FINAL SAMPLE PROBLEMS

David Kauchak
CS62 – Spring 2021

Admin

Assignment 10 due tomorrow

Last mentor hours tomorrow

Final exam 5/12, 2-5pm (PST)

- 4 pages of notes (2 double-sided)
- sakai (can take on sakai or pdf)
- Will post zoom link
- If you can't take it then, let me know and you can take it any time that day

Sample question 1: True or False

A binary tree with *n* nodes has at most height O(*log n*)

Heapsort is always O(n log n)

A complete graph is connected.

The following code will print out 1:

```
String[] strings = new String[10];
strings[0] = "banana";
System.out.println(strings.length);
```

Sample question 2

Fill in the details for the helper findLessThan helper method (see next slide) that takes a Node, a value, and an ArrayList and populates the ArrayList with all data items in the tree that are less than the value passed in. You should not explore more of the tree than you need to (e.g., don't traverse the entire tree!).

You may assume that the value in the tree are unique. You can directly access the node instance variables or assume set/get methods, whichever you prefer.

```
public class BinaryTree<E extends Comparable<E>> {
    private Node root;

    public ArrayList<E> findLessThan(E value) {
        ArrayList<E> values = new ArrayList<E>();
        findLessThan(root, value, values);
        return values;
    }

    private void findLessThan(Node t, E value, ArrayList<E> list) {
        // fill in details here!
    }

    private class Node {
        private E value;
        private Node left;
        private Node right;

        public Node(Node left, Node right, E value) {
            this.left = left;
            this.right = right;
            this.value = value;
        }
    }
}
```

Sample question 3

What is a hashtable collision? Why do they happen? What are two ways of dealing with them?

Sample question 4

What is the difference between strongly connected and connected graphs? Why do we make the distinction?

Sample question 5

Explain the problem with the following code that tries to print out all of the even values and suggest how to fix it:

```
public void printEven(Iterator<Integer> iter) {
  while (iter.hasNext()) {
    if (iter.next() % 2 == 0) {
      System.out.println(iter.next());
    }
  }
}
```

Sample question 6

In a graph where the shortest path from A to B is 3 edges, while the shortest path from A to C is 7 edges, **will breadth-first search** starting at A explore B or C first, or does it depend on the structure of the graph? **Justify your answer.**

Sample question 7

a. **Draw a diagram** of a doubly-linked list with both head and tail pointers that contains the elements 17, 23, and 31. **Use arrows to indicate** pointers, including the head and tail pointers, and the next and previous pointers of each node. **For null pointers**, draw an arrow pointing to the word "NULL."

b. Assuming we wanted to insert the value 47 between the 23 and 31 in our list, **write a list of which pointers** would have to be changed. **Include pointers** that are part of the new node. You can label nodes based on their values, e.g., 17.next would be the next reference for the node with value 17.

Sample question 8

Draw a directed, acyclic graph with 5 nodes that is **not** a tree that has at least 5 edges (and no self edges). *Label your nodes with the letters A through E.*

Sample question 9

Assuming that the following method is called a node of a full *binary search tree* the node is not a leaf, what does the method below do?

```
public String mystery(Node t){
  Node temp = t.right;

  while( temp.left != null ){
    temp = temp.left;
  }

  return temp.value
}
```

Sample question 1 solution

False. At most O(n)

True.

True. It has edges between every pair of vertices.

False. Even though it only has one entry in it, strings will always have length 10 since an arrays length never changes.

Sample question 2 solution

```
private void findLessThan(Node t, E value, ArrayList<E> list) {
    if( t != null ) {
        if( t.value.compareTo(value) < 0 ) {
            list.add(t.value);
            // need to explore both left and right since
            // values to the right could be also be less than
            findLessThan(t.left, value, list);
            findLessThan(t.right, value, list);
        } else {
            // look for values that are less, which is in the left branch
            findLessThan(t.left, value, list);
        }
    }
}
```

Sample question 3: solution

Collisions happen when you have two **different** objects (i.e., objects that are not equals) that have the same hashCode

They happen because hash codes are mapping from a larger space of possibilities (e.g., all strings) down to a finite set of hash codes (e.g., 64-bit integers)

Collision resolution by chaining or open addressing

Sample question 4: solution

They're similar ideas. Both state that a path exists from any vertex to every other vertex.

Strongly-connected is for directed graphs, while connected is for undirected.

We make the distinction to allow for a distinction in directed graphs between strongly-connected and something weaker, e.g., weakly-connected, where each vertex is adjacent to at least one other vertex (visually, it would look "connected").

Note: We did not talk about weakly connected in class, so I wouldn't expect you to actually use this terminology, but to note how directed graphs are more nuanced.

Sample question 5: solution

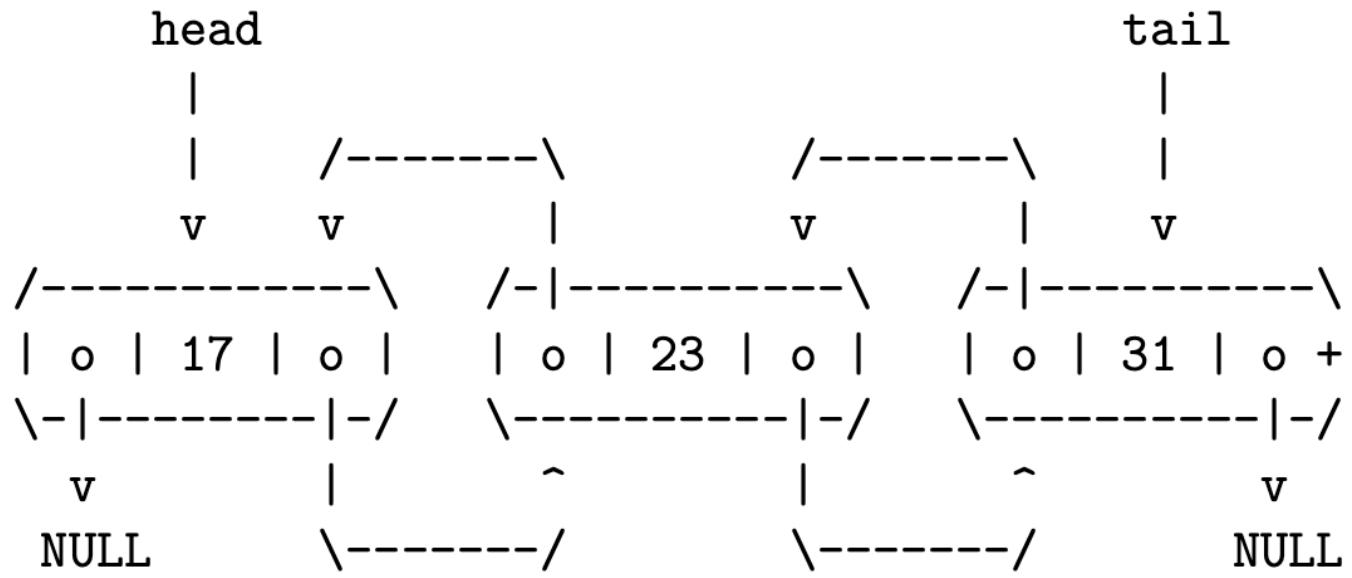
This code calls next once in the condition of the while loop, and then again in the body, getting two different values each time. It needs to call next only once, and store the result.

Sample question 6: solution

It will explore B first, because breadth-first search always explores closer nodes (in terms of path length) before farther ones.

Sample question 7: solution

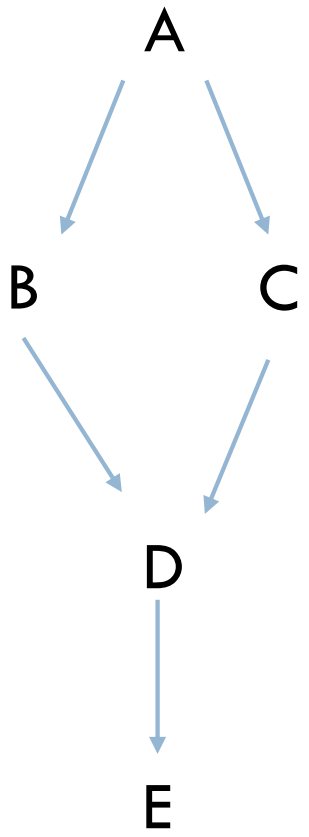
a.



b.

23.next
31.previous
47.previous
47.next

Sample question 8: solution



Sample question 9: solution

Finds the value that comes immediately after the root in sorted order (i.e., the successor)