Compsci 201
Maps and Linked Lists

K is for …

- **Kernel**
  - Low-level foundation for an operating system

- **Key Pairs**
  - Public & private key make encryption happening, from Git to SSL

Announcements

- Exam 1 – Do not discuss until with anyone until handed back
- APT Quiz 1 out today
  - Do by yourself
- Assignment P3 out Friday – due 2/27
  - Builds on P2 Markov

PFWAE1

- Quick Review of Maps
- Linked List from high-level to low-level
  - Similar to how we viewed ArrayList/array
  - Low-level linked lists have history and current pedigree
- Iterators, Interfaces, Idioms
  - From design patterns to APIs
- APT Quiz ready today, Exam 1 not graded yet
### The java.util.Map interface, concepts

**HashMap <Key,Value> or <K,V**

<table>
<thead>
<tr>
<th>Method</th>
<th>return</th>
<th>purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>map.size()</td>
<td>int</td>
<td># keys</td>
</tr>
<tr>
<td>map.get(K)</td>
<td>V</td>
<td>get value</td>
</tr>
<tr>
<td>map.keySet()</td>
<td>Set&lt;K&gt;</td>
<td>Set of keys</td>
</tr>
<tr>
<td>map.values()</td>
<td>Collection&lt;V&gt;</td>
<td>All values</td>
</tr>
<tr>
<td>map.containsKey(K)</td>
<td>boolean</td>
<td>Is key in Map?</td>
</tr>
<tr>
<td>map.put(K,V)</td>
<td>V (ignored)</td>
<td>Insert (K,V)</td>
</tr>
<tr>
<td>map.entrySet()</td>
<td>Set&lt;Map.Entry&lt;K,V&gt;&gt;</td>
<td>Get (K,V) pairs</td>
</tr>
<tr>
<td>map.clear()</td>
<td>void</td>
<td>Remove all keys</td>
</tr>
<tr>
<td>map.putIfAbsent(K,V)</td>
<td>V (ignored)</td>
<td>Insert if not there</td>
</tr>
</tbody>
</table>

---

### BigWord APT

**Problem Statement**

In days of yore, aka BG (Before Google), search engines ranked webpages in part by the number of occurrences of a word on the page. You should write method most to determine and return the word that occurs most often in an array of sentences. This most frequently occurring word will be unique — that is you don’t need to worry about two words both occurring more often than any other word. The word returned should be all lower-case regardless of the case of letters in sentences.

Each string in sentences represents several words, each word is delimited by spaces from other words. Words should be considered the same without respect to case, so BIG is the same word as big, for example.

**Examples**

1. sentences = ["one fish two", "fish red fish blue", "fish this fish is black"]
   
   ```java
   public String most(String[] sentences) {
     Map<String, Integer> map = new HashMap<>();
     for (String one : sentences) {
       String s = one.toLowerCase().split(" ");
       for (String s : one.toLowerCase().split(" ")) {
         map.putIfAbsent(s, 0);
         map.put(s, map.get(s) + 1);
       }
     }
     int mx = Collections.max(map.values());
     for (String key : map.keySet()) {
       if (map.get(key) == mx) return key;
     }
     return "never";
   }
   ```

   The word "fish" occurs five times, which is more than any other word.

---

### Investigate Map Solution

- One pass over the data instead of many passes
- Understand all map methods
- Why is line 39 never executed? Still needed?

---

### What is a java.util.List in Java?

- **Interface** for collection of elements
  - Add, remove, traverse, …
  - What can a list do to itself?
  - What can we do to a list?

- **Why more than one kind of list: Array and Linked?**
  - Useful in different applications
  - How do we analyze differences?
  - How do we use them in code?
Remember? \texttt{list.remove(0)}

- What is “faster”? LinkedList or ArrayList

  ```java
  public double removeFirst(List<String> list) {
      double start = System.nanoTime();
      while (list.size() != 1) {
          list.remove(0);
      }
      double end = System.nanoTime();
      return (end - start) / 1e9;
  }
  ```

- Must shift N-1 elements
  - Details in code below? Some matter, some …

- Shifting N elements is O(N²): why?

Random Access v Splicing…

- How does find-a-track work? Fast forward?
  - Quick survey of linked list code

  ```java
  @Override
  public E remove(int index) {
      rangeCheck(index);
      E hold = (E) myStorage[index];
      for(int k=index; k < myStorage.length-1; k++) {
          myStorage[k] = myStorage[k+1];
      }
      mySize--; myStorage[mySize] = null;
      return hold;
  }
  ```

Conceptual: array[] to Node

- How do we implement ArrayList? Use array[]
  - \texttt{list.get(n)} is O(1), BUT
  - \texttt{list.remove(0)} is O(N)

- How do we implement LinkedList? Use Node
  - \texttt{list.get(n)} is O(n), BUT
  - \texttt{list.remove(0)} is O(1)

- Tradeoffs: what does sequence of nodes provide?

http://bit.ly/201playRecord
What’s in a Node?

• Some information
• Place to snap another node

In Java we’ll see
• String reference: info
• Node reference: next

Visualizing/Understanding Nodes

• [https://coursework.cs.duke.edu/rodger/diyad-new](https://coursework.cs.duke.edu/rodger/diyad-new)
  • `diyad.linkedlist.SimpleLinkedList`
  • Like pair, note: `this` not needed below
  • Instance variables for String and "next node"

```
private class Node{
  String info;
  Node next;
  public Node(String key, Node link){
    this.info = key;
    this.next = link;
  }
}
```

remove(0) for linked list?

• Looking at remove(0) not remove(n) now
  • Instance variables `myFirst` and `myLast`
  • Initially null, but we’ll see what `.add` does

```
public String remove(int index) {
  if (index < 0) {
    throw new IndexOutOfBoundsException("negative index: "+index);
  }
  if (mySize == 0) {
    throw new IndexOutOfBoundsException("index too large for \n",index,mySize));
  }
  if (index == 0) {
    String ret = myFirst.info;
    if (myFirst == myLast) myLast = null;
    myFirst = myFirst.next;
    mySize--;
    return ret;
  }
```

Adding nodes to end: `.add( . . )`

• Class invariant: `myLast` references last node
  • Symmetry: `myFirst` references first node
  • When the list is empty, special case?
    • Always add node to end of list

```
public void add(String s) {
  if (myLast == null) {
    myFirst = myLast = new Node(s,null);
    mySize = 1;
    return;
  }
  myLast.next = new Node(s,null);
  myLast = myLast.next;
  mySize += 1;
}
```
Adding New Nodes

• To add to the end of a linked list
  • Maintain reference to first node
    • only through first node can we access entire list
  • Need reference to last node
    • To add a new last node
  • Often need initialization code
    • First node anchors list
      • Must do before loop
      • Loop will add over and over to end

Visualizing, Thinking, Understanding

• How to picture myLast and myLast.next
  • Both are Node pointers aka Node references
  • Like all Java Object variables: memory location

• Conceptually? An arrow, a pointer
  • References a Node: label for memory location
Another program for understanding

- Not modeling a List class, just plain Nodes
  - [https://coursework.cs.duke.edu/201spring20/classcode/blob/master/src/LowLevelLinkDemo.java](https://coursework.cs.duke.edu/201spring20/classcode/blob/master/src/LowLevelLinkDemo.java)
- LowLevelLinkDemo: add to back, keep front
- Local variable last: always point to last node
  - Each time through loop? True, thus an invariant

```java
private Node 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++) {
        last.next = new Node(array[k], null);
        last = last.next;
    }
    return first;
}
```

Main in LowLevelLinkDemo.java

```java
public static void main(String[] args) {
    String[] vg = {"squash", "corn", "potato", "onion", "peas"};
    LowLevelLinkDemo ld = new LowLevelLinkDemo();
    Node list1 = ld.createList(vg);
    Node list2 = ld.createListFront(vg);
}
```

Visualizing Code – CreateList

Add nodes to end of list

- Using Java Tutor:
- See first and last: both Node variables

Adding first node to linked list

- Repeatedly add first element, initially null
  - New first node points at previous first node
- first references/points to new first node
  - Can use first = new Node(vg[k], first)

```java
private Node createListFront(String[] vg) {
    Node first = null;
    for (int k=0; k < vg.length; k++) {
        Node nf = new Node(vg[k], first);
        first = nf;
    }
    return first;
}
```
Visualizing Code – CreateListFront

Add nodes to **front** of list

- Using Java Tutor:
- See first and last: both Node variables

List Traversal

- Visiting (printing) every value in an array
  - Start with first node, print .info, advance .next
  - Done when current node is null

Reference: Array Traversal

- Visiting (printing) every value in an array
  - Initialize index, print w/index, increment index
  - Elements of array are adjacent in memory

```java
private void print(String[] list) {
    for (int i = 0; i < list.length; i++) {
        System.out.printf(“%s”, list[i]);
        System.out.print(“ “);
    }
    System.out.println();
}
```

WOTO (correctness counts)


You can install Java Tutor in IntelliJ – see course website Resources tab
John Tukey: 1915-2000

• Cooley-Tukey FFT
• Bit is a binary digit
• Box or Box and Whiskers Plots

Far better an approximate answer to the right question, which is often vague, than an exact answer to the wrong question, which can always be made precise.

The combination of some data and an aching desire for an answer does not ensure that a reasonable answer can be extracted from a given body of data.

Removing some values: filter

• Remove all occurrences of X, or ...
  • When we remove in array, we shift. Trouble?
  
• ListRemoveAndCount: exception thrown!
  • ConcurrentModificationException

```java
public List<String> removeAll(String target, List<String> list) {
    for(String w : list) {
        if (w.equals(target)) {
            list.remove(w);
        }
    }
    return list;
}
```

Incorrect Results

• See ListRemoveAndCount.java
  • You shouldn’t do this: results in errors
  • Remove the kth element (think 0)

```java
public List<String> removeAllWrong(String target, List<String> list) {
    for(int k=0; k < list.size(); k++) {
        String w = list.get(k);
        if (w.equals(target)) {
            list.remove(k);
        }
    }
    return list;
}
```

Iterators to the Rescue

• Iterators are soooo nice. But timing?
  • Why $O(N)$ linked list and $O(N^2)$ array?

```java
public List<String> removeAllWithIterator(String target, List<String> list) {
    Iterator<String> iter = list.iterator();
    while (iter.hasNext()) {
        String w = iter.next();
        if (w.equals(target)) {
            iter.remove();
        }
    }
    return list;
}
```
From Iterator to Iterable

- **Enhanced for:**
  ```java
  for(String s : list) {
    ...
  }
  ```
- Underneath, uses iterator
- Code below \(O(N)\) for both lists!

```
public int iterateEach(List<String> list) {
    int total = 0;
    for(String s : list) {
        total += s.length();
    }
    return total;
}
```

- **What if indexing loop used?,**
  ```java
  e.g., list.get(k)
  ```
- Code below is ?

```
public int iterate(List<String> list) {
    int total = 0;
    for(int k=0; k < list.size(); k++) {
        total += list.get(k).length();
    }
    return total;
}
```

From Iterator to Iterable

- **Compare the two**
  ```java
  ListSplicer.java
  ```

  ```java
  ListSplicer.java  (linked list first, the arrayList)
  ```

```
iterateEach iterate
```

Linked list too slow with .get
APT Practice Quiz

- APT Practice Quiz is on Sakai now
  - NOT FOR CREDIT, Just for practice
  - You can see how an APT quiz works
  - Only Available through Sunday 11:59pm

- RECOMMEND trying APT Practice Quiz before taking the APT Quiz

APT Quiz Details

- APT Quiz 1 available on Sakai
  - Wed. Feb 19 at 8pm – Mon. Feb 24 11:59pm

- Once you start, you get 2.5 hours
  - You cannot stop and restart it.
  - More time if you get accommodations

- Must start by 9:29pm Monday night!
- Recommend you take it BEFORE Monday

- You can see the timer in Sakai
  - We will not grade anything you submit after time runs out

- You CANNOT, CANNOT, CANNOT collaborate on the quiz.
  We run reasonably sophisticated similarity detection software

APT Quiz

- We expect that everyone will get the first problem
  - Sometimes we are wrong. But it’s designed to be straightforward. If you’ve done the APTs? You’ll succeed

- We expect everyone will know how to solve the other problems, but sometimes coding and debugging is not easy
  - There is a time limit, if stuck? Try next problem