LAB-03 08/10/24

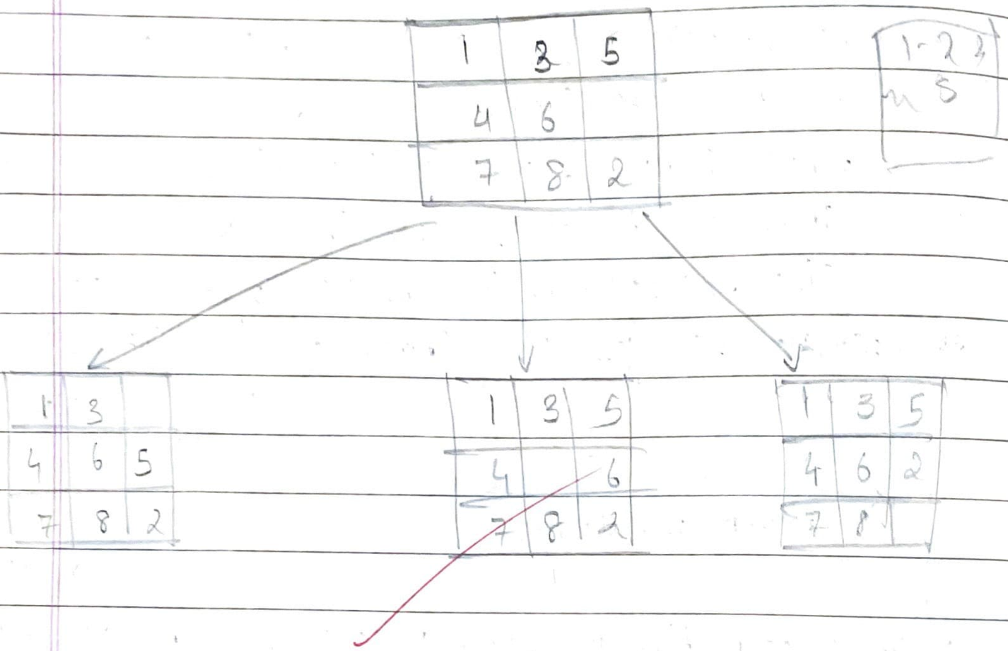
3. 8-Puzzle Problem

→ Manhattan - A* algorithm

1. Start at initial ~~array~~ list of the given matrix (3x3).
2. Compare each element in the index to the final state and see how far it is from the final state
3. c → cost of each space
   h → heuristic search
   f → total cost (c+h)
   } Intialization
4.
```
manhatten ( current_state, final_state)
    if the tile is not in blank tile
    (current x, current y) = position of the current tile
total dista~~goal x, goal y~~) = (current x - goal x) + (current y -
                                                        goal y)

    return totaldistance
```
5. If at each tile it matches the goal state we terminate the algorithm & traceback the path.

→ DFS

```
start_state = [ - - - ]
goal_state = [ - - - ]
stack = push (start_state)
F(i, j) = goal_state
visited_state = add . ~~start~~_state , moves=0
if start_state = goal_state
if (not in visited_state)
~~moves=0~~ {
        left = F(i, j-1)
        right = F(i, j+1)
        up = F(i-1, j)
        down = F(i+1, j)
      }
  }
  print moves;
```



1	3	5
4	6	
7	8	2

1	2	3
4	5	

1	3	
4	6	5
7	8	2

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

1	3	5
4	6	2
7	8	

Snehal B

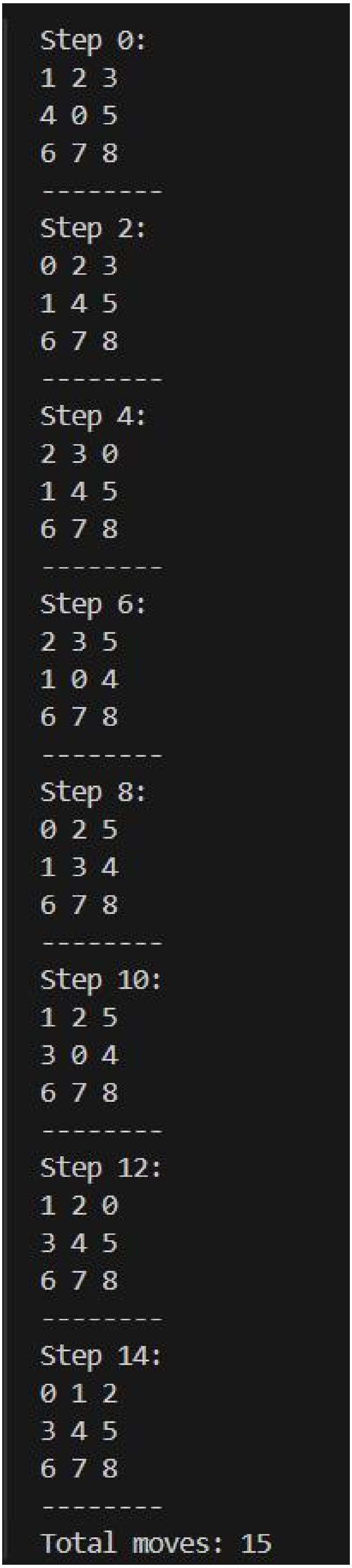
→ Code:

```
def Manhattan(puzzle, goal):
    dist = 0
    for i in range (9):
        if puzzle [i] != 0:
            goal_idx = goal.index (puzzle [i])
            dist += abs(i //3 - goal_idx //3) +
                    abs(i %3 - goal_idx %3)
    return dist


def manhattan_dfs( puzzle, goal, visited, path):
    if (puzzle == goal):
        return path
    visited.add (tuple (puzzle))
    idx = puzzle.index (0)
    moves = [(1, 3), (-1, 3), (3, 1), (-3, 1)]
    next_states = []
    for move, cond in moves:
        new_idx = idx + move
        if 0 <= new_index < 9 and (new_index // 3 ==
                        idx //3 or new_idx%3
                        == idx%3):
            new_puzzle = puzzle[:]
            new_puzzle [idx], new_puzzle [new_idx] =
            new_puzzle [new_idx], new_puzzle [idx]
            if tuple (new_puzzle) not in visited:
                next_states.append((new_puzzle, manhattan
                (new_puzzle, goal)))
    next_states.sort (key = lambda x: x[1])
    for state, _ in next_states:
        res = dfs_manhattan (state, goal, visited, path + [state])
    if res:
        return res
return None


def prettify (res):
    i=0
    for j in range (3):
        for k in range (3) :
            print (res[i], end=" ")
            i+=1
        print ("\n")
start = [1, 2; 3, 4 , 0, 5, 6, 7, 8 ]
goal  = [0, 1, 2, 3, 4, 5 , 6 , 7; 8 ]
result = dfs_manhattan (start, goal , set() [start])

for i in result:
    prettify(i)
    print(" - - -")
```

o) Output:-

```
1)  1 | 2 | 3        2)  1 | 2 | 3
    4 | 0 | 5            0 | 4 | 5
    6 | 7 | 8            6 | 7 | 8


3)  0 | 1 | 2
    3 | 4 | 5
    6 | 7 | 1
```

Sneha B 8/10/24



```
Step 0:
1 2 3
4 0 5
6 7 8
--------
Step 2:
0 2 3
1 4 5
6 7 8
--------
Step 4:
2 3 0
1 4 5
6 7 8
--------
Step 6:
2 3 5
1 0 4
6 7 8
--------
Step 8:
0 2 5
1 3 4
6 7 8
--------
Step 10:
1 2 5
3 0 4
6 7 8
--------
Step 12:
1 2 0
3 4 5
6 7 8
--------
Step 14:
0 1 2
3 4 5
6 7 8
--------
Total moves: 15
```