Amer Aziz : 202169770

<u>The Sliding-Tile Puzzle</u>

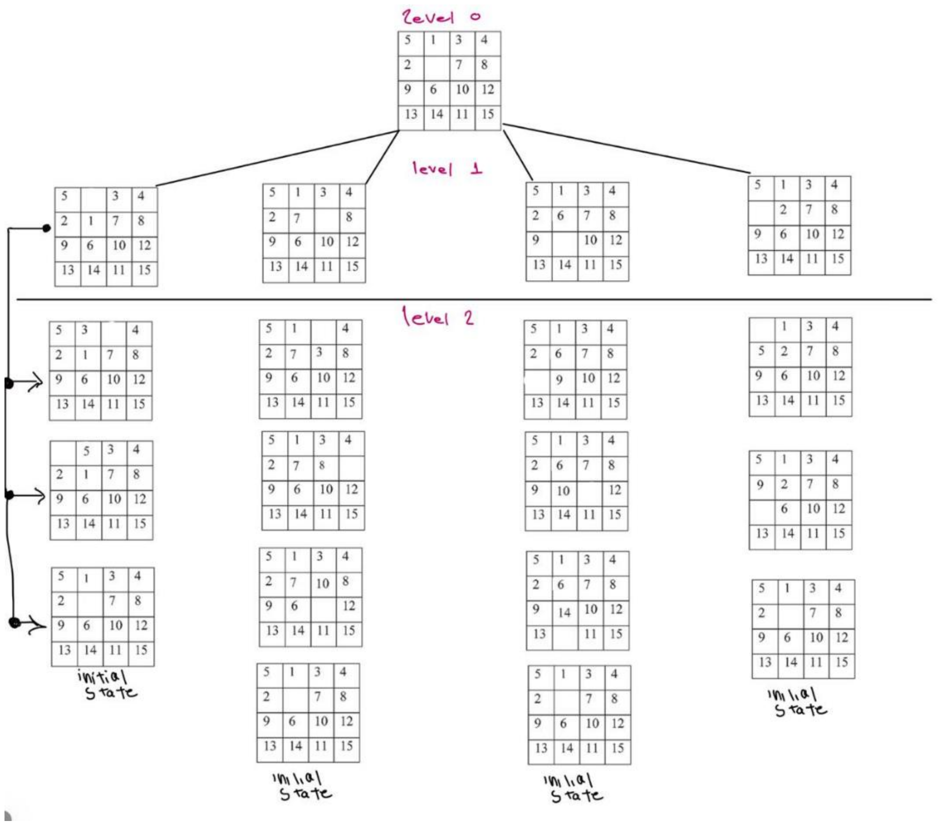
Consider the *m-puzzle* variant of the *sliding-tile puzzle* discussed in Section 3.2.1; $m = n^2 - 1$, $3 \leq n$, and n is an integer. Consider the formulation discussed in Section 3.2.1.

1. Comment on the *branching factor*.

   The branching factor in this problem is not fixed it depends on the position of the blank square or box. If it is on the middle the maximum branching factor for all n>2 where n is positive integer is 4. But some positions might get less than this for example if it is on the corners the branch factor reduce to 2 since there is 2 move to take (for example if the blank in top-right corner it can move to either *left* or *down*) and others positions which on sides of box (excluding corners) takes 3.

   So as it not uniform branch factor we have to take the average

2. Draw a diagram of the corresponding state space search tree (*depth* 0, 1, and 2) for *n=4*.



3. Implement ***BFS***, ***DFS***, and ***DFS with revisit check*** search solutions to the problem.

   a. Explain your representation.

   I represent the problem using data structures like 2D and 1D lists,

   and queues for BFS because it follow FIFO behavior to explore states level by level also I used Sets to keeps track of already explored states to prevent redundant processing.

   and for DFS I used stacks, and sets, the stacks Implements LIFO behavior to explore as deep as possible along each branch before backtracking. For DFS-revited node I initialize visited set like BFS to avoid revisited nodes

4. Run your program for $3 \leq n \leq 6$.

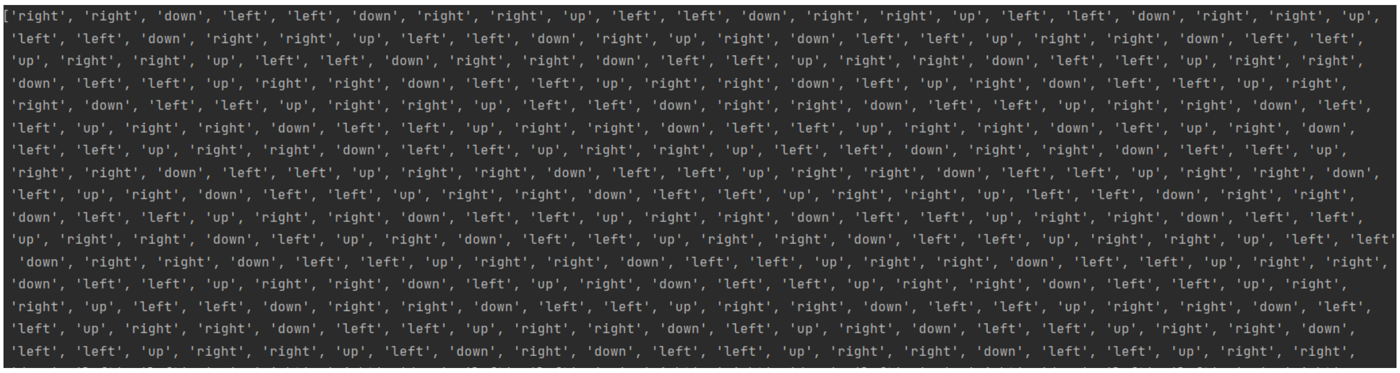
Amer Aziz : 202169770

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

   Can be done using '4. Run Benchmark' options in *console_solver.py* file

b. Report the *Initial State*, *Final State*, and the *Solution Sequence of Actions*.

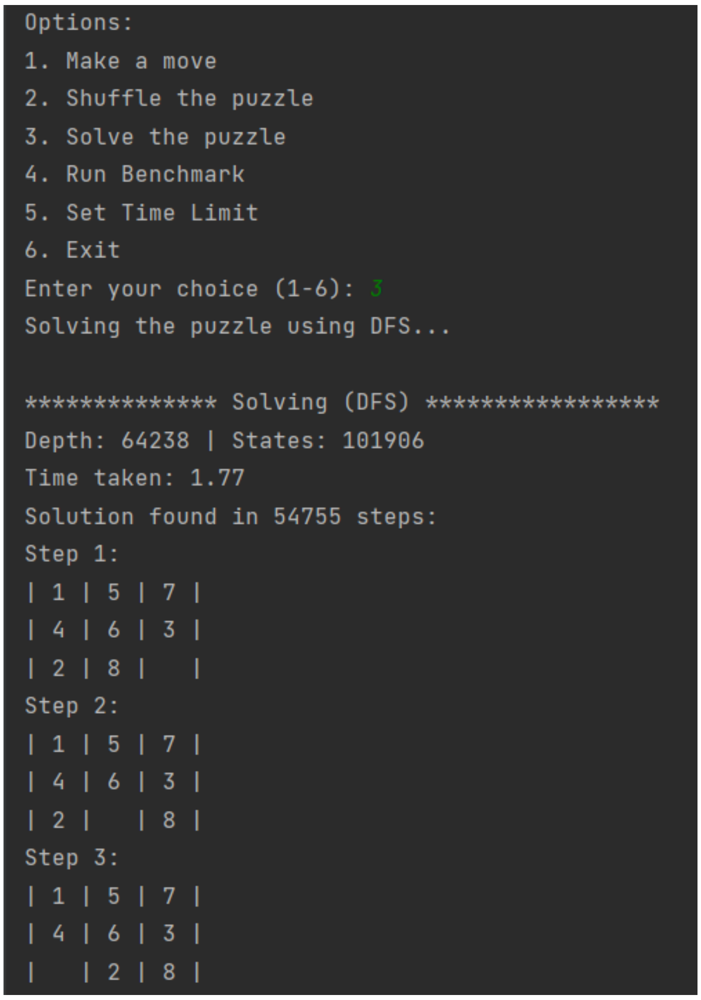
   Note: For the sequence of actions it can be viewd after solving the puzzle in Menu option '3. Solve the puzzle' but it cant show the sequence in the 10 running because the formatting issue (it can be applied but I don't recommend) instead I provide the number of steps/action needed.
   #to show the sequence uncomment lines 170,171 in *console_solver.py*
   here is the format issue: it hard to follow with 10 runs when large size puzzle applied



```
['right', 'right', 'down', 'left', 'left', 'down', 'right', 'right', 'up', 'left', 'left', 'down', 'right', 'right', 'up', 'left', 'left', 'down', 'right', 'right', 'up',
 'left', 'left', 'down', 'right', 'right', 'up', 'left', 'left', 'down', 'right', 'up', 'right', 'down', 'left', 'left', 'up', 'right', 'right', 'down', 'left', 'left',
 'up', 'right', 'right', 'up', 'left', 'left', 'down', 'right', 'right', 'down', 'left', 'left', 'up', 'right', 'right', 'down', 'left', 'left', 'up', 'right', 'right',
 'down', 'left', 'left', 'up', 'right', 'right', 'down', 'left', 'left', 'up', 'right', 'right', 'down', 'left', 'up', 'right', 'down', 'left', 'left', 'up', 'right',
 'right', 'down', 'left', 'left', 'up', 'right', 'right', 'up', 'left', 'left', 'down', 'right', 'right', 'down', 'left', 'left', 'up', 'right', 'right', 'down', 'left',
 'left', 'up', 'right', 'right', 'down', 'left', 'left', 'up', 'right', 'right', 'down', 'left', 'left', 'up', 'right', 'right', 'down', 'left', 'up', 'right', 'down',
 'left', 'left', 'up', 'right', 'right', 'down', 'left', 'left', 'up', 'right', 'right', 'up', 'left', 'left', 'down', 'right', 'right', 'down', 'left', 'left', 'up',
 'right', 'right', 'down', 'left', 'left', 'up', 'right', 'right', 'down', 'left', 'left', 'up', 'right', 'right', 'down', 'left', 'left', 'up', 'right', 'right', 'down',
 'left', 'up', 'right', 'down', 'left', 'left', 'up', 'right', 'right', 'down', 'left', 'left', 'up', 'right', 'right', 'up', 'left', 'left', 'down', 'right', 'right',
 'down', 'left', 'left', 'up', 'right', 'right', 'down', 'left', 'left', 'up', 'right', 'right', 'down', 'left', 'left', 'up', 'right', 'right', 'down', 'left', 'left',
 'up', 'right', 'right', 'down', 'left', 'up', 'right', 'down', 'left', 'left', 'up', 'right', 'right', 'down', 'left', 'left', 'up', 'right', 'right', 'up', 'left', 'left'
  'down', 'right', 'right', 'down', 'left', 'left', 'up', 'right', 'right', 'down', 'left', 'left', 'up', 'right', 'right', 'down', 'left', 'left', 'up', 'right', 'right',
 'down', 'left', 'left', 'up', 'right', 'right', 'down', 'left', 'up', 'right', 'down', 'left', 'left', 'up', 'right', 'right', 'down', 'left', 'left', 'up', 'right',
 'right', 'up', 'left', 'left', 'down', 'right', 'right', 'down', 'left', 'left', 'up', 'right', 'right', 'down', 'left', 'left', 'up', 'right', 'right', 'down', 'left',
 'left', 'up', 'right', 'right', 'down', 'left', 'left', 'up', 'right', 'right', 'down', 'left', 'up', 'right', 'down', 'left', 'left', 'up', 'right', 'right', 'down',
 'left', 'left', 'up', 'right', 'right', 'up', 'left', 'down', 'right', 'down', 'left', 'left', 'up', 'right', 'right', 'down', 'left', 'left', 'up', 'right', 'right',
```

   I used `Solution found in 57385 steps/actions:` instead when ruuning 10 Run Benchmark, **However** I provide alternative way when solving only a single or individual staes form the menu when selecting #3.

Amer Aziz : 202169770



```
Options:
1. Make a move
2. Shuffle the puzzle
3. Solve the puzzle
4. Run Benchmark
5. Set Time Limit
6. Exit
Enter your choice (1-6): 3
Solving the puzzle using DFS...

************** Solving (DFS) *****************
Depth: 64238 | States: 101906
Time taken: 1.77
Solution found in 54755 steps:
Step 1:
| 1 | 5 | 7 |
| 4 | 6 | 3 |
| 2 | 8 |   |
Step 2:
| 1 | 5 | 7 |
| 4 | 6 | 3 |
| 2 |   | 8 |
Step 3:
| 1 | 5 | 7 |
| 4 | 6 | 3 |
|   | 2 | 8 |
```

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

   Benchmark Results of 2 running:

   Depth - Min: 65021, Max: 65559, Average: 65218.33

   States Explored - Min: 131330, Max: 148590, Average: 137649.33

d. Across the 10 repetitions, report the descriptive statistics (*minimum*, *maximum*, and *average*) of the ***maximum number of states <u>concurrently</u> stored*** for each *n* for each solution.

   Same as above

5. Comment on the results.

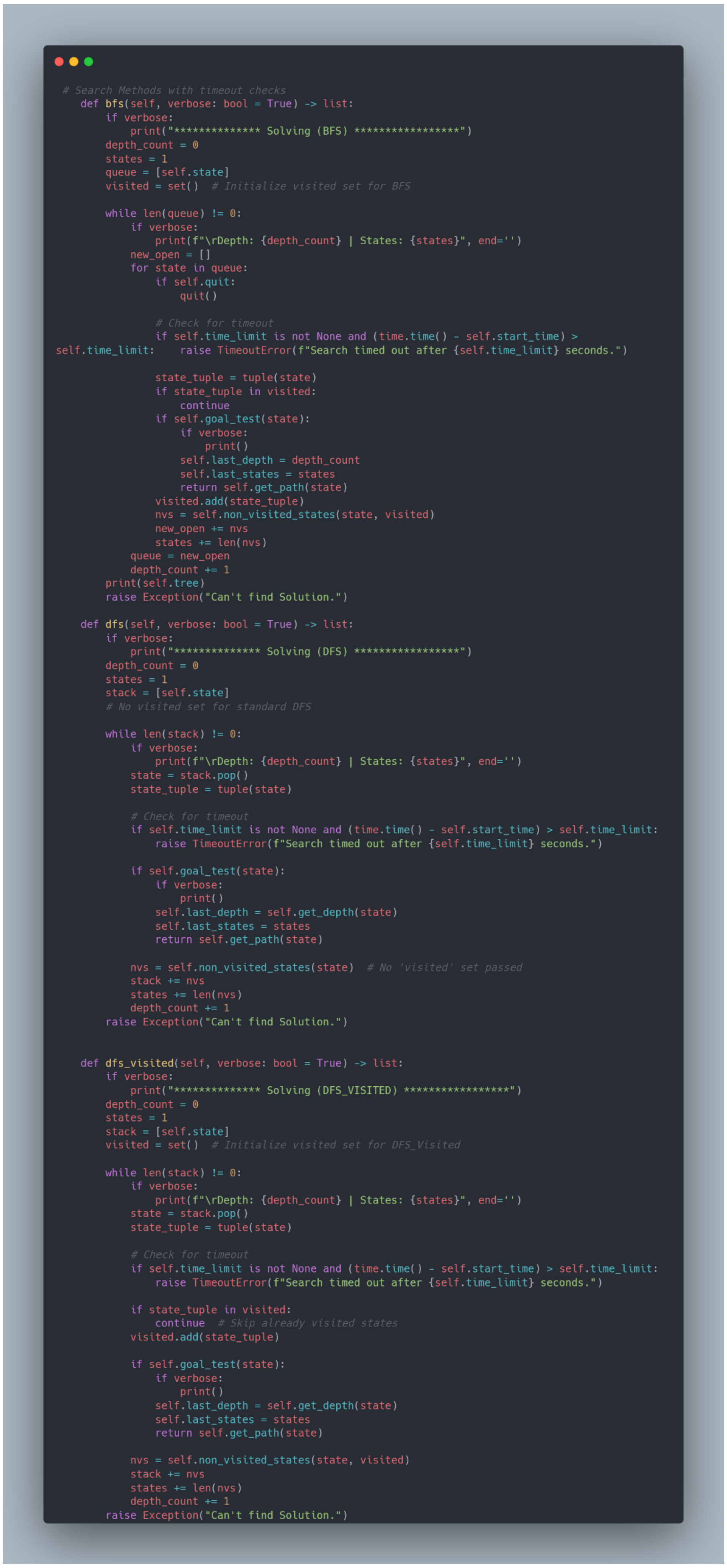
   For *BFS* it finds the solution when it exist in low depth easily but takes times when the size get bigger, and for *dfs* with revisited check nodes it might not noticeable difference in small board size but when the size become bigger the time difference between *dfs* and *df-visted* become bigger due to the additional processing of checking whether the node exist in a set or not while exploring nodes for *df-visted*. However with dfs and sometimes even with bfs there is no result appears ,since the goal-state is far way, so as additional solution a implement I time constrain (100 sec) to reach closer to the goal from if there is no result move to next state and if you didn't find solution with specific time then report no solution. To conclude, BFS is optimal for shallow puzzles but becomes nightmare for larger ones. DFS can be memory-inefficient without cycle prevention. DFS-Visited improves upon DFS by preventing cycle.

Notes:

Amer Aziz : 202169770

- This is an *individual-type* assignment, no group work is accepted.
- Submit the Report and the Code (as Appendix) in PDF.
- Source code also be submitted in a separate file(s). - No Zip Files.

Amer Aziz : 202169770

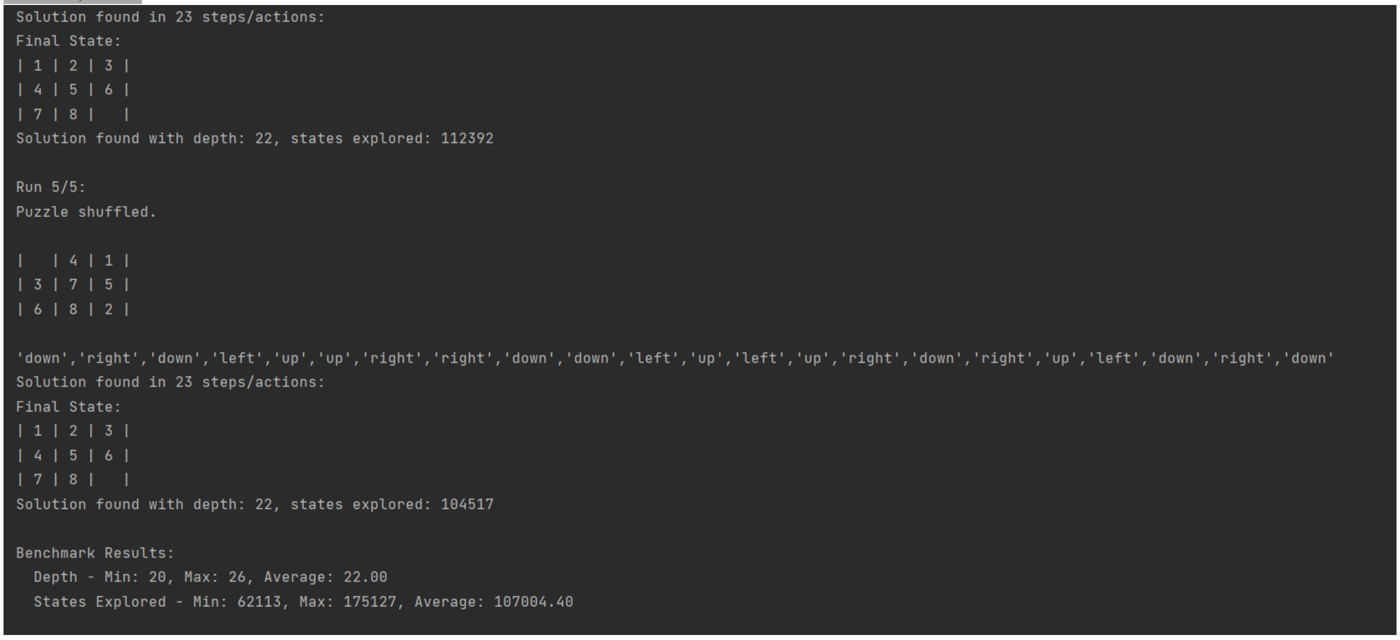


```python
# Search Methods with timeout checks
    def bfs(self, verbose: bool = True) -> list:
        if verbose:
            print("************** Solving (BFS) *****************")
        depth_count = 0
        states = 1
        queue = [self.state]
        visited = set()  # Initialize visited set for BFS

        while len(queue) != 0:
            if verbose:
                print(f"\rDepth: {depth_count} | States: {states}", end='')
            new_open = []
            for state in queue:
                if self.quit:
                    quit()

                # Check for timeout
                if self.time_limit is not None and (time.time() - self.start_time) >
self.time_limit:    raise TimeoutError(f"Search timed out after {self.time_limit} seconds.")

                state_tuple = tuple(state)
                if state_tuple in visited:
                    continue
                if self.goal_test(state):
                    if verbose:
                        print()
                    self.last_depth = depth_count
                    self.last_states = states
                    return self.get_path(state)
                visited.add(state_tuple)
                nvs = self.non_visited_states(state, visited)
                new_open += nvs
                states += len(nvs)
            queue = new_open
            depth_count += 1
        print(self.tree)
        raise Exception("Can't find Solution.")

    def dfs(self, verbose: bool = True) -> list:
        if verbose:
            print("************** Solving (DFS) *****************")
        depth_count = 0
        states = 1
        stack = [self.state]
        # No visited set for standard DFS

        while len(stack) != 0:
            if verbose:
                print(f"\rDepth: {depth_count} | States: {states}", end='')
            state = stack.pop()
            state_tuple = tuple(state)

            # Check for timeout
            if self.time_limit is not None and (time.time() - self.start_time) > self.time_limit:
                raise TimeoutError(f"Search timed out after {self.time_limit} seconds.")

            if self.goal_test(state):
                if verbose:
                    print()
                self.last_depth = self.get_depth(state)
                self.last_states = states
                return self.get_path(state)

            nvs = self.non_visited_states(state)  # No 'visited' set passed
            stack += nvs
            states += len(nvs)
            depth_count += 1
        raise Exception("Can't find Solution.")


    def dfs_visited(self, verbose: bool = True) -> list:
        if verbose:
            print("************** Solving (DFS_VISITED) *****************")
        depth_count = 0
        states = 1
        stack = [self.state]
        visited = set()  # Initialize visited set for DFS_Visited

        while len(stack) != 0:
            if verbose:
                print(f"\rDepth: {depth_count} | States: {states}", end='')
            state = stack.pop()
            state_tuple = tuple(state)

            # Check for timeout
            if self.time_limit is not None and (time.time() - self.start_time) > self.time_limit:
                raise TimeoutError(f"Search timed out after {self.time_limit} seconds.")

            if state_tuple in visited:
                continue  # Skip already visited states
            visited.add(state_tuple)

            if self.goal_test(state):
                if verbose:
                    print()
                self.last_depth = self.get_depth(state)
                self.last_states = states
                return self.get_path(state)

            nvs = self.non_visited_states(state, visited)
            stack += nvs
            states += len(nvs)
            depth_count += 1
        raise Exception("Can't find Solution.")
```

•

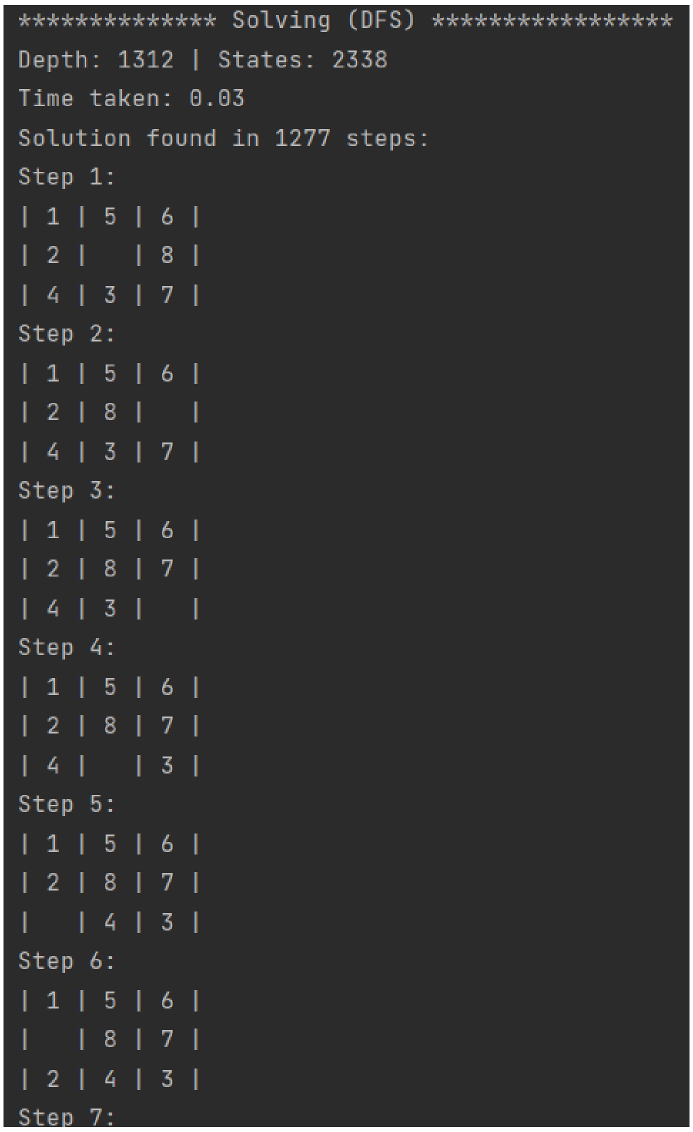
Amer Aziz : 202169770

N=3 ,BFS



```
Solution found in 23 steps/actions:
Final State:
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 |   |
Solution found with depth: 22, states explored: 112392

Run 5/5:
Puzzle shuffled.

|   | 4 | 1 |
| 3 | 7 | 5 |
| 6 | 8 | 2 |

'down','right','down','left','up','up','right','right','down','down','left','up','left','up','right','down','right','up','left','down','right','down'
Solution found in 23 steps/actions:
Final State:
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 |   |
Solution found with depth: 22, states explored: 104517

Benchmark Results:
  Depth - Min: 20, Max: 26, Average: 22.00
  States Explored - Min: 62113, Max: 175127, Average: 107004.40
```

N=3 , DFS

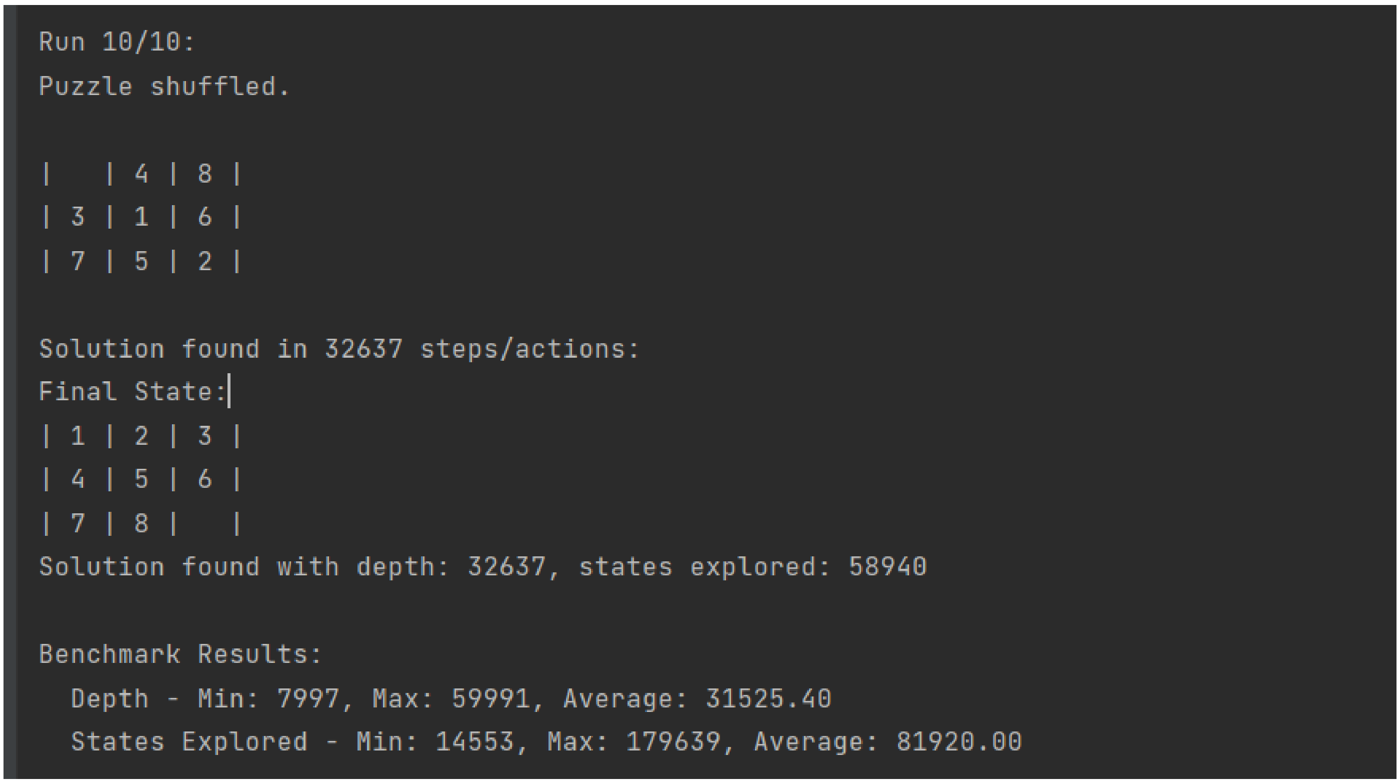


```
*************** Solving (DFS) *****************
Depth: 1312 | States: 2338
Time taken: 0.03
Solution found in 1277 steps:
Step 1:
| 1 | 5 | 6 |
| 2 |   | 8 |
| 4 | 3 | 7 |
Step 2:
| 1 | 5 | 6 |
| 2 | 8 |   |
| 4 | 3 | 7 |
Step 3:
| 1 | 5 | 6 |
| 2 | 8 | 7 |
| 4 | 3 |   |
Step 4:
| 1 | 5 | 6 |
| 2 | 8 | 7 |
| 4 |   | 3 |
Step 5:
| 1 | 5 | 6 |
| 2 | 8 | 7 |
|   | 4 | 3 |
Step 6:
| 1 | 5 | 6 |
|   | 8 | 7 |
| 2 | 4 | 3 |
Step 7:
```

Amer Aziz : 202169770

N=3 , DFS_reVisted

Note: since the sequence of actions huge I provide only #actions ,see ‘**4.b**’

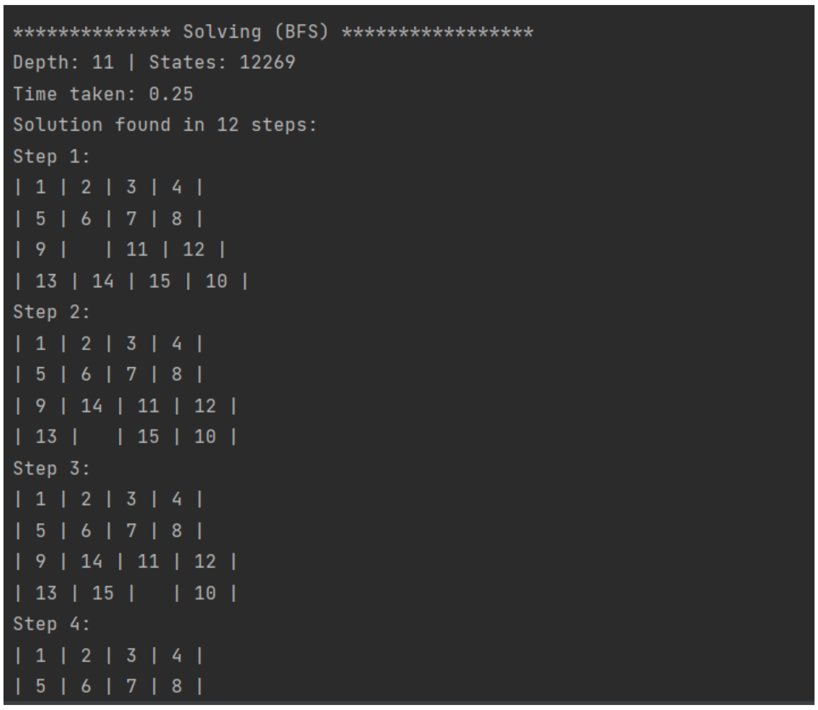


```
Run 10/10:
Puzzle shuffled.

|   | 4 | 8 |
| 3 | 1 | 6 |
| 7 | 5 | 2 |

Solution found in 32637 steps/actions:
Final State:
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 |   |
Solution found with depth: 32637, states explored: 58940

Benchmark Results:
  Depth - Min: 7997, Max: 59991, Average: 31525.40
  States Explored - Min: 14553, Max: 179639, Average: 81920.00
```

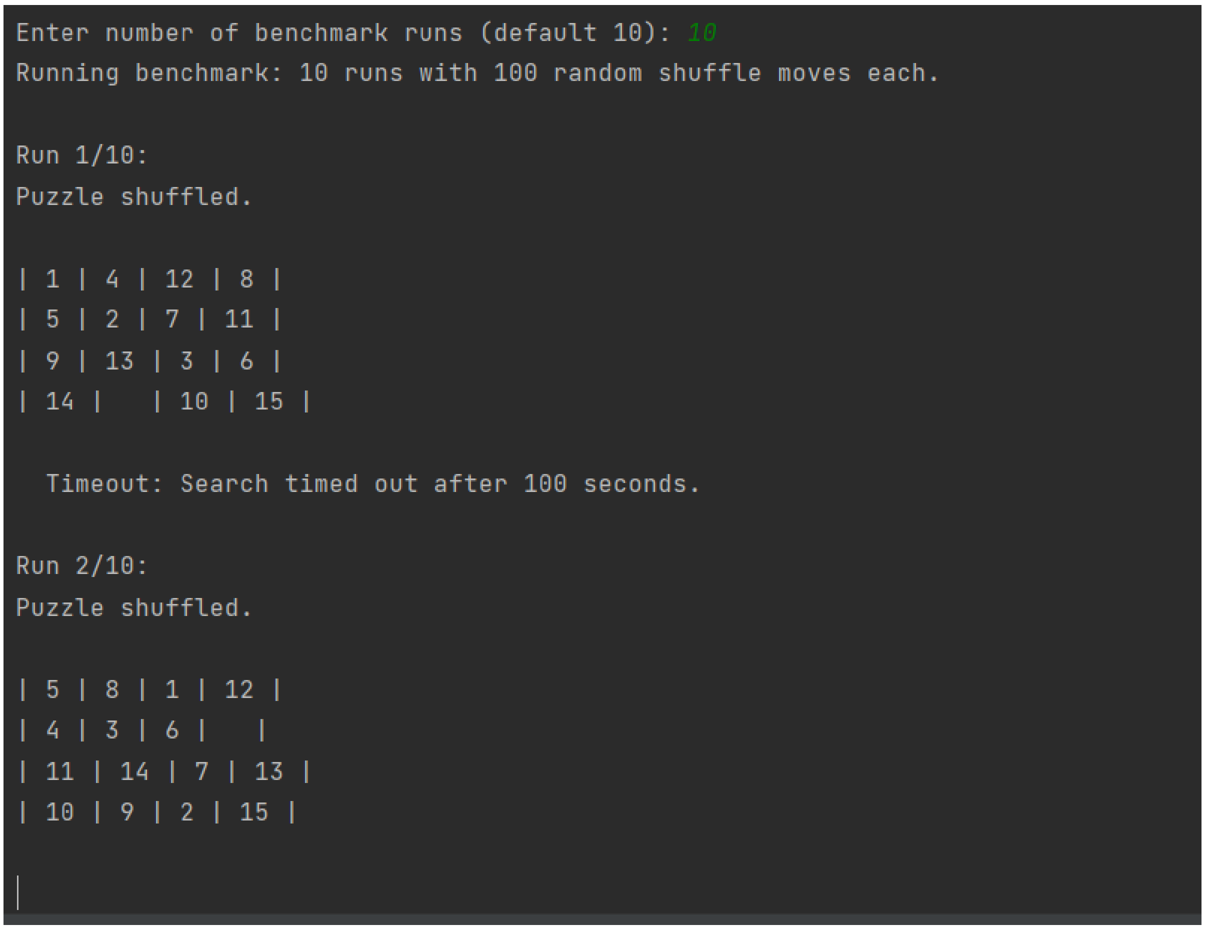
N=4 ,BFS

Simple case and random case. The remain algorithms is same for n=4,5…

Amer Aziz : 202169770



```
************** Solving (BFS) *****************
Depth: 11 | States: 12269
Time taken: 0.25
Solution found in 12 steps:
Step 1:
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 |   | 11 | 12 |
| 13 | 14 | 15 | 10 |
Step 2:
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 14 | 11 | 12 |
| 13 |   | 15 | 10 |
Step 3:
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 14 | 11 | 12 |
| 13 | 15 |   | 10 |
Step 4:
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
```



```
Enter number of benchmark runs (default 10): 10
Running benchmark: 10 runs with 100 random shuffle moves each.

Run 1/10:
Puzzle shuffled.

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

  Timeout: Search timed out after 100 seconds.

Run 2/10:
Puzzle shuffled.

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