Midterm Exam

Autumn 2024

Name _____

Net ID _____ (@uw.edu)

Academic Integrity: You may not use any resources on this exam except for your one-page (front and back) reference sheet, writing instruments, your own brain, and the exam packet itself. This exam is otherwise closed notes, closed neighbor, closed electronic devices, etc.. The last two pages of this exam provide a list of potentially helpful identities as well as room for scratch work (respectively). Please detach those last two pages from the exam packet. No markings on these last two pages will be graded. Your answer for each question must fit in the answer box provided.

Instructions: Before you begin, **Put your name and UW Net ID at the top of this page.** Make sure that your name and ID are LEGIBLE. Please ensure that all of your answers appear within the boxed area provided.

Section	Max Points
ADTs and Data Structures	8
Asymptotic Analysis	13
Heaps	13
AVL Trees	13
Hashing	11
Algorithms	6
Extra Credit	(+2)
Total	64

Section 1: ADTs and Data Structures

(4 pts)**Question 1: ADT vs Data Structure**
For each of the following, indicate whether it is a Data Structure or an Abstract Data Type by writing DS or ADT (respectively) in the box provided.

1. Array List: [ ]
2. Priority Queue: [ ]
3. Queue: [ ]
4. Circular Array: [ ]

(4 pts)**Question 2: AVL vs. Hash Table**
AVL Trees and Hash Tables both implement the dictionary ADT. Describe one advantage than an AVL tree has over Hash Tables and one advantage that a Hash Table has over an AVL tree.

Section 2: Asymptotic Analysis

(4 pts)**Question 3: Asymptotic Analysis of Code**

Each method below adds all contents of the given array to a different data structure. Assume that all items in the array are distinct (no item appears multiples times). In all cases, the value being inserted is also either the priority (for priority queues) or the key (for dictionaries).

For each method, give a simplified Θ bound on its running time for the scenarios where the array is in ascending order (`arr[i]<arr[i+1]`) and the array is in descending order (`arr[i]>arr[i+1]`).

```
// add into an AVL tree
public void addAVL(int[] arr){
    AVLTree<Integer> avl = new AVLTree<>();
    for(int i = 0; i < arr.length; i++){
        avl.insert(arr[i], arr[i]);
    }
}
```

1. Ascending: $\Theta($ $)$
2. Descending: $\Theta($ $)$

```
// add into a Binary Min Heap
public void addHeap(int[] arr){
    MinHeap<Integer> heap = new MinHeap<>();
    for(int i = 0; i < arr.length; i++){
        heap.insert(arr[i], arr[i]);
    }
}
```

3. Ascending: $\Theta($ $)$
4. Descending: $\Theta($ $)$

```
// add into a Binary Max Heap
public void addHeap(int[] arr){
    MaxHeap<Integer> heap = new MaxHeap<>();
    for(int i = 0; i < arr.length; i++){
        heap.insert(arr[i], arr[i]);
    }
}
```

5. Ascending: $\Theta($ $)$
6. Descending: $\Theta($ $)$

```
// add into a Linear Probing Hash Table
// Give the expected running time assuming
// a good hash function and good rehashing.
public void addHash(int[] arr){
    HashMap<Integer> table = new ProbingTable<>();
    for(int i = 0; i < arr.length; i++){
        table.insert(arr[i], arr[i]);
    }
}
```

7. Ascending: $\Theta($ $)$
8. Descending: $\Theta($ $)$

(4 pts)**Question 4: Tree Method**

Suppose that the running time of an algorithm is expressed by the recurrence relation:

$$T(n) = 3 \cdot T\left(\frac{n}{3}\right) + n^3$$

$$T(1) = 1$$

For the following questions, use the tree method to solve the recurrence relation. We have broken up the process into subquestions to guide you through your answer. You may assume that n is always a power of 3.

1) Sketch the tree in space below. Include at least the first 3 levels of the tree (i.e. the root, its children, and its grandchildren), **make clear what the input size is for each recursive call as well as the work per call**.

2) Indicate exactly the total amount of work done at level i of the tree (define the root to be level 0). Include all constants and non-dominant terms.

3) Indicate the level of the tree in which the base cases occur.

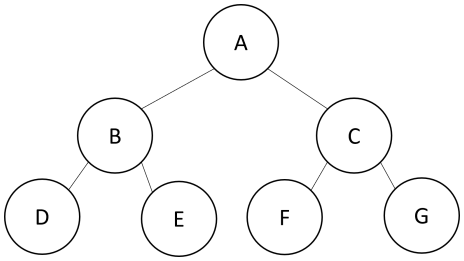
4) Give a simplified Θ bound on the solution. When simplified, n should not appear in any exponents.

$\Theta\left(\quad\right)$

(5 pts)**Question 5: Big O Proof**
Show that $5n^3 + \frac{1}{2}n^2 - 2n$ belongs to $O(n^3)$.

Section 3: Heaps

(4 pts)**Question 6: Always, Sometimes, Never**
Suppose that the tree below is a valid minheap, and its contents are integers represented by the variables A through G. For each statement below, indicate whether it is always true, sometimes true, or never true by writing "A", "S" or "N" in the box provided. (You may assume all of the heap's contents are unique).



1. $B < C$

2. $E < C$

3. A is the smallest value in the heap.

4. G is the largest value in the Heap

(4 pts)**Question 7: Inserting x**
Now suppose we insert a new integer x into the heap above and it becomes the left child of A. Answer each question regarding the resulting heap. You may assume x is different from all of the integers A through G.

1. Supposing we represented the heap using a 0-indexed array, at what index will we find G?

2. Supposing we represented the heap using a 0-indexed array, at what index will we find D?

3. List all items of the heap that are guaranteed to be smaller than x.

4. List all items of the heap that are guaranteed to be larger than x.

(5 pts)**Question 8: Binary Heap Math**

Answer each question below as it relates to a 0-indexed binary min heap containing 64 items. 0-indexed means the root of the tree is at index 0 in its array representation.

1. What is the height of the tree (recall that a one-node tree has height 0)?

2. How many items are on the last level of the tree?

3. If we call percolate up on index 62, which index will we compare to?

4. If we call percolate down on index 23, which two indices will we compare to?

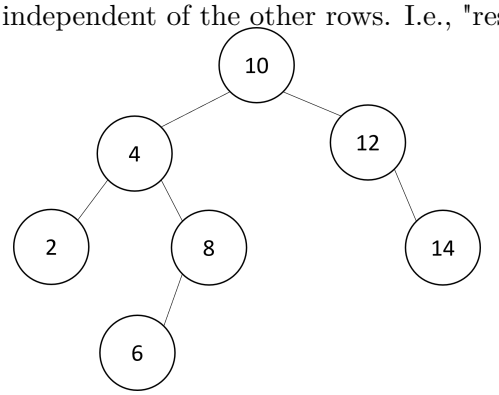
5. What is the smallest index which contains a leaf?

Section 4: AVL Trees

(10 pts)**Question 9: Rotations**

Use the following AVL Tree to complete the table. The first column should give an item not already in the tree. The second column should give the number of rotations that would be done when inserting that item. The third column should identify the type of rotation(s) performed (in order).

Each row's value will be inserted independent of the other rows. I.e., "reset" to this tree between rows.



New Value Inserted	Number of Rotations Done	Rotation type/types
	0	None
7		
	1	left rotation
9		
	2	right rotation then left rotation
5		

(3 pts)**Question 10: Structure and Order**

Using 2-3 Sentences, explain at a high level why the Order property of binary search trees and the structure property of AVL trees are *both* necessary to achieve a worst-case $O(\log n)$ running time for find.

Section 5: Hash Tables

(3 pts)**Question 11: Separate Chaining**
In a separate chaining hash table, any dictionary data structure could be used as a separate chain. In EX05 we used a linked list, but we could have used an AVL tree instead. Name one advantage and one disadvantage of using an AVL tree there.

(3 pts)**Question 12: Hash Table Deletion**
The next two questions relate to deletion performed on the open-addressing hash table below. This hash table was constructed by inserting (in order) the values 14, 24, 34, and 36 using the hash function $h(x) = x$ and linear probing.

				14	24	34	36		
0	1	2	3	4	5	6	7	8	9

Suppose we next performed the following two operations:

delete(24)

insert(54)

1. If the hash table used tombstone deletion, fill in the final hash table contents.

0	1	2	3	4	5	6	7	8	9

2. If the hash table did not use tombstone deletion, fill in the final hash table contents.

0	1	2	3	4	5	6	7	8	9

(5 pts)**Question 13: Quadratic Probing**
Insert 12, 13, 22, 23, 32, 33 (in that order) into the open addressing hash table below. You should use the primary hash function $h(k) = k\%10$. In the case of collisions, use quadratic probing for collision resolution. If an item cannot be inserted into the table, indicate this and continue inserting the remaining values. Do not resize the hash table.

Items that could not be inserted:

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

Section 6: Algorithms

(6 pts)**Question 14: Replace Min**

Suppose we want to implement a new operation called replaceMin. The result of replaceMin(x) will be to remove the smallest value from the data structure, then add x to the data structure. The operation does not have a return value.

Using 1-3 sentences, briefly describe an algorithm to perform replaceMin for each data structure given. Give the best and worst case running times for each algorithm. Use x as both the key and the value. You may assume the size of the data structure is at least 1.

1. Binary Search Tree:

   Worst Case Running Time: $\Theta\left(\quad\right)$.

   Best Case Running Time: $\Theta\left(\quad\right)$.

2. Linear Probing Hash Table:

   Worst Case Running Time: $\Theta\left(\quad\right)$.

   Best Case Running Time: $\Theta\left(\quad\right)$.

Extra Credit

(2 pts)**Question Extra Credit: Before**
In the space below, draw a picture of how you were feeling coming into this exam.

(2 pts)**Question Extra Credit: After**
In the space below, draw a picture of how you were feeling coming into this exam.

Identities

Nothing written on this page will be graded.

Summations

$$\sum_{i=0}^{\infty} x^i = \frac{1}{1-x} \text{ for } |x| < 1$$

$$\sum_{i=0}^{n-1} 1 = \sum_{i=1}^{n} 1 = n$$

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

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

$$\sum_{i=0}^{n} i^3 = \left(\frac{n(n+1)}{2}\right)^2 = \frac{n^4}{4} + \frac{n^3}{2} + \frac{n^2}{4}$$

$$\sum_{i=0}^{n-1} x^i = \frac{1-x^n}{1-x}$$

$$\sum_{i=0}^{n-1} \frac{1}{2^i} = 2 - \frac{1}{2^{n-1}}$$

Logs

$$x^{\log_x(n)} = n$$

$$\log_a(b^c) = c \log_a(b)$$

$$a^{\log_b(c)} = c^{log_b(a)}$$

$$\log_b(a) = \frac{\log_d(a)}{\log_d(b)}$$

Scratch Work

Nothing written on this page will be graded.

