

# CSE 417

# Algorithms and Complexity

Richard Anderson  
Lecture 13, Winter 2020  
Recurrences

# Announcements

- Midterm, Monday, February 10, 2020
  - Coverage through KT 5.2
  - Sample midterm questions will be posted
  - 50 minutes
  - Closed book
  - No notes
  - No calculators or electronic devices

# 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

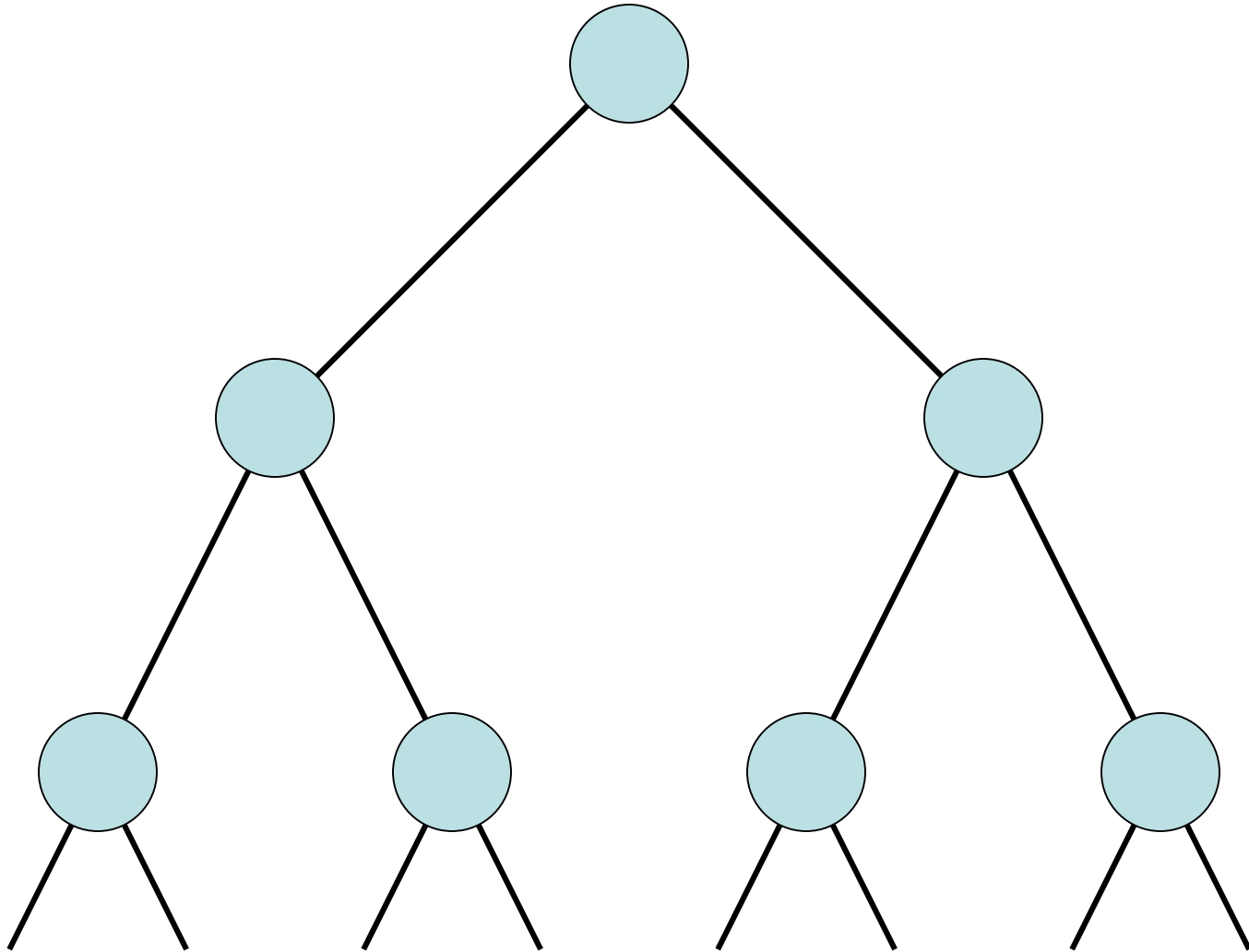
- Cost of Merge
- Cost of Mergesort

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

# Recurrence Analysis

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

# Unrolling the recurrence





$$T(n) = 2T(n/2) + n; T(1) = 1;$$

# Substitution

Prove  $T(n) \leq n (\log_2 n + 1)$  for  $n \geq 1$

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