October 1, 2025

DOUBT SESSION

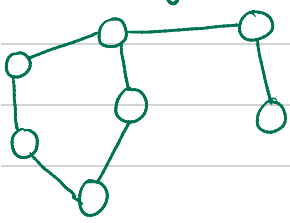
Ques 1. Edge Removals

https://maang.in/problems/Edge-Removals-41

You have given an undirected graph G with N nodes, indexed from 1 to N and M edges, indexed from 1 to M. There are two types of operations. **1 X**: Remove the edge numbered X. **2**: Print the number of connected components in the graph.

Example :

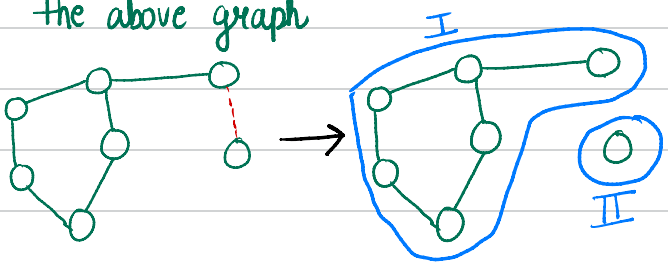
We are given a graph –



Here,

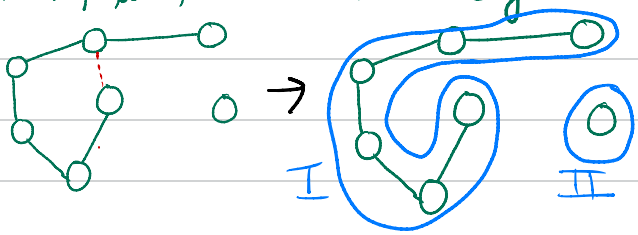
No. of components = 1

Let's say we remove an edge from the above graph



Number of components after removing the above edge = 2.

Now, we remove another edge



Number of components = 2

How do we represent an edge b/w two nodes a & b :

or,

```
g[a].push_back[b]
g[b].push_back[a]
```

Choices we have to remove edge :

1. • Erase from vector O(N)
   • Run normal BFS to calculate no. of components O(N)
   ⇒ O(N) for one edge.
2. Optimised :

In adjacency matrix, we have

```
g[a][b] = 1
g[b][a] = 1
```

For removal of an edge, we could simply replace 1 with 0 for that respective node :

```
g[a][b] = 0
g[b][a] = 0
```

This can be done in O(1).

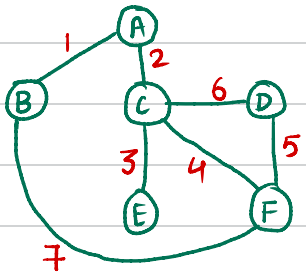
Then, Run Normal BFS to find #component.

⇒ O(N)

For M number of edges, our T.C will be O(N*M) This shall give us TLE for given constraints.

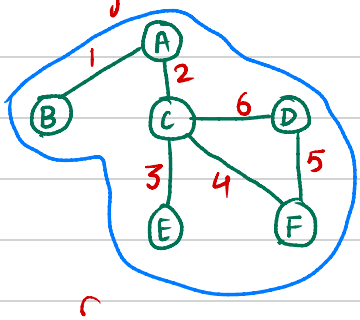
So, we cannot use the above approaches for our solution and have to optimise it to fit the constraints.

Example :



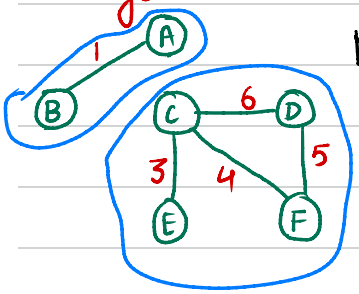
Now, we will be given the edges that needs to removed. Let's first do that & calc the #components.

→ Edge 7



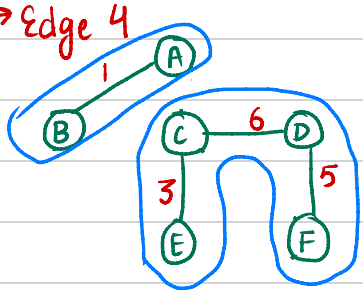
No. of component ⇒ 1

→ Edge 2



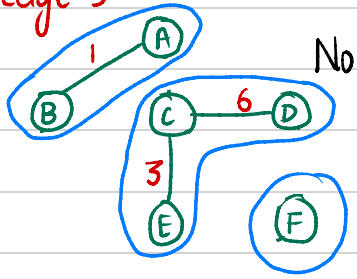
No. of component ⇒ 2

→ Edge 4



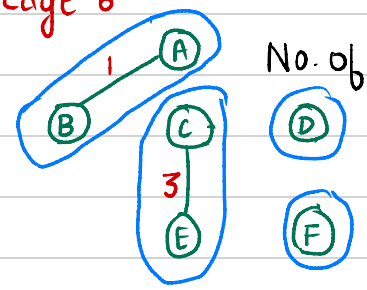
No. of component ⇒ 2

→ Edge 5



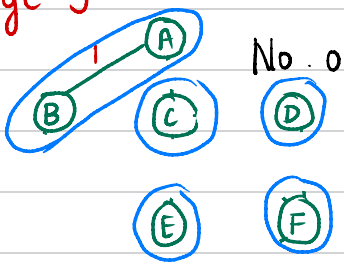
No. of component ⇒ 3

Edge 6



No. of components = 4

Edge 3



No. of components = 5

Edges Removed	7	2	4	5	6	3
No. of component	1	2	2	3	4	5

Now, let us reverse the whole process where we remove all the required edges for removal all at once and then add them to observe the effect on # component.

We will see that the no. of component will only change if the nodes that are to be connected a & b :

p[a] != p[b] (parent)
Merge a & b
comp --

The code of the following :

```cpp
int par[101000];
int ran[101000];
int com;
int find_par(int x)
{
    if (par[x] == x)
    {
        return x;
    }
    return par[x] = find_par(par[x]);
}
void merge(int x, int y)
{
    x = find_par(x), y = find_par(y);
    if (x != y)
    {
        if (ran[x] > ran[y])
        {
            swap(x, y);
        }
        ran[y] += ran[x];
        par[x] = y;
        com--;
    }
}
```

// DSU

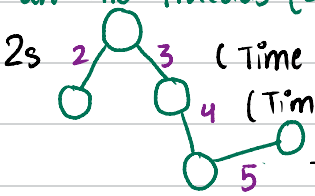
```cpp
int n, m, q;
cin >> n >> m >> q;
com = n;
f(i, 1, m + 1)
{
    par[i] = i, ran[i] = 1;
}
vector<pair<int, int>>e;
f0(i, m)
{
    int a, b;
    cin >> a >> b;
    e.push_back({a, b});
}
vector<pair<int, int>>query;
vector<int>v(m, 1);
while (q--)
{
    int a, b = -1;
    cin >> a;
    if (a == 1)
    {
        cin >> b;
        b--;
        v[b] = 0;
    }
    query.push_back({a, b});
}
f0(i, m)
{
    if (v[i])
    {
        merge(e[i].first, e[i].second);
    }
}
vi ans;
for (int i = sz(query) - 1; i >= 0; i--)
{
    if (query[i].second == -1)
    {
        ans.pb(com);
    }
    else
    {
        merge(e[query[i].second].first, e[query[i].second].second);
    }
}
reverse(all(ans));
for (auto i : ans)
{
    cout << i << endl;
}
```

Ques 2. Burn them all

https://maang.in/problems/Burn-them-All-618

You have given an undirected graph of N vertices and M edges. Edge weight d on edge between nodes u and v represents that u and v are connected by a thread of length d units. You set node A on to the fire. It takes the fire 1 unit of time to travel 1 unit of distance via threads. Let T be the minimum time in which all the threads will be burned out. Your task is to find $10T$. We can prove that $10T$ will always be an integer number. See the sample test cases for more explanation.

Example : Given a root node where the fire starts, min time in which all the threads (edges) will be burnt:



(Time to burn = 3s)

(Time to burn = 3+4 = 7s)

Time to burn = 7+5 = 12s

Total = 12s.

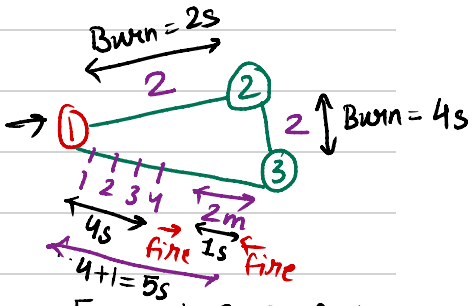
Test Case :

1 2 2

2 3 2

1 3 6

1 ← Node where fire begins



From 1→2→3, it took 4s to burn.

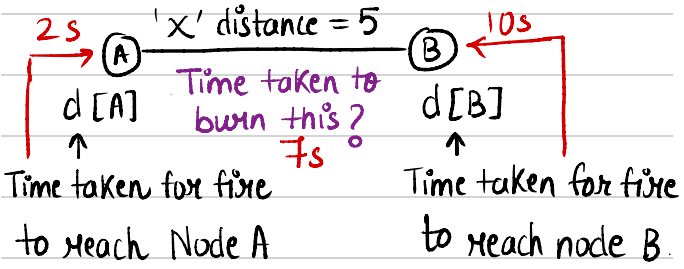
Simultaneously 1→3 is also burning and by the time 1→2→3 was burnt, distance = 4 was covered for 1→3. Now, the remaining distance is 6-4=2. After 4s, the fire approached from left & right BOTH. We have to cover 2m distance with 1m/s fire from left & 1m/s right.

AlgoZenith

Distance to cover = 2m
Speed = 2m/s
Time taken = $\frac{2}{2}$ = 1s

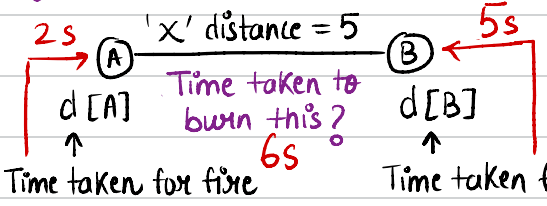
Total = 4+1 = 5s

Now, let us generalise our approach:



Distance of thread A→B = 5
Time taken for burning A→B : 7s
From A node, the fire will reach first as it is 2s (it will finish burning node y→A in 2s and reach to start burning A→B) meanwhile the fire that is suppose to start from Node B hasn't finished burning Node Z→B which takes 10s. Here the fire burns node y→A in 2s and cover the distance = 5m in 5s. Total=7s.
So the edge A→B would have burnt before the fire from B could even start.

```
if (min(d[A], d[B])+X <= max(d[A],d[B]))
{
  time = min(d[A], d[B])+X
}
```



In this case, the fire will start from A but will cover only distance = 3 as 3+2=5 which means that fire from the side B will also start as it has reached B in 5s.
Distance remaining = 5-3=2m.
Fire coming from both sides = 2m/s
Time taken = 1s.
Initially, we covered 3m in 3s due to fire coming from A
Then, we covered remaining 2m in 1s
Total = 2+3+1 = 6s.

```
if (min(d[A], d[B])+X <= max(d[A],d[B]))
{
  time = min(d[A], d[B])+X
}
else
{
time = min(d[A], d[B]) + [X - (abs(d[B]-d[A]))] / 2
}
```

(length covered; speed)

So, we have 2 expressions now that can be the possible output but we have to output 10T.

①

$$\left(\min(d[A], d[B]) + \left(x + \frac{abs(d[B]-d[A])}{2}\right)\right) \times 10$$

$$10 * \min(d[A], d[B]) + 5 * \left(x + \frac{abs(d[B]-d[A])}{2}\right)$$

② $(\min(d[A], d[B]) + x) * 10$

```
void solve()
{
    int n;
    cin >> n;
    int m;
    cin >> m;
    vector<vector<pair<int, int>>>g(n + 1);
    vector<pair<int, pair<int, int>>>e;
    f0(i, m)
    {
        int a, b, c;
        cin >> a >> b >> c;
        e.push_back({c, {a, b}});
        g[a].push_back({b, c});
        g[b].push_back({a, c});
    }
    int s;
    cin >> s;
    priority_queue<pair<int, int>>pq;
    pq.push({0, s});
    vector<int>d(n + 1, 1e18), vis(n + 1, 0);
    d[s] = 0;
    while (!pq.empty())
    {
        pair<int, int>p = pq.top();
        int wt = p.first, i = p.second;
        pq.pop();
        if (vis[i])continue;
        vis[i] = 1;
        for (auto j : g[i])
        {
            int node = j.first, w = j.second;
            if (d[node] > w + d[i])
            {
                d[node] = w + d[i];
                pq.push({ -d[node], node});
            }
        }
    }
    int mx = 0;
    for (auto i : e)
    {
        int diff = d[i.second.first] - d[i.second.second];
        int x = i.first, f;
        if (abs(diff) >= x)
        {
            f = (min(d[i.second.first], d[i.second.second]) + x) * 10;
        }
        else
        {
            f = ((d[i.second.first] + d[i.second.second] + x) * 5);
        }
        mx = max(mx, f);
    }
    cout << mx << endl;
}
signed main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(0);
// #ifndef ONLINE_JUDGE
//     freopen("input.txt", "r", stdin);
//     freopen("output.txt", "w", stdout);
// #endif // ONLINE_JUDGE
    int t = 1;
    // cin >> t;
    while (t--)
    {
        solve();
    }
    return 0;
}
```

Ques 3. One Piece

https://maang.in/problems/One-Piece-902

Monkey D. Luffy, on his journey of becoming the **"King of Pirates"** and to conquer the **"One Piece"**, wants to travel across the Grand Line. Grand Line is a mysterious sea, and is in the shape of a $N \times M$ grid S with every cell denoting the wind direction. The sign of $S[i][j]$ can be: 1 which means wind in the cell flows to the right (i.e. from $S[i][j]$ to $S[i][j+1]$), 2 which means wind in the cell flows to the left (i.e. from $S[i][j]$ to $S[i][j-1]$), 3 which means wind in the cell flows downwards (i.e. from $S[i][j]$ to $S[i+1][j]$), or 4 which means wind in the cell flows upwards (i.e. from $S[i][j]$ to $S[i-1][j]$). Notice that there could be some signs on the cells of the grid S that point outside the Grand Line sea grid.

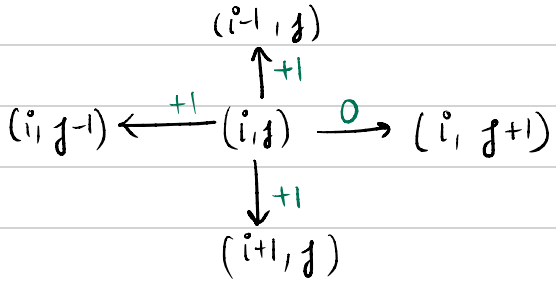
Merry, Luffy's ship, can only sail along the wind direction and cannot go outside the Grand Line sea grid S at any point. Luffy can modify the wind's direction on a cell with a $cost = 1$ (he can modify the sign on a cell one time only).

Find the minimum cost to make Luffy's voyage from the top-left corner of the Grand Line, i.e., $S[1][1]$, to its bottom-right corner, i.e., $S[N][M]$, possible.

Use of Dijkstra

Have to go from top-left to bottom right at min cost.

Takes +1 cost to move left, up & down.



Ques 4. Cycle Finding

https://cses.fi/problemset/task/1197

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.

Use of Bellman Ford to find -ve cycle

```
signed main() {
    int n, m;
    cin >> n >> m;
    vector<pair<int, pair<int, int>>>edges;
    for (int i = 0; i < m; i++)
    {
        int a, b, c;
        cin >> a >> b >> c;
        edges.push_back({ a,{b,c} });
    }
    if (m == 1)
    {
        if (edges[0].first == edges[0].second.first and edges[0].second.second < 0)
        {
            cout << "YES\n" << 1 << " " << 1 << endl;
        }
        else
        {
            cout << "NO\n";
        }
        return 0;
    }
    vector<int>par(n + 1, -1), dis(n + 1, 1e18);
    dis[1] = 0;
    for (int i = 0; i < n - 1; i++)
    {
        for (auto j : edges)
        {
            if (dis[j.second.first] > dis[j.first] + j.second.second)
            {
                dis[j.second.first] = dis[j.first] + j.second.second;
                par[j.second.first] = j.first;
            }
```

```
    }
    if (curr != -1)
    {
        cout << "YES\n";
        set<int>s;
        vector<int>v;
        while (curr!=-1)
        {
            v.push_back(curr);
            if (s.find(curr) != s.end())
            {
                break;
            }
            s.insert(curr);
            curr = par[curr];
        }
        vector<int>vis;
        vis.resize(n + 1, 0);
        for (auto i : v)vis[i]++;
        int flag = 0;
        vector<int>ans;
        for (auto i : v)
        {
            if (flag)
            {
                ans.push_back(i);
                continue;
            }
            if (vis[i] == 2)
            {
                ans.push_back(i);
                flag = 1;
                continue;
            }
        }
        reverse(ans.begin(), ans.end());
        for (auto i : ans)cout << i << " ";
        cout << endl;
    }
    else
    {
        cout << "NO\n";
    }
    return 0;
}
```

Ques 5. Gotham rises

https://maang.in/problems/Gotham-rises-866

Gotham is in ruins. After the destruction caused by the Joker's latest rampage, the city is in chaos. Bruce Wayne has returned as Batman and has neutralised most of the Joker's associates. However, the city needs to be rebuilt and most of the people need urgent medical attention. The city's prominent Gotham General hospital has already been blown up by the Joker. Hence instead of building one large hospital which may easily become the next target for the Joker, Batman asks the mayor to build small clinics throughout the city.

The city consists of N blocks. The mayor also has to ensure that every block has access to at least one clinic. The connectivity is established by repairing the existing roads.

The cost to build a new clinic is c and to repair a road is r.

Your task is to find the minimum cost such that for every block in the city either there is a clinic in the block or the block has a path via repaired roads to a block with a clinic.

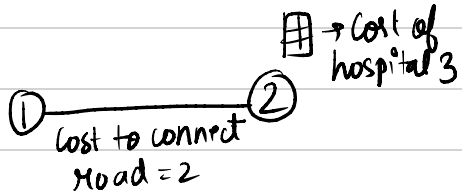
Example :

5 Blocks

4 roads

Cost to build hospital = 3

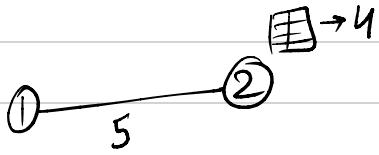
Cost to build road = 2



- 5 units

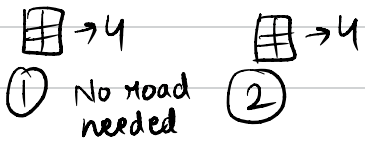
Cost of hospital = 4

Cost to build road = 5



→ Cost = 9

Or,



Cost = 8

if (cost of hospital <= cost of road)

{

ans = N * cost of hospital //No road

(N ↑ no. of Block)

Cost of hospital = 5

Cost to build road = 4