết cho k là t = (f × k − cnt)%k, và tìm được số lớn nhất cũng như vị trí trong đoạn sẽ là mx[x][t].

+ Sau khi duyệt qua các đoạn ta tìm được số lớn nhất và vị trí của số đó, ta thực hiện việc xóa số đó đi bằng cách cập nhật lại đoạn chứa số đó. Ta cập nhật c[x] − −, và cập nhật lại mảng mx[x] bằng cách duyệt lần lượt các phần tử còn lại của đoạn này.

Với cả 2 thao tác trên ta đều chỉ tốn O($\sqrt{n}$), và thực hiện n lần nên ta có O(n $\sqrt{n}$).

Bài VII. Đoạn thẳng *(6 điểm)*

Bài gồm 50 test.

- **Subtask 1:** Có 10% số test ứng với 10% số điểm có $M \leq 10$; **(5 test)**

  Duyệt quay lui
- **Subtask 2:** 10% số test khác ứng với 10% số điểm có $M \leq 20$; **(5 test)**

  DP bitmask

  Dễ thấy nếu có thể được X đoạn thì $X-1$ đoạn cũng thỏa mãn nên ta chặt nhị phân đáp án. Ta nói đoạn $[l,\ r]$ và số X có thể ghép cặp với nhau nếu $X < l$ hoặc $X < r$. Khi đó bài toán chuyển về hãy kiểm tra xem với n đoạn và m con số, có thể ghép cặp được tối thiểu $n-k$ cặp thoả mãn hay không.
- **Subtask 3:** 20% số test khác ứng với 20% số điểm có $M \leq 10^3$; **(10 test)**

  Tới đây ta xây đồ thị hai phía giữa các đoạn và các số. Có cạnh nối nếu đó là một cặp ghép hợp lý. Chạy thuật toán cặp ghép nếu ra kết quả lớn hơn hoặc bằng $n-k$ tức là ghép được.
- **Subtask 4:** 20% số test khác ứng với 20% số điểm có $M \leq 10^5$; $K = 0$; **(10 test)**

  Sắp xếp các đoạn và các số theo thứ tự tăng dần. Ta có nhận xét như sau:

  Đặt $f(i)$ là chỉ số của đoạn trái nhất mà $L_{f(i)} > a_i$. Tương tự đặt $g(i)$ là chỉ số của đoạn phải nhất mà $R_{g(i)} < a_i$. Hay nói cách khác, số a_i có thể ghép cặp với các đoạn $1, 2, \dots, g(i)$ và $f(i), f(i+1), \dots, n$.

  Giả sử có hai số a_i, a_j và hai đoạn p, q sao cho $R_p < a_i < a_j < L_q$, hoặc $a_j < a_i < L_p < L_q$, hoặc $R_p < R_q < a_j < a_i$. Khi đó, thay vì ghép cặp (i,p) và (j,q) thì ta có thể ghép cặp (i,q) và (j,p).

  Từ đó ta có thể rút ra một nhận xét rằng, nếu có một cách nào đó để ghép được đầy đủ X cặp cho X đoạn thì chắc chắn ta có thể sử dụng y số trái nhất để ghép cặp với y đoạn phải nhất, và $X-y$ số phải nhất để ghép cặp với $X-y$ đoạn trái nhất $(0 \leq y \leq X)$. Cụ thể hơn, ta có thể ghép cặp như sau:

Số thứ	**Đoạn thứ**	**Số thứ**	**Đoạn thứ**
1	$X-y+1$	$m-(X-y)+1$	1
2	$X-y+2$	$m-(X-y)+2$	2
3	$X-y+3$	$m-(X-y)+3$	3
...	...	...	...
i	$X-y+i$	j	$j-(m-(X-y))$
y	X	m	$X-y$

  Điều kiện cần và đủ để số thứ i $(1 \leq i \leq y)$ có thể ghép cặp với đoạn tương ứng là:

  $f(i) \leq X - y + i$, hay tương đương với: $y \leq X + i - f(i)$

  Điều kiện cần và đủ để số thứ j $(m-(X-y)+1 \leq j \leq m)$ có thể ghép cặp với đoạn tương ứng là $g(j) \geq j-(m-(X-y))$, hay tương đương với $y \geq X - m + j - g(j)$

  Do đó, ta chỉ cần tìm một giá trị $y \leq X$ bất kì thoả mãn toàn bộ các điều kiện trên thì chắc chắn tồn tại cách ghép thoả mãn. Ngược lại nếu không tìm thấy giá trị y nào như vậy thì không có cách ghép nào thoả mãn. Việc tìm một giá trị k như vậy có thể làm bằng cách duyệt qua tất

cả các giá trị y và lưu trữ min, max các giá trị cần thiết của các tiền tố và hậu tố tương ứng trong $O(m)$.

Tóm lại: Vậy với mỗi truy vấn, ta chặt nhị phân đáp án trong $O(log(n))$, việc kiểm tra cần phải sắp xếp lại các số và các đoạn trong $O(mlog(m))$ và xử lý trong $O(m)$. Do đó, độ phức tạp thuật toán là: $O(m \times log(m) \times log(n))$.

Ghi chú: Có thể có cách khác không cần tìm điều kiện của f, g mà chỉ cần chặt nhị phân giá trị y và kiểm tra $O(m)$ là được. Độ phức tạp tổng thể vẫn là $O(m \times log(m) \times log(n))$.

- **Subtask 5:** 20% số test khác ứng với 20% số điểm có $M \leq 10^5$; **(10 test)**

  Giống Subtask 4, chỉ thay đổi điều kiện một chút:

  Ghép y số trái nhất với y đoạn phải nhất

  Ghép $X - y - k$ số phải nhất với $X - y - k$ đoạn trái nhất

  Khi đó, điều kiện cần và đủ của f, g là:

  $f(i) \leq X - y + i$, hay tương đương với $y \leq X + i - f(i)$

  $g(j) \geq j - (m - (X - y - k))$, hay tương đương với $y \geq X - m - k + j - g(j)$

- **Subtask 6:** 20% số test còn lại ứng với 20% số điểm không có ràng buộc gì thêm. **(10 test)**

Hiện tại hàm kiểm tra đang có hai phần:

- Sắp xếp các đoạn và các số $(O(mlogm))$
- Tìm min max của tiền tố, hậu tố tương ứng $(O(m))$

Để cải thiện thuật toán hơn nữa, ta cần tăng tốc pha sắp xếp.

Việc sắp xếp các số chỉ cần một lần sắp xếp trước khi chặt nhị phân là được.

Vậy làm thế nào để đẩy nhanh việc sắp xếp các đoạn?

Nhận xét rằng vì ta đang chặt nhị phân, nên các đoạn mà ta cần sắp xếp luôn là một tiền tố các đoạn trong input. Giả sử đang có tiền tố p các đoạn đã được sắp xếp. Để sắp xếp một tiền tố khác, ta có thể tận dụng việc p đã được sắp xếp để chèn thêm, hoặc xoá đi các đoạn vào dãy này ở những vị trí hợp lý để sao cho dãy tiền tố mới cũng được sắp xếp.

Nói cách khác, ta phải xây dựng CTDL để có cách chèn, xoá các đoạn vào các vị trí phù hợp và thỏa mãn các điều kiện sau:

1. Chi phí tìm vị trí phù hợp để chèn / xoá vào đủ nhỏ
2. Chi phí chèn, xoá đủ nhỏ
3. Chi phí truy cập vào mọi tiền tố, hậu tố tổng cộng là tuyến tính để không làm ảnh hưởng tới pha xử lý tìm min, max tiền tố hậu tố

Từ tính chất $2, 3$, ta có thể nghĩ tới sử dụng $linked\ list$ để chèn, xoá các đoạn này trong $O(1)$.

Ta đồng thời lưu thêm 1 set để mỗi lần tìm vị trí cần chèn trong $O(log)$.

Như vậy, độ phức tạp tổng cộng để sắp xếp là: $O((số\ lần\ chèn, xoá\ đoạn) \times log)$

Nhận xét rằng khi ta chặt nhị phân, tổng số đoạn ta cần quan tâm thực ra chỉ là:

$$n / 2 + n / 4 + n / 8 + ... = n$$

Do đó, thực ra ta chỉ cần chèn, xoá tổng cộng n lần, dẫn tới độ phức tạp của việc sắp xếp sau tất cả các lần check là $O(nlogn)$

Vậy độ phức tạp tổng thể là $O(nlogn + mlogm)$

…….. **Hết** ……..