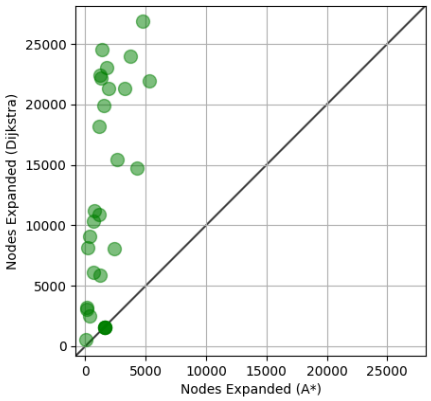
Question 1

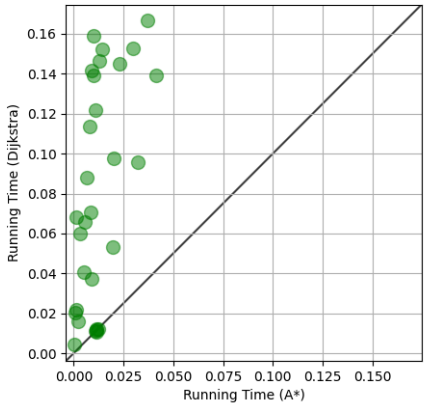
a) Why is the overall distribution of points in the plot the way it is? That is, what is the overall story the distribution of points tells?

Diverging from the diagonal line which shows that both A* and Dijkstra expand the same number of nodes in the first plot, indicating they have equal performance, the points of distribution show that A* is consistently efficient than Dijkstra in terms of the maps for testing.

b) Explain the difference in the distribution of points between the expansions plot and the running time plot. In particular, what causes the shift in distribution?

In the running-time plot, it compares running time for the pair of two algorithms on the same instances. While the general trend mirrors the node-expansion plot, which still shows that Dijkstra tends to take longer, many points lie closer to the diagonal. Additionally, pre-evaluation for heuristic function in A* for per node guides the expansion, saving the time spent searching and expanding in terms of the Dijkstra. Therefore, The gap is narrower than the difference in node counts.

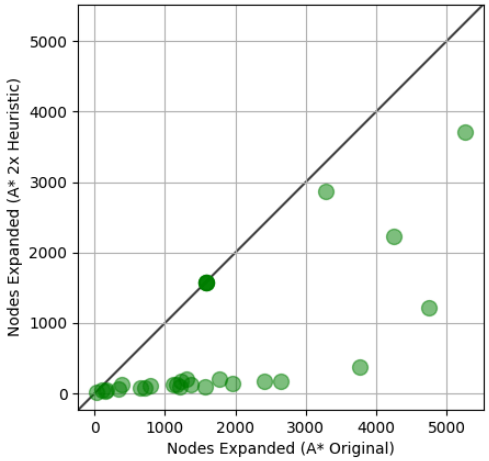




Question 2

What happens when you multiply the heuristic value for A* by 2? Explain what you observe. It can be helpful to plot a scatter plot of the number of expansions for A* as you had originally implemented (in one axis) and A* with the heuristic multiplied by 2 (in the other axis). Note that if you decide to generate this plot, you need to modify main.py.

It is observed that the points lie below the diagonal. This reflects the general trend of weighted heuristic search, which expands fewer nodes. Points where the weighted version lies noticeably below the diagonal correspond to cases where the inflated heuristic made the search much greedier.



Question 3

What happens when you change your Dijkstra's and A* implementations so that the search does not update the cost and parent pointer to a node n once it finds a better path to *n*? Please discuss what you observe in terms of solution cost and solution path.

When removing the statements related to updating the cost and parent pointer, the first path found to some node will be locked in since the node's g-value, stored cost, and parent pointer will never be improved due to the lack of value update. As the search continues, it still stumbles upon the goal, and the recorded information corresponds to the first path.

This will lead to suboptimal solution cost, which is higher than the true optimal path that should be found by the correct whichever Dijkstra or A*. This violates the optimality guarantees that both algorithms provide when they are able to relax edges a couple of times.

In consequence, suboptimal solution cost brings an inaccurate solution path. There may exist the case where the OPEN list eventually includes a cheaper state, the parent pointer that would reconstruct the path remains the original one.