***1)BFS\_CODE :***

***2) path ofBFS:***

***1)BFS CODE :***

// بختصار دة هيجبل بعد العمودى كل نود عن نقطة انا بعتهالة

//بص يغالى هو الفكرة انى بمسك كل رقم اطلعة ودخل العيال بتاعتة الى فى المستوى الى تحتة لحاد ميخلص المستوى وهكذا

//مثال هو انا دخلت فى الاول خالص الواحد والبعد بتاعة صفر هحوش الواحد ودخل عيالة الى هم مثلا 2 3 4 والبعد بتاعهم 1 هحوش ال2 ودخل عيالها الى هما مثلا 5 6 والبعد بتاعهم 2 وبعدين احوش ال 3 ومثلا ملهاش عيال فمدخلش حاجة وحوش ال4 ودخل عيالها الى هى مثلا 7 والبعد 2 وهكذا

vector<int> BFS(int s, vector<vector<int> > & adjList) {

vector<int> len(sz(adjList), OO);

queue< pair<int, int> > q;

q.push(MP(s, 0)), len[s] = 0;

int cur, dep;

while(!q.empty()) {

pair<int, int> p = q.front(); q.pop();

cur = p.first, dep = p.second;

rep(i, adjList[cur]) if (len[adjList[cur][i]] == OO) //دى عشان لو فى اكتر من اب للحاجة هاخد اول واح وصلها

q.push(MP(adjList[cur][i], dep+1)), len[adjList[cur][i]] = dep+1;

}

return len; //cur is the furthest node from s with depth dep

}

***In main:***

int n, e;

cin>>n>>e;

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

lp(i, e) {

int from, to;

cin>>from>>to;

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

}

***2) BFS code in beater way:***  //half time &&momery from above

ector<int> BFS2(int s, vector<vector<int> > & adjList) {

vector<int> len(sz(adjList), OO);

queue<int> q;

q.push(s), len[s] = 0;

int dep = 0, cur = s, sz = 1;

for (; !q.empty(); ++dep, sz = q.size()) {

while (sz--) {

cur = q.front(), q.pop();

rep(i, adjList[cur]) if (len[adjList[cur][i]] == OO)

q.push(adjList[cur][i]), len[adjList[cur][i]] = dep+1;

}

}

return len; //cur is the furthest node from s with depth dep

}

// الدللة دة بتعمل اية اقلك بص هو هيديك بداية ونهاية وعاوز كل الارقام من البتداية للنهاية

//يعنى الكود الى فوق كان بيديك بعد كل النقاط عن نقطة معطاه تمام الى جاى دى بيديك المسار بقا يعنى كذا متوصلة بكذا لحاد النقطة النهائية الى هو عطهالك

vector<int> BFSPath(int s, int d, vector<vector<int> > & adjList) {

vector<int> len(sz(adjList), OO);

vector<int> par(sz(adjList), -1);

queue<int> q;

q.push(s), len[s] = 0;

int dep = 0, cur = s, sz = 1;

bool ok = true;

for ( ; ok && !q.empty(); ++dep, sz = q.size()) {

while (ok && sz--) {

cur = q.front(), q.pop();

rep(i, adjList[cur]) if (len[adjList[cur][i]] == OO)

{

q.push(adjList[cur][i]), len[adjList[cur][i]] = dep+1, par[ adjList[cur][i] ] = cur;

if(adjList[cur][i] == d) // we found target no need to continue

{

ok = false;

break;

}

}

}

}

vector<int> path;

while(d != -1) {

path.push\_back(d);

d = par[d];

}

reverse( all(path) );

return path;

}