Sample 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	11
Asymptotic Analysis	11
Heaps	15
AVL Trees	15
Hashing	11
Algorithms	4
Extra Credit	(+2)
Total	67

Section 1: ADTs and Data Structures

(5 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. Binary Search Tree:
2. Heap:
3. Queue:
4. Dictionary:
5. AVL Tree:

(6 pts)**Question 2: Name that ADT**

Write the name of the ADT that would be most appropriate for each use case below. Please name only ADTs that we have discussed in this quarter thus far.

1. There's a long line for seeking help in office hours, and we want to help students in the order that they've arrived.
2. There's a long line for seeking help in office hours, and we want to help students in order of who has received the most hearts on Ed (keeping in mind students may gain hearts as they're waiting).
3. We want to track and update the number of times that each student has come to office hours before.

Section 2: Asymptotic Analysis

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

Give a simplified Θ bound on the best and worst case running times for each method below. (By simplified we mean it should contain no constant coefficients or non-dominant terms.)

Each method 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). The comment beside each insert operation is included as a reminder of the order of the parameters.

```
// 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], i); // key, value
    }
}
```

1. Best Case: Θ( )
2. Worst Case: Θ( )

```
// add into a Binary Search Tree
public void addBST(int[] arr){
    BSearchTree<Integer> bst = new BSearchTree<>();
    for(int i = 0; i < arr.length; i++){
        bst.insert(arr[i], i); // key, value
    }
}
```

3. Best Case: Θ( )
4. Worst Case: Θ( )

```
// 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], i); // value, priority
    }
}
```

5. Best Case: Θ( )
6. Worst Case: Θ( )

```
// 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], i); // value, priority
    }
}
```

7. Best Case: Θ( )
8. Worst Case: Θ( )

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

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

$$T(n) = 3 \cdot T\left(\frac{n}{2}\right) + n^2$$

$$T(1) = 1$$

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 2.

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.

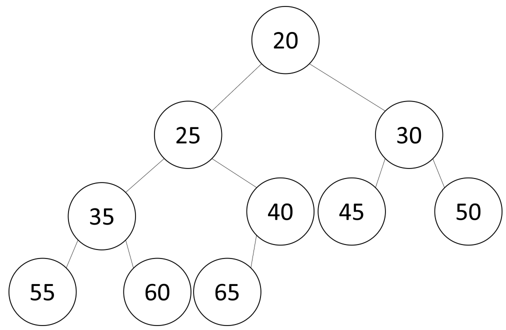
$\Theta\left(\qquad \right)$

(5 pts)**Question 5: Big O Proof**

Show that $\frac{2n(2n+1)}{2}$ belongs to $O(n^2)$.

Section 3: Heaps

The next three questions relate to the given binary min heap.



(4 pts)**Question 6: True or False**

Suppose we added a new node containing x as the right child of the node containing 40. Supposing this heap is still valid, consider each statement below. If that statement guaranteed to be true, then write "True", otherwise write "False".

1. $x \geq 65$
2. $x \geq 40$
3. $x \leq 45$
4. $x > 20$

(2 pts)**Question 7: Array Representation**

Give the array representation of the original heap given above. Place the root at index 0 of the array.

(4 pts)**Question 8: Extract**

Perform an extract operation on the heap and give the array representation of the resulting heap. Place the root at index 0 of the array.

(5 pts) **Question 9: Binary Heap Math**

Answer each question below as it relates to a 0-indexed binary min heap containing 45 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 8, which index will we compare to?

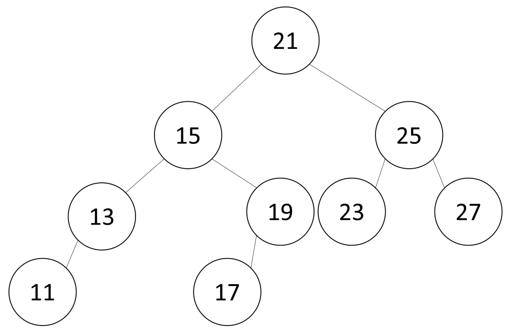
4. If we call percolate down on index 8, which two indices will we compare to?

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

Section 4: AVL Trees

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

Answer the following questions about the AVL Tree below. Each question should be considered completely independently (i.e. "reset" to the image between questions)



1. Give an integer key which, when inserted into the given AVL tree, would cause a double rotation.

2. Give the smallest integer key which, when given as the argument to insert, would not result in any rotation.

3. If we insert 10 into the tree, which node will become the deepest unbalanced node (from which we will do a rotation)?

4. If we insert 10 into the tree, what type of rotation will we do (Left, Right, Left-Right, or Right-Left)?

5. Give the shortest sequence of keys which, when inserted, would cause in a single left rotation.

(2 pts)**Question 11: Structure property**

The AVL structure property requires that, for every node in the tree, the heights of its left and right subtrees differ by at most 1. Why do we *not* require the subtrees' heights to exactly match? Answer using 1-2 sentences.

(3 pts)**Question 12: Structure Property**

Using 1-2 sentences, describe, at an intuitive level, how the AVL tree structure property improves its worse-case running time.

Section 5: Hash Tables

(2 pts)**Question 13: Clustering**

Using 1-2 sentences, explain how quadratic probing addresses the clustering problem of linear probing.

(2 pts)**Question 14: Double Hashing v. Quadratic Probing**

Using 1-2 sentences, explain one improvement that double hashing makes compared to quadratic probing.

(2 pts)**Question 15: Separate Chaining v. Open Addressing**

In 1-2 sentences, explain why, in general, a separate chaining hash table permits a larger load factor before resizing compared to an open addressing hash table.

(5 pts)**Question 16: Quadratic Probing**

Insert 27, 39, 18, 17, 29, 37, 16 (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

(4 pts)**Question 17: Find Next Largest**

For each data structure below, describe an algorithm which, when given an item that is currently in the data structure, finds the next largest item in the data structure. Assume that the item given is indeed already present in the data structure.

By next largest item we mean either the item with the smallest key larger than the one given for dictionaries, or the smallest priority that is larger than the one given for priority queues (e.g. min heaps keep the smallest priorities near the root). Make the algorithm as efficient as you can (asymptotically). You may assume the size of the data structure is greater than 1.

1. Binary Min Heap:

   Asymptotic Running Time: $\Theta\left(\qquad \right)$.

2. AVL Tree:

   Asymptotic Running Time: $\Theta\left(\qquad \right)$.

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.

Scratch Work

Nothing written on this page will be graded.

Identities

Nothing written on this page will be graded.

Summations

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

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

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

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

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

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

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

Logs

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

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

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

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