4. Single Source Shortest Paths: Dijkstra's Algorithm

Code:

```
import heapq
def dijkstra(graph, start):
    pq = [(0, start)]
    distances = {vertex: float('inf') for vertex in graph}
    distances[start] = 0
    visited = set()
    while pq:
        (current_distance, current_vertex) = heapq.heappop(pq)
        if current_vertex in visited:
            continue
        visited.add(current_vertex)
        for neighbor, weight in graph[current_vertex]:
            distance = current_distance + weight
            if distance < distances[neighbor]:
                distances[neighbor] = distance
                heapq.heappush(pq, (distance, neighbor))
    return distances
graph = {
    'A': [('B', 1), ('C', 4)],
    'B': [('A', 1), ('C', 2), ('D', 5)],
    'C': [('A', 4), ('B', 2), ('D', 1)],
    'D': [('B', 5), ('C', 1)]}
start_vertex = 'A'
print(dijkstra(graph, start_vertex))
```

output:

Time complexity:

F(n)=o((v+e)logv)