***1)code dfs(Deapth \_firs\_ search)in page ();***

***aplcation on it:***

***1)count compounnet page();***

***2)view for page();***

***3) EdgeClassification in page ();***

***4)*** Flood Fill...given maze where cells are . or X. You start at 0, 0..how many cells you could reach?

//بختصار دة هيمشى على كل النقط تمام بشكل عمودى كلهم دى كل مهمتة وفى تطبيقات ممكن استخدمها بسببة هنشوفها

***Code dfs :***

int n, e;

vector< vector<int> > adj;

vector<bool> visited;

void dfs(int node)

{

visited[node] = true;

rep(i, adj[node]) // rep from 0 to size().

{

int child = adj[node][i];

if (!visited[child]) // To avoid cyclic behavior

dfs(child);

}

//topsort.push\_back(node); // DAG // Other way Indegree / Outdegree

}

//in main

cin>>n>>e;

adj = vector< vector<int> >(n);

visited = vector<bool>(n);

lp(i, e) {

int from, to;

cin>>from>>to;

adj[from-1].push\_back(to-1);

}

***1)count number of componrnt***

//عينى هيعد كام مجموعة يعنى لو ل النقط توصلة ببعض كدة مجموعة وحدة ل

***//لو فى مجموعة اربع نقط متصبين مع بعض باى شكل المهم متصلين مع بعض وفى زيهم 3 نقط متصلين مع بعض التلت نقط مش متصلة بالاربع نقط كدة فى مكونين***

int ConnectedComponenetsCnt()

{

int cnt = 0;

lp(i, n)

{

if(!visited[i]) // Then no one reach this isolated node yet and its neighbors.

{

dfs(i);

cnt++;

}

}

return cnt;

}

***2)secont aplacation :***

//دة عشان يعرض الحاجة مش بس يعلم زى الى فوق

// مثال

vector<int> topsort; //above main

int n, e;

vector< vector<int> > adj;

vector<bool> visited;

void dfs(int node)

{

visited[node] = true;

rep(i, adj[node]) // rep from 0 to size().

{

int child = adj[node][i];

if (!visited[child]) // To avoid cyclic behavior

dfs(child);

}

topsort.push\_back(node); // DAG // Other way Indegree / Outdegree🡸🡸🡸🡸🡸secnd type

}

***3)third application :***

***// عشان احدد نوع الكبرى هو اية بالضبط عموما اهم نوع الى هو دخل فى دايرة***

vector<int> start, finish; //above main

bool anyCycle = 0;

int timer = 0;

void dfs\_EdgeClassification(int node)

{

start[node] = timer++;

rep(i, adj[node])

{

int child = adj[node][i];

if (start[child] == -1) // Not visited Before. Treed Edge

dfs\_EdgeClassification(child);

else {

if(finish[child] == -1)// then this is ancestor that called us and waiting us to finish. Then Cycle. Back Edge

anyCycle = 1;

else if(start[node] < start[child]) // then you are my descendant

; // Forward Edge

else

; // Cross Edge

}

}

finish[node] = timer++;}

***In main***

lp(i, n) if(start[i] == -1)

dfs\_EdgeClassification(i);cout<<anyCycle;

--------------------------------------------------------------------------------------------------------------------------------------------------------------

***4)four aplacation Flood Fill:***

// Flood Fill...given maze where cells are . or X. You start at 0, 0..how many cells you could reach?

***//consider evry node is a edage***

/\*

..X.

.X.X

..X.

...x.

..x..

.x...

x....

....X...

....XXXX

..X.....

.X....XX

..X.X.X.

..X...X.

...X..XX

\*/

// A reachable block is called connected components. Each set of positions reachable together are connected component.

const int MAX = 100;

bool valid(int i, int j) { return 1; }//complet it

char maze[MAX][MAX];

bool vis[MAX][MAX];

int cnt = 0;

void cntReachalbleCells(int r, int c)

{

if( !valid(r, c) || maze[r][c] == 'X' || vis[r][c] == 1)

return; // invalid position or block position

vis[r][c] = 1; // we just visited it, don't allow any one back to it

cnt++;

// Try the 4 neighbor cells

cntReachalbleCells(r, c-1);

cntReachalbleCells(r, c+1);

cntReachalbleCells(r-1, c);

cntReachalbleCells(r+1, c);

}