BÀI TẬP CẤU TRÚC DỮ LIỆU NÂNG CAO

1. Đếm hình bình hành (CntPar.*)

Cho (N<=2000) điểm trên mặt phẳng , đếm số hình bình hành tạo bởi N điểm .

Dữ liệu:

- số N
- N điểm nguyên có giá trị tuyệt đối của tọa độ không quá 100.

Kết quả:

Số hình bình hành

Ví dụ:

Input	Output
4 0 1 1 0 1 1 2 0	1

Hướng dẫn:

Cách 1: Với bài toán này, đơn giản nhất là sử dụng cách sinh tổ hợp 4 điểm để kiểm tra xem có phải hình bình hành không. Hoặc cách khác là sử dung kĩ thuật quản lí trung điểm bằng cách đánh dấu bằng mảng. Nhìn chung 2 cách cài đặt này khá dài dòng và phức tạp.

Cách 2: Dùng map để quản lí trung điểm, với mỗi cặp điểm trong N điểm , gọi C là trung điểm của cặp điểm đó , CNT là số lượng cặp điểm có C là trung điểm. như vậy kết quả của bài toán là $\sum(CNT_i*(CNT_i -1)/2)$.

Dễ thấy việc sử dụng map đã làm cho việc cài đặt bài toàn này trở nên rất đơn giản.

2. Lại tìm thấy (FOUND2.*)

Cho dãy số $a_1, a_2, \dots, a_n$. Xác định xem số x xuất hiện trong dãy a bao nhiêu lần?

Dữ liệu: vào từ file **FOUND2.INP**

- Dòng đầu chứa hai số nguyên dương n, m $(n, m \leq 10^5)$, với m là số trường hợp cần kiểm tra.
- Dòng thứ 2 chứa n số nguyên dương $a_1, a_2, \dots, a_n$ $(a_i \leq 10^9)$.
- M dòng tiếp theo, mỗi dòng chứa một số nguyên dương x là số cần kiểm tra.

Kết quả: Ghi ra file **FOUND2.OUT** m dòng, mỗi dòng tương ứng là số lần xuất hiện của số x tương ứng trong dãy đã cho.

FOUND2.INP	**FOUND2.OUT**
5 4 **3 5 2 7 3** **4** **3** **7** **8**	**0** **2** **1** **0**

3. Khối lập phương (CUBICS.*)

Quà sinh nhật của Jimmylà một bộ khối lập phương xếp hình. Jimmy xếp thành ***n*** tháp, tháp thứ ***i*** có độ cao là a_i ($1 \leq a_i \leq 10^9$, $1 \leq n \leq 10^5$, $i = 1 \div n$). Jimmy rất có cảm tình với số nguyên ***k***, vì vậy dãy liên tục các tháp được coi là hài hòa nếu chúng có độ cao trung bình là ***k*** ($1 \leq k \leq 10^9$).

Yêu cầu: Cho ***n***, ***k*** và a_i($i = 1 \div n$). Hãy xác định dãy tháp hài hòa dài nhất, chỉ ra tháp đầu tiên và độ dài của dãy tìm được. Nếu tồn tại nhiều dãy cùng độ dài thì chỉ ra dãy tháp có vị trí đầu nhỏ nhất. Nếu không tồn tại dãy tháp thì đưa ra một số 0.

Dữ liệu: Vào từ file văn bản CUBICS.INP:

- Dòng đầu tiên chứa 2 số nguyên ***n*** và ***k***,
- Dòng thứ 2 chứa ***n*** số nguyên $a_1, a_2, \dots, a_n$.

Kết quả: Đưa ra file văn bản CUBICS.OUT trên một dòng gồm: độ dài của dãy tìm được và số thứ tự của tháp đầu tiên hoặc một số 0 nếu không tồn tại dãy.

Ví dụ:

CUBICS.INP	CUBICS.OUT
5 3 **1 2 3 4 6**	**3 2**

4. Thống kê (statis.*)

Nguồn bài: Đề thi HSG lớp 10 năm học 2019-2020 tỉnh Thái Nguyên

Để điều tra số con trong mỗi gia đình ở huyện A, người ta chọn ra k=mxn gia đình, thống kê số con các gia đình đó và thu được mẫu số liệu dạng một bảng các số nguyên có m dòng, n cột thể hiện số con của các gia đình. Công việc tiếp theo của nhân viên điều tra là cần phân tích mẫu số liệu thu được để lập bảng phân bố tần số các số liệu trong mẫu. Hãy viết chương trình giúp nhân viên điều tra thực hiện công việc trên.

Yêu cầu: Từ mẫu số liệu đã có hãy xuất ra bảng phân bố tần số của các số liệu dưới dạng cột theo thứ tự có tần số giảm dần.

Dữ liệu: vào từ file STATIS.INP

- Dòng đầu chứa 2 số nguyên dương m và n thể hiện kích thước của mẫu số liệu.

- m dòng tiếp theo, mỗi dòng chứa n số nguyên dương (các số này không quá 100) hoặc số 0, mỗi số cách nhau 1 khoảng trắng, thể hiện số con trong mỗi gia đình tương ứng.

Kết quả: Ghi ra file STATIS.OUT

Bảng phân bố tần số của các số liệu dưới dạng cột theo thứ tự có tần số giảm dần.

Ví dụ :

STATIS.INP	**STATIS.OUT**
4 5 8 3 6 4 2 4 5 0 2 6 1 8 9 5 5	5:4 4:3 8:3 1:2

1 8 4 9 5	2:2 6:2 9:2 0:1 3:1

5. Thả xốp (SPONGE.*)

Nguồn bài tập: Chuyên đề hội thảo khoa học trại hè Hùng Vương năm 2019

Có N hạt xốp, hạt thứ i có khối lượng a_i, được thả lần lượt xuống một ống nước đặc biệt được thiết kế sao cho tại mỗi thời điểm chỉ có một hạt xốp nhẹ nhất nổi lên trên bề mặt. Trước mỗi lần thả, hạt xốp đang nổi trên bề mặt sẽ bị ngấm nước và tăng gấp đôi khối lượng. Hỏi sau khi thả hạt xốp cuối cùng vào ống thì khối lượng xốp tăng so với tổng khối lượng ban đầu là bao nhiêu?

Input: Vào từ file văn bản **SPONGE.INP**

- Dòng 1: Số nguyên dương N (N<=10^6)
- Dòng 2: N số nguyên dương

Output: Ghi ra file văn bản **SPONGE.OUT**

- Ghi 1 số duy nhất là đáp án của bài toán

Ví dụ:

SPONGE.INP	SPONGE.OUT
3 2 1 3	3

Gợi ý:

Cách 1: Nếu áp dụng theo cách thông thường thì mỗi lần thả hạt xốp xuống thì phải tìm phần tử bé nhất sau đó tăng gấp đôi khối lượng của hạt xốp đó khi đó độ phức tạp sẽ là $O(n^2)$ và kết quả chính là tổng khối lượng trong ống sau khi thả trừ đi khối lượng ban đầu.

Cách 2: Sử dụng hàng đợi ưu tiên để làm nhưng ta không sử dụng phép có sẵn là less mà ta dùng greater khi đó hạt xốp bé sẽ nổi ở đầu queue. Bài toán

trở nên đơn giản với độ phức tạp là O(n.logn).

6. Xếp hàng ưu tiên (MSE07B.*)

Nguồn bài tập: https://vn.spoj.com/problems/MSE07B/

Ngân hàng BIG-Bank mở một chi nhánh ở Bucharest và được trang bị một máy tính hiện đại với các công nghệ mới nhập, C2#, VC3+ ... chỉ chuối mỗi cái là không ai biết lập trình. Họ cần một phần mềm mô tả hoạt động của ngân hàng như sau: Mỗi khách hàng có một mã số là số nguyên K, và khi đến ngân hàng giao dịch, họ sẽ nhận được 1 số P là thứ tự ưu tiên của họ. Các thao tác chính như sau:

(0) Kết thúc phục vụ

(1KP) Thêm khách hàng K vào hàng đợi với độ ưu tiên P

(2) Phục vụ người có độ ưu tiên cao nhất và xóa khỏi danh sách hàng đợi

(3) Phục vụ người có độ ưu tiên thấp nhất và xóa khỏi danh sách hàng đợi.

Tất nhiên là họ cần bạn giúp rồi.

Input: Mỗi dòng của input là 1 yêu cầu, và chỉ yêu cầu cuối cùng mới có giá trị là 0. Giả thiết là khi có yêu cầu 1 thì không có khách hàng nào khác có độ ưu tiên là P **(**K<=10^6, P<= 10^7, tổng yêu cầu mỗi loại không vượt quá 10^5.Một khách hàng có thể yêu cầu phục vụ nhiều lần và với các độ ưu tiên khác nhau.

Output: Với mỗi yêu cầu 2 hoặc 3, in ra trên 1 dòng mã số của khách hàng được phục vụ tương ứng. Nếu có yêu cầu mà hàng đợi rỗng, in ra số 0.

Ví dụ:

MSE07B.INP	MSE07B.OUT
2 1 20 14 1 30 3 2 1 10 99 3 2 2 0	0 20 30 10 0

Gợi ý:

Nhận xét: Nếu danh sách khách hàng được đưa vào mảng không được

sắp xếp theo độ ưu tiên P, thì mỗi khi có thao tác loại 2 hoặc 3 thì ta lại phải đi tìm min, max.

Mỗi thao tác loại 2 hoặc 3 đều có độ phức tạp thuật toán phụ thuộc tuyến tính vào số lượng khách hàng. Như vậy chương trình không chạy được trong thời gian cho phép.

Do vậy, danh sách khách hàng cần được sắp xếp theo độ ưu tiên tăng (hoặc giảm). Các thao loại 1, 2, 3 là xen kẽ nhau nên danh sách khách hàng liên tục biến động, việc quản lý cũng tương đối phức tạp.

Tuy nhiên, nếu sử dụng kiểu dữ liệu SET thì vấn đề trở nên đơn giản.

7. Khôi phục lại mảng (ANUMLA.*)

Nguồn tham khảo:

https://www.codechef.com/problems/ANUMLA

Mahesh có một mảng tuyệt đẹp tên A làm quà sinh nhật từ cô bạn gái xinh đẹp Nam-ra-tha. Có N số nguyên dương trong mảng đó. Mahesh yêu thích mảng này đến mức anh bắt đầu dành nhiều thời gian cho nó hàng ngày. Một ngày nọ, anh viết ra tất cả các tập con có thể có của mảng. Sau đó, với mỗi tập hợp con, anh ta tính tổng các phần tử trong tập hợp con đó và viết nó xuống một tờ giấy. Thật không may, Mahesh đã mất mảng đẹp đó. Anh ấy vẫn còn tờ giấy mà đã viết tất cả các tập hợp con. Nhiệm vụ của bạn là xây dựng lại mảng A đẹp giúp cặp đôi hạnh phúc này nhé.

Input: Dòng đầu tiên chứa số nguyên dương T ($T \leq 50$) là số bộ dữ liệu. Tiếp theo là T nhóm dòng, mỗi nhóm dòng mô tả một bộ dữ liệu với cấu trúc:

- Dòng đầu tiên chứa số nguyên dương ($1 \leq N \leq 15$)
- Dòng thứ hai chứa 2^N số nguyên là tổng của các tập con mà Nam ghi được.

Các số nguyên này có giá trị không vượt quá 10^9.

Ouptut:

Với mỗi bộ dữ liệu in ra trên một dòng số tìm được theo giá trị không giảm. Hai số trên một dòng ghi cách nhau một dấu trống.

Ví dụ:

ANUMLA.INP	ANUMLA.OU

	T
2 1 0 10 2 0 1 1 2	10 1

Gợi ý:

Đây là bài tập về việc sử dụng mutiset trong C++ (do có thể có nhiều tập con có tổng bằng nhau). Tư tưởng là với mỗi testcase ta sắp xếp lại tổng theo thứ tự tăng dần, rồi lặp N lần, mỗi lần lấy phần tử nhỏ nhất trong các tổng còn khả dụng làm phần tử tiếp theo. Xóa tất cả các tổng liên quan đến phần tử này... *Cụ thể:* giả sử tại bước thứ i ta đã chọn được $a[i]$ thì ta cần xóa tất cả các tổng được tạo thành từ tổ hợp của $a[i]$ với các tổ hợp khác rỗng của $\{a[1], a[2], \dots a[i-1]\}$. Phần tử bé nhất còn lại của mảng tổng nhập vào ban đầu sẽ là phần tử $a[i+1]$ của bước thứ $i+1$.

8. Karate (Karate.*)

Giải thi đấu võ Karate tại Thái Nguyên, cả nam lẫn nữ có tất cả n vận động viên được xếp thành một hàng ngang đánh số từ *1* đến n. Vận động viên thứ i có năng lực chiến đấu là một số nguyên a_i.

Trưởng ban tổ chức muốn chọn ra một số vận động viên đại diện cho tỉnh mình để tham gia giải đấu toàn quốc nhưng ông ta chưa biết chọn thế nào, đành cho các vận động viên của mình đấu với nhau theo nguyên tắc như sau:

- Chỉ thi đấu với nhau khi cùng giới tính.
- Mỗi vận động viên sẽ được đấu với các vận động viên đứng trước họ từ gần đến xa, đến khi nào thua trận thì thôi. (Vận động viên thứ i chỉ đấu thắng vận động viên thứ j nếu $a_i > a_j$).

Yêu cầu: Với mỗi vận động viên hãy cho biết họ thắng bao nhiêu vận động viên khác theo nguyên tắc trên.

Input: Đọc từ tệp karate.inp có cấu trúc như sau:

- Dòng 1: Chứa số nguyên dương n là số lượng võ sỹ nhà dế mèn.
- n dòng sau: Mỗi dòng chứa hai số

nguyên, dòng thứ *i+1* chứa số nguyên a_i, b_i là năng lực chiến đấu và giới

tính của võ sĩ *i* ($0 < a_i \le 10^9, 0 \le b_i \le 1$).

Output: Ghi vào tệp karate.out gồm *n* số nguyên số thứ *i* là số lượng vận động viên mà vận động viên thứ *i* sẽ được chiến đấu với vận động viên *i* thắng trận.

Ví dụ:

karate.inp	karate.out
10 9 0 18 1 11 0 12 0 3 0 12 1 2 0 2 1 7 1 6 0	0 0 1 2 0 0 0 0 1 2

Gợi ý:

Nhiệm vụ 1: Ta dùng thuật toán duyệt toàn bộ bằng 2 vòng lặp. Với mỗi võ sĩ thứ *i* ta đếm xem có bao nhiêu võ sỹ cùng giới tính, đứng ở bên trái từ gần đến xa cho đến khi gặp người có năng lực chiến đấu không nhỏ hơn a_i. Với độ phức tạp là $O(n^2)$

Nhiệm vụ 2:

- Ta dùng hàng đợi hai đầu, một đầu dành để lưu trữ năng lực chiến đấu của các võ sỹ nam, một đầu dành để lưu trữ năng lực chiến đấu của các võ sỹ nữ.

- Khi bổ sung vào hàng đợi ta phải loại bỏ các phần tử nhỏ hơn nó để tìm ra phần tử có giá trị lớn hơn hoặc bằng nó và gần nó nhất về bên trái.

- Độ phức tạp của thuật toán là O(n).

9. Liên thông mạnh (ltmanh.*)

Cho đồ thị G có hướng gồm N đỉnh, các đỉnh được đánh số từ 1 đến N.

Yêu cầu: Liệt các thành phần liên thông mạnh của đồ thị

Dữ liệu: tệp ltmanh.inp

- Dòng 1: số nguyên dương N (N <= 10^5)
- Các dòng tiếp theo mỗi dòng chứa 2 số nguyên u, v, thể hiện cung tối từ đỉnh u đến đỉnh v.

Kết quả: tệp ltmanh.out

- Các dòng đầu, mỗi dòng ghi các đỉnh thuộc cùng thành phần liên thông mạnh
- Dòng cuối ghi số thành phần liên thông mạnh của đồ thi.

Ví dụ:

Ltmanh.inp	**Ltmanh.out**
8 1 2 2 3 3 4 4 2 1 5 5 4 5 6 6 7 7 8 8 5	4 3 2 8 7 6 5 1 3

Hướng dẫn giải:

Sử dụng thuật toán Tarjan để giải bài toán. Với việc sử dụng vector đã làm cho việc biểu diễn đồ thị ở dạng danh sách kề trở nên đơn giản và ngắn gọn. Sử dụng cấu trúc dữ liệu Stack giúp việc cài đặt đơn giản hơn.

Chương trình tham khảo:

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

```
const int nmax = round(1e5);
int n, ti ,dem, number[nmax], low[nmax];
vector <int> adj[nmax];
stack <int> s;
bool dd[nmax];
void readf()
{
   cin>>n;
   int x,y;
   while (!cin.eof())
   {
      cin>>x>>y;
      adj[x].push_back(y);
   }
}
void dfs(int u)
{
   number[u] = ++ti; low[u] = nmax + 1;
   s.push(u);
   for(int i = 0; i< adj[u].size();i++)
   {
      int v = adj[u][i];
      if (dd[v] == false)
         if (number[v] != 0)
            low[u] = min(low[u],number[v]);
         else
         {
            dfs(v);
```

```
            low[u] = min(low[u],low[v]);
        }
    }
    if (number[u] <= low[u])
    {
        dem++;
        while (s.top() != u)
        {
            cout<<s.top()<<" "; dd[s.top()] = true;
            s.pop();
        }
        cout<<s.top()<<endl; dd[s.top()] = true;
        s.pop();
    }
}
int main()
{
    freopen("ltmanh.inp","r",stdin);
    freopen("ltmanh.out","w",stdout);
    readf();
    memset(dd,false,sizeof(dd));
    memset(number,0,sizeof(number));
    dem = 0;
    for(int i = 1;i <= n ; i++)
        if (dd[i] == false)
            dfs(i);
    cout<<dem;
    return 0;
```

```
}
```

II. MỘT SỐ BÀI TẬP THAM KHẢO

1. Công cụ sắp xếp kì lạ (SORTTOOL.*)

Những bài toán về sắp xếp tăng dần hay giảm dần theo giá trị của khóa cho trước đã trở nên quá đỗi quen thuộc với các bạn học sinh, để đỡ nhàm chán, thầy giáo giao cho học sinh bài tập xây dựng công cụ sắp xếp theo yêu cầu:

Cho dãy số có $N(1 \le N \le 10^5)$ số nguyên $a_1,a_2,...a_N$ $(|a_i| \le 10^9)$, hãy sắp xếp các số trên theo thứ tự giảm dần theo tần số xuất hiện, nếu có những số có cùng tần số xuất hiện thì số nào được xuất hiện trước thì sẽ xếp trước.

Input: Dòng đầu là số N; dòng tiếp theo chứa N số $a_1,a_2,...a_N$.

Output: Dãy được sắp xếp theo yêu cầu đã đưa ra.

Example:

SORTTOOL.INP	SORTTOOL.OUT
7 2 3 3 3 2 1 2	2 2 2 3 3 3 1
4 2 1 2 2	2 2 2 1

2. Chỗ ngồi trong nhà hát (SEATS.*)

Trong một nhà hát có N chỗ ngồi, chúng được xếp thành một hàng dài đánh số từ 1 đến N và từ trái qua phải. Ghế số 1 gần khán đài nhất và ghế số N là ghế xa nhất. Khi thấy phía trong nhà hát còn ghế trống thì nhân viên bán vé mới bán vé cho khán giả vào. Ban đầu tất cả các ghế đều trống, khách đầu tiên vào chắc chắn sẽ ngồi ghế trên cùng (ghế số 1). Mỗi khi có khán giả vào thêm, họ luôn chọn chỗ sao cho khoảng cách từ họ đến người gần nhất là xa nhất có thể. Nếu có nhiều chỗ như vậy thì họ chọn ghế có số thứ tự nhỏ nhất.

Trong suốt buổi hòa nhạc, nhân viên bán vé thấy có Q người ra và vào. Hỏi số ghế mỗi người vào là sau là số nào theo cách chọn chỗ như trên.

Input:

Dòng 1 là 2 số N,Q (N là số ghế; Q là số người ra, vào).

Q dòng tiếp theo mô tả người ra, người vào:

Nếu là (1) thì có người vào và cần tìm số ghế mà người đó chọn. Nếu là (2, i) thì là người thứ i đi ra khỏi nhà hát.

Biết rằng $1 \le N \le 10^{18}$;$1 \le Q \le 10^5$.

Output:

Gồm nhiều số tương ứng với số ghế của những người vào sau đã chọn **Example:**

sSEATS.INP	SEAT.OUT	Giải thích
2 7 1 1 2 1 1 2 2 2 3	1 2 1 1	Có 2 ghế và 7 lượt vào ra. *Người 1 vào, chọn ghế 1.* *Người 2 vào, chọn ghế 2.* Người 1 ra, ghế 1 trống. *Người 3 vào, chọn ghế 1.* Người 2 ra, ghế 2 trống. Người 3 ra, ghế 1

3. Đoạn con tổng 0 (SUMSEQ0.*)

Cho một dãy số nguyên gồm N phần tử: a_1,a_2,...a_n. Một đoạn con liên tiếp của dãy A có điểm đầu L, điểm cuối R với ($L \le R$) là tập hợp tất cả các phần tử a_i với ($L \le i \le R$). Đếm số đoạn con có tổng tất cả các phần tử bằng 0.

Input: Dòng đầu là số tự nhiên N.

Dòng thứ 2 là N số nguyên a_1,a_2,...a_N ($|a_i| \le 10^9$).

Output: Ghi số lượng đoạn con tìm được.

Example:

SUMSEQ0.INP	SUMSEQ0.OUT	Giải thích
5 2 1 -1 -2 0	4	Có 4 đoạn có tổng bằng 0 là: [2,3],[1,4],[1,5],[5,5]