Applying Dijkstra's Algorithm

1. Introduction Dijkstra's Algorithm is applied to find the shortest path from a source node to all other nodes in a weighted graph. This document presents a step-by-step application of Dijkstra's Algorithm starting from **Node A**.

2. Given Graph The graph consists of the following edges and weights:

- (A, B) -> 4
- (A, C) -> 2
- (B, C) -> 5
- (B, D) -> 10
- (C, E) -> 3
- (E, D) -> 4
- (D, F) -> 11

3. Initialization A distance table is maintained where:

- **d(A) = 0** (starting node)
- All other nodes are set to ∞ (unreachable initially)
- A **visited set** is used to keep track of processed nodes

Node	**A**	**B**	**C**	**D**	**E**	**F**
d[]	0	∞	∞	∞	∞	∞

4. Step-by-Step Execution

Step 1: Select the smallest distance from non-visited vertices → **A** (d = 0)

- Possible vertices A can visit: **B, C**
- Compute new distances:
  - d(B) = min(d(B), d(A) + cost(A, B))
    - d(B) = min(∞, 0 + 4) = 4
  - d(C) = min(d(C), d(A) + cost(A, C))
    - d(C) = min(∞, 0 + 2) = 2

Node	A	B	C	D	E	F
d[]	0	4	2	∞	∞	∞

Step 2: Select the smallest distance from non-visited vertices → **C** (d = 2)

- Possible vertices C can visit: **B, E**
- Compute new distances:
  - d(B) = min(d(B), d(C) + cost(C, B))
    - d(B) = min(4, 2 + 5) = 4
  - d(E) = min(d(E), d(C) + cost(C, E))
    - d(E) = min(∞, 2 + 3) = 5

Node	A	B	C	D	E	F
d[]	0	4	2	∞	5	∞

Step 3: Select the smallest distance from non-visited vertices → **B** (d = 4)

- Possible vertices B can visit: **D**
- Compute new distances:
  - d(D) = min(d(D), d(B) + cost(B, D))
    - d(D) = min(∞, 4 + 10) = 14

Node	A	B	C	D	E	F
d[]	0	4	2	14	5	∞

Step 4: Select the smallest distance from non-visited vertices → **E** (d = 5)

- Possible vertices E can visit: **D**
- Compute new distances:
  - d(D) = min(d(D), d(E) + cost(E, D))
    - d(D) = min(14, 5 + 4) = 9

Node	A	B	C	D	E	F
d[]	0	4	2	9	5	∞

Step 5: Select the smallest distance from non-visited vertices → **D** (d = 9)

- Possible vertices D can visit: **F**
- Compute new distances:
  - d(F) = min(d(F), d(D) + cost(D, F))
    - d(F) = min(∞, 9 + 11) = 20

Node	A	B	C	D	E	F
d[]	0	4	2	9	5	20

Step 6: Select the smallest distance from non-visited vertices → **F** (d = 20)

- No updates, as **F** has no unvisited neighbors.

5. Final Shortest Distances from Node A

- A → A = 0
- A → B = 4
- A → C = 2
- A → D = 9
- A → E = 5
- A → F = 20

6. Conclusion Dijkstra's Algorithm successfully computes the shortest path from **Node A** to all other nodes. The computed shortest distances confirm the correctness of the algorithm's step-by-step execution.