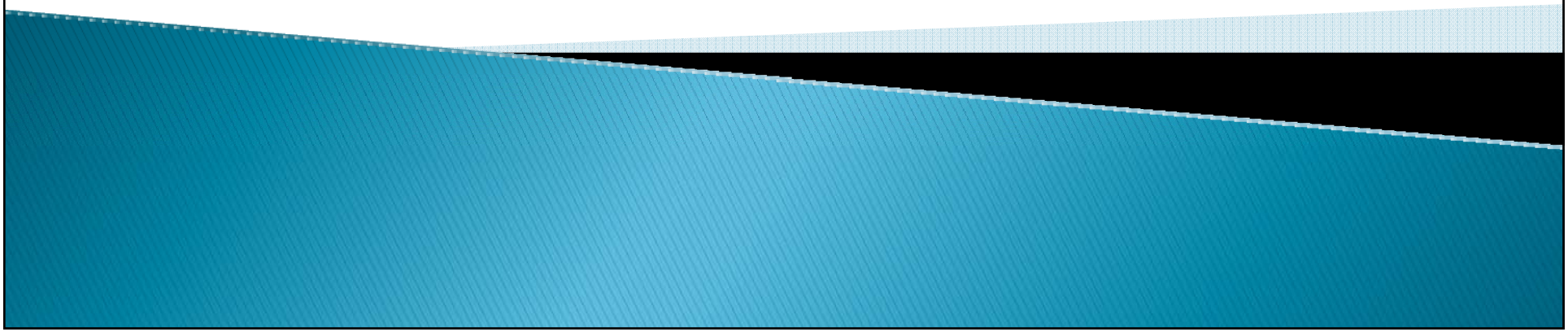
CSSE 220 Day 16

More Big Oh

Review of Generic Methods

Searching (sequential, Binary, Interpolation)



CSSE 220 Day 16

- Key Concepts Quiz has been set so that you can see your answers and the correct answers.
- Paint team/team members peer review due by 5 PM Wednesday.
- You will be asked to review several teams' Paint programs for functionality issues before the end of the week.
  - More details later.
- Next Programming assignment: Hardy's Taxi.
  Due next Tuesday, but begin thinking about it today.
  - An individual assignment.
- Markov assignment will be done in pairs. You can choose your partner again.
  - Must be different than your Paint partner

Recap: O, Ω, Θ

- f(N) is O(g(N) if there is a constant c such that for sufficiently large N, $f(N) \leq cg(N)$
  - Informally, the growth rate of f is bounded above by the growth rate of g
- f(N) is Ω(g(N) if there is a constant c such that for sufficiently large N, $f(N) \geq cg(N)$
  - Informally, the growth rate of f is bounded below by the growth rate of g
  - f(N) is Θ(g(N) if f(N) is O(g(n)) and f(N) is Ω(g(N))
  - Informally, the growth rate of f is the same as the growth rate of g

Recap: Limits and asymptotics

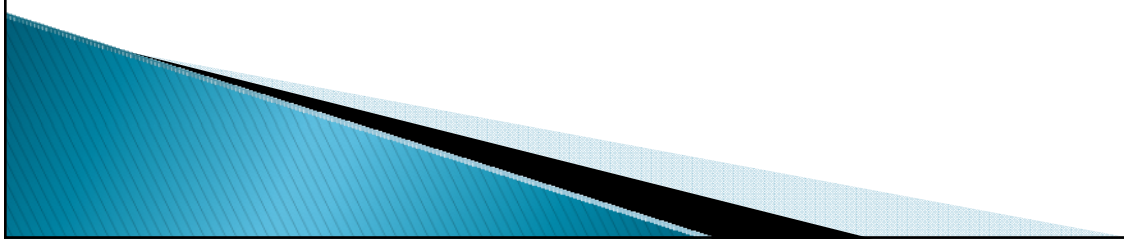
$\lim_{n\to\infty} \frac{f(n)}{g(n)}$	Conclusions		
0	f(n) is O(g(n))	g(n) is not O(f(n))	f(n) is not Θ(g(n))
non-zero	f(n) is O(g(n))	g(n) is O(f(n))	f(n) is Θ(g(n))
∞	f(n) is not O(g(n))	g(n) is O(f(n))	f(n) is not Θ(g(n))
undefined	We cannot conclude anything from the limit of the ratios.		

Apply this limit property to the following pairs of functions

1. N and N^2
2. $N^2 + 3N + 2$ and N^2
3. **$N + \sin(N)$ and N**
4. $\log N$ and N
5. $N \log N$ and N^2
6. **N^a and N^N**
7. a^N and b^N $(a < b)$
8. $\log_a N$ and $\log_b N$ $(a < b)$

Today's agenda

- Review of generic methods
- Searching (sequential, binary, interpolation)
- Abstract Data types and Data Structures



Generic methods: the need

- Consider the following methods:

```
public static void main(String[] args) {
    String [] ss = {"abc", "def", "ghij"};
    Integer [] ii = {new Integer(5), new Integer(6)};
    print(ss);
    print(ii);
 }

 public static void print(String[] strings){
    for (String s: strings)
       System.out.println(s);
 }

 public static void print(Integer[] ints){
    for (Integer i: ints)
       System.out.println(i);
 }
}
```

- Can we write **print** in a generic way so we do not have to have a separate method for each type of array?

Generic method: simple solution

```
public static <T> void print (T[] a){
    for (T obj: a)
        System.out.println(obj);
 }
```

- The type variable <T> before the method's return type tells the compiler: T will be a generic type for this method. Substitute for it the actual type of the argument.
- This method can be called with any array of objects.
- For some other methods, we need to constrain the generic type used (next slide)

Generic method: type constraint

- Suppose want a generic method to take an array as its only argument, and return the smallest item in the array.
- This only makes sense if the base type of the array implements the **Comparable** interface.

```
public static <T extends Comparable> T min (T[] a) {
     T smallest = a[0];
     for (int i=1; i<a.length; i++)
        if (smallest.compareTo(a[i]) > 0)
           smallest = a[i];
     return smallest;
  }
```

- This works, but gives a warning
  - Type safety: The method compareTo(Object) belongs to the raw type Comparable. References to generic type Comparable<T> should be parameterized
- How to fix it?

Generic method: fix the warning

```
public static <T extends Comparable<T>> T min (T[] a) {
     T smallest = a[0];
     for (int i=1; i<a.length; i++)
        if (smallest.compareTo(a[i]) > 0)
           smallest = a[i];
     return smallest;
  }
```

- Note that in this context "extends" means either "extends" or "implements".
- But this could be too restrictive. Perhaps we want to be able to be able to compare elements of a subclass with elements of a superclass (as in the Shape hierarchy from a couple of weeks ago).

Generic method: more generally

```
public static <T extends Comparable<? super T>> T min (T[] a) {
      T smallest = a[0];
      for (int i=1; i<a.length; i++)
         if (smallest.compareTo(a[i]) > 0)
            smallest = a[i];
      return smallest;
   }
```

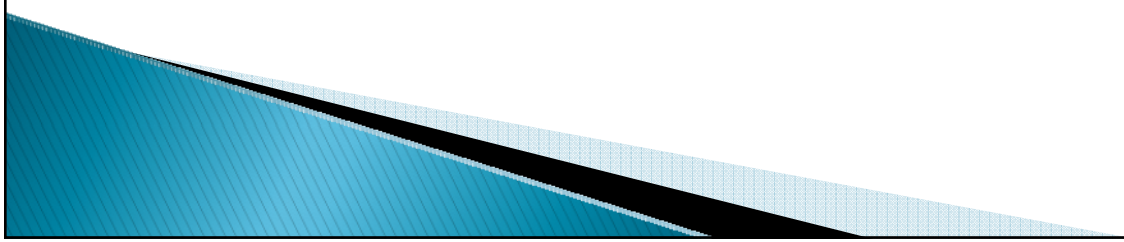
- The ? is a "wild card". <? super T> says we can compare to an element of any superclass of T.
- For more on wild cards (optional) see Weiss sections 4.7.2-4.7.4 or http://www.devarticles.com/c/a/Java/Wildcards-and-Generic-Methods-in-Java/

Search

- I'm thinking of a number between 1 and 100.
  - When you guess incorrectly, I'll say **higher or lower**
  - What's your strategy?
- How does your strategy change when your only feedback is yes/no?

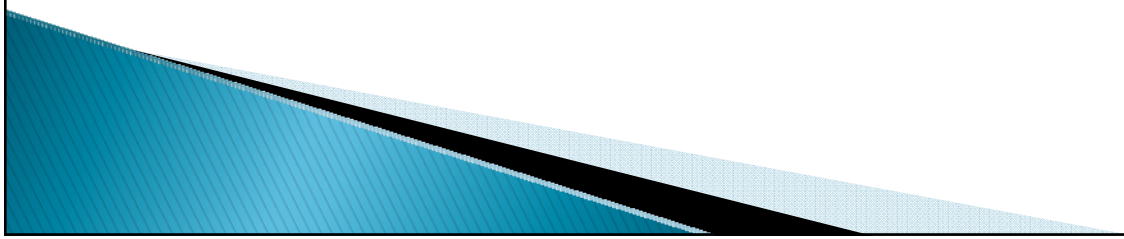
Search Efficiency

- I'm thinking of a number between 1 and 100 and say "higher" or "lower" if you are incorrect.
- How many guesses does it take in the best case? Worst-case? (Exact numbers)
  - Get with a partner and devise scenarios for each
- How does this change if the feedback is in the form yes/no?



Searching: Problem statement

- Search a collection of data for an item (or all items) whose key is a certain value (or has some relationship to a certain value.
- The key is usually one particular field of an object, but it may be a combination of fields.
- In today's examples, we assume that the collection is an array of N items.



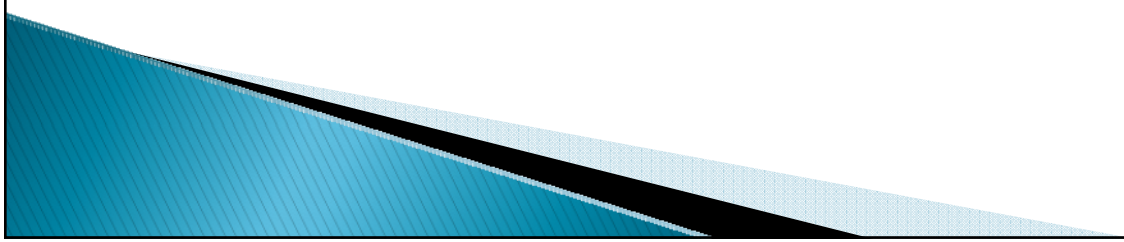
Sequential search

```
public static final int NOT_FOUND = -1;
public static <AnyType extends Comparable<? super AnyType>>
             int indexOf( AnyType [ ] a, AnyType x ) {
   for (int i=0; i<a.length; i++)
      if (a[i].compareTo(x)==0)
          return i;
   return NOT_FOUND;
}
```

- Best case (exact), worst-case (exact)
- Is the best you can do?
- No, but only on one condition…
  - The data must be **sorted.**

Binary Search

- *Divide-and-conquer*
- Useful only when the data is sorted (objects must be Comparable).
- Guess the middle position.
  - If it's correct, return that position
  - If it's too small, then recurse on the first half of the array
  - If it's too big, then recurse on the second half of the array.



Code: Binary Search

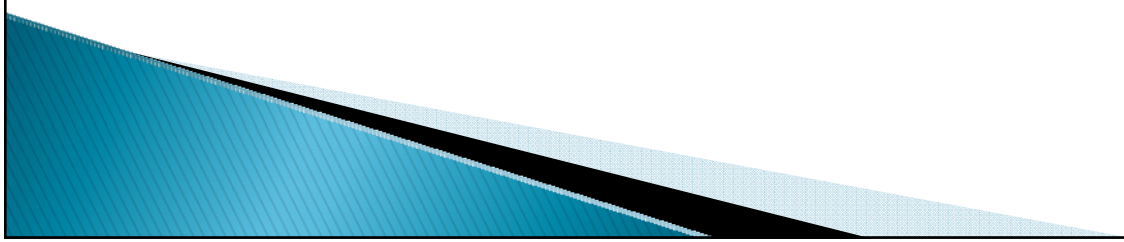
```
public static final int NOT_FOUND = -1;
public static <T extends Comparable<? super T>>
   int binarySearch( T[ ] a, T x ) {
       int low = 0;
       int high = a.length - 1;
       int mid;
       while( low <= high ) {
           mid = ( low + high ) / 2;

           if( a[ mid ].compareTo( x ) < 0 )
               low = mid + 1;
           else if( a[ mid ].compareTo( x ) > 0 )
               high = mid - 1;
           else
               return mid;
       }
       return NOT_FOUND;      // NOT_FOUND = -1
   }
```

Analysis of Search Time

- Sequential search: $n/2 = O(n)$ each time
- Binary Search: $O(n\log(n))$ once to sort
  $+ O(\log n)$ each time

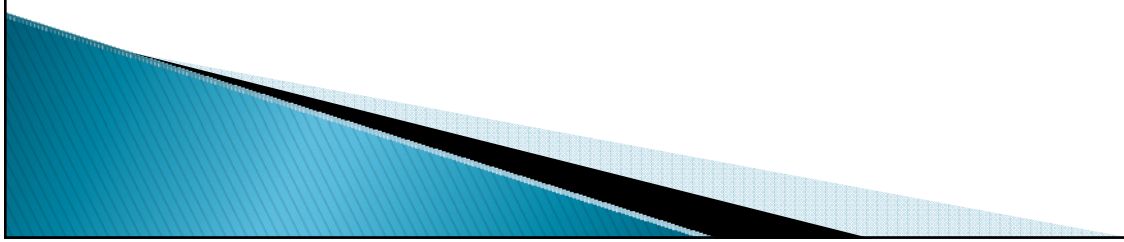
- Which is faster if we had an array with 1M elements and we wanted to search it 1M times?

- Demo



Interpolation search

- A more natural appraoch.
- If you were looking for my name in the phone book, would you start your search in the middle?
- In interpolation search, we choose where in the table to probe based on the value of the key relative to the first and last keys in the part of the table we are searching.



Interpolation search

- general formula: when looking for item x in a[low] … a[high], the next place to search is:

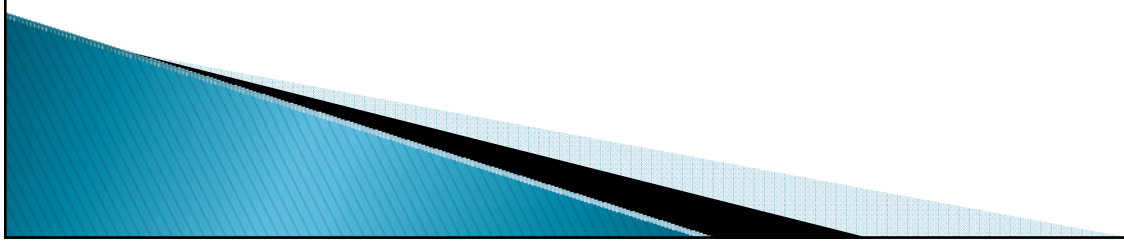
$$next = low + \left[\frac{x - a[low]}{a[high] - a[low]} * (high - low)\right]$$

- Average case # of probes:

- Simple references: Weiss Section 5.6.3, http://en.wikipedia.org/wiki/Interpolation_search

Interpolation search limitation

- What if the data is not uniform?
- Phone book
- RHIT CSSE staff members, 1986–2007.

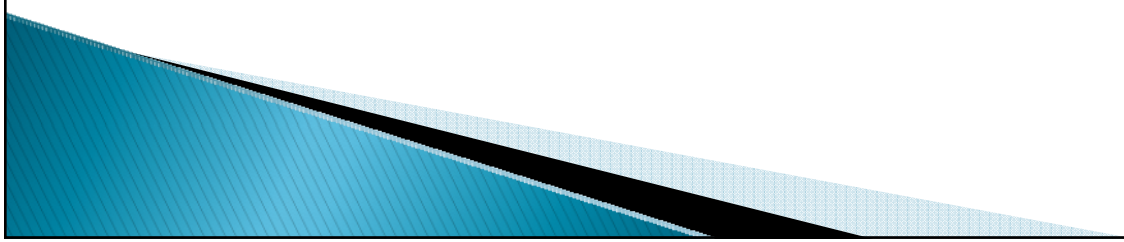


RHIT Staff Members, 1986-2007

anderson
atkins
ardis
azhar
bagert
baker
boutell
bowman
chenoweth
chidanandon
clifton
criss
curry
dalkolic
defoe
degler
jeschke
kaczmzrczyk
kinley
laxer
mohan
mellor
merkle
mutchler
oexmann
sengupta
surendran
sullivan
wollowski
young

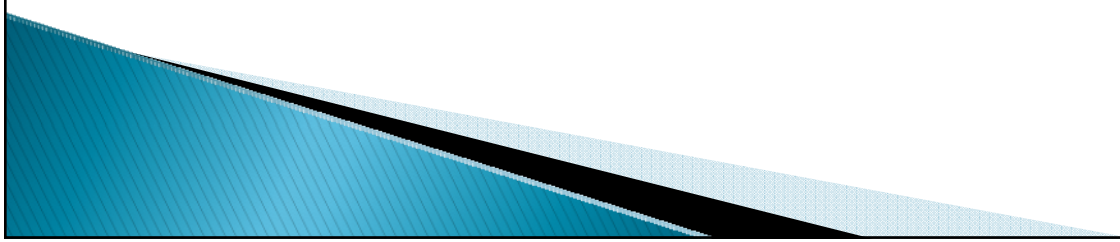
The downsides of binary search and interpolation search

- Initially sorting the array (expensive)
  - But may be worth it
- Keeping it sorted if the data changes
  - That's why we call these techniques "static search"
  - Other approaches (such as trees and hash tables) work better for dynamic data

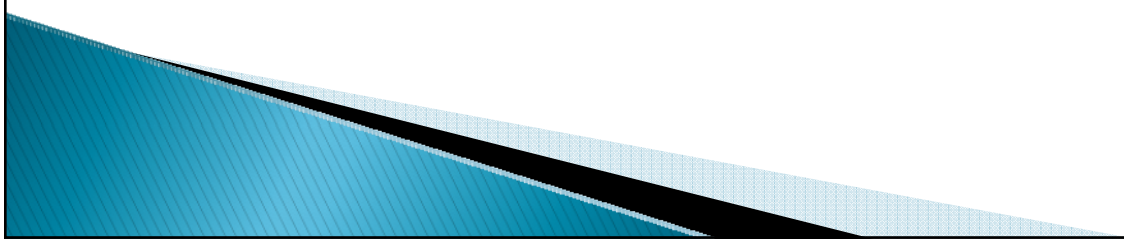


Searching and sorting are ubiquitous

- In the classic book series *The Art of Computer Programming*, Donald Knuth devotes a whole volume (about 700 pages) to sorting and searching.
  - Claims that about 70% of all CPU time is spent on these activities.



Break



Interlude

Fighting for their rights

More than 3,000 law students in India protested official policies against cheating. They were upset at a policy that bans copying on exams, according to Australia's Courier-Mail newspaper. Things got violent after police were called to help confiscate all their cheat sheets. Some students blocked a highway and started burning tires. They claim copying answers is a tradition that authorities must respect. "We found almost all students carrying books and photocopied notes hidden on their body," education official Radhanath Mishra told the paper. "We asked them to hand over all the illegally smuggled study materials. But they did not listen to us."

Reference (unfortunately it disappeared):

http://www.worldmag.com/world/issue/07-26-03/opening_5.asp

Another reference to this story (still live):

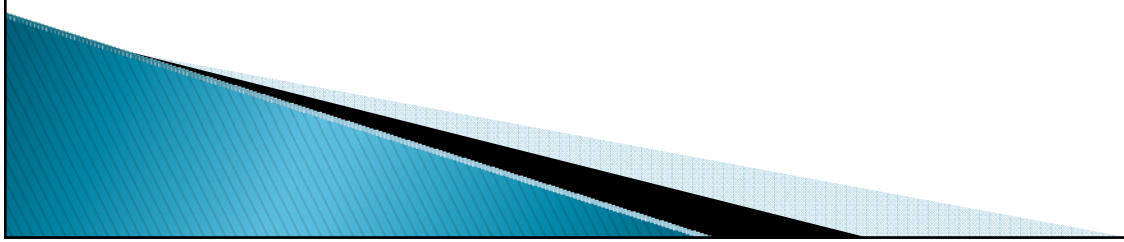
http://www.hinduonnet.com/thehindu/2003/07/09/stories/2003070904161200.htm

WORLD ON THE WEB

ARCHIVE FROM:
July 26, 2003
Volume 18
Number 28

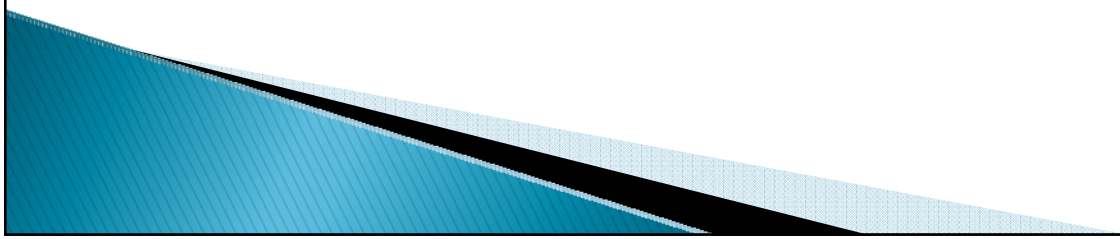
Weiss Book Overview

- Chapters 1-5: Review of Java, and foundations of algorithm analysis
- 6 Data Structure interface and usage
- 7-9 Fundamental algorithms
- 10-14 Applications of data structures and algorithms
- 15-21 Implementation of basic data structures.
- 22-24 Advanced data structures



Data Structures and the Java Collections Framework

- What is data?
- What do we mean by "structure"
  - A data type
    - But what *is* a data type, really?
  - An interpretation of the bits
    - An interpretation is basically a set of operations.
    - The interpretation may be provided by the hardware, as for `int` and `double` types, or by software, as for the `java.math.BigInteger` type.
    - Or by software with much assistance from the hardware, as for the `java.lang.Array` type.



What is an Abstract Data Type (ADT)?

- A mathematical model of a data type. Specifies:
  - The type of data stored
  - the operations supported
  - the types and return values of these operations
  - Specifies **what** each operation does, but not **how** it is implemented.
- **Example: Non-negative integer ADT.**
  A special value: *zero*:
- Basic operations include *succ pred isZero* .
  Derived operations include *plus* .
  - **Sample rules**:
    isZero(succ(n)) ➔ false
    plus(n, zero) ➔ n
    plus(n, succ(m)) ➔ succ(plus(n, m))

Standard implementation: Binary numbers. But there are many other possibilities.

Rules are independent of implementation.

- Next week:
  - Markov assignment will be done in pairs. You can choose your partner again.
  - Must be different than your Paint partner

- This week:
  - Next Programming assignment: Hardy's Taxi.
    Due next Tuesday, but begin thinking about it today.
  - An individual assignment.
  - Read it now and starting thinking
  - Lots of HW due Thursday (reading quiz, written problems)