Assignment 2

By: Chinmayee Prabhakar

Code and algorithm explanation

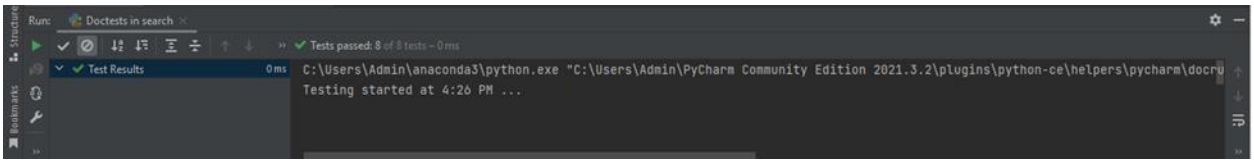
The algorithm takes two approaches Dijkstra and A* to find the goal node in a 2 dimensional map initialized with the Start and Goal as well as obstacle nodes.

We initially split the grid elements to two sets of nodes : Lists with nodes containing Obstacles and Lists with nodes that are Obstacle Free.

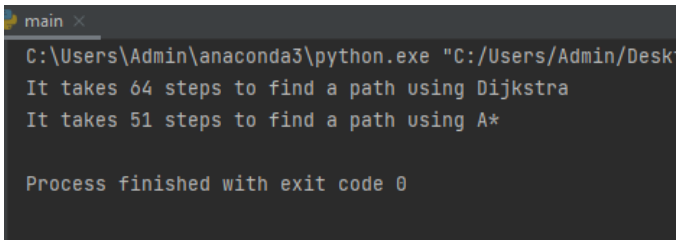
When it comes to the Dijkstra implementation, the algorithm starts from the source node. We have a 'Cost' dictionary that keeps track of the nodes as key and their cost-to-come as their value. The algorithm finds the adjacent nodes of source that are obstacle free and not already a part of the dictionary and adds these nodes to 'Cost' along with their cost-to-come value. As each adjacent node is found, their cost to come is evaluated at the same time and added to the 'Cost' dictionary. This way , starting from 'start' node we move forward by selecting the adjacent nodes with the least cost-to-come and at the same time updating their values if they are found to be sub-optimal iteratively until we reach the 'Goal' node. Upon reaching the 'Goal', the iteration going through each element in 'Cost' using the 'For Loop' is broken out of and the collected parent nodes from each node during the implementation is used for tracing the path from start to goal.

Similarly, when implementing A* we start with the 'Source' node and compute its adjacent nodes out of which only one that satisfies two conditions i.e. is not an obstacle node and is not already present in 'Cost' dictionary is selected. We select the successive node based on the sum of their cost-to-come and heuristic value stored in the 'H' dictionary. The node having the least sum stored in 'Q' is selected and this cycle is continuously repeated until we reach the 'Goal'. Then the parent nodes are traced and the path from start to goal is calculated.

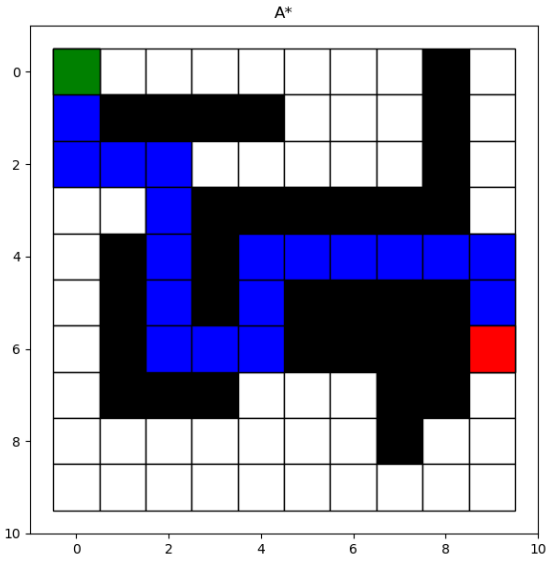
1. For the given test case (map.csv) in the assignment the results for Dijkstra and A* are as follows:

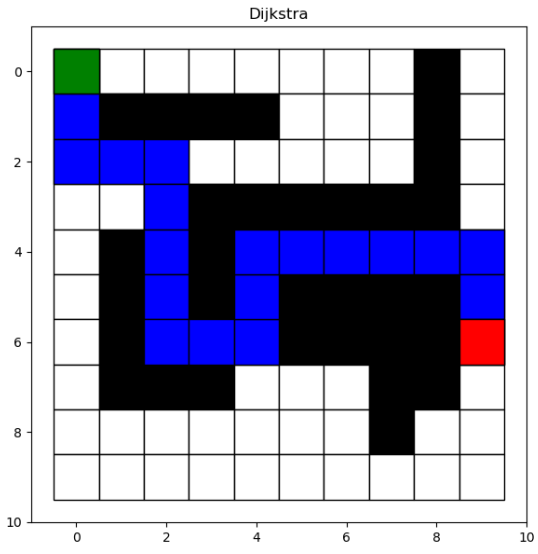


The Doctests



No. of Steps taken to find path.





The results are very different as the approaches are different. In Dijkstra as expected the implementation takes longer or more number of steps to reach the goal whereas in A* the number of steps are less. But both are optimal in finding the shortest path for the given obstacle map as can be seen in the images above.