CS 331 - Eight Puzzle Programming Assignment
04/25/2023
Edson Fuentes
Nathan Rumsey

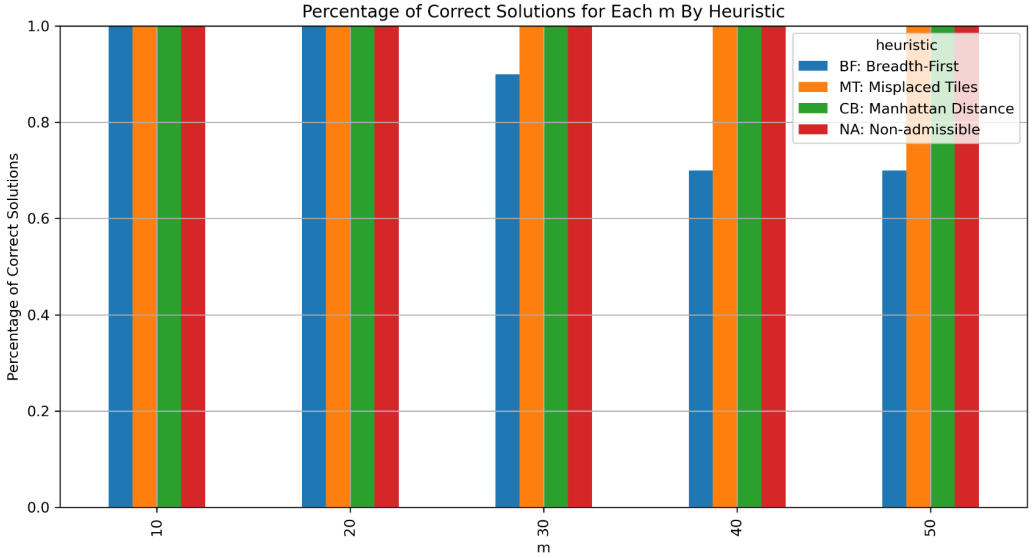
Contributions Statement
Nathan Rumsey: All code and development for this assignment. Generated report plots.
Edson Fuentes: All report text and contributed ideas for non-admissible heuristic and a* algorithm implementation.

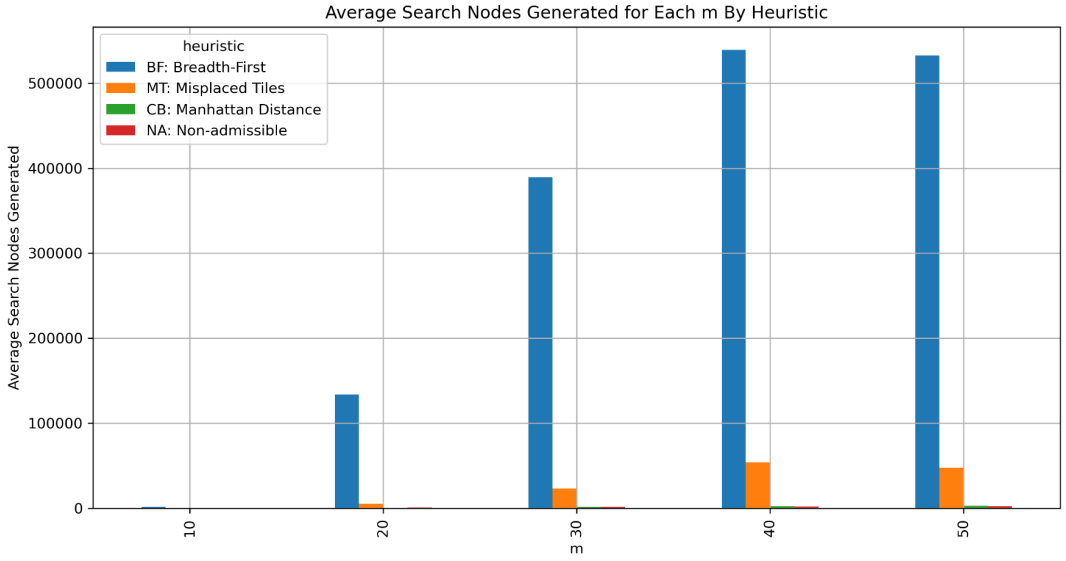
Heuristic Testing on 5 Random Problems

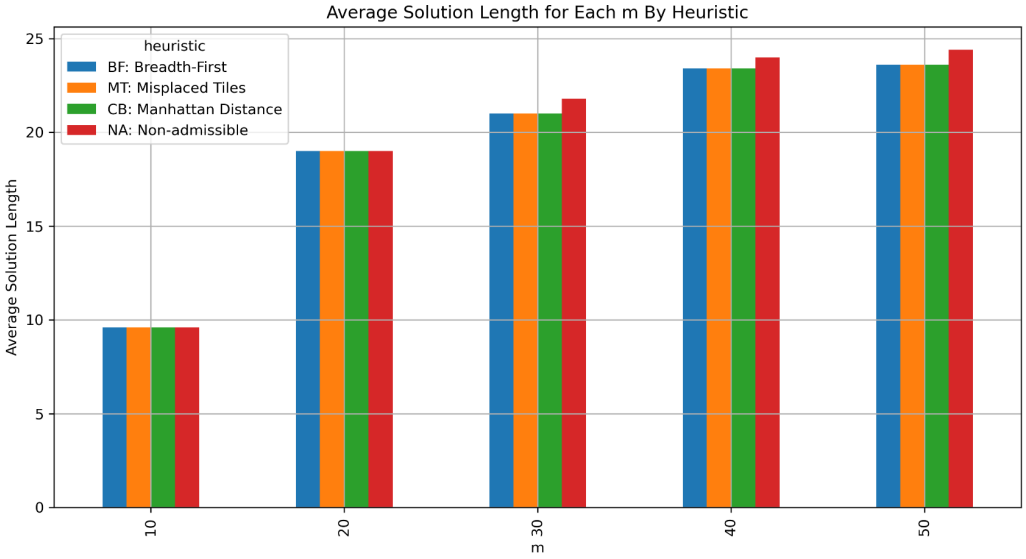
```
(cs331) PS C:\Users\natha\Documents\Programming\OSU\Classes\CS331\8_Puzzle> python .\test.py
 Tests for BF Heuristic
Optimal soultion for problem 0 where m=10 and s=21
Optimal soultion for problem 1 where m=20 and s=77
Optimal soultion for problem 2 where m=30 and s=39
Solution Timed Out for problem 3 where m=40 and s=8
Optimal soultion for problem 4 where m=50 and s=402
Time elapsed:  87.38626313209534 seconds

Tests for MT Heuristic
Optimal soultion for problem 0 where m=10 and s=21
Optimal soultion for problem 1 where m=20 and s=77
Optimal soultion for problem 2 where m=30 and s=39
Optimal soultion for problem 3 where m=40 and s=8
Optimal soultion for problem 4 where m=50 and s=402
Time elapsed:  38.281673192977905 seconds

Test for CB Heuristic
Optimal soultion for problem 0 where m=10 and s=21
Optimal soultion for problem 1 where m=20 and s=77
Optimal soultion for problem 2 where m=30 and s=39
Optimal soultion for problem 3 where m=40 and s=8
Optimal soultion for problem 4 where m=50 and s=402
Time elapsed:  1.511673927307129 seconds

Test for NA Heuristic
Optimal soultion for problem 0 where m=10 and s=21
Suboptimal soultion for problem 1 where m=20 and s=77
Optimal soultion for problem 2 where m=30 and s=39
Optimal soultion for problem 3 where m=40 and s=8
Optimal soultion for problem 4 where m=50 and s=402
Time elapsed:  1.508819580078125 seconds
```

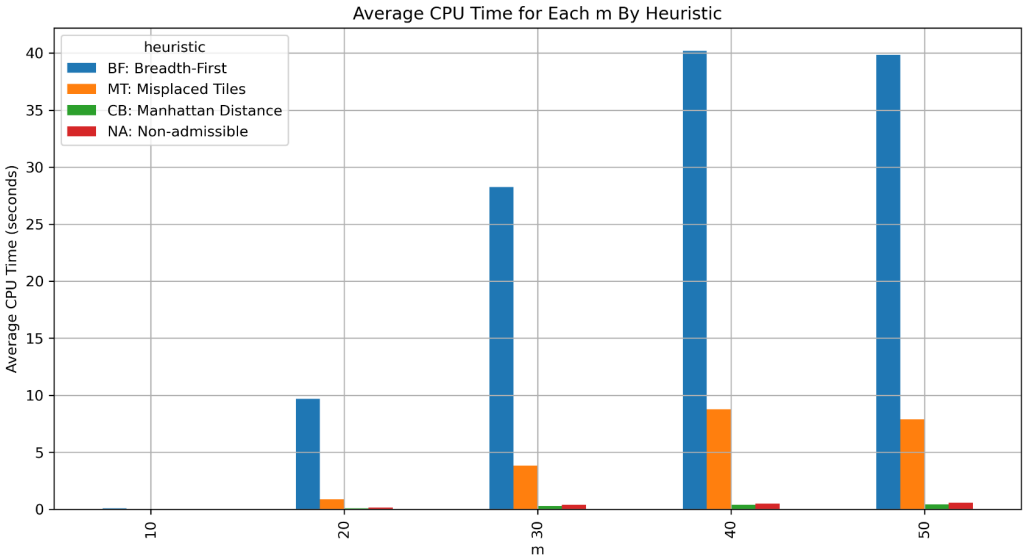
* A* Algorithm times out after 60 seconds

Result Plots






Average Solution Length for Each m By Heuristic
heuristic
BF: Breadth-First
MT: Misplaced Tiles
CB: Manhattan Distance
NA: Non-admissible
Average Solution Length
0
5
10
15
20
25
10
20
30
40
50
m


Average CPU Time for Each m By Heuristic
heuristic
BF: Breadth-First
MT: Misplaced Tiles
CB: Manhattan Distance
NA: Non-admissible
Average CPU Time (seconds)
0
5
10
15
20
25
30
35
40
10
20
30
40
50
m

Discussion Questions

1. How did the two algorithms and the three heuristics compare in terms of the number of nodes searched, the solution length, and the CPU time?
Taking a look at the plots above, we can make several observations about the efficiency of the different algorithms and heuristics. Breadth First Search appears to be the least efficient with regards to CPU times, number of nodes searched, and the percentage of correct solutions. The heuristics of A* performed much better than BFS in almost all areas. The solution lengths for all the algorithms and heuristics were similar, but in every other category the Misplaces Tiles, Manhattan Distance, and the Non-Admissible Heuristic performed better. While the Non-Admissible Heuristic is comparable to the CB (Manhattan Distance Heuristic) with the others, it yields non-optimal solutions (evidenced by the higher on average solution length). This makes the Manhattan Distance Heuristic by far the best heuristic of these for time, efficiency, memory usage, and accuracy (finding optimal solutions).

2. Is there a clear preference ordering among the heuristics? Is there a tradeoff between solution length and the number of nodes searched or between CPU time and the number of nodes searched?
If we were to look at the plots above, specifically the relationship between the average nodes searched and the average solution length, we can see that there is a tradeoff between these two. Breadth First Search appears to have no tradeoff. The other three heuristics on the other hand showcase that the fewer nodes we search, the higher the average solution length is.

3. How did you come up with your heuristic evaluation function?
As stated in the notes, a heuristic is admissible if and only if h(n) <= h*(n), or the predicted distance is always less than or equal to the overall distance. Our heuristic function is the sum of the misplaced tiles and the manhattan distance plus 1. This yields an overestimated distance when at the goal state (1 instead of 0) for example. This single case makes it non-admissible, but other cases also overestimate, such as the case where all the 8 outer tiles need to be shifted once clockwise. The true distance would be 8 since it would take 8 actions to achieve the goal state, but the heuristic function would return 7+7+1=15 (#misplaced tiles+7*(manhattan distance for each tile - 1)+1).

4. Is there anything you are surprised by or learn from the experiment?
The main thing that surprised us was the efficiency of CB (Manhattan Distance Heuristic) in the sense that it captures the amount of steps rather than the distance. Misplaced Tiles is much worse due to the fact that it is naive. It is only taking into account the fact that the current tile is misplaced as an indicator of distance and not how many actions it might take to achieve the goal state. The Manhattan Distance heuristic does take this into account and knows the actions needed to achieve the goal state.