

CHUYÊN ĐỀ: MỘT SỐ BÀI TOÁN VỀ ĐỒ THỊ

Bài 1. BGRAPH

Cho đơn đồ thị vô hướng G gồm n đỉnh và m cạnh (đồ thị không có khuyên và cạnh lặp). Bob muốn gán cho mỗi đỉnh một giá trị 1, 2 hoặc 3 sao cho tổng 2 số bất kì ở 2 đầu mút của một cạnh là số lẻ. Hiển nhiên đây là bài toán dễ và Bob tìm được lời giải ngay sau khi code thuật toán Brute Force duyệt toàn bộ các khả năng trên siêu máy tính của mình. Bob thắc mắc liệu có bao nhiêu cách đánh số thỏa mãn. Hai cách đánh số được coi là khác nhau nếu tồn tại một đỉnh được đánh số khác nhau ở trong hai cách. Anh ấy đưa ra câu đố này cho các bạn thi môn Phân tích và thiết kế thuật toán. Vì kết quả có thể rất lớn nên anh ấy chỉ cần in ra kết quả theo modul 998244353.

Dữ liệu: Vào từ file BGRAPH.INP gồm

- Dòng đầu bao gồm 2 số n và m là số đỉnh và số cạnh của đồ thị đã cho.
- m dòng sau mỗi dòng chứa 2 số u và v cho biết có cạnh nối giữa u và v .

Kết quả: Ghi ra file BGRAPH.OUT gồm một số là kết quả của bài toán.

Ví dụ:

BGRAPH.INP	BGRAPH.OUT
2 1 1 2	4

Ràng buộc:

- $1 \leq n \leq 3 \times 10^5$; $0 \leq m \leq 3 \times 10^5$.
- Có 50% test với $n \leq 10$.

- Có 20% test đồ thị liên thông.

Thuật toán:

Xây dựng các thành phần liên thông, mỗi thành phần liên thông (TPLT) là một đồ thị hai phía trong đó một phía chỉ bao gồm đỉnh chẵn, phía còn lại chỉ bao gồm đỉnh lẻ

Gọi a_i, b_i lần lượt là số đỉnh của phía đỉnh chẵn và phía đỉnh lẻ của TPLT thứ i

Kết quả cho TPLT thứ i là $2^{a_i} + 2^{b_i}$

Kết quả bài toán là $\prod 2^{a_i} + 2^{b_i}$

CODE tham khảo

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

using namespace std;

#define y1 as214

#define ii pair < int , int >

#define iii pair < int , ii >

#define iv pair < ii , ii >

#define fi first

#define se second

#define fr front()

#define pb push_back

#define t top()

#define FOR(i , x , n) for(int i = x ; i <= n ; ++i)

#define REP(i , n) for(int i = 0 ; i < n ; ++i)

#define FORD(i , x , n) for(int i = x ; i >= n ; --i)
```

```

#define ll long long
#define oo 1e17
#define int long long
#define pow poww
const int N = 3e5 + 5;
int n , m , mod = 998244353 , z;
vector < int > g[N];
int pow[N];
int ans[3];
int ok[N];
int add(int a , int b)
{
    return (a + b) % mod;
}
int mul(int a , int b)
{
    return (a * b) % mod;
}
bool DFS(int i , int j , int k)
{
    ans[k]++;
    ok[i] = k;
    REP(s , g[i].size())

```

```

{
    int u = g[i][s];
    if(u != j)
    {
        if(ok[u] == k)
            z = 0;
        if(ok[u] == 0)
            DFS(u , i , 3 - k);
    }
}

main()
{
    freopen("BGRAPH.inp","r",stdin);
    freopen("BGRAPH.out","w",stdout);

    cin >> n >> m;
    FOR(i , 1 , m)
    {
        int x , y;
        cin >> x >> y;
        g[x].pb(y);
        g[y].pb(x);
    }
}

```

```

pow[0] = 1;
FOR(i , 1 , n)
    pow[i] = mul(pow[i - 1] , 2);
int res = 1;
FOR(i , 1 , n)
    if(ok[i] == 0)
    {
        z = 1;
        ans[1] = ans[2] = 0;
        DFS(i , -1 , 1);
        if(z == 1)
            res = mul(res , add(pow[ans[1]] , pow[ans[2]]));
        else
            res = 0;
    }
cout << res << "\n";
}

```

Test tham khảo tại

<https://www.mediafire.com/file/hkgc9o74vksxbi1/BGRAPH.zip/file>

Bài 2. DỊCH CHUYỂN VÒNG

Giáo sư X đưa các bé trường mầm non SuperKids thăm hồ Big-O, nơi có huyền thoại về sự xuất hiện của những người ngoài hành tinh. Các bé được vui chơi tự do và đến cuối ngày sẽ có các xe đón về.

Con đường bao quanh hồ Big-O có độ dài n km, dọc theo con đường có n bến xe cách đều nhau đánh số từ 1 tới n theo một chiều đi quanh hồ gọi là **chiều đánh số**. Có a_i bé đứng tại bến i .

Trường có k xe, mỗi xe sẽ được điều đến một bến nào đó để đợi đón các bé. Nếu một bé đứng ở bến không có xe đón, bé sẽ phải **di chuyển trên con đường theo chiều đánh số** cho tới bến có xe đón.

Giáo sư X muốn tìm vị trí các bến cho xe đợi ở đó sao cho tổng độ dài quãng đường các bé phải di chuyển là nhỏ nhất. Sau một hồi phân tích, ông nhận ra đó là một thách thức nổi tiếng của những người ngoài hành tinh để lại trên Trái Đất: Bài toán Circular-Shift Problem (CSP)

Dữ liệu: Vào từ file văn bản CSP.INP

Dòng 1 chứa hai số nguyên dương n, k ($n \leq 800; k \leq n$) tương ứng là số bến và số lượng xe đón học sinh.

Dòng 2 chứa n số nguyên a_1, a_2, \dots, a_n là số học sinh tại các bến. Tổng số học sinh không vượt quá 10^6 .

Các số trên một dòng được ghi cách nhau bởi dấu cách

Kết quả: Ghi ra file văn bản CSP.OUT gồm một số nguyên duy nhất là tổng độ dài quãng đường các bé phải di chuyển (tính bằng km) theo phương án tối ưu tìm được.

Ví dụ:

CSP.INP	CSP.OUT	CSP.INP	CSP.OUT
5 1	3	8 2	5
1 2 0 0 1		1 0 0 1 1 0 1 2	

Ràng buộc:

Bộ test chia làm các subtasks:

Subtask 1 (30% số điểm): $k \leq 3$

Subtask 2 (30% số điểm): $n \leq 60$

Subtask 3 (40% số điểm): $n \leq 300$

Thuật toán

Subtask 1 cần tính tổng quãng đường di chuyển ứng với một phương án phân phối xe. Việc giải quyết subtask 1 với kích thước n lớn còn gợi ý dùng phương pháp 2 con trỏ, kỹ thuật sẽ được nâng cấp thành Knuth optimization.

Subtask 2 gợi ý rằng bài toán này có thể giải bằng quy hoạch động:

Để tiện cho việc xử lý vòng tròn, ta tăng số bến lên gấp đôi, bến $n + 1$ trùng với bến 1, bến $n + 2$ trùng với bến 2, ..., bến $2n$ trùng với bến n . Khi đó mảng A được nhân đôi kích thước: $a_1, a_2, \dots, a_n, a_{n+1}, a_{n+2}, \dots, a_{2n}$

Bây giờ một đoạn liên tiếp gồm đúng n bến tương đương với bến n vòng quanh hồ theo chiều đánh số, vì vậy ta chỉ cho phép các học sinh di chuyển từ bến chỉ số nhỏ hơn tới bến chỉ số lớn hơn.

Chi phí để đưa các học sinh ở bến t về bến j ($t \leq j$) bằng:

$$a_t \times (j - t)$$

Xét các bến từ i đến j ($i \leq j$), chi phí để đón các học sinh trong phạm vi đó về bến j được tính bằng:

$$\text{Cost}(i, j) = \sum_{t=i}^j (a_t \times (j - t))$$

Gọi $f(p, i, j)$ là chi phí nhỏ nhất (tính bằng tổng độ dài quãng đường di chuyển của các học sinh) nếu đón bởi p xe. Lưu ý là nếu số xe dư thừa thì một vài xe có thể đón ở cùng bến, trong trường hợp một xe số hiệu nhỏ nhất tại bến đón học sinh, các xe khác cùng bến coi như vô dụng.

Giá trị $f(k, i, i + n - 1)$ chính là chi phí ít nhất để đón tất cả các học sinh bởi k xe trong đó xe cuối cùng đón ở bến $i + n - 1$. Đáp số là:

$$\min\{f(k, i, i + n - 1)\}, \forall i: 1 \leq i \leq n$$

Cách tính $f(p, i, j)$ như sau:

- Nếu $p = 1$, thì $f(1, i, j) = \text{Cost}(i, j)$.
- Nếu $p > 1$ thì xe thứ $p - 1$ sẽ phải đón ở bến t nào đó thuộc $[i \dots j]$, tất cả các học sinh từ bến $t + 1$ tới j sẽ phải di chuyển tới bến j để lên xe thứ p , vậy:

$$f(p, i, j) = \min \{ f(p - 1, i, t) + \text{Cost}(t + 1, j) \}, \forall t : i \leq t \leq j$$

Subtask 3: Xây dựng ma trận chi phí M , ta sử dụng thuật toán tính toán trên ma trận.

Khởi tạo ma trận M trong đó $M[i][j] = f(1, i, j) = \text{Cost}(i, j)$.

Sau đó tính ma trận:

$$M = M^k \quad (k \text{ ma trận } M)$$

Phần tử $M[i][j]$ lúc này chính là $f(k, i, j)$.

CODE tham khảo

```
#define taskname "CSP"
#include <iostream>
#include <cstdio>
#include <algorithm>
using namespace std;
const int maxN = 800;
const int maxCost = maxN * 1000000;
const int maxN2 = 2 * maxN;
using TMatrix = int[maxN2 + 1][maxN2 + 1];
int n, k, n2;
int cnt[maxN2 + 1], acc[maxN2 + 1];
TMatrix MatA, MatB, MatC, trace;
inline int ReadInt()
{
    char c;
```



```

while (c = getchar(), c < '0' || c > '9');
int x = c - '0';
while (c = getchar(), c >= '0' && c <= '9')
    x = x * 10 + c - '0';
return x;
}
void WriteInt(int x)
{
    if (x > 9) WriteInt(x / 10);
    putchar(x % 10 + '0');
}
inline bool Minimize(int& Target, int Value)
{
    if (Value < Target)
    {
        Target = Value;
        return true;
    }
    return false;
}
void ReadInput()
{
    n = ReadInt();
    k = ReadInt();
    cnt[0] = 0;
    for (int i = 1; i <= n; ++i)
        cnt[i] = ReadInt();
    n2 = n * 2;
    copy(cnt + 1, cnt + n + 1, cnt + n + 1);
    acc[0] = 0;
    for (int i = 1; i <= n2; ++i)
        acc[i] = acc[i - 1] + cnt[i] * i;
    for (int i = 1; i <= n2; ++i)

```

```

        cnt[i] += cnt[i - 1];
    }
    //Chỉ phí chuyển mọi hs in [low, high] về vị trí high
    inline int Cost(int low, int high)
    {
        --low;
        return high * (cnt[high] - cnt[low]) - (acc[high] - acc[low]);
    }
    void CalOriginalMatrix()
    {
        for (int i = 1; i <= n2; ++i)
        {
            fill(&MatA[i][1], &MatA[i][n2 + 1], 0);
            fill(&MatB[i][1], &MatB[i][n2 + 1], 0);
        }
        for (int i = 1; i <= n2; ++i)
            for (int j = i; j <= n2 && j < i + n; ++j)
                MatB[i][j] = MatA[i][j] = Cost(i, j);
    }
    //Matrix multiplication with knuth optimization
    void Mul(const TMatrix& x, const TMatrix& y, TMatrix& res)
    {
        for (int step = 0; step < n; ++step)
            for (int i = 1; i + step < n2; ++i)
            {
                int j = i + step; //Tính res[i][i + step];
                res[i][j] = x[i][j];
                trace[i][j] = j;
                if (i == j) continue;
                int k = trace[i][j - 1];
                int endk = min(trace[i + 1][j] + 1, j);
                for (; k <= endk; ++k)
                    if (Minimize(res[i][j], x[i][k] + y[k + 1][j]))

```

```

        trace[i][j] = k;
    }
}
void Solve()
{
    int p = 1;
    while (p * 2 <= k) p *= 2;
    k %= p;
    p /= 2;
    TMatrix* b = &MatB;
    TMatrix* c = &MatC;
    for (; p > 0; k %= p, p /= 2)
    {
        Mul(*b, *b, *c);
        swap(b, c);
        if (k / p == 1)
        {
            Mul(*b, MatA, *c);
            swap(b, c);
        }
    }
    int Result = maxCost;
    for (int i = 1; i <= n; ++i)
        Minimize(Result, (*b)[i][i + n - 1]);
    WriteInt(Result);
}
int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    freopen(taskname".INP", "r", stdin);
    freopen(taskname".OUT", "w", stdout);
    ReadInput();
}

```

```
CalOriginalMatrix();  
Solve();  
}
```

Test tham khảo tại

<https://www.mediafire.com/file/eic0ar4pg2g2iaz/CSP.rar/file>

Bài 3. REBUILD

Hệ thống giao thông trong thành phố Hội An không còn hiệu quả nữa, một số tuyến đường có lưu lượng xe tham gia giao thông rất lớn thì lại đi qua vùng dân cư đông đúc.

Chính quyền thành phố quyết định sẽ cải tạo hệ thống đó để khắc phục tình trạng mất an toàn và giảm mật độ giao thông khi các phương tiện giao thông đi qua các khu vực đông dân cư. Hiện tại thành phố có N nút giao thông, $N - 1$ đoạn đường phố hai chiều nối các nút giao thông và có thể đi từ một nút giao thông bất kỳ đến tất cả các nút còn lại. Hệ thống giao thông mới được thiết kế cũng trên N nút giao thông đó, nhưng với một số đoạn đường phố khác thay thế cho các đoạn đường phố cũ (và vẫn đảm bảo giao thông giữa tất cả các nút).

Để duy trì sự giao thông giữa các nút người ta tiến hành xây dựng một đoạn đường tại một thời điểm. Mỗi tuần sẽ có một đoạn đường ngừng hoạt động và một đoạn đường mới được xây dựng và đưa vào sử dụng. Với cách làm như vậy hệ thống giao thông vẫn đủ $N - 1$ đoạn đường phố và vẫn đảm bảo sự giao thông giữa N nút giao thông của thành phố trong tuần thi công đó.

Yêu cầu: Bạn hãy giúp chính quyền thành phố tìm một cách thi công hệ thống giao thông mới thỏa các điều kiện trên và sao cho số tuần thi công là ít nhất có thể.

Dữ liệu vào: Từ tệp văn bản REBUILD.INP gồm

- Dòng 1: ghi số nguyên dương N .
- $N - 1$ dòng tiếp theo mỗi dòng ghi hai số nguyên a, b ($0 \leq a, b < N$) mô tả một đoạn đường phố nối giữa hai nút giao thông a, b . Và hệ thống đã cho luôn đảm bảo sự đi lại giữa hai nút giao thông bất kỳ qua một hay một dãy các đoạn đường phố.
- $N - 1$ dòng tiếp theo mô tả các đoạn đường phố trong hệ thống giao thông mới.

Kết quả: Ghi ra tệp văn bản REBUILD.OUT gồm

- Dòng 1: ghi số nguyên dương K là số tuần ít nhất để thi công hệ thống giao thông mới.

- K dòng tiếp theo, mỗi dòng ghi bốn số nguyên a_1, b_1, a_2, b_2 trong đó a_1, b_1 là đoạn đường ngừng hoạt động và a_2, b_2 là đoạn đường mới được đưa vào sử dụng của tuần đó.

Ví dụ:

REBUILD.INP	REBUILD.OUT
3	1
0 1	2 1 2 0
1 2	
0 1	
0 2	

REBUILD.INP	REBUILD.OUT
4	2
0 1	3 0 3 2
0 2	0 2 1 2
0 3	
0 1	
1 2	
2 3	

Ràng buộc:

- Có 35% số lượng test thỏa mãn $N \leq 10$
- Có 35% số lượng test thỏa mãn $N \leq 1000$
- Có 30% số lượng test thỏa mãn $N \leq 100000$.

Thuật toán

Cho hai cây đồ thị T_S và T_T , bài toán yêu cầu thực hiện chuyển từ cây này sang một cây kia bằng cách di chuyển ít cạnh nhất mà vẫn duy trì sự liên thông của cây (vẫn là cây) tại mỗi bước thực hiện.

Đầu tiên ta nhận xét rằng số cạnh ít nhất cần di chuyển là số cạnh khác nhau giữa hai cây. Gọi S là tập các cạnh có trong cây T_S mà không có trong cây T_T , T là tập các cạnh có trong cây T_T mà không có trong cây T_S và C là tập cạnh chung của hai cây khi đó ta có $|S| = |T| = (N - 1) - |C|$. Giả thiết là ta cần di chuyển số cạnh ít nhất trong S .

Gọi T_c là cây thu được trong quá trình xây dựng, lúc đầu T_c bằng cây thứ nhất, với cạnh $e \in S$, sau khi bỏ cạnh e thì cây T_c sẽ chia thành hai cây con T_1, T_2 . Trong T sẽ tồn tại một

vài cạnh e_2 nối hay cây T_1, T_2 thành một cây. Thay e bởi e_2 ta giảm đi được một cạnh trong S .

Khi này bài toán đưa về việc tìm e_2 sao cho hiệu quả?

Thuật toán 1: Duyệt mọi cách chọn $e_2 \in T$, thay dần các cạnh $e \in S$ của cây thứ nhất. Bằng cách chọn lần lượt $e \in S$, với mỗi e ta duyệt lần lượt $e_2 \in T$. Thay e bởi e_2 và kiểm tra tính liên thông của cây T_c . Nếu T_c liên thông thì ta bỏ e khỏi S và e_2 khỏi T và dừng (S giảm đi 1), ngược lại ta duyệt qua e_2 mới.

Thuật toán thực hiện việc bỏ, dựng cây và kiểm tra tính liên thông nhiều lần nên dùng cấu trúc union-find hiệu quả hơn khi cài đặt.

Thuật toán duyệt qua S và T , mỗi lần kiểm tra tính liên thông mất $O(N)$. Nên độ phức tạp của thuật toán $O(N^3)$.

Thuật toán 2: Bỏ lần lượt các cạnh $e \in S$ trên cây T_c , mỗi lần bỏ sẽ chia cây thu được thành hai cây, $DFS()$ từ một đỉnh bất kỳ T_c ta thu được tập đỉnh V_1 của cây T_1 và V_2 của cây T_2 và tìm $e_2 \in T$ sao cho $e_2 = (u, v)$ có $u \in V_1$ và $v \in V_2$. Thêm e_2 vào cây T_c . Cây T_c đã thay thêm một cạnh.

Duyệt $e \in S$ mất $O(N)$, $DFS()$ mất $O(N)$, việc tìm e_2 chỉ là $O(1)$ nên thuật toán có độ phức tạp là $O(N^2)$.

Thuật toán 3: Thay đổi cách chọn e và e_2 , nếu ta chọn e nối một nút lá của cây T_S , và chọn e_2 tương ứng với nút lá đó của cây T_T thì ta không cần phải kiểm tra tính liên thông của cây T_c . Để thực hiện ta dùng một danh sách lưu các cạnh nối đến mỗi $u \in T_S$ và mỗi $v \in T_T$ và dùng một hàng đợi để lưu các nút sẽ trở thành nút lá trong quá trình thay.

Để ý các nút được nối với nhau trong C thì cũng sẽ nối với nhau sau khi chuyển các cạnh nên ta xem các cạnh này sẽ tạo ra các cây (nối thêm các cạnh trong S thì ta được cây T_S). Trong trường hợp xấu nhất thì cây này cũng có ít nhất một nút lá và ta có thể áp dụng cách làm trên.

Để thực hiện ta dùng cấu trúc union – find để ghép hết các cạnh trong C , sau đó ghép dần các cạnh (đi ra) từ danh sách đã lưu.

Có tối đa N nút lá, mỗi thao tác trong union – find mất $O(\log N)$ nên độ phức tạp của thuật toán là $O(N \log N)$.

CODE tham khảo

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

#define rep(i, from, to) for (int i = from; i < (to); ++i)
#define trav(a, x) for (auto& a : x)
#define all(x) x.begin(), x.end()
#define sz(x) (int)(x).size()
typedef long long ll;
typedef pair<int, int> pii;
typedef vector<int> vi;

struct Edge {
    int u, v, id;
};

struct UF {
    vi v, deg;
    vector<vector<Edge>> ed, ed2;
    UF(int n) : v(n, -1), deg(n), ed(n), ed2(n) {}
    int find(int x) { return v[x] < 0 ? x : v[x] = find(v[x]); }
    void join(int a, int b) {
        a = find(a), b = find(b);
        if (a == b) return;
        if (-v[a] < -v[b]) swap(a, b);
        v[a] += v[b];
```



```

    deg[a] += deg[b];
    ed[a].insert(ed[a].end(), all(ed[b]));
    ed2[a].insert(ed2[a].end(), all(ed2[b]));
    v[b] = a;
}
};

int main() {
    cin.sync_with_stdio(false);
    cin.exceptions(cin.failbit);
    freopen("rebuild.inp", "r", stdin);
    freopen("rebuild.out", "w", stdout);

    int N;
    cin >> N;

    if (N == 1) {
        cout << 0 << endl;
        return 0;
    }

    UF uf(N);

    int u, v;
    set<pii> source, target;
    rep(i,0,N-1) {
        cin >> u >> v;
        if (u > v) swap(u, v);
        source.insert(pii(u,v));
    }
}

```

```

rep(i,0,N-1) {
    cin >> u >> v;
    if (u > v) swap(u, v);
    if (source.count(pii(u,v))) {
        uf.join(u, v);
        source.erase(pii(u,v));
    }
    else
        target.insert(pii(u,v));
}

int delta = sz(source);
assert(delta == sz(target));

vector<bool> dead;
int eid = 0;
trav(e, source) {
    tie(u,v) = e;
    Edge edg = {u, v, eid++};
    uf.ed[uf.find(u)].push_back(edg);
    uf.ed[uf.find(v)].push_back(edg);
    dead.push_back(false);
}
trav(e, target) {
    tie(u,v) = e;
    Edge edg = {u, v, eid++};
    uf.ed2[uf.find(u)].push_back(edg);
    uf.ed2[uf.find(v)].push_back(edg);
}

```

```

    dead.push_back(false);
}

vi leaves;
rep(i,0,N) {
    uf.deg[i] = sz(uf.ed[i]);
    if (uf.deg[i] == 1) leaves.push_back(i);
}

vector<tuple<int, int, int, int>> out;
while (!leaves.empty()) {
    int theLeaf = leaves.back();
    int comp = uf.find(theLeaf);
    leaves.pop_back();

    if (uf.ed[comp].empty()) continue;
    Edge e = uf.ed[comp].back(), e2;
    uf.ed[comp].pop_back();
    if (dead[e.id]) {
        leaves.push_back(theLeaf);
        continue;
    }

    int x = e.u, y = e.v;
    if (uf.find(x) != comp) swap(x, y);
    assert(uf.find(x) == comp);
    for (;;) {
        assert(uf.ed2[comp].size() >= 1);
        e2 = uf.ed2[comp].back();

```

```

        uf.ed2[comp].pop_back();
        if (!dead[e2.id]) break;
    }
    int x2 = e2.u, y2 = e2.v;
    if (uf.find(x2) != comp) swap(x2, y2);
    assert(uf.find(x2) == comp);
    out.emplace_back(x, y, x2, y2);
    dead[e.id] = true;
    dead[e2.id] = true;
    --uf.deg[comp];
    if (--uf.deg[uf.find(y)] == 1) {
        leaves.push_back(y);
    }
    uf.join(x, y2);
}
cout << sz(out) << endl;
trav(t, out) {
    int a,b,c,d;
    tie(a,b,c,d) = t;
    cout << a << ' ' << b << ' ' << c << ' ' << d << '\n';
}
assert(sz(out) == delta);
}

```

Test tham khảo tại

<https://www.mediafire.com/file/33u3twkf6iz9nsi/REBUILD.rar/file>

Bài 4.CEZAR

Tại TP, có tất cả n thượng nghị sĩ ở trong n ngôi nhà (đánh số từ 1 đến n), giữa hai ngôi nhà có một đường đi duy nhất: đường đi trực tiếp hoặc không trực tiếp (đi qua các con đường khác). Tất cả các ngôi nhà đều đi được đến nhau.

Mỗi thượng nghị sĩ khi đi từ nhà mình đến thượng viện, phải trả 1 USD khi đi qua một con phố (con phố là đường nối trực tiếp hai ngôi nhà bất kỳ). Người đứng đầu thượng viện đã nghĩ cách làm sao cho số tiền mà các thượng nghị sĩ phải trả là tối thiểu. Vì vậy, ông ra quyết định:

- Có k con phố miễn phí (thượng nghị sĩ sẽ không phải trả tiền khi đi trên con phố này).
- Đặt tòa nhà thượng viện ở một trong n ngôi nhà.

Yêu cầu: Hãy viết chương trình tính xem chi phí tối thiểu là bao nhiêu?

Dữ liệu vào từ tệp văn bản **CEZAR.INP** gồm:

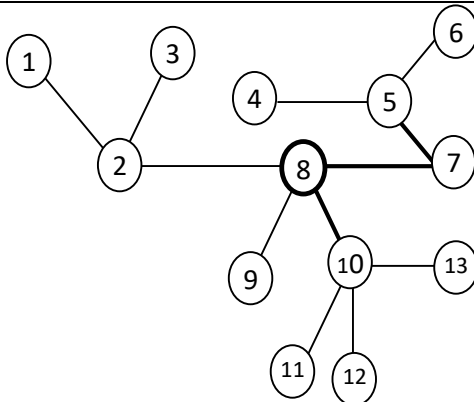
- Dòng 1: Số nguyên n và k tương ứng là số ngôi nhà ở TP và số con phố miễn phí.
- $n - 1$ dòng tiếp theo, mỗi dòng chứa 2 số i, j có nghĩa là có con phố nối ngôi nhà i và ngôi nhà j .

$$(1 < n \leq 10000; 0 < k < n; 1 \leq i, j \leq n; i \neq j)$$

Kết quả đưa ra tệp văn bản **CEZAR.OUT** gồm: Một số duy nhất là chi phí tối thiểu phải trả.

Ví dụ:

CEZAR.INP	CEZAR.OUT	Giải thích
-----------	-----------	------------

13 3	11	 <p>Có nhiều phương án giải quyết: Đây là 1 phương án: 5 - 7, 7 - 8, 8 - 10 là những con phố miễn phí và 8 là thượng viện. Chi phí tối thiểu là: 11.</p>
1 2		
2 3		
2 8		
7 8		
7 5		
5 4		
5 6		
8 9		
8 10		
10 11		
10 12		
10 13		

Ràng buộc:

- Có 30% số test ứng với 30% số điểm của bài có $n \leq 30$
- Có 40% số test ứng với 40% số điểm của bài có $30 < n \leq 5000$
- Có 30% số test ứng với 30% số điểm của bài có $5000 < n \leq 10000$.

Thuật toán

Tìm cách tính tất cả các tổng khoảng cách nếu chọn mỗi đỉnh làm gốc. Trước hết chọn đỉnh 1 làm gốc, từ đó tính được các đỉnh con của đỉnh 1, và cứ như thế...

Dùng cấu trúc đặc biệt BIT để lưu trữ trọng số các cạnh rồi sử dụng chập nhị phân để tính tổng của $(N-1-k)$ cạnh có trọng số nhỏ nhất. ĐPT: $O(N \cdot \log N \cdot \log N)$

CODE tham khảo

```
#include <bits/stdc++.h>
using namespace std;
// Takes in two strings, adds an edge if possible
// returns whether or not it is possible using this config
bool cmp(vector<vector<bool>>& adj, vector<int>& deg, string s1, string s2) {
```

```

for(int i = 0; i < min(s1.size(), s2.size()); i++) {
    if(s1[i] != s2[i]) {
        if(!adj[s1[i]-'a'][s2[i]-'a']) {
            adj[s1[i]-'a'][s2[i]-'a'] = true;
            deg[s2[i]-'a']++;
        }
        return true;
    }
}
return s1.size() < s2.size();
}
// From the toposort order, build the answer
string convert(string s) {
    string ans(26, 'a');
    for(int i = 0; i < s.size(); i++) {
        ans[s[i]-'a'] = i+'a';
    }
    return ans;
}
int main() {
    int n;
    cin >> n;
    vector<string> input(n);
    for(auto& i : input) {
        cin >> i;
    }
    vector<string> sorted(n);
    for(int i = 0; i < n; i++) {
        int t;
        cin >> t;
        t--;
        sorted[i] = input[t];
    }
    vector<int> deg(26, 0);
    vector<vector<bool>> adj;
    adj.resize(26, vector<bool>(26, false));
    // For each pair, do a string compare, and add edge or detect
    // if impossible
    bool works = true;
    for(int i = 1; i < n; i++) {

```

```

        works &= cmp(adj, deg, sorted[i-1], sorted[i]);
    }
    // Prepare toposort
    queue<int> q;
    for(int i = 0; i < 26; i++) {
        if(deg[i] == 0) {
            q.push(i);
        }
    }
    string order = "";
    // Run toposort
    while(!q.empty()) {
        int curr = q.front();
        q.pop();
        order.push_back('a'+curr);
        for(int i = 0; i < 26; i++) {
            if(adj[curr][i]) {
                deg[i]--;
                if(deg[i] == 0) {
                    q.push(i);
                }
            }
        }
    }
    // Print answer
    if(works && order.size() == 26) {
        cout << "DA" << endl;
        order = convert(order);
        cout << order << endl;
    }
    else {
        cout << "NE" << endl;
    }
}

```

Test tham khảo tại

<https://www.mediafire.com/file/qzxnwsixeks4q0/CEZAR.rar/file>

Bài 5. HỆ THỐNG TIỀN TỆ

Hệ thống tiền tệ tại một quốc gia luôn đảm bảo việc lưu thông buôn bán một cách thuận tiện nhất. Theo đó nó phải có khả năng thanh toán cho tất cả các mức giá nguyên dương. Tuy vậy, do lạm phát mà mệnh giá tiền ngày càng tăng lên và các mệnh giá hiện tại có thể không giữ được tính chất cố hữu đó nữa. Ví dụ với các tờ tiền có mệnh giá là 100, 200, 500, 1000, 2000, 5000 thì rõ ràng không thể chi trả số tiền là 2020.

Cụ thể hơn, một số tiền x được gọi là thanh toán được bằng hệ thống tiền tệ hiện tại nếu có thể chọn số lượng cho mỗi mệnh giá sao cho tổng giá trị được chọn là x . Theo định kỳ thì Ủy ban tài chính quốc gia sẽ có những khảo sát để đánh giá mức độ thuận tiện của hệ thống tiền tệ, trước hết người ta chọn một số nguyên dương T , thường là giới hạn các giao dịch từng được sử dụng. Sau đó họ tính số lượng các số nguyên x ($0 \leq x \leq T$), có thể thanh toán được.

Yêu cầu: Bạn hãy giúp Ủy ban tài chính quốc gia tính số lượng các số nguyên x , có thể thanh toán được trong hệ thống tiền tệ hiện tại.

Dữ liệu vào: Từ tệp văn bản YENOM.INP gồm

- Dòng đầu tiên ghi hai số nguyên dương n ($n \leq 2000$) và T , với n là số lượng mệnh giá;
- Dòng thứ hai gồm n số nguyên dương là các mệnh giá: a_1, a_2, \dots, a_n ($1 \leq a_i \leq 2000$).

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

Kết quả: Ghi ra tệp văn bản YENOM.OUT một số nguyên dương là kết quả bài toán.

Ví dụ:

YENOM.INP	YENOM.OUT
3 5 1 4 2	6

YENOM.INP	YENOM.OUT
5 10000 100 200 500 1000 2000	101

Ràng buộc:

- Có 40% số test ứng với 40% số điểm của bài có $T \leq 2 \times 10^3$;
- Có 20% số test ứng với 20% số điểm của bài có $T \leq 2 \times 10^4$;
- Có 20% số test ứng với 20% số điểm của bài có $T \leq 2 \times 10^5$;
- Có 20% số test khác ứng với 20% số điểm còn lại của bài có $T \leq 10^{18}$.

Thuật toán

Thuật toán 1: Với ràng buộc $T \leq 2 \times 10^4$.

Gọi $F[x] = \text{true}$ nếu có cách thanh toán số tiền x .

$$F[0] = \text{true}$$

$$F[x] = \bigvee_{i=1}^n F[x - a_i], x > 0.$$

Thuật toán 2: Với ràng buộc $T \leq 2 \times 10^5$.

Lưu mảng F bằng một bitset.

$$F[0] = 1$$

$$\text{Xét } i = 1..n, F = F | (F \ll a_i).$$

Thuật toán 3: Với ràng buộc $T \leq 10^{18}$.

Sử dụng thuật toán Dijkstra.

Giả sử $x \% a_1 = r$, khi đó ta sẽ đếm phân hoạch theo r (tức là xét từng r và đếm số nghiệm x của bài toán mà $x \% a_1 = r$).

Dựng đồ thị với a_1 đỉnh $0, 1, \dots, a_1 - 1$. Từ đỉnh k sẽ có n cung, cung thứ i có trọng số a_i , nối k với $(k + a_i) \% a_1$.

Tìm đường đi ngắn nhất từ 0 đến tất cả các đỉnh khác.

- Nếu $d_r > T$ thì không có nghiệm nào ứng với trường hợp $x \% a_1 = r$.

- Nếu $d_r \leq T$ thì tất cả các nghiệm x ứng với trường hợp $x \% a_1 = r$ là: $d_r, d_r + a_1, d_r + 2a_1, \dots$ cho đến khi vượt quá T. Có thể hiểu d_r là số nhỏ nhất tạo được từ tập a mà đồng dư với r theo mod a_1 .

Độ phức tạp thuật toán là: $O(a_1^2)$

CODE tham khảo

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

int32_t main()
{
    freopen("YENOM.inp", "r", stdin);
    freopen("YENOM.out", "w", stdout);
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    int n;
    long long T;
    cin >> n >> T;
    vector<int> e;
    while(n--){
        int a;
        cin >> a;
        e.push_back(a);
    }
    n = e.back();
    vector<ll> d(n, T+1);
    vector<int> mark(n);
    d[0] = 0;
    for (int i = 1; i < n; ++i){//Dijkstra
```

```

    int x = -1;
    for (int j = 0; j < n; ++j) if (!mark[j]){
        if (x == -1 || d[j] < d[x]) x = j;
    }
    if (d[x] > T) break;
    mark[x] = 1;
    for (int i : e)
    {
        int y = (x + i) % n;
        d[y] = min(d[y], d[x] + i);
    }
}
ll ans = 0;
for (int i = 0; i < n; ++i) if (d[i] <= T){
    ans += (T - d[i]) / n + 1;
}
cout << ans << endl;
}

```

Test tham khảo tại

<https://www.mediafire.com/file/qp6jdz7ua7mu0jh/YENOM.rar/file>

MỘT SỐ BÀI TOÁN THAM KHẢO

Bài 6: VÉ MIỄN PHÍ

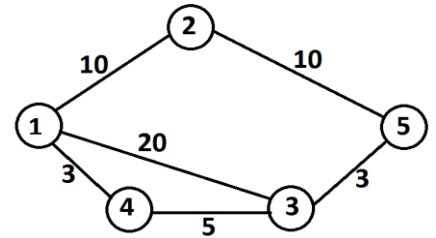
Tham gia trò chơi nhảy lò cò, thật may mắn, Nam đã giành giải nhất của cuộc thi. Phần thưởng mà Nam nhận được là k vé xe buýt miễn phí để đi thăm quan thành phố Hạ Long. Mỗi vé xe chỉ được sử dụng một lần và có thể sử dụng cho bất kỳ tuyến xe buýt nào trong thành phố. Thành phố có n nút giao thông được đánh số từ 1 đến n và m tuyến xe buýt hai chiều. Mỗi cặp nút giao thông i, j có không quá một tuyến xe buýt hai chiều, nếu có thì để đi từ nút i đến nút j (hoặc từ nút j đến nút i) với giá vé là $c_{ij} = c_{ji}$ đồng. Xuất phát từ nút giao thông s , Nam muốn di chuyển đến nút giao thông t và anh luôn lựa chọn đường đi với chi phí ít nhất.

Ví dụ: thành phố có 5 nút giao thông và 6 tuyến xe buýt:

Tuyến 1: 1-2 giá vé 10 đồng; Tuyến 2: 2-5 giá vé 10 đồng;

Tuyến 3: 1-4 giá vé 3 đồng; Tuyến 4: 3-4 giá vé 5 đồng;

Tuyến 5: 3-5 giá vé 3 đồng; Tuyến 6: 1-3 giá vé 20 đồng.



Xuất phát từ nút 1 đến nút 5, đi theo hành trình $1 \rightarrow 4 \rightarrow 3 \rightarrow 5$ hết 11 đồng là đường đi với chi phí ít nhất. Tuy nhiên, nếu Nam sử dụng 1 vé xe miễn phí thì đường đi $1 \rightarrow 3 \rightarrow 5$ hết 3 đồng là ít nhất (vé xe miễn phí được sử dụng tại tuyến 1-3).

Yêu cầu: Cho biết các tuyến xe buýt với giá vé tương ứng và các giá trị s, t, k . Hãy tính chi phí ít nhất để đi từ nút giao thông s đến nút giao thông t mà không sử dụng quá k vé xe miễn phí.

INPUT: Vào từ file văn bản FREEBUS.INP

- Dòng đầu tiên ghi năm số nguyên dương n, m, k, s, t ;
- m dòng sau, mỗi dòng 3 số nguyên i, j, c_{ij} mô tả có tuyến xe buýt $i - j$ hết c_{ij} đồng.

OUTPUT: Đưa ra file văn bản FREEBUS.OUT một số duy nhất là chi phí ít nhất để đi từ nút giao thông s đến nút giao thông t mà không sử dụng quá k vé xe miễn phí.

Ví dụ:

FREEBUS.INP	FREEBUS.OUT
5 6 1 1 5	3
1 2 10	
2 5 10	
1 4 3	
3 4 5	
3 5 3	
1 3 20	

Ghi chú:

- Có 40% số test ứng với 40% số điểm có $n \leq 100$, $m \leq 1000$ và $k = 1$;
- Có 20% số test ứng với 20% số điểm có $n \leq 10^5$, $m \leq 10^5$ và $k = 1$;
- Có 40% số test còn lại ứng với 40% số điểm có $n \leq 10^5$, $m \leq 10^5$ và $k \leq 5$.

CODE tham khảo

```
#include <iostream>
#include <vector>
#include <queue>
using namespace std;

#define INF 123456789123456789LL

int n, m, k, s, t;
vector<int> c[100001];
vector<long long> p[100001];
long long d[100001][6];
bool inQueue[100001][6];
queue<int> q1, q2;
```

```

void push(int x, int y) {
    if (!inQueue[x][y]) {
        q1.push(x);
        q2.push(y);
        inQueue[x][y] = true;
    }
}

int main() {
    ios_base::sync_with_stdio(false);
    cin >> n >> m >> k >> s >> t;
    for (int i = 0; i < m; i++) {
        int u, v, w;
        cin >> u >> v >> w;
        c[u].push_back(v);
        p[u].push_back(w);
        c[v].push_back(u);
        p[v].push_back(w);
    }
    for (int x = 1; x <= n; x++)
        for (int y = 0; y <= k; y++) {
            d[x][y] = INF;
            inQueue[x][y] = false;
        }
    q1.push(s);
    q2.push(k);
    d[s][k] = 0;
    while (!q1.empty()) {
        int x = q1.front(); q1.pop();
        int y = q2.front(); q2.pop();
    }
}

```

```

inQueue[x][y] = false;
for (int i = 0; i < c[x].size(); i++) {
    int xx = c[x][i];
    long long price = p[x][i];
    //take free bus to xx
    if (y) {
        int yy = y - 1;
        if (d[xx][yy] > d[x][y]) {
            d[xx][yy] = d[x][y];
            push(xx, yy);
        }
    }
    //pay the price
    int yy = y;
    if (d[xx][yy] > d[x][y] + price) {
        d[xx][yy] = d[x][y] + price;
        push(xx, yy);
    }
}
long long res = INF;
for (int kk = 0; kk <= k; kk++) {
    if (d[t][kk] < res) res = d[t][kk];
}
cout << res << endl;
cerr << res << endl;
return 0;
}

```


BÀI 7: TRUY TÌM TỘI PHẠM

Để truy bắt tội phạm, cảnh sát xây dựng một hệ thống máy tính mới. Bản đồ khu vực bao gồm N thành phố và E đường nối 2 chiều. Các thành phố được đánh số từ 1 đến N.

Cảnh sát muốn bắt các tội phạm di chuyển từ thành phố này đến thành phố khác. Các điều tra viên, theo dõi bản đồ, phải xác định vị trí thiết lập trạm gác. Hệ thống máy tính mới phải trả lời được 2 loại truy vấn sau:

1. Đối với hai thành phố A, B và một đường nối giữa hai thành phố G_1, G_2 , hỏi tội phạm có thể di chuyển từ A đến B nếu đường nối này bị chặn (nghĩa là tên tội phạm không thể sử dụng con đường này) không?
2. Đối với 3 thành phố A, B, C, hỏi tội phạm có thể di chuyển từ A đến B nếu như toàn bộ thành phố C bị kiểm soát (nghĩa là tên tội phạm không thể đi vào thành phố này) không?

Dữ liệu vào: Vào từ file **CRIMINAL.INP**

- Dòng đầu tiên chứa 2 số nguyên N và E ($2 \leq N \leq 100\,000$, $1 \leq E \leq 500\,000$), số thành phố và số đường nối.
- Mỗi dòng trong số E dòng tiếp theo chứa 2 số nguyên phân biệt thuộc phạm vi [1, N] - cho biết nhãn của hai thành phố nối với nhau bởi một con đường. Giữa hai thành phố có nhiều nhất một đường nối.
- Dòng tiếp theo chứa số nguyên Q ($1 \leq Q \leq 300\,000$), số truy vấn được thử nghiệm trên hệ thống.
- Mỗi dòng trong Q dòng tiếp theo chứa 4 hoặc 5 số nguyên. Số đầu tiên cho biết loại truy vấn - 1 hoặc 2.
 - Nếu loại truy vấn là 1, tiếp theo trên cùng dòng là 4 số nguyên A, B, G_1 , G_2 với ý nghĩa như đã mô tả. A khác B; G_1, G_2 mô tả một con đường có sẵn.
 - Nếu loại truy vấn là 2, tiếp theo trên cùng dòng là 3 số nguyên A, B, C với ý nghĩa như đã mô tả. A, B, C đôi một khác nhau.

Dữ liệu được cho sao cho ban đầu luôn có cách di chuyển giữa hai thành phố bất kỳ.

Kết quả: Xuất ra file **CRIMINAL.OUT**

Gồm Q dòng, mỗi dòng chứa câu trả lời cho một truy vấn. Nếu câu trả lời là khẳng định, in ra "yes". Nếu câu trả lời là phủ định, in ra "no".

CRIMINAL.INP	CRIMINAL.OUT
13 15	yes
1 2	yes
2 3	yes
3 5	no
2 4	yes
4 6	
2 6	
1 4	
1 7	
7 8	
7 9	
7 10	
8 11	
8 12	
9 12	
12 13	

5	
1 5 13 1 2	
1 6 2 1 4	
1 13 6 7 8	
2 13 6 7	
2 13 6 8	

CODE tham khảo

```

#include <iostream>
#include <vector>
using namespace std;
const int N = 1e5+5;
int n, m, low[N], num[N], pa[N][21];
int tin[N], tout[N], Time;
vector<int> a[N];
void dfs(int p, int u) {
    tin[u] = ++Time;
    num[u] = ++num[0]; low[u] = n+1;
    pa[u][0] = p;
    FOR(i,1,20) pa[u][i] = pa[pa[u][i-1]][i-1];

    for (auto v : a[u]) {
        if (v == p) continue;
        if (num[v])
            low[u] = min(low[u], num[v]);
    }
}

```

```

        else {
            dfs(u,v);
            low[u] = min(low[u], low[v]);
        }
    }
    tout[u] = ++Time;
}

bool thuoc(int v, int u) {
    return tin[u] <= tin[v] && tout[v] <= tout[u];
}

int FindParent(int p, int v) {
    FOD(i,20,0) {
        int u = pa[v][i];
        if (u > 0 && u != p && thuoc(u,p)) v = u;
    }
    return v;
}

bool solve1(int,int,int,int);
bool solve2(int,int,int);

int main() {
    scanf("%d%d", &n,&m);
    while (m--) {
        int i, j; scanf("%d%d", &i,&j);
        a[i].pb(j); a[j].pb(i);
    }
    dfs(0,1);
}

```

```

scanf("%d", &m);
while (m--) {
    int t; scanf("%d", &t);
    if (t == 1) {
        int A, B, G1, G2;
        scanf("%d%d%d%d", &A, &B, &G1, &G2);
        if (solve1(A, B, G1, G2)) puts("yes"); else puts("no");
    }
    else {
        int A, B, C;
        scanf("%d%d%d", &A, &B, &C);
        if (solve2(A, B, C)) puts("yes"); else puts("no");
    }
}

return 0;
}

bool solve1(int A, int B, int G1, int G2) {
    if (num[G1] > num[G2]) swap(G1, G2);
    if (pa[G2][0] != G1) return true;
    if (low[G2] <= num[G1]) return true;
    if (thuoc(A,G2) == thuoc(B,G2)) return true;
    return false;
}

bool solve2(int A, int B, int C) {
    if (num[A] > num[B]) swap(A,B);
    if (pa[B][0] == A) return true;

```

```

if (thuoc(B,A)) {
    if (thuoc(C,A) && thuoc(B,C)) {
        B = FindParent(C,B);
        if (low[B] < num[C]) return true;
        return false;
    }
    return true;
}
if (thuoc(C,A) || thuoc(C,B)) return true;
if (thuoc(A,C) && thuoc(B,C)) {
    A = FindParent(C,A); B = FindParent(C,B);
    if (A == B) return true;
    if (low[A] < num[C] && low[B] < num[C]) return true;
    return false;
}
if (thuoc(A,C) || thuoc(B,C)) {
    if (thuoc(B,C)) swap(A,B);
    A = FindParent(C,A);
    if (low[A] < num[C]) return true;
    return false;
}
return true;
}

```

Bài 8. XE BUÝT

Một xe buýt của công ty có nhiệm vụ đón nhân viên đến trụ sở làm việc. Trên hành trình xe buýt sẽ tiếp nhận nhân viên đứng chờ ở các điểm hẹn nếu như xe còn chỗ trống. Xe buýt có thể sẽ đỗ lại để chờ những công nhân còn chưa kịp đến điểm hẹn. Cho biết thời điểm mà mỗi nhân viên đến điểm hẹn của mình và thời gian cần thiết để xe buýt di chuyển từ một điểm hẹn đến điểm hẹn tiếp theo. Giả thiết rằng xe buýt đến điểm hẹn đầu tiên tại thời điểm 0, thời gian xếp khách lên xe được coi là bằng 0.

Yêu cầu: Hãy xác định khoảng thời gian ngắn nhất để xe buýt có thể chở một số lượng nhiều nhất các nhân viên đến trụ sở làm việc.

Dữ liệu: Vào từ file BUS.INP gồm

- Dòng đầu tiên chứa 2 số nguyên dương N, M theo thứ tự là số điểm hẹn và số chỗ ngồi của xe buýt;
- Dòng thứ i trong số N dòng tiếp theo chứa số nguyên t_i là thời gian cần thiết để xe buýt di chuyển từ điểm hẹn i đến điểm hẹn $i+1$ (điểm hẹn thứ $N+1$ sẽ là trụ sở làm việc của công ty), số nguyên K_i là số lượng nhân viên đến điểm hẹn i và tiếp đến là K_i số nguyên là các thời điểm đến điểm hẹn của K_i nhân viên.

Các số trên cùng dòng được ghi cách nhau bởi dấu cách.

Giới hạn: $1 \leq M \leq 2000, 1 \leq N, K \leq 200000$.

Kết quả: Ghi ra file văn bản BUS.OUT khoảng thời gian ngắn nhất tìm được.

Ví dụ:

BUS.INP	BUS.OUT
3 5 1 2 0 1 1 1 2 1 4 0 2 3 4	4

CODE tham khảo

```
#include <iostream>
#include <fstream>
#include <vector>
#include <map>
using namespace std;
const int debug = 0;
int n, m, k[200005];
long long t[200005], sum[200005], ans;
vector<long long> v[200005];
map<long long, int> mp;

int main()
{
    ifstream cin("bus.inp");
    ofstream cout("bus.out");
    ios_base::sync_with_stdio(0);
    cin.tie(NULL);
    cin >> n >> m;
    for(int i = 1; i <= n; i++)
    {
        cin >> t[i] >> k[i];
        for(int j = 1; j <= k[i]; j++)
        {
            long long x;
            cin >> x;
            v[i].push_back(x);
        }
    }
    for(int i = 1; i <= n; i++)
```



```

{
    sum[i] = sum[i - 1] + t[i];
}
for(int i = 1; i <= n; i++)
{
    for(int j = 0; j < k[i]; j++)
    {
        long long x = v[i][j];
        mp[max(0LL, x - sum[i - 1])]++;
    }
}
ans = sum[n];
int cnt = 0;
for(map<long long, int>::iterator it = mp.begin(); it != mp.end(); it++)
{
    ans = sum[n] + it->first;
    cnt += it->second;
    if(cnt >= m)
    {
        break;
    }
}
cout << ans << "\n";
if(debug)
{
    cout << "n = " << n << ", m = " << m << "\n";
    cout << "t[] =";
    for(int i = 1; i <= n; i++)
    {

```

```

        cout << " " << t[i];
    }
    cout << "\n";
    cout << "k[] =";
    for(int i = 1; i <= n; i++)
    {
        cout << " " << k[i];
    }
    cout << "\n";
    for(int i = 1; i <= n; i++)
    {
        cout << "v[" << i << "] =";
        for(int j = 0; j < k[i]; j++)
        {
            cout << " " << v[i][j];
        }
        cout << "\n";
    }
    cout << "sum[] =";
    for(int i = 1; i <= n; i++)
    {
        cout << " " << sum[i];
    }
    cout << "\n";
    for (map<long long, int>::iterator it = mp.begin(); it != mp.end(); it++)
    {
        cout << "mp[" << it->first << "] = " << it->second << "\n";
    }
}

```

```

    return 0;
}

```

Bài 9. XẾP HÀNG

Để trình diễn một tiết mục trong màn khai mạc Đại hội thể thao quốc tế, đạo diễn Q đã mời n vận động viên tham gia. Theo kịch bản, n vận động viên sẽ được xếp thành một khối có dạng hình chữ nhật gồm một số hàng và một số cột. Cụ thể, các vận động viên đứng ở các vị trí có tọa độ nguyên và liên tiếp nhau, xếp thành các hàng song song với trục tọa độ để tạo thành một khối có dạng hình chữ nhật. Hiện tại, vận động viên thứ i đang ở vị trí (x_i, y_i) , nếu vận động viên này di chuyển đến vị trí (u_i, v_i) thì sẽ mất năng lượng là $|x_i - u_i| + |y_i - v_i|$.

Yêu cầu: Hãy giúp đạo diễn xác định cách xếp hàng để tổng năng lượng di chuyển của cả n vận động viên là nhỏ nhất.

Dữ liệu: Vào từ file văn bản QUEUE.INP gồm

- Dòng đầu ghi hai số nguyên dương n ;
- Tiếp theo là n dòng, dòng thứ i chứa hai số nguyên x_i, y_i , các số có giá trị tuyệt đối không vượt quá 10^9 .

Kết quả: Ghi ra file văn bản QUEUE.OUT gồm một dòng, chứa một số nguyên là tổng năng lượng di chuyển của cả n vận động viên.

Ví dụ:

QUEUE.INP	QUEUE.OUT
3 1 1 1 2 3 3	2

Ràng buộc:

- Có 20% số test ứng với 20% số điểm của bài có $0 \leq x_i, y_i \leq 100$;
- $n \leq 11$ và n là số nguyên tố;

- Có 20% test khác ứng với 20% số điểm của bài có $0 \leq x_i, y_i \leq 100; n \leq 11$;
- Có 20% test khác ứng với 20% số điểm của bài có $0 \leq x, y_i \leq 10000; n < 1000$ và n là số nguyên tố;
- Có 20% test khác ứng với 20% số điểm của bài có $n < 50000$ và n là số nguyên tố;
- Có 20% số test còn lại ứng với 20% số điểm của bài có $n \leq 50000$.

CODE tham khảo

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

```
using namespace std;
```

```
#define TASK "QUEUE"
```

```
#define input_file TASK".INP"
```

```
#define output_file TASK".OUT"
```

```
#define fi first
```

```
#define se second
```

```
#define IT(i,v) for (i=v.begin();i!=v.end();i++)
```

```
#define FORU(i,a,b) for (int i=(a); i<=(b); i++)
```

```
#define FORD(i,a,b) for (int i=(a); i>=(b); i--)
```

```
#define abs(x) ((x)>=0?(x):- (x))
```

```
#define sqr(x) (x)*(x)
```

```
#define max3(a,b,c) max((a),max((b),(c)))
```

```
#define min3(a,b,c) min((a),min((b),(c)))
```

```
#define SET_ARR(a,v) memset(a,v,sizeof(a))
```

```
#define ALL(x) (x).begin(),(x).end()
```

```
typedef long long ll;
```

```
typedef unsigned long long ull;
```

```
typedef pair<int,int> ii;  
typedef pair<int,ii> iii;  
typedef pair<double,double> dd;  
typedef pair<double,dd> ddd;  
const int oo = 1123123123;  
const ll loo = 1123123123123123;
```

```
const int MAXN = 50004;
```

```
struct point {  
    int x, y;  
} a[MAXN], b[MAXN];
```

```
int n;
```

```
bool isPrime (int n) {  
    return true;  
}
```

```
void Read() {  
    scanf("%d", &n);  
    FORU(i, 1, n) {  
        int x, y;  
        scanf("%d %d", &x, &y);  
        a[i].x = x;  
        a[i].y = y;  
        b[i] = a[i];  
    }  
}
```

```

bool cmpx(point x, point y) {
    if (x.x < y.x) return true;
    return false;
}

```

```

bool cmpy(point x, point y) {
    if (x.y < y.y) return true;
    return false;
}

```

```

struct Case1 {
    void Run() {
        sort(a + 1, a + n + 1, cmpx);
        sort(b + 1, b + n + 1, cmpy);
        int xx = a[n / 2 + 1].x;
        int cost = 0;
        FORU(i, 1, n) {
            cost += abs(a[i].x - xx);
        }
        int yy = b[n / 2 + 1].y;
        int tmp = n / 2;
        FORU(i, 1, n / 2 + 1) {
            cost += abs(b[i].y - yy) - tmp;
            tmp--;
        }
        tmp = n / 2;
        FORU(i, n / 2 + 2, n) {
            cost += abs(b[i].y - yy) - tmp;

```

```

        tmp--;
    }
    cout << cost << "\n";
}

} Solve_NPrime;

int main() {
#ifdef ONLINE_JUDGE
    freopen(input_file, "r", stdin);
    freopen(output_file, "w", stdout);
#endif
    Read();
    if (isPrime(n)) {
        Solve_NPrime.Run();
    }
}

```

Bài 10. PHÂN NHÓM

Để có công việc làm thêm cho những X-men, giáo sư X mở thêm trường dạy chó X-dogs. Trường có n con chó đánh số từ 1 tới n . Mỗi con chó có thể bắt hòa với không quá 3 con chó khác. Giả thiết quan hệ bắt hòa ở đây là quan hệ hai chiều tức là nếu con chó a bắt hòa với con chó b thì con chó b cũng bắt hòa với con chó a và ngược lại. Hàng ngày các X-men có nhiệm vụ dắt chó đi dạo. Để giúp lũ chó được thoải mái, hạn chế việc xảy ra xung đột diện rộng, các X-men muốn chia n con chó vào hai nhóm đi hai nơi khác nhau sao cho trong mỗi nhóm, mỗi con chó bắt hòa với không quá 1 con chó khác.

Yêu cầu: Hãy giúp chia các con chó thành hai nhóm thỏa mãn yêu cầu trên.

Dữ liệu: Vào từ file văn bản GROUPDIV.INP

- Dòng đầu chứa số nguyên dương $n \leq 3.105$

- Dòng thứ i trong n dòng tiếp theo chứa các không quá 4 số: số đầu là số lượng những con chó bất hòa với con chó thứ i , tiếp theo là chỉ số của các con chó đó đó.

Kết quả: Ghi ra file văn bản GROUPDIV.OUT

- Dòng 1 ghi từ YES nếu có phương án chia n con chó vào hai nhóm thỏa mãn yêu cầu, ghi từ NO nếu không tồn tại phương án Trong trường hợp có tồn tại phương án chia nhóm
 - Dòng 2 ghi chỉ số các con chó trong nhóm thứ nhất
 - Dòng 3 ghi chỉ số các con chó trong nhóm thứ hai
- Các số trên một dòng phải ghi cách nhau ít nhất một dấu cách.

Ví dụ:

GROUPDIV.INP	GROUPDIV.OUT
7	YES
3 2 3 4	1 4 6 7
3 1 3 6	2 3 5
2 1 2	
2 1 5	
1 4	
1 2	
0	

CODE tham khảo

```
#include <bits/stdc++.h>
#define Task "GROUPDIV"

using namespace std;

vector<int> hate[300005];
bool gr[300005];
int n;
```



```

int cnt(int u){
    int res = 0;
    for(int v: hate[u]) res += (gr[u] == gr[v]);
    return res;
}

int main(){
    freopen(Task".inp","r",stdin);
    freopen(Task".out","w",stdout);
    ios::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);
    cin >> n;
    for(int i = 1, x, ssize; i<=n; ++i){ cin >> ssize;
        for(int j = 1; j<=ssize; ++j) cin >> x, hate[i].push_back(x); }
    queue<int> q;
    for(int i = 1; i<=n; ++i) q.push(i);
    while (!q.empty()){
        int u = q.front(); q.pop();
        if (cnt(u) > 1){
            gr[u] ^= 1;
            for(int v : hate[u]) if (gr[u] == gr[v]) q.push(v);
        }
    }
    cout << "YES\n";
    for(int u = 1; u<=n; ++u) if (gr[u]) cout << u << " "; cout << "\n";
    for(int u = 1; u<=n; ++u) if (!gr[u]) cout << u << " "; cout << "\n";
    return 0;
}

```

TÀI LIỆU THAM KHẢO

1. <http://codeforces.com/>
2. <https://vn.spoj.com/>
3. Sách giáo khoa chuyên tin tập 1,2,3– nhà xuất bản giáo dục năm 2009.
- 4 . Giải thuật và lập trình của TS. Lê Minh Hoàng ĐHSP Hà Nội.