Dijkstra's algorithm is a **shortest path algorithm** used to find the shortest distance from a **single source vertex** to all other vertices in a weighted graph (with non-negative weights). It is widely used in network routing, GPS navigation, and graph-related problems.

**Algorithm Steps**

1. **Initialize**:
   * Set the source node’s distance to **0** and all other nodes’ distances to **infinity**.
   * Create a priority queue (or min-heap) to track nodes with the smallest known distance.
2. **Process Nodes**:
   * Extract the node with the **smallest** distance from the priority queue.
   * Update the distances of its adjacent nodes if a **shorter path** is found.
   * Repeat until all nodes have been processed.
3. **Terminate**:
   * When all reachable nodes have been visited, the shortest path from the source is determined.

**Example**

Consider a graph:

css

CopyEdit

(A)

/ \

4 1

/ \

(B) --2-- (C)

**Adjacency List Representation**

css

CopyEdit

A -> B (4), C (1)

B -> C (2)

**Find the shortest path from A:**

* Start at **A**:
  + Distance to B = 4, Distance to C = 1
* Process **C**:
  + Distance to B via C = 1 + 2 = 3 (update B)
* Process **B**:
  + No further updates

**Final Shortest Distances from A**:

rust

CopyEdit

A -> A = 0

A -> B = 3

A -> C = 1

**Python Code**

import heapq

def dijkstra(graph, start):

pq = [] # Min-heap (priority queue)

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

distances[start] = 0

heapq.heappush(pq, (0, start))

while pq:

current\_distance, current\_node = heapq.heappop(pq)

if current\_distance > distances[current\_node]:

continue

for neighbor, weight in graph[current\_node]:

distance = current\_distance + weight

if distance < distances[neighbor]:

distances[neighbor] = distance

heapq.heappush(pq, (distance, neighbor))

return distances

# Example Graph (Adjacency List)

graph = {

'A': [('B', 4), ('C', 1)],

'B': [('C', 2)],

'C': []

}

print(dijkstra(graph, 'A'))

**Time Complexity**

* **O((V + E) log V)** using a priority queue (Binary Heap).
* **O(V²)** using a simple array.

Let me know if you need further clarification! 🚀