```
def a_star(graph, start, goal, heuristic):
    open_set = [start]  # Only keep nodes
    g_score = {node: float('inf') for node in graph}
    g_score[start] = 0
    came_from = {}

    while open_set:
        # Find node in open_set with the lowest f = g + h
        current = min(open_set, key=lambda node: g_score[node] + heuristic[node])

        if current == goal:
            path = []
            while current in came_from:
                path.append(current)
                current = came_from[current]
            path.append(start)
            return path[::-1]

        open_set.remove(current)

        for neighbor, cost in graph[current].items():
            tentative_g = g_score[current] + cost
            if tentative_g < g_score[neighbor]:
                came_from[neighbor] = current
                g_score[neighbor] = tentative_g
                if neighbor not in open_set:
                    open_set.append(neighbor)

    return None

# Define graph
graph = {
    'A': {'B': 1, 'C': 3},
    'B': {'D': 1, 'E': 4},
    'C': {'F': 2},
    'D': {'G': 2},
    'E': {'G': 1},
    'F': {'G': 5},
    'G': {}
}

# Define heuristic values
heuristic = {
    'A': 6,
    'B': 4,
    'C': 4,
    'D': 2,
    'E': 1,
    'F': 3,
    'G': 0
}

# Run A* Search
start = 'A'
goal = 'G'
path = a_star(graph, start, goal, heuristic)

# Print the result
if path:
    print(f"Shortest path from {start} to {goal}: {path}")
else:
    print("No path found.")
```