Ai Experiment-5

March 4, 2024

[ ]:
```
from queue import PriorityQueue

'''
i = Tile Number
b = Current Position
g = Target Position
'''
def h(puzzle, target):
    return sum(abs(b%3 - g%3) + abs(b//3 - g//3) for b, g in ((puzzle.index(i),␣
 ↪target.index(i)) for i in range(1, 9)))

def solve(puzzle, target):
    queue = PriorityQueue()
    queue.put((0, puzzle))
    came_from = {tuple(puzzle): None}
    cost_so_far = {tuple(puzzle): 0}
    while not queue.empty():
        _, current = queue.get()
        if current == target:
            path = []
            while current:
                path.append(current)
                current = came_from[tuple(current)]
            path.reverse()
            return path
        for new in neighbors(current):
            new_cost = cost_so_far[tuple(current)] + 1
            if tuple(new) not in cost_so_far or new_cost <␣
 ↪cost_so_far[tuple(new)]:
                cost_so_far[tuple(new)] = new_cost
                priority = new_cost + h(new, target)
                queue.put((priority, new))
                came_from[tuple(new)] = current

def neighbors(current):
    neighbors = []
    i = current.index(0)
```

```
    if i in [3, 4, 5, 6, 7, 8]:
        neighbors.append(swap(list(current), i, i - 3))
    if i in [1, 2, 4, 5, 7, 8]:
        neighbors.append(swap(list(current), i, i - 1))
    if i in [0, 1, 3, 4, 6, 7]:
        neighbors.append(swap(list(current), i, i + 1))
    if i in [0, 1, 2, 3, 4, 5]:
        neighbors.append(swap(list(current), i, i + 3))
    return neighbors

def swap(puzzle, i, j):
    puzzle[i], puzzle[j] = puzzle[j], puzzle[i]
    return puzzle

puzzle = [1, 2, 5, 3, 4, 0, 6, 7, 8]
target = [0, 1, 2, 3, 4, 5, 6, 7, 8]
path = solve(puzzle, target)
for i in path:
    print(i)
```

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