Noah Miles

CS4200-01

Project 1 Report

02/23/20

Approach:

My A* Search Strategy's parameters were the problem, heuristic, and a boolean flag. The problem contained a goal state, a method to test if the current state was the goal state, a method to generate a random initial state, a method to test if the initial state was solvable, and a method to generate children states of the current state. The heuristics were passed into the search strategy because they were not included in the problem statement. The boolean flag determines whether or not it was to be a tree search (explored set disabled) or a graph search (explored set enabled).

Comparisons:

Throughout my testing, I found the graph search with Manhattan heuristic was the most efficient method of finding the goal state for most initial cases. Occasionally, there were some cases where the tree search found the goal state faster. For either case, the Manhattan heuristic outperformed the Hamming Heuristic

Analysis:

Charts: (See outputForReport.txt)

	Search Cost (Tree-Search)		Time (ms)	
Depth	A* (using h1)	A* (using h2)	A* (using h1)	A* (using h2)
12	1116	268	14.3493	2.2763
14	20007	880	32.6182	1.0968
16	178273	12601	188.5799	12.8591

	Search Cost (Graph-Search)		Time (ms)	
Depth	A* (using h1)	A* (using h2)	A* (using h1)	A* (using h2)
12	1116	268	3.7617	0.7064
14	20007	880	26.2133	0.9446
16	178273	12601	159.2236	9.3296

Findings:

Throughout this project, I found out how quickly the branching factor increases as the depth of the search increases. The time necessary to perform searches on depth greater than 20 initial states was exponentially larger than those of less depth. Also, I ran into memory issues due to the exponential branching of the tree and the need to keep many items in memory when on the frontier. It forced me to revisit my code and optimize otherwise I would run out of memory when trying to solve a deep initial state.