Lecture09

CSE 421
Introduction to Algorithms

Richard Anderson

Lecture 9, Winter 2024

Recurrences

1

Announcements

- Divide and Conquer and Recurrences
  - Recurrence Techniques
  - Fast Matrix Multiplication
  - Counting Inversions (5.3)
  - Closest Pair (5.4)
  - Multiplication (5.5)
  - Quicksort and Median Finding
- Dynamic Programming
- Midterm, Friday, February 9

2

Merge $O(n)$

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);
}
```

3

Algorithm Analysis

- Cost of Merge = $Merge(n) = O(n)$
- Cost of Mergesort

$$MS(n) = cn + 2\,MS(\tfrac{n}{2})$$

$$MS(1) = 1$$

4

$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/3 \rfloor) + \ldots$

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

5

Recurrence Analysis

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

6

Useful Math Facts

$$k^{\log_k n} = n$$

$$\log_k n = \frac{\log_2 n}{\log_2 k}$$

$$k^{\log_2 n} = n^{\log_2 k}$$

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

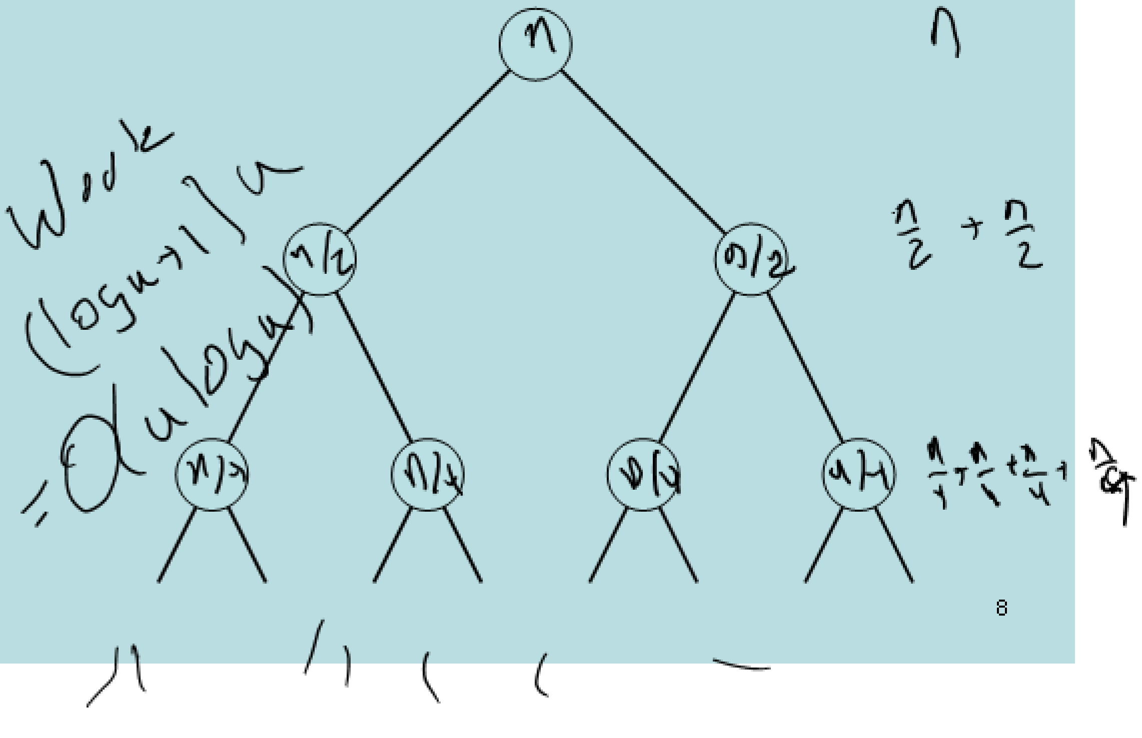
$\log k^{\log_k n} = \log_2 n \cdot \log_2 k$

$7^{\log_2 n} = n^{\log_2 n}$

7

$T(n) = 2T(\frac{n}{2}) + n \; ; \; T(1) = 1$

Unrolling the recurrence



8

n

$2 \cdot \frac{n}{2} = n$

$4 \cdot \frac{n}{4} = n$

$2^k \cdot \frac{n}{2^k} = n$

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: $1 = T(1)$

Induction Hypothesis: $T(n/2) \leq \frac{n}{2}(\log \frac{n}{2} + 1)$

$T(n) = 2T(\frac{n}{2}) + n$

$\leq 2\frac{n}{2}(\log \frac{n}{2} + 1) + n \leq n(\log n - 1 + 1) + n$

$= n(\log n + 1)$ ✓

9

Master Theorem

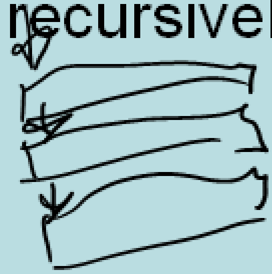
- $T(n) = a\,T(n/b) + O(n^d)$

- $T(n) = O(n^d)$ if $d > \log_b a$
- $T(n) = O(n^d \log n)$ if $d = \log_b a$
- $T(n) = O(n^{\log_b a})$ if $d < \log_b a$

10

A better mergesort (?)

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



$O(n)$

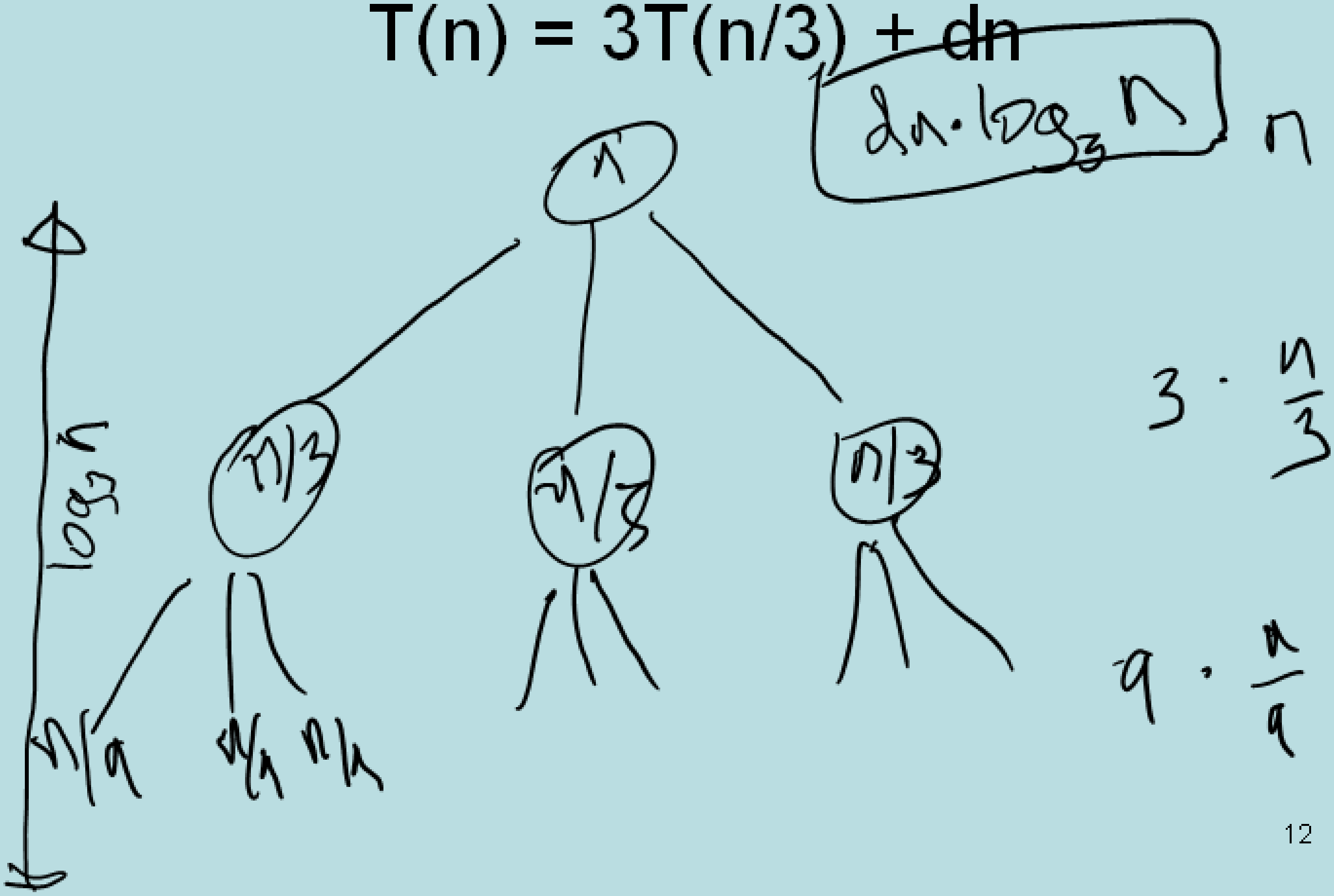
$$T(n) = 3T\left(\frac{n}{3}\right) + n$$

$$T(1) = 1$$

What is the recurrence?

11

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



12

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

13

$BS(n) = BS(\frac{n}{2}) + 1$

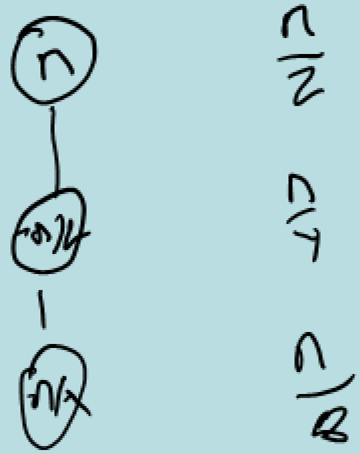
T(n) = T(n/2) + cn

Where does this recurrence arise?

Tournament.

$T(n) = \frac{n}{2} + T(\frac{n}{2})$

$T(1) = 0$

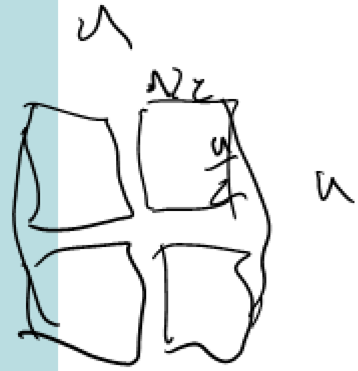


$T(n) = T(\frac{2n}{3}) + n$

14

Standard MM = $O(n^3)$

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 x N matrix can be viewed as a 2 x 2 matrix with entries that are (N/2) x (N/2) matrices.

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

Recursive Matrix Multiplication

- How many recursive calls are made at each level? 8
- How much work in combining the results? $O(n^2)$
- What is the recurrence?

$$T(n) = 8\,T\left(\frac{n}{2}\right) + n^2$$

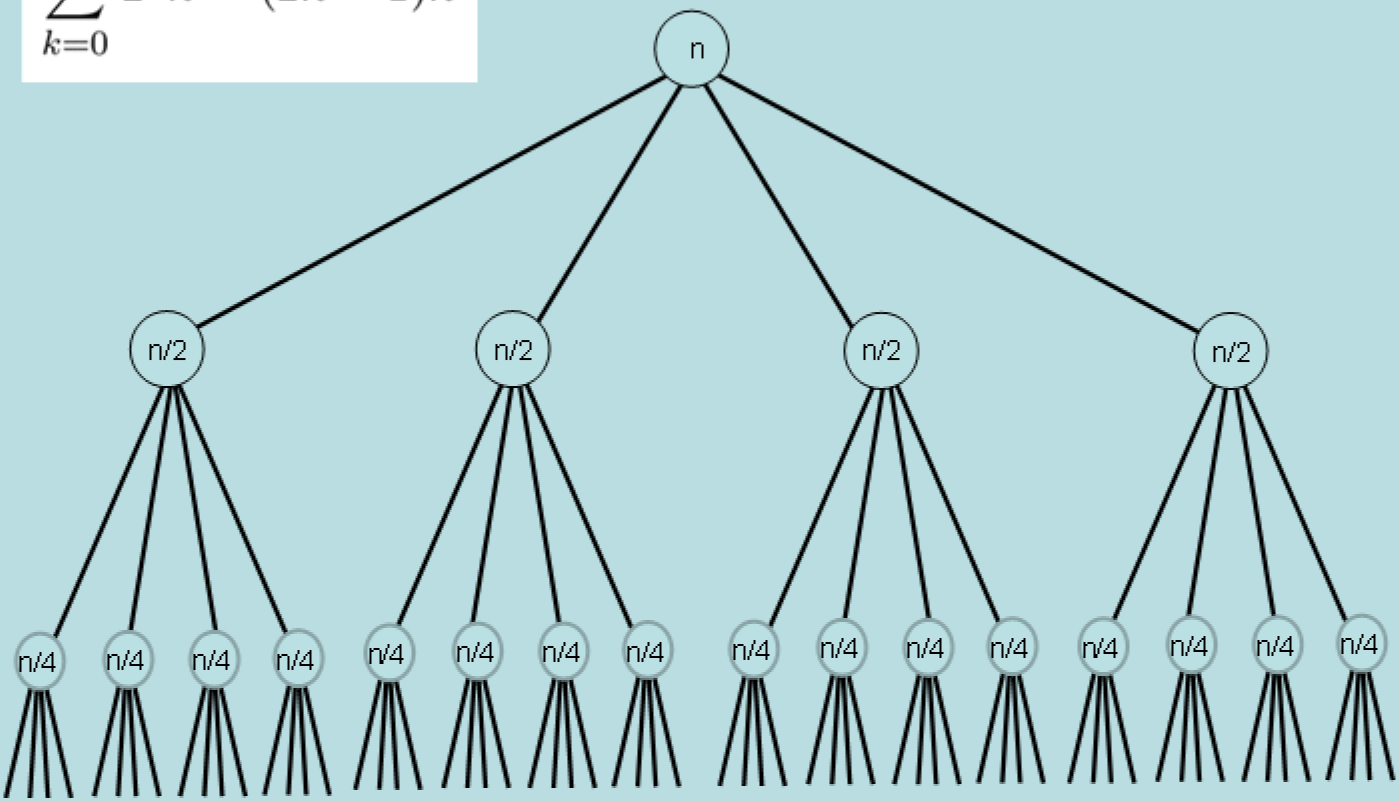
What is the run time for the recursive Matrix Multiplication Algorithm?

- Recurrence:

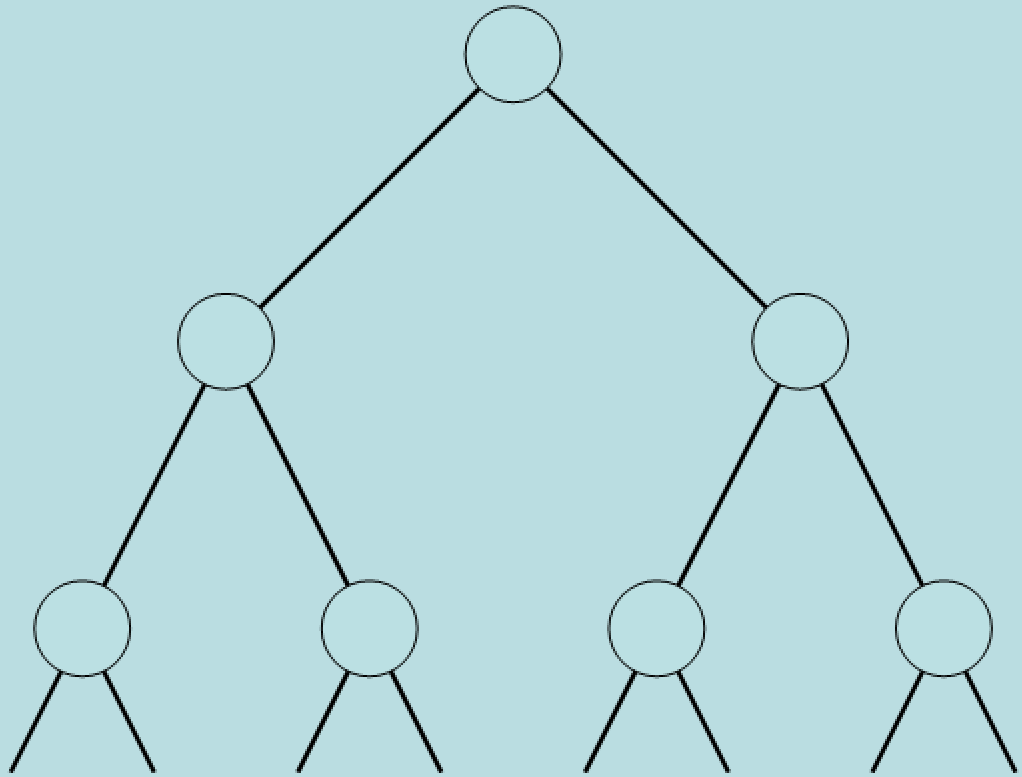
Total Work

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

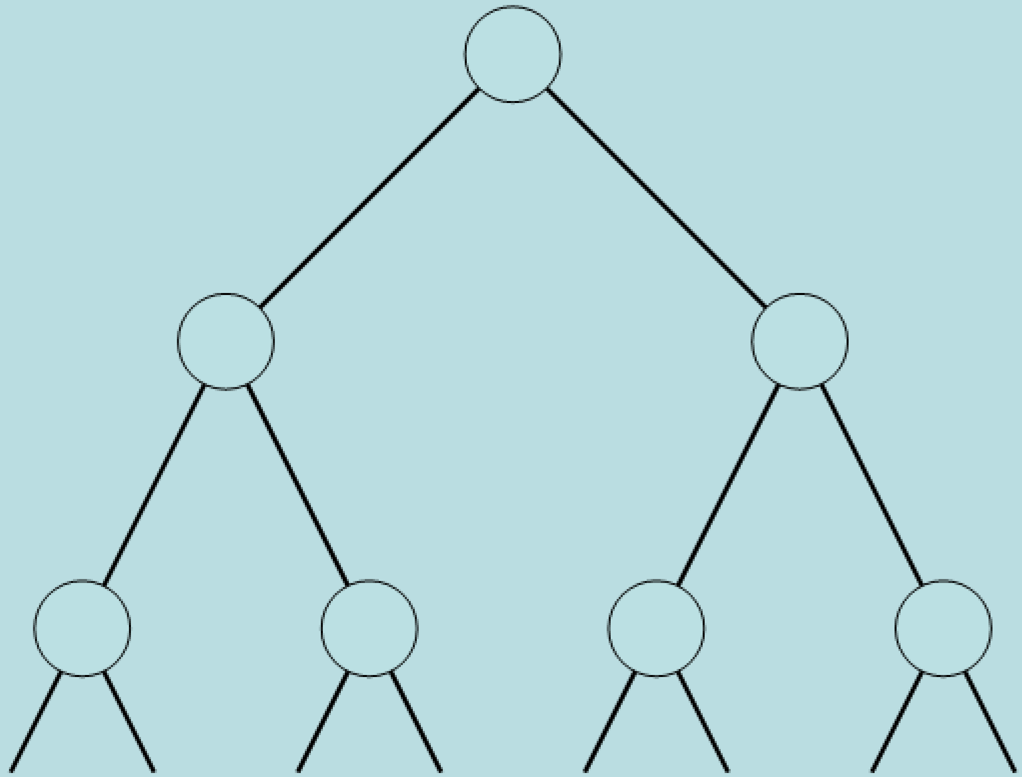
T(n) = 4T(n/2) + n



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



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



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)

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