CS571 - ARTIFICIAL INTELLIGENCE LAB

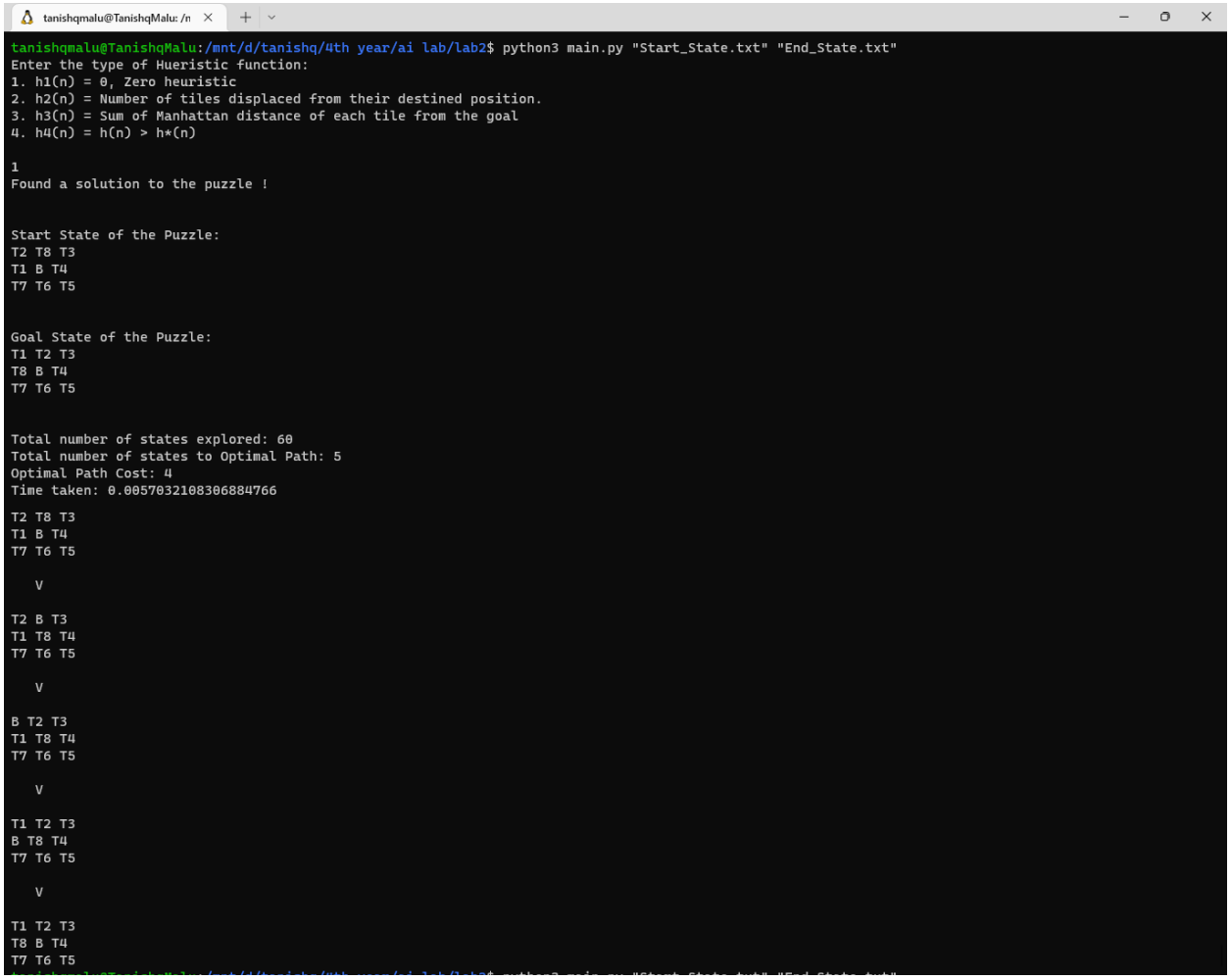
Lab - 2
A* Algorithm

Jenish Monpara	1901CS28
Tanishq Malu	1901CS63
Tarusi Mittal	1901CS65

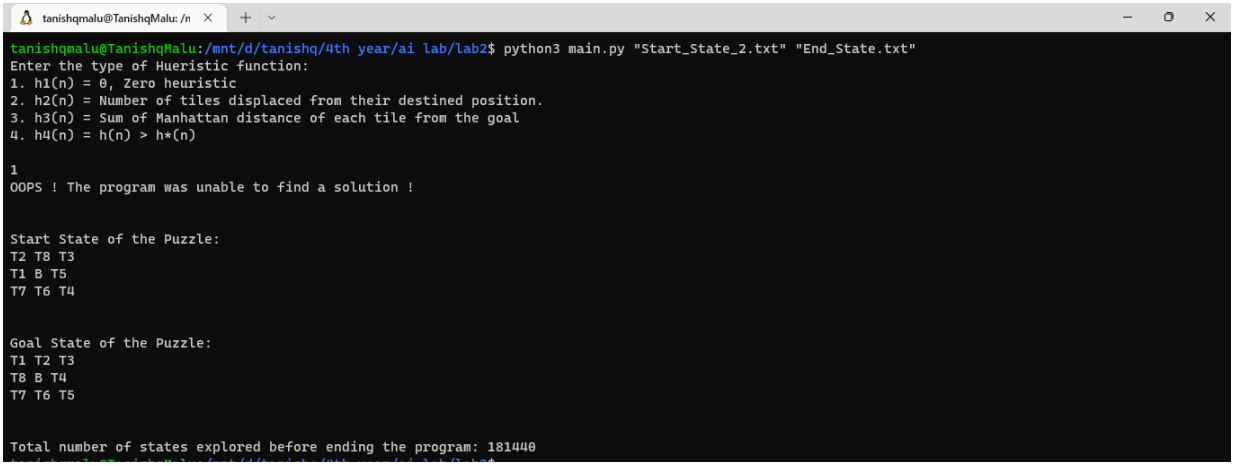
Sample Input-Output for each case for Success and failure cases:

Heuristic type 1: Zero Heuristic

Solution exists case

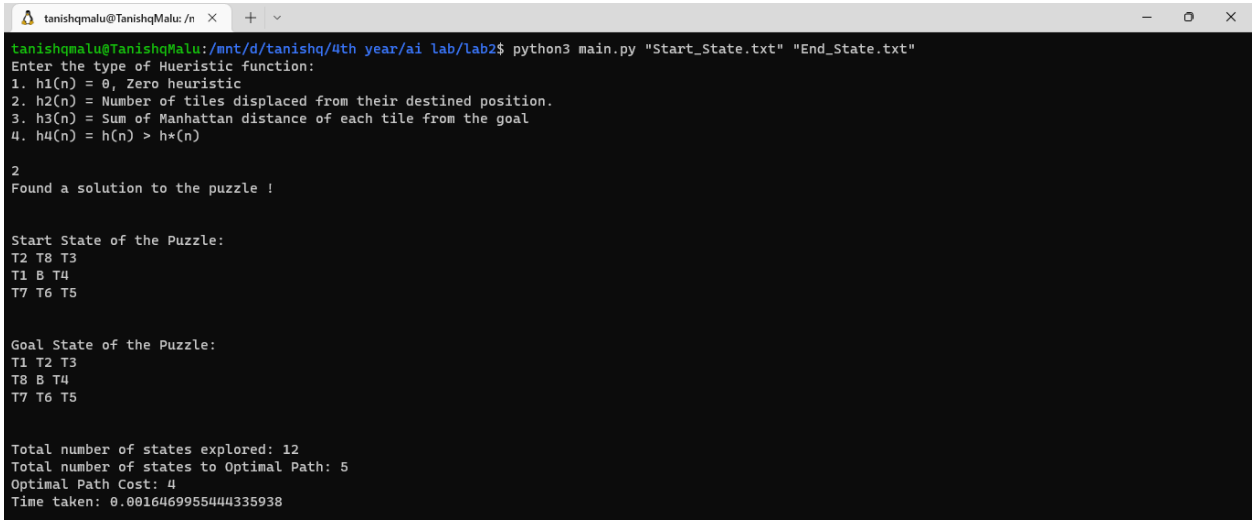


Solution does not exists case



Heuristic type 2: Number of tiles displaced from their destined position

Solution exists case



Solution does not exists case

```
tanishqmalu@TanishqMalu:/mnt/d/tanishq/4th year/ai lab/lab2$ python3 main.py "Start_State_2.txt" "End_State.txt"
Enter the type of Hueristic function:
1. h1(n) = 0, Zero heuristic
2. h2(n) = Number of tiles displaced from their destined position.
3. h3(n) = Sum of Manhattan distance of each tile from the goal
4. h4(n) = h(n) > h*(n)

2
OOPS ! The program was unable to find a solution !


Start State of the Puzzle:
T2 T8 T3
T1 B T5
T7 T6 T4


Goal State of the Puzzle:
T1 T2 T3
T8 B T4
T7 T6 T5


Total number of states explored before ending the program: 181440
tanishqmalu@TanishqMalu:/mnt/d/tanishq/4th year/ai lab/lab2$
```

```
T2 T8 T3
T1 B T4
T7 T6 T5

   V

T2 B T3
T1 T8 T4
T7 T6 T5

   V

B T2 T3
T1 T8 T4
T7 T6 T5

   V

T1 T2 T3
B T8 T4
T7 T6 T5

   V

T1 T2 T3
T8 B T4
T7 T6 T5
```

Heuristic type 3: Sum of Manhattan distance of each tiles from the goal position.

Solution exists case

```
tanishqmalu@TanishqMalu: /n
tanishqmalu@TanishqMalu:/mnt/d/tanishq/4th year/ai lab/lab2$ python3 main.py "Start_State.txt" "End_State.txt"
Enter the type of Hueristic function:
1. h1(n) = 0, Zero heuristic
2. h2(n) = Number of tiles displaced from their destined position.
3. h3(n) = Sum of Manhattan distance of each tile from the goal
4. h4(n) = h(n) > h*(n)

3
Found a solution to the puzzle !


Start State of the Puzzle:
T2 T8 T3
T1 B T4
T7 T6 T5


Goal State of the Puzzle:
T1 T2 T3
T8 B T4
T7 T6 T5


Total number of states explored: 10
Total number of states to Optimal Path: 5
Optimal Path Cost: 4
Time taken: 0.0012984275817871094
```

Solution does not exists case

```
tanishqmalu@TanishqMalu:/mnt/d/tanishq/4th year/ai lab/lab2$ python3 main.py "Start_State_2.txt" "End_State.txt"
Enter the type of Hueristic function:
1. h1(n) = 0, Zero heuristic
2. h2(n) = Number of tiles displaced from their destined position.
3. h3(n) = Sum of Manhattan distance of each tile from the goal
4. h4(n) = h(n) > h*(n)

3

OOPS ! The program was unable to find a solution !


Start State of the Puzzle:
T2 T8 T3
T1 B T5
T7 T6 T4


Goal State of the Puzzle:
T1 T2 T3
T8 B T4
T7 T6 T5


Total number of states explored before ending the program: 181440
```

Heuristic type 4: Squared sum of difference in rows and difference in columns

for each tile:

ri = find number of rows it needs to be swapped

ci = find number of column it needs to be swapped to reach original

h(n) = (ri)* (ci) + 1

Solution exists case

```
tanishqmalu@TanishqMalu: /n
tanishqmalu@TanishqMalu:/mnt/d/tanishq/4th year/ai lab/lab2$ python3 main.py "Start_State.txt" "End_State.txt"
Enter the type of Hueristic function:
1. h1(n) = 0, Zero heuristic
2. h2(n) = Number of tiles displaced from their destined position.
3. h3(n) = Sum of Manhattan distance of each tile from the goal
4. h4(n) = h(n) > h*(n)

4
Found a solution to the puzzle !


Start State of the Puzzle:
T2 T8 T3
T1 B T4
T7 T6 T5


Goal State of the Puzzle:
T1 T2 T3
T8 B T4
T7 T6 T5


Total number of states explored: 34
Total number of states to Optimal Path: 5
Optimal Path Cost: 4
Time taken: 0.00530242919921875
```

Solution does not exists case

```
tanishqmalu@TanishqMalu:/mnt/d/tanishq/4th year/ai lab/lab2$ python3 main.py "Start_State_2.txt" "End_State.txt"
Enter the type of Hueristic function:
1. h1(n) = 0, Zero heuristic
2. h2(n) = Number of tiles displaced from their destined position.
3. h3(n) = Sum of Manhattan distance of each tile from the goal
4. h4(n) = h(n) > h*(n)

4

OOPS ! The program was unable to find a solution !


Start State of the Puzzle:
T2 T8 T3
T1 B T5
T7 T6 T4


Goal State of the Puzzle:
T1 T2 T3
T8 B T4
T7 T6 T5


Total number of states explored before ending the program: 181440
```

1. Observe and verify that better heuristics expands lesser states.

Lets try to observe this for the following test case

Start Matrix :

2	8	3
1	B	4
7	6	5

Goal Matrix:

1	2	3
8	B	4
7	6	5

```
1
Found a solution to the puzzle !


Start State of the Puzzle:
T1 T2 T3
B T8 T4
T6 T5 T7


Goal State of the Puzzle:
T1 T2 T3
T8 B T4
T7 T6 T5


Total number of states explored: 10606
Total number of states to Optimal Path: 16
Optimal Path Cost: 15
Time taken: 0.8431782722473145
```

```
2
Found a solution to the puzzle !


Start State of the Puzzle:
T1 T2 T3
B T8 T4
T6 T5 T7


Goal State of the Puzzle:
T1 T2 T3
T8 B T4
T7 T6 T5


Total number of states explored: 886
Total number of states to Optimal Path: 16
Optimal Path Cost: 15
Time taken: 0.08797907829284668
```

```
3
Found a solution to the puzzle !


Start State of the Puzzle:
T1 T2 T3
B T8 T4
T6 T5 T7


Goal State of the Puzzle:
T1 T2 T3
T8 B T4
T7 T6 T5


Total number of states explored: 240
Total number of states to Optimal Path: 16
Optimal Path Cost: 15
Time taken: 0.027650117874145508
```

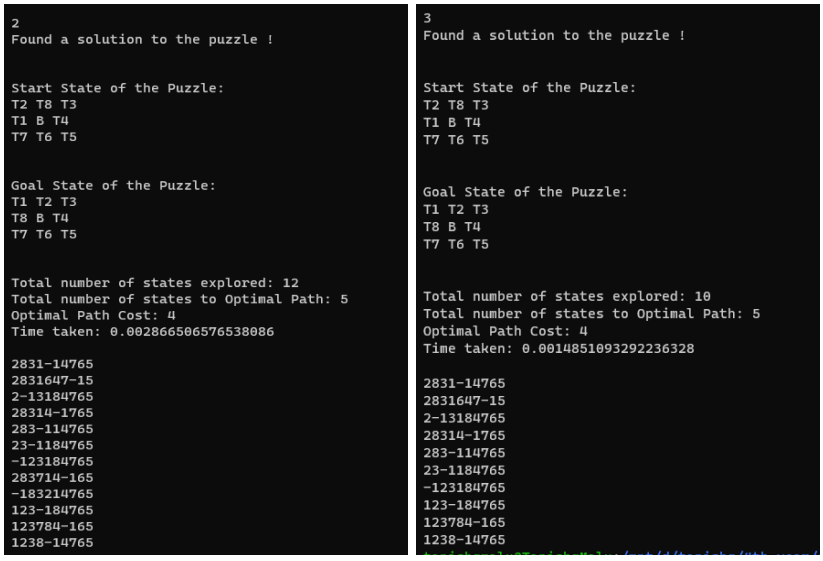
```
4
Found a solution to the puzzle !


Start State of the Puzzle:
T1 T2 T3
B T8 T4
T6 T5 T7


Goal State of the Puzzle:
T1 T2 T3
T8 B T4
T7 T6 T5


Total number of states explored: 2950
Total number of states to Optimal Path: 16
Optimal Path Cost: 15
Time taken: 0.2956256866455078
```

Number of states explored by:

Heuristic	h1	h2	h3	h4
No. of states explored	10606	886	240	2950

Thus we can verify from the above case that Manhattan heuristic(h3) provides better results from the given rest of the heuristic functions.

2. Observe and verify that all the states expanded by better heuristics should also be expanded by inferior heuristics.



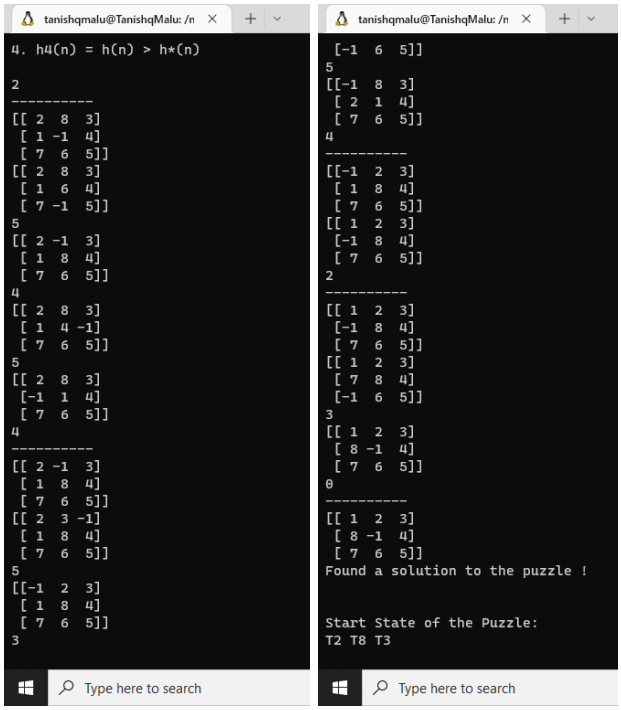
For the above test case, when we try to run it for h2 and h3, we see that h3 explores 10 states and h2 explores 12 states. On examining the elements in closed list we can see that all the state expanded by h3 are also expanded by h2

3. Observe and verify monotone restriction on the heuristics.

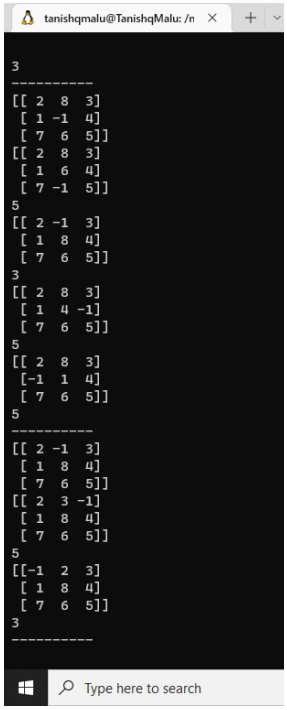
Monotone restriction: h(n) <= cost(n,m) + h(m)

a. h2(n) = number of tiles displaced from their destined position.

b. h3(n) = sum of Manhattan distance of each tiles from the goal position.



The above ss for h2 heuristic clearly shows montonic resriction of h2.Similarly for h3:



4. Observe un-reachability and provide a proof.

A pair (A[i], A[j]) is said to be in inversion if:

- A[i] > A[j]
- i < j

The idea is based on the fact the parity of inversions remains same after a set of moves, i.e., if the inversion count is odd in initial stage, then it remain odd after any sequence of moves and if the inversion count is even, then it remains even after any sequence of moves. In the goal state, there are 0 inversions. **So we can reach goal state only from a state which has even inversion count.**

When we slide a tile, we either make a row move (moving a left or right tile into the blank space), or make a column move (moving a up or down tile to the blank space).

a) A row move doesn't change the inversion count. See following example

```
1   2   3    Row Move     1   2   3

4   _   5   ---------->   _   4   5

8   6   7                8   6   7
```

Inversion count remains 2 after the move

```
1   2   3    Row Move     1   2   3

4   _   5   ---------->   4   5   _

8   6   7                8   6   7
```

Inversion count remains 2 after the move

b) A column move does one of the following three.

…..(i) Increases inversion count by 2. See following example.

```
1   2   3   Column Move     1   _   3

4   _   5   ----------->    4   2   5

8   6   7                  8   6   7
```

Inversion count increases by 2 (changes from 2 to 4)

…..(ii) Decreases inversion count by 2

```
1   3   4    Column Move     1   3   4
5   _   6   ------------>     5   2   6
7   2   8                    7   _   8
```

Inversion count decreases by 2 (changes from 5 to 3)

…..(iii) Keeps the inversion count same.

```
1   2   3    Column Move     1   2   3
4   _   5   ------------>     4   6   5
7   6   8                    7   _   8
```

Inversion count remains 1 after the move

So if a move either increases/decreases inversion count by 2, or keeps the inversion count same, then it is not possible to change parity of a state by any sequence of row/column moves.

for example:

```
tanishqmalu@TanishqMalu:/mnt/d/tanishq/4th year/ai lab/lab2$ python3 main.py "Start_State_2.txt" "End_State.txt"
Enter the type of Hueristic function:
1. h1(n) = 0, Zero heuristic
2. h2(n) = Number of tiles displaced from their destined position.
3. h3(n) = Sum of Manhattan distance of each tile from the goal
4. h4(n) = h(n) > h*(n)

4

OOPS ! The program was unable to find a solution !


Start State of the Puzzle:
T2 T8 T3
T1 B T5
T7 T6 T4


Goal State of the Puzzle:
T1 T2 T3
T8 B T4
T7 T6 T5


Total number of states explored before ending the program: 181440
```

The input state: 2831-5764 has 1 + 0 + 1 + 0 + 1 + 2 + 2 + 0 = 7 (odd number) of inversion state and hence it cannot reach the final state.

6. Observe and verify that if the cost of the empty tile is added (considering empty tile as another tile) then monotonicity will be violated.

Let us try to understand with respect to the below test case:

Start Matrix :

6	2	3
B	5	4
1	8	7

Goal Matrix :

B	2	3
6	5	4
1	8	7

Let us try to examine this for h2 and h3, For the start matrix,
h2(n) = 2
h3(n) = 2
Suppose we swap with tile 6 to take it to new state n',
then, c(6,B) = 1

Now our state n' becomes,
Calculating the heuristic score for n'

B	2	3
6	5	4
1	8	7

h2(n') = 0
h3(n') = 0
Thus the condition of,
h(n) <= c(n,n') + h(n') is not satisfied as

For h2,
2 (h2(n)) > 1 (c(n,n')) + 0 (h2(n'))

For h3,
2 (h3(n)) > 1 (c(n,n')) + 0 (h3(n'))

Question: Please make a table that should list the following for all the heuristics:

a. Total number of states explored.

b. Total number of states on optimal path.

c. Optimal path.

d. Optimal Cost of the path.

e. Total time taken for execution

Start Matrix :

6	2	3
B	5	4
1	8	7

Goal Matrix:

1	2	3
8	B	4
7	6	5

Heuristic	h1	h2	h3	h4
No. of states explored	26372	1654	510	4618
No. of states on optimal path	18	18	18	18
Optimal cost of the path	17	17	17	17
Total time taken for execution	2.88335	0.17471	0.06421	0.47322

Question: Compare and contrast between the results of all four heuristics h1(n), h2(n), h3(n), and h4(n) and state the reasons in a document file 'Why one heuristic is better than the other one?'. While explaining, please comment on the optimality, time complexity etc

Time Complexity: The time complexity for all the above heuristic is exponential in nature since A* algorithm guarantees an optimal solution but the time complexity still remains exponential in nature.

The choice of a better heuristic function helps in reducing the time complexity but the worst case complexity still remains exponential in nature.

Space Complexity: Space complexity for all the heuristic is exponential in nature.

Completeness: Except h4(n), All heuristic add bands of increasing f Unless there are infinitely many nodes with f<f(G). Thus h1(n), h2(n) and h3(n) are complete in nature.

Optimality: As we observed h*>h3 > h2 > h1, thus h1,h2 and h3 are admissible and h3 is the most optimal amongst the 3 heuristic.

For h4, since h4 > h* thus it is not admissible and thus it may give sub optimal results. Therefore h4 is not optimal in nature

h3 which is Manhattan heuristic is most suitable heuristic among the given heuristics.

—----------END—----------