CSCI 3104 PS2b

Luna Mcbride

TOTAL POINTS

39 / 51

QUESTION 1

1 2 / 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

lim n->inf f(n) does not equal lim n->inf f(n)*f(n)

QUESTION 2

2 4 / 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

3^(5(n+1)) = 3^5n*3^5. Not 3^5n*3

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 6 / 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 2 / 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.

You need to give attention to the min() function.

6.2 3 / 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 1 / 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 7 / 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

Name: Luna McBride
ID: 107607144

CSCI 3104, Algorithms
Problem Set 2b (51 points)

Profs. Hoenigman & Agrawal
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: Luna McBride
ID: 107607144

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.
Limit Comparison: $\frac{lim}{n->\infty}\frac{f(n)}{g(n)}$, L'Hopital's Rule: $\frac{lim}{n->\infty}\frac{f(n)}{g(n)} = \frac{lim}{n->\infty}\frac{f`(n)}{g`(n)}$

$f(x) = ln(n), g(x) = \sqrt{n}$

$\frac{lim}{n->\infty}\frac{ln(n)}{\sqrt{n}}$

$\frac{\infty}{\infty} --> $ Infinity over Infinity, so L'Hopital's rule is required.

$\frac{lim}{n->\infty}\frac{ln(n)}{\sqrt{n}} = \frac{lim}{n->\infty}\frac{\frac{1}{n}}{\frac{1}{2\sqrt{n}}}$

$\frac{lim}{n->\infty}\frac{2\sqrt{n}}{n}$

$\frac{lim}{n->\infty}(\frac{2\sqrt{n}}{n})^2 --> \frac{4n}{n^2}$

$\frac{lim}{n->\infty}\frac{4n}{n^2} --> \frac{\infty}{\infty^2}$

The denominator is decreasing at a faster speed than the top is increasing, making the value go to 0. As such, since this has to be 0 to be a big O, $\ln(n) \in \mathcal{O}(\sqrt{n})$.

Name: Luna McBride
ID: 107607144

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.
Ratio Test:$\frac{a_{n+1}}{a_n}$

$a_n = \frac{(n-3)!}{3^{5n}}, a_{n+1} = \frac{(n-2)!}{3^{5n+1}}$

$\frac{lim}{n->\infty} \frac{\frac{(n-2)!}{3^{5n+1}}}{\frac{(n-3)!}{3^{5n}}}$

$\frac{lim}{n->\infty} \frac{(n-2)!*3^{5n}}{(n-3)!*3^{5n+1}}$

$\frac{lim}{n->\infty} \frac{(n-3)!*(n-2)*3^{5n}}{(n-3)!*3*3^{5n}}$

$\frac{lim}{n->\infty} \frac{(n-2)*3^{5n}}{3*3^{5n}}$

$\frac{lim}{n->\infty} \frac{(n-2)}{3}$

$\frac{\infty}{3} --> \infty$

Since the relation goes to ∞ instead of a constant or 0, this relationship is a $f(n) \in \Omega(g(n))$ relationship.

Name: Luna McBride
ID: 107607144

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.

   $a = 4, b = 5$ (taken from aT($\frac{n}{b}$)

   $log_5(4) = 0.8614$

   Use trick from Professor Shiv's lecture notes

   $\in = 0.1$

   $0.8614 > 0.1$

   $T(n) = \Theta(n^{log_5(4)})$

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.
   With help from CA Alici

   $T(0) = a$
   $T(3)$ is being used as a constant, so I will use c to represent it

   $T(n) = T(n-3) + c + n$
   $T(n) = [T(n-6) + c + (n-3)] + c + n - - > T(n-6) + 2c + n + (n-3)$
   $T(n) = [T(n-9) + c + (n-6)] + 2c + n + (n-3) - - > T(n-9) + 3c + n + (n-3) + (n-6)$

   $T(n) = T(n-3i) + ci + (n - \sum_{k=0}^{i-1} 3k) - - > T(n) = T(n-3i) + ci + (n - 3i^2)$

   $n - 3i = 0$
   $n = 3i$
   $i = \frac{n}{3}$

   $T(n) = T(n - 3 * \frac{n}{3}) + \frac{cn}{3} + (n - 3(\frac{n}{3})^2) - - > a + \frac{cn}{3} + (n - \frac{n^2}{3})$

   The largest term is order n^2, so $\Theta(n^2)$

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.
With help from CA Alici
Base case: T(0)=a=1

T(2n)=aT(g(n))+f(n)
T(2n)=T(2n-2)+2

$T(2n) = [T(2n-4)+2]+2 -- > T(2n-4)+4$
$T(2n) = [T(2n-6)+2]+4 -- > T(2n-6)+6$
$T(2n) = T(2n-2n)+2n -- > T(0)+2n$
$a+2n -- > 2n+1$

Constants and multiplication factors do not matter as much, so $\Theta(n)$

Name: Luna McBride
ID: 107607144

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.

   There are 3 calls to FindMin, so a=3

   The FindMins go in ranges of $\frac{n}{3}$ and that is structured $\frac{n}{b}$, so b=3

   Cost per line of the rest per worst case:
   if A.length $==0 -->1$
   return infinity $-->0$
   else if A.length $==1 -->1$
   return A[0] $-->0$
   else if A.length $==2$: $-->1$
   return min(A[0], A[1]) $-->0$

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

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

Solution.

a=3, b=3. 3=3*n^0, so c=0

$log_b(a) = log_3(3) = 1$

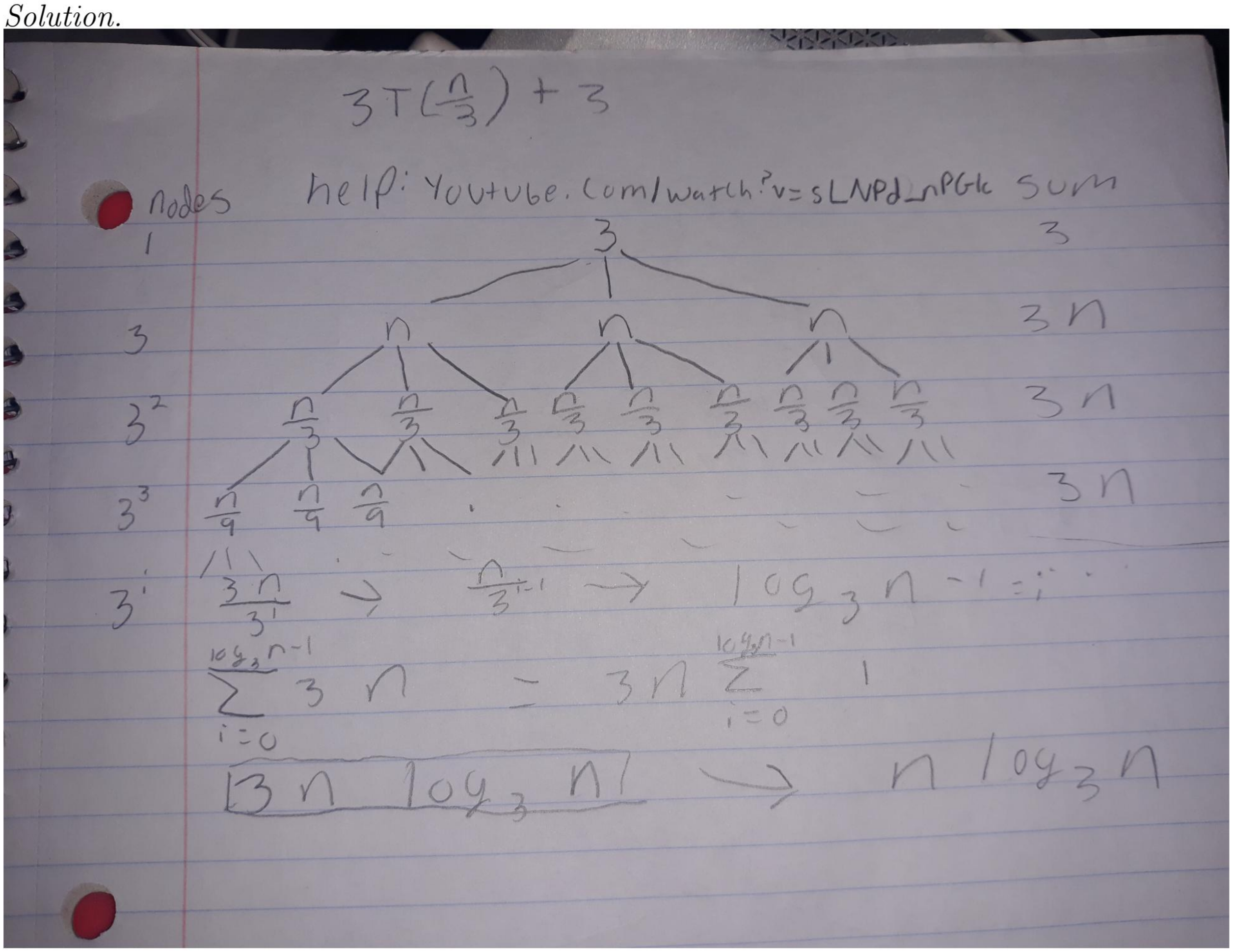
$log_b(a) > c - - > 1 > 0$

$T(n) = \Theta(n^{log_b(a)})$

$T(n) = \Theta(n)$

(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.



$T(n) = \Theta(nlog_3(n))$

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

Solution.

Return calls occur once every recursive step. Since the array lasts 1 to n (or 0 to (n-1) really, but for us counting, that is 1 to n being counted). All values need to be checked for minimum status, so there is a return every value, making for n returns no matter worst case or best case.

Therefore, T(n)=$\Theta(n)$

Name: Luna McBride
ID: 107607144

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.

T(0)=c
a=1, as there is one recursive call to buffSort.
t(n)=T(n-1)+n+2
$T(n) = [T(n-2) + (n-1) + 2] + n + 2 -- > T(n) = T(n-2) + n + (n-1) + 4$
$T(n) = [T(n-3) + (n-2) + 2] + n + (n-1) + 4 -- > T(n-3) + n + (n-1) + (n-2) + 6$
$T(n) = T(n-n) + \sum_{k=0}^{n} n - k + 2n -- > c + \frac{n}{2}(n+1) + 2n$
$T(n) = c + \frac{1}{2}n^2 + \frac{1}{2}n + 2n -- > c + \frac{1}{2}n^2 + \frac{5}{2}n$

The highest value is $\frac{1}{2}n^2$, which we can ignore the front on.
$\Theta(n^2)$