8/10/24

Lab - 3

8- Puzzle

→ **Algorithm :-**

Step 1 :- Goal state and moves

goal state = [[1, 2, 3]
[4, 5, 6]
[7, 8, -]]

moves = [(-1, 0) up
(1, 0) = Down
(0, -1) = Left
(0, 1) = Right]

Step 2 :- To Calculate manhattan distance

```
def manhattan_distance (state)
    for i in range (3)
        for j in range (3)
            if state [i] [j] = '-' :
                goal_i, goal_j = divmod [state [i] [j] - 1, 3]
                distance += abs (i - goal_i) + abs (j - goal_j)
    return distance
```

Step 3 : Check if curnt state matches goal state

```
def curent (State):
    return goal == State == State
```

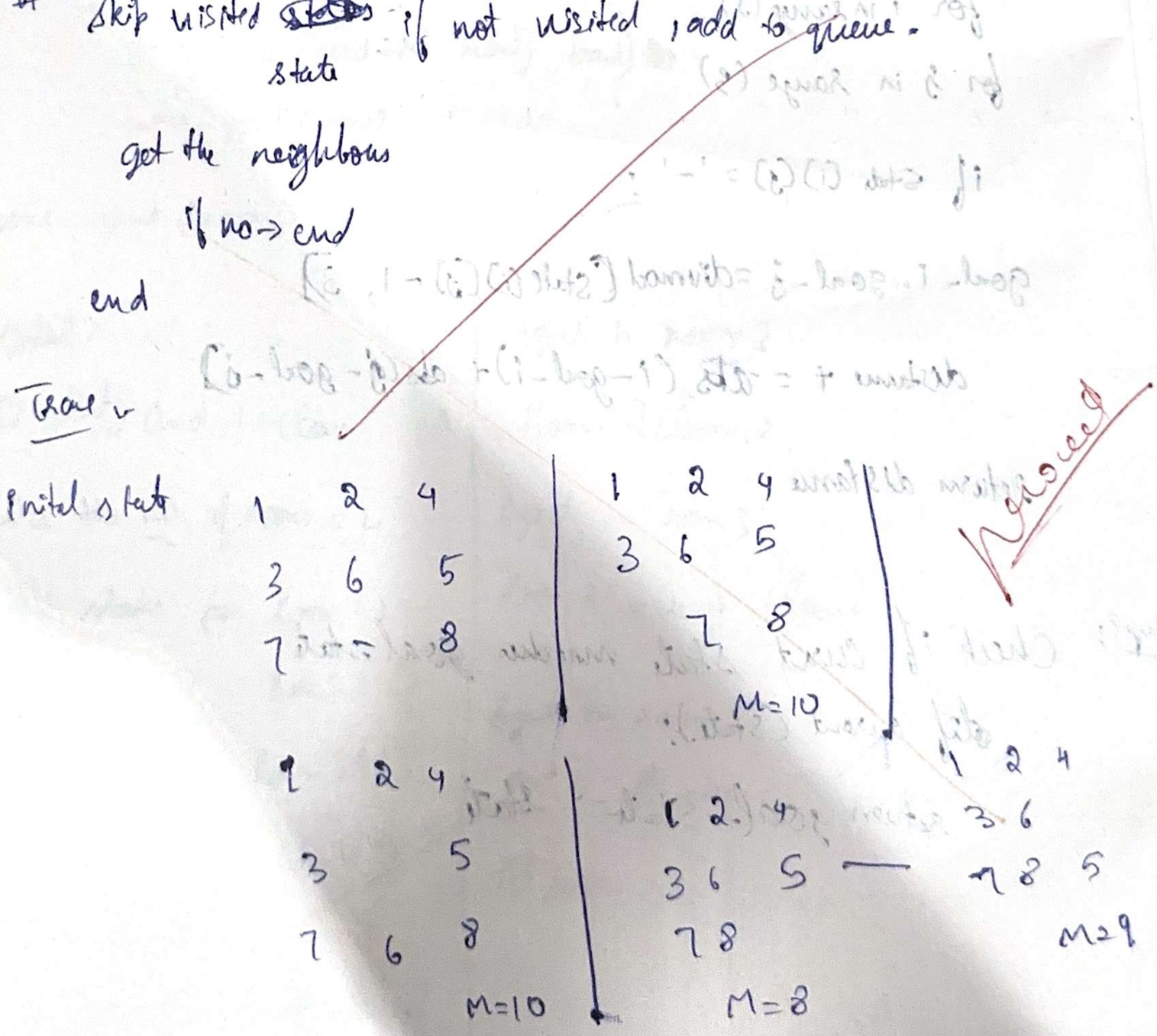
if board[i][j] == '-' ⇒ check for all 4 directions using the move matrix.

Return the neighbors array.

Step 5 :-

```
def dfs(state):
    queue = deque([(state, state)])
    visited = set()
    while queue
        current_state, path = queue.popleft()
# check if current state is goal state
# skip visited state, if not visited, add to queue.
    get the neighbours
        if no -> end
    end
```

Trace :-




```
from collections import deque

GOAL_STATE = [[1, 2, 3],
              [4, 5, 6],
              [7, 8, '-']
             ]

MOVES = [(-1, 0),
         (1, 0),
         (0, -1)
         (0, 1)
        ]

def manhattan_distance(state):
    distance = 0
    for i in range(3):
        for j in range(3):
            if state[i][j] != '-' :
                goal_i, goal_j = di
                distance += abs(i - j
    return distance

def get_neighbors(state):
    neighbors = []
    for i in range(3):
        for j in range(3):
            if state[i][j] == '
                for move in Mo
                    new_i, ne
                if 0 <= new_i <
                    new_state = [row
                    new_state[i][j],
```

```
from collections import deque

GOAL_STATE = [ [1, 2, 3],
               [4, 5, 6],
               [7, 8, '-'] ]

MOVES = [(-1, 0),
         (1, 0),
         (0, -1)
         (0, 1)
        ]

def manhattan_distance(state):
    distance = 0
    for i in range(3):
    for j in range(3):
        if state[i][j] != '-':
            goal_i, goal_j = divmod(state[i][j]-1, 3)
            distance += abs(i - goal_i) + abs(j - goal_j)
    return distance

def get_neighbors(state):
    neighbors = []
    for i in range(3):
        for j in range(3):
            if state[i][j] == '-':
                for move in MOVES:
                    new_i, new_j = i + move[0], j + move[1]
                    if 0 <= new_i < 3 and 0 <= new_j < 3:
                        new_state = [row[:] for row in state]
                        new_state[i][j], new_state[new_i][new_j] = new_state
                        [new_i][new_j],
```

```
            new_state[i][j]

        neighbors.append(new_state)

    return neighbors

def dfs(state):
    queue = deque([[state, []]])
    visited = set()

    while queue:
        current_state, path = queue.popleft()
        if is_goal_state(current_state):
            return path

        if tuple(map(tuple, current_state)) in visited:
            continue

        visited.add(tuple(map(tuple, current_state)))

        for neighbor in get_neighbours(current_state):
            queue.append((neighbour, path + [neighbour]))

    return None

initial_state = [ [4, 1, 3],
                  [7, 2, 6],
                  [5, 8, -1]
                ]

path = dfs(initial_state)
```

```
if path :
    print ("Solution found :")
    for state in path :
        for row in state :
            print (row)
    print ()
else :
    print ("No solution found.")
```

Output :

$$\begin{bmatrix} 4,1,3 \\ 7,2,6 \\ 5,8,- \end{bmatrix} \rightarrow \begin{bmatrix} 4,1,3 \\ 7\ 2\ 6 \\ 5\ -\ 8 \end{bmatrix} \rightarrow \begin{bmatrix} 4 & 1 & 3 \\ - & 2 & 6 \\ 7 & 5 & 8 \end{bmatrix}$$

$$\downarrow$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & - & 8 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & - & 3 \\ 4 & 2 & 6 \\ 7 & 5 & 8 \end{bmatrix} \leftarrow \begin{bmatrix} - & 1 & 3 \\ 4 & 2 & 6 \\ 7 & 5 & 8 \end{bmatrix}$$

$$\downarrow$$

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

Lab 8
8/10/24