PROGRAM [5]:

```
graph = {
   'A': {'B': 10, 'C': 20},
   'B': {'A': 10, 'D': 5, 'E': 15},
   'C': {'A': 20, 'F': 30},
   'D': {'B': 5},
   'E': {'B': 15, 'F': 5},
   'F': {'C': 30, 'E': 5}
}

# Define the heuristic function for A* algorithm
def heuristic(a, b):
   return abs(ord(a) - ord(b))

# Define the BFS function
def bfs(graph, start, end):
   queue = [(start, [start], 0)]
   while queue:
      node, path, cost = queue.pop(0)
      for next_node in graph[node]:
         if next_node == end:
            return path + [next_node], cost + graph[node][next_node]
         else:
            queue.append((next_node, path + [next_node], cost + graph[node][next_node]))

# Define the A* function
```

OUTPUT [5:

```
BFS:  (['A', 'C', 'F'], 50)
A*:  (['A', 'B', 'E', 'F'], 30)
```

```
def a_star(graph, start, end):
    queue = [(0, start, [start], 0)]
    visited = set()
    while queue:
        f_cost, node, path, cost = queue.pop(0)
        if node in visited:
            continue
        visited.add(node)
        if node == end:
            return path, cost
        for next_node in graph[node]:
            g_cost = cost + graph[node][next_node]
            h_cost = heuristic(next_node, end)
            queue.append((g_cost + h_cost, next_node, path + [next_node], g_cost))
        queue.sort(key=lambda x: x[0])

# Test the algorithms
print("BFS: ", bfs(graph, 'A', 'F'))
print("A*: ", a_star(graph, 'A', 'F'))
```