Page No.:
Date: / /

Date: 4/2/2022 Roll: 31145

Batch: M1 Sub.: LP2 (AI)

• ASSIGNMENT 2 •

* Problem Statement:

Implement A* algorithm for any game search algorithm problem.

* Objectives:

① To be able to design and implement A* algorithm to solve the game search problem.

② To take source and destination co-ordinates as i/p and find out the shortest path from source to destination.

* Outcomes:

① Implement A* algorithm which can be used to solve complex search problems.

② Find the shortest distance between source and destination in real-life situations like maps, games with obstacles.

* Theory:

• A* algorithm:

It is a searching algorithm that searches for shortest path between initial and final state. It is used in applications like maps. A* search is best and most popular path finding technique for ~~garp~~ graph traversals. It is also optimal meaning that there is no other algorithm better at optimization than A*.

- **Working:**

Consider a square grid having many obstacles and were given a starting, an ending cell. We want to reach target from source as quickly as possible.

At each set step the algorithm tries to find the lowest value of 'f' which is sum of parameters 'g' and 'h'. Thus, $f(n) = g(n) + h(n)$

where, $g(n)$ = Actual cost path from start to current.

$h(n)$ = Heuristic cost from current to target.

$f(n)$ = Actual cost from start to target.

We consider two lists in which we maintain already visited as well as not visited cells. At each iteration, we can move forward in eight directions.

At current cell,

① For calculating $g(n)$, all the straight directions (Up, Down, Left, Right) are added by 1 to the current $g(n)$.

② For all diagonal cells, $\sqrt{2}$ is added to the current $g(n)$

③ There are a number of ways to find heuristic distance. However we will consider euclidean distance.

④ Euclidean distance is the distance between current cell and goal cell using distance formula.

⑤ Select the direction with the lowest $g(n)+h(n)$.

* **Algorithm:**

① Start

② Initialize open and close lists. (Start node in open list)

③ while open list is not empty

3a) Find node with least 'f' on open list. Call it 'q'

3b) Pop 'q' from open list.

3c) Generate 'q's 8 successors and set their parents to q.

3d) For each successor –

3d.1) Calculate 'f'

3d.2) If a node with same position with successor is present in open list, skip the successor.

3d.3) Similarly, skip if a node in close list, else add to open list.

3e) Push 'q' to closed list.

④ End

* Test Case :

Note: 1 indicates cell is blocked
0 indicates cell is free

Input Grid	Source	Destination
[[1,0,1,1,1,1,0,1,1,1], [1,1,1,0,1,1,1,0,1,1], [1,1,1,0,1,1,0,1,0,1], [0,0,1,0,1,0,0,0,0,1], [1,1,1,0,1,1,1,0,1,0], [1,0,1,1,1,1,0,1,0,0], [1,0,0,0,0,1,0,0,0,1], [1,0,1,1,1,1,0,1,1,1], [1,1,1,0,0,0,1,0,0,1]]	(8,0)	(0,0)

Page No.:
Date: / /

Expected Output:

Destination found

Path: (8,0) → (7,0) → (6,0) → (5,0) → (4,1) → (3,2) → (2,1) → (1,0) → (0,0)

Result :

Passed

* Conclusion:

Through the assignment we have implemented A* algorithm using C++.