

OWLS Practice Final  
January 2022

This practice midterm has many questions for you to use to study for the exam. Note, this exam is way longer than the exam Professor Geitz is going to give, we have done that intentionally so that you have many questions to practice with.

Good Luck! You got this!

- Here is a list of data: **[4,2,18,1,3,7,9,0,6]** For each of the following structures I will walk through the list in order, add each item to the structure and then go into a loop in which I remove elements one at a time from the structure and print them as I remove them. **In what order do I print the items for**
  - A stack
  - A queue
  - A priority queue
  - Forming a graph: each value is nodes, and there are **edges from any value to any value except itself that it divides evenly into**. In the removal stage I do a topological sort of the graph.
  - Forming a hash table of size 9 with linear open addressing using each data value as its own hash code (so the hash value is the remainder when we divide the value by 9). In the remove stage I remove and print the data at index 0, then the data at index 1, then index 2, etc.
- Here is the start of a class that represents binary search trees that hold integer values:

```
class BST {
    Node root;
    class Node {
        int data;
        Node left, right;

        Node(int d) {
            data = d;
            left = null;
            right = null;
        }
    }
    BST( ) {
        root = null;
    }
}
```

As you can see, the BST class has a nested Node class; there is a constructor for Nodes. Give a BST method

```
boolean insert(int x)
```

If x is already in the tree your method should make no changes and return false. If x is not in the tree your method should insert it (preserving the Binary Search Tree property) and return true.

- I have an `ArrayList<String>` and I want to sort it by increasing string length. The `ArrayList` class has a method `sort (Comparator<String> comp)`. Give a class that implements the `Comparator<String>` interface that I can use to sort my `ArrayList`.

4. I have a priority queue with 11 entries stored as a heap in the following array:

0	1	2	3	4	5	6	7	8	9	10	11	12
	10	15	20	30	40	25	20	30	40	45	40	...

The digits on top are the array indices. I want to poll the queue to remove its smallest value. What is the sequence of entries in the array after this removal? Your answer should look like this: 15 20 20 25 30 30 40 40 40 45, though that isn't the correct answer.

5. I have  $n$  integer values stored in an ArrayList that I want to use as a priority queue. Instead of the heaps that we used in class, I am going to just sort the ArrayList and use that as the basis for my priority queue. For each of the following methods give an algorithm in English and a worst-case Big-Oh estimate of how long this takes.

- (a) **peek( )**: Return the smallest value in the sorted ArrayList without changing the list.
- (b) **poll( )**: Remove and return the smallest value in the sorted ArrayList
- (c) **offer(x)**: Add element  $x$  to the sorted array list.

6. I have a directed graph with  $N$  nodes. How can I tell if the graph has a cycle? Your answer should be an algorithm in English. It is nice but not essential for you to give the name of the algorithm. You should describe how the algorithm works.

7. Oh, no! You fell asleep during the CS 151 final and now you find yourself inside a video game. You are in a gigantic house with many rooms and doors connecting the rooms. The name of each room is painted on the ceiling so you know where you are; you just have to find your way out. Instead of a map you have a file that lists all of the doors of the house; a line in the file such as

Dining Room : Kitchen

means there is a door connecting the Dining Room and the Kitchen. You need to find your way to the room called Exit. To make matters worse, all of the doors in the house are guarded by trolls. To go through a door you have to distract its troll by giving it a cookie. You have a small bag of cookies and no way to get any more. If you can find your way out before you run out of cookies, you will wake up in the wonderful Land of Internships. If you don't find your way out, those trolls look very hungry

Give an algorithm, in English, for how to find your way out.

8. In each part give a Big-Oh estimate of the worst-case time it takes to find a specific entry in the given structure:

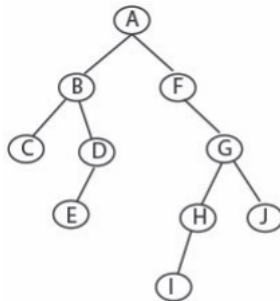
- (a) An unsorted Linked List with  $n$  entries
- (b) A sorted Array List with  $n$  entries
- (c) A sorted Linked List with  $n$  entries
- (d) A Binary Search tree with  $n$  entries
- (e) An AVL Tree with  $n$  entries
- (f) A Heap (Priority Queue) with  $n$  entries

9. We might have a binary tree structure with integer nodes defined as:

```
class Tree {  
    int data;  
    Tree left, right;  
}
```

Let's assume missing or empty trees are represented as null and a leaf is a tree that has null left and right children. Write a method **int height( )** for this Tree class that returns the height of the tree rooted at the current node. Leaves should have height 0, a tree with leaves for both children should have height 1, and so forth. Note that this method is slightly different from the code you wrote for Lab 5 because there is no EmptyTree class; you have to handle the base case of the recursion yourself.

10. Here is a picture of a binary tree with 10 nodes.



I am going to run an algorithm that uses a “structure” that might be a stack or a queue. I start by putting the root node A into the structure. Then at each step I remove the next node from the structure, print its letter, add its left child into the structure and then add its right child into the structure. This continues until the structure is empty.

- (a) In what order are the nodes printed if the structure is a stack? An answer might look like “AGHBCDFEIJ” (though that’s not the right answer).
- (b) In what order are the nodes printed is the structure is a queue?
11. I want to build a Hashmap in an array of size 10 (so the array indices are 0 through 9). I will use linear open addressing to resolve collisions. Here is a table of 9 entries and their hash values:

Entry	A	B	C	D	E	F	G	H	I
Hash Value	0	8	9	6	1	3	7	5	6

I add the data in the order: D B C I G A E F H

List the entries in the order they appear in the array: index 0 first, then index 1, and so forth

12. We might have a Node class defined as

```
class Node {  
    private int data;  
    private Node next;  
}
```

and a List class with an empty sentinel node Head at the front and a variable Tail pointing to the last node in the list.

Write a method List reverse( List L ) that will build the reversal of List L. So if L starts as the list shown, reverse( L ) will produce the list 30, 20, 10. It is acceptable to destroy L in the process of making its reversal (for example, you might want to make the Head node point to the node containing 30, and that node point to the one containing 20, and so forth).

The following questions are not from Professor Geitz and are written by us or the previous OWLs!!

1. Below is a fill in the gap:

- (a) The java syntax "implements" is used for \_\_\_\_\_ while "extends" is used for \_\_\_\_\_
- (b) One main difference between an array and an ArrayList is \_\_\_\_\_
- (c) \_\_\_\_\_ algorithm can be used to find the shortest path in a graph with negative edge weights.
- (d) We perform \_\_\_\_\_ when we add an element to a heap and \_\_\_\_\_ when we remove an element from a heap.
- (e) The \_\_\_\_\_ operation is used to remove an element from a stack
- (f) The element removed from a queue is the \_\_\_\_\_ element added to the queue.
- (g) Dijkstra's algorithm produces a \_\_\_\_\_
- (h) If you're looking for the next empty location in an array and you search in a squared number sequence, then you are using a hashing technique known as \_\_\_\_\_
- (i) An IF statement with 3 nested for-loops in one of its conditions has a big-oh runtime \_\_\_\_\_
- (j) Once the load factor exceeds its predefined value, the hash map must be \_\_\_\_\_

2. Put the following Big-O runtimes in order (Indicate if two are equivalent):  $O((1/5)n)$ ,  $O(\log n + n^2)$ ,  $O(\log(n))$ ,  $O(2^n)$ ,  $O(n^6)$ ,  $O(n \log n)$ ,  $O(x^n)$

3. Write the Big-O runtime of the following functions:

- (a) 

```
public int size(){
    return size;
}
```
- (b) 

```
public int fact(int n){
    int x = 1;
    for (int i = 1; i <= n; i++){
        x = x*i;
        if (n>1){
            n--;
        }
    }
    return x;
}
```

```

(c) public int ILoveOWLS(int n){
    int result = 0;
    while (n > 1){
        n = n/2;
        result += 1;
    }
    return result;
}

(d) public int IWillMissThisClass(int n){
    int result = 0;
    for (int i = 0; i < n; i++){
        for (int j = 0; j < n-1; j++){
            for (int k = 3; k < n-5; k++){
                result ++;
            }
        }
    }
}

```

4. Give the expected Big-O runtime for the following operations using the given data structures. Ignore the cost of doubling the array, or consider it amortized throughout all the add calls

Operation	ArrayList	Singly Linked List (with head pointer and no tail pointer)
add(0,x)		
add(x)		
get(i)		
remove(0)		

5. Stack vs Queue Assume that you are implementing a class that can use either a Stack or a Queue to manage a worklist of items that need to be processed. If the following method calls occur in the order given below, complete the table showing: 1) The changes being made to the worklist. 2) What the method call will return. If the method does not return anything, write "null".

**Stack:**

Method	Worklist	Return value
push("a")		
push("b")		
peek()		
push("c")		
pop()		
pop()		

**Queue:**

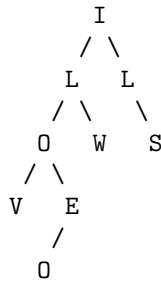
Method	Worklist	Return value
enqueue("a")		
enqueue("b")		
head()		
enqueue("c")		
dequeue()		
dequeue()		

6. Draw the binary search tree that will result from inserting the following numbers in the following order:  
7, 25, 12, 78, 35, 3, 1, 4, 15, 9, 265

7. Heaps

- Describe the structure of a min-heap, how do each nodes compare to their children?
- Fill in the runtimes of the min-heap operations
  - GetMin()
  - RemoveMin()
  - Insert()
- Fill in the array index locations of each node (When relevant, relative to node
  - The root
  - Node i's Left Child
  - Node i's Right Child
  - Node i's Parent

8. Show and label the results of inorder, postorder, level order, and preorder traversals of the tree below.  
Label which traversals are depth-first and which are breadth-first.



9. Why, in a binary search tree with  $n$  nodes, are there exactly  $n-1$  different possible opportunities for single rotations? (Think about the structure of a rotation, what's needed?)
10. Give recursive algorithms that perform preorder and postorder tree traversals in  $O(n)$  time on a tree of  $n$  nodes.
11. ArrayList Implementation
- Implement the resize function to create a new array that is 3 times the length of the old one, copies all the data into the new one, changes this.data in the relevant way.
  - Implement a new function, shuffle that shuffles the array by swapping the first and last element, the  $n-2$  and 1st element, and so on. Include the runtime of your function.



12. Give the Big-Oh for the following:

- (a) Bubble Sort:
- (b) Selection Sort:
- (c) Heap Sort:

13. Sort the following list using merge sort [15, 6, 4, 22, 3, 11, 2, 9]. Show all of your steps.