Date __/__/__
Page 11

Lab 3

8 word puzzle using DFS and Manhattan distance

→	1	2	3	1	2	3
	7	6	5	4	5	6
	8	4		7	8	

In an 8 word puzzle we have 9 boxes with 8 with movable puzzle

DFS

start_state = [ ]

goal_state = = [ ]

stack = push (start_state)

visited_set = { }

moves = 0

```
f(i,j)
{
    visited_set.add(current_state)
    if (current_state == goal_state)
        return moves
    if (not in visited_set)
    {
        left = f(i, j-1)
        right = f(i, j+1)
        up = f(i-1, j)
        down = f(i+1, j)
    }
}
print moves
```

Date / /
Page

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 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], 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])
```

Date __/__/__
Page _____

```
if res:
    return res
return None

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

output

1

1	2	4
3	6	5
0	7	8

2

1	2	4
3	6	5
7	0	8

1st 2

3

3	1	2
0	4	5
6	7	8

4

0	1	2
3	4	5
6	7	8

last 2

8/10/24

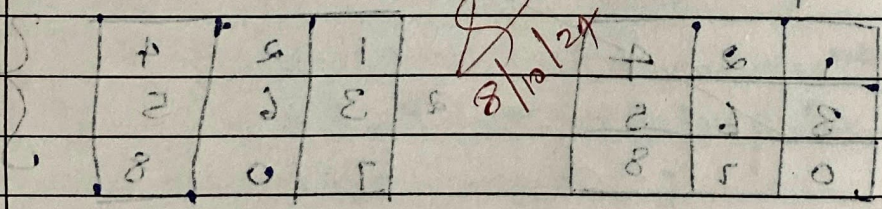
Date / /
Page

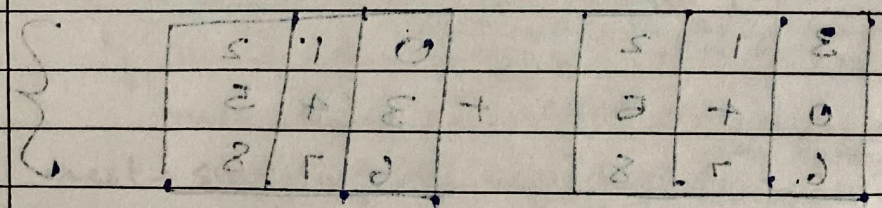
Manhattan distance

→ start at initial list of the given matrix (3x3)

→ compare each element in the index to the final state and see how far it is from the final state

manhatten (current_state, final_state)
    if the tile is not in blank tile
    [(current x, current Y)] = position of the current tile
    distance = (current x - goal x) + (current Y - goal Y)
return total distance





```
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
```