Classical Search Report

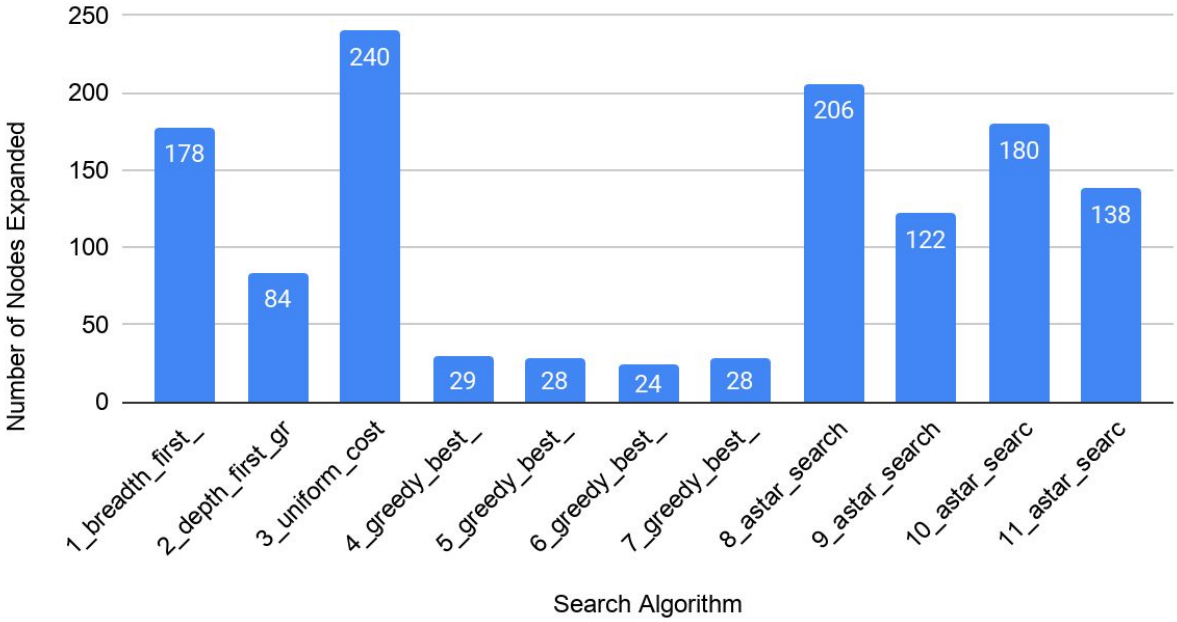
Author: Pei Zhao

1. Analyze the number of nodes expanded against the number of actions in the domain.

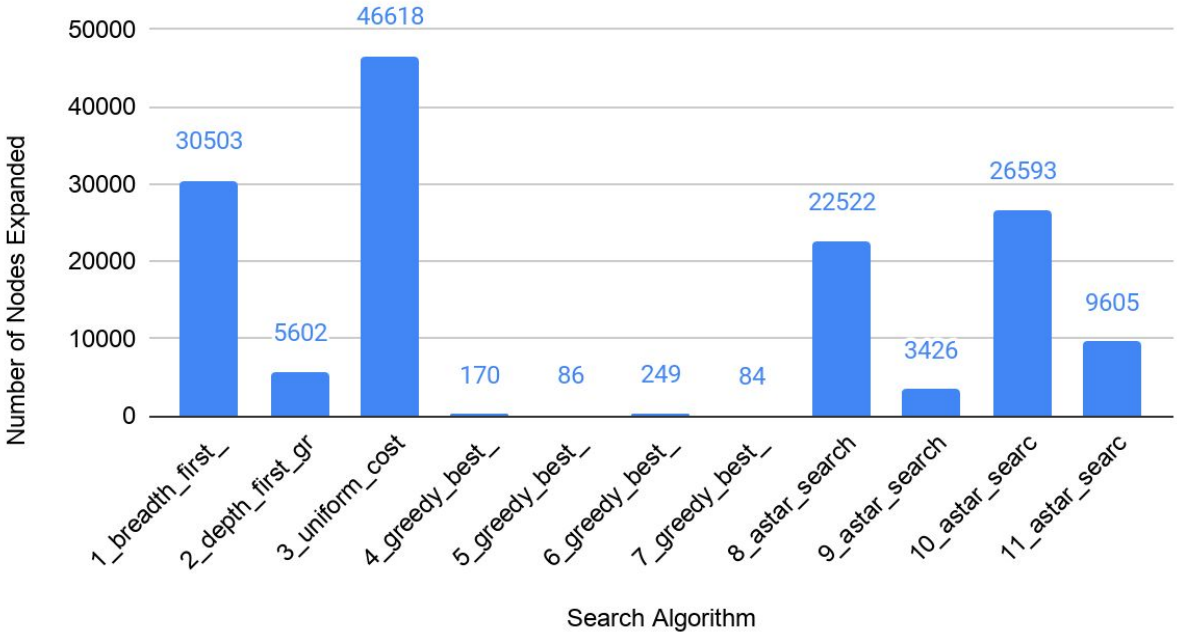
For uninformed search algorithms and heuristic A* algorithms, the number of nodes expanded sort of grows **exponentially** as the number of actions in the problem domain.

For heuristic greedy search algorithms, the number of nodes expanded sort of grows **linearly** as the number of actions in the problem domain.

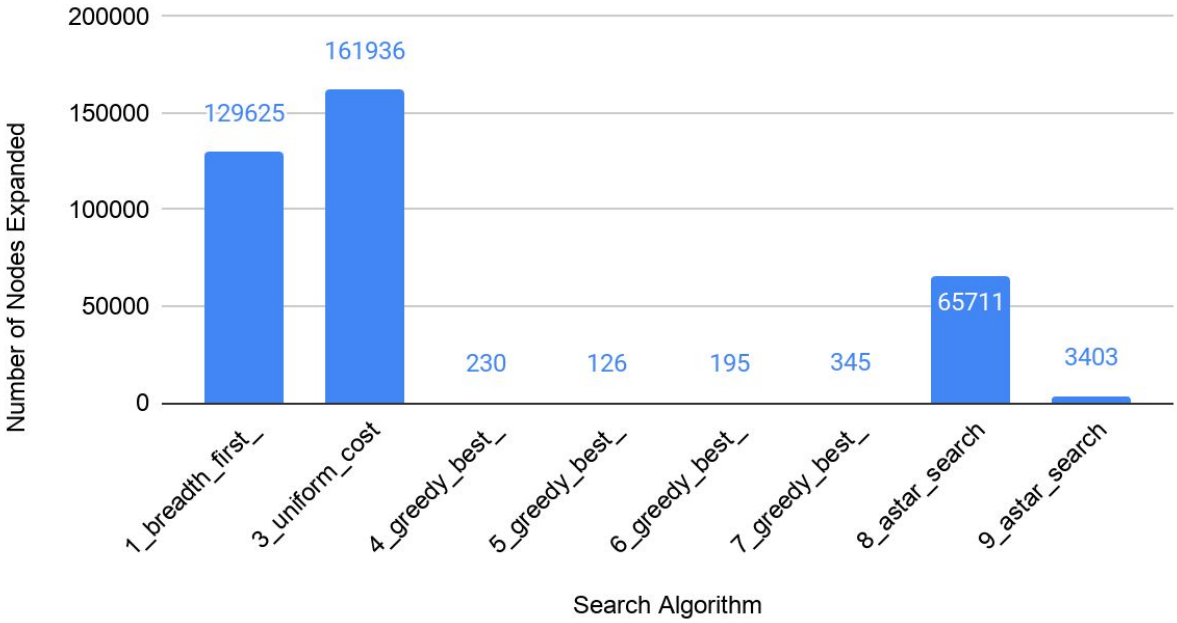
Problem 1 (20 actions) - Number of Nodes Expanded



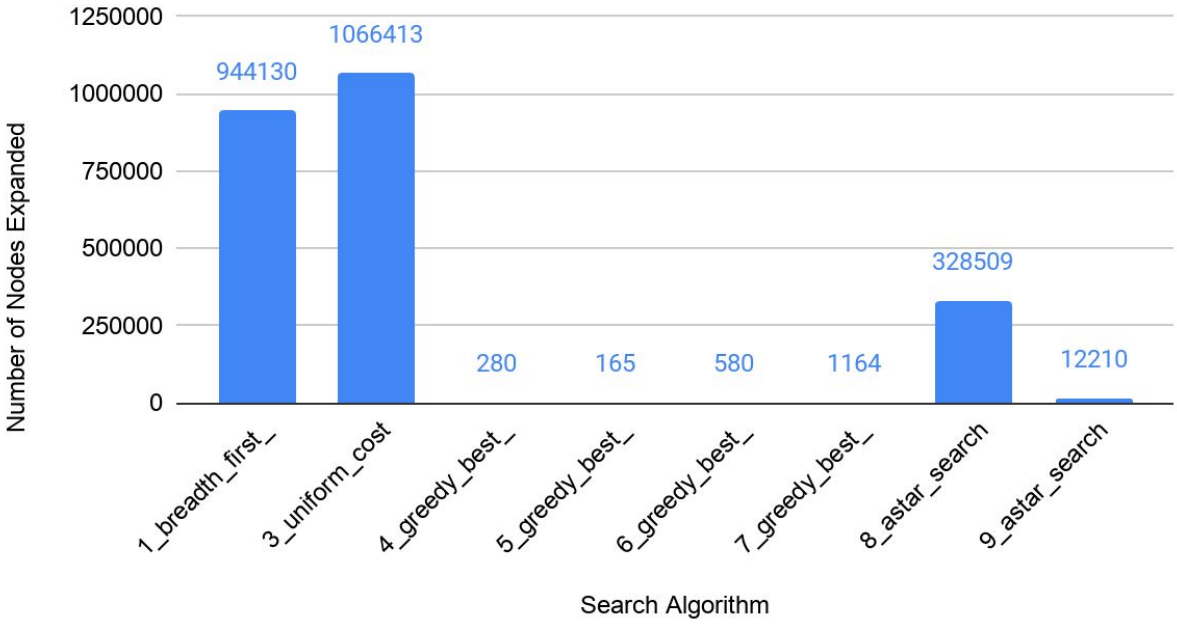
Problem 2 (72 actions) - Number of Nodes Expanded



Problem 3 (88 actions) - Number of Nodes Expanded



Problem 4 (104 actions) - Number of Nodes Expanded



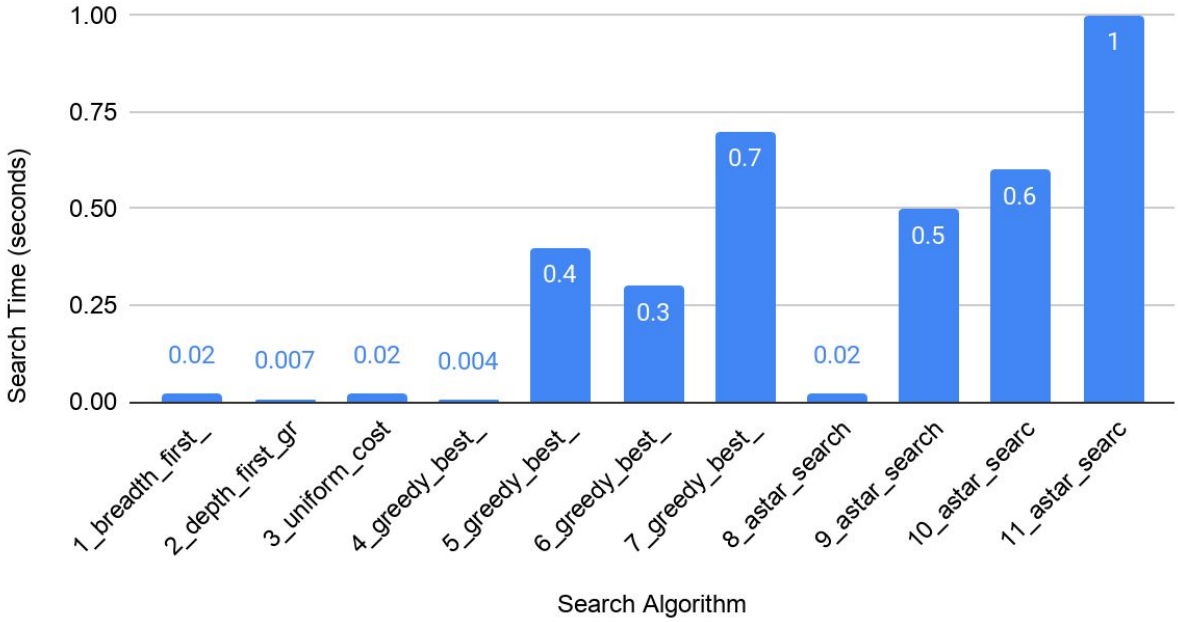
2. Analyze the search time against the number of actions in the domain.

For uninformed search algorithms, the search time grows slowly as the problem size increases.

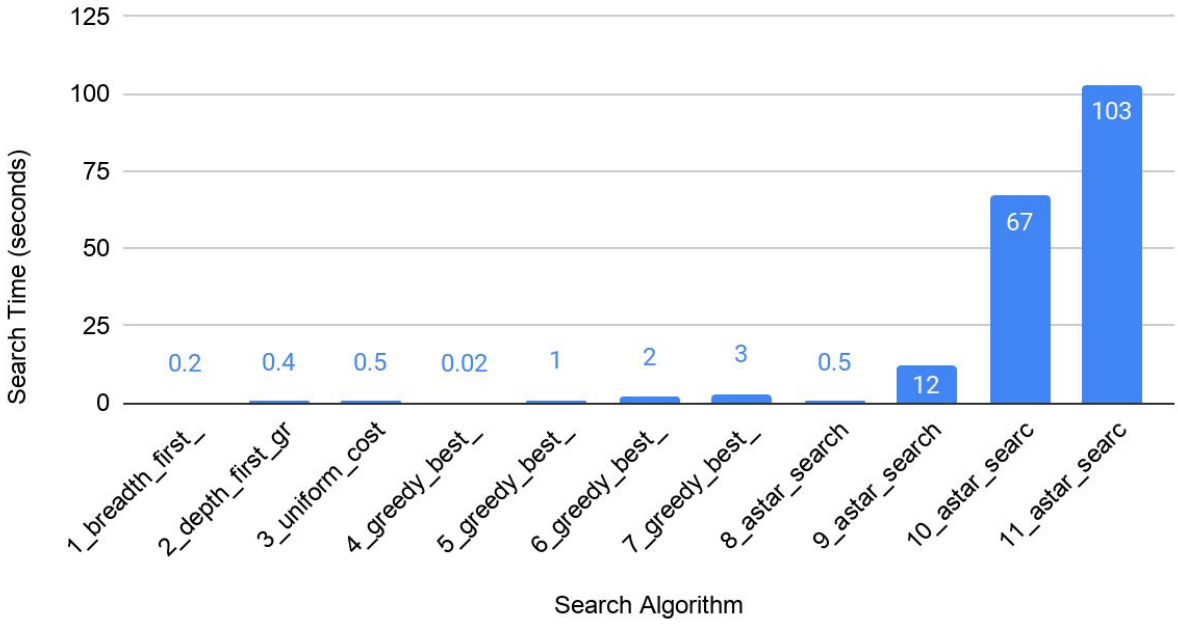
For heuristic greedy search algorithms, the search time grows relatively fast as the problem size increases. Among heuristic greedy search algorithms, unmet goals >> level sum >> max level >> set level (>> means faster).

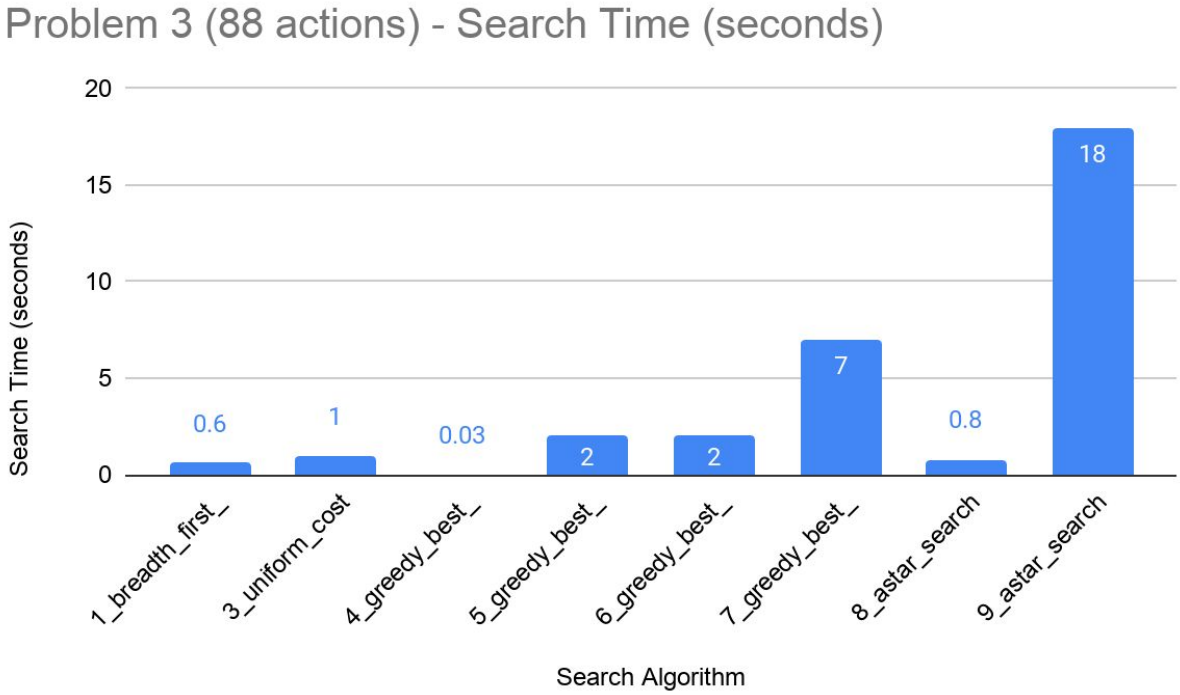
For heuristic A* search algorithms, the search time grows extremely fast as the complexity of the problem grows. As for problem 2, with 72 actions in domain, astar_search h_pg_setlevel already takes 103 seconds to find a solution. Among heuristic A* search algorithms, unmet goals >> level sum >> max level >> set level (>> means faster).

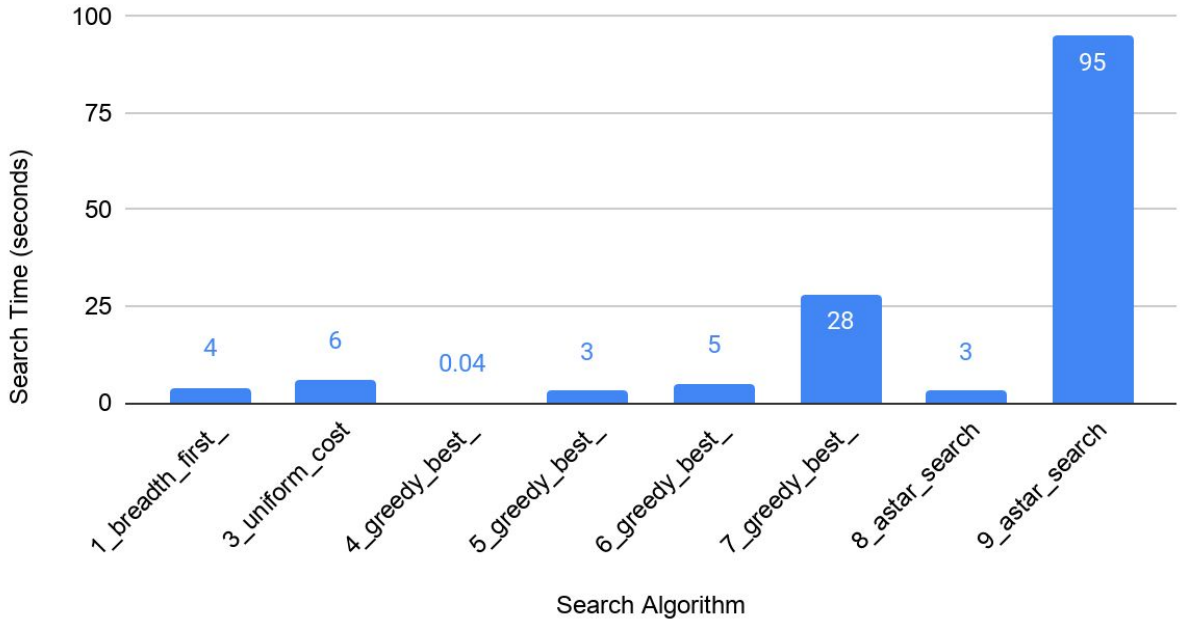
Problem 1 (20 actions) - Search Time (seconds)



Problem 2 (72 actions) - Search Time (seconds)




Problem 3 (88 actions) - Search Time (seconds)
Search Time (seconds)
20
15
10
5
0
0.6
1
0.03
2
2
7
0.8
18
1_breadth_first_
3_uniform_cost
4_greedy_best_
5_greedy_best_
6_greedy_best_
7_greedy_best_
8_astar_search
9_astar_search
Search Algorithm


Problem 4 (104 actions) - Search Time (seconds)
Search Time (seconds)
100
75
50
25
0
4
6
0.04
3
5
28
3
95
1_breadth_first_
3_uniform_cost
4_greedy_best_
5_greedy_best_
6_greedy_best_
7_greedy_best_
8_astar_search
9_astar_search
Search Algorithm

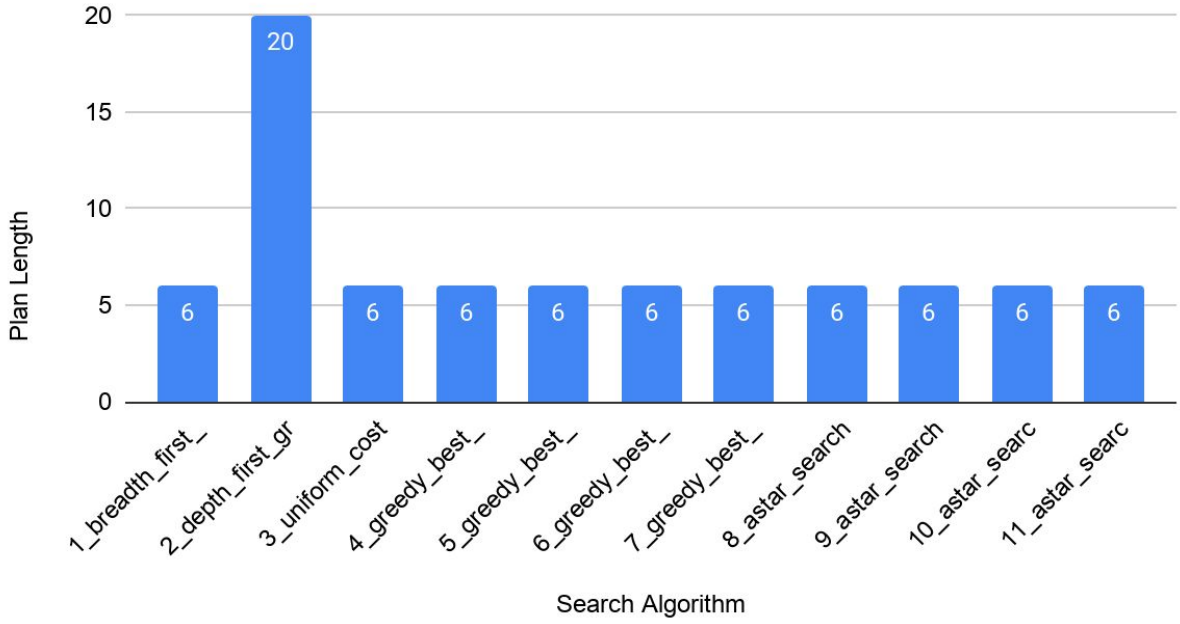
3. Analyze the length of the plans returned by each algorithm on all search problems.

For uninformed search algorithms, BFS and UCS always return the shortest plan (if computationally allowed in terms of CPU and memory). DFS simply returns a plan, usually much longer than the shortest.

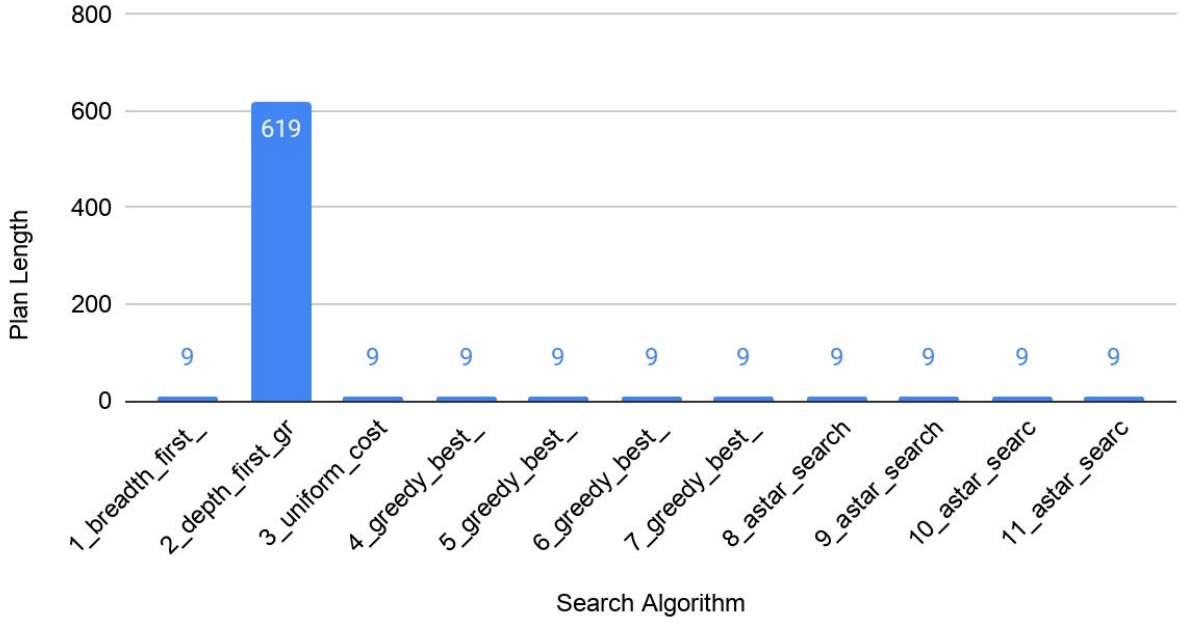
For heuristic greedy algorithms, the shortest plan is not guaranteed. However, the solution is usually close to optimal.

For heuristic A* algorithms, the shortest plan is also not guaranteed unless the heuristic is optimistic. However, the solution is usually close to optimal.

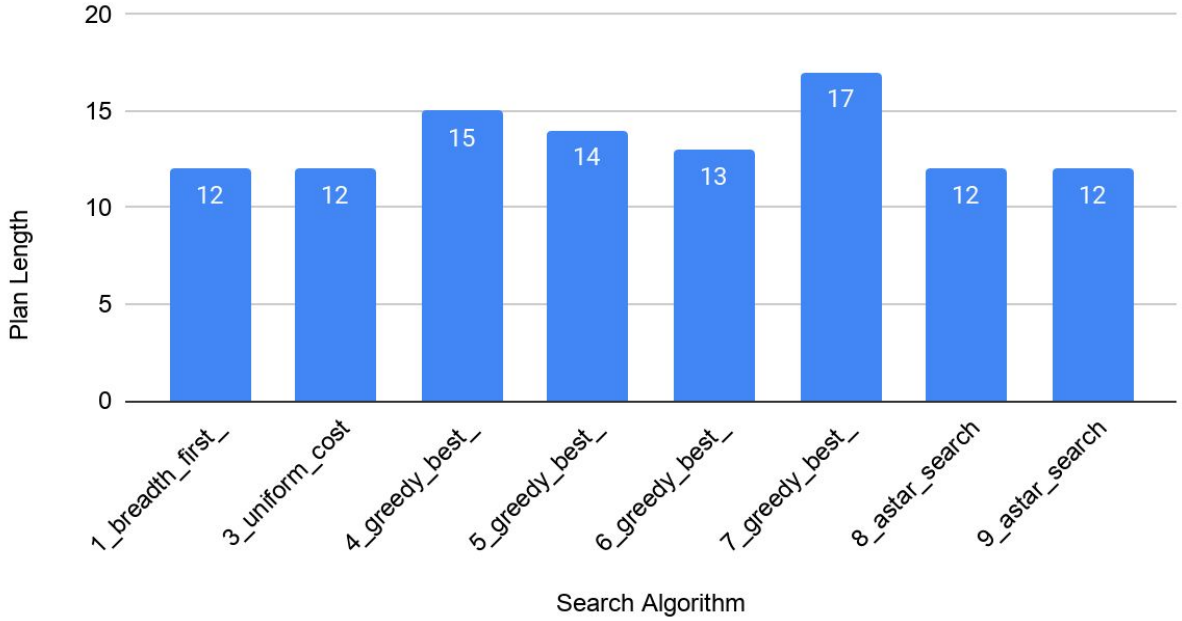
Problem 1 - Plan Length



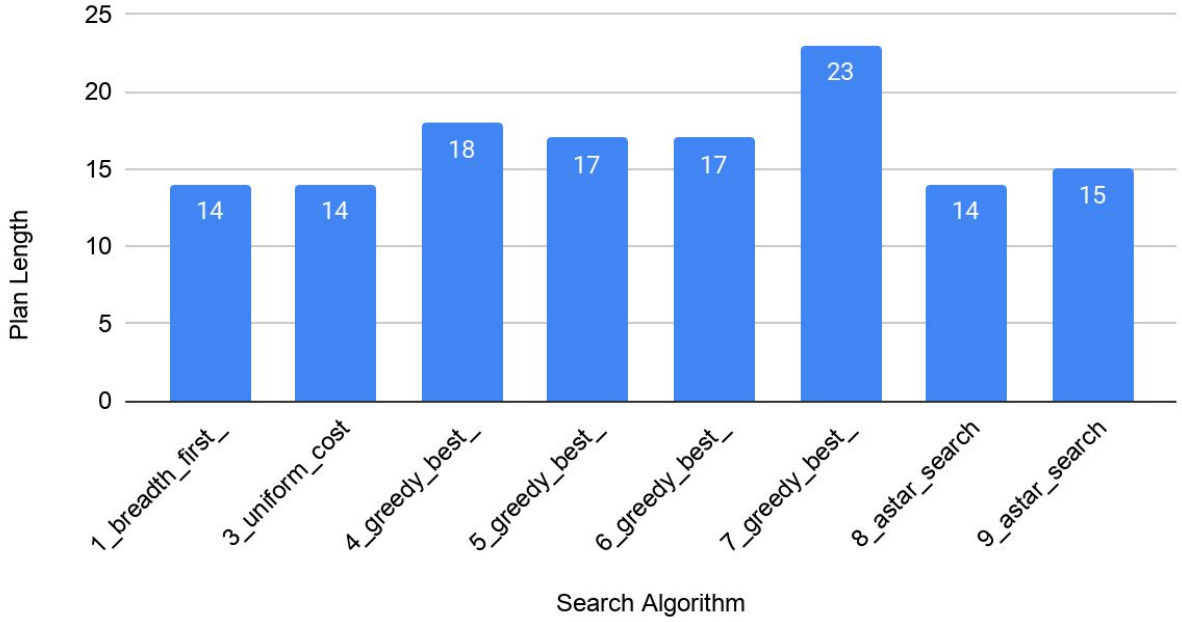
Problem 2 - Plan Length



Problem 3 - Plan Length



Problem 4 - Plan Length



4. Which algorithm or algorithms would be most appropriate for planning in a very restricted domain (i.e., one that has only a few actions) and needs to operate in real time?

BFS and UCS, because:

1. Guarantee to find the best plan
2. Simple implementation
3. Won't expand many nodes, and therefore won't consume much memory

5. Which algorithm or algorithms would be most appropriate for planning in very large domains (e.g., planning delivery routes for all UPS drivers in the U.S. on a given day)

Using the elimination method:

- Uninformed algorithms 1, 2, and 3 won't work, because they expand too many nodes
- Greedy algorithms 7 isn't great, because their plans are further from optimal
- A* algorithms 10, 11 won't work, they take too much time.

Among the rest algorithms 4, 5, 6, 8, 9:

- 8 expand more nodes
- 9 takes more time

And that leaves algorithms 4, 5 and 6.

6. Which algorithm or algorithms would be most appropriate for planning problems where it is important to find only optimal plans?

Algorithms 1, 3, and 8 guarantee to find the optimal plans. 1 and 3 would expand too many nodes when the problem is complex. And that leaves 8 as our only choice.