CS3000: Algorithms & Data
Paul Hand

Lecture 6:

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

Jan 28, 2019

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

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 $O\left(n^{\boldsymbol{d}}\right)$
- Recipe for recurrences of the form:
  - $T(n) = \boldsymbol{a} \cdot T(n/\boldsymbol{b}) + Cn^{\boldsymbol{d}}$

not run in parallel

- Three cases:
  - $\left(\frac{\boldsymbol{a}}{\boldsymbol{b}^{\boldsymbol{d}}}\right) > 1 : T(n) = \Theta\left(n^{\log_{\boldsymbol{b}} \boldsymbol{a}}\right)$ cost dominated by base cases (pay polynomial factor)
  - $\left(\frac{\boldsymbol{a}}{\boldsymbol{b}^{\boldsymbol{d}}}\right) = 1 : T(n) = \Theta\left(n^{\boldsymbol{d}} \log n\right)$ only pay log factor
  - $\left(\frac{\boldsymbol{a}}{\boldsymbol{b}^{\boldsymbol{d}}}\right) < 1 : T(n) = \Theta\left(n^{\boldsymbol{d}}\right)$ cost dominated by merge

Recursion Tree

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

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

Level	Size	Work spent merging
0	n	n^d
1	n/b ... n/b a calls	$a \times \left(\frac{n}{b}\right)^d = \left(\frac{a}{b^d}\right) \cdot n^d$
2	n/b^2 ... n/b^2 a^2 calls	$a^2 \times \left(\frac{n}{b^2}\right)^d = \left(\frac{a}{b^d}\right)^2 \cdot n^d$
i	n/b^i a^i calls	$\left(\frac{a}{b^d}\right)^i \cdot n^d$
$\log_b n$	1 1 1 $a^{\log_b n}$	$a^{\log_b n} = \left(\frac{a}{b^d}\right)^{\log_b n} \cdot n^d$

Total work

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

(

geometric series

Recursion Tree

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

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

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

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

apply geometric summation

most expensive level of recursion tree is the last

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

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

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

Fact:

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

Why?

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

Recursion Tree

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

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

$$= \boxed{\Theta(n^d \log_b n)}$$

all levels
of tree are
equally expensive

Recursion Tree

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

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

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
=	2	3	5	6

$\Theta(n)$

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

				1	2	3	4
			x	1	1	2	2
			=				
+				2	4	6	8
+			2	4	6	8	0
+		1	2	3	4	0	0
	1	2	3	4	0	0	0

n additions
of #s w/ at most
2n digits

$\Theta(n^2)$

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	$\boldsymbol{c}$	$\boldsymbol{d}$

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

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

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

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

too many. Notice mergesort had only 2

- Four $n/2$-digit mults, three n-digit adds
  - Multiplying by 10^n is "free" because it's a shift
- Recurrence: $T(n) = 4T\left(\frac{n}{2}\right) + 3n^{1}$

a b d

- 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:
  - $\left(\frac{\boldsymbol{a}}{\boldsymbol{b}^{\boldsymbol{d}}}\right) > 1: T(n) = \Theta(n^{\log_{\boldsymbol{b}} \boldsymbol{a}})$
  - $\left(\frac{\boldsymbol{a}}{\boldsymbol{b}^{\boldsymbol{d}}}\right) = 1: T(n) = \Theta(n^{\boldsymbol{d}} \log n)$
  - $\left(\frac{\boldsymbol{a}}{\boldsymbol{b}^{\boldsymbol{d}}}\right) < 1: T(n) = \Theta(n^{\boldsymbol{d}})$

$$T(n) = \Theta(n^{\log_2 4})$$

$$= \Theta(n^2)$$

$$\frac{a}{b^d} = \frac{4}{2} = 2$$

Is this good or bad? Why? BAD

Why did mergesort beat insertion sort but this did not beat naive multiplication?

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)!

ac
bd
$(b-a)(c-d)$

Additions we do are
$b-a$
$c-d$
$(b-a)(c-d) + ac + bd$

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
```

computes $a \cdot c$

compute bd

computes $(b-a)(c-d)$

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

Proof by induction

① Base case. Trivial. $n=1$

② General case.

Assume $H(1), H(2), \dots H(n-1)$ are true

We prove $H(n)$:

By ind Hyp, $e = ac$
$f = bd$
$g = (b-a)(c-d)$

Returns

$10^{2m} ac + 10^m (e+f+g) + bd$

$x \cdot y \quad = 10^{2m} ac + 10^m (\cancel{ac} + \cancel{bd} + bc - \cancel{bd} - \cancel{ac} + ad) + bd$

Running Time of Karatsuba

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

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

  Let e ← Karatsuba(a,c,m)                             T(n/2)
      f ← Karatsuba(b,d,m)                             T(n/2)
      g ← Karatsuba(b-a,c-d,m)                         T(n/2)

  Return 10^(2m) e + 10^m (e + f + g) + f              O(n)
```

$T(n) = \underset{a}{3}\,T\left(\frac{n}{\underset{b}{2}}\right) + cn^{1}$ — d

Master Theorem for Recurrences

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

$a = 3$

$b = 2$

$d = 1$

$\frac{a}{b^d} = \frac{3}{2}$

$T(n) = \Theta\left(n^{\log_2 3}\right)$

$= \Theta\left(n^{\log_2 3}\right)$

$\simeq \Theta\left(n^{1.59}\right)$

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 — ~~or [illegible]~~
- Analyze running time via recursion tree & Master Thm
  - $T(n) = 3T(n/2) + Cn$

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]

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

Discuss w/ neighbors: What is a reasonable place to start attacking this problem.

Find 3 things

- Finding an approach you need to beat
- Study a smaller instance and try to solve it in your head
- Consider special cases (Eg pos/neg entries)
- Do you understand problem/vocabulary

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]

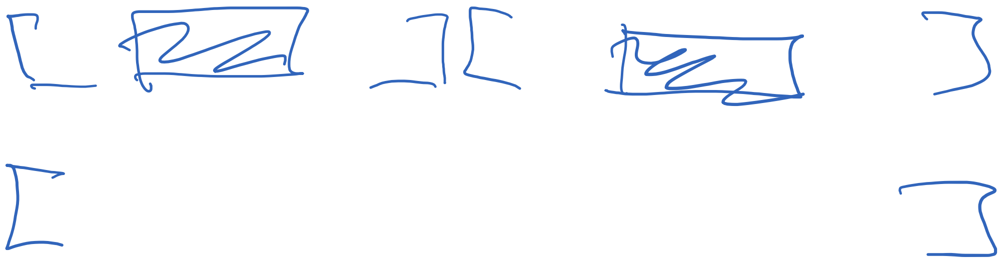
max sum subarray ??

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

What do I need to show to get divide + conquer to work?

Base case

If you've solved smaller problems, how do you combine them efficiently



difficulty: combining is hard bec the max sum subarray may not live in either half alone