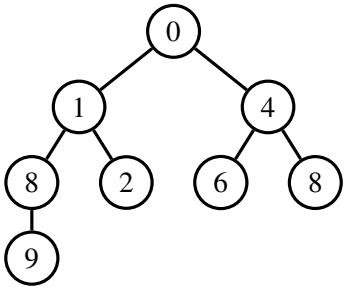
CS 61B Summer 2020 | Small Group Tutoring
Section 9: Hashing and Heaps | Worksheet 9

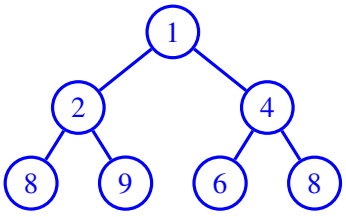
1 Basic Operations

1. **Min Heap**

   (a) Draw the Min Heap that results if we delete the smallest item from the heap.

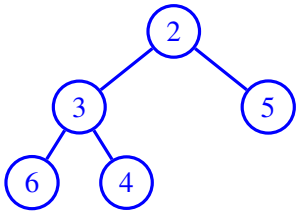


Solution:



   (b) Draw the Min Heap that results if we insert the elements 6, 5, 4, 3, 2 into an empty heap.

Solution:



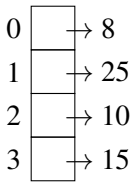
   (c) Given an array, heapify it such that after heapification it represents a Max Heap.

```
int[] a = {5, 12, 64, 1, 37, 90, 91, 97}
```

   After heapifying the array to represent a Max Heap, the array's elements will be ordered as follows: $\{97, 37, 91, 12, 5, 90, 64, 1\}$.

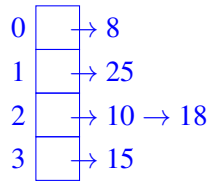
2. **External Chaining**

   Consider the following External Chaining Hash Set below, which resizes when the load factor reaches 1.5. Assume that we're using the default hashCode for integers, which simply returns the integer itself.



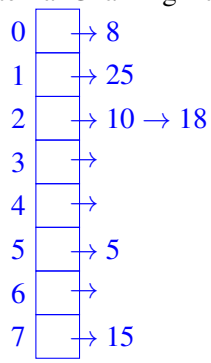
(a) Draw the External Chaining Hash Set that results if we insert 18.

Solution:



(b) Draw the External Chaining Hash Set that results if we insert 5 after the insertion done in part (a).

Solution:



2 Invalid Hashing

Which of the hashCodes are invalid? Assume we are trying to hash the following class:

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

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

Solution: Invalid. The return type of the hashCode function should be an integer.

(b) public int hashCode() {
Random randomGenerator = new Random();
return randomGenerator.nextInt(Int); }

Solution: Invalid. Using a random generator results in the hashcode function to be not deterministic. There is a possibility that the hashcode for the same Point object could be different across two attempts to hash the object.

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

Solution: Valid. This is a good and safe hashcode function because the fields this.x and this.y are private variables. However, if these fields were not private, then any changes made the variables would result in an entirely different hashcode function. For instance, assume that the variables x and y are public. First, hash a Point object and then change its variables (x and y). Now, it will be impossible to find the initial Point object because the hashcode function will not return the same hash value.

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

Solution: Invalid. Since count is a static variable (it is shared by all instances of Point), the same object will not return the same hash code if another Point instance is created between calls.

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

Solution: Valid. However, this is a bad hashcode function because all Point objects will be put in to the same bucket.

3 Search Structures Runtime

Assume you have N items. Using Theta notation, find the worst case runtime of each function.

Function	Unordered List	Sorted Array	Bushy Search Tree	"Good" Hash Table	Heap
find	Θ(N)	$\Theta(\log N)$	$\Theta(\log N)$	$\Theta(1)$	$\Theta(N)$
add (Amortized)	$\Theta(1)$	$\Theta(N)$	$\Theta(\log N)$	$\Theta(1)$	$\Theta(\log N)$
find largest	$\Theta(N)$	$\Theta(1)$	$\Theta(\log N)$	$\Theta(N)$	$\Theta(1)$
remove largest	$\Theta(N)$	$\Theta(1)$	$\Theta(\log N)$	$\Theta(N)$	$\Theta(\log N)$