**Dijkstra's single source shortest path algorithm:-**

**import heapq**

**# define the graph and its weights as a dictionary of dictionaries**

**graph = {}**

**# get the number of edges from the user**

**n = int(input("Enter the number of edges: "))**

**# get the edges and their weights from the user and build the graph**

**for i in range(n):**

**u, v, w = input("Enter an edge (u v w): ").split()**

**w = int(w)**

**if u not in graph:**

**graph[u] = {}**

**if v not in graph:**

**graph[v] = {}**

**graph[u][v] = w**

**graph[v][u] = w**

**# get the starting node from the user**

**start\_node = input("Enter the starting node: ")**

**# initialize the distances dictionary and the priority queue**

**distances = {node: float('inf') for node in graph}**

**distances[start\_node] = 0**

**heap = [(0, start\_node)]**

**# Dijkstra's algorithm**

**visited = set()**

**while heap:**

**(d, u) = heapq.heappop(heap)**

**if u in visited:**

**continue**

**visited.add(u)**

**for v, w in graph[u].items():**

**if v in visited:**

**continue**

**if d + w < distances[v]:**

**distances[v] = d + w**

**heapq.heappush(heap, (distances[v], v))**

**# print the shortest distances from the starting node to all other nodes**

**print("Shortest distances from", start\_node)**

**for node in graph:**

**print("Shortest dist from "+node+" is ",distances[node])**