Week 7 - Wednesday

COMP 4500

Last time

- What did we talk about last time?
- Closest pair of points

Questions?

Logical warmup

- A man has two 10 gallon jars
- The first contains 6 gallons of wine and the second contains 6 gallons of water
- He poured 3 gallons of wine into the water jar and stirred
- Then he poured 3 gallons of the mixture in the water jar into the wine jar and stirred
- Then he poured 3 gallons of the mixture in the wine jar into the water jar and stirred
- He continued the process until both jars had the same concentration of wine
- How many pouring operations did he do?

Three-Sentence Summary of Integer Multiplication

Integer Multiplication

Multiplication by hand

- How long does it take to do multiplication by hand when we count single digit addition or multiplication as an operation?

```
  123
x 456
-----
  738
 615
492
-----
56088
```

- Let's assume that the length of the numbers is ***n*** digits
- (***n*** multiplications + ***n*** carries) x ***n*** digits + (***n*** + 1 digits) x ***n*** additions
- Running time: $O(n^2)$

Can we do better?

- Imagine that we're in base 2, because it keeps things simple
- I want to find the product $x \cdot y$
- We can break x (and similarly y) into high-order part x_1 and low-order part x_0 such that $x = x_1 \cdot 2^{\frac{n}{2}} + x_0$
- $xy = \left(x_1 \cdot 2^{\frac{n}{2}} + x_0\right)\left(y_1 \cdot 2^{\frac{n}{2}} + y_0\right)$
- $= x_1 y_1 \cdot 2^n + (x_1 y_0 + x_0 y_1) \cdot 2^{\frac{n}{2}} + x_0 y_0$

Did that help things?

- Not really!
- We turned the multiplication of two ***n***-bit numbers into the multiplication (and then addition) of four ***n***/2-bit numbers
- $T(n) \leq 4T\left(\frac{n}{2}\right) + cn$
- Which is $O\left(n^{\log_2 4}\right) = O(n^2)$, which is … the same

We need a trick

- We want $x_1y_1 \cdot 2^n + (x_1y_0 + x_0y_1) \cdot 2^{\frac{n}{2}} + x_0y_0$
- What if we compute
  - $a = (x_1 + x_0) \cdot (y_1 + y_0) = x_1y_1 + x_0y_1 + x_1y_0 + x_0y_0$
  - $b = x_1y_1$
  - $c = x_0y_0$
- Then, $b \cdot 2^n + (a - b - c) \cdot 2^{\frac{n}{2}} + c =$
- $x_1y_1 \cdot 2^n + (x_1y_0 + x_0y_1) \cdot 2^{\frac{n}{2}} + x_0y_0$

Running time

- We do two additions before the multiplies: O(***n***)
- We do three recursive multiplies of ***n***/2-bit numbers
- We do two additions and two subtractions after the multiplies: O(***n***)
- $T(n) \leq 3T\left(\frac{n}{2}\right) + cn$
- Which is $O\left(n^{\log_2 3}\right) \approx O(n^{1.59})$, which is better!

Note about multiplication

- Integer multiplication can be viewed as:
  - Polynomial multiplication
  - Vector convolution
- We will not cover section 5.6, but it describes the Fast Fourier Transform (FFT)
- The FFT can perform polynomial multiplication in $O(n \log n)$ time: even better than $O(n^{1.59})$
- However, the FFT has a big constant
- Using the FFT to multiply two integers only makes sense when the numbers are **large**, e.g., millions of digits

Master Theorem

Master Theorem

- Has a great name …
- Allows us to determine the Big Theta running time of many recursive functions that would otherwise take more effort to determine

Basic form the recurrence relation must take

$$T(n) = aT\left(\frac{n}{b}\right) + f(n),$$
$$\text{where } a \geq 1 \text{ and } b > 1$$

- ***a*** is the number of recursive calls made
- ***b*** is how much the quantity of data is divided by each recursive call
- ***f*(*n*)** is the non-recursive work done at each step

Case 1

- If $f(n)$ is $O\left(n^{\log_b(a)-\epsilon}\right)$ for some constant $\epsilon > 0$, then

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

Case 2

- If $f(n)$ is $\Theta\left(n^{\log_b(a)} \log^k n\right)$ for some constant $k \geq 0$, then

$$T(n) \text{ is } \Theta\left(n^{\log_b(a)} \log^{k+1} n\right)$$

Case 3

- If $f(n)$ is $\Omega\left(n^{\log_b(a)+\epsilon}\right)$ for some constant $\epsilon > 0$, and if

$$af\left(\frac{n}{b}\right) \leq cf(n)$$

for some constant $c < 1$ and sufficiently large n, then

$$T(n) \text{ is } \Theta(f(n))$$

Stupid Sort

Stupid Sort algorithm (recursive)

- Base case: List has size less than 3
  - Swap out of order items if necessary
- Recursive case:
  - Recursively sort the first 2/3 of the list
  - Recursively sort the second 2/3 of the list
  - Recursively sort the first 2/3 of the list **again**

Stupid Sort

- How long does Stupid Sort take?
- We need to know $\log_b a$
- $a = 3$
- $b = 3/2 = 1.5$
- Because I'm a nice guy, I'll tell you that the $\log_{1.5} 3$ is about 2.7

Binary Search

- We know that binary search takes O(log ***n***) time
- Can we use the Master Theorem to check that?

Practicing the Master Theorem

- One way to practice is to try to create a problems that different cases of the Master Theorem apply to
- Give a recurrence relation that uses Case 1
- Give a recurrence relation that uses Case 2
- Give a recurrence relation that uses Case 3

Quiz

Upcoming

Next time…

- More Master Theorem examples
- Solved Exercises

Reminders

- **Work on Assignment 4**
  - **Due next Monday**
- Exam 2 is next Wednesday
  - Review Chapters 4 and 5