Spike: Task 17

Title: Graphs, Paths, and Searches

Author: Sam Huffer, 101633177

Instructions

- In the box_world.py module there is a variable named min_edge_cost used by the A* algorithm. It is currently set to the wrong value! Change this value and demonstrate (find a case) where the behaviour between the old value (which can give non-optimal results) and your new value (which should give optimal results) are different. The console will show the path cost to assist in your comparisons.
- Currently the navigation graph only uses N-S-E-W edges between boxes. A* is also currently using Manhattan distance to estimate cost. Find the code in the reset_navgraph method of the BoxWorld class (in the box_world.py module) and uncomment the code needed to add diagonal edges.
- Is the use of Manhattan distance calculation correct now? Change this to another method by editing the reset_navgraph method of the BoxWorld class. (The correct code is currently commented out.)

Min Edge Cost

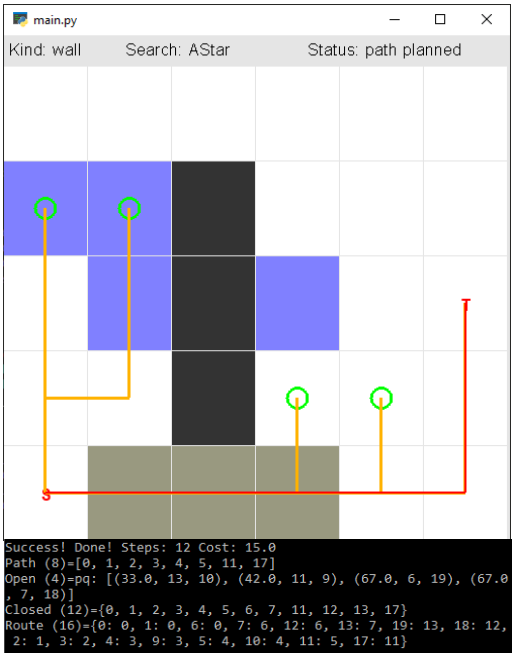


Figure 1: original result on an unaltered map 1 with min_edge_cost set to 10. Seems simple enough.

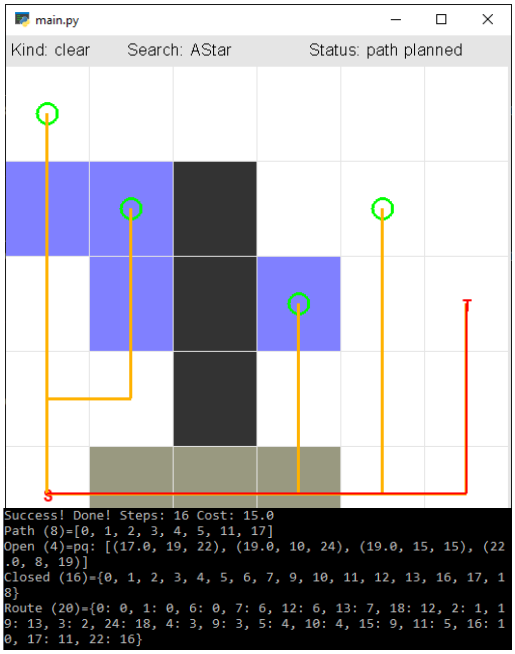


Figure 2: new result with min_edge_cost set to 1. Already here one can see that the A algorithm is now trying more options to find the best path.*

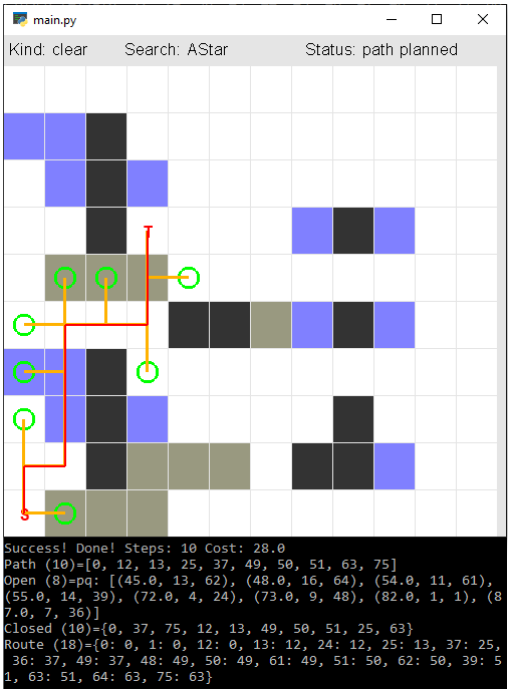


Figure 3: original result on an unaltered map 2 with min_edge_cost set to 10. Here you can see that it is not taking an optimal path, instead going across 2 water blocks when it could save itself effort by crossing only one.

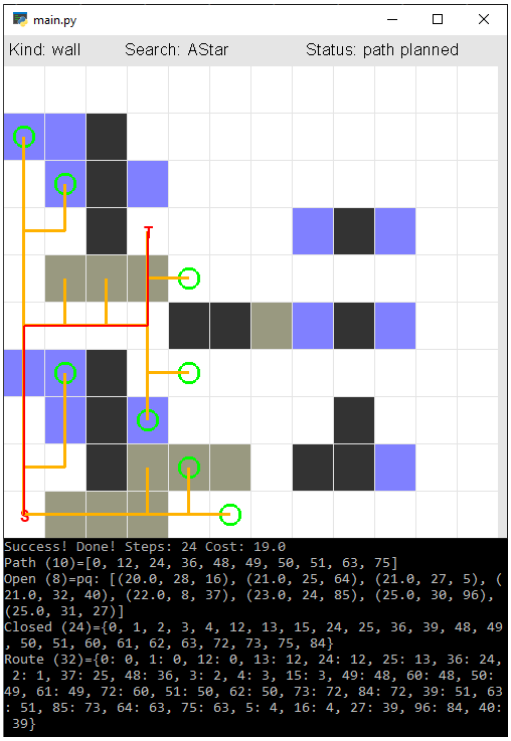


Figure 4: new result with min_edge_cost set to 1. With the new value, the A algorithm has found the optimal path to the target. The size of the tree shows that this time, the algorithm experimented more, which led it to a better solution than it originally found.*

Diagonal Edges

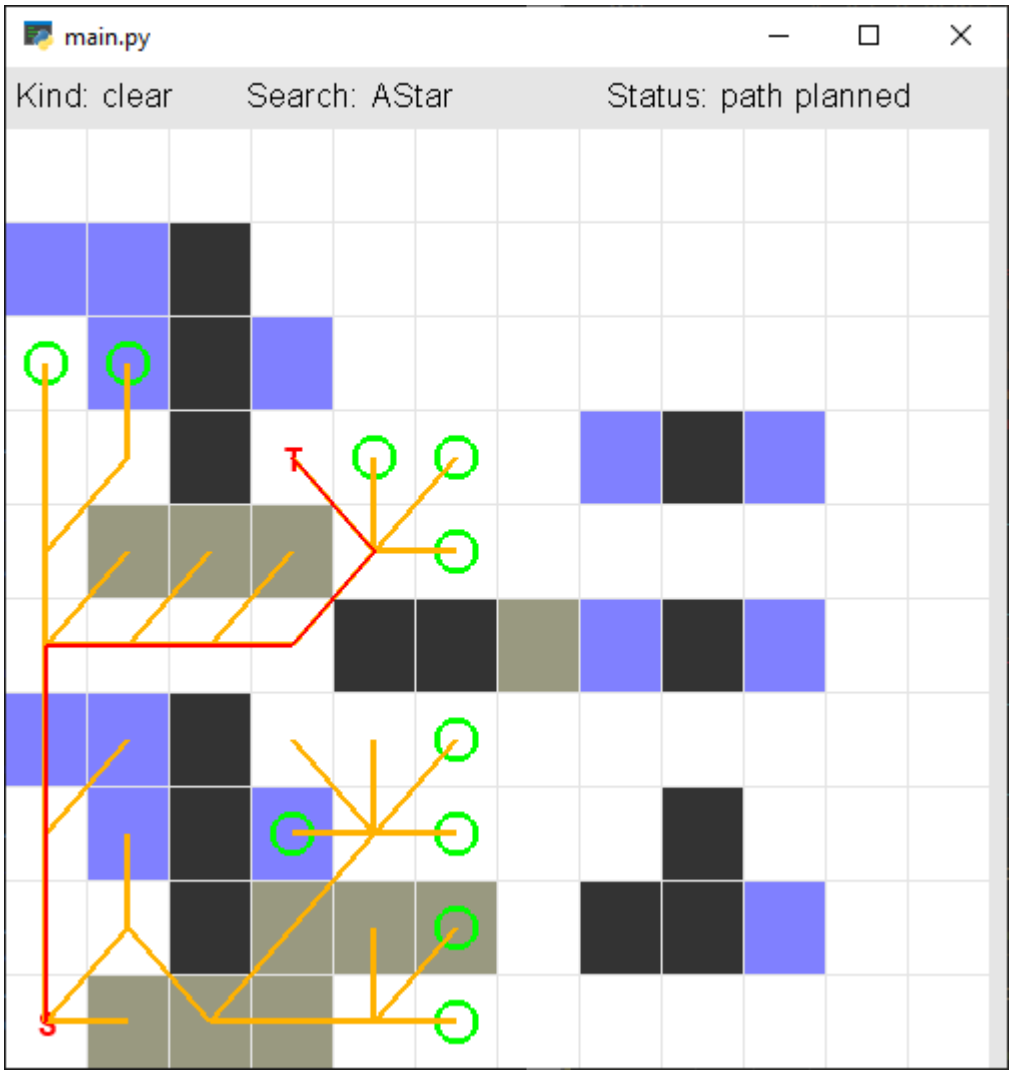


Figure 5: the results of the diagonal edges code being uncommented. There was a line that prints stuff to the terminal, I do have to wonder why that was there.

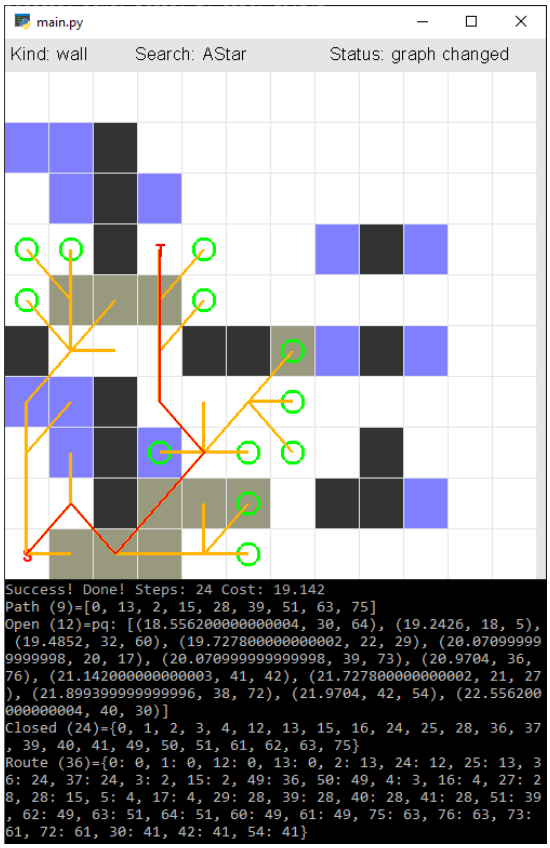
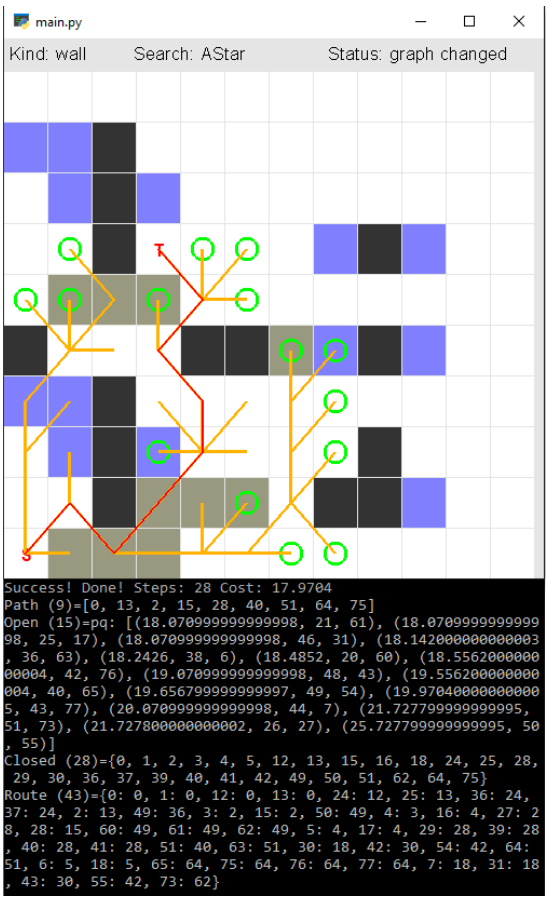


Figure 6: the output on map 2 using the Manhattan distance. The map has been changed to include an extra wall in the centre-west to obstruct the path.



Manhattan Distance Calculation

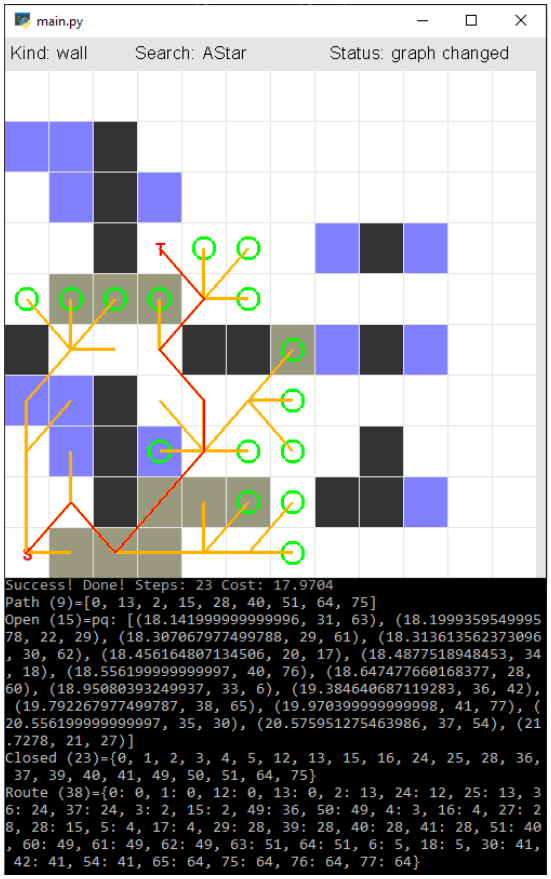


Figure 7: the output using the hypotenuse. Here the algorithm finds a shorter distance by going around some mud rather than through it. As such, the hypotenuse is a better measure of diagonal distances than the Manhattan distance. The results of the algorithm also differ here in that the algorithm hasn't searched as far north in the centre-west, but has gone further east towards the south.

Figure 8: the output using the maximum distance ("the straight-line distance between two points on a 2-D Cartesian plane", a description it shares in the code's comments with the hypotenuse). Its search pattern extends further east and one cell further north, but comes to the same conclusion as the hypotenuse and therefore is also more suitable than the Manhattan distance. Given that the better method in the first section on min_edge_cost explored further, it would seem to me that this method might have an edge over the hypotenuse as well as over the Manhattan distance.