DIJKSTRA

What is Dijkstra ?

Dijkstra’s Algorithm is a shortest-path algorithm.
It finds the minimum cost (shortest distance) from a starting node to all other nodes in a weighted graph (where edges have costs).

Main Ideas of Dijkstra ?

1. Find the shortest path in a weighted graph
2. Always choose the smallest distance node next
3. Start with distance = 0, others = infinity
4. Relaxation updates distances
5. Once a node is finalized, its distance is the shortest
6. Works only with positive weights

Where Dijkstra is used

1. Networking
2. Maps and Navigation
3. Robotics and AI
4. Operations Research

Examples

GPS Navigation / Maps

Computer Networks / Internet Routing

Game Development

Robotics / Path Planning

Social Networks

Graph

```
        (A)
        / \
      2/   \4
      /     \
   (B)---1---(C)
```

Source: `A`
Edges: `A-B=2`, `A-C=4`, `B-C=1`

How dijkstra works

1. Mark the distance to the source node as `0`.
2. Mark the distance to all other nodes as infinity (∞).
3. Create a visited/unvisited set (all nodes start unvisited).
4. Pick the Minimum Distance Node
5. Update Neighbor Distances
6. Mark Node as Visited
7. Repeat
8. Result

Pseudocode :

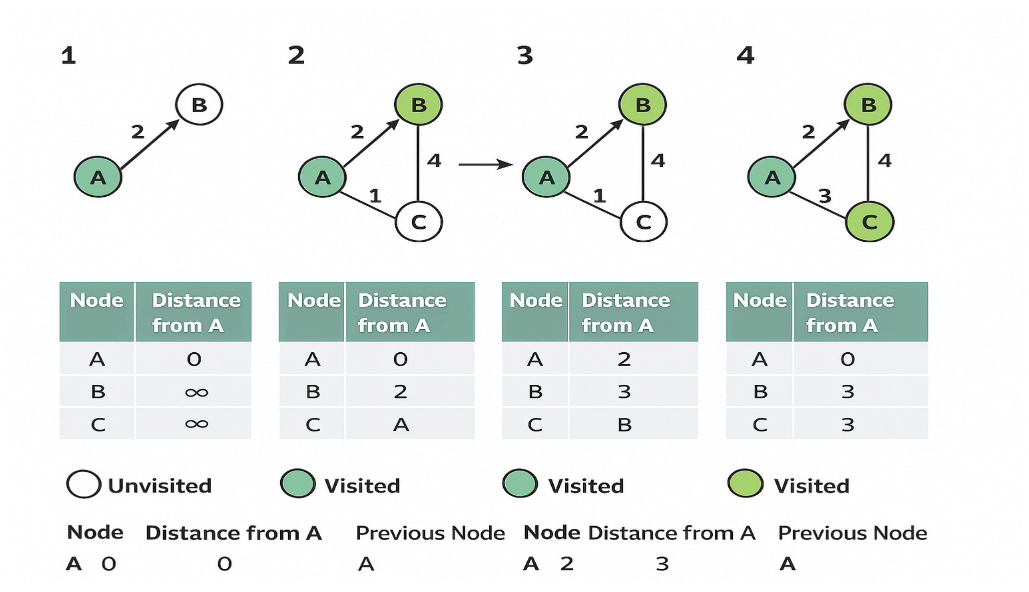
```
            Dijkstra(Graph, source):
for each vertex v in Graph:
    dist[v] := infinity
    previous[v] := undefined
dist[source] := 0
Q := all vertices in Graph

while Q is not empty:
    u := vertex in Q with smallest dist[u]
    remove u from Q

    for each neighbor v of u:
        alt := dist[u] + weight(u, v)
        if alt < dist[v]:
            dist[v] := alt
            previous[v] := u

return dist[], previous[]
```

Dijkstra:



Node	Distance from A
A	0
B	∞
C	∞

Node	Distance from A
A	0
B	2
C	A

Node	Distance from A
A	2
B	3
C	B

Node	Distance from A
A	0
B	3
C	3

○ Unvisited ● Visited ● Visited ● Visited

Node	Distance from A	Previous Node	Node	Distance from A	Previous Node
A 0	0	A	A 2	3	A

After learning Dijkstra :

Bellman-Ford – handles negative weights.

Floyd-Warshall – all-pairs shortest paths.

A* – heuristic-based pathfinding.

Minimum Spanning Tree – Prim’s and Kruskal’s algorithms.

Advanced graph problems – DAG shortest paths, Johnson’s algorithm, and real-world graph applications.