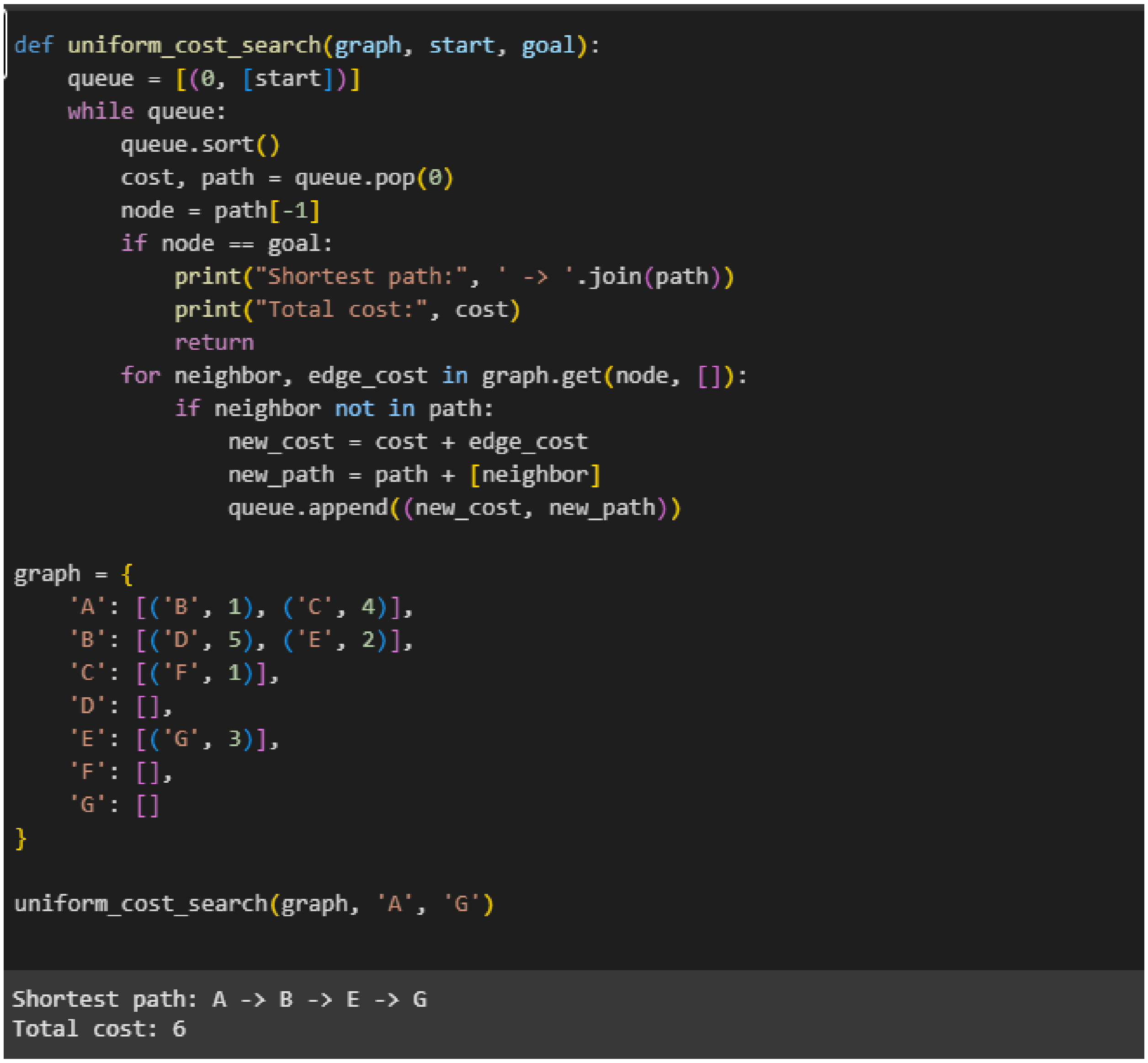
1. Ans:



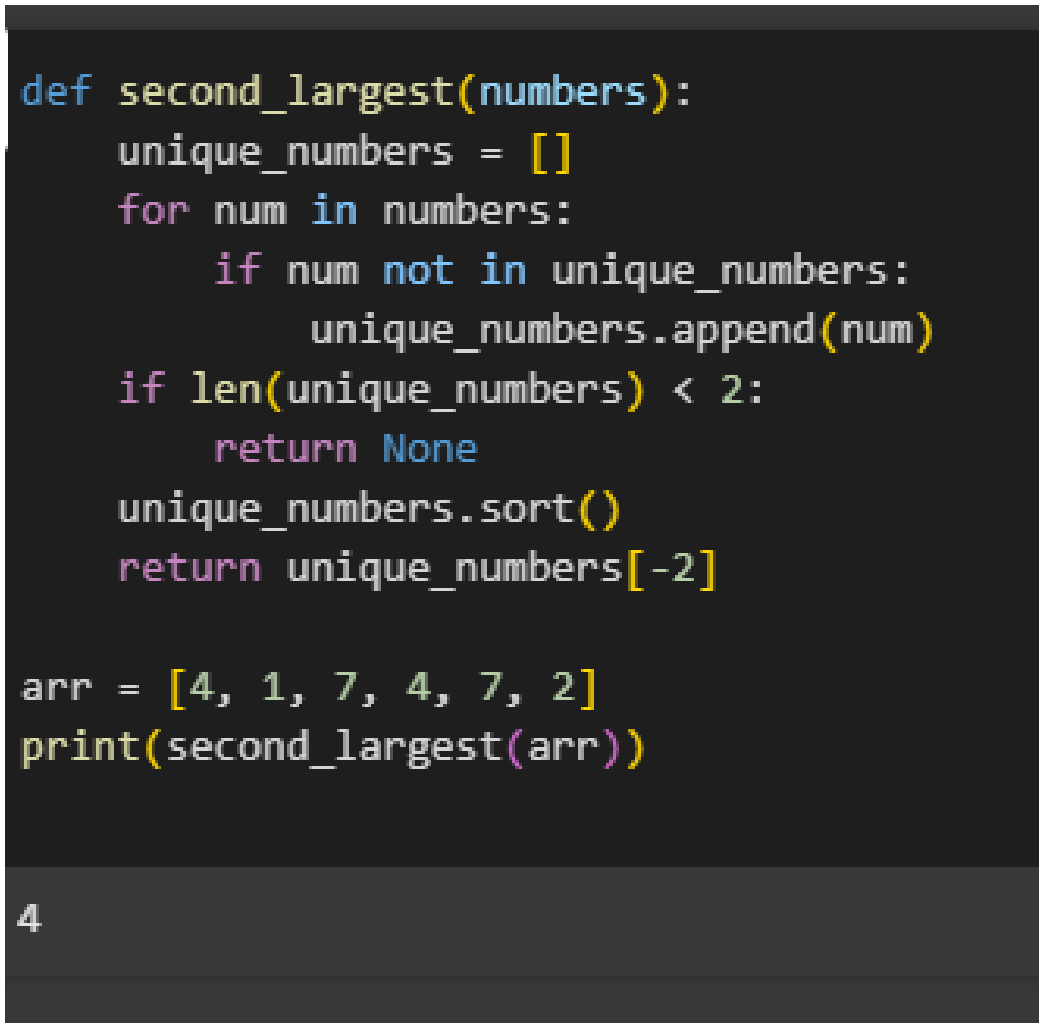
```python
def uniform_cost_search(graph, start, goal):
    queue = [(0, [start])]
    while queue:
        queue.sort()
        cost, path = queue.pop(0)
        node = path[-1]
        if node == goal:
            print("Shortest path:", ' -> '.join(path))
            print("Total cost:", cost)
            return
        for neighbor, edge_cost in graph.get(node, []):
            if neighbor not in path:
                new_cost = cost + edge_cost
                new_path = path + [neighbor]
                queue.append((new_cost, new_path))

graph = {
    'A': [('B', 1), ('C', 4)],
    'B': [('D', 5), ('E', 2)],
    'C': [('F', 1)],
    'D': [],
    'E': [('G', 3)],
    'F': [],
    'G': []
}

uniform_cost_search(graph, 'A', 'G')
```

```
Shortest path: A -> B -> E -> G
Total cost: 6
```

2. Ans:



```python
def second_largest(numbers):
    unique_numbers = []
    for num in numbers:
        if num not in unique_numbers:
            unique_numbers.append(num)
    if len(unique_numbers) < 2:
        return None
    unique_numbers.sort()
    return unique_numbers[-2]

arr = [4, 1, 7, 4, 7, 2]
print(second_largest(arr))
```

```
4
```