92. Single Source Shortest Paths: Dijkstra's Algorithm

Code:

```
import heapq

def dijkstra(graph, start):
    n = len(graph)
    distances = [float('inf')] * n
    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]:
            distance = current_distance + weight
            if distance < distances[neighbor]:
                distances[neighbor] = distance
                heapq.heappush(priority_queue, (distance, neighbor))

    return distances

graph = [
    [(1, 4), (2, 1)],
    [(3, 1)],
    [(1, 2), (3, 5)],
    []
]

start_vertex = 0
distances = dijkstra(graph, start_vertex)
print(f"Shortest distances from vertex {start_vertex}: {distances}")
```

Output:

```
Shortest distances from vertex 0: [0, 3, 1, 4]
```

Time Complexity:

- T(n)= O((E+V)logv)