CS 141 homework1

Celyna Su

TOTAL POINTS

88 / 100

QUESTION 1

1 Problem 0 20 / 20

✓ **- 0 pts** **Everyone gets 20 points**

QUESTION 2

2 Problem 1 20 / 20

✓ **- 0 pts** **Correct**

- 20 pts No answer

Assuming 1st function to be 2^(2^(n+1)) instead of 2^(2n+1).

QUESTION 3

3 Problem 2 15 / 20

✓ **- 0 pts** **Correct**

- 20 pts No answer

- 5 pts The inner loop (while k) time complexity is incorrect/ not explained

- 5 pts The for loop (loop for j) time complexity is incorrect/ not explained

- 5 pts The final time complexity is incorrect

- 5 pts Analysis of loop for i incorrectly/ not explained

✓ **- 5 pts** **Solution is correct but the explanation is not clear**

Next time please provide more explanation.

QUESTION 4

4 Problem 3 13 / 20

- 0 pts Correct

- 12 pts Incorrect solution

- 20 pts No solution

✓ **- 7 pts** **O(n^2) solution provided**

- 1 pts We should also consider the case in which n is a power of 2.

- 0.5 pts Extra 2^k is missing.

- 1 pts We should also consider the case in which n is not a power of 2.

QUESTION 5

5 Problem 4 20 / 20

- 0 pts Click here to replace this description.

+ 1 **Point adjustment**

CS 141, Spring 2019 Homework 1

Posted: April 4th, 2019 Due: April 11th, 2019, 11:59pm

Name: Celyna Su

Student ID #: 862037643

- You are expected to work on this assignment on your own
- Use pseudocode, Python-like or English to describe your algorithms. Absolutely no C++/C/Java
- When designing an algorithm, you are allowed to use any algorithm or data structure we explained in class, without giving its details, unless the question specifically requires that you give such details
- Always remember to analyze the time complexity of your algorithms
- Homework has to be submitted electronically on Gradescope by the deadline. No late assignments will be accepted

Problem 0. (20 points)

- Go to Gradescope (`https://gradescope.com/` or follow the link from the class CS 141 webpage) and sign up using the entry code M2B7GD; please make sure that **your name is correctly typed and matches university records**, also please **enter your student id correctly**; once you have completed this homework, submit the PDF via Gradescope by April 11th, 2019, 11:59pm

- Go to the Piazza discussion board (`https://piazza.com/ucr/spring2019/cs141/` or follow the link from the class CS 141 webpage), register yourself, then select `hw1` from the top menu and post a short message to introduce yourself.

Answer: I did it!

Problem 1. (20 points) Order the following list of functions by the big-Oh notation, i.e., rank them by order of growth. Group together (for example, by underlining) those functions that are big-Theta of one another. Logarithms are base two unless indicated otherwise.

$3n+2$	n^3+n^2	$\log^2 n$	$\log\log n$	2^{2^n}	4^{n-1}
$2^{\log n}$	$2^{\sqrt{2\log n}}$	$\sqrt{n}$	n^3	1	$3^{n/3}$
$10000n^2$	$2^{2^{n+1}}$	e^n	$n^{\log\log n}$	$\log n$	$(\log n)^2$
2^n	$n3^n$	$4^{\log n}$	$4n^2/\sqrt{n}$	$\sqrt{\log n}$	$n\log n$

Answer:

$2^{(2n+1)}\quad 2^{2^n}\quad 4^{n-1}\quad n3^n\quad e^n$

$2^n\quad 3^{n/3}\quad n^{\log\log n}\quad \underline{n^3+n^2, n^3}\quad \underline{1000n^2,\ 4^{\log n}}$

$\frac{4n^2}{\sqrt{n}}\quad n\log n\quad \underline{3n+2, 2^{\log n}}\quad \sqrt{n}\quad 2^{\sqrt{2\log n}}$

$\underline{\log^2 n,\ (\log n)^2}\quad \log n\quad \sqrt{\log n}\quad \log\log n\quad 1$

Problem 2. (20 points)

Give a tight bound (using the big-theta notation) on the time complexity of following method as a function of n. For simplicity, you can assume n to be a power of two.

```
Algorithm WeirdLoop ($n$ : integer)
  $i \leftarrow n$
  while $i \geq 1$ do
    for $j \leftarrow 1$ to $i$ do
      $k \leftarrow 1$
      while $k \leq n$ do
        $k \leftarrow 2k$
    $i \leftarrow i/2$
```

Answer:
$(n + \frac{n}{2} + \frac{n}{4} + ... + 2 + 1) \log n \leq 2n \log n \in \Theta(n \log n)$

Problem 3. (20 points)

You are facing a high wall that stretches infinitely in both directions. There is a door in the wall, but you don't know how far away or in which direction. It is pitch dark, but you have a very dim lighted candle that will enable you to see the door when you are right next to it. Show that there is an algorithm that enables you to find the door by walking at most $O(n)$ steps, where n is the number of steps that you would have taken if you knew where the door is and walked directly to it. What is the constant multiple in the big-O bound for your algorithm?

Answer:
$T(n) = 2(1) + 2(2) + 2(3) + ... + 2(n) + n$
$T(n) = 2(1 + 2 + 3 + ... + n) + n$
$= 2(\frac{n(n+1)}{2}) + n$
$= n^2 + 2n + 1$
$= O(n^2)$
$c = 1$

Problem 4. (20 points) Consider the following "proof" that the solution $T(n)$ to the following recurrence relation

$$T(n) = \begin{cases} 1 & n = 1 \\ T(n-1) + n & n > 1 \end{cases}$$

is $O(n)$.

"Proof": *Base case* ($n = 1$)*:* $T(1) = 1$ which is $O(1)$.
Induction step ($n > 1$)*:* Assume that the claim is true for $n' < n$. Consider the recurrence for n, $T(n) = T(n-1) + n$. By induction hypothesis $T(n-1) \in O(n-1)$. Also, $n \in O(n)$. Then, $T(n) \in O((n-1)+n)$ by the properties of the big-Oh. Therefore, $T(n) \in O(n)$, since $O((n-1)+n)$ is $O(n)$.

We know however that this recurrence relation has solution $T(n) \in \Theta(n^2)$ (see slides). What is wrong with this "proof"?

Answer:
The error is in the line $T(n-1) \in O(n-1)$.
With $T(n-1) = O(n)$, that means there is some constant c such that $T(n-1) \leq cn$.
Upon adding n, you get $T(n) \leq (c+1)n$ which changes the constant.
Thus, the constant keeps changing, meaning we don't have one fixed constant c' such that $T(n) \leq c'n$. Instead, we have a sequence of constants.
In conclusion, this is not the definition of big O, and this proof fails.