CS Theory (Spring '25) February 18, 2025

Lecture Note: Asymptotic Notation

Instructor: *Josh Alman*

We have been studying different models of computation, including DFAs, NFAs and regular expressions. So far we focus on whether the model can solve a problem, and we haven't talked about how efficient the model can solve the problem. For example, we have seen the construction of a DFA from an NFA. Although they recognize the same language, it has a huge (exponential) blow-up in the number of states. Starting now, we will not only focus on whether a computation model can, but also how efficient it can solve a problem.

1 Big-O Notation

The exact running time of an algorithm is often complicated, but we usually only care about an estimate.

We use big-O notation to capture estimates. It helps us to understand how a function behaves on large inputs. We usually care more about large inputs, because even an inefficient algorithm can be fast on small inputs.

Example 1. $f(n) \coloneqq 6n^3 + 2n^2 + 20n + 45 = O(n^3)$.

For polynomials, we consider only the highest order term and disregard coefficients to get an estimation in big-O form.

Below we give a general definition for big-O notation.

Definition 2. Let $f, g : \mathbb{N} \to \mathbb{R}^+$ (positive integers to positive reals) be functions. We say $f(n) = O(g(n))$ if there exist constants $c, n_0 > 0$, such that for every $n \geq n_0$, we have $f(n) \leq c \cdot g(n)$.

In the definition, we have the coefficient c so that we don't need to care about the coefficient, and we have $n \geq n_0$ so that we only need to care about big enough inputs.

Recall Example 1. Let $f(n) = 6n^3 + 2n^2 + 20n + 45$, $g(n) = n^3$. We can choose $c = 9$ and $n_0 = 45$ to fit in the above definition.[1] If $n \geq 45$, then $f(n) = 6n^3 + 2n^2 + 20n + 45$. Since $2 \leq n$, $20 \leq n$ and $45 \leq n$, we have

$$f(n) \leq 6n^3 + n \cdot n^2 + n \cdot n + n = 7n^3 + n^2 + n \leq 7n^3 + n^3 + n^3 = 9n^3.$$

Example 3. $f(n) \coloneqq 6n^3 + 2n^2 + 20n + 45 = O(n^4)$. *This is true because n^4 is even bigger than n^3.*

Example 4. $f(n) \coloneqq 6n^3 + 2n^2 + 20n + 45 \neq O(n^2)$, *because $6n^3$ grows much faster than n^2 does. Formally, no matter what n_0 and c we pick, there is always an $n \geq n_0$ with $f(n) > c \cdot n^2$. For example, if we choose $n = 6c + 1$, then*

$$f(n) \geq 6n^3$$

[1] Although $c = 7$ should work, picking a larger c makes the proof easier. There are a lot of possible choices of c and n_0.

Example 5. *$f(n) = \log_2(n)$, $g(n) = \log_e(n)$. Then, $f(n) = O(g(n))$, because*

$$\log_2(n) = \frac{\log_e(n)}{\log_e(2)}.$$

For this reason, in big-O notation we usually don't care about the base of logarithm, and hence the notation $O(\log(n))$ (where the base is not specified).

Example 6. $\log(n) \neq O(\log_{\sqrt{n}}(n))$. *Note that* $\log_{\sqrt{n}}(n) = 2$.

Example 7. $\log_2(n) \neq O(\log_e \log_e(n))$. *If we substitute $k = \log_e(n)$, then this becomes $k \neq O(\log k)$.*

Example 8. $n \cdot \log\log n = O(n \cdot \log n)$. *We know* $\log\log n = O(\log n)$ *and we multiply both sides by n.*

Example 9. $\log n \neq O((\log\log n)^{1000})$. *If we substitute $k = \log n \log n$, this becomes $2^k \neq O(k^{1000})$.*

Example 10. $\log n = O((\log\log n)^n)$. *This is because* $\log n = O(2^n) = O((\log\log n)^n)$.

Example 11. $3^n \neq O(2^n)$. *Actually,* $(2^n)^{\log_2 3} \approx (2^n)^{1.585}$. *Another way to see this is* $3^n/2^n = 1.5^n \to \infty$ *as $n \to \infty$.*

Below we use big-O notation for arithmetic expressions.

Example 12. $\sum_{i=1}^n i = \frac{n(n+1)}{2} = \frac{1}{2}n^2 + \frac{1}{2}n = O(n^2)$.

Example 13. $\sum_{i=1}^n i \leq \sum_{i=1}^n n = n^2 \leq O(n^2)$.

If we actually care about the leading constant and want a bound on the remaining terms, we can also use big-O notation, in a way similar to the following.

Example 14. $\sum_{i=1}^n i = \frac{1}{2}n^2 + \frac{1}{2}n = \frac{1}{2}n^2 + O(n)$.

Example 15. $\sum_{i=1}^n 2^i = \sum_{i=1}^n O(2^n) \leq O(n \cdot 2^n)$. *But this is not tight. Actually,* $\sum_{i=1}^n 2^i = 2^{n+1} - 2 = O(2^n)$.

Example 16. $3^n = 2^{O(n)}$. *The reason is* $3^n = 2^{\log_2 3 \cdot n} = 2^{O(n)}$. *$O(n)$ here is replacing an "anonymous function" that is bounded by $O(n)$. Compare to Example 11.*

Example 17. $n^3 = 2^{O(\log n)}$. *In fact,* $n^3 = 2^{3\log_2 n}$.

$2^{O(\log n)}$ captures all polynomials in n (with positive leading coefficient). For example, $2^{100 \cdot \log_2 n} = n^{100}$, $2^{10000000 \cdot \log_2 n} = n^{10000000}$. Similarly, $n^{O(1)}$ also captures all polynomials in n. We use the notation $\text{poly}(n)$ for all polynomials in n.

Definition 18. $\text{poly}(n) \coloneqq n^{O(1)}$

Thus, $f(n) = \text{poly}(n)$ if and only if $f(n) = n^{O(1)}$, which is equivalent to $f(n) = 2^{O(\log n)}$.

Example 19. $n \cdot \log_2(n) = n^{O(1)}$. *This is because $n \cdot \log(n) = O(n^2)$ is a polynomial in n.*

Example 20. $(\log n)^{100} = \text{poly}(n)$. *Actually* $(\log n)^{100} = O(n)$.

Example 21. $(\log n)^{\log\log(n)} = \text{poly}(n)$. *Note that* $\log n = 2^{\log\log n}$, *so* $(\log n)^{\log\log n} = 2^{(\log\log n)^2} = 2^{O(\log n)} = \text{poly}(n)$.

Example 22. $\binom{n}{4} = \text{poly}(n)$. *This is because* $\binom{n}{4} = \frac{n(n-1)(n-2)(n-3)}{24} = O(n^4)$.

Example 23. *It is not true that for all* $0 \le k \le n$, $\binom{n}{k} \le \text{poly}(n)$. *Although* $\binom{n}{k} = O(n^k)$, k *may depend on* n *and is not necessarily a constant here. When* $k = n/2$,

$$\binom{n}{n/2} = \frac{n!}{((n/2)!)^2} \approx 2^n \cdot \sqrt{\frac{2}{\pi n}}.$$

However, if k *is a fixed constant, then* $\binom{n}{k}$ *is a polynomial in* n.

Here we formally prove it is false that $\binom{n}{k} \le \text{poly}(n)$ for all $0 \le k \le n$.

Proof. Assume to the contrary that $\binom{n}{k} \le \text{poly}(n)$ for all k. Then,

$$\sum_{k=0}^{n} \binom{n}{k} \le \sum_{k=0}^{n} \text{poly}(n) \le n \cdot \text{poly}(n) \le \text{poly}(n).$$

On the other hand, $\sum_{k=0}^{n} \binom{n}{k} = 2^n$.[2] This is not a poly$(n)$, so there is a contradiction. □

2 Big-Θ notation

Definition 24. Let $f, g : \mathbb{N} \to \mathbb{R}^+$ be functions. We say $f(n) = \Theta(g(n))$ if $f(n) = O(g(n))$ and $g(n) = O(f(n))$.

Roughly speaking, using Θ means "we have found the best big-O bound".

Example 25. $f(n) \coloneqq 6n^3 + 2n^2 + 20n + 45$, $g(n) = n^3$, *then* $f(n) = \Theta(g(n))$.
However, if $h(n) = n^4$, *then* $f(n) \neq \Theta(h(n))$, *although it is true that* $f(n) = O(h(n))$.

Example 26. $\log_2(n) = \Theta(\log_e(n))$.

Example 27. $\sum_{i=1}^{n} i = \Theta(n^2)$.

Example 28. $3^n = 2^{\Theta(n)}$.

Example 29. $n^3 = 2^{\Theta(\log n)} = n^{\Theta(1)}$. *In general,* $\text{poly}(n) = n^{\Theta(1)}$.

3 Additional notations (not required material)

There are some other similar notations used in computer science.

Definition 30 (Small-o notation)**.** Let $f, g : \mathbb{N} \to \mathbb{R}^+$ be functions. We say $f(n) = o(g(n))$ if for all $c > 0$, there exists $n_0 > 0$ such that for all $n \ge n_0$, $f(n) < c \cdot g(n)$.

Example 31. $f(n) \coloneqq 6n^3 + 2n^2 + 20n + 45$, *then* $f(n) = o(n^4)$ *but* $f(n) \neq o(n^3)$. *We can see the latter by choosing* $c = 1$ *in the definition.*

[2] This can be proved by using two ways to count the number of subsets of a size-n set. On one hand, for each element, there are two possible choices: to be in the subset or not in the subset, so the total number of subsets is 2^n. On the other hand, the number of subsets of size k is $\binom{n}{k}$, so the total number is $\sum_{k=0}^{n} \binom{n}{k}$.

Example 32. *$f(n) \coloneqq 6n^3 + 2n^2 + 20n + 45$, then $f(n) = 6n^3 + o(n^3) = 6n^3(1 + o(1))$.*

Example 33. *$n^2/\log n = o(n^2)$.*

If $f(n) = o(g(n))$, then it always holds that $f(n) = O(g(n))$, and it cannot hold that $f(n) = \Theta(g(n))$. However, it is not true that if $f(n) = O(g(n))$, then either $f(n) = o(g(n))$ or $f(n) = \Theta(g(n))$.

Example 34. *If*

$$f(n) = \begin{cases} n^3 & n \text{ is even} \\ n^4 & n \text{ is odd} \end{cases},$$

then $f(n) = O(n^4)$, but $f(n) \neq o(n^4)$ and $f(n) \neq \Theta(n^4)$.

Example 35. *$\log n = o(n)$.*

Definition 36 (Big-Ω notation)**.** Let $f, g : \mathbb{N} \to \mathbb{R}^+$ be functions. We say $f(n) = \Omega(g(n))$ if $g(n) = O(f(n))$.

Definition 37 (Small-ω notation)**.** Let $f, g : \mathbb{N} \to \mathbb{R}^+$ be functions. We say $f(n) = \Omega(g(n))$ if $g(n) = o(f(n))$.

Ω and ω are similar to O and o, except that they are for lower bounds instead of upper bounds.

Example 38. *$f(n) \coloneqq 6n^3 + 2n^2 + 20n + 45$, then $f(n) = \Omega(n^3)$, $f(n) = \Omega(n^2)$, $f(n) = \omega(n^2)$, $f(n) \neq \omega(n^3)$.*