CS 261	**Latex Homework5**
Fall 2022	**Due:** Week 5

My Introduction Write your biography of 150 words at least here

---

Problems

problem 1-1:

0.0.1 1-1 Comparison of running times

For each function f(n) and time t in the following table, determine the largest size n of a problem that can be solved in time t, assuming that the algorithm to solve the problem takes f(n) microseconds.

1 Second	1 Minute	1 Hour	1 Day	1 Month	1 Year
lg n					
Root n					
n					
n lg n					
n^2					
n^3					
2^n					
n!					

0.1 Problem 4-6:

Professor Diogenes has n supposedly identical integrated-circuit chips that in principle are capable of testing each other. The professor's test jig accommodates two chips at a time. When the jig is loaded, each chip tests the other and reports whether it is good or bad. A good chip always reports accurately whether the other chip is good or bad, but the professor cannot trust the answer of a bad chip. Thus, the four possible outcomes of a test are as follows:

Chip A says	Chip B says	Conclusion
B is good	A is good	both are good, or both are bad
B is good	A is bad	at least one is bad
B is bad	A is good	at least one is bad
B is bad	A is bad	at least one is bad

a. Show that if at least n=2 chips are bad, the professor cannot necessarily determine which chips are good using any strategy based on this kind of pairwise test. Assume that the bad chips can conspire to fool the professor.

Now you will design an algorithm to identify which chips are good and which are bad, assuming that more than n=2 of the chips are good. First, you will determine how to identify one good chip.

b. Show that bn=2c pairwise tests are sufûcient to reduce the problem to one of nearly half the size. That is, show how to use bn=2c pairwise tests to obtain a set with at most dn=2e chips that still has the property that more than half of the chips are good.

c. Show how to apply the solution to part (b) recursively to identify one good chip. Give and solve the recurrence that describes the number of tests needed to identify one good chip.

You have now determined how to identify one good chip.

d. Show how to identify all the good chips with an additional ,.n/ pairwise tests

0.2 Problem 13-1:

0.2.1 13-1 Persistent dynamic sets

During the course of an algorithm, you sometimes ûnd that you need to maintain past versions of a dynamic set as it is updated. We call such a set persistent. One way to implement a persistent set is to copy the entire set whenever it is modiûed, but this approach can slow down a program and also consume a lot of space. Sometimes, you can do much better. Consider a persistent set S with the operations I NSERT, DELETE, and SEARCH, which you implement using binary search trees as shown in Figure 13.8(a). Maintain a separate root for every version of the set. In order to insert the key 5 into the set, create a new node with key 5. This node becomes the left child of a new node with key 7, since you cannot modify the existing node with key 7. Similarly, the new node with key 7 becomes the left child of a new node with key 8 whose right child is the existing node with key 10. The new node with key 8 becomes, in turn, the right child of a new root r 0 with key 4 whose left child is the existing node with key 3. Thus, you copy only part of the tree and share some of the nodes with the original tree, as shown in Figure 13.8(b). Assume that each tree node has the attributes key, left , and right but no parent. (See also Exercise 13.3-6 on page 346.)

a. . For a persistent binary search tree (not a red-black tree, just a binary search tree), identify the nodes that need to change to insert or delete a node.

b. Write a procedure PERSISTENT-TREE-INSERT .T; ´/ that, given a persistent binary search tree T and a node ´ to insert, returns a new persistent tree T 0 that is the result of

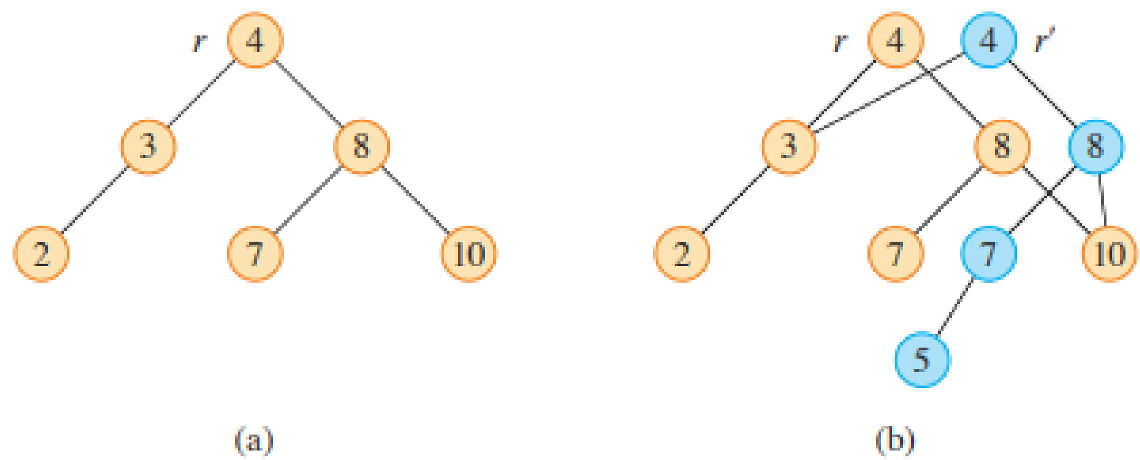

Figure 1: (a) A binary search tree with keys 2; 3; 4; 7; 8; 10. (b) The persistent binary search tree that results from the insertion of key 5. The most recent version of the set consists of the nodes reachable from the root r 0 , and the previous version consists of the nodes reachable from r. Blue nodes are added when key 5 is inserted.

inserting ´ into T . Assume that you have a procedure COPY-NODE.x/ that makes a copy of node x, including all of its attributes.

c. c. If the height of the persistent binary search tree T is h, what are the time and space requirements of your implementation of PERSISTENT-TREE-INSERT? (The space requirement is proportional to the number of nodes that are copied.)

d. Suppose that you include the parent attribute in each node. In this case, the PERSISTENT-TREE-INSERT procedure needs to perform additional copying. Prove that PERSISTENT-TREE-INSERT then requires n time and space, where n is the number of nodes in the tree.

e. Suppose that you include the parent attribute in each node. In this case, the PERSISTENT-TREE-INSERT procedure needs to perform additional copying. Prove that PERSISTENT-TREE-INSERT then requires n time and space, where n is the number of nodes in the tree.