A*

In [6]:

```
class Node:
    def __init__(self, name, g=0, h=0, parent=None):
        self.name = name
        self.g = g
        self.h = h
        self.f = g + h
        self.parent = parent


def get_lowest_f(open_list):
    lowest = open_list[0]
    for node in open_list:
        if node.f < lowest.f:
            lowest = node
    return lowest


def a_star(graph, heuristics, start, goal):
    open_list = []
    closed_list = []

    start_node = Node(start, g=0, h=heuristics[start])
    open_list.append(start_node)

    while open_list:
        current = get_lowest_f(open_list)

        if current.name == goal:
            path = []
            while current:
                path.append(current.name)
                current = current.parent
            return path[::-1]

        open_list.remove(current)
        closed_list.append(current.name)


        for neighbor, cost in graph[current.name].items():
            if neighbor in closed_list:
                continue

            g = current.g + cost
            h = heuristics[neighbor]
            neighbor_node = Node(neighbor, g, h, current)

            skip = False
            for open_node in open_list:
                if open_node.name == neighbor and open_node.f <= neighbor_node.f:
                    skip = True
```

```
                    break
            if skip:
                continue

            open_list.append(neighbor_node)

    return None


if __name__ == "__main__":
    graph = {
        'A': {'B': 1, 'C': 3},
        'B': {'A': 1, 'D': 3, 'E': 1},
        'C': {'A': 3, 'F': 5},
        'D': {'B': 3, 'E': 2, 'G': 2},
        'E': {'B': 1, 'D': 2, 'G': 3},
        'F': {'C': 5, 'G': 2},
        'G': {'D': 2, 'E': 3, 'F': 2}
    }

    heuristics = {
        'A': 7,
        'B': 6,
        'C': 5,
        'D': 3,
        'E': 2,
        'F': 4,
        'G': 0
    }

    start, goal = 'A', 'G'
    path = a_star(graph, heuristics, start, goal)
    print("Shortest Path:", path)
```

```
Shortest Path: ['A', 'B', 'E', 'G']
```

In [ ]: