CSCI 3104 PS2b

Jonathan Phouminh

TOTAL POINTS

38 / 51

QUESTION 1

1 4 / 4

✓ **+ 1 pts** **Correct asymptotic relation**

✓ **+ 3 pts** **Correct application of L'Hospital's rule**

+ 1 pts Partially correct application of L'Hospital's rule

+ 0 pts Incorrect

QUESTION 2

2 6 / 6

✓ **+ 1 pts** **Correct asymptotic relation**

✓ **+ 5 pts** **Correct application of ratio test or other method**

+ 2 pts Partially correct application of ratio test or other method

+ 0 pts Incorrect

- 1 pts Incorrect order of operations for 5(n+1)

+ 5 pts Minor Algebra Error

QUESTION 3

3 4 / 4

✓ **+ 1 pts** **Totally Correct (no logical flaws)**

✓ **+ 1 pts** **Showed work (such as finding a & b, setting N^(epsilon)**

✓ **+ 1 pts** **Found out that it follows the case where log_b(a) is bigger than c of the master theorem**

✓ **+ 1 pts** **got the final notation right; big-theta(or O) of n^log_5(4)**

+ 0 pts No work or totally incorrect

- 0.5 pts C should not be zero.

QUESTION 4

4 4 / 6

+ 2 pts Unrolled the equation correctly and pattern found is correct.

✓ **+ 1 pts** **Minor mistakes while expanding the equation and finding the pattern. Please refer to the solution file.**

✓ **+ 2 pts** **Showed where the recursion ends using appropriate base case.**

+ 1 pts Minor mistakes while showing where the recursion ends and/or not identifying the appropriate base case. Please refer to the solution file.

+ 2 pts Substituted and solved the recurrence relation correctly by showing correct g(n).

✓ **+ 1 pts** **Minor mistakes in either substituting or completely solving the recurrence relation in the final step. Please refer to the solution file.**

+ 0 pts Empty solution or incorrect approach submitted. Please refer to the solution file.

QUESTION 5

5 8 / 8

✓ **+ 2 pts** **Right recurrence relation**

✓ **+ 4 pts** **Solving the recurrence**

✓ **+ 2 pts** **The correct worst case time complexity**

+ 2 pts Partially correct recurrence solving.

+ 1 pts Partially correct worst case time complexity

+ 1 pts Partially correct recurrence relation

+ 0 pts Incorrect or not attempted

QUESTION 6

16 pts

6.1 3 / 3

✓ **+ 0.5 pts** **Get base case correctly**

✓ **+ 1.5 pts** **Get recurrence of recursive part correctly--3T(n/3).**

✓ **+ 1 pts** **Get time complexity at each step of recursion correctly--O(1). (Which is the min() function)**

+ **0 pts** Incorrect/Not attempted.

It would be better if you can divide your expression into two parts.

6.2 1 / 3

✓ + **1 pts** **Correct values of a, b and f(n)**

+ **1 pts** Shown precise work for the answer
+ **1 pts** Correct final runtime complexity
+ **0 pts** Incorrect answer or not attempted
+ **1 pts** Partially correct answer
+ **0 pts** Incorrect recurrence relation

6.3 2 / 6

✓ + **2 pts** **Get the height of the tree correctly.**

+ **2 pts** Get the cost at each level correctly.
+ **1 pts** Get the right mathematic expression correctly.
+ **1 pts** Get the right answer correctly.--(O(n))
+ **0 pts** Incorrect/Not attempted.

6.4 2 / 4

+ **2 pts** Correct recurrence relation mentioned

✓ + **1 pts** **Minor mistakes in obtaining recurrence relation equation. Please refer to the solution file.**

+ **2 pts** Correct tight bound mentioned either by completely proving it or referring to the previous mentioned proof

✓ + **1 pts** **Minor mistakes in mentioning correct tight bound or proving correct tight bound. Please refer to the solution file.**

+ **0 pts** Empty solution or incorrect approach. Please refer to the solution file.
+ **4 pts** Correct but a more detailed solution is expected with recurrence relation mentioned.
+ **0.5 pts** Constants present in tight bound

QUESTION 7

7 4 / 7

✓ + **2 pts** **Correct recurrence relation**

+ **1 pts** Correct Base case

✓ + **2 pts** **Shown work for worst case time complexity**

+ **2 pts** Correct final runtime complexity
+ **1 pts** Partially correct recurrence relation.
+ **0 pts** Incorrect answer or not attempted
+ **1 pts** Work shown for time complexity is not precise enough
+ **0.5 pts** Final runtime complexity is not precise

Big-O is the upper bound, whereas Big-theta is the tight bound notation.

Name: Jonathan Phouminh

ID: 106054641

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**

Problem Set 2b (51 points) **Fall 2019, CU-Boulder**

Instructions for submitting your solution:

- The solutions **should be typed** and we cannot accept hand-written solutions. Here's a short intro to Latex.
- You should submit your work through **Gradescope** only.
- If you don't have an account on it, sign up for one using your CU email. You should have gotten an email to sign up. If your name based CU email doesn't work, try the identikey@colorado.edu version.
- Gradescope will only accept **.pdf** files (except for code files that should be submitted separately on Gradescope if a problem set has them) and **try to fit your work in the box provided**.
- You cannot submit a pdf which has less pages than what we provided you as Gradescope won't allow it.
- Verbal reasoning is typically insufficient for full credit. Instead, write a logical argument, in the style of a mathematical proof.
- For every problem in this class, you must justify your answer: show how you arrived at it and why it is correct. If there are assumptions you need to make along the way, state those clearly.
- You may work with other students. However, **all solutions must be written independently and in your own words.** Referencing solutions of any sort is strictly prohibited. You must explicitly cite any sources, as well as any collaborators.

Name: Jonathan Phouminh

ID: 106054641

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**

Problem Set 2b (51 points) **Fall 2019, CU-Boulder**

1. *(4 pts) Using L'Hopital's Rule, show that* $\ln(n) \in \mathcal{O}(\sqrt{n})$.

Solution.

Show that

$$\lim_{n\to\infty} \frac{ln(n)}{\sqrt{n}} = 0$$

would tell us that $g(n)$ grows faster than $f(n)$

$$\lim_{n\to\infty} \frac{ln(n)}{\sqrt{n}} = \frac{\infty}{\infty} \xrightarrow{LH}$$

$$\lim_{n\to\infty} \frac{\frac{1}{n}}{\frac{1}{2}n^{-\frac{1}{2}}} = \frac{\infty}{\infty} \rightarrow$$

$$\lim_{n\to\infty} \frac{\frac{1}{n}}{\frac{1}{2}n^{-\frac{1}{2}}} \rightarrow$$

$$\lim_{n\to\infty} \frac{2\sqrt{n}}{n} \xrightarrow{LH}$$

$$\lim_{n\to\infty} \frac{1}{\sqrt{n}} = 0$$

Since the limit equals 0, $f(n) \in O(g(n))$ by the limit comparison test.

Name: Jonathan Phouminh

ID: 106054641

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**

Problem Set 2b (51 points) **Fall 2019, CU-Boulder**

2. *(6 pts) Let $f(n) = (n-3)!$ and $g(n) = 3^{5n}$. Determine which of the following relations* ***best*** *applies: $f(n) \in \mathcal{O}(g(n))$, $f(n) \in \Omega(g(n))$, or $f(n) \in \Theta(g(n))$. Clearly justify your answer. You may wish to refer to Michael's Calculus Review document on Canvas.*

   Solution.

   For this problem we must use the ratio test to observe the limit $\frac{f(n))}{g(n))}$ as it goes to infinity.

   Ratio Test:

$$\lim_{n\to\infty} \frac{\frac{(n-2)!}{3^{5n+5}}}{\frac{(n-3)!}{3^{5n}}} \to \lim_{n\to\infty} \frac{(n-2)!}{3^{5n}+3^5} * \frac{3^{5n}}{(n-3)!} \xrightarrow[3^{5n}]{cancel} \lim_{n\to\infty} \frac{(n-2)!}{3^5(n-3)!} \to \lim_{n\to\infty} \frac{(n-2)(n-3)!}{3^5(n-3)!}$$

$$\lim_{n\to\infty} \frac{(n-2)}{3^5} = \infty$$

   By the ratio test we end up with the limit being infinity, therefore $f(n)$ grows faster thus, $f(n) \in \Omega(g(n))$

Name: Jonathan Phouminh

ID: 106054641

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**

Problem Set 2b (51 points) **Fall 2019, CU-Boulder**

3. *(4 pts) Let $T(n) = 4T(n/5) + \log(n)$, where $T(n)$ is constant when $n \leq 2$.* ***Using the Master Theorem****, determine tight asymptotic bounds for $T(n)$. That is, use the Master Theorem to find a function $g(n)$ such that $T(n) \in \Theta(g(n))$. Clearly show all your work.*

   Solution.

   We are given the equation, T(n)=4T(n/5)+log(n), to use masters theorem we require the equation to be in the form $T(n) = AT(n/b) + n^c$.
   So we will consider the equation,

$$T(n) < 4T(n/5) + n^{\epsilon}$$

   where

$$\epsilon = 0.0001(\epsilon > 0)$$

   We know that

$$a = 4, b = 5, \log_5 4 = 0.86$$

   we also know that

$$n^{\epsilon}$$

   grows faster asymptotically than

$$log(n)$$

   By Masters Theorem we will encounter the case where

$$\log_b a = c$$

   so by MT we have that T(n) is bounded by $O(n^{.83})$ since

$$T(n) < 4T(n/5) + n^{\epsilon}$$

Name: Jonathan Phouminh

ID: 106054641

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 2b (51 points) **Fall 2019, CU-Boulder**

4. *(6 pts) Let $T(n) = T(n-3) + T(3) + n$, where $T(n)$ is constant when $n \leq 3$.* ***Using unrolling****, determine tight asymptotic bounds for $T(n)$. That is, find a function $g(n)$ such that $T(n) \in \Theta(g(n))$. Clearly show all your work.*

Solution.

$$T(n) = T(n-3) + T(3) + n$$

is constant for n $\leq$ 3 so let our base case be equal to T(0) = a also T(3) is always constant so we can immediately disregard that value when unrolling and we will have

$$T(n) = T(n-3) + n$$

$$T(n) = T(n-6) + n + (n-3)$$

$$T(n) = T(n-9) + n + (n-3) + (n-6)$$

$$-$$

$$-$$

$$-$$

From here we can generalize the recursive process,

$$T(n) = T(n-3k) + kn - 3^{k-1}$$

To hit our base case we solve for what value k makes $T(n-3k$ equal zero and substitute for all k's, $k = \frac{n}{3}$

$$T(n) = T(n - \frac{3n}{3}) + \frac{n^2}{3} - 3^{\frac{n}{3}-1}$$

$$T(n) = T(n-n) + \frac{n^2}{3} - 3^{\frac{n}{3}-1}$$

we hit our base case and are left with

$$T(n) = \frac{n^2}{3} - 3^{\frac{n}{3}-1}$$

Therefore,

$$T(n) \in O(n^2)$$

QED

Name: Jonathan Phouminh

ID: 106054641

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**

Problem Set 2b (51 points) **Fall 2019, CU-Boulder**

5. *(8 pts) Consider the following algorithm, which takes as input a string of nested parentheses and returns the number of layers in which the parentheses are nested. So for example, "" has 0 nested parentheses, while ((())) is nested 3 layers deep. In contrast, ()() is* ***not*** *valid input. You may assume the algorithm receives only valid input. For the sake of simplicity, the string will be represented as an array of characters.*

   Find a recurrence for the worst-case runtime complexity of this algorithm. Then ***solve*** *your recurrence and get a tight bound on the worst-case runtime complexity.*

```
CountParens(A[0, ..., 2n-1]):
    if A.length == 0:
        return 0
    return 1 + CountParens(A[1, ..., 2n-2])
```

   Solution.

Cost	Time
c1	1
c2	1
c3	T(n-2) + 1

$$T(n) = c1 + c2 + T(n-2)$$
$$BaseCase : T(0) = a$$
$$T(n) = c1 + c2 + T(n-2)$$
$$T(n) = T(n-2) + 1$$
$$T(n) = T(n-4) + 1 + 1$$
$$T(n) = T(n-6) + 1 + 1 + 1$$
$$-$$
$$-$$
$$T(n) = T(n-n) + n$$

   base case hits. . .

$$T(n) = n$$

   Therefore,

$$T(n) \in O(n)$$

   QED

Name: Jonathan Phouminh

ID: 106054641

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 2b (51 points) **Fall 2019, CU-Boulder**

6. *(16 pts) For the given algorithm to find* ***min****, solve the following.*

You may assume the existence of a ***min*** *function taking $\mathcal{O}(1)$ time, which accepts at most three arguments and returns the smallest of the three.*

```
FindMin(A[0, ..., n-1]):
     if A.length == 0:
        return infinity
     else if A.length == 1:
         return A[0]
     else if A.length == 2:
        return min(A[0], A[1])
     return min( FindMin(A[0, ..., floor(n/3)],
                 FindMin(A[floor(n/3) + 1, ..., floor(2n/3)],
                 FindMin(A[floor(2n/3) + 1, ..., n-1])
             )
```

(a) (3pts) Find a recurrence for the worst-case runtime complexity of this algorithm.
Solution.

Cost	Time c1
1	
c2	2
c3	3
c4	4
c5	5
c6	2

Number of sub-problems: 3

Size of sub-problems: n/3

Time to combine(and divide) sub-problems:

$$\Theta(1)$$

Therefore,

$$T(n) = 3T(\frac{n}{3}) + \Theta(1)$$

(b) (3 pts) Solve your recurrence **using the Master's Method** and get a tight bound on the worst-case runtime complexity.

Solution.

$$T(n) = 3T(\frac{n}{3}) + \Theta(1)$$

$$a = 3, b = 3, c^n = 1$$

Therefore $c = 1$

$$\log_b a = c$$

so by Masters Theorm we have

$$T(n) \in \Theta(nlog(n))$$

(c) (6 pts) Solve your recurrence **using the recurrence tree method** and get a tight bound on the worst-case runtime complexity. (It's ok to put an image of your hand drawn tree but label it neatly.)

Solution.

Image of tree diagram inserterd below, tried my best to make it as big as possible while being readable.

Name: Jonathan Phouminh

ID: 106054641

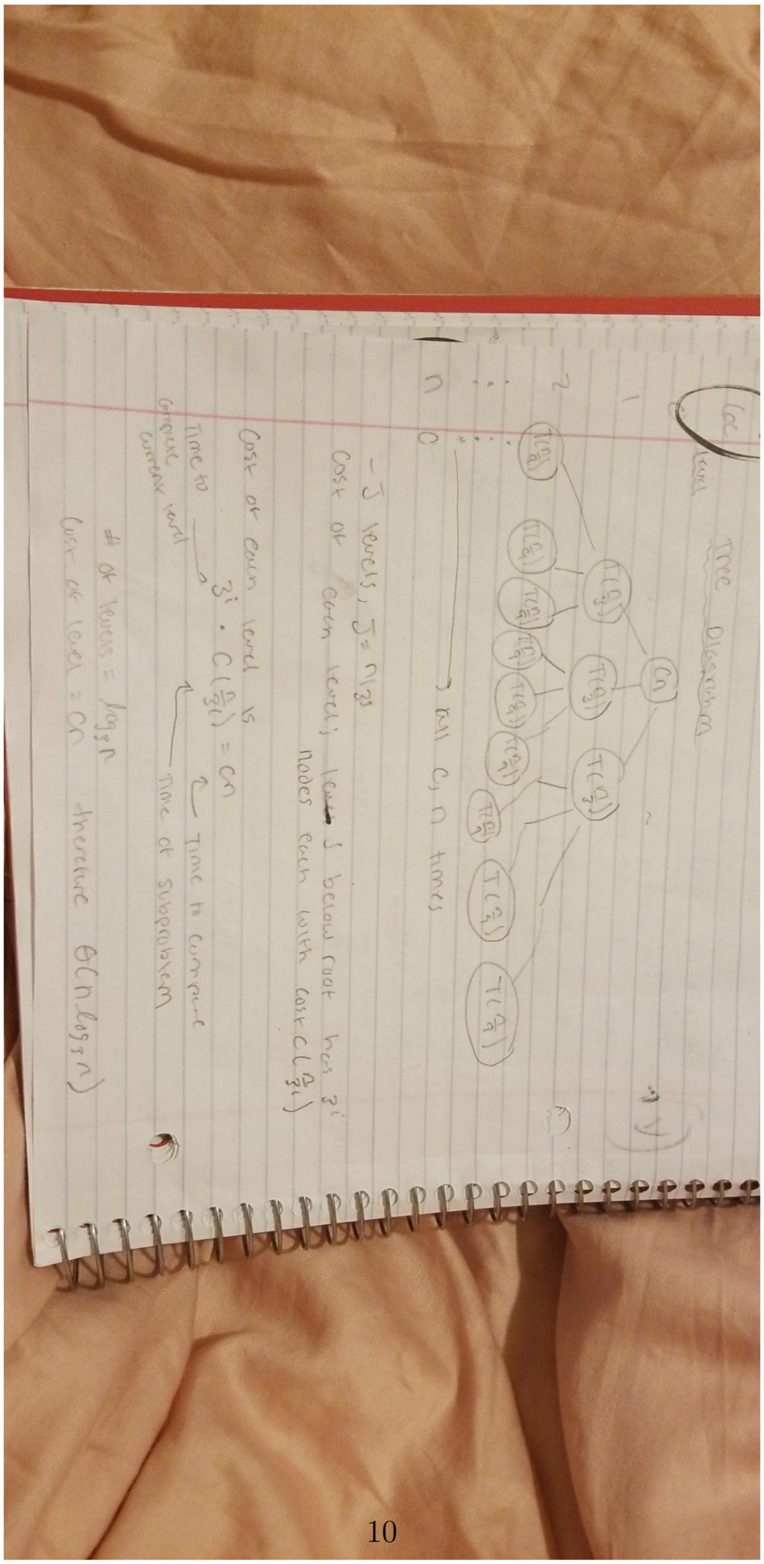

Figure 1: An image of a galaxy

(d) (4 pts) Give a tight bound (Θ bound) on the number of `return` calls this algorithm makes. Justify your answer.

Solution.

$\sum_{n=0}^{n} 3^i - 1.$ is the summation of all returns from level zero to the nth level. Every level has 3^n nodes so you would add them all up and then subtract 1 for the root node which doesn't return.

The summation is a geometric sequence so it can be written in

$$S = \frac{1 - 3^n}{1 - r}$$

Thus, number of return call is bounded by

$$\Theta(3^n)$$

Name: Jonathan Phouminh

ID: 106054641

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**

Problem Set 2b (51 points) **Fall 2019, CU-Boulder**

---

7. *(7 pts) Consider the following algorithm that sorts an array.*

   Express and provide the worst-case runtime complexity of this algorithm as a function of n, where n represents the size of the array. Provide a tight bound on the worst-case runtime complexity.

```
buffSort(A, size):
    if size <= 1:
        return

    buffSort(A, size-1)

    foo = Arr[size-1]

    for(index = size-2; index >= 0 AND A[index] > foo; index--)
        A[index+1] = A[index]

    A[index+1] = foo
```

Solution.

Cost	Time
c1	1
c2	1
c3	T(n-1)
c4	1
c5	n-2
c6	1
c7	1

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

$$T(n-1) = T(n-2) + 2n + 5$$

$$T(n-2) = T(n-3) + 3n - 9$$

$$+4n - 14$$

$$-$$

$$-$$

$$-$$

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

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

$$O(n^2)$$

COLLABORATION PARTNERS:
BRANDON ALLEN
BAO NGUYEN
ZACHARY CHOMMALLA
TRISTAN THOMAS
BENJAMIN GONZALES
AUSTIN PARK
MICHAEL REN
MARVIN NGUYEN