Report

Chinmay Kate

Note- Most of the extra code is to visualize therefore it is big code, but repetitive items are made into one function and used into the different algorithms. It may seem big but just many functions are written to visualize.

Code and Algorithm explanation-

- Common Code-
- `gen_moves(self, pos):`
  This Function generates the moves by moving right, down, left, up by taking position as input.

- `valid_move(self,move):`
  This Function checks if the move is valid with obstacles and boundary conditions by taking position as input.

- `str_to_array(self,string)`
  This Function converts string to array.

- `get_possible_moves(self):`
  This Function takes potential moves from the gen_moves() function and iterates over potential move also check if the move is valid or not by calling valid_move function. It also check if its in explored or not_explored dictonary and if it is not then populated the not_explored dictonary with the depth value. It moves current position from not_explored to explored.

- `goal_found(self)`
  This Function check if the goal is in the not_explored dictonary. And stores the path of the goal and depth.

- `explore_next_move(self):`
  This Function Determine next move to explore with the sorting Function based on the application like BFS, DFS, A*, Dijkstara and computes the path and depth value.

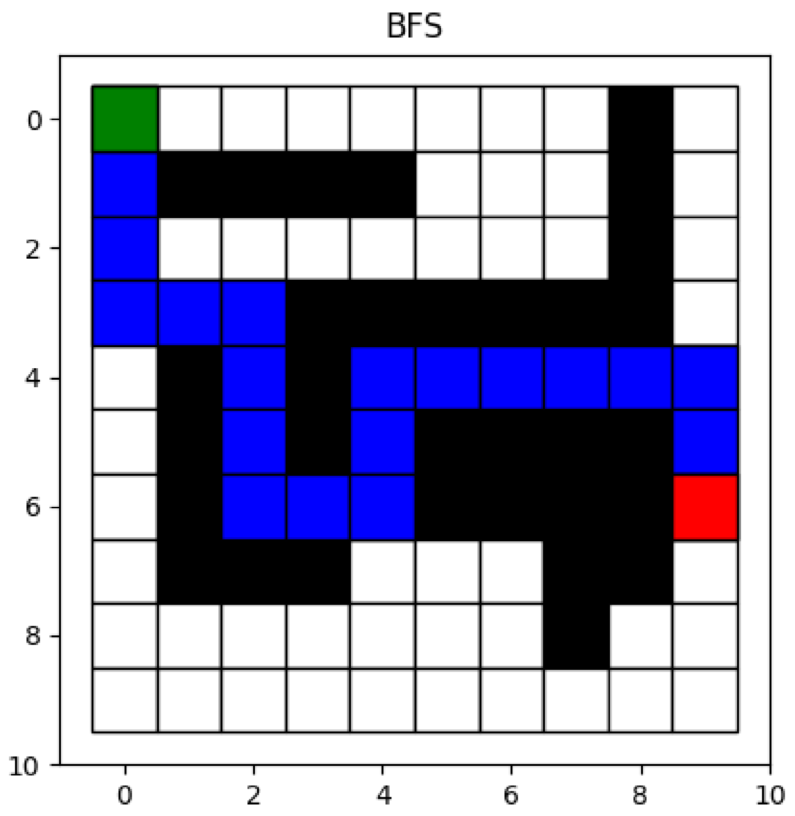
  Other Functions are similar and just twicked based on their algorithms.

  Every thing is commented well in the code.

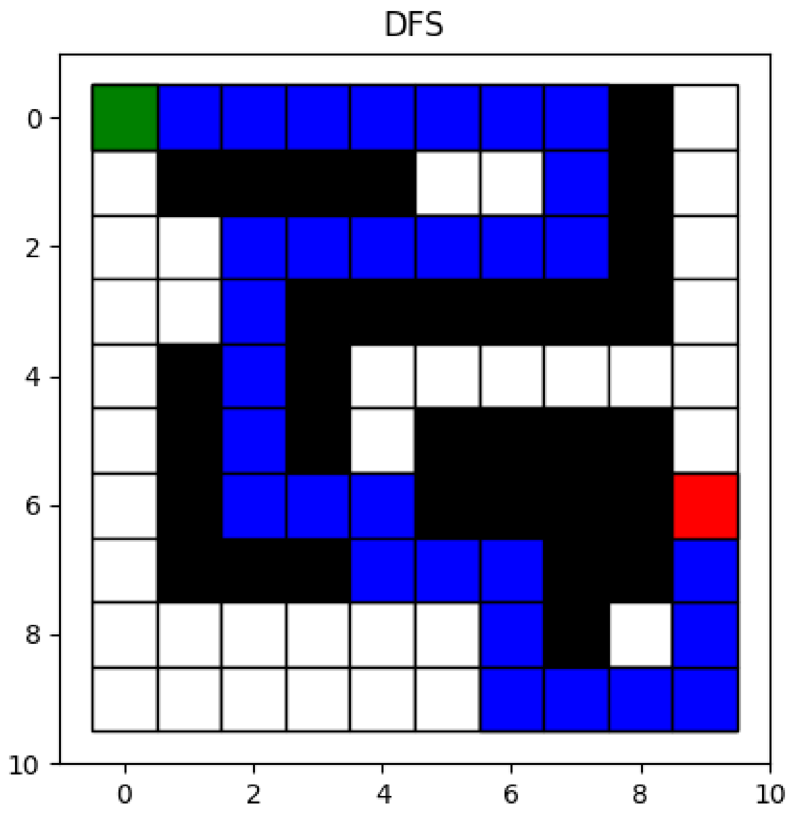
i) BFS- Here the neighbors are searched first and and neighbors' neighbors are searched till the goal is reached.

ii) DFS- Here one neighbor is selected and proceeded till the goal is reached or till it doesn't find the path. It backtracks if the node is already visited.
iii) Dijkstra- Here it works the same as BFS but min cost to go node is selected.
iv) A*- Here both cost to come and Heuristic cost is selected as parameter to visit the nodes and Heuristic is calculated by Manhattan distance.

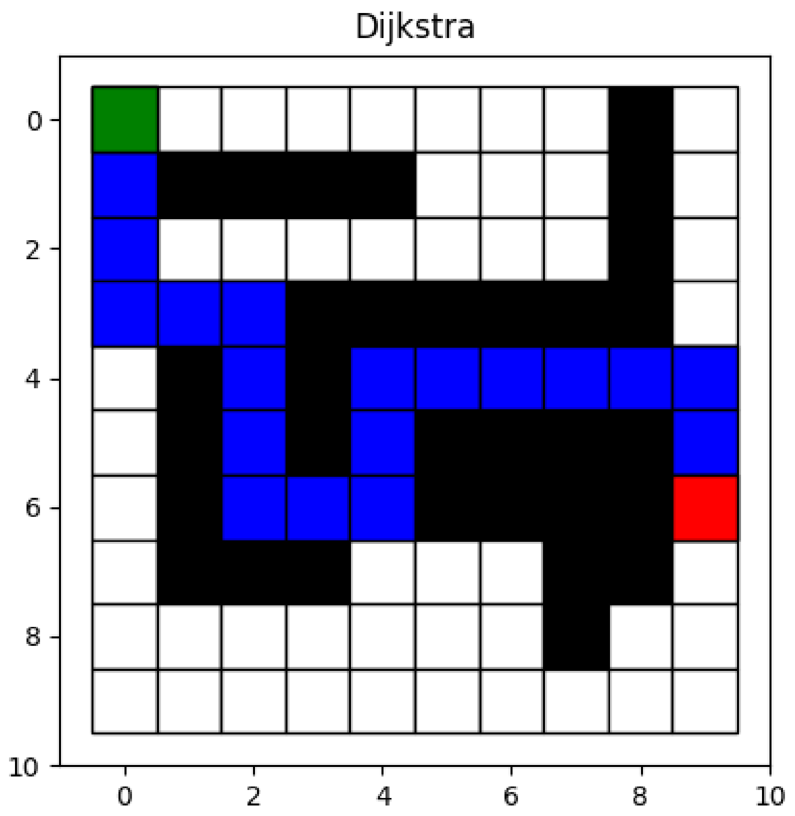
<u>Map.csv Results</u>



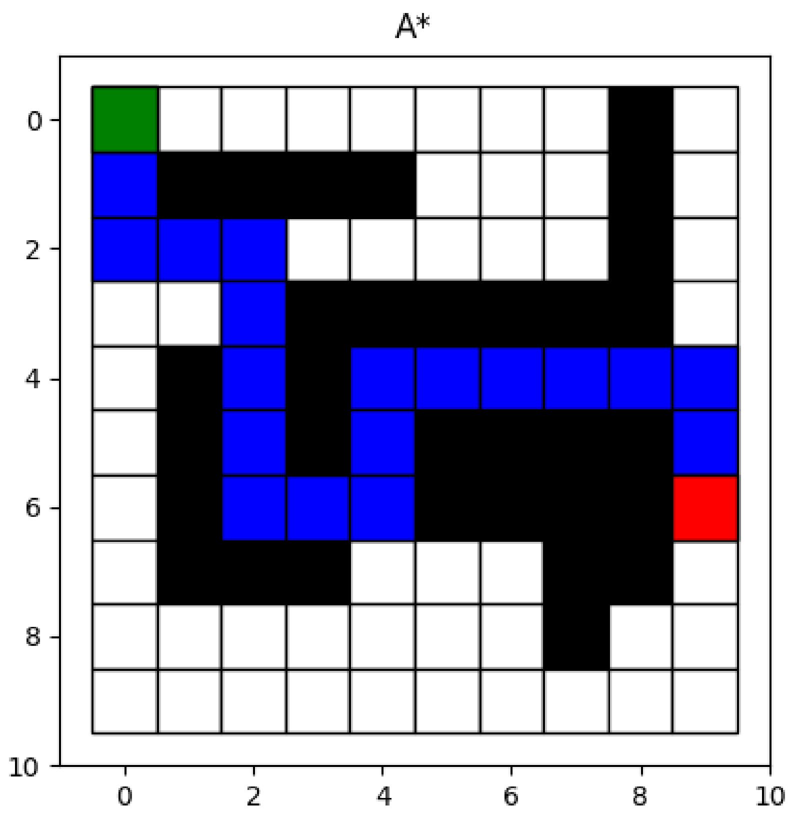
Here it takes upper route from (6,4) as it takes less steps to reach the goal that from the bottom route from (6,4) as well as (3,0)



Here it moves from the right because priority to move right is first and then it moves towards that goal based on priority and also the steps taken to move towards the goal.



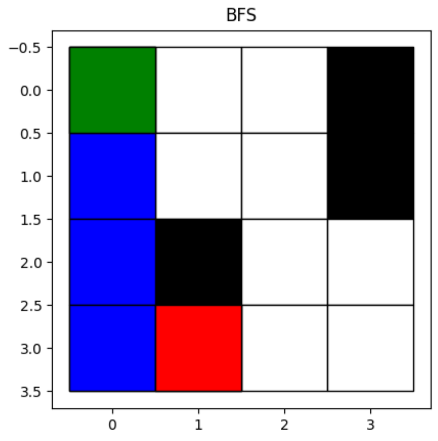
Here It moves similar to BFS as even the cost to visit node is 1. But in general the node is selected with the minimum cost to move. It works exactly same like BFS in exploring the neighbor nodes.

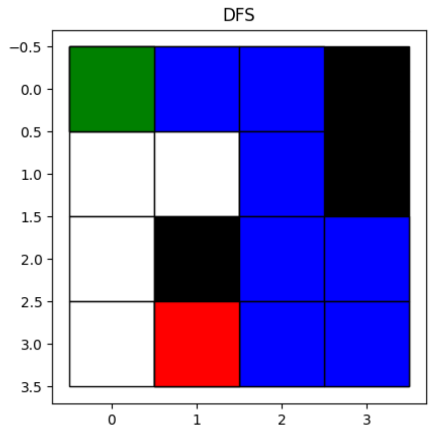


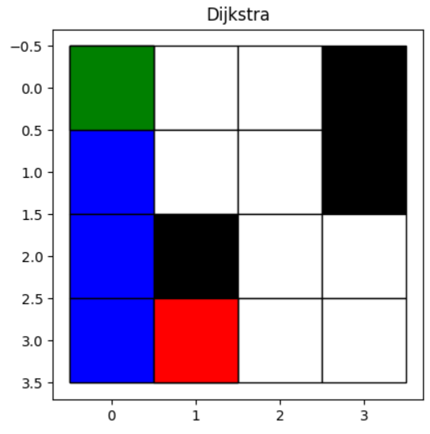
Here A* uses cost to come as well as Heuristic cost i.e Manhattan distance towards the goal is chosen. At every turn the Manhattan distance is less than the other direction to move from (2,0) (6,4). It chose the less cost node from the (2,0) and (6,4) also it will check for obstacles and boundary conditions.

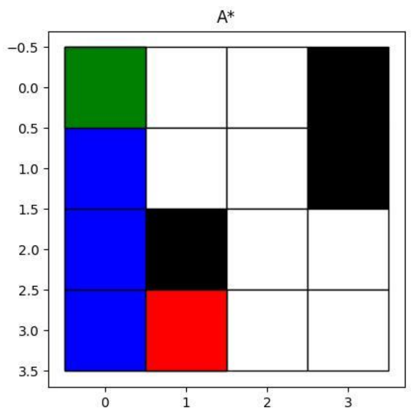
References.

- Geeks from geeks https://www.geeksforgeeks.org/breadth-first-traversal-bfs-on-a-2d-array/?ref=rp
- Professor Notes to understand pseudo code.
- https://github.com/GKPr0/Autonomous-Robots-Path-Planning to implement some functions like sorting.


BFS
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
1
2
3


DFS
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
1
2
3


Dijkstra
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
1
2
3


A*
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
1
2
3