CSE332: Data Abstractions

Section 2

HyeIn Kim

Spring 2013

Section Agenda

- Bugs & Testing
- Induction Review
- Recurrence Relations
- Asymptotic Analysis
- Homework Tips & Questions

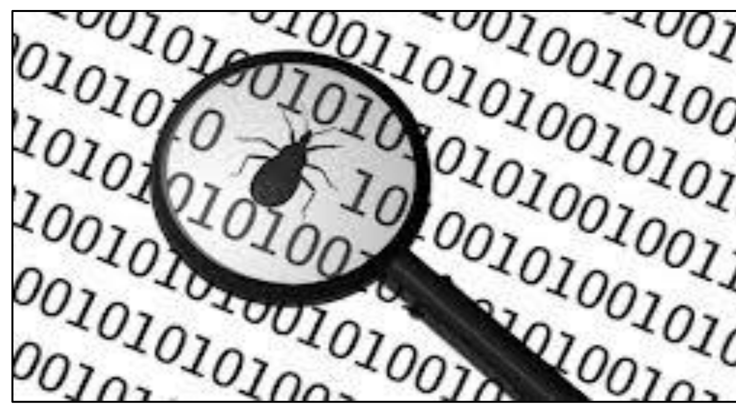
Bugs & Testing

Bugs & Testing

- **Software Bugs**

- Error in a computer program

- Causes program to behave in unexpected ways



Bugs & Testing

- **Why Testing?**

Bugs can be costly

- Cost points in homework
- Can cost $$$ and even life (Therac-25)

Interesting Bug References

- List of bugs http://en.wikipedia.org/wiki/List_of_software_bugs
- History's worst http://www.wired.com/software/coolapps/news/2005/11/69355?currentPage=all
- Bugs of the month http://www.gimpel.com/html/bugs.htm

Bugs & Testing

- **Reverse.java**
  does not test your stack!!

  - Stack can still have lots of bugs
    when working perfectly with Reverse.java

  - Some extreme case in past quarter:
    it only worked with Reverse.java
    (Not a good stack!)

Bugs & Testing

- **Tips for Testing**

  - Make sure program meets the spec

  - Test if each method works independently

  - Test if methods work together

  - Test for edge cases

Bugs & Testing

- **Make sure program meets the spec**
  - What is wrong with this implementation?

```
public class ListStack implements DStack {
    private LinkedList<Double> myStack;

    public ListStack() {
        myStack = new LinkedList<Double>();
    }
    public void push(double d) {
        myStack.add(d);
    }
    …
}
```

Bugs & Testing

- **Test if each method works**

  - Four public methods

  boolean isEmpty()

  True if no elements, false otherwise

  void push(E elem)

  Add element on top of stack

  E pop()

  Remove & return top element, exception when empty

  E peek()

  Return (but don't remove) top element, exception when empty

Bugs & Testing

- **Test if each method works**

Thorough commenting can help

- Think about what each method is supposed to do

- Check if the method actually does what you think it should do

Bugs & Testing

- **Test if methods work together**

  - Should work in any order

```
stack.push(3.3)
stack.isEmpty()
stack.push(9.3)
stack.peek()
stack.pop()
stack.push(100)
stack.push(4343)
…
```

Bugs & Testing

- **Test for edge cases**

  - Empty stack

  - Push after resizing

  - Anything else?

Bugs & Testing

- **Testing tools:** JUnit Testing

  - Not required for Project 1

  - Required for Project 2

  - Covered in section next week

Induction Review

Induction Review

- **Proof by Induction**

  - Prove that the **first** statement in the infinite sequence of statements is true (Base case)

  - Prove that if **any one** statement in the infinite sequence of statements is true, then so is the **next** one. (Inductive case)

Induction Review

- **Proof by Induction**

  To prove statement P(n),

- Base Case:

  Prove that P(1) is true

- Inductive Case:

  Assuming P(k) is true,

  prove that P(k+1) is true

Recurrence Relations

Recurrence Relations

- **Recursively defines a Sequence**

  - Example: $T(n) = T(n-1) + 3$, $T(1) = 5$

    ^ Has $T(x)$ in definition

- **Solving Recurrence Relation**

  - Eliminate recursive part in definition

    = Find “Closed Form”

  - Example: $T(n) = 3n + 2$

Recurrence Relations

- **Expansion Method example**

  - Solve $T(n) = T(n-1) + 2n – 1, \quad T(1) = 1$

$$T(n) = T(n-1) + 2n – 1$$

$$T(n-1) = T([n-1]-1) + 2[n-1] – 1$$

$$= T(n-2) + 2(n-1) – 1$$

$$T(n-2) = T([n-2]-1) + 2[n-2] – 1$$

$$= T(n-3) + 2(n-2) – 1$$

Recurrence Relations

- **Expansion Method example**

$$T(n) = T(n-1) + 2n - 1$$
$$T(n-1) = T(n-2) + 2(n-1) - 1$$
$$T(n-2) = T(n-3) + 2(n-2) - 1$$

$$T(n) = [T(n-2) + 2(n-1) - 1] + 2n - 1$$
$$= T(n-2) + 2(n-1) + 2n - 2$$
$$T(n) = [T(n-3) + 2(n-2) - 1] + 2(n-1) + 2n - 2$$
$$= T(n-3) + 2(n-2) + 2(n-1) + 2n - 3$$

Recurrence Relations

- **Expansion Method example**

$T(n) = T(n-1) + 2n - 1$

$T(n) = T(n-2) + 2(n-1) + 2n - 2$

$T(n) = T(n-3) + 2(n-2) + 2(n-1) + 2n - 3$

…

$T(n) = T(n-k) + [2(n-(k-1)) + \ldots + 2(n-1) + 2n] - k$

$= T(n-k) + [2(n-k+1) + \ldots + 2(n-1) + 2n] - k$

Recurrence Relations

- **Expansion Method example**

$T(n) = T(n-k) + [2(n-k+1) + \ldots + 2(n-1) + 2n] - k$

When expanded all the way down, $T(n-k) = T(1)$

$n-k = 1$, $k = n-1$

$T(n) = T(n-[n-1]) + [2(n-[n-1]+1) + \ldots + 2(n-1) + 2n] - [n-1]$

$= T(1) + [2(2) + \ldots + 2(n-1) + 2n] - n + 1$

Recurrence Relations

- **Expansion Method example**

$$
\begin{aligned}
T(n) &= T(1) + [2(2) + \ldots + 2(n\text{-}1) + 2n] - n + 1 \\
&= T(1) + 2[2 + \ldots + (n\text{-}1) + n] - n + 1 \\
&= T(1) + 2[(n+1)(n/2) - 1] - n + 1 \\
&= T(1) + (n+1)(n) - 2 - n + 1 \\
&= T(1) + (n^2+n) - n - 1 \\
&= T(1) + n^2 - 1 \\
&= 1 + n^2 - 1 \\
&= n^2
\end{aligned}
$$

Recurrence Relations

- **Expansion Method example** **Check it!**

$T(n) = T(n-1) + 2n - 1, \quad T(1) = 1$

$T(n) = n^2$

$T(1) = 1$ same as 1^2

$T(2) = T(1) + 2(2) - 1 = 4$ same as 2^2

$T(3) = T(2) + 2(3) - 1 = 9$ same as 3^2

$T(4) = T(3) + 2(4) - 1 = 16$ same as 4^2

Asymptotic Analysis

Asymptotic Analysis

- **Describe Limiting behavior of F(n)**

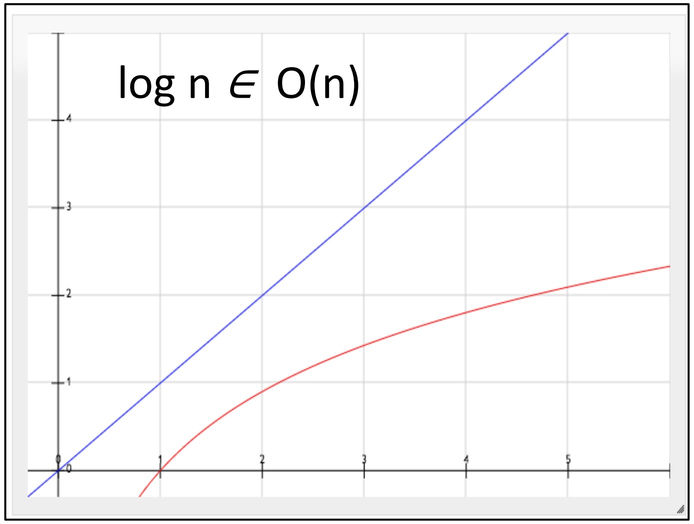
  - Characterize growth rate of F(n)

  - Use O(g(n)), *Ω*(g(n)), *Θ*(g(n)) for set of functions with asymptotic behavior □, □, □ & □ to g(n)

- **Upper Bound: O(n)**

  f(n) ∈ O(g(n)) if and only if there exist positive constants c and n_0 such that

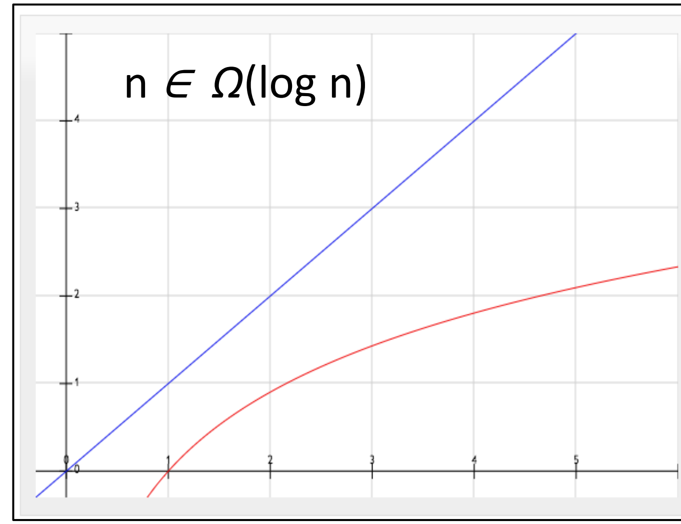
  f(n) □ c*g(n) for all n_0 □ n



Asymptotic Analysis

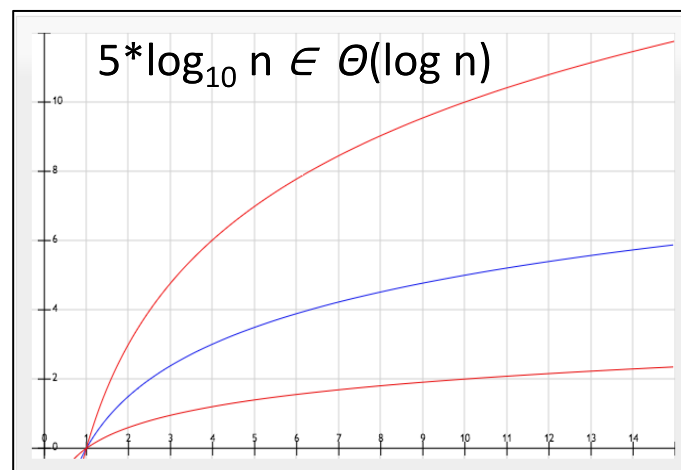
- **Lower Bound: Ω(n)**

  $f(n) \in \Omega(g(n))$ if and only if
  if there exist positive constants
  c and n_0 such that
  $c*g(n)$ □ $f(n)$ for all n_0 □ n



- **Tight Bound: Θ(n)**

  $f(n) \in \Theta(g(n))$ if and only if
  $f(n) \in \Omega(g(n))$ and
  $f(n) \in O(g(n))$



Asymptotic Analysis

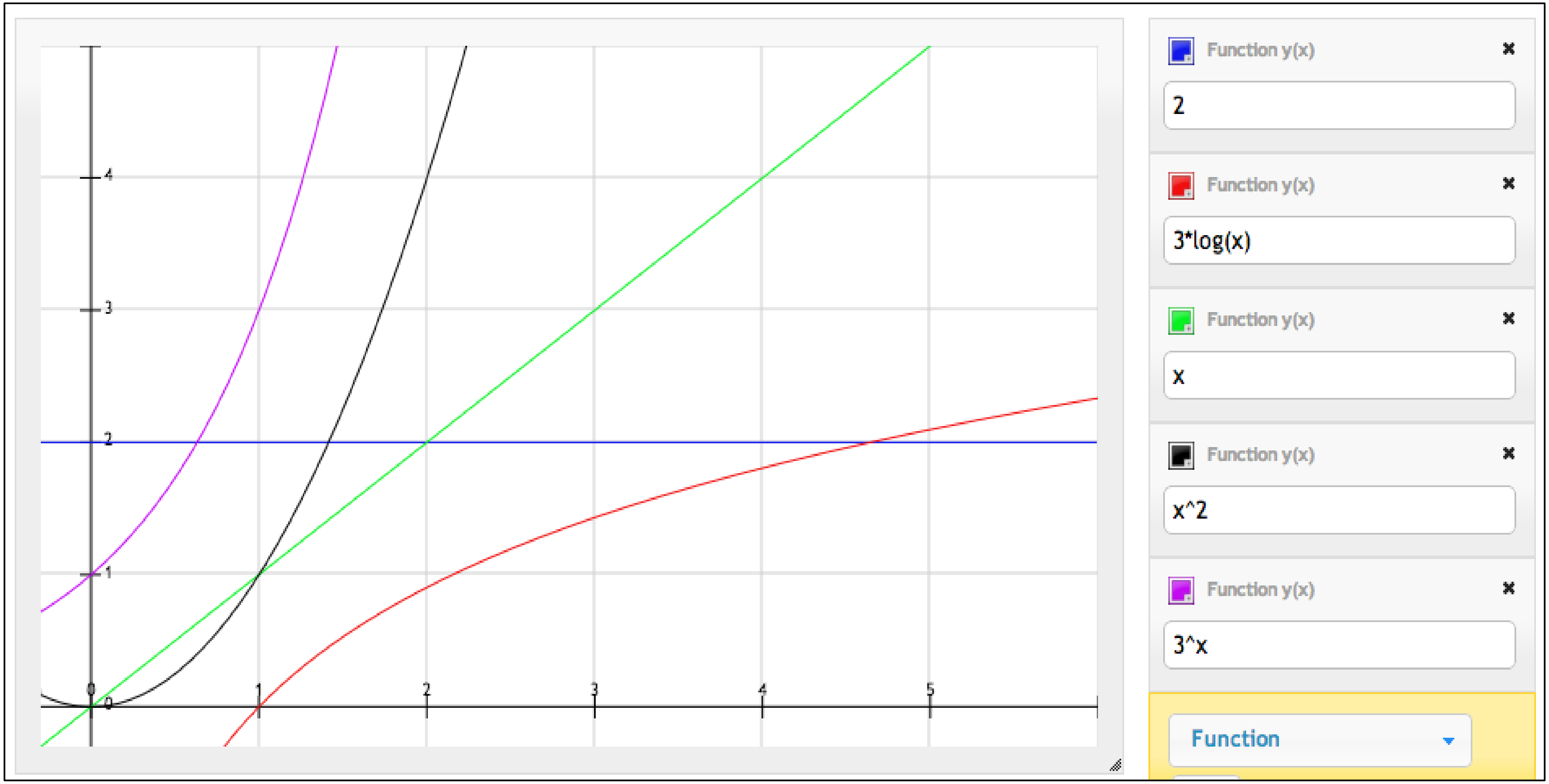
- **Ordering Growth rates (k = constant)**
  - Ignore Low-Order terms & Coefficients

O(k)	constant
O(log n)	logarithmic
O(n)	linear
$O(n^k)$	polynomial
$O(k^n)$	exponential (k > 1)

Increasing Growth rate

Asymptotic Analysis

- **Ordering Growth rates**



Asymptotic Analysis

- **Ordering Growth rates**
  - $\log^k n \in O(n^b)$ if $1 < k$ & $0 < b$
  - $n^k \in O(b^n)$ if $0 < k$ & $1 < b$

- **Ordering Example**

$2n^{100} + 10n$	n^{100}	4
$2^{n/100} + 2^{n/270}$	$2^{n/100}$	5
$1000n + \log^8 n$	n	3
$23785n^{1/2}$	$n^{1/2}$	2
$1000 \log^{10} n + 1^{n/300}$	$\log^{10} n$	1

Asymptotic Analysis

- **Proof Example: $f(n) \in O(g(n))$**

  - Prove or disprove $n\log n \in O(3n)$

$n\log n \quad \in O(3n)$
$n\log n \quad □ \; c*(3n)$, for $0 < c$ && $0 < n_0$ □ n
$(1/3)\log n$ □ c

but as $n \to \infty$, $\log n \to \infty$
Finite constant c always greater than log n cannot exist, no matter what n_0 we choose

$n\log n \notin O(3n)$

Homework Tips

Homework Tips

- **Problem #1**

  - Use formula in the book
  (You don't have to derive it by yourself)

- **Problem #2**

  - Use following rules:

  1. $\left\lfloor \frac{\lfloor \frac{x}{m} \rfloor}{n} \right\rfloor = \lfloor \frac{x}{mn} \rfloor$ which means $\left\lfloor \frac{\lfloor \frac{n}{2} \rfloor}{2} \right\rfloor = \lfloor \frac{n}{2^2} \rfloor$

  2. $\lfloor x \rfloor = m$ if and only if $m \leq x < m + 1$

Homework Tips

- **Problem #3**

  - $f(n) \times 10^{-6}$ sec □ t sec, solve for n

- **Problem #4 <= Not in this HW**

  - Remember that when you are proving P(k+1), you are assuming P(k) no matter how silly it is!
  - Find flaw in inductive reasoning

Homework Tips

- **Problem #5**

  - Use definitions and show you can/cannot find the constant c

- **Problem #6**

  - Analyze runtime of each loop & merge when appropriate
  - Practice finding exact runtime when you can
  - Think about maximum iteration of each loop