CS561-Assignment-05

2.)

Example1:

```
Source:
['T2', 'T7', 'T5']
['B', 'T1', 'T8']
['T3', 'T4', 'T6']
Target:
['T1', 'T2', 'T3']
['T4', 'T5', 'T6']
['T7', 'T8', 'B']
--------------

Source:
['T2', 'T7', 'T5']
['B', 'T1', 'T8']
['T3', 'T4', 'T6']
Target state reached using tilesdisplaced
--------------
Optimal Path:
Number of states exceeded 100. Printing the first 100 states:
```

```
Source:
['T2', 'T7', 'T5']
['B', 'T1', 'T8']
['T3', 'T4', 'T6']
Total states visited = 24762
Target state not reached using manhatt_distance
Time taken for execution: 1.3231427669525146
```

Example2:

```
Source:
['T3', 'T7', 'B']
['T1', 'T6', 'T2']
['T5', 'T8', 'T4']
Target:
['T1', 'T2', 'T3']
['T4', 'T5', 'T6']
['T7', 'T8', 'B']
--------------

Source:
['T3', 'T7', 'B']
['T1', 'T6', 'T2']
['T5', 'T8', 'T4']
Total states visited = 25133
Target state not reached using tilesdisplaced
Time taken for execution: 0.9373958110809326
--------------

Source:
['T3', 'T7', 'B']
['T1', 'T6', 'T2']
['T5', 'T8', 'T4']
Target state reached using manhatt_distance
--------------
Optimal Path:
Number of states exceeded 100. Printing the first 100 states:
```

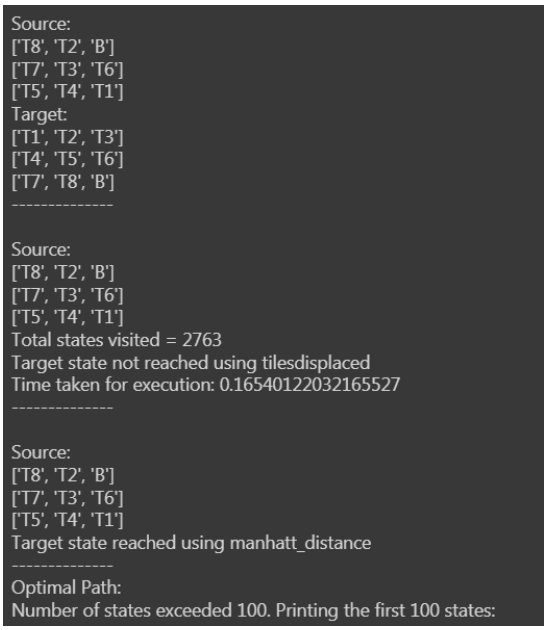
Example3:

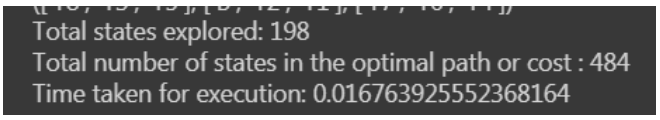
```
Source:
['T1', 'T2', 'T3']
['T4', 'B', 'T6']
['T7', 'T5', 'T8']
Target:
['T1', 'T2', 'T3']
['T4', 'T5', 'T6']
['T7', 'T8', 'B']
--------------
```

```
Source:
['T1', 'T2', 'T3']
['T4', 'B', 'T6']
['T7', 'T5', 'T8']
Target state reached using tilesdisplaced
--------------
Optimal Path:
Move 0:
(['T1', 'T2', 'T3'], ['T4', 'B', 'T6'], ['T7', 'T5', 'T8'])
Move 1:
(['T1', 'T2', 'T3'], ['T4', 'T5', 'T6'], ['T7', 'B', 'T8'])
Move 2:
(['T1', 'T2', 'T3'], ['T4', 'T5', 'T6'], ['B', 'T7', 'T8'])
Move 3:
(['T1', 'T2', 'T3'], ['T4', 'T5', 'T6'], ['T7', 'B', 'T8'])
Move 4:
(['T1', 'T2', 'T3'], ['T4', 'T5', 'T6'], ['T7', 'B', 'T8'])
Move 5:
(['T1', 'T2', 'T3'], ['T4', 'T5', 'T6'], ['B', 'T7', 'T8'])
Move 6:
(['T1', 'T2', 'T3'], ['B', 'T5', 'T6'], ['T4', 'T7', 'T8'])
Move 7:
(['T1', 'T2', 'T3'], ['T4', 'T5', 'T6'], ['B', 'T7', 'T8'])
Move 8:
(['T1', 'T2', 'T3'], ['T4', 'T5', 'T6'], ['T7', 'B', 'T8'])
Move 9:
(['T1', 'T2', 'T3'], ['T4', 'T5', 'T6'], ['B', 'T7', 'T8'])
Move 10:
(['T1', 'T2', 'T3'], ['T4', 'T5', 'T6'], ['T7', 'B', 'T8'])
Move 11:
(['T1', 'T2', 'T3'], ['T4', 'T5', 'T6'], ['T7', 'T8', 'B'])
Total states explored: 4
Total number of states in the optimal path or cost : 11
Time taken for execution: 0.0004291534423828125
```

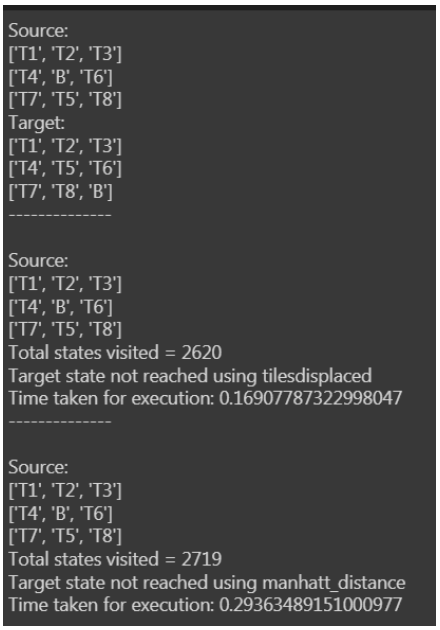
```
Source:
['T1', 'T2', 'T3']
['T4', 'B', 'T6']
['T7', 'T5', 'T8']
Target state reached using manhatt_distance
--------------
Optimal Path:
Move 0:
(['T1', 'T2', 'T3'], ['T4', 'B', 'T6'], ['T7', 'T5', 'T8'])
Move 1:
(['T1', 'T2', 'T3'], ['T4', 'T6', 'B'], ['T7', 'T5', 'T8'])
Move 2:
(['T1', 'T2', 'B'], ['T4', 'T6', 'T3'], ['T7', 'T5', 'T8'])
Move 3:
(['T1', 'T2', 'T3'], ['T4', 'T6', 'B'], ['T7', 'T5', 'T8'])
Move 4:
(['T1', 'T2', 'T3'], ['T4', 'B', 'T6'], ['T7', 'T5', 'T8'])
Move 5:
(['T1', 'T2', 'T3'], ['T4', 'T5', 'T6'], ['T7', 'B', 'T8'])
Move 6:
(['T1', 'T2', 'T3'], ['T4', 'T5', 'T6'], ['T7', 'T8', 'B'])
Total states explored: 4
Total number of states in the optimal path or cost : 6
Time taken for execution: 0.00034689903259277344
```

Example4:





Example5:



The Simulated Annealing algorithm is used to find a sequence of moves from the initial state to the goal state.

We have used two heuristic function tiles displaced and the Manhattan heuristic function. From this We can see that in this algorithm in one test case if the source reaches the target state if we run the same test case then that test case might not work(Example 5 and 3). Most of the cases Manhattan distance works better when compared to the tiles displaced heuristic function.

a)Comparison of Both Heuristics:

Tiles Displaced Heuristic: This heuristic calculates the number of tiles that are not in their correct positions compared to the goal state. It's a simple heuristic that gives higher values for states that are far from the goal. In the context of the simulated annealing algorithm, it encourages exploration but may not necessarily lead to the shortest path.

Manhattan Distance Heuristic: This heuristic calculates the Manhattan distance of each tile from its correct position in the goal state. It takes into account the distance each tile needs to move to reach its goal position. It provides a more informed estimate of the state's distance from the goal.

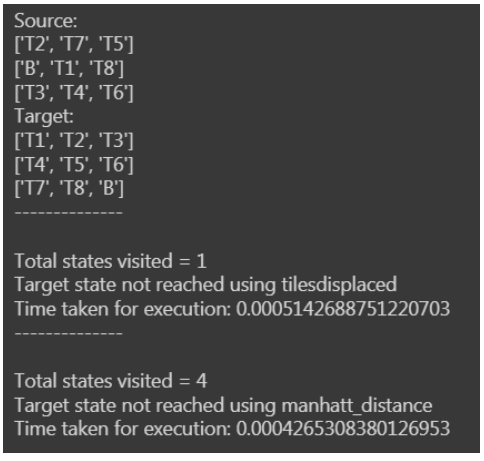
b)Result Analysis:
Across all four test cases, the Manhattan Distance Heuristic consistently yielded superior results compared to the Tiles Displaced Heuristic. Notably, the simulated annealing algorithm's behaviour exhibited stochasticity; in some instances, if the initial source state successfully reached the target state, rerunning the same test case might not produce the same outcome. This variability arises due to the algorithm's probabilistic nature, making it a versatile tool for navigating complex search spaces and finding optimal solutions.

c)Difference in Approach:
The difference in approach between the heuristics lies in their ability to estimate the distance of a state from the goal. The Tiles Displaced Heuristic considers only the number of misplaced tiles, which can be misleading in some cases. In contrast, the Manhattan Distance Heuristic provides a more accurate estimate by considering the actual distances tiles need to move. In the simulated annealing algorithm, the choice of heuristic affects how likely it is to accept a move to a neighbouring state. The Manhattan Distance Heuristic tends to guide the algorithm toward the goal state more effectively.

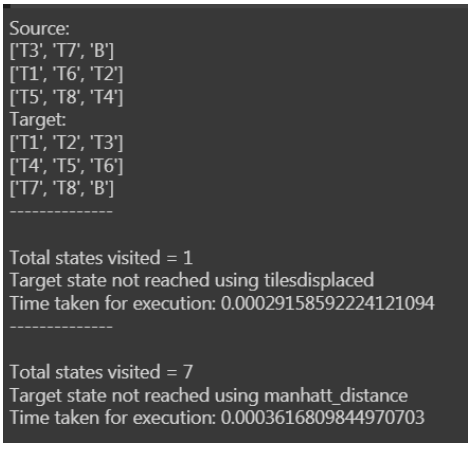
d. Algorithm Performance:
For this particular problem, the **Manhattan Distance** Heuristic is expected to perform better overall. It provides a more accurate estimate of the distance from the goal, leading to more efficient exploration of state space. This matches the results seen in the examples where the Manhattan Distance Heuristic outperforms the Tiles Displaced Heuristic.
Intuitively, the Manhattan Distance Heuristic is preferable because it guides the algorithm toward states that are closer to the goal. This leads to faster convergence to the optimal solution.

In conclusion, the choice of heuristic can significantly impact the performance of the simulated annealing algorithm. In most cases, heuristics that provide more accurate estimates of the distance from the goal, such as the Manhattan Distance Heuristic, are expected to perform better for pathfinding problems like the one described.

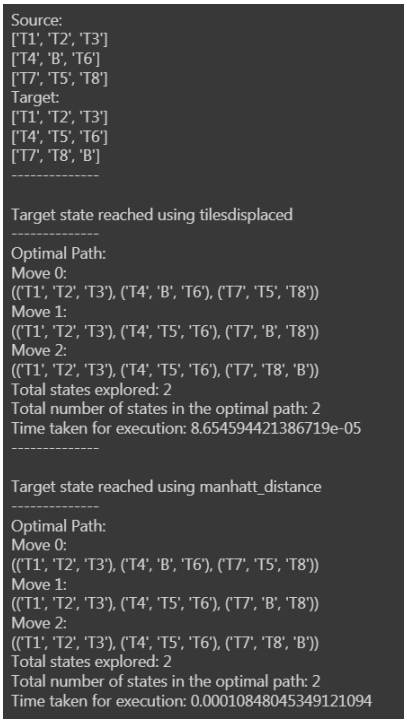
3).
Running the Hill Climbing for same test cases :
Example 1:



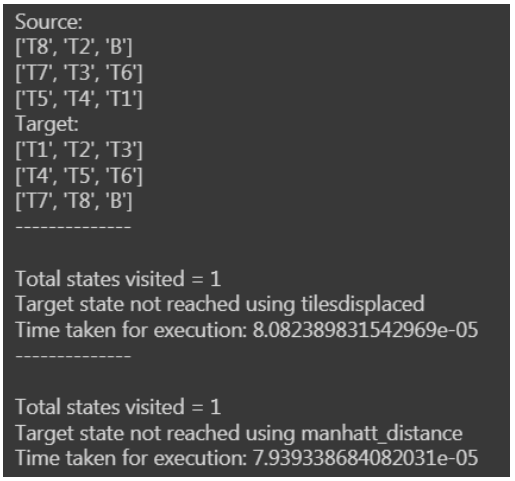
Example2:



Example3:



Example4:



b)**Analysis of Results:**

Hill Climbing: This algorithm showed consistency in its behaviour. In cases where the initial state could lead to the target, Hill Climbing often reached the goal successfully. However, it tends to get stuck in local optima, making it less effective in more complex search spaces.

Simulated Annealing: Simulated Annealing demonstrated more variability due to its probabilistic nature. While it might initially choose suboptimal paths, it can escape local optima, exploring the search space more effectively. This stochasticity makes it suitable for complex problems.

c. Performance Comparison:

In general, for this problem, **Simulated Annealing performed better**. Its probabilistic nature allowed it to explore a broader solution space, increasing the chances of finding an optimal solution. It was less likely to get stuck in **local optima** compared to Hill Climbing.

d. Intuition vs. Results:

Our intuition suggests that for problems with a complex search space, where local optima are prevalent, **Simulated Annealing** should perform better due to its ability to escape these optima. The results match with this intuition, showing that Simulated Annealing outperforms Hill Climbing in such scenarios.

However, **Hill Climbing may excel** in simpler problems with fewer local optima and clear paths to the goal.

Team Members:
1.Devanand 2001CS19
2.Gnaneshwar 2001CS15
3.Teja Vardhan 2001CS26