- Which heuristics did you use for the A* algorithm?

  The algorithm makes all possible choices, when the choices are made, it will compare the given new paths with the desired state, then choose this as the new location and repeat until the goal state is reached.

- Test your program with a couple of different problems. Increase the size of the problem to test the limits of your program. Make a table comparing **how many nodes are searched** to find the answer for each problem. For this table, you should compare a number of different problems (at least 3) to avoid a statistical bias. Which of the three algorithms (UCS, A *with consistent and and A* with an inconsistent heuristic) searches the least nodes and which one take the most?

  The algorithm is not perfect, but it show it defends itself in theory it should be able to find the goal state every single time at the expense of memory and execution time, it is not efficient, but it is consistent.

  A* with inconsistent heuristic will make the worst case, it can visit every single state in the worst-case scenario.

  A* depending on how good the heuristic is it will almost every time get the best path.

  UCS will get the  most consistent result, A* with consistent heuristic and USC will almost get the same result on every case.

- Why does this happen?

  Bad heuristics is the main reason also the efficiency of the algorithm to choose a better path with the given information

- Which algorithms are optimal? Why?

  UCS and A consistent is optimal because of the path it takes, it allows a good path finding with optimal path

- In your opinion, what are the benefits of simpler algorithms versus more complex ones?

  Better execution time and memory use on not so complex problems, it allows a fast-easy search on path that not require a major depth.