# Artificial Intelligence
# Assignment 1
# N-Puzzle Problem

Yingtao Xu
yx2318

June 6th, 2016

## Introduction

The aim of this assignment is to find a solution for the n-puzzle problem (Figure 1) by designing an agent. The solution should give the path of the movement from a random board to a board in order.

In this homework, the order with zero locating on the upper left instead of on the right bottom is used.

Different searching algorithms including BFS, DFS, and A $^\star$ searching have been implemented and the results and comparison among these algorithms will be demonstrated in later sections.

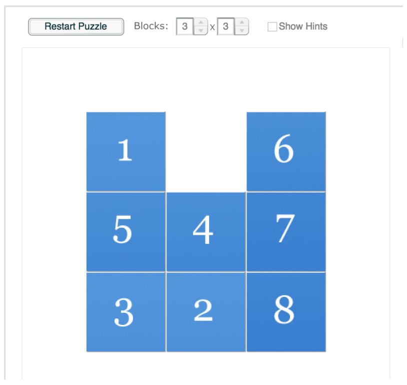


Figure 1: N-Puzzle Problem (n = 3) [1]

As mentioned in the lectures, the design of this homework is based on the class of state and then implements different search algorithms based on the class.

The state class contains different member variables including the size of puzzle, n, the total size of the board which is the power of 2 of n, the path, the current state, the goal state.

The most important function in the class is the method to expand the current node to get the child nodes or the method to find the possible next states from the current state.

There are some main design ideas:

- The total size of the board is also the total number of the possible arrangements of the board or the total possible states for a given n.

- Each state represents one arrangement of the board.

- The child states of the parent states are generated by the possible moves of the from the current zero position, that is, UP, DOWN, LEFT, RIGHT and each move will lead to one arrangement of the board, that is, one state.

- The initial state is generated randomly and the goal state is determined which is arrangement of the board in order and the position of zero locates on the upper left corner of the board.

- BFS uses queue to store the states, DFS uses stack to hold the states, and $A^\star$ uses heap (priority queue) to hold the states

- Some cases will have no solution at all will be checked first and then ignored.[2]

## Implementation of Different Searching Algorithms

### 1. BFS

BFS uses the search strategy as expanding shallowest node. The main idea to the implementation of BFS is to use the data structure queue which has the behaviour of FIFO. Namely, the nodes will be search as the order when they are put into the queue. The first node put in the queue will be searched or expanded first.

For example, in Figure 2, the root node will be push into the queue first and then the child nodes of root and then the child nodes of the child nodes, and so forth. Thus, the searching or the expanding order will be as the order in the graph.

In this homework, one state is one arrangement of the board. So BFS will start from a random arrangement of the board as its root node and then search the path to reach the goal arrangement of the board.

Detailed implementation and algorithm can be found in the codes with comments.

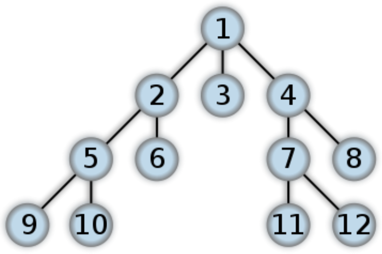


Figure 2: BFS Searching Order[3]

## 2. DFS

In contrast to BFS, DFS uses the strategy for search as expanding deepest node. Thus, the main idea to implement of DFS is to use the data structure stack which has the behaviour of LIFO, which uses different order as FIFO. That is, the last node put into the stack will be searched or explored first.

For example, Figure 3 gives the search order for DFS. It will search from the root node and then go one child node. The from that child node, it will go deeper and deeper until it finds the goal or it reach the leave. Then, it will continue search the deepest node at the bottom level and if there is no goal still it will go back to one shallower level and so forth.

Similarly, in this homework, as one arrangement represents one state or node, the search by using DFS will begin from a randomly generated node, and then this root node will be expanded if no goal found. After that, from one of the child nodes, it will keep expanding the nodes deeper and deeper until it find the goal.

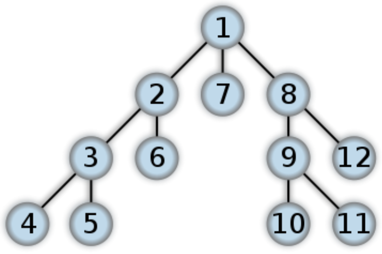


Figure 3: DFS Searching Order[4]

## 3. A$^\star$ Search

Unlike the uninformed search as BFS or DFS previously, informed search uses a heuristic function that estimates how close a state is to the goal. One of the most popular algorithm is A $^\star$ search. The main idea of this algorithm is to expand the node in consideration of the total estimated solution cost by using the evaluation function $h(n)$ (heuristic). It will always expand the node with minimum cost. That is, as the cost function $f(n)$ shown below:

$$f(n) = g(n) + h(n)$$

- g(n) is the cost to reach node n
- h(n) is the cost to get from n to the goal

In this homework, the evaluation function or the heuristic used is the total manhattan distance (the total of the horizontal and vertical distances) from the current node to the goal.

For instance, the current state for a 8-puzzle and the corresponding goal state are shown in Figure 4 below. The Manhattan distance from current 1 to the 1 in goal is the sum of 2 in vertical direction and 1 in horizontal direction so its Manhattan distance is 3. Similarly, from current 2 to the 2 in the goal state only one Manhattan distance in horizontal direction. For 3, 4, etc., the calculation is the same. Thus, the total Manhattan distance for the current state to the goal state $h(n) = 3+1+2+2+2+3+3+2 = 18$.

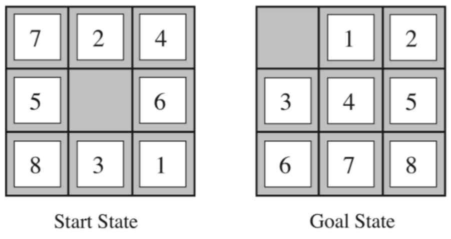


Figure 4: Example for Manhattan Distance Calculation[5]

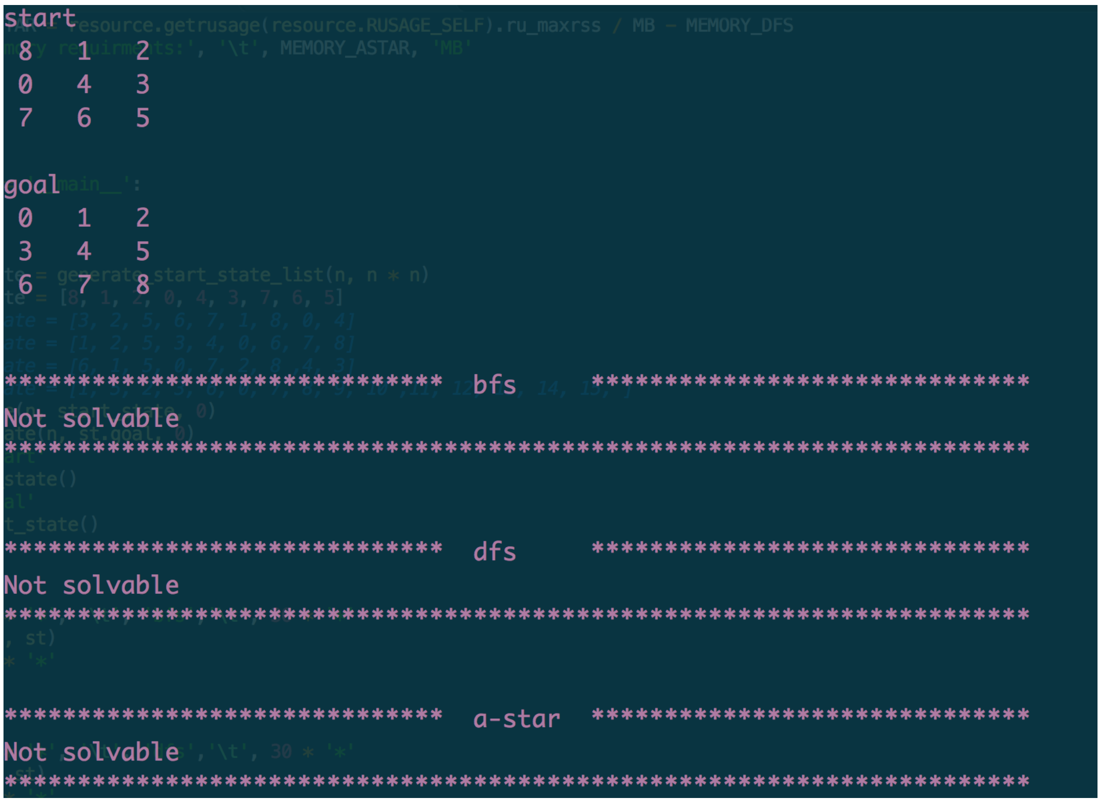
To ensure the node with the minimum sum of cost is expanded before the one with more cost, the data structure used in this part is priority queue (heap) so that whenever the head has been pop out, it will be the minimum one in that queue.

More details of the implementation can be found in the code also.

## Tests and Results

For any initial state, the puzzle solver will firstly will check wether there is a solution for the arrangement or not. If there is no solution, it will just give the answer as “ Not solvable”.

one example can be seen in Figure 5.



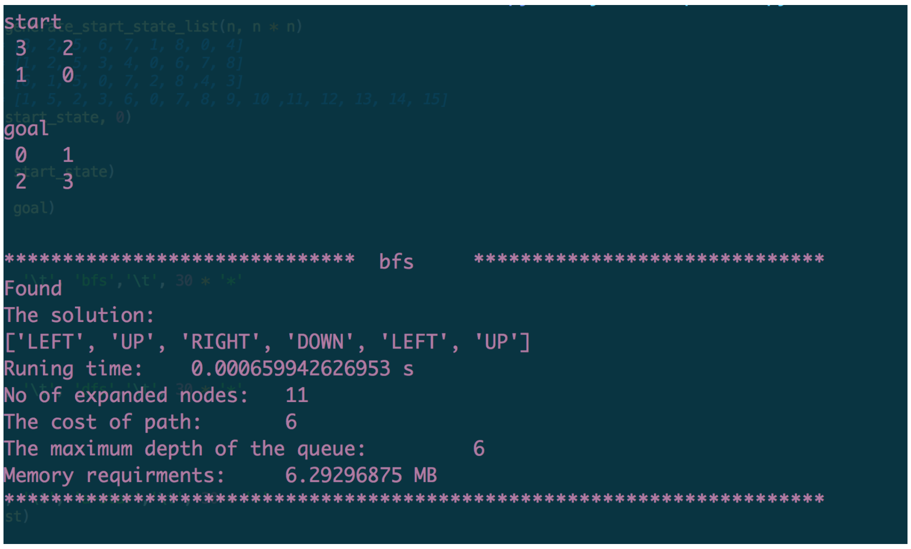
```
start
 8   1   2
 0   4   3
 7   6   5

goal
 0   1   2
 3   4   5
 6   7   8

******************************  bfs     ******************************
Not solvable
**********************************************************************

******************************  dfs     ******************************
Not solvable
**********************************************************************


******************************  a-star  ******************************
Not solvable
**********************************************************************
```

Figure 5: Example for unsolvable case

### N-Puzzel Solver as N = 2

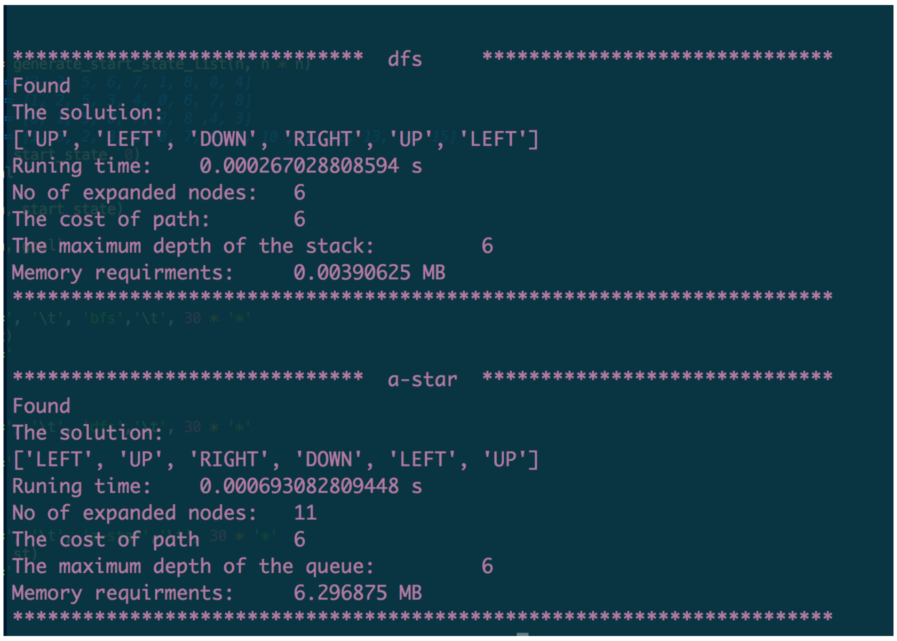
The extreme example for N = 2 is that the order of the number is totally reverse.

The results by using BFS, DFS, and A$^\star$ are shown in Figure 6 and Figure 7.



```
start
 3  2
 1  0
goal
 0  1
 2  3

******************************  bfs     ******************************
Found
The solution:
['LEFT', 'UP', 'RIGHT', 'DOWN', 'LEFT', 'UP']
Runing time:    0.000659942626953 s
No of expanded nodes:   11
The cost of path:       6
The maximum depth of the queue:         6
Memory requirments:     6.29296875 MB
**********************************************************************
```

Figure 6: Example for N-Puzzel Solver as N = 2

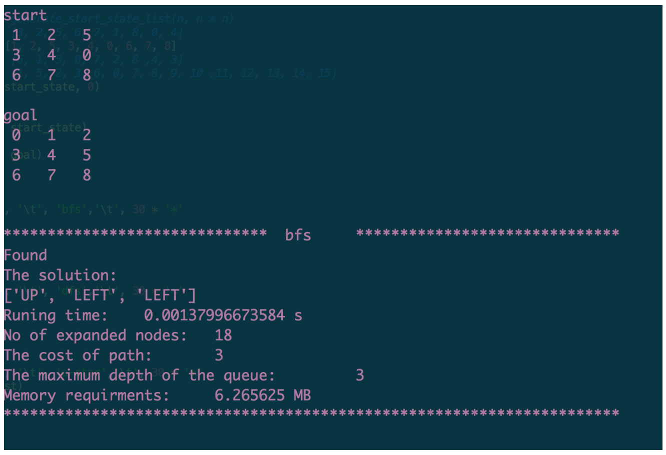


```
******************************  dfs     ******************************
Found
The solution:
['UP', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEFT']
Runing time:    0.000267028808594 s
No of expanded nodes:   6
The cost of path:       6
The maximum depth of the stack:         6
Memory requirments:     0.00390625 MB
**********************************************************************

******************************  a-star  ******************************
Found
The solution:
['LEFT', 'UP', 'RIGHT', 'DOWN', 'LEFT', 'UP']
Runing time:    0.000693082809448 s
No of expanded nodes:   11
The cost of path   6
The maximum depth of the queue:         6
Memory requirments:     6.296875 MB
**********************************************************************
```

Figure 7: Example for N-Puzzle Solver as N = 2

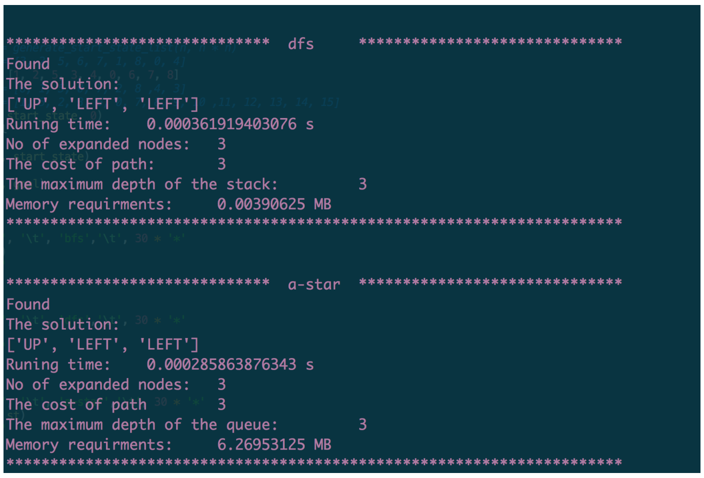
### N-Puzzel Solver as N = 3

For a simple case, as the example in the homework instruction. The results can be seen in Figure 8 and Figure 9.



```
start
 1   2   5
 3   4   0
 6   7   8

goal
 0   1   2
 3   4   5
 6   7   8


******************************  bfs     ******************************
Found
The solution:
['UP', 'LEFT', 'LEFT']
Runing time:    0.00137996673584 s
No of expanded nodes:   18
The cost of path:       3
The maximum depth of the queue:         3
Memory requirments:     6.265625 MB
**********************************************************************
```

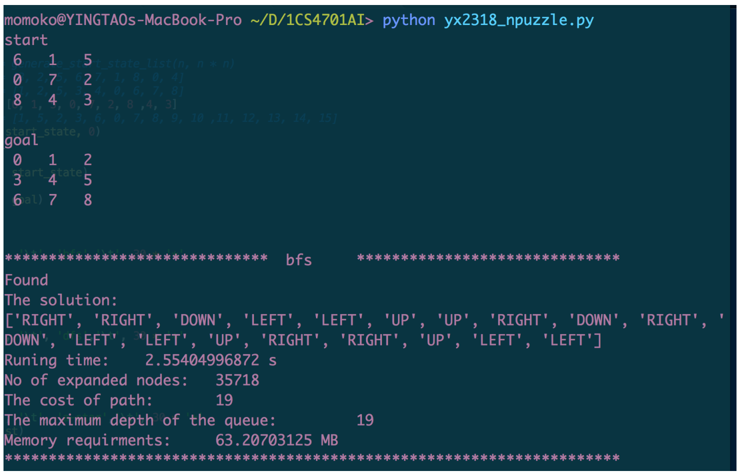
Figure 8: Simple level for N-Puzzel Solver as N = 3



```
******************************  dfs     ******************************
Found
The solution:
['UP', 'LEFT', 'LEFT']
Runing time:    0.000361919403076 s
No of expanded nodes:   3
The cost of path:       3
The maximum depth of the stack:         3
Memory requirments:     0.00390625 MB
**********************************************************************

******************************  a-star  ******************************
Found
The solution:
['UP', 'LEFT', 'LEFT']
Runing time:    0.000285863876343 s
No of expanded nodes:   3
The cost of path        3
The maximum depth of the queue:         3
Memory requirments:     6.26953125 MB
**********************************************************************
```

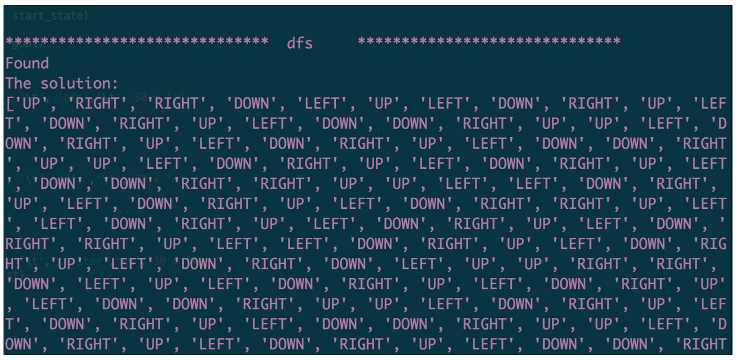
Figure 9: Simple level for N-Puzzle Solver as N = 3

For a middle level case, the results can be seen from Figure 10 to Figure 13. Since the path by DFS is too long to show, only part of the solution has been screenshot for results. For full solution please simply run the code and get the result.



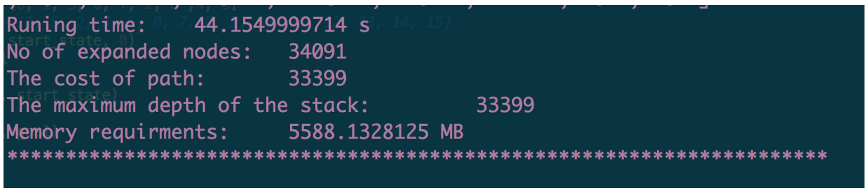
```
momoko@YINGTAOs-MacBook-Pro ~/D/1CS4701AI> python yx2318_npuzzle.py
start
 6   1   5
 0   7   2
 8   4   3
goal
 0   1   2
 3   4   5
 6   7   8


******************************  bfs     ******************************
Found
The solution:
['RIGHT', 'RIGHT', 'DOWN', 'LEFT', 'LEFT', 'UP', 'UP', 'RIGHT', 'DOWN', 'RIGHT', '
DOWN', 'LEFT', 'LEFT', 'UP', 'RIGHT', 'RIGHT', 'UP', 'LEFT', 'LEFT']
Runing time:    2.55404996872 s
No of expanded nodes:   35718
The cost of path:       19
The maximum depth of the queue:         19
Memory requirments:     63.20703125 MB
**********************************************************************
```

Figure 10: Middle level for N-Puzzel Solver as N = 3



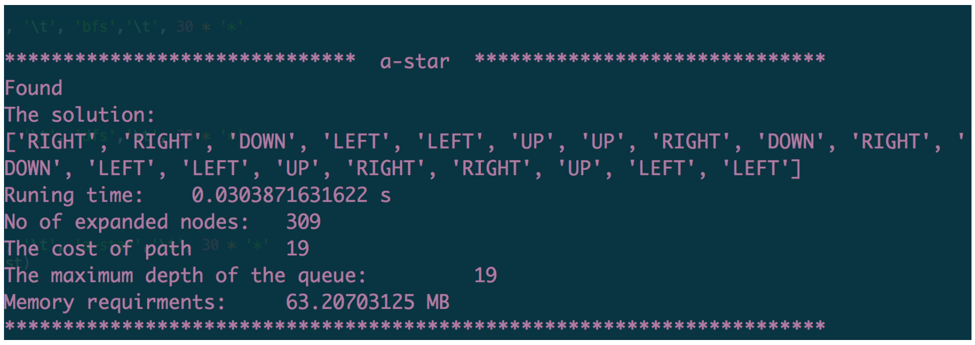
```
******************************  dfs     ******************************
Found
The solution:
['UP', 'RIGHT', 'RIGHT', 'DOWN', 'LEFT', 'UP', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEF
T', 'DOWN', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'DOWN', 'RIGHT', 'UP', 'UP', 'LEFT', 'D
OWN', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'DOWN', 'RIGHT
', 'UP', 'UP', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEFT
', 'DOWN', 'DOWN', 'RIGHT', 'RIGHT', 'UP', 'UP', 'LEFT', 'LEFT', 'DOWN', 'RIGHT',
'UP', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'RIGHT', 'RIGHT', 'UP', 'LEFT
', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEFT', 'DOWN', '
RIGHT', 'RIGHT', 'UP', 'LEFT', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'RIG
HT', 'UP', 'LEFT', 'DOWN', 'RIGHT', 'DOWN', 'LEFT', 'UP', 'UP', 'RIGHT', 'RIGHT',
'DOWN', 'LEFT', 'UP', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'RIGHT', 'UP'
, 'LEFT', 'DOWN', 'DOWN', 'RIGHT', 'UP', 'UP', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEF
T', 'DOWN', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'DOWN', 'RIGHT', 'UP', 'UP', 'LEFT', 'D
OWN', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'DOWN', 'RIGHT
```

Figure 11: Middle level for N-Puzzle Solver as N = 3



```
Runing time:   44.1549999714 s
No of expanded nodes:    34091
The cost of path:        33399
The maximum depth of the stack:          33399
Memory requirments:     5588.1328125 MB
**********************************************************************
```

Figure 12: Middle level for N-Puzzle Solver as N = 3

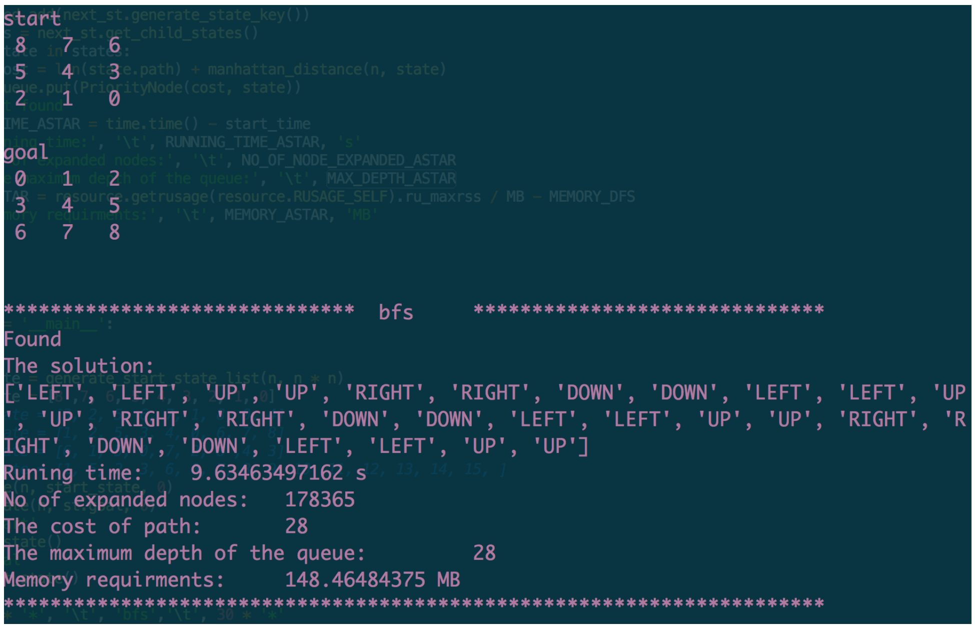


```
******************************  a-star  ******************************
Found
The solution:
['RIGHT', 'RIGHT', 'DOWN', 'LEFT', 'LEFT', 'UP', 'UP', 'RIGHT', 'DOWN', 'RIGHT', '
DOWN', 'LEFT', 'LEFT', 'UP', 'RIGHT', 'RIGHT', 'UP', 'LEFT', 'LEFT']
Runing time:    0.0303871631622 s
No of expanded nodes:   309
The cost of path        19
The maximum depth of the queue:         19
Memory requirments:     63.20703125 MB
**********************************************************************
```

Figure 13: Middle level for N-Puzzle Solver as N = 3

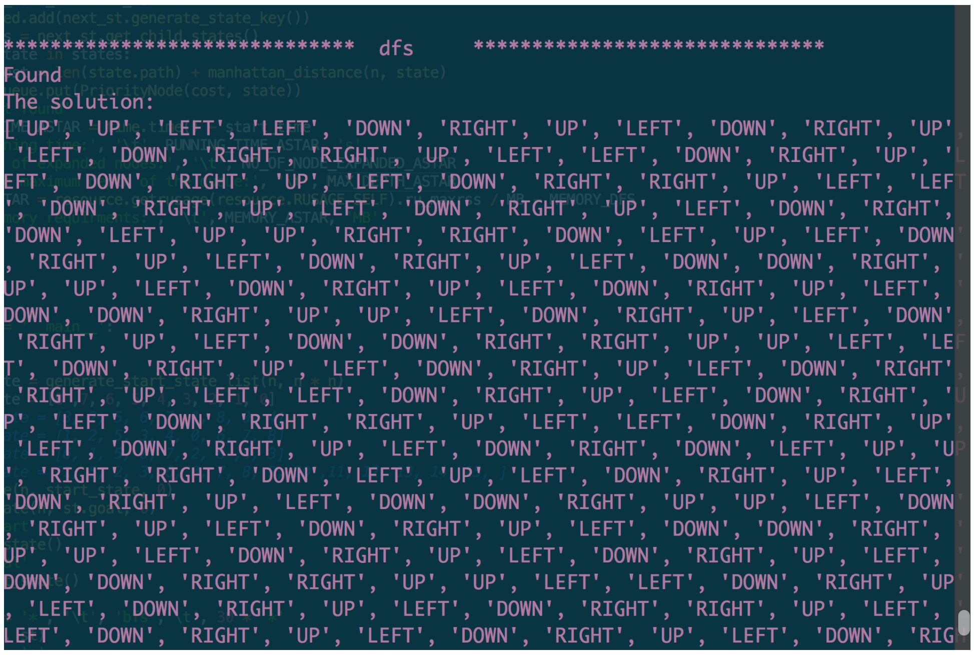
For extreme case, the results can be seen in Figure 14 to Figure 16.

Similarity, only part of the solution by DFS will be shown.



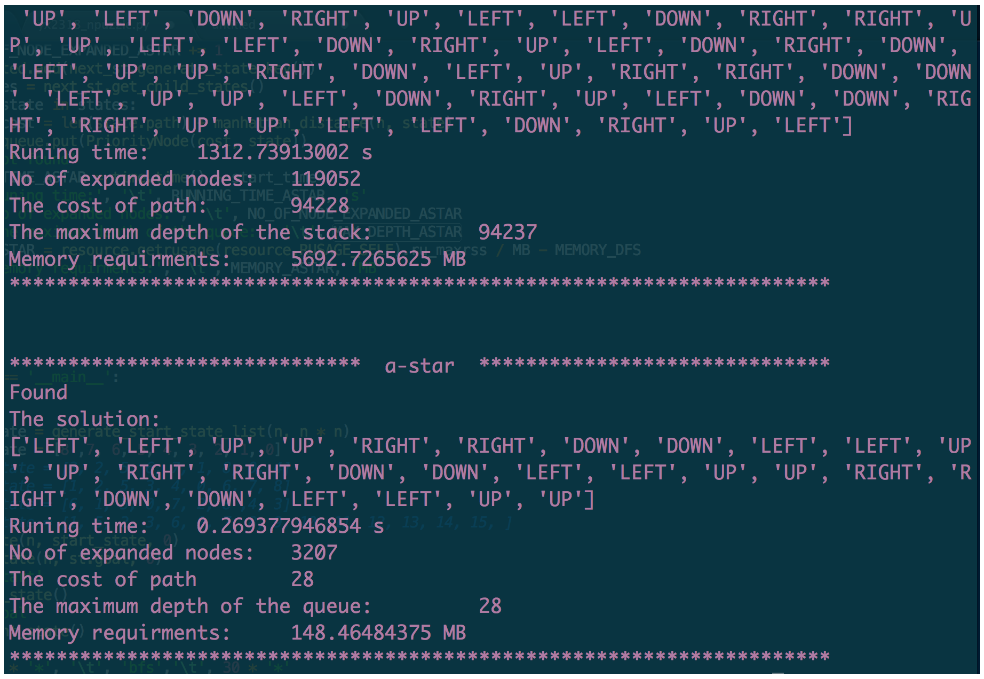
```
start
 8   7   6
 5   4   3
 2   1   0

goal
 0   1   2
 3   4   5
 6   7   8



******************************  bfs     ******************************
Found
The solution:
['LEFT', 'LEFT', 'UP', 'UP', 'RIGHT', 'RIGHT', 'DOWN', 'DOWN', 'LEFT', 'LEFT', 'UP
', 'UP', 'RIGHT', 'RIGHT', 'DOWN', 'DOWN', 'LEFT', 'LEFT', 'UP', 'UP', 'RIGHT', 'R
IGHT', 'DOWN', 'DOWN', 'LEFT', 'LEFT', 'UP', 'UP']
Runing time:    9.63463497162 s
No of expanded nodes:   178365
The cost of path:       28
The maximum depth of the queue:         28
Memory requirments:     148.46484375 MB
**********************************************************************
```

Figure 14: Extreme level for N-Puzzel Solver as N = 3



```
******************************  dfs    ******************************
Found
The solution:
['UP', 'UP', 'LEFT', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'RIGHT', 'UP',
 'LEFT', 'DOWN', 'RIGHT', 'RIGHT', 'UP', 'LEFT', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'L
EFT', 'DOWN', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'RIGHT', 'RIGHT', 'UP', 'LEFT', 'LEFT
', 'DOWN', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'RIGHT',
'DOWN', 'LEFT', 'UP', 'UP', 'RIGHT', 'RIGHT', 'DOWN', 'LEFT', 'UP', 'LEFT', 'DOWN'
, 'RIGHT', 'UP', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'DOWN', 'RIGHT', '
UP', 'UP', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEFT', '
DOWN', 'DOWN', 'RIGHT', 'UP', 'UP', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEFT', 'DOWN',
 'RIGHT', 'UP', 'LEFT', 'DOWN', 'DOWN', 'RIGHT', 'RIGHT', 'UP', 'UP', 'LEFT', 'LEF
T', 'DOWN', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'RIGHT',
 'RIGHT', 'UP', 'LEFT', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'RIGHT', 'U
P', 'LEFT', 'DOWN', 'RIGHT', 'RIGHT', 'UP', 'LEFT', 'LEFT', 'DOWN', 'RIGHT', 'UP',
 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'RIGHT', 'DOWN', 'LEFT', 'UP', 'UP
', 'RIGHT', 'RIGHT', 'DOWN', 'LEFT', 'UP', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEFT',
'DOWN', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'DOWN', 'RIGHT', 'UP', 'UP', 'LEFT', 'DOWN'
, 'RIGHT', 'UP', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'DOWN', 'RIGHT', '
UP', 'UP', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEFT', '
DOWN', 'DOWN', 'RIGHT', 'RIGHT', 'UP', 'UP', 'LEFT', 'LEFT', 'DOWN', 'RIGHT', 'UP'
, 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'RIGHT', 'RIGHT', 'UP', 'LEFT', '
LEFT', 'DOWN', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'RIG
```

Figure 15: Extreme level for N-Puzzle Solver as N = 3

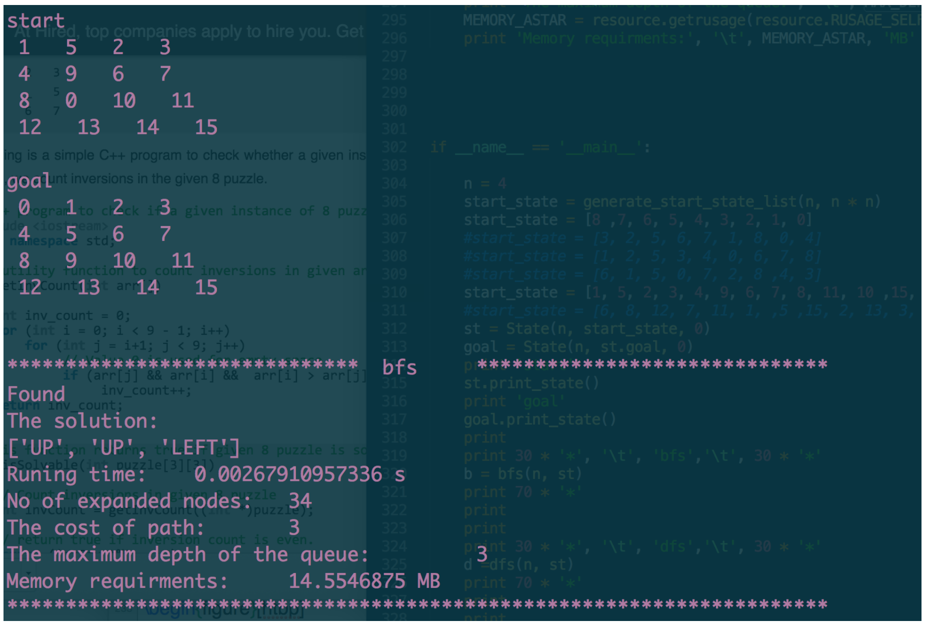


```
 'UP', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEFT', 'LEFT', 'DOWN', 'RIGHT', 'RIGHT', 'U
P', 'UP', 'LEFT', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'RIGHT', 'DOWN',
'LEFT', 'UP', 'UP', 'RIGHT', 'DOWN', 'LEFT', 'UP', 'RIGHT', 'RIGHT', 'DOWN', 'DOWN
', 'LEFT', 'UP', 'UP', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'DOWN', 'RIG
HT', 'RIGHT', 'UP', 'UP', 'LEFT', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEFT']
Runing time:    1312.73913002 s
No of expanded nodes:   119052
The cost of path:       94228
The maximum depth of the stack:         94237
Memory requirments:     5692.7265625 MB
**************************************************************



******************************  a-star  ******************************
Found
The solution:
['LEFT', 'LEFT', 'UP', 'UP', 'RIGHT', 'RIGHT', 'DOWN', 'DOWN', 'LEFT', 'LEFT', 'UP
', 'UP', 'RIGHT', 'RIGHT', 'DOWN', 'DOWN', 'LEFT', 'LEFT', 'UP', 'UP', 'RIGHT', 'R
IGHT', 'DOWN', 'DOWN', 'LEFT', 'LEFT', 'UP', 'UP']
Runing time:    0.269377946854 s
No of expanded nodes:   3207
The cost of path        28
The maximum depth of the queue:         28
Memory requirments:     148.46484375 MB
**************************************************************
```

Figure 16: Extreme level for N-Puzzle Solver as N = 3

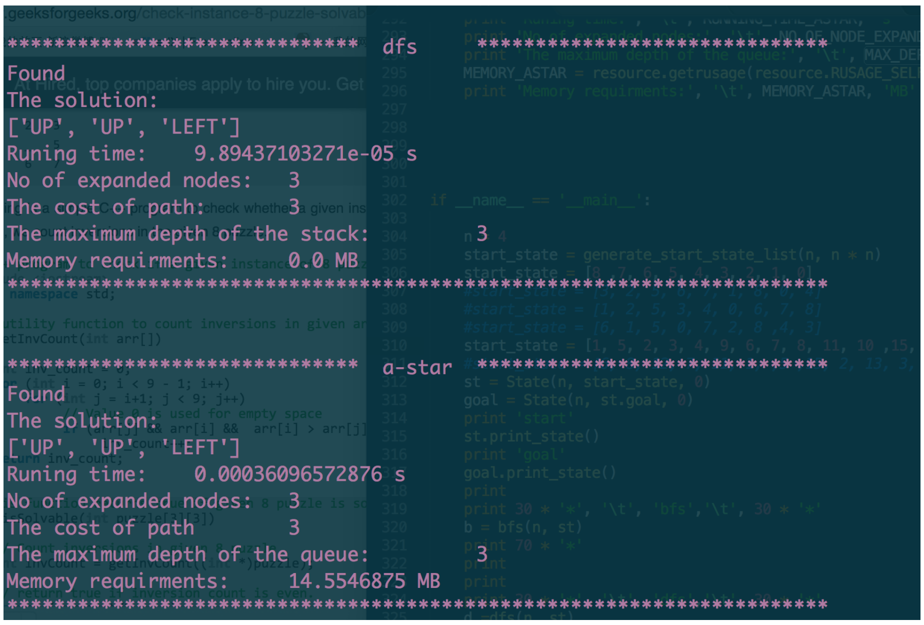
## N-Puzzel Solver as N = 4

The results for one simple case can be seen in Figure 17 and Figure 18. Since the time for solving the extreme case is relatively long for BFS and DFS (larger than two hours) and there is little time for me to wait for finish, I only tested A⋆. Also, the reverse order is not solvable which can be seen in Figure 19 so I chose a randomly ordered case for test. Even using A*, the time cost memory needed is very large. The result can be found in Figure 20.



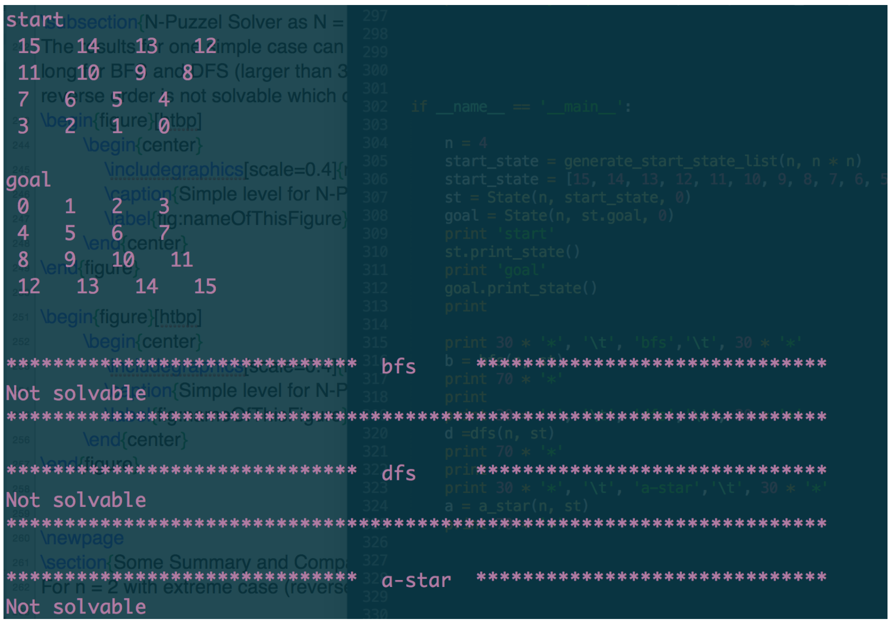
```
start
 1   5   2   3
 4   9   6   7
 8   0   10   11
 12   13   14   15

goal
 0   1   2   3
 4   5   6   7
 8   9   10   11
 12   13   14   15
******************************  bfs  ******************************
Found
The solution:
['UP', 'UP', 'LEFT']
Runing time:    0.00267910957336 s
No of expanded nodes:  34
The cost of path:     3
The maximum depth of the queue:     3
Memory requirments:     14.5546875 MB
**********************************************************************
```

Figure 17: Simple level for N-Puzzle Solver as N = 4



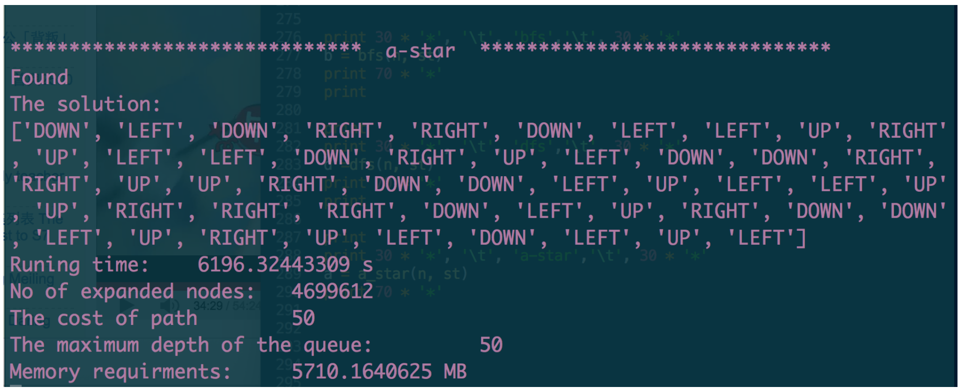
```
******************************  dfs    ******************************
Found
The solution:
['UP', 'UP', 'LEFT']
Runing time:    9.89437103271e-05 s
No of expanded nodes:   3
The cost of path:       3
The maximum depth of the stack:   3
Memory requirments:     0.0 MB
**********************************************************************

******************************  a-star ******************************
Found
The solution:
['UP', 'UP', 'LEFT']
Runing time:    0.00036096572876 s
No of expanded nodes:   3
The cost of path        3
The maximum depth of the queue:   3
Memory requirments:     14.5546875 MB
**********************************************************************
```

Figure 18: Simple level for N-Puzzle Solver as N = 4



```
start
 15   14   13   12
 11   10   9    8
 7    6    5    4
 3    2    1    0

goal
 0    1    2    3
 4    5    6    7
 8    9    10   11
 12   13   14   15

******************************  bfs  ******************************
Not solvable
**********************************************************************

******************************  dfs  ******************************
Not solvable
**********************************************************************

******************************  a-star  ******************************
Not solvable
```

Figure 19: Unsolvable extreme level for N-Puzzle Solver as N = 4



```
*****************************  a-star  *****************************
Found
The solution:
['DOWN', 'LEFT', 'DOWN', 'RIGHT', 'RIGHT', 'DOWN', 'LEFT', 'LEFT', 'UP', 'RIGHT'
, 'UP', 'LEFT', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'DOWN', 'RIGHT',
'RIGHT', 'UP', 'UP', 'RIGHT', 'DOWN', 'DOWN', 'LEFT', 'UP', 'LEFT', 'LEFT', 'UP'
, 'UP', 'RIGHT', 'RIGHT', 'RIGHT', 'DOWN', 'LEFT', 'UP', 'RIGHT', 'DOWN', 'DOWN'
, 'LEFT', 'UP', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'LEFT', 'UP', 'LEFT']
Runing time:    6196.32443309 s
No of expanded nodes:   4699612
The cost of path        50
The maximum depth of the queue:         50
Memory requirments:     5710.1640625 MB
```

Figure 20: A random case using A$^\star$ only for N-Puzzle Solver as N = 4

## Some Summary and Comparison in Table From the Results

For n = 2 with extreme case (reverse order):

| Comparisons of Diffrent Implementation | | | | | |
|---|---|---|---|---|---|
| | Cost of path | Expanded nodes | Max depth | Memory(MB) | Running time (ms) |
| BFS | 6 | 11 | 6 | $\approx 6.29$ | $\approx 0.6$ |
| DFS | 6 | 6 | 6 | $\approx 0.004$ | $\approx 0.27$ |
| A$^\star$ | 6 | 11 | 6 | $\approx 6.3$ | $\approx 0.69$ |

n = 3 with extreme case (reverse order):

| Comparisons of Diffrent Implementation | | | | | |
|---|---|---|---|---|---|
| | Cost of path | Expanded nodes | Max depth | Memory(MB) | Running time(s) |
| BFS | 28 | 178365 | 28 | $\approx 148.46$ | $\approx 9.635$ |
| DFS | 94228 | 119052 | 94237 | $\approx 5692.73$ | $\approx 1312.739$ |
| A$^\star$ | 28 | 3207 | 28 | $\approx 148.46$ | $\approx 0.269$ |

## Justification of the choice of heuristic

In this homework, I have chosen Manhattan distance as the heuristic. Actually, any heuristic is acceptable as long as it is admissible heuristic. For an admissible heuristic ($h(n)$), it should satisfy the inequality below:

$$0 \leq h(n) \leq h(n^*)$$

, where $h(n^*)$ is the true cost to reach the goal from n.

For n-puzzle board, the minimum distance to move any number from one point to anther point is always the sum of the horizontal and the vertical distance, which is equal to the Manhattan distance. Thus, Manhattan distance will always give the cost less than or equal to the true cost to reach the goal state from any state. Thus, the choice of the heuristic is admissible and acceptable.

## Discussion

From the results obtained form the homework, I have obtained the main features of uninformed and informed search. For uninformed search, no matter for BFS or DFS, the brute force is actually the truth and the core. If there is a solution, BFS and DFS will always find the solution but in a really inefficient way because they will explore all possible states until they find the goal. Thus, BFS and DFS will cost much unnecessary time. Judging from the results of this homework, when N is 2, the performance of BFS and DFS is similar. However, as N becomes larger than 2, it seems DFS performs less better than BFS in terms of both time cost and memory needed.

In contrast to uninformed search, informed search performs much better. The results obtained from A$^\star$ give a very obvious comparison. Especially, the time cost for the A$^\star$ is much less than BFS and DFS. When N is larger than 3, the time cost for both BFS and DFS significantly increases the increase of A$^\star$ is very less. For instance, for the extreme case of N = 3, the time cost for BFS is around 10 seconds and the time cost for DFS is even worse which needs around 20 mins but the time cost for A$^\star$ is just around 20ms. The difference is huge.

However, the memory needed for A$^\star$ has no significant difference than BFS. So one of the possible methods to improve is to implement IDA$^\star$, which is still complete and optimal but needs less memory. Due to the time limit, IDA$^\star$has not be implemented but after the homework, it will be tested.

# Bibliography

[1] N-Puzzel Problem: http://mypuzzle.org/sliding

[2] Check unsolvable cases: http://www.geeksforgeeks.org/check-instance-8-puzzle-solvable/

[3] BFS: https://en.wikipedia.org/wiki/Breadth-first_search#/media/File:Breadth-first-tree.svg

[4] DFS: https://en.wikipedia.org/wiki/Depth-first_search#/media/File:Depth-first-tree.svg

[5] Manhattan Distance from Lecture notes: https://courseworks2.columbia.edu/courses/7069/files/folder/lectures?preview=503027