CS 201

HW2: Computational Complexity

For this assignment, you can once again work with up to one partner, but you must write up all of your answers yourself. You can, however, have high-level discussions with your classmates.

You will submit this assignment electronically via Gradescope. You can either modify this PDF digitally and submit it to Gradescope, or print this out, write your answers on it, and take a clear photo of each page to submit. (Gradescope also has a mobile app, in case you find that more handy.)

1. For each of the following code fragments:
   - indicate how many times the output statement is displayed (the exact number, not an approximation, relative to n)
   - indicate whether that number is better described as $O(n)$ or $O(n^2)$
   - provide a brief but accurate justification of your answer

   a)

```
Kotlin
for (i in 0..<n) {
        for (j in 0..<n) {
                println("$i $j")
                }
        }
```

n times

n times

$n \times n = n^2$

$\Rightarrow O(n^2)$

A ~~nested~~ for loop has the operations inside the for loop multiplied by how many times that loop iterates to find $T(n)$. for a nested for loop, there is $n \times n$ or n^2, which means that it can be represented with $O(n^2)$.

b)

```
Kotlin
for (i in 0..<n) {
    for (j in 0..<2) {
        println("$i $j")
    }
}
```

n times
2 times
$\Rightarrow T(n) = 2n$
$\Rightarrow O(n)$

As we know from explanation in (a), $T(n)$ can be found by counting the number of iterations in the for loops and multiplying. for this nested for loops, it's 2 times every n, or $2 \times n$. as $n \to \infty$, the 2 part doesn't really matter, so it should be written as $O(n)$.

c)

```
Kotlin
for (i in 0..<n) {
    for (j in n-1 downTo 0) {
        println("$i $j")
    }
}
```

n times
n times $(n-1 \to 0)$
$\Rightarrow T(n) = n^2$
$\Rightarrow O(n^2)$

As we knew from the explanation in (b), $T(n)$ can be found by counting the number of iterations in the for loops and multiplying. for these nested for loops, $T(n)$ is n times n, or n^2; which can be represented as $O(n^2)$.

d)

```kotlin
for (i in 0..<n) {
    for (j in 0..<i) {
        if (j % i == 0) {
            println("$i $j")
        }
    }
}
```

n! ?

n times

i times

-1 times

$\Rightarrow T(n) = n-1$ times

$\Rightarrow O(n)$

Stemming off of the explanation in (a), $T(n)$ can be found by counting the number of iterations in the for loops. Here, however, the modules, or remainder operation, evaluating to zero weakly constrains the iteration. Because, only dividing by the exact number leaves zero as a remainder, and 0 divided by 0 doesn't count.

2. Below is a code snippet to match students with advisors:

```kotlin
fun makeAdvisingMatch(students:List<String>, professors:List<String>){
    for (student in students){
        for(professor in professors){
            println("Possible Match: $student & $professor")
        }
    }
}
```

CS 201 HW2: Computational Complexity

a) What is the Big-O run time of this function?

$n \times n$

for n Student & n Professors $T(n) = n^2$

Therefore Big-O = $O(n^2)$

b) Provide a brief but accurate justification for your answer.

because the program is iterating over two (4)s, with one iteration inside the other nested, to find $T(n)$ we add the number of operations inside the for loop, and because there's another for loop, it's n. So $n \times n = n^2$ and that's $O(n) \rightarrow O(n^2)$

3. Reflection: Were there any particular issues or challenges that you dealt with in completing this assignment? How long did you spend on this assignment?

Write a brief discussion (a sentence or two is fine). Also include your collaboration statement here; if you worked alone, say so.

Initially I thought that I was unable to write the code and have the computer compute it, but after that I was able to ensure what I thought was happening was actually happening.

I took 1 hour on this assignment.

(Q1) part d) T(n) is and can only ever be n, or n-1, because of the order of the mod. Because its i mod j, j will always be 1 smaller than i due to the loop increments, the only time when the values are printed is when j is zero because then 0 % i == 0 ✓. But T(n) is not n, its n-1. This is because, in the case of i = 0, its undefined not 0. Therefore instead of n, its n-1, , its undefined not 0.

Addendum to my reflection

I did this work alone.