**Bfs [2D], O(V+E)**

**Dx[], Dy[] is corresponding move. cost[i][j] is the minumum distance from source**

**Tested on: UVA 439 Knight Moves**

int Dx[]= {-2,-1, 1, 2, 2,1,-1,-2}; //for knight move

int Dy[]= {-1,-2,-2,-1, 1,2, 2, 1}; //for knight move

void bfs(int sx,int sy,int dx,int dy) //sx-sy source position, dx-dy destination position

{

int x, y;

queue< pair<int,int> >Q;

pair<int, int>pii;

pii.first=sx, pii.second=sy;

Q.push(pii);

cost[sx][sy]=0;

CLR(vis);

vis[sx][sy]=true;

while(!Q.empty())

{

pii=Q.front();

Q.pop();

if(pii.first==dx && pii.second==dy)

{

cout<<cost[pii.first][pii.second]<<"\n";

break;

}

for(int i=0; i<8; i++) //8=number of possible moves

{

x=pii.first + Dx[i];

y=pii.second + Dy[i];

if( x>=1 && x<=8 && y>=1 && y<=8 && !vis[x][y]) //boundary and visited check

{

vis[x][y]=true;

cost[x][y]=cost[pii.first][pii.second]+1;

Q.push(make\_pair(x, y));

}

}

}

return;

}