CST 370 Design and Analysis of Algorithms
Spring A 2020
Midterm-II

Name: _____Ivan Alejandre_____

Four-digits Class ID: _____8889_____

- Test time is **2 hours and 30 minutes**.
- There are **12 problems**
- This is a **closed book exam**. You **can't use a calculator** during the exam. However, as a reference during the exam, you can prepare "two pages (= total of 4 sides)" **cheat sheet**. The cheat sheet can be typed or hand-written.
- If possible, enter your answers directly into the Word file to increase the readability of your answers. However, if it is too difficult or time consuming to type in a Word file, you can write down your answer on a paper. Then, take a picture and insert the picture into the Word file.
- During the exam, you must sign into the **Zoom** session and **turn on the video**. We will record the video. However, **turn off the audio** on your computer.
- If you have **a question** during the exam, please **use "Chat"** in Zoom. I will answer.
- When you finish the exam, submit your file in PDF format (optional Word file) on the iLearn.
- Use your time wisely—make sure to answer the questions you know first.
- Read the questions carefully.

1. (5 points) Suppose we are sorting an array of eight integers using quicksort, and we have just finished the first partitioning with the array looking like this:

2, 5, 1, 7, 9, 12, 11, 10

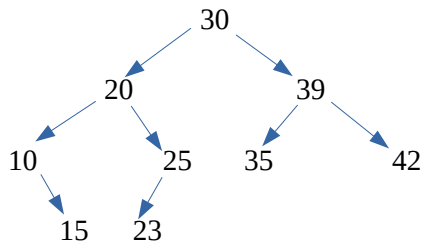
Which statement is correct? Why?

(A) The pivot could be either the 7 or the 9.
(B) The pivot could be the 7, but it is not the 9
(C) The pivot is not the 7, but it could be the 9
(D) Neither the 7 nor the 9 is the pivot.

7: all values left are less and all values right are greater
9: same reasoning applies, left are less and right are greater.

2. (5 points) Draw a binary search tree with 9 nodes labeled 10, 15, 20, 23, 25, 30, 35, 39, 42 in such a way that the **preorder traversals** of the tree yield the following lists:

30, 20, 10, 15, 25, 23, 39, 35, 42



If you can't completely draw a binary search tree with the given information, explain why.

3. (10 points) Consider the following master theorem as we covered in the class

$T(n) = aT(n/b) + f(n)$ where $f(n) \in \Theta(n^d)$, $d \geq 0$

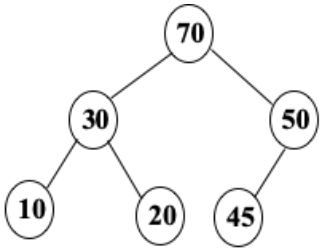
Master Theorem: If $a < b^d$, $T(n) \in \Theta(n^d)$
If $a = b^d$, $T(n) \in \Theta(n^d \log n)$
If $a > b^d$, $T(n) \in \Theta(n^{\log_b a})$

Based on the above theorem, determine the time efficiency of the following formula T(n).

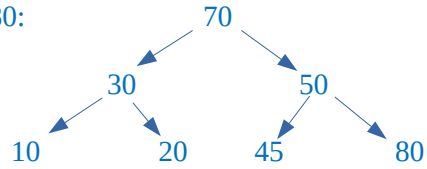
(a) $T(n) = 8 * T(n/2) + 2n^2 + 2n + 1$
$a=8,\ b=2, f(n)=2n^2+2n+1 \rightarrow f(n) \in \Theta(n^2)$, d=2
a ?= b^2 → no, 8 != 4 → 8 > 4
so T(n) = $\Theta(n^{\log_b a})$

(b) $T(n) = 3 * T(n/2) + 6n$
a=3, b=2, f(n)=6n, f(n) $\in \Theta(n)$, d=1
3 ? 2^1 → 3>2
so T(n) = $\Theta(n^{\log_b a})$

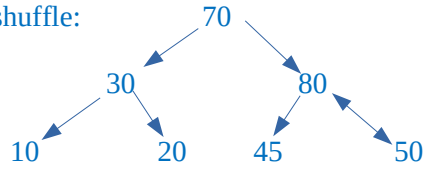
4. (5 points) Add 80 to the following heap. After adding the node, show intermediate diagrams (if any) that make it a heap again



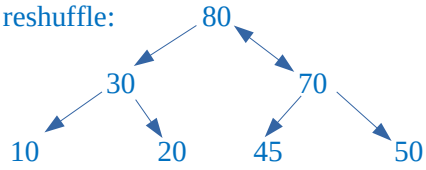
insert 80:



first reshuffle:

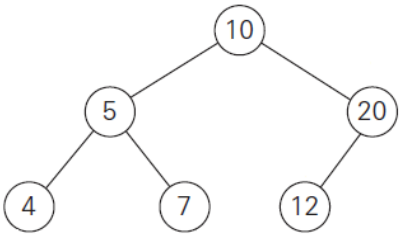


second reshuffle:

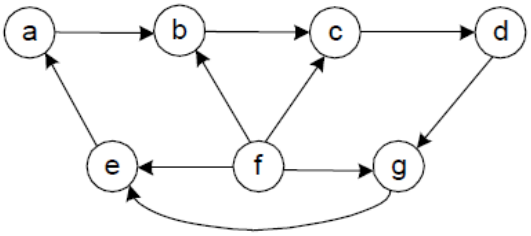


5. (15 points)

(a) Is this an AVL tree? (**Yes**/ No)

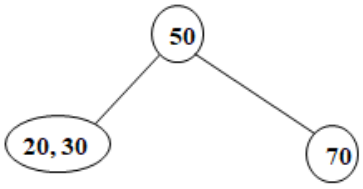


(b) Is the following graph a DAG (= directed acyclic graph)? (Yes/**No**)

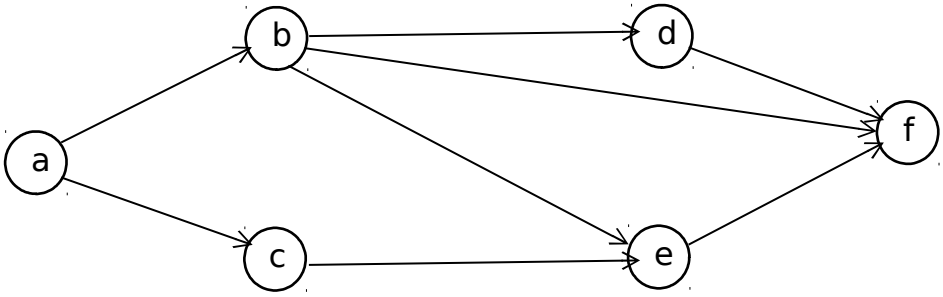


cycle present: E → A → B → C → D → G → E

(c) Is this a 2-3 tree? (**Yes**/ No)



6. (10 points) Consider the following **directed graph**.

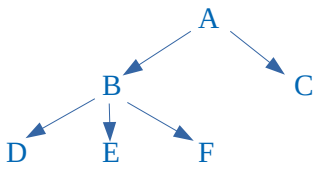


Starting at vertex ***a***, traverse the graph using the **breadth-first search** algorithm. You should present the **mark array** and **BFS tree** with only tree edges as we covered in the class. For the algorithm, you have to use our convention of the class (= ascending order for the alphabetical characters).

Mark array:

a	b	c	d	e	f
1	2	3	4	5	6

BFS edge tree:



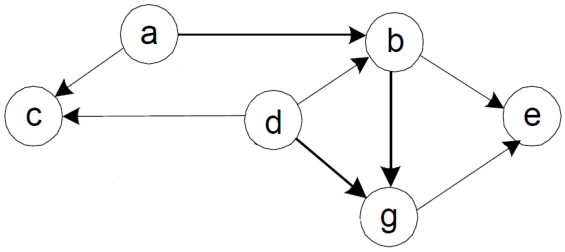
7. (5 points) Let T be a complete binary tree with n nodes. Finding a path from the root of T to a given vertex v in T using breadth-first search takes O(log n) time. Is this true or not? Explain your answer.

Since the tree is a complete binary tree, when we search for a path to the given vertex v, we halve the amount of searches we need to do each time we descend a node. This search is recursive in nature. Thus, we can use the master theorem and set up T(n) as

$$T(n) = 2T(n/2) + n$$

applying the theorem, we find a=2, b=2, d=1. According to the theorem, when $a=b^d$ ($2=2^1$), then the theorem states that time complexity is upper and lower bounded **$\Theta(\log n)$**

8. (10 points) For the following graph, present the source vertices (or starting vertices) when you use the **source-removal algorithm** to get the topological order. And then, present the result (= **topological order**) of the algorithm. For the problem, you have to follow our convention of alphabetical order removing.



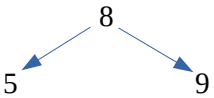
Sources: A, D
Topological order: A → D → B → C → G → E

9. (5 points) Construct a 2-3 tree with the list of **9**, **5**, **8, 3, 2, 4**, and **7**.

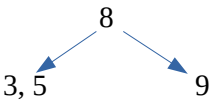
insert 9: 9

insert 5: 5, 9

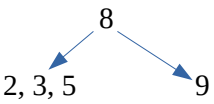
insert 8: 5, 8, 9 // promote 8



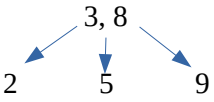
insert 3:



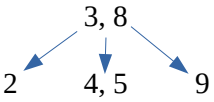
insert 2:



// promote 3



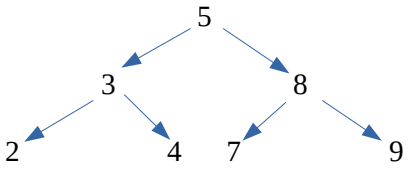
insert 4:



insert 7:

// promote 5

// promote 5

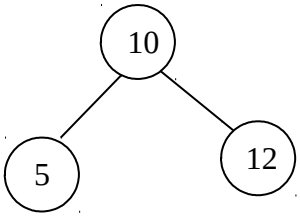


10. (10 points) Assume that you have an array A[0.. n–1] which contains n integers from 0 to n in the increasing order. Because the size of the array is n, one integer in the range from 0 to n is missing. For the problem, you have to design an efficient algorithm to find the missing integer. For example, let's assume that the array A has eight elements such as 0, 1, 2, 3, 4, 5, 7, and 8. For the array A, your algorithm should return 6 as the missing integer number. Assume that your algorithm receives the array A and the array size n as input arguments. Describe your algorithm in English. Determine the complexity of your algorithm in Big O notation.

```
Algorithm findMissing(A[0..n-1], n)
int i ← 0
For i to n-2 do
        if A[i+1] != A[i] + 1
                return A[i] + 1
```

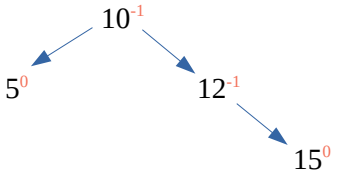
In English, the algorithm is to compare the value of the current index to the following value. If the current value plus 1 is equal to the following value, then we continue to the next index. However, if the comparison fails, then it'll return the value of the current index plus 1 which would be the missing value. Our time complexity for this algorithm would be **O(n)** since we iterate through the whole array once (which would be the **worst** case)

11. (15 points) Consider an AVL tree as below.

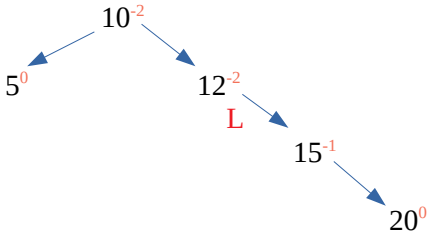


For all answers, superscript represents balance factor

(a) Add a vertex with the value **15** to the above AVL tree. If there is a rotation, you should present it clearly.

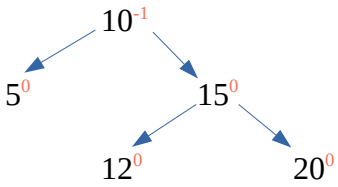


(b) To your solution of the question (a), add a vertex with the value **20**. If there is a rotation, you should present it clearly.

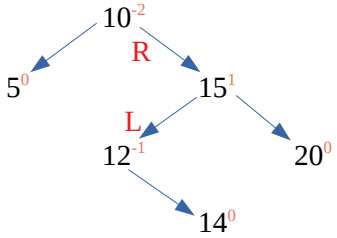


// rotate L on pivot node "12"

after rotation:

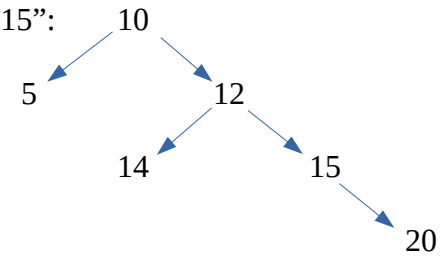


(c) To your solution of the above question (b), add a vertex with the value **14**. If there is a rotation, you should present it clearly.

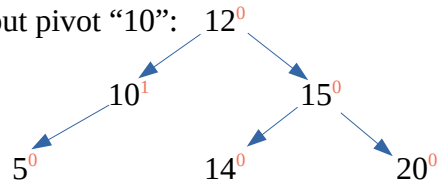


// rotate RL about pivot "10"

first rotation R about pivot “15”:



second rotation L about pivot “10”:



12. (5 points) Note that this is a **puzzle** problem. Assume that you have **100** identical-looking coins and a two-pan balance scale with no weights. One of the coins is fake, but it is not known whether it is lighter or heavier than the genuine 99 coins. Describe your idea to determine in the minimum number of weighings whether the fake coin is lighter or heavier than the others. Present the minimum number of weighings and your answer clearly.

Not that you don’t need to find out which coin is fake. The question is to identify whether the fake coin is heavier or lighter than the real coins.

1) Weigh half the coins with the other half. This should reveal one pile is heavier than the other.

2) Take the lighter stack, split 25/25, and weigh again. If they are equal, then the other original stack of 50 coins contains the **heavier** coin. This is the minimum(lucky) amount of weighings.