Homework 1

COMP221 Spring 2024 - Suhas Arehalli

Complete the problems below. Note that point values are roughly inversely correlated with expected difficulty.

- Write up your solutions to all parts of this homework so that they may be exported to a typeset pdf(LaTeX/Overleaf is recommended, but any typeset PDF is acceptable).
- You'll submit this assignment through Moodle.
- **The due date for this assignment will be posted on the course website.**
- If you discussed problems with other students, list their names at the top of your assignment (there will be no penalty for this — it's encouraged!).
- However, don't look at other group's written solutions, or share your written solutions with other groups. Consult the syllabus for more details on academic integrity policies! You should make sure your groups are sharing approaches to a problem, not full solutions

Problems

1. **Ignoring coefficients** (*15pts*): In COMP128, we learned that we can drop constant coefficients in time complexities — all we care about is what we need for Big-O notation. A version of this claim can be written formally as the following: if $f(n) \in \Theta(g(n))$, $kf(n) \in \Theta(g(n))$ for $k > 0$. Prove this to be correct using the c, n_0 definition of big-Θ!

   HINT: *Try and generalize from more concrete examples. Is $3n \in O(n)$? Is $3n^2 \in O(n^2)$? What about $9n^2$? Can you see a way to construct the c and n_0 to prove $9n^2 \in O(n^2)$ from the c' and n_0' that show $3n^2 \in O(n^2)$?*

2. **Counting for Quadratics** (*10pts*) In class we talked about how we developed all of our Big-O formalisms so we could be adequately lazy in talking about time complexity. In class, I was often actively sloppy when counting simple operations when proving the time complexities of our n^2 sorts. In these kinds of nested-loop algorithms, as long as we have a constant number of simple operations within our 2 loops, we get a time complexity of the form

   $$f(n) = a_1 n^2 + a_2 n + a_3$$

   where $a_1, a_2, a_3 \in \mathbb{Z}$ (i.e., they're integers) and $a_1 > 0$, It's a quadratic function with a positive n^2 term! My claim is that if I'm sloppy with counting operations, this will affect the values of a_1, a_2, and a_3, but won't result in a time complexity growth function of a different form (convince yourself this is true!).

Prove that for *any* integers $a_1 > 0, a_2, a_3$ that $f(n) \in \Theta(n^2)$ using the c, n_0 definition of big-O and big-Ω. You may use either definition of big-Θ we discussed. If helpful, you may assume that $f(n) > 0$ for all $n > 0$.

HINT: *Try and get the quadratic to be something easily factorable. Try by cases: If $a_3 = 0$, if $a_3 > 0$, or if $a_3 < 0$. Use the tricks we found in class.*

3. **Polynomials of Higher Degree** (*5pts*) Of course, the above only shows that this works for quadratic time algorithms. Let's extend it to all polynomials: Show that if

$$\begin{aligned} f(n) &= \sum_{i=0}^{k} a_i n^i \\ &= a_k n^i + \cdots + a_1 n + a_0 \end{aligned}$$

for some $a_i \in \mathbb{Z}$, then $f(n) \in O(n^k)$. Use the c, n_0 definition of big-O. Note I'm only asking for big-0 here, not big-Θ!

HINT: We'll need a slightly different strategy (one more general than the small bounding trick I suggested for quadratics). Consider $c = \sum_{i=0}^{n} |a_i|$. Can you see a way to show $cn^k \geq f(n)$ for all $n \geq$ some n_0?