import heapq

class PuzzleState:

def \_\_init\_\_(self, board, parent, move, depth, cost):

self.board = board

self.parent = parent

self.move = move

self.depth = depth

self.cost = cost

def \_\_lt\_\_(self, other):

return self.cost < other.cost

def print\_board(board):

print("+---+---+---+")

for i in range(0, 9, 3):

row = "|"

for tile in board[i:i+3]:

if tile == 0:

row += " |"

else:

row += f" {tile} |"

print(row)

print("+---+---+---+")

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

moves = {'U': -3, 'D': 3, 'L': -1, 'R': 1}

def manhattan\_distance(board):

distance = 0

for i in range(9):

if board[i] != 0:

current\_row, current\_col = divmod(i, 3)

goal\_row, goal\_col = divmod(board[i] - 1, 3)

distance += abs(current\_row - goal\_row) + abs(current\_col - goal\_col)

return distance

def make\_move(board, move, blank\_pos):

new\_board = board[:]

new\_blank\_pos = blank\_pos + moves[move]

new\_board[blank\_pos], new\_board[new\_blank\_pos] = new\_board[new\_blank\_pos], new\_board[blank\_pos]

return new\_board

def a\_star\_search(start\_board):

open\_list = []

closed\_set = set()

start\_state = PuzzleState(start\_board, None, None, 0, manhattan\_distance(start\_board))

heapq.heappush(open\_list, start\_state)

while open\_list:

current\_state = heapq.heappop(open\_list)

if current\_state.board == goal:

return current\_state

closed\_set.add(tuple(current\_state.board))

blank\_pos = current\_state.board.index(0)

for move in moves:

if move == 'U' and blank\_pos < 3:

continue

if move == 'D' and blank\_pos > 5:

continue

if move == 'L' and blank\_pos % 3 == 0:

continue

if move == 'R' and blank\_pos % 3 == 2:

continue

new\_board = make\_move(current\_state.board, move, blank\_pos)

if tuple(new\_board) in closed\_set:

continue

new\_depth = current\_state.depth + 1

new\_cost = new\_depth + manhattan\_distance(new\_board)

new\_state = PuzzleState(new\_board, current\_state, move, new\_depth, new\_cost)

heapq.heappush(open\_list, new\_state)

return None

# ---- Main Program ----

initial\_state = [

1, 2, 3,

4, 0, 5,

6, 7, 8

]

print("Initial State:")

print\_board(initial\_state)

print("Goal State:")

print\_board(goal)

result = a\_star\_search(initial\_state)

if result:

print(f" Solution found in {result.depth} steps.")

else:

print(" No solution found.")