23su CSE332 Midterm

Full Name: Sample Solutions

Email Address (UW NetID): samplesolutions **@uw.edu**

Instructions:

- The allotted time is 1 hour.
- Do not turn the page until the staff says to do so.
- Read the directions carefully, especially for problems that require you to show work or provide an explanation.
- This is a closed-book and closed-notes exam.
- You are NOT permitted to access electronic devices including calculators.
- You must put your final answer inside the box.
    - If you run out of space, indicate where the answer continues.
    - Try to avoid writing on the very edges of the pages as we scan the exams.
- Unless otherwise noted, O, Ω, or Θ bound must be **simplified** and **tight**.
- Unless otherwise noted, $\log$s are base 2.
- Unless otherwise noted, all material is assumed as in lecture.
- For answers that involve bubbling in a ◯ or ☐, fill in the shape completely.
- A formula sheet has been included at the end of the exam.

Advice:

- If you feel like you're stuck on a problem, you may want to skip it and come back at the end if you have time.
- Look at the question titles on the cover page to see if you want to start somewhere other than problem 1.
- **Relax and take a few deep breaths. You've got this! :-).**

Q1: Short Answer Questions (20 pts)

- O, Ω, or Θ bound must be **simplified** and **tight**. This means that, for example, $7n + 3 \in O(7n + 3)$ (not simplified) or $n \in O\left(2^{n!}\right)$ (not tight enough) are unlikely to get points.
- Unless otherwise specified, all $\log$s are base 2.
- For questions with a mathematical answer, you may leave your answer as an unsimplified formula (e.g., $7 \cdot 103$).
- Points are all or nothing (i.e., we will only grade what is inside the box).

a) (2 pts) Give a simplified, tight bound for the function $f(n) = \log\left(n!^{n}\right) + 1.001^{n}$.

$$O(\boxed{1.001^{n}})$$

b) (1 pt) Give a simplified, tight bound for the function $f(n) = \log^{2}(n) + \log\log n$.

or $O(\log(n) \cdot \log(n))$

$$O(\boxed{\log^{2} n})$$

c) (1 pt) Let $n_{0} = 1$. Give the **smallest** possible c to show that $3n^{2} + 5n + 2 \in O\left(n^{2}\right)$.

$$c = \boxed{10}$$

d) (3 pts) Let a recurrence be defined by $T(1) = a$, $T(n) = 2T(\frac{n}{2}) + c_{0}$.

Give the simplified closed form of $T(a)$ (in terms of a). You should also remove constants and lower-order terms as if you were finding the asymptotic bound.

$$T(a) = \boxed{a^{2}}$$

Hint: Picture the same tree shown in lecture except the leaves are different.

e) (2 pts) Give the worst-case runtime to delete the median element in a BST containing n elements.

$O(\boxed{n})$

f) (2 pts) Give the worst-case runtime to find the kth smallest element in a binary Min Heap containing n elements.

$O(\boxed{k \log n})$

g) (2 pts) Give the **best**-case runtime of `decreaseKey(k, p)` in a binary Min Heap containing n elements.

$O(\boxed{1})$

h) (2 pts) Give the formula to find the 4th child of a node (in index i) in a 5-ary heap. Recall the root is at index 1.

$\boxed{5i}$

i) (2 pts) Give the **best**-case runtime of finding the minimum element in an AVL Tree containing n elements.

$O(\boxed{\log n})$

j) (3 pts) Give the worst-case runtime of finding an element in a B-Tree containing n elements with parameters M and L.

or in detail, $O\left(\log(M) \cdot \log_M(n) + \log L\right)$

$O(\boxed{\log n + \log L})$

Q2: Code Analysis (18 pts)

- O, Ω, or Θ bound must be **simplified** and **tight**.
- You should use n to denote the input size.

a) (4 pts)

```
void silly(int dogs) {
  if (dogs <= 0) {
    return;
  }
  if (dogs % 2 == 0) {
    for (int i = 0; i < dogs; i++) {
      print("Woof!");
    }
  } else {
    for (int i = 0; i < dogs * dogs; i++)
{
      print(Howl!!");
    }
  }
  silly(dogs - 1);
}
```

$T(0) = c_0$

$T(n) = n^2 + T(n-1)$

Recurrence that should be memorized but also very easy to logically derive quickly.

$O(\boxed{n^3})$

b) (4 pts)

```
void seagulls(int n, int funny) {
  int i = n;
  while (i > 0) {
    for (int j = 0; j < n * n; j++) {
      funny++;
    }
    i = i / 2;
  }
  return funny;
}
```

$O(\boxed{n^2 \log n})$

c) (4 pts)

```
int hipHop(int n) {
  int i = 1;
  while (i < n * n) {
    i *= 2;
  }
  while (i > n) {
    i /= 2;
  }
  return i;
}
```

$O(\boxed{\log n})$

d) (6 pts) Assume that `BinaryMinHeap` is implemented as in lecture.

```
int heapsOfInserts(int n) {
  BinaryMinHeap output = new
BinaryMinHeap();
  int x = 0;
  for (int i = 0; i < n; i++) {
    if (i % 2 == 0) {
      x += i;
    } else {
      for (int j = i; j < n; j += 2) {
        output.insert(x);
        x += j;
      }
    }
  }
  return output;
}
```

Notice that `x` is always monotonically increasing so `output.insert(x);` is a constant time operation (i.e. directly inserting into an array with no percolation)

$O(\boxed{n^2})$

Q3: Asymptotics (10 pts)

For each of the following statements, indicate whether it is **Always True**, **Sometimes True**, or **Never True**.

- Assume that the domain and codomain of all functions in this problem are natural numbers (1, 2, 3, ...) (e.g., $f_a: \mathbb{N} \to \mathbb{N}$).
- You only need to <u>fully shade</u> the circle (i.e., no explanation needed).

a) (3 pts) $\log\left(\log\left(n^2\right)\right) \in O(\log n)$.

● **Always True**

○ **Sometimes True**

○ **Never True**

b) (3 pts) $f_b(n) \cdot n^2 \in \Theta\left(f_b(n)^2\right)$.

False when $f_b(n) = 1$

○ **Always True**

● **Sometimes True**

○ **Never True**

c) (4 pts) If $f_a(n) \in \Theta\left(g_a(n)\right)$ and $g_a(n) \in \Omega\left(h_a(n)\right)$, then $f_a(n) \in O\left(h_a(n)\right)$.

○ **Always True**

● **Sometimes True**

○ **Never True**

Q4: Solve a Recurrence (15 pts)

Solve the recurrence $T(1) = c_0$, $T(n) = 2 \cdot T\left(\frac{n}{4}\right) + n$ with the **Tree** Method.

- Assume any functions of n (e.g., $\log n$) resolves nicely to a whole number.
- To earn full credit, <u>fill in all the boxes</u> as in lecture.

a) (12 pts) Draw the tree (Note: the **Tree** itself does not have to be drawn perfectly).

i	$T(n)$ Call	# Nodes	Tree		Sum Work
0	$T(n)$	1	n	=	n
1	$T\left(\frac{n}{4}\right)$	2	$\frac{n}{4} + \frac{n}{4}$	=	$2 \cdot \frac{n}{4}$
2	$T\left(\frac{n}{4^2}\right)$	4	$\frac{n}{4^2} + \frac{n}{4^2} + \frac{n}{4^2} + \frac{n}{4^2}$	=	$2^2 \cdot \frac{n}{4^2}$
$\vdots$	$\vdots$	$\vdots$			$\vdots$
i	$T\left(\frac{n}{4^i}\right)$	2^i	$\frac{n}{4^i} + \frac{n}{4^i} + \cdots + \frac{n}{4^i} + \frac{n}{4^i}$	=	$2^i \cdot \frac{n}{4^i}$

2. Total Non-Recursive + Recursive Work = $\displaystyle\sum_{i=0}^{\log_4(n)-1} \frac{n}{2^i}$

i	$T(n)$ Call	# Nodes	Tree
$\vdots$	$\vdots$	$\vdots$	
$\log_4 n$	$T(1)$	$2^{\log_4 n}$	$c_0 + c_0 + \cdots + c_0 + c_0$

Find when the base case occurs:

When $1 = \frac{n}{4^i}$ (i.e., $i = \log_4 n$)

1. Total Base Case Work = $\displaystyle\sum_{i=1}^{2^{\log_4(n)}} c_0$

b) (1 pts) Find General Formula.

Multiple answers were accepted beyond this due to the simplicity of the base case.

$T(n) =$ Total Base Case Work + (Total Non-Recursive + Recursive Work)

$$T(n) = \boxed{\sum_{i=1}^{2^{\log_4(n)}} c_0 + \sum_{i=0}^{\log_4(n)-1} \frac{n}{2^i}}$$

c) (2 pts) Find Closed Form. Simplify as much as possible.

$$T(n) = \sum_{i=1}^{2^{\log_4(n)}} c_0 + \sum_{i=0}^{\log_4(n)-1} \frac{n}{2^i}$$

$$= \sum_{i=1}^{\sqrt{n}} c_0 + n \sum_{i=0}^{\log_4(n)-1} \frac{1}{2^i}$$

$$= \sqrt{n} \cdot c_0 + n \cdot \left(2 - \frac{1}{2^{\log_4(n)-1}}\right)$$

$$= \sqrt{n} \cdot c_0 + n \cdot \left(2 - \frac{1}{\frac{1}{2}\sqrt{n}}\right)$$

$$T(n) = \boxed{2n - 2\sqrt{n} + c_0\sqrt{n}}$$

Q5: Heaps (11 pts)

a) (1 pt each) Describe the two properties of a Min Heap.

Structure Property:	Complete Tree
Order Property:	Every (non-root) node's priority value ≥ parent's priority value

b) (4 pts) **Given a Min 3-Heap that contains values 1, 2, 3, …, 1000.**

Can the value **5** exist at this **depth** (multiple choice)?

Recall that depth is defined as the number of edges (arrows) from the root to the node (in this case, the node that represents 5).

Depth -1 is undefined and makes no sense.

Depth 0 is the root.

- [ ] **-1**
- [ ] **0**
- [x] **1**
- [x] **2**
- [x] **3**
- [x] **4**
- [ ] **5**
- [ ] **6**

c) (5 pts) Fill in the array representation of a **Binary** Min Heap after inserting **5, 7, 3, 1, 9, 2** (in that order) one by one. Recall that the root starts at index 1.

Use the blank papers for rough workings before giving your final answer.

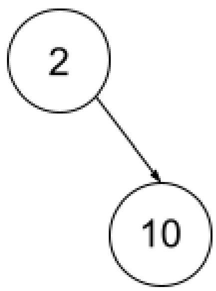
	1	3	2	7	9	5				
0	1	2	3	4	5	6	7	8	9	10

Q6: AVL Tree (7 pts)

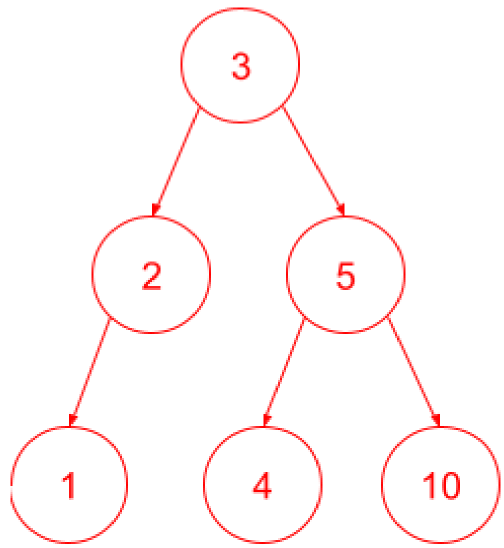
a) (4 pts) Insert **5, 3, 4, 1** (in that order) into the Initial AVL Tree.

Use the blank papers for rough workings before giving your final answer.

Initial AVL Tree:



Final AVL Tree after inserting **5, 3, 4, 1**:



Total # of single rotations required:

2

Total # of double rotations required:

1

b) (1 pt) Give the **minimum** height of an AVL Tree with **8 nodes**.

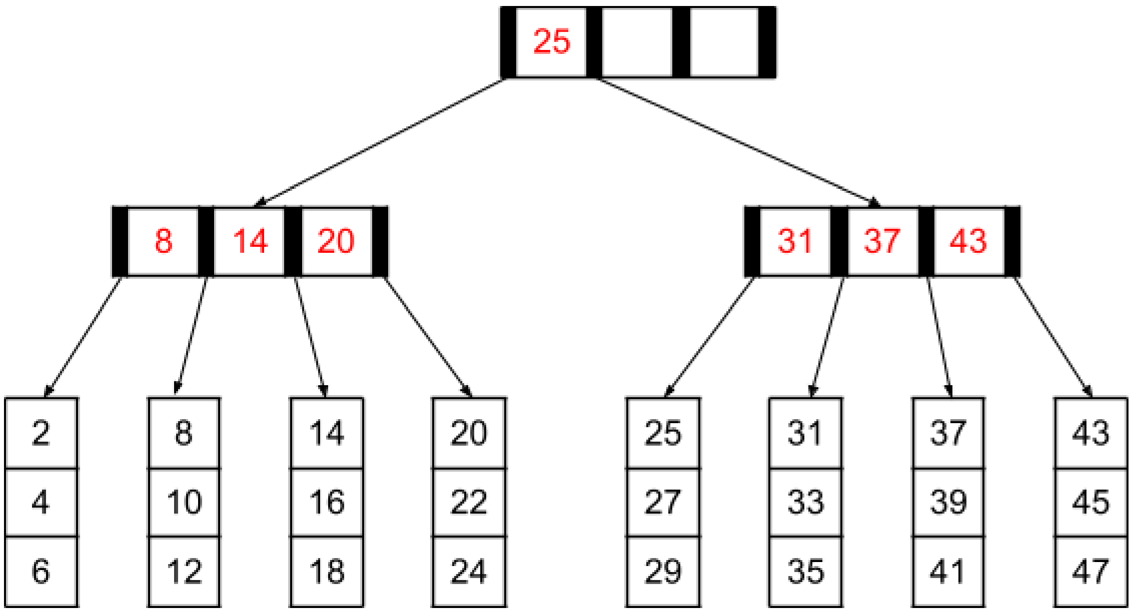
3

c) (2 pts) Give the **maximum** height of an AVL Tree with **8 nodes**.

3

Q7: B-Trees (5 pts)

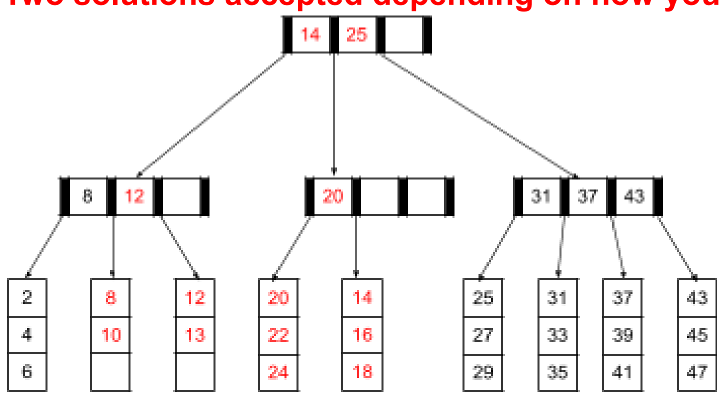
a) (1 pt) In this B-Tree with parameters $M = 4$ and $L = 3$ shown, write in the correct values for the internal nodes.

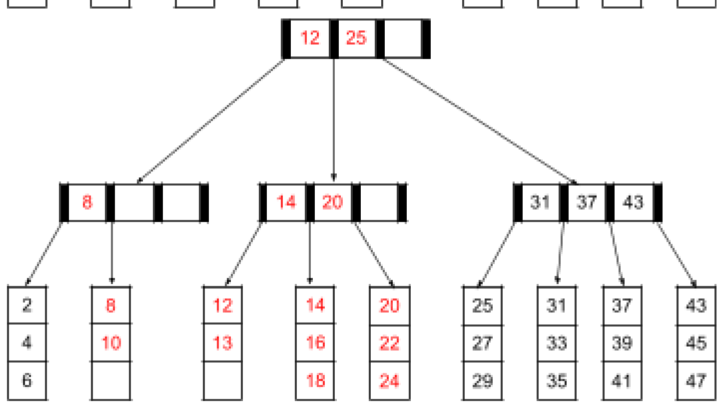


b) (3 pts) Starting with the **B-Tree from a)**, draw the B-Tree after **inserting 13**.

Use the blank papers for rough workings before giving your final answer.

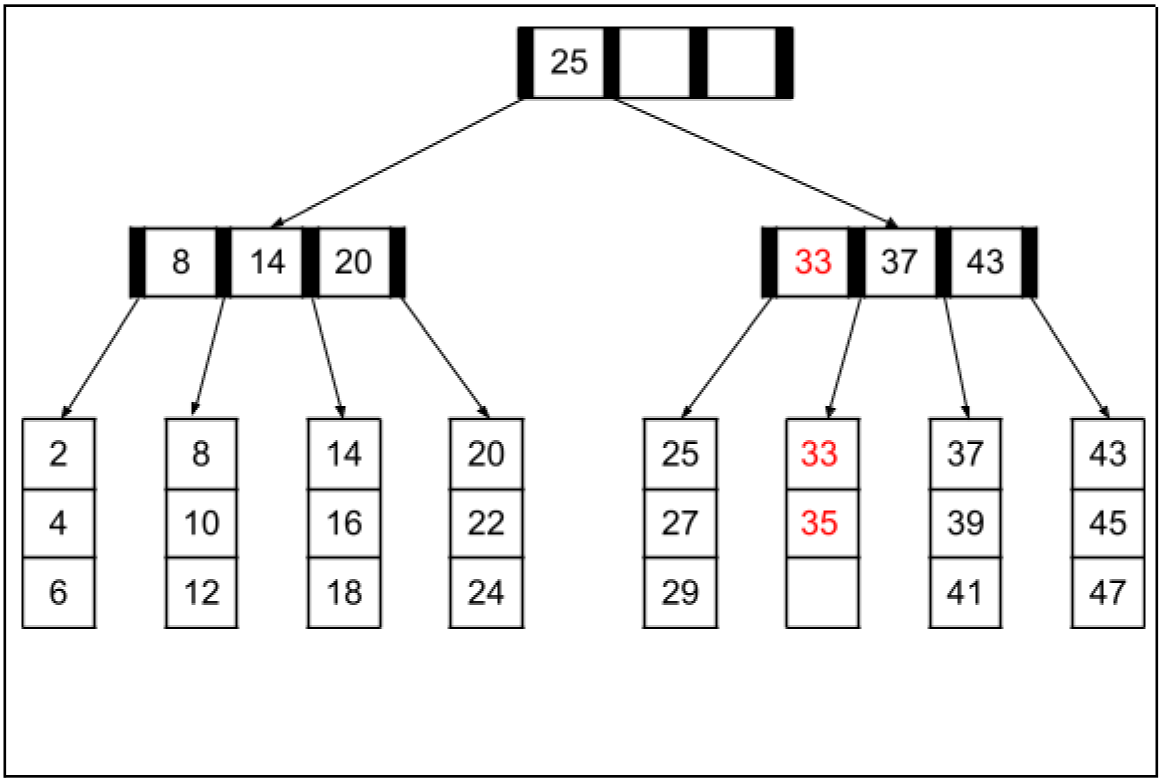
Two solutions accepted depending on how you split the internal node:





c) (1 pt) Starting with the **B-Tree from a)**, draw the B-Tree after **deleting 31**.

Use the blank papers for rough workings before giving your final answer.



CSE 332: Data Structures and Parallelism

Useful Math Identities

Summations

1. $\sum_{i=0}^{\infty} x^{i} = \frac{1}{1-x}$ for $|x| < 1$
2. $\sum_{i=0}^{n-1} 1 = \sum_{i=1}^{n} 1 = n$
3. $\sum_{i=0}^{n} i = 0 + \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$
4. $\sum_{i=1}^{n} i^{2} = \frac{n(n+1)(2n+1)}{6} = \frac{n^{3}}{3} + \frac{n^{2}}{2} + \frac{n}{6}$
5. $\sum_{i=1}^{n} i^{3} = \left(\frac{n(n+1)}{2}\right)^{2} = \frac{n^{4}}{4} + \frac{n^{3}}{2} + \frac{n^{2}}{4}$
6. $\sum_{i=0}^{n-1} x^{i} = \frac{1-x^{n}}{1-x}$
7. $\sum_{i=0}^{n-1} \frac{1}{2^{i}} = 2 - \frac{1}{2^{n-1}}$

Logs

1. $x^{\log_{x} n} = n$
2. $a^{\log_{b} c} = c^{\log_{b} a}$
3. $\log_{b} a = \frac{\log_{d} a}{\log_{d} b}$

This page has been intentionally left blank.