CSE 537 – ARTIFICIAL INTELLIGENCE

ASSIGNMENT 1

SNIGDHA KAMAL
110937472

1. A heuristic is said to be admissible if it never overestimates the cost of reaching the goal from a current state (that is, the distance it estimates should not be greater than the lowest possible cost of reaching the goal from that state).

   0<= h(n) <=h*(n) for all n

   The two heuristics I used are:

   a. Number of misplaced tiles: This heuristic takes in the present configuration of the puzzle, compares it with the goal state of the puzzle and counts the numbers which are not in their right position (excluding the blank tile). The number of misplaced tiles is summed and returned by this heuristic function.

      This heuristic is admissible, as for any configuration, the total number of misplaced tiles will always be less than or equal to the actual number of steps that the blank tile will have to move to make sure all the other numbers are in their correct positions.

      1 2 3 Here, the number of misplaced tiles is 2 (#7, #8, position of blank space (0)
      4 5 6 not taken into account) and the distance the blank tile will move for the
      0 7 8 right configuration is also 2. Hence, number of misplaced tiles < = actual
      Distance and the heuristic is admissible.

   b. Manhattan Distance: This heuristic takes the present configuration of the puzzle and compares it with the final goal state of the puzzle. For each number that is not in its correct position, it returns the city-block or the Manhattan distance between the current position of the number and its right position

      Manhattan-Distance + = absolute(i-k) + absolute(j-l), where,
      i, j = index of out of place number
      k, l = index of correct position of number

      This heuristic is admissible as it assumes that you can only move 1 block at a time in any of the 4 directions yielding that best case scenario that there is a clear unobstructed path to the goal, which is generally not the case in reality as obstructions (other number tiles) need to be taken into account too

      1 3 2
      5 6 7
      4 0 8
      Manhattan distance = 1 + 1 + 1 + 1 + 1 + 3 + 1 = 9
      As is clearly obvious, the number of states that need to be checked before the goal state can be achieved will be greater than 9, making this heuristic admissible.

2. The code for the 15 puzzle took a very long time to yield an answer. A* algorithm saved all the child nodes it encountered on a queue which it then sorts by priority to extract the element with the least F value (F=G+H). All the other nodes take up memory space while they are stored on the queue, and this exhausts the available memory quickly. The memory complexity grows exponentially with the size of the input and the space complexity of A* becomes O(b^d). Hence the memory required to solve the 15 puzzle will be in exponential order, which reduces the efficiency of the algorithm.

3. The memory bound search algorithm I have used is IDA* Algorithm (Iterative Deepening A* search Algorithm). It is a depth first Iterative Deepening search with the F value used as the depth bound. Anytime a state with F value greater than the bound is encountered, the value is saved and the search starts again from the beginning with the new F value as the bound.

   (F = G+H)

   This algorithm performs a series of depth-first searches, hence its memory requirement was linear with respect to the maximum search depth as it only keeps one path in memory at a time, unlike the A * search Algorithm. In this manner, IDA* eliminates memory constraints of A* search without compromising on optimality. IDA* is recursive, much easier to implement and yields results faster than the A* algorithm.

   IDA* is complete and optimal. It is complete as it will find a solution if one exists.
   IDA* is optimal because it traverses all paths of lengths iteratively larger and halts only when it expands the goal node with a cost such that all paths of an inferior cost have been traversed already.
   The space usage of IDA* is directly proportional to the depth of the solution. Each iteration is a depth first search, hence the space requirement is b*d where b is the branching factor and d is the depth of the tree. Hence the space requirement is O(bd) which is a linear requirement.

4.

S. NO.	A STAR ALGORITHM HEURISTIC: MANHATTAN DISTANCE			A STAR ALGORITHM HEURISTIC: # OF MISPLACED TILES		
	NO. OF STATES EXPLORED	TIME TAKEN(ms)	DEPTH OF SOLUTION	NO. OF STATES EXPLORED	TIME TAKEN(ms)	DEPTH OF SOLUTION
1.	4	0.6668	3	4	5.232	3
2.	24	7.77	8	58	6.769	8
3.	4	0.89	3	4	0.577	3
4.	3	0.59	2	3	0.462	2
5.	24	8.89	9	125	58.082	9
6.	9	1.345	8	21	4.686	8
7.	4	0.711	3	4	1.269	3
8.	23	3.716	12	132	97.999	12
9.	10	1.46	6	12	1.576	6
10.	5	0.924	4	5	0.66	4
11.	4	0.98	3	4	0.637	3
12.	7	1.574	6	7	1.035	6
13.	6	1.15	5	6	0.707	5
14.	95	122.14	12	196	122.703	12
15.	15	4.37	9	19	2.410	9
16.	5	1.011	4	5	0.627	4
17.	22	38.95	13	132	69.342	13
18.	5	1.046	4	5	0.883	4
19.	52	40.341	11	196	83.002	11
20.	11	2.820	10	11	2.128	10

Resources Used:

1. For IDA* pseudocode:

   https://algorithmsinsight.wordpress.com/graph-theory-2/ida-star-algorithm-in-general/

   https://en.wikipedia.org/wiki/Iterative_deepening_A*

2. For A* Pseudocode:
   https://algorithmsinsight.wordpress.com/graph-theory-2/a-star-in-general/