# Dijkstra's Algorithm: Single Source Shortest Path

Mayank Pratap Singh

## 1 Dijkstra's Algorithm

Dijkstra's Algorithm is used to find the shortest path from a single source vertex to all other vertices in a weighted, directed graph. The algorithm efficiently uses a priority queue (min-heap) to continuously choose the vertex with the smallest tentative distance.

### 1.1 Key Components

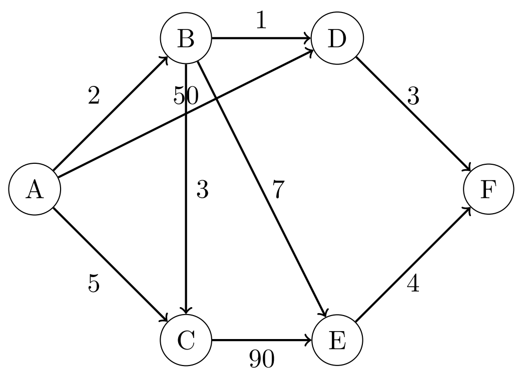
- **Single Source Shortest Path**: The algorithm starts from a source vertex and calculates the shortest path to all other vertices.
- **Directed, Weighted Graph**: The algorithm works on graphs where each edge has a direction and a weight.

### 1.2 Algorithm Steps

1. **Initialization**: Set the distance of the source vertex to 0 and all other vertices to infinity.
2. **Priority Queue (Min-Heap)**: Use a priority queue to select the vertex with the smallest distance that hasn't been processed.
3. **Update Distances**: For each neighbor of the selected vertex, update the distance if the path through this vertex is shorter.
4. **Repeat**: Continue the process until all vertices have been processed.

### 1.3 Example Graph

Consider the following directed, weighted graph:




In this graph: - The numbers on the edges represent the weights of the edges. - We will start from vertex **A** as the source.

### 1.4 Steps of Dijkstra's Algorithm

1. **Initialization**:

| Vertex | Distance | Parent |
|---|---|---|
| $A$ | 0 | $-$ |
| $B$ | $\infty$ | $-$ |
| $C$ | $\infty$ | $-$ |
| $D$ | $\infty$ | $-$ |
| $E$ | $\infty$ | $-$ |
| $F$ | $\infty$ | $-$ |

2. **First Iteration** (Start from A): - Pick vertex A (distance 0). - Update distances to its neighbors (B, C, D).

| Vertex | Distance | Parent |
|---|---|---|
| $A$ | 0 | $-$ |
| $B$ | 2 | $A$ |
| $C$ | 5 | $A$ |
| $D$ | 50 | $A$ |
| $E$ | $\infty$ | $-$ |
| $F$ | $\infty$ | $-$ |

3. **Second Iteration** (Pick vertex B): - Pick vertex B (distance 2). - Update distances to its neighbors (C, D, E).

| Vertex | Distance | Parent |
|---|---|---|
| $A$ | 0 | $-$ |
| $B$ | 2 | $A$ |
| $C$ | 5 | $A$ |
| $D$ | 3 | $B$ |
| $E$ | 9 | $B$ |
| $F$ | $\infty$ | $-$ |

4. **Third Iteration** (Pick vertex D): - Pick vertex D (distance 3). - Update distance to its neighbor (F).

| Vertex | Distance | Parent |
|---|---|---|
| $A$ | 0 | $-$ |
| $B$ | 2 | $A$ |
| $C$ | 5 | $A$ |
| $D$ | 3 | $B$ |
| $E$ | 9 | $B$ |
| $F$ | 6 | $D$ |

5. **Fourth Iteration** (Pick vertex C): - Pick vertex C (distance 5). No updates since it leads to vertex E with no shorter path.
6. **Fifth Iteration** (Pick vertex F): - Pick vertex F (distance 6). No updates.

## 1.5 Final Result

The shortest paths from A to all other vertices are:
- A $\rightarrow$ B: 2 - A $\rightarrow$ C: 5 - A $\rightarrow$ D: 3 - A $\rightarrow$ E: 9 - A $\rightarrow$ F: 6

## 1.6 Time Complexity

The time complexity of Dijkstra's Algorithm is:

$$O(V + E \log V)$$

Where: - $V$ is the number of vertices. - $E$ is the number of edges. - The logarithmic factor comes from the use of a priority queue (min-heap) to select the vertex with the smallest distance.