8-12-23

8-Puzzle problem using ID-DFS:

Code:

```
def id_dfs(puzzle, goal, get_moves):
    import itertools
    # get_moves → possible_moves
    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], depth-1)
                if next_route:
                    return next_route

    for depth in itertools.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')
    pos_moves = []
    for i in d:
        pos_moves.append(generate(state, i, b))
    return pos_moves
```

```
def generate(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 temp

initial = [1,2,3,0,4,6,7,5,8]
goal = [1,2,3,4,5,6,7,8,0]
route = id_dfs(initial, goal, possible_moves)

if route:
    print("Success!!")
    print("Path:", route)
else:
    print("Failed to find a solution").
```

ID_DFS - Combination of BFS and DFS
- DFS in BFS manner