BFS

import numpy as np

def bfs(src, target):

queue = [(src, None)] # State and last move

visited = set()

state\_count = 0 # Initialize state count

while queue:

state, last\_move = queue.pop(0)

state\_tuple = tuple(state) # Convert state to tuple for set operations

if state\_tuple not in visited:

visited.add(state\_tuple)

state\_count += 1 # Increment the state count

print\_board(state)

if last\_move:

print(f"Current move: {last\_move}\n")

if state == target:

print("Goal state achieved!")

break

for move, direction in possible\_moves(state):

if tuple(move) not in visited:

queue.append((move, direction))

print(f"Total unique states explored: {state\_count}")

def possible\_moves(state):

b = state.index(0)

directions = []

if b not in [0, 1, 2]: directions.append('u')

if b not in [6, 7, 8]: directions.append('d')

if b not in [0, 3, 6]: directions.append('l')

if b not in [2, 5, 8]: directions.append('r')

return [(gen(state, d, b), d) for d in directions]

def gen(state, direction, b):

temp = state.copy()

if direction == 'u': temp[b], temp[b - 3] = temp[b - 3], temp[b]

if direction == 'd': temp[b], temp[b + 3] = temp[b + 3], temp[b]

if direction == 'l': temp[b], temp[b - 1] = temp[b - 1], temp[b]

if direction == 'r': temp[b], temp[b + 1] = temp[b + 1], temp[b]

return temp

def print\_board(state):

board = np.array(state).reshape(3, 3)

print(board)

# Initial configuration and target configuration

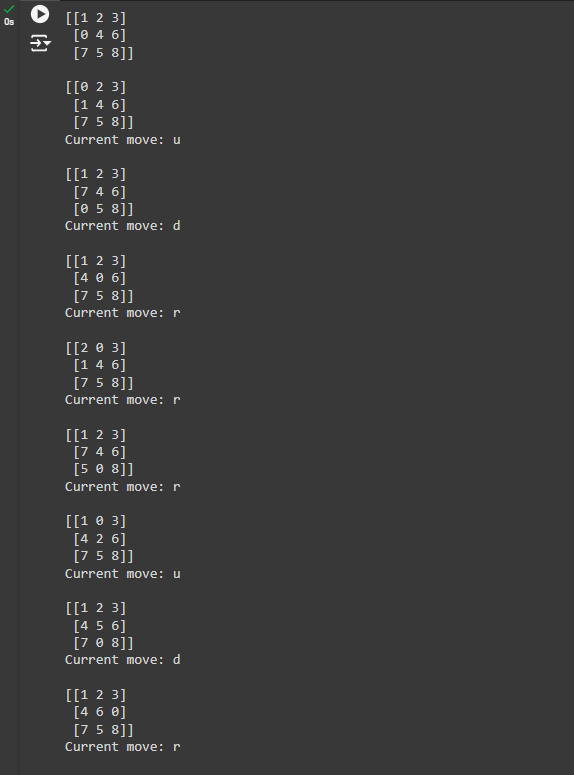
src = [1, 2, 3, 0, 4, 6, 7, 5, 8]

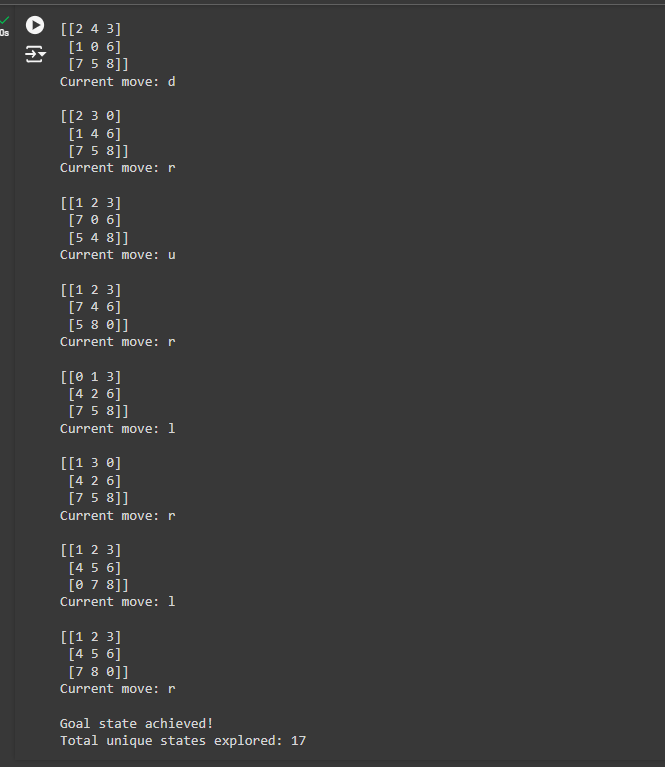
target = [1, 2, 3, 4, 5, 6, 7, 8, 0]

# Run BFS to solve the puzzle

bfs(src, target)

OUTPUT:





DFS