Dijkstra's Algorithm

1. Introduction

Dijkstra's Algorithm is used to find the shortest path from a source node to all other nodes in a weighted graph. This document illustrates the step-by-step execution 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

- **Distance Table:**
  A distance table is maintained to keep track of the shortest distances from the source node.
  - `d(A) = 0` (starting node)
  - All other nodes are set to infinity (∞) initially, as they are unreachable at the start.

Initial Distance Table

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

Visited Set: {}
Unvisited Set: {A, B, C, D, E, F}

---

4. Step-by-Step Execution

Step 1: Select the minimum cost vertex from non-visited vertices → A (d = 0)

- Possible vertices A can visit: B, C

Update distances:

- `d[B] = min{d[B], d[A] + cost(A, B)} = min{∞, 0 + 4} = 4`
- `d[C] = min{d[C], d[A] + cost(A, C)} = min{∞, 0 + 2} = 2`

Node	A	B	C	D	E	F
d[]	0	4	2	∞	∞	∞

Visited Set: {A}
Unvisited Set: {B, C, D, E, F}

Step 2: Select the minimum cost vertex from non-visited vertices → C (d = 2)

- Possible vertex C can visit: E

Update distances:

- `d[E] = min{d[E], d[C] + cost(C, E)} = min{∞, 2 + 3} = 5`

Node	A	B	C	D	E	F
d[]	0	4	2	∞	5	∞

Visited Set: {A, C}
Unvisited Set: {B, D, E, F}

Step 3: Select the minimum cost vertex from non-visited vertices → B (d = 4)

- Possible vertices B can visit: C, D

Update distances:

- `d[D] = min{d[D], d[B] + cost(B, D)} = min{∞, 4 + 10} = 14`

Node	A	B	C	D	E	F
d[]	0	4	2	14	5	∞

Visited Set: {A, C, B}
Unvisited Set: {D, E, F}

Step 4: Select the minimum cost vertex from non-visited vertices → E (d = 5)

- Possible vertex E can visit: D

Update distances:

- `d[D] = min{d[D], d[E] + cost(E, D)} = min{14, 5 + 4} = 9`

Node	A	B	C	D	E	F
d[]	0	4	2	9	5	∞

Visited Set: {A, C, B, E}
Unvisited Set: {D, F}

Step 5: Select the minimum cost vertex from non-visited vertices → D (d = 9)

- Possible vertex D can visit: F

Update distances:

- `d[F] = min{d[F], d[D] + cost(D, F)} = min{∞, 9 + 11} = 20`

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

Visited Set: {A, C, B, E, D}
Unvisited Set: {F}

Step 6: Select the minimum cost vertex from non-visited vertices → F (d = 20)

- No updates, as F has no unvisited neighbors.

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

Visited Set: {A, C, B, E, D, F}
Unvisited Set: {}

---

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.