

# CSE 421

# Algorithms

Richard Anderson

Lecture 13, Autumn 2019

Recurrences, Part 2

# Announcements

- Midterm, Oct 30
  - 50 minutes
  - In class
  - Closed book
  - No calculators

# Mergesort Recurrence

- $T(n) = 2 T(n/2) + n$ ;  $T(1) = 1$ 
  - $O(n \log n)$

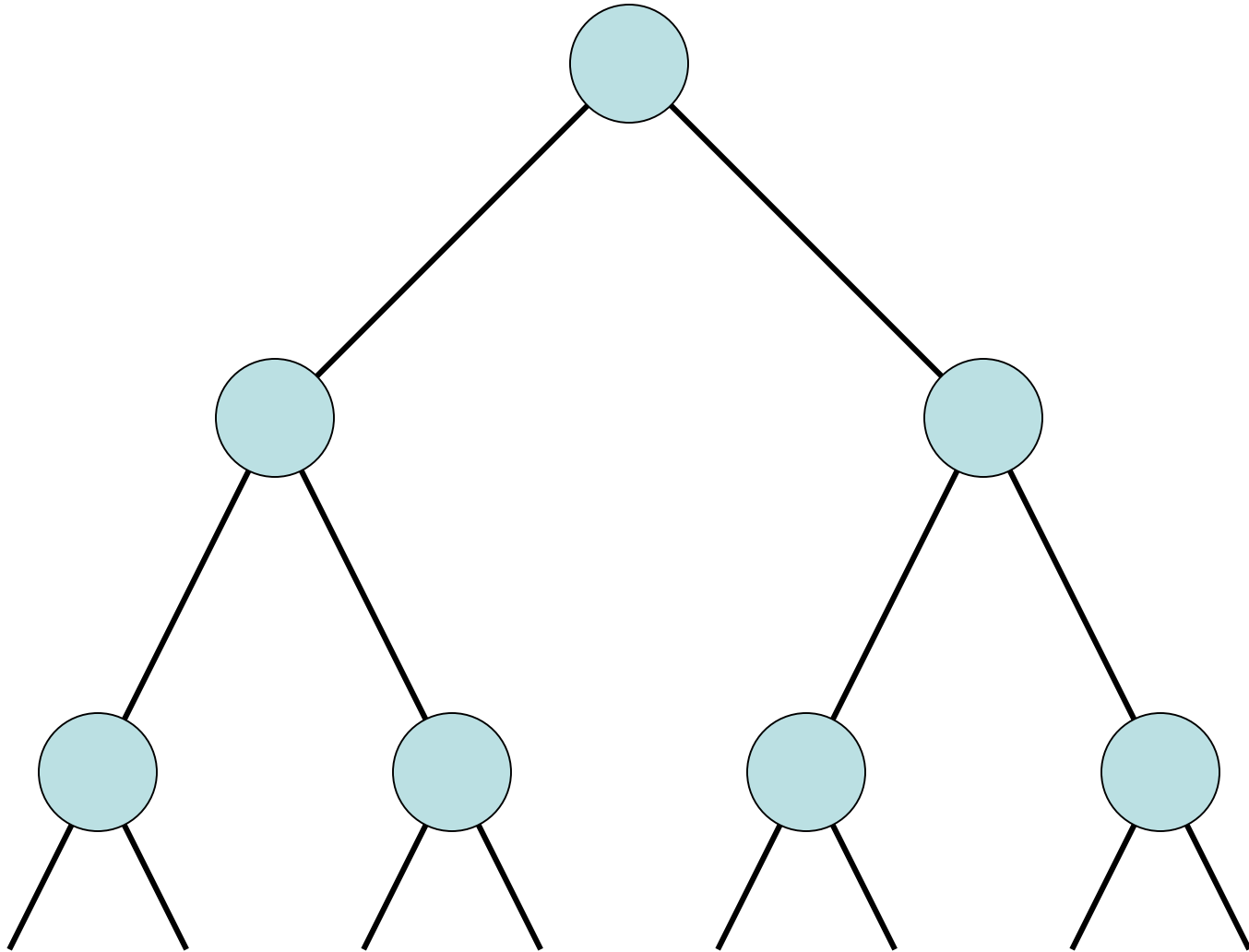
- Useful facts:

- $\log_k n = \log_2 n / \log_2 k$

- $k^{\log n} = n^{\log k}$

$$\sum_{i=0}^n x^i = \frac{1 - x^{n+1}}{1 - x}$$

# Unrolling the recurrence



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

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

Where does this recurrence arise?

# Solving the recurrence exactly

# Recursive Matrix Multiplication

Multiply 2 x 2 Matrices:

$$\begin{vmatrix} r & s \\ t & u \end{vmatrix} = \begin{vmatrix} a & b \\ c & d \end{vmatrix} \begin{vmatrix} e & g \\ f & h \end{vmatrix}$$

$$r = ae + bf$$

$$s = ag + bh$$

$$t = ce + df$$

$$u = cg + dh$$

A  $N \times N$  matrix can be viewed as a  $2 \times 2$  matrix with entries that are  $(N/2) \times (N/2)$  matrices.

The recursive matrix multiplication algorithm recursively multiplies the  $(N/2) \times (N/2)$  matrices and combines them using the equations for multiplying  $2 \times 2$  matrices



# Recursive Matrix Multiplication

- How many recursive calls are made at each level?
- How much work in combining the results?
- What is the recurrence?

# What is the run time for the recursive Matrix Multiplication Algorithm?

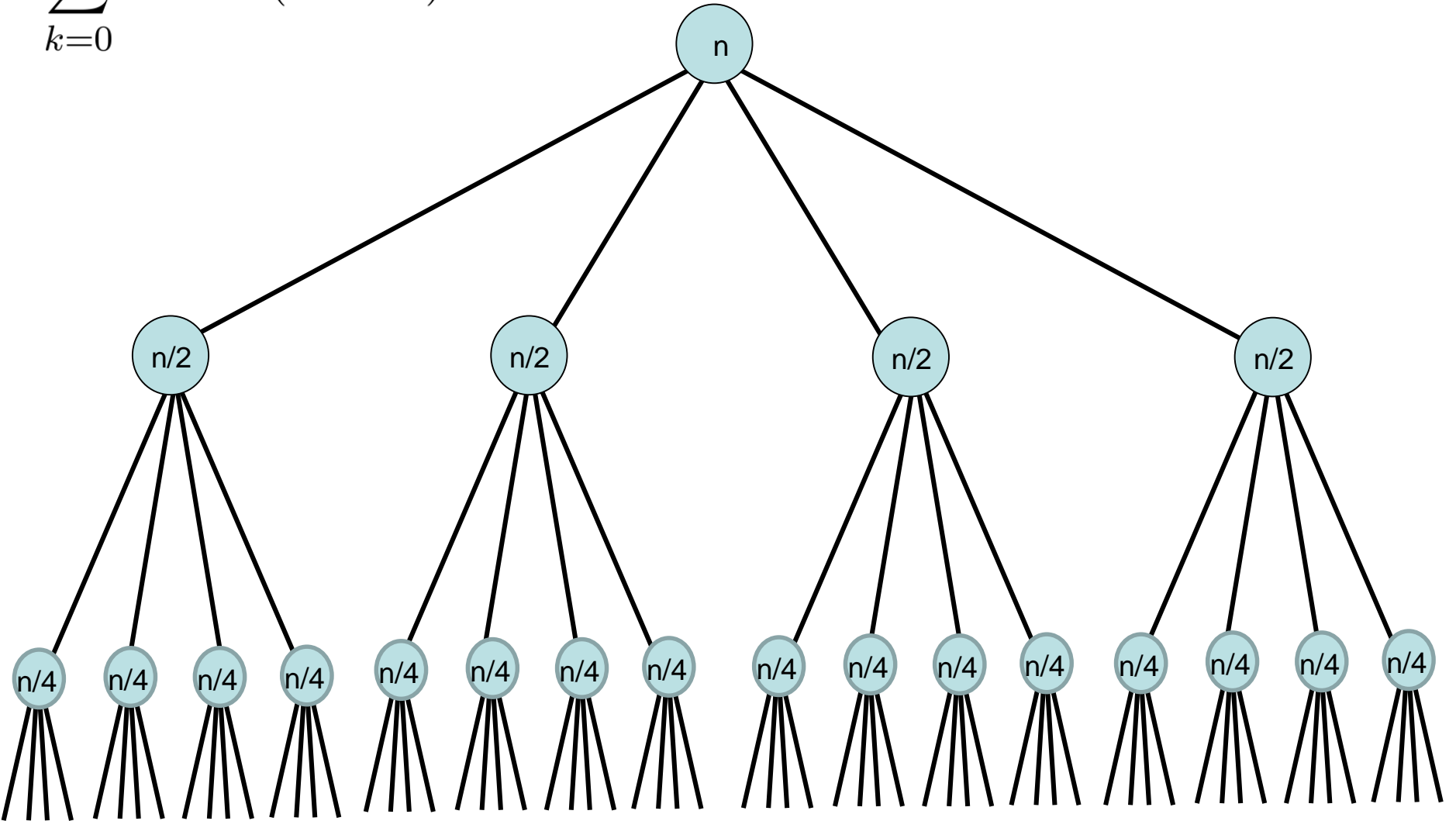
- Recurrence:

Total Work

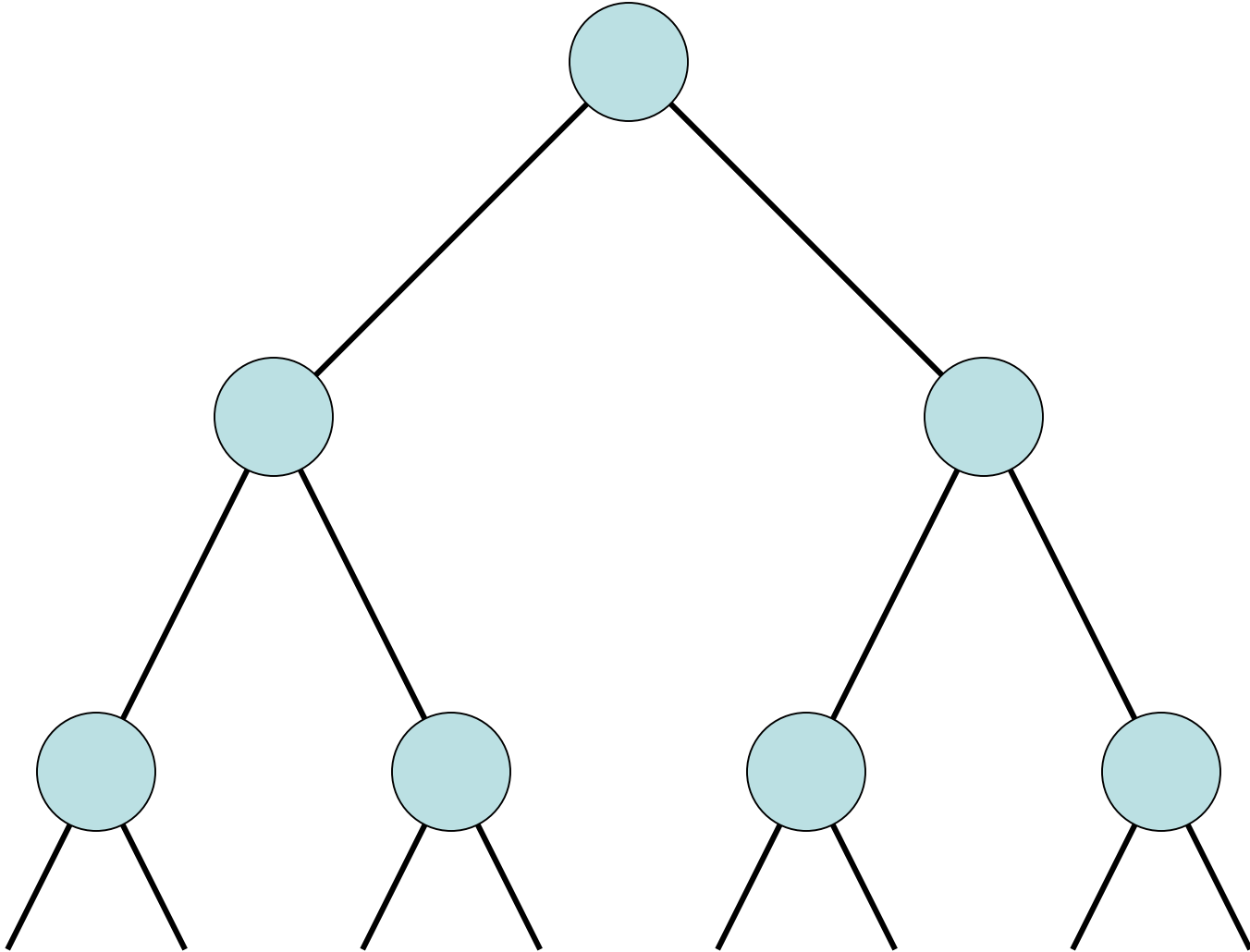
$\log n$

$$\sum_{k=0} 2^k n = (2n - 1)n$$

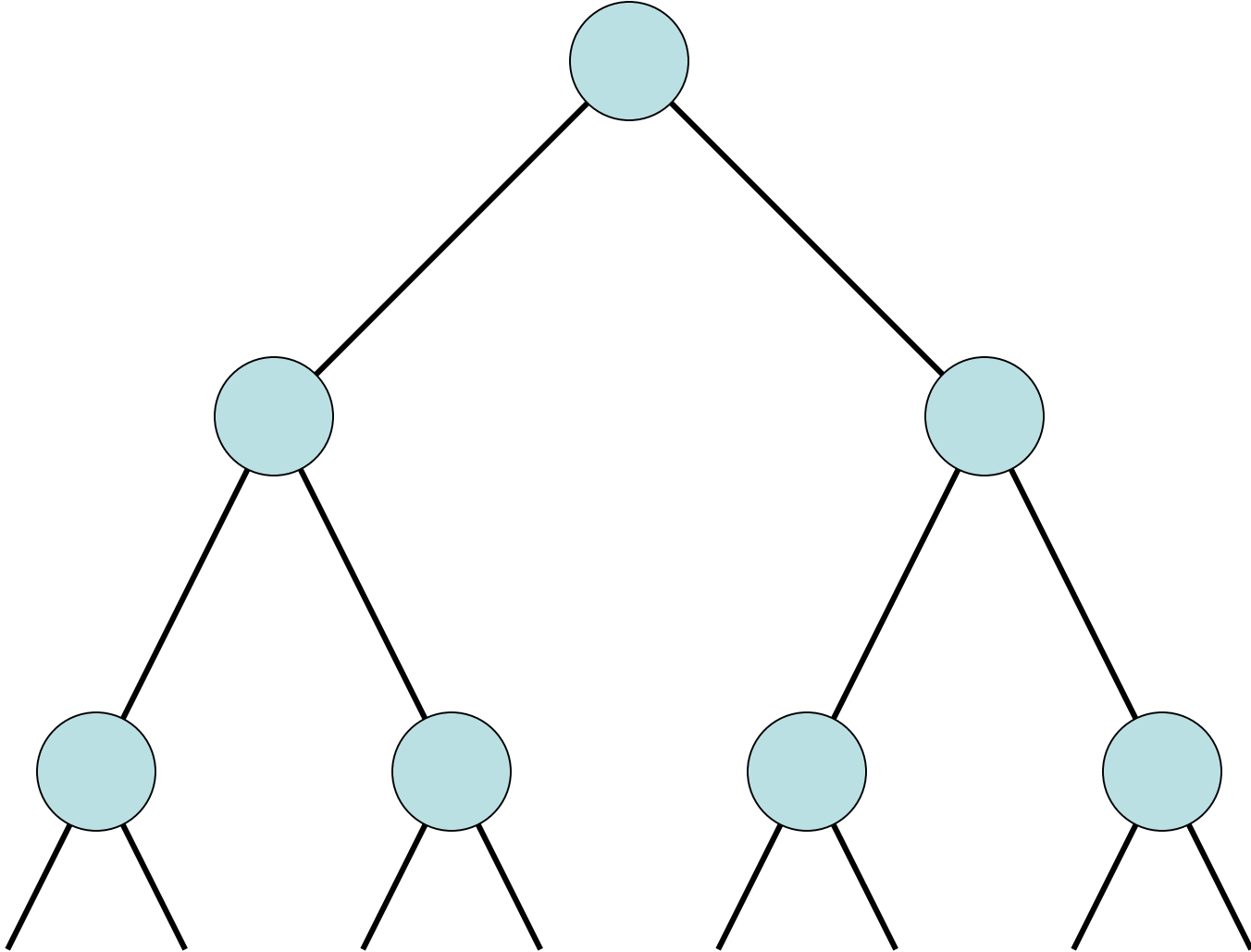
$$T(n) = 4T(n/2) + n$$



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



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



# Recurrences

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

# What you really need to know about recurrences

- Work per level changes geometrically with the level
- Geometrically increasing ( $x > 1$ )
  - The bottom level wins
- Geometrically decreasing ( $x < 1$ )
  - The top level wins
- Balanced ( $x = 1$ )
  - Equal contribution

# Classify the following recurrences (Increasing, Decreasing, Balanced)

- $T(n) = n + 5T(n/8)$
- $T(n) = n + 9T(n/8)$
- $T(n) = n^2 + 4T(n/2)$
- $T(n) = n^3 + 7T(n/2)$
- $T(n) = n^{1/2} + 3T(n/4)$



# Strassen's Algorithm

Multiply 2 x 2 Matrices:

$$\begin{array}{|c|c|} \hline r & s \\ \hline t & u \\ \hline \end{array} = \begin{array}{|c|c|} \hline a & b \\ \hline c & d \\ \hline \end{array} \begin{array}{|c|c|} \hline e & g \\ \hline f & h \\ \hline \end{array}$$

$$r = p_1 + p_2 - p_4 + p_6$$

$$s = p_4 + p_5$$

$$t = p_6 + p_7$$

$$u = p_2 - p_3 + p_5 - p_7$$

Where:

$$p_1 = (b - d)(f + h)$$

$$p_2 = (a + d)(e + h)$$

$$p_3 = (a - c)(e + g)$$

$$p_4 = (a + b)h$$

$$p_5 = a(g - h)$$

$$p_6 = d(f - e)$$

$$p_7 = (c + d)e$$

# Recurrence for Strassen's Algorithms

- $T(n) = 7 T(n/2) + cn^2$
- What is the runtime?

$$\log_2 7 = 2.8073549221$$

# BFPRT Recurrence

$$T(n) \leq T(3n/4) + T(n/5) + 20n$$

What bound do you expect?