CS109A Notes for Lecture 1/26/96

Running Time

A program or algorithm has a running time $T(n)$, where n is the measure of the size of the input.

- $T(n)$ is the largest amount of time the program takes on any input of size n.

Example: For a sorting algorithm, we normally choose n to be the number of elements to be sorted. For Mergesort, $T(n) = n \log n$; for Selection-sort or Quicksort, $T(n) = n^2$.

- But there is an unknowable constant factor that depends on various factors, such as machine speed, quality of the compiler, load on the machine.

Why Measure Running Time?

- Guides our selection of an algorithm to implement.
- Helps us explore for better solutions without expensive implementation, test, and measurement.

Arguments Against Running-Time Measurement

- Algorithms often perform much better on average than the running time implies (e.g., quicksort is $n \log n$ on a "random" list but n^2 in the worst case, where each division separates out only 1 element).
  - ☐ But for most algorithms, the worst case is a good predictor of typical behavior.
  - ☐ When average and worst cases are radically different, we can do an average-case analysis.
- Who cares? In a few years machines will be so fast that even bad algorithms will be fast.

- The faster computers get, the more we find to do with them and the larger the size of the problems we try to solve.
- Asymptotic behavior (growth rate) of the running time becomes *more* important, not less, because we are getting closer to the asymptote.

- Constant factors hidden by "big-oh" are more important than the growth rate of running time.
    - Only for small instances, and anything is OK when your input is small.
- Benchmarking (running program on a popular set of test cases) is easier.
    - Sometimes true, but you've committed yourself to an implementation already.

Big-Oh

- A notation to let us ignore the unknowable constant factors and focus on growth rate of the running time.

Say $T(n)$ is $O\big(f(n)\big)$ if for "large" n, $T(n)$ is no more than proportional to $f(n)$.

- More formally: there exist constants n_0 and $c > 0$ such that for all $n \geq n_0$ we have $T(n) \leq cf(n)$.
- n_0 and c are called *witnesses* to the fact that $T(n)$ is $O\big(f(n)\big)$.

Example: $10n^2 + 50n + 100$ is $O(n^2)$. Pick witnesses $n_0 = 1$ and $c = 160$. Then for any $n \geq 1$, $10n^2 + 50n + 100 \leq 160n^2$.

- Other choices of witness are possible, e.g., $(n_0 = 10,\ c = 16)$.
- General rule: any polynomial is big-oh of its leading term with coefficient of 1.

Example: n^{10} is $O(2^n)$.

- Note that n^{10} can be *very* large compared to 2^n for "small" n.
  - □ $n^{10} < 2^n$ is the same as saying $10 \log_2 n \leq n$. (False for $n = 32$; true for $n = 64$.)
- Pick witnesses $n_0 = 64$ and $c = 1$. For $n \geq 64$ we have $n^{10} \leq 1 \times 2^n$.

Growth Rates of Common Functions

- The base of a logarithm doesn't matter. $\log_a n$ is $O(\log_b n)$ for any bases a and b because $\log_a n = \log_a b \log_b n$ (i.e., witnesses are $n_0 = 1, c = \log_a b$).
  - □ Thus, we omit the base when talking about big-oh.
- Logarithms grow slower than any power of n, e.g. $\log n$ is $O(n^{1/10})$.
- An *exponential* is c^n for some constant $c > 1$.
- Polynomials grow slower than any exponential, e.g. n^{10} is $O(1.001^n)$.
- Generally, exponential running times are impossibly slow; polynomial running times are tolerable.

Proofs That a Big-oh Relationship is False

Example: n^3 is not $O(n^2)$. In proof: suppose it were. Then there would be witnesses n_0 and c such that for all $n \geq n_0$ we have $n^3 \leq cn^2$.

Choose n_1 to be

1. At least as large as n_0.
2. At least as large as $2c$.

- $n^3 \leq cn^2$ holds for $n = n_1$, because $n_1 \geq n_0$ by (1).
- If $n_1^3 \leq cn_1^2$, then $n_1 \leq c$.
- But by (2), $n_1 \geq 2c$.

- Since $c > 0$ (holds for any witness c), it is not possible that $2c \leq n_1 \leq c$.
- Thus, our assumption that we could find witnesses n_0 and c was wrong, and we conclude n^3 is *not* $O(n^2)$.

General Idea of Non-Big-Oh Proofs

- Template p. 101 of FCS.

1. Assume witnesses n_0 and c exist.
2. Select n_1 in terms of n_0 and c.
3. Show that $n_1 \geq n_0$, so the inequality $T(n) \leq cf(n)$ must hold for $n = n_1$.
4. Show that for the particular n_1 chosen, $T(n_1) > cf(n_1)$.
5. Conclude from (3) and (4) that n_0 and c are not really witnesses. Since we assumed nothing special about witnesses n_0 and c, we conclude that no witnesses exist, and therefore the big-oh relationship does not hold.