Program 2 : 8 Puzzle Breadth First Search Algorithm

Code:

```
def bfs(src,target):
    queue = []
    queue.append(src)

    exp = []

    while len(queue) > 0:
        source = queue.pop(0)
        exp.append(source)

        print(source)

        if source==target:
            print("success")
            return

        poss_moves_to_do = []
        poss_moves_to_do = possible_moves(source,exp)

        for move in poss_moves_to_do:

            if move not in exp and move not in queue:
                queue.append(move)
def possible_moves(state,visited_states):
    #index of empty spot
    b = state.index(-1)

    #directions array
    d = []
    #Add all the possible directions

    if b not in [0,1,2]:
        d.append('u')
    if b not in [6,7,8]:
        d.append('d')
    if b not in [0,3,6]:
        d.append('l')
    if b not in [2,5,8]:
        d.append('r')
```

```
    # If direction is possible then add state to move
    pos_moves_it_can = []

    # for all possible directions find the state if that move is played
    ### Jump to gen function to generate all possible moves in the given directions

    for i in d:
        pos_moves_it_can.append(gen(state,i,b))

    return [move_it_can for move_it_can in pos_moves_it_can if move_it_can not in visited_states]
def gen(state, m, b):
    temp = state.copy()

    if m=='d':
        temp[b+3],temp[b] = temp[b],temp[b+3]

    if m=='u':
        temp[b-3],temp[b] = temp[b],temp[b-3]

    if m=='l':
        temp[b-1],temp[b] = temp[b],temp[b-1]

    if m=='r':
        temp[b+1],temp[b] = temp[b],temp[b+1]


    # return new state with tested move to later check if "src == target"
    return temp
src = [1,2,3,-1,4,5,6,7,8]
target = [1,2,3,4,5,-1,6,7,8]
bfs(src, target)
```

Observation:

Date: 24/11/23

8 Puzzle Problem Using BFS

In the problem, the square with have N+1 tiles where N = 8, 15, 24 so on

N=8 means square will have 9 tiles (3 rows and 3 columns)

In the problem, initial state will be given and and we have do reach goal state.

Suppose

Initial state

1	2	3
	4	6
7	5	8

Goal state

1	2	3
4	5	6
7	8	

Rules

- Empty space can move in 4 directions
  1) Up 2) Down 3) Right 4) Left
- It cannot move diagonally
- It can only take one step at a time

O	X	O
X	#	X
O	X	O

Tiles at O — no. of possible moves=2
Tiles at X — no of possible moves=3
Tile at # — no. of possible moves=4

Using Breadth First Search, which is uninformed search. This problem is solved by non heuristic approach.

Complexity $O(b^d)$ where
b — branching factor
d depth

Date : 24/11/23

For 8 puzzle

Branching factor b = all possible moves of empty tile at each position / no. of tiles

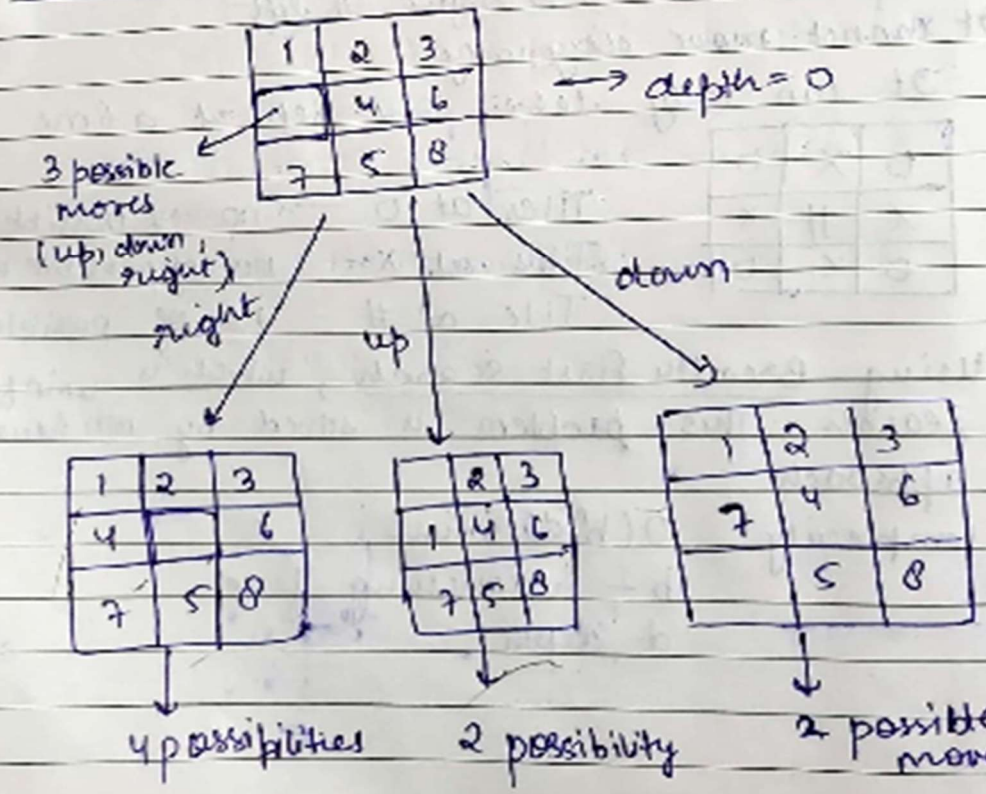
$= \frac{24}{9} = 2.67 \sim 3$

depth factor = initially 0

As we explore the nodes, depth factor increases

Every time, we check all possible moves of tiles branch it and consider every case and finally to goal state

For example:



Output:

```
Kanjika Singh-1BM21CS086
1 | 2 | 3
4 | 5 | 6
0 | 7 | 8

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

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

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

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

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

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

success
```