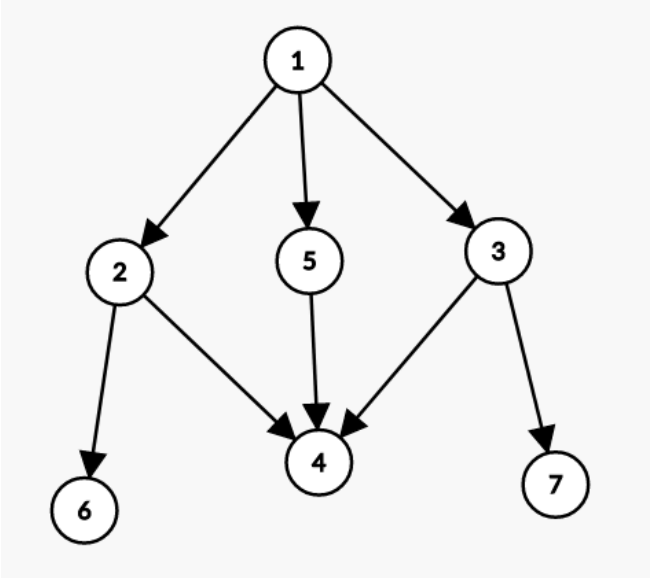
GRAPHS

Taking input of a graph

```
vector <int> adj[n+1];
    for(int i=0;i<a;i++)
    {
        int a, b;
        adj[a].pb(b);
        adj[b].pb(a);
    }
```

Breadth First Search (BFS)

FINAL ORDER OF TRAVERSAL: 1 2 5 3 6 4 7
Also known as level order traversal.



1 -> visited
q.push(1);
while(!q.empty())

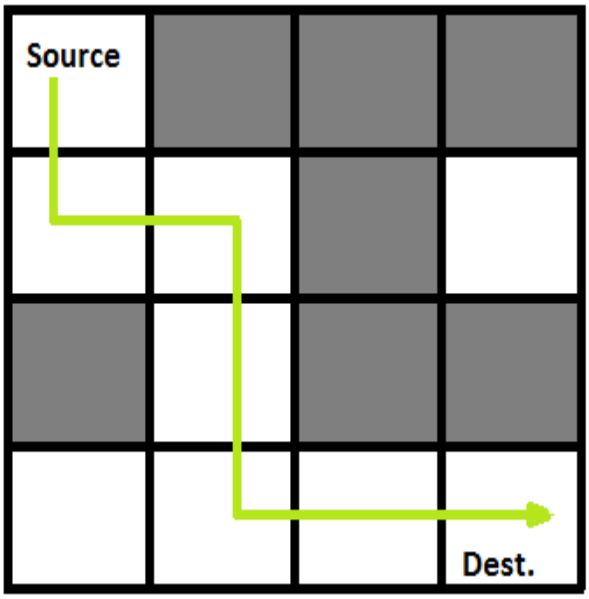
1
2 5 3
5 3 6 4
6 4 7
4 7
7

1 2 5 3 6 4 7

```
queue <int> q;
    bool vis[n+1];
    memset(vis, false, sizeof(vis));
    for(int i=1;i<=n;i++)
    {
        if(vis[i])
        continue;
        q.push(i);
        vis[i]=true;
        while(!q.empty())
        {
            int u=q.front();
            q.pop();
            for(auto to:adj[u])
            {
                if(!vis[to])
                {
                    vis[to]=true;
                    q.push(to);
                }
            }
        }
    }
```

BFS in a grid

You are given a n*m grid with some cells blocked and some unblocked you have to find the length of the shortest path from starting to ending cell?



		2			
	2	X-1, y	2		
2	X, y-1	X, y	X, y+1	2	
	2	X+1, y	2		
		2			

0 -> 1 1 1 1 -> 1 1 1 2 2 2 -> 1 1 2 2 2 2 2’

```
int n, m;
bool val(int x, int y)
{
    if(x<1 || y<1 || x>n || y>m || a[x][y]==1)
    return false;
    return true;
}
int32_t main()
{
    IOS;
    int dx[]={1, -1, 0, 0};
    int dy[]={0, 0, 1, -1};
    int a[n+1][m+1];
    for(int i=1;i<=n;i++)
    {
        for(int j=1;j<=m;j++)
        cin>>a[i][j];
    }
    int x1, y1, x2, y2;
    queue <pii> q;
    q.push({x1, y1});
    int dist[n+1][m+1];
    fill(dist, -1);
    dist[x1][y1]=0;
    while(!q.empty())
```

```
    {
        pii p=q.front();
        int x=p.ff, y=p.ss;
        q.pop();
        for(int i=0;i<4;i++)
        {
            int nx=x+dx[i], ny=y+dy[i];
            if(val(nx, ny) && dist[nx][ny]==-1)
            {
                dist[nx][ny]=dist[x][y]+1;
                q.push({nx, ny});
            }
        }
    }
    cout<<dist[x2][y2];
}
```

Homework Problems:

Counting Rooms

CSES - Labyrinth

Building Roads

Message Route

Round Trip

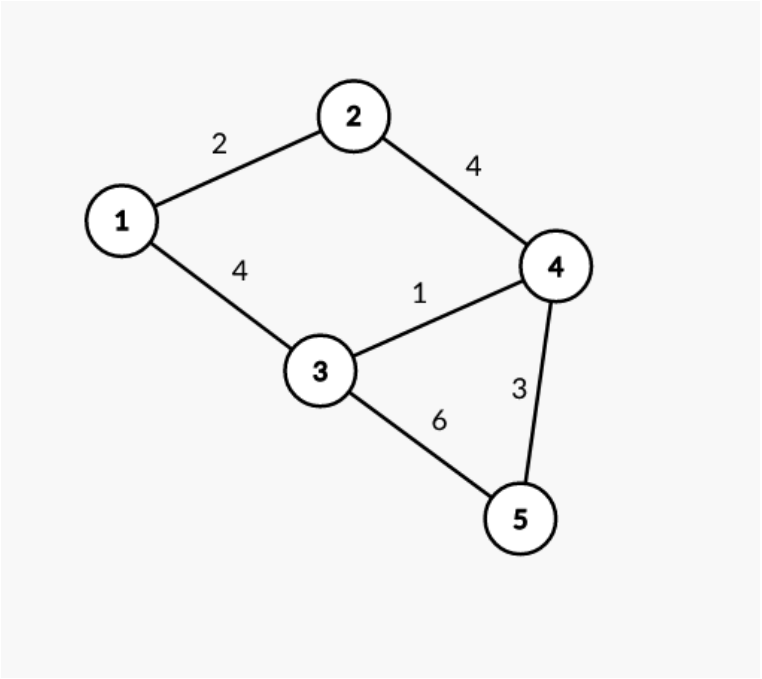
CSES - Monsters

Shortest Routes I

Shortest Routes II

Bonus: Building Teams Try to solve this.
PPT-https://docs.google.com/presentation/d/1QNMbZQcJNHm1e5-1xpXLUxkQ5fFF9FGeYNdbZDvodFo/edit#slide=id.gfc6e1da210_0_397

Dijkstra Algorithm



Undirected Weighted graph

Queue

Distance: 1 2 3 4 5
0 2 4 5 8

Queue - {distance, node}
{0, 1}
{4, 3}, {2, 2}
{4, 3}, {6, 4}
{6, 4}, {5, 4}, {10, 5}
{6, 4}, {10, 5}, {8, 5}
{10, 5}, {8, 5}
{10, 5}

- Single Source Shortest Path
- Does not work in the case of negative cycles
- Does not work in undirected graph having negative weights
- Directed graph with negative weights but no cycle - ??
- O((N+E)log(E))

Problem - [Problem - 20C - Codeforces]

```
/*

Priority Queue - MaxHeap - priority_queue<type> pq;
{1, 3, 2, 6} -> pq.top() -> 6

MinHeap - priority_queue<type, vector<type>, greater<type>> pq;
{1, 3, 2, 6} -> pq.top() -> 1

*/
```

```
     // Min heap using priority queue
    priority_queue<pair<int,int>, vector<pair<int,int>>,
greater<pair<int,int>>> pq;
    int dist[n+1], par[n+1];
    for(int i=0;i<=n;i++){
        dist[i] = INF;
        par[i]=i;
    }

    dist[1]=0;
    pq.push({0, 1});

    while(!pq.empty()){
        pair<int,int> p = pq.top();
        pq.pop();
        int u = p.second;

        // Check for stale node
        if(p.first>dist[u]) continue;

        for(auto nxt:g[u]){
            int v = nxt.first;
            int w = nxt.second;

            // If current edge relaxates the distance of node v
            if(dist[v]>dist[u]+w){
                dist[v] = dist[u]+w;
                par[v] = u;
                pq.push({dist[v], v});
            }
        }
    }

    if(dist[n]==INF){
        cout<<-1;
        return 0;
    }

    vector<int> path;

    int on = n;
```

```
    path.push_back(on);
    while(par[on]!=on){
        on = par[on];
        path.push_back(on);
    }

    reverse(path.begin(), path.end());

    for(auto i:path) cout<<i<<" ";
```

Dijkstra Using set

```
s.insert({0, 1});
    while(!s.empty()){
        auto p = *s.begin();
        s.erase(s.begin());
        int u = p.second;
        for(auto to: g[u]){
            int v = to.first;
            int w = to.second;
            if(dist[v] > dist[u]+ w){
                auto it = s.find({dist[v], v});
                if(it != s.end()) s.erase(it);
                dist[v]  = dist[u] + w;
                s.insert({dist[v],v});
                par[v] = u;
            }
        }
    }
```

[SPOJ.com - Problem KATHTHI]

NORMAL BFS
0
1 1 1 1 1
1 1 1 2 2 2 2
1 1 2 2 2 2 2
2 2 2 2 2 2
2 2 2 2 2 2 3 3

Multisource
0 0 0 0 0
0 0 0 0 1 1
0 0 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 11 1 1 1 1 2 2 2 2

0
0 1 0 1
1 0 1 0 1 1 0
 0 1 0 1 1 0 2 2 1 1

—----------

0-1 BFS
0 weighted edges insert at front
1 weighted insert at end

0
0 0 1 1
0 0 0 1 1 1 1
1 1 1 1 1 1 1 1 1 1 2 2 2

Code:

```
    int test;
    cin>>test;

    while(test--){
        int r, c;
        cin>>r>>c;

        string grid[r];
        for(int i=0;i<r;i++){
            cin>>grid[i];
        }

        int dx[] = {-1, 1, 0, 0};
        int dy[] = {0, 0, -1, 1};

        int dis[r][c];
        for(int i=0;i<r;i++){
            for(int j=0;j<c;j++){
                dis[i][j]=INF;
```

```
            }
        }

        dis[0][0]=0;
        deque<pair<int,int>> q;
        q.push_back({0, 0});

        while(!q.empty()){
            pair<int,int> p = q.front();
            q.pop_front();

            int x = p.first;
            int y = p.second;
            for(int k=0;k<4;k++){
                int nx = x + dx[k];
                int ny = y + dy[k];


if(nx<0||nx>=r||ny<0||ny>=c||dis[nx][ny]<=dis[x][y]) continue;

                // 0-edge
                if(grid[x][y]==grid[nx][ny]){
                    dis[nx][ny] = dis[x][y];
                    q.push_front({nx, ny});
                }else if(dis[nx][ny]>dis[x][y]+1){
                    // 1-edge
                    dis[nx][ny] = dis[x][y]+1;
                    q.push_back({nx, ny});
                }
            }

        }
        cout<<dis[r-1][c-1]<<"\n";
    }
```