From data to information

- Data that’s organized can be processed
  - Is this a requirement?
  - What does “organized” means

- Purpose of map in Markov assignment?
  - Properties of keys?
  - Comparable v. Hashable

- TreeSet v. HashSet
  - Speed v. order
  - Memory considerations

Goldilocks and the Hashtable

- A hashtable is a collection of buckets
  - Find the right bucket and search it
  - Bucket organization?
    - Array, linked list, search tree

Structuring Data: The inside story

- How does a hashtable work? (see SimpleHash.java)
  - What happens with put(key, value) in a HashMap?
  - What happens with getValue(key)?
  - What happens with remove(key)?

```java
ArrayList<ArrayList<Combo>> myTable;
public void put(String key, int value) {
    int bucketIndex = getHash(key);
    ArrayList<Combo> list = myTable.get(bucketIndex);
    if (list == null) {
        list = new ArrayList<Combo>();
        myTable.set(bucketIndex, list);
    }
    list.add(new Combo(key, value));
    mySize++;
}
```

How do we compare times? Methods?

- Dual 2Ghz Power PC
  - King James Bible: 823K words time to arraylist hash: 5.524
time to default hash: 6.137
time to link hash: 4.933
arraylist hash size = 34027
Default hash size = 34027
link hash size = 34027

- Linux 2.4 Ghz, Core Duo
  - King James Bible: 823K words
time to arraylist hash: 1.497
time to default hash: 1.128
time to link hash: 1.03
arraylist hash size = 34027
Default hash size = 34027
link hash size = 34027

- OS X Laptop 2.4 Ghz, Core Duo
  - King James Bible: 823K words
time to arraylist hash: 2.415
time to default hash: 2.143
time to link hash: 2.085
arraylist hash size = 34027
Default hash size = 34027
link hash size = 34027
What’s the Difference Here?

- How does find-a-track work? Fast forward?

Contrast LinkedList and ArrayList

- See ISimpleList, SimpleLinkedList, SimpleArrayList
  - Meant to illustrate concepts, not industrial-strength
  - Very similar to industrial-strength, however
- ArrayList --- why is access O(1) or constant time?
  - Storage in memory is contiguous, all elements same size
  - Where is the 1st element? 40th? 360th?
  - Doesn’t matter what’s in the ArrayList, everything is a pointer or a reference (what about null?)

What about LinkedList?

- Why is access of Nth element linear time?
- Why is adding to front constant-time O(1)?

ArrayLists and linked lists as ADTs

- As an ADT (abstract data type) ArrayLists support
  - Constant-time or O(1) access to the k-th element
  - Amortized linear or O(n) storage/time with add
    - Total storage used in n-element vector is approx. 2n, spread over all accesses/additions (why?)
    - Adding a new value in the middle of an ArrayList is expensive, linear or O(n) because shifting required
- Linked lists as ADT
  - Constant-time or O(1) insertion/deletion anywhere, but...
  - Linear or O(n) time to find where, sequential search
- Good for sparse structures: when data are scarce, allocate exactly as many list elements as needed, no wasted space/copying (e.g., what happens when vector grows?)
Linked list applications

- Remove element from middle of a collection, maintain order, no shifting. Add an element in the middle, no shifting
  - What’s the problem with a vector (array)?
  - Emacs visits several files, internally keeps a linked-list of buffers
  - Naively keep characters in a linked list, but in practice too much storage, need more esoteric data structures
- What’s \((3x^3 + 2x^2 + x + 5) + (2x^4 + 5x^2 + x^2 + 4x)\)?
  - As a vector \((3, 0, 2, 0, 1, 5)\) and \((0, 2, 5, 1, 4, 0)\)
  - As a list \(((3, 5), (2, 3), (1, 1), (5, 0))\) and ________?
  - Most polynomial operations sequentially visit terms, don’t need random access, do need “splicing”
- What about \((3x^{100} + 5)\)?

Linked list applications continued

- If programming in C, there are no “growable-arrays”, so typically linked lists used when # elements in a collection varies, isn’t known, can’t be fixed at compile time
  - Could grow array, potentially expensive/wasteful especially if # elements is small.
  - Also need # elements in array, requires extra parameter
  - With linked list, one pointer used to access all the elements in a collection
- Simulation/modeling of DNA gene-splicing
  - Given list of millions of CGTA… for DNA strand, find locations where new DNA/gene can be spliced in
    - Remove target sequence, insert new sequence

Linked lists, CDT and ADT

- As an ADT
  - A list is empty, or contains an element and a list
    - ( ) or \((x, (y, ())))\)
- As a picture

```
0
p
```

- As a CDT (concrete data type) pojo: plain old Java object

```java
public class Node {
    String value;
    Node next;
    Node(String s, Node link){
        value = s;
        next = link;
    }
}
```
```
// _ declarations here
Node list = null;
while (scanner.hasNext()) {
    list = new Node(scanner.next(), list);
}
```

Building linked lists

- Add words to the front of a list (draw a picture)
  - Create new node with next pointing to list, reset start of list

```java
public class Node {
    String value;
    Node next;
    Node(String s, Node link){
        value = s;
        next = link;
    }
}
```

- What about adding to the end of the list?
Dissection of add-to-front

- List initially empty
- First node has first word

\[
\text{list} = \text{new Node(word, list);} \\
\text{Node(String s, Node link)} \\
\{ \text{info} = s; \text{next} = \text{link}; \}
\]

- Each new word causes new node to be created
  - New node added to front
- Rhs of operator = completely evaluated before assignment

Standard list processing (iterative)

- Visit all nodes once, e.g., count them or process them

\[
\text{public int size(Node list){} \\
\text{int count = 0;} \\
\text{while (list != null) {} \\
\text{count++;} \\
\text{list = list.next;}} \\
\text{return count;} \\
\}
\]

- What changes in code if we generalize what process means?
  - Print nodes?
  - Append “s” to all strings in list?

Nancy Leveson: Software Safety

Founded the field
- Mathematical and engineering aspects
  - Air traffic control
  - Microsoft word

"C++ is not state-of-the-art, it's only state-of-the-practice, which in recent years has been going backwards"

- Software and steam engines: once extremely dangerous?
- THERAC 25: Radiation machine that killed many people

Building linked lists continued

- What about adding a node to the end of the list?
  - Can we search and find the end?
  - If we do this every time, what's complexity of building an N-node list? Why?

- Alternatively, keep pointers to first and last nodes of list
  - If we add node to end, which pointer changes?
  - What about initially empty list: values of pointers?
    - Will lead to consideration of header node to avoid special cases in writing code

- What about keeping list in order, adding nodes by splicing into list? Issues in writing code? When do we stop searching?
Standard list processing (recursive)

- Visit all nodes once, e.g., count them

  ```java
  public int recsize(Node list) {
    if (list == null) return 0;
    return 1 + recsize(list.next);
  }
  ```

- Base case is almost always empty list: null pointer
  - Must return correct value, perform correct action
  - Recursive calls use this value/state to anchor recursion
  - Sometimes one node list also used, two “base” cases

- Recursive calls make progress towards base case
  - Almost always using list.next as argument

Recursion with pictures

- Counting recursively

  ```java
  int recsize(Node list) {
    if (list == null) return 0;
    return 1 + recsize(list.next);
  }
  ```

Recursion and linked lists

- Print nodes in reverse order
  - Print all but first node and...
    - Print first node before or after other printing?

  ```java
  public void print(Node list) {
    if (list != null) {
      System.out.println(list.info);
      System.out.println(list.info);
      System.out.println(list.info);
      System.out.println(list.info);
      System.out.println(list.info);
      System.out.println(list.info);
      System.out.println(list.info);
    }
  }
  ```

Complexity Practice

- What is complexity of Build? (what does it do?)

  ```java
  public Node build(int n) {
    if (null == n) return null;
    Node first = new Node(n, build(n-1));
    for(int k = 0; k < n-1; k++) {
      first = new Node(n, first);
    }
    return first;
  }
  ```

- Write an expression for T(n) and for T(0), solve.
  - Let T(n) be time for build to execute with n-node list
  - T(n) = T(n-1) + \( O(n) \)
Changing a linked list recursively

- Pass list to method, return altered list, assign to list
  - Idiom for changing value parameters

```java
list = change(list, "apple");
public Node change(Node list, String key) {
    if (list != null) {
        list.next = change(list.next, key);
        if (list.info.equals(key)) return list.next;
        else return list;
    }
    return null;
}
```
- What does this code do? How can we reason about it?
  - Empty list, one-node list, two-node list, n-node list
  - Similar to proof by induction

Analyzing Algorithms

- Consider three solutions to SortByFreqs
  - Sort, then scan looking for changes
  - Insert into Set, then count each unique string
  - Find unique elements without sorting, sort these, then count each unique string
  - Use a Map (TreeMap or HashMap)
- We want to discuss trade-offs of these solutions
  - Ease to develop, debug, verify
  - Runtime efficiency
  - Vocabulary for discussion

What is big-Oh about? (preview)

- Intuition: avoid details when they don’t matter, and they don’t matter when input size (N) is big enough
  - For polynomials, use only leading term, ignore coefficients
    - $y = 3x$     $y = 6x-2$     $y = 15x + 44$
    - $y = x^2$     $y = x^2-6x+9$     $y = 3x^2+4x$
- The first family is $O(n)$, the second is $O(n^2)$
  - Intuition: family of curves, generally the same shape
  - More formally: $O(f(n))$ is an upper-bound, when n is large enough the expression $cf(n)$ is larger
  - Intuition: linear function: double input, double time, quadratic function: double input, quadruple the time

Recall adding to list (class handout)

- Add one element to front of ArrayList
  - Shift all elements
  - Cost $N$ for $N$-element list
  - Cost $1 + 2 + ... + N = N(N+1)/2$ if repeated
- Add one element to front of LinkedList
  - No shifting, add one link
  - Cost is independent of $N$, constant-time cost
  - Cost $1 + 1 + ... + 1 = N$ if repeated
More on O-notation, big-Oh

- Big-Oh hides/obscures some empirical analysis, but is good for general description of algorithm
  - Allows us to compare algorithms in the limit
    - $20N$ hours vs $N^2$ microseconds: which is better?
- O-notation is an upper-bound, this means that $N$ is $O(N)$, but it is also $O(N^2)$; we try to provide tight bounds.

Formally:
- A function $g(N)$ is $O(f(N))$ if there exist constants $c$ and $n$ such that $g(N) < cf(N)$ for all $N > n$.

Big-Oh calculations from code

- Search for element in an array:
  - What is complexity of code (using O-notation)?
  - What if array doubles, what happens to time?

```java
for(int k=0; k < a.length; k++) {
    if (a[k].equals(target)) return true;
}
return false;
```

- Complexity if we call $N$ times on $M$-element vector?
  - What about best case? Average case? Worst case?

Amortization: Expanding ArrayLists

- Expand capacity of list when `add()` called
- Calling `add N` times, doubling capacity as needed

<table>
<thead>
<tr>
<th>Item #</th>
<th>Resizing cost</th>
<th>Cumulative cost</th>
<th>Resizing Cost per item</th>
<th>Capacity After add</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3-4</td>
<td>4</td>
<td>6</td>
<td>1.5</td>
<td>4</td>
</tr>
<tr>
<td>5-8</td>
<td>8</td>
<td>14</td>
<td>1.75</td>
<td>8</td>
</tr>
</tbody>
</table>

- What if we grow size by one each time?

Some helpful mathematics

- $1 + 2 + 3 + 4 + \ldots + N$
  - $N(N+1)/2$, exactly $N^2/2 + N/2$ which is $O(N^2)$ why?
- $N + N + N + \ldots + N$ (total of $N$ times)
  - $N*N = N^2$ which is $O(N^2)$
- $N + N + N + \ldots + N$ (total of $3N$ times)
  - $3N*N = 3N^2$ which is $O(N^2)$
- $1 + 2 + 4 + \ldots + 2^N$
  - $2^{N+1} - 1 = 2^N - 1$ which is $O(2^N)$
- Impact of last statement on adding $2^N+1$ elements to a vector
  - $1 + 2 + \ldots + 2^N + 2^{N+1} = 2^{N+2} - 1 = 4x2^N - 1$ which is $O(2^N)$

**resizing + copy = total (let x = 2^N)**
### Running times @ 10^6 instructions/sec

<table>
<thead>
<tr>
<th>N</th>
<th>O(log N)</th>
<th>O(N)</th>
<th>O(N log N)</th>
<th>O(N^2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.000003</td>
<td>0.00001</td>
<td>0.000033</td>
<td>0.0001</td>
</tr>
<tr>
<td>100</td>
<td>0.000007</td>
<td>0.00010</td>
<td>0.000664</td>
<td>0.1000</td>
</tr>
<tr>
<td>1,000</td>
<td>0.000010</td>
<td>0.00100</td>
<td>0.010000</td>
<td>1.0</td>
</tr>
<tr>
<td>10,000</td>
<td>0.000013</td>
<td>0.01000</td>
<td>0.132900</td>
<td>1.7 min</td>
</tr>
<tr>
<td>100,000</td>
<td>0.000017</td>
<td>0.10000</td>
<td>1.661000</td>
<td>2.78 hr</td>
</tr>
<tr>
<td>1,000,000</td>
<td>0.000020</td>
<td>1.0</td>
<td>19.9</td>
<td>11.6 day</td>
</tr>
<tr>
<td>1,000,000,000</td>
<td>0.000030</td>
<td>16.7 min</td>
<td>18.3 hr</td>
<td>318 centuries</td>
</tr>
</tbody>
</table>

### Getting in front

- **Suppose we want to add a new element**
  - At the back of a string or an ArrayList or a ...
  - At the front of a string or an ArrayList or a ...
- **Is there a difference? Why? What's complexity?**

- **Suppose this is an important problem: we want to grow at the front (and perhaps at the back)**
  - Think editing film clips and film splicing
  - Think DNA and gene splicing

- **Self-referential data structures to the rescue**
  - References, reference problems, recursion, binky