Compsci 201
Exam 2 Review, Comparators, APTs
Part 1 of 4

us is for …

- Unix
  - Basis for Linux and Android and GNU and …
- User Interface/UI, User Experience/UX
  - User is the heart and soul

Susan Rodger
April 8, 2020

Google interface

- Marissa Mayer
  - Google – employee number 20
- Yahoo CEO
- Lumi Labs

Announcements

- APT-6 due Tuesday, April 7
- APT-7 out this week and due April 16 (note Thursday)
- Assignment P5 Percolation
  - Code due Thursday, April 9, grace thru April 13
- Assignment P6 Huffman out end of week, due April 22
- Discussion 12 Monday, April 13
  - Pre-discussion out at end of week
- Exam 2 is April 10 – Your own work!
- APT Quiz 2 is April 11-15 – Your own work!
Exam 2

- Take any time on Friday April 10. Maybe earlier.
- Take on Gradescope, DO NOT GO TO GRADESCOPE UNTIL YOU ARE READY
  - ONCE YOU CLICK ON IT, YOUR TIMER STARTS
- If you cannot take it on Friday, April 10
  - Fill out form to tell us why, and the soonest day you can take it
- Fill out a reflect form after taking the exam
  - Get 2 extra points
  - Give us feedback

More Exam 2 Details - GradeScope

- MC, short answer, and short code segments.
- You will type in, or click on answers
- Suggest: write code on paper and then type it in.
- Submit each question as you go.
  - Lose internet, just connect back in
- Exam is about 1 hour, 15 min
- You get an extra 1 hour, 15 min for logistics
- Total time is 2 hr, 30 min
- Those with accommodations, Kate will email

Exam 2 – Honor Code

- The exam is your work only
- Use books, open notes, code you have written
- Write code on paper only, then type into Gradescope
- DO NOT write code and run it in IntelliJ, Jshell or other computer means
- DO NOT Search on the web for answers to problems
- DO NOT Talk to any humans about the exam during the exam period
- DO NOT Talk to any humans about the exam until the exam is handed back
Now let’s review for the exam

Runtime and Other Analysis

• Asymptotic complexity of runtime or memory
  • Return value: `bleem(10)=88, bleem(20)=360`
  • \(1+2+\ldots+n=O(n^2)\)

• Runtime and value
  • \(\text{sum }+=1\)

Intuition and Reality

• "fun" x 4 = "funfunfunfun"
  • https://www.youtube.com/watch?v=VbrEsOLu75c
  • Analyze only in terms of \(n\), say with "fun"
    • It's all about line 34, how much does it matter?
  • \(3, 6, 9, 12, \ldots, 3n = 3(1 + 2 + \ldots + n)\) so big-Oh?
    • What if length of base is \(m\), then big-Oh?

Buffering … and faster

• "fun" x 4 = "funfunfunfun"
  • It's all about line 41, how much does it matter?
  • \(3, 3, 3, \ldots, 3 = 3n\) and \(O(n)\)
    • What if length of base is \(m\), then big-Oh?
Recursive Insert and Recurrences

- Recurrence relation for .add below? (TreeDemo.java)
  - https://coursework.cs.duke.edu/201spring20/classcode/
  - T(N) is time to add one value to tree of N values, what is complexity of adding N values one-at-a-time?

```java
publicTreeNode add(TreeNode root, String value) {
    if (root == null) {
        return new TreeNode(value, null, null);
    }
    int comp = value.compareTo(root.info);
    if (comp <= 0) {
        root.left = add(root.left, value);
    } else {
        root.right = add(root.right, value);
    }
    return root;
}
```

Understand and Analyze

- If tree has N values
  - One recursive call: T(n) = T(n/2) + O(1)
    - worst: T(n) = T(n-1) + O(1), solutions?
  - We do this n times. Upper bound n log n, but ...
    - log(1) + log(2) + ... + log(n) = log(1 x 2 x...x n)
    - log(n!)
    - n log n
    - **Stirling's Formula**
## Recurrence Relations

No need to derive, remember or look up

<table>
<thead>
<tr>
<th>Recurrence</th>
<th>Algorithm</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>T(n) = T(n/2) + O(1)</td>
<td>binary search</td>
<td>O(log n)</td>
</tr>
<tr>
<td>T(n) = T(n−1) + O(1)</td>
<td>sequential search</td>
<td>O(n)</td>
</tr>
<tr>
<td>T(n) = 2T(n/2) + O(1)</td>
<td>tree traversal</td>
<td>O(n)</td>
</tr>
<tr>
<td>T(n) = T(n/2) + O(n)</td>
<td>quicksort partition</td>
<td>O(n)</td>
</tr>
<tr>
<td>T(n) = 2T(n/2) + O(n)</td>
<td>mergesort, quicksort</td>
<td>O(n log n)</td>
</tr>
<tr>
<td>T(n) = T(n−1) + O(n)</td>
<td>selection or bubble sort</td>
<td>O(n²)</td>
</tr>
</tbody>
</table>

## WOTO Midterm Tree Review


## Compsci 201

Exam 2 Review, Comparators, APTs

Part 3 of 4

```
public TreeNode add(TreeNode root, String value) {
  if (root == null) {
    return new TreeNode(value, null, null);
  }
  int comp = value.compareTo(root.info);
  if (comp < 0) {
    root.left = add(root.left, value);
  } else {
    root.right = add(root.right, value);
  }
  return root;
}
```

## Dina Katabi

- Born in Syria, Professor at MIT
  - ACM Hopper Award, ACM Fellow
  - MacArthur Genius Award
  - AI that sees through walls?

When I was a kid, I was, ..., very much fascinated by Star Wars, and what fascinated me the most is this notion of the Force, this energy that connects all people and all objects and allows you to feel people that you can’t even see. And I remember many nights, I would be sitting at home, just, like, concentrating and focusing, trying to feel the Force, and I didn't feel anything, don't worry.

https://www.nytimes.com/2013/10/20/opinion/sunday/dina-katabi.html
Linked List - LowLevelLinkDemo.java

- Convert array to linked-list, values in same order
  - https://coursework.cs.duke.edu/201spring20/classcode/
- Choices: add to front or back of list being built
  - If we add to front? loop over array values …
  - If we add to back? Need pointer to front too
- Invariant?
  - last ...

```java
def createList(String[] array) {
    if (array.length == 0) return null;
    Node first = new Node(array[0], null);
    Node last = first;
    for (int k=1; k < array.length; k += 1) {
        last.next = new Node(array[k], null);
        last = last.next;
    }
    return first;
}
```

Add to Back Invariant

- Choices: add to front or back of list being built
  - If we add to back? Need pointer to front too
- Invariant
  - last is last node of list with values in range [0,k)
  - True before loop guard first tested? [0,1)
  - Updated in loop?

```java
def createList(String[] array) {
    if (array.length == 0) return null;
    Node first = new Node(array[0], null);
    Node last = first;
    for (int k=1; k < array.length; k += 1) {
        last.next = new Node(array[k], null);
        last = last.next;
    }
    return first;
}
```

Linked List Concepts Reviewed

- Loop with statement list = list.next
- Recurse with call doit(list.next)
- Must assign to .next field to create a linked list
  - Can be via node constructor
- Verify code works: empty list and last node of list
  - void null-pointer dereference

WOTO Link Review

ClientsList APT

- Nested class that can be compared: Person
  - Two approaches
    - Implement Comparable and a newer approach
  - Solve? Translate, Process, Translate

Sorting APTs

- Creating inner/nested classes to help
  - [https://www2.cs.duke.edu/csed/newapt/clientlist.html](https://www2.cs.duke.edu/csed/newapt/clientlist.html)
  - [https://www2.cs.duke.edu/csed/newapt/medaltable.html](https://www2.cs.duke.edu/csed/newapt/medaltable.html)

- Sorting Map.Entry<Key,Value> pairs
  - [https://www2.cs.duke.edu/csed/newapt/sortbyfreqs.html](https://www2.cs.duke.edu/csed/newapt/sortbyfreqs.html)

- Simple Use of API
  - [https://www2.cs.duke.edu/csed/newapt/lengthsort.html](https://www2.cs.duke.edu/csed/newapt/lengthsort.html)

How to sort: old and new

- This is the OLD way to sort in Java

```java
public class ClientsList {
    public class Person implements Comparable<Person> {
        String myFirst;
        String myLast;
        public Person(String fname, String lname) {
            myFirst = fname;
            myLast = lname;
        }
    }
    public String getFirst() {
        return myFirst;
    }
    @Override
    public int compareTo(Person p) {
        int ldiff = myLast.compareTo(p.myLast);
        if (ldiff != 0) return ldiff;
        return myFirst.compareTo(p.myFirst);
    }
}
```

- Here is newer way, no need for Comparable

```java
Arrays.sort(list, Comparator.comparing(Person::getLast)
.thenComparing(Person::getFirst));
```
**SortByFreqs, SortedFreqs**

- Can use TreeMap to keep count of each string
  - `Map.Entry` class is like inner class `Person`
  - However, does efficiency matter?

- Suppose for SortedFreqs we create a sorted set of the unique strings, why TreeSet?

  ```java
  public int[] freqs(String[] data) {
    TreeSet<String> set = new TreeSet<>(Arrays.asList(data));
  }
  ```

- Simply return count of each string

**Efficiency for APTs?**

- If we have M unique values from a set of N
  - This code is O(MN), we can do better

  ```java
  int[] ret = new int[set.size()];
  int index = 0;
  for(String s : set) {
    ret[index] = Collections.frequency(Arrays.asList(data), s);
    index++;
  }
  return ret;
  ```

- Extract values: `TreeMap<String, Integer>`
SortByFreqs

- **TreeMap**: `<String, Integer>`, keys in order
  - Keys are in sorted order
  - What song will you use for green dance?
  - [https://www.youtube.com/watch?v=XjYNIG5cZyQ](https://www.youtube.com/watch?v=XjYNIG5cZyQ)

```
TreeMap<String, Integer> map = new TreeMap<>();
for(String s : data) {
    map.putIfAbsent(s, 0);
    map.put(s, map.get(s) + 1);
}
```

Using TreeMap and Map.Entry

- Map.Entry is like a pair class for maps
  - Use `Map.Entry<Key, Value>`
  - Getting this right is not pretty

```
ArrayList<Map.Entry<String, Integer>> list =
    new ArrayList<>(map.entrySet());
Collections.sort(list,
    Map.Entry.comparingByValue(Comparator.reverseOrder()));
```

Using Lambda to sort in Java

- Perhaps more intuitive: sort strings using corresponding value in a map
  - Given string, find value
  - Leverage alphabetical order of strings and stable sort: "equal" keys don't change order, where equal is count of occurrences from map

```
ArrayList<String> list = new ArrayList<>(map.keySet());
Comparator<String> comp =
    Comparator.comparingInt(s -> map.get(s));
Collections.sort(list, comp.reversed());
return list.toArray(new String[0]);
```