**Provide an optimal plan for Problems 1, 2, and 3**

For the planning exercise 1, 6 of the 7 planning algos had the optimal length of 6. It would be reasonable to select planning algo by time elapsed. Therefore, the winning algo was Greedy Best First Graph Search algo. We see that for this particular algo, the expansions, goal tests and new nodes were the fewest among all the algos.

For the planning exercise 2, 5 of the 7 planning algos had the optimal length of 9. It would also be reasonable to select planning algo by time elapsed. Therefore, the winning algo were Uniform Cost Search, A-star with null or ignore precondition heuristics. They have very similar performances regarding time elapsed, and their expansions, goal tests, and new nodes are the same.

For the planning exercise 3, 5 of the 7 planning algos had the optimal length of 9. It would also be reasonable to select planning algo by time elapsed. Therefore, the winning algo were also Uniform Cost Search, A-star with null or ignore precondition heuristics. They have very similar performances regarding time elapsed, and their expansions, goal tests, and new nodes are the same.

**Compare and contrast non-heuristic search result metrics (optimality, time elapsed, number of node expansions) for Problems 1,2, and 3. Include breadth-first, depth-first, and at least one other uninformed non-heuristic search in your comparison.**

I chose Breath First, Depth First and Uniform Cost searches for this exercise. As seen in the previous tables, we could quickly determine the optimal algos by first determine if the plan length is optimal. We can then select the optimal algo by looking at time elapsed and number of node expansions. For problem 1, because the environment was small, difference in time elapsed and node expansions weren’t apparent. However, Breath First was the winner because it had both the minimal node expansion and time elapsed.

For problem 2, as the environment was larger, the differences in performance metrics were more apparent. I believe the winner was the Uniform Cost search, as it had 10% performance improvement over Breath First on the time elapsed metric.

For problem 3, we could really see the difference is performances. The winner was also the Uniform Cost search, as it had 45% performance improvement over Breath First on the time elapsed metric.

**Compare and contrast heuristic search result metrics using A\* with the "ignore preconditions" and "level-sum" heuristics for Problems 1, 2, and 3.**

All the A\* algos were able to produce the correct results. However, the metrics, especially time elapsed, varied wildly differently. For problem 1, ignore precondition performed 40x faster than level-sum. For problem 2, ignore precondition performed 100x faster than level-sum. For problem 3, ignore precondition performed 130x faster than level-sum. In all three problems, ignore precondition came out on top. One important thing to note was that when the problem was small, the difference between ignore precondition and level-sum was not apparent. However, when the problem grew, the difference in performance was quite large.

**What was the best heuristic used in these problems? Was it better than non-heuristic search planning methods for all problems? Why or why not?**

The best heuristics were null and ignore precondition. I was quite surprised that null heuristic, which just outputs constant, was very similar to ignore preconditions, which checked if actions would achieve goals. In addition, A\* performance was very similar to uniform Cost Search. This is because Uniform Cost was similar to the null heuristic in that it used the pre-defined path costs, which were uniform. To improve on the performance of these algorithms, we needed to have real world ways to effectively measure costs. For example, path costs should be specific to operational expenses (fuel, personnel) and time;, ignore precondition should return cost such as distance between current state and goal state.