8/12/2023

Lab-5

→ 8 Puzzle problem using Iterative Deepening Search Algorithm:

(1) Node (data, level) : initialize node with puzzle state & level.

(2) def puzzle:
→ accept () ← start & goal state
→ dls() (node, goal, depth) ← perform dls

(3) def DLS (node, goal, depth):
if (current_state == goal):
return solution
else:
generate child node, recursively call with increase level.

(4) def IDS (start, goal)
→ Start with depth : 0
→ Repeat until goal is found :
→ perform DLS with current depth
→ if sol" found ← exit
increment depth

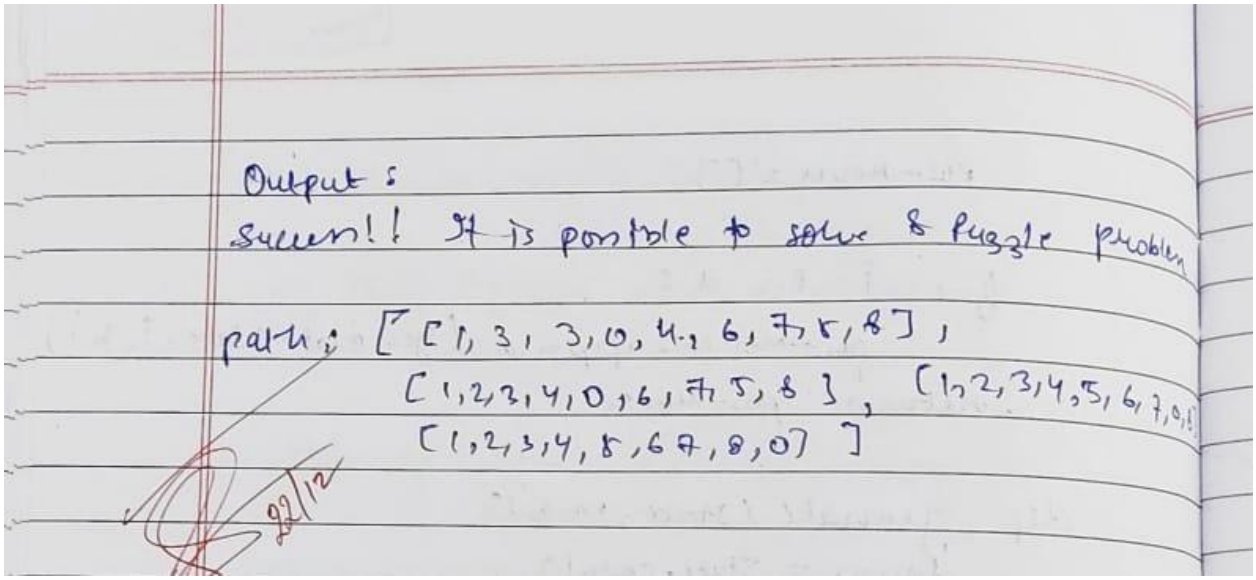
(5) → Create puzzle instance
→ Call IDS (start, goal)

→ Code Iterative_Deepening_search :-

```
→ def id_dfs(puzzle, goal, get_moves)
      import intertools

    def dfs(route, depth):
        if (depth == 0:
            return
        if route[-1] == goal:
            return route
        for move in get_moves(route[-1]):
            if move not in route:
            next_route = dfs(route + [move], dfs
            if next_route:
                return next_route
    for depth in intertools.count():
        route = dfs([puzzle], depth)
        if route:
            return route

def possible_moves(state):
    b = state.index(0)
    d = []
    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')
```

→ Code Iterative_Deepening_search :-

```
→ def id_dfs(puzzle, goal, get_moves)
      import intertools

    def dfs(route, depth):
        if (depth == 0:
            return
        if route[-1] == goal:
            return route
        for move in get_moves(route[-1]):
            if move not in route:
            next_route = dfs(route + [move], dfs
            if next_route:
                return next_route
    for depth in intertools.count():
        route = dfs([puzzle], depth)
        if route:
            return route

def possible_moves(state):
    b = state.index(0)
    d = []
    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')
```



Output:

```
Success!! It is possible to solve 8 Puzzle problem
Path: [[1, 2, 3, 0, 4, 6, 7, 5, 8], [1, 2, 3, 4, 0, 6, 7, 5, 8], [1, 2, 3, 4, 5, 6, 7, 0, 8], [1, 2, 3, 4, 5, 6, 7, 8, 0]]
```

```
[ ] Start coding or generate with AI.
```