1BM18CS062
Nilanshu Ranjan

AI Lab

A* algorithm to solve 8 puzzle :

heuristic : Manhattan distance

```
def manhattan(state, target) -> to calculate manhattan
                                  distance

def possible_moves(state, visited_states) :-> to get the
                                  possible moves from
                                  current state

def gen(state, direction, b) -> to generate action from
                                  possible state (direction is
                                  direction to which we chose to
                                  move, b is index of empty cell).

def print_grid(state) -> to print current state

def a_star(source, target):
    states = [source]
    g=0  # here we use it as depth
    visited_states = set()
    while(len(states) && g<=3):
        moves=[]
        for state in states:
            visited_states.add(tuple(state))
        print_grid(state)
        if state == target:
            print("Success")
            return
        moves += [move for move in possible_moves(state,
            visited_states) if move not in moves]
```

```
f=[g + manhattan(move, target) for move in moves
    # f(n) = g(n) + h(n)
 states=[moves[i] for i in range(len(moves)) if
             f[i] = min(f)] # taking minimum co
   g+=1  # increasing depth
print("fail")
```