TRẠI HÈ HÙNG VƯƠNG
LẦN THỨ XIV - PHÚ THỌ 2018

HƯỚNG DẪN GIẢI ĐỀ THI CHỌN HỌC SINH GIỎI
MÔN: TIN HỌC - KHỐI 11
Ngày thi: 29 tháng 7 năm 2018
Hướng dẫn này gồm có 03 trang

ĐỀ CHÍNH THỨC

Bài 1. Đấu vật

Với nhóm Tests I: So sánh trực tiếp ($a_1, a_2, \ldots, a_n$) lần lượt với ($b_1, b_2, \ldots, b_n$), ($b_2, b_3, \ldots, b_{n-1}, b_n$), . . ., ($b_n, b_1, b_2, \ldots, b_{n-1}$), rút các kết quả cần thiết,

Với nhóm Tests II: Chuẩn bị mảng số liệu so sánh (để tránh tạo lại mảng hoặc liên tục kiểm tra, tính lại chỉ số) theo nguyên lý xử lý dữ liệu vòng tròn:

Tạo mảng ($b_1, b_2, \ldots, b_n, b_1, b_2, \ldots, b_n$), lần lượt so sánh ($a_1, a_2, \ldots, a_n$) với ($b_i, b_{i+1}, \ldots, b_{n+i-1}$), $i = 1 \div n-1$.

Độ phức tạp: $O(n^2)$.

Với nhóm Tests III: Từng b_i ($i = 1 \div n$) tính số lượng vị trí cần chạy thêm để b_i gặp đối thủ cũ, tổng hợp các kết quả theo số lượng vị trí cần chạy thêm. Việc sử dụng các hàm tìm kiếm cung cấp trong hệ thống lập trình C++ cho phép giải quyết bài toán trong phạm vi thời gian đã cho.

Với nhóm Tests IV: Cần giải thuật tối ưu.

Hoán vị vòng tròn.

Xây dựng mảng $P = (p_1, p_2, \ldots, p_n)$, trong đó p_i – vị trí người thứ i của đội **A** trong danh sách xếp hàng: $p_{a_i} = i$.

Người thứ i trong danh sách xếp hàng của **B** là b_i,

Để gặp được đối thủ cũ của mình người b_i cần đứng cạnh sàn đấu p_{b_i}.

Như vậy, để từ vị trí i sang vị trí p_{b_i} số sàn đấu cần chạy thêm sau một vòng theo chiều kim đồng hồ sẽ là `k = (i-p[b[i]]+n)%n;`

Giá trị `k` nằm trong phạm vi từ `0` đến `n-1`.

Với mỗi người của B tính giá trị k tương ứng, tích lũy số người có cùng giá trị k và tìm max trong các giá trị tích lũy được.

Độ phức tạp của giải thuật: $O(n)$.

Tổ chức dữ liệu:

- Mảng `vector<int> p(n+1)` – Lưu vị trí của người thứ i của **A**, $i = 1 \div n$,
- Mảng `vector<int> r(n,0)` – Lưu số người gặp đối thủ cũ khi chạy thêm 0,1, 2, . . ., `n-1` sàn đấu.

Chương trình

```
#include <bits/stdc++.h>
#define NAME "chess."
using namespace std;
```

```
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,k,ans=0;

int main()
{
    fi>>n;
    vector<int> p(n+1),r(n,0);
    for(int i=1; i<=n; ++i){fi>>k; p[k]=i;}
    for(int i=1; i<=n; ++i)
    {
        fi>>k;
        k = (i-p[k]+n)%n;
        ++r[k];
    }
    for(int i=0; i<n; ++i)if(ans<r[i]) ans=r[i], k=i;
    fo<< k <<' '<< ans;
    //fo<<"\nTime: "<<clock()/(double)1000<<" sec";
}
```

Bài 2. Cột điện

Subtask 1. Sử dụng duyệt chọn các giá trị từng cột. Độ phức tạp $O(3^n)$

Subtask 2. Gọi $F[i][x]$ là chi phí tối thiểu của i cột đầu tiên nếu cột i có chiều cao x.

Tính: $F[i][x] = \min F[i-1][k] + c \times |x-k| + (k-h_i)^2$ với $k \geq h_i; |k-x| \leq d$.

Đáp số: $\min F[n][x]$. Độ phức tạp: $O(n \times h^2)$

Subtask 3.

Nhận thấy

$$F[i][x] = (F[i-1][k] + ck) - cx + (k-h_i)^2 \text{ với } k \in [x, x+d]$$

$$F[i][x] = (F[i-1][k] - ck) + cx + (k-h_i)^2 \text{ với } k \in [x-d, x]$$

Đặt:

$$Low[i-1][x] = min(F[i-1][k] - ck) \;\; với\; k \in [x-d, x]$$

$$High[i-1][x] = \min((F[i-1][k] + ck) \; với\; k \in [x, x+d]$$

Khi đó: $F[i][x] = \min(Low[i-1][x] + cx, High[i-1][x] - cx) + (k-h_i)^2$

Ta có thể tính: $Low[i][x], High[i][x]$ sử dụng RMQ hoặc Interval Tree.

Độ phức tạp $O(n \times hmax \times \log_2 h)$

Subtask 3: Sử dụng Deque để tính Low, High. Độ phức tạp: $O(n \times hmax)$

Bài 3. Thành phần liên thông

Subtask 1: Với mỗi truy vấn thực hiện bỏ các cạnh và đếm lại số thành phần liên thông.

Độ phức tạp là $O(k \times (m+n))$

Subtask 2: Có thể cải tiến lại cách 2 như sau:

Trước hết tạo ra rừng cho đồ thị (vì đồ thị không liên thông nên không thể có 1 cây mà là 1 rừng cây):

+ Tạo một rừng xét theo chiều xuôi các cạnh từ 1 đến M, trong đó cần lưu lại chỉ số các cạnh và các cạnh của rừng

+ Tạo một rừng xét theo chiều ngược các cạnh từ M về 1, trong đó cần lưu lại chỉ số các cạnh và các cạnh của rừng

Sẽ có những cạnh của đồ thị không thuộc rừng cây thứ nhất, nhưng nó sẽ thuộc rừng cây thứ 2 và ngược lại. Đó chính là chìa khóa để giải quyết bài toán.

Xét truy vấn [l_i, r_i]:

- Ban đầu số thành phần liên thông là N

- Xét các cạnh thuộc rừng cây thứ nhất nhưng không thuộc đoạn chỉ số [l_i, r_i], nối cạnh đó lại (nhớ tìm gốc chung cho 2 đỉnh thuộc cạnh bằng DSU) và giảm số tplt 1 đơn vị.

- Sau bước trên ta thấy sẽ có những cạnh không thuộc rừng cây thứ nhất nhưng cũng không thuộc đoạn [l_i, r_i], nhưng những cạnh đó lại thuộc rừng cây thứ hai và ta xét các cạnh đó ở rừng cây thứ 2 (nhớ tìm gốc chung cho 2 đỉnh thuộc cạnh bằng DSU) và giảm số tplt 1 đơn vị.

Số tplt cuối cùng còn lại chính là kết quả của truy vấn.

Độ phức tạp thuật toán $O(k \times N \times \alpha)$

Subtask 3:

Xây dựng trước bảng đáp án. Gọi $res[u][v]$ là số thành phần liên thông nếu sử dụng u cạnh đầu tiên của rừng thứ nhất và v cạnh đầu tiên của rừng thứ 2. Dễ dàng tính được $res[u][v]$ với trong thời gian $O(n^2\alpha)$

Với mỗi cặp l, r, tìm u là cạnh cuối cùng thuộc rừng thứ nhất có chỉ số $< l$, tìm cạnh v là cạnh cuối cùng thuộc rừng thứ 2 (rừng này xét từ đầu về cuối) có chỉ số $> r$. Kết quả cho phép truy vấn là $res[u][v]$.

Độ phức tạp: $O(n^2\alpha + klog_2 n)$