Midterm Exam

Winter 2026

Name Answer Key

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	6
Asymptotic Analysis	13
Heaps	12
AVL Trees	14
Algorithms	9
Extra Credit	(+1)
Total	54

Section 1: ADTs

(6 pts) **Question 1: ADT Applications**
For each scenario below, identify which of the ADT options provided is the most appropriate choice. Options: Stack, Queue, Priority Queue, Ordered Dictionary

1. You are building a compiler. When a function is called, you must pause the current execution, save the current state, and start the new function. When that function finishes, you must immediately return to the most recently paused state.

   Stack

2. During a match of Pokémon Unite, the game generates a stream of notifications. These notifications are displayed one at a time on the screen. Once a notification is acknowledged, the next chronological notification is shown, maintaining a consistent and fair ordering of information for all players.

   Queue

3. As an exam is being graded, student scores are recorded one by one. At any point, the system must be able to identify the current highest score. After all exams are graded, the top score is used to determine benchmarks such as the maximum score or how the class average compares to the highest performing student, regardless of the order in which the exams were graded.

   Priority Queue

4. For the 2026 Grammy Awards, the organizers maintain a directory of nominated artists. Staff members must be able to quickly look up information for a specific artist by name, but they already frequently need to generate a list of all artists whose names fall alphabetically between "G" and "M" for scheduling and presentation purposes.

   Ordered Dictionary

5. Adobe Photoshop has an undo feature. It records each change made, and selecting "undo" will revert the image back by one change.

   Stack

6. A programming competition is being held across multiple locations simultaneously. We need to maintain a live "medal tracker" which displays the current top three competitors (who would receive gold, silver, and bronze medals).

   Priority Queue

Section 2: Asymptotic Analysis

(4 pts) **Question 2: Code Analysis**
Give the best and worst case running times for each algorithm below. Both algorithms add all of the contents from a binary search tree of size n into a binary min heap of size m, but do so in a different order. The key of the item in the BST will act as the priority of the item in the heap.

Your running time may depend on *both* the values of n and m.

1. Suppose we populate the heap using an **in-order** traversal as follows:

```
public void PopulateMinHeap1(TreeNode root, MinHeap heap){
    if(root != null){
        PopulateMinHeap1(root.left, heap);
        heap.insert(root);
        PopulateMinHeap1(root.right, heap);
    }
}
```

Best Case Running Time: $\theta(n)$

Worst Case Running Time: $\theta(n \log m)$

2. Suppose we populate the heap using a **pre-order** traversal as follows:

```
public void PopulateMinHeap2(TreeNode root, MinHeap heap){
    if(root != null){
        heap.insert(root);
        PopulateMinHeap2(root.left, heap);
        PopulateMinHeap2(root.right, heap);
    }
}
```

Best Case Running Time: $\theta(n)$

Worst Case Running Time: $\theta(n \log (m+n))$

(5 pts) **Question 3: Tree Method**

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

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

$$T(1) = 1$$

For the following questions, use the tree method to solve the recurrence relation. We have broken up the process into parts to guide you through your answer. You may assume that $2n/3$ is always an integer.

1) Fill in the values of a and b.

$a =$ 3 $b =$ $\frac{3}{2}$

2) 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 the input size for each recursive call as well as the work per call.

...

3) 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.

$n \cdot 2^i$

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

$\log_{3/2}(n) = \log_2(n)/\log_2(1.5)$.

5) Give a simplified Θ bound on the solution. When simplified, n should not appear in any exponents. (Hint: the log rule $a^{\log_b(c)} = c^{log_b(a)}$ may be helpful!)

$\Theta\left(n \cdot 2^{\log_{3/2}(n)} = n \cdot n^{\log_{3/2}(2)} = n^{1+\log_{3/2}(2)} = n^{\log_{3/2}(3)} \right)$

(4 pts) **Question 4: Change c or n_0**

For each part below, we give $f(n)$ and $g(n)$. Our goal is to show that $f(n) = O(g(n))$. We have provided a choice of c and n_0 that **do not** work to show that $f(n) = O(g(n))$. First, show *why* that choice of c and n_0 does not work, then correct the issue by suggesting either a different c or a different n_0 (you must change one and the other must stay the same). You do not need to justify that your c and n_0 work.

1. $f(n) = 2n^2 + 10n - 10$, $g(n) = n^2$, $c = 2$, $n_0 = 2$

   Justification:

   consider $n = 3$. In this case $f(3) = 2 \cdot 3^2 + 10 \cdot 3 - 10 = 18 + 30 - 10 = 38$
   $c \cdot g(n) = 2 \cdot 3^2 = 18$

   Variable you're changing (c or n_0): c

   New value: any ≥ 4.5

2. $f(n) = \log_{10} n$, $g(n) = (\log_{10} n)^2$, $c = 1$, $n_0 = 1$.

   Hint: consider when squaring a value makes it smaller.

   Justification:

   Consider $n = 2$. In this case $\log_{10} 2 < 1$ and so $(\log_{10} 2)^2 < \log_{10} 2$.

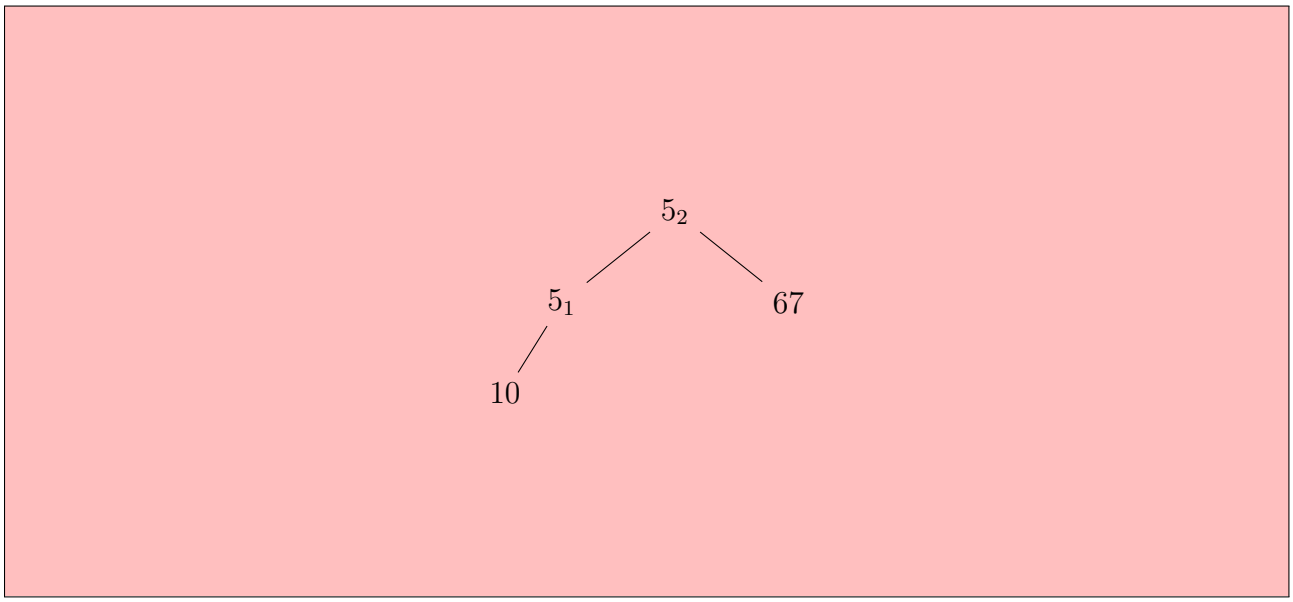
   Variable you're changing (c or n_0): n_0

   New value: any ≥ 10

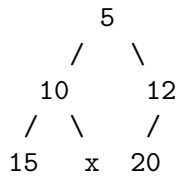
Section 3: Heaps

(5 pts) **Question 5: Heap Inserts**
Draw the Binary Min-Heap that results from performing the following operations in order: Insert(10), Insert(5_1), Insert(67), Insert(5_2), Insert(2), extract()

Assume that elements with equal keys are not swapped during insertion or deletion.



(4 pts) **Question 6: Heap Property**
Consider the following Binary Min-Heap:



1. What is the smallest integer value for x that maintains the heap property?

   10

2. Give an integer key which, when inserted, will cause exactly **2 swaps** during `percolateUp`.

   any < 5

(3 pts) **Question 7: Heap Arithmetic**
Consider a Binary Max-Heap with n nodes.

1. How many leaf nodes are in a heap with $n = 21$ nodes.

   $\text{leaves} = \lceil \frac{15}{2} \rceil = \lceil 10.5 \rceil = 11$

2. How many nodes are on the last level of a heap with 21 nodes.

   6

3. What is a height of a heap with 100 nodes? (*height* = number of edges from root to the deepest node).

   $\text{height} = \lfloor \log_2 100 \rfloor = \lfloor 6.64 \rfloor = 6$

Section 4: AVL Trees

(6 pts) **Question 8: Largest Two Items**

For each question below we give a data structure and an operation (that is not an ADT operation). For each, describe an algorithm for that operation (in either English or pseudocode), then give your algorithm's worst case asymptotic running time. Only algorithms with the minimum possible asymptotic running time will receive full credit. Correct but slower algorithms will receive partial credit.

1. Find the **maximum** element in an AVL Tree of size n.

   Algorithm:

   from the root, if the node has a right node, move to that one. Return the first node found without a right child.

   Worst Case Running time: $O(\log n)$

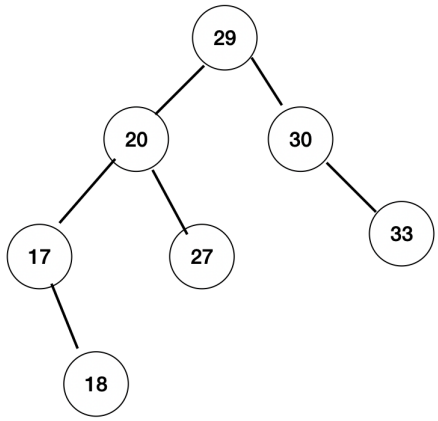
2. Find the **second largest** element in an AVL tree of size n.

   Algorithm:

   navigate to the maximum node, then find its predecessor (the next smaller key). This will be the largest of: the parent of the maximum node, the largest node in maximum's left subtree.

   Worst Case Running time: $O(\log n)$

(8 pts) **Question 9: Rotations**
Consider the following AVL tree:



For each question below, indicate the *problem node* and *type of rotation* that would occur if the given insertions are done on the tree above. Write one of the following to indicate your answer:

– **R** for a single right rotation

– **L** for a single left rotation

– **LR** for a left-then-right double rotation

– **RL** for a right-then-left double rotation

– **NR** for no rotation (write this for the "Problem node" box as well)

Each question should be considered completely independently (i.e. "reset" to the image between questions).

1. `insert(19)`

   Problem node: 17 Rotation type: L

2. `insert(15)`
   `insert(16)`

   Problem node: 29 Rotation type: R

3. `insert(28)`

   Problem node: NR Rotation type: NR

4. `insert(21)`
   `insert(22)`

   Problem node: 27 Rotation type: LR

Section 5: Algorithms - Summing the Deepest Leaves

Welcome to the last section of the exam! You're nearly there! (Unless you worked out of order.)

For both questions in this section, suppose we have a collection of n integers stored in the tree-like data structure name in the question. Our goal is to the **sum of the integers stored in the leaves of maximal depth**. For example, if we had a tree of height 3, then we would want to sum the the integers stored in leaves that are exactly 3 edges away from the root.

(5 pts) **Question 10: Binary Search Tree**
You may assume the following TreeNode class is used by BST.

```
public TreeNode(K key, V value) {
    this.key = key;
    this.value = value;
    height = 0;
    this.left = null;
    this.right = null;
}
```

1. Give an asymptotically optimal algorithm that achieves this for the Binary Seach Tree data structure.

   From the current node, if its height is 0 then return its value. Otherwise, return the sum of the recursive calls for each child whose height is 1 less than the current.

2. What is the best case runtime for your algorithm? $O(\log n)$

3. What is the worst case runtime for your algorithm? $O(n)$

4. What is the worst case runtime for your algorithm if the BST is an AVL Tree? $O(n)$

5. If the worst running time is different, give an intuitive justification for why. If the worst running time is the same, give an intuitive justification for why. 1-2 sentences should be sufficient

   It is the same because AVL trees may still have a linear number of leaves, even if the height is smaller.

(4 pts) **Question 11: Binary Min Heap**

Assume the binary min heap is represented in an array with the root stored at index 1.

1. In the array representing our tree, what is index of the first element of depth d? Assume that we define the root to have depth 1.

   $2^d - 1$

2. Give an asymptotically optimal algorithm that achieves the goal of summing the values at the leaves of largest depth of the Binary Min Heap.

   The first node at level d appears at index $2^d - 1$. The height of the heap is $\lfloor \log_2 n \rfloor$. Sum together all values from index $2^{\lfloor \log_2 n \rfloor} - 1$ up to $n - 1$

3. Give the best case runtime of your algorithm.

   $O(1)$

4. Give the worst case runtime of your algorithm.

   $O(n)$

Section 6: Extra Credit

(+0.5 pts)**Extra 1: Before**
In the space below, draw a picture of how you were feeling coming into this exam.

:$

(+0.5 pts)**Extra 2: After**
Now draw a picture of how you are feeling coming out of it.

:)

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

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

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

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

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

