Math 2300: Discrete Mathematics

Algorithms 1 - Definition of Big O Textbook Section

Terms, Concepts, and Examples

Big O notation is a standard way mathematicians and computer scientists use to describe how much time and how much memory is required for an algorithm to run.

- Let f and g be functions from the set of integers to the set of real numbers. We say that $f(x)$ is $O(g(x))$ if there exists positive integers A and n such that

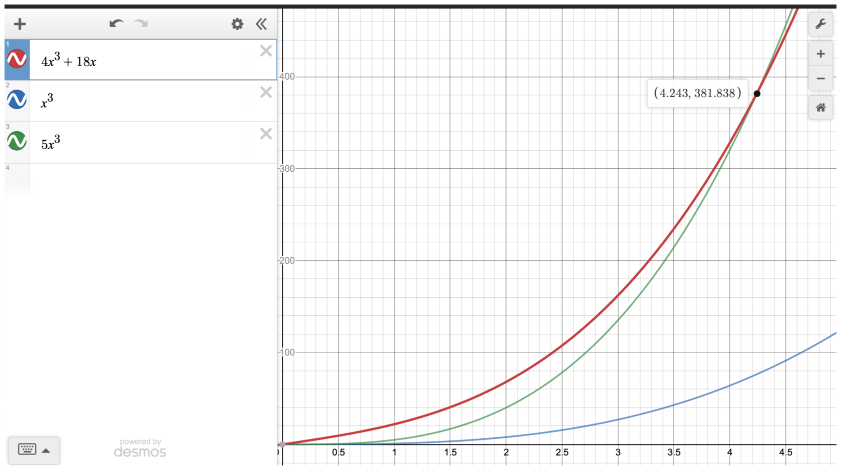
  $$|f(x)| \leq A|g(x)|$$

  whenever $x > n$. This is read as "$f(x)$ is **big-oh** of $g(x)$."

  To determine if a function $f(x)$ is $O(g(x))$, amounts to identifying the, positive constants A and n. That is, we must find the factor A and the point n for which $f(x) \leq Ag(x)$ whenever $x > n$.

  Example Show that $f(x) = 3x^3 + 18x$ is $O(x^3)$ with $A = 4$ and $n = 5$.

  Solution: Notice that $x^3 > 18x$ when $x \geq 5$. This means that $3x^3 + x^3 > 4x^3 + 18x$ when $x > 5$. In other words, $4x^3 > 3x^3 + 18x$ whenever $x > 5$, confirming $A = 4$ and $n = 5$.



  Video Example of Big O Definition

- Important Facts about Big O

  - $n!$ is $O(n^n)$.
  - $\log n!$ is $O(n \log n)$.
  - $\log n$ is $O(n)$.
  - If $f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$ is a polynomial of degree n where $a_0, a_1, \ldots, a_{n-1}, a_n$ are real numbers. Then $f(x)$ is $O(x^n)$.
  - n^c is $O(n^d)$ as long as $1 < c < d$.
  - n^d is $O(b^n)$ for any d positive and $b > 1$.
  - b^n is $O(c^n)$ as long as $1 < b < c$.

- Suppose that $f_1(x)$ is $O(g_1(x))$ and that $f_2(x)$ is $O(g_2(x))$. Then $(f_1 + f_2)(x)$ is $O(\max(|g_1(x)|, |g_2(x)|))$. That is, Big O of the sum of two functions is Big O of the "larger" function.
- Suppose that $f_1(x)$ is $O(g_1(x))$ and that $f_2(x)$ is $O(g_2(x))$. Then $(f_1 f_2)(x)$ is $O(g_1(x)g_2(x))$. That is, Big O of the product of two functions is the product of Big O of the functions.

Example Give a Big-O estimate for $f(n) = 4n \log(n!) + (n^3 - 2)\log n$, where n is a positive integer.

Solution: The product $4n\log(n!)$ is $O(n^2 \log n)$. To see this recall, $4n$ is $O(n)$ from the polynomial rule and $\log n!$ is $O(n \log n)$. Then using the product, $4n\log(n!)$ is $O(n * n\log n) = O(n^2 \log n)$.

Also $(n^3 - 2)\log n$ is $O(n^4)$. To see this, recall $n^3 - 2$ is $O(n^3)$ from the polynomial rule and $\log n$ is $O(n)$. Then using the product again, $(n^3 - 2)\log n$ is $O(n^3 * n) = O(n^4)$.

Using the sum rule and the fact $n^4 > n^2 \log n$ (so the max of the two is n), the function $f(n) = 4n\log(n!) + (n^3 - 2)\log n$ is $O(n^4)$.

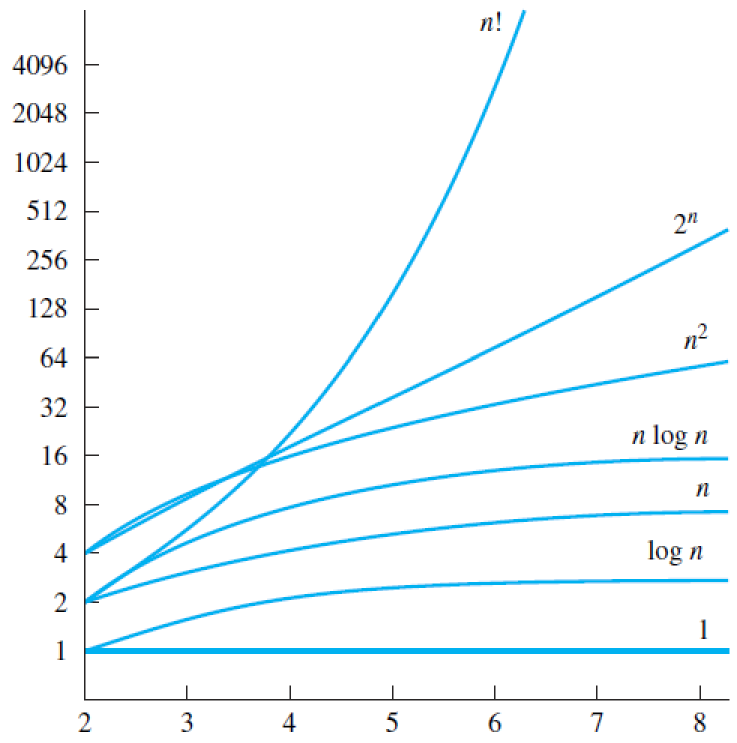
Video Example of Using the Facts

Another Video Example of Using the Facts

- Recall that Big O is used for estimating the number of operations needed to run an algorithm. Here are functions commonly found in these estimates:

$$1, \log n, n, n\log n, n^2, 2^n, n!$$

This list is in "order," in that each function in the list is smaller than the next. We can see this in the graph, which has a log scale.



Video Example of Ordering Functions

Practice Problems

1. Use the definition of "$f(x)$ is $O(g(x))$" to show that $x^4 + 8x^3 - 3x + 5$ is $O(x^4)$.
2. Determine whether each of these functions is $O(x^2)$.
   (a) $f(x) = 16x + 9$
   (b) $f(x) = x^2 + 500$
   (c) $f(x) = x \log x$
   (d) $f(x) = x^4/2$
   (e) $f(x) = 2^x$
3. Give as good a Big O estimate as possible for each of these functions.
   (a) $(n^2 + 7)(x + 3)$
   (b) $(n \log n + n^2)(n^3 + 2)$
   (c) $(n^3 + n^2 \log n)(\log n + 1)$
   (d) $(2^n + n^2)(n! + 5^n)$