CS 61B Spring 2017 Guerrilla Section 5 Worksheet

18 March 2017

Directions: In groups of 4-5, work on the following exercises. Do not proceed to the next exercise until everyone in your group has the answer and *understands why the answer is what it is*. Of course, a topic appearing on this worksheet does not imply that the topic will appear on the midterm, nor does a topic not appearing on this worksheet imply that the topic will not appear on the midterm.

1 Hashing Mechanics

Suppose we insert the following words into an initially empty hash table, in this order: **galumphing, frumious, slithy, borogroves, mome, bandersnatch**. Assume that the hash code of a String is just its length (note that this is not actually the hash code for Strings in Java). Use external chaining to resolve collisions. Assume 4 is the initial size of the hash table's internal array, and double this array's size when the load factor is equal to 1. Illustrate this hash table below with a box-and-pointer diagram.

STOP!

Don't proceed until everyone in your group has finished and understands all exercises in this section!

2 Hash Code Design

Describe a potential problem with each of the following:

1. An implementation of the hashCode method of the String class that simply returns the length of the string (i.e. the hash code used in problem 1).

2. An implementation of the hashCode method of the String class that simply returns a random number each time.

3. Overriding the equals method of a class without overriding the hashCode method.

4. Overriding the hashCode method of a class without overriding the equals method.

5. Modifying an object after inserting it into a HashSet.

STOP!

Don't proceed until everyone in your group has finished and understands all exercises in this section!

3 Hash Table Runtimes

1. What is the worst-case runtime for inserting a single entry into a HashMap containing N elements? You may assume that hashing a key requires a constant amount of time. Provide your answer using Θ notation.

2. Describe a situation in which we would achieve the above runtime.

3. After finishing Lab 9, Janice decides to create her own hash table implementation: `SmallMap`. In order to avoid costly resizing operations, `SmallMap`'s internal array does not resize; it has a fixed length of 1,024. What is the best-case runtime for insertion into a `SmallMap` containing N elements? The worst-case runtime? Assume that the N elements are distributed uniformly. Provide your answer using Θ notation.

STOP!

Don't proceed until everyone in your group has finished and understands all exercises in this section!

4 Writing Hashcode Functions

Implement a `hashCode` function for the class `Politician`. Try to design your hash code to minimize collisions (e.g. George H. W. Bush and George W. Bush should hash to different buckets). Your hash code does not need to be a perfect hash.

```
public class Politician {
    String firstName;
    String lastName;
    int birthYear;

    @Override
    public boolean equals(Object o) {
        if(this.getClass() != o.getClass()) {
            return false;
        }
        Politician other = (Politician) o;
        return other.firstName.equals(this.firstName) &&
                other.lastName.equals(this.lastName) &&
                 other.birthYear == this.birthYear;
    }

    @Override
    public int hashCode() {




    }
}
```

Now, write a `hashCode` method for the `BinString` class. A binary string is a string made up of just two characters: 0 and 1.

```
public class BinString {
    String binStr;

    @Override
    public boolean equals(Object o) {
        return this.getClass() == o.getClass() && ((BinString) o).binStr.equals(this.binStr);
    }

    @Override
    public int hashCode() {









    }
}
```

5 Assorted Heap Questions

(a) Describe a way to modify the usual max heap implementation so that finding the minimum element takes constant time without incurring more than a constant amount of additional time and space for the other operations.

(b) In class, we looked at one way of implementing a priority queue: the binary heap. Recall that a binary heap is a nearly complete binary tree such that any node is smaller than all of its children. There is a natural generalization of this idea called a d-ary heap. This is also a nearly complete tree where every node is smaller than all of its children. But instead of every node having two children, every node has d children for some fixed constant d.

i. Describe how to insert a new element into a d-ary heap (this should be very similar to the binary heap case). What is the running time in terms of d and n (the number of elements)?

ii. What is the running time of finding the minimum element in a d-ary heap with n nodes in terms of d and n?

iii. Describe how to remove the minimum element from a d-ary heap (this should be very similar to the binary heap case). What is the running time in terms of d and n?

(c) Suppose a value in a max heap were changed. To recreate the heap to reflect the modified value, would you bubble the value up, bubble it down, both, or neither? Briefly defend your answer.

STOP!

Don't proceed until everyone in your group has finished and understands all exercises in this section!

6 Zork Sort

Suppose you are given an array A with n elements such that no element is more than k slots away from its position in the sorted array. Assume that $k > 0$ and that k, while not constant, is much less than n ($k \ll n$).

(a) Fill in the blanks such that the array A is sorted after execution of this method. The important operations on a `PriorityQueue` are `add(x)`, `remove()` (remove smallest), and `isEmpty()`. Your solution should be as fast as possible.

```
public static void zorkSort(int[] A, int k) {
        int n = A.length;
        int i = 0;
        PriorityQueue<Integer> pq = new PriorityQueue<Integer>();
        while (i < k) {
                ___________________________________________
                i += 1;
        }
        while (_____________________) {
                A[i - k] = ____________________________________
                ___________________________________________
                i += 1;
        }
        while (_____________________) {
                ___________________________________________
                i += 1;
        }
}
```

(b) What is the running time of this algorithm, as a function of n and k? Justify your answer.

STOP!

Don't proceed until everyone in your group has finished and understands all exercises in this section!

7 Linguistics

Donald Trump's speaking style has attracted a great deal of attention in recent weeks. As curious students, we're interested in learning what Trump's favorite words are. Consider this problem: we'd like to determine the top k most common words found in the list of all n words that The Donald has ever spoken. Explain how you would solve this problem efficiently and describe which data structure(s) you would use. Hint: we want our algorithm to return Trump's k favorite words in $O(n \log(k))$. You may assume $k \ll n$.

STOP!

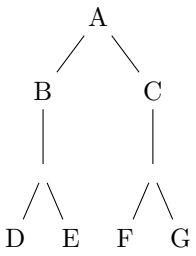
Don't proceed until everyone in your group has finished and understands all exercises in this section!

8 Hidden Message

(a) Given the post-order sequence (G A U F H) and in-order sequence (G U A H F), construct the tree and find the pre-order sequence.

(b) Given the post-order sequence (T R D I E) and pre-order sequence (E D T R I), construct the tree and find the in-order sequence. For this question you can assume the tree is full (a full tree means that each node will always have two children).

(c) Suppose we are searching for nodes in the following tree:



To search this tree, we may use either of the following two traversal methods: preorder depth-first and level-order. Assume that both traversals will break ties by going left. Which method will require us to look at the fewest nodes, assuming we're looking for node D? What about for nodes C and G? Keep in mind the answer may be a tie.

STOP!

Don't proceed until everyone in your group has finished and understands all exercises in this section!