8-PUZZLE USING ITERATIVE DEEPENING SEARCH

def iterative\_deepening\_search(src, target):

depth\_limit = 0

while True:

result = depth\_limited\_search(src, target, depth\_limit, [])

if result is not None:

print("Success")

return

depth\_limit += 1

if depth\_limit > 30: # Set a reasonable depth limit to avoid an infinite loop

print("Solution not found within depth limit.")

return

def depth\_limited\_search(src, target, depth\_limit, visited\_states):

if src == target:

print\_state(src)

return src

if depth\_limit == 0:

return None

visited\_states.append(src)

poss\_moves\_to\_do = possible\_moves(src, visited\_states)

for move in poss\_moves\_to\_do:

if move not in visited\_states:

print\_state(move)

result = depth\_limited\_search(move, target, depth\_limit - 1, visited\_states)

if result is not None:

return result

return None

def possible\_moves(state, visited\_states):

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\_it\_can = []

for i in d:

pos\_moves\_it\_can.append(gen(state, i, b))

return [move\_it\_can for move\_it\_can in pos\_moves\_it\_can if move\_it\_can not in visited\_states]

def gen(state, m, b):

temp = state.copy()

if m == 'd':

temp[b + 3], temp[b] = temp[b], temp[b + 3]

elif m == 'u':

temp[b - 3], temp[b] = temp[b], temp[b - 3]

elif m == 'l':

temp[b - 1], temp[b] = temp[b], temp[b - 1]

elif m == 'r':

temp[b + 1], temp[b] = temp[b], temp[b + 1]

return temp

def print\_state(state):

print(f"{state[0]} {state[1]} {state[2]}\n{state[3]} {state[4]} {state[5]}\n{state[6]} {state[7]} {state[8]}\n")

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

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

iterative\_deepening\_search(src, target)

OUTPUT:



