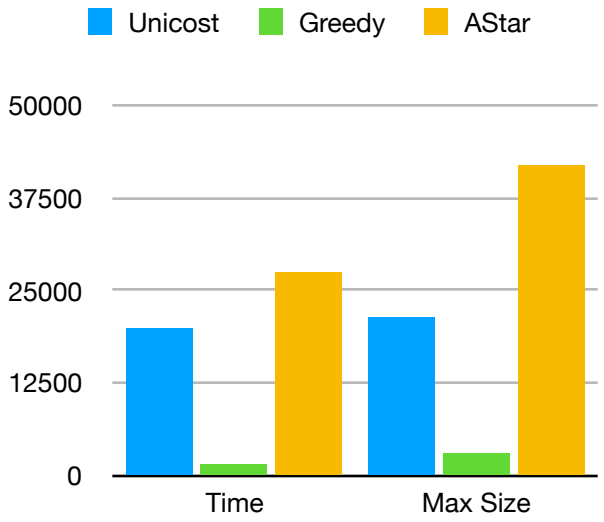
Christopher Grant
CS 1571 Homework 1 Writeup

Note: Please see data.pdf in the submission to see the exact numbers generated using each algorithm for each problem

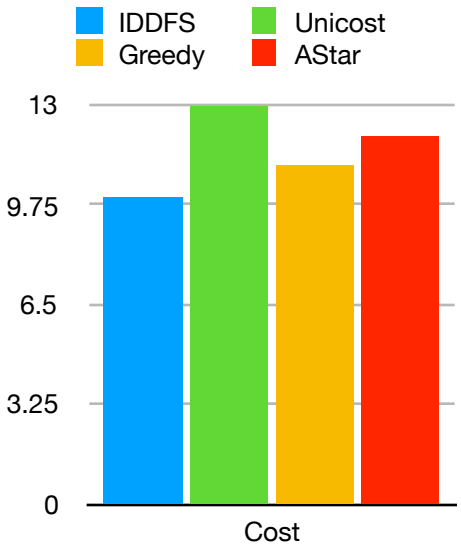
Note: For problems that produced “No Results” in data.pdf, their output is not included in the output file for that problem, as it produced no results due to timing out.

Problem 1:

Looking at the two example monitor examples, the results show how this problem can balloon in size very quickly when using a more exact search algorithm like unicost, greedy or astar. This chart shows how unicost and astar are much more costly searches, while greedy is relatively cheap.



However, looking at the chart that shows the overall time that all targets will remain monitored, shows how IDDFS produces a close result for very quick search time (doesn’t even appear on the above chart)



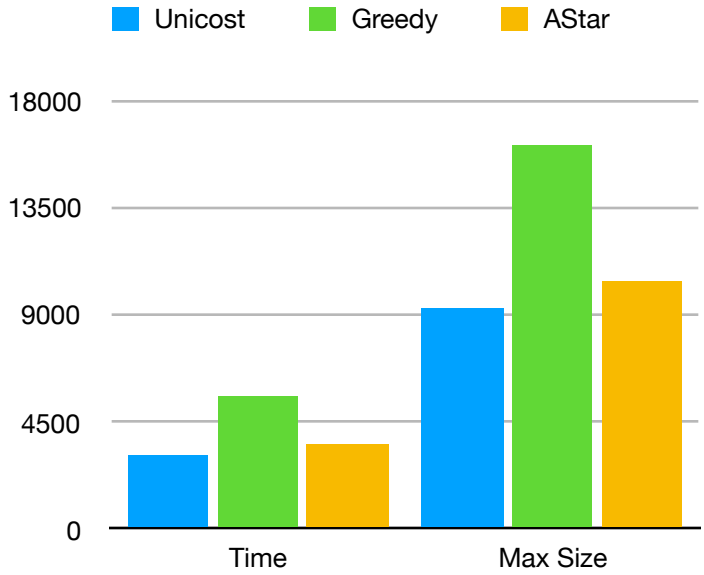
For the heuristic function for problem 1, I chose to use euclidian distance, as in general most sensors would last longer for targets that are closer. This isn't necessarily always true, and sometimes a better choice can be made, but that accuracy is sacrificed for speed in greedy search. In astar search, this isn't ideal because it combines euclidian distance and battery / distance, which produces results that are neither ideal nor quick.

In conclusion, a better heuristic function could improve search results to produce more optimal result quicker. However with the current heuristic, either IDDFS, unicost or greedy should be used based on the needs of the problem (fast results, complete results, or somewhere in the middle)

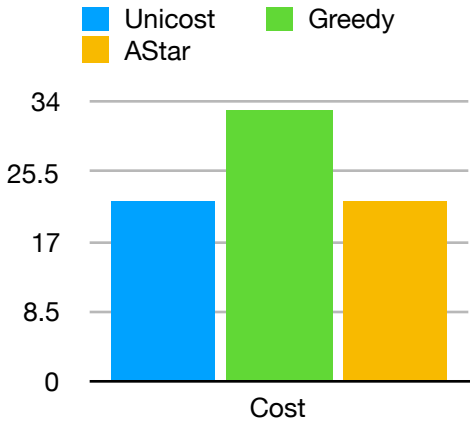
Problem 2:

Looking at the 3 problems provided for the aggregation problem, I'll be primarily addressing the 3rd example, as it provides a good mix between speed and solvability.

Looking at the more optimal algorithms, we see that unicost and astar have similar results, with unicost being slightly more ideal, however greedy is much more costly than unicost or astar in terms of both search time and max frontier size.



Looking at the total cost of the path generated by the algorithms, we continue to see this trend that unicost and astar produce better results than greedy.



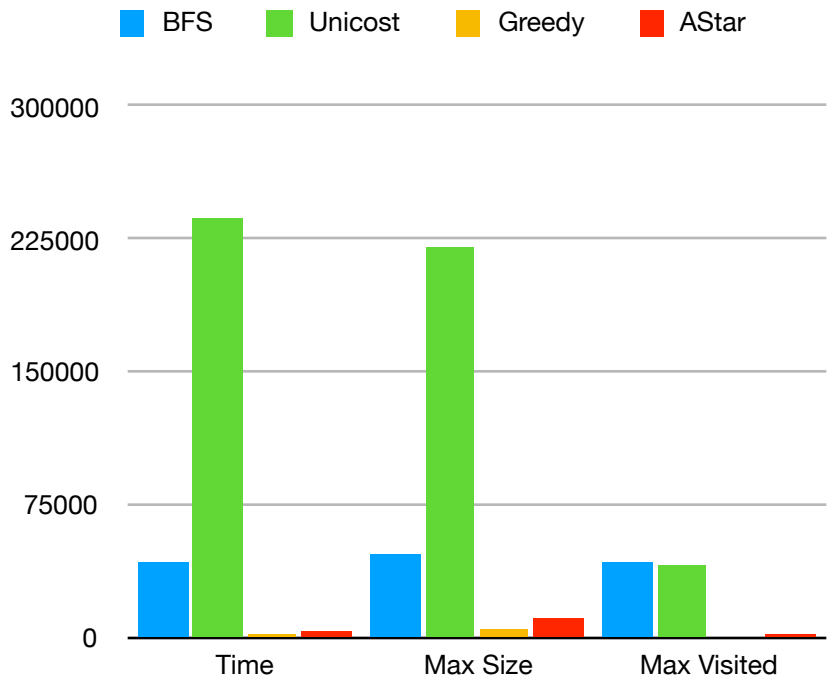
For the heuristic, I chose to use euclidian distance, as that will generally navigate from one node to the next node with the minimum cost. This is generally more effective when going from a specific node to a specific node rather than visiting all nodes, but still an effective strategy.

In conclusion, unicost should be used when looking for an exact result, as it produces smaller a smaller footprint than astar but produces an equally optimal result. However, if you are looking for a quick solution, one of the non optimal algorithms should be used, either IDDFS or BFS. These algorithms will produce a solution for a large problem much quicker than unicost, greedy or astar for a smaller problem.

Problem 3:

Looking at problem 3, using the original 2 provided samples, I found myself frustrated because only greedy and astar found a result for pancakes1, with no results found for any of the algorithms for pancakes2. However, with Pancakes3, I was able to find results for 4 out of the 5 algorithms, so I will be using the data for that sample for the writeup.

Looking at the results in the chart, we can see that Greedy and astar are the only reasonable results for the problem. This is due to the heuristic that was chosen. I did some research and found a heuristic that will find a quicker solution than using bfs or unicost. The paper that goes into detail for the heuristic that I implemented can be found here.



In conclusion, for solving this problem, greedy is the only choice. It provides the exact result quicker than any other of the algorithms, however the problem still grows very quickly, so it is still difficult to solve large problems.