

# Design and Analysis of Algorithms

## L22: Dijkstra's Algorithm

Single Source Shortest Path

+

Bellman-Ford Algorithm

Dr. Ram P Rustagi  
Sem IV (2020-Even)  
Dept of CSE, KSIT  
[rprustagi@ksit.edu.in](mailto:rprustagi@ksit.edu.in)

# Resources

- Text book 1: Sec 9.1-5.4 - Levitin
- R1: Introduction to Algorithms
  - Cormen et al.

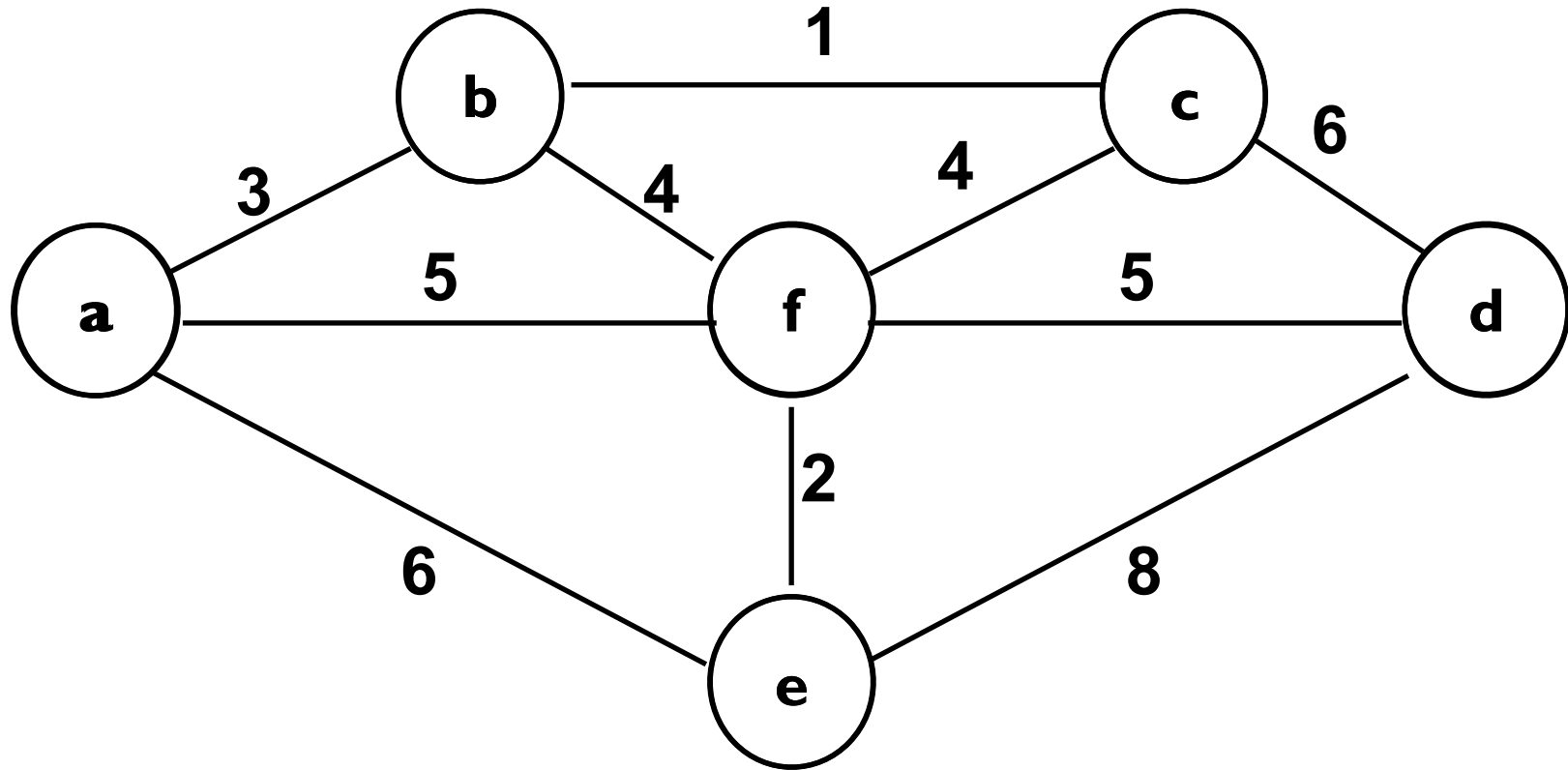
# Single Source Shortest Path

- Applications
  - Supplying deliveries from a factory to various godowns
    - Minimum time/cost
  - KSIT: Moving from quadrangle to your class rooms
    - Minimum time taken

# Single Source Shortest Path

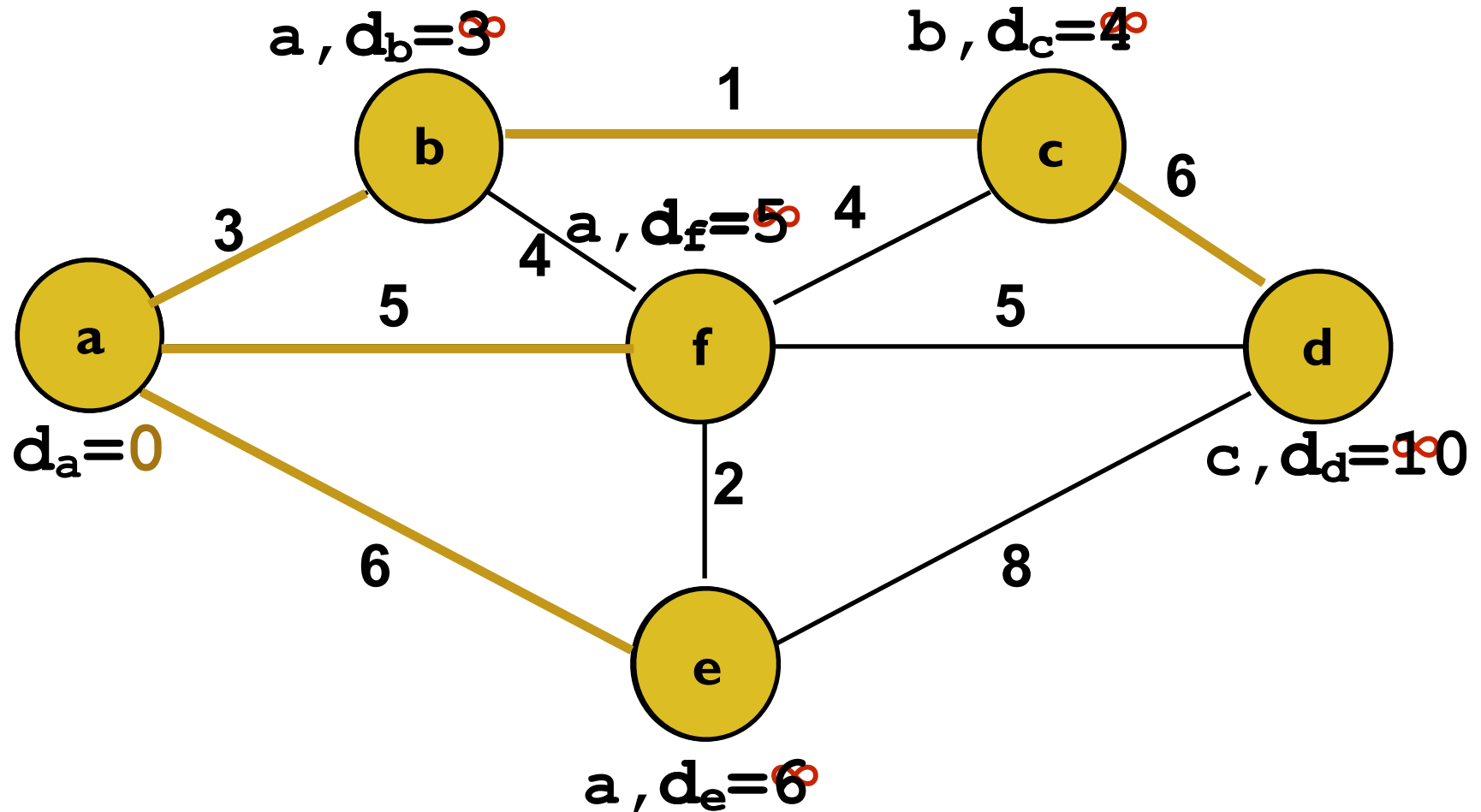
- Goal: Given a weighted connected (directed) graph  $G$ , find shortest paths from source vertex  $s$  to each of the other vertices
- Dijkstra's algorithm
  - Similar to Prim's algorithm for MST
  - Computes numerical labels differently
  - Among vertices not in the tree,
    - Find the vertex  $v$  with the smallest sum  $d_u + w(u, v)$ , where
      - $u \in V$  whose shortest path found in previous iteration
      - $d_u$  is the length of shortest path from  $s$  to  $u$
      - $w(u, v)$  is the weight of edge  $u \rightarrow v$

# Example: Dijkstra's Algorithm



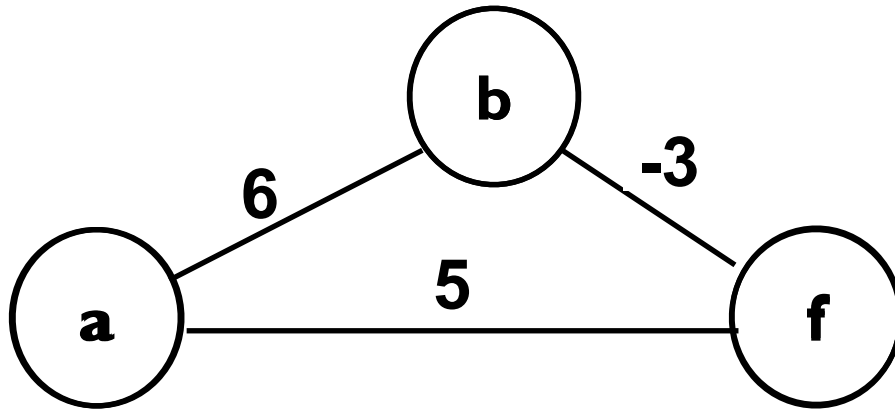
- Q: Construct an SSSP using Dijkstra's algo starting from vertex a

# Example: Dijkstra's Algorithm



# Notes on Dijkstra's Algorithm

- Proof of correctness:
  - Using induction
- Works with graph with +ve weights only
  - Build a counter example with -ve weight where Dijkstra's algorithm does not work
- Works for both directed and undirected graphs



# Algorithm: Dijkstra's Algorithm

```
Algo Dijkstra( $G, s$ )  
// i/p: a weighted connected graph  $G = (V, E)$ , and src  $s$   
//      all edges are non-negative weights  
// o/p: Length  $d_v$  of a shortest path from  $s$  to  $v$ .  
//      along with it predecessor vertex from  $v$  to  $s$ .  
Initialize( $Q$ ) // priority queue of vertices is empty initially  
for each vertex  $v \in V$ , do  
     $d_v \leftarrow \infty$ ;  $p_v \leftarrow \text{Null}$ ;  
    Insert( $Q, v, d_v$ ) // initialize vertex priority in priority  $Q$   
 $d_s \leftarrow 0$ ;  
Decrease( $Q, s, d_s$ )  
 $p_s \leftarrow \text{Null}$ ;  
 $V_T \leftarrow \emptyset$ 
```



# Algorithm: Dijkstra's Algorithm...

```
Algo Dijkstra( $G, s$ ) ...
```

```
  for  $i=0$  to  $|V|-1$  do
```

```
     $u = \text{DeleteMin}(Q)$  //time implementation based
```

```
     $V_T = V_T \cup \{u\}$ 
```

```
    for every vertex  $w \in V - V_T$  adjacent to  $u$ , do
```

```
      if  $d_u + \text{weight}(u, w) < d_w$ , then
```

```
         $d_w \leftarrow d_u + \text{weight}(u, w)$ 
```

```
         $p_w \leftarrow u$ 
```

```
        Decrease( $Q, w, d_w$ ) //time implementation based
```

```
      fi
```

```
    end //for  $w \in V - V_T$ 
```

```
  end //for  $i=0$ 
```

```
end //algo
```

# Analysis: Dijkstra's Algorithm...

- Implementation using Adjacency matrix
  - priority Q using unsorted array
  - Outer for loop ( $i=0$  to  $|V|-1$ ) :  $O(|V|)$  times
  - DeleteMin takes  $O(|V|)$  times
    - Total time for all vertices:  $O(|V|^2)$
  - Decrease( $Q, w, d_w$ ) takes  $O(1)$  time
    - Total time for all vertices:  $O(|E|)$
  - Time Complexity:  $O(|V|^2)$

# Analysis: Dijkstra's Algorithm

- Implementation using Adjacency List
  - priority Q using Heap
  - Outer for loop ( $i=0$  to  $|V|-1$ ):  $O(|V|)$  times
  - DeleteMin takes  $O(\lg |V|)$  times
    - Total time for all vertices:  $O(|V| \lg |V|)$
  - Decrease( $Q, w, d_w$ ) takes  $O(\lg |V|)$  time
    - Total time for all vertices:  $O(|E| \lg |V|)$
  - Time Complexity:  $O(|E| \lg |V|)$

# Questions

- Q1: what adjustments if any need to be made in Dijkstra's algorithm to solve the single-source shortest-paths problem for directed weighted graphs.
- Ans:
  - Do we need any changes? Just follow the directed edges.

# Questions

- Q2: Find a shortest path between two given vertices of a weighted graph or digraph. (This variation is called the single-pair shortest-path problem.)
- Ans:
  - Start from one vertex as source
  - Iterate the for loop till you find 2nd vertex.

# Questions

- Q3: Find the shortest paths to a given vertex from each other vertex of a weighted graph. (This variation is called the single destination shortest-paths problem.)
- Ans: Undirected graph
  - Start from the destination vertex as source
  - Find the shortest path from this to all other vertices
  - Reverse the path
- Ans: directed graph
  - Reverse the direction of all edges.
  - Start from the destination vertex as source
  - Find the shortest path from this src to all other vertices
  - Reverse the path

# Questions

- Q4: Solve the single-source shortest-path problem in a graph with non-negative numbers assigned to its vertices (and the length of a path defined as the sum of the vertex numbers on the path).
- Hint:
  - The weight of the edge is sum of non-negative numbers assigned to vertices of the corresponding edge.

# Summary

- Dijkstra's algorithm
  - Keeps shortest length for each vertex from source  $s$
  - Keep predecessor with each vertex towards  $s$
  - Different from Prim's algorithm
    - Dijkstra: Chooses vertex with min shortest length
    - Prim: chooses edges with minimum weight.