

## 8-PUZZLE-A\*

```
import queue as Q

goal = [[1, 2, 3], [4, 5, 6], [7, 8, 0]]

def isGoal(state):
    return state == goal

def HeuristicValue(state):
    cnt = 0
    for i in range(len(goal)):
        for j in range(len(goal[i])):
            if goal[i][j] != state[i][j]:
                cnt += 1
    return cnt

def getCoordinates(currentState):
    for i in range(len(goal)):
        for j in range(len(goal[i])):
            if currentState[i][j] == 0:
                return (i, j)

def isValid(i, j) -> bool:
    return 0 <= i < 3 and 0 <= j < 3

def A_Star(state, goal) -> int:
    visited = set()
    pq = Q.PriorityQueue()
    pq.put((HeuristicValue(state), 0, state))

    while not pq.empty():
        _, moves, currentState = pq.get()
        if currentState == goal:
            return moves

        if tuple(map(tuple, currentState)) in visited:
            continue

        visited.add(tuple(map(tuple, currentState)))
        coordinates = getCoordinates(currentState)
        i, j = coordinates[0], coordinates[1]
```

```

for dx, dy in [(0, 1), (0, -1), (1, 0), (-1, 0)]:
    new_i, new_j = i + dx, j + dy
    if isValid(new_i, new_j):
        new_state = [row[:] for row in currentState]
        new_state[i][j], new_state[new_i][new_j] = new_state[new_i][new_j], new_state[i][j]
        if tuple(map(tuple, new_state)) not in visited:
            pq.put((HeuristicValue(new_state)+moves, moves + 1, new_state))
            if new_state == goal: # Print only states leading to the goal
                print(new_state)
return -1

state = [[1, 2, 3], [4, 0, 5], [6, 7, 8]]
moves = A_Star(state, goal)
if moves == -1:
    print("NO way to reach the given state")
else:
    print("Reached in " + str(moves) + " moves")

```

#### OUTPUT:

```

[[1, 2, 3], [4, 5, 6], [7, 8, 0]]
Reached in 14 moves

```