```
[1]//********* Bismillahir Rahmanir Rahim *********

// mahfahim51

#include<bits/stdc++.h>
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
using namespace std;
#define pb push_back
#define nl cout<<"\n"
#define all(x) x.begin(), x.end()
#define allr(x) x.rbegin(), x.rend()
#define f(a,b,c) for(int a=b;a<c;a++)
#define cin(vec,n) f(i,0,n){cin >> vec[i];}
#define endl "\n"
const int NN = 2e5+10;
const int MM = 1e9+7;
#define ll long long
#define ldb long double
template <typename T> using pbds = tree<T, null_type, less<T>, rb_tree_tag, tree_order_statistics_node_update>;
//template <typename T> using pbds = tree<T, null_type, less_equal<T>, rb_tree_tag, tree_order_statistics_node_update>;

void solve(void)
{

}
signed main()
{
    ios::sync_with_stdio(false);
    cin.tie(NULL);

    int t=1;
    //cin >> t;
    while(t--)
    {
      solve();
    }
    return 0;
    //********* Alhamdulillah *********
}

//……………………………………………………………………………………………………………………………………………
```

```
// bit manipulation

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

int check_kth_bit_on_or_off(int x, int k) {
    return (x >> k) & 1;
}

void print_on_and_off_bits(int x) {
    for (int k = 0;k <= 31;k++) {
        if (check_kth_bit_on_or_off(x, k)) {
            cout << 1 << " ";
        }
```

```
// path printing

#include <bits/stdc++.h>
using namespace std;
vector<int> v[1005];
bool vis[1005];
int level[1005];
int parent[1005];
void bfs(int src)
{
    queue<int> q;
    q.push(src);
    vis[src] = true;
    level[src] = 0;
```

[1] PSTU_Logic_Builder

```
        else {
            cout << 0 << " ";
        }
    }
    cout << '\n';
}

int turn_on_kth_bit(int x, int k) {
    return (x | (1 << k));
}

int turn_off_kth_bit(int x, int k) {
    return (x & (~(1 << k)));
}

int toggle_kth_bit(int x, int k) {
    return (x ^ (1 << k));
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(NULL);

    // cout << check_kth_bit_on_or_off(44, 3) << '\n';
    // cout << check_kth_bit_on_or_off(44, 4) << '\n';
    // print_on_and_off_bits(44);
    // cout << turn_on_kth_bit(44, 4);
    // cout << turn_off_kth_bit(44, 3);
    cout << toggle_kth_bit(44, 3) << '\n';
    return 0;
}
//………………………………………………………
// bit masking

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

int main() {
    ios::sync_with_stdio(false);
    cin.tie(NULL);

    int n;
    cin >> n;

    for (int i = 0;i < (1 << n);i++) {
        for (int k = 0;k < n;k++) {
            if ((i >> k) & 1) {
                cout << 1 << " ";
            }
            else {
                cout << 0 << " ";
            }
        }
        cout << '\n';
    }

    return 0;
}

//………………………………………………………………

// binary search

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

int main() {
    ios::sync_with_stdio(false);
    cin.tie(NULL);
```

```
    while (!q.empty())
    {
        int par = q.front();
        q.pop();
        for (int child : v[par])
        {
            if (vis[child] == false)
            {
                q.push(child);
                vis[child] = true;
                level[child] = level[par] + 1;
                parent[child] = par;
            }
        }
    }
}
int main()
{
    int n, e;
    cin >> n >> e;
    while (e--)
    {
        int a, b;
        cin >> a >> b;
        v[a].push_back(b);
        v[b].push_back(a);
    }
    int src, des;
    cin >> src >> des;
    memset(vis, false, sizeof(vis));
    memset(level, -1, sizeof(level));
    memset(parent, -1, sizeof(parent));
    bfs(src);
    int x = des;
    vector<int> path;
    while (x != -1)
    {
        path.push_back(x);
        x = parent[x];
    }
    reverse(path.begin(), path.end());
    for (int val : path)
    {
        cout << val << " ";
    }
    return 0;
}


// bfs on 2D

#include <bits/stdc++.h>
using namespace std;
bool vis[20][20];
int dis[20][20];
vector<pair<int, int>> d = {{0, 1}, {0, -1}, {-1, 0}, {1, 0}};
int n, m;
char a[20][20];
bool valid(int i, int j)
{
    if (i < 0 || i >= n || j < 0 || j >= m)
        return false;
    return true;
}

void bfs(int si, int sj)
{
    queue<pair<int, int>> q;
    q.push({si, sj});
    int n, key;
    cin >> n >> key;
    vector<int> a(n);
    for (int i = 0;i < n;i++) {
        cin >> a[i];
    }

    int l = 0, r = n - 1, mid, idx = -1;
    while (l <= r) {
        mid = (l + r) / 2;
        if (key == a[mid]) {
            idx = mid;
            break;
        }
        else if (key < a[mid]) {
            r = mid - 1;
        }
        else {
            l = mid + 1;
        }
    }

    if (idx == -1) {
        cout << "Element not found" << '\n';
    }
    else {
        cout << "Element found at index " << idx << '\n';
    }

    return 0;
}
```

// binary search // Maximum median

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

int main() {
    ios::sync_with_stdio(false);
    cin.tie(NULL);

    int n, k;
    cin >> n >> k;
    vector<int> a(n);
    for (int i = 0;i < n;i++) {
        cin >> a[i];
    }

    sort(a.begin(), a.end());

    auto ok = [&](long long mid) {
        long long cnt = 0;
        for (int i = (n / 2);i < n;i++) {
            cnt += (a[i] < mid ? (mid - a[i]) : 0);
        }
        return cnt <= k;
        };

    long long l = 1, r = 2e9, mid, ans = 0;
    while (l <= r) {
        mid = l + (r - l) / 2;
        if (ok(mid)) {
            ans = mid;
            l = mid + 1;
        }
        else {
            r = mid - 1;
        }
```

```
    vis[si][sj] = true;
    dis[si][sj] = 0;
    while (!q.empty())
    {
        pair<int, int> par = q.front();
        int a = par.first, b = par.second;
        q.pop();
        for (int i = 0; i < 4; i++)
        {
            int ci = a + d[i].first;
            int cj = b + d[i].second;
            if (valid(ci, cj) == true && vis[ci][cj] == false)
            {
                q.push({ci, cj});
                vis[ci][cj] = true;
                dis[ci][cj] = dis[a][b] + 1;
            }
        }
    }
}
int main()
{
    cin >> n >> m;
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            cin >> a[i][j];
        }
    }
    int si, sj;
    cin >> si >> sj;
    memset(vis, false, sizeof(vis));
    memset(dis, -1, sizeof(dis));
    bfs(si, sj);
    cout << dis[2][3];
    return 0;
}
```

// component

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 5;
vector<int> v[N];
bool vis[N];

void dfs(int src)
{
    cout << src << endl;
    vis[src] = true;
    for (int child : v[src])
    {
        if (vis[child] == false)
            dfs(child);
    }
}
int main()
{
    int n, e;
    cin >> n >> e;
    while (e--)
    {
        int a, b;
        cin >> a >> b;
        v[a].push_back(b);
        v[b].push_back(a);
    }
```

```cpp
    }
    cout << ans << '\n';

    return 0;
}
```

// difference array

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

int main() {
    ios::sync_with_stdio(false);
    cin.tie(NULL);

    int n, q;
    cin >> n >> q;
    vector<int> a(n + 1);
    for (int i = 1;i <= n;i++) {
        cin >> a[i];
    }

    vector<int> d(n + 2);
    for (int i = 1;i <= q;i++) {
        int l, r, x;
        cin >> l >> r >> x;
        d[l] += x;
        d[r + 1] -= x;
    }

    // for (int i = 0;i <= n + 1;i++) {
    //     cout << d[i] << " ";
    // }
    // cout << '\n';
    for (int i = 1;i <= n;i++) {
        d[i] = d[i - 1] + d[i];
    }
    // for (int i = 0;i <= n + 1;i++) {
    //     cout << d[i] << " ";
    // }
    // cout << '\n';
    for (int i = 1;i <= n;i++) {
        cout << a[i] + d[i] << " ";
    }
    cout << '\n';
    return 0;
}
```

//prime factorization

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

int main() {
    ios::sync_with_stdio(false);
    cin.tie(NULL);

    int n;
    cin >> n;
    map<int, int> cnt;

    for (int i = 2;i * i <= n;i++) {
        if (n % i == 0) {
            while (n % i == 0) {
                cnt[i]++;
                n /= i;
            }
        }
```

```cpp
    memset(vis, false, sizeof(vis));
    int c = 0;
    for (int i = 0; i < n; i++)
    {
        if (vis[i] == false)
        {
            dfs(i);
            c++;
        }
    }
    cout << "component - " << c << endl;
    return 0;
}
```

// dfs

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 5;
vector<int> v[N];
bool vis[N];

void dfs(int src)
{
    cout << src << endl;
    vis[src] = true;
    for (int child : v[src])
    {
        if (vis[child] == false)
            dfs(child);
    }
}
int main()
{
    int n, e;
    cin >> n >> e;
    while (e--)
    {
        int a, b;
        cin >> a >> b;
        v[a].push_back(b);
        v[b].push_back(a);
    }
    memset(vis, false, sizeof(vis));
    dfs(0);
    return 0;
}
```

// dfs on 2d

```cpp
#include <bits/stdc++.h>
using namespace std;
char a[20][20];
bool vis[20][20];
vector<pair<int, int>> d = {{0, 1}, {0, -1}, {-1, 0}, {1, 0}};
int n, m;
bool valid(int i, int j)
{
    if (i < 0 || i >= n || j < 0 || j >= m)
        return false;
    return true;
}
void dfs(int si, int sj)
{
    cout << si << " " << sj << endl;
    vis[si][sj] = true;
    for (int i = 0; i < 4; i++)
    }

    if (n > 1) {
        cnt[n]++;
    }

    for (auto [x, y] : cnt) {
        cout << x << " " << y << '\n';
    }
    return 0;
}
```

//prime factiorization using sieve

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

int main() {
    ios::sync_with_stdio(false);
    cin.tie(NULL);

    int n = 20;
    vector<bool> prime(n + 1, true);
    for (int i = 2;i * i <= n;i++) {
        if (prime[i]) {
            for (int j = i + i;j <= n;j += i) {
                prime[j] = false;
            }
        }
    }

    vector<int> all_primes;
    for (int i = 2;i <= n;i++) {
        if (prime[i]) {
            all_primes.push_back(i);
        }
    }

    // for (auto val : all_primes) {
    //     cout << val << " ";
    // }
    // cout << '\n';

    map<int, int> cnt;
    int x, idx = 0;
    cin >> x;
    while (x > 1) {
        if (x % all_primes[idx] == 0) {
            cnt[all_primes[idx]]++;
            x /= all_primes[idx];
        }
        else {
            idx++;
        }
    }

    for (auto [x, y] : cnt) {
        cout << x << " " << y << '\n';
    }
    return 0;
}
```

// sieve

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

int main() {
```

```
    {
        int ci = si + d[i].first;
        int cj = sj + d[i].second;
        if (valid(ci, cj) == true && vis[ci][cj] == false)
        {
            dfs(ci, cj);
        }
    }
}
int main()
{
    cin >> n >> m;
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            cin >> a[i][j];
        }
    }
    int si, sj;
    cin >> si >> sj;
    memset(vis, false, sizeof(vis));
    dfs(si, sj);
    return 0;
}
```

// cycle detect in directed graph

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 5;
bool vis[N];
bool pathVisit[N];
vector<int> adj[N];
bool ans;
void dfs(int parent)
{
    vis[parent] = true;
    pathVisit[parent] = true;
    for (int child : adj[parent])
    {
        if (pathVisit[child])
        {
            ans = true;
        }
        if (!vis[child])
        {
            dfs(child);
        }
    }
    // kaj sesh
    pathVisit[parent] = false;
}
int main()
{
    int n, e;
    cin >> n >> e;
    while (e--)
    {
        int a, b;
        cin >> a >> b;
        adj[a].push_back(b);
        // adj[b].push_back(a);
    }
    memset(vis, false, sizeof(vis));
    memset(pathVisit, false, sizeof(pathVisit));
    ans = false;
    for (int i = 0; i < n; i++)
    ios::sync_with_stdio(false);
    cin.tie(NULL);

    int n;
    cin >> n;
    vector<bool> prime(n + 1, true);
    for (int i = 2;i * i <= n;i++) {
        if (prime[i]) {
            for (int j = i + i;j <= n;j += i) {
                prime[j] = false;
            }
        }
    }

    for (int i = 2;i <= n;i++) {
        if (prime[i]) {
            cout << i << " ";
        }
    }
    cout << '\n';

    return 0;
}
```

// gcd and lcm

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

int gcd(int a, int b) {
    return __gcd(a, b);
}

int lcm(int a, int b) {
    // return ((a * b) / gcd(a, b));
    return ((a / gcd(a, b)) * b);
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(NULL);

    int a, b;
    cin >> a >> b;
    cout << gcd(a, b) << " " << lcm(a, b) << '\n';
    return 0;
}
```

// pbds - 1 //Queries about less or equal elements

```
#include <bits/stdc++.h>
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
using namespace std;
template <typename T> using pbds = tree<T, null_type,
less_equal<T>, rb_tree_tag,
tree_order_statistics_node_update>;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(NULL);

    int n, m;
    cin >> n >> m;
    pbds<int> p;
    for (int i = 1;i <= n;i++) {
        int x;
        cin >> x;
```

```
    {
        if (!vis[i])
        {
            dfs(i);
        }
    }
    if (ans)
        cout << "Cycle detected";
    else
        cout << "Cycle not detected";
    return 0;
}
```

// cycle detect using bfs

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 5;
bool vis[N];
vector<int> adj[N];
int parentArray[N];
bool ans;
void bfs(int s)
{
    queue<int> q;
    q.push(s);
    vis[s] = true;
    while (!q.empty())
    {
        int parent = q.front();
        // cout << parent << endl;
        q.pop();
        for (int child : adj[parent])
        {
            if (vis[child] == true && parentArray[parent] != child)
            {
                ans = true;
            }
            if (vis[child] == false)
            {
                vis[child] = true;
                parentArray[child] = parent;
                q.push(child);
            }
        }
    }
}
int main()
{
    int n, e;
    cin >> n >> e;
    while (e--)
    {
        int a, b;
        cin >> a >> b;
        adj[a].push_back(b);
        adj[b].push_back(a);
    }
    memset(vis, false, sizeof(vis));
    memset(parentArray, -1, sizeof(parentArray));
    ans = false;
    for (int i = 0; i < n; i++)
    {
        if (!vis[i])
        {
            bfs(i);
        }
    }
    if (ans)
```

```
        p.insert(x);
    }

    for (int i = 1;i <= m;i++) {
        int x;
        cin >> x;
        cout << p.order_of_key(x + 1) << " ";
    }
    cout << '\n';
    return 0;
}
```

// pbds - 2 // sliding window median

```
#include <bits/stdc++.h>
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
using namespace std;
template <typename T> using pbds = tree<T, null_type,
less<T>, rb_tree_tag, tree_order_statistics_node_update>;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(NULL);

    int n, k;
    cin >> n >> k;
    vector<int> a(n);
    for (int i = 0;i < n;i++) {
        cin >> a[i];
    }

    int l = 0, r = 0;
    pbds<pair<int, int>> p;
    while (r < n) {
        p.insert({ a[r],r });
        if (r - l + 1 == k) {
            int pos = k / 2;
            if (k % 2 == 0) {
                pos--;
            }
            auto it = p.find_by_order(pos);
            cout << it->first << " ";
            p.erase({ a[l],l });
            l++;
        }
        r++;
    }
    cout << '\n';
    return 0;
}
```

// fixed sliding window

```
class Solution {
public:
    long maximumSumSubarray(int k, vector<int>& a, int n) {
        int l = 0, r = 0;
        long long sum = 0, ans = 0;
        while (r < n) {
            sum += a[r];
            if ((r - l + 1) == k) {
                ans = max(ans, sum);
                sum -= a[l];
                l++;
                r++;
            }
            else {
```

```
    {
        cout << "Cycle found";
    }
    else
    {
        cout << "Cycle not found";
    }
    return 0;
}
```

// cycle detect using dfs

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 5;
bool vis[N];
vector<int> adj[N];
int parentArray[N];
bool ans;
void bfs(int s)
{
    queue<int> q;
    q.push(s);
    vis[s] = true;
    while (!q.empty())
    {
        int parent = q.front();
        // cout << parent << endl;
        q.pop();
        for (int child : adj[parent])
        {
            if (vis[child] == true && parentArray[parent] != child)
            {
                ans = true;
            }
            if (vis[child] == false)
            {
                vis[child] = true;
                parentArray[child] = parent;
                q.push(child);
            }
        }
    }
}
int main()
{
    int n, e;
    cin >> n >> e;
    while (e--)
    {
        int a, b;
        cin >> a >> b;
        adj[a].push_back(b);
        adj[b].push_back(a);
    }
    memset(vis, false, sizeof(vis));
    memset(parentArray, -1, sizeof(parentArray));
    ans = false;
    for (int i = 0; i < n; i++)
    {
        if (!vis[i])
        {
            bfs(i);
        }
    }
    if (ans)
    {
        cout << "Cycle found";
    }
```

```
            r++;
        }
    }
    return ans;
    }
};
```

// variable sliding window

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

int main() {
    ios::sync_with_stdio(false);
    cin.tie(NULL);

    int n;
    long long s;
    cin >> n >> s;
    vector<int> a(n);
    for (int i = 0;i < n;i++) {
        cin >> a[i];
    }

    long long l = 0, r = 0, ans = 0, sum = 0;

    while (r < n) {
        sum += a[r];
        if (sum <= s) {
            ans += (r - l + 1);
        }
        else {
            while (sum > s && l < r) {
                sum -= a[l];
                l++;
            }
            if (sum <= s) {
                ans += (r - l + 1);
            }
        }
        r++;
    }

    cout << ans << '\n';

    return 0;
}
```

// bfs traversal

```
#include <bits/stdc++.h>
using namespace std;
vector<int> v[1005];
bool vis[1005];
void bfs(int src)
{
    queue<int> q;
    q.push(src);
    vis[src] = true;
    while (!q.empty())
    {
        int par = q.front();
        q.pop();
        cout << par << endl;
        for (int child : v[par])
        {
            if (vis[child] == false)
            {
                q.push(child);
```

```
    else
    {
        cout << "Cycle not found";
    }
    return 0;
}
```

//dikstra

```
#include <bits/stdc++.h>
using namespace std;
const int N = 100;
vector<pair<int, int>> v[N];
int dis[N];
class cmp
{
public:
    bool operator()(pair<int, int> a, pair<int, int> b)
    {
        return a.second > b.second;
    }
};
void dijkstra(int src)
{
    priority_queue<pair<int, int>, vector<pair<int, int>>, cmp>
pq;
    pq.push({src, 0});
    dis[src] = 0;
    while (!pq.empty())
    {
        pair<int, int> parent = pq.top();
        pq.pop();
        int node = parent.first;
        int cost = parent.second;
        for (pair<int, int> child : v[node])
        {
            int childNode = child.first;
            int childCost = child.second;
            if (cost + childCost < dis[childNode])
            {
                // path relax
                dis[childNode] = cost + childCost;
                pq.push({childNode, dis[childNode]});
            }
        }
    }
}
int main()
{
    int n, e;
    cin >> n >> e;
    while (e--)
    {
        int a, b, c;
        cin >> a >> b >> c;
        v[a].push_back({b, c});
        v[b].push_back({a, c});
    }
    for (int i = 0; i < n; i++)
    {
        dis[i] = INT_MAX;
    }
    dijkstra(0);
    for (int i = 0; i < n; i++)
    {
        cout << i << "-> " << dis[i] << endl;
    }
    return 0;
}
```

```
                vis[child] = true;
            }
        }
    }
}
int main()
{
    int n, e;
    cin >> n >> e;
    while (e--)
    {
        int a, b;
        cin >> a >> b;
        v[a].push_back(b);
        v[b].push_back(a);
    }
    int src;
    cin >> src;
    memset(vis, false, sizeof(vis));
    bfs(src);
    return 0;
}
```

// bfs with level

```
#include <bits/stdc++.h>
using namespace std;
vector<int> v[1005];
bool vis[1005];
int level[1005];
void bfs(int src)
{
    queue<int> q;
    q.push(src);
    vis[src] = true;
    level[src] = 0;
    while (!q.empty())
    {
        int par = q.front();
        q.pop();
        for (int child : v[par])
        {
            if (vis[child] == false)
            {
                q.push(child);
                vis[child] = true;
                level[child] = level[par] + 1;
            }
        }
    }
}
int main()
{
    int n, e;
    cin >> n >> e;
    while (e--)
    {
        int a, b;
        cin >> a >> b;
        v[a].push_back(b);
        v[b].push_back(a);
    }
    int src;
    cin >> src;
    memset(vis, false, sizeof(vis));
    memset(level, -1, sizeof(level));
    bfs(src);
    for (int i = 0; i < n; i++)
    {
```

// belman ford

```
#include <bits/stdc++.h>
using namespace std;
class Edge
{
public:
    int u, v, c;
    Edge(int u, int v, int c)
    {
        this->u = u;
        this->v = v;
        this->c = c;
    }
};
const int N = 1e5 + 5;
int dis[N];
int main()
{
    int n, e;
    cin >> n >> e;
    vector<Edge> EdgeList;
    while (e--)
    {
        int u, v, c;
        cin >> u >> v >> c;
        EdgeList.push_back(Edge(u, v, c));
    }
    for (int i = 0; i < n; i++)
    {
        dis[i] = INT_MAX;
    }
    dis[0] = 0;
    for (int i = 1; i <= n - 1; i++)
    {
        for (Edge ed : EdgeList)
        {
            int u, v, c;
            u = ed.u;
            v = ed.v;
            c = ed.c;
            if (dis[u] < INT_MAX && dis[u] + c < dis[v])
            {
                dis[v] = dis[u] + c;
            }
        }
    }
    for (int i = 0; i < n; i++)
        cout << i << " -> " << dis[i] << endl;
    return 0;
}
```

//detect negative cycle

```
#include <bits/stdc++.h>
using namespace std;
class Edge
{
public:
    int u, v, c;
    Edge(int u, int v, int c)
    {
        this->u = u;
        this->v = v;
        this->c = c;
    }
};
const int N = 1e5 + 5;
```

```
        cout << i << " " << level[i] << endl;
    }
    return 0;
}
```

// bfs with level pair

```
#include <bits/stdc++.h>
using namespace std;
vector<int> v[1005];
bool vis[1005];
void bfs(int src, int des)
{
    queue<pair<int, int>> q;
    q.push({src, 0});
    vis[src] = true;
    bool paisi = false;
    while (!q.empty())
    {
        pair<int, int> p = q.front();
        q.pop();
        int par = p.first;
        int level = p.second;
        // cout << par << endl;
        if (par == des)
        {
            cout << level << endl;
            paisi = true;
        }
        for (int child : v[par])
        {
            if (vis[child] == false)
            {
                q.push({child, level + 1});
                vis[child] = true;
            }
        }
    }
    if (paisi == false)
    {
        cout << -1 << endl;
    }
}
int main()
{
    int n, e;
    cin >> n >> e;
    while (e--)
    {
        int a, b;
        cin >> a >> b;
        v[a].push_back(b);
        v[b].push_back(a);
    }
    int src;
    cin >> src;
    memset(vis, false, sizeof(vis));
    bfs(src, 9);
    return 0;
}
```

…………………………………………………………………..

```
int dis[N];
int main()
{
    int n, e;
    cin >> n >> e;
    vector<Edge> EdgeList;
    while (e--)
    {
        int u, v, c;
        cin >> u >> v >> c;
        EdgeList.push_back(Edge(u, v, c));
    }
    for (int i = 0; i < n; i++)
    {
        dis[i] = INT_MAX;
    }
    dis[0] = 0;
    for (int i = 1; i <= n - 1; i++)
    {
        for (Edge ed : EdgeList)
        {
            int u, v, c;
            u = ed.u;
            v = ed.v;
            c = ed.c;
            if (dis[u] < INT_MAX && dis[u] + c < dis[v])
            {
                dis[v] = dis[u] + c;
            }
        }
    }
    for (int i = 0; i < n; i++)
        cout << i << " -> " << dis[i] << endl;
    return 0;
}
```

// MST with kruskal with DSU

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

const int N = 1e5+5;
int par[N];
int level[N];
class Edge
{
    public:
       int u,v,w;
       Edge(int x,int y,int z)
       {
         u=x;
         v=y;
         w=z;
       }
};
int dsu_find(int node)
{
    if(par[node]==-1) return node;
    int leader = dsu_find(par[node]);
    par[node]=leader;
    return leader;
}
void dsu_union_by_rank(int a,int b)// firt = c second = d
{
    int leaderA = dsu_find(a);//-1
    int leaderB = dsu_find(b);//c
    if(level[leaderA] < level[leaderB])
    {
        par[leaderA]=leaderB;
        level[leaderB]++;
    }
    else
    {
        par[leaderB]=leaderA;
        level[leaderA]++;
    }
}
bool cmp(Edge list1, Edge list2)
{
    return list1.w < list2.w;
}
int main()
{

  int n,e;
  cin >> n >> e;
  vector<Edge> list;
  while(e--)
  {
    int a,b,c;
    cin >> a >> b >> c;
    list.push_back(Edge(a,b,c));
  }
  sort(list.begin(),list.end(),cmp);
  memset(par,-1,sizeof(par));
  memset(level,0,sizeof(level));
  int totalcost = 0;
  for(Edge ed:list)
  {
    int ledA = dsu_find(ed.u);
    int ledB = dsu_find(ed.v);
    if(ledA == ledB) continue;
    else
    {
        dsu_union_by_rank(ed.u,ed.v);
        totalcost += ed.w;
        cout << ed.u << " " << ed.v << " " << ed.w << endl;

    }
  }
  // cout << totalcost << endl;

}



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

const int N = 1e5;
vector<pair<int,int>> mat[N];
bool vis[N];

class cmp
{
    public:
           bool operator()(pair<int,int> a,pair<int,int> b)
           {
               return a.second > b.second;
           }
};
int  prims(int src)
{
   priority_queue<pair<int,int>,vector<pair<int,int>>,cmp> pq;
   pq.push({src,0});
   int totalcost = 0;
   while(!pq.empty())
   {
       pair<int,int> papa = pq.top();
       int pnode = papa.first;
       int pcost = papa.second;
       if(vis[pnode]==false)
       {
        totalcost += pcost;
       // cout << pnode << " "<< pcost << endl;
       }
       pq.pop();
       vis[pnode]=true;
       for(pair<int,int> child:mat[pnode])
       {
           int cnode = child.first;
           int ccost = child.second;
           if(vis[cnode]==false)
           {
             pq.push({cnode,ccost});
           }
       }

   }
   return totalcost;
}

int main()
{
    int n,e;
    cin >> n >> e;
    while(e--)
    {
        int a,b,c;
        cin >> a >> b >> c;
        mat[a].push_back({b,c});
        mat[b].push_back({a,c});
    }
    memset(vis,false,sizeof(vis));
    int totalcost = prims(1);
    // cout << totalcost << endl;

}
```