LAB 2

8 Puzzle problem

```
import heapq
import numpy as np

goal = [[0,1,2], [3,4,5], [6,7,8]]
vis = set()
q = []
parent_map = {}
move_map = {}

def manhattan(curr):
    ans = 0
    pos = {goal[i][j]: (i,j) for i in range(3) for j in range(3)}
    for i in range(3):
        for j in range(3):
            x,y = pos[curr[i][j]]
            ans += abs(i-x) + abs(j-y)
    return ans

def move(curr):
    x,y = [(i,j) for i in range(3) for j in range(3) if curr[i][j]==0][0]
    pos = [[0,-1,'left'], [-1,0,'up'], [1,0,'down'], [0,1,'right']]
    for dx,dy,direction in pos:
        nx,ny = x+dx, y+dy
        if 0<=nx<3 and 0<=ny<3:
            curr1 = [row[:] for row in curr]
            curr1[x][y], curr1[nx][ny] = curr1[nx][ny], curr1[x][y]
            tuple_curr1 = tuple(map(tuple,curr1))
            if tuple_curr1 not in vis:
                heapq.heappush(q, (manhattan(curr1), curr1))
                vis.add(tuple_curr1)
                parent_map[tuple(map(tuple,curr1))] = curr
```

```
move_map[tuple(map(tuple, curr))] = directions

def dfs(curr):
    vis.add(tuple(map(tuple, curr)))
    if curr == goal:
        return True
    moves(curr)
    if q:
        curr = heapq.heappop(q)[1]
        if dfs(curr):
            return True
    return False

def display_board(board):
    print("+---+---+---+")
    for row in board:
        print("|" + "|".join(str(x) if x!=0 else ' ' for x in row) + "|")
        print("+---+---+---+")

C = [[3,0,2], [5,1,7], [8,4,6]]
dfs(C)
result_path[0] = []
directions = []
state = goal
while state:
    result_path.append(state)
    directions.append(move_map.get(tuple(map(tuple, state)), None))
    state = parent_map.get(tuple(map(tuple, state)))

for ind, (state, direction) in enumerate(reversed(list(zip(result_path,
    directions)))):
    print("Step", ind:")
    display_board(state)
    if ind == 0:
```

```
        print("initial state")
    if drawing:
        print(f"Move empty space to {row} {col}")
    print()
print(f"steps taken: {len(result_path)-1}")
```

O/P

Trying depth limit 0
1
2
3
4
5

Solution found at depth 5 with path:

1 2 3 5 6 0 4 7 8	1 2 3 5 6 0 4 7 8
1 2 3 5 0 6 4 7 8	1 2 3 4 5 6 7 0 8
1 2 3 0 5 6 4 7 8	1 2 3 4 5 6 7 8 0

True

URBAN EDGE

Iterative Deepening Depth first search :

```
class Graph:
    def __init__(self, vertices):
        self.V = vertices
        self.graph = {i: [] for i in range(vertices)}

    def add_edge(self, u, v):
        self.graph[u].append(v)

    def dls(self, src, target, limit):
        if src == target:
            return True
        if limit <= 0:
            return False
        for neighbor in self.graph[src]:
            if self.dls(neighbor, target, limit - 1):
                return True
        return False

    def iddfs(self, src, target, max_depth):
        for depth in range(max_depth + 1):
            if self.dls(src, target, depth):
                return True
        return False

g = Graph(7)
g.add_edge(0, 1)
g.add_edge(0, 2)
g.add_edge(1, 3)
g.add_edge(1, 4)
g.add_edge(2, 5)
g.add_edge(2, 6)
src = 0
target = 6
```

max_depth = 3

If dfs(src, target, max_depth):
    print("Target (target) found within depth (max_depth)")
else:
    print("Target (target) not found within depth (max_depth)")

O/P

Target 6 found within depth 3.

Solutions for the search problems & techniques.