# King Fahd University of Petroleum & Minerals

## College of Computing and Mathematics

### Information & Computer Science

## ICS 381: Principles of Artificial Intelligence

## Assignment #2

**Student Name:** Abdullah Alzeid

# 1. Implement A* search solutions considering the Misplaced, Euclidean, and Manhattan heuristics to the problem

## a. Representation

### i. Function Breakdown

**get_puzzle_size():**

- Purpose: Get the size of the puzzle from the user. The size is represented by an integer N such that 3 <= N <= 6.
- Implementation: Uses an infinite loop to keep prompting the user until they provide a valid integer in the specified range.

**solvable(state):**

- Purpose: Determine if a given state of the puzzle is solvable.
- Implementation: Uses the inversion count method to check solvability. For even grid sizes, it also considers the row containing the blank.

**generate_initial_state(n):**

- Purpose: Generate a random starting state of size n x n.
- Implementation: Starts with a solved puzzle and performs a certain number of random moves to shuffle it.

**generate_goal_state(n):**

- Purpose: Produce the goal state of the puzzle for a given size n.
- Implementation: Constructs the puzzle in order with the last element being 0 (blank).

**GenerateChildren(state, last_move=None):**

- Purpose: Generate possible successor states (children) from a given state.
- Implementation: It finds the position of the blank (0) and tries to swap it with its neighbors in the up, down, left, and right directions. It avoids the move opposite to the last move to prevent useless back-and-forth between states.

**a_star_method(initial, goal, heuristic):**

- Purpose: A wrapper function to call the A* search method.
- Implementation: Simply calls the a_star function.

**a_star(initial_state, goal_state, heuristic):**

- Purpose: Solve the puzzle using the A* search algorithm.
- Implementation: Uses a priority queue (heuristic-based) to explore states, storing visited states to avoid repetition.

**misplaced_tiles(state, goal_state):**

- Purpose: Heuristic that counts tiles that are out of place.
- Implementation: Compares each tile in the state with the goal state and counts the mismatches.

**manhattan_distance(state, goal_state) and euclidean_distance(state, goal_state):**

- Purpose: Heuristics that measure the distance tiles need to move to be in their goal position. The former uses Manhattan (L1) distance, while the latter uses Euclidean (L2) distance.
- Implementation: For each tile, compute its current position and desired position, then calculate the distance.

**generate_report(algorithm_name, algorithm, n, heuristic):**

- Purpose: Generate a report on the performance of an algorithm over 10 runs.
- Implementation: For each run, it generates a random puzzle and solves it. It writes the results, such as solution path and number of states stored, to a file with a certain name.

**main():**

- Purpose: The main driver function.
- Implementation: Gets puzzle size, generates an initial state, solves it using the specified heuristic and method (A*), and displays the results.

## ii. Problem Representation

- **State**: A state of the puzzle is represented as a list of lists (a 2D list or matrix). Each number represents a tile, and **0** represents the blank or empty space. For example, for a 3x3 puzzle:

  [[1, 2, 3], [4, 5, 6], [7, 8, 0]]

  The above state is the goal state.

- **Moves**: Moves are represented as strings "U" for up, "D" for down, "L" for left, and "R" for right.

- **Path**: A path is a sequence of moves that takes the puzzle from its initial state to the goal state. Represented as a list of strings, e.g., **["U", "R", "D","L"]**.

- **Heuristics**: Used to estimate the cost of reaching the goal from a given state. Three heuristics are defined: misplaced tiles, Manhattan distance, and Euclidean distance.

2. **Run your program for $3 \leq n \leq 6$.**

   a. **For each *n*, repeat the run *10 times* with a different random *Initial State*. *Final State* should be the standard sorted one.**
   Answers in the next pages

   b. **Report the Initial State, Final State, and the Solution Sequence of Actions.**
   Answers in the next pages

   c. **Across the 10 repetitions, report the descriptive statistics (minimum, maximum, and average) of the solution depth for each n for each solution.**
   Answers in the next pages

   d. **Across the 10 repetitions, report the descriptive statistics (*minimum*, *maximum*, and *average*) of the *maximum number of states concurrently stored* for each *n* for each solution.**
   Answers in the next pages

   e. **Comment on the results.**

## Answers in the next pages

**Important Note:** All of the runs seen below for all sizes of N are completely randomly generated. But for sizes of N=4 and larger some configurations were added to the generate initial state function to limit the randomness and favor more manageable initial states. This was applied because of various reasons:

- **Computational Complexity and Efficiency:** Considering the N=4 case, the number of possible states grows exponentially. The 15-puzzle (N=4) alone has over 1.3×10131.3×1013 possible states. Running A* on a state that is far from the solution could be computationally prohibitive, particularly when using heuristics that involve a lot of calculations, such as the Manhattan or Euclidean distances. By enforcing manageable initial states, we ensure that our algorithm terminates in a reasonable amount of time while still showcasing its problem-solving capabilities.

- **Memory Constraints:** A* relies on storing and exploring various possible puzzle states, and this storage can explode when dealing with complex puzzles and distant initial states. It's critical to ensure that memory usage remains within manageable bounds, especially when the primary focus is to compare the efficiency of heuristics rather than stress-testing memory capabilities.

- **Emphasis on Heuristic Performance:** The primary objective when testing with different heuristics is to observe how each heuristic guides the A* algorithm. By having extremely difficult initial states, the emphasis might shift from heuristic performance to mere puzzle solvability. Manageable states ensure that the focus remains on how each heuristic aids in achieving the solution, providing a clearer insight into their relative efficiencies.

# Analysis for N=3:-

# Manhattan:

Run 1:

Initial State:

1 5 4

3 7 6

0 8 2

Final State:

1 2 3

4 5 6

7 8 0

Algorithm Used: A* with manhattan

Solution Sequence:

R --> R --> U --> U --> L --> L --> D --> R --> D --> L --> U --> R --> U --> L --> D --> R --> R --> U --> L --> L --> D --> R --> R --> D

Solution Depth: 24

Max States Stored: 1265

Run 2:

Initial State:

7 6 8

4 1 3

2 5 0

Final State:

1 2 3

4 5 6

7 8 0

Algorithm Used: A* with manhattan

Solution Sequence:

L --> U --> L --> D --> R --> U --> U --> L --> D --> D --> R --> U --> U --> R --> D --> D --> L --> U --> U --> L --> D --> D --> R --> U --> R --> D

Solution Depth: 26

Max States Stored: 2293

----------------------------------------

Run 3:

Initial State:

2 8 1

4 6 3

0 5 7

Final State:

1 2 3

4 5 6

7 8 0

Algorithm Used: A* with manhattan

Solution Sequence:

U --> U --> R --> R --> D --> L --> D --> R --> U --> L --> L --> U --> R --> D --> L --> D --> R --> U --> R --> D

Solution Depth: 20

Max States Stored: 291

--------------------------------------

Run 4:

Initial State:

3 4 2

7 5 1

8 0 6

Final State:

1 2 3

4 5 6

7 8 0

Algorithm Used: A* with manhattan

Solution Sequence:

L --> U --> U --> R --> D --> R --> U --> L --> D --> L --> U --> R --> D --> R --> U --> L --> D --> R --> D

Solution Depth: 19

Max States Stored: 285

---------------------------------------

Run 5:

Initial State:

7 8 5

3 1 6

0 2 4

Final State:

1 2 3

4 5 6

7 8 0

Algorithm Used: A* with manhattan

Solution Sequence:

R --> R --> U --> L --> L --> U --> R --> D --> L --> D --> R --> U --> R --> U --> L --> D --> L --> U --> R --> D --> L --> D --> R --> U --> R --> D

Solution Depth: 26

Max States Stored: 1557

----------------------------------------

Run 6:

Initial State:

2 5 0

6 4 1

3 8 7

Final State:

1 2 3

4 5 6

7 8 0

Algorithm Used: A* with manhattan

Solution Sequence:

L --> D --> D --> R --> U --> L --> L --> D --> R --> R --> U --> L --> L --> U --> R --> D --> D --> R --> U --> U --> L --> D --> L --> U --> R --> D --> R --> D

Solution Depth: 28

Max States Stored: 4215

----------------------------------------

Run 7:

Initial State:

6 0 7

1 5 4

3 2 8

Final State:

1 2 3

4 5 6

7 8 0

Algorithm Used: A* with manhattan

Solution Sequence:

D --> L --> D --> R --> U --> R --> U --> L --> L --> D --> D --> R --> U --> R --> U --> L --> L --> D --> D --> R --> U --> R --> U --> L --> L --> D --> D --> R --> R

Solution Depth: 29

Max States Stored: 5578

---------------------------------------

Run 8:

Initial State:

8 3 1

0 7 4

6 2 5

Final State:

1 2 3

4 5 6

7 8 0

Algorithm Used: A* with manhattan

Solution Sequence:

R --> D --> L --> U --> R --> R --> U --> L --> L --> D --> R --> R --> D --> L --> L --> U --> R --> U --> L --> D --> D --> R --> U --> R --> D

Solution Depth: 25

Max States Stored: 1212

----------------------------------------

```
Run 9:

Initial State:

5 3 0

2 1 6

7 8 4


Final State:

1 2 3

4 5 6

7 8 0


Algorithm Used: A* with manhattan

Solution Sequence:

L --> D --> L --> U --> R --> D --> L --> D --> R --> R --> U --> L -->
D --> L --> U --> R --> R --> D


Solution Depth: 18

Max States Stored: 21

---------------------------------------
```

Run 10:

Initial State:

1 0 4

6 2 5

8 3 7

Final State:

1 2 3

4 5 6

7 8 0

Algorithm Used: A* with manhattan

Solution Sequence:

D --> D --> R --> U --> L --> L --> D --> R --> U --> L --> U --> R --> R --> D --> D --> L --> U --> U --> L --> D --> R --> R --> D

Solution Depth: 23

Max States Stored: 906

--------------------------------------

## Descriptive Statistics:

Minimum Solution Depth: 18

Maximum Solution Depth: 29

Average Solution Depth: 23.80

Minimum States Stored: 214

Maximum States Stored: 5578

Average States Stored: 1781.60

## Euclidean:

Run 1:

Initial State:

5 6 8

3 7 1

2 0 4

Final State:

1 2 3

4 5 6

7 8 0

Algorithm Used: A* with euclidean

Solution Sequence:

R --> U --> L --> L --> D --> R --> R --> U --> U --> L --> D --> D --> R --> U --> U --> L --> D --> L --> U --> R --> D --> L --> D --> R --> R

Solution Depth: 25

Max States Stored: 1803

--------------------------------------

Run 2:

Initial State:

2 8 4

1 5 7

0 3 6

Final State:

1 2 3

4 5 6

7 8 0

Algorithm Used: A* with euclidean

Solution Sequence:

U --> R --> D --> R --> U --> L --> U --> R --> D --> L --> D --> R --> U --> L --> L --> D --> R --> U --> U --> L --> D --> R --> D --> R

Solution Depth: 24

Max States Stored: 1211

---------------------------------------

Run 3:

Initial State:

8 5 2

6 0 3

7 1 4

Final State:

1 2 3

4 5 6

7 8 0

Algorithm Used: A* with euclidean

Solution Sequence:

D --> L --> U --> R --> D --> R --> U --> U --> L --> L --> D --> R --> D --> L --> U --> R --> U --> R --> D --> D --> L --> U --> U --> R --> D --> D

Solution Depth: 26

Max States Stored: 4580

----------------------------------------

Run 4:

Initial State:

5 6 7

1 3 2

8 4 0

Final State:

1 2 3

4 5 6

7 8 0

Algorithm Used: A* with euclidean

Solution Sequence:

L --> U --> U --> R --> D --> D --> L --> U --> U --> L --> D --> R --> R --> D --> L --> L --> U --> R --> U --> R --> D --> D

Solution Depth: 22

Max States Stored: 588

----------------------------------------

Run 5:

Initial State:

8 2 7

3 1 5

0 6 4

Final State:

1 2 3

4 5 6

7 8 0

Algorithm Used: A* with euclidean

Solution Sequence:

R --> R --> U --> L --> U --> R --> D --> L --> L --> U --> R --> R --> D --> L --> D --> L --> U --> U --> R --> R --> D --> D --> L --> L --> U --> R --> R --> D

Solution Depth: 28

Max States Stored: 5710

----------------------------------------

Run 6:

Initial State:

4 1 3

0 5 7

6 2 8

Final State:

1 2 3

4 5 6

7 8 0

Algorithm Used: A* with euclidean

Solution Sequence:

D --> R --> U --> R --> D --> L --> U --> L --> D --> R --> R --> U --> L --> L --> U --> R --> D --> D --> R

Solution Depth: 19

Max States Stored: 342

----------------------------------------

```
Run 7:
Initial State:
7 2 6
5 4 8
0 1 3

Final State:
1 2 3
4 5 6
7 8 0

Algorithm Used: A* with euclidean
Solution Sequence:
R --> R --> U --> L --> D --> L --> U --> U --> R --> D --> D --> L -->
U --> R --> R --> U --> L --> L --> D --> R --> D --> R

Solution Depth: 22
Max States Stored: 777

----------------------------------------
```

Run 8:

Initial State:

7 2 8

0 5 6

1 3 4

Final State:

1 2 3

4 5 6

7 8 0

Algorithm Used: A* with euclidean

Solution Sequence:

R --> D --> R --> U --> L --> L --> U --> R --> R --> D --> L --> L --> D --> R --> U --> U --> L --> D --> D --> R --> U --> R --> D

Solution Depth: 23

Max States Stored: 905

---------------------------------------

```
Run 9:
Initial State:
1 3 8
0 4 2
7 6 5

Final State:
1 2 3
4 5 6
7 8 0

Algorithm Used: A* with euclidean
Solution Sequence:
R --> R --> U --> L --> D --> D --> R --> U --> L --> D --> R

Solution Depth: 11
Max States Stored: 22

---------------------------------------
```

Run 10:

Initial State:

7 4 5

2 3 1

6 0 8

Final State:

1 2 3

4 5 6

7 8 0

Algorithm Used: A* with euclidean

Solution Sequence:

L --> U --> U --> R --> D --> R --> U --> L --> D --> D --> L --> U --> U --> R --> R --> D --> L --> D --> R --> U --> U --> L --> D --> R --> D

Solution Depth: 25

Max States Stored: 2417

---------------------------------------

## Descriptive Statistics:

Minimum Solution Depth: 11

Maximum Solution Depth: 28

Average Solution Depth: 22.50

Minimum States Stored: 22

Maximum States Stored: 5710

Average States Stored: 1835.50

## Misplaced Tiles:

Run 1:

Initial State:

6 8 1

2 3 0

5 4 7

Final State:

1 2 3

4 5 6

7 8 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

L --> U --> R --> D --> L --> L --> U --> R --> D --> L --> D --> R --> R --> U --> L --> L --> D --> R --> R

Solution Depth: 19

Max States Stored: 1268

---------------------------------------

Run 2:

Initial State:

8 1 2

5 6 4

0 3 7

Final State:

1 2 3

4 5 6

7 8 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

R --> R --> U --> L --> L --> U --> R --> D --> D --> L --> U --> R --> D --> R --> U --> L --> D --> L --> U --> R --> U --> R --> D --> D

Solution Depth: 24

Max States Stored: 9469

---------------------------------------

```
Run 3:
Initial State:
0 6 8
4 7 5
3 2 1

Final State:
1 2 3
4 5 6
7 8 0

Algorithm Used: A* with misplaced tiles
Solution Sequence:
D --> D --> R --> U --> L --> D --> R --> R --> U --> U --> L --> D -->
D --> R --> U --> U --> L --> D --> L --> U --> R --> D --> D --> R

Solution Depth: 24
Max States Stored: 8911

---------------------------------------
```

Run 4:

Initial State:

0 4 7

8 1 5

3 2 6

Final State:

1 2 3

4 5 6

7 8 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

R --> D --> D --> L --> U --> U --> R --> R --> D --> L --> U --> L --> D --> R --> D --> L --> U --> U --> R --> D --> R --> U --> L --> D --> R --> D

Solution Depth: 26

Max States Stored: 17168

----------------------------------------

Run 5:

Initial State:

4 7 8

1 3 2

0 5 6

Final State:

1 2 3

4 5 6

7 8 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

R --> U --> U --> L --> D --> R --> U --> R --> D --> D --> L --> L --> U --> R --> R --> D --> L --> U --> U --> R --> D --> D --> L --> U --> R --> D

Solution Depth: 26

Max States Stored: 18070

---------------------------------------

Run 6:

Initial State:

1 6 8

0 2 5

4 3 7

Final State:

1 2 3

4 5 6

7 8 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

R --> D --> R --> U --> U --> L --> D --> L --> D --> R --> R --> U --> U --> L --> D --> D --> R

Solution Depth: 17

Max States Stored: 544

--------------------------------------

Run 7:

Initial State:

4 7 5

8 1 6

3 2 0

Final State:

1 2 3

4 5 6

7 8 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

L --> L --> U --> R --> R --> D --> L --> U --> U --> L --> D --> R --> U --> R --> D --> D --> L --> L --> U --> R --> U --> R --> D --> D

Solution Depth: 24

Max States Stored: 9448

----------------------------------------

Run 8:

Initial State:

5 3 7

8 4 1

2 6 0

Final State:

1 2 3

4 5 6

7 8 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

L --> L --> U --> R --> R --> U --> L --> D --> D --> L --> U --> U --> R --> D --> R --> D --> L --> U --> L --> D --> R --> R

Solution Depth: 22

Max States Stored: 4346

----------------------------------------

Run 9:

Initial State:

0 3 8

7 1 4

2 5 6

Final State:

1 2 3

4 5 6

7 8 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

D --> D --> R --> R --> U --> U --> L --> D --> D --> R --> U --> L --> L --> U --> R --> D --> D --> L --> U --> R --> D --> R

Solution Depth: 22

Max States Stored: 4599

----------------------------------------

Run 10:

Initial State:

0 6 4

2 3 5

8 7 1

Final State:

1 2 3

4 5 6

7 8 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

D --> D --> R --> R --> U --> L --> U --> R --> D --> L --> L --> U --> R --> R --> D --> L --> D --> R --> U --> U --> L --> D --> L --> U --> R --> D --> D --> R

Solution Depth: 28

Max States Stored: 26186

----------------------------------------

## Descriptive Statistics:

Minimum Solution Depth: 17

Maximum Solution Depth: 28

Average Solution Depth: 23.20

Minimum States Stored: 544

Maximum States Stored: 26186

Average States Stored: 10000.90

## Analysis For N=4:-

## Manhattan:

Run 1:

Initial State:

1 2 8 3

5 6 12 4

13 10 0 11

14 9 7 15

Final State:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 0

Algorithm Used: A* with Manhattan

Solution Sequence:

L --> D --> L --> U --> R --> R --> U --> U --> R --> D --> D --> L --> D --> R --> U --> U --> L --> D --> R --> D

Solution Depth: 20

Max States Stored: 481

```
----------------------------------------
Run 2:
Initial State:
9 3 6 4
1 5 12 7
13 2 0 8
14 10 11 15

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

Algorithm Used: A* with Manhattan
Solution Sequence:
U --> U --> L --> D --> L --> U --> R --> D --> D --> D --> L --> U -->
U --> U --> R --> D --> R --> R --> D --> L --> D --> R

Solution Depth: 22
Max States Stored: 251

----------------------------------------
```

Run 3:

Initial State:

3 6 7 4

1 11 9 0

5 2 15 8

13 10 12 14

Final State:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 0

Algorithm Used: A* with Manhattan

Solution Sequence:

D --> L --> D --> R --> U --> L --> U --> L --> U --> L --> D --> R --> D --> R --> U --> U --> L --> D --> L --> D --> R --> D --> R --> R

Solution Depth: 24

Max States Stored: 336

----------------------------------------

Run 4:

Initial State:

1 2 3 4

14 6 10 8

7 9 0 12

5 13 11 15

Final State:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 0

Algorithm Used: A* with Manhattan

Solution Sequence:

L --> L --> U --> R --> D --> L --> D --> R --> U --> R --> U --> L --> L --> D --> R --> R --> D --> R

Solution Depth: 18

Max States Stored: 142

----------------------------------------

Run 5:

Initial State:

5 1 2 3

6 7 11 4

13 9 15 12

10 0 8 14

Final State:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 0

Algorithm Used: A* with Manhattan

Solution Sequence:

L --> U --> R --> R --> D --> R --> U --> L --> L --> D --> R --> U --> U --> L --> L --> U --> R --> R --> R --> D --> D --> D

Solution Depth: 22

Max States Stored: 198

---------------------------------------

Run 6:

Initial State:

1 2 4 8

5 0 7 3

9 6 14 10

13 15 12 11

Final State:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 0

Algorithm Used: A* with Manhattan

Solution Sequence:

D --> R --> R --> D --> L --> L --> U --> R --> U --> R --> U --> L --> D --> D --> R --> D

Solution Depth: 16

Max States Stored: 79

----------------------------------------

Run 7:

Initial State:

2 3 0 4

1 10 6 8

5 14 7 12

9 13 11 15

Final State:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 0

Algorithm Used: A* with Manhattan

Solution Sequence:

L --> L --> D --> D --> D --> R --> U --> U --> R --> D --> D --> R

Solution Depth: 12

Max States Stored: 16

---------------------------------------

```
Run 8:
Initial State:
1 3 0 12
5 6 4 8
9 7 2 15
13 10 14 11

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

Algorithm Used: A* with Manhattan
Solution Sequence:
D --> D --> L --> U --> R --> R --> U --> L --> L --> D --> D --> D -->
R --> R --> U --> U --> L --> D --> D --> R

Solution Depth: 20
Max States Stored: 352

----------------------------------------
```

Run 9:

Initial State:

6 1 3 8

2 0 4 12

5 13 7 14

10 9 15 11

Final State:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 0

Algorithm Used: A* with Manhattan

Solution Sequence:

L --> U --> R --> R --> D --> D --> R --> D --> L --> U --> R --> U --> U --> L --> L --> D --> L --> D --> D --> R --> U --> L --> D --> R --> R --> R

Solution Depth: 26

Max States Stored: 1004

----------------------------------------

```
Run 10:
Initial State:
0 1 2 4
5 7 3 12
9 6 8 10
13 14 11 15

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

Algorithm Used: A* with Manhattan
Solution Sequence:
R --> R --> D --> D --> R --> U --> L --> L --> D --> R --> D --> R

Solution Depth: 12
Max States Stored: 24

----------------------------------------
```

## Descriptive Statistics:

Minimum Solution Depth: 12

Maximum Solution Depth: 26

Average Solution Depth: 19.20

Minimum States Stored: 16

Maximum States Stored: 1004

Average States Stored: 288.30

## Euclidean:

Run 1:

Initial State:

1 6 0 4

9 5 2 8

13 10 3 7

14 11 15 12

Final State:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 0

Algorithm Used: A* with euclidean

Solution Sequence:

D --> D --> R --> D --> L --> L --> L --> U --> U --> R --> U --> R --> D --> D --> D --> R

Solution Depth: 16

Max States Stored: 56

-----------------------------------------

Run 2:

Initial State:

1 3 6 4

5 2 7 8

13 15 9 11

14 0 10 12

Final State:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 0

Algorithm Used: A* with euclidean

Solution Sequence:

U --> R --> D --> L --> L --> U --> R --> R --> U --> U --> L --> D --> R --> D --> R --> D

Solution Depth: 16

Max States Stored: 82

---------------------------------------

Run 3:

Initial State:

6 5 1 3

7 9 2 4

0 13 11 12

10 14 8 15

Final State:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 0

Algorithm Used: A* with euclidean

Solution Sequence:

R --> U --> L --> U --> R --> R --> D --> D --> D --> L --> L --> U --> R --> D --> R --> R --> U --> L --> U --> L --> L --> U --> R --> R --> R --> D --> D --> D

Solution Depth: 28

Max States Stored: 5441

---------------------------------------

Run 4:

Initial State:

6 3 0 4

2 1 7 11

5 14 8 10

9 13 15 12

Final State:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 0

Algorithm Used: A* with euclidean

Solution Sequence:

D --> D --> R --> U --> L --> U --> L --> D --> L --> U --> R --> D --> L --> D --> D --> R --> U --> R --> R --> D

Solution Depth: 20

Max States Stored: 172

---------------------------------------

Run 5:

Initial State:

6 3 7 4

2 1 11 8

0 5 12 15

13 9 10 14

Final State:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 0

Algorithm Used: A* with euclidean

Solution Sequence:

R --> D --> R --> R --> U --> L --> U --> U --> L --> D --> L --> U --> R --> D --> L --> D --> R --> D --> R --> R

Solution Depth: 20

Max States Stored: 95

-----------------------------------------

Run 6:

Initial State:

1 2 3 4

5 6 7 8

13 9 0 10

14 12 11 15

Final State:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 0

Algorithm Used: A* with euclidean

Solution Sequence:

R --> D --> L --> L --> L --> U --> R --> R --> D --> R --> U --> L --> D --> R

Solution Depth: 14

Max States Stored: 95

-------------------------------------

Run 7:

Initial State:

6 10 8 3

1 0 2 7

5 14 13 4

9 15 12 11

Final State:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 0

Algorithm Used: A* with euclidean

Solution Sequence:

D --> R --> R --> U --> U --> L --> D --> D --> R --> D --> L --> L --> U --> U --> U --> L --> D --> D --> D --> R --> U --> U --> U --> R --> D --> R --> U --> L --> D --> D --> R --> D

Solution Depth: 32

Max States Stored: 45187

---------------------------------------

```
Run 8:

Initial State:

1 3 4 7

10 2 15 6

0 13 11 8

5 9 14 12


Final State:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 0


Algorithm Used: A* with euclidean

Solution Sequence:

D --> R --> U --> R --> U --> R --> U --> L --> L --> D --> L --> D -->
D --> R --> R --> U --> L --> U --> R --> R --> D --> D


Solution Depth: 22

Max States Stored: 152


-------------------------------------
```

Run 9:

Initial State:

1 4 3 8

5 2 10 11

9 13 15 7

14 0 6 12

Final State:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 0

Algorithm Used: A* with euclidean

Solution Sequence:

R --> U --> U --> U --> L --> D --> L --> D --> R --> D --> L --> U --> U --> R --> D --> R --> R --> U --> U --> L --> D --> D --> R --> D

Solution Depth: 24

Max States Stored: 1664

---------------------------------------

```
Run 10:

Initial State:

2 5 3 4

1 0 8 12

9 6 7 15

13 10 14 11


Final State:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 0


Algorithm Used: A* with euclidean

Solution Sequence:

U --> L --> D --> R --> D --> D --> R --> R --> U --> U --> L --> D -->
D --> R


Solution Depth: 14

Max States Stored: 33


---------------------------------------
```

## Descriptive Statistics:

Minimum Solution Depth: 14

Maximum Solution Depth: 32

Average Solution Depth: 20.60

Minimum States Stored: 33

Maximum States Stored: 45187

Average States Stored: 5297.70

## Misplaced Tiles:

Run 1:

Initial State:

6 13 2 4

1 0 9 7

5 10 3 8

14 15 11 12

Final State:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

U --> L --> D --> R --> R --> D --> D --> L --> U --> L --> U --> R --> D --> D --> L --> U --> U --> R --> U --> R --> D --> R --> D --> D

Solution Depth: 24

Max States Stored: 28487

---------------------------------------

Run 2:

Initial State:

2 6 4 8

9 10 1 7

5 3 15 11

13 0 14 12

Final State:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

R --> U --> L --> U --> R --> D --> L --> L --> U --> R --> R --> R --> U --> L --> L --> D --> D --> R --> U --> U --> L --> L --> D --> R --> R --> D --> R --> D

Solution Depth: 28

Max States Stored: 327736

----------------------------------------

```
Run 3:
Initial State:
5 1 2 4
9 7 3 8
13 6 10 12
14 0 11 15

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

Algorithm Used: A* with misplaced tiles
Solution Sequence:
L --> U --> U --> U --> R --> R --> D --> L --> D --> R --> D --> R

Solution Depth: 12
Max States Stored: 16

----------------------------------------
```

Run 4:

Initial State:

1 3 4 8

5 6 2 7

0 14 9 12

13 10 11 15

Final State:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

D --> R --> U --> R --> D --> L --> L --> U --> R --> U --> R --> R --> U --> L --> L --> D --> D --> D --> R --> R

Solution Depth: 20

Max States Stored: 3582

----------------------------------------

Run 5:

Initial State:

1 6 4 7

5 3 11 2

9 8 0 12

13 10 14 15

Final State:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

L --> D --> R --> R --> U --> L --> U --> R --> U --> L --> D --> L --> U --> R --> D --> R --> D --> D

Solution Depth: 18

Max States Stored: 1675

---------------------------------------

Run 6:

Initial State:

1 2 3 4

6 10 7 8

5 15 12 14

9 13 11 0

Final State:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

L --> U --> R --> D --> L --> U --> L --> U --> L --> D --> D --> R --> R --> U --> R --> D

Solution Depth: 16

Max States Stored: 589

----------------------------------------

```
Run 7:
Initial State:
5 1 2 3
7 0 8 4
10 6 11 12
9 13 14 15

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

Algorithm Used: A* with misplaced tiles
Solution Sequence:
L --> U --> R --> R --> R --> D --> L --> L --> D --> L --> D --> R -->
R --> R

Solution Depth: 14
Max States Stored: 37

------------------------------------
```

Run 8:

Initial State:

1 5 2 3

9 6 4 8

10 7 12 15

13 14 11 0

Final State:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

U --> U --> L --> L --> U --> R --> R --> D --> D --> L --> L --> L --> U --> R --> R --> D --> D --> R

Solution Depth: 18

Max States Stored: 824

----------------------------------------

Run 9:

Initial State:

6 4 3 8

1 0 15 2

5 9 7 11

13 10 14 12

Final State:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

R --> U --> L --> L --> D --> D --> R --> U --> R --> R --> U --> L -->
D --> L --> U --> R --> D --> D --> L --> D --> R --> U --> R --> D

Solution Depth: 24

Max States Stored: 29913

----------------------------------------

Run 10:

Initial State:

1 6 2 7

5 0 4 3

9 15 11 8

13 10 14 12

Final State:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

U --> R --> D --> D --> L --> D --> R --> U --> U --> R --> U --> L --> D --> R --> D --> D

Solution Depth: 16

Max States Stored: 422

---------------------------------------

## Descriptive Statistics:

Minimum Solution Depth: 12

Maximum Solution Depth: 28

Average Solution Depth: 19.00

Minimum States Stored: 16

Maximum States Stored: 327736

Average States Stored: 39328.10

## Analysis For N=5:-

## Manhattan:

Run 1:

Initial State:

6 1 2 3 4

11 12 8 9 14

16 13 17 10 5

0 18 7 19 15

21 22 23 24 20

Final State:

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 0

Algorithm Used: A* with manhattan

Solution Sequence:

U --> R --> R --> D --> L --> U --> L --> U --> U --> R --> R --> D --> D --> L --> U --> R --> R --> R --> D --> L --> U --> L --> U --> R --> R --> D --> D --> D --> D

Solution Depth: 29

Max States Stored: 1694

Run 2:

Initial State:

1 2 3 13 4

6 7 8 9 10

11 0 12 5 15

16 17 18 14 20

21 22 23 19 24

Final State:

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 0

Algorithm Used: A* with manhattan

Solution Sequence:

R --> U --> R --> D --> R --> U --> L --> U --> R --> D --> D --> L --> U --> L --> D --> R --> D --> D --> R

Solution Depth: 19

Max States Stored: 471

Run 3:

Initial State:

7 8 3 5 10

6 1 2 4 0

11 12 19 9 14

16 17 13 18 24

21 22 23 20 15

Final State:

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 0

Algorithm Used: A* with manhattan

Solution Sequence:

U --> L --> L --> L --> D --> R --> U --> R --> D --> D --> L --> L --> L --> U --> U --> R --> D --> L --> D --> R --> R --> D --> R --> R --> D --> L --> U --> U --> R --> D --> D

Solution Depth: 31

Max States Stored: 7435

Run 4:

Initial State:

2 7 3 8 4

1 12 9 15 5

6 14 13 19 10

11 17 18 23 0

16 21 22 24 20

Final State:

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 0

Algorithm Used: A* with manhattan

Solution Sequence:

D --> L --> U --> U --> U --> L --> D --> L --> U --> U --> R --> R --> R --> D --> D --> L --> L --> U --> U --> L --> L --> D --> D --> D --> D --> R --> R --> R --> R

Solution Depth: 29

Max States Stored: 879

Run 5:

Initial State:

2 6 3 4 5

1 7 13 8 10

16 11 9 15 24

17 22 12 20 14

21 23 18 0 19

Final State:

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 0

Algorithm Used: A* with manhattan

Solution Sequence:

U --> R --> U --> L --> L --> D --> D --> L --> U --> L --> U --> R --> R --> U --> L --> U --> L --> D --> R --> R --> R --> D --> D --> D --> R --> U --> L --> D --> R

Solution Depth: 29

Max States Stored: 789

Run 6:

Initial State:

12 1 8 3 5

2 6 0 7 9

17 11 14 4 10

16 22 13 19 15

21 23 18 24 20

Final State:

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 0

Algorithm Used: A* with manhattan

Solution Sequence:

L --> L --> U --> R --> D --> D --> L --> U --> R --> R --> R --> D --> L --> D --> D --> L --> U --> U --> L --> U --> R --> R --> U --> R --> D --> R --> D --> D --> D

Solution Depth: 29

Max States Stored: 689

Run 7:

Initial State:

1 2 13 3 5

0 6 7 4 9

11 12 14 8 10

18 16 17 19 15

21 22 23 24 20

Final State:

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 0

Algorithm Used: A* with manhattan

Solution Sequence:

D --> D --> R --> U --> L --> U --> R --> R --> U --> R --> D --> D --> L --> L --> D --> R --> U --> U --> R --> R --> D --> D --> D

Solution Depth: 23

Max States Stored: 840

Run 8:

Initial State:

1 2 3 4 5

16 6 18 13 10

7 21 11 8 14

0 22 12 9 15

23 17 24 19 20

Final State:

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 0

Algorithm Used: A* with manhattan

Solution Sequence:

R --> U --> L --> U --> R --> D --> R --> U --> R --> D --> D --> D --> L --> L --> L --> U --> R --> R --> D --> L --> L --> U --> U --> R --> D --> R --> U --> U --> R --> D --> R --> D --> D

Solution Depth: 33

Max States Stored: 1102

Run 9:

Initial State:

1 2 12 3 5

7 8 4 9 0

6 22 13 19 10

11 16 17 15 14

21 18 23 24 20

Final State:

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 0

Algorithm Used: A* with manhattan

Solution Sequence:

D --> D --> L --> U --> U --> L --> U --> R --> D --> D --> L --> U --> L --> L --> D --> D --> R --> U --> R --> R --> R --> D --> D --> L --> L --> L --> U --> R --> D --> R --> R

Solution Depth: 31

Max States Stored: 5185

Run 10:

Initial State:

6 1 8 3 4

2 12 7 9 0

11 18 14 10 5

16 13 17 19 15

21 22 23 24 20

Final State:

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 0

Algorithm Used: A* with manhattan

Solution Sequence:

D --> L --> L --> L --> D --> R --> U --> L --> U --> L --> U --> R --> D --> R --> U --> R --> R --> D --> D --> D --> D

Solution Depth: 21

Max States Stored: 59

## Descriptive Statistics:

Minimum Solution Depth: 19

Maximum Solution Depth: 33

Average Solution Depth: 27.40

Minimum States Stored: 59

Maximum States Stored: 7435

Average States Stored: 1914.30

## Euclidean:

Run 1:

Initial State:

1 2 3 4 5

7 8 9 10 15

6 11 12 23 0

16 17 14 13 18

21 22 19 24 20

Final State:

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 0

Algorithm Used: A* with euclidean

Solution Sequence:

U --> L --> L --> L --> L --> D --> R --> R --> D --> R --> U --> L --> D --> R --> R --> D --> L --> L --> U --> R --> D --> R

Solution Depth: 22

Max States Stored: 353

Run 2:

Initial State:

1 2 3 9 4

6 7 8 0 5

11 12 13 15 10

16 17 18 14 20

21 22 23 19 24

Final State:

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 0

Algorithm Used: A* with euclidean

Solution Sequence:

U --> R --> D --> D --> L --> D --> D --> R

Solution Depth: 8

Max States Stored: 12

Run 3:

Initial State:

1 2 8 3 5

6 12 7 4 10

0 11 20 9 19

16 18 13 23 14

21 17 22 24 15

Final State:

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 0

Algorithm Used: A* with euclidean

Solution Sequence:

R --> U --> R --> U --> R --> D --> D --> R --> D --> D --> L --> U --> U --> L --> D --> L --> D --> R --> R --> U --> U --> R --> D --> L --> D --> R

Solution Depth: 26

Max States Stored: 1758

Run 4:

Initial State:

1 2 4 9 5

11 8 12 3 10

7 6 18 13 14

16 0 22 19 15

17 21 23 24 20

Final State:

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 0

Algorithm Used: A* with euclidean

Solution Sequence:

R --> U --> U --> R --> U --> L --> D --> L --> D --> L --> D --> D --> R --> U --> L --> U --> U --> R --> D --> R --> R --> R --> D --> D

Solution Depth: 24

Max States Stored: 492

Run 5:

Initial State:

13 2 3 4 5

8 7 1 9 10

0 6 12 14 15

11 17 18 19 20

16 21 22 23 24

Final State:

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 0

Algorithm Used: A* with euclidean

Solution Sequence:

U --> U --> R --> D --> D --> L --> U --> R --> R --> D --> L --> L --> U --> R --> R --> D --> L --> U --> U --> L --> D --> D --> D --> D --> R --> R --> R --> R

Solution Depth: 28

Max States Stored: 13510

Run 6:

Initial State:

1 2 3 4 5

6 7 8 9 10

11 14 18 13 15

21 23 17 0 24

12 16 22 20 19

Final State:

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 0

Algorithm Used: A* with euclidean

Solution Sequence:

D --> R --> U --> L --> L --> L --> D --> L --> U --> R --> D --> R --> U --> R --> U --> L --> L --> D --> R --> R --> U --> L --> D --> R --> D --> R

Solution Depth: 26

Max States Stored: 4920

Run 7:

Initial State:

1 2 8 3 9

6 7 13 10 5

11 12 19 4 0

16 17 14 15 24

21 22 18 20 23

Final State:

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 0

Algorithm Used: A* with euclidean

Solution Sequence:

L --> U --> R --> D --> L --> D --> D --> R --> U --> L --> L --> U --> U --> U --> R --> D --> R --> U --> L --> D --> R --> D --> L --> D --> L --> D --> R --> R

Solution Depth: 28

Max States Stored: 7974

Run 8:

Initial State:

6 1 3 4 5

2 8 12 9 10

11 17 18 7 13

16 22 14 24 15

21 23 0 19 20

Final State:

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 0

Algorithm Used: A* with euclidean

Solution Sequence:

R --> U --> L --> U --> R --> R --> D --> D --> L --> L --> L --> U --> U --> R --> U --> L --> L --> U --> R --> D --> D --> R --> R --> D --> D --> R

Solution Depth: 26

Max States Stored: 1398

Run 9:

Initial State:

8 7 2 4 5

16 1 3 9 10

6 17 19 14 0

11 13 12 18 24

21 22 23 20 15

Final State:

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 0

Algorithm Used: A* with euclidean

Solution Sequence:

L --> L --> D --> L --> L --> U --> U --> R --> D --> L --> D --> R --> U --> U --> U --> L --> D --> D --> D --> R --> U --> U --> U --> R --> D --> L --> D --> R --> D --> R --> R --> D --> L --> U --> U --> R --> D --> D

Solution Depth: 38

Max States Stored: 295318

Run 10:

Initial State:

6 8 1 3 4

11 2 13 9 5

16 7 18 14 10

21 23 19 0 15

22 12 17 24 20

Final State:

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 0

Algorithm Used: A* with euclidean

Solution Sequence:

L --> L --> D --> R --> U --> U --> U --> L --> U --> R --> D --> L --> D --> D --> D --> L --> U --> U --> U --> U --> R --> R --> R --> R --> D --> D --> D --> D

Solution Depth: 28

Max States Stored: 311

## Descriptive Statistics:

Minimum Solution Depth: 8

Maximum Solution Depth: 38

Average Solution Depth: 25.40

Minimum States Stored: 12

Maximum States Stored: 295318

Average States Stored: 32604.60

## Misplaced Tiles:

Run 1:

Initial State:

1 2 3 4 5

6 0 8 9 10

11 7 12 14 15

16 17 13 24 19

21 22 18 23 20

Final State:

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

D --> R --> D --> D --> R --> U --> R --> D

Solution Depth: 8

Max States Stored: 15

Run 2:

Initial State:

1 2 3 4 5

6 7 8 9 10

16 11 17 14 15

21 13 23 18 20

0 12 22 19 24

Final State:

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

U --> U --> R --> D --> D --> R --> U --> U --> L --> D --> R --> R --> D --> R

Solution Depth: 14

Max States Stored: 129

Run 3:

Initial State:

1 9 8 3 5

11 0 2 4 10

21 6 13 14 15

12 7 18 19 20

17 16 22 23 24

Final State:

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

D --> D --> L --> U --> U --> R --> R --> U --> L --> D --> R --> U --> R --> D --> L --> L --> D --> D --> D --> L --> U --> R --> D --> R --> R --> R

Solution Depth: 26

Max States Stored: 20861

Run 4:

Initial State:

1 2 3 4 14

6 7 8 10 9

12 16 13 19 5

11 0 17 18 15

21 22 23 24 20

Final State:

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

U --> L --> D --> R --> R --> R --> U --> U --> R --> U --> L --> D --> R --> D --> L --> U --> U --> R --> D --> D --> D --> D

Solution Depth: 22

Max States Stored: 9947

Run 5:

Initial State:

1 2 8 3 4

6 7 9 0 5

11 12 13 19 10

22 17 18 24 14

16 21 23 20 15

Final State:

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

L --> U --> R --> R --> D --> D --> D --> D --> L --> U --> L --> L --> L --> D --> R --> U --> R --> R --> U --> R --> D --> D

Solution Depth: 22

Max States Stored: 3003

Run 6:

Initial State:

2 4 8 5 10

1 6 15 13 9

11 7 3 19 0

16 12 18 14 20

21 17 22 23 24

Final State:

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

L --> U --> L --> U --> L --> L --> D --> R --> R --> D --> R --> U --> R --> U --> L --> L --> D --> L --> D --> D --> D --> R --> R --> R --> U --> U --> L --> D --> R --> D

Solution Depth: 30

Max States Stored: 115721

Run 7:

Initial State:

6 1 3 4 5

11 2 8 9 10

21 7 0 14 15

12 17 13 19 20

22 16 18 23 24

Final State:

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

D --> L --> L --> U --> U --> U --> R --> D --> D --> D --> D --> L --> U --> R --> R --> D --> R --> R

Solution Depth: 18

Max States Stored: 310

Run 8:

Initial State:

0 8 6 9 5

2 1 4 3 10

11 7 13 14 15

16 12 17 18 20

21 22 23 19 24

Final State:

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

D --> R --> U --> R --> D --> R --> U --> L --> D --> L --> U --> L --> D --> R --> D --> D --> R --> R --> D --> R

Solution Depth: 20

Max States Stored: 939

Run 9:

Initial State:

1 2 3 4 5

6 7 13 8 10

16 11 12 9 15

21 17 18 14 20

22 23 0 19 24

Final State:

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

L --> L --> U --> U --> R --> R --> U --> R --> D --> D --> D --> R

Solution Depth: 12

Max States Stored: 19

Run 10:

Initial State:

2 7 3 4 5

1 17 8 13 10

6 16 12 9 14

11 0 19 18 15

21 22 23 24 20

Final State:

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

U --> R --> D --> L --> U --> U --> R --> R --> D --> L --> L --> D --> R --> U --> U --> L --> U --> L --> D --> D --> D --> R --> R --> R --> U --> R --> D --> D

Solution Depth: 28

Max States Stored: 252523

## Descriptive Statistics:

Minimum Solution Depth: 8

Maximum Solution Depth: 30

Average Solution Depth: 20.00

Minimum States Stored: 15

Maximum States Stored: 252523

Average States Stored: 40346.70

# Demo Runs For N=6:-

# Manhattan:

Run 1:

Initial State:

2 8 9 3 5 6

7 1 15 4 11 12

0 20 14 10 18 23

13 27 26 17 22 24

19 25 21 16 28 29

31 32 33 34 35 30

Final State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 17 18

19 20 21 22 23 24

25 26 27 28 29 30

31 32 33 34 35 0

Algorithm Used: A* with manhattan

Solution Sequence:

U --> R --> U --> L --> D --> D --> D --> D --> R --> U --> R --> D --> L --> U --> U --> R --> U --> U --> R --> D --> D --> D --> D --> R --> R --> U --> U --> L --> L --> D --> R --> R --> D --> D

Solution Depth: 34

Max States Stored: 2051

Run 2:

Initial State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 23 17

19 26 22 0 18 24

25 21 20 34 28 35

31 32 27 33 30 29

Final State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 17 18

19 20 21 22 23 24

25 26 27 28 29 30

31 32 33 34 35 0

Algorithm Used: A* with manhattan

Solution Sequence:

L --> D --> L --> U --> R --> D --> D --> R --> U --> R --> D --> R --> U --> U --> L --> U --> R --> D --> D --> L --> D --> R

Solution Depth: 22

Max States Stored: 238

Run 3:

Initial State:

1 2 3 4 5 6

7 15 14 9 10 11

13 8 21 16 18 12

19 20 28 22 17 24

25 26 35 23 34 29

31 32 27 33 30 0

Final State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 17 18

19 20 21 22 23 24

25 26 27 28 29 30

31 32 33 34 35 0

Algorithm Used: A* with manhattan

Solution Sequence:

L --> U --> L --> L --> U --> U --> U --> L --> D --> R --> U --> R --> R --> R --> D --> L --> D --> D --> L --> L --> D --> R --> R --> U --> R --> D

Solution Depth: 26

Max States Stored: 75

Run 4:

Initial State:

1 2 3 4 5 6

7 8 9 10 11 12

13 15 20 16 17 18

19 26 14 21 24 30

0 25 27 28 22 35

31 32 33 23 34 29

Final State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 17 18

19 20 21 22 23 24

25 26 27 28 29 30

31 32 33 34 35 0

Algorithm Used: A* with manhattan

Solution Sequence:

R --> U --> R --> U --> L --> D --> R --> R --> D --> D --> R --> R --> U --> U --> L --> D --> L --> U --> R --> D --> D --> R

Solution Depth: 22

Max States Stored: 120

Run 5:

Initial State:

1 2 3 4 5 6

7 9 16 15 10 12

13 8 14 11 17 18

19 20 21 22 24 29

25 26 27 28 23 30

31 32 33 34 35 0

Final State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 17 18

19 20 21 22 23 24

25 26 27 28 29 30

31 32 33 34 35 0

Algorithm Used: A* with manhattan

Solution Sequence:

U --> U --> L --> U --> L --> U --> L --> L --> D --> R --> R --> U --> R --> D --> D --> D --> R --> D

Solution Depth: 18

Max States Stored: 59

Run 6:

Initial State:

1 9 3 11 5 6

8 2 14 4 16 12

7 13 21 15 10 17

19 0 20 22 23 18

25 26 27 28 29 24

31 32 33 34 35 30

Final State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 17 18

19 20 21 22 23 24

25 26 27 28 29 30

31 32 33 34 35 0

Algorithm Used: A* with manhattan

Solution Sequence:

R --> U --> U --> R --> U --> L --> L --> D --> L --> D --> R --> R --> R --> R --> U --> L --> L --> U --> R --> D --> D --> R --> R --> D --> D --> D

Solution Depth: 26

Max States Stored: 561

Run 7:

Initial State:

1 2 3 4 5 6

13 7 9 10 17 11

0 8 14 15 16 18

19 20 21 22 12 24

25 26 27 28 23 30

31 32 33 34 29 35

Final State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 17 18

19 20 21 22 23 24

25 26 27 28 29 30

31 32 33 34 35 0

Algorithm Used: A* with manhattan

Solution Sequence:

U --> R --> D --> R --> R --> R --> D --> R --> U --> L --> U --> R --> D --> D --> L --> D --> D --> R

Solution Depth: 18

Max States Stored: 205

Run 8:

Initial State:

1 2 3 4 5 6

7 8 10 16 11 12

19 13 21 15 17 18

14 27 22 9 23 28

25 20 0 34 29 24

31 26 32 33 35 30

Final State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 17 18

19 20 21 22 23 24

25 26 27 28 29 30

31 32 33 34 35 0

Algorithm Used: A* with manhattan

Solution Sequence:

U --> L --> L --> U --> R --> D --> D --> D --> R --> R --> U --> L --> U --> R --> R --> R --> D --> L --> U --> L --> U --> L --> D --> R --> U --> U --> L --> D --> D --> R --> D --> R --> R --> D

Solution Depth: 34

Max States Stored: 39876

Run 9:

Initial State:

1 2 3 4 5 6

7 8 9 10 17 11

13 14 15 21 16 12

25 19 20 22 18 0

26 32 28 23 29 24

31 33 27 34 35 30

Final State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 17 18

19 20 21 22 23 24

25 26 27 28 29 30

31 32 33 34 35 0

Algorithm Used: A* with manhattan

Solution Sequence:

L --> L --> U --> R --> U --> R --> D --> D --> D --> L --> L --> L --> D --> L --> U --> L --> U --> R --> R --> R --> R --> D --> R --> D

Solution Depth: 24

Max States Stored: 361

Run 10:

Initial State:

1 2 3 4 5 6

7 8 9 11 12 0

13 14 16 10 23 17

19 20 15 21 28 18

25 26 27 34 22 24

31 32 33 29 30 35

Final State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 17 18

19 20 21 22 23 24

25 26 27 28 29 30

31 32 33 34 35 0

Algorithm Used: A* with manhattan

Solution Sequence:

L --> L --> D --> L --> D --> R --> R --> D --> D --> L --> U --> U --> R --> U --> R --> D --> D --> L --> D --> R

Solution Depth: 20

Max States Stored: 40

## Descriptive Statistics:

Minimum Solution Depth: 18

Maximum Solution Depth: 34

Average Solution Depth: 24.40

Minimum States Stored: 40

Maximum States Stored: 39876

Average States Stored: 4358.60

## Euclidean:

Run 1:

Initial State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 21 16 17 18

19 20 22 0 15 24

25 26 27 28 23 35

31 32 33 34 30 29

Final State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 17 18

19 20 21 22 23 24

25 26 27 28 29 30

31 32 33 34 35 0

Algorithm Used: A* with euclidean

Solution Sequence:

R --> U --> L --> D --> L --> U --> R --> R --> D --> D --> D --> R --> U --> L --> D --> R

Solution Depth: 16

Max States Stored: 386

Run 2:

Initial State:

1 2 9 3 4 5

7 0 8 10 11 6

13 15 16 23 18 12

19 14 21 17 22 24

25 20 33 27 29 30

31 26 32 28 34 35

Final State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 17 18

19 20 21 22 23 24

25 26 27 28 29 30

31 32 33 34 35 0

Algorithm Used: A* with euclidean

Solution Sequence:

R --> U --> R --> R --> R --> D --> D --> L --> L --> D --> R --> U --> L --> L --> L --> D --> D --> D --> R --> U --> R --> D --> R --> R

Solution Depth: 24

Max States Stored: 55

Run 3:

Initial State:

1 2 9 3 6 12

7 8 15 4 11 5

13 14 0 10 16 17

19 20 21 23 24 18

25 26 27 22 28 29

31 32 33 34 35 30

Final State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 17 18

19 20 21 22 23 24

25 26 27 28 29 30

31 32 33 34 35 0

Algorithm Used: A* with euclidean

Solution Sequence:

U --> U --> R --> D --> D --> R --> U --> R --> U --> L --> D --> D --> R --> D --> L --> L --> D --> R --> R --> D

Solution Depth: 20

Max States Stored: 132

Run 4:

Initial State:

1 2 3 4 12 5

7 8 9 0 11 6

13 14 15 10 17 18

19 20 21 16 22 24

25 27 34 28 23 35

31 26 32 33 30 29

Final State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 17 18

19 20 21 22 23 24

25 26 27 28 29 30

31 32 33 34 35 0

Algorithm Used: A* with euclidean

Solution Sequence:

R --> U --> R --> D --> L --> L --> D --> D --> D --> L --> L --> D --> R --> R --> U --> U --> R --> D --> D --> R --> U --> L --> D --> R

Solution Depth: 24

Max States Stored: 929

Run 5:

Initial State:

1 2 3 4 5 6

7 14 8 16 10 12

19 13 9 15 11 18

32 26 27 22 17 24

0 21 25 28 23 29

20 31 33 34 35 30

Final State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 17 18

19 20 21 22 23 24

25 26 27 28 29 30

31 32 33 34 35 0

Algorithm Used: A* with euclidean

Solution Sequence:

U --> R --> D --> L --> D --> R --> U --> R --> U --> L --> L --> D --> R --> U --> L --> U --> R --> U --> R --> D --> R --> U --> R --> D --> D --> D --> R --> D

Solution Depth: 28

Max States Stored: 631

Run 6:

Initial State:

1 2 3 4 5 6

7 9 10 15 11 12

13 8 14 22 16 23

19 20 34 18 17 0

25 26 21 27 28 24

31 32 33 35 30 29

Final State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 17 18

19 20 21 22 23 24

25 26 27 28 29 30

31 32 33 34 35 0

Algorithm Used: A* with euclidean

Solution Sequence:

L --> L --> L --> D --> R --> U --> U --> U --> L --> L --> D --> R --> R --> R --> D --> R --> U --> L --> D --> R --> D --> D --> L --> L --> U --> R --> R --> D

Solution Depth: 28

Max States Stored: 1093

Run 7:

Initial State:

1 3 4 10 5 6

7 2 9 0 17 12

13 8 21 11 16 18

31 15 14 22 23 24

20 19 26 33 27 30

32 25 34 29 28 35

Final State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 17 18

19 20 21 22 23 24

25 26 27 28 29 30

31 32 33 34 35 0

Algorithm Used: A* with euclidean

Solution Sequence:

D --> R --> U --> L --> U --> L --> L --> D --> D --> D --> D --> L --> U --> R --> R --> U --> L --> D --> D --> D --> L --> U --> R --> R --> R --> R --> D --> L --> L --> U --> R --> R --> D --> R

Solution Depth: 34

Max States Stored: 1695

Run 8:

Initial State:

7 1 2 10 4 6

13 9 3 16 5 11

8 20 14 22 17 12

19 15 27 21 23 18

25 32 26 28 29 24

31 33 34 0 35 30

Final State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 17 18

19 20 21 22 23 24

25 26 27 28 29 30

31 32 33 34 35 0

Algorithm Used: A* with euclidean

Solution Sequence:

L --> L --> U --> R --> U --> L --> U --> L --> U --> U --> R --> R --> D --> L --> D --> R --> D --> R --> U --> U --> U --> R --> D --> R --> D --> D --> D --> D

Solution Depth: 28

Max States Stored: 75

Run 9:

Initial State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 0 28 24 17

19 20 16 15 22 18

26 31 21 35 27 29

25 32 33 34 23 30

Final State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 17 18

19 20 21 22 23 24

25 26 27 28 29 30

31 32 33 34 35 0

Algorithm Used: A* with euclidean

Solution Sequence:

D --> R --> U --> L --> D --> D --> D --> L --> U --> L --> D --> R --> R --> R --> U --> R --> D --> L --> L --> U --> R --> U --> R --> U --> R --> D --> L --> D --> R --> D

Solution Depth: 30

Max States Stored: 6018

Run 10:

Initial State:

1 2 3 5 11 6

7 8 9 4 12 18

13 14 15 10 16 24

19 0 20 22 17 29

25 26 21 34 28 23

31 32 27 33 35 30

Final State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 17 18

19 20 21 22 23 24

25 26 27 28 29 30

31 32 33 34 35 0

Algorithm Used: A* with euclidean

Solution Sequence:

R --> D --> D --> R --> U --> R --> R --> U --> U --> U --> L --> U --> L --> D --> D --> R --> D --> D --> R --> D

Solution Depth: 20

Max States Stored: 33

## Descriptive Statistics:

Minimum Solution Depth: 16

Maximum Solution Depth: 34

Average Solution Depth: 25.20

Minimum States Stored: 33

Maximum States Stored: 6018

Average States Stored: 1104.70

# Misplaced Tiles:

Run 1:

Initial State:

1 2 3 4 5 6

7 8 9 10 12 18

13 14 15 16 11 24

25 19 21 22 17 30

20 33 27 28 0 23

31 26 32 34 29 35

Final State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 17 18

19 20 21 22 23 24

25 26 27 28 29 30

31 32 33 34 35 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

L --> L --> L --> L --> U --> R --> D --> D --> R --> U --> R --> R --> R --> U --> U --> U --> L --> D --> D --> D --> D --> R

Solution Depth: 22

Max States Stored: 1463

Run 2:

Initial State:

1 2 3 4 5 6

7 8 9 10 11 12

14 15 27 16 17 18

13 25 21 0 23 24

20 19 26 22 28 29

31 32 33 34 35 30

Final State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 17 18

19 20 21 22 23 24

25 26 27 28 29 30

31 32 33 34 35 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

L --> U --> L --> L --> D --> D --> R --> U --> L --> D --> R --> R --> U --> R --> D --> R --> R --> D

Solution Depth: 18

Max States Stored: 615

Run 3:

Initial State:

1 8 2 4 5 6

13 7 3 15 10 11

0 14 16 9 18 12

19 20 21 22 17 24

25 26 27 28 23 29

31 32 33 34 35 30

Final State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 17 18

19 20 21 22 23 24

25 26 27 28 29 30

31 32 33 34 35 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

U --> R --> U --> R --> D --> R --> D --> L --> U --> R --> R --> R --> D --> L --> D --> D --> R --> D

Solution Depth: 18

Max States Stored: 85

Run 4:

Initial State:

1 3 4 5 6 11

7 2 9 10 17 18

13 8 14 16 0 12

19 20 15 28 22 24

25 26 21 27 23 29

31 32 33 34 35 30

Final State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 17 18

19 20 21 22 23 24

25 26 27 28 29 30

31 32 33 34 35 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

U --> R --> U --> L --> L --> L --> L --> D --> D --> R --> D --> D --> R --> U --> R --> U --> U --> R --> D --> L --> D --> D --> R --> D

Solution Depth: 24

Max States Stored: 1505

Run 5:

Initial State:

1 8 2 3 5 6

7 9 4 16 10 12

13 14 21 15 17 18

19 0 27 11 22 23

25 20 26 28 29 24

31 32 33 34 35 30

Final State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 17 18

19 20 21 22 23 24

25 26 27 28 29 30

31 32 33 34 35 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

D --> R --> U --> U --> R --> D --> R --> U --> L --> U --> L --> L --> U --> R --> R --> D --> R --> D --> D --> R --> D --> D

Solution Depth: 22

Max States Stored: 1371

Run 6:

Initial State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 17 18

19 20 21 0 23 24

25 27 32 22 34 29

31 26 33 35 28 30

Final State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 17 18

19 20 21 22 23 24

25 26 27 28 29 30

31 32 33 34 35 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

D --> R --> D --> L --> L --> U --> L --> D --> R --> R --> U --> R --> R --> D

Solution Depth: 14

Max States Stored: 237

Run 7:

Initial State:

1 2 3 4 5 6

7 8 9 16 10 0

13 14 15 11 18 12

19 20 21 22 17 24

25 26 27 28 23 30

31 32 33 34 29 35

Final State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 17 18

19 20 21 22 23 24

25 26 27 28 29 30

31 32 33 34 35 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

D --> L --> L --> U --> R --> D --> D --> D --> D --> R

Solution Depth: 10

Max States Stored: 28

Run 8:

Initial State:

8 1 2 3 10 4

7 13 9 11 6 5

0 20 14 16 17 12

19 21 22 15 24 18

25 26 27 28 23 30

31 32 33 34 29 35

Final State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 17 18

19 20 21 22 23 24

25 26 27 28 29 30

31 32 33 34 35 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

U --> U --> R --> R --> R --> R --> R --> D --> L --> L --> D --> D --> L --> L --> U --> U --> L --> D --> R --> R --> R --> U --> U --> R --> R --> D --> D --> D --> L --> D --> D --> R

Solution Depth: 32

Max States Stored: 39594

Run 9:

Initial State:

1 3 9 4 5 6

7 2 14 10 11 12

0 13 8 15 18 23

19 21 27 16 22 29

25 26 20 28 24 17

31 32 33 34 35 30

Final State:

1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 17 18

19 20 21 22 23 24

25 26 27 28 29 30

31 32 33 34 35 0

Algorithm Used: A* with misplaced tiles

Solution Sequence:

D --> D --> R --> R --> U --> L --> D --> L --> U --> U --> R --> R --> U --> U --> L --> D --> D --> R --> R --> D --> R --> R --> D --> L --> U --> R --> U --> L --> D --> R --> D --> D

Solution Depth: 32

Max States Stored: 806556

Run 10:

Initial State:

```
1 2 3 4 5 6

7 8 10 0 11 12

19 14 9 15 16 18

25 13 21 22 17 23

31 20 26 28 29 24

32 33 27 34 35 30
```

Final State:

```
1 2 3 4 5 6

7 8 9 10 11 12

13 14 15 16 17 18

19 20 21 22 23 24

25 26 27 28 29 30

31 32 33 34 35 0
```

Algorithm Used: A* with misplaced tiles

Solution Sequence:

L --> D --> L --> D --> D --> R --> D --> L --> L --> U --> U --> U --> R --> R --> R --> R --> D --> R --> D --> D

Solution Depth: 20

Max States Stored: 121

## Descriptive Statistics:

Minimum Solution Depth: 10

Maximum Solution Depth: 32

Average Solution Depth: 21.20

Minimum States Stored: 28

Maximum States Stored: 806556

Average States Stored: 85157.50

# Results of Assignment 1 for N=3:

## BFS:-

Descriptive Statistics:

Minimum Solution Depth: 7

Maximum Solution Depth: 24

Average Solution Depth: 19.5

Minimum States Stored: 181

Maximum States Stored: 61384

Average States Stored: 40613.4

## IDDFS (DFS with Iterative Deepening):-

Minimum Solution Depth: 18

Maximum Solution Depth: 25

Average Solution Depth: 21.80

Minimum States Stored: 17

Maximum States Stored: 23

Average States Stored: 19.70

## DFS (With Revisit):-

Minimum Solution Depth: 25449

Maximum Solution Depth: 105411

Average Solution Depth: 70549.2

Minimum States Stored: 20053

Maximum States Stored: 83069

Average States Stored: 55419.5

## 3. Comment:

**Note:** The following comparisons is based on the results of puzzle size N=3

**Breadth-First Search (BFS):**

- **Solution Depth**: This algorithm has an average depth of 19.5, with a minimum of 7 and a maximum of 24.
- **States Stored**: On average, BFS stored 40,613.4 states. The maximum number of states stored reached an impressive 61,384, which is quite high.

**Comment**: BFS explores all possible states level by level, ensuring it finds the shortest path to the solution. This often results in a large number of states being stored, especially for larger solution depths, as seen in the results. The breadth of exploration is evident in the vast number of states stored.

**IDDFS (DFS with Iterative Deepening):**

- **Solution Depth**: This method averaged a solution depth of 21.80.
- **States Stored**: The average number of states stored is very low, at only 19.70.

**Comment**: IDDFS combines the benefits of BFS's completeness with DFS's space efficiency. It guarantees finding the shortest path like BFS but uses much less memory. The results show the space efficiency, with a significantly lower average number of states stored compared to BFS.

**DFS (With Revisit):**

- **Solution Depth**: The solution depths are remarkably high, with an average of 70,549.2.
- **States Stored**: The average number of states stored is 55,419.5.

**Commentary**: DFS dives deep into the tree before backtracking. This depth-first behavior can lead to extremely long solution paths when a more direct solution exists, as evidenced by the high solution depth numbers. This method also stores a large number of states, indicating significant exploration.

**A* with Manhattan Heuristic:**

- **Solution Depth**: The average solution depth is 23.80.
- **States Stored**: The average number of states stored is 1,781.60.

**Commentary**: The Manhattan distance heuristic tends to be effective for this type of problem as it gives an estimate of the cost based on the distance tiles are from their goal positions. The average solution depth and states stored are reasonable, showcasing the efficiency of A* with this heuristic.

**A* with Euclidean Heuristic:**

- **Solution Depth**: Average solution depth is 22.50.
- **States Stored**: The algorithm stored an average of 1,835.50 states.

**Commentary**: Euclidean distance tends to be less accurate than Manhattan for grid-based problems because it doesn't consider the grid's constraints. The results are close to those of the Manhattan heuristic, but with slightly different depth and states stored.

**A* with Misplaced Tiles Heuristic:**

- **Solution Depth**: This heuristic resulted in an average solution depth of 23.20.
- **States Stored**: On average, 10,000.90 states were stored.

**Commentary**: Counting misplaced tiles provides a simpler heuristic, which can be less informative and may result in exploring more states than other heuristics. This is evident in the significantly higher number of states stored compared to the Manhattan and Euclidean heuristics.

**Overall Analysis:**

For a puzzle of size N the choice of algorithm and heuristic will depend on specific requirements:

1. **Efficiency in Terms of Depth**: If you're looking for the shortest solution path (fewest steps to solve), then BFS offers the smallest minimum solution depth. However, this comes at the cost of storing a potentially large number of states.

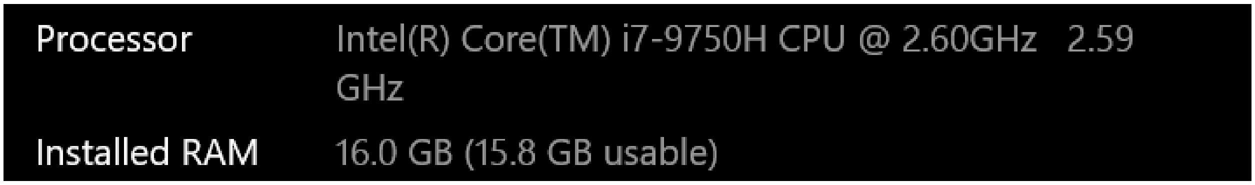
2. **Memory Consumption**: If memory usage is a concern, IDDFS stands out with its minimal states stored, but this efficiency comes at the expense of a higher average solution depth.

3. **Predictability**: DFS with revisits has a widely varying solution depth, making it unpredictable in terms of performance. It might find a solution quickly in some cases, but it might take a very long path in others.

4. **Heuristics and A***: Among the A* heuristics:

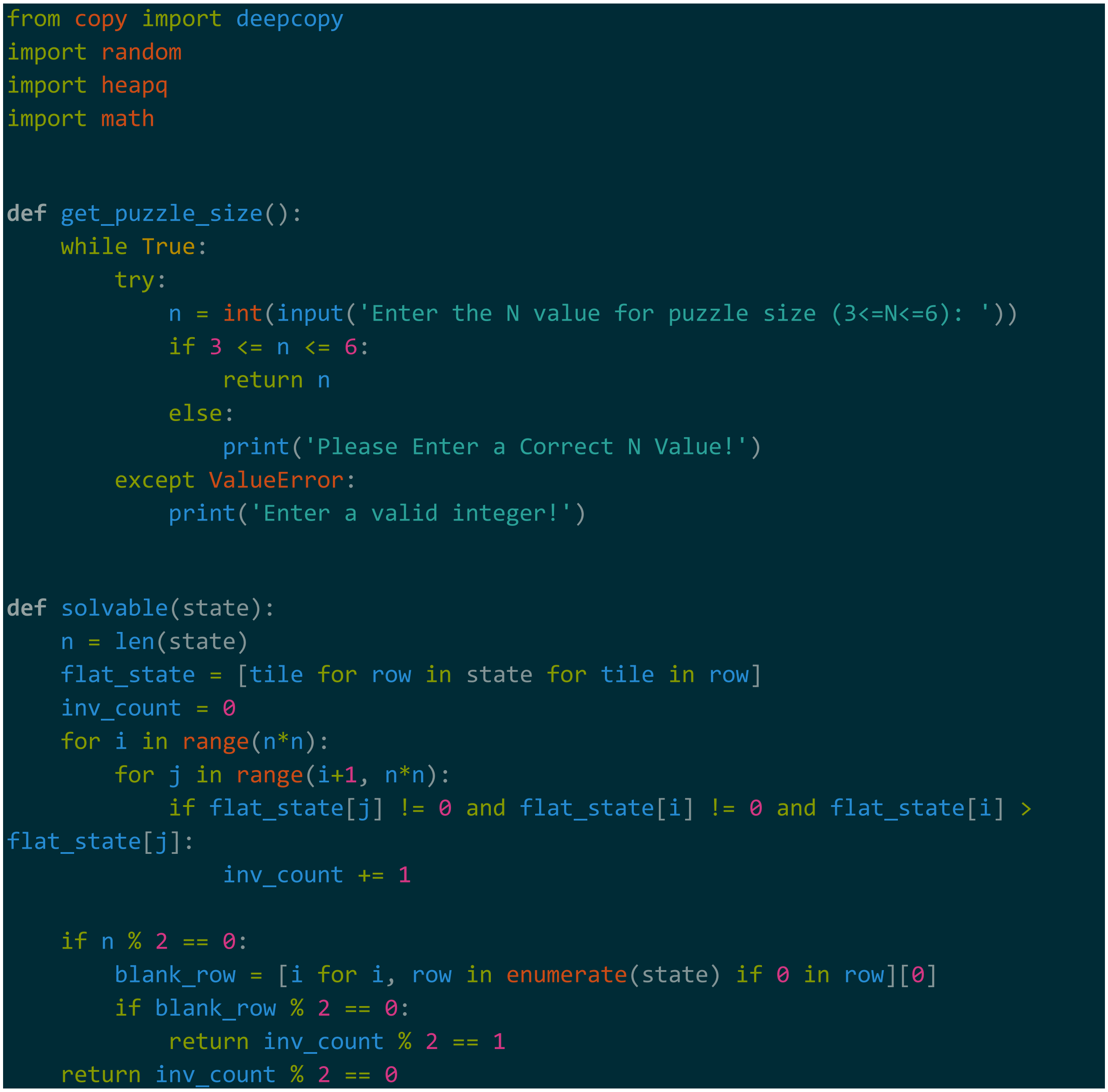
    - **Manhattan** is balanced in terms of solution depth and the number of states stored.
    - **Euclidean** offers a slightly better average solution depth than Manhattan but at a slightly higher memory cost.
    - **Misplaced Tiles**, although effective, tends to store more states compared to the other heuristics, potentially consuming more memory.

# Specifications:



| | |
|---|---|
| Processor | Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz   2.59 GHz |
| Installed RAM | 16.0 GB (15.8 GB usable) |

# Appendix:

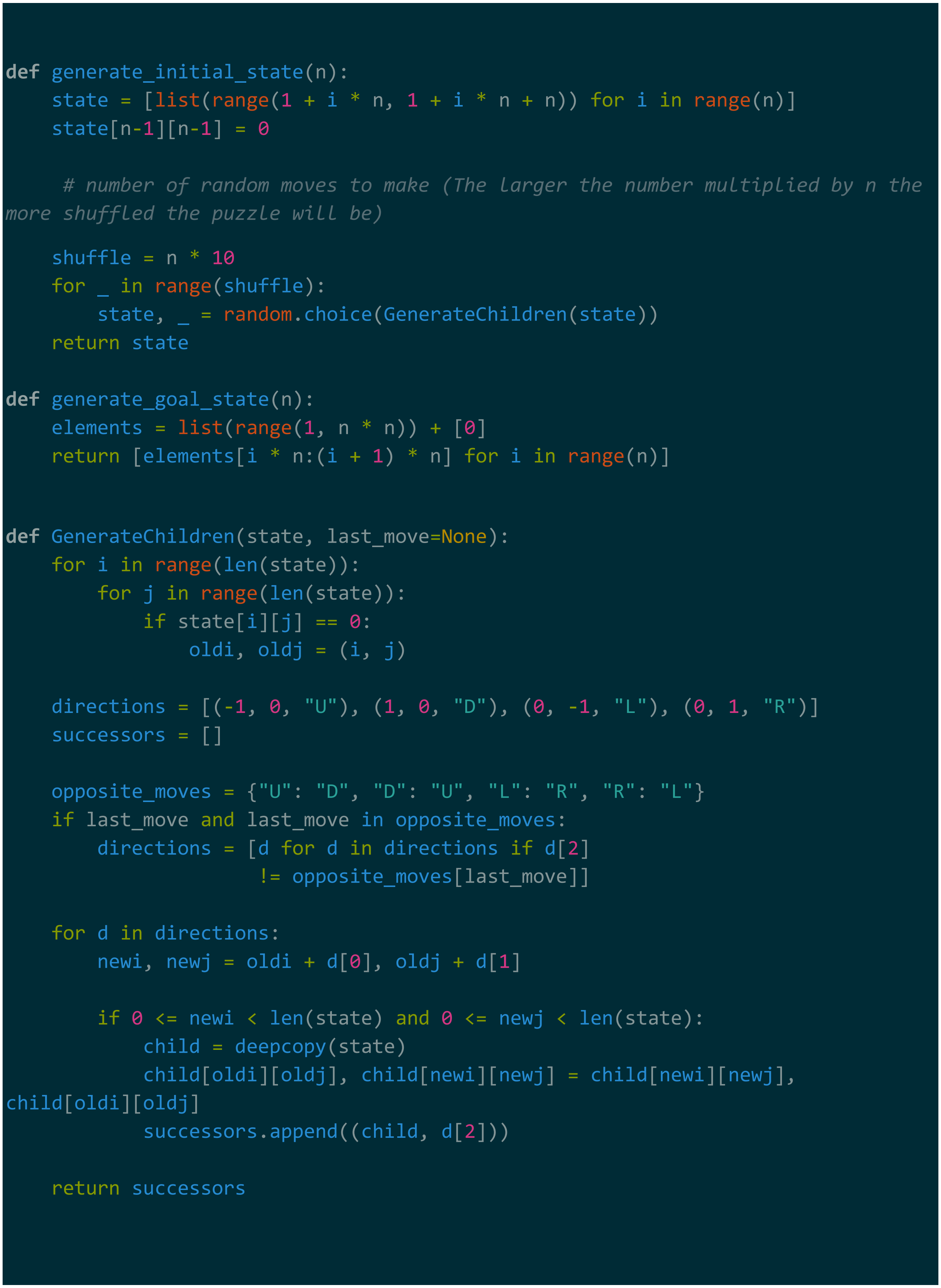


```
from copy import deepcopy
import random
import heapq
import math


def get_puzzle_size():
    while True:
        try:
            n = int(input('Enter the N value for puzzle size (3<=N<=6): '))
            if 3 <= n <= 6:
                return n
            else:
                print('Please Enter a Correct N Value!')
        except ValueError:
            print('Enter a valid integer!')


def solvable(state):
    n = len(state)
    flat_state = [tile for row in state for tile in row]
    inv_count = 0
    for i in range(n*n):
        for j in range(i+1, n*n):
            if flat_state[j] != 0 and flat_state[i] != 0 and flat_state[i] >
flat_state[j]:
                inv_count += 1

    if n % 2 == 0:
        blank_row = [i for i, row in enumerate(state) if 0 in row][0]
        if blank_row % 2 == 0:
            return inv_count % 2 == 1
    return inv_count % 2 == 0
```



```
def generate_initial_state(n):
    state = [list(range(1 + i * n, 1 + i * n + n)) for i in range(n)]
    state[n-1][n-1] = 0

    # number of random moves to make (The larger the number multiplied by n the
more shuffled the puzzle will be)

    shuffle = n * 10
    for _ in range(shuffle):
        state, _ = random.choice(GenerateChildren(state))
    return state

def generate_goal_state(n):
    elements = list(range(1, n * n)) + [0]
    return [elements[i * n:(i + 1) * n] for i in range(n)]


def GenerateChildren(state, last_move=None):
    for i in range(len(state)):
        for j in range(len(state)):
            if state[i][j] == 0:
                oldi, oldj = (i, j)

    directions = [(-1, 0, "U"), (1, 0, "D"), (0, -1, "L"), (0, 1, "R")]
    successors = []

    opposite_moves = {"U": "D", "D": "U", "L": "R", "R": "L"}
    if last_move and last_move in opposite_moves:
        directions = [d for d in directions if d[2]
                      != opposite_moves[last_move]]

    for d in directions:
        newi, newj = oldi + d[0], oldj + d[1]

        if 0 <= newi < len(state) and 0 <= newj < len(state):
            child = deepcopy(state)
            child[oldi][oldj], child[newi][newj] = child[newi][newj],
child[oldi][oldj]
            successors.append((child, d[2]))

    return successors
```

```
# This is a wrapper for A* method
def a_star_method(initial, goal, heuristic):
    return a_star(initial, goal, heuristic=heuristic)


def a_star(initial_state, goal_state, heuristic):
    visited = set()
    pq = [(heuristic(initial_state, goal_state), 0, initial_state, [])]
    max_states = 0

    while pq:
        max_states = max(max_states, len(pq))
        f, g, current_state, path = heapq.heappop(pq)

        if current_state == goal_state:
            return path, max_states

        hashed_state = tuple(map(tuple, current_state))
        if hashed_state in visited:
            continue

        visited.add(hashed_state)

        last_move = path[-1] if path else None
        for neighbor, move in GenerateChildren(current_state, last_move):
            hashed_neighbor = tuple(map(tuple, neighbor))
            if hashed_neighbor not in visited:
                new_g = g + 1
                new_f = new_g + heuristic(neighbor, goal_state)
                heapq.heappush(pq, (new_f, new_g, neighbor, path + [move]))

    return [], max_states


def misplaced_tiles(state, goal_state):
    return sum(1 for i in range(len(state)) for j in range(len(state)) if
state[i][j] != 0 and state[i][j] != goal_state[i][j])


def manhattan_distance(state, goal_state):
    distance = 0
    for num in range(1, len(state) * len(state)):
        x1, y1 = next((i, j) for i, row in enumerate(state)
```

```python
                      for j, cell in enumerate(row) if cell == num)
        x2, y2 = next((i, j) for i, row in enumerate(goal_state)
                      for j, cell in enumerate(row) if cell == num)
        distance += abs(x1 - x2) + abs(y1 - y2)
    return distance


def euclidean_distance(state, goal_state):
    distance = 0
    for num in range(1, len(state) * len(state)):
        x1, y1 = next((i, j) for i, row in enumerate(state)
                      for j, cell in enumerate(row) if cell == num)
        x2, y2 = next((i, j) for i, row in enumerate(goal_state)
                      for j, cell in enumerate(row) if cell == num)
        distance += math.sqrt((x1 - x2)**2 + (y1 - y2)**2)
    return distance


def generate_report(algorithm_name, algorithm, n, heuristic):

    splitter = "-" * 40 + "\n"

    with open("N3_mis.txt", "w") as f:
        solution_depths = []
        states_stored = []

        for i in range(10):
            initial = generate_initial_state(n)
            final = generate_goal_state(n)

            solution_sequence, max_states_stored = algorithm(
                initial, final, heuristic=heuristic)
            solution_depth = len(solution_sequence)

            f.write(f"Run {i + 1}:\n")
            f.write("Initial State:\n")
            for row in initial:
                f.write(" ".join(map(str, row)) + "\n")
            f.write("\nFinal State:\n")
            for row in final:
                f.write(" ".join(map(str, row)) + "\n")
            f.write(f"\nAlgorithm Used: {algorithm_name}\n")
            f.write("Solution Sequence:\n")
            f.write(" --> ".join(solution_sequence) + "\n")
            f.write(f"\nSolution Depth: {solution_depth}\n")
            f.write(f"Max States Stored: {max_states_stored}\n\n")

            solution_depths.append(solution_depth)
            states_stored.append(max_states_stored)

            f.write(splitter)

        f.write("Descriptive Statistics:\n")
        f.write(f"Minimum Solution Depth: {min(solution_depths)}\n")
        f.write(f"Maximum Solution Depth: {max(solution_depths)}\n")
        f.write(
            f"Average Solution Depth: {sum(solution_depths) /
len(solution_depths):.2f}\n")
        f.write(f"Minimum States Stored: {min(states_stored)}\n")
        f.write(f"Maximum States Stored: {max(states_stored)}\n")
        f.write(
            f"Average States Stored: {sum(states_stored) /
len(states_stored):.2f}\n")


def main():

    n = get_puzzle_size()

    # generate report used to get summary of the algorithm of choice
    # generate_report("A* with misplaced tiles",
    #                 a_star_method, n, misplaced_tiles)

    Initial_State = generate_initial_state(n)

    Goal_State = generate_goal_state(n)

    print("\nInitial State:")
    for row in Initial_State:
        print(row)
    print("\nGoal State:")
    for row in Goal_State:
        print(row)

    # This sets method to our wrapper function which calls A*
    method = a_star_method

    # Set the heuristic you want to use with A*
    heuristic_used = manhattan_distance
    print("\nFinding a Path Solution using A* with",
          heuristic_used.__name__, "heuristic . . .\n")

    path, max_states = method(
        Initial_State, Goal_State, heuristic=heuristic_used)

    if path:
        print(
            f"Solution found through A* using {heuristic_used.__name__}
heuristic!")
        print(" --> ".join(path))
        print(f"Solution depth: {len(path)}")
        print(f"Maximum number of states concurrently stored: {max_states}")
    else:
        print(
            f"No solution found using A* with {heuristic_used.__name__}
heuristic.")


if __name__ == "__main__":
    main()
```