

# CSE 417

# Algorithms and Complexity

Autumn 2020

Lecture 11

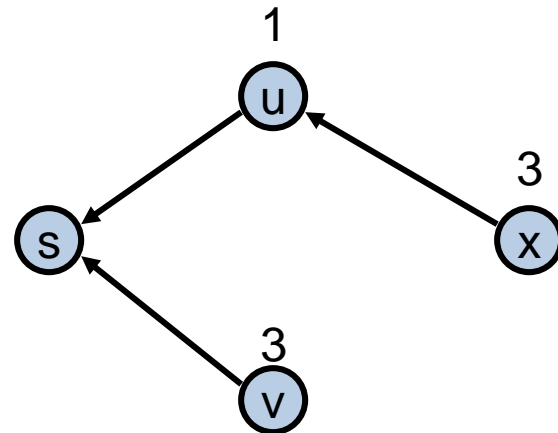
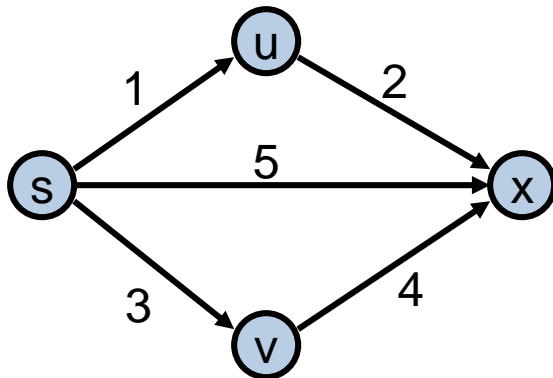
Dijkstra's algorithm

# Upcoming lectures

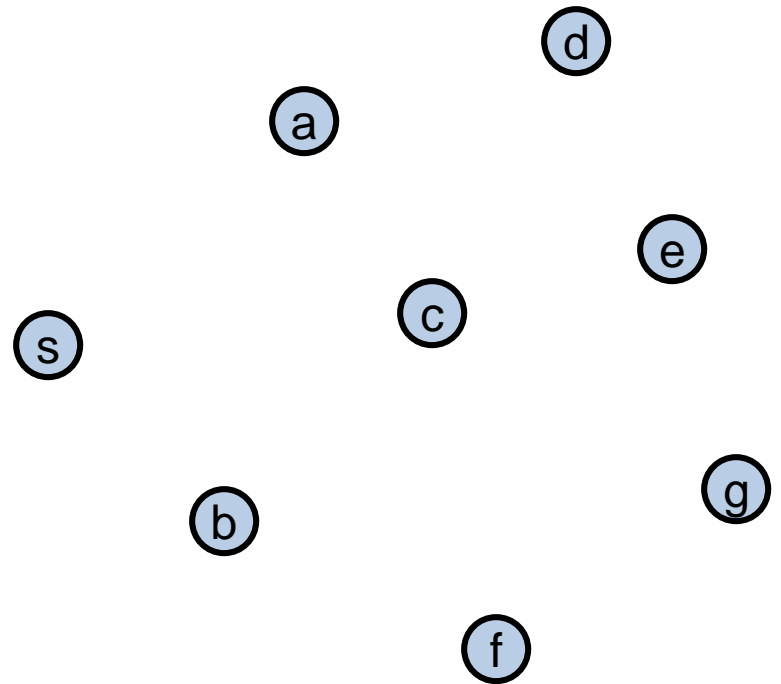
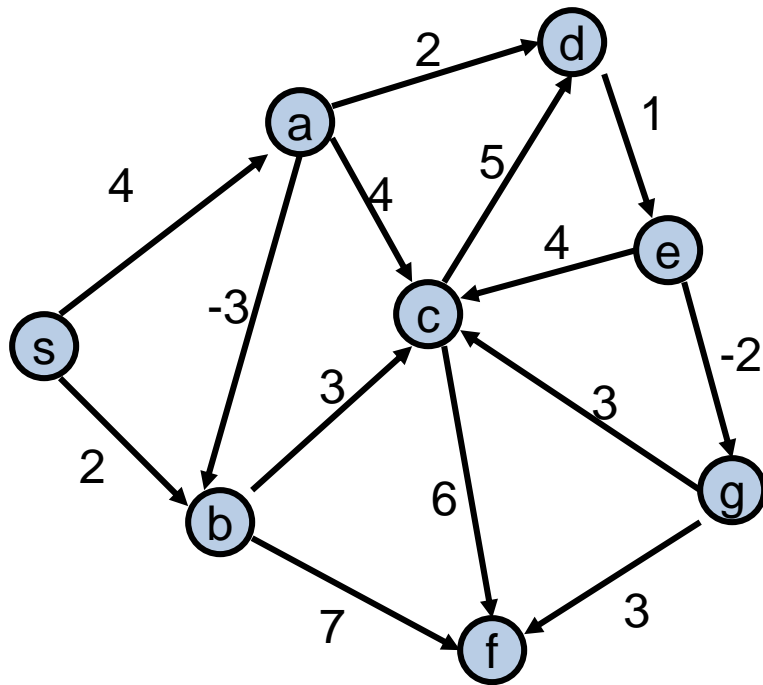
- Topics
  - Dijkstra's Algorithm (Section 4.4)
  - Monday: Minimum Spanning Trees
- Reading
  - 4.4, 4.5, 4.7, 4.8

# Single Source Shortest Path Problem

- Given a graph and a start vertex  $s$ 
  - Determine distance of every vertex from  $s$
  - Identify shortest paths to each vertex
    - Express concisely as a “shortest paths tree”
    - Each vertex has a pointer to a predecessor on shortest path

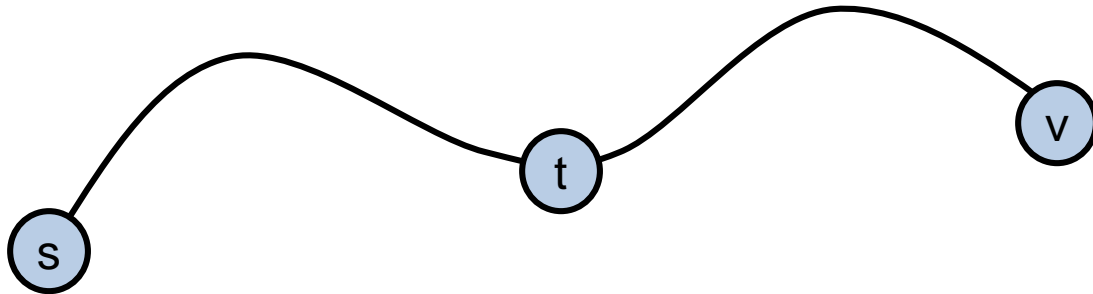


# Construct Shortest Path Tree from s



# Warmup

- If  $P$  is a shortest path from  $s$  to  $v$ , and if  $t$  is on the path  $P$ , the segment from  $s$  to  $t$  is a shortest path between  $s$  and  $t$



- WHY?

Assume all edges have non-negative cost

# Dijkstra's Algorithm

$S = \{ \}$ ;  $d[s] = 0$ ;  $d[v] = \text{infinity}$  for  $v \neq s$

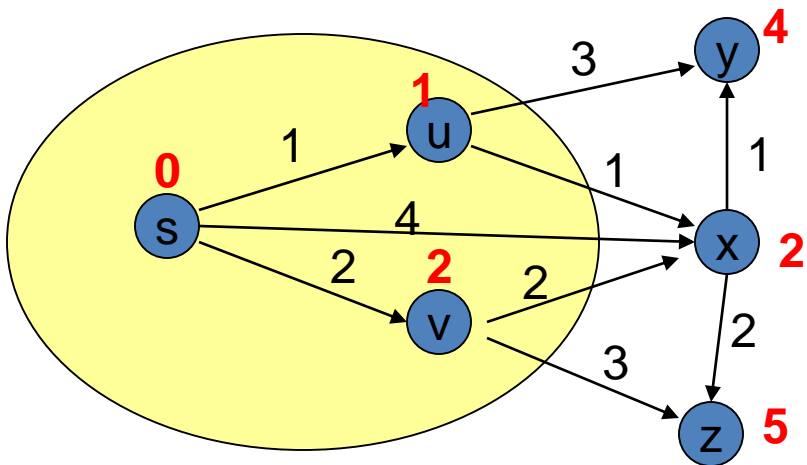
While  $S \neq V$

    Choose  $v$  in  $V-S$  with minimum  $d[v]$

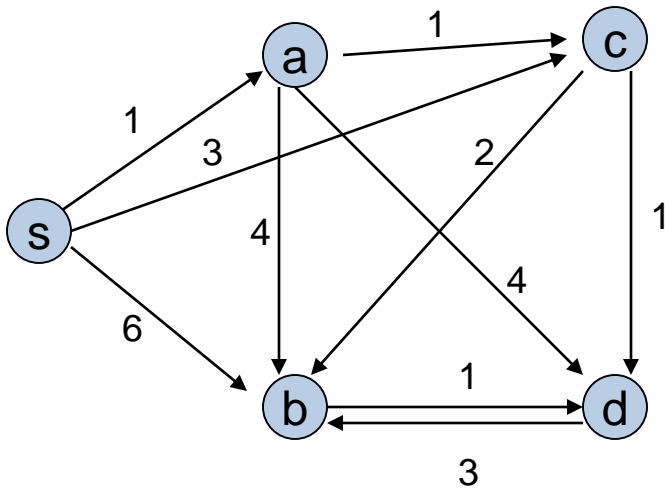
    Add  $v$  to  $S$

    For each  $w$  in the neighborhood of  $v$

$$d[w] = \min(d[w], d[v] + c(v, w))$$



# Simulate Dijkstra's algorithm (starting from s) on the graph



Round	Vertex Added	s	a	b	c	d
1						
2						
3						
4						
5						

# Who was Dijkstra?

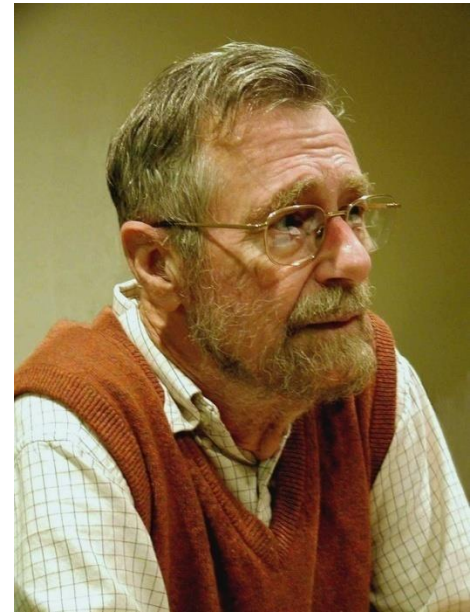
- What were his major contributions?





<http://www.cs.utexas.edu/users/EWD/>

- Edsger Wybe Dijkstra was one of the most influential members of computing science's founding generation. Among the domains in which his scientific contributions are fundamental are
  - algorithm design
  - programming languages
  - program design
  - operating systems
  - distributed processing
  - formal specification and verification
  - design of mathematical arguments

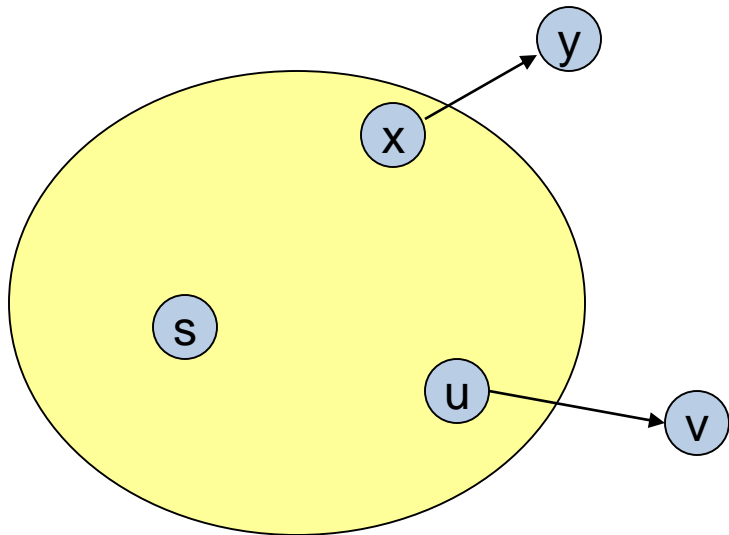


# Dijkstra's Algorithm as a greedy algorithm

- Elements committed to the solution by order of minimum distance

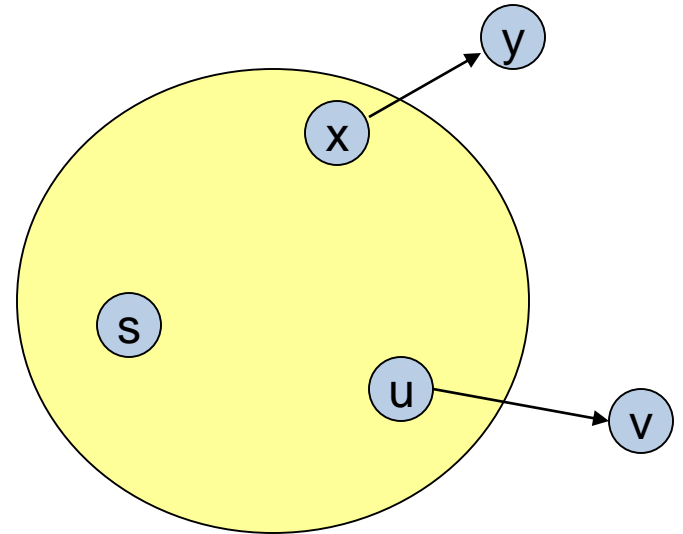
# Correctness Proof

- Elements in  $S$  have the correct label
- Key to proof: when  $v$  is added to  $S$ , it has the correct distance label.



# Proof

- Let  $v$  be a vertex in  $V-S$  with minimum  $d[v]$
- Let  $P_v$  be a path of length  $d[v]$ , with an edge  $(u,v)$
- Let  $P$  be some other path to  $v$ . Suppose  $P$  first leaves  $S$  on the edge  $(x, y)$ 
  - $P = P_{sx} + c(x,y) + P_{yv}$
  - $\text{Len}(P_{sx}) + c(x,y) \geq d[y]$
  - $\text{Len}(P_{yv}) \geq 0$
  - $\text{Len}(P) \geq d[y] + 0 \geq d[v]$



# Negative Cost Edges

- Draw a small example a negative cost edge and show that Dijkstra's algorithm fails on this example

# Dijkstra Implementation

$S = \{ \}; \quad d[s] = 0; \quad d[v] = \text{infinity for } v \neq s$

While  $S \neq V$

    Choose  $v$  in  $V-S$  with minimum  $d[v]$

    Add  $v$  to  $S$

    For each  $w$  in the neighborhood of  $v$

$$d[w] = \min(d[w], d[v] + c(v, w))$$

- Basic implementation requires Heap for tracking the distance values
- Run time  $O(m \log n)$

# $O(n^2)$ Implementation for Dense Graphs

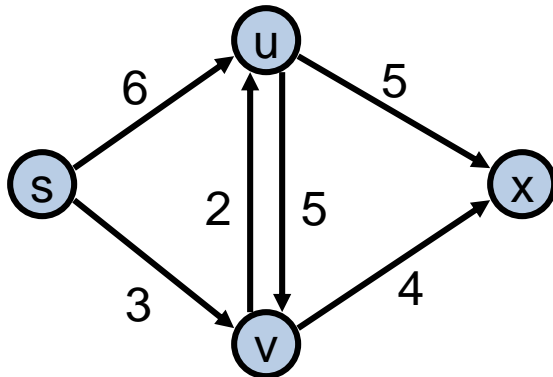
```
FOR i := 1 TO n
    d[i] := Infinity;  visited[i] := FALSE;
d[s] := 0;

FOR i := 1 TO n
    v := -1;  dMin := Infinity;
    FOR j := 1 TO n
        IF visited[j] = FALSE AND d[j] < dMin
            v := j;  dMin := d[j];
    IF v = -1
        RETURN;
    visited[v] := TRUE;

    FOR j := 1 TO n
        IF d[v] + len[v, j] < d[j]
            d[j] := d[v] + len[v, j];
            prev[j] := v;
```

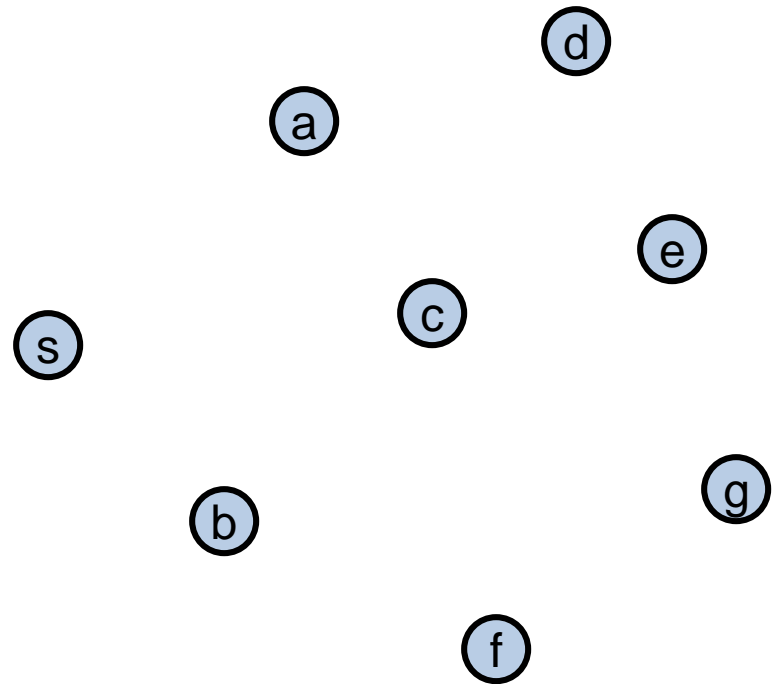
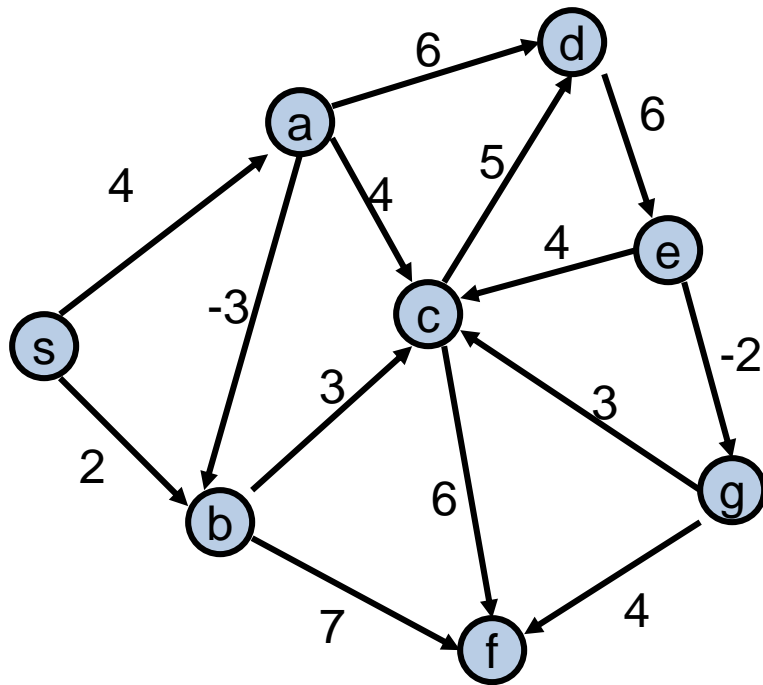
# Bottleneck Shortest Path

- Define the bottleneck distance for a path to be the maximum cost edge along the path





# Compute the bottleneck shortest paths



How do you adapt Dijkstra's algorithm to handle bottleneck distances

- Does the correctness proof still apply?