**DAA ASSIGNMENT**

**TEAM MEMBERS: -**

1. Pranav Prashob - RA1911032010036
2. Varun Deshpande - RA1911032010047

**Question:**

Maze game is a well-known problem, where we are given a grid of 0’s and 1’s, 0’s corresponds to a place that can be traversed, and 1 corresponds to a place that cannot be traversed (i.e. a wall or barrier); the problem is to find a path from bottom left corner of grid to top right corner; immediate right, immediate left, immediate up and immediate down only are possible (no diagonal moves). We consider a variant of the maze problem where a cost (positive value) or profit (negative value) is attached to visiting each location in the maze, and the problem is to find a path of least cost through the maze.

**Code:**

#include<bits/stdc++.h>

using namespace std;

#define debug(x) cout << #x << " " << x <<endl;

#define endll "\n"

#define pb push\_back

#define pr(x,y) pair<x,y>

#define ff first

#define ss second

#define mpr make\_pair

#define ll long long

#define pp pr(ll,pr(ll,ll))

const ll inf=1e9;

bool vis[1001][1001];

ll adj[1001][1001];

ll wgt[1001][1001];

ll dist[1001][1001];

ll dir[] = {-1, 0, 1, 0, -1};

bool isSafe(ll r, ll c, ll n, ll m) {

if(r<=0 || r>n || c<=0 || c>m || vis[r][c] || adj[r][c]) {

return false;

}else {

return true;

}

}

void dijkstra(ll x, ll y, ll n, ll m) {

priority\_queue<pp, vector<pp>, greater<pp>> pq;

pq.push(mpr(wgt[n][1],mpr(x ,y)));

vis[x][y]=true;

dist[x][y]=wgt[x][y];

while(!pq.empty()) {

pp p=pq.top();

ll d=p.ff;

ll r=p.ss.ff;

ll c=p.ss.ss;

pq.pop();

for(ll i=0;i<4;i++) {

if(isSafe(r+dir[i],c+dir[i+1],n,m) && d+wgt[r+dir[i]][c+dir[i+1]]<dist[r+dir[i]][c+dir[i+1]]) {

ll rr=r+dir[i];

ll cc=c+dir[i+1];

vis[rr][cc]=true;

dist[rr][cc]=dist[r][c]+wgt[rr][cc];

pq.push(mpr(dist[rr][cc],mpr(rr,cc)));

}

}

}

}

void solve() {

ll n,m,x,y;

cout<<"INPUT THE DIMENSIONS :";

cin>>n>>m;

cout<<endl;

cout<<"INPUT THE MAZE :"<<endl;

for(ll i=1;i<=n;i++) {

for(ll j=1;j<=m;j++) {

cin>>adj[i][j];

}

}

cout<<"INPUT THE COST :"<<endl;

for(ll i=1;i<=n;i++) {

for(ll j=1;j<=m;j++) {

cin>>wgt[i][j];

}

}

for(ll i=1;i<=n;i++) {

for(ll j=1;j<=m;j++) {

dist[i][j]=inf;

if(adj[i][j]) {

wgt[i][j]=inf;

}

}

}

ll r1=n,c1=1; // Start Points

dijkstra(r1,c1,n,m);

ll r2=1,c2=m; // End Points

if(dist[r2][c2]==inf) {

cout<<"THERE'S NO PATH FROM (N,1) to (1,M)"<<endll;

}else {

cout<<"Minimum Cost to reach the Top Left position is : ";

cout<<dist[r2][c2]<<endll;

}

}

int main() {

solve();

return 0;

}

**Explanation:**

The Path finding problem in a maze with obstacles in some cells can be broken down into smaller problems, so as to make it easier. Considering the whole maze as a graph with each vertex depicting a cell having 0 to 4 edges connected to it.

Now, to distinguish a cell with an obstacle, we are going to make that vertex have no edge, or here what we have done is, we made the edges with the largest cost possible(1e9). Hence, that vertex won’t ever be considered in the path with minimum cost.

So, as all the preparations to make the maze into a graph has been done. Now, we have to find a suitable algorithm to find the minimum cost from a source, i.e. (n,1), to the destination, i.e. (1,m). As for the algorithm, we have chosen Dijkstra’s algorithm. At each vertex we keep the minimum possible cost that is connected from the source.

TIME COMPLEXITY: O(ElogV)

**Sample Input 1:**

2 2

0 0

0 0

2 3

4 5

**Sample Output 1:**

Minimum Cost to reach the Top Left position is: 9

**Sample Input 2:**

2 2

1 0

0 1

2 3

4 5

**Sample Output 2:**

THERE'S NO PATH FROM (N, 1) to (1, M)