```
import heapq

def dijkstra(n, edges, src):
    graph = {i: [] for i in range(1, n+1)}
    for u, v, w in edges:
        graph[u].append((v, w))
        graph[v].append((u, w))
    dist = {i: float('inf') for i in range(1, n+1)}
    dist[src] = 0
    visited = {i: False for i in range(1, n+1)}
    heap = [(0, src)]
    while heap:
        d, u = heapq.heappop(heap)
        if visited[u]:
            continue
        visited[u] = True
        for v, w in graph[u]:
            if not visited[v] and d + w < dist[v]:
                dist[v] = d + w
                heapq.heappush(heap, (dist[v], v))
    return dist

def main():
    n = int(input("Enter number of vertices: "))
    m = int(input("Enter number of edges: "))
    edges = []
    print("Enter each edge in format: u v weight")
    for _ in range(m):
        u, v, w = map(int, input().split())
        edges.append((u, v, w))
    src = int(input("Enter source vertex (1-based): "))
    dist = dijkstra(n, edges, src)
    print("Shortest distances from source:")
    for i in range(1, n+1):
        d = dist[i]
        if d == float('inf'):
            print(f"Vertex {i} -> unreachable")
        else:
            print(f"Vertex {i} -> {d}")

if __name__ == "__main__":
    main()
```