

```

for(i = 0; i < n; i++)
{
    if (*(ptr + i) <= m)
    {
        continue;
    }

    for (j = i + 1; j < n; j++)
    {
        int temp = *(ptr + i) & *(ptr + j);

        if(temp > m)
        {
            m = temp;
        }
    }
}

printf("%d", m);
}

```

## GRAPH:-

You have already solved this challenge! Though you can run the code with different logic!

Course	Session	Question Information
DS	Graph	Level 1 Challenge 81

**Question description**  
 Consider a network consisting of  $n$  computers and  $m$  connections. Each connection specifies how fast a computer can send data to another computer. Kotivalo wants to download some data from a server. What is the maximum speed he can do this, using the connections in the network?

**Input**  
 The first input line has two integers  $n$  and  $m$ : the number of computers and connections. The computers are numbered  $1, 2, \dots, n$ . Computer 1 is the server and computer  $n$  is Kotivalo's computer. After this, there are  $m$  lines describing the connections. Each line has three integers  $a$ ,  $b$  and  $c$ : computer  $a$  can send data to computer  $b$  at speed  $c$ .

**Output**  
 Print one integer: the maximum speed Kotivalo can download data.

**Constraints**

- $1 \leq n \leq 500$
- $1 \leq m \leq 1000$
- $1 \leq a, b \leq n$
- $1 \leq c \leq 109$

**Logical Test Cases**

**Test Case 1**  
 INPUT (STDIN)  
 4 5  
 1 2 10  
 1 3 100  
 2 4 1  
 3 4 1

**Test Case 2**  
 INPUT (STDIN)  
 4 5  
 1 2 10  
 1 3 100  
 2 4 1  
 3 4 1

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

```
using namespace std;
```

```

using ll = long long;

#define FOR(i,a) for(int i=0; i<(a); i++)
#define FOR(i,a,b) for(int i=(a); i<=(b); i++)

int n, m;

ll adj[501][501], oadj[501][501];

ll flow[501];

bool V[501];

int pa[501];

void link(int i,int h){}

int bfs(int n,int s,int t){return 1;}

bool reachable() {
    memset(V, false, sizeof(V));

    queue<int> Q; Q.push(1); V[1]=1;

    while(!Q.empty()) {
        int i=Q.front(); Q.pop();

        FOR(j,1,n) if (adj[i][j] && !V[j])
            V[j]=1, pa[j]=i, Q.push(j);
    }

    return V[n];
}

int main() {

    bfs(1,1,1);

    link(1,1);

    cin >> n >> m;

    FOR(i,1,n) FOR(j,1,n) adj[i][j] = 0;

    FOR(i,m) {
        ll a,b,c; cin >> a >> b >> c;

        adj[a][b] += c;
    }

    int v, u;

    ll maxflow = 0;

    while(reachable()) {
        ll flow = 1e18;

        for (v=n; v!=1; v=pa[v]) {
            u = pa[v];

            flow = min(flow, adj[u][v]);
        }
    }
}

```

```

maxflow += flow;

for (v=n; v!=1; v=pa[v]) {
    u = pa[v];
    adj[u][v] -= flow;
    adj[v][u] += flow;
}

}

cout << maxflow << '\n';
}

```

The screenshot shows a web browser window with the URL `care.srmup.in/srmcnetelab/#/srmcnetelab/student/home`. The page content includes a notification bar, a navigation menu, and a main area for a coding challenge. The challenge is titled "Question description" and describes a game involving rooms and teleporters. The constraints are  $2 \leq n \leq 500$ ,  $1 \leq m \leq 1000$ , and  $1 \leq a, b \leq n$ . The input format specifies two integers  $n$  and  $m$  on the first line, followed by  $m$  lines of teleporter data. The output format requires printing the maximum number of days and the route descriptions. Two test cases are provided for logical testing.

Test Case 1	Test Case 2
INPUT (STDIN)	INPUT (STDIN)
6 7	6 7
1 2	1 4
1 3	3 5
2 6	4 6
3 4	5 6
3 5	2 2

```
#include <stdio.h>
```

```
#define N 500
```

```
#define M 1000
```

```

struct L {
    struct L *next;
    int h;
} aa[N * 2];

```

```
int ij[M + N], cc[(M + N) * 2], dd[N * 2];
```

```
int bfs(int n,int s,int t) {
```

```

static int qq[N * 2];

int head, cnt, h, i, j, d;

for (i = 0; i < n; i++)
    dd[i] = n;

dd[s] = 0;

head = cnt = 0;

qq[head + cnt++] = s;

while (cnt) {
    struct L *l;

    i = qq[cnt--, head++];

    d = dd[i] + 1;

    for (l = aa[i].next; l; l = l->next)
        if (cc[h = l->h]) {
            j = i ^ ij[h >> 1];

            if (dd[j] == n) {
                dd[j] = d;

                if (j == t)
                    return 1;

                qq[head + cnt++] = j;
            }
        }
    }

return 0;
}

```

```

int dfs(int n, int i, int t) {
    struct L *l;

    int h, j, d;

    if (i == t)
        return 1;

    d = dd[i] + 1;

    for (l = aa[i].next; l; l = l->next)
        if (cc[h = l->h]) {
            j = i ^ ij[h >> 1];

            if (dd[j] == d && dfs(n, j, t)) {

```

```

        cc[h]--, cc[h ^ 1]++;

        return 1;
    }

}

dd[i] = n;

return 0;
}

```

```

int dinic(int n, int s, int t) {

    int f = 0;

    while (bfs(n, s, t))

        while (dfs(n, s, t))

            f++;

    return f;
}

```

```

void link(int i, int j, int h, int c) {

    static struct L l91[(M + N) * 2], *l = l91;

    ij[h] = i ^ j;

    cc[h << 1] = c;

    l->h = h << 1;

    l->next = aa[i].next, aa[i].next = l++;

    l->h = h << 1 ^ 1;

    l->next = aa[j].next, aa[j].next = l++;

}

```

```

int qq[N];

```

```

int path(int i, int t) {

    int cnt = 0;

    while (i != t) {

        struct L *l;

        int h;

        qq[cnt++] = i;
    }
}

```

```

    for (l = aa[i].next; l; l = l->next)
        if (((h = l->h) & 1) == 0 && cc[h ^ 1]) {
            cc[h]++, cc[h ^ 1]--;
            i ^= ij[h >> 1];
            break;
        }
    }

    qq[cnt++] = t;

    return cnt;
}

```

```

int main() {

    int n, m, h, i, j, k, s, t, cnt;

    scanf("%d%d", &n, &m);
    for (h = 0; h < m; h++) {
        scanf("%d%d", &i, &j), i--, j--;
        link(i << 1 ^ 1, j << 1, h, 1);
    }

    for (i = 0; i < n; i++)
        link(i << 1, i << 1 ^ 1, m + i, n);

    s = 0, t = (n - 1) << 1 ^ 1;
    k = dinic(n * 2, s, t);
    printf("%d\n", k);
    while (k--) {
        cnt = path(s, t);
        printf("%d\n", cnt / 2);
        for (i = 0; i < cnt; i += 2)
            printf("%d ", (qq[i] >> 1) + 1);
        printf("\n");
    }

    return 0;
}

```

care.srmup.in/srmncretelab/#/srmncretelab/student/home

You have already solved this challenge! Though you can run the code with different logic!

Course	DS	Session	Graph	Question Information	Level 1	Challenge 83
<p><b>Question description</b></p> <p>Teddy is visiting the country wonderland. wonderland has <math>n</math> cities and <math>m</math> bidirectional roads. There are <math>k</math> types of tokens. Token <math>i</math> costs <math>c_i</math>. The costs of the tokens are such that for all <math>2 \leq i \leq k</math>, <math>c_i \geq 2c_{i-1}</math>. For each road, you need to have a particular set of tokens, if you want to travel it. Note that you don't have to give the tokens, you just need to show them. Thus, one token can be used at any number of roads, where it is required. Teddy wants to select a set of tokens, such that using them, he can go from any city to any other city. You have to help him minimize the total cost of tokens he buys.</p> <p><b>Constraints</b></p> <p><math>1 \leq n \leq 10^5</math></p> <p><math>1 \leq m \leq 10^5</math></p> <p>No road connects a city to the same city. However, there can be multiple roads between two cities.</p> <p><math>1 \leq c_i \leq 10^{18}</math></p> <p>For all <math>2 \leq i \leq k</math>, <math>c_i \geq 2c_{i-1}</math></p> <p><b>Input:</b></p> <ul style="list-style-type: none"> <li>The first line contains three space separated integers, <math>n</math>, <math>m</math> and <math>k</math>.</li> <li>The second line contains <math>k</math> space separated integers, where the <math>i^{\text{th}}</math> integer denotes the price of <math>i^{\text{th}}</math> token, i.e. <math>c_i</math>.</li> <li><math>i^{\text{th}}</math> of the next <math>m</math> lines contains three integers <math>u_i, v_i, l_i</math>, where <math>l_i</math> is the number of tokens required by the <math>i^{\text{th}}</math> road, followed by <math>l_i</math> indices denoting the tokens required. This road connects cities <math>u_i</math> and <math>v_i</math>.</li> </ul> <p><b>Output:</b></p> <ul style="list-style-type: none"> <li>Print one integer containing the minimum cost of tokens Teddy has to buy, such that he can travel from any city to any other city. If it is impossible to choose such a set of tokens, print <math>-1</math>.</li> </ul> <p>Logical Test Cases</p> <p>Test Case 1</p> <p>Test Case 2</p>						

```
#include <stdio.h>
```

```
#if defined( _WIN32 )
```

```
typedef __int64 az_int64_t;
```

```
typedef unsigned __int64 az_uint64_t;
```

```
#define I64(x) x ## I64
```

```
#define F64 "I64"
```

```
#else
```

```
typedef long long az_int64_t;
```

```
typedef unsigned long long az_uint64_t;
```

```
#define I64(x) x ## ll
```

```
#define F64 "ll"
```

```
#endif
```

```
#define MAXN (100*1024)
```

```
struct link
```

```
{
```

```
    az_int64_t t;
```

```
    int u, v;
```

```
};
```

```
struct link links[MAXN];
```

```

int n, m, k;

az_int64_t c[64];

int gr[MAXN];

int getgr( int g )
{
    return (g == gr[g]) ? g : (gr[g] = getgr( gr[g] ));
}

int test( az_int64_t r )
{
    int i, left = n-1, u, v;

    for(i=1;i<=n;++i) gr[i] = i;

    for( i = 0; i < m; ++i)

        if( (links[i].t & r) == 0 &&

            (u = getgr( links[i].u )) != (v = getgr( links[i].v )) )

        {

            gr[v] = u;

            if( --left == 0 ) return 1;

        }

    return 0;
}

int main( void )
{
    az_int64_t rejected = 0, sum = 0;

    int i;

    scanf( "%d %d %d", &n, &m, &k);

    for( i = 0; i < k; ++i) scanf( "%" F64 "d", &c[i]);

    for( i = 0; i < m; ++i)

    {

        int l, id;

        scanf( "%d %d %d", &links[i].u, &links[i].v, &l);

        while( l-- > 0 )

        {

            scanf( "%d", &id);

```



```

links[i].t |= l64(1) << (id-1);
}
}

if( !test( 0 ) )
{
printf( "-1\n" );
return 0;
}

for( i = k-1; i >= 0; --i)
{
az_int64_t f = l64(1) << i;
if( test( rejected | f ) ) rejected |= f; else sum += c[i];
}

printf( "%F64 "d\n", sum);
return 0;
}

```

Course DS Session Graph Question Information Level 1 Challenge 84

**Question description**  
You are playing a game consisting of  $n$  planets. Each planet has a teleporter to another planet (or the planet itself). You start on a planet and then travel through teleporters until you reach a planet that you have already visited before. Your task is to calculate for each planet the number of teleportations there would be if you started on that planet.

**Input**  
The first input line has an integer  $n$ : the number of planets. The planets are numbered  $1, 2, \dots, n$ .  
The second line has  $n$  integers  $i_1, i_2, \dots, i_n$ : for each planet, the destination of the teleporter. It is possible that  $i_i = i$ .

**Output**  
Print  $n$  integers according to the problem statement.

**Constraints**

- $1 \leq n \leq 10^5$
- $1 \leq i_i \leq n$

**Logical Test Cases**

Test Case 1	Test Case 2
INPUT (STDIN) 5 2 4 3 1 4	INPUT (STDIN) 6 2 1 3 5 6 4

```

#include <stdio.h>

#include <string.h>

```

```
#define N 200000
```

```
int main() {
```

```
    static int aa[N], cc[N], dd[N], qq[N];
```

```
    int n, i, j, c, d, q, cnt;
```

```
    scanf("%d", &n);
```

```
    for (i = 0; i < n; i++)
```

```
        scanf("%d", &aa[i]), aa[i]--;
```

```
    memset(cc, -1, n * sizeof *cc);
```

```
    cnt = 0;
```

```
    for(i = 0; i < n; i++) {
```

```
        if (cc[i] != -1)
```

```
            continue;
```

```
        d = 0;
```

```
        j = i;
```

```
        while (cc[j] == -1) {
```

```
            cc[j] = -2;
```

```
            d++;
```

```
            j = aa[j];
```

```
        }
```

```
        if (cc[j] == -2) {
```

```
            c = cnt++;
```

```
            q = 0;
```

```
            while (cc[j] == -2) {
```

```
                cc[j] = c;
```

```
                q++;
```

```
                j = aa[j];
```

```
            }
```

```
            qq[c] = q;
```

```
            d -= q;
```

```
        } else {
```

```
            c = cc[j];
```

```
            d += dd[j];
```

```
        }
```

```
        j = i;
```

```
        while (cc[j] == -2) {
```

```

        cc[j] = c;

        dd[j] = d--;

        j = aa[j];
    }
}

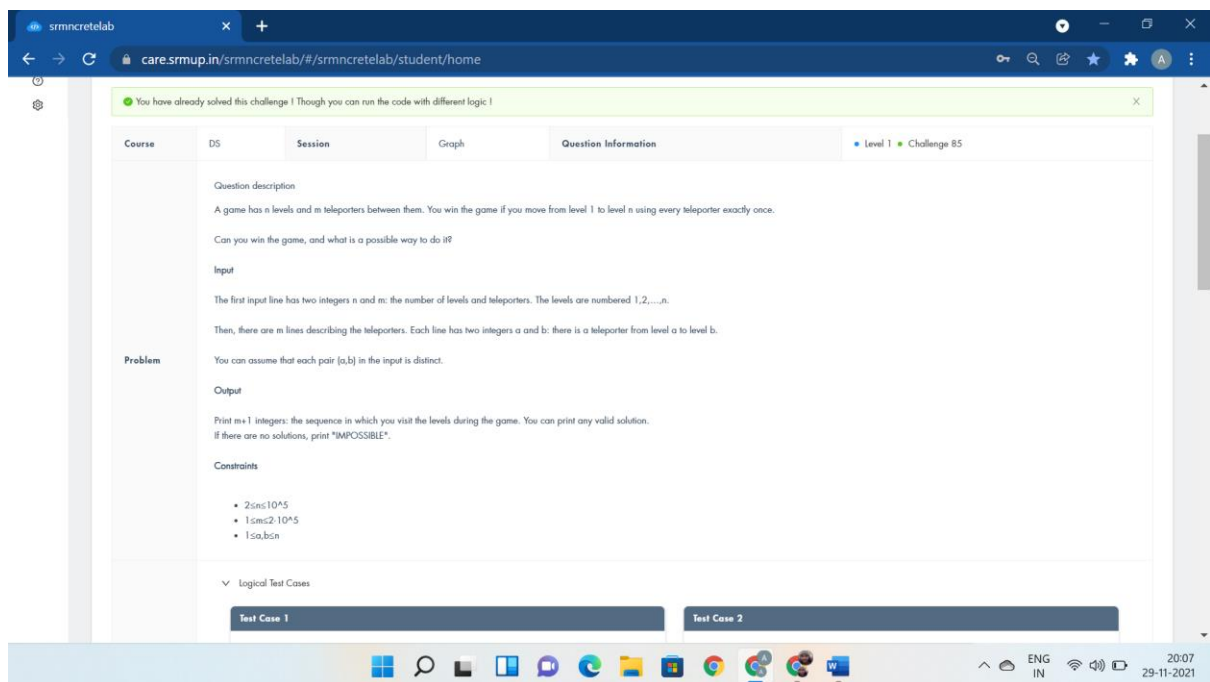
for (i = 0; i < n; i++)

    printf("%d ", dd[i] + qq[cc[i]]);

printf("\n");

return 0;
}

```



```
#include <stdio.h>
```

```
#define N 100000
```

```
#define M 200000
```

```
struct L {
```

```
    struct L *next;
```

```
    int j;
```

```
} *aa[N];
```

```
struct L *new_L(int j) {
```

```
    static struct L l91[M + 1 + M], *l = l91;
```

```

l->j = j;

return l++;
}

```

```

void link(int i,int j) {

    struct L *l = new_L(j);

    l->next = aa[i]; aa[i] = l;
}

```

```

void hierholzer(struct L *e) {

    struct L *f = e->next, *l;

    int i = e->j;

```

```

    while ((l = aa[i])) {

        aa[i] = l->next;

        e = e->next = new_L(l->j);

        i = l->j;
    }

    e->next = f;
}

```

```

int main() {

    static int din[N], dout[N];

    struct L *e_, *e;

    int n, m, h, i, j;

    scanf("%d%d", &n, &m);

    for (h = 0; h < m; h++) {

        scanf("%d%d", &i, &j), i--, j--;

        link(i, j);

        dout[i]++, din[j]++;
    }

    if (dout[0] - din[0] != 1 || din[n - 1] - dout[n - 1] != 1) {

        printf("IMPOSSIBLE\n");

        return 0;
    }
}

```

```

for (i = 1; i < n - 1; i++)

    if (dout[i] != din[i]) {

        printf("IMPOSSIBLE\n");

        return 0;

    }

e_ = new_L(0);

m++;

hierholzer(e_);

for (e = e_; e; e = e->next) {

    hierholzer(e);

    m--;

}

if (m != 0) {

    printf("IMPOSSIBLE\n");

    return 0;

}

for (e = e_; e; e = e->next)

    printf("%d ", e->j + 1);

printf("\n");

return 0;

}

```

The screenshot shows a web browser window with the URL `care.srmup.in/srmncretelab/#/srmncretelab/student/home`. A green notification bar at the top states: "You have already solved this challenge! Though you can run the code with different logic!".

The main content area is titled "Level 1 Challenge 86" and contains the following information:

- Course:** DS
- Session:** Graph
- Question Information:** Level 1 Challenge 86

**Question description:**

In the country of India, there are  $N$  cities and  $M$  bi-directional roads. We need to transport essential items from City 1 to all other cities. (There exists a path always) But every road has some positive amount of Toll Charge associated with it say  $C$  (it is not same for all roads). We need to find the minimum amount of charge that it required to deliver essential items for each city. Fortunately, to our rescue we have  $K$  special offers, which means while travelling from City 1 to any other city we can select at most  $K$  roads and we will not be charged for using those roads.

Can you now find the minimum charge that we have to pay to deliver essential items for each city. (Remember we require to provide answers for each destination city separately i.e. we have  $K$  offers for every city and not as a whole)

**Constraints :**

- $1 \leq N \leq 10^5, 1 \leq M \leq 5 \times 10^5$
- $1 \leq U, V \leq N$
- $1 \leq W \leq 10^5$
- $1 \leq K \leq 18$

**Input :**

- First line contain three integers  $N, M, K$ .
- Next  $M$  lines contain three integers  $U, V, W$ , denoting a road between city  $U$  and city  $V$  with Toll Charge  $W$ .

**Output :**

Print  $N$  space separated integers, denoting the minimum charge we require to pay for each city, where first integer represent cost for City 1, second for City 2 and so on.

**Logical Test Cases:**

Test Case 1	Test Case 2
INPUT (STDIN)	INPUT (STDIN)
5 4 1	5 4 1
1 2 2	1 2 3

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

```

using namespace std;

void solve(){}

int main(){

    solve();

    long long int n,m;

        int k;

    cin>>n>>m>>k;

    vector<pair<long long int,long long int>> adjList[n+1];

    for(long long int i=0;i<m;++i){

        long long int a,b,c;

        cin>>a>>b>>c;

        /*if((a==1 && b==n) || (a==n && b==1)){

            cout<<"0\n";

            return 0;

        }*/

        adjList[a].push_back(pair<long long int,long long int>{b,c});

        adjList[b].push_back(pair<long long int,long long int>{a,c});

    }

    vector<vector<long long int>> dp(n+1,vector<long long int>(k+1,1000000000000));

    queue<pair<long long int,long long int>> q;

    dp[1][0]=0;

    q.push(pair<long long int,long long int>{0,1});

    while(!q.empty()){

        long long int from=q.front().first;

        long long int now=q.front().second;

        q.pop();

        bool change=false;

        for(auto to:adjList[now]){

            if(to.first==from){

                continue;

            }

            for(int i=0;i<=k;++i){

                if(i!=k && dp[to.first][i+1] > dp[now][i]){

                    dp[to.first][i+1] = dp[now][i];

                    change=true;

                }

            }

            //for(int i=0;i<2;++i){

```

```

        if(dp[to.first][i] > dp[now][i]+to.second){
            dp[to.first][i] = dp[now][i]+to.second;
            change=true;
        }
    //}

    }

    if(change){
        q.push(pair<long long int,long long int>{now,to.first});
    }
}

for(long long int i=1; i<=n; i++)
{
    long long int ans = 1000000000000000;
    for(long long int j =0; j<=k; j++)
    {
        ans = min(ans,dp[i][j]);
    }
    cout<<ans<<" ";
}

return 0;
}

```

srmmcretelab

care.srmup.in/srmmcretelab/#/srmmcretelab/student/home

You have already solved this challenge! Though you can run the code with different logic!

Course	DS	Session	Graph	Question Information
				Level 1 Challenge 87

**Problem**

**Question description**

There are  $n$  boys and  $m$  girls in a school. Next week a school dance will be organized. A dancing pair consists of a boy and a girl, and there are  $k$  potential pairs.

Your task is to find out the maximum number of dance pairs and show how this number can be achieved.

**Constraints**

- $1 \leq n, m \leq 500$
- $1 \leq k \leq 1000$
- $1 \leq a \leq n$
- $1 \leq b \leq m$

**Input**

The first input line has three integers  $n$ ,  $m$  and  $k$ : the number of boys, girls, and potential pairs. The boys are numbered  $1, 2, \dots, n$ , and the girls are numbered  $1, 2, \dots, m$ .

After this, there are  $k$  lines describing the potential pairs. Each line has two integers  $a$  and  $b$ : boy  $a$  and girl  $b$  are willing to dance together.

**Output**

First print one integer  $r$ : the maximum number of dance pairs. After this, print lines describing the pairs. You can print any valid solution.

Logical Test Cases

Test Case 1

Test Case 2

20:08 29-11-2021

```
#include <stdio.h>
```

```
#define N 500
```

```
#define M 1000
```

```
struct L {
    struct L *next;
    int v;
} aa[N + 1];
```

```
int vv[N + 1], uu[N + 1], dd[N + 1];
```

```
void link(int u, int v) {
    static struct L l91[M], *l = l91;

    l->v = v;
    l->next = aa[u].next, aa[u].next = l++;
}
```

```
int bfs(int n) {
    static int qq[N];
    int u, head, cnt, d;
```



```

head = cnt = 0;

dd[0] = n;

for (u = 1; u <= n; u++)

    if (vv[u] == 0) {

        dd[u] = 0;

        qq[head + cnt++] = u;

    } else

        dd[u] = n;

while (cnt) {

    struct L *l;

    u = qq[cnt--, head++];

    d = dd[u] + 1;

    for (l = aa[u].next; l; l = l->next) {

        int v = l->v, w = uu[v];

        if (dd[w] == n) {

            dd[w] = d;

            if (w == 0)

                return 1;

            qq[head + cnt++] = w;

        }

    }

}

return 0;

}

```

```

int dfs(int n, int u) {

    struct L *l;

    int d;

    if (u == 0)

        return 1;

    d = dd[u] + 1;

    for (l = aa[u].next; l; l = l->next) {

        int v = l->v, w = uu[v];

        if (dd[w] == d && dfs(n, w)) {

```

```

        vv[u] = v;

        uu[v] = u;

        return 1;

    }

}

dd[u] = n;

return 0;

}

int hopcroft_karp(int n) {

    int m = 0;

    while (bfs(n)) {

        int u;

        for (u = 1; u <= n; u++)

            if (vv[u] == 0 && dfs(n, u))

                m++;

    }

    return m;

}

int main() {

    int n, n_, m, u, v;

    scanf("%d%d%d", &n, &n_, &m);

    while (m--) {

        scanf("%d%d", &u, &v);

        link(u, v);

    }

    printf("%d\n", hopcroft_karp(n));

    for (u = 1; u <= n; u++)

        if (vv[u])

            printf("%d %d\n", u, vv[u]);

    return 0;

}

```

care.srmup.in/srmncretelab/#/srmncretelab/student/home

You have already solved this challenge! Though you can run the code with different logic!

Course DS Session Graph Question Information Level 1 Challenge 88

**Question description**

Byteland has  $n$  cities and  $m$  roads between them. The goal is to construct new roads so that there is a route between any two cities. Your task is to find out the minimum number of roads required, and also determine which roads should be built.

**Constraints**

$1 \leq n \leq 10^5$   
 $1 \leq m \leq 2 \cdot 10^5$   
 $1 \leq a, b \leq n$

**Input**

The first input line has two integers  $n$  and  $m$ : the number of cities and roads. The cities are numbered  $1, 2, \dots, n$ .

After that, there are  $m$  lines describing the roads. Each line has two integers  $a$  and  $b$ : there is a road between those cities.

A road always connects two different cities, and there is at most one road between any two cities.

**Output**

First print an integer  $k$ : the number of required roads.

Then, print  $k$  lines that describe the new roads. You can print any valid solution.

Logical Test Cases

Test Case 1 Test Case 2

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

```
using namespace std;
```

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

```
#define trav(a, x) for(auto& a : x)
```

```
#define all(x) begin(x), end(x)
```

```
#define sz(x) (int)(x).size()
```

```
typedef long long ll;
```

```
typedef pair<int, int> pii;
```

```
typedef vector<int> vi;
```

```
vi val, comp, z, cont;
```

```
int Time, ncomps;
```

```
template<class G, class F> int dfs(int j, G& g, F& f) {
```

```
    int low = val[j] = ++Time, x; z.push_back(j);
```

```
    trav(e, g[j]) if (comp[e] < 0)
```

```
        low = min(low, val[e] ? dfs(e, g, f);
```

```
    if (low == val[j]) {
```

```
        do {
```

```
            x = z.back(); z.pop_back();
```

```
            comp[x] = ncomps;
```

```
            cont.push_back(x);
```

```

    } while (x != j);

    f(cont); cont.clear();

    ncomps++;
}

return val[j] = low;
}

template<class G, class F> void scc(G& g, F f) {

    int n = sz(g);

    val.assign(n, 0); comp.assign(n, -1);

    Time = ncomps = 0;

    rep(i,0,n) if (comp[i] < 0) dfs(i, g, f);
}

int main() {

    cin.sync_with_stdio(0); cin.tie(0);

    cin.exceptions(cin.failbit);

    int n, m;

    cin >> n >> m;

    vector<vi> g(n);

    while(m--) {

        int a, b;

        cin >> a >> b;

        a--, b--;

        g[a].push_back(b);

        g[b].push_back(a);

    }

    vi r;

    scc(g, [&](vi &c) { r.push_back(c[0]); });

    cout << sz(r)-1 << '\n';

    rep(i, 1, sz(r))

    cout << r[0]+1 << " " << r[i]+1 << '\n';

    return 0;
}

```

care.srmup.in/srmncretelab/#srmncretelab/student/home

You have already solved this challenge! Though you can run the code with different logic!

Course	DS	Session	Graph	Question Information
				Level 1 Challenge 89

**Problem**

**Question description**  
Your task is to deliver mail to the inhabitants of a city. For this reason, you want to find a route whose starting and ending point are the post office, and that goes through every street exactly once.

**Input**  
The first input line has two integers  $n$  and  $m$ : the number of crossings and streets. The crossings are numbered  $1, 2, \dots, n$ , and the post office is located at crossing 1.  
After that, there are  $m$  lines describing the streets. Each line has two integers  $a$  and  $b$ : there is a street between crossings  $a$  and  $b$ . All streets are two-way streets.  
Every street is between two different crossings, and there is at most one street between two crossings.

**Output**  
Print all the crossings on the route in the order you will visit them. You can print any valid solution.  
If there are no solutions, print "IMPOSSIBLE".

**Constraints**

- $2 \leq n \leq 10^5$
- $1 \leq m \leq 2 \cdot 10^5$
- $1 \leq a, b \leq n$

Logical Test Cases

Test Case 1	Test Case 2
INPUT (STDIN)	INPUT (STDIN)

```
#include <stdio.h>
```

```
#define N 100000
```

```
#define M 200000
```

```
struct L {
    struct L *next;
    int h;
} *aa[N];
```

```
int ij[M + 1];
char lazy[M + 1];
```

```
struct L *new_L(int h) {
    static struct L l91[M * 2 + 1 + M], *l = l91;

    l->h = h;
    return l++;
}
```

```
void link(int i, int h) {
    struct L *l = new_L(h);
```

```

l->next = aa[i]; aa[i] = l;
}

```

```

void hierholzer(struct L *e, int i) {
    struct L *f = e->next, *l;

    while ((l = aa[i])) {
        int h = l->h;

        if (lazy[h])
            aa[i] = l->next;
        else {
            lazy[h] = 1;
            e = e->next = new_L(h);
            i ^= ij[h];
        }
    }
    e->next = f;
}

```

```

int main() {
    static int dd[N];

    struct L *e_, *e;

    int n, m, h, i, j;

    scanf("%d%d", &n, &m);
    for (h = 1; h <= m; h++) {
        scanf("%d%d", &i, &j), i--, j--;
        ij[h] = i ^ j;
        link(i, h), link(j, h);
        dd[i]++, dd[j]++;
    }
    for (i = 0; i < n; i++)
        if (dd[i] % 2) {
            printf("IMPOSSIBLE\n");
            return 0;
        }
    e_ = new_L(0);
}

```

```

i = 0;

m++;

for (e = e_; e = e->next) {

    i ^= ij[e->h];

    hierholzer(e, i);

    m--;

}

if (m != 0) {

    printf("IMPOSSIBLE\n");

    return 0;

}

i = 0;

for (e = e_; e = e->next) {

    i ^= ij[e->h];

    printf("%d ", i + 1);

}

printf("\n");

return 0;

}

```

The screenshot shows a web browser window with the URL `care.srmup.in/srmncretelab/#/srmncretelab/student/home`. The page content includes a notification bar, a navigation menu, and a main content area for a coding challenge.

**Notification:** You have already solved this challenge! Though you can run the code with different logic!

**Course:** DS    **Session:** Graph    **Question Information:** Level 1 Challenge 90

**Question description:** You are given a directed graph, and your task is to find out if it contains a negative cycle, and also give an example of such a cycle.

**Constraints:**

- $1 \leq n \leq 2500$
- $1 \leq m \leq 5000$
- $1 \leq a, b \leq n$
- $-10^9 \leq c \leq 10^9$

**Input Format:** The first input line has two integers  $n$  and  $m$ : the number of nodes and edges. The nodes are numbered  $1, 2, \dots, n$ . After this, the input has  $m$  lines describing the edges. Each line has three integers  $a, b$ , and  $c$ : there is an edge from node  $a$  to node  $b$  whose length is  $c$ .

**Output Format:** If the graph contains a negative cycle, print first "YES", and then the nodes in the cycle in their correct order. If there are several negative cycles, you can print any of them. If there are no negative cycles, print "NO".

**Logical Test Cases:**

Test Case 1	Test Case 2
<b>INPUT (STDIN)</b> 4 5 1 2 1 2 4 1 3 1 1 4 1 -3 4 3 -2	<b>INPUT (STDIN)</b> 5 6 1 2 1 2 4 2 3 1 -1 5 1 -5 4 3 5

```
#include <stdio.h>
```

```
#define N 2500
```

```
#define M 5000
```

```
int main() {
```

```
    static int aa[M], bb[M], cc[M], pp[N], ii[1 + N];
```

```
    static char used[N];
```

```
    static long long dd[N];
```

```
    int n, m, h, r, a, b, c, k;
```

```
    long long d;
```

```
    scanf("%d%d", &n, &m);
```

```
    for (h = 0; h < m; h++)
```

```
        scanf("%d%d%d", &aa[h], &bb[h], &cc[h]), aa[h]--, bb[h]--;
```

```
    for (r = 0; r < n; r++)
```

```
        for (h = 0; h < m; h++) {
```

```
            a = aa[h], b = bb[h], c = cc[h];
```

```
            d = dd[a] + c;
```

```
            if (dd[b] > d) {
```

```
                dd[b] = d;
```

```
                pp[b] = a;
```

```
                if (r == n - 1) {
```

```
                    while (!used[b]) {
```

```
                        used[b] = 1;
```

```
                        b = pp[b];
```

```
                    }
```

```
                    k = 0;
```

```
                    while (used[b]) {
```

```
                        used[b] = 0;
```

```
                        ii[k++] = b;
```

```
                        b = pp[b];
```

```
                    }
```

```
                    ii[k++] = b;
```

```
                    printf("YES\n");
```

```
                    while(k--)
```

```
                        printf("%d ", ii[k] + 1);
```

```
                    printf("\n");
```

```
                    return 0;
```

```
                }
```

```
        }
```



```

    }

    printf("NO\n");

    return 0;
}

```

## HASHING :-

You have already solved this challenge ! Though you can run the code with different logic !

Course	DS	Session	Hashing	Question Information
				Level 1 Challenge 91

**Problem Description**

You are given with integers  $a, b, c, d, m$ . These represent the modular equation of a curve  $y^2 \bmod m = (ax^3 + bx^2 + cx + d) \bmod m$

Also, you are provided with an array  $A$  of size  $N$ . Now, your task is to find the number of pairs in the array that satisfy the given modular equation.

If  $(A_i, A_j)$  is a pair then  $A_j^2 \bmod m = (aA_i^3 + bA_i^2 + cA_i + d) \bmod m$ .

Since the answer could be very large output it modulo  $10^9 + 7$ .

**Note:** A pair is counted different from some other pair if either  $A_i$  of the two pairs is different or  $A_j$  of the two pairs is different. Also for the convenience of calculations, we may count  $(A_i, A_i)$  as a valid pair if it satisfies given constraints.

**Constraints**

- $1 \leq T \leq 10$
- $1 \leq N \leq 10^5$
- $-2 \times 10^9 \leq a, b, c, d, A_i \leq 2 \times 10^9$
- $1 \leq m \leq 2 \times 10^9$

**Input Format**

First line of the input contains number of test cases  $T$ .

First line for each test case consists of 5 space-separated integers  $a, b, c, d, m$ , corresponding to modular equation given.

Next line contains a single integer  $N$ .

Next line contains  $N$  space-separated integers corresponding to values of array  $A$ .

**Output Format**

For each test case, output a single line corresponding to number of valid pairs in the array mod  $10^9 + 7$ .

Logical Test Cases

```

#include <bits/stdc++.h>

using namespace std;

const int md = 1E9 + 7;

map<long long, int> mp;

int main() {

    int t;

    cin >> t;

    while(t--) {

        long long a, b, c, d, m;

        cin >> a >> b >> c >> d >> m;

        int n;

        cin >> n;

        int arr[n];

        for(int i = 0; i < n; i++) {

            cin >> arr[i];

            mp[(((arr[i] * arr[i]) % m) + m) % m]++;

```