CSE 373 18au: Midterm (key)

Name:

UW email address:

Instructions

- **Do not start the exam until told to do so**.
- You have 50 minutes to complete the exam.
- This exam is closed book and closed notes.
- You may not use a calculator, cell phone, or any other electronic devices.
- Write your answers neatly in the provided space. Be sure to leave some room in the margins: we will be scanning your answers.
- If you need extra space, use the back of the page.
- If you have a question, ~~start singing a song~~ raise your hand.

Question	Points	Score
Binary Trees	12	
Recurrence	12	
AVL Trees	15	
Hash	15	
Heap	10	
Big-Oh	12	
Asymptotic Analysis	12	
Design choices	12	
Total:	100	

Advice

- Read questions carefully. Understand a question before you start writing.
- Write down thoughts and intermediate steps so you can get partial credit.
- The questions are not necessarily in order of difficulty. **Skip around**
- Relax. You are here to learn.

1. (12 points) Assume that the tree in Figure 1 is a binary search tree that contains only integer values and no duplicates. Note that nodes A and B are part of the tree and should be considered as such while answering all the following questions about the tree.

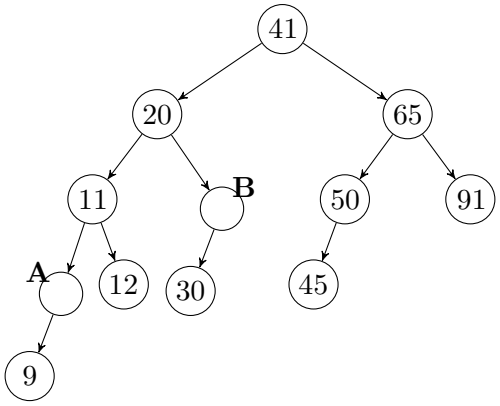

Figure 1: A binary tree.

(a) What could be the value of node A?

(a) **9**

(b) What could be the value of node B?

(b) **30 < B < 41**

(c) What is the height of the tree?

(c) **4**

(d) How many nodes can you insert in this tree without increasing its height?

(d) **19**

(e) What is the minimum number of nodes that need to be inserted to make this tree a complete tree?[14] Recall that a complete binary tree is a binary tree in which every level, except possibly the last, is completely filled, and all nodes are as far left as possible.

(e) **4**

(f) This tree is an AVL tree. Name one leaf node, which, if removed, will break the AVL balance property? (There are multiple answers to this. You just need to give one.)

(f) **12 (30, 45, or 91)**

(g) Write the in-order traversal sequence of the tree (include nodes A and B in your traversal)

9	**A**	**11**	**12**	**20**	**30**	**B**	**41**	**45**	**50**	**65**	**91**

2. Consider the following method that traverses a binary tree and prints the node values.

```
1 public void traverseTree(Node node) {
2     System.out.println('#');
3     if (node == null) {
4         return ;
5     }
6     System.out.println(node.data);
7     traverseTree(node.left);
8     traverseTree(node.right);
9 }
```

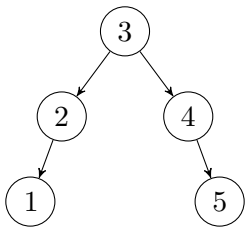

Figure 2: Binary search tree; test input for `traverseTree()`

(a) (2 points) What type of tree traversal does the method `traverseTree()` represent?

A. In-order
B. Post-order
C. Pre-order
D. Other

(b) (2 points) If the `traverseTree()` method is called on the root of the tree in Figure 2, how many times will the character '#' get printed?

(b) **11**

(c) (4 points) Give the recurrence $T(n)$ for the runtime of the following snippet of the code. Use variables appropriately for constants (e.g. c_1, c_2) in your recurrence (you do not need to attempt to count the exact number of operations). **YOU DO NOT NEED TO SOLVE** this recurrence, just give the base case and a recurrence case.

```
public int bar(int n) {
    if (n <= 10) {
        return 0;
    }
    int r = 0;
    for (int i = 0; i < n; i++) {
        r++;
    }
    return bar(n/3) + bar(n-1);
}
```

Solution:

$$T(n) = \begin{cases} c_1 & \text{if } n \leq 10 \\ T(n/3) + T(n-1) + c_2 n + c_3 & \text{otherwise} \end{cases}$$

(d) (4 points) Similar to how you solved the above question (c), give the recurrence $T(n)$ for the following snippet of the code

```
public int foo(int n) {
    if (n < 10) {
        return 0;
    } else if (n < 1000) {
        return foo(n - 1);
    }
    return foo(n - 1) + foo(n - 2);
}
```

Solution:

$$T(n) = \begin{cases} c_1 & \text{if } n < 10 \\ T(n-1) + c_2 & \text{if } 10 \leq n < 1000 \\ T(n-1) + T(n-2) + c_3 & \text{otherwise} \end{cases}$$

3. AVL Trees

(a) (5 points) Insert the following sequence of values into an empty AVL tree *in the given order*.

7, 2, -5, 8, 10, 9

Draw your final tree in the following figure (Figure 3). You may use the space below Figure 3 or the back of this page (or the previous page) to draw your intermediary trees.

You do not have to show the intermediary steps, but **a wrong answer and no intermediary steps will result in no credit**. We strongly encourage you to draw your intermediary trees, so we can give you partial credit if your answer is wrong.

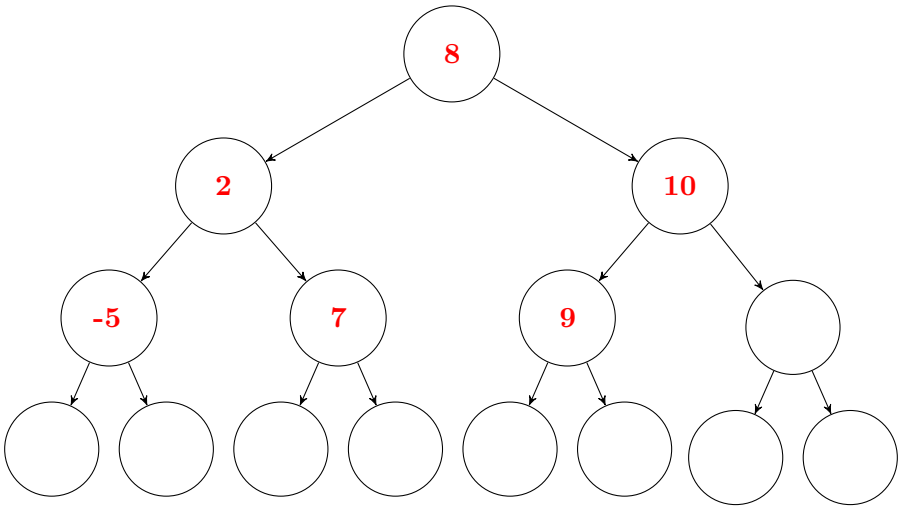

Figure 3: Fill this tree with your final answer. Leave unused nodes empty.

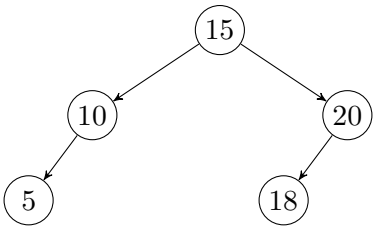

Figure 4: A five-node AVL tree.

(b) (4 points) Figure 4 shows a five-node AVL tree that was created when the following sequence of values were inserted in an empty AVL tree.

15	20	10	18	5

There are many other insertion sequences (orderings of these five numbers) that would result in the above AVL tree. Some sequences (like the one above) will cause no rotation; some sequences will cause only single rotation(s); and some sequences will cause double rotation(s).

Give an insertion sequence that will result in the AVL tree shown in Figure 4, but will cause at least one double rotation in the process.

20	**10**	**15**	**18**	**5**

Solution: There are multiple answers to this question; the above sequence is just one of those. For example, 10, 20, 15, 18, 5 is another sequence that would cause a double rotation. The key observation here is that to cause a double rotation we need to create a 'kink', and to do that we need to pick the third insertion value that is in between the first and second values. In the above sequence, the third value 15 is between 20 and 10 (the first and third values).

(c) (6 points) The tree in Figure 5 is an invalid AVL tree. The last inserted node (node 5, also labeled in the figure) breaks the AVL balance property of the tree.

i. In Figure 5, draw a rectangle around the lowest unbalanced node (i.e., an unbalanced node that will be rotated in an AVL rotation that will fix the AVL tree).

ii. Choose the case that represents the violation of the AVL balancing property. For reference, see the AVL rotation table and AVL rotation figure on your cheat sheet.

A. Case 1 (also called left line)

B. Case 2 (also called left kink)

C. Case 3 (also called right kink)

D. Case 4 (also called right line)

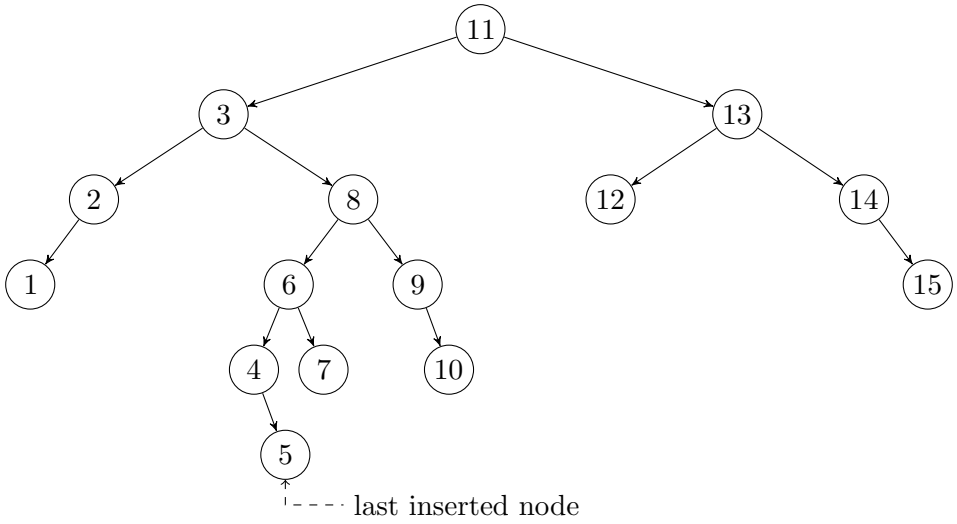

Figure 5: The last inserted node is node 5.

iii. Fix the invalid AVL tree in Figure 5 by doing the appropriate rotation(s). Draw your answer in Figure 6 (the following figure). Leave unused nodes empty. (*Hint: Check the AVL rotation table on the cheat sheet.*)

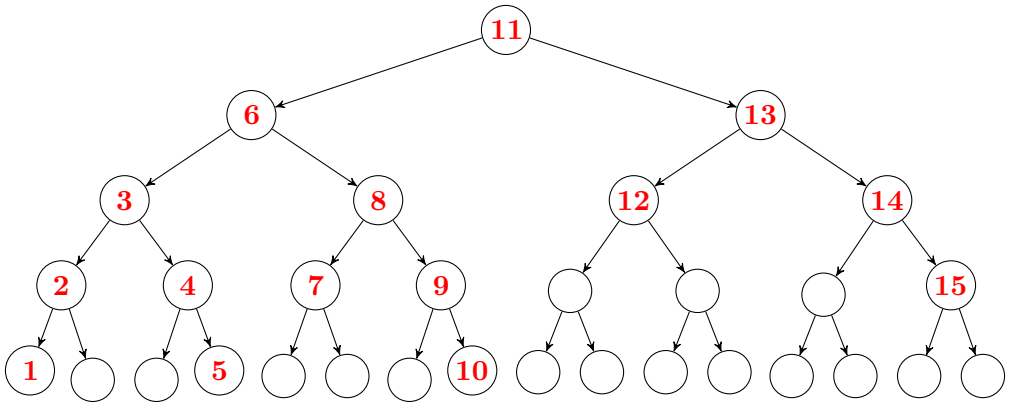

Figure 6: Fill this tree with your answer. Leave unused nodes empty.

4. (15 points) Consider inserting data with integer keys 4, 0, 1, 10, 13, 15, 30, 17 in that order into a hash table of size 15 where the hashing function is $h(key) = key$.

(a) Show an open addressing hash table that uses linear probing for resolving collisions after doing the above insertions. Do not worry about resizing.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	**1**	**15**	**30**	**4**	**17**					**10**			**13**	

(b) Show an open addressing hash table that uses quadratic probing for resolving collisions after doing the same insertions. Do not worry about resizing.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	**1**	**17**		**4**		**30**			**15**	**10**			**13**	

Consider inserting data with integer keys 4, 0, 1, 10, 13, 15, 30, 17 in that order into a hash table of size 15 where the hashing function is $h(key) = key$.

(c) Show an open addressing hash table that uses double hashing resolving collisions after doing the above insertions. The second hash function is $g(key) = key \ \% \ 9$. Do not worry about resizing.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	**1**	**17**	**30**	**4**		**15**				**10**			**13**	

5. (10 points) Insert the following sequence of values in the given order into the min-heap **shown in Figure 7**. (Do not insert these value into an empty min-heap.)

5, 10, 3, 2

Fill Figure 8 with your final answer. You may use the space below Figure 8 or the back of this page (or the previous page) to draw your intermediary trees.
You do not have to show the intermediary steps, but **a wrong answer and no intermediary steps will result in no credit**. We strongly encourage you to draw your intermediary trees, so we can give you partial credit if your answer is wrong.

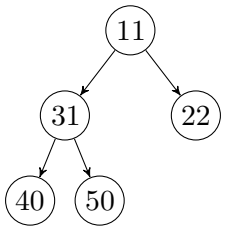

Figure 7: min-heap.

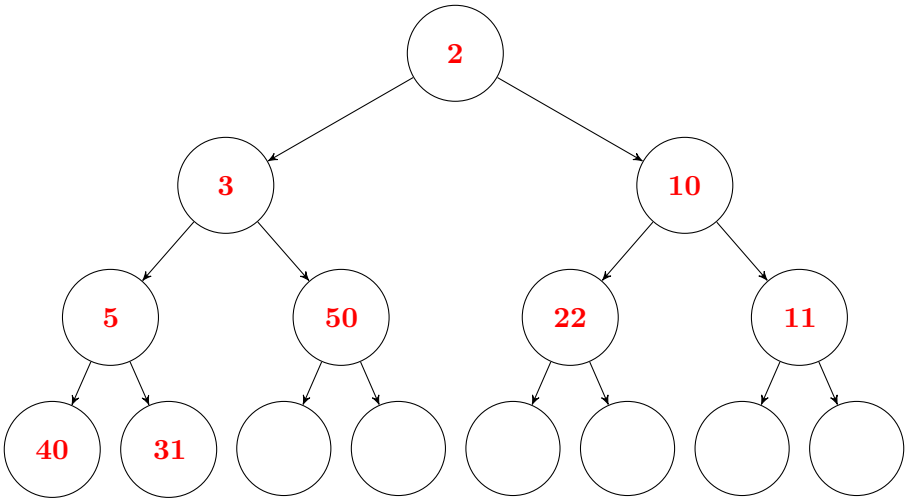

Figure 8: Fill this tree with your final answer. Leave unused nodes empty.

6. (12 points) For each of the following, describe the worst-case running time with respect to variable n. Please give a tight big-$\mathcal{O}$ bound and in the most simplified form. You do not need to justify your answer.

(a) Give a tight big-$\mathcal{O}$ bound in terms of variables n: $\mathcal{O}(n^2)$

```
public void foo(int n) {
    j = 100;
    for (int i = 0; i < n; i++) {
        j = bar(n);
        for (int k = 0; k < j; k++) {
            System.out.println("foo");
        }
    }
    return ;
}

public int bar(int n) {
    int i;
    for (i = 0; i < n; i++) {
        System.out.println("bar");
    }
    return i/2;
}
```

(b) Give a tight big-$\mathcal{O}$ bound with respect to variable n: $\mathcal{O}(n)$

```
public void zoo(int n) {
    int j = n - 1;
    while (j < n) {
        for (int i = 0; i < j; i++) {
            for (int k = 0; k < 1000; k++) {
                System.out.println("#");
            }
        }
        j++;
    }
    return ;
}
```

(c) Give a tight big-$\mathcal{O}$ for function $f(n)$. $f(n) = n^2(n + \log n)$: : $\mathcal{O}(n^3)$

(d) Give a tight big-$\mathcal{O}$ for function $f(n)$. $f(n) = 8n + n^3 + 2^n$: : $\mathcal{O}(2^n)$

7. **Asymptotic Analysis**

(a) (5 points) Demonstrate that $10n^2 + 100n + \log n$ is dominated by n^2 by finding a c and n_0. **For any credit, show your work**. Clearly state your c and n_0.

Solution:

By definition of "dominated by" and big-$\mathcal{O}$, we need to show $10n^2 + 100n + \log n \leq c \cdot n^2$ for all $n \geq n_0$ where c and n_0 are constant values.

Scratch work: Our goal is to show that $f(n) = 10n^2 + 100n + \log n$ is dominated by $g(n) = n^2$. We start by comparing terms.

$$10n^2 \leq 10n^2 \qquad \text{for all } n$$
$$100n \leq n^2 \qquad \text{if } n \geq 100$$
$$\log n \leq n^2 \qquad \text{if } n > 0$$

We add together the inequalities to get

$$10n^2 + 100n + \log n \leq 12n^2 \qquad \text{if } n \geq 100$$

So our $n_0 = 100$ and $c = 12$.

Thus, we know that $10n^2 + 100n + \log n \leq c \cdot n^2$ is true for all $n \geq n_0$, and for our chosen values of $c = 12$ and $n_0 = 100$.

(b) (7 points) Simplify the following summation to a closed form. Clearly state any summation identities that you use. You only need to simply to a point where there are no summations in your expression. You do not need to simplify any more than that.

$$8 + \sum_{i=3}^{n} i(i+1)$$

Solution:

$$8 + \sum_{i=3}^{n} i(i+1) = 8 + \sum_{i=3}^{n} (i^2 + i)$$

Adjusting summation bounds

$$= 8 + \sum_{i=0}^{n} (i^2 + i) - \sum_{i=0}^{2} (i^2 + i)$$

Splitting the sum

$$= 8 + \sum_{i=0}^{n} i^2 + \sum_{i=0}^{n} i - \sum_{i=0}^{2} i^2 - \sum_{i=0}^{2} i$$

Applying Gauss's identity

$$= 8 + \sum_{i=0}^{n} i^2 + \frac{n(n+1)}{2} - \sum_{i=0}^{2} i^2 - \frac{2 \cdot 3}{2}$$

Applying sum of squares

$$= 8 + \frac{n(n+1)(2n+1)}{6} + \frac{n(n+1)}{2} - \frac{2 \cdot (2+1) \cdot (4+1)}{6} - \frac{2 \cdot 3}{2}$$

We can stop here, but if we want to simplifying a little further we get

$$= 8 + \frac{n(n+1)(2n+1)}{6} + \frac{n(n+1)}{2} - 5 - 3$$

$$= \frac{n(n+1)(2n+1)}{6} + \frac{3n(n+1)}{6}$$

$$= \frac{2n^3 + 3n^2 + n}{6} + \frac{3n^2 + 3n}{6}$$

$$= \frac{2n^3 + 6n^2 + 4n}{6}$$

$$= \frac{n^3 + 3n^2 + 2n}{3}$$

8. (12 points) **Design choices:** For the following, choose the most appropriate data structure or ADT to solve the problem and briefly (in no more than 1-2 sentences) justify your answer.

   Stack, Queue, AVL tree, Binary search tree, Heap, Hash table, Linked List, Array

   If there are multiple equally good candidates, list them all.

   (a) Messaging service where guaranteed delivery and fairness (i.e., messages get delivered in the order they are sent) is more important than timeliness.

   Solution: We need a Queue here. Linked Lists are ideal to implement a FIFO queue. Arrays (with resizing) is also a valid answer.

   (b) A company has a large amount of data where data elements are *comparable*, but each data item is very large (in GBs). So comparing two data items takes a constant but long time. With the data, the company does mostly find operations and very few insert or remove operation.

   Solution: Hash table, because they are efficient at find operations. On average, in hash tables find is a constant time operation.

(c) An issue management system that receives hundreds of issues every hour needs to fetch the most critical issue to flag and process it.

Solution: Heap. The operation that we need to optimize here is removing an issue with the highest critical priority, while allowing efficient insert operations, and heaps are ideal for this.

(d) A job scheduler that needs to schedule jobs with the highest priority. In this system, the priority of a job is inversely proportional to how long that job has been in the queue. A job has the highest priority the moment it is submitted, but as it waits in the queue for the scheduler, the job's priority decreases.

Solution: Stack. Although the expected behavior may seem like a priority queue, based on how the priority of jobs change, the expected behavior actually matches that of a Stack. Among the data structures, both linked list or arrays (with resizing) are good choices to implement a Stack.