Search

Jon Park
Feb 24, 2017
Announcements

• Lab 3 due on Monday!
• A message from Morgan and Bill:
  “OHs will be canceled next week!”
Last Time

- Asymptotic analysis (Big-O)
- Recursion
Today’s Outline

• Review
  • Asymptotic Analysis (Big-O)

• Search
  • Linear Search
  • Binary Search

• DefiningSortableClasses
  • Comparable
  • Comparator
Asymptotic Analysis (Big-O)

• “How scalable is the algorithm?”
• Commonly split into the following classes:
  • $O(1)$: “constant”
  • $O(\log n)$: “logarithmic” or “log n”
  • $O(n)$: “linear”
  • $O(n \log n)$: “$n \log n$”
  • $O(n^c)$: “polynomial”
    • $O(n^2)$: “quadratic”
    • $O(n^3)$: “cubic”
  • $O(c^n)$: “exponential”
Today’s Outline

• Review
  • Asymptotic Analysis (Big-O)

• Search
  • Linear Search
  • Binary Search

• Defining Sortable Classes
  • Comparable
  • Comparator
Search

• What is search?
  • Locating an element in data

• Examples:
  - documents
  - phone #
  - file
Searching for a Card

Time Complexity:
A. O(1) ← Best case
B. O(log n)
C. O(n) ← worst case
D. O(n log n)
E. Not sure
public boolean linearSearch (int[] data, int num) {
    for (int i = 0; i < data.length; i++) {
        if (num == data[i]) {
            return true;  \(O(1)\)
        }
    }
    return false;
}
Linear Search Summary

- **Precondition:**
  - None (The data can be in any order)

- **Time complexity:**
  - **Best case:**
    - $O(1)$
  - **Worst case:**
    - $O(n)$
  - **Average case**
    - $O(n)$
Searching for a Card (Again!)

Time Complexity:
A. $O(1)$  
B. $O(\log n)$  
C. $O(n)$  
D. $O(n \log n)$  
E. Not sure

\[
\frac{n}{2^x} = 1 \\
n = 2^x \\
\log_{2^x} n = x
\]
public boolean binarySearch (int[] data, int num) {
    int low = 0;
    int high = data.length - 1;
    while (high >= low) {
        int mid = (low + high) / 2;
        if (data[mid] == num)
            return true;
        if (data[mid] < num)
            low = middle + 1;
        else // data[mid] > num
            high = middle - 1;
    }
    return false;
}
Binary Search Summary

• Precondition:
  • The data is sorted

• Time complexity:
  • Best case:
    • O(1)
  • Average case
    • O(log n)
  • Worst case:
    • O(log n)
Search Time Complexity

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Best</th>
<th>Worst</th>
<th>Ave</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>$O(1)$</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Binary</td>
<td>$O(1)$</td>
<td>$O(\log n)$</td>
<td>$O(\log n)$</td>
</tr>
</tbody>
</table>

- Caveat: Binary search only works on sorted data.
- Some classes are sortable: Integer, String, …
- But how do we define new sortable classes?
Today’s Outline

• Review
  • Asymptotic Analysis (Big-O)

• Search
  • Linear Search
  • Binary Search

• Defining Sortable Classes
  • Comparable
  • Comparator
2 Types of Sortable Classes

• One “obvious” way to compare/sort
  • Examples: Integer
  • Make the given class Comparable (=implement Comparable)

• Multiple ways to compare/sort
  • Examples: PatientRecord
  • Make Comparator classes
    • E.g. nameComparator, idComparator
Today’s Outline

• Review
  • Asymptotic Analysis (Big-O)
• Search
  • Linear Search
  • Binary Search
• DefiningSortableClasses
  • Comparable
  • Comparator
public interface Comparable<T> {
    public int compareTo(T o);
}

Comparable<E> Interface

• “Class X implement Comparable<X>”
  ➔ “X contains compareTo(X obj) method”
  ➔ “X can compare objects of class X”

• To compare object A to object B (both of type X), call A.compareTo(B), which returns an int:
  • Negative when A is “<” in rank
  • Zero A and B are “=” in rank
  • Positive when A is “>” in rank
public class Integer ... {
    ...
}

Example: \textit{Integer}

```java
public class Integer ... implements Comparable<Integer> {
    ...
    // From Java6 Integer implementation (renamed variables)
    public int compareTo(Integer anotherInteger) {
        int x = this.value;
        int y = anotherInteger.value;
        return (x < y ? -1 : (x == y ? 0 : 1));
    }
}
```
Exercise: *Integer* class

```
Integer A = new Integer(1);
Integer B = new Integer(2);
int val = A.compareTo(B);
```

Given the lines above, which of the following is true?

A. `val < 0`
B. `val = 0`
C. `val > 0`
D. Not sure
Today’s Outline

• Review
  • Asymtotic Analysis (Big-O)

• Search
  • Linear Search
  • Binary Search

• Defining Sortable Classes
  • Comparable
  • Comparator
public interface Comparator<T> {
    ...
    int compare(T o1, T o2);
    ...
}
Comparator<E> Interface

- “Class X implements Comparator<Y>”
  ➞ “X contains compare(Y obj1, Y obj2) method.”
  ➞ “X can compare objects of class Y”
- To compare object A to object B (both of type Y), create an object C (of type X) and call C.compare(A, B), which returns an int:
  - Negative when A is “<” in rank
  - Zero A and B are “=” in rank
  - Positive when A is “>” in rank
Example: Comparator

class Patient {
    protected int age;
    protected String name;
    public Patient (String s, int a) {name = s; age = a;}
    public String getName() { return name; }
    public int getAge() {return age;}
}

class AgeComparator implements Comparator<Patient>{
    public int compare(Patient a, Patient b) {
        return a.getAge() - b.getAge();
    }
}

class NameComparator implements Comparator<Patient>{
    public int compare(Patient a, Patient b) {
        return a.getName().compareTo(b.getName());
    }
}
Exercise: Patient class

Patient A = new Patient(“Deepak”, 27);
Patient B = new Patient(“Jenny”, 52);
Given the lines above, how would you figure out the order between A and B?

"Name

Name Comparator c = new NameComparator();
c.compare (A, B);
2 Types of Sortable Classes

- Classes with 1 “obvious” way to compare/sort (a) vs Classes with multiple ways to compare/sort (b)

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implements Comparable(&lt;E&gt;)</td>
<td>O</td>
<td>X</td>
</tr>
<tr>
<td>(i.e. contains compareTo(E otherObj))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Need comparator classes containing compare(E obj1, E obj2)</td>
<td>X</td>
<td>O</td>
</tr>
<tr>
<td>The class itself supports comparison</td>
<td>O</td>
<td>X</td>
</tr>
<tr>
<td>Can be compared/sorted in multiple ways</td>
<td>X</td>
<td>O</td>
</tr>
</tbody>
</table>