### PFTFWo?
- Review recurrences, use them to motivate solving Isomorphic and QuasiIsomorphic
  - Complexity of: do trees have same shape
  - It Depends on definition of same shape?
- Motivation for PriorityQueue
  - Solve top M of N, Autocomplete
  - Lead-in to Huffman Compression

### Midterm

### Trees and Recursion
- Structure and code are very much related
  - Similar to searching folder, which requires ...
- Establish base cases: simple to solve, no recursion
  - Typically empty tree, one node tree/leaf
- Make recursive calls on .left, .right w/ other params
  - Must adjust parameters to reflect subtree call
  - Must use results of recursion to create new result
- As a root, do one thing: combine results of children

### Recurrence Summary

<table>
<thead>
<tr>
<th>Recurrence</th>
<th>Algorithm</th>
<th>complexity</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) = 2T(n/2) + O(n) )</td>
<td>Quicksort</td>
<td>( O(n \log n) )</td>
</tr>
<tr>
<td>( T(n) = T(n-1) + O(n) )</td>
<td>Selection Sort</td>
<td>( O(n^2) )</td>
</tr>
<tr>
<td>( T(n) = 2T(n-1) + O(1) )</td>
<td>Towers of Hanoi</td>
<td>( O(2^n) )</td>
</tr>
</tbody>
</table>

- \( T(n) \): time is labeled, \( n \) is size of input
  - Typically we say \( T(1) \) is \( O(1) \) in solving recurrence.

### Leafsum
- Sum all the values in leaves of tree
  - Base cases?
  - Recursive calls?
- Aside: min value in search tree
  - Max value?
  - Complexity in average case
  - In worst case?
  - N-node trees
- Red-black ensures average!
Isomorphic Trees

- How do we determine two trees have same shape?
  - What are the base cases, there are two parameters
  - How do you combine results of recursive calls?

- What is complexity for two trees, total N nodes?

QuasIsomorphic Trees

- How do we determine two trees have same shape?
  - What are the base cases, there are two parameters
  - How do you combine results of recursive calls?

- What is complexity for two trees, total N nodes?

Tree Questions


- Base cases, recursion, and more...

Looking up?
Algorithms and Data Structures

- Finding the top M of N elements, consider autocomplete for example

```
wh when winning the election
where to vote
who won the election
where can i vote
```

Store only the top (500) numbers …

- https://git.cs.duke.edu/201fall16/sorting-stuff/blob/master/src/TopMsorts.java
- Need an efficient structure that keeps elements ordered, but not too ordered
  - PriorityQueue
  - Add elements, remove elements (like Queue)
  - However, remove means "remove smallest"

```
Priority<Integer> pq = new PriorityQueue<>();
// add 10-million random integers to pq??
for (all of 10-million int values)
  pq.add(value);
if (pq.size() > 500) pq.remove();
while (pq.size() > 0) top2.add(pq.remove());
```

Sometimes simple is good, but …

- https://git.cs.duke.edu/201fall16/sorting-stuff/blob/master/src/TopMsorts.java
- Add all elements to an array/list, sort, find last M
  - Advantages? Disadvantages?
  - Do we need to store a 10-million numbers and sort them to find the top 500?

```
ArrayList<Integer> nums = new ArrayList<>();
// add 10-million random integers to nums
for (all of 10-million int values)
  nums.add(value);
Collections.sort(nums);
top1 = nums.subList(nums.size() - 500, nums.size());
```

Java 8 Streams Aside

- Streams aren't part of 201, but they're useful in this and other situations
  - Create a stream from some source
  - Alter the stream: filter or limit or …
  - Collect results, or forEach them, or …
- Chain results of streams: create new streams or terminate the streams, e.g., limit and forEach

```
Random r = new Random(1234);
IntStream is = r.ints(low, high);
is.limit(1000000).forEach(e -> System.out.println(e));
```
 Streams and Big Data

- Google originally implemented MapReduce
  - [https://en.wikipedia.org/wiki/MapReduce](https://en.wikipedia.org/wiki/MapReduce) now open sourced, e.g., Hadoop
- Distributed Storage and processing
  - Not everything fits on one disk, one computer, ...
  - How to coordinate and combine data
- Lazy evaluation: only compute when needed
  - To some just the even numbers in lazy stream, ...
  - Filter the even numbers, sum everything

 Why good for autocomplete?

- Advantageous to store fewer than a billion terms?
  - Assume terms are “weighted” by popularity
  - We want maximally weighted terms

 YAQ, YAQ, haha! (Yet Another Queue)

- What is the dequeue policy for a Queue?
  - Why do we implement Queue with LinkedList
  - Can we remove an element other than first?
  - How does queue help word-ladder/shortest path?
  - First item enqueued/added is the one we want
  - What if different element is “best”?
- PriorityQueue has a different dequeue policy
  - Best item is dequeued, queue manages itself to ensure operations are efficient

 PriorityQueue raison d’être

- Algorithms Using PQ for efficiency
  - Shortest Path: Google Maps to Internet Routing
    - How is this like word-ladder? How different?
  - Event based simulation
    - Coping with explosion in number of particles or things
  - Optimal A* search, game-playing, AI,
    - Can’t explore entire search space, can estimate good move
- Data compression facilitated by priority queue
  - All-time best assignment in a CompSci course?
    - Subject to debate, of course
  - From A-Z, soup-to-nuts, bits to abstractions
Priority Queue sorting

● See PQDemo.java, now with streams!
  ➢ https://git.cs.duke.edu/201fall16/sorting-stuff/blob/master/src/PQDemo.java
  ➢ code below sorts, complexity?

String[] array = {...}; // array filled with data
PriorityQueue<String> pq = new PriorityQueue<String>();
for(String s : array) pq.add(s);
for(int k=0; k < array.length; k++){
  array[k] = pq.remove();
}

● Bottlenecks, operations in code above
  ➢ Add words one-at-a-time to PQ v. all-at-once
  ➢ We’d like PQ to have tree characteristics, why?

Priority Queue top-M sorting

● What if we have lots and lots and lots of data
  ➢ code below sorts top-M elements, complexity?

Scanner s = ... // initialize;
PriorityQueue<String> pq = new PriorityQueue<String>();
while (s.hasNext()) {
  pq.add(s.next());
  if (pq.size() > M) pq.remove();
}

● What’s advantageous about this code?
  ➢ Store everything and sort everything?
  ➢ Store everything, sort first M?
  ➢ What is complexity of sort: O(n log n)

Craig Gentry

Duke ’95, Harvard Law, Stanford
Compsci PhD
ACM 2010 Hopper Award for...

"Fully homomorphic encryption is a bit like enabling a layperson to perform flawless neurosurgery while blindfolded, and without later remembering the episode. We believe this breakthrough will enable businesses to make more informed decisions, based on more studied analysis, without compromising privacy."

IBM VP, Software Research
Data Structures for AutoComplete

- We want M of N, ordered by weight/importance
  - Typically N is very, very large
- We can use brute force, if we type "the", find everything that matches "the", sort by weight, done
  - O(N) to search through everything
  - O(M log M) to sort list of M items
- We can use priority queue, insert matches of "the"
  - If we want only top 50 of M, limit size of PQ
  - O(log M) for PQ, done N times... O(N log M)

Use TreeSet (balanced Search Tree)

- tree.subSet(4,12)
  - https://docs.oracle.com/javase/8/docs/api/java/util/TreeSet.html#subSet-E-boolean-E-boolean-

Trie

- reTRIEval structure supporting very efficient lookup, O(w) where w is length of query, regardless of number of entries in structure!
  - 26-way branching
  - N-way branching
- Map if sparse branching

Trie, Trie, and Trie again

- https://git.cs.duke.edu/201fall16/set-examples/blob/master/src/TrieSet.java

- Method .contains is similar to others
  - What does Node class look like?

```java
public boolean contains(String s) {
    Node t = myRoot;
    for (int k = 0; k < s.length(); k++) {
        char ch = s.charAt(k);
        t = t.children.get(ch);
        if (t == null)
            return false; // no path below? done
    }
    return t.isWord; // was this marked as a word?
}
```