```
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 II 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 // path printing #include <bits/stdc++.h> #include <bits/stdc++.h> using namespace std; using namespace std; vector<int> v[1005]; int check\_kth\_bit\_on\_or\_off(int x, int k) { bool vis[1005]; return (x >> k) & 1; int level[1005]; int parent[1005]; void bfs(int src) void print\_on\_and\_off\_bits(int x) { for (int $k = 0; k \le 31; k++$ ) { queue<int> q; if (check\_kth\_bit\_on\_or\_off(x, k)) { cout << 1 << " "; q.push(src); vis[src] = true; level[src] = 0;

<sup>&</sup>lt;sup>1</sup> PSTU\_Logic\_Builder

```
while (!q.empty())
     else {
        cout << 0 << " ";
                                                                              int par = q.front();
                                                                              q.pop();
  cout << '\n';
                                                                              for (int child : v[par])
                                                                                if (vis[child] == false)
int turn_on_kth_bit(int x, int k) {
                                                                                   q.push(child);
  return (x | (1 << k));
                                                                                   vis[child] = true;
                                                                                   level[child] = level[par] + 1;
int turn_off_kth_bit(int x, int k) {
                                                                                   parent[child] = par;
  return (x & (\sim(1 << k)));
                                                                             }
                                                                          }
int toggle_kth_bit(int x, int k) {
  return (x ^ (1 << k));
                                                                        int main()
                                                                           int n, e;
int main() {
                                                                           cin >> n >> e;
  ios::sync_with_stdio(false);
                                                                           while (e--)
  cin.tie(NULL);
                                                                              int a, b;
  // cout << check_kth_bit_on_or_off(44, 3) << '\n';
                                                                             cin >> a >> b;
  // cout << check_kth_bit_on_or_off(44, 4) << '\n';
                                                                              v[a].push_back(b);
  // print_on_and_off_bits(44);
                                                                              v[b].push_back(a);
  // cout << turn on kth bit(44, 4);
  // cout << turn_off_kth_bit(44, 3);
                                                                           int src, des;
  cout << toggle_kth_bit(44, 3) << '\n';
                                                                           cin >> src >> des;
                                                                           memset(vis, false, sizeof(vis));
   return 0;
                                                                           memset(level, -1, sizeof(level));
                                                                           memset(parent, -1, sizeof(parent));
// bit masking
                                                                           bfs(src);
                                                                           int x = des;
#include <bits/stdc++.h>
                                                                           vector<int> path;
                                                                           while (x != -1)
using namespace std;
int main() {
                                                                              path.push_back(x);
   ios::sync_with_stdio(false);
                                                                              x = parent[x];
  cin.tie(NULL);
                                                                           reverse(path.begin(), path.end());
  int n;
                                                                           for (int val : path)
  cin >> n;
                                                                             cout << val << " ";
   for (int i = 0; i < (1 << n); i++) {
     for (int k = 0; k < n; k++) {
                                                                           return 0;
        if ((i >> k) & 1) {
          cout << 1 << " ":
                                                                        // bfs on 2D
        else {
           cout << 0 << " ";
                                                                        #include <bits/stdc++.h>
        }
                                                                        using namespace std;
     cout << '\n';
                                                                        bool vis[20][20];
                                                                        int dis[20][20];
                                                                        vector<pair<int, int>> d = \{\{0, 1\}, \{0, -1\}, \{-1, 0\}, \{1, 0\}\};
                                                                        int n, m;
  return 0;
                                                                        char a[20][20];
                                                                        bool valid(int i, int j)
                                                                           if (i < 0 || i >= n || j < 0 || j >= m)
// binary search
                                                                             return false;
                                                                           return true;
#include <bits/stdc++.h>
using namespace std;
                                                                        void bfs(int si, int sj)
int main() {
   ios::sync_with_stdio(false);
                                                                           queue<pair<int, int>> q;
   cin.tie(NULL);
                                                                           q.push({si, sj});
```

```
vis[si][sj] = true;
                                                                            dis[si][sj] = 0;
  int n, key;
  cin >> n >> key;
                                                                            while (!q.empty())
  vector<int> a(n);
  for (int i = 0; i < n; i++) {
                                                                              pair<int, int> par = q.front();
     cin >> a[i];
                                                                              int a = par.first, b = par.second;
                                                                              q.pop();
                                                                              for (int i = 0; i < 4; i++)
  int I = 0, r = n - 1, mid, idx = -1;
  while (I \leq r) {
                                                                                 int ci = a + d[i].first;
                                                                                 int cj = b + d[i].second;
     mid = (I + r) / 2;
     if (key == a[mid]) {
                                                                                 if (valid(ci, cj) == true && vis[ci][cj] == false)
        idx = mid;
        break;
                                                                                    q.push({ci, cj});
                                                                                    vis[ci][cj] = true;
     else if (key < a[mid]) {
                                                                                    dis[ci][cj] = dis[a][b] + 1;
       r = mid - 1;
                                                                              }
     else {
                                                                           }
       I = mid + 1;
                                                                         int main()
  }
                                                                            cin >> n >> m;
                                                                            for (int i = 0; i < n; i++)
  if (idx == -1) {
     cout << "Element not found" << '\n';
                                                                              for (int j = 0; j < m; j++)
  else {
     cout << "Element found at index " << idx << '\n';
                                                                                 cin >> a[i][j];
  return 0;
                                                                           int si, sj;
                                                                           cin >> si >> sj;
                                                                            memset(vis, false, sizeof(vis));
// binary search // Maximum median
                                                                           memset(dis, -1, sizeof(dis));
                                                                           bfs(si, sj);
#include <bits/stdc++.h>
                                                                            cout << dis[2][3];
using namespace std;
                                                                            return 0;
int main() {
  ios::sync_with_stdio(false);
  cin.tie(NULL);
                                                                         // component
  int n, k;
                                                                         #include <bits/stdc++.h>
  cin >> n >> k;
                                                                         using namespace std;
  vector<int> a(n);
                                                                         const int N = 1e5 + 5;
  for (int i = 0; i < n; i++) {
                                                                         vector<int> v[N];
                                                                         bool vis[N];
     cin >> a[i];
                                                                         void dfs(int src)
  sort(a.begin(), a.end());
                                                                         {
                                                                            cout << src << endl;
                                                                           vis[src] = true;
  auto ok = [&](long long mid) {
     long long cnt = 0;
                                                                            for (int child : v[src])
     for (int i = (n / 2); i < n; i++) {
                                                                              if (vis[child] == false)
        cnt += (a[i] < mid ? (mid - a[i]) : 0);
                                                                                 dfs(child);
     return cnt <= k;
                                                                         int main()
  long long I = 1, r = 2e9, mid, ans = 0;
                                                                           int n, e;
  while (I \leq r) {
     mid = I + (r - I) / 2;
                                                                            cin >> n >> e;
     if (ok(mid)) {
                                                                            while (e--)
        ans = mid;
        I = mid + 1;
                                                                              int a, b;
                                                                              cin >> a >> b;
                                                                              v[a].push_back(b);
     else {
        r = mid - 1;
                                                                               v[b].push_back(a);
```

```
memset(vis, false, sizeof(vis));
  cout << ans << '\n';
                                                                            int c = 0;
                                                                            for (int i = 0; i < n; i++)
  return 0;
                                                                               if (vis[i] == false)
                                                                                  dfs(i);
// difference array
                                                                               }
#include <bits/stdc++.h>
using namespace std;
                                                                            cout << "component - " << c << endl;
                                                                            return 0;
int main() {
  ios::sync_with_stdio(false);
  cin.tie(NULL);
                                                                         // dfs
  int n, q;
                                                                         #include <bits/stdc++.h>
  cin >> n >> q;
                                                                         using namespace std;
  vector<int> a(n + 1);
  for (int i = 1; i \le n; i++) {
                                                                         const int N = 1e5 + 5;
     cin >> a[i];
                                                                         vector<int> v[N];
                                                                         bool vis[N];
  vector<int> d(n + 2);
                                                                         void dfs(int src)
  for (int i = 1; i \le q; i++) {
                                                                         {
                                                                            cout << src << endl;
     int I, r, x;
     cin >> I >> r >> x;
                                                                            vis[src] = true;
     d[l] += x;
                                                                            for (int child : v[src])
     d[r + 1] = x;
  }
                                                                               if (vis[child] == false)
                                                                                  dfs(child);
  // for (int i = 0;i \le n + 1;i++) {
  // cout << d[i] << " ";
  //}
                                                                         int main()
  // cout << '\n';
                                                                         {
  for (int i = 1; i \le n; i++) {
                                                                            int n, e;
     d[i] = d[i - 1] + d[i];
                                                                            cin >> n >> e;
                                                                            while (e--)
  // for (int i = 0; i <= n + 1; i++) {
  // cout << d[i] << " ";
                                                                               int a, b;
  //}
                                                                               cin >> a >> b;
  // cout << '\n';
                                                                               v[a].push_back(b);
  for (int i = 1; i \le n; i++) {
                                                                               v[b].push_back(a);
     cout << a[i] + d[i] << " ";
                                                                            memset(vis, false, sizeof(vis));
  cout << '\n';
                                                                            dfs(0);
                                                                            return 0;
  return 0;
//prime factorization
                                                                         // dfs on 2d
#include <bits/stdc++.h>
using namespace std;
                                                                         #include <bits/stdc++.h>
                                                                         using namespace std;
int main() {
                                                                         char a[20][20];
  ios::sync_with_stdio(false);
                                                                         bool vis[20][20];
  cin.tie(NULL);
                                                                         vector<pair<int, int>> d = \{\{0, 1\}, \{0, -1\}, \{-1, 0\}, \{1, 0\}\};
                                                                         int n, m;
                                                                         bool valid(int i, int j)
  int n;
  cin >> n;
                                                                            if (i < 0 || i >= n || j < 0 || j >= m)
  map<int, int> cnt;
                                                                              return false;
  for (int i = 2; i * i <= n; i++) {
                                                                            return true;
     if (n \% i == 0) {
        while (n \% i == 0) {
                                                                         void dfs(int si, int sj)
           cnt[i]++;
           n /= i;
                                                                            cout << si << " " << si << endl;
       }
                                                                            vis[si][sj] = true;
     }
                                                                            for (int i = 0; i < 4; i++)
```

```
int ci = si + d[i].first;
  if (n > 1) {
                                                                                     int cj = sj + d[i].second;
     cnt[n]++;
                                                                                     if (valid(ci, cj) == true && vis[ci][cj] == false)
                                                                                       dfs(ci, cj);
  for (auto [x, y] : cnt) {    cout << x << " " << y << '\n';
                                                                                 }
  return 0;
                                                                              int main()
                                                                                  cin >> n >> m;
                                                                                  for (int i = 0; i < n; i++)
//prime factiorization using sieve
                                                                                     for (int j = 0; j < m; j++)
#include <bits/stdc++.h>
using namespace std;
                                                                                       cin >> a[i][j];
int main() {
  ios::sync_with_stdio(false);
cin.tie(NULL);
                                                                                  int si, sj;
                                                                                  cin >> si >> sj;
   int n = 20;
                                                                                  memset(vis, false, sizeof(vis));
  vector<br/>vector<br/>prime(n + 1, true);<br/>for (int i = 2;i * i <= n;i++) {
                                                                                  dfs(si, sj);
                                                                                 return 0;
     if (prime[i]) {
         for (int j = i + i; j <= n; j += i) {
           prime[j] = false;
                                                                               // cycle detect in directed graph
     }
  }
                                                                               #include <bits/stdc++.h>
                                                                               using namespace std;
                                                                               const int N = 1e5 + 5;
   vector<int> all_primes;
   for (int i = 2; i \le n; i++) {
                                                                               bool vis[N];
                                                                              bool pathVisit[N];
     if (prime[i]) {
         all_primes.push_back(i);
                                                                               vector<int> adj[N];
                                                                              bool ans;
                                                                               void dfs(int parent)
   // for (auto val : all_primes) {
                                                                                  vis[parent] = true;
                                                                                  pathVisit[parent] = true;
for (int child : adj[parent])
  // cout << val << " ";
  //}
  // cout << '\n';
                                                                                     if (pathVisit[child])
  map<int, int> cnt;
   int x, idx = 0;
                                                                                       ans = true;
  cin >> x;
   while (x > 1) {
                                                                                     if (!vis[child])
     if (x \% all\_primes[idx] == 0) {
         cnt[all_primes[idx]]++;
                                                                                        dfs(child);
         x /= all_primes[idx];
     else {
                                                                                 // kaj sesh
        idx++;
                                                                                  pathVisit[parent] = false;
                                                                              int main()
  for (auto [x, y] : cnt) {
    cout << x << " " << y << '\n';
                                                                                  int n, e;
                                                                                  cin >> n >> e;
                                                                                  while (e--)
  return 0;
                                                                                     int a, b;
                                                                                     cin >> a >> b;
                                                                                     adj[a].push_back(b);
                                                                                     // adj[b].push_back(a);
#include <bits/stdc++.h>
                                                                                  memset(vis, false, sizeof(vis));
                                                                                 memset(pathVisit, false, sizeof(pathVisit));
using namespace std;
                                                                                  ans = false;
                                                                                  for (int i = 0; i < n; i++)
int main() {
```

```
ios::sync_with_stdio(false);
  cin.tie(NULL);
                                                                            if (!vis[i])
                                                                               dfs(i);
  int n;
  cin >> n;
  vector<bool> prime(n + 1, true);
  for (int i = 2;i \dot{x} i <= \dot{n};i++) {
                                                                         if (ans)
     if (prime[i]) {
                                                                            cout << "Cycle detected";
        for (int j = i + i; j <= n; j += i) {
          prime[j] = false;
                                                                            cout << "Cycle not detected";
                                                                         return 0;
       }
    }
                                                                       // cycle detect using bfs
  for (int i = 2; i \le n; i++) {
                                                                       #include <bits/stdc++.h>
     if (prime[i]) {
       cout << i << " ";
                                                                      using namespace std;
                                                                       const int N = 1e5 + 5;
                                                                      bool vis[N];
  cout << '\n';
                                                                      vector<int> adj[N];
                                                                      int parentArray[N];
  return 0;
                                                                      bool ans;
                                                                       void bfs(int s)
// gcd and lcm
                                                                         queue<int> q;
                                                                         q.push(s);
#include <bits/stdc++.h>
                                                                         vis[s] = true;
                                                                         while (!q.empty())
using namespace std;
int gcd(int a, int b) {
                                                                            int parent = q.front();
  return gcd(a, b);
                                                                            // cout << parent << endl;
                                                                            q.pop();
                                                                            for (int child : adj[parent])
int lcm(int a, int b) {
                                                                               if (vis[child] == true && parentArray[parent] != child)
  // return ((a * b) / gcd(a, b));
  return ((a / gcd(a, b)) * b);
                                                                                 ans = true;
int main() {
                                                                               if (vis[child] == false)
  ios::sync_with_stdio(false);
  cin.tie(NULL);
                                                                                 vis[child] = true;
                                                                                 parentArray[child] = parent;
  int a, b;
                                                                                 q.push(child);
  cin >> a >> b;
  cout << gcd(a, b) << " " << lcm(a, b) << '\n';
  return 0;
                                                                         }
                                                                      int main()
// pbds - 1 //Queries about less or equal elements
                                                                         int n, e;
#include <bits/stdc++.h>
                                                                         cin >> n >> e;
                                                                         while (e--)
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace gnu pbds;
                                                                            int a, b;
                                                                            cin >> a >> b;
using namespace std;
template <typename T> using pbds = tree<T, null_type,
                                                                            adj[a].push_back(b);
less_equal<T>, rb_tree_tag,
                                                                            adj[b].push_back(a);
tree_order_statistics_node_update>;
                                                                         memset(vis, false, sizeof(vis));
int main() {
                                                                         memset(parentArray, -1, sizeof(parentArray));
  ios::sync_with_stdio(false);
                                                                         ans = false:
  cin.tie(NULL);
                                                                         for (int i = 0; i < n; i++)
  int n, m;
                                                                            if (!vis[i])
  cin >> n >> m;
  pbds<int> p;
                                                                               bfs(i);
  for (int i = 1;i <= n;i++) {
     int x;
     cin >> x;
                                                                         if (ans)
```

```
p.insert(x);
                                                                           cout << "Cycle found";
  for (int i = 1; i \le m; i++) {
                                                                        else
     cin >> x;
                                                                           cout << "Cycle not found";
     cout << p.order_of_key(x + 1) << " ";
                                                                        return 0;
  cout << '\n';
  return 0;
                                                                      // cycle detect using dfs
// pbds - 2 // sliding window median
                                                                     #include <bits/stdc++.h>
                                                                     using namespace std;
#include <bits/stdc++.h>
                                                                      const int N = 1e5 + 5;
#include<ext/pb_ds/assoc_container.hpp>
                                                                     bool vis[N];
#include<ext/pb_ds/tree_policy.hpp>
                                                                     vector<int> adj[N];
                                                                      int parentArray[N];
using namespace __gnu_pbds;
using namespace std;
                                                                      bool ans;
template <typename T> using pbds = tree<T, null_type,
                                                                     void bfs(int s)
less<T>, rb_tree_tag, tree_order_statistics_node_update>;
                                                                        queue<int> q;
int main() {
                                                                        q.push(s);
  ios::sync_with_stdio(false);
                                                                        vis[s] = true;
  cin.tie(NULL);
                                                                        while (!q.empty())
                                                                           int parent = q.front();
  int n, k;
                                                                           // cout << parent << endl;
  cin >> n >> k;
  vector<int> a(n);
                                                                           q.pop();
  for (int i = 0; i < n; i++) {
                                                                           for (int child : adj[parent])
     cin >> a[i];
                                                                              if (vis[child] == true && parentArray[parent] != child)
  int I = 0, r = 0;
                                                                                ans = true:
  pbds<pair<int, int>> p;
  while (r < n) {
                                                                              if (vis[child] == false)
     p.insert({ a[r],r });
     if (r - I + 1 == k) {
                                                                                vis[child] = true;
        int pos = k/2;
                                                                                parentArray[child] = parent;
        if (k \% 2 == 0) {
                                                                                q.push(child);
          pos--;
                                                                           }
        auto it = p.find_by_order(pos);
       cout << it->first << " ";
       p.erase({ a[l],l });
                                                                     int main()
       |++;
                                                                        int n, e;
                                                                        cin >> n >> e;
                                                                        while (e--)
  cout << '\n';
  return 0;
                                                                           int a, b;
                                                                           cin >> a >> b;
                                                                           adj[a].push_back(b);
// fixed sliding window
                                                                           adj[b].push_back(a);
                                                                        memset(vis, false, sizeof(vis));
class Solution {
public:
                                                                        memset(parentArray, -1, sizeof(parentArray));
  long maximumSumSubarray(int k, vector<int>& a, int n) {
                                                                        ans = false;
     int I = 0, r = 0;
                                                                        for (int i = 0; i < n; i++)
     long long sum = 0, ans = 0;
     while (r < n) {
                                                                           if (!vis[i])
        sum += a[r];
        if ((r-l+1) == k) {
                                                                              bfs(i);
          ans = max(ans, sum);
          sum -= a[l];
          |++;
                                                                        if (ans)
          r++;
                                                                           cout << "Cycle found";
        else {
```

```
else
           r++;
        }
                                                                            cout << "Cycle not found";
     return ans;
                                                                         return 0;
  }
};
// variable sliding window
                                                                       //dikstra
#include <bits/stdc++.h>
                                                                       #include <bits/stdc++.h>
using namespace std;
                                                                       using namespace std;
                                                                       const int N = 100;
                                                                       vector<pair<int, int>> v[N];
int main() {
   ios::sync_with_stdio(false);
                                                                       int dis[N];
  cin.tie(NULL);
                                                                       class cmp
                                                                       public:
   int n;
                                                                         bool operator()(pair<int, int> a, pair<int, int> b)
   long long s;
  cin >> n >> s;
   vector<int> a(n);
                                                                            return a.second > b.second;
  for (int i = 0; i < n; i++) {
     cin >> a[i];
                                                                       void dijkstra(int src)
   long long I = 0, r = 0, ans = 0, sum = 0;
                                                                         priority_queue<pair<int, int>, vector<pair<int, int>>, cmp>
                                                                          pq.push({src, 0});
  while (r < n) {
                                                                          dis[src] = 0;
     sum += a[r];
     if (sum <= s) {
                                                                          while (!pq.empty())
        ans += (r - l + 1);
                                                                            pair<int, int> parent = pq.top();
     else {
                                                                            pq.pop();
        while (sum > s && I < r) {
                                                                            int node = parent.first;
           sum -= a[l];
                                                                            int cost = parent.second;
           |++;
                                                                            for (pair<int, int> child : v[node])
                                                                               int childNode = child.first;
        if (sum <= s) {
           ans += (r - l + 1);
                                                                               int childCost = child.second;
                                                                               if (cost + childCost < dis[childNode])
                                                                                  // path relax
                                                                                  dis[childNode] = cost + childCost;
                                                                                  pq.push({childNode, dis[childNode]});
  cout << ans << '\n';
                                                                         }
  return 0;
                                                                       int main()
// bfs traversal
                                                                          int n. e:
#include <bits/stdc++.h>
                                                                          cin >> n >> e;
using namespace std;
                                                                         while (e--)
vector<int> v[1005];
bool vis[1005];
                                                                            int a, b, c;
void bfs(int src)
                                                                            cin >> a >> b >> c;
                                                                            v[a].push_back({b, c});
   queue<int> q;
                                                                            v[b].push_back({a, c});
   q.push(src);
  vis[src] = true;
                                                                          for (int i = 0; i < n; i++)
   while (!q.empty())
                                                                            dis[i] = INT_MAX;
     int par = q.front();
                                                                          dijkstra(0);
     q.pop();
     cout << par << endl;
                                                                          for (int i = 0; i < n; i++)
     for (int child : v[par])
                                                                            cout << i << "-> " << dis[i] << endl;
        if (vis[child] == false)
                                                                         return 0;
           q.push(child);
```

```
vis[child] = true;
                                                                       // belman ford
       }
                                                                       #include <bits/stdc++.h>
     }
  }
                                                                       using namespace std;
                                                                       class Edge
int main()
                                                                       public:
  int n, e;
                                                                          int u, v, c;
  cin >> n >> e;
                                                                          Edge(int u, int v, int c)
  while (e--)
                                                                            this->u = u;
                                                                            this->v = v;
     int a, b;
     cin >> a >> b;
                                                                            this->c = c;
     v[a].push_back(b);
     v[b].push_back(a);
                                                                       };
                                                                       const int N = 1e5 + 5;
                                                                       int dis[N];
  int src;
  cin >> src;
                                                                       int main()
  memset(vis, false, sizeof(vis));
  bfs(src);
                                                                          int n, e;
                                                                          cin >> n >> e;
  return 0;
                                                                          vector<Edge> EdgeList;
                                                                          while (e--)
// bfs with level
                                                                            int u, v, c;
#include <bits/stdc++.h>
                                                                            cin >> u >> v >> c;
using namespace std;
                                                                            EdgeList.push_back(Edge(u, v, c));
vector<int> v[1005];
bool vis[1005];
                                                                          for (int i = 0; i < n; i++)
int level[1005];
void bfs(int src)
                                                                            dis[i] = INT MAX;
  queue<int> q;
                                                                          dis[0] = 0;
                                                                          for (int i = 1; i \le n - 1; i++)
  q.push(src);
  vis[src] = true;
  level[src] = 0;
                                                                            for (Edge ed : EdgeList)
  while (!q.empty())
                                                                               int u, v, c;
     int par = q.front();
                                                                               u = ed.u;
     q.pop();
                                                                               v = ed.v;
     for (int child : v[par])
                                                                               c = ed.c;
                                                                               if (dis[u] < INT\_MAX && dis[u] + c < dis[v])
        if (vis[child] == false)
                                                                                  dis[v] = dis[u] + c;
          q.push(child);
          vis[child] = true;
                                                                            }
          level[child] = level[par] + 1;
                                                                          for (int i = 0; i < n; i++)
                                                                            cout << i << " -> " << dis[i] << endl;
  }
                                                                          return 0;
int main()
  int n, e;
                                                                       //detect negative cycle
  cin >> n >> e;
  while (e--)
                                                                       #include <bits/stdc++.h>
                                                                       using namespace std;
     int a, b;
                                                                       class Edge
     cin >> a >> b;
     v[a].push_back(b);
                                                                       public:
     v[b].push_back(a);
                                                                          int u, v, c;
                                                                          Edge(int u, int v, int c)
  int src;
  cin >> src;
                                                                            this->u = u;
  memset(vis, false, sizeof(vis));
                                                                            this->v = v;
  memset(level, -1, sizeof(level));
                                                                            this->c = c;
  bfs(src);
  for (int i = 0; i < n; i++)
                                                                       };
                                                                       const int N = 1e5 + 5;
```

```
cout << i << " " << level[i] << endl;
                                                                      int dis[N];
                                                                      int main()
  return 0;
                                                                         int n, e;
                                                                         cin >> n >> e;
                                                                         vector<Edge> EdgeList;
// bfs with level pair
                                                                         while (e--)
#include <bits/stdc++.h>
                                                                            int u, v, c;
using namespace std;
                                                                            cin >> u >> v >> c;
                                                                            EdgeList.push_back(Edge(u, v, c));
vector<int> v[1005];
bool vis[1005];
void bfs(int src, int des)
                                                                         for (int i = 0; i < n; i++)
  queue<pair<int, int>> q;
                                                                            dis[i] = INT_MAX;
  q.push({src, 0});
  vis[src] = true;
                                                                         dis[0] = 0;
  bool paisi = false;
                                                                         for (int i = 1; i \le n - 1; i++)
  while (!q.empty())
                                                                            for (Edge ed : EdgeList)
     pair<int, int> p = q.front();
     q.pop();
                                                                              int u, v, c;
     int par = p.first;
                                                                              u = ed.u;
     int level = p.second;
                                                                              v = ed.v;
     // cout << par << endl;
                                                                              c = ed.c;
     if (par == des)
                                                                              if (dis[u] < INT\_MAX && dis[u] + c < dis[v])
        cout << level << endl;
                                                                                 dis[v] = dis[u] + c;
        paisi = true;
                                                                           }
     for (int child : v[par])
                                                                         for (int i = 0; i < n; i++)
                                                                            cout << i << " -> " << dis[i] << endl;
        if (vis[child] == false)
                                                                         return 0:
          q.push({child, level + 1});
          vis[child] = true;
                                                                      // MST with kruskal with DSU
     }
  if (paisi == false)
                                                                      //kruskals_with_DSU
                                                                      #include<bits/stdc++.h>
     cout << -1 << endl;
                                                                      using namespace std;
                                                                      const int N = 1e5+5;
int main()
                                                                      int par[N];
                                                                      int level[N];
                                                                      class Edge
  int n, e;
  cin >> n >> e;
                                                                         public:
  while (e--)
                                                                           int u,v,w;
     int a, b;
                                                                           Edge(int x,int y,int z)
     cin >> a >> b;
     v[a].push_back(b);
                                                                            u=x:
     v[b].push_back(a);
                                                                            v=y;
                                                                            w=z;
  int src;
  cin >> src;
  memset(vis, false, sizeof(vis));
                                                                      int dsu find(int node)
  bfs(src, 9);
  return 0;
                                                                         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;
#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;
```