Announcements

- Assignment 1 was due today.
- Assignment 2 goes out today.
Assignment 2: Serafini

Due next Saturday

Y.E.A.H. Hours
Tonight
Fri, Oct 7th 4-5pm
420-041
Will post video

Due next Saturday
Interesting Puzzle

Counterfeiter

You (Distributor)

User
Fake Medicine is a Problem

700,000 deaths from fake malaria and tuberculosis drugs in 2013 [1]

Equivalent of this many crashes per day

Abstract Data Types

- Vector
- Grid
- Stack
- Queue
- Map
- Set
Vectors

• Can grow to any size you need.
• Supports random access.
Stack

- Last in, First Out (so rude)
- Slightly faster than vector.
- Great style.
- First in, First Out (very fair)
- Slightly faster than vector.
- Great style.
Abstract Data Types

Vector  Grid  Map
Stack   Queue  Set
Today’s Goal

1. Learn how to use Sets
2. Learn how to use Maps
3. Be prepared for Assn 2
Today’s Plan

- Moby
- Sets
- Maps
- Anagrams

Friday
Problem #1: Count Unique Words
Sets

• Collection of elements with no duplicates

Maps

• Collection of Key/value pairs
• Use a key to find its associated value;

Both are blindingly fast to lookup an element
Sets

**set**: A collection of values with no duplicates.

Add, contains, remove operation are all fast
We don’t think of sets as having indices

```
set.contains("to")  # true
set.contains("be")  # false
```

```
set = 
"if"  "the"  "of"
"down"  "from"
"by"  "she"  "you"
"in"  "why"  "him"
```
Set<string> friends;
friends.add("chris");
friends.add("anton");
cout << friends.contains("voldemort") << endl;
for(string person : friends) {
    cout << person << endl;
}
The Set Essentials

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>set.size()</code></td>
<td>Returns the number of elements in the set.</td>
</tr>
<tr>
<td><code>set.add(value)</code></td>
<td>Adds a new value to the set (ignores it if the value already was in the set).</td>
</tr>
<tr>
<td><code>Set.contains(value)</code></td>
<td>Return true if the value is in the set.</td>
</tr>
</tbody>
</table>

Sets also have `remove`, `isEmpty` and other helpful methods. See the online docs for more.
Looping Over a Set

```cpp
for (type currElem : set) {
    // process elements one at a time
}
```

can't use a normal `for` loop and get each element `[i]`

```cpp
for (int i = 0; i < set.size(); i++) {
    // does not compile
    cout << set[i];
}
```
Types of Sets

Set

- Iterate over elements in sorted order
- Really Fast
- Implemented using a binary search tree

HashSet

- Element order is jumbled
- Really, Ridiculously Fast
- Implemented using a hash table
Count Unique Words: Revisited
### Set Operands

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>s1 == s2</code></td>
<td>true if the sets contain exactly the same elements</td>
</tr>
<tr>
<td><code>s1 != s2</code></td>
<td>true if the sets don't contain the same elements</td>
</tr>
<tr>
<td><code>s1 + s2</code></td>
<td>returns the union of <code>s1</code> and <code>s2</code> (elements from either)</td>
</tr>
<tr>
<td><code>s1 * s2</code></td>
<td>returns intersection of <code>s1</code> and <code>s2</code> (elements in both)</td>
</tr>
<tr>
<td><code>s1 - s2</code></td>
<td>returns difference of <code>s1</code>, <code>s2</code> (elements in <code>s1</code> but not <code>s2</code>)</td>
</tr>
</tbody>
</table>
Set\langle string\rangle\ allStudents = loadStanford();

Set\langle string\rangle\ left;
for(string student : allStudents){
    if(audienceLeft(student)) {
        left.add(student);
    }
}

Set\langle string\rangle\ hats;
for(string student : allStudents){
    if(isWearingHat(student)) {
        hats.add(student);
    }
}
Stand allStudents, hats, left;
Set Operands

allStudents, hats, left

stand(left);
Set Operands

allStudents, hats, left

Set<string> combine;
combine = left + hats;
stand(combine);

Elements in either set
Set Operands

allStudents, hats, left

Set<string> combine;
combine = left * hats;
stand(combine);

Elements in both sets
Today’s Plan

- Anagrams
- Maps
- Sets
- Moby

Friday
**Maps**

**map**: A collection of **pairs** \((k, v)\), sometimes called key/value pairs, where \(v\) can be found quickly if you know \(k\).

a.k.a. dictionary, associative array, hash

a generalization of an array, where the "indexes" need not be **ints**.
A map allows you to get from one half of a pair to the other. 

*Store an association from "Suzy" to "206-685-2181".*

```java
// key value
// m["Suzy"] = "206-685-2181";
m.put("Suzy", "206-685-2181");
```

What was Suzy’s phone number?

```java
// m["Suzy"]
m.get("Suzy")
```

"206-685-2181"
Maps are Everywhere

Key = Title, Value = Article

**Key:**
“Mantis Shrimp”

**Value:**
Mantis shrimp
From Wikipedia, the free encyclopedia

*Mantis shrimp* or *stomatopods* are marine crustaceans, the members of the order *Stomatopoda*. Most species can grow to around 10 centimetres (3.9 in) in length, though a few species reach up to 36 cm (14 in). The largest ever caught has a length of 46 cm (18 in) in the ocean near Fort Pierce, Florida of USA. The compound eyes of mantis shrimp cover only the rear part of the head and the first four segments of the thorax. There are more than 450 species of mantis shrimp. Varieties range from shades of brown to vivid colours, and are among the most important predators in many shallow, tropical and subtropical marine habitats. Despite being common, they are poorly understood as many species spend most of their life tucked away in burrows and holes.  

**Key:**
“Antelope Canyon”

**Value:**
Antelope Canyon is a slot canyon in the American Southwest. It is located on Navajo land east of Page, Arizona. Antelope Canyon includes two separate, photogenic slot canyon sections, referred to individually as Upper Antelope Canyon or The Crack, and Antelope Canyon or The Contour. The Navajo name for Upper Antelope Canyon is T'al-bi-ni-bi-shi, which means “the place where water runs through rocks.” Lower Antelope Canyon is renowned (advertised as “Horseshoe Bend” by the Navajo Parks and Recreation Department) or “tower rock arches.” Both are located within the Lee’s Ferry Chapter of the Navajo Nation.  

Contents [hide]
1 Geology
2 Tourism and photography
2.1 Upper Antelope Canyon
Creating Maps

Requires 2 type parameters: one for keys, one for values.

```
// maps from string keys to integer values
Map<string, int> votes;

// maps from double keys to Vector<int> values
Map<string, Vector<string>> friendMap;
```
### Map Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>m.clear();</code></td>
<td>removes all key/value pairs from the map</td>
</tr>
<tr>
<td><code>m.containsKey(key)</code></td>
<td>returns true if the map contains a mapping for the given key</td>
</tr>
<tr>
<td><code>m[key]</code> or <code>m.get(key)</code></td>
<td>returns the value mapped to the given key; if key not found, <strong>adds</strong> it with a default value (e.g. 0, &quot;&quot;)</td>
</tr>
<tr>
<td><code>m.isEmpty()</code></td>
<td>returns true if the map contains no k/v pairs (size 0)</td>
</tr>
<tr>
<td><code>m.keys()</code></td>
<td>returns a Vector copy of all keys in the map</td>
</tr>
<tr>
<td><code>m[key] = value;</code> or <code>m.put(key, value);</code></td>
<td>adds a mapping from the given key to the given value; if the key already exists, <strong>replaces</strong> its value with the given one</td>
</tr>
<tr>
<td><code>m.remove(key);</code></td>
<td>removes any existing mapping for the given key</td>
</tr>
<tr>
<td><code>m.size()</code></td>
<td>returns the number of key/value pairs in the map</td>
</tr>
<tr>
<td><code>m.toString()</code></td>
<td>returns a string such as &quot;{a:90, d:60, c:70}&quot;</td>
</tr>
<tr>
<td><code>m.values()</code></td>
<td>returns a Vector copy of all values in the map</td>
</tr>
</tbody>
</table>
Map Example

Map<\texttt{string}, \texttt{string}> \texttt{wiki};

// adds name / text pair to dataset
\texttt{wiki.put("Stanford Band", articleHTML);};

// returns corresponding articleHTML
\texttt{cout << \texttt{wiki.get("Mantis Shrimp");}};

// removes the article
\texttt{wiki.remove("Britain in the E.U.");}
Types of Maps

Map
- Iterate over elements in sorted order
- Really Fast
- Implemented using a binary search tree

HashMap
- Element order is jumbled
- Really, Ridiculously Fast
- Implemented using a hash table
## Tallies

**count digits:** 22092310907

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
**Tallies**

Count votes:
```
// (M)ilk, (S)tokes, (R)ogers
"MMMRMSSMSSMMMMRRRMMMMRRRRM" 
```

<table>
<thead>
<tr>
<th>key</th>
<th>&quot;M&quot;</th>
<th>&quot;S&quot;</th>
<th>&quot;R&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>17</td>
<td>7</td>
<td>3</td>
</tr>
</tbody>
</table>

```
keys

"S"    

"R"    

values

7

3

17

*In 1976 Harvey Milk became the first openly gay elected official in the US*
Problem #2: Tally Words
Map<string, double> gpa = load();

for (string name : gpa) {
    cout << name << "'s GPA is ";
    cout << gpa[name] << endl;
}

*The order is unpredictable in a HashMap*
Today's Plan

- Anagrams
- Maps
- Sets
- Moby
Write a program to find all anagrams of a word the user types.

Type a word [Enter to quit]: scared
Anagrams of scared:
cadres
cedars
sacred
scared

What is the appropriate collection to use to solve this problem?

Hint: Use a compound collection...
Insight

Every word has a sorted form where its letters are arranged into alphabetical order

<table>
<thead>
<tr>
<th>word</th>
<th>sorted form</th>
</tr>
</thead>
<tbody>
<tr>
<td>fare</td>
<td>aefr</td>
</tr>
<tr>
<td>fear</td>
<td>aefr</td>
</tr>
<tr>
<td>swell</td>
<td>ellsw</td>
</tr>
<tr>
<td>wells</td>
<td>ellsw</td>
</tr>
</tbody>
</table>

Notice that anagrams have the same sorted form as each other
Lexicon: A set of vocabulary

Lexicon: Special class for sets of strings

- Often used to store vocabularies
- Great syntax for loading from file.
- Same methods as set.
- Also has a `containsPrefix` method.

Historically the Lexicon was created for file size reasons (3MB was too large)
Lexicon Example

// great syntax for loading from file
Lexicon english("english-dictionary.txt");

// true: hello is a word
cout << english.contains("hello") << endl;

// true: hel is not a word but its a prefix for hello
cout << english.containsPrefix("hel") << endl;

// same syntax for looping over all elements as set
for(string word : english) {
    cout << word << endl;
}
Problem #3 Anagrams
Aka how to beat your friends at word games
I’m really into this anagram solution. It feels like it can be used for so much more…
Bananagrams

Bananagrams vs computer

Bananagrams online

number of legal moves in bananagrams

different species of bananas
This may be the first…
Bananagrams

Tiles: DTGEMAOTOWKJEAAESELXO
Bananagrams code online
Today’s Goal

1. Learn how to use Sets (and Lexicons)
2. Learn how to use Maps
3. Be prepared for Assn 2
What Data Structure?

A. Vector<int>
B. Set<int>
C. Map<int, string>
D. Stack<int>
M Pedigree

342302033 → “legitimate”

927216398 → “used”

927216398 → “legitimate”
M Pedigree

Random number

342302033

Unique Id
Happy Friday
Extra Reading

Hashing: https://en.wikipedia.org/wiki/Hash_table

Relational Databases: https://en.wikipedia.org/wiki/Relational_database
(especially indices)