

CSE 202 - Algorithms

Euclidean Algorithm

Divide and Conquer

4/3/2003

CSE 202 - More Math



Euclidean Algorithm

GCD stands for "greatest common divisor".

E.g. $GCD(10, 25) = 5$. If $GCD(A, B) = 1$, we say A and B are relatively prime.

Note that $GCD(N, 0) = N$. (At least for $N > 0$.)

Thm: If $A + kB = C$ and $B \neq 0$, then $GCD(A, B) = GCD(B, C)$.

Proof. Let $g = GCD(A, B)$ and $h = GCD(B, C)$.

Since g divides A and B , it must divide $A + kB = C$.

Since g is a common divisor of B and C , it must be \leq their *greatest* common divisor, i.e. $g \leq h$.

Reversing the roles of A and C , we conclude that $h \leq g$.

Thus, $g = h$. QED

The Euclidean Algorithm finds $GCD(A, B)$.

Given $A > B > 0$, set $C = A \bmod B$ (so $C = A - kB$ for some k and $C < B$).

This reduces the problem of finding $GCD(A, B)$ to the smaller problem, finding $GCD(B, C)$. Eventually, we'll get to $GCD(X, 0) = X$.

Euclidean Algorithm

Example: Find GCD of 38 and 10.

$$38 \bmod 10 = 8$$

$$10 \bmod 8 = 2$$

$$8 \bmod 2 = 0$$

$$\text{GCD}(2,0) = 2$$

Extended Euclidean Algorithm

Example: Find GCD of 38 and 10.

$$38 \bmod 10 = 8 \quad \text{Lets you write 8 in terms of 38 and 10} \quad 8 = 38 - 3 \times 10$$

$$10 \bmod 8 = 2 \quad \text{Lets you write 2 in terms of 10 and 8.} \quad 2 = 10 - 1 \times 8$$

$$8 \bmod 2 = 0 \quad \text{Change 8's to 10's and 38's} \quad = 10 - 1 \times (38 - 3 \times 10)$$

$$\text{GCD}(2,0) = 2 \quad \text{Voila!} \quad 2 = 4 \times 10 - 1 \times 38$$

Can write "2" as linear combination of 10 and 38.

Euclidean Magic

Linear equations over integers

- Solve Diophantine Equations:
 - E.g. " $a \times 38 + b \times 10 = 6$ "
 - First write $GCD(38,10)$ using 10 & 38 ($2 = 4 \times 10 - 1 \times 38$),
then multiply by $6/GCD = 3$ ($6 = 12 \times 10 - 3 \times 38$).
 - Rational approximations:
 - E.g. $31416 = 3 \times 10000 + 1416$
 $10000 = 7 \times 1416 + 88$
 - Let's pretend $88 \approx 0$. We then have $1416 \approx 10000/7$
Thus, $31416 \approx 3 \times 10000 + 10000/7 = 10000 \times (3 + 1/7)$
Thus, $3.1416 \approx 3 + 1/7 = 22/7$ (= 3.142857...)
- We can get all "close" approximations this way.

5

CSE 202 - More Math

Morals

- Once you have solved something (e.g. GCD), see if you can solve other problems (e.g. linear integer equations, approximations, ...)
- I've shown some key ideas without writing out the algorithms - they are actually simple.
"Continued fractions" does this all nicely!
- If this is your idea of fun, try cryptography.

6

CSE 202 - More Math

Divide & Conquer

- Basic Idea
 - Break big problem into subproblems
 - Solve each subproblem
 - Directly if it's very small
 - Otherwise recursively break it up, etc.
 - Combine these solutions to solve big problem

7

CSE 202 - More Math

Faster Multiplication ??

Standard method for multiplying long numbers:

$$(1000a+b) \times (1000c+d) = 1,000,000 ac \\ + 1000 (ad + bc) \\ + bd$$

4 multiplies:
ac, ad, bc, bd

Clever trick:

$$(1000a+b) \times (1000c+d) = 1,000,000 ac \\ + 1000 ((a+b)(c+d) - ac - bd) \\ + bd$$

3 multiplies:
ac, (a+b)(c+d), bd

One length-k multiply = 3 length-k/2 multiplies and a bunch of additions and shifting.

On computer, might use 2^{16} in place of 1000

Fine print: If "a+b" has 17 bits, drop the top bit, but add extra c+d in the right place. Handle overflow for c+d similarly.

8

CSE 202 - More Math

Faster Multiplication !!

To magnify savings, use recursion (Divide & Conquer).

Let $T(n)$ = time to multiply two n -chunk numbers.

For n even, we have $T(n) < 3T(n/2) + c n$

This is called a recurrence equation

Make c big enough so that cn time lets you do all the adds, subtracts, shifts, handling carry bits, and dealing with +/-.

Reasonable, since operations are on n -chunk numbers.

Recursively divide until we have 1-chunk numbers

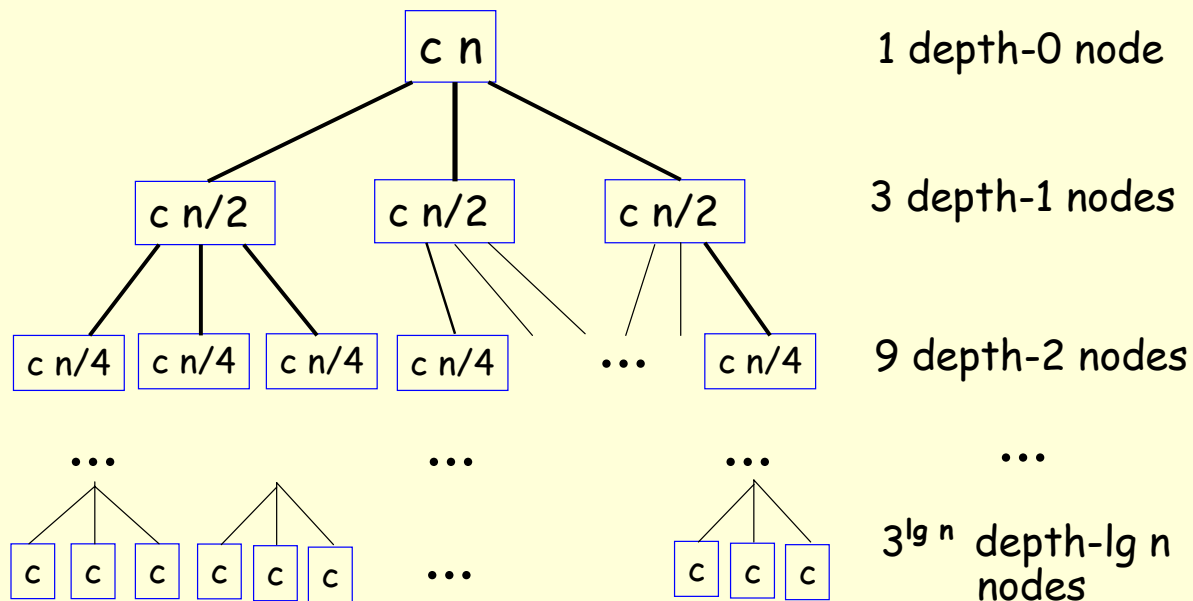
Need $\lg n$ "divide" steps.

Also make c big enough so $T(1) < c$.

9

CSE 202 - More Math

Recursion Tree



10

CSE 202 - More Math

From the picture, we know ...

$$T(n) < cn (1 + 3(1/2) + 9(1/4) + \dots + 3^{\lg n} (1/ 2^{\lg n})) \\ < cn (1 + 3/2 + (3/2)^2 + \dots + (3/2)^{\lg n}).$$

Lemma 1: For $a \neq 1$, $a^k + a^{k-1} + \dots + 1 = (a^{k+1} - 1) / (a-1)$

Proof: $(a^k + a^{k-1} + \dots + 1) (a-1) = a^{k+1} - 1$

It's obvious! (Multiply it out - middle terms cancel.)

By lemma 1, $T(n) < cn ((3/2)^{\lg n + 1} - 1) / ((3/2)-1)$

$$< cn ((3/2)^{\lg n} (3/2) - 0) / (1/2) \\ = cn (3/2)^{\lg n} (3/2) / (1/2) \\ = 3cn (3/2)^{\lg n} .$$

And finally ...

Lemma 2: $a^{\lg b} = b^{\lg a}$

Proof: $a^{\lg b} = (2^{\lg a})^{\lg b} = 2^{\lg a \lg b} = (2^{\lg b})^{\lg a} = b^{\lg a}$

Applying Lemma 2, we have

$$T(n) < 3cn (3/2)^{\lg n} = 3cn (n^{\lg(3/2)}) = 3cn^{1+\lg(3/2)} .$$

Thus, $T(n) \in O(n^{1+\lg(3/2)}) = O(n^{\lg 3}) = O(n^{1.58\dots})$.

Key points

- Divide & Conquer can magnify a small advantage.
- Recursion tree gives a method of solving recurrence equations.
 - We'll look at other methods next.
- Lemmas 1 and 2 are useful.
 - Make sure you're comfortable with the math.