# Introduction

Analysis of Algorithms
CSCI 5620

# Topics

- Intro
- Functions
- Induction
- Complexity

# Prime numbers

- A positive integer p>1, is ***<u>prime</u>*** if the only positive integers which divide p are 1 and p.
  - Exs: 2, 3, 5, 7, 11, 13, 17, 19, 23

# Example

- Input: A n-bit number, L>2
- Output: Is L prime? Yes/No

1 Write a SIMPLE algorithm to solve this problem.

2 If L is prime, how many steps does your algorithm take in terms of n?

# For a 100 bit number

- http://www.top500.org/news/articles/article_21.php
- 35.86 Tflop/s ("teraflops" trillions of calculations per second)  (2005)
- 8 Petaflops/s (quadrillion) 2011
- 2013 NuDT Tiahe-2: 33.86 PFLOPS
- Could take as long as 1,142,465,658 years with this algorithm (2005). 51,211,102 years (2011)
- There is a much faster (not exponential) algorithm, which can determine this is in a matter of seconds.
- In cryptography, the "key" usually involves prime numbers in some manner.
  - Ex in RSA the public key is a pair (e,n) where n is the product of two primes.

# Algorithms

- An Algorithms is a well-defined computational procedure that takes some values (as input) and produces values as (output)
- Examples: Insertion Sort, Bubble Sort, Heap Sort, Merge Sort, Quick Sort
- Algorithms are described in pseudocode in this course.

# Analyzing Algorithms

- Is the algorithm correct and as fast as possible?
  - A algorithm is correct if for every instance, it halts with the correct output.
- Analyzing algorithms : predicting resources that the algorithm requires.
- Running time: number of steps performed
  - Expressed as a function ( order of growth)
  - Machine independent
- Worst case, average case, best case
  - Worst case: longest running time for any input of size n.
  - Average case : Expected running time
  - Best Case: Shortest running time for any input of size n.

# Functions in CS

- (mod-n functions) $f_n(m) = m \pmod{n}$
  - Example $f_3(7)=1$, $f_3(8)=2$, $f_3(111)=0$
- $f(n)=n!$
  - $f(1)=1$, $f(2)=2$, $f(3)=6$, $f(4)=24$, $f(5)=20$, etc
- (Floor function) $f(x)=\lfloor x \rfloor$
  - $f(2.1)=2$
  - $f(2.9)=2$
  - $f(2.999999)=2$
  - $f(3)=3$
  - $f(-2.3)=-3$

# Functions in CS

- Polynomial of degree d,
  - $P(n) = a_0 + a_1 n + a_2 n^2 + \ldots + a_d n^d$
  - $a_i$ are all reals

# Functions in CS

- Exponential example $f(x)=2^x$
  - $f(1)=2$
  - $f(2)=4$
  - $f(2.2)=4.4957\ldots$
  - $F(0)=1$
  - $f(-1)=1/2$
  - $f(-2)=1/4$

# Functions in CS

- Logarithm $f_n(x)=\log_n(x)$
  - $f_2(4)=2$
  - $f_2(8)=3$
  - $f_2(2)=1$
  - $f_2(1)=0$
  - $f_2(1/2)=-1$
  - $f_2(1/16)=-4$
  - $f_2(0)=$ undefined

# Logs

- $\lg n = \log_2 n$
- $\ln n = \log_e n$
- $\log n = \log_{10} n$
- $\log_b a = \dfrac{\log_c a}{\log_c b}$
- $\log_b a^n = n\log_b a$
- $\log_b ac = \log_b a + \log_b c$

# Logs continue

- lg (32) = ?
- lg (1/8) =?
- lg (15) = log 15/ log ?
- Convert lg n to log, lg n = ?
- lg (15) =lg (5) + lg ?
- lg (15) = lg (30/?)
- lg ($3^5$)= 5 * ?
- lg ($2^n$) =?
- The number of bits to represent a number N is ?

# Mathematical Induction

- $P(n) := 1+2+3+\ldots+n = n(n+1)/2$
- Show P(n) is true for all n>=1
  - Show P(1) is true (Basis Step)
  - Show $P(k) \Rightarrow P(k+1)$
  - There true for all n
- Proof On Board
- Quiz
- More induction proofs

# More Induction Problems

- Fibonacci numbers: 0, 1, 1, 2, 3, 5, 8, 13, 21, …
- $f_0=0$, $f_1=1$, $f_i=f_{i-1}+f_{i-2}$ for $i\geq 2$
- Prove $f_n=f_0+f_1+\ldots f_{n-2}+1$

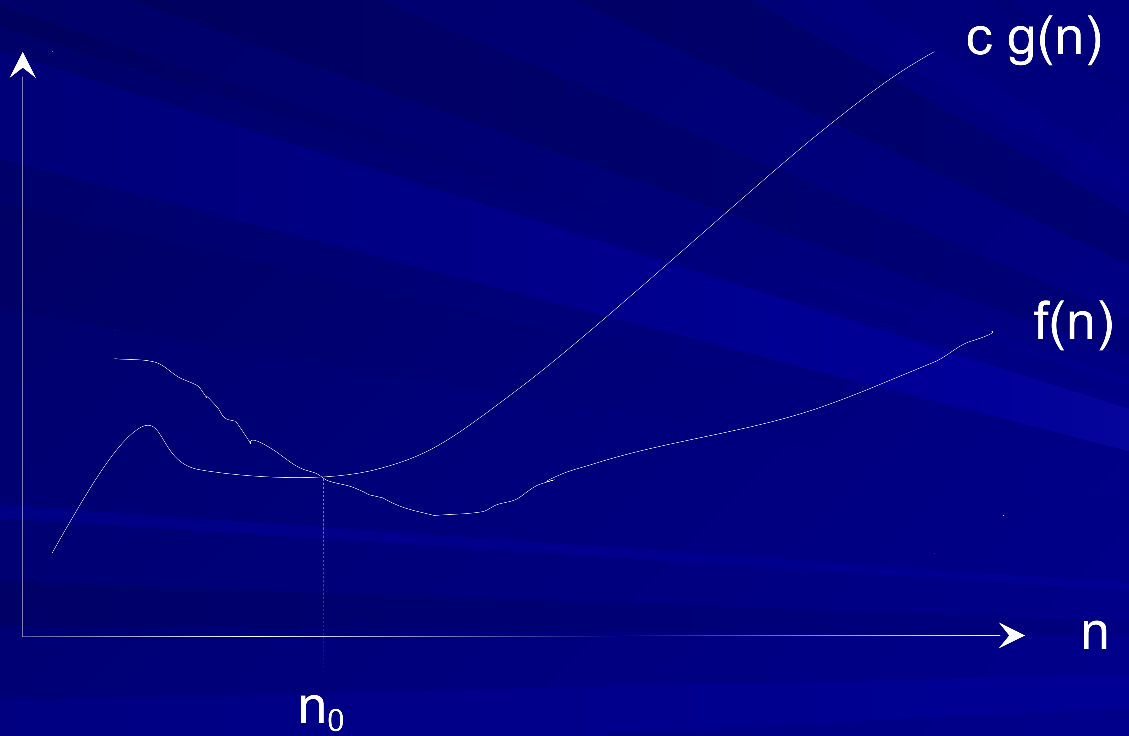
# More Induction Examples

- The maximum number of regions the plane is divided into by n lines is

$$\frac{1}{2}(n^2+n+2)$$

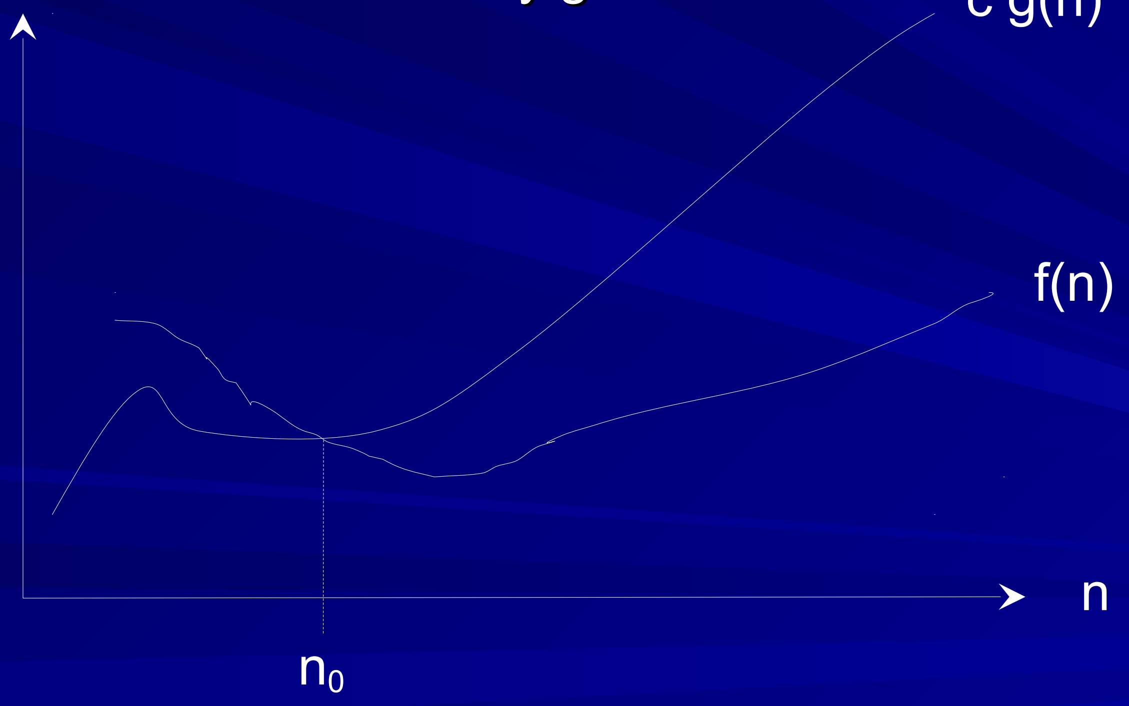
# Big Oh: Start

- f(n) is bounded above by c g(n)



# Big-Oh notation

- $f \in O(g)$, if there exist c and k so that $|f(n)| \leq c|g(n)|$ for all $n \geq k$.
  - f grows no faster than g does
  - f is bounded above by g



# Big – Oh Notation

- Notation $f \in O( g )$
  - Verbalize as “f is in Big-Oh of g ”
- We say that $f \in O( g )$ if
  - There exists two constants c and $n_0$ such that

$$f(n) < c * g(n) \qquad \text{for all} \qquad n > n_0$$

- Important implications
  - g serves as an upper bound on f
  - Ignores behavior of f for small values

# Big – Oh (cont)

- For continuous functions
  - $f \in O(g)$ if

$$\lim_{x \to \infty} \frac{f(x)}{g(x)} = c \quad \text{for some} \quad c \in \Re^*$$

- You must learn and be able to apply both
  - Learn definition ≡ memorize

# L'Hospital's Rule

- Frequently the second definition will yield one of the indeterminate forms

$$\frac{0}{0} \text{ or } \frac{\infty}{\infty}$$

- Which can be resolved by the use of L'Hospital's rule

$$\lim_{x \to \infty} \frac{f(x)}{g(x)} = \lim_{x \to \infty} \frac{f'(x)}{g'(x)}$$

  – provided the derivatives $f'(x)$ and $g'(x)$ exist

# Big-Oh notation

- Show $2n+1 \in O(n^2)$.
  - $C=3$, $k=2$
- What is in $O(n^3)$?
  - $f(n)=10n^3$
  - $f(n)=n^2$
  - $f(n)=n^3+100n^2-n$
  - $f(n)=2$
  - $f(n)=\lg n$
  - $f(n)=(\lg n)n$

# “Standard” Functions

| Function | Name |
|---|---|
| 1 | Constant |
| lg n, log n, ln n | Logarithmic |
| n | Linear |
| n lg n | n lg n |
| $n^2$ | Quadratic |
| $n^3$ | Cubic |
| $2^n$ | Exponential |
| $3^n$ | Exponential |
| n! | Factorial |

In order of increasing run time

Proof Quiz

# Big-theta notation

- $f \in \Theta(g)$, if f and g have same order, i.e. $f \in O(g)$ and $g \in O(f)$
- What is in $\Theta(n^3)$
  - $n^3 + 100n^2 - n$ is
  - 2 is not
  - $n^2$ is not
  - lg n is not
  - (lg n)n is not

# Big-omega notation

- $f \in \Omega(g)$, if there exist c and k so that $cg(n) \leq f(n)$ for all $n \geq k$.
  - f grows at least as fast as g
- Example $n^2 \in \Omega(n)$.

- What is in $\Omega(n^3)$?
  - $f(n)=10n^3$
  - $f(n)=n^2$
  - $f(n)=n^3+100n^2-n$
  - $f(n)=2$
  - $f(n)=\lg n$
  - $f(n)=(\lg n)n$
  - $f(n)=n^4+100n^2-n$

# Limits:

- $f \in o(g)$
- $o \sim$ ‘<‘ $\quad \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$
- $O \sim$ ‘≤’ $\quad \lim_{n\to\infty} \frac{f(n)}{g(n)} = c$, where $c \in$ ***R***
- $\Theta \sim$ ‘=‘ $\quad \lim_{n\to\infty} \frac{f(n)}{g(n)} = c$, where $c \in$ ***R, c≠0***
- $\Omega \sim$ ‘≥’
  - $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$ or

# Running time

- The $\Theta$-class of a function that describes the number of steps performed by an algorithm is called the ***running time*** of the algorithm.
- This allows us to compare algorithms for a specify task.
- For example bubble sort is $\Theta(n^2)$ and quicksort is in $\Theta(n(\lg n))$, but what is the fastest running time sorting algorithm?

# Efficiency

- Insertion Sort $\Theta( n^2 )$
- Merge Sort $\Theta (n \lg n)$
- For n=1,000,000
  - $n^2 =$ 1,000,000,000,000
  - $n \lg n \approx$ 19,931,569

# Operation Counts

| Input Size | Constant (1) | lg N | N | N lg N | $N^2$ | $2^N$ | N! |
|---|---|---|---|---|---|---|---|
| 10 | 1 | 4 | 10 | 40 | $10^2$ | 1024 | $\sim 4 \times 10^6$ |
| 100 | 1 | 7 | 100 | 700 | $10^4$ | $\sim 10^{30}$ | $\sim 9 \times 10^{157}$ |
| 1000 | 1 | 10 | 1000 | 10,000 | $10^6$ | - | - |
| $10^6$ | 1 | 20 | $10^6$ | $2 \times 10^7$ | $10^{12}$ | - | - |

# Homework

- I will post homework tomorrow in Elearn..