LAB-3

8/10/24

8 Puzzle Problem

DFS method -

```
puzzle board = [ [], [], [] ]
for i in range :
    for j in range [3]:
        puzzle board [j] [i] = input()

goal = [0, 1, 2, 3, 4, 5, 6, 7, 8]
moves = { left : -1 , right : 1 , up : -3 , down : 3 }
lis = [ ]

dfs ( board , final_move ):
    if ( board == goal ):
        return
    else :
        if ( board . move() not in lis ());
            board . move()
            lis . append ( board )
            dfs ( board )
```

Manhattan distance

Step -1 Initialise board

current_board = [2D list]

goal = $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix}$

Step -2

Calculate dist b/w current and targest position using sum of distances.

dist = abs(current_row - goal_row) + (current_col - goal_col)

Step -3

Sum all distances and return distance.

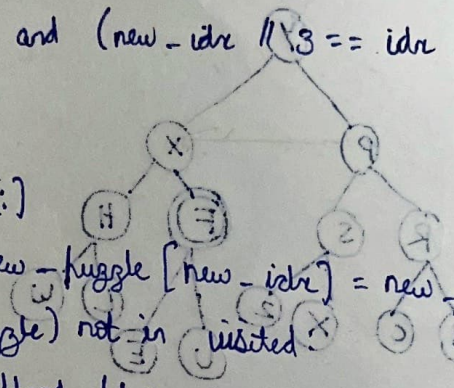
Code-

```
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)
                        % 3)
            return dist

def dfs_manhattan (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_idx < 9 and (new_idx // 3 == idx // 3 or new_idx %
    3 == idx % 3):
    new_puzzle = puzzle[:]
    new_puzzle[idx], new_puzzle[new_idx] = new_puzzle[new_idx],
        if tuple(new_puzzle) not in visited:
            next_states.append((new_puzzle, manhattan(new_puzzle, goal)))
for state, _ in next_states
    res = dfs_manhattan(state, goal, visited, path + [state])
    if res:
        return res
    return None.

def prettify(res): # To display output
```



Output -

1	0	2
3	7	5
4	6	8

0	1	2
3	7	5
4	6	8

0	1	2
3	4	5
6	7	8

Goal State.