CS 404 - Homework 1

1)

Initial State:

The input of the user, the state that game starts and our 2x1x1 block is on the available tiles (O's), represented by single or double 'S'

Goal State:

The state, in which our block is exactly on top of the hole, fitting to the shape of hole (exactly 'one S' on the 'G').

Successor States:

Any of the states that our block is placed on the tiles (not on the X's, not even half of the block).

Successor State Function:

The function which will generate successor states when given a current state as a parameter.

Goal Test:

The test that checks if the current state is the goal state or not

State Cost:

Cost of proceeding into that step. Since X's are not considered as available places, they are also dismissed from the successor states. Thus, every other possible successor state will have the same cost, and since every move is only 1 keystroke in the game itself, cost is defined as "1" in this homework.

3)

Heuristic:

As heuristic, i used Manhattan Distance, which is the sum of the differences between goal and initial coordinates.

It is admissible since, any distance calculation via this method will be positive (assuming the destination and the starting point is different).

c(n1->n2) >= x > 0 holds.

Assume h*(n) be the cost of an optimal path from n to a goal node. Euclidean Distance also holds for the equation *h(n) <= h*(n)* regarding the cost is 1.

Hence, it also holds for *h(g)=0*, when we are at goal, the Manhattan Distance will be 0 to the goal state.

5)

Results are as expected. It can be said that, A* is overall faster and more efficient than UCS. Also in this homework, since costs are always same (1), UCS is acting exactly same as the BFS, so this table is also comparing A* to BFS, all the interpretations of the results are also applicable to BFS to A* comparison.

In some cases, the results are quite similar and the efficiency difference is not much, when inspected, it can be seen that, in these instances, for example, a path is seemed more close to the goal at the first (for the a*) but in the long run, its a dead end, so in the long run, nearly all the roads are traveled, in these cases UCS is also traveling nearly all possible paths, so the score is very similar in terms of efficiency.

The Table is below:

Test Case #	Time Consumption (UCS / A*)	Memory Consumption (UCS / A*)	Iteration Amount (UCS / A*)
1	0.006627 / 0.000465	9498 / 332	70 / 11
2	0.006312 / 0.003517	11769 / 7392	91 / 66
3	0.010876 / 0.006310	14429 / 12609	111 / 102
4	0.001971 / 0.000995	4163 / 1021	55 / 27
5	0.004425 / 0.003436	8173 / 8112	87 / 86
6	0.000073 / 0.000064	2 / 2	NA / NA(no solution, all possibilities have been tried)
7	0.000781 / 0.0004920	954 / 290	26 / 14

Memory Consumption is actually showing how many times there is an operation executing on lists, nodes (such as creating a node, popping from a heap, putting into a heap, comparing a node if that exist in the visited list, etc…)