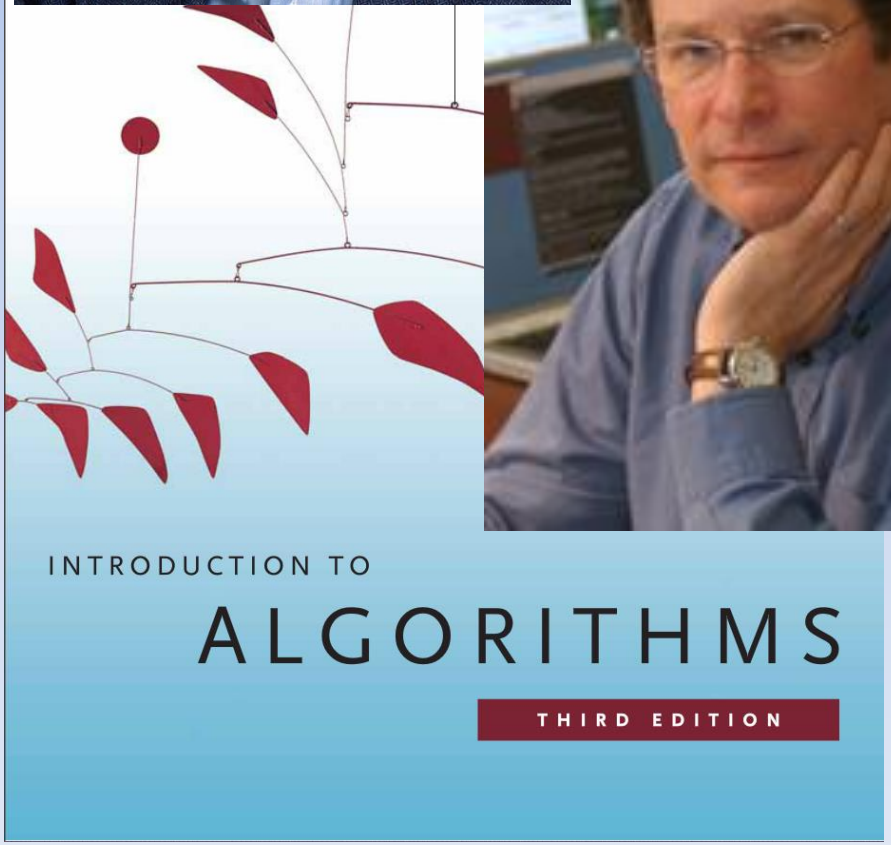
Design and Analysis of Algorithms

Topic: Algorithms



Chapter 1: Algorithms : Computer Science Topics

- Algorithms break down into:
  - Constructs:
    - Variables
    - Branching
    - Looping
    - Functions
    - Recursion
  - Data
    - Data Structures

Computer Science Topics : Tools & Technology

- Languages
  - PHP, Python, C++, Java
- Frameworks
  - Django, Ruby on Rails
- Tools
  - MySQL, MongoDB
- Platforms
  - Amazon Web Services

Computer Science Topics : Scientific Support

- Mathematics:
  - Probability Theory
  - Calculus
  - Linear Algebra
  - Discrete Math
- Biology
  - Human Intelligence
  - Genetic Algorithms

Our Focus: The Algorithm

IN: Hacking Scene
w/ Mark Zuckerberg speaking to Eduardo Saverin

Mark Zuckerberg:
I need the algorithm..
I need the algorithm..

- A procedure or formula for solving a problem.
- A tool for solving a well-specified computational problem.
  - The statement of the problem specifies in general terms the desired input/output relationship.
  - The algorithm describes a specific computational procedure for achieving that input/output relationship.

Algorithm's History

- Algorithms existed long before computers.
- Al-Khwārizmī(Abū ʿAbdallāh Muḥammad ibn Mūsā al-Khwārizmī)
  - (Persian: عَبْدَالله مُحَمَّد بِن مُوسَى اَلْخْوَارِزْمِي
  - Earlier transliterated as Algoritmi or Algaurizin
  - (c. 780 – c. 850) was a Persian mathematician, astronomer and geographer during the Abbasid Caliphate, a scholar in the House of Wisdom in Baghdad.

Courtesy: Wikipedia

Algorithm's History

- Words reflecting the importance of al-Khwarizmi's contributions to mathematics.
  - "Algebra"
    - derived from al-jabr,
    - one of the two operations he used to solve quadratic equations.
  - Algorism and algorithm stem from Algoritmi, the Latin form of his name.
  - His name is also the origin of (Spanish) guarismo and of (Portuguese) algarismo, both meaning digit.

- **"On the Calculation with Hindu Numerals"**
  - **written about 825,**
  - **principally responsible for spreading the Indian system of numeration throughout the Middle East and Europe.**
  - Translated into Latin as Algoritmi de numero Indorum.
  - Al-Khwārizmī, rendered as (Latin) Algoritmi, led to the term "algorithm".

Courtesy: Wikipedia

Algorithms as Technology

- With Infinitely fast computers… Would Algorithms matter?

- YES: You would still need to know the algorithm would terminate with the correct answer!

Algorithms as Technology

- Computers are not infinitely fast, and Memory is not free.
- Algorithm efficiency for same problem can vary drastically!

Problem 1-1

- 1-1 Comparison of running times For each function f(n) and time t in the following table, determine the largest size n of a problem that can be solved in time t, assuming that the algorithm to solve the problem takes f(n) microseconds.

Problem 1-1

	1 second	1 minute	1 hour	1 day	1 month	1 year	1 century
$\lg n$							
$\sqrt{n}$							
n							
$n \lg n$							
n^2							
n^3							
2^n							
$n!$							

Times

- 1 second
- 1 minute (60 seconds)
- 1 hour (3600 seconds)
- 1 day (24 hours or 86400 seconds)
- 1 month (30 days or 2.592 million seconds)
- 1 year (31.5 million seconds)
- 1 century (3.15 billion seconds)

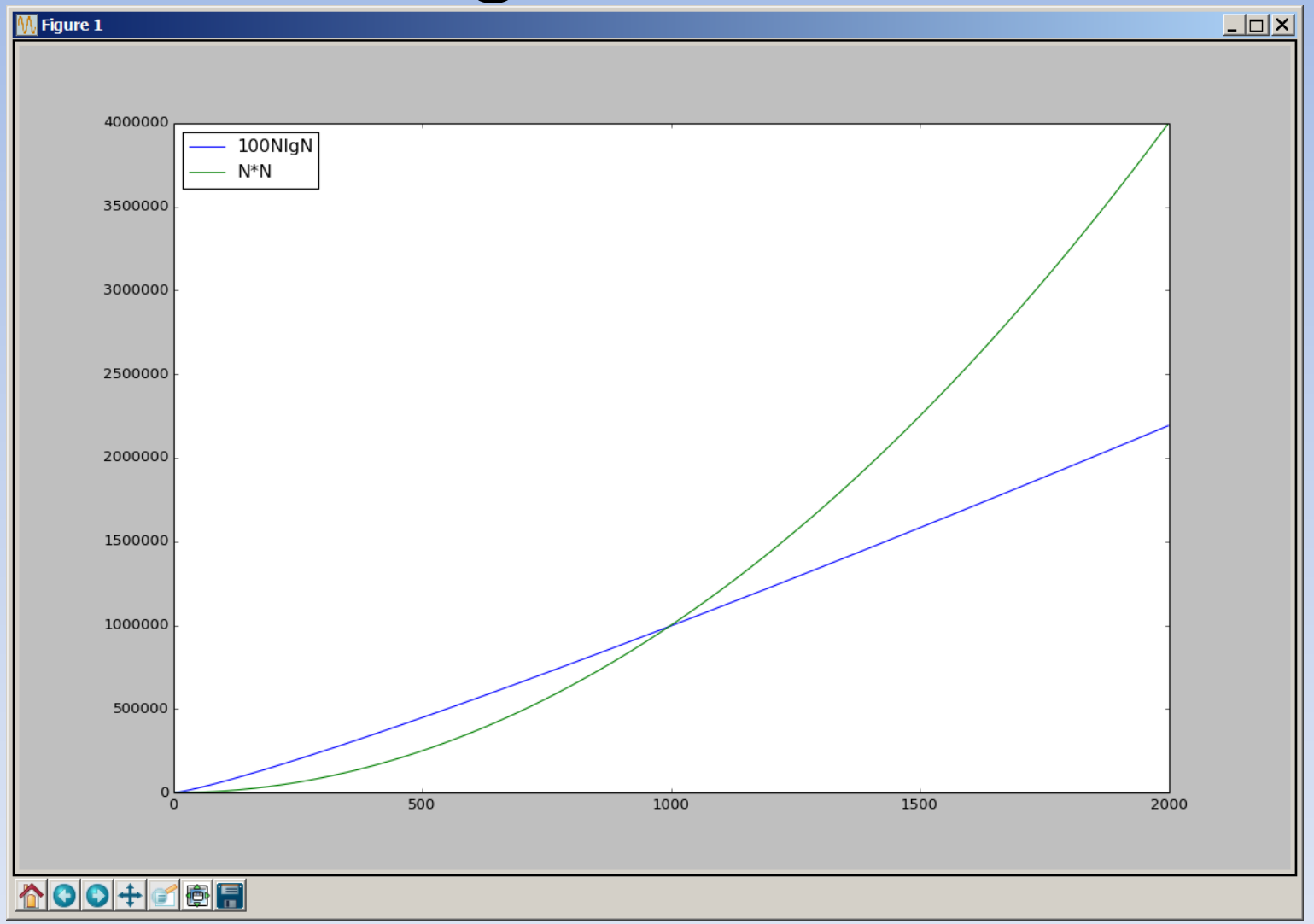
- 1 second = 1 million microseconds

Size of problem solvable in time N w/ O(n), O(NlgN), O(N^2)

```
24  def Sizeit(fn, n):
25      i = 1
26      k = 10
27      while (fn(i)< n):
28          i += k
29          if ( round(math.log(i,10)) > math.log(k) ):
30              k *= k
31      return i
```

- [1000000, 60000000, 3600000000L, 86400000000L, 2592000000000L, 31536000000000L, 315360000000000L]
- N*N: 1021; 7821; **61,721**; 301721; 1611721; 5621721; 56,161,721
- NlgN: 71721; 2801721; **133,381,721**, 2755151721; 71962281,721; 797662281721; 68610962281721
- N: 1,001,721; 60,001,721; **3,662,281,721**; 86462281721; 2592062281721; 31536062281721; 315360062281721;

100NlgN versus N*N



Algorithms & Performance Results

- Sorting Algorithms

Interesting Problems: Computational Geometry (CH33)

- Finding the Convex Hull

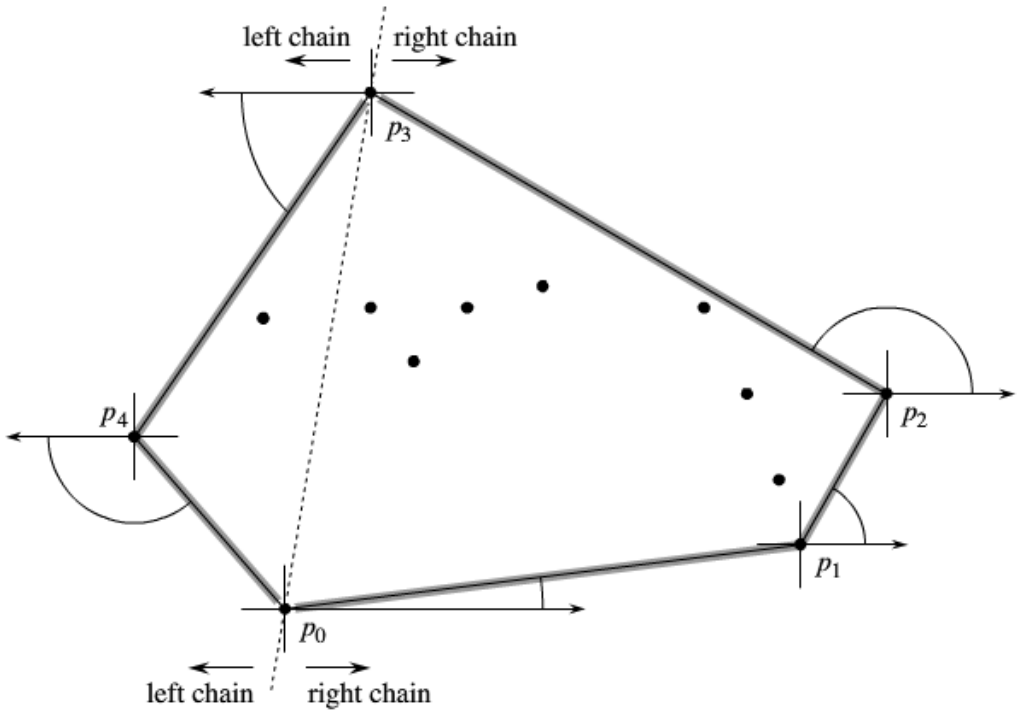
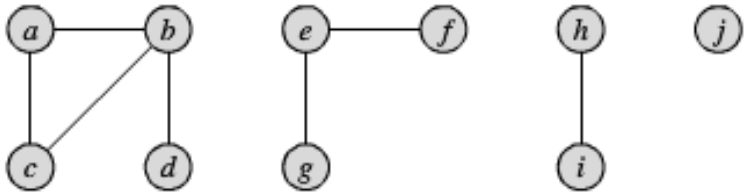


Figure 33.9 The operation of Jarvis's march. We choose the first vertex as the lowest point p_0. The next vertex, p_1, has the smallest polar angle of any point with respect to p_0. Then, p_2 has the smallest polar angle with respect to p_1. The right chain goes as high as the highest point p_3. Then, we construct the left chain by finding smallest polar angles with respect to the negative x-axis.

Interesting Problem: Disjoint Sets (CH21)



(a)

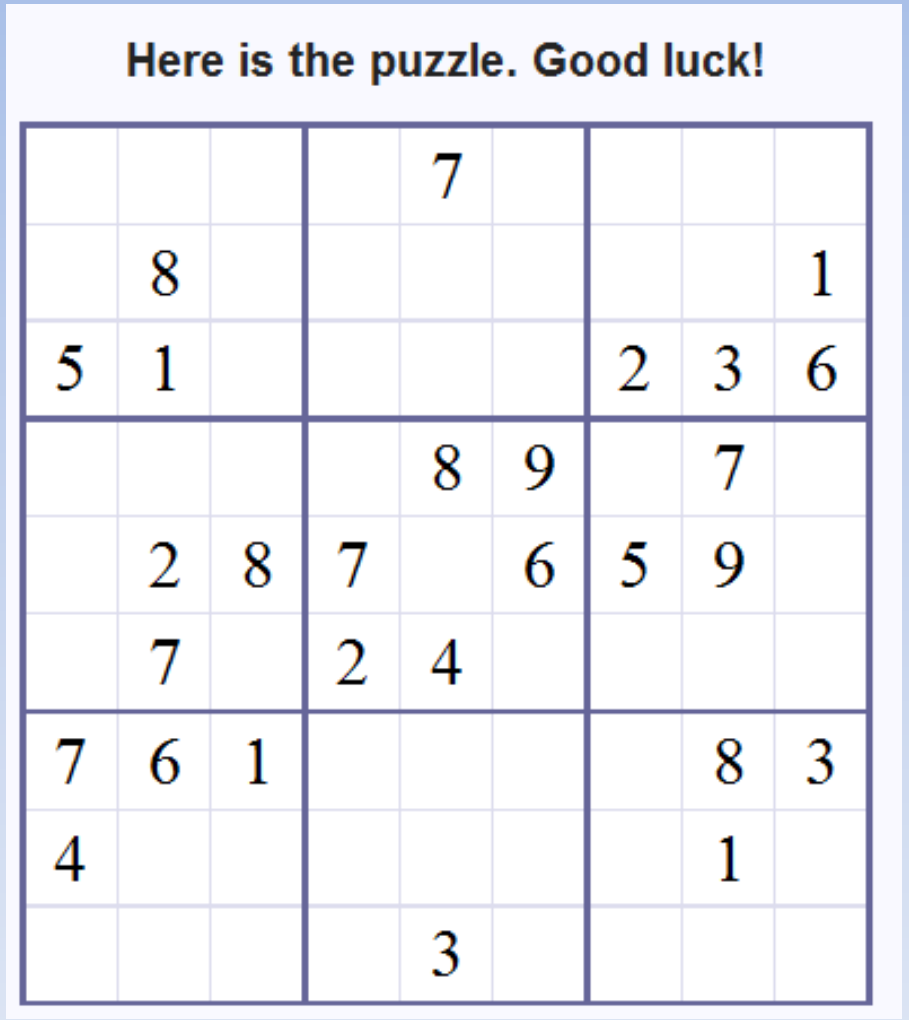
Edge processed	Collection of disjoint sets									
initial sets	{a}	{b}	{c}	{d}	{e}	{f}	{g}	{h}	{i}	{j}
(b,d)	{a}	{b,d}	{c}		{e}	{f}	{g}	{h}	{i}	{j}
(e,g)	{a}	{b,d}	{c}		{e,g}	{f}		{h}	{i}	{j}
(a,c)	{a,c}	{b,d}			{e,g}	{f}		{h}	{i}	{j}
(h,i)	{a,c}	{b,d}			{e,g}	{f}		{h,i}		{j}
(a,b)	{a,b,c,d}				{e,g}	{f}		{h,i}		{j}
(e,f)	{a,b,c,d}				{e,f,g}			{h,i}		{j}
(b,c)	{a,b,c,d}				{e,f,g}			{h,i}		{j}

(b)

Figure 21.1 **(a)** A graph with four connected components: $\{a, b, c, d\}$, $\{e, f, g\}$, $\{h, i\}$, and $\{j\}$. **(b)** The collection of disjoint sets after processing each edge.

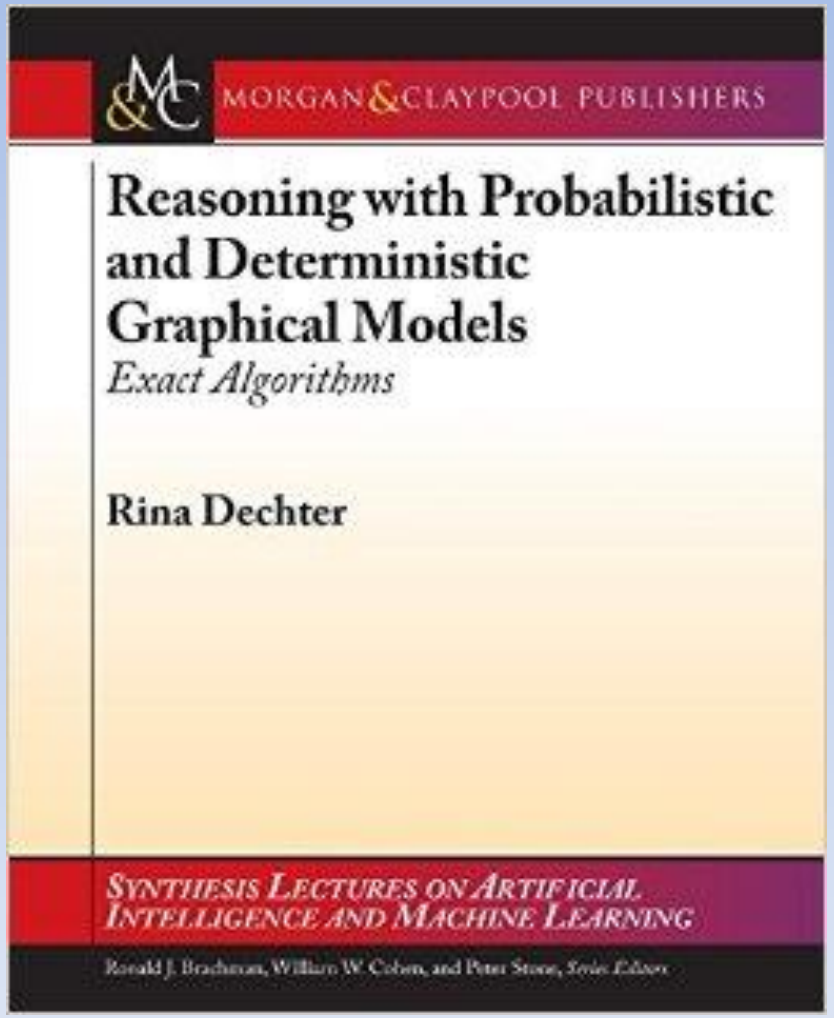
Interesting Problem: Constraint Satisfaction Problems

- Sudoku

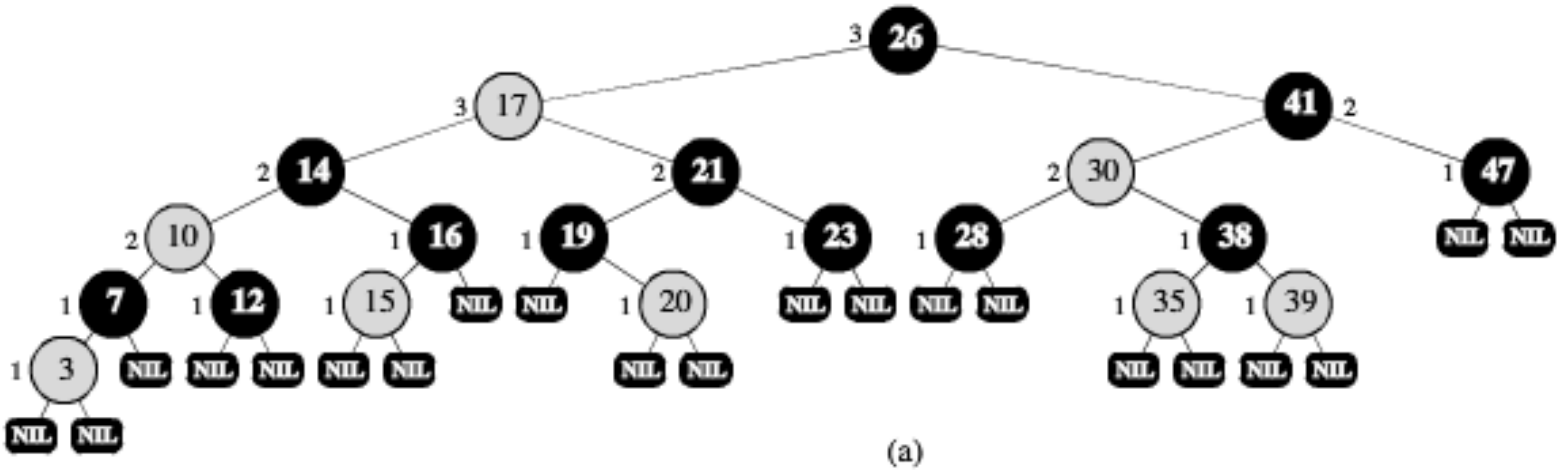


Interesting Problem: Bayesian Inference

- Variable Elimination Algorithm for Bayesian Networks

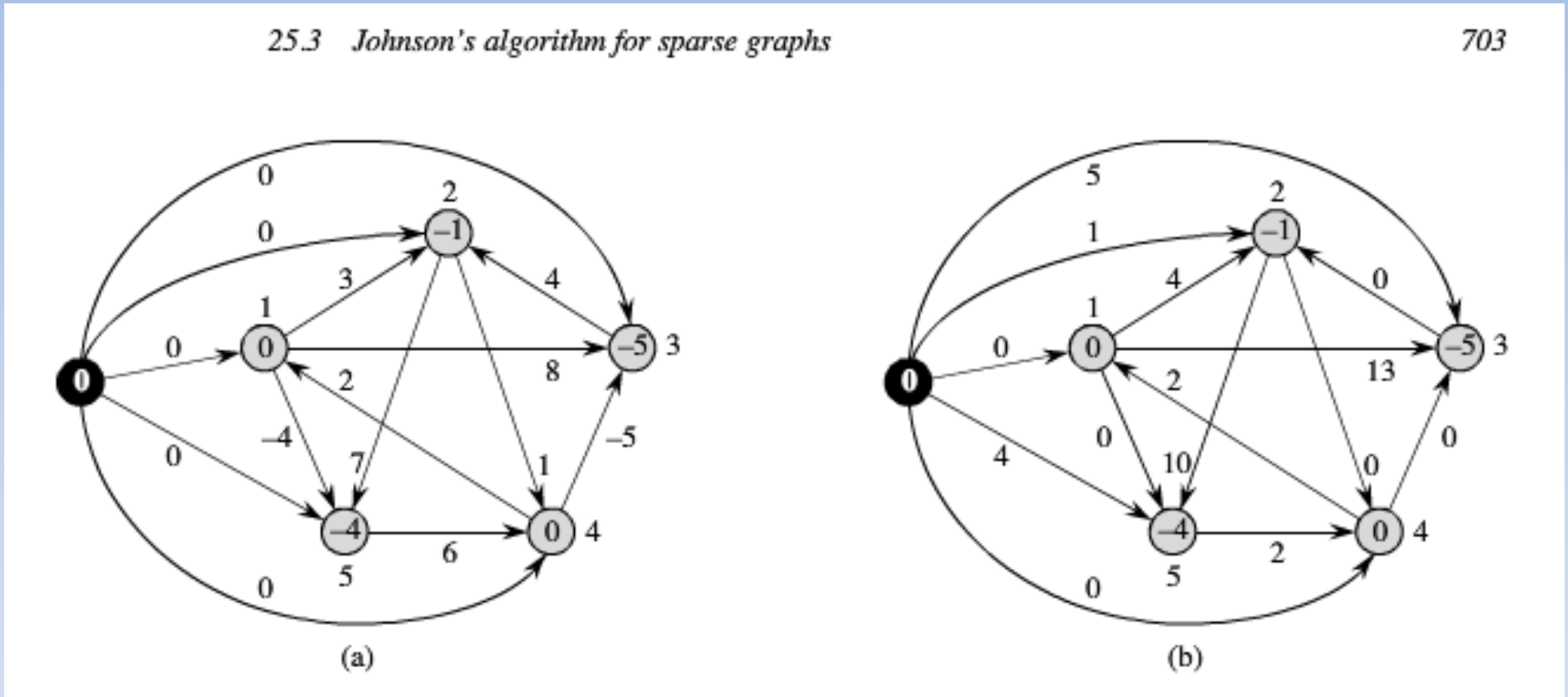


Interesting Problem: Balanced Search Tree (CH13)



(a)

Interesting Problem: Graphs (CHs 22, 23, 24, 25, 26)



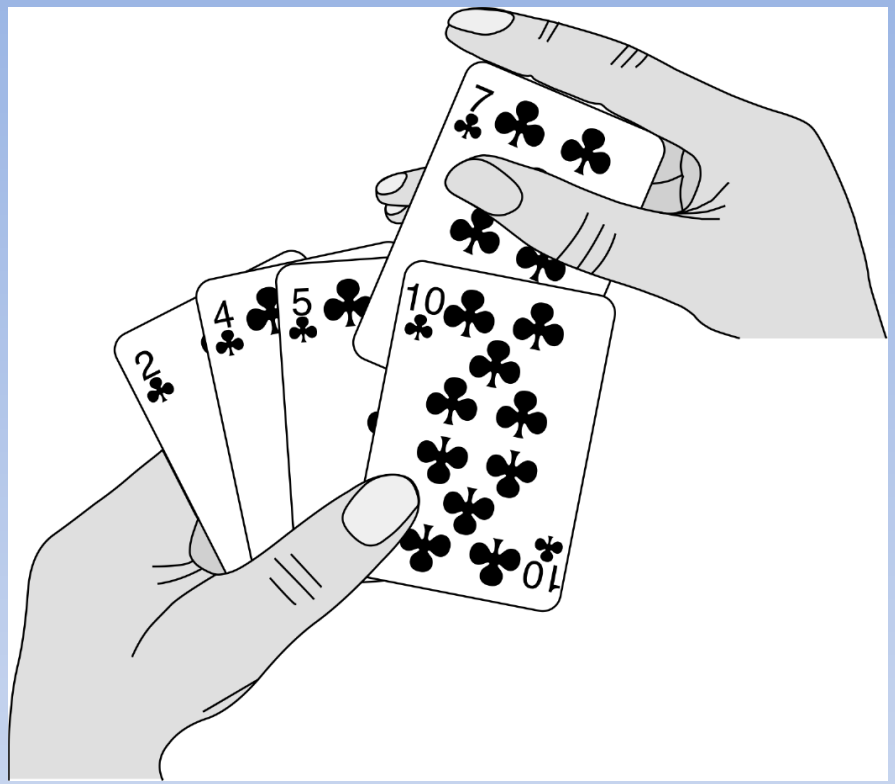
- Find shortest paths between all pairs of vertices.

Chapter 2: Getting Started

- Analyzing Algorithms:
  - Introduces Framework used by book.

Classic Problem: Sorting

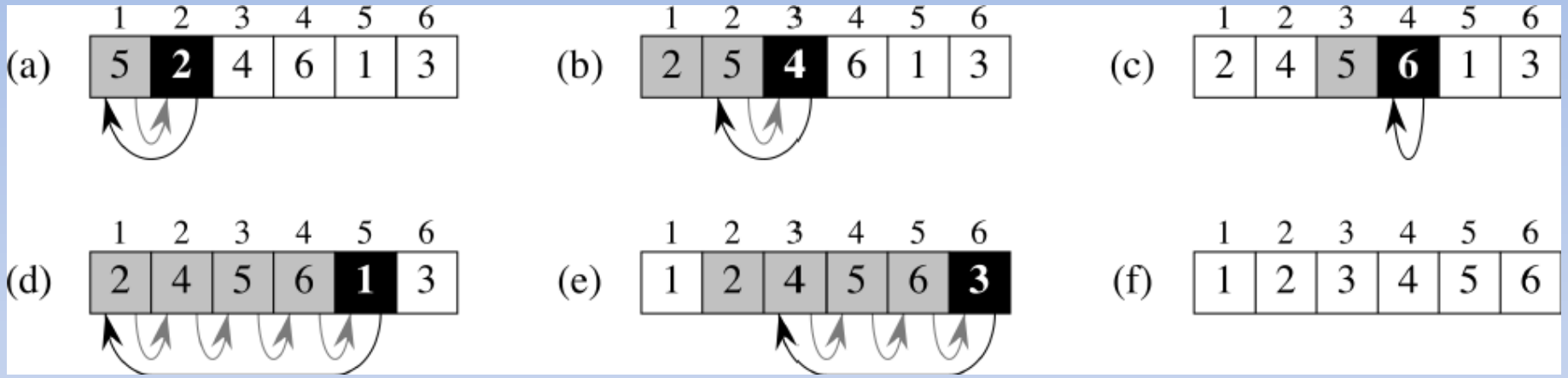
- What is sorting?
- Algorithms?
- Complexity?
  - Space
  - Time
- Best Case?



- Add card correctly into sorted set.
- Start with set size 1.
- Add cards into right space, until all cards in hand.

Pseudocode

```
INSERTION-SORT(A)
1  for j = 2 to A.length
2      key = A[j]
3      // Insert A[j] into the sorted
           sequence A[1 .. j − 1].
4      i = j − 1
5      while i > 0 and A[i] > key
6          A[i + 1] = A[i]
7          i = i − 1
8      A[i + 1] = key
```


(a)
1 2 3 4 5 6
5 2 4 6 1 3
(b)
1 2 3 4 5 6
2 5 4 6 1 3
(c)
1 2 3 4 5 6
2 4 5 6 1 3
(d)
1 2 3 4 5 6
2 4 5 6 1 3
(e)
1 2 3 4 5 6
1 2 4 5 6 3
(f)
1 2 3 4 5 6
1 2 3 4 5 6

Need to Analyze Performance

- To Do Analysis need Model Of Computation
  - RAM
  - Pointer Model
  - Python Model

Models of Computation

- Specifies:
  - What operations an algorithm is allowed
  - Cost (time, …) of each op.
- Random Access Machine (RAM)
  - Random access memory modeled by big array.
  - Θ(1) time can
    - Load words
    - Do Θ(1) computations
    - Store Θ(1) words
  - Θ(1) registers
  - Word is w bits
    - W >= log(size of memory)

Models of Computation (2)

- Pointer Machine
  - Dynamically allocated objects
  - Object has $\Theta(1)$ fields
  - Field =
    - Word
    - Pointer
      - Points to another object
      - Null, nil, None

Models of Computation (3)
Python Model

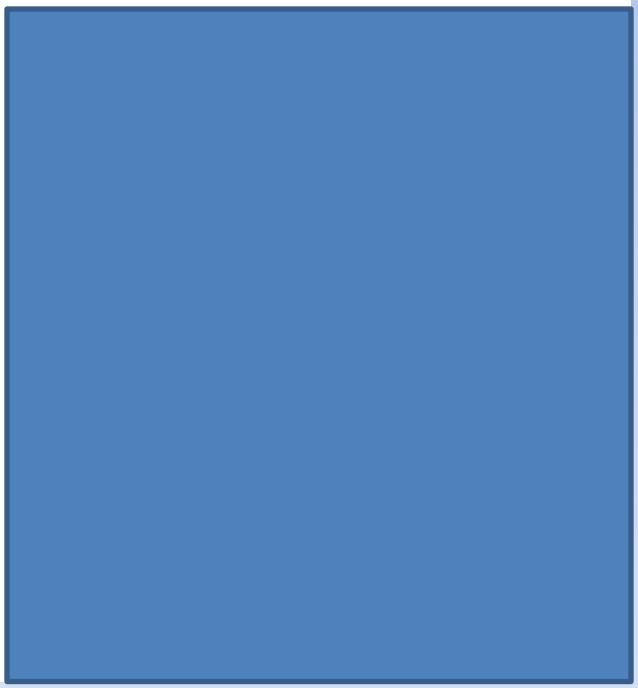
- "List" = array
  - L[i] = l[j] + 5 {takes constant time} Θ(1)
- Object with Θ(1) attributes
  - For reasonable number of attributes
  - X = X.next {Θ(1)}
- L.append(x)
  - Table doubling (L9)
  - Θ(1)
- L1 + L2
  - For x in L1: L.append(x)
  - For x in L2: L.append(x)
  - Θ(1 + length(L1) + length(L2))
- X in L
  - Θ(length(L)), Θ(n)
- Len(L)
  - Θ(1)
  - Counter built in

Models of Computation (3)
Python Model

- L.sort() -> $\Theta(|L| \lg |L|)$
  - L3
- Dict:
  - D[key] = val {$\Theta(1)$}
  - Hash Table
  - Constant Time **With High Probability** (w.h.p.)
  - Dictionaries are GREAT!
- Longs:
  - X+Y {$\Theta(|X| + |Y|)$}
  - X*Y {$\Theta((|X| + |Y|)^{\log 3})$}
- Heapq

```
INSERTION-SORT(A)
1  for j = 2 to A.length
2      key = A[j]
3      // Insert A[j] into the sorted
           sequence A[1 .. j − 1].
4      i = j − 1
5      while i > 0 and A[i] > key
6          A[i + 1] = A[i]
7          i = i − 1
8      A[i + 1] = key
```

```
INSERTION-SORT(A)                                   cost      times
1  for j = 2 to A.length                            c1        n
2      key = A[j]
3      // Insert A[j] into the sorted
           sequence A[1 .. j - 1].
4      i = j - 1
5      while i > 0 and A[i] > key
6          A[i + 1] = A[i]
7          i = i - 1
8      A[i + 1] = key
```



INSERTION-SORT(A)	*cost*	*times*
1 **for** $j = 2$ **to** $A.length$	c_1	n
2 $key = A[j]$	c_2	$n-1$
3 // Insert $A[j]$ into the sorted sequence $A[1..j-1]$.	0	$n-1$
4 $i = j-1$	c_4	$n-1$
5 **while** $i > 0$ and $A[i] > key$		
6 $A[i+1] = A[i]$		
7 $i = i-1$		
8 $A[i+1] = key$		

INSERTION-SORT(A)	*cost*	*times*
1 **for** $j = 2$ **to** $A.length$	c_1	n
2 $key = A[j]$	c_2	$n-1$
3 // Insert $A[j]$ into the sorted sequence $A[1..j-1]$.	0	$n-1$
4 $i = j-1$	c_4	$n-1$
5 **while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6 $A[i+1] = A[i]$		
7 $i = i-1$		
8 $A[i+1] = key$		

- t_j = the number of times the loop 5-6-7 executes for each value of j.

INSERTION-SORT(A)	*cost*	*times*
1 **for** $j = 2$ **to** $A.length$	c_1	n
2 $key = A[j]$	c_2	$n-1$
3 // Insert $A[j]$ into the sorted sequence $A[1..j-1]$.	0	$n-1$
4 $i = j-1$	c_4	$n-1$
5 **while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6 $A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
7 $i = i-1$		
8 $A[i+1] = key$		

- t_j = the number of times the loop 5-6-7 executes for each value of j.

INSERTION-SORT(A)	*cost*	*times*
1 **for** $j = 2$ **to** $A.length$	c_1	n
2 $key = A[j]$	c_2	$n-1$
3 // Insert $A[j]$ into the sorted sequence $A[1..j-1]$.	0	$n-1$
4 $i = j-1$	c_4	$n-1$
5 **while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6 $A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n}(t_j-1)$
7 $i = i-1$	c_7	$\sum_{j=2}^{n}(t_j-1)$
8 $A[i+1] = key$		

- t_j = the number of times the loop 5-6-7 executes for each value of j.

INSERTION-SORT(A)	*cost*	*times*
1 **for** $j = 2$ **to** $A.length$	c_1	n
2 $key = A[j]$	c_2	$n-1$
3 // Insert $A[j]$ into the sorted sequence $A[1..j-1]$.	0	$n-1$
4 $i = j-1$	c_4	$n-1$
5 **while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6 $A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n}(t_j-1)$
7 $i = i-1$	c_7	$\sum_{j=2}^{n}(t_j-1)$
8 $A[i+1] = key$	c_8	$n-1$

- t_j = the number of times the while loop test of line 5 executes.

What is the total cost?

- It can vary based on the value of t_j!

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n} (t_j - 1) + c_7 \sum_{j=2}^{n} (t_j - 1) + c_8(n-1) .$$

5	**while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6	$A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
7	$i = i - 1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$

How about Best Case ??

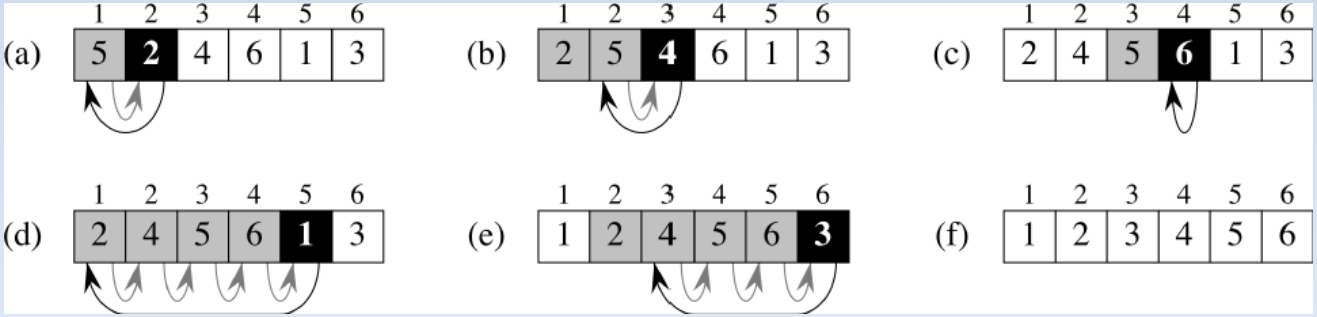
What is the total cost?

- It can vary based on the value of t_j!

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n} (t_j - 1) + c_7 \sum_{j=2}^{n} (t_j - 1) + c_8(n-1) .$$

5	**while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6	$A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
7	$i = i - 1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$

How about Best Case ??



What is the total cost?

- It can vary based on the value of t_j!

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n} (t_j - 1) + c_7 \sum_{j=2}^{n} (t_j - 1) + c_8(n-1) .$$

5	**while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6	$A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
7	$i = i - 1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$

How about Best Case ?? **t_j = 1 for all j**

$$\begin{aligned} T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5(n-1) + c_8(n-1) \\ &= (c_1 + c_2 + c_4 + c_5 + c_8)n - (c_2 + c_4 + c_5 + c_8) . \end{aligned}$$

T(n) = O(n)

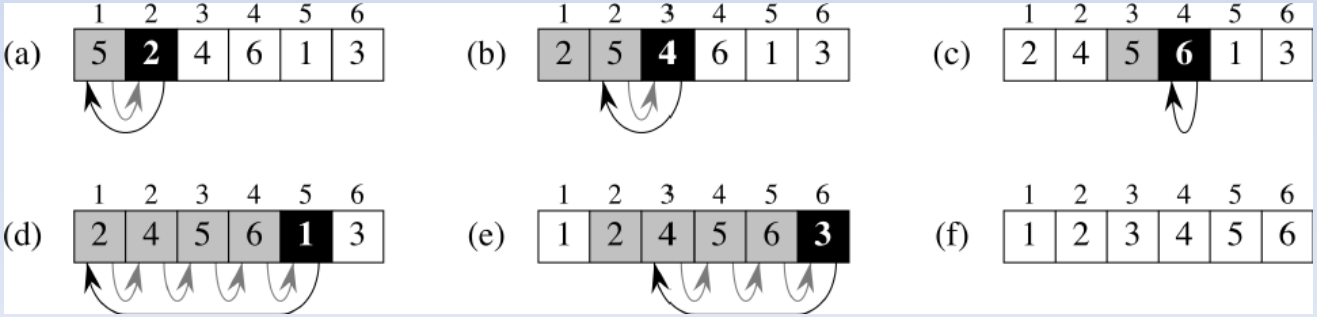
What is the total cost?

- It can vary based on the value of t_j!

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n} (t_j - 1) + c_7 \sum_{j=2}^{n} (t_j - 1) + c_8(n-1) .$$

5	**while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6	$A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
7	$i = i - 1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$

How about Worst Case ??



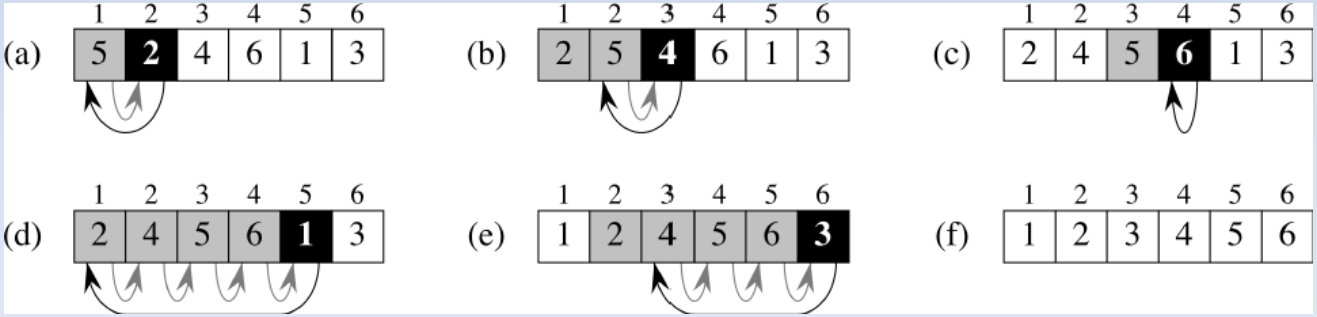
What is the total cost?

- It can vary based on the value of t_j!

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n} (t_j - 1) + c_7 \sum_{j=2}^{n} (t_j - 1) + c_8(n-1) .$$

5	**while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6	$A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
7	$i = i - 1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$

How about Worst Case ?? $\mathbf{t_j = j}$



Analysis w/ Insertion Sort

$$\sum_{j=2}^{n} j = \frac{n(n+1)}{2} - 1$$

and

$$\sum_{j=2}^{n} (j-1) = \frac{n(n-1)}{2}$$

- SO:

$$\begin{aligned} T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5\left(\frac{n(n+1)}{2} - 1\right) \\ &\quad + c_6\left(\frac{n(n-1)}{2}\right) + c_7\left(\frac{n(n-1)}{2}\right) + c_8(n-1) \\ &= \left(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2}\right) n^2 + \left(c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8\right) n \\ &\quad - (c_2 + c_4 + c_5 + c_8) \, . \end{aligned}$$

Analysis w/ Insertion Sort

$$\sum_{j=2}^{n} j = \frac{n(n+1)}{2} - 1$$

and

$$\sum_{j=2}^{n} (j-1) = \frac{n(n-1)}{2}$$

- SO: **$T(n) = O(n^2)$**

$$\begin{aligned} T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5\left(\frac{n(n+1)}{2} - 1\right) \\ &\quad + c_6\left(\frac{n(n-1)}{2}\right) + c_7\left(\frac{n(n-1)}{2}\right) + c_8(n-1) \\ &= \left(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2}\right) n^2 + \left(c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8\right) n \\ &\quad - (c_2 + c_4 + c_5 + c_8) . \end{aligned}$$

ch2.1.py - D:/Documents/GitHub/Fall14/174/Resources/py/ch2.1.py

File Edit Format Run Options Windows Help

```
import random
''' Insertion Sort: Chapter 2 '''

def InsertionSort(A):
    for j in range(1, len(A)):
        key = A[j]
        i = j - 1
        while i >=0 and A[i] > key:
            A[i+1] = A[i]
            i -= 1
        A[i+1] = key

x = [i for i in range(1,25)]
random.shuffle(x)
print x
InsertionSort(x)
print x
```

Ln: 14 Col: 17

Divide & Conquer

- We'll get back to this later…

Chapter 3 :
Growth of Functions

- Set of functions:

O(g(n)) =

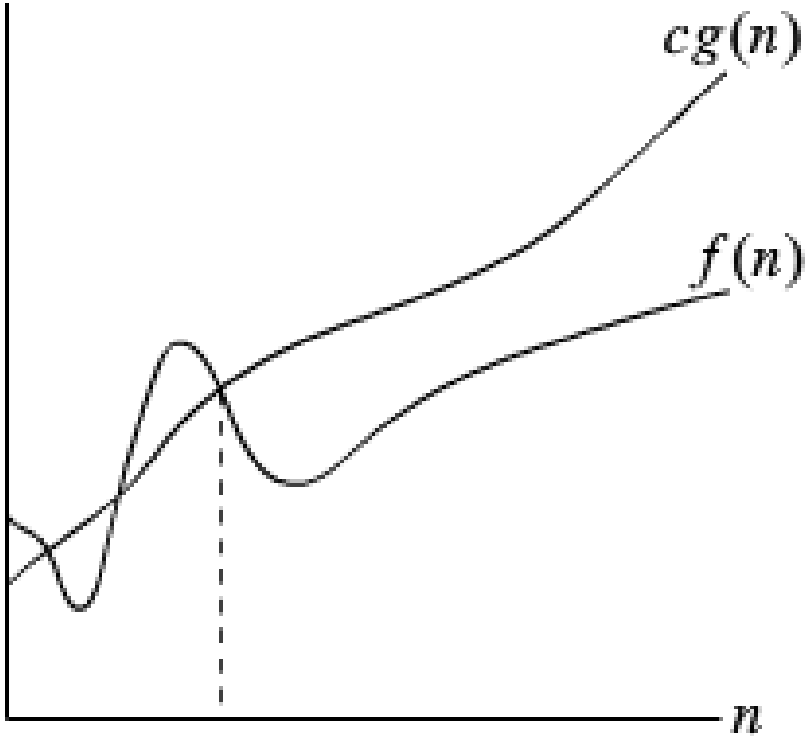
{

f(n) : there exist positive constants c and n_0 such that:

$$0 \leq f(n) \leq cg(n)$$
$$for\ all\ n \geq n_0$$

}

f(n) = O(g(n)



$f(n) = O(g(n))$

(b)

Chapter 3 :
Growth of Functions

- Set of functions:

Θ(g(n)) =

{

f(n) : there exist positive constants c_1, c_2, and n_0 such that:

$$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$$
$$for\ all\ n \geq n_0$$

}

Chapter 3 :
Growth of Functions

- Set of functions:

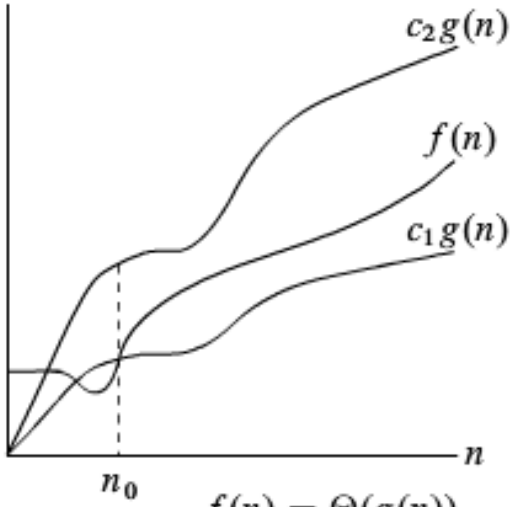
Ω(g(n)) =

{

f(n) : there exist positive constants c and n_0 such that:

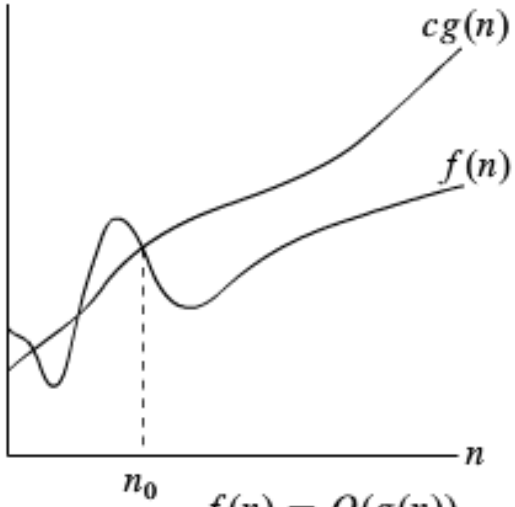
$$0 \leq cg(n) \leq f(n)$$
$$for\ all\ n \geq n_0$$

}



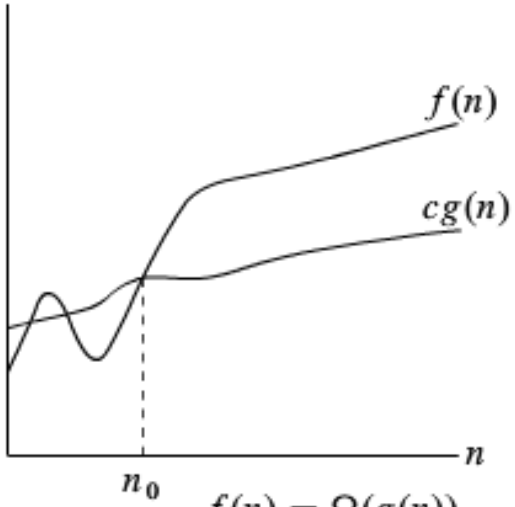
$f(n) = \Theta(g(n))$

(a)



$f(n) = O(g(n))$

(b)



$f(n) = \Omega(g(n))$

(c)

Theorem 3.1

Theorem 3.1

For any two functions $f(n)$ and $g(n)$, we have $f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$. ■

Question

- IF: f(n) = O(g(n)).
- IS: $f(n) * \log_2(f(n)^c) = O(\ g(n) * \log_2\big(g(n)\big)\ )$

Question

- IF: f(n) = O(g(n)).
- IS: $f(n) * \log_2(f(n)^c) = O(\ g(n) * \log_2\big(g(n)\big)\ )$

$$f(n) * \log_2(f(n)^c) = f(n) * c\log_2(f(n))$$
$$= O(\ g(n) * c\log_2\big(g(n)\big)\ )$$
$$= O(\ g(n) * \log_2\big(g(n)\big)\ )$$

o-notation: definitely NOT asymptotically tight

o(g(n)) =

{

f(n) : for ANY positive constant c>0, there exists a constant n_0>0 such that $0 \leq f(n) \leq cg(n)$ for all n > n_0

}

o-notation: definitely NOT asymptotically tight

o(g(n)) =

f(n) : ANY positive constant c>0, there exists a constant n_0>0 such that

$$0 \leq f(n) \leq cg(n)$$
$$for\ all\ n \geq n_0$$

- Versus

O(g(n)) =

f(n) : EXISTS positive constants c and n_0 such that:

$$0 \leq f(n) \leq cg(n)$$
$$for\ all\ n \geq n_0$$

3.2: Standard Notations & Common Functions

- modular arithmetic!

$$a \bmod n = a - n\lfloor a/n \rfloor$$

$$0 \leq a \bmod n < \mathrm{n}$$

- (a mod n) = (b mod n) then we write

$$a \equiv b \pmod{n}$$

  - a is equivalent to b, modulo n.

3.2: Standard Notations & Common Functions

- Logarithms

$$\log_b a = \frac{\log_c a}{\log_c b}$$

  - $\log_c b$ is a constant!

3.2: Standard Notations & Common Functions

- Logarithms

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \frac{x^5}{5} \cdots$$

$$\frac{x}{1+x} \leq \ln(1+x) \leq x \text{ , for x} > -1$$

3.2: Standard Notations & Common Functions

- Fibonacci Numbers
  - Have many kewl properties.
  - Talk more about these later.

$$F_0 = 0,$$
$$F_1 = 0,$$
$$F_i = F_{i-1} + F_{i-2}, \text{ for } i \geq 2$$

Fibonacci

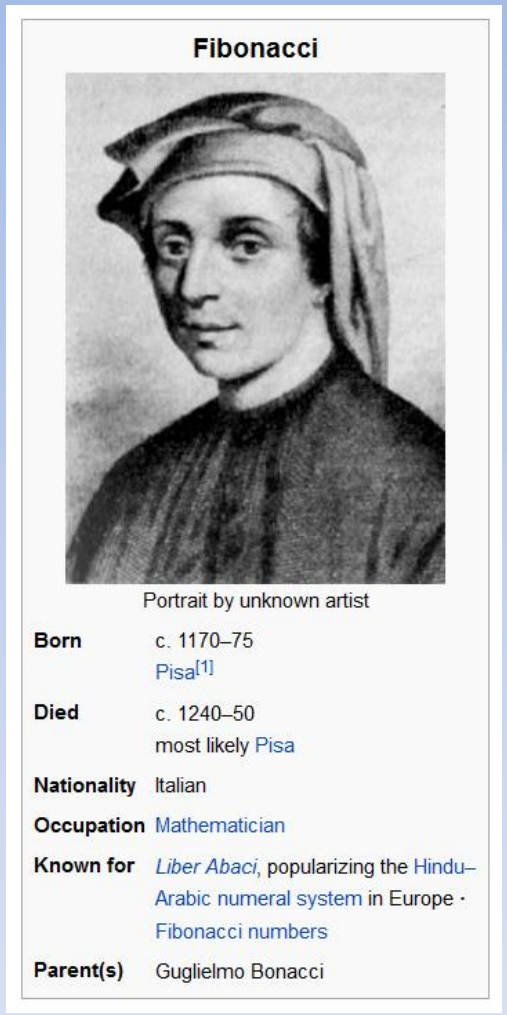
Portrait by unknown artist

Born	c. 1170–75 Pisa[1]
Died	c. 1240–50 most likely Pisa
Nationality	Italian
Occupation	Mathematician
Known for	*Liber Abaci*, popularizing the Hindu–Arabic numeral system in Europe · Fibonacci numbers
Parent(s)	Guglielmo Bonacci

Leonardo Bonacci
(c. 1170 – c. 1250)

3.2: Standard Notations & Common Functions

- Fibonacci Numbers & Golden Ratio
  - Fibonacci numbers are related to golden ratio ϕ
  - ϕ and $\widehat{\phi}$ are the two roots the equation $x^2 = x + 1$
    - This property can be used in very interesting ways w/ mathematical analysis.
  - $\phi = \frac{1+\sqrt{5}}{2}, \widehat{\phi} = \frac{1-\sqrt{5}}{2}$



Leonardo Bonacci
(c. 1170 – c. 1250)

Fibonacci Numbers & Golden Ratio

- Turns out there is a direct relationship between the golden ratio and the number in the Fibonacci Sequence!
- i^{th}number of the Fibonacci Sequence will be designated F_i
- $F_i = \frac{\Phi^i - \widehat{\Phi}^i}{\sqrt{5}}$, Proof is by Induction

Fibonacci Numbers & Golden Ratio

- $F_i = \frac{\phi^i - \hat{\phi}^i}{\sqrt{5}}$
- $\frac{|\hat{\phi}^i|}{\sqrt{5}} < \frac{1}{\sqrt{5}} < \frac{1}{2}$, since $|\hat{\phi}^i| \approx 0.61803\ldots < 1$

$$F_i = \left\lfloor \frac{\phi^i}{\sqrt{5}} + \frac{1}{2} \right\rfloor$$

- $F_i = \frac{\phi^i}{\sqrt{5}}$ rounded to the nearest integers SO:
- Fibonacci numbers grow exponentially!

Git & Github

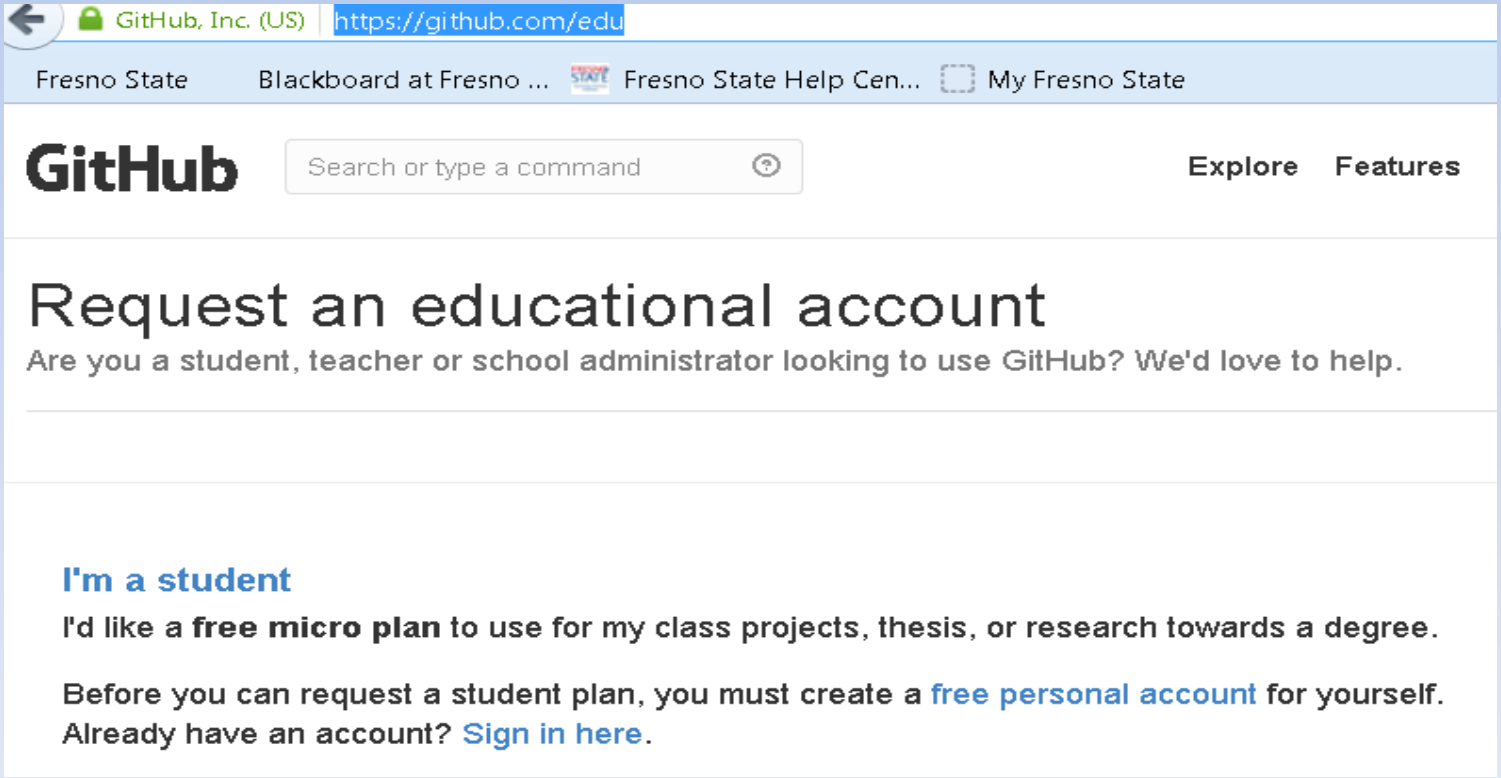
- Introduce Some Technology

GitHub

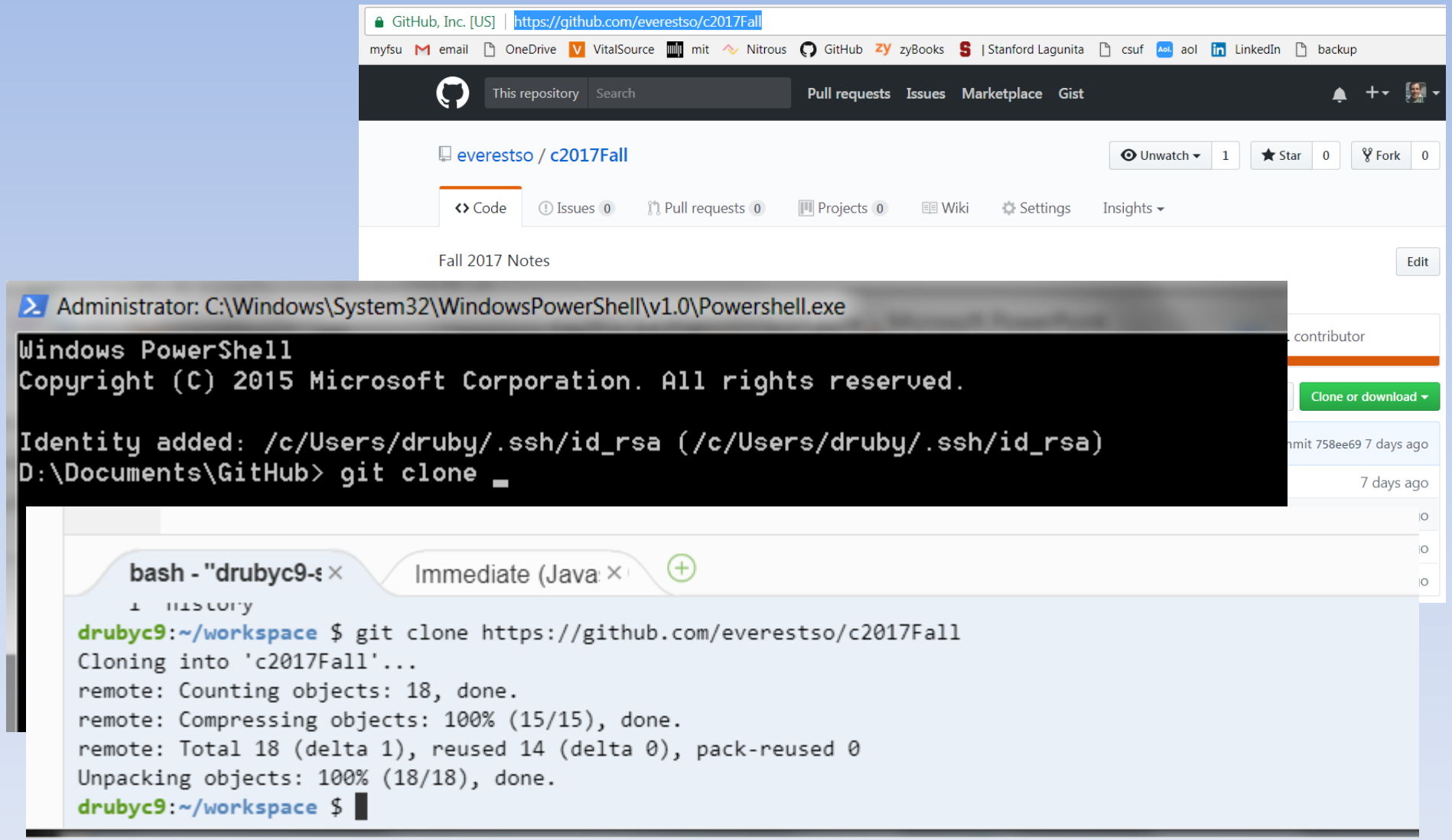
- Popular Code Repository for Open-Source Projects
- Free Student Accounts
- In-Class: Code uploaded for class use.
- Student Projects: Interviews/Personal Development
- Great repository when looking for code ideas.

GitHub
github.com

- Consider requesting student account
  - https://Github.com/edu

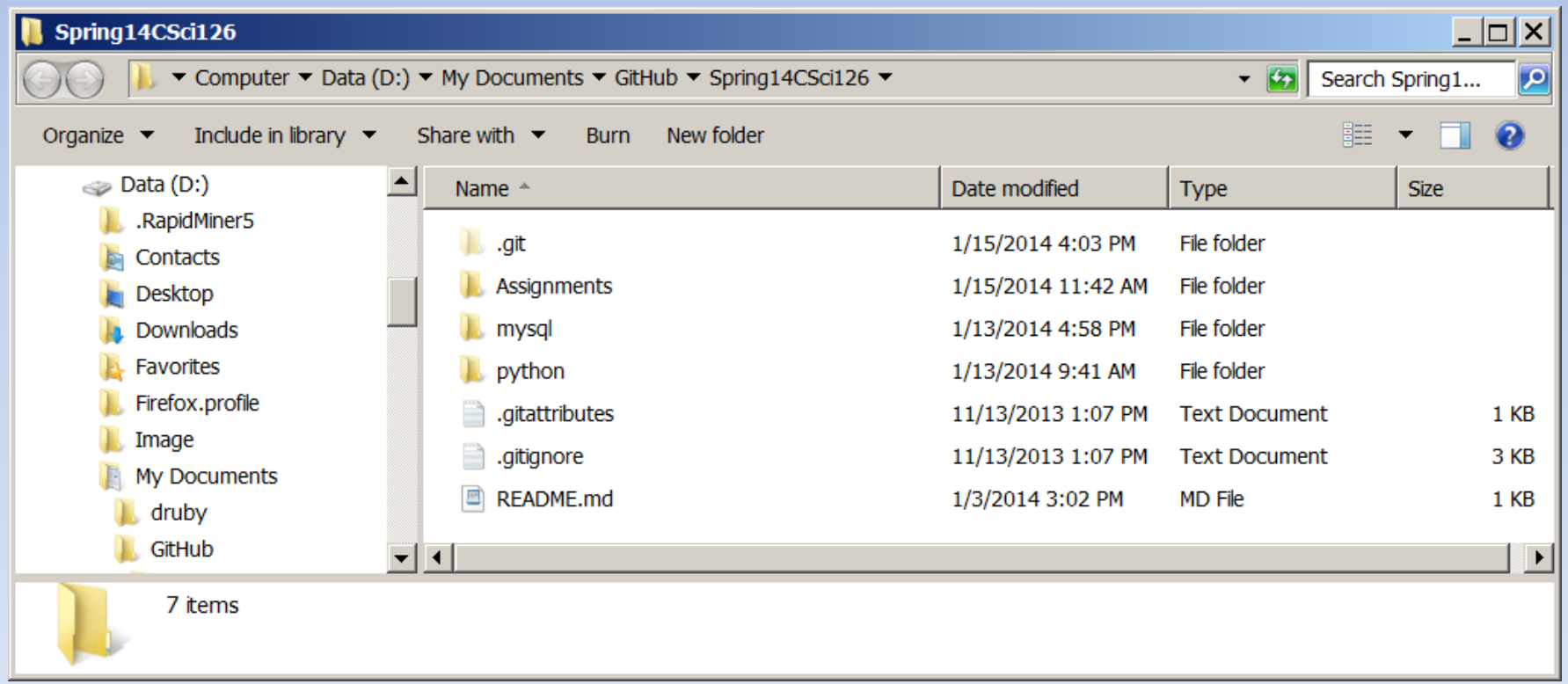


Using Git


GitHub, Inc. [US] https://github.com/everestso/c2017Fall
myfsu email OneDrive VitalSource mit Nitrous GitHub zyBooks | Stanford Lagunita csuf aol LinkedIn backup
This repository Search Pull requests Issues Marketplace Gist
everestso / c2017Fall
Unwatch 1 Star 0 Fork 0
Code Issues 0 Pull requests 0 Projects 0 Wiki Settings Insights
Fall 2017 Notes
Edit
Administrator: C:\Windows\System32\WindowsPowerShell\v1.0\Powershell.exe
Windows PowerShell
Copyright (C) 2015 Microsoft Corporation. All rights reserved.
Identity added: /c/Users/druby/.ssh/id_rsa (/c/Users/druby/.ssh/id_rsa)
D:\Documents\GitHub> git clone
contributor
Clone or download
bash - "drubyc9-s
Immediate (Java
drubyc9:~/workspace $ git clone https://github.com/everestso/c2017Fall
Cloning into 'c2017Fall'...
remote: Counting objects: 18, done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 18 (delta 1), reused 14 (delta 0), pack-reused 0
Unpacking objects: 100% (18/18), done.
drubyc9:~/workspace $

Using Git

- Directories loaded onto your system after Cloned In!



More Git

- http://proquest.safaribooksonline.com/video/software-engineering-and-development/version-control/9781771372473#toc

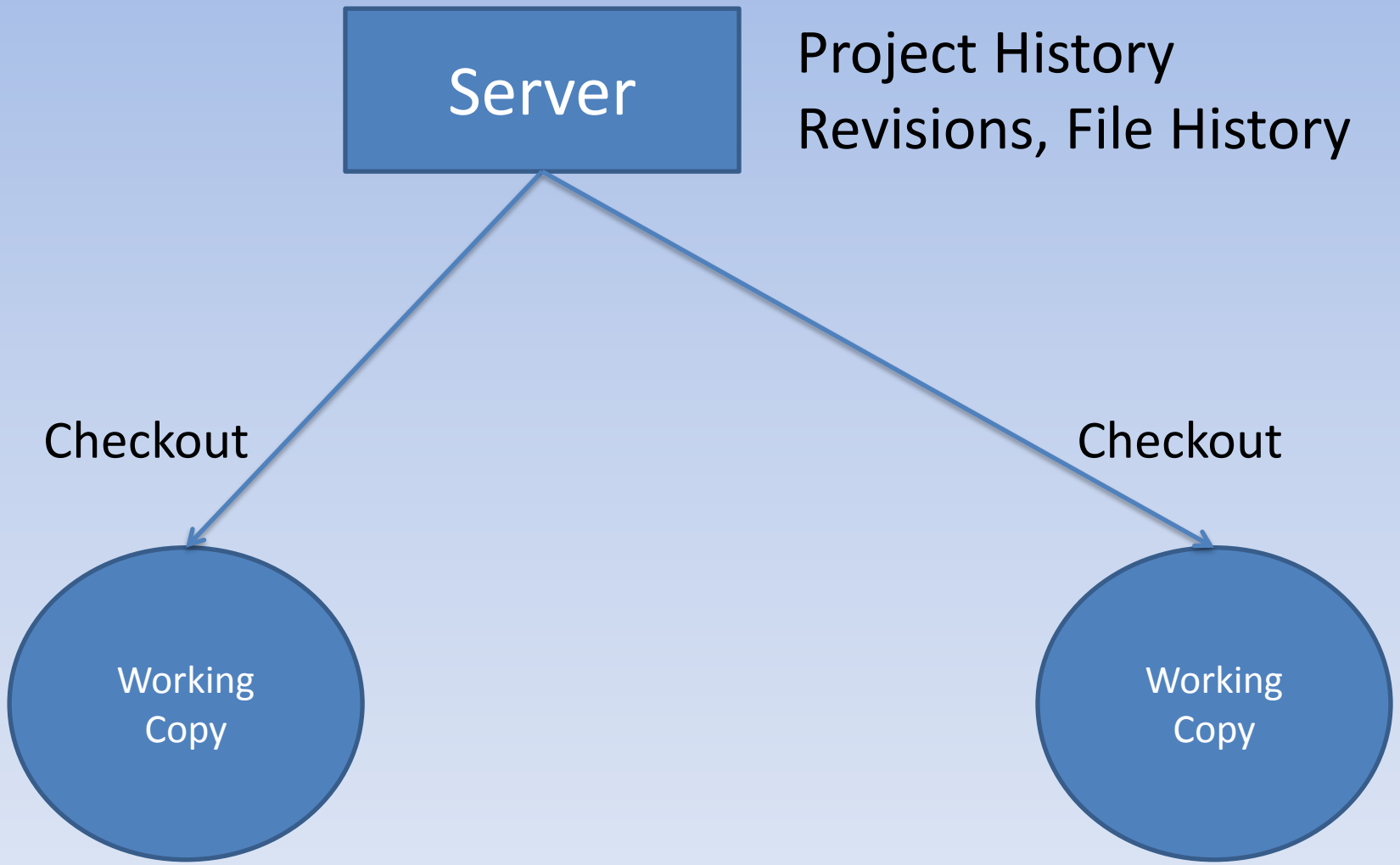
More Git

- Source Control & Git
- Git Basics
- Repository Operations
- Social Workflows with GitHub

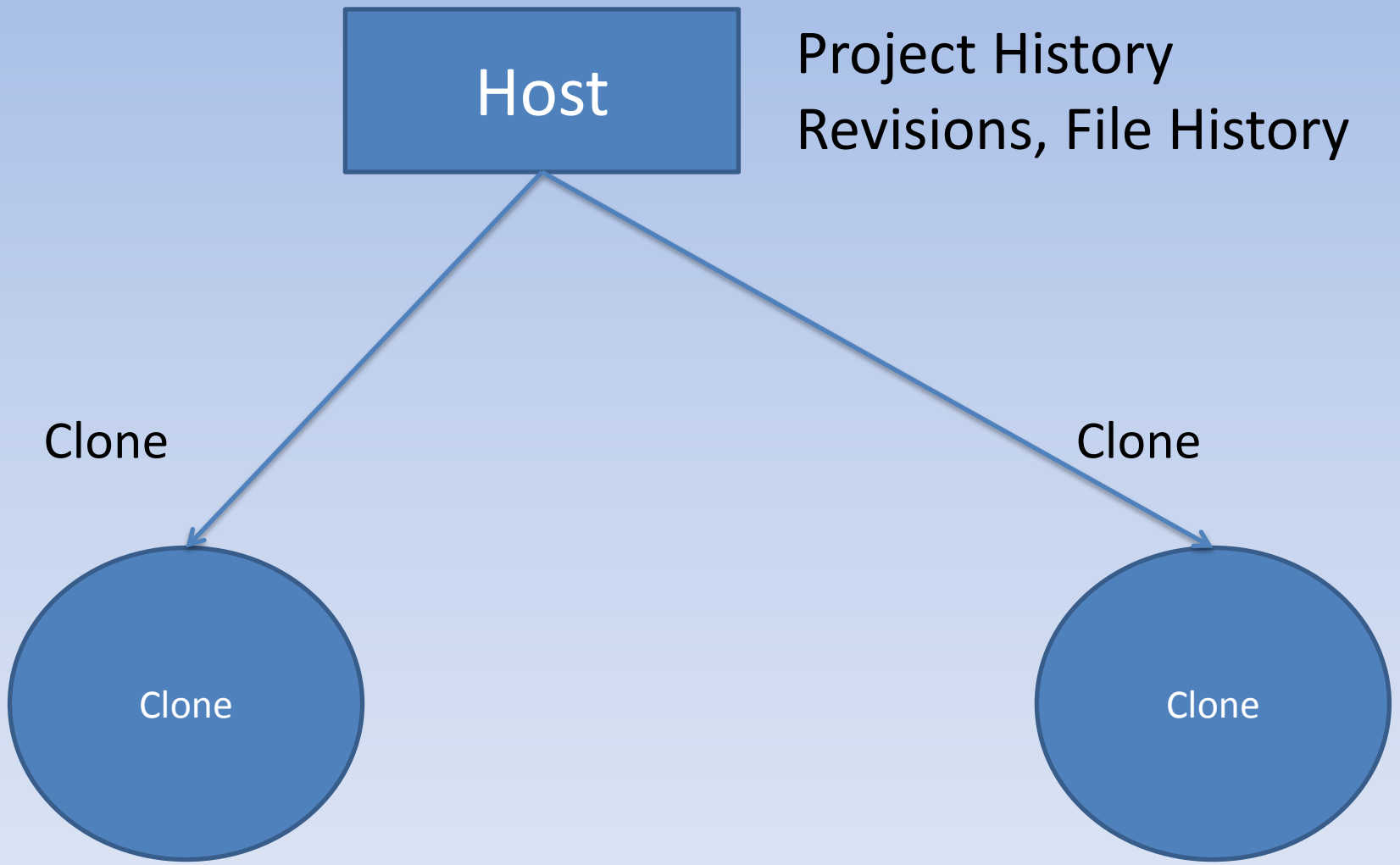
Managing Source Code

- Source code involves a lot of files
- Especially when working with a team, managing files is challenge.
- A tool that manages and tracks different versions of software or other content is called:
  - Version Control System (VCS)
  - Source Code Manager (SCM)
  - Revision Control System (RCS)

Traditional Source Control



Git


Host
Project History
Revisions, File History
Clone
Clone
Clone
Clone

Git History

- Git was invented by Linus Torvalds to support development of Linux Kernel in April 2005.

Key Dates in VCS History

- Source Code Control System (SCCS) 1970s
- Revision Control System (RCS) 1980s
- CVS (Concurrent Version System): 1986
  - Very popular
- Subversion (SVN): 2001
  - Added better support for branches
- BitKeeper (May 2000) and Mercurial (April 2005) went in a different direction
  - Eliminated central repository
  - Each developer has a shareable copy of the project
  - Peer-To-Peer Model
  - Git is derived from this approach.

Motivation

- ERROR:
- ‘You have an issue with your repository’
- ‘Open Git Shell and DEBUG!’

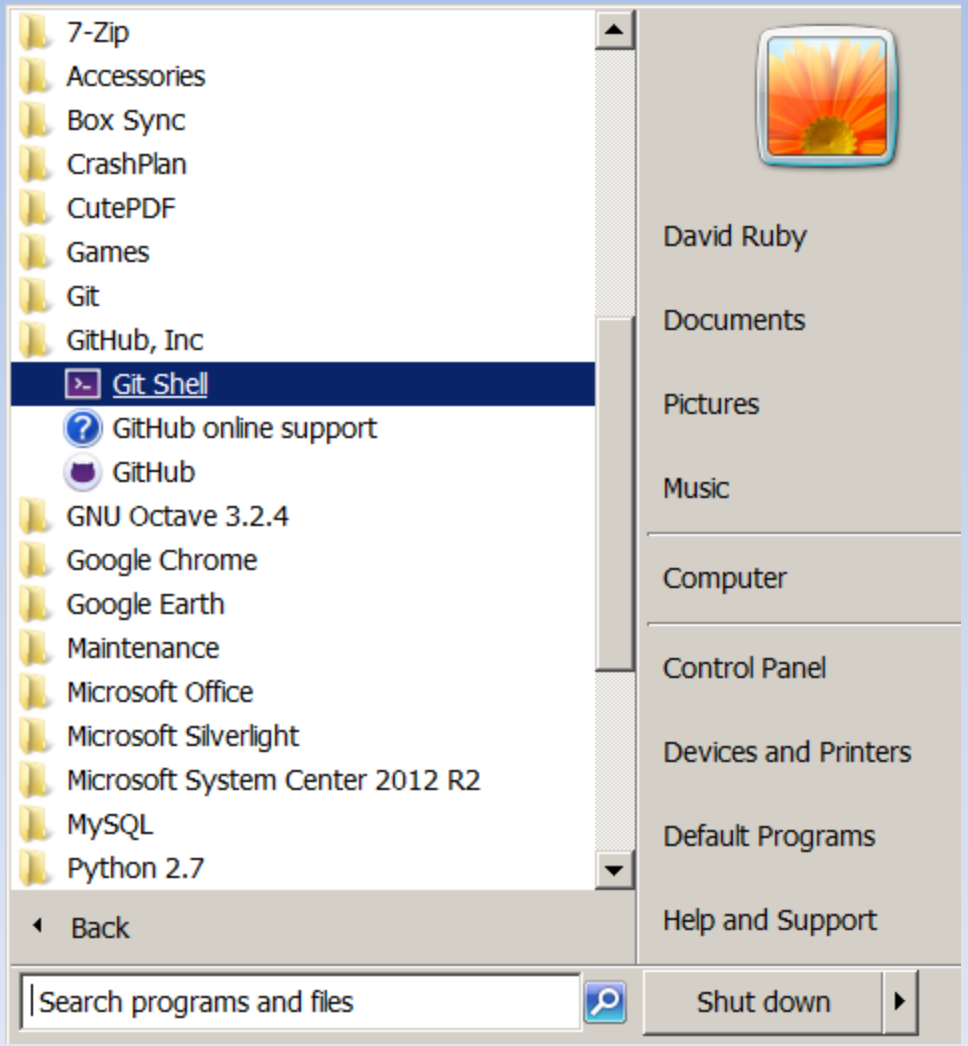
- OH NO!

Git Basic Concepts

- Repository
  - A database of all the information needed to retain and manage the revisions and history of a project.
  - A set of configuration value are kept within each repository.
  - Object Store and Index are two primary data structures for Repository
- Object Store
  - Contains the original data files and all the log messages, author information, dates.
  - Contains all info needed to rebuild any version or branch of the project.
- Index
  - Temporary dynamic file that allows stage changes before a commit.

Git

- Git Shell



Git Configuration

- git config --list
- git help git
- git help git-config
- git config --help

- git config --global user.name “John Doe100”
- git config --global user.email ‘JD@test.com’

New Repository

- Create a directory for the new repository
  - `mkdir ~/MyRepo`
  - `cd ~/MyRepo`
  - `echo 'Test Web Site!' > index.html`
- To turn *~/public_html* or any directory into a Git repository, run *git init*:
  - `git init`
  - Initialized empty Git repository in .git/
- To signify that your directory is a Git repository
  - *git init* command creates a hidden directory, called *.git*, at the top level of your project.
  - CVS and SVN place revision information in *CVS* and *.svn* subdirectories within each of your project's directories,
  - Git places all its revision information in this one, top-level *.git* directory.
- Everything in your *~/MyRepo* directory remains untouched.
- Git considers it your project's *working directory*,
- In contrast, the repository hidden within *.git* is maintained by Git.

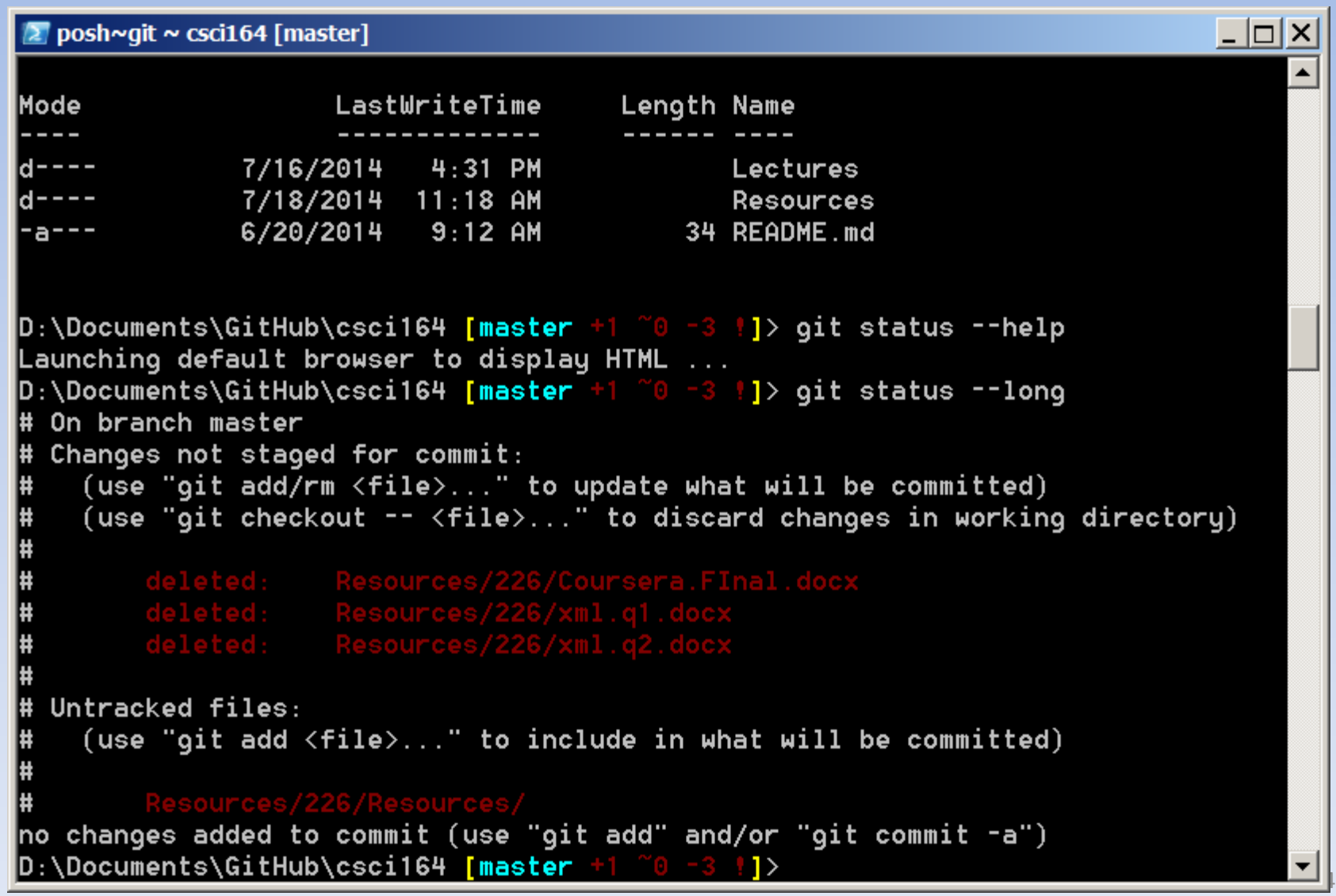
Making Changes

- git add .
  - Adds all new files
- git add –u
  - Updates tracking for files that changed names or were deleted
- git add –A
  - Does both of the previous
- Git Add update the Index Repository Structure

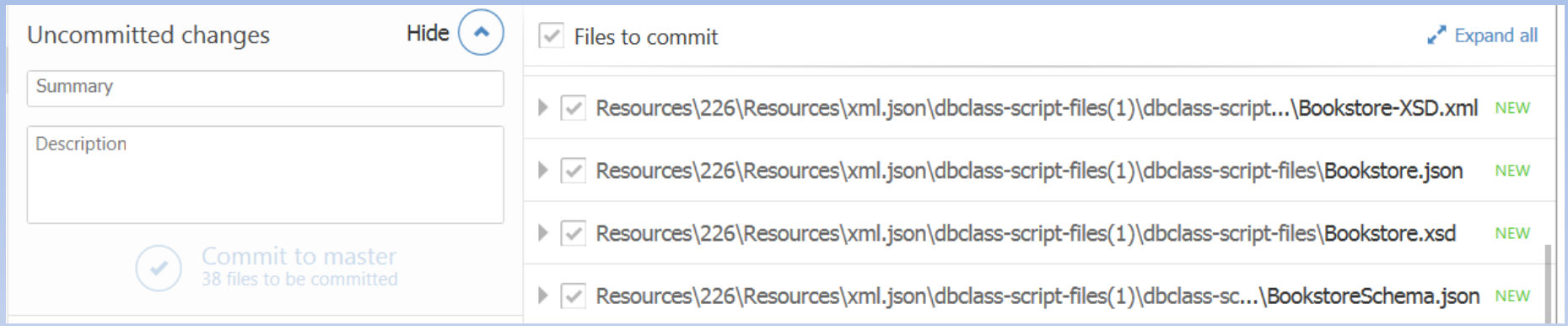
Git Status

- git status
  - Show the working tree status
- After making changes to files GitHub for Windows will show uncommitted changes.
- Git Status will indicate changes are not staged.

Git Status


posh~git ~ csci164 [master]
Mode LastWriteTime Length Name
d---- 7/16/2014 4:31 PM Lectures
d---- 7/18/2014 11:18 AM Resources
-a--- 6/20/2014 9:12 AM 34 README.md
D:\Documents\GitHub\csci164 [master +1 ~0 -3 !]> git status --help
Launching default browser to display HTML ...
D:\Documents\GitHub\csci164 [master +1 ~0 -3 !]> git status --long
On branch master
Changes not staged for commit:
(use "git add/rm <file>..." to update what will be committed)
(use "git checkout -- <file>..." to discard changes in working directory)
deleted: Resources/226/Coursera.FInal.docx
deleted: Resources/226/xml.q1.docx
deleted: Resources/226/xml.q2.docx
Untracked files:
(use "git add <file>..." to include in what will be committed)
Resources/226/Resources/
no changes added to commit (use "git add" and/or "git commit -a")
D:\Documents\GitHub\csci164 [master +1 ~0 -3 !]>

GitHub for Windows



Git Commit

- git commit
  - Will record changes to repository.
- git log
  - Shows all prior commits
- git push
  - Pushes changes to remote repository.

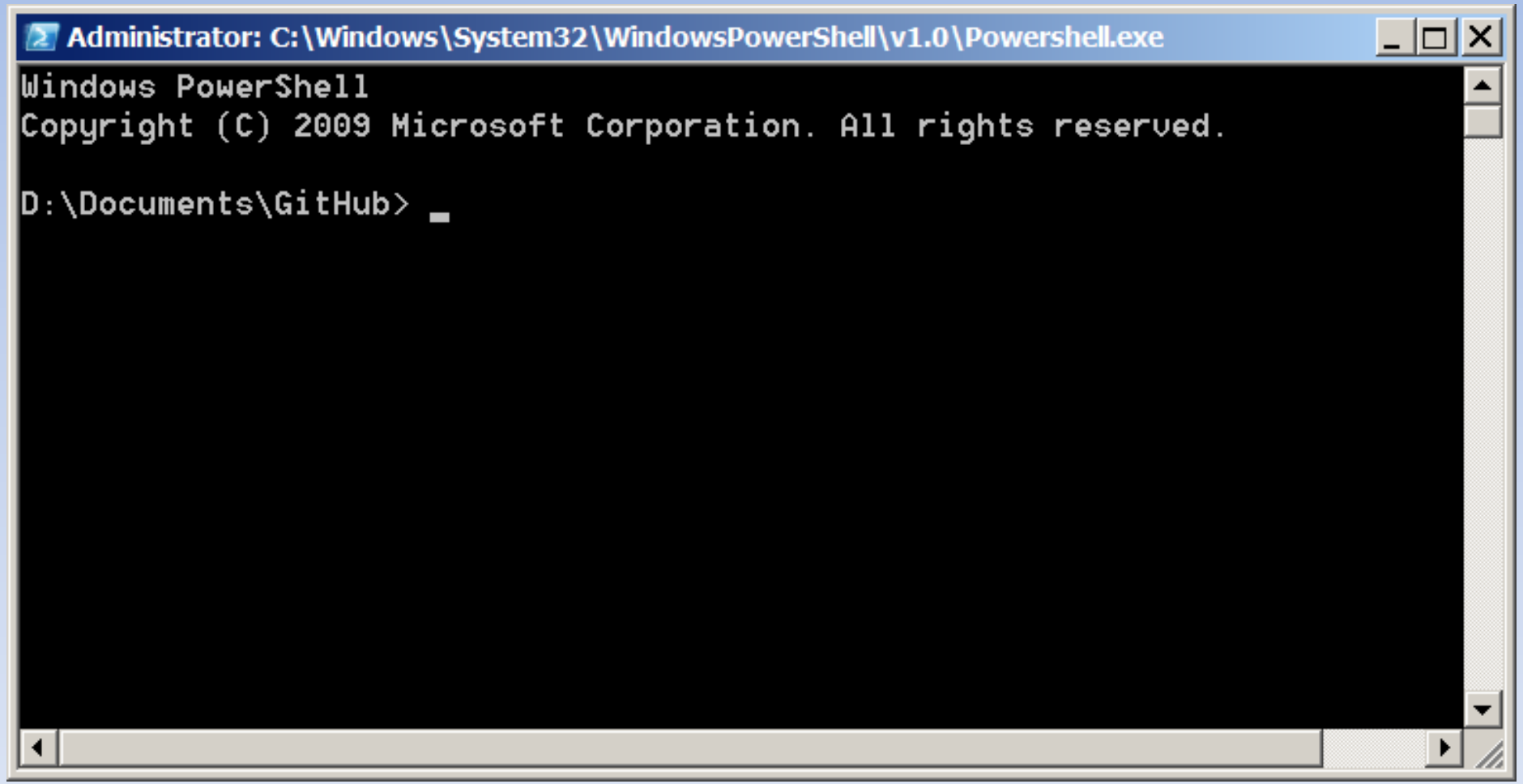
My Error

- GitHub kept failing to push changes.
  - Kept generating an error to debug repository status.
- Undid commit and retried the process.
  - Lost data
  - Recovered data
- Finally walked through each step of the staging process in shell
- Final stage: Git Push
  - Git Push generated an error from the shell I hadn't seen before.
  - File size limit.
  - Deleted the large file, and re-staged.
  - Push to remote server worked.

Another Example

- Was working on a local Git Repository
- Actually started it online, but wanted to take if offline a while
- Continue it offline
- Then wanted to push the new version online.

Showing Your Remotes



Showing Your Remotes

Showing Your Remotes

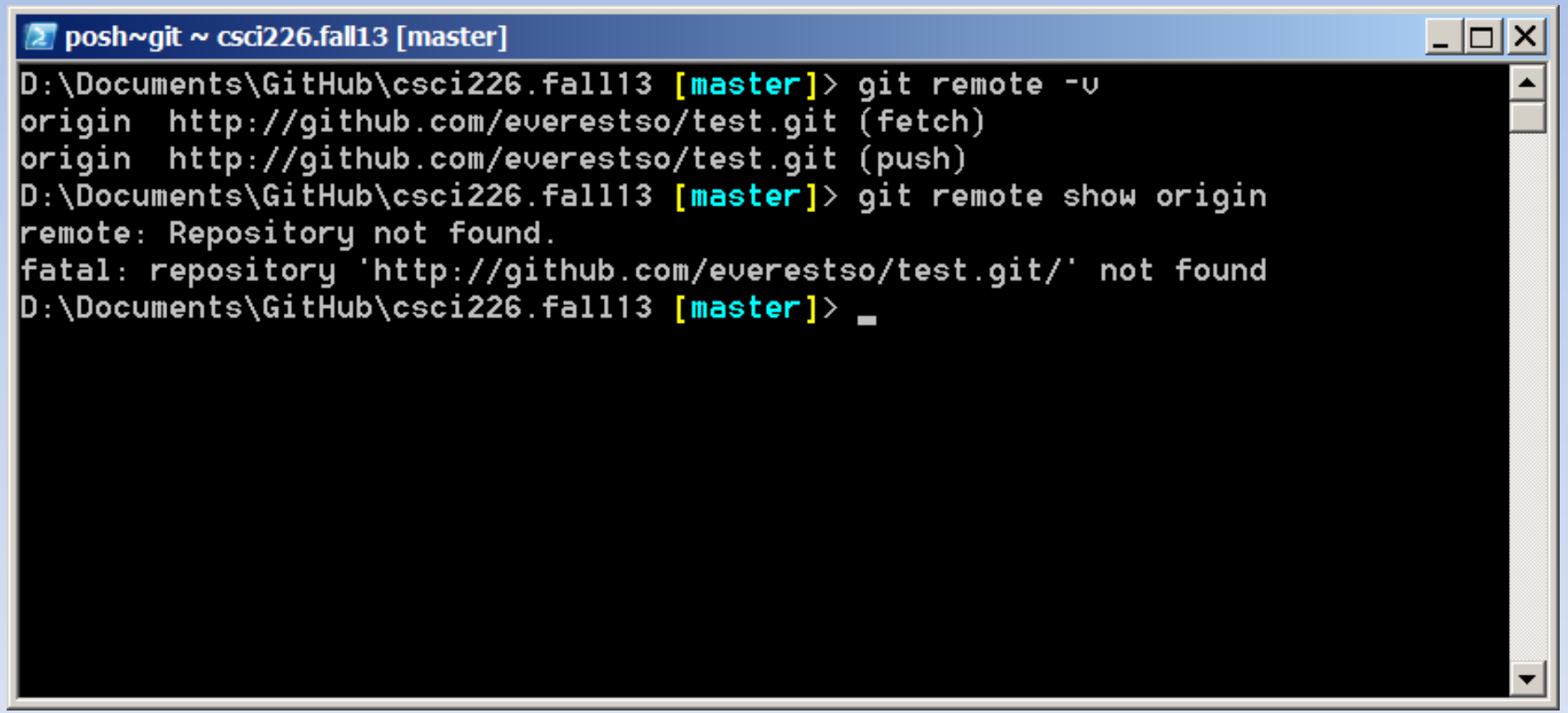
- git remote -v

Changing Remote URL

- git remote set-url origin
  - https//github.com/everestso/name.git

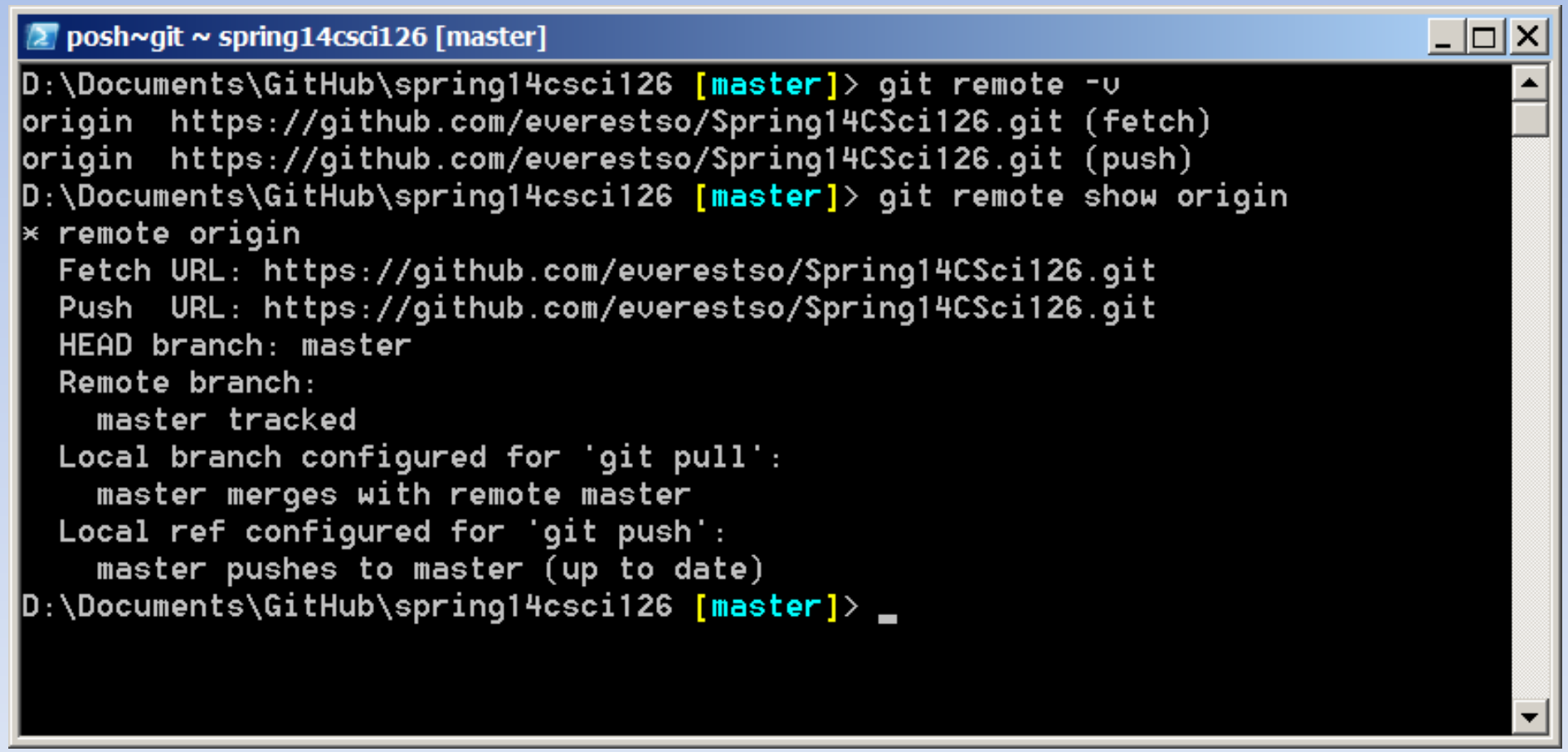
Check the Status

- git remote show origin



Check the Status

- git remote show origin



Python

- Readable
- Garbage Collected
- Dynamically Typed

Courtesy WikiPedia

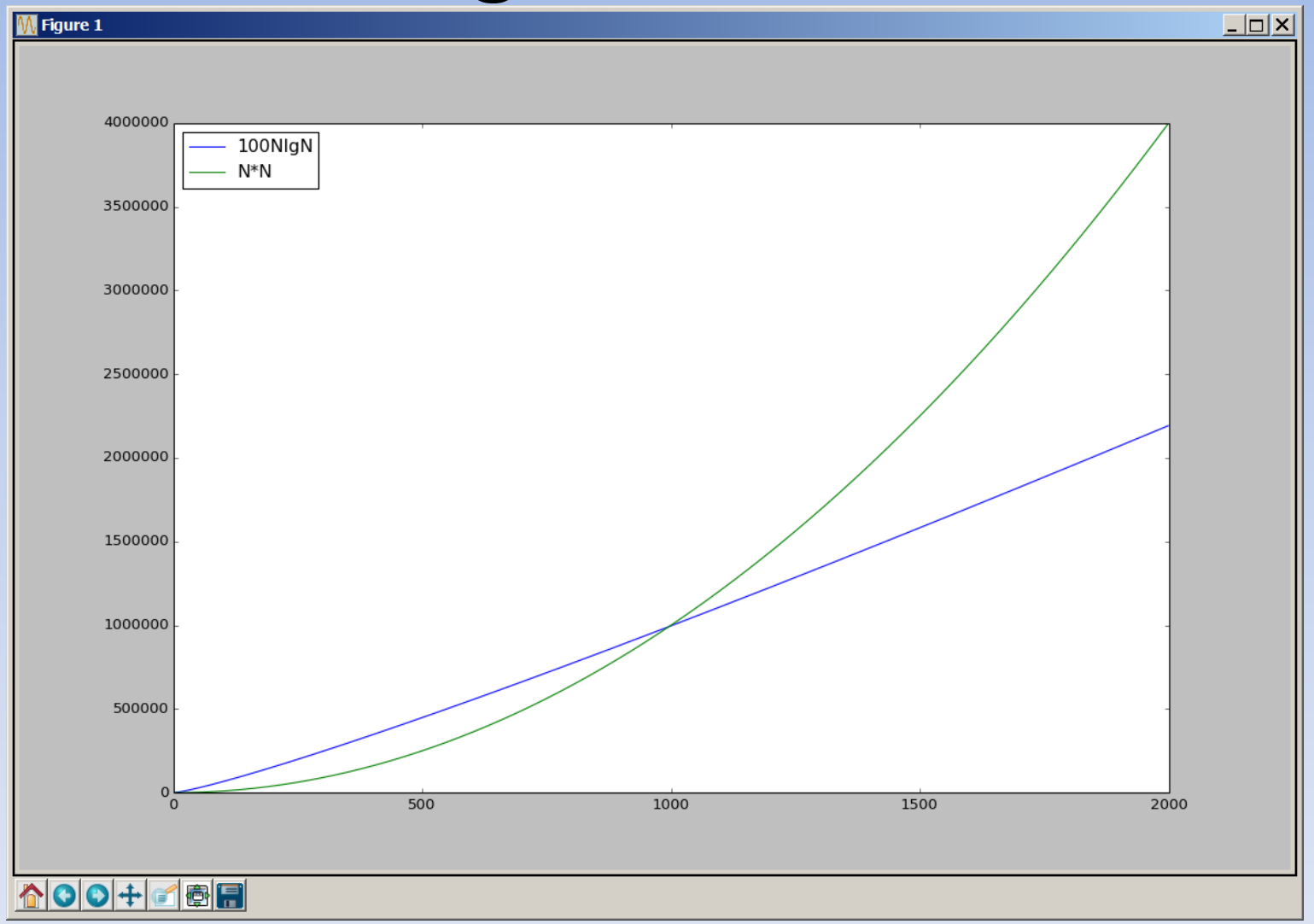
- Conceived in late 80s, with implementation beginning in December ‘89 by Guido Van Rossum @ CWI in Netherlands.
- Guido Van Rossum is Python's principal author, and his continuing central role in deciding the direction of Python is reflected in the title given to him by the Python community, BENEVOLENT DICTATOR FOR LIFE (BDFL).
- Python 2.0 was released on 16 October 2000, with many major new features including a full garbage collector and support for Unicode. With this release the development process was changed and became more transparent and community-backed.
- Python 3.0 (also called Python 3000 or py3k), a major, backwards-incompatible release, was released on 3 December 2008 after a long period of testing. Many of its major features have been backported to the backwards-compatible Python 2.6 and 2.7.

Size of problem solvable in time N w/ O(n), O(NlgN), O(N^2)

```
24  def Sizeit(fn, n):
25      i = 1
26      k = 10
27      while (fn(i)< n):
28          i += k
29          if ( round(math.log(i,10)) > math.log(k) ):
30              k *= k
31      return i
```

- [1000000, 60000000, 3600000000L, 86400000000L, 2592000000000L, 31536000000000L, 315360000000000L]
- N*N: 1021; 7821; **61,721**; 301721; 1611721; 5621721; 56,161,721
- NlgN: 71721; 2801721; **133,381,721**, 2755151721; 71962281,721; 797662281721; 68610962281721
- N: 1,001,721; 60,001,721; **3,662,281,721**; 86462281721; 2592062281721; 31536062281721; 315360062281721;

100NlgN versus N*N



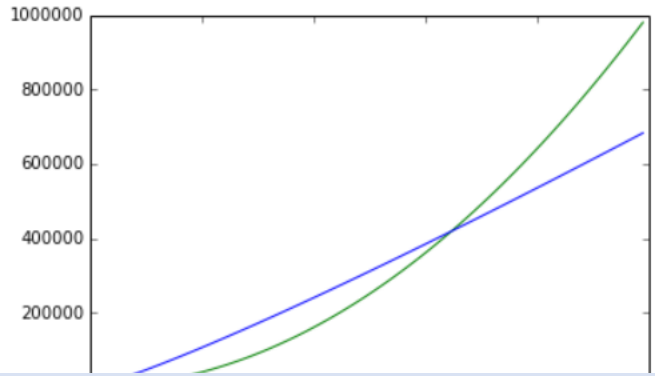
iPython Notebook

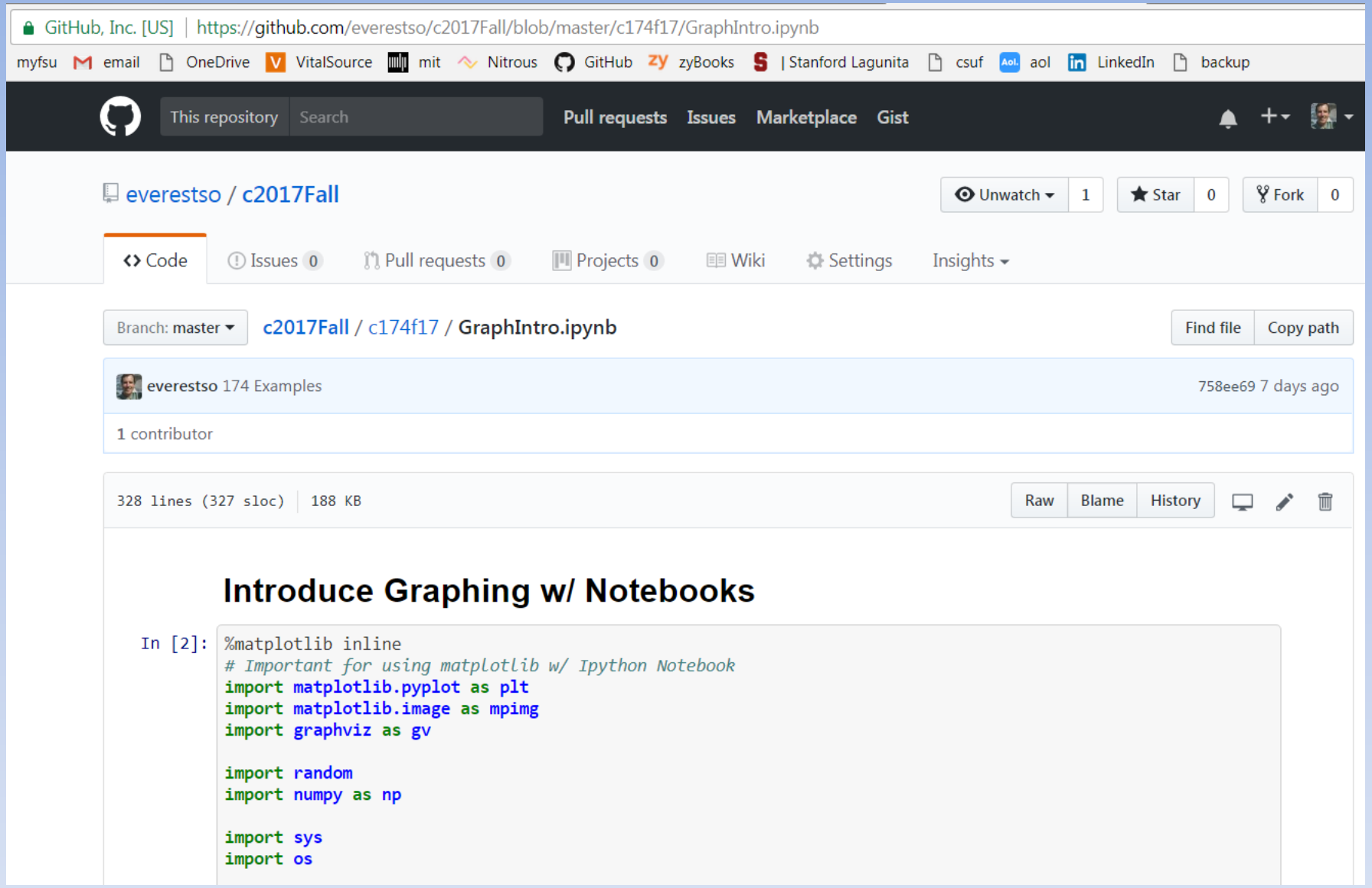
Growth of Functions

This notebook illustrates a growth of functions

In [4]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import math
```

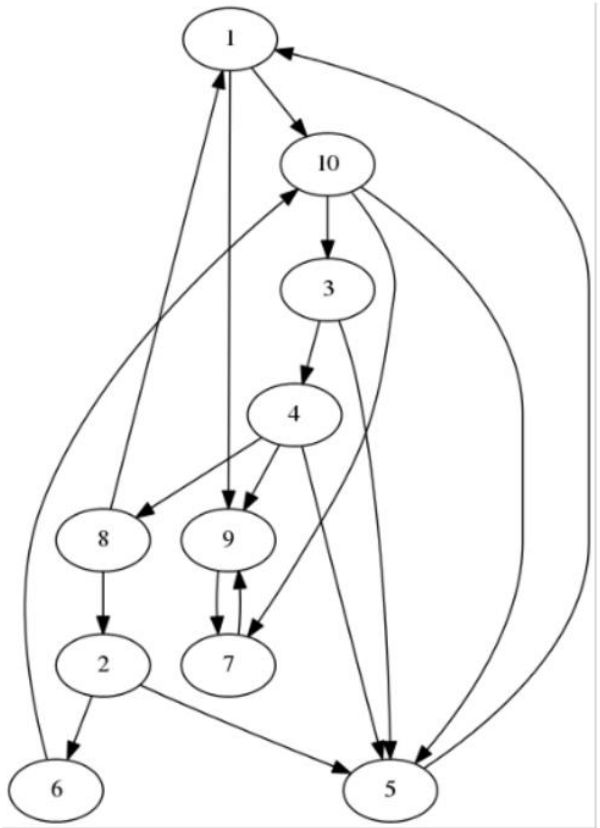
In [11]:
```
x=[i for i in range(10,1000, 10)]
y = [n*n for n in x]
y2 = [100*n*math.log(n) for n in x]
plt.plot(x, y,'g-')
plt.plot(x, y2,'b-')
plt.show()
```



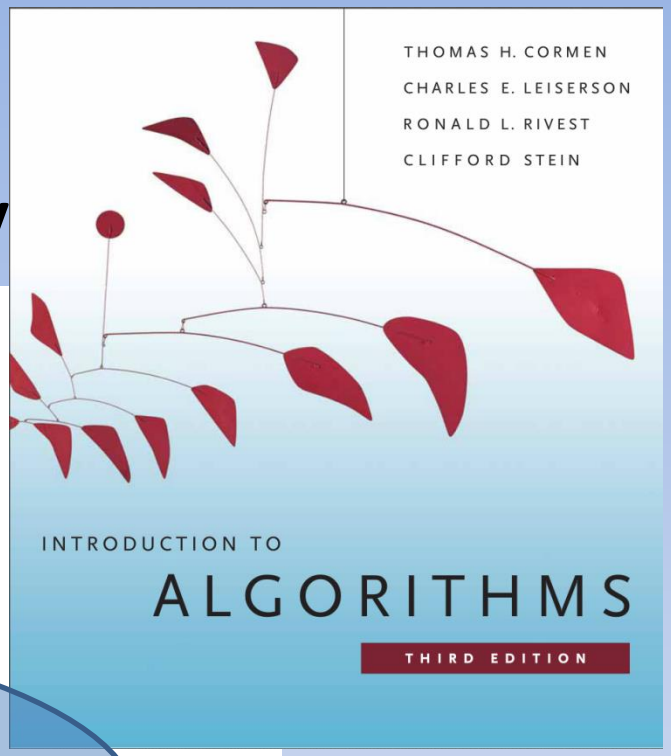

GitHub, Inc. [US] | https://github.com/everestso/c2017Fall/blob/master/c174f17/GraphIntro.ipynb
myfsu email OneDrive VitalSource mit Nitrous GitHub zyBooks | Stanford Lagunita csuf aol LinkedIn backup
This repository Search Pull requests Issues Marketplace Gist
everestso / c2017Fall
Unwatch 1 Star 0 Fork 0
Code Issues 0 Pull requests 0 Projects 0 Wiki Settings Insights
Branch: master c2017Fall / c174f17 / GraphIntro.ipynb
Find file Copy path
everestso 174 Examples
758ee69 7 days ago
1 contributor
328 lines (327 sloc) 188 KB
Raw Blame History
Introduce Graphing w/ Notebooks
In [2]:
%matplotlib inline
# Important for using matplotlib w/ Ipython Notebook
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import graphviz as gv
import random
import numpy as np
import sys
import os

```
def GraphIt(V, E, w={}, D=True, C={}):
    'Plots a graph'
    G = gv.Digraph(filename='graph', format='png') if D else gv.Graph(filename='graph', format='png')

    for v in V:
        G.node(str(v))
    for u in E:
        for v in E.get(u, []):
            if not D and u>v: continue
            if (u,v) in C or ((v,u) in C and not D):
                G.edge(str(u),str(v), label = str(w.get((u,v), '')), color='blue')
            else:
                G.edge(str(u),str(v), label = str(w.get((u,v), '')))
    G.render()
    img=mpimg.imread('graph.png')

    fig, ax = plt.subplots(frameon=False)
    fig.set_size_inches(7,7)
    imgplot = plt.imshow(img)

    ax.axis('off')
    plt.box(on=None)
    plt.show()
    print "Vertex: Edge List"
    for u in V:
        print u, ": ", E.get(u, [])
```

```
5]: V, E = RandomEdgePermute(10, 0.2)
    GraphIt(V, E)
```



Chapter 33
Computational Geometry



Sign of the Crossproduct:
Clockwise versus Counterclockwise

- Sign of the Crossproduct p_1 X p_2 is positive, then p_1 is counterclockwise from p_2.
- Sign of the Crossproduct p_1 X p_2 is 0, then p_1 and p_2 are colinear.
- Dark Region contains vectors that are CounterClockwise with respect to P
  - Sign of the Crossproduct Negative