Assignment 1 Analysis

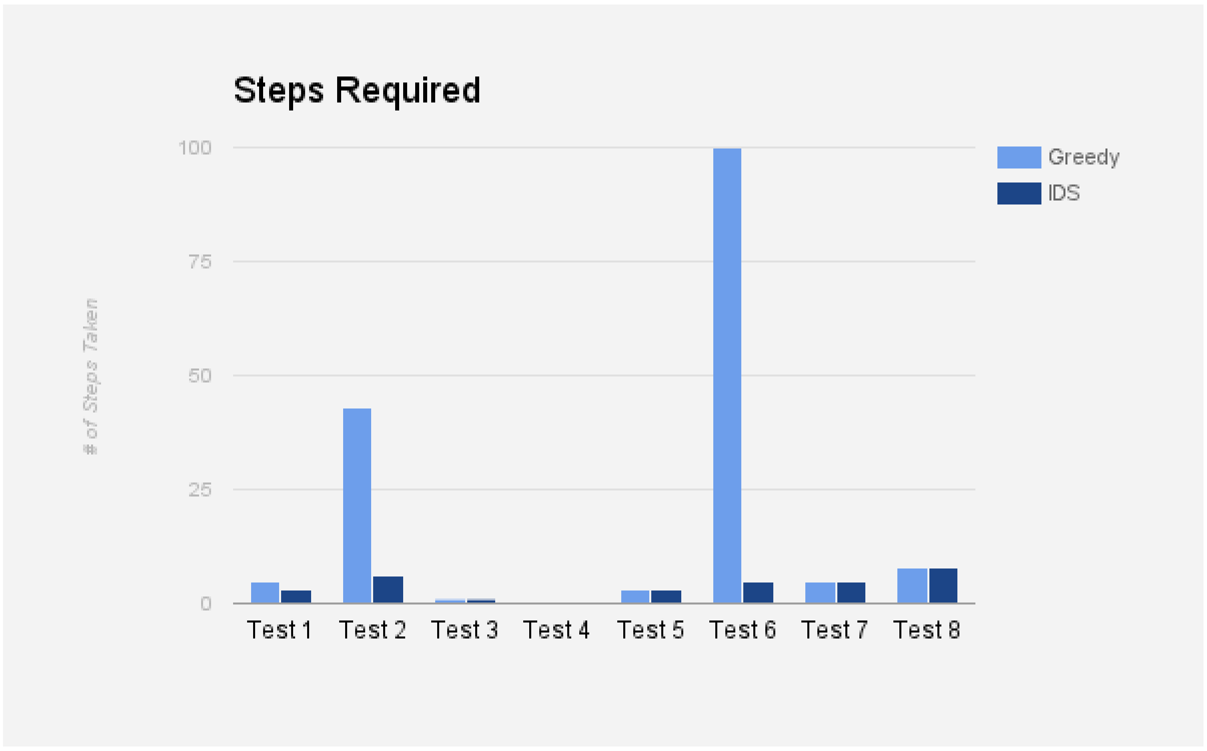
Charlie Lovering, Jonas McGowan-Martin, Anqi Lu, James Honicker

Comparison Of Greedy Search and Iterative Deepening Search

Test Results:

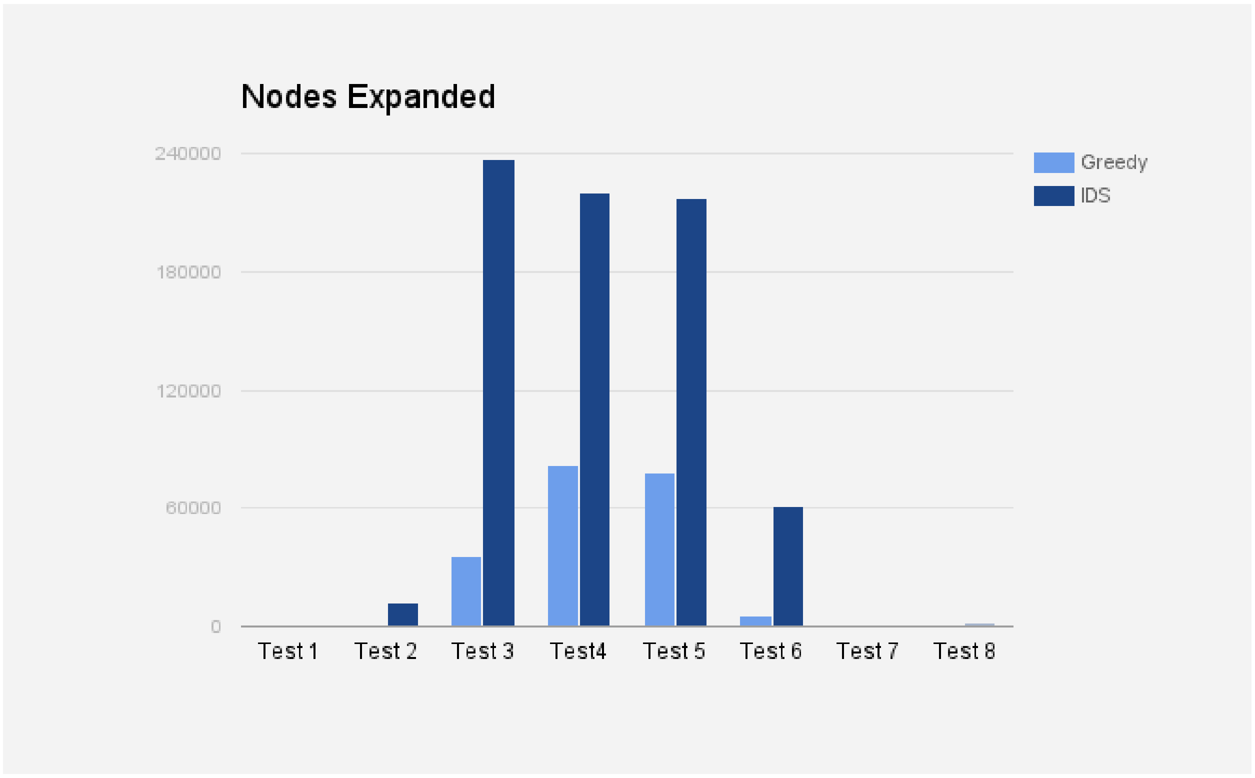
Test #	Algorithm	Error	Result	Steps	Time (sec)	Nodes Expanded	Max Depth	Branching Factor
1	Greedy	0	11	5	0.00335	45	5	5
1	IDS	0	11	3	0.00397	137	4	5
2	Greedy	0	1100	43	0.00717	215	43	5
2	IDS	0	1100	6	0.14207	12438	7	5
3	Greedy	3	8	1	~2	36002	36002	1
3	IDS	3	8	1	~2	237016	688	1
4	Greedy	1	2	0	~2	81620	40810	2
4	IDS	1	2	0	~2	219811	17	2
5	Greedy	1	12	3	~2	77982	1521	2
5	IDS	1	12	3	~2	216983	17	2
6	Greedy	0	1000	100	0.14218	6110	109	10
6	IDS	956	44	5	~1	61273	6	10
7	Greedy	0	10	5	0.00040	10	5	2
7	IDS	0	10	5	0.00075	63	6	2
8	Greedy	0	10	5	0.00030	10	5	2
8	IDS	0	10	8	0.01463	1013	9	2

Steps Required:



As shown in the graph, the number of steps taken by greedy search in order to find the goal is usually equal to or greater than the number of steps taken by IDS. In some cases, such as tests 2 and 6, greedy search can take many more steps to reach the goal than IDS due to its over-optimistic approach. This is because greedy will overshoot its goal, and end up taking far more steps than necessary. This occurs because since it does not take into account the number of steps it took (at any given decision), and thus can take a suboptimal path. Iterative is complete and optimal, given enough time, so greedy can never take less steps. This holds true if iterative fails to find the goal and returns its current best node, because that node is necessarily less ‘deep’ than the actual answer.

Nodes Expanded (and Memory Considerations):



As can be seen above, the IDS search requires many times more nodes to be expanded in every test as it looks through all possible solutions. However, IDS is very space efficient because it does not require a large amount of memory at any given time. Since it is built on depth first search it is O(d); it recursively runs through a single branch at a time. It will always find the shortest path to the goal given enough time and memory because it applies the depth first search iteratively. Greedy creates less nodes, but it will often go much deeper down a branch. Not only will this lead it to sometimes find a suboptimal path, but it will lead to it requiring more memory at a given time. That being said, it is generally faster because it uses its heuristic to attempt to follow a more ‘informed’ path. Since it has to go through less nodes it takes less time, but may not end up with the optimal solution.

Effective Branching Factor:

Evaluations	Type	Nodes Expanded	Depth to Goal	Effective Branching Factor
Evaluation 1	IDS	219811	17	2
Evaluation 2	Greedy	215	43	1.02
Evaluation 3	Greedy	10	5	1.2
Evaluation 4	IDS	63	6	1.74
Evaluation 5	Greedy	45	5	1.8
Evaluation 6	IDS	1013	9	2
Evaluation 7	IDS	45	4	3.11
Evaluation 8	IDS	12438	7	3.68

Average EBF for Greedy = 1.3:
Optimal case = 1.02
Worst case = 1.8

Average EBF for IDS = 2.5
Optimal case = 1.74
Worst case = 3.68

Effective Branching Factor (EBF) is a good metric for heuristic is performance. By using the formula $N + 1 = 1 + b^* + (b^*)^2 + \dots + (b^*)^d$, where N is total nodes expanded and d is depth of search to goal the EBF, b*, can be calculated. A good heuristic will approach an EBF of 1 and as the searches get larger become more consistent. In our best case tested an EBF of 1.02 was achieved for a very deep search that led to a viable solution. For that same problem an IDS search was run and an EBF of nearly 3.7 was achieved. The branching factor for this problem was five, meaning the greedy search reduced its average branching down to nearly 1 while the IDS could only pull it down to 3.8. This demonstrate the strength of the heuristic and how a greedy search is especially strong at rapidly descending down a tree to find a viable solution.