CS 32 Worksheet 10

This worksheet is entirely **optional**, and meant for extra practice. Some problems will be more challenging than others and are designed to have you apply your knowledge beyond the examples presented in lecture, discussion or projects. All exams will be done on paper, so it is in your best interest to practice these problems by hand and not rely on a compiler.

If you have any questions or concerns please go to any of the LA office hours.

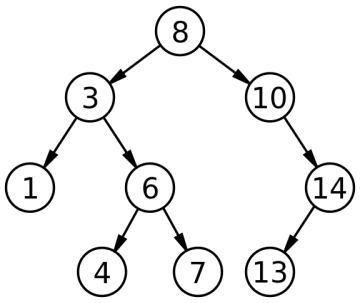
Concepts

Stacks, Queues, Recursion, STL, Algorithm Analysis, Sorting, Trees

1. You are given the following STL data structure:
   hash_map<student, set<class>> studentClass

   There are S students and an average of C classes per student.

   a. What is the big-O complexity of printing all students alphabetically and for each student, listing each of their classes alphabetically?
   b. What is the big-O complexity of determining who's taking CS32?
   c. What is the big-O complexity of determining if Joe Smith is taking CS32?
   d. Suppose we have the data structure hash_map<string, set<int>> and we wish for a particular class L to print the ID numbers of all the students in that course in sorted order (assume there are S students in a class). What is the big-O complexity?

2. 

   a. What are the preorder and inorder traversal for the tree?
   b. Using the simplest binary search tree (BST) insertion algorithm (no balancing), what is the resulting tree after inserting the nodes 80, 65, 71, 15, 39

and 25, then deleting 10, 6 and 39 in that order.
c. What is the postorder traversal for the resulting tree in part 2?
d. Given the following postorder and inorder traversal of a binary tree (not necessarily a BST), draw the corresponding tree:

INORDER: 1 5 0 8 4 3 7 6 2 POSTORDER: 1 8 4 0 5 2 6 7 3

3. Given the following code:

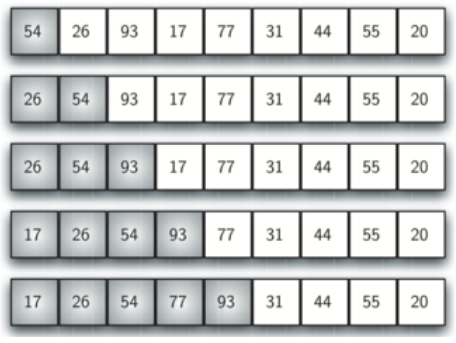
```
void func(int arr[], int n){
   set<int> s;
   for (int i=0;i<n;i++){
       for (int j=0;j<i;j++)
            s.insert(arr[i] * arr[j]);
       }
}
```

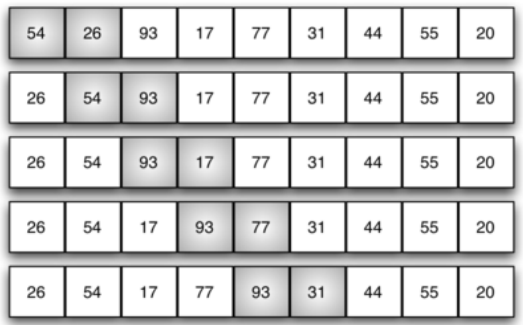
a. What is the time complexity of the code?
b. If we implement `s` with an `unordered_map` instead, what is the time complexity of the code?
c. What if we just implement it using an array that's sorted? What is its time complexity?

4.

a. Given the following array of integers `{-4, 10, 8, 2, -7, 3, 1, 0}`. Show the resulting array after three iterations of i) insertion sort, ii) bubble sort, and iii) quicksort with the first element as the pivot
b. What sorting algorithm is used for the iterations shown below?

i) ii)





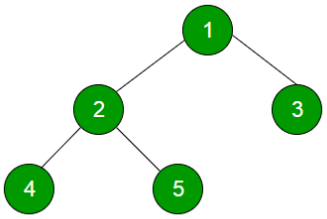
c. Now given the following array of integers {-2, -1, 0, 3, 10, 5, 7, 9}. Which sorting

algorithm is most efficient for you to use? Insertion sort, merge sort or heapsort?

5. a. Given a binary tree, write a code that returns all root-to-leaf paths. Use the function header:

   vector<string> rootToLeaf(Node* head)

   Example:



   All root-to-leaf paths are: ["1->2->4", "1->2->5", “1->3”]

   Note: Use this Node definition

```
struct Node {
        int val;
        Node* left, right;
};
```

   b. What is the time complexity of this function for each recursive call?

6. Evaluate the following infix to postfix expression. Show how the stack looks after each operation. (^ means exponentiation, with higher precedence than * and /).

   A / (B * C) - (D / E ^ F) * G

7. Consider the following code:

```
void mystery(queue<int>& q) {
   if (q.empty()) {
        return;
    }

   int data = q.front();
   q.pop();
```

```
    mystery(q);
    q.push(data);
}
```

Given the queue containing [3, 7, 5, 9, 2, 0, 8]

a. What is the queue being passed in on the first and last recursive call?

b. What does this function do?

8. Evaluate the following *prefix* expression using integer arithmetic. The answer is a single integer:

   + / * 2 - 8 2 4 * + 1 0 - 6 2

9. At UCLA, each department has its own set of classes. There are D departments and an average of C classes in a department. Given the following data structures:

   unordered_map<department, set<class>> SC1
   set<pair<department, set<class>> SC2

   a. Which one would be better (more efficient) if we want to print out all the classes in each department, with the departments and classes printed in alphabetical order? What is the big-O of the *less efficient* data structure for doing this?
   b. What is the big-O of determining what department a specific class belongs to if we use the SC1 data structure?
   c. What is the *worst* data structure to use if we want to print out all the classes in a specific department? What is the big-O?
   d. What is the big-O for each data structure if we want to see if a certain class is in a specific department?
   e. Say we have another data structure that stores the student IDs of all the students enrolled in a class (with total of C classes, an average of S students in each class, and each student is enrolled in N classes) defined as unordered_map<class, list<studentIDs>>, what is the big-O if:
      i. We want to print out all the student IDs enrolled in a specific class in an increasing order?
      ii. We want to print out all the classes a specific student is enrolled in alphabetically?
      iii. We want to find if a certain student is enrolled in a certain class?

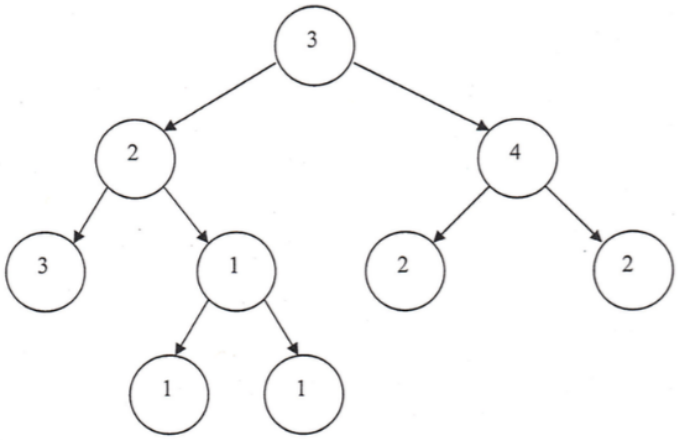
10. Let's say a tree is in balance if the tree is empty, or the *weight* (defined to be the total of the vals in the nodes of the tree) of the left and right subtrees are equal, and each of those subtrees is itself in balance. Write a function called isInBalance that takes a pointer to the root node of a tree and returns true if the tree is balanced, or false otherwise. The node is defined as:

```
struct Node{
        int val;
        Node* left, right;
};
```

You may write additional functions, but you must not use any global variables, not any variables other than bool, bool&, int, int&, or Node*. You must not use the keywords while, for, goto or static. Your solution shouldn't be more than 30 lines of code. Use the function header: bool isInBalance(Node* root)

Hint: Your solution must be such that a call to isInBalance results in each node in each tree being visited no more than once.

For example, this tree is in balance:



11. You are given an STL set< list<int>*>. In other words, you have a set of pointers, and each pointer points to a list of ints. Write a function that removes the lists with odd sums from the set. The lists with odd sums should be deleted from memory and their pointers should be removed from the set. This function should also return the number of lists that are removed from the set. If a list is empty, treat its sum as zero. You may assume that none of the pointers is null.

```
int deleteOddSumLists(set<list<int>*>& s) {
      // Fill in code here
}
```