

# Data Structures and Algorithms

Prof. Ganesh Ramakrishnan,  
Prof. Ajit Diwan,  
Prof. D.B. Phatak

Department of Computer Science and Engineering  
IIT Bombay

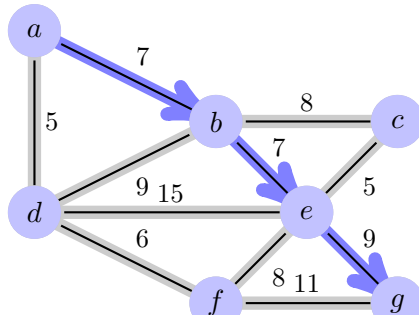
Session: Shortest Path Algorithm  
(Dijkstra's Algorithm)

# Finding Shortest Paths in Graphs

1. Shortest Path in Weighted Graphs
2. Shortest Path Properties in Weighted Graphs
3. Dijkstra's algorithm
4. Edge Relaxaton

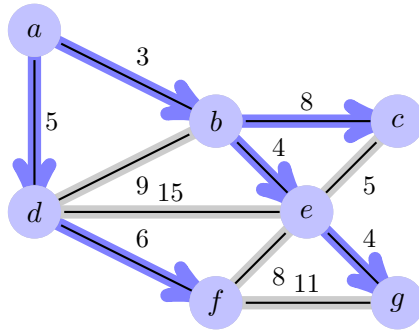
# Shortest Path in Weighted Graphs

1. Given two vertices  $a$  and  $g$  in the weighted graph below, find a path of minimum total weight between them.
2. Length of path = sum of weight of constituent edges
3. Applications: Routing of vehicles, flights, internet packets



# Shortest Path Properties in Weighted Graphs

1. **Optimal Substructure:** A subpath of a shortest path is also a shortest path
2. **Tree of shortest paths** exists<sup>1</sup> from a start vertex  $s$  to every other vertex



<sup>1</sup>Illustrated below is the tree of shortest paths from  $a$

# Dijkstra's Algorithm and Shortest Path Properties

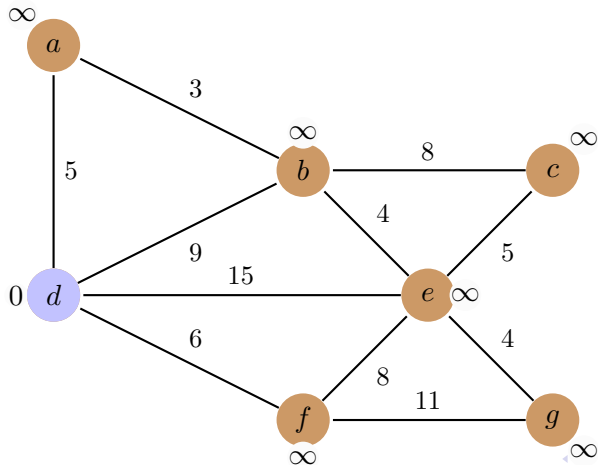
1. **Optimal Substructure:** A subpath of a shortest path is also a shortest path
2. **Tree of shortest paths** exist from a start vertex  $s$  to every other vertex
3. Dijkstra's Algorithm combines the two:
  - ▶ Store for each  $v$  a label  $d(v)$  = the distance of  $v$  from  $s$
  - ▶ Successively computes  $d(v)$ , starting from the neighbors of  $s$ , assuming
    - ▶ graph is connected
    - ▶ edges are undirected
    - ▶ edge weights are nonnegative

# Dijkstra's Shortest Path Algorithm

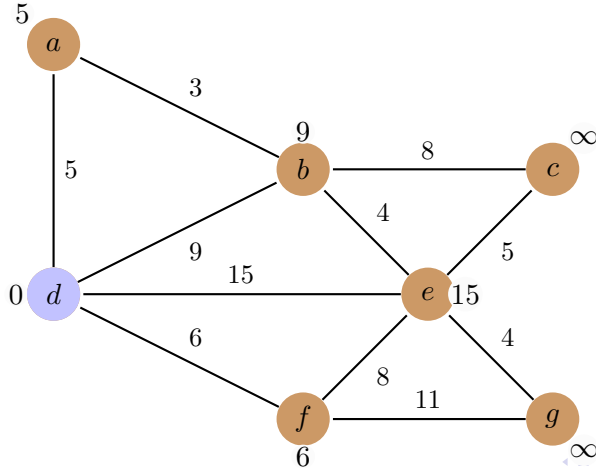
```
Algorithm ComputeDijkstraSPs( $G, s$ )  
Output: Array  $SP[.]$  with length of shortest path from  $s$  to  $v$  stored in  $SP[v]$ .  
 $P \leftarrow$  new Min-heap  
for  $v \in G.getVertices()$  do  
    if  $v = s$  then  
         $SP[v] = 0$   
    else  
         $SP[v] = \infty$   
    end if  
     $P.insert(v, SP[v])$   
end for  
while  $P.isNotEmpty()$  do  
     $u \leftarrow P.getMin()$   
     $P.removeMin()$  //Remove  $u$   
    for  $e \in G.incidentEdges(u)$  do  
         $w \leftarrow G.otherVertex(e, u)$   
         $r \leftarrow SP[u] + weight(e)$  //Relax  $e$   
        if  $r < SP[w]$  then  
             $SP[w] = r$   
             $P.update(w, r)$  //Update the value for key  $w$   
        end if  
    end for  
end while
```

Figure: Computing Shortest Paths using Dijkstras Algorithm

## Shortest Path (from $d$ )

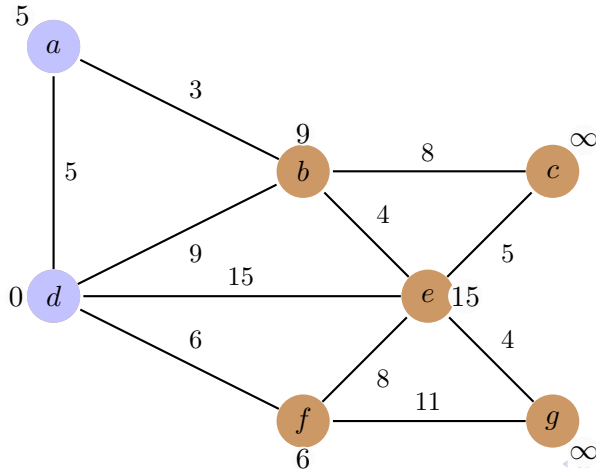


## Shortest Path (from $d$ ) contd.

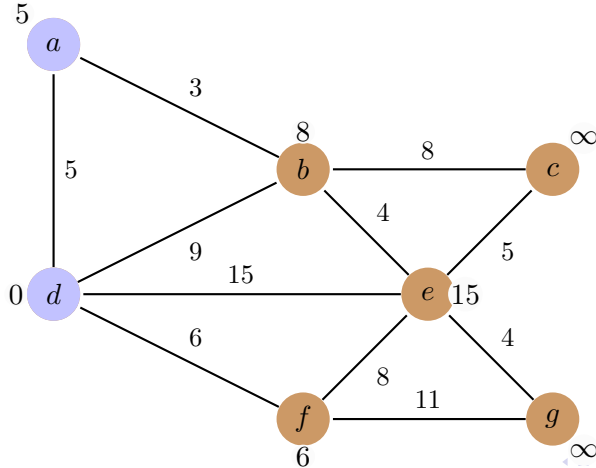




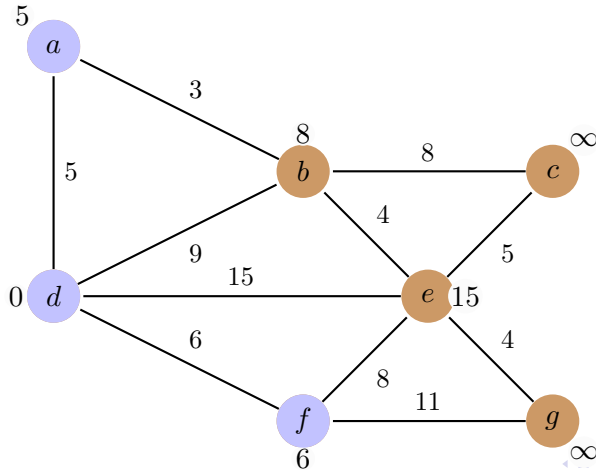
## Shortest Path (from $d$ ) contd.



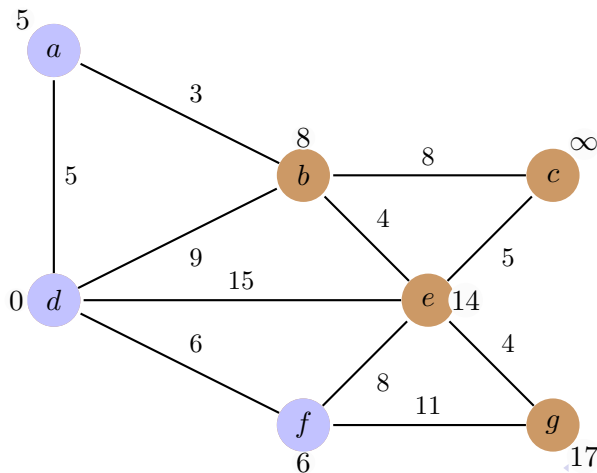
## Shortest Path (from $d$ ) contd.



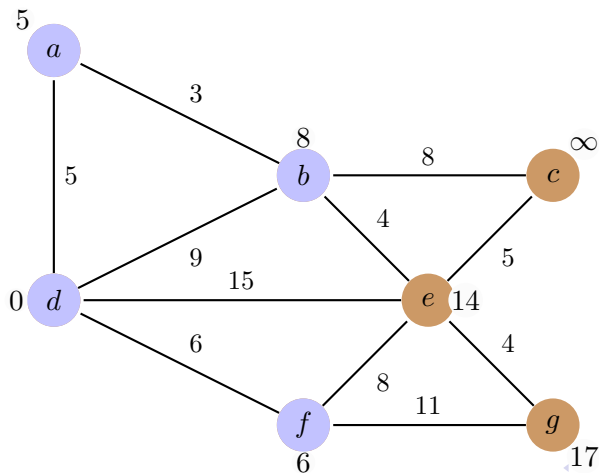
## Shortest Path (from $d$ ) contd.



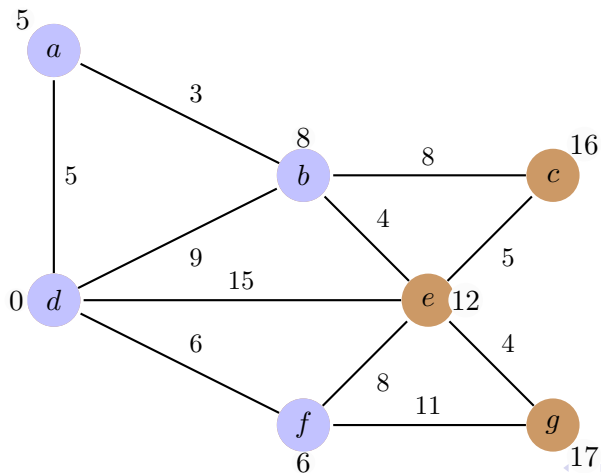
## Shortest Path (from $d$ ) contd.



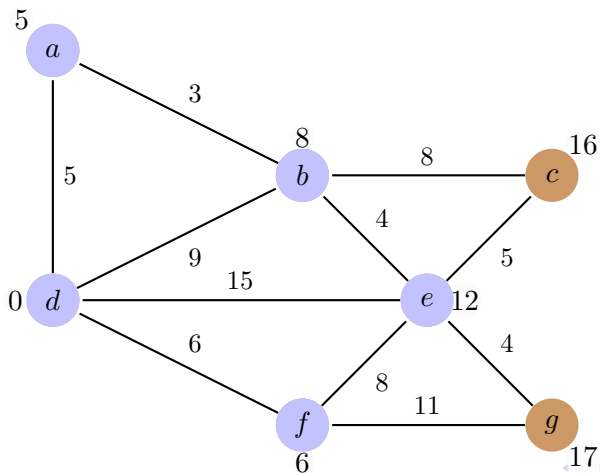
## Shortest Path (from $d$ ) contd.



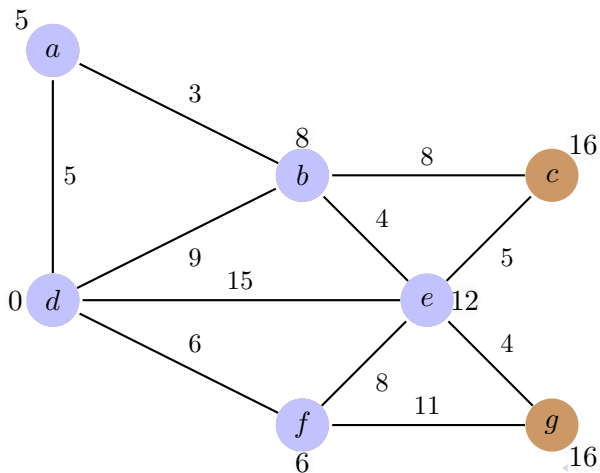
## Shortest Path (from $d$ ) contd.



## Shortest Path (from $d$ ) contd.

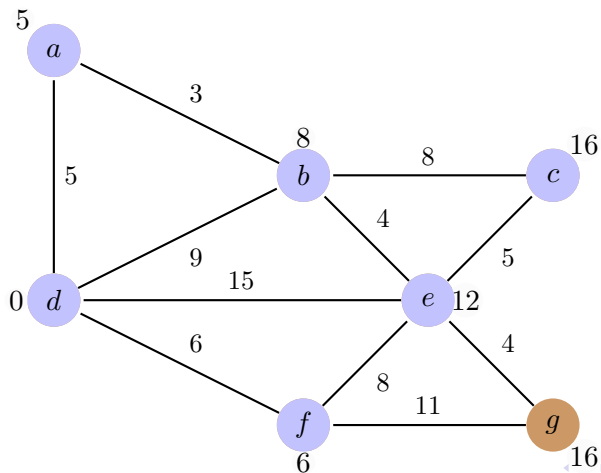


## Shortest Path (from $d$ ) contd.

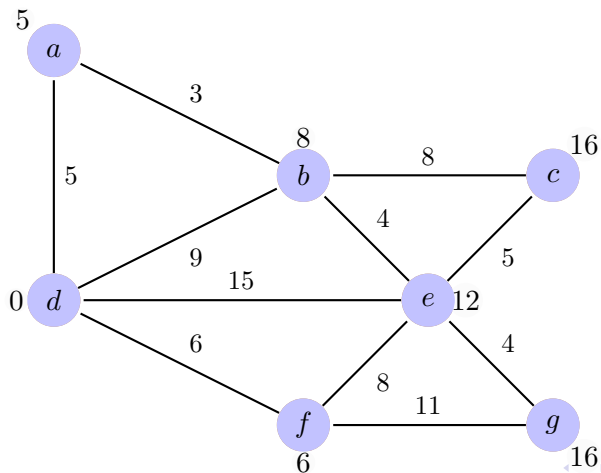




## Shortest Path (from $d$ ) contd.



## Shortest Path (from $d$ ) contd.



# Analysis of Dijkstra's Algorithm

**Algorithm** ComputeDijkstraSPs( $G, s$ )

**Output:** Array  $SP[.]$  with length of shortest path from  $s$  to  $v$  stored in  $SP[v]$ .

$P \leftarrow$  new Min-heap

**for**  $v \in G.getVertices()$  **do**

**if**  $v = s$  **then**

$SP[v] = 0$

**else**

$SP[v] = \infty$

**end if**

$P.insert(v, SP[v])$

**end for**  $\implies c_1 \times |V|$  times

**while**  $P.isNotEmpty()$  **do**

$u \leftarrow P.getMin()$   $\implies c_2 \times \log |V|$  times

$P.removeMin()$  //Remove  $u$

**for**  $e \in G.incidentEdges(u)$  **do**

$w \leftarrow G.otherVertex(e, u)$

$r \leftarrow SP[u] + weight(e)$  //Relax  $e$

**if**  $r < SP[w]$  **then**

$SP[w] = r$

$P.update(w, r)$  //Update the value for key  $w \implies c_3 \times \log |V|$  times

**end if**

**end for**  $\implies c_4 \times deg(v)$  times

**end while**  $\implies c_5 \times |V|$  times

$$T(n) = c_3 c_4 \log |V| \sum_{v \in V} deg(v) + c_2 c_5 |V| (\log |V|) + c_1 |V| = O(|E| + |V|) \log |V|$$

**Thank you**