91) Single Source Shortest Paths: Dijkstra's Algorithm

CODE:

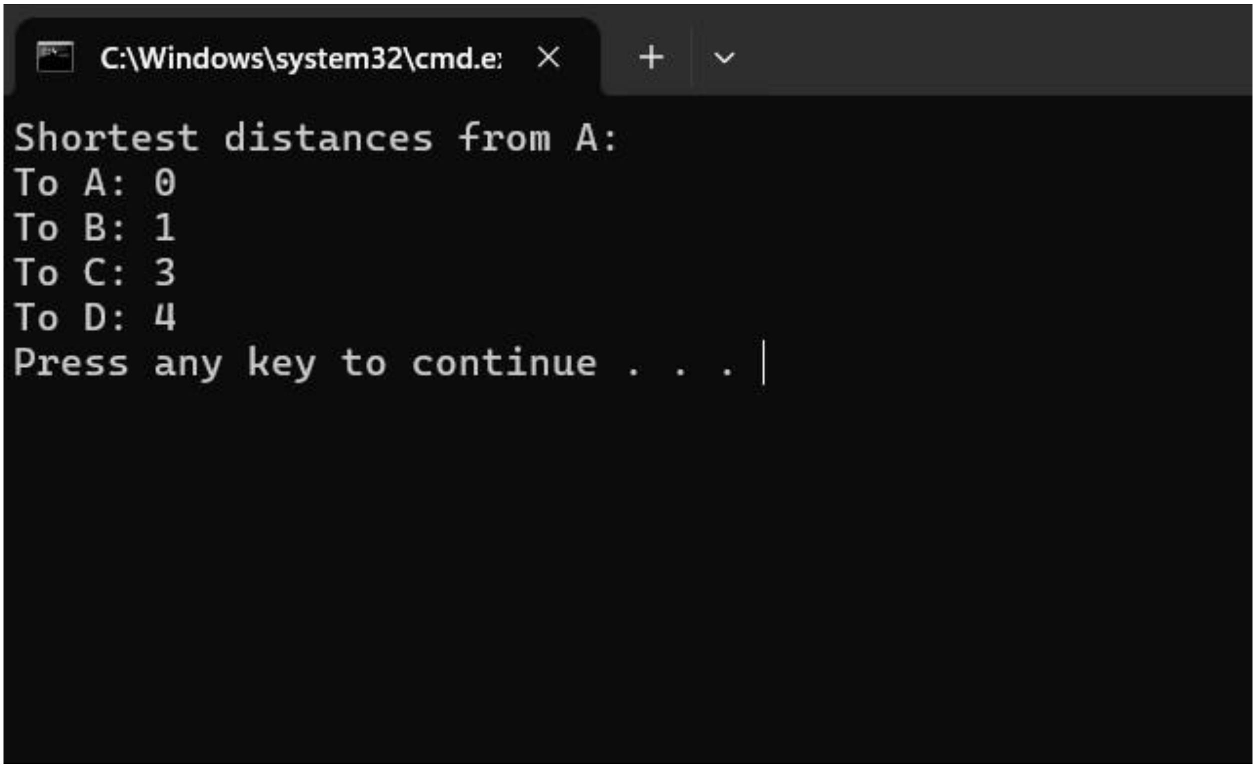
```
import heapq
 def dijkstra(graph, start):     distances = {vertex: float('infinity') for vertex in graph}
distances[start] = 0

    priority_queue = [(0, start)]
         while priority_queue:
        current_distance, current_vertex = heapq.heappop(priority_queue)
                  if current_distance > distances[current_vertex]:
              continue
                  for neighbor, weight in graph[current_vertex].items():
                   distance = current_distance + weight
                  if distance < distances[neighbor]:
distances[neighbor] = distance
                 heapq.heappush(priority_queue, (distance, neighbor))

    return distances
 if __name__ == "__main__":
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'
    shortest_distances = dijkstra(graph, start_vertex)

    print(f"Shortest distances from {start_vertex}:")
     for vertex, distance in shortest_distances.items():
         print(f"To {vertex}: {distance}")
```

OUTPUT:



```
Shortest distances from A:
To A: 0
To B: 1
To C: 3
To D: 4
Press any key to continue . . .
```

TIME COMPLEXITY : O((V+E)logV)