Name: Sanchit Kripalani
Batch: M1
Roll no: 31145

Date of submission: 4/2/22

Group A [Artificial Intelligence]

Assignment 2

- **Problem Statement:**

  Implement A* algorithm for any game search problem

- **Objective:**

  1. To be able to design and implement A star algorithm to solve the game search problem.
  2. To take source and destination co-ordinates as input, and find out the shortest path from source to destination in a given 2D grid.

- **Outcomes:**

  Students will be able to:-

  1. Implement A* algorithm which can be used to solve complex search problems.
  2. Find the shortest distance between source and destination in real-like situations like maps, games with obstacles.

- **Theory:**

  A* Algorithm-

  It is a searching algorithm that searches for the shortest path between initial and final state. It is used in applications like maps.

  Therefore, A* search is the best and most popular path-finding technique for graph traversals. It is also optimally efficient, meaning that there is no other algorithm better at optimization than A*.

Working -

consider a square grid having many obstacles and we are given a starting and target cell. we want to reach the target ~~for~~ from the source as quickly (lowest cost) as possible.

At each step, the algorithm tries to find the lowest value of 'f', which is in turn the 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.

There are 2 lists where we maintain the already visited as well as not visited cells. [Named closedList, openList resp.]

At each and every point of iteration, we can move forward in 8-directions [all the cells adjacent to current cell]

At current cell -

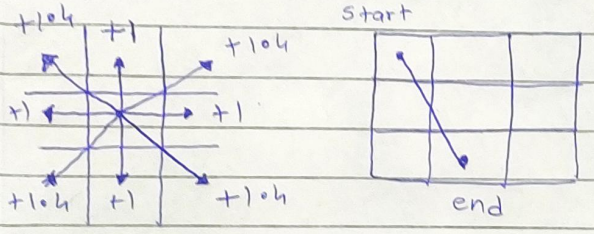
1. For calculating $g(n)$, all the straight directions (up, down, left, right) are added by 1 to the current $g(n)$.

2. For all the diagonal cells, $\sqrt{2} \approx 1.41$ is added to the current $g(n)$.

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

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

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



G(n) Calculation

$G'(n) = G(n) + 1 \,|\, 1.4$

h(n) Calculation

$H'(n) = \text{sqrt}[(\text{current.x} - \text{goal.x})^2 + (\text{current.y} - \text{goal.y})^2]$

- **Algorithm :**

① Initialize open and close list [start node in open list]

② while open list is not empty –

2.a) Find node with least f on open list, call it 'q'

2.b) Pop 'q' off open list

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

2.d) For each successor –

2.d.1) Calculate 'f' as mentioned in worked.

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

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

2.e) Push q to closed list

③ End Algorithm.

- Test Case

Note :- 1 = cell not blocked. 0 = cell blocked.

Grid

```
[ [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]
]
```

Src: (8,0)

Dest: (0,0)

Expected O/P :-

Destination found.

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

Actual O/P :-

Destination found.

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

Result = Success

- Conclusion :-

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