Task results

Blank indicating which takes over 10 minutes to execute

All tasks are run in parallel with *run_search_bulk.py*

	Problem 1			Problem 2			Problem 3		
	Expansions	Goal tests	New nodes	Expansions	Goal tests	New nodes	Expansions	Goal tests	New Nodes
BFS	43	56	180	3608	5168	32851	14120	17673	124926
BFTS	1458	1459	5960						
DFS(graph)	12	13	48	808	809	8404	677	678	5608
DFS(limited)	101	271	414						
Uniform	55	57	224	5628	5630	50677			
Recursive best first h1	4229	4230	17029						
Greedy best first graph h1	7	9	28	549	551	4493	5578	5580	49150
A* h1	55	57	224	5628	5630	50677			
A* Ignored preconds	41	43	170	1567	1569	13930	5118	5120	45650
A* Level sum	11	13	50	136	138	1091	414	416	3818

	Problem 1		Problem 2		Problem 3	
	Path	Time	Path	Time	Path	Time
BFS	6	0.0135s	9	26.62s	12	314.18s
BFTS	6	0.70s				
DFS(graph)	12	0.075s	774	7.78s	660	14.58s
DFS(limited)	50	0.037s				
Uniform	6	0.020s	9	141.70s		
Recursive Best first h1	6	2.26s				
Greedy Best first h1	6	0.0052s	20	3.68s	22	356.44s
A* h1	6	0.029s	9	150.00s		
A* Ignored Preconds	6	0.027s	9	26.01s	12	302.54s
A* Level sum	6	0.80s	9	65.38s	12	392.55s

Optimal plan:
According to the statistics, greedy best first search with h_1 heuristic is the most optimal for Problem 1, because it finds the correct path with minimal time and node exploration, due to its poorer performance later in Problem 2 and 3, the optimal path is probably due to chance or problem setup. For Problem 2, if more weights are put onto finding the correct path as well as solving within less amount of time, A* search with ignored preconditions is the most optimal as it finds the correct path while maintaining a reasonably fast speed. As for problem 3, A* search with ignored preconditions is also the most optimal based on similar criteria. As a side mention, BFS also achieved really good results in the end, but due to its memory issue with the exponential growing number of nodes visited, it is not optimal.

Comparisons of non-heuristic search results:
In this context, I will choose BFS, DFS(graph) and Uniform cost search for comparison. Among these 3 search methods, BFS always guarantees to get the goal with optimality as well as reasonable time complexity, however, like mentioned, BFS works by keeping storing each node in new branches, thus when branching factor is high, the memory footprint could be an issue when facing more complexed problems. DFS(graph)

on the other hand, does not guarantee shortest path or optimal result, but it works really fast as it stops after it searches through the end of one branch, a.k.a. found a path, but obvious it is dissatisfying in this problem, there would be so much waste in effort if 660 actions have to be taken instead of 12. Uniform cost search, on the other hand, seems to expand more nodes even that BFS search, probably due to its non-directional nature, and it is much slower, but guarantees to find the most optimal path, if time allowed.

Comparisons between A* ignored conditions and A* Level sum:
Both heuristics have its own advantages, with level sum, since it must build up a plan graph in real time as the search goes, it costs much more time, as much as ~40 seconds for Problem 2 and ~90 seconds for Problem 3, but with its planning nature, it expands much fewer nodes and saves memory footprint, for example, in Problem 3, Ignore condition heuristic results in more that 10 times more nodes to be searched that that from level sum.

Heuristic or Non-Heuristic:
Clearly heuristic methods produce more consistently optimal results as both ignored precondition and level sum produce the best path for all problems, whereas only BFS succeeds in the same way. However, if memory consumption is considered, BFS is definitely a much worse choice than Heuristics. Overall, A* with ignored preconditions seems to be the best choice for this type of problems as it costs least amount of time of finding an optimal solution with reasonable memory footprint.