Brian Legarth
Classical Planning Project Results

During this project, I was given many algorithms to write and I had to analylize the outcomes for each of them. The algorithms truly gave me a better understanding of planning motives and search algorithms.

Number of Nodes Expanded against the Number of Actions in the Domain								
Problem 1		Problem 2		Problem 3		Problem 4		
Actions	Nodes Expanded	Actions	Nodes Expanded	Actions	Nodes Expanded	Actions	Nodes Expanded	
20	43	72	3343	88	14663	104	99736	breadth first
20	21	72	624					depth first
20	60	72	5154					uniform cost
20	7	72	17	88	25	104	29	GBFGS unmet
20	6	72	9	88	14	104	17	GBFGS levelsum
20	6	72	27					GBFGS maxlevel
20	6	72	9					GBFGS setlevel
20	50	72	2467	88	7388	104	34330	astar unmet
20	28	72	357	88	369	104	1208	astar levelsum
20	43	72	2887					astar maxlevel
20	33	72	1037					astar setlevel

As the problem increases in size, the growth of the nodes expanding increases exponential for most algorithms. The most obvious exponential growers would be breadth first, uniform cost, astar unmet goals, and astar max level. Some of the algorithms grow in an almost linear rate which makes them able to scale when given a larger problem. These include all of the Greedy Best First algorithms.

Search Time against the Number of Actions in the Domain								
Problem 1		Problem 2		Problem 3		Problem 4		
Length	Time	Length	Time	Length	Time	Length	Time	
	0.006310		1.785565		10.18673		92.75843	
6	32	9	19	12	52	14	63	breadth first
	0.003068		2.735424					
20	55	619	45					depth first
	0.009445		3.195911					
6	57	9	21					uniform cost
	0.001591		0.019774		0.035883		0.058461	GBFGS
6	56	9	45	15	17	18	46	unmet
	2.506232		18.48995		40.89701		77.68940	GBFGS
6	42	9	67	14	07	17	66	levelsum
	1.876601		35.92970					GBFGS
6	32	9	92					maxlevel
	5.657055		39.67598					GBFGS
6	84	9	53					setlevel
	0.009712		2.039877		8.217968		54.38713	
6	56	9	17	12	32	14	49	astar unmet
	6.610096		448.2436		729.6814		4172.601	astar
6	12	9	9	12	82	15	1	levelsum
	6.982466		2726.135					astar
6	09	9	39					maxlevel
	14.28532		3109.794					
6	24	9	53					astar setlevel

When comparing the actions to the search time, the most important distinction to make is which algorithms would be able to run in real time allowing for the quickest results. From my tests the only algorithm that does very well is the Greedy Best First Unmet Goals Algorithm. This one keeps the length almost optimal while still making the time nearly instantaneous. All the other algorithms begin taking a short about of time but quickly increase past the limit that we want them to with some taking over an hour.

Optimality of the Solution								
Problem 1		Problem 2		Problem 3		Problem 4		
Length	O/N	Length	O/N	Length	O/N	Length	O/N	
6	O	9	O	12	O	14	O	breadth first
20	N	619	N					depth first
6	O	9	O					uniform cost
6	O	9	O	15	N	18	N	GBFGS unmet
6	O	9	O	14	N	17	N	GBFGS levelsum
6	O	9	O					GBFGS maxlevel
6	O	9	O					GBFGS setlevel
6	O	9	O	12	O	14	O	astar unmet
6	O	9	O	12	O	15	N	astar levelsum
6	O	9	O					astar maxlevel
6	O	9	O					astar setlevel

When trying to pick the best algorithms, certain situations must be taken into account. For example, the algorithm that works best with a small domain may not work well with a large domain. If the algorithm must be able to find an optimal solution it will not technically be the fastest algorithm as well. Taking these into account I have decided that when the best algorithm for a restricted domain and one that can run in real time, the Greedy Best First Unmet Goals algorithm should be used. This was my fastest algorithm; others that are comparable in speeds would be the breadth first, depth first, uniform cost, and astar unmet goals. When looking for an algorithm that can work with very large domains, the two most important things are low time and a low number of expanded nodes. I found that the best algorithms were the Greedy Best First algorithms. If the optimal plans were needed to be found every time, I found that the astar unmet goals and the breadth first search tended to be the best options.