Artificial Intelligence Lab

CSE-0408 Summer 2021

SANJIDA AKTER
Department of Computer Science and Engineering
State University of Bangladesh (SUB)
Dhaka, Bangladesh
Muktaeva11@gmail.com

n

***Index Terms*—8 puzzle**

I. Code

include¡bits/stdc++.h¿ using namespace std; define D(x) cerr¡¡$_{LINE_{<<":"<<x<<"->"<<x<<endldefinerep(i,j)for(inti=0;i<3;i++)for(intj=0;j<3;j++)definePIIpair<int,int>typedefvector<vector<int>>vec2D;}}$

const int MAX = 1e5+7; int t=1, n, m, l, k, tc;

int dx[4] = 0, 0, 1, -1; int dy[4] = 1, -1, 0, 0;

vec2D init 8, 1, 2, 3, 6, 4, 0, 7, 5 ; vec2D goal 1, 3, 2, 8, 0, 4, 7, 6, 5 ; /// using a structure to store information of each state struct Box vec2D mat 0,0,0 , 0,0,0, 0,0,0 ; int diff, level; int x, y; int lastx, lasty; Box(vec2D a,int b = 0, int c = 0, PII p = 0,0, PII q = 0,0) rep(i,j) mat[i][j] = a[i][j]; diff = b; level = c; x = p.first; y = p.second; lastx = q.first; lasty = q.second; ;

/// operator overload for which bases priority queue work bool operator ¡ (Box A, Box B) if(A.diff == B.diff) return A.level ¡ B.level; return A.diff ¡ B.diff;

/// heuristic function to calculate mismatch position int heuristic$_function(vec2Da, vec2Db)intret(0); rep(i,j)if(a[i][j]! = b[i][j])ret - -; returnret;$

/// checking puzzle boudaries bool check(int i, int j) return i¿=0 and i¡3 and j¿=0 and j¡3;

/// this function used to show state status void print(Box a) rep(i,j) cout ¡¡ a.mat[i][j] ¡¡ (j == 2 ? "" : " "); cout ¡¡ " heuristic Value is : " ¡¡ -a.diff ¡¡ ""; cout ¡¡ " Current level is : " ¡¡ -a.level ¡¡ "";

/// used to get new state which can be jump from current state Box get$_new_state(Boxnow, intxx, intyy)Boxtemp = now; swap(temp.mat[temp.x][temp.y], temp.mat[xx][yy]); temp.diff = heuristic_f$

/// this is modified version of dijkstra shortest path algorithms /// basically work on those state first which heuristic value lesser void dijkstra(int x, int y) map ¡ vec2D, bool ¿ mp; priority$_queue < Box > PQ; intnD = heuristic_function(init, goal); Boxsrc = init, nD, 0, x, y, -1, -1; PQ.push(src); intstate = 0; while(!PQ.empty())state + +; Boxnow = PQ.top(); PQ.pop(); cout << "Stepno : " << state << ""; print(now); if(!now.d$

signed main() puts("Current State:"); rep(i,j) cout ¡¡ init[i][j] ¡¡ (j == 2 ? "" : " "); puts(""); puts("Goal State:"); rep(i,j) cout ¡¡ goal[i][j] ¡¡ (j == 2 ? "" : " "); puts("............Search Started..............."); rep(i,j) if(!init[i][j]) dijkstra(i,j); /// this will find zero-th position and start return 0;

II. BFS

include¡bits/stdc++.h¿ using namespace std;

define MX 110

vector ¡ int ¿ graph[MX]; bool vis[MX]; int dist[MX]; int parent[MX];

void bfs(int source) queue ¡ int ¿ Q; // initialization vis[source] = 1; dist[source] = 0; Q.push(source);

while(!Q.empty()) int node = Q.front(); Q.pop();

for (int i = 0; i ¡ graph[node].size(); i++) int next = graph[node][i]; if (vis[next] == 0) vis[next] = 1; // visit dist[next] = dist[node] + 1; // update Q.push(next); // push to queue

// set parent parent[next] = node;

/* input: 7 9 1 2 1 3 1 7 2 3 3 7 2 4 4 5 3 6 5 6 1 */

// path printing functions

// recursive function void printPathRecursive(int source, int node) if (node == source) cout ¡¡ node ¡¡ " "; // print from source return; printPathRecursive(source, parent[node]); cout ¡¡ node ¡¡ " ";

// iterative function void printPathIterative(int source, int node) vector¡int¿ path$_vector;$

while(node != source) path$_vector.push_back(node); node = parent[node]; path_vector.push_back(source); //insertingsource$

for (int i = path$_vector.size() - 1; i >= 0; i - -)cout << path_vector[i] << "";$

int main() int nodes, edges; cin ¿¿ nodes ¿¿ edges;

for (int i = 1; i ¡= edges; i++) int u, v; cin ¿¿ u ¿¿ v; graph[u].push$_back(v); graph[v].push_back(u);$

int source; cin ¿¿ source;

bfs(source);

cout ¡¡ "From node " ¡¡ source ¡¡ endl; for (int i = 1; i ¡= nodes; i++) cout ¡¡ "Distance of " ¡¡ i ¡¡ " is : " ¡¡ dist[i] ¡¡ endl; cout ¡¡ endl;

// path printing example

// recursive version for (int i = 1; i ¡= nodes; i++) cout ¡¡ "Path from " ¡¡ i ¡¡ " to source: "; printPathRecursive(source, i); cout ¡¡ endl;

cout ¡¡ endl;

// iterative version for (int i = 1; i ¡= nodes; i++) cout ¡¡ "Path from " ¡¡ i ¡¡ " to source: "; printPathIterative(source, i); cout ¡¡ endl;

return 0;

ACKNOWLEDGMENT

I would like to thank my honourable**Khan Md. Hasib Sir** for his time, generosity and critical insights into this project.