MA/CSSE 473 Day 05

Factors and Primes
Recursive division algorithm

MA/CSSE 473 Day 05

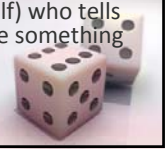
- HW 2 due tonight, 3 is due Monday
- **Student Questions**
- Asymptotic Analysis example: summation
- Review topics I don't plan to cover in class
- Continue Algorithm Overview/Review
  - Integer Primality Testing and Factoring
  - Modular Arithmetic intro
  - Euclid's Algorithm

MA/CSSE 473 Day 05

- HW 2 due tonight, 3 is due Monday
- **Student Questions**
- Asymptotic Analysis example: summation
- Continue Algorithm Overview/Review
  - Integer Primality Testing and Factoring
  - Modular Arithmetic intro

- Address the issue in my email from Tuesday evening:
- Some possible misunderstanding about my disclaimer
  - Was not intended to be funny, as the limerick was.
  - It was not my opinion, it was just a bunch of facts copied from a NIH web page (four years ago I think)
- http://www.niaaa.nih.gov/alcohol-health/special-populations-co-occurring-disorders/college-drinking

- I don't really like the term "date rape".
  - Rape is rape, and it is a horrendous crime.
- It is all about the perpetrator, not the victim.
  - In my opinion, the perp. should have a part of his anatomy cut off, or something worse.
- The impact on the victim is greater than anyone who has not experienced it can imagine.
- Nothing that the victim does can legitimize any kind of unwanted sexual advance.
- Anyone who implies that it is the victim's fault is just plain wrong!
  - But so many people believe this, that I can see how someone might interpretthe NIH statement as a "blame the victim" statement. This is a very serious and emotional issue.
  - So I again apologize for including that in the disclaimer.
- Anyone who won't take a single "no" at face value and respect it is evil, IMHO.
- Unfortunately such evil people exists, and unfortunately the people they choose as victims are often people whom they know, people who would never suspect them of being capable of something so horrendous.
- IMHO, the NIH web page was simply stating that alcohol is often involved in rape and other crimes. It can embolden the perp. And impair the victim's extrication ability.
- If anyone here is victim of rape or anyother kind of violence, my heart goes out to you.
- As you struggle to find healing, don't listen to anyone (including yourself) who tells you that you are at fault or that you in any way less of a person because something happened to you that was totally beyond your control.



Asymptotic Analysis Example

- Find a simple big-Theta expression (as a function of n) for the following sum
  - when $0 < c < 1$
  - when $c = 1$
  - when $c > 1$
- $f(n) = 1 + c + c^2 + c^3 + \ldots + c^n$

Quick look at review topics in textbook

REVIEW THREAD

Textbook Topics I Won't Cover in Class

- **Chapter 1 topics** that I will not discuss in detail unless you have questions. They should be review For some of them, there will be review problems in the homework
  - Sieve of Eratosthenes (all primes less than n)
  - Algorithm Specification, Design, Proof, Coding
  - Problem types : sorting, searching, string processing, graph problems, combinatorial problems, geometric problems, numerical problems
  - Data Structures: ArrayLists, LinkedLists, trees, search trees, sets, dictionaries,

Textbook Topics I Won't Cover*

- Chapter 2
  - Empirical analysis of algorithms should be review
  - I believe that we have covered everything else in the chapter except amortized algorithms and recurrence relations
  - We will discuss amortized algorithms
  - Recurrence relations are covered in CSSE 230 and MA 375. We'll review particular types as we encounter them.

***Unless you ask me to**

Textbook Topics I Won't Cover*

- Chapter 3 - Review
  - Bubble sort, selection sort, and their analysis
  - Sequential search and simple string matching

*Unless you ask me to

Textbook Topics I Won't Cover*

- Chapter 4 - Review
  - Mergesort, quicksort, and their analysis
  - Binary search
  - Binary Tree Traversal Orders (pre, post, in, level)

*Unless you ask me to

Textbook Topics I Won't Cover*

- Chapter 5 - Review
  - Insertion Sort and its analysis
  - Search, insertion, delete in Binary Tree
  - AVL tree insertion and rebalance

*Unless you ask me to

Integer Division
Modular arithmetic
Euclid's Algorithm
Heading toward Primality Testing

ARITHMETIC THREAD

FACTORING and PRIMALITY

- Two important problems
  - FACTORING: Given a number N, express it as a product of its prime factors
  - PRIMALITY: Given a number N, determine whether it is prime
- Where we will go with this eventually
  - Factoring is hard
    - The best algorithms known so far require time that is exponential in the number of bits of N
  - Primality testing is comparitively easy
  - A strange disparity for these closely-related problems
  - Exploited by cryptographic algorithms
- More on these problems later
  - First, more math and computational background…

Recap: Arithmetic Run-times

- For operations on two k-bit numbers:
- Addition: $\Theta(k)$
- Multiplication:
  - Standard algorithm: $\Theta(k^2)$
  - "Gauss-enhanced": $\Theta(k^{1.59})$, but with a lot of overhead.
- Division (We won't ponder it in detail, but see next slide): $\Theta(k^2)$

Algorithm for Integer Division

```
def divide(x, y):
    """ Input: Two non-negative integers x and y, where y>=1.
        Output: The quotient and remainder when x is divided by y."""
    if x == 0:
        return 0, 0
    q, r = divide(x // 2, y)   # max recursive calls:
    q, r = 2 * q, 2 * r        #   number of bits in x
    if x % 2 == 1:
        r = r + 1
    if r >= y:                 # note that all of the multiplications
        q, r = q + 1, r - y    # and divisions are by 2:
    return q, r                #     simple bit shifts
```

Let's work through divide(19, 4).

Analysis?

Modular arithmetic definitions

- **x modulo N** is the remainder when x is divided by N. I.e.,
  - If x = qN + r, where 0 ≤ r < N (**q and r are unique!**),
  - then **x modulo N** is equal to r.
- x and y are **congruent modulo N**, which is written as $x \equiv y \pmod{N}$, if and only if N divides (x-y).
  - i.e., there is an integer k such that x-y = kN.
  - In a context like this, **a divides b** means "divides with no remainder", i.e. "a is a factor of b."
- Example: $253 \equiv 13 \pmod{60}$

Modular arithmetic properties

- Substitution rule
  - If $x \equiv x' \pmod N$ and $y \equiv y' \pmod N$, then $x + y \equiv x' + y' \pmod N$, and $xy \equiv x'y' \pmod N$
- Associativity
  - $x + (y + z) \equiv (x + y) + z \pmod N$
- Commutativity
  - $xy \equiv yx \pmod N$
- Distributivity
  - $x(y+z) \equiv xy + yz \pmod N$

Modular Addition and Multiplication

- To **add** two integers x and y modulo N (where k = $\lceil \log N \rceil$ (the number of bits in N), begin with regular addition.
  - x and y are in the range_____, so x + y is in range _____
  - If the sum is greater than N-1, subtract N.
  - Run time is Θ ( )
- To **multiply** x and y modulo N, begin with regular multiplication, which is quadratic in k.
  - The result is in range _____ and has at most ____ bits.
  - Compute the remainder when dividing by N, quadratic time. So entire operation is Θ( )

Modular Addition and Multiplication

- To **add** two integers x and y modulo N (where k = $\lceil \log N \rceil$, begin with regular addition.
  - x and y are in the range **0 to N-1**, so x + y is in range **0 to 2N-1**
  - If the sum is greater than N-1, subtract N.
  - Run time is Θ (**k**)
- To **multiply** x and y, begin with regular multiplication, which is quadratic in n.
  - The result is in range 0 to **$(N-1)^2$** and has at most **2k** bits.
  - Then compute the remainder when dividing by N, quadratic time in k. So entire operation is Θ(**k^2**)

Modular Exponentiation

- In some cryptosystems, we need to compute x^y modulo N, where all three numbers are several hundred bits long. Can it be done quickly?
- Can we simply take x^y and then figure out the remainder modulo N?
- Suppose x and y are only 20 bits long.
  - x^y is at least $(2^{19})^{(2^{19})}$, which is about 10 million bits long.
  - Imagine how big it will be if y is a 500-bit number!
- To save space, we could repeatedly multiply by x, taking the remainder modulo N each time.
  - If y is 500 bits, then there would be 2^{500} bit multiplications.
  - This algorithm is exponential in the length of y.
  - Ouch!

Modular Exponentiation Algorithm

```
def modexp(x, y, N):
    if y==0:
        return 1
    z = modexp(x, y/2, N)
    if y%2 == 0:
        return (z*z) % N
    return (x*z*z) % N
```

- Let n be the maximum number of bits in x, y, or N
- The algorithm requires at most ___ recursive calls
- Each call is Θ( )
- So the overall algorithm is Θ( )

Modular Exponentiation Algorithm

```
def modexp(x, y, N):
    if y==0:
        return 1
    z = modexp(x, y/2, N)
    if y%2 == 0:
        return (z*z) % N
    return (x*z*z) % N
```

- Let n be the maximum number of bits in x, y, or N
- The algorithm requires at most **n** recursive calls
- Each call is Θ(**n^2**)
- So the overall algorithm is Θ(**n^3**)