Maximum weight node

```
class Solution
{
  public:
  int maxWeightCell(int N, vector<int> Edge)
  {
      // code here
        int arr[N];
        int ans=INT_MIN,j=-1;
        for(int i=0;i<N;i++){
            arr[i]= 0;
        }
        for(int i=0; i<N; i++){
            if(Edge[i]!= -1){
                arr[Edge[i]] += i;
            }
        }
        for(int i=0;i<N;i++){
            if(ans<=arr[i]){
                ans = arr[i];
                j = i;
            }
        }
        if(ans!=INT_MIN)
            return j;
        return -1;
  }
};
```

Largest sum cycle

```
vector<vector<int>> g;
vector<int> vis, par, tmp;

class Solution
{
    public:
    long long dfs(int node, int p = -1) {
        vis[node] = 1;
        par[node] = p;
        tmp.push_back(node);
        for (auto ng : g[node]) {
```

```
        if (vis[ng] == 0) {
            long long z = dfs(ng, node);
            if (z != -1) {
                return z;
            }
        }
        else if (vis[ng] == 1) {
            long long sum = ng;
            while (node != ng) {
                sum += node;
                node = par[node];
            }
            if (node == ng) {
                return sum;
            }
            return -1;
        }
    }
    return -1;
}

long long largestSumCycle(int n, vector<int> Edge)
{
// code here
    long long ans = -1;
    vis = vector<int>(n);
    g = vector<vector<int>>(n);
    par = vector<int>(n);

    for (int i = 0; i < n; i++) {
        if (Edge[i] != -1) {
            g[i].push_back(Edge[i]);
        }
    }

    for (int i = 0; i < n; i++) {
        if (!vis[i]) {
            ans = max(ans, dfs(i));
            for (auto j : tmp) {
                vis[j] = 2;
            }
            tmp.clear();
        }
    }
```

```
        return ans;
    }
};
```

Longest sum cycle

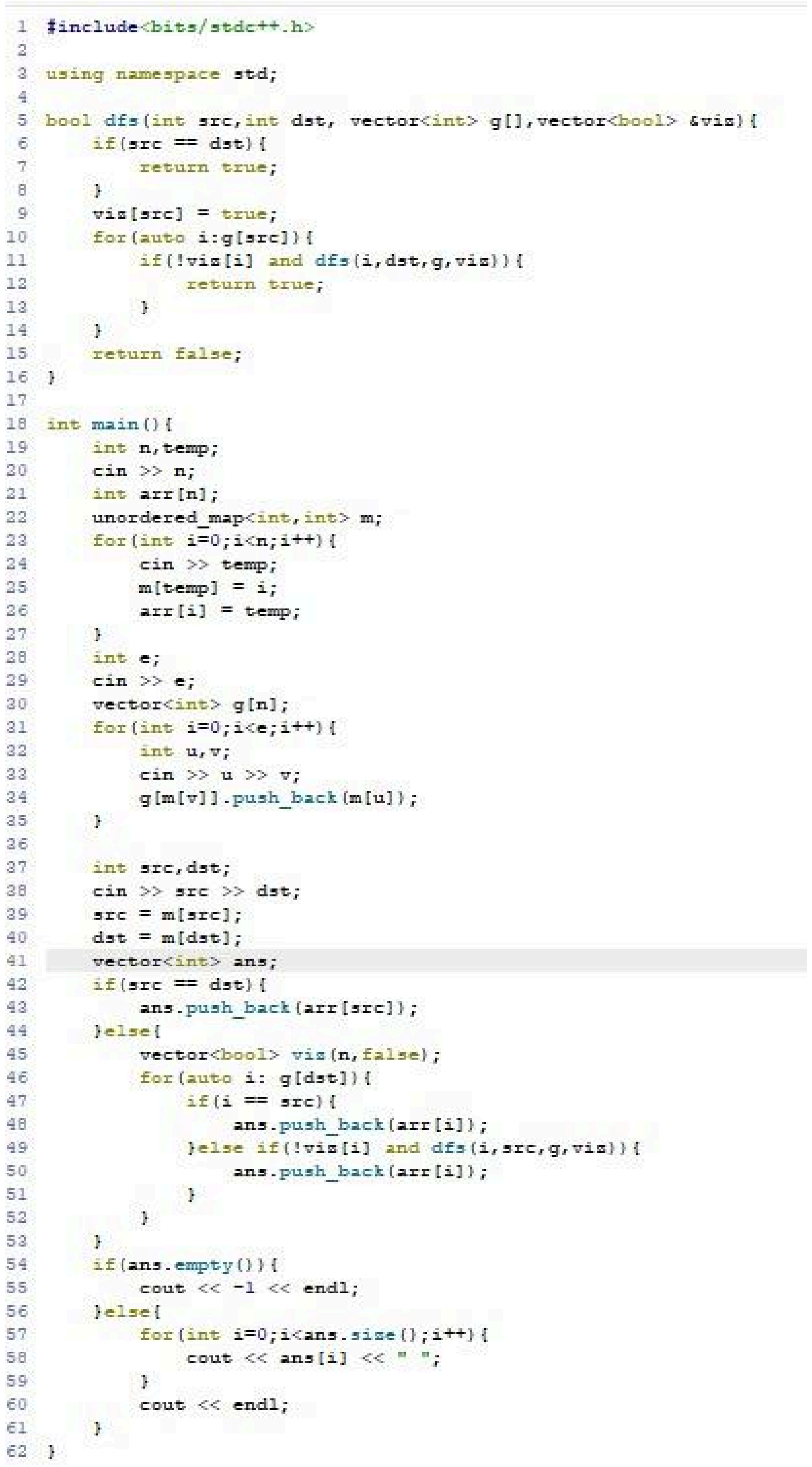
```
class Solution {
public:
    void dfs(int node, vector<int> &dist_node1, vector<bool> &visited, vector<int> &edges, int
distance, int &ans, vector<bool> &extra) {
        if (node != -1) {
            if (!visited[node]) {
                visited[node] = true;
                extra[node] = true;
                dist_node1[node] = distance;
                dfs(edges[node], dist_node1, visited, edges, distance + 1, ans, extra);
            }
            else if (extra[node]) {
                ans = max(ans, distance - dist_node1[node]);
            }
            extra[node] = false;
        }
    }

    int longestCycle(vector<int>& edges) {
        vector<int> dist_node(edges.size(), 0);
        vector<bool> visited(edges.size(), false);
        vector<bool> extra(edges.size(), false);
        int ans = -1;

        for (int i = 0; i < edges.size(); i++) {
            if (!visited[i]) {
                dfs(i, dist_node, visited, edges, 0, ans, extra);
            }
        }
        return ans;
    }
};
```

Nearest meeting cell

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

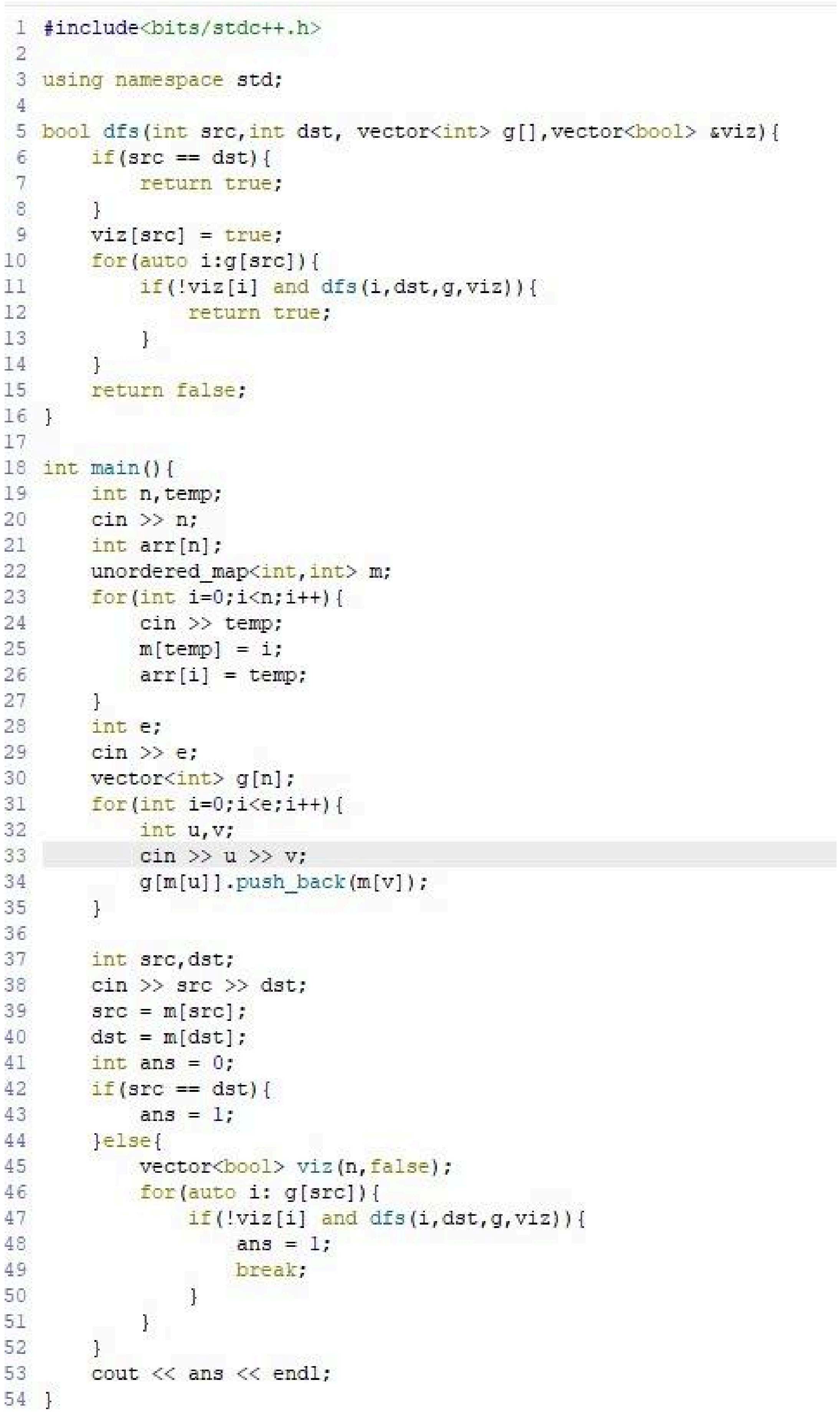
```
vector<int> shortPath(vector<int> adj[], int c1, int n){

vector<int> dist(n, INT_MAX);
queue<int> q;
q.push(c1);
dist[c1] = 0;

while(!q.empty()){
    int u = q.front();
    q.pop();

    for(auto &v: adj[u]){
        if(dist[v] > dist[u] + 1){
            dist[v] = dist[u] + 1;
            q.push(v);
        }
    }
}
return dist;
}

int main()
{
int n;
cin>>n;

vector<int> edges(n);
for(int i=0 ; i<n ; i++)
    cin >> edges[i];

int c1, c2;
cin>>c1>>c2;

vector<int> adj[n];
for(int i=0 ; i<n ; i++){
    if(edges[i] == -1) continue;
    adj[i].push_back(edges[i]);
}
vector<int> v1 = shortPath(adj, c1, n);
vector<int> v2 = shortPath(adj, c2, n);

int mn = INT_MAX, node = -1;
for(int i=0 ; i<n ; i++){
    if(v1[i] == INT_MAX || v2[i] == INT_MAX)
        continue;
    if(v1[i] + v2[i] < mn){
        mn = v1[i] + v2[i];
        node = i;
    }
}
cout << node << endl;
return 0;
}
```

The nagging react newbie



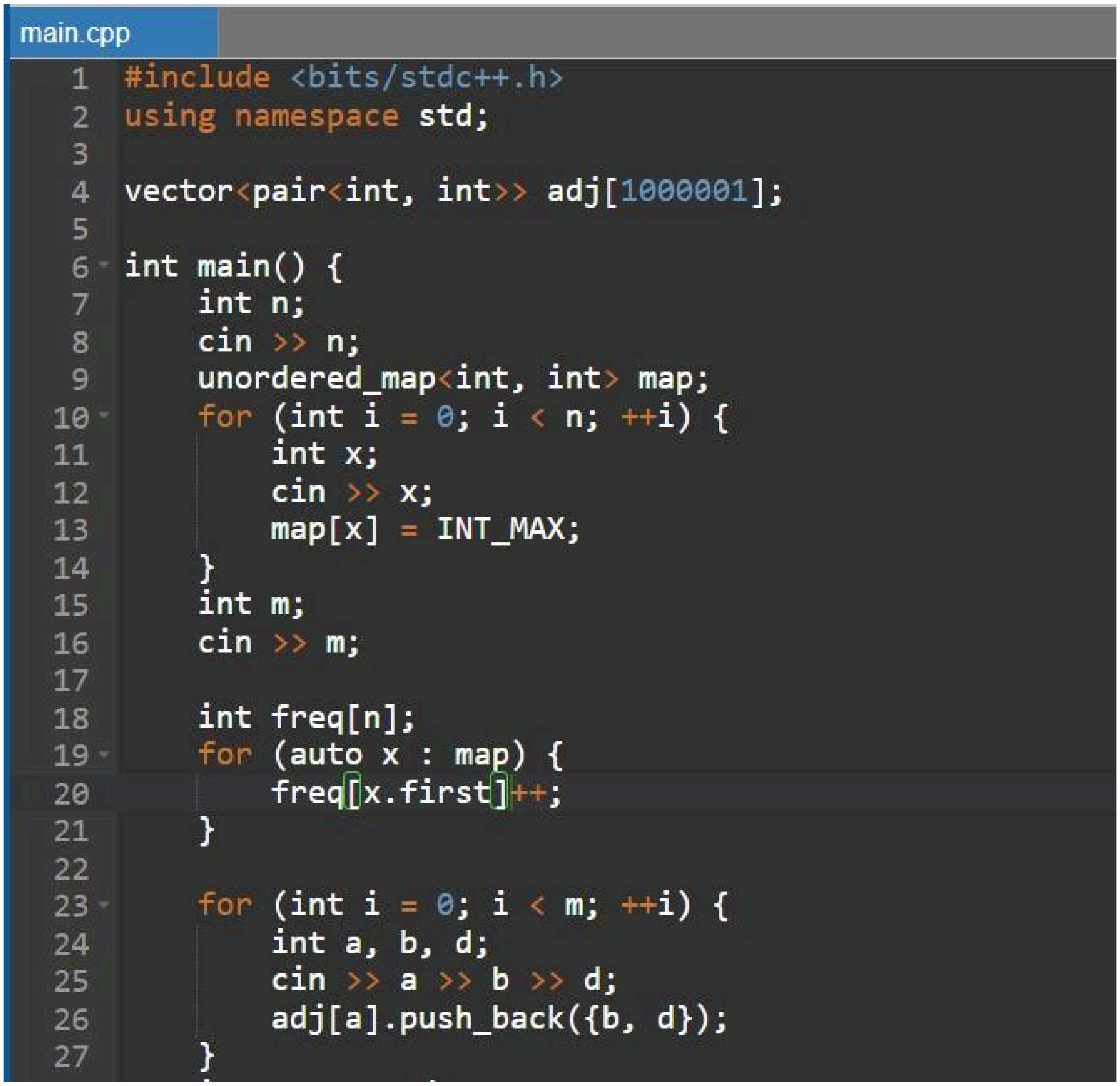
```cpp
#include<bits/stdc++.h>

using namespace std;

bool dfs(int src,int dst, vector<int> g[],vector<bool> &vis){
    if(src == dst){
        return true;
    }
    vis[src] = true;
    for(auto i:g[src]){
        if(!vis[i] and dfs(i,dst,g,vis)){
            return true;
        }
    }
    return false;
}

int main(){
    int n,temp;
    cin >> n;
    int arr[n];
    unordered_map<int,int> m;
    for(int i=0;i<n;i++){
        cin >> temp;
        m[temp] = i;
        arr[i] = temp;
    }
    int e;
    cin >> e;
    vector<int> g[n];
    for(int i=0;i<e;i++){
        int u,v;
        cin >> u >> v;
        g[m[v]].push_back(m[u]);
    }

    int src,dst;
    cin >> src >> dst;
    src = m[src];
    dst = m[dst];
    vector<int> ans;
    if(src == dst){
        ans.push_back(arr[src]);
    }else{
        vector<bool> vis(n,false);
        for(auto i: g[dst]){
            if(i == src){
                ans.push_back(arr[i]);
            }else if(!vis[i] and dfs(i,src,g,vis)){
                ans.push_back(arr[i]);
            }
        }
    }
    if(ans.empty()){
        cout << -1 << endl;
    }else{
        for(int i=0;i<ans.size();i++){
            cout << ans[i] << " ";
        }
        cout << endl;
    }
}
```

Find reachability

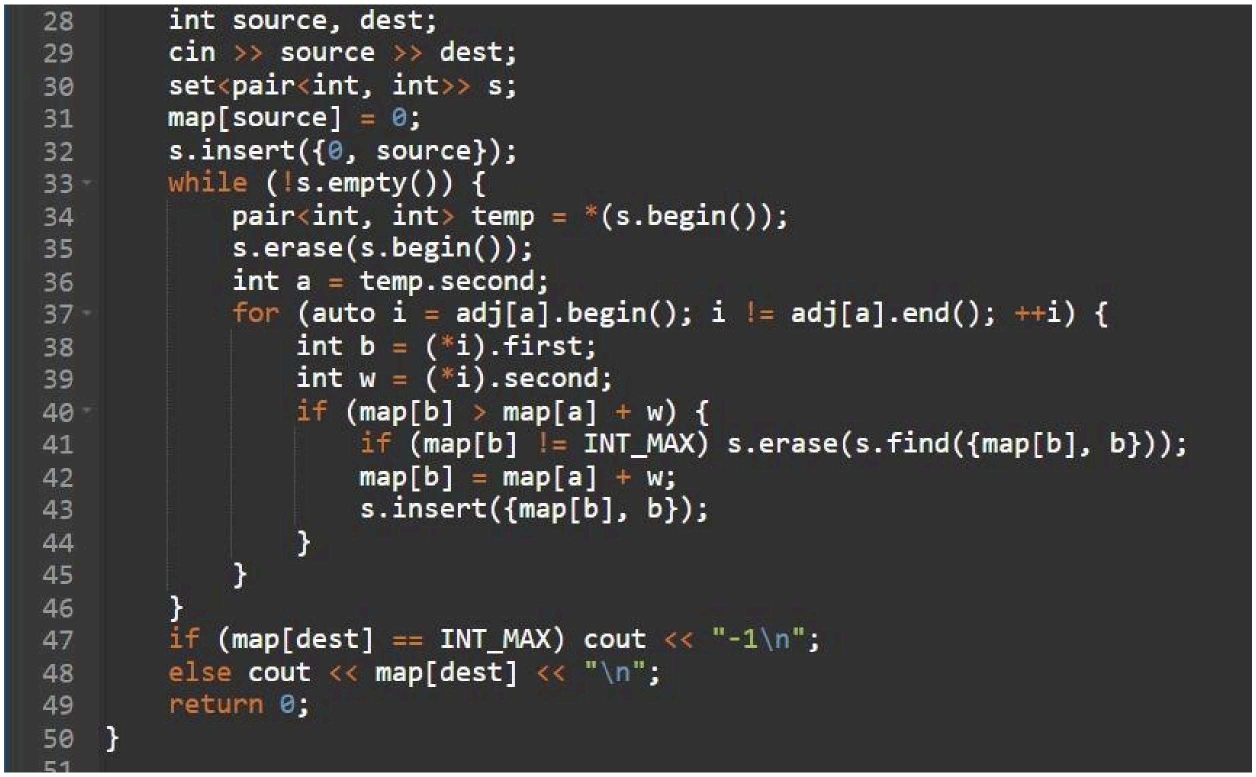


```cpp
#include<bits/stdc++.h>

using namespace std;

bool dfs(int src,int dst, vector<int> g[],vector<bool> &viz){
    if(src == dst){
        return true;
    }
    viz[src] = true;
    for(auto i:g[src]){
        if(!viz[i] and dfs(i,dst,g,viz)){
            return true;
        }
    }
    return false;
}

int main(){
    int n,temp;
    cin >> n;
    int arr[n];
    unordered_map<int,int> m;
    for(int i=0;i<n;i++){
        cin >> temp;
        m[temp] = i;
        arr[i] = temp;
    }
    int e;
    cin >> e;
    vector<int> g[n];
    for(int i=0;i<e;i++){
        int u,v;
        cin >> u >> v;
        g[m[u]].push_back(m[v]);
    }

    int src,dst;
    cin >> src >> dst;
    src = m[src];
    dst = m[dst];
    int ans = 0;
    if(src == dst){
        ans = 1;
    }else{
        vector<bool> viz(n,false);
        for(auto i: g[src]){
            if(!viz[i] and dfs(i,dst,g,viz)){
                ans = 1;
                break;
            }
        }
    }
    cout << ans << endl;
}
```

Learn JS



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

vector<pair<int, int>> adj[1000001];

int main() {
    int n;
    cin >> n;
    unordered_map<int, int> map;
    for (int i = 0; i < n; ++i) {
        int x;
        cin >> x;
        map[x] = INT_MAX;
    }
    int m;
    cin >> m;

    int freq[n];
    for (auto x : map) {
        freq[x.first]++;
    }

    for (int i = 0; i < m; ++i) {
        int a, b, d;
        cin >> a >> b >> d;
        adj[a].push_back({b, d});
    }
```



```
    int source, dest;
    cin >> source >> dest;
    set<pair<int, int>> s;
    map[source] = 0;
    s.insert({0, source});
    while (!s.empty()) {
        pair<int, int> temp = *(s.begin());
        s.erase(s.begin());
        int a = temp.second;
        for (auto i = adj[a].begin(); i != adj[a].end(); ++i) {
            int b = (*i).first;
            int w = (*i).second;
            if (map[b] > map[a] + w) {
                if (map[b] != INT_MAX) s.erase(s.find({map[b], b}));
                map[b] = map[a] + w;
                s.insert({map[b], b});
            }
        }
    }
    if (map[dest] == INT_MAX) cout << "-1\n";
    else cout << map[dest] << "\n";
    return 0;
}
```