MA/CSSE 473 Summer 2010 Additional comments on 200910 PowerPoint slides

Day 2

Slide 2

Some parts of the definition of "Algorithm":

Sequence of instructions

For solving a problem

Unambiguous (including order)

Can depend on input

Terminates in a finite amount of time

Day 3

Slide 4

"until we dig deeper" refers to the cost of doing the additions themselves.

Slide 8

Several reasons

Floating point arithmetic and especially square roots are expensive to compute

Also, they do not give exact values.

Each F_n computed this way is an integer, but if we use floating-point arithmetic to do it, we will not get exact values.

Slide 10 Whole thing is $O(n^2)$

Slide 11

It is correct, because the same numbers get added as in the more "traditional" algorithm.

If we are multiplying n-bit numbers, the number of times through the loop is n.

And each time through the loop requires at most n steps.

And still we have to add up to n numbers of 2n bits each.

So it is O(n^2)

Slide 15

T(n) = 4T(n/2) +O(n). Solution?

Solution is case 3 of Master Theorem.

log[2] 4 is 2, so it is n^2.

Day 4

Slide 8

Prove: If $f(n)\in O(h(n))$ and $g(n)\in O(h(n))$, then $f(n)+g(n)\in O(h(n))$

If f(n))$\in O(h(n))$, then there are constants c_1 and n_1 such that $\forall n > n_1$, $f(n) \leq c_1\, h(n)$.

If g(n))$\in O(h(n))$, then there are constants c_2 and n_2 such that $\forall n > n_2$, $g(n) \leq c_2\, h(n)$.

Now let $n_0 = \max(n_1, n_2)$.

Then $\forall n > n_0$, $f(n)+g(n) \leq c_1\, h(n) + \leq c_2\, h(n). = (c_1+ c_2)\, h(n)$.

Slide 20

T(n) = 3T(n/2) +O(n).

Solution is case 3 of Master Theorem.

It is $\Theta(n\text{^}(\log_2 3))$, which is approximately n^1.59

You don't really want to recur down to the 1-bit level. Stop at the machine's integer size (32 or 64), since that many bits can already be multiplied efficiently.

Because there is some overhead with this method, it doesn't actually beat the straightforward recursive approach unless x and y are several hundred bits long.

Day 5

Slide 3

C < 1. The infinite series converges to 1/(1-c), so Theta(1)

C = 1 f(n) = n, so it is Theta(n)

C > 1 f(n) = (c^(n+1) – 1) / (c – 1). The limit of f(n) divided by c^n as n→infinity is c/(c-1), so f(n) is Theta(c^n).

Slide 8

Note on recursive: The length of F(n) is .694n, which is O(n).

Note on last one: The # of bits in the numbers in the matrix at most doubles with each matrix multiplication.

Thus we get AN UPPER BOUND OF M(1) + M(2) + M(4) + M(8) + … + M(F(n))

If a = 2, we get (Maple notation):

> **sum((2^i)^2), i=0..log[2](n));**

> **simplify(sum((2^i)^log[2](3), i=0..log[2](n)));**

Slide 14

To do better, we can use an algorithm like our previous recursive exponentiation algorithm

Day 6

Slide 7

Stamps: Proof is by strong induction.

Five base cases: 24 = 7*2+5*2, 25=5*5, 26=7*3 + 5, 27=7 + 4*5, 28 = 4*7

Induction step: Let k be any number that is > 28. Show that if for all j<k, we can achieve j cents using 5 and 7 cent stamps, then we can also achieve k cents using 5 and 7 cent stamps.

In particular (by the induction assumption) j=k-5 can be achieved with 5 and 7-cent stamps. Add one more 5-cent stamp to get k cents.

Day 7

Slide 5

Base case. P(1), 3 people. Suppose A and B are the closest pair, and C is the third person. Since all of the distances are different, the distances between A and C, and between B and C are strictly larger than the distance between A and B. Thus A and B throw pies at each other, and C is the survivor.

Induction step. Assume that P(k) is true (in every pie fight with 2k+1 people, there is a survivor). Show that P(k+1) is true (In every pie fight with 2K+3 people, there is a survivor).

Let A and B be the closest pair in this group of 2k+3 people. They throw pies at each other.

If someone else throws a pie at one of them, then for the remaining 2k+1 people, there are only 2k pies to hit them, so somene survives.

If no one else throws a pie at A or B, then the other people comprise a 2k+1 pie fight, which has a survivor, by the induction assumption.

Slide 8

Place the first tromino so that it covers 3 of the 4 center squares, the three that are not in the lower-right quadrant. Now all of the four quadrants are deficient rectangles of size 2^k by 2^k.

Slide 14

Does it work? It's clear that it works when b is 0.

Is it clear that b will eventually be 0? Yes.

SO by Euclid's rule, it works.

Efficiency? Each time, (a, b) gets replaced with (b, a mod b). How big of a reduction is that?

Day 8

Slide 4

Second case of lemma proof: If b > a/2, then a – b < a – a/2 = a/2.

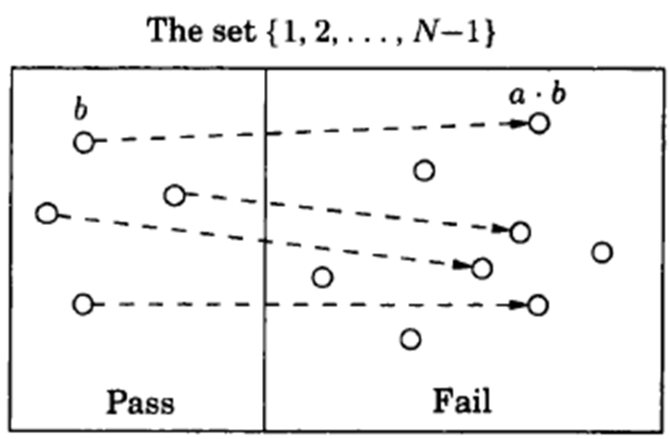
Day 10

Slide 12

Of course it doesn't work, we might just happen to pick an a for which a^(p-1) is congruent to 1, but N is not prime.

In fact, there are numbers (called Carmichael numbers) which are composite, but for which $a^{N-1} \equiv 1 \pmod N$ for all a that are relatively prime to N. These numbers are rare, and we'll see later how to deal with them.

Slide 13 (the diagram mentioned in the slides)



Slide 14

Do the test for k randomly-generated values of a < N.

Probability of error is < (1/2)^k

If k=100, dasgupta says the probability of error is less than the probability of a cosmic ray flipping some bits and messing up your computer's computation

Day 11

Slide 5 (2^{nd} bullet)

u is odd?

Or should I say "u are odd". To most of the world outside Rose-Hulman, if u would take this course or any 400-level CSSE course, u must be odd!

Note that this factorization of N-1 is fast. Just count how many bits at the end of N-1 are 0 to get t, and then bit-shift N-1 to get u.

Slide 8

(Journal of Number Theory 12 (1980) no. 1, pp 128-138)

Slide 17

Another example:

N=55 = 5*11. e = 3, d = 3^{-1} mod 40 = 27.

13^3 = 52 mod 55, 52^27 = 13 mod 55.