CSM 61B
Fall 2017

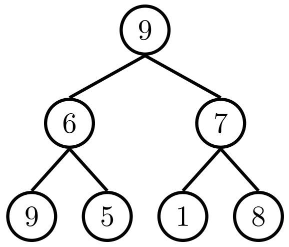
Heaps & Hashing

Mentoring 9: October 23, 2017

1 Min-Heapify This

1.1 In general, there are 4 ways to heapify. Which 2 ways actually work?

- Level order, bubbling up
- Level order, bubbling down
- Reverse level order, bubbling up
- Reverse level order, bubbling down



1.2 (a) Show the heapification of the tree.

(b) Now, insert the value 2.

(c) Finally, remove the value 1.

2 Motivation

2.1 (a) In the worst case, how long does it take to index into a linked list?

(b) In the worst case, how long does it take to index into an array?

(c) In the worst case, how long does it take to insert into a linked list?

(d) Assuming there's space, how long does it take to put a element in an array?

(e) What if we assume there is no more space in the array?

(f) Given what we know about linked lists and arrays, how could we build a data structure with efficient access and efficient insertion?

3 Hash Table Basics

3.1
```
public class BadHashMap<K, V> implements Map<K, V> {
    private V[] array;
    public void put(K key, V value) {
        this.array[key.hashCode() % this.array.length] = value;
    }
}
```

```
interface Map<K, V> {
    boolean containsKey(K key);
    V get(K key);
    void put(K key, V value);
    int size();
}
```

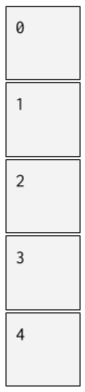
(a) Why do we use the `%` (modulo) operator?

(b) What are collisions? What data structure can we use to address them?

(c) Why is this a bad `HashMap`?

3.2 (a) Draw the diagram that results from the following operations on a Java `HashMap`. `Integer::hashCode` returns the integer's value.

```
put(3, "monument");
put(8, "shrine");
put(3, "worker");
put(5, "granary");
put(13, "worker");
```



(b) Suppose a resize occurs, doubling the array to size 10. What changes?

4 Hash Codes *Extra Practice*

4.1 What does it mean for a hashcode to be valid?

4.2 Which of the following hashcodes are valid? Good?

```java
class Point {
    private int x, y;
    private static int count = 0;
    public Point(int x, int y) {
        this.x = x;
        this.y = y;
        count += 1;
    }
}
```

(a)
```java
public void hashCode() {
    System.out.print(this.x + this.y);
}
```

(b)
```java
public int hashCode() {
    Random random = new Random();
    return random.nextInt();
}
```

(c)
```java
public int hashCode() {
    return this.x + this.y;
}
```

(d)
```java
public int hashCode() {
    return count;
}
```

(e)
```java
public int hashCode() {
    return 4;
}
```