DIJKSTRA

Part 1: What problem does Dijkstra solve?

Dijkstra helps you find:

1.The **shortest distance** from a starting node to all other nodes

2.Works only when weights are **positive**

3.Used in **Google Maps**, **network routing**, **games**, etc.

Example:
If cities are nodes and road distances are weights, Dijkstra tells you the **minimum travel distance** from your city to all other cities.

Part 2: Main Idea

Think of the algorithm like this:

1.You start at a node with distance **0**

2.All other distances are **∞ (infinity)**

3.You always go to the **closest unvisited node**

4.Update (relax) the distances of its neighbors

5.Repeat until all nodes are visited

6.It’s like expanding outward from the source step-by-step, always choosing the cheapest next path.

Part 3: Key Concepts

1. Distance Array

Stores the shortest distance found so far:

dist[node] = minimum cost from source to node

2. Visited Set

Marks nodes whose shortest path is already fixed.

3. Priority Queue (Min-Heap)

Always picks the node with the **smallest distance** next.

Part 4: Step-by-Step Example

Suppose edges:

A --5-- BA --2-- C
C --1-- DB --3-- D

Start from A.

Steps:

1.Initially

dist[A] = 0

dist[B] = ∞

dist[C] = ∞

dist[D] = ∞

2.Pick A (distance 0)
Update:

B = 5

C = 2

3.Pick C (distance 2)
Update:

D = 2 + 1 = 3

4.Pick D (distance 3)
Update:

B = min(5, 3+3=6) → still

5.Pick B (distance 5)
Done.

Shortest distances:
A=0, C=2, D=3, B=5.

Part 5: Why is this algorithm correct?

Because **when you pick the smallest unvisited distance**, it is always the final shortest path (since weights are positive).

Part 6: Dijkstra Pseudocode

```
Dijkstra(Graph, source):

   For each node:

      dist[node] = infinity

   dist[source] = 0

   Create a min-priority-queue pq

   pq.push( (0, source) )

   While pq is not empty:

      (d, u) = pq.pop()
```

```
    If u is already visited:
        continue
    Mark u as visited

    For each neighbor v of u with edge weight w:
        If dist[u] + w < dist[v]:
            dist[v] = dist[u] + w
            pq.push( (dist[v], v) )
```

Part 7: What Dijkstra gives you

1.Shortest distance from source to all nodes

2.You can reconstruct shortest paths too

3.Works fast using priority queue

8.Dijkstra Diagram

```
      (5)
 A -------- B
  |         |
 (2)       (3)
  |         |
 C -------- D
     (1)
```

Edges with weights:

A → B = 5

A → C = 2

C → D = 1

B → D = 3

How Dijkstra explores (step-by-step visually)

Start at A (distance = 0)

A(0)   B(∞)   C(∞)   D(∞)

Step 1: From A, update neighbors

A(0) → B = 5
A(0) → C = 2
A(0)   B(5)   C(2)   D(∞)

Step 2: Pick the smallest distance → C(2)

Update neighbors of C:

C(2) → D = 2 + 1 = 3
A(0)   B(5)   C(2)   D(3)

Step 3: Pick next smallest → D(3)

Update neighbors:

D → B = 3 + 3 = 6 But B already has 5 → keep 5

No change:

A(0)   B(5)   C(2)   D(3)

Step 4: Pick next smallest → B(5)

No more updates.

⭑ Final Shortest Distances

A = 0 C = 2 D = 3 B = 5