CSE 421
Algorithms

Richard Anderson

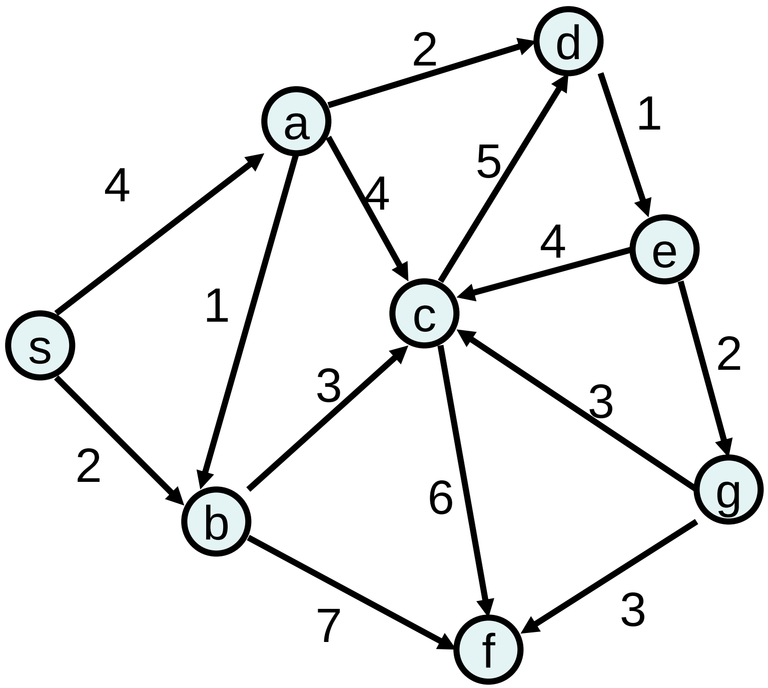
Lecture 12, Autumn 2019

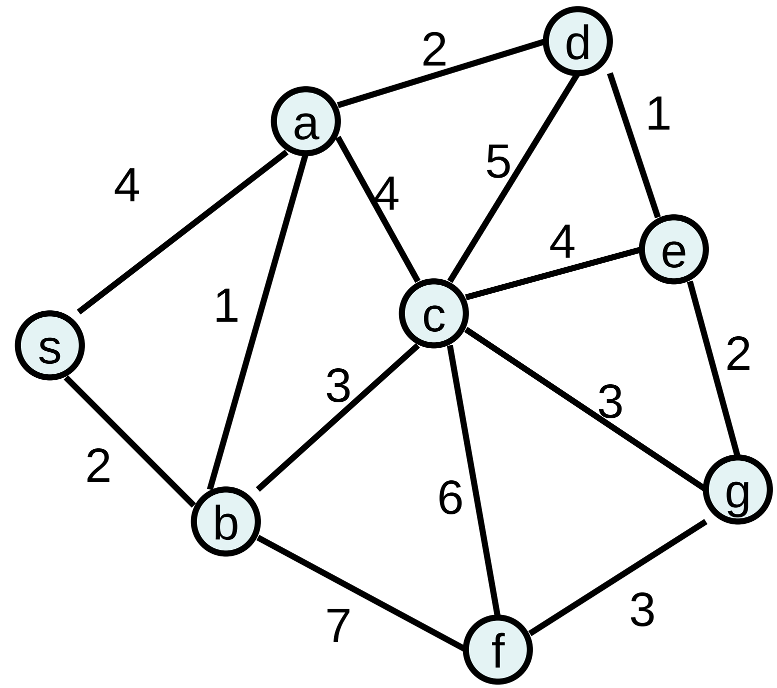
Recurrences

Announcements

- Midterm, Wednesday, October 30
  - Coverage through KT 5.5
  - Old midterms posted

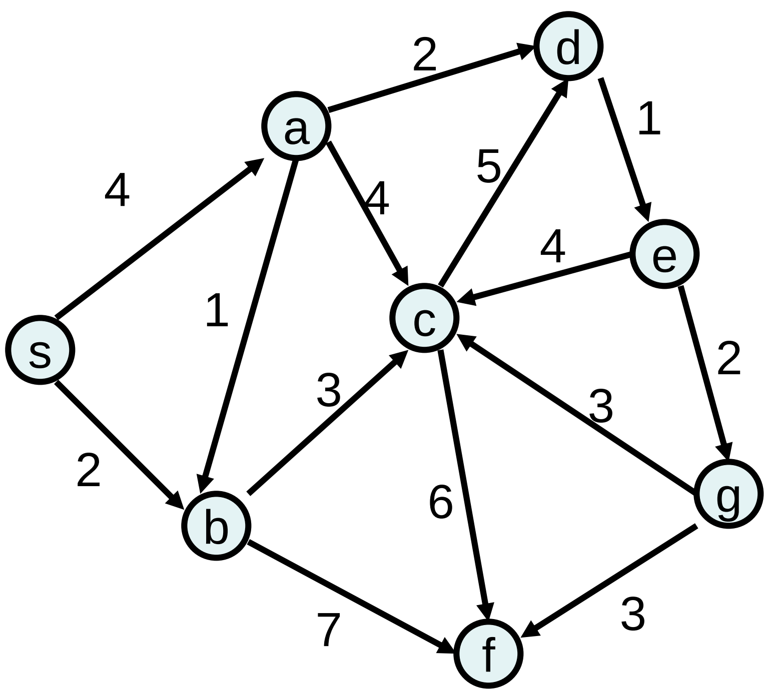
Shortest paths in directed graphs vs undirected graphs



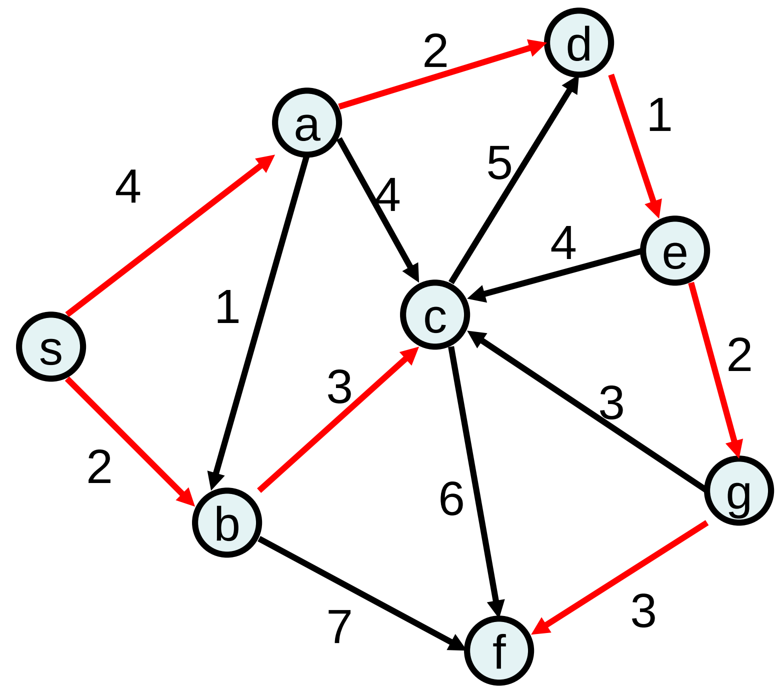


What about the minimum spanning tree of a directed graph?

- Must specify the root r
- Branching: Out tree with root r

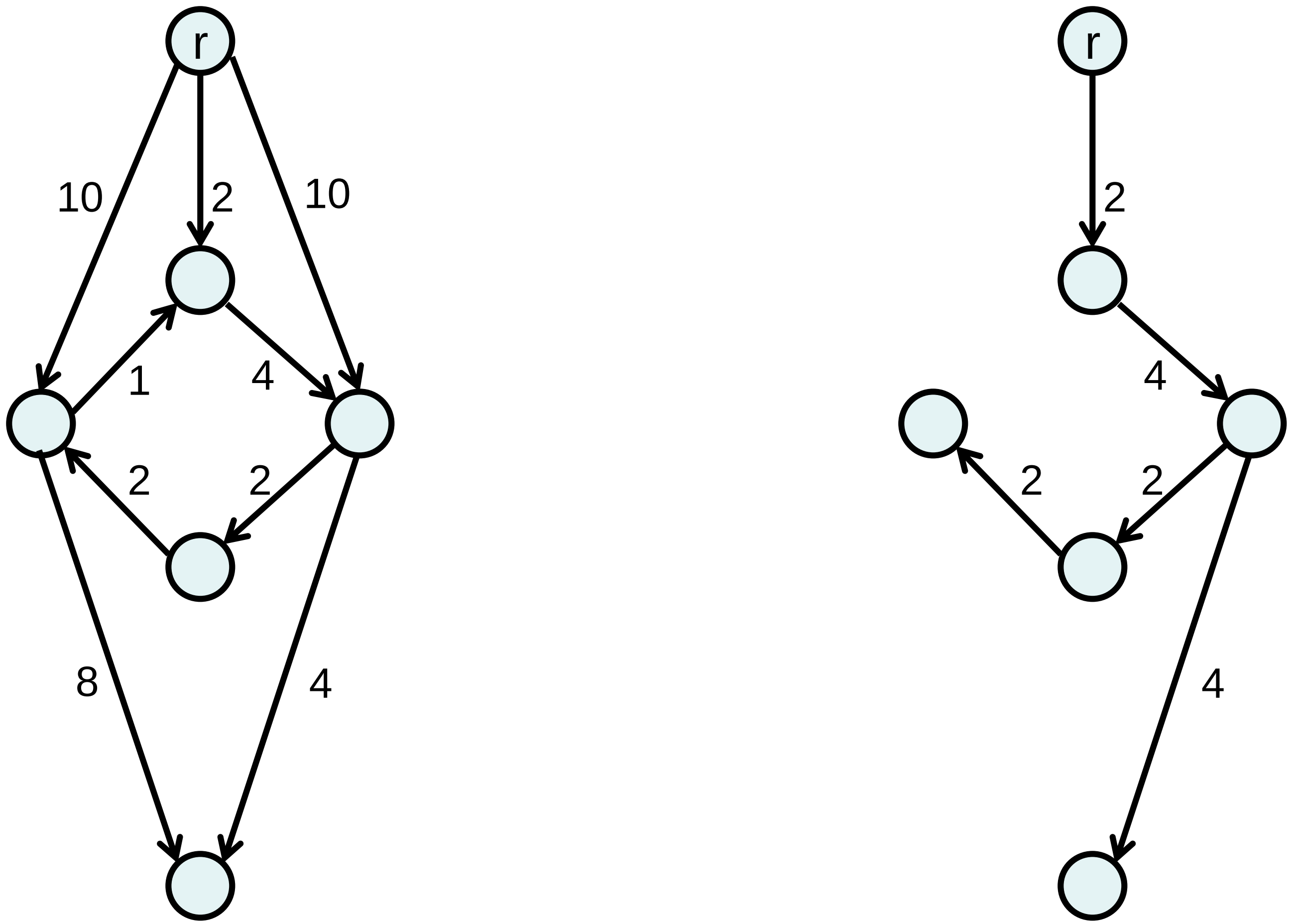


Assume all vertices reachable from r



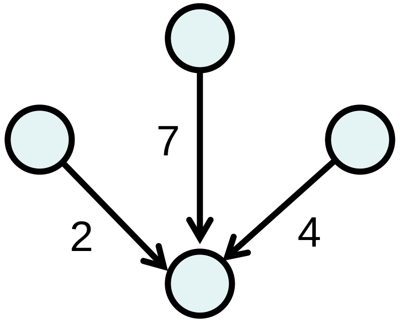
Also called an arborescence

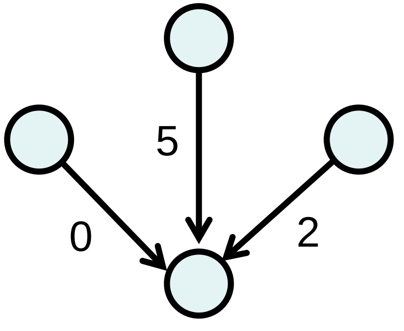
Finding a minimum branching



Finding a minimum branching

- Remove all edges going into r
- Normalize the edge weights, so the minimum weight edge coming into each vertex has weight zero



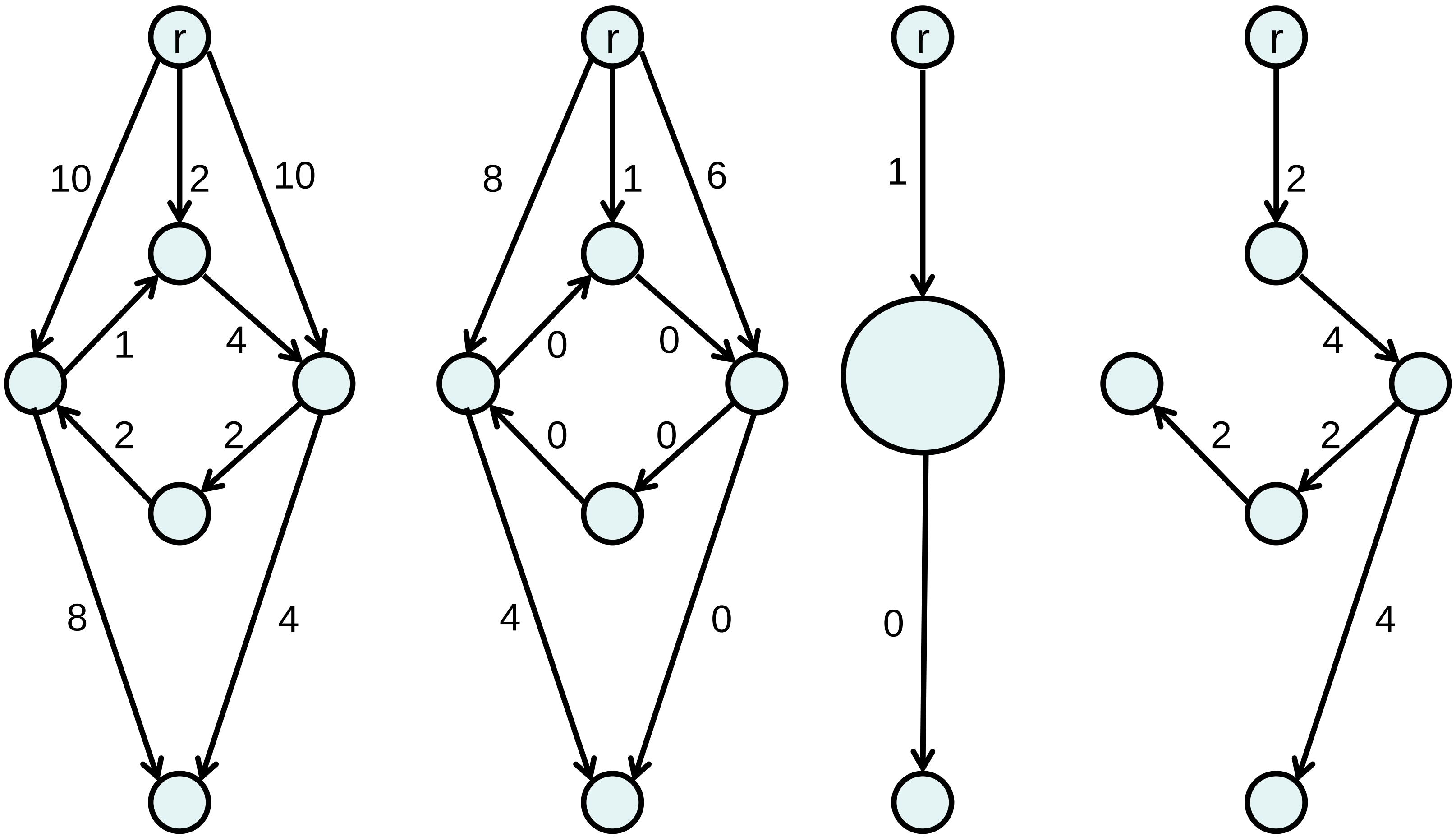


This does not change the edges of the minimum branching

Finding a minimum branching

- Consider the graph that consists of the minimum cost edge coming in to each vertex
  - If this graph is a branching, then it is the minimum cost branching
  - Otherwise, the graph contains one or more cycles
    - Collapse the cycles in the original graph to super vertices
    - Reweight the graph and repeat the process

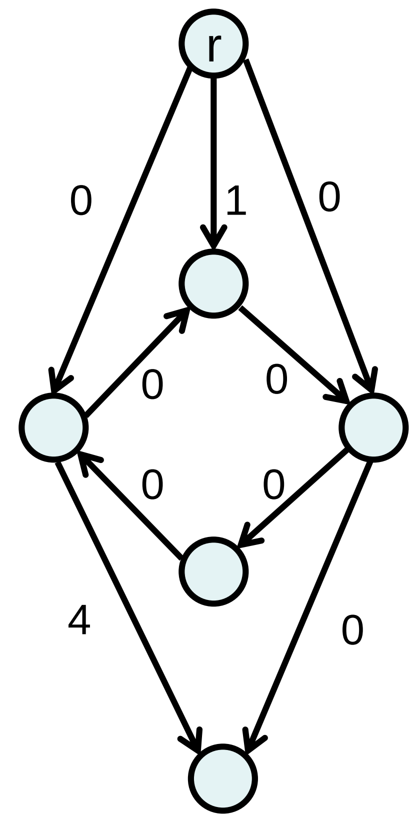
Finding a minimum branching



Correctness Proof

Lemma 4.38 Let C be a cycle in G consisting of edges of cost 0 with r not in C. There is an optimal branching rooted at r that has exactly one edge entering C.

- The lemma justifies using the edges of the cycle in the branching
- An induction argument is used to cover the multiple levels of compressing cycles



Divide and Conquer

- Recurrences, Sections 5.1 and 5.2
- Algorithms
  - Fast Matrix Multiplication
  - Counting Inversions (5.3)
  - Closest Pair (5.4)
  - Multiplication (5.5)

Divide and Conquer

```
Array Mergesort(Array a){
        n = a.Length;
        if (n <= 1)
                return a;
        b = Mergesort(a[0 .. n/2]);
        c = Mergesort(a[n/2+1 .. n-1]);
        return Merge(b, c);
}
```

Algorithm Analysis

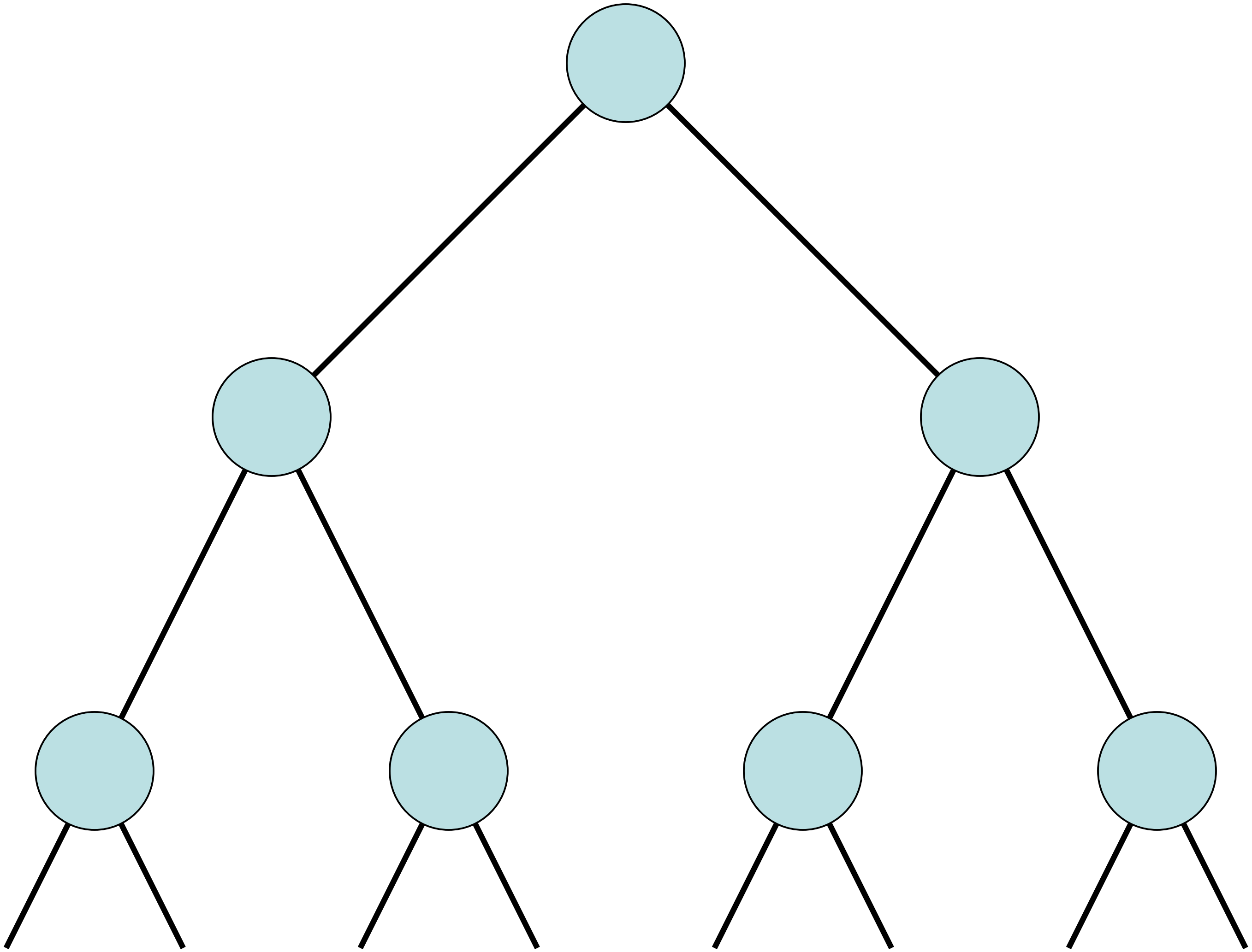
- Cost of Merge
- Cost of Mergesort

$$T(n) = 2T(n/2) + cn; \; T(1) = c;$$

Recurrence Analysis

- Solution methods
  - Unrolling recurrence
  - Guess and verify
  - Plugging in to a “Master Theorem”

Unrolling the recurrence



Substitution

Prove $T(n) <= cn (\log_2 n + 1)$ for $n >= 1$

Induction:

Base Case:

Induction Hypothesis:

A better mergesort (?)

- Divide into 3 subarrays and recursively sort
- Apply 3-way merge

What is the recurrence?

Unroll recurrence for $T(n) = 3T(n/3) + dn$

$$T(n) = aT(n/b) + f(n)$$

$$T(n) = T(n/2) + cn$$

Where does this recurrence arise?

Solving the recurrence exactly

$$T(n) = 4T(n/2) + n$$

$$T(n) = 2T(n/2) + n^2$$

$$T(n) = 2T(n/2) + n^{1/2}$$

Recurrences

- Three basic behaviors
  - Dominated by initial case
  - Dominated by base case
  - All cases equal – we care about the depth