CIF Crash Course - Lecture 3

Intro to Data Structures

Most useful data structures

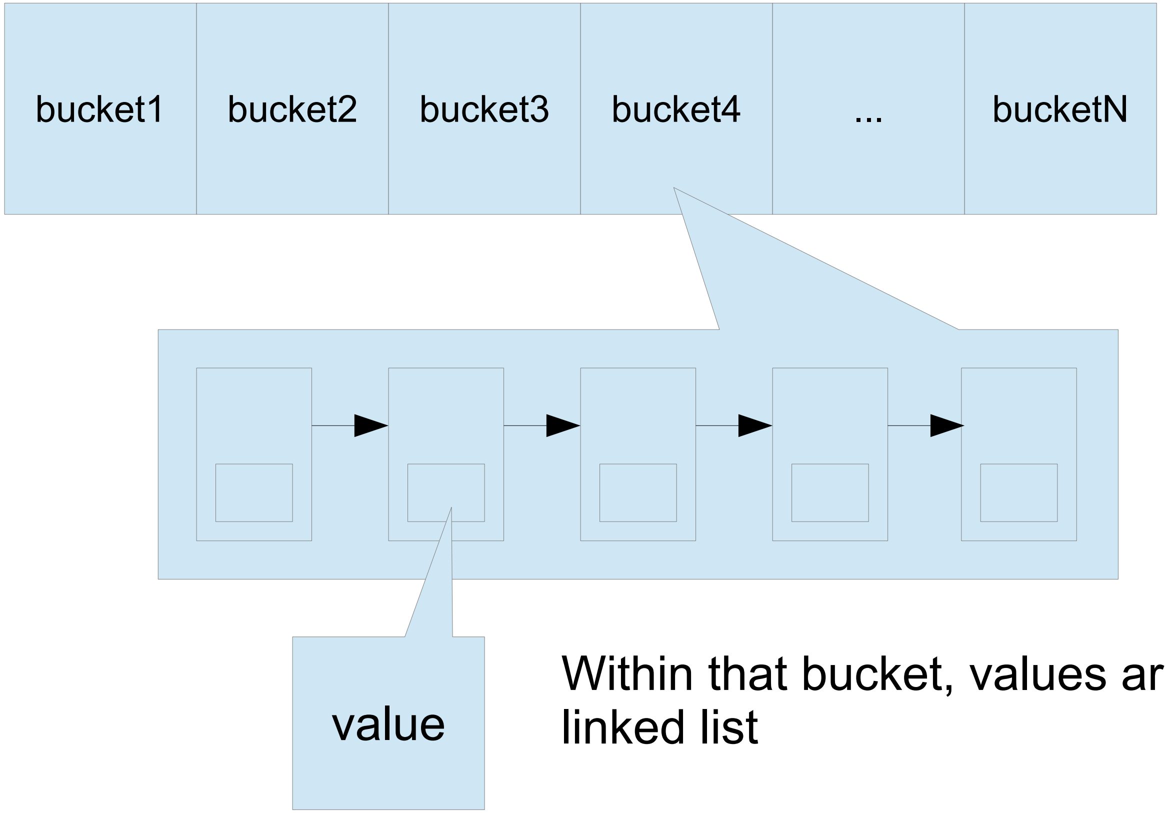
- Lists
  - Array
  - ArrayList
  - LinkedList
    - Single linked list
    - Double linked list
  - CircularList

- Maps
  - HashMap
  - TreeMap

Hashing

How does a hash map work?

A hash function uses the hashCode method of a key to determine which bucket holds a value. A good hash function evenly distributes the keys.



Within that bucket, values are held in a linked list

How does a hashing algorithm work?

- A good hashing algorithm is one that can quickly and inexpensively (in terms of computing resources) map keys uniformly to a set of bins
- For example, suppose that we can quickly compute a hash code for a key and that hash code doesn't appear to have any particular pattern – it's just a number between 1 and 2 billion
- That hash code modulo the number of bins in our hash will be a good way to evenly distribute our objects among our bins

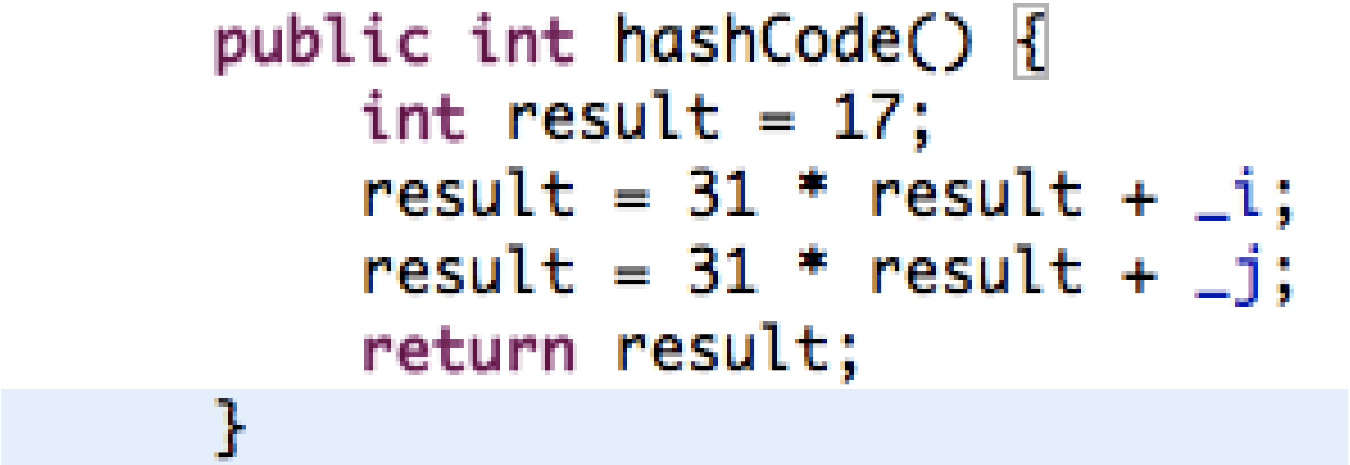
The subject of hashing is huge.

See article on hashes in Wikipedia

See example of how to write the equals / hashCode methods

Test_Bad_HashCode.java

How to write hash code



```
public int hashCode() {
    int result = 17;
    result = 31 * result + _i;
    result = 31 * result + _j;
    return result;
}
```

If a String or other object is one of the fields, call its hashCode method and use the integer as one of the fields

Should we cache the hash code?

- If a hash code is expensive to calculate – many fields – and the hashCode method will be called frequently, the hashCode should be calculated when there's a change in state and saved to an instance variable
- Be careful that you can still find the object in a collection when a state change alters its hash code

Suppose we know keys ahead of time, eg list of all tickers on NYSE?

We can make a “perfect hash” - a hash that avoids all collisions

gperf – perfect hash builder

- Open a terminal on your Mac or Cygwin on your Windows machine or go to a terminal window on your Unix / Linux machine
- Create a text file with two symbols
  - Edit the file using any editor, eg: vim zz.txt
  - File contents
    - IBM
    - DELL
- Call gperf to create a perfect hash – no collissions:
  - gperf < zz.txt

gperf is clever!

```
static const char * wordlist[] =
    {
      "", "", "",
      "IBM",
      "DELL"
    };
```

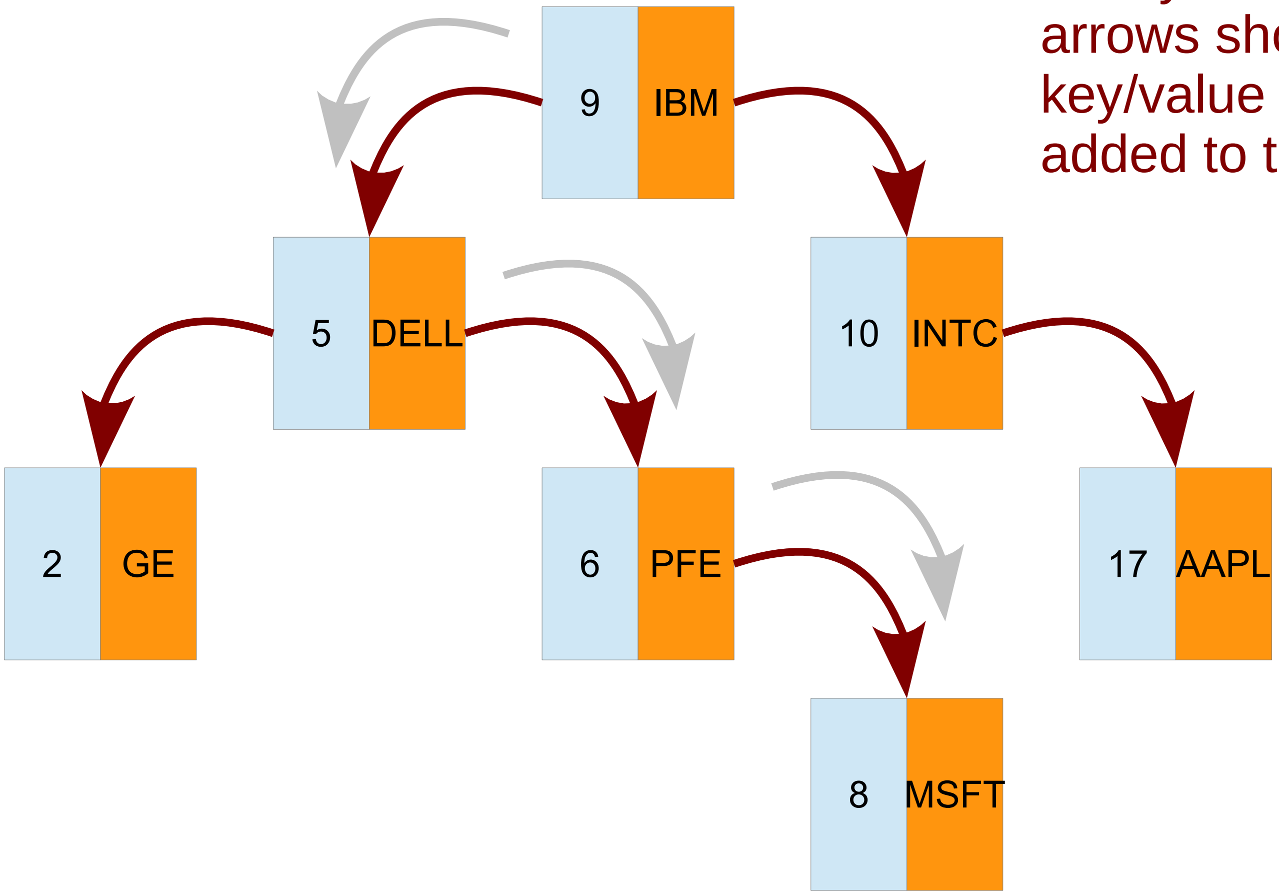
Length = 3

Length = 4

In this case, it sets up an array that uses the length of the two strings as an index to find them!

Binary Trees

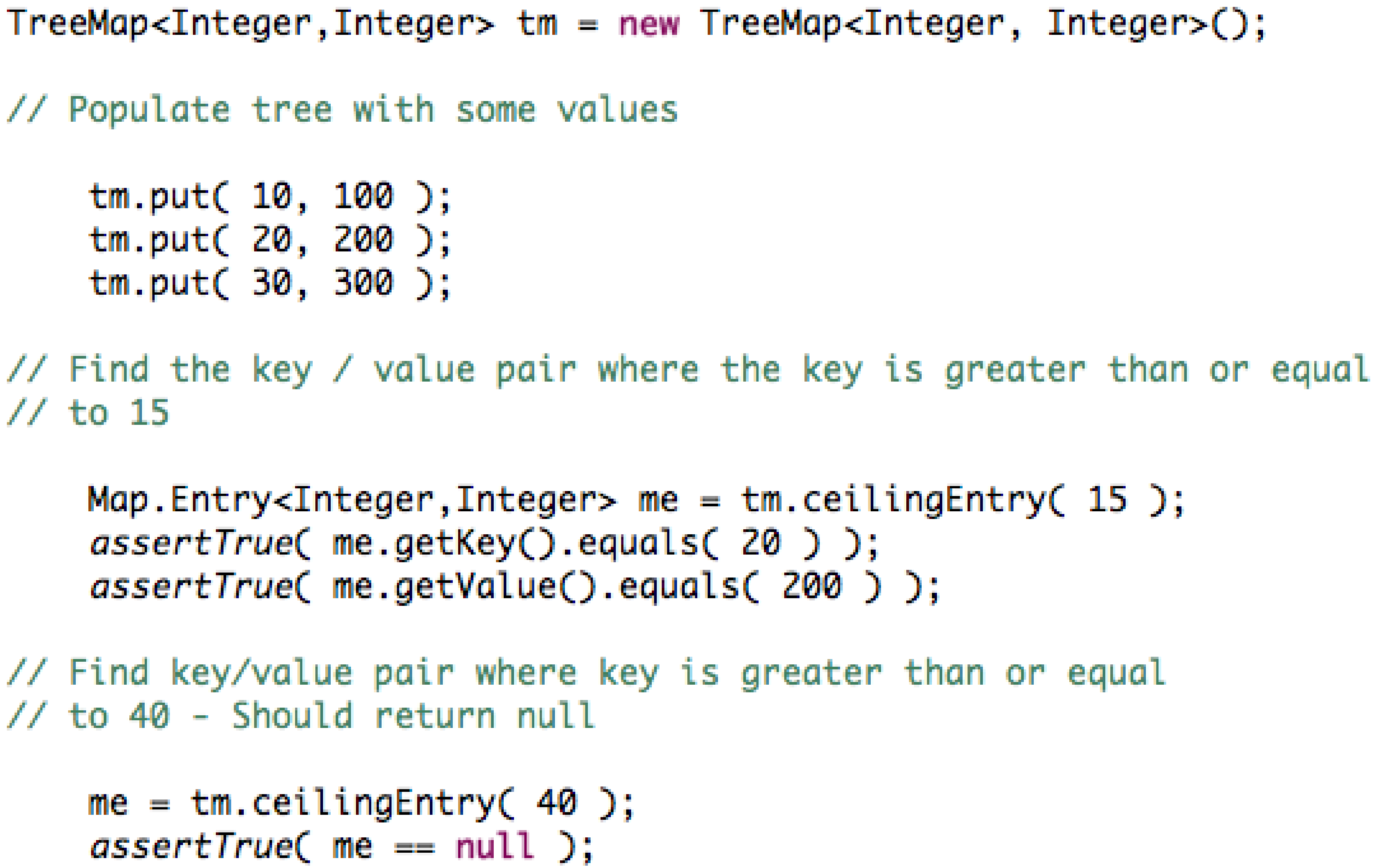
To efficiently model the order book, we will need a tree map implementation



The red arrows are links between key/value pairs in a binary tree map. The gray arrows show how the key/value pair 8/MSFT is added to this tree.

Binary tree data structure

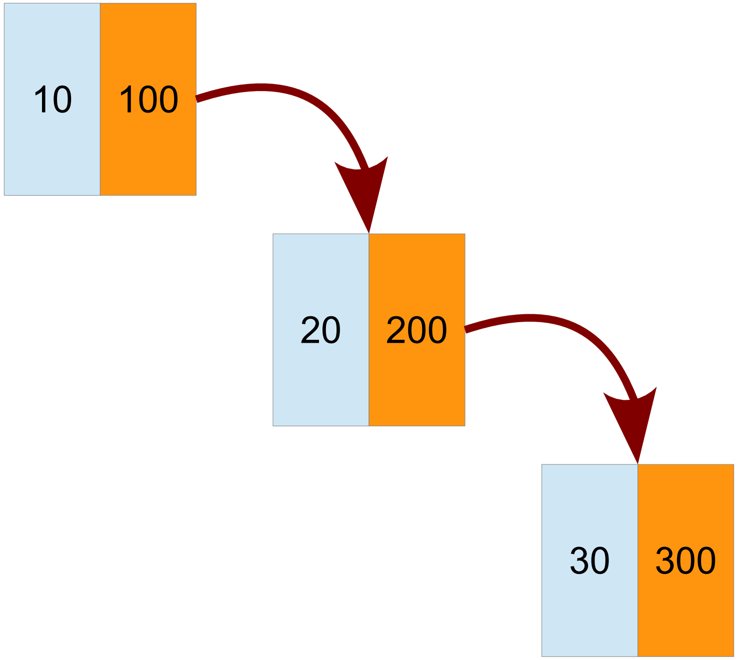
- Disadvantages compared to hash map
  - Slower look-up than a hash map for a well written hash function
  - Requires a comparator
- Advantages compared to hash map
  - Always in order
    - Allows us to find a key that is greater than or less than some other key
    - In a hash map, keys are distributed randomly among the hash buckets



```
TreeMap<Integer,Integer> tm = new TreeMap<Integer, Integer>();

// Populate tree with some values

    tm.put( 10, 100 );
    tm.put( 20, 200 );
    tm.put( 30, 300 );

// Find the key / value pair where the key is greater than or equal
// to 15

    Map.Entry<Integer,Integer> me = tm.ceilingEntry( 15 );
    assertTrue( me.getKey().equals( 20 ) );
    assertTrue( me.getValue().equals( 200 ) );

// Find key/value pair where key is greater than or equal
// to 40 - Should return null

    me = tm.ceilingEntry( 40 );
    assertTrue( me == null );
```

What kind of map will this code produce?

The order in which we added key/value pairs to this map is not random, so we got a map that is out of balance. The search times for this map will not be O(log(n)) but O(n)



To prevent this from happening, the Java collections framework implements the tree as a Red/Black tree, a version of the binary tree that keeps track of whether each node is in balance. When the tree is significantly out of balance, it is rebuilt.