CS3000: Algorithms & Data
Paul Hand

Lecture 6:

- Master Theorem for Recurrences
- Integer Multiplication
- Divide and Conquer Example – Similar to HW

Jan 28, 2019

Solving Recurrences:
“The Master Theorem”

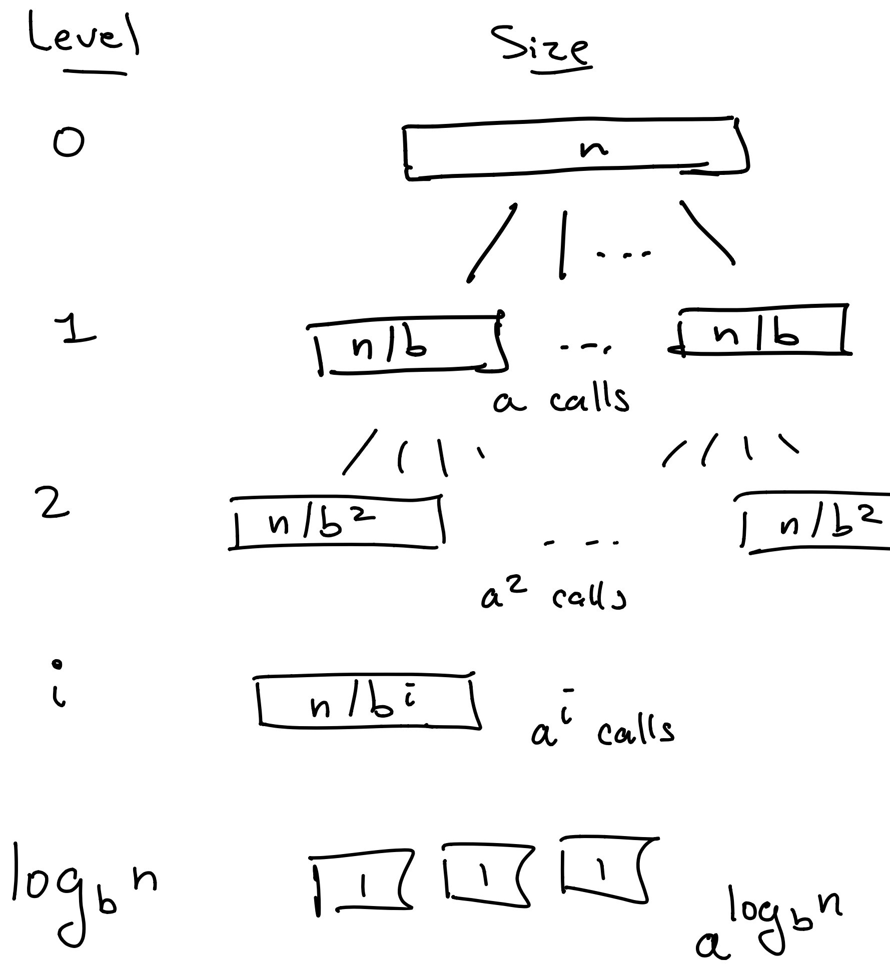
The "Master Theorem"

- Generic divide-and-conquer algorithm:
  - Split into $\boldsymbol{a}$ pieces of size $\frac{\boldsymbol{n}}{\boldsymbol{b}}$ and merge in time $O(n^{\boldsymbol{d}})$
- Recipe for recurrences of the form:
  - $T(n) = \boldsymbol{a} \cdot T(n/\boldsymbol{b}) + Cn^{\boldsymbol{d}}$
- Three cases:
  - $\left(\frac{\boldsymbol{a}}{\boldsymbol{b}^{\boldsymbol{d}}}\right) > 1: \; T(n) = \Theta\left(n^{\log_{\boldsymbol{b}} \boldsymbol{a}}\right)$
  - $\left(\frac{\boldsymbol{a}}{\boldsymbol{b}^{\boldsymbol{d}}}\right) = 1: \; T(n) = \Theta\left(n^{\boldsymbol{d}} \log n\right)$
  - $\left(\frac{\boldsymbol{a}}{\boldsymbol{b}^{\boldsymbol{d}}}\right) < 1: \; T(n) = \Theta\left(n^{\boldsymbol{d}}\right)$

Recursion Tree

- $T(n) = \boldsymbol{a}T(n/\boldsymbol{b}) + n^{\boldsymbol{d}}$

$$S = \sum_{i=0}^{\ell} r^i = \frac{r^{\ell+1} - 1}{r - 1}$$



Work

n^d

$a \times \left(\frac{n}{b}\right)^d = \left(\frac{a}{b^d}\right) \cdot n^d$

$a^2 \times \left(\frac{n}{b^2}\right)^d = \left(\frac{a}{b^d}\right)^2 \cdot n^d$

$\left(\frac{a}{b^d}\right)^i \cdot n^d$

$a^{\log_b n} = \left(\frac{a}{b^d}\right)^{\log_b n} \cdot n^d$

Recursion Tree

- $T(n) = \boldsymbol{a}T(n/\boldsymbol{b}) + n^{\boldsymbol{d}}$
- $\left(\frac{\boldsymbol{a}}{\boldsymbol{b}^{\boldsymbol{d}}}\right) > 1$

$$\sum_{i=0}^{\log_b n} \left(\frac{a}{b^d}\right)^i n^d = n^d \frac{\left(\frac{a}{b^d}\right)^{\log_b n + 1} - 1}{\left(\frac{a}{b^d}\right) - 1} = \Theta\left(\left(\frac{a}{b^d}\right)^{\log_b n} \cdot n^d\right)$$

apply geometric summation

$$= \Theta\left(n^{\log_b a - d} \cdot n^d\right) = \Theta\left(n^{\log_b a}\right)$$

Use $\left(\frac{a}{b^d}\right)^{\log_b n} = n^{\log_b \left(\frac{a}{b^d}\right)}$

$= n^{\log_b a - d} =$

$$T(n) = \Theta\left(n^{\log_b a}\right)$$

most expensive level of recursion tree is the last

Fact:

$x^{\log_b n} = n^{\log_b x}$

Why?

$x^{\log_b n} = b^{\log_b (x^{\log_b n})} = b^{\log_b n \log_b x} = (b^{\log_b n})^{\log_b x} = n^{\log_b x}$

Recursion Tree

- $T(n) = \boldsymbol{a}T(n/\boldsymbol{b}) + n^{\boldsymbol{d}}$
- $\left(\frac{\boldsymbol{a}}{\boldsymbol{b}^{d}}\right) = 1$

$$\sum_{i=0}^{\log_b n} \left(\frac{a}{b^d}\right)^i n^d = \sum_{i=0}^{\log_b n} n^d = \Theta(\log_b n \cdot n^d)$$

$$= \Theta(n^d \log_b n)$$

all levels of tree are equally expensive

Recursion Tree

- $T(n) = \boldsymbol{a}T(n/\boldsymbol{b}) + n^{\boldsymbol{d}}$
- $\left(\frac{\boldsymbol{a}}{\boldsymbol{b}^{\boldsymbol{d}}}\right) < 1$

$$\sum_{i=0}^{\log_b n} \left(\frac{a}{b^d}\right)^i n^d = n^d \frac{\left(\frac{a}{b^d}\right)^{\log_b n + 1} - 1}{\left(\frac{a}{b^d}\right) - 1} = \Theta(n^d)$$

apply geometric summation

because $\left(\frac{a}{b^d}\right) < 1$,

$\left(\frac{a}{b^d}\right)^{\log_b n} \to 0$ as $n \to \infty$

expensive part is at the 0^{th} level of recursion tree

Practice

- Use the Master Theorem to Solve:
  - $T(n) = 16 \cdot T\left(\frac{n}{4}\right) + n^2$
  - $T(n) = 21 \cdot T\left(\frac{n}{5}\right) + n^2$
  - $T(n) = 2 \cdot T\left(\frac{n}{2}\right) + 1$
  - $T(n) = 1 \cdot T\left(\frac{n}{2}\right) + 1$

- Recipe for recurrences of the form:
  - $T(n) = \boldsymbol{a} \cdot T(n/\boldsymbol{b}) + Cn^{\boldsymbol{d}}$
- Three cases:
  - $\left(\frac{\boldsymbol{a}}{\boldsymbol{b^d}}\right) > 1 : \; T(n) = \Theta\left(n^{\log_b \boldsymbol{a}}\right)$
  - $\left(\frac{\boldsymbol{a}}{\boldsymbol{b^d}}\right) = 1 : \; T(n) = \Theta\left(n^{\boldsymbol{d}} \log n\right)$
  - $\left(\frac{\boldsymbol{a}}{\boldsymbol{b^d}}\right) < 1 : \; T(n) = \Theta\left(n^{\boldsymbol{d}}\right)$

Integer Multiplication: Karatsuba's Algorithm

Activity: Arithmetic Algorithms

What is the time complexity of the grade school algorithm for the following?

- Given n-digit numbers x, y output $x + y$

	1	2	3	4
+	1	1	2	2
=				

- Given n-digit numbers x, y output $x \cdot y$

	1	2	3	4
x	1	1	2	2
=				

Divide and Conquer Multiplication

	1	2	3	4
x	1	1	2	2

$$\boldsymbol{x} = 10^2 \cdot 12 + 34$$
$$\boldsymbol{y} = 10^2 \cdot 11 + 22$$

	$\boldsymbol{a}$	$\boldsymbol{b}$
x	$\boldsymbol{c}$	$\boldsymbol{d}$

$$\boldsymbol{x} = 10^{n/2}\boldsymbol{a} + \boldsymbol{b}$$
$$\boldsymbol{y} = 10^{n/2}\boldsymbol{c} + \boldsymbol{d}$$

Divide and Conquer Multiplication

	$\boldsymbol{a}$	$\boldsymbol{b}$
x	$\boldsymbol{c}$	$\boldsymbol{d}$

$$\boldsymbol{x} = 10^{n/2}\boldsymbol{a} + \boldsymbol{b}$$

$$\boldsymbol{y} = 10^{n/2}\boldsymbol{c} + \boldsymbol{d}$$

$$\boldsymbol{x} \cdot \boldsymbol{y} = (10^{n/2}\boldsymbol{a} + \boldsymbol{b})(10^{n/2}\boldsymbol{c} + \boldsymbol{d})$$

$$= 10^{n}\boldsymbol{ac} + 10^{n/2}(\boldsymbol{ad} + \boldsymbol{bc}) + \boldsymbol{bd}$$

- Four $n/2$-digit mults, three n-digit adds
  - Multiplying by 10^n is "free" because it's a shift
- Recurrence: $T(n) = 4T\left(\frac{n}{2}\right) + 3n$
- Total cost of algorithm

Master Theorem for Recurrences

- Recipe for recurrences of the form:
  - $T(n) = \boldsymbol{a} \cdot T(n/\boldsymbol{b}) + Cn^{\boldsymbol{d}}$
- Three cases:
  - $\left(\frac{\boldsymbol{a}}{\boldsymbol{b}^{d}}\right) > 1 : T(n) = \Theta\left(n^{\log_{\boldsymbol{b}} \boldsymbol{a}}\right)$
  - $\left(\frac{\boldsymbol{a}}{\boldsymbol{b}^{d}}\right) = 1 : T(n) = \Theta\left(n^{\boldsymbol{d}} \log n\right)$
  - $\left(\frac{\boldsymbol{a}}{\boldsymbol{b}^{d}}\right) < 1 : T(n) = \Theta\left(n^{\boldsymbol{d}}\right)$

Karatsuba's Algorithm

	a	b
x	c	d

$$\boldsymbol{x} = 10^{n/2}\boldsymbol{a} + \boldsymbol{b}$$

$$\boldsymbol{y} = 10^{n/2}\boldsymbol{c} + \boldsymbol{d}$$

$$x \cdot y = 10^{n}\boldsymbol{ac} + 10^{n/2}(\boldsymbol{ad} + \boldsymbol{bc}) + \boldsymbol{bd}$$

- Key Identity
  - $(\boldsymbol{b} - \boldsymbol{a})(\boldsymbol{c} - \boldsymbol{d}) = \boldsymbol{ad} + \boldsymbol{bc} - \boldsymbol{ac} - \boldsymbol{bd}$

- Only three $n/2$-digit mults (plus some adds)!

Karatsuba's Algorithm

```
Karatsuba(x,y,n):
  If (n = 1): Return x·y                    // Base Case

  Let m ← ⌈n/2⌉                              // Split
  Write x = 10^m a + b,  y = 10^m c + d

  Let e ← Karatsuba(a,c,m)                  // Recurse
      f ← Karatsuba(b,d,m)
      g ← Karatsuba(b-a,c-d,m)

  Return 10^{2m} e + 10^m (e + f + g) + f   // Merge
```

- **Claim:** The algorithm **`Karatsuba`** is correct

Running Time of Karatsuba

```
Karatsuba(x,y,n):
  If (n = 1): Return x·y

  Let m ← ⌈n/2⌉
  Write x = 10^m a + b,  y = 10^m c + d

  Let e ← Karatsuba(a,c,m)
      f ← Karatsuba(b,d,m)
      g ← Karatsuba(b-a,c-d,m)

  Return 10^{2m} e + 10^m (e + f + g) + f
```

Master Theorem for Recurrences

- Recipe for recurrences of the form:
  - $T(n) = \boldsymbol{a} \cdot T(n/\boldsymbol{b}) + Cn^{\boldsymbol{d}}$
- Three cases:
  - $\left(\frac{a}{b^d}\right) > 1 : T(n) = \Theta\left(n^{\log_b a}\right)$
  - $\left(\frac{a}{b^d}\right) = 1 : T(n) = \Theta\left(n^d \log n\right)$
  - $\left(\frac{a}{b^d}\right) < 1 : T(n) = \Theta\left(n^d\right)$

Karatsuba Wrapup

- Multiply n digit numbers in $O(n^{1.59})$ time
  - Improves over naïve $O(n^2)$ time algorithm
  - **Fast Fourier Transform:** multiply in $\approx O(n \log n)$ time
- Divide-and-conquer approach
  - Uses a clever algebraic trick to split
  - **Key Fact:** adding is faster than multiplying
- Prove correctness via induction
- Analyze running time via recursion tree
  - $T(n) = 3T(n/2) + Cn$

Practice Problem:
Maximum Sum Subarray Problem

Maximum Sum Subarray Problem

- Input: Array A[1:n] of integers
- Problem: Find a subarray A[i:j] with the largest possible sum
- Example: A = [3, -4, 5, -2, -2, 6, -3, 5, -3, 2]

  Maximum sum = 9

- Task: Devise a divide and conquer algorithm to solve this problem. Consider an algorithm that divides A into two halves.