# Homework 1: Time complexity

COS 226 – Fall 2020

Assigned: 5 Sep 2020 Due: 11 Sep 2020

In case this is useful:

---

**Multiplication by constant:**
If $f(n)$ is $\mathcal{O}(g(n))$, then $cf(n)$ is, too ($a > 0$)

**Addition:**
If $a(n)$ and $b(n)$ are $\mathcal{O}(f(n))$ and $\mathcal{O}(g(n))$, respectively, then $a(n) + b(n)$ is $\mathcal{O}(f(n) + g(n))$

**Multiplication:**
If $a(n)$ and $b(n)$ are $\mathcal{O}(f(n))$ and $\mathcal{O}(g(n))$, respectively, then $a(n)b(n)$ is $\mathcal{O}(f(n)g(n))$

**Transitivity:**
If $a(n)$ is $\mathcal{O}(f(n))$ and $f(n)$ is $\mathcal{O}(g(n))$, then $a(n)$ is $\mathcal{O}(g(n))$

**Polynomials:**
If $f(n)$ is a polynomial of degree $d$, then it is $\mathcal{O}(n^d)$

**Exponential bound on polynomial:**
$n^x$ is $\mathcal{O}(a^n)$ for any fixed $x > 0$ and $a > 1$

**Log of power:**
$\log n^x$ is $\mathcal{O}(\log n)$ for any fixed $x > 0$

**Power of log:**
$\log^x n$ is $\mathcal{O}(n^y)$ for any fixed $x > 0$ and $y > 0$

---

1. Let $f(n) = (n + 3)(n^2 + 1)$

   (a) Find $g(n)$ such that $f(n)$ is $O(g(n))$.

   $(n+3)\left(n^2+1\right)$

   $n^3+3n^2+n+3$

   ***g* (*n*) =** $n^3+3n^2+n+3$

   (b) What are $c$ and $n_0$ that shows your answer is correct?

   ***c* = 10**
   $\boldsymbol{n_0 = 3}$

2. If $f(n) = n^{1000} + 3n^2$ and $g(n) = 2^n$, is $f(n) \in o(g(n))$? Why or why not?

   **No. *f* (*n*)'s running time is far larger than *g*(*n*) could ever touch.**

3. Suppose $f(n) = (\log^6 n)(\log n^3)$. Show that $f(n)$ is $O(n \log n)$.

   $\log n^3$ is $O(\log n)$ (Log of power identity)
   $\log^6 n$ is $O(n)$ (Power of log identity)

4. Suppose we have the following algorithm:

```
Algorithm Cartesian(A, B, n)

    Input: A and B, two n-element lists
    Output: The Cartesian product of the two lists: [[A[0], B[0]], [A[0], B[1]],
    ...]
    Let C be an empty list
    for i from 0 to n − 1 do
        for j from 0 to n − 1 do
            Add [A[i], B[j]] to the end of C
    return C
End.
```

(a) What is the time complexity using the RAM model (i.e., directly counting operations)? Make sure you explain your answer in terms of the operations you consider primitive.

**Loop i and j - 2 loops. 5 operations being done per iteration - fetch element of A, fetch element of B, add A's element to C, add B's element to C, return C. It's running time is O($n^2$).**

(b) Is this algorithm's running time $O(n^3)$? Why or why not?

**No, it's worst case will be O($n^2$).**

(c) Is this algorithm's running time $\Theta(n^2)$? Why or why not?

**Yes. The running time will be between $n^2 * c_1$ and $n^2 * c_2$ where both constants are the running times of the primitive operations.**

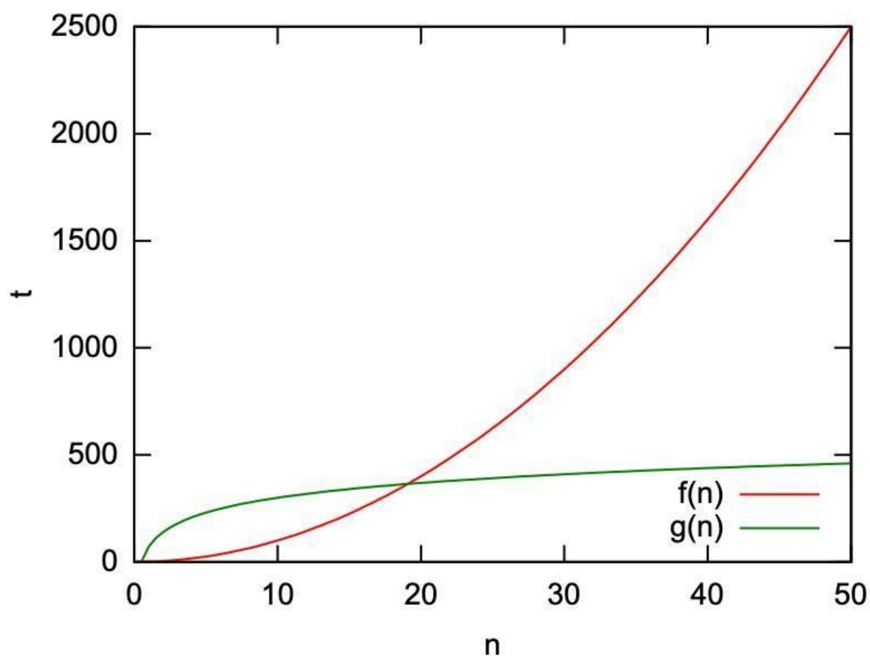
(d) Is this algorithm's running time $\Omega(n)$? Why or why not?

**Yes. The algorithm's running time will be *at least* $\Omega(n)$.**

5. Suppose that we add a loop between lines 7 and 8 of the algorithm in question 1 that prints the list, one element at a time. Which of the answers you gave in question 4 would be different now? Why?

**Part B will now be different. The algorithm's running time would change to O($n^3$).**

6. Given the graph below:




What can you say about the relationship between $f(n)$ and $g(n)$? Make sure you reference $c$ and $n_0$ in your answers.

**$f(n)$ is $O(g(n))$ until $n$ reaches about 19**
**$g(n)$ is $\Omega(f(n))$ until $n$ reaches about 19**

**Without the graph, I would assume that their time complexities are roughly the same. I would have concluded that $f(n)$ is $O(g(n))$.**