COMPSCI 311: Introduction to Algorithms

Lecture 12: Divide and Conquer

Dan Sheldon

University of Massachusetts Amherst

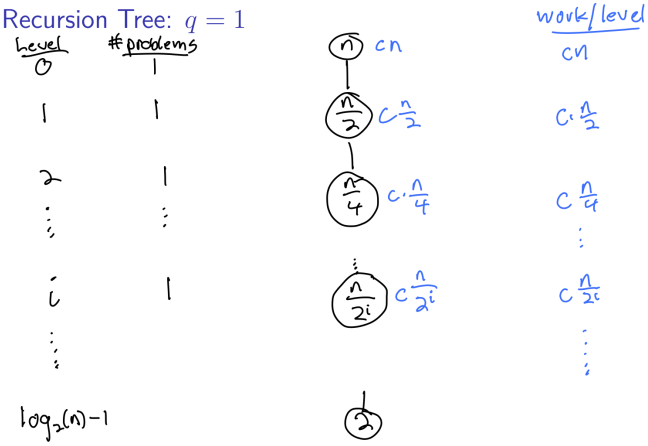
A More General Recurrence

$$T(n) \le q \cdot T(n/2) + cn$$
$$T(2) \le c$$

- What does the algorithm look like?
  - q recursive calls to itself on problems of half the size
  - $O(n)$ work outside of the recursive calls
- Exercises / board work: $q = 1$, $q > 2$ (recursion trees)

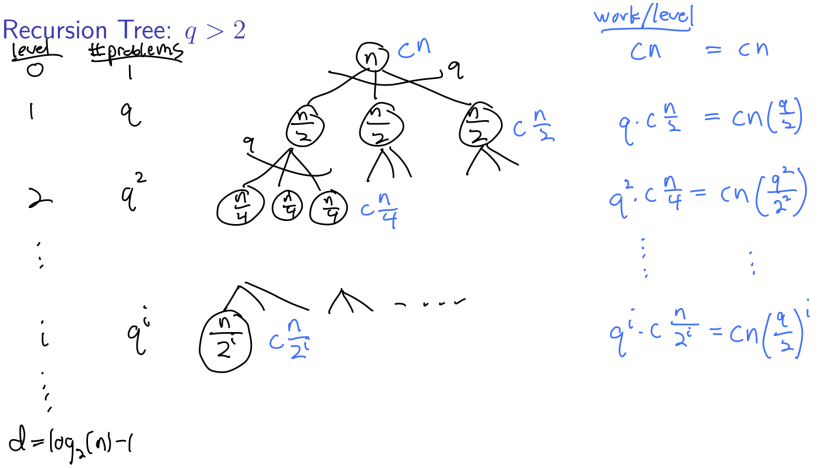
Review

Recursion Tree: $q = 1$



Review

Recursion Tree: $q > 2$



General Case

Work at level j of recursion tree:

$$\underbrace{q^j}_{\text{num. subproblems}} \times \underbrace{cn/2^j}_{\text{work per subproblem}} = \left(\frac{q}{2}\right)^j cn$$

Total work:

$$T(n) = cn \cdot \sum_{j=0}^{d} \left(\frac{q}{2}\right)^j \qquad (d = \log_2 n - 1)$$

$q = 1$? Easy to see $T(n) \leq 2cn = O(n)$.

Useful Fact: Geometric Sum

If $r \neq 1$ then

$$1 + r + r^2 + \ldots + r^d = \frac{r^{d+1} - 1}{r - 1}$$

General Case ($q > 2$)

$$T(n) = cn \cdot \sum_{j=0}^{d} \left(\frac{q}{2}\right)^j \qquad (d = \log_2 n - 1)$$

Let $r = q/2 > 1$. Then

$$\begin{aligned}\sum_{j=0}^{d} r^j &= \frac{r^{d+1} - 1}{r - 1} \\ &\leq \tfrac{1}{r-1} r^{d+1} \\ &= \tfrac{1}{r-1} \left(\tfrac{q}{2}\right)^{\log_2 n} \\ &= \tfrac{1}{r-1} n^{\log_2 \frac{q}{2}} \\ &= O(n^{\log_2 q - 1})\end{aligned}$$

Therefore,

$$\begin{aligned}T(n) &= cn \cdot O(n^{\log_2 q - 1}) \\ &= O(n^{\log_2 q})\end{aligned}$$

E.g., $q = 3$, $T(n)$ is $O(n^{1.59})$

Summary

Useful general recurrence and its solutions:

$$\boxed{T(n) \leq q \cdot T(n/2) + cn}$$

1. $q = 1$: $T(n) = O(n)$ — dominated by root
2. $q = 2$: $T(n) = O(n \log n)$ — same work every level
3. $q > 2$: $T(n) = O(n^{\log_2 q})$ — dominated by leaves

Work at is either exponentially decreasing, staying same, or exponentially increasing with level

Algorithms with these recurrences?

1. ???
2. MSS, Mergesort
3. Integer multiplication...

Clicker Question

Which of the following is *not* true ?

A. $n \log n = O(n^2)$

B. $n \log n = O(n^{1.1})$

C. There exists some k such that $n \log n = \Theta(n^k)$.

D. $n \log n = \Omega(n \log \log n)$

Integer Multiplication

Motivation: multiply two 30-digit integers?

```
  153819617987625488624070712657
x 925421863832406144537293648227
  --------------------------------
```

- Multiply two 300-digit integers?
- Cannot do this in Java with built-in data types
- 64-bit unsigned integer can only represent integers up to ~20 digits ($2^{64} \approx 10^{20}$)

Warm-Up: Addition

Input: two n-digit binary integers x and y
Goal: compute $x + y$

Let's do everything in base-10 instead of binary to make examples more familiar.

Grade-school algorithm:

```
   1854
 + 3242
 -------
   5096
```

Running time? $\Theta(n)$

Integer Multiplication Problem

Input: two n-digit base-10 integers x and y
Goal: compute xy

Can anyone think of an algorithm?

Grade-School Algorithm (Long Multiplication)

Example: $n = 3$

```
    287
  x 132
 ------
    574
   861
  287
--------
  37884
```

$$287 \times 132 = (2 \times 287) + 10 \cdot (3 \times 287) + 100 \cdot (1 \times 287)$$

Running time? $\Theta(n^2)$
But xy has at most $2n$ digits. Can we do better?

Divide and Conquer – First Try: An Example

Idea: split x and y in half (assume n is a power of 2)

$$x = \underbrace{3380}_{x_1}\underbrace{2367}_{x_0}$$
$$y = \underbrace{4508}_{y_1}\underbrace{1854}_{y_0}$$

Then use distributive law

$$\begin{aligned} xy &= (10^{n/2}x_1 + x_0) \times (10^{n/2}y_1 + y_0) \\ &= 10^n x_1 y_1 + 10^{n/2}(x_1 y_0 + x_0 y_1) + x_0 y_0 \end{aligned}$$

Have reduced the problem to multiplications of $n/2$-digit integers and additions of n-digit numbers.
(Ignore time to multiply by 10^k. Why?)

Divide and Conquer – First Try: Analysis

Recursive algorithm:

$$xy = 10^n x_1 y_1 + 10^{n/2}(x_1 y_0 + x_0 y_1) + x_0 y_0$$

Running time?

Four multiplications of $n/2$ digit numbers plus three additions of at most n-digit numbers

$$\begin{aligned} T(n) &\leq 4T\left(\frac{n}{2}\right) + cn \\ &= O(n^{\log_2 4}) \\ &= O(n^2) \end{aligned}$$

We did not beat the grade-school algorithm. :(

Better Divide and Conquer

Same starting point:

$$xy = 10^n x_1 y_1 + 10^{n/2}(x_1 y_0 + x_0 y_1) + x_0 y_0$$

Trick: use three multiplications to compute the following:

$$\begin{aligned} A &= (x_1 + x_0)(y_1 + y_0) = x_1 y_1 + x_1 y_0 + x_0 y_1 + x_0 y_0 \\ B &= x_1 y_1 \\ C &= x_0 y_0 \end{aligned}$$

Then

$$xy = 10^n B + 10^{n/2}(A - B - C) + C$$

Total: three multiplications of $n/2$-digit integers, six additions

Better Divide and Conquer

Total: three multiplications of $n/2$-digit integers, six additions of at most n-digit integers

$$\begin{aligned} T(n) &\leq 3T\left(\frac{n}{2}\right) + cn \\ &= O(n^{\log_2 3}) \\ &\approx O(n^{1.59}) \end{aligned}$$

We beat long multiplication!

Can be done even faster (split x and y into k parts instead of two)

Master Theorem

Consider the general recurrence:

$$T(n) \leq aT\left(\frac{n}{b}\right) + cn^d$$

Clicker. How much work is done outside recursion at the root of the recursion tree?

A. $\Theta(n^d)$

B. $\Theta(n^a)$

C. $\Theta(n^b)$

D. None of the above

Master Theorem

Consider the general recurrence:

$$T(n) \leq aT\left(\frac{n}{b}\right) + cn^d$$

Clicker. How many leaves are in the recursion tree?

A. $\Theta(a^{\log_b n})$

B. $\Theta(n^{\log_b a})$

C. $\Theta(b^{\log_a n})$

D. Both a and b

Master Theorem

Consider the general recurrence:

$$T(n) \leq aT\left(\frac{n}{b}\right) + cn^d$$

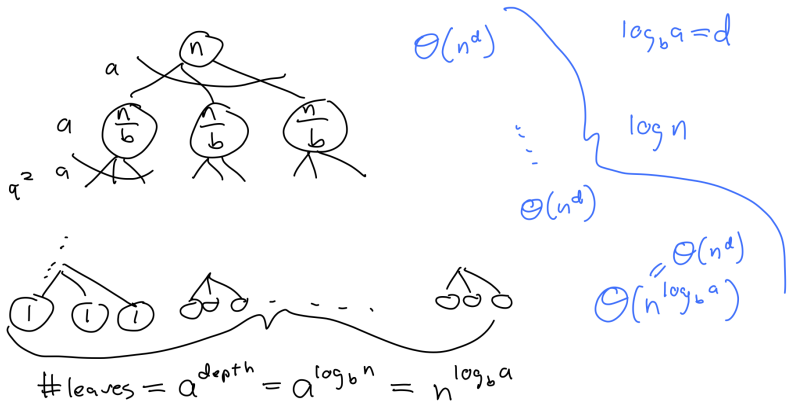
This solves to:

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } \log_b a < d \\ \Theta(n^d \log n) & \text{if } \log_b a = d \\ \Theta(n^{\log_b a}) & \text{if } \log_b a > d \end{cases}$$

Intuition: work at each level of the recursion tree is (1) decreasing exponentially, (2) staying the same, (3) increasing exponentially.

Pick the largest of work at root vs. work at leaves; multiyply by $\log n$ if same.

Master Theorem Intuition Review



Clicker

Master Theorem:

$$T(n) \leq aT\left(\frac{n}{b}\right) + cn^d$$

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } \log_b a < d \\ \Theta(n^d \log n) & \text{if } \log_b a = d \\ \Theta(n^{\log_b a}) & \text{if } \log_b a > d \end{cases}$$

Suppose $T(n) = 9T(n/3) + n^d$. What is the largest value for d below such that $T(n) = \Theta(n^2)$?

A. $d = 1$
B. $d = 1.5$
C. $d = 2$
D. $d = 3$