8 puzzle

8/10/24.

```
goalstate = [ [1, 2, 3],
              [4, 5, 6],
                ~~8 Puzzle.~~
              [7, 8, 0] ] zero
def ~~manhattandistance~~ pu~~zzle~~. (state )
    for i in range.(0, 3)
        for j in range(0, 3)
            if state[i][j]==0
                return i,j      // zero state.


def manhattandistance.
    sum = 0
    for i in range(0, 3)
        for j in range(0, 3)
            distance = state[i]-[j.] - goalstate.
    sum+=distance.


def. generationmoves (state)
    directions = [ [-1, 0] , [0, 1] , [0, -1] , [.1, 0] ] ,
                   // up   ,  left    ,  ri

    ~~if~~ state [i] [j] == 0
       for x, y in directions.
           newposition = state.[i] [j] + dx , state [j] + dy

    ~~if~~ ~~ne~~ swap newstate and oldstate.


def dfs (state).

    ~~newst[]~~.
    vistied[]
    if state == goal state
        return True.
    newstate~~[]~~= generationmoves(state)
    if newstate != visited. -> min(manhattandistance
        dfs(newstate)
    visited.append(newstate)
```

```
Sort(manhattandistance(newstates))
~~if new~~
for newstate i in range of len(newstates):
    if newstate[i] = visited:
        dfs(newstate).
    ~~vis tri~~
    visited.append(newstate).
```

```
                -1,0
          0  -1  ·00   01
initial state.   1    2    4                1 -> 0
               10                           2 -> 0
                 3    6    5                3 -> 3
                 7         8                4 -> 3
                                            5 -> 1
                                            6 -> 1
 1  2  4        1    2    4       1  2  4   7 -> 0
 3  6  5        3         5       3  6  5   8 -> 1
    7  8        7    6    8       7  8
     M=10            M=10.        M=8,

                  1   2   4         1   2   4
                  3   6             3   6   5
                  7   8   5         7       8
                      M=9                M=9
```

```
from copy import deepcopy

directions = [(0,1), (1,0), (0,-1), (-1,0)]
goalstate = [[1,2,3], [4,5,6], [7,8,0]]

def zero(puzzle):
    for i in range(3):
        for j in range(3):
            if puzzle[i][j] == 0:
                return i,j
```

```
def generationmoves(state):
    new_states = [ ]
    p, q = findo (state)

  for x, y in directions:
      newx, newy = p + x , q + y

      if 0< = newx < 3 and 0< = newy < 3 :
           newstate = deepcopy(state)

           newstate[x][y], newstate [newx][newy]
                = newstate[newx] [newy], newstate[x][
          newstates . append (newstate)
  return newstates


def manhattan(state):
   dist = 0
  for i in range(3):
     for j in range(3):
        if state[i][j] != 0:
           xcurrent, ycurrent = i, j
           xgoal, ygoal = divmod(state[i][j]-1, 3
           dist~~ance~~ + = abs(xcurrent - xgoal)
                        + abs(ycurrent - ygoal)
 return dist


def dfs (state , visited):
    if state = = goalstate:
        return. True, [state]

  visited.append(state)

    new_states = generationmoves(state).
```

```
newstates.sort(key=lambda s: manhattan(s))

for newstate in newstates:
    if newstate not in visited:
        success, path = dfs(newstate, visited)
        if success:
            return True, [state]+path

return False, []

def print(puzzle):
    for row in puzzle:
        print(''.join(str(x) if x!=0 else
                ''  for x in row))
    print().

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

visited = []
success, solution = dfs(initialstate, visited)

if success:
    print("Solution found")
    for step in solution.
        print(step)
else:
    print("No solution")
```

```
Solution found!
1 2 3
4   6
7 5 ~~8~~
```

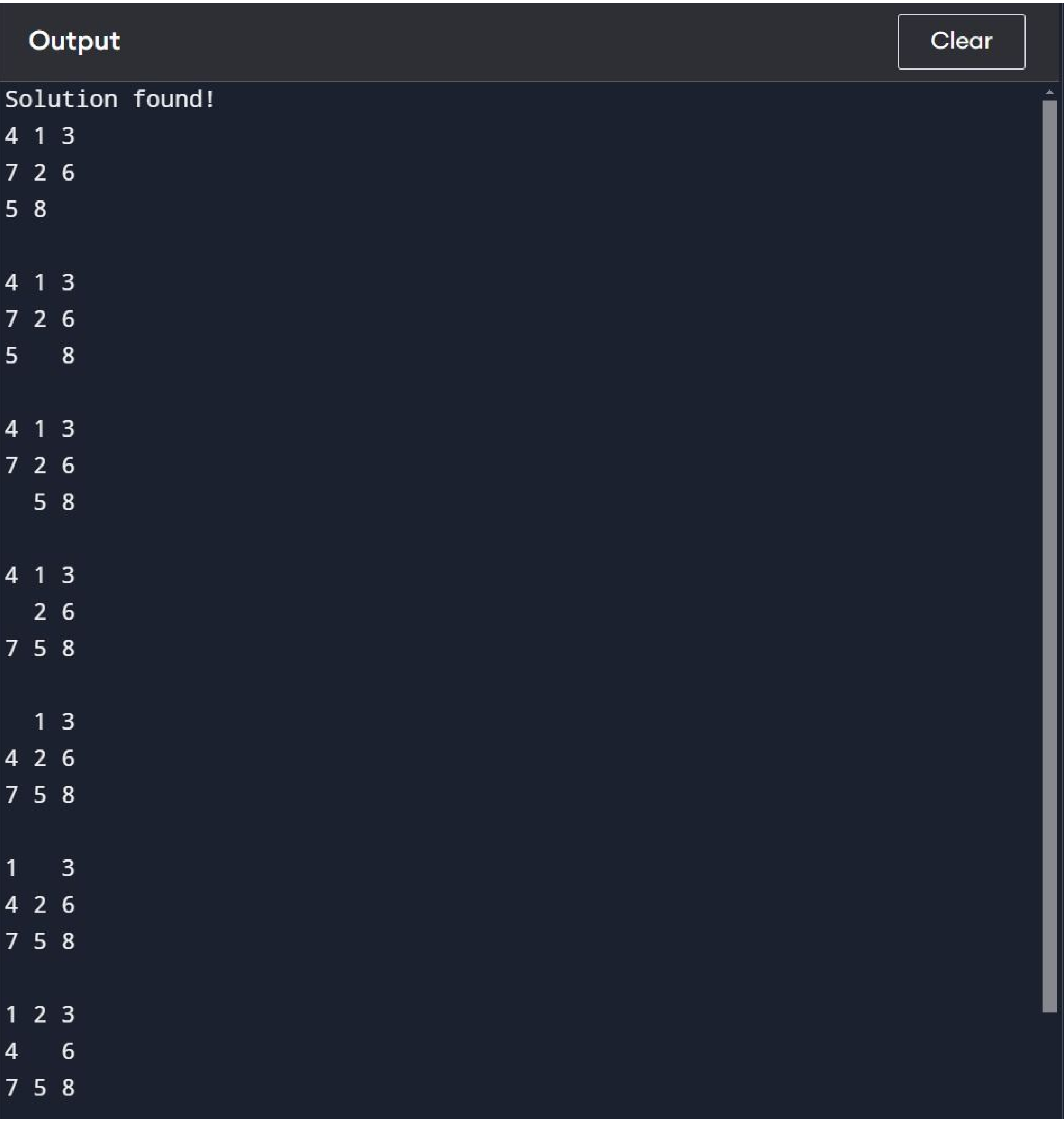
Solution found.

```
1 2 3
4 5 6
7   8

1 2 3
4 5 6
7 8
```

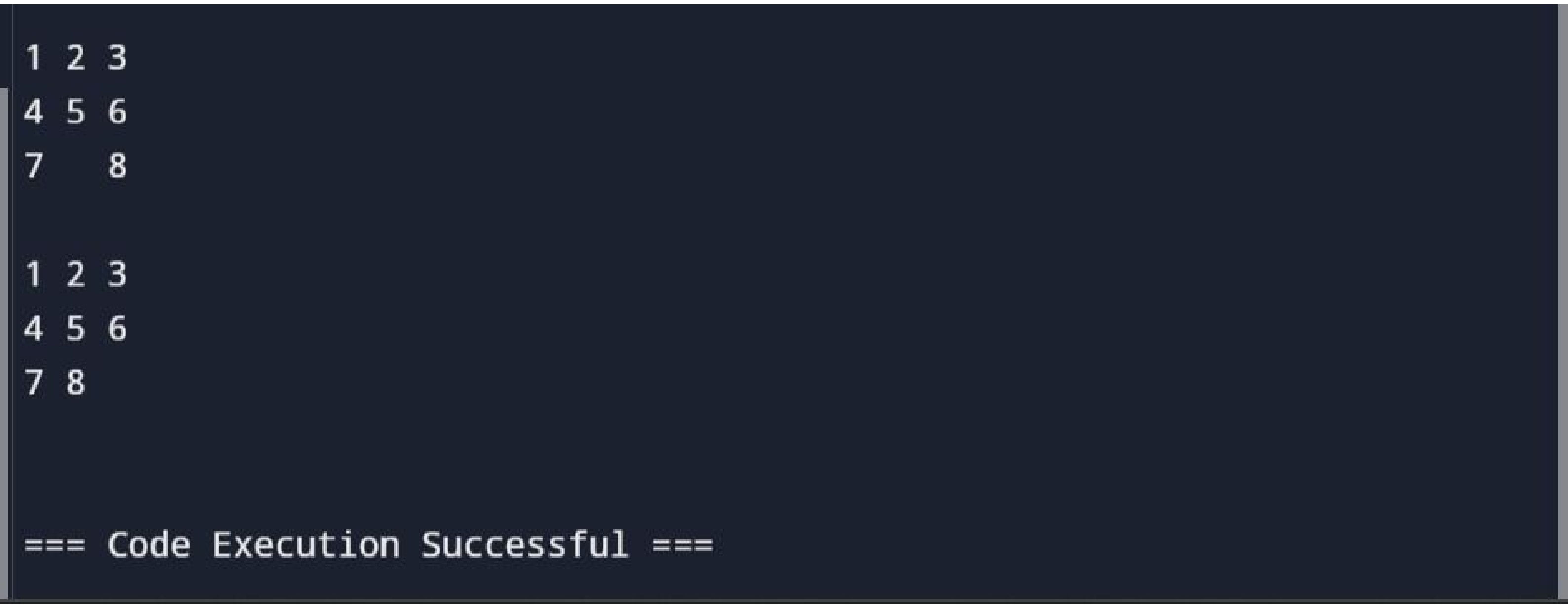
```
4 1 3
7 2 6
5 8

4 1 3
7 2 6
5   8

4 1 3
7 2 6
  5 8
```

```
4 1 3
  2 6
7 5 8

  1 3
4 2 6
7 5 8

1   3
4 2 6
7 5 8

1 2 3
4   6
7 5 8

1 2 3
4 5 6
7   8

1 2 3
4 5 6
7 8
```

Sneha S B



```
Solution found!
4 1 3
7 2 6
5 8  

4 1 3
7 2 6
5   8

4 1 3
7 2 6
  5 8

4 1 3
  2 6
7 5 8

  1 3
4 2 6
7 5 8

1   3
4 2 6
7 5 8

1 2 3
4   6
7 5 8
```



```
1 2 3
4 5 6
7   8

1 2 3
4 5 6
7 8


=== Code Execution Successful ===
```