Problem1:

Modify ***AStarMaze*** to compare the behaviors of the ***Greedy Best-First*** and ***A* search*** algorithms.

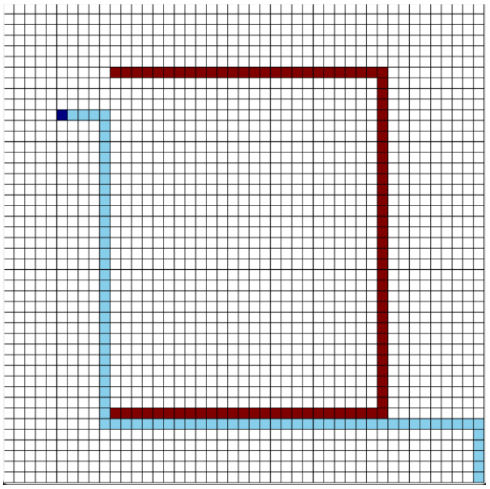


Fig1: A* search

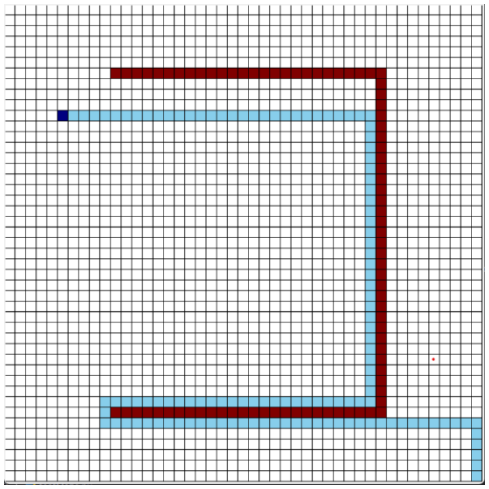


Fig2: Greedy Best-First search

Description: Agent needs to move in the maze (size: 45 rows, 45 columns), find the path (blue sky color) from Start (10, 5) to Goal (44, 44); agent can only go up, down, left, right with the cost of 1 hop (equals 1 small square (node) of the grid) per move; agent can't go through the wall (maroon color).

The search algorithm uses evaluation function: f(n) = g(n) + h(n) to evaluate the cost of movement. From the current node, the agent will hop to the *neighbor_node* which has smallest f(n) value.

Heuristic value h(n):

$$h(n) = |y_{(\text{current position})} - y_{(\text{goal position})}| + |x_{(\text{current position})} - x_{(\text{goal position})}|$$

where x, y are row, column position, respectively.
For example: $h(\text{Start}) = |10 - 44| + |5 - 44| = 73$.
$h(\text{Goal}) = |44 - 44| + |44 - 44| = 0$.
From Start, if agent moves right or down 1 hop to *neighbor_node*, it will get closer to the Goal and the h(*neighbor_node*) value will be lesser than h(Start) 1 hop.

From Start, if agent moves left or up 1 hop to *neighbor_node*, it will get farther from the Goal and the h(*neighbor_node*) value will greater than h(Start) 1 hop.

In this maze, therefore, the agent will move right and down to target the Goal.

Actual cost g(n): when agent moves 1 hop (any direction), g(n) will be added 1 unit.

Example:
From Start, g(Start) = 0, if agent moves 1 hop to *neighbor_node* (any direction), g(*neighbor_node*) = 1

Generally, when agent get closer to the Goal, h(n) → 0 and g(n) → higher value (depends on the path length).

In **Fig1**, using A* search algorithm with evaluation function: f(n) = g(n) + h(n), the agent moves from the Start straightly right or down to approach the Goal. This path has f(n) = 73, consistently, for every node because for each node to which the agent moves, h(n) value decreases 1 hop, and g(n) value increases 1 hop. Although f(n) is unchanged, but agent uses h(n) with reduced value to target the Goal and considers the g(n) value – actual cost – to choose the optimal path. Therefore, this is the cost-optimal algorithm (Total: 73 hops).

In **Fig2**, the agent uses Greedy Best-First search algorithm. In this algorithm, the actual cost is set as 0 (g(n) = 0, by changing the code in ***AStarMaze*** algorithm that setting the cost for moving to new position ***new_g = 0***). The evaluation function will be f(n) = h(n), which means that the agent will rely only on the estimate function h(n) to evaluate the movement's cost for choosing the next node. By this way, agent choose the node in a path that h(n) value goes down to zero, which **appears** to be closest to the Goal. Therefore, from Start (h(Start)=73), agent will move straightly far-most to the right. It reaches the wall, then turns down targeting the Goal. Unfortunately, it reaches the wall again at the bottom right where h(n)=17. This makes the agent keep moving along the wall to approach the Goal. The path from Start to Goal is not cost-optimal (Total: 123 hops).

Problem 2:

Repeat the above experiment but this time:

- Use the Euclidean Distance heuristic.

Change the code of heuristic function in ***AStarMaze*** algorithm as below:

def heuristic(self, pos):
return ((math.sqrt((pos[0] - self.goal_pos[0])**2 + (pos[1] - self.goal_pos[1])**2)))

- The agent is allowed to make diagonal moves (i.e., NE, NW, SE, SW) in addition to the usual N, S, E, and W moves.
  Change the code for agent's movement in ***AStarMaze*** algorithm as below:

  for dx, dy in [(0, 1), (0, -1), (1, 0), (-1, 0), (1, 1), (1, -1), (-1, 1), (-1, -1)]
  new_pos = (current_pos[0] + dx, current_pos[1] + dy)

- The moves are made randomly and not in any specific order.

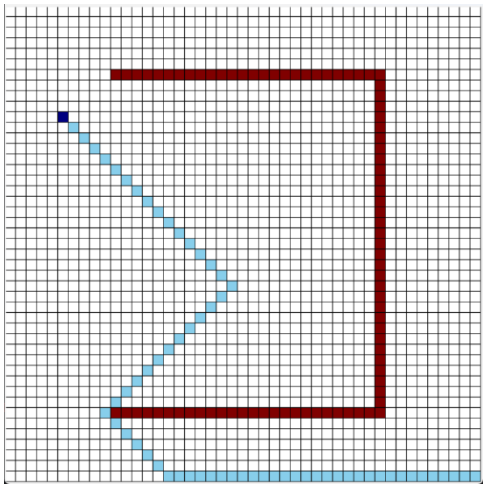


Fig3: A* search result

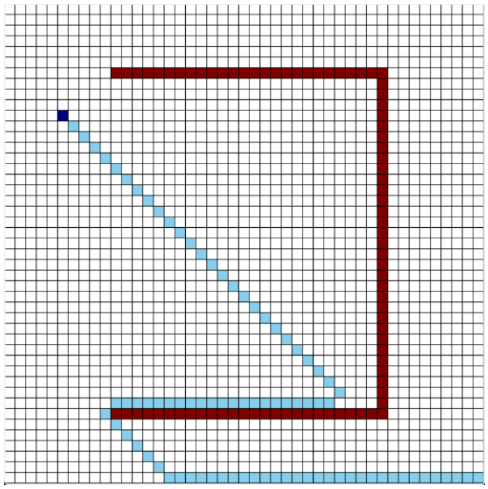


Fig4: Greedy Best-First search result

A* search takes 63 hops to reach the Goal (less than above A* search in Fig1)

Greedy Best-First search takes 83 hops to reach the Goal (less than above condition in Fig2)

When agent is allowed to make diagonal moves in combination with using Euclidian distance heuristic, both A* search and Greedy Best-First search algorithms help agent approach the Goal with less hops than the above conditions (Fig 1 & 2).

Like above (Fig2), Greedy Best-First search makes agent move toward the Goal using only heuristic value to choose the next moves closest to the Goal. Only when it reaches the wall, it changes direction to other way around. Therefore, this path is not optimal.

A* search in this case is more optimal than above (Fig1) because it takes diagonal move.

Problem 3:

The evaluation function in **AstarMaze** is defined as **f(n) = g(n) + h(n)**. A weighted version of the function can be defined as: $f(n) = \alpha*g(n) + \beta*h(n)$ where $\alpha, \beta \geq 0$

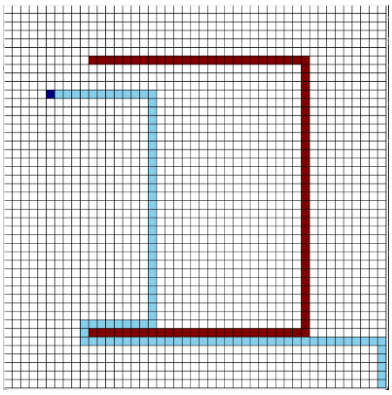
1. Explain different values of α and β affect the A* algorithm's behavior

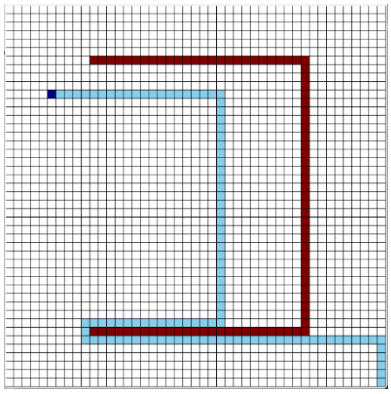
α	β	**Observed Behavior**
> 1	1	Agent moves from Start to Goal by going straightly down to the bottom, then turn right to approach the Goal. The path cost is optimal. (similar to Fig.1 – A* search)
1	>1	Agent takes a longer path to go from Start to Goal: (similar to Fig.2 – Greedy Best-First search)

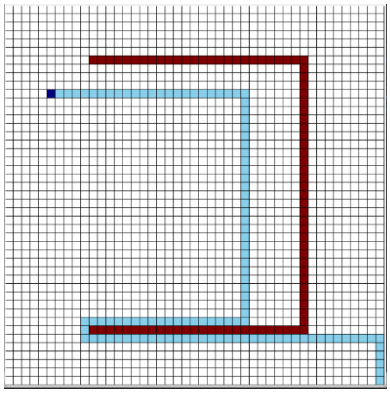
2. β can be considered the algorithm's bias towards states that are closer to goal. Run the algorithm for various values of the bias.

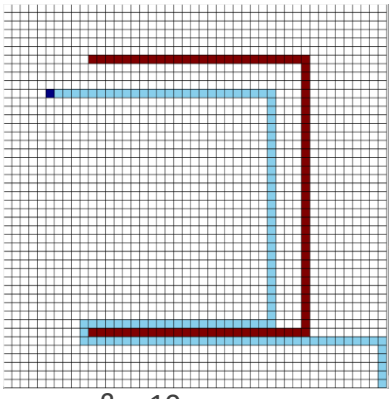
Change the code in AstarMaze as follows: (α=1, β various values: 2,4,6,10,26,27)

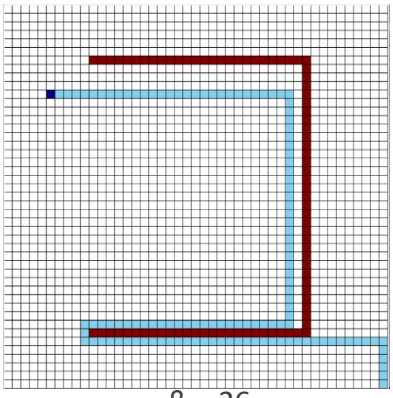
self.cells[new_pos[0]][new_pos[1]].f=α *new_g+ β *self.cells[new_pos[0]][new_pos[1]].h

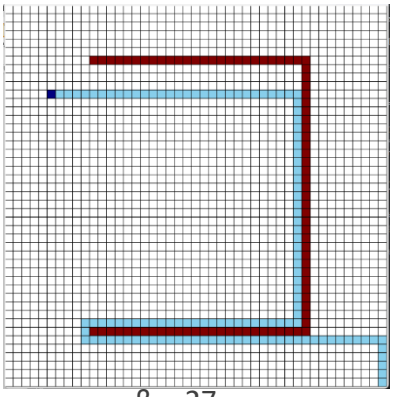

β = 2


β = 4


β = 6


β = 10


β = 26


β = 27

Increasing β value makes the algorithm become “greedier”, which means that it behaves like the Greedy Best-First search. Agent with this algorithm will choose moving to the node closer to the Goal. This bias sometimes makes the agent lost because the actual cost is not considered.

The optimal path is obtained when $\alpha/\beta >= 1$.