In [4]:
```
#KETHARNATH R 111723102089 CSE C
```

In [5]:
```
class Solution:
    def solve(self, board):
        state_dict = {}
        flatten = []

        for i in range(len(board)):
            flatten += board[i]

        flatten = tuple(flatten)
        state_dict[flatten] = 0

        if flatten == (0, 1, 2, 3, 4, 5, 6, 7, 8):
            return 0

        return self.get_paths(state_dict)

    def get_paths(self, state_dict):
        cnt = 0

        while True:
            current_nodes = [x for x in state_dict if state_dict[x] == cnt]
            if len(current_nodes) == 0:
                return -1

            for node in current_nodes:
                next_moves = self.find_next(node)
                for move in next_moves:
                    if move not in state_dict:
                        state_dict[move] = cnt + 1
                        if move == (0, 1, 2, 3, 4, 5, 6, 7, 8):
                            return cnt + 1

            cnt += 1

    def find_next(self, node):
        moves = {
            0: [1, 3],
            1: [0, 2, 4],
            2: [1, 5],
            3: [0, 4, 6],
            4: [1, 3, 5, 7],
            5: [2, 4, 8],
            6: [3, 7],
            7: [4, 6, 8],
            8: [5, 7],
        }

        results = []
        pos_0 = node.index(0)

        for move in moves[pos_0]:
            new_node = list(node)
            new_node[move], new_node[pos_0] = new_node[pos_0], new_node[move]
            results.append(tuple(new_node))

        return results
```

```
# Example usage
ob = Solution()
matrix = [
    [3, 1, 2],
    [4, 7, 5],
    [6, 8, 0]
]
print(ob.solve(matrix))
```

```
4
```

In [ ]: