A* Algorithm for 8-puzzle problem

```
function A_STAR_8_PUZZLE(start_state, goal_state):
    goal_flat = flatten(goal_state)
    open_list = priorityQueue()
    closed_list = Set()

    start_node = Node(start_state, g=0, h= MANHATTAN_DISTANCE
                                    (start_state, goal_flat))
    open_list.push(start_node)
    while open_list is not empty:
        current_node = open_list.pop()
        if current_node.state == goal_state
            return RECONSTRUCT_path(current_node)
        closed_list.add(current_node.state)
        for neighbor in GET_Neighbors(current_node)
            if neighbor.state in closed_list:
                continue
            neighbor.g = current_node.g + 1
            neighbor.h = MANHATTAN_DISTANCE
            neighbor.f = neighbor.g + neighbor.h

            if neighbour.state not in open_list or
                neighbor.f < open_list.get_f_value
    return "No solution found"
function Get_Neighbors(node):
    neighbors = []
    (x,y) = FIND_BLANK(node.state)
    directions = ["up", "down", "left", "right"]
    for each direction in directions:
        if valid move(direction, x, y):
            new_state = swap_tiles
            neighbors.add(Node(new_state, parent=node, move=
                                                direction))
    return neighbors

function Reconstruct_path(node):
    path = []
    while node.parent is not null:
        path.append(node.move)
        node = node.parent
    return reverse(path)

function MANHATTAN_DISTANCE(state, goal_flat):
    distance = 0
    for each tile in state:
        if tile != 0:
            goal_position = find(tile in goal_flat)
            current_position = find(tile in state)
            distance += abs(goal_position.x - current_position.x)
                        + abs(goal_position.y - current_position.y)
    return distance.
```