SỞ GD & ĐT TỈNH NINH BÌNH
Trường THPT Chuyên Lương Văn Tụy

ĐỀ THI CHỌN HỌC SINH GIỎI THPT
CÁC TRƯỜNG CHUYÊN MIỀN DUYÊN HẢI ĐỒNG BẰNG BẮC BỘ
LẦN THỨ XIV – NĂM HỌC 2022 - 2023
(Đề thi đề nghị)

MÔN: TIN HỌC LỚP 10
(Thời gian làm bài 180 phút)

Tổng quan đề thi

Tên bài	Tên file	Dữ liệu vào	Dữ liệu ra	Điểm
Đếm dãy con liên tiếp	ASUM.*	ASUM.INP	ASUM.OUT	7
Xây dựng mảng	AD.*	AD.INP	AD.OUT	7
Độ dính kết	DDK.*	DDK.INP	DDK.OUT	6

Dấu * được hiểu phần mở rộng bài làm là pas, cpp hoặc py, thể hiện bài làm của học sinh sử dụng ngôn ngữ lập trình PASCAL, C++ hoặc PYTHON

Bài 1. Đếm dãy con liên tiếp

Cho dãy số A có n số nguyên $a_1, a_2, \dots, a_n$. Một dãy con liên tiếp các số hạng của dãy A là dãy các số hạng từ số hạng a_i đến số hạng a_j $(1 \le i \le j \le n)$. Hãy cho biết dãy A có bao nhiêu dãy con liên tiếp mà giá trị tuyệt đối của tổng các số hạng trong dãy con đó lớn hơn một số nguyên dương S cho trước.

Dữ liệu:

- Dòng thứ nhất chứa hai số nguyên dương n và S $(n \le 10^5, S \le 10^{14})$.
- Dòng thứ hai chứa n số nguyên $a_1, a_2, .., a_n$ $(|a_i| \le 10^9)$.

Hai số liên tiếp trên cùng dòng được ghi cách nhau bởi dấu cách.

Kết quả:

Một số nguyên duy nhất là số dãy con liên tiếp thỏa mãn yêu cầu của bài toán.

Ví dụ:

ASUM.INP	ASUM.OUT
4 4 5 -1 8 -5	6
10 7 -4 9 2 -11 -3 8 -6 5 -3 1	12

Giải thích: *Trong ví dụ đầu tiên có 6 dãy con thỏa mãn yêu cầu là*: $\{5\}, \{8\}, \{-5\}, \{-1; 8\}, \{5; -1; 8\}$ và $\{5; -1; 8; 5\}$.

Ràng buộc:

- Có 50% số test ứng với 50% số điểm của bài có $n \leq 100$.
- Có 30% test khác ứng với 30% số điểm của bài có $n \leq 10^3$.
- Có 20% test còn lại ứng với 20% số điểm của bài có $n \leq 10^5$.

Bài 2. Xây dựng mảng

Cho một mảng gồm n phần tử A[1], A[2], …, A[n]. Các phần tử là các số nguyên trong nằm trong đoạn [0, m] với m là số cho trước. ($1 \leq n \leq 10^5$, $1 \leq m \leq 100$)

Đếm số cách thay số 0 có trong mảng bằng các số nằm trong đoạn [1, m] sao cho chênh lệch tuyệt đối giữa hai giá trị liền kề nhiều nhất là 1.

Dữ liệu:

Dòng đầu vào đầu tiên có hai số nguyên n và m: kích thước mảng và giới hạn trên cho mỗi giá trị.

Dòng tiếp theo có n số nguyên A[1], A[2], …, A[n].

Kết quả:

In một số nguyên: số lượng dãy chia lấy dư cho 10^9+7.

AD.INP	AD.OUT	Giải thích
3 5 2 0 2	3	Các dãy [2,1,2], [2,2,2], [2,3,2] thoả mãn yêu cầu.

Ràng buộc:

- 50% số test tương ứng với 50% số điểm chỉ có 1 số có giá trị bằng 0 trong dãy A.
- 50% số test tương ứng với 50% số điểm không có ràng buộc gì thêm.

Bài 3. ĐỘ DÍNH KẾT

Đất nước Happy có N thành phố được nối với nhau bởi M đường nối hai chiều. Hệ thống các con đường được thiết lập sao cho giữa 2 thành phố bất kỳ luôn có một đường đi bao gồm một hoặc nhiều con đường trực tiếp giữa hai thành phố. Mỗi con đường thực hiện việc di chuyển giữa hai thành phố theo cả hai chiều.

Trung tâm điều khiển của đất nước đưa ra khái niệm độ dính kết giữa cặp hai thành phố A và B được xác định như là số lượng các con đường mà việc bỏ đi một trong số chúng (các con đường khác vẫn thực hiện bình thường) dẫn đến không thể đi từ thành phố A đến thành phố B.

Một nghiên cứu cho biết rằng, trong điều kiện thời tiết xấu, tổng độ dính kết giữa các cặp thành phố phải đạt đến một giá trị nhất định thì hệ thống đường đi mới được gọi là an toàn.

Yêu cầu: Hãy tính tổng độ dính kết giữa mọi cặp thành phố.

Dữ liệu:

- Dòng đầu tiên chứa hai số nguyên N (1≤N≤300000) và M (1≤M≤300000)
- Mỗi dòng trong số N dòng tiếp theo chứa thông tin về một con đường, bao gồm hai số nguyên dương trong khoảng từ 1 đến N: chỉ số của hai thành phố được nối bởi một con đường.

Kết quả:

In ra 1 số nguyên duy nhất là tổng độ dính kết giữa mọi cặp thành phố (A, B) (với A<B).

Ví dụ:

DDK.INP	**DDK.OUT**
5 5 1 2 4 2 4 5 3 2 3 1	**10**

- 30% số test tương ứng 30% số điểm có n<100.
- 30% số test tương ứng 30% số điểm có n<10000.
- 40% số test tương ứng với 40% số điểm không có dàng buộc gì thêm.

----------Hết----------

Hướng dẫn Thuật toán:

Bài 1:

Sub 1: Duyệt mọi đoạn con từ i đến j. Với mỗi đoạn tính tổng a[i] + a[i+1] +..+a[j] và kiểm tra tổng đoạn con có thoả mãn điều kiện hay ko.

Độ phức tạp $O(n^3)$

Sub 2: Gọi F[i] là tổng a[1] + a[2] + … + a[i].

Tính tổng đoạn con từ i đến j bằng cách sử dụng mảng tổng cộng dồn F.

a[i] + a[i+1] +..+a[j] = F[j] – F[i-1]

Độ phức tạp $O(n^2)$

Sub 3: Sắp xếp mảng F theo thứ tự tăng dần.

Kết quả bài toán là tổng số lượng các F[j] > F[i] + k với (1 <= i < j<=n) (với mỗi i sử dụng upper_bound để tìm j nhỏ nhất thoả mãn F[j] >= F[i] + k)

```
#include <bits/stdc++.h>
using namespace std;
long long n,f[1000005],a[1000005],k,kq,x,vt;
map <int,int> m;
int main()
{
    freopen("asum.inp","r",stdin);
    freopen("asum.out","w",stdout);
    cin>>n>>k;
    for (int i=1;i<=n;i++)
    {
        cin>>x;
        a[i]=x;
        f[i]=f[i-1]+a[i];
    }
    sort(f,f+1+n);
    for (int i=0;i<=n;i++)
    {
        vt=upper_bound(f+i,f+1+n,k+f[i])-f;
        if (vt<n+1) kq+=n-vt+1;
    }
```

```
    cout<<kq;
}
```

Bài 2:

Sub 1: Tìm vị trí x[i] = 0. Có 3 trường hợp xảy ra

x[i] = x[i-1] hoặc x[i] = x[i-1] + 1 hoặc x[i] = x[i-1] – 1

Với mỗi trường hợp kiểm tra điều kiện | x[i] – x[i+1] | <= 1 và x[i] có nằm trong đoạn 1..m hay ko

Sub 2: Quy hoạch động:

Gọi `F[i][j]` là số cách thay số 0 từ vị trí 1 đến vị trí i thoả mãn điều kiện đề bài và thoả mãn `a[i] = j`

Khởi tạo:

Nếu `x[1] = 0` thì `F[1][j] = 1;` với `j = 1..m`

Nếu `x[1] != 0` thì `F[1][x[1]] = 1; F[1][j] = 0` với `j = 1..m`

QHĐ

Xét phần tử x[i],

TH1: Nếu `x[i] = 0`. Ta thay thế phần tử `x[i] = j` thì phần tử `x[i-1]` phải bằng `j-1` hoặc `j` hoặc `j+1`

⇨ `f[i][j] = f[i-1][j-1] + f[i-1][j] + f[i-1][j+1]`

TH2: Nếu `x[i] != 0` thì phần tử `x[i-1]` phải bằng `x[i]-1` hoặc `x[i]` hoặc `x[i]+1`

⇨ `f[i][x[i]] = f[i-1][x[i]-1] + f[i-1][x[i]] + f[i-1][x[i]+1]`

Kết quả của bài toán là tổng các `f[n][j]` với `j = 1..m`

Code mẫu:

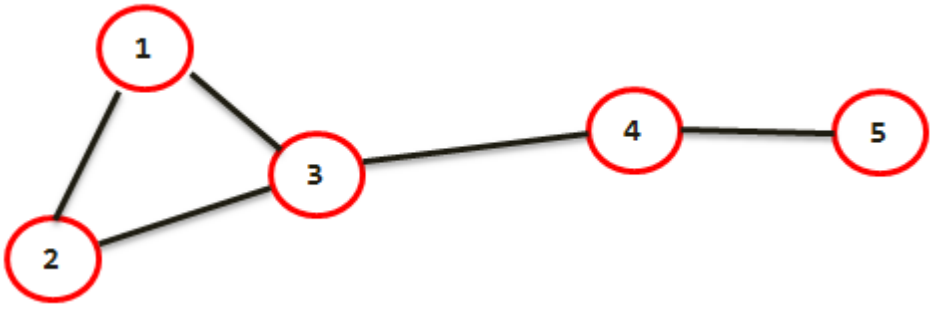
```
#include <bits/stdc++.h>

using namespace std;

#define ll long long

int n, m, x[100001];

int f[100001][101];
```

```
int main()
{
    ios_base::sync_with_stdio(0); cin.tie(0); cout.tie(0);
    freopen("AD.inp", "r", stdin);
    freopen("AD.out", "w", stdout);
    cin >> n >> m;
    for (int i=1; i<=n; i++) cin >> x[i];

    int i=1;
    if (x[1] == 0)
    {
        for (int j=1; j<=m; j++)
            {
                f[i][j] = 1;
            }
    }
    else f[i][x[1]] = 1;;

    int mod = 1e9+7;
    for (int i=2; i<=n; i++)
    {
        if (x[i] == 0)
        {
            for (int j=1; j<=m; j++)
            {
                f[i][j] = (f[i][j] + f[i-1][j-1]) % mod;
                f[i][j] = (f[i][j] + f[i-1][j+1]) % mod;
```

```
                f[i][j] = (f[i][j] + f[i-1][j]) % mod;
            }
        }
        else
        {
            f[i][x[i]] = (f[i][x[i]] + f[i-1][x[i]-1]) % mod;
            f[i][x[i]] = (f[i][x[i]] + f[i-1][x[i]+1]) % mod;
            f[i][x[i]] = (f[i][x[i]] + f[i-1][x[i]]) % mod;
        }
    }
    int kq = 0;
    for (int j=1; j<=m; j++)
        kq = (kq + f[n][j]) % mod;
    cout << kq;
}
```

Bài 3:

Với mỗi cặp đỉnh `(u,v)` ta chỉ quan tâm các cạnh là **cầu** của đồ thị, xem từ `u` có thể đến được `v` mà cần qua bao nhiêu cầu. Với mỗi cạnh (u,v) là cầu của đồ thị, giả sử `u` là cha của `v`, số cặp đỉnh cần qua cầu này là `f[v] * (n - f[v])` với `f[v]` là số đỉnh trong cây DFS gốc `v`.



Hình trên `(3,4)` là **cầu**, số cặp lúc đó là `3*2`. (`f[4] = 2`, `4` là con `3`)

`f[u]` sẽ được tính bằng QHĐ trong lúc DFS.

Độ phức tạp $O(N+M)$.

```
#include <bits/stdc++.h>
using namespace std;
#define maxn 500009
int n,m;
vector<int> a[maxn];
int par[maxn];
int id[maxn];
int cnt = 0;
int cau = 0;
int low[maxn];
long long f[maxn]; // f[i] =  số đỉnh trong cây DFS gốc v.
long long KQ = 0;
void DFS(int u){
    f[u] = 1;
    int child;
    if (par[u] == -1) child = 0;
    else child = 1;

    low[u] = id[u] = ++cnt;
  //  cout << u << " " << id[u] << endl;
    for (auto v:a[u])
    if (v != par[u])
        if (par[v] == 0  ) // v chưa thăm
        {
            par[v] = u;
            DFS(v);
```

```
            f[u] = f[u]+f[v];
            low[u] = min (low[u], low[v]);
            // kiểm tra u - v có phải là cầu ?
            if (low[v] == id[v]) {
                cau++;
                KQ = KQ + (long long) f[v] * (n - f[v]);
            }
            // cout << u << " " << v << endl
            if (low[v] >= id[u]){  // sửa
                child++; // tăng số nhánh con gốc u
            }
        }
        else    // v đã thăm
            low[u] = min (low[u], id[v]);
}

main()
{
    ios_base::sync_with_stdio(0); cin.tie(0); cout.tie(0);
    freopen("DDK.inp", "r", stdin);
    freopen("DDK.out", "w", stdout);
    cin >> n >> m;
    for (int i=1; i<=m; i++)
    {
        int u,v;
        cin >> u >> v;
        a[u].push_back(v);
```

```
        a[v].push_back(u);
    }
    for (int i=1; i<=n; i++)
    if (par[i] == 0)
    {
        par[i] = -1; // i là gốc
        DFS(i);
    }
    cout << KQ;
}
```