ECS 36C: Homework #1 - Written part

Joël Porquet, UC Davis

Due before 7:00pm, Tuesday, October 9th, 2018

Name: ______________________ **Student ID:** ______________________

Preamble

Goal The goal of this homework is to further your understanding of computational complexity, as well as the Big-O notation.

Submission This document is to be filled out and uploaded to Gradescope by the due date and time.

You can either print this document, fill it out manually, and scan it back into a digital document; or you can, with a PDF editor, work directly on the digital version.

In any case, you **cannot alter** the document's formatting, as originally published. If your submission doesn't respect the expected formatting, you will be penalized.

Code of Conduct This homework is to be worked **alone**.

Each question will be entirely graded by a single TA so it is easy to notice identical or highly similar answers. Any suspicion of cheating beyond reasonable doubt, will be reported to SJA and might incur academic and disciplinary sanctions.

Question 1 ***(20 pts)*** Find the computational complexity for the following code fragments. Express your answers using the Big-O notation and justify.

```
for(int count = 0, i = 0; i < n; i++)
    for(int j = 0; j < n; j++)
        count++;
```

```
for(int x = 1, count = 0, i = 0; i < n; i++) {
    for(int j = 0; j <= x; j++)
        count++;
    x = 2;
}
```

```
for(int count = 0, i = 0; i < n; i++)
    for(int j = 0; j < i; j++)
        count++;
```

```
for(int count = 0, i = 0; i < n; i++)
    for(int j = 0; j < n * n; j++)
        count++;
```

```
for(int count = 0, i = 0; i < n * n; i++)
    if( i % n == 0)
        for(int j = 0; j < i; j++)
            count++;
```

Question 2 ***(20 pts)*** Let $p(x)$ be a polynomial of degree n, that is, $p(x) = \sum_{i=0}^{n} a_i x^i$.

- Describe a simple $\Theta(N^2)$ time method for computing $p(x)$ (you can write pseudo-C++ code if it helps).

Now consider a rewriting of $p(x)$ as $p(x) = a_0 + x(a_1 + x(a_2 + x(a_3 + ... + x(a_{n-1} + xa_n)...)))$, which is known as *Horner's method*.

- Apply this algorithm to solve $f(x) = 4x^4 + 8x^3 - x + 2$ when $x = 3$.

- Using the big-Oh notation, characterize the number of arithmetic operations this method executes.

Question 3 ***(20 pts)*** Let $f(n) = 3n^2 + 2n + 4$. Use the definition of Big-O to prove that $f(n) = O(N^2)$.

Question 4 ***(20 pts)*** Rank the following time bounds. That is, write them as $f_1, f_2, ..., f_6$ implying that $f_i = O(f_{i+1})$ for all $1 \leq i \leq 5$.

Note that we can determine the relative growth rates of two functions $f(n)$ and $g(n)$ by computing $\lim_{n\to\infty} f(n)/g(n)$. If the result is 0, then $f(n) = O(g(n))$.

It usually helps to use L'Hôpital's rule to perform the limits computations. L'Hôpital's rule states that if $\lim_{n\to\infty} f(n) = \infty$ and $\lim_{n\to\infty} g(n) = \infty$, then $\lim_{n\to\infty} f(n)/g(n) = \lim_{n\to\infty} f'(n)/g'(n)$, where $f'(n)$ and $g'(n)$ are the derivatives of $f(n)$ and $g(n)$, respectively.

- $n^3 + 2n + 1$
- $n \log(n^2)$
- $n^2 \log n$
- 2^n
- 3^n
- $1023n^2 + 2n + 45$

Question 5 ***(20 pts)*** An algorithm takes 0.5 ms for input size 100. How large a problem can be solved in 1 min if the running time is the following (assume low-order terms are negligible)?

1. linear
2. O(N log N)
3. quadratic
4. cubic