Exam 3 Study Guide

Thursday, July 20, 2023, 9 AM - 10:50 AM
Building 15, 3rd floor, RF-Smart Lab, Room 3130

- ☞ If possible, arrive a few minutes early, login to a desktop and your Canvas (Duo app maybe needed).
- ☞ Cheats sheets: ***upto five***, written/typed on both sides; feel free to include slides and diagrams. You can use the same cheat sheets for both the exams.
- ☞ Feel free to bring blank papers and pencils for scratch work.
- ☞ Feel free to use earplugs to avoid getting distracted.

1 Coding, 50 points, 1 hr 15 minutes, 9 AM - 10:15 AM + 5 minutes for uploading the Java file; cheats sheets are allowed

Exam 3 coding will be worth 50 points. You will be asked to complete 2 methods. There will be a third bonus method worth 15 points. Code outline and tests will be given to you, like HW problems. Like the HWs, you need to download the package, complete coding, test, and then upload your code to Canvas. Multiple submissions are allowed. Your latest submission will be graded.

Syllabus. **Mainly binary trees but basic understanding of other data structures such as stacks, queues, heaps, BSTs may be needed for some of the methods. See the practice exam on GitHub to learn more.** ***On the test, if required, the related data structure classes will be included inside the IntelliJ project. However, you may also use the built-in Java data structures if you want to.***

Practice problems. **Pull from GitHub. Download the package exam3 and start practicing. The problem is on maintaining a tree of student records. Complete the methods given to you. The method signatures should explain to you what to implement. Please try yourself before you Google for solutions.**

Alert. ***Absolutely NO collaboration and/or copying from the internet during the exam will be allowed. This is a closed notes exam. Only cheat sheets are allowed. LANSchool will be used to monitor the desktops in the room.***

2 MCQs, 10 questions, 50 points, 30 minutes, 10:20 AM - 10:50 AM; cheats sheets are allowed; bonus problem(s) worth 10 points

- ☞ You may start the MCQ exam immediately after the coding exam submission; no need to wait till 10:20 AM. However, even if you start early, you still get 30 minutes to solve the MCQ exam.
- ☞ Once you start the quiz, make sure that you do not tab out from the quiz. Canvas records your activity during the quiz.

Excluded slides. `1-Introduction`, `2-OOD`

2.1 Slides. `3-Analysis`

Understand how to derive time complexity using Big-Oh notation. You may be given a small piece of code and asked to derive its time complexity.

Practice problems. What are the time complexities of the following pieces of code snippets?

1.
```java
public static boolean doContainSameContent1(Integer[] arrayA, Integer[] arrayB) {
        if( arrayA.length != arrayB.length )
              return false;

        QuickSort.sort(arrayA);
        QuickSort.sort(arrayB);

        for(int pos = 0; pos < arrayA.length; pos++)
              if( !arrayA[pos].equals(arrayB[pos]) )
                    return false;

        return true;
}
```

Answer. $O(n^2)$

2.
```java
public static boolean doContainSameContent2(Integer[] arrayA, Integer[] arrayB) {
        if( arrayA.length != arrayB.length )
              return false;

        Arrays.sort(arrayA);
        QuickSort.sort(arrayB);

        for(int pos = 0; pos < arrayA.length; pos++)
              if( !arrayA[pos].equals(arrayB[pos]) )
                    return false;

        return true;
}
```

Answer. $O(n^2)$

3.
```
public static boolean doContainSameContent3(Integer[] arrayA, Integer[] arrayB) {
        if( arrayA.length != arrayB.length )
                return false;

        Arrays.sort(arrayA);
        Arrays.sort(arrayB);

        for(int pos = 0; pos < arrayA.length; pos++)
                if( !arrayA[pos].equals(arrayB[pos]) )
                        return false;

        return true;
}
```

Answer. $O(n \log n)$

4.
```
public static boolean doContainSameContent4(Integer[] arrayA, Integer[] arrayB) {
        if( arrayA.length != arrayB.length )
                return false;

        MergeSort.sort(arrayA);
        MergeSort.sort(arrayB);

        for(int pos = 0; pos < arrayA.length; pos++)
                if( !arrayA[pos].equals(arrayB[pos]) )
                        return false;

        return true;
}
```

Answer. $O(n \log n)$

5.
```
public static boolean doContainSameContent5(Integer[] arrayA, Integer[] arrayB) {
        if( arrayA.length != arrayB.length )
                return false;

        HeapSort.sort(arrayA);
        HeapSort.sort(arrayB);

        for(int pos = 0; pos < arrayA.length; pos++)
                if( !arrayA[pos].equals(arrayB[pos]) )
                        return false;

        return true;
}
```

Answer. $O(n \log n)$

6.
```
public static boolean doContainSameContent6(Integer[] arrayA, Integer[] arrayB) {
        if( arrayA.length != arrayB.length )
                return false;
        InsertionSort.sort(arrayA);
        InsertionSort.sort(arrayB);

        for(int pos = 0; pos < arrayA.length; pos++)
              if( !arrayA[pos].equals(arrayB[pos]) )
                    return false;

        return true;
}
```

Answer. $O(n^2)$

7.
```
public static TreeSetRBTree<Integer> union(Integer[] arrayA, Integer[] arrayB) {
        TreeSetRBTree<Integer> S = new TreeSetRBTree<>();

        for(var item : arrayA )
              S.add(item);

        for (var item : arrayB)
              S.add(item);

        return S;
}
```

Answer. $O(n \log n)$

8.
```
public static void reverse( Integer[] array ){
        LinkedStack<Integer> stack = new LinkedStack<>();

        for(var item : array)
              stack.push(item);

        int pos = 0;

        while( !stack.isEmpty() )
              array[pos++] = stack.pop();
}
```

Answer. $O(n)$

2.2 Slides. 4-ArraysandLLs

Understand the difference between an array of primitives and an array of objects. Understand how linked lists work and the time complexity of various operations mentioned on the slides. Given a small piece of linked-list code, you may need to figure out what the code is doing. Reviewing the methods taught in the class should be enough. Understand how arraylists work in Java.

Practice problem. What is the purpose and time complexity of the following code snippet when invoked on a singly linked-list?

```
public void doSomething(E e) {
        if( head == null ) {
                head = tail = new Node<>(e,null);
                return;
        }

        if( head == tail ) {
                head = new Node<>(e,tail);
                return;
        }

        Node<E> current = head, newNode = new Node<>(e,null);

        while( current.next != tail )
                current = current.next;

        current.next = newNode;
        newNode.next = tail;
}
```

2.3 Slides. 5-StacksandQueues

Understand how stacks, queues, and deques work. Basic understanding of the operations and their time complexities should be sufficient. Ignore the applications for the test. You may be given a sequence of operations and asked the final state of the stack/queue/deque.

Practice problem. Start with an empty integer deque and execute the following operations. How does the deque look like in the end?

`addFIrst(20); addFirst(30); addFirst(40); addlast(90); removeFIrst(); removeLast();`

2.4 Slides. 6-Recursion

Given a small piece of recursive code, you should be able to trace it and figure out the output/return value.

Practice problem. Predict the return value of the following recursive method when invoked with $x = 5, y = 6$.

```
public static int myFunc(int x, int y ) {
        if( x <= 0 && y <= 0 )
                return 2;
        else
                return 2 * myFunc(x-1, y-2);
}
```

2.5 Slides. 7-Trees

Understand the three traversal algorithms. Given a tree, you should be able to figure out its pre, in, and post-order traversal sequences. Practice using the trees on the slides and also from the tree HW to gain more experience. Tries are excluded from this test.

Practice problem. Figure out the preorder, inorder, and postorder traversals of the following binary tree.

$$\begin{tikzcd}
 & & & \textsc{Eric} \arrow[dll] \arrow[drr] & & & \\
 & \textsc{Alice} \arrow[dl] \arrow[dr] & & & & \textsc{Emma} \arrow[dl] \arrow[dr] & \\
\textsc{Pat} & & \textsc{Bob} \arrow[d] & & \textsc{John} \arrow[dl] \arrow[dr] & & \textsc{David} \\
 & & \textsc{Maya} & \textsc{Donald} & & \textsc{Sophia} &
\end{tikzcd}$$

2.6 Slides. `8-BSTs`

Understand the basic idea of binary search trees and the algorithms for searching, insertion, and deletion (3 cases). Use examples on the slides to understand the three operations. Insert a few items and delete a few to check your understanding. You may be asked questions on insertion and deletion.

Understand how to insert a new item into a RB-Tree and understand how the colors changes after insertion. You may be asked questions about the structure of the tree and node colors after inserting a new item.

Practice problem. Insert the following integers in the given sequence into an empty plain binary search tree. Now delete 10 and then 5, and finally -10. Draw how the tree looks like in the end.

$$-10, 0, 2, 10, 5, 7, 99, 12, -55, 4, 6$$

Practice problem. Insert the following integers in the given sequence into an empty red-black tree and draw how the tree looks like in the end along with the colors of the nodes.

$$4, 7, 12, 15, 3, 5, 14$$

2.7 Slides. `9-Hashing`

Understand how separate chaining (including rehashing) and open addressing insertion works using examples. Make sure you can insert new items by hand for both the hashing techniques.

Practice problem. Insert the following records in the given sequence into an empty separate chaining hash map and show how the map looks like in the end. The values are omitted from the records to make things simpler. Assume that in the beginning the length of the array is 10 and the target load-factor is 0.75; rehash accordingly.

$$10, 0, 2, 11, 5, 7, 99, 12, 55, 4, 6$$

Practice problem. Insert the following integers in the given sequence into an empty open addressing hash map and show how the map looks like in the end. The values are omitted from the records to make things simpler. Assume that in the beginning the length of the array is 20.

$$10, 0, 2, 11, 5, 7, 99, 12, 55, 4, 6$$

2.8 Slides. `10-Sets`

Understand the set operations and the algorithms for executing the operations.

Practice problem. What is the purpose of the following code snippet?

```
public static TreeSetRBTree<Integer> doSomething(TreeSetRBTree<Integer> T1,
                                                   TreeSetRBTree<Integer> T2) {
        TreeSetRBTree<Integer> S = new TreeSetRBTree<>();

        for( var e : T1 )
                if( T2.contains(e) )
                        S.add(e);

        return S;
}
```

2.9 Slides. `11-PQsandHeaps`

Understand how to insert and delete from a heap and the time complexities. Understand how heaps are represented using arrays. Study the examples on the slides. Try to insert a few items into the tree and execute a few removeMin() operations. After every operation, draw the array representation of the heap by hand.

Practice problem. Insert the following integers in the given sequence into an empty min-heap and show how the heap looks like in the end.

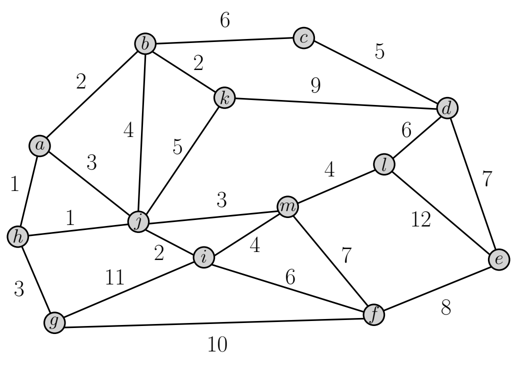
$$10, 0, 2, 11, 5, 7, 99, 12, 55, 4, 6$$

Now, execute removeMin() three times. Draw the final array representation of the min-heap.

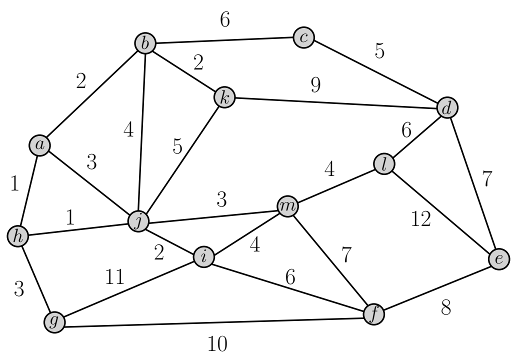
2.10 Slides. `12-Graphs`

Understand the uses of the algorithms: BFS, DFS, Prim's, Dijkstra. Given a graph, you should be able to execute the four graph algorithms by hand.

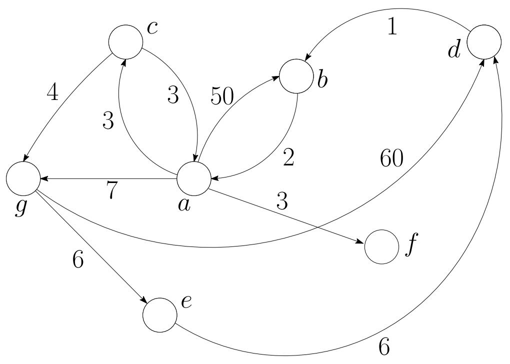
Practice problem. Consider the following graph and execute BFS and DFS on it. Start at vertex *m*. Whenever required, consider the neighbors in alphabetical order.



Practice problem. Consider the following graph and execute the Prim's algorithm on it. Start at vertex *j*. In case of ties, choose the vertex whose ID is alphabetically the smallest.



Practice problem. Consider the following graph and execute the Dijkstra's algorithm on it. Start at vertex a. In case of ties, choose the vertex whose ID is alphabetically the smallest.



Alert. ***Absolutely NO collaboration and/or copying from the internet during the exam will be allowed. This is a closed notes exam. Only cheat sheets are allowed. LANSchool will be used to monitor the desktops in the room.***