CHUYÊN ĐỀ: MỘT SỐ BÀI TẬP VỀ ĐỒ THỊ

Tổ Tin học – Trường THPT chuyên Lam Sơn Thanh Hóa

Bài 1. Distribute

Morioh là một thành phố lớn gồm N điểm du lịch và N – 1 con đường nối từ điểm du lịch này sang điểm du lịch khác sao cho từ một điểm du lịch bất kì có thể đi được tới tất cả các điểm còn lại.

Josuke được ngài thị trưởng giao trách nhiệm đặt các trạm thu phí trên mỗi con đường sao cho chi phí của mỗi con đường là một số nguyên dương và số lượng con đường có chi phí là 1 phải nhỏ nhất có thể. Josuke rất thích số K nên tích tất cả các chi phí trên mỗi con đường phải bằng K. Vì K là một số rất lớn nên K sẽ được biểu diễn sang tích M thừa số nguyên tố $p_1, p_2, \dots p_M$. ($p_1 \cdot p_2 \cdot \dots \cdot p_M = K$). Chi phí của đường đi từ điểm du lịch u đến điểm du lịch v là tổng tất cả các chi phí trên đường đi từ u đến v. Gọi d(i, j) là chi phí để đi từ i đến j. Hãy giúp Josuke phân phối các chi phí sao cho tổng chi phí để di chuyển giữa các điểm du lịch phải lớn nhất có thể. Hay nói cách khác : $\sum_{i}^{n-1} \sum_{j}^{n} d(i,j)$ phải lớn nhất có thể.

***Dữ liệu*:** Vào từ file DISTRIBUTE.INP gồm:

- Dòng đầu tiên gồm số N ($N \le 10^5$)
- N – 1 dòng sau, dòng thứ i gồm 2 số u_i và v_i ($1 \le u_i, v_i \le n$), tức là có đường đi từ điểm du lịch u_i đến điểm du lịch v_i
- Dòng tiếp theo gôm số M ($M \le 6 \cdot 10^4$)
- Dòng tiếp theo gồm M số $p_1, p_2, \dots, p_M$ ($2 \le p_i \le 6 \cdot 10^4$)

***Kết quả*:** Ghi ra file DISTRIBUTE.OUT gồm 1 dòng là kết quả tìm được. Vì kết quả rất lớn nên hãy lấy kết quả mod 1000000007.

***Ví dụ*:**

DISTRIBUTE.INP	DISTRIBUTE.OUT
4 1 2 2 3	17

3 4 2 2 2	
7 6 1 2 3 4 6 7 3 5 1 3 6 4 7 5 13 3	286

Thuật toán:

- Có thể thấy thành phố có dạng cây, ta sẽ đặt đỉnh 1 làm gốc.
- Gọi cnt[i] là số đường đi giữa 2 đỉnh bất kì mà đi qua cạnh thứ i.

Chúng ta có thể dễ dàng tính được cnt[i] bằng cách:

cnt[i] = f[v_i] * (n – f[v_i]) với cạnh thứ i nối điểm u_i và v_i và f[u] là số đỉnh thuộc cây con gốc u.

- Để thu được tổng chi phí lớn nhất, chúng ta sẽ cho cạnh có nhiều đường đi qua nhất chi phí lớn nhất. Gọi c_1, c_2, …, c_{N-1} là chi phí đặt cho cạnh thứ 1, 2, … N – 1. Vì yêu cầu số lượng c_i = 1 phải nhỏ nhất có thể, nên đến đây chúng ta sẽ có 2 trường hợp:

+ TH1: M <= N – 1

Với trường hợp này, chúng ta sẽ đặt cạnh có nhiều đường đi qua nhất chi phí p_i lớn nhất, cứ làm như vậy đến khi nào không đặt được nữa thì những cạnh còn lại sẽ có c_i = 1.

+ TH2 : M > N – 1

Sort mảng p lại, chúng ta sẽ đặt cạnh có ít đường đi qua nhất là p_1 , ít thứ nhì là p_2 , … cứ làm như vậy với N-2 cạnh ít đường đi qua nhất. với cạnh nhiều đường đi qua nhất thì ta sẽ cho c = p_{N-1} * p_N * … * p_M

Kết quả sẽ là $\sum_{i=1}^{N-1} cnt[i] * c[i]$

Code tham khảo:

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll mod = 1e9 + 7;
int n, m;
vector<int> adj[100005];
ll f[100005];
vector<ll> cnt;
ll ans[100005];
void dfs(int u, int pa){
    f[u] = 1;
    for(int v : adj[u]){
        if(v == pa) continue;
        dfs(v, u);
        f[u] += f[v];
        cnt.push_back(f[v] * (n - f[v]));
    }
}
int main()
{
    ios_base::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);
    freopen("DISTRIBUTE.inp", "r", stdin);
    freopen("DISTRIBUTE.out", "w", stdout);
    cin >> n;
    ll res = 0;
    for(int i = 1; i < n; i++){
        int u, v;
        cin >> u >> v;
        adj[u].push_back(v);
        adj[v].push_back(u);
    }
    dfs(1, 1);
    sort(cnt.begin(), cnt.end());
    cin >> m;
    vector<ll> p(m);
    for(int i = 0; i < m; i++)
        cin >> p[i];
    sort(p.begin(), p.end());
    if(m >= n - 1){
```

```
        for(int i = 0; i < n - 2; i++)
            res = (res + cnt[i] * p[i]) % mod;
        ll tmp = cnt[n - 2];
        for(int i = n - 2; i < m; i++)
            tmp = (tmp * p[i]) % mod;
        res = (res + tmp) % mod;
    }
    else{
        reverse(cnt.begin(), cnt.end());
        reverse(p.begin(), p.end());
        for(int i = 0; i < m; i++)
            res = (res + cnt[i] * p[i]) % mod;
        for(int i = m; i < n - 1; i++)
            res = (res + cnt[i]) % mod;
    }
    cout << res;
    return 0;
}
```

Test tham khảo: https://drive.google.com/file/d/1A5chcSxf7s3lSbuItFJ8KNHcGm_00IIu/view?usp=sharing

Bài 2. KPATH

Vương quốc Belzerg là một vương quốc vô cùng rộng lớn với N thị trấn và M con đường 1 chiều nối giữa các thị trấn. Con đường thứ i ($1 \leq i \leq M$) nối thị trấn u_i đến thị trấn v_i.

Kazuma là một anh hùng nổi tiếng của thị trấn Axel với biết bao chiến công lẫy lừng. Một ngày nọ anh vào cửa hàng để mua một thanh kiếm mới và đặt tên nó sao thật ngầu, nhưng chưa kịp đặt tên thì một đồng bọn của anh là Megumin đã lấy thanh kiếm và đặt ngay cái tên Chunchunmaru cho nó. Megumin ra điều kiện cho anh nếu muốn lấy lại và đổi tên thanh kiếm: cho anh K ngày, mỗi ngày anh phải đi từ một thị trấn đến thị trấn khác bằng những con đường đã cho và kiếm được nhiều đồng Eris nhất có thể bằng cách chiến đấu với quái vật trên các con đường (chiến đấu với quái vật trên con đường thứ i sẽ được w_i đồng Eris, quái vật sẽ hồi sinh sau mỗi ngày). Megumin cũng ra điều kiện rằng Kazuma phải đi chính xác K con đường không thì cô ấy sẽ dùng phép "EXPLOSION!!!" lên thanh kiếm mới của anh, cô ấy cũng thấy rất tội lỗi nên cho phép Kazuma xuất phát từ thị trấn bất kì và được đi lại các thị trấn cũng như các đường mà anh đã đi qua. Kazuma

rất muốn lấy lại và đổi tên thanh kiếm mới này vì anh đã để dành biết bao nhiêu tiền từ các nhiệm vụ để mua được nó.

Kazuma muốn nhờ các bạn tìm một lộ trình đi qua chính xác K con đường trong K ngày ấy sao cho tổng số đồng Eris kiếm được là nhiều nhất có thể nhé!

Dữ liệu: Vào từ file KPATH.inp gồm:

- Dòng đầu gồm 3 số N, M, K ($1 \le N \le 100$, $1 \le M \le 10000$, $1 \le K \le 10^9$).
- M dòng tiếp theo, dòng thứ i gồm bộ ba số u_i, v_i, w_i trong đó

($1 \le u_i, v_i \le N$) và $1 \le w_i \le 10^9$.

Kết quả: Ghi ra file KPATH.out gồm: 1 dòng duy nhất là tổng số đồng Eris kiếm được nhiều nhất trên lộ trình ấy, nếu không có lộ trình nào thỏa mãn thì in ra -1.

Ví dụ:

KPATH.INP	KPATH.OUT
4 4 6 1 2 10 2 3 3 3 4 3 4 2 3	25
4 5 4 1 2 10 2 3 3 3 4 3 1 4 5 2 4 7	-1

Ràng buộc:

- 40% số test có $K \le 100$.
- 60% còn lại không có ràng buộc gì thêm.

Thuật toán:

- Với K nhỏ:

Ta có thể dùng qhđ F[i][j] là đường đi có số tiền nhận được lớn nhất khi đến đỉnh i và đã đi qua j cạnh. Kết quả sẽ là $\sum_{i=1}^{N} F[i][k]$

- K lớn

Ta sẽ sử dụng nhân ma trận với ma trận trung gian có dạng f[i][j] = -1 với mọi i, j ≤ N, $f[u_i][v_i] = w_i$ với 1 ≤ i ≤ M.

Nhân 2 ma trận a, b thì ta duyệt tất cả các đỉnh u, v cho ma trận kết quả rồi cập nhật f[u][v] = max(a.f[u][t] + b.f[t][v]) với t là đỉnh trung gian để đi u->t->v và f[u][t] với f[t][v] phải khác -1. Từ đó ta được f[u][v] là đường đi nhận được số tiền lớn nhất từ u->v. Ma trận kết quả cuối cùng sẽ là (Ma trận trung gian)^k, đặt là res thì kết quả của bài toán là max(res[u][v]) với mọi 1 ≤ u, v ≤ N.

Code tham khảo:

```
#include <bits/stdc++.h>
#define fi first
#define se second
using namespace std;
typedef long long ll;
const int maxn = 105;
int n, m;
struct matrix{
   ll f[maxn][maxn];
   matrix(){
      for(int i = 1; i < maxn; i++)
         for(int j = 1; j < maxn; j++)
         f[i][j] = - 1;
   }
};
matrix Mul(matrix a, matrix b){
   matrix c;
   for(int u = 1; u <= n; u++)
      for(int v = 1; v <= n; v++)
         for(int t = 1; t <= n; t++)
         if(a.f[u][t] > 0 && b.f[t][v] > 0)
         c.f[u][v] = max(c.f[u][v], a.f[u][t] + b.f[t][v]);
   return c;
}
matrix Pow(matrix x, int k){
   matrix tmp = x; k--;
   while(k){
      if(k&1) tmp = Mul(tmp, x);
```

```
        x = Mul(x, x);
        k >>= 1;
    }
    return tmp;
}
int main()
{
    ios_base::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);
    freopen("KPATH.inp", "r", stdin);
    freopen("KPATH.out", "w", stdout);
    int k;
    cin >> n >> m >> k;
    matrix base;
    for(int i = 1; i <= m; i++){
        int u, v;
        ll w;
        cin >> u >> v >> w;
        base.f[u][v] = max(base.f[u][v], w);
    }
    base = Pow(base, k);
    ll res = -1;
    for(int u = 1; u <= n; u++)
        for(int v = 1; v <= n; v++)
            res = max(res, base.f[u][v]);
    cout << res;
    return 0;
}
```

Test tham khảo: https://drive.google.com/file/d/1QEvaiFpIwgLBrIW1wUIgeX-5hZQSC8if/view?usp=sharing

Bài 3. ROAD

Bác VA có N đồng cỏ, được kết nối với nhau bởi N-1 đoạn đường 2 chiều (có cấu trúc giống 1 cây). Lần này, vì quá già yếu, bác muốn chia N-1 đoạn đường thành nhiều con đường 2 chiều dài hơn bằng cách ghép một số đoạn đường 2 chiều lại với nhau và giao cho những người làm thuê của mình quản lí. Để tìm người quản lí xứng đáng thay thế mình, bác quyết định mỗi con đường phải có cùng độ dài. Bác đang tự hỏi độ dài mỗi con đường bằng bao nhiêu.

Nhưng bác VA lại nghĩ ra 1 vấn đề khác, đó là với mỗi độ dài K thỏa mãn 1<=K<=N-1, liệu có thể chia N-1 đoạn đường thành các con đường độ dài K hay không. Biết mỗi đoạn đường có độ dài là 1.

Dữ liệu: Vào từ file ROAD.INP gồm

+ Dòng đầu tiên là số N (2<=N<=100000)

+ N-1 dòng tiếp theo gồm 2 số u và v (u khác v) cho biết tồn tại đoạn đường từ u đến v. (1<=a,b<=N).

Kết quả: Ghi ra file ROAD.OUT gồm N-1 số a[1],a[2],…,a[n-1] một cách liên tiếp. Trong đó a[i]=1 nếu tồn tại cách chia N-1 đoạn đường thành các con đường lớn có độ dài là i, a[i]=0 nếu không chia được.

Ví dụ :

ROAD.INP	ROAD.OUT
13 1 2 2 3 2 4 4 5 2 6 6 7 6 8 8 9 9 10 8 11 11 12 12 13	111000000000

Giải thích:

Thỏa mãn để chia thành các con đường độ dài 1,2 hoặc 3.

Với độ dài K=3 thì có thể chia thành các con đường gồm các đoạn đường nhỏ sau :

+ (13−12−11−8)

+(10−9−8−6)

+(7−6−2−3)

+(5−4−2−1)

Ràng buộc:

Subtask1 : 30% số điểm có N<=1000

Subtask2: 70% số điểm không có rằng buộc gì thêm

Thuật toán:

Ban đầu, ta thấy nếu độ dài K sẽ không thỏa mãn nếu (N-1) không chia hết cho K.

Do đó, ta sẽ sử dụng DFS. Với mỗi đỉnh x, ta gọi child[x] là tổng các con của cây con gốc x và lưu 1 vector chỉ số lượng child [v] với v là con của x.

Tiếp theo, với mỗi đỉnh u, ta chia cây con gốc u thành một số con đường lớn độ dài là K và 1 con đường phụ độ dài <K và kết thúc tại u. Sau đó ta chỉ cần xử lý ghép các con bất kì của mỗi đỉnh xem có thỏa mãn yêu cầu đề bài hay không. Nếu tất cả các đỉnh đều thỏa mãn, in ra 1, ngược lại in ra 0.

Code tham khảo:

```
#include "bits/stdc++.h"
using namespace std;
#define f first
#define s second
const int mod = 1e9+7;
const int N = 1e5+5;
int n,child[N],cur[N];
vector<int> ke[N],num[N];
bool bad=0;
void dfs(int u, int cha) {
        child[u]=1;
        for(int i=0;i<ke[u].size();i++){
           int v=ke[u][i];
    if (v==cha) continue;
                dfs(v,u);
                child[u]+=child[v];
                num[u].push_back(child[v]);
        }
        if (child[u]!=n) num[u].push_back(n-child[u]);
}
bool ok(int len) {
```

```
        if ((n-1)%len!=0) return 0;
        for (int i=0;i<len;i++) cur[i]=0;
        for (int i=1;i<=n;i++) {
                int cnt=0;
                for (int j=0;j<num[i].size();j++) {
        int t=num[i][j];
                        int z=t%len;
        if (z==0) continue;
                        if (cur[len-z]) cur[len-z]--, cnt--;
                        else cur[z]++, cnt++;
                }
                if (cnt) return 0;
        }
        return 1;
}
int main() {
   ios_base::sync_with_stdio(0);
   cin.tie(0);
   cout.tie(0);
   freopen("road.inp", "r", stdin);
   freopen("road.out", "w", stdout);
        cin>>n;int a,b;
        for (int i=1;i<n;i++) {
      cin>>a>>b;
      ke[a].push_back(b);
      ke[b].push_back(a);
        }
        dfs(1,0);
        for (int i=1;i<n;i++) {
                if (ok(i)) cout<<1;
                else cout<<0;
        }
}
```

Test tham khảo: *https://drive.google.com/file/d/1aSgJ5o9P4d_0KrFfCNryZCX7LA_F9cHK/view?usp=sharing*

Bài 4: MKVISIT

Bác VA có N đồng cỏ , được kết nối bởi N-1 con đường (có cấu tạo giống như 1 cây). Mỗi đồng cỏ có một con bò được đánh số may mắn, đồng cỏ thứ i có con bò được đánh số là a[i] (1<=a[i]<=N).

Bác VA là một người rất thân thiện và hay tổ chức mời bạn bè đến thăm những đồng cỏ của mình. Bác có M người bạn đến thăm. Với người bạn thứ i, bác sẽ dẫn họ đi từ đồng cỏ Ai đến đồng cỏ Bi (Ai có thể bằng Bi), sao cho trên quãng đường đó không có đồng cỏ nào được thăm quá 1 lần. Họ sẽ thăm những con bò trên mỗi cánh đồng đi qua. Mỗi người bạn của bác Va rất yêu thích toán học, họ quyết định có cho mình 1 con số may mắn, người thứ i có số may mắn là x[i] . Nếu 1 người bạn của bác đi qua cánh đồng chứa con bò có số may mắn của họ, họ sẽ được bác VA tặng 1 món quà.

Hãy chỉ ra xem trong M người bạn của bác VA, ai sẽ được tặng quà.

Dữ liệu: ào từ file MKVISIT.INP gồm

+ Dòng đầu tiên là 2 số N và M

+ Dòng thứ 2 gồm N số : a[1],a[2],..,a[n]. Với a[i] là số may mắn của con bò tại đồng cỏ i.

+ N-1 dòng tiếp theo gồm 2 số u và v (1<=u,v<=N) là đường nối từ cánhđồng u đến cánh đồng v

+ M dòng tiếp theo , dòng thứ i gồm 3 số Ai, Bi, Ci. Với Ai và Bi là điểm đầu và điểm cuối trong hành trình thăm đồng cỏ của người bạn thứ i, và Ci là số may mắn của người đó. (1<=Ci<=N)

Kết quả: Ghi ra file MKVISIT.OUT là xâu nhị phân gồm i phần tử, phần tử thứ i chỉ '1' nếu người bạn thứ i được tặng quà và chỉ '0' nếu người bạn đó không được tặng.

Ví dụ:

MKVISIT.INP	MKVISIT.OUT
5 5 1 1 2 1 2 1 2 2 3 2 4 1 5 1 4 1 1 4 2 1 3 2 1 3 1	10110

5 5 1	

***Ràng buộc*:**

subtask 1: 30% số điểm

+ 1<=a[i],x[i]<=2

+ N<=2000, M<=2000

Subtask2 : 70% số điểm

+ N<=100000, M<=100000

+ a[i],x[i]<=N;

Thuật toán:

Với bài toán này, chúng ta sẽ sử dụng xử lí offline cho tất cả truy vấn. Ban đầu , chúng ta tính LCA của 2 điểm A và B của mỗi truy vấn. Sau đó ta duyệt toàn bộ cây bằng DFS, duy trì 1 stack là color[x] tương ứng là số đỉnh có màu x trong lúc duyệt. Khi mỗi điểm được đi qua, chỉ cần xét xem đường dẫn từ 1 điểm đến LCA đó có chứa màu này hay không,. Việc đánh giá này rất đơn giản. Vì trong các stack, các đỉnh đều nằm trên đường từ đỉnh gốc đến đỉnh đang xét, nên chỉ cần lấy điểm top của ngăn xếp và so sánh chiều sâu của nó với LCA đang xét . Nếu nó lớn hơn hoặc bằng thì tức là đường đi từ Ai đến Bi sẽ đi qua màu đó, và đáp án là 1.

Code tham khảo:

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
#define tkn "mkvisit"
#define bignum vector<int>
#define ii pair<int,int>
#define X first
#define Y second
const int N=100500;
const int M=6e4+7;
const ll base=311;
const ll mod=1e9+7;
int n,m,tt=20;
int f[N][30],t[N],pa[N],h[N],c[N],res[N],root[N];
vector<int> ke[N],qid[N];
```

```
stack<int> col[N];
void init(){
    memset(f,-1,sizeof(f));
    for (int i=1;i<=n;i++)
        f[i][0]=pa[i];
    for (int j=1;j<=tt;j++)
        for (int i=1;i<=n;i++){
            if (f[i][j-1]!=-1)
                f[i][j]=f[f[i][j-1]][j-1];
        }
}
void dfs(int u,int cha,int hi){
    h[u]=hi;
    for (int i=0;i< ke[u].size();i++){
        int v=ke[u][i];
        if (v==cha) continue;
        if (h[v]) continue;
        pa[v]=u;
        dfs(v,u,hi+1);
    }
}
int lca(int u,int v){
    if (h[u]<h[v]) swap(u,v);
    int tmp=0;
    while((1<<tmp)<=h[u]) tmp++;
    tmp--;
    if (tmp<0) tmp=0;
    for (int i=tmp;i>=0;i--){
        if (h[u]-h[v]>=(1<<i)) u=f[u][i];
    }
    if (u==v) return u;
    for (int i=tmp;i>=0;i--){
        if (f[u][i]!=-1&&f[u][i]!=f[v][i]){
            u=f[u][i];
            v=f[v][i];
        }
    }
    return pa[u];
}
void sol(int x,int cha){
```

```
    col[t[x]].push(x);
    for (int i = 0; i < qid[x].size(); ++i) {
        int id = qid[x][i];
        if (col[c[id]].size())
            res[id] |=h[col[c[id]].top()]>=h[root[id]];
    }
    for (int i=0;i< ke[x].size();i++ ){
        int v=ke[x][i];
        if (v==cha) continue;
        sol(v,x);
    }
    col[t[x]].pop();
}
int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    freopen(tkn".inp", "r", stdin);
    freopen(tkn".out", "w", stdout);
    cin>>n>>m;
    for (int i=1;i<=n;i++) cin>>t[i];
    for (int i=1;i<n;i++){
        int u,v;
        cin>>u>>v;
        ke[u].push_back(v);
        ke[v].push_back(u);
    }
    h[1]=1; pa[1]=-1;
    dfs(1,-1,1);int a,b;
    init();
    for (int i=1;i<=m;i++){
        cin>>a>>b>>c[i];
        root[i]=lca(a,b);
        qid[a].push_back(i);
        qid[b].push_back(i);
    }
    sol(1,0);
    for (int i=1;i<=m;i++) cout<<res[i];
    return 0;
```

}

Test tham khảo: *https://drive.google.com/file/d/1Ei2xCo-iQlZ7F6k_CR0ukCFPQiVOzmVV/view?usp=sharing*

Bài 5: MOOD

Có n thành phố và n-1 con đường 2 chiều nối trực tiếp 2 thành phố với nhau. Người ở một thành phố bất kì đều có thể đến thành phố khác thông qua n-1 con đường này. Các thành phố được đánh số từ 1 đến n và thành phố 1 là thủ đô. Nói cách khác, quốc gia có cấu trúc dạng cây.

Có m công dân sống ở n thành phố. Có p_i công dân sinh sống tại thành phố i nhưng tất cả bọn họ đang làm việc tại thủ đô. Đến tối, các công dân quay về thành phố của mình bằng con đường ngắn nhất.

Mỗi người có một tâm trạng khác nhau. Một vài người quay về trong tâm trạng tốt, số còn lại quay về trong tâm trạng xấu. Hơn nữa, một người cũng có thể phá hủy tâm trạng của chính mình trên đường về thành phố. Nếu một người đang trong tâm trạng xấu, anh ta sẽ không thể cải thiện được nó.

Mỗi thành phố có một chỉ số hạnh phúc h_i, được tính bằng cách lấy số người đến thành phố trong tâm trạng tốt trừ đi số người đến thành phố trong tâm trạng xấu. Mỗi thành phố được lắp đặt một cỗ máy hiện đại, cho biết tâm trạng của mỗi người đến thành phố. Một người sẽ không thay đổi tâm trạng của mình trong thành phố.

Cỗ máy này vẫn đang trong giai đoạn phát triển vì vậy sẽ có khả năng xảy ra sai sót khi đánh giá tâm trạng của một người. Vào một buổi đêm muộn, khi mọi công dân đã về nhà, chính phủ đã mời thầy Phú, nhà lập trình bậc nhất đến để kiểm tra các cỗ máy. Nhận thấy yêu cầu này quá dễ dàng, thầy Phũ đã nhường nó cho những lập trình viên trẻ tuổi để thử sức và tích lũy thêm kinh nghiệm. Liệu bạn có hoàn thành được yêu cầu này không?

Dữ liệu: Vào từ file MOOD.INP gồm:

+ Dòng đầu tiên là số t ($1 <= t <= 10^4$) - số bộ test.

+ Dòng đầu tiên của mỗi bộ test là hai số nguyên n,m ($1 <= n <= 10^5$, $0 <= m <= 10^9$) - số thành phố và số công dân.

+ Dòng thứ hai là n số nguyên p_1, p_2 , p_3,..., p_n ($0 <= p_i <= m$, $p_1 + p_2 +... + p_n = m$), trong đó p_i là số người sống ở thành phố i.

+ Dòng thứ ba chứa n số nguyên h_1, h_2, h_3,.., h_n ($-10^9 <= h_i <= 10^9$), trong đó h_i là chỉ số hạnh phúc của thành phố i.

Trong n-1 dòng tiếp theo, mỗi dòng chứa hai số nguyên phân biệt x_i, y_i ($1 <= x_i, y_i <= n$) miêu tả đường nối giữa thành phố x_i và thành phố y_i. Có thể đảm bảo rằng tổng của tất cả số n trong t bộ test không quá 2.10^5.

Kết qủa: Ghi ra file MOOD.OUT: với mỗi bộ test, in ra YES nếu như dữ liệu thu thập được là chính xác. Ngược lại in ra NO.

Ví dụ:

MOOD.INP	MOOD.OUT	MOOD.INP	MOOD.OUT
2	YES	2	NO
7 4	YES	4 4	NO
1 0 1 1 0 1 0		1 1 1 1	
4 0 0 -1 0 -1 0		4 1 -3 -1	
1 2		1 2	
1 3		1 3	
1 4		1 4	
3 5		3 13	
3 6		3 3 7	
3 7		13 1 4	
5 11		1 2	
1 2 5 2 1		1 3	
-11 -2 -6 -2 -1			
1 2			
1 3			
1 4			
3 5			

Thuật toán:

Bài này sử dụng thuật toán DFS 2 lần

Gọi sum[i] là số người đến thành phố i. Sử dùng thuật toán DFS để tính sum[i].

Đặt kt= true

DFS từ đỉnh 1, ta có

sum[i]=p[i] + ∑ (sum[j]) - j là các thành phố có đường nối trực tiếp đến i trừ cha của i.
Lại DFS 1 lần nữa từ đỉnh 1 , ở mỗi đỉnh gọi good[i] là số người có tâm trạng tốt đến thành phố i và bad[i] là số người có tâm trạng xấu đến thành phố i. Dễ dàng tính được good[i] bad[i] dựa vào tổng hiệu.
Kiểm tra mối quan hệ giữa sum[i], h[i], good[i], bad[i], nếu không phù hợp kt=false
Gọi goodd[i] là số người có tâm trạng tốt từ thành phố i đi đến các thành phố có đường nối trực tiếp với thành phố i trừ cha của i
→ goodd[i] = ∑ (good[j])
Nếu goodd[i] > good[i] kt=false
Với kt=true ỉn ra YES
Ngược lại in ra NO

Code tham khảo

```
#include <bits/stdc++.h>
#define fi first
#define se second
#define ll long long
#define getbit(n,i) ((n>>(i))&1)
#define offbit(n,i) (n^(1<<(i)))
#define onbit(n,i) (n|(1<<(i)))
#define cntone(x) (__builtin_popcount(x))
const int mod=1e9+7;
using namespace std;
template <typename T> inline void read(T & x)
{
   char c; bool nega=0;
   while((!isdigit(c=getchar()))&&c!='-');
   if(c=='-')
   {
      c=getchar();
      nega=1;
   }
   x=c-48;
   while(isdigit(c=getchar()))
   {
      x=x*10+c-48;
   }
```

```
    if(nega) x=-x;
}
template <typename T> void Write(T x) {if (x > 9) Write(x/10); putchar(x%10+48);}
template <typename T> void write(T x) {if (x < 0) {putchar('-'); x = -x;} Write(x);}
int n,m;
ll p[100005];
ll h[100005];
ll sum[100005];
bool ok;
vector<int> adj[100005];
void dfs(int u,int pa){
    sum[u] = p[u];
    for(int v :adj[u]){
        if(v==pa) continue;
        dfs(v,u);
        sum[u] += sum[v];
    }
}
pair<ll,ll> dfs1(int u,int pa){
    if(sum[u]<abs(h[u]))
        ok=0;
    ll good = (sum[u] + h[u]) / 2;
    ll bad = (sum[u] - h[u]) / 2;
    if(good+bad!=sum[u]||good-bad!=h[u]){
        ok=0;
    }
    ll goodd=0,badd=0;
    for(int v :adj[u]){
        if(v==pa) continue;
        pair<ll,ll> cur=dfs1(v,u);
        goodd += cur.first;
        badd += cur.second;
    }
    if(goodd>good)
        ok=0;
    return {good,bad};
}
int main()
{
    freopen("mood.inp", "r", stdin);
```

```
    freopen("mood.out", "w", stdout);
    int test;
    read(test);
    while(test--){
        ok=1;
        read(n);read(m);
        for(int i=1;i<=n;i++)
            adj[i].clear();
        for(int i=1 ; i<= n; i++)
            read(p[i]);
        for(int i= 1;i <= n ;i++)
            read(h[i]);
        for(int i= 1;i< n;i++){
            int u,v;
            read(u);read(v);
            adj[u].push_back(v);
            adj[v].push_back(u);
        }
        dfs(1,1);
        dfs1(1,1);
        if(ok) {
            putchar('Y');
            putchar('E');
            putchar('S');
            putchar('\n');
        }
        else {
            putchar('N');
            putchar('O');
            putchar('\n');
        }
    }
    return 0;
}
```

Test tham khảo: https://drive.google.com/file/d/1SmmfQB3PHnuN-KXcC5pE-Swqs7nnWIa2/view?usp=sharing

TÀI LIỆU THAM KHẢO

1. http://codeforces.com/

2. https://vn.spoj.com/