

main.py | sort heapq

```
2 goal_state = (1, 2, 3,
3             4, 5, 6,
4             7, 8, 0)
5 rows = 3
6 cols = 3
7 def heuristic(state):
8     distance = 0
9     for i, tile in enumerate(state):
10         if tile == 0:
11             continue
12         goal_x, goal_y = (tile - 1) // cols, (tile - 1) % cols
13         cur_x, cur_y = i // cols, i % cols
14         distance += abs(goal_x - cur_x) + abs(goal_y - cur_y)
15     return distance
16 def get_neighbors(state):
17     neighbors = []
```

Solution requires 2 moves:

(1, 2, 3)
(4, 0, 6)
(7, 5, 8)

(1, 2, 3)
(4, 5, 6)
(7, 0, 8)

(1, 2, 3)
(4, 5, 6)
(7, 8, 0)

=== Code Execution Successful ===

```
18 zero_pos = state.index(0)
19 x, y = zero_pos // cols, zero_pos % cols
20 moves = {
21     "UP": (x - 1, y),
22     "DOWN": (x + 1, y),
23     "LEFT": (x, y - 1),
24     "RIGHT": (x, y + 1)
25 }
26 for action, (nx, ny) in moves.items():
27     if 0 <= nx < rows and 0 <= ny < cols:
28         new_pos = nx * cols + ny
29         new_state = list(state)
30         new_state[zero_pos], new_state[new_pos] =
            new_state[new_pos], new_state[zero_pos]
31         neighbors.append(tuple(new_state))
32 return neighbors
33 def a_star(start_state):
```

main.py

```
open_list = []
35 heapq.heappush(open_list, (0 + heuristic(start_state), 0,
    start_state, None))
36 visited = {}
37 came_from = {}
38 while open_list:
39     f, g, state, parent = heapq.heappop(open_list)
40     if state in visited:
41         continue
42     visited[state] = g
43     came_from[state] = parent
44     if state == goal_state:
45         break
46     for neighbor in get_neighbors(state):
47         if neighbor not in visited:
48             new_g = g + 1
49             new_f = new_g + heuristic(neighbor)
```

```
51     path = []
52     current = goal_state
53     while current:
54         path.append(current)
55         current = came_from[current]
56     path.reverse()
57     return path
58 start_state = (1, 2, 3,
59               4, 0, 6,
60               7, 5, 8)
61 solution_path = a_star(start_state)
62 print("Solution requires", len(solution_path)-1, "moves:\n")
63 for step in solution_path:
64     print(step[0:3])
65     print(step[3:6])
66     print(step[6:9])
67     print()
```