Project 1 Part 1

Joshua Westbrook & Arifur Rahman

1 The maps

The map files are available at `https://github.com/Joshua-Westbrook/AIproject1maps`

They are also in the folder that opens after unzipping the .zip file attached and importing it into Eclipse as an existing project.

2 The algorithm

The algorimths can be found in in the searches package. The UCS algorithm is fully implemented in SearchAlgo.java but uses UCS.java so it has a specific name. A* and Weighted A* overwrite the h(n) being zero in UCS with them using either the heuristic from DistanceHeuristic.java or the heuristic times the weight. UCS and A* always return the cheapest path while Weighted A* may or may not depending on the weight and the specific map with its start and goal point. In some situations such as map5j even a weight of 3.0 still has Weighted A* return the right path while other maps such as map5b return a longer path with a weight of 2.

3 Optimizations

One trade-off made in the algorithm is the use of a 2d array to store every node once it is added to the fringe. This allows for much quicker look up times of a node already in the fringe when checking if the new path found to that node is more optimal faster but also increases the memory requirements of the program because it will always require space to store all the nodes on the map. Due to the nodes not requiring much memory to store this is a good trade-off. In other situations the memory cost of allocating the 2d array might be worse than the time cost of iterating through the fringe, which is Java's priority queue, to retrieve the previously found hcost of reaching that node.

4 Heuristics

The Heuristic we finally used was a modified Euclidean distance . We then multiply the Euclidean distance between the currNode and goal by .25 since in a best case scenario the entire path to the goal is an highway/river with no hard to travel in which case

all movements is the distance divided by 4 which is the same thing as times 0.25 The formula for the heuristic is $0.25 * \sqrt{(currNode.x - end.x)^2 + (currNode.y - end.y)^2}$.

Other attempted heuristics were the following.

1. Euclidean distance from currNode to goal. Not admissable because it is not optimistic.
2. ???
3. ???
4. ???

5 benchmarks

Table 1: The average across all 50 maps

UCS nodes	12170.16
UCS time	26.88
UCS cost	112.19421
A* nodes	9427.98
A* time	15.56
A* cost	112.19421
2 weight A* nodes	6539.04
2 weight A* time	8.22
2 weight A* cost	112.85156
3 weight A* nodes	3644.6
3 weight A* time	2.8
3 weight A* cost	116.00762

The excel sheet attached contains the cost for each map with its individual start and end points in tables. They are not fitting nicely in the latex sheet unfortunately.

6 Final Explanation

The results clearly show that when using an admissable heuristic while UCS and A* find the same path A* is able to do it with much less nodes expanding and a much quicker speed. Adding a weight increases the speed and decreasing the nodes expanded even more but also results in a slightly higher cost found on average. This makes it clear that using A* when you have an admissable heuristic is always better than using UCS and UCS should only be used in situations where a heuristic isn't available. It also shows that weighted A* can be used to increase the speed of the algorithm and reduce the nodes expanding but that will also reduce the ability to get an optimal path. Many maps, such as 5a and 1j, see a pure increase in performance when using a weight of 2 which says that it is often worth using at least a small weight to increase your performance because the path found is very close to optimal at least on the maps we have. This is also shown

by the average cost of A* with a weight of 2 only being 0.76 higher than the average optimal path from UCS/A* which is less than the cost of moving between two normal nodes. Meanwhile the average cost for A* with a weight of 3 is around 4 higher than the optimal path which makes it very clear that using larger weights will almost always cause a noticable increase in the cost of the found path.