Analysis of Algorithms

September 12, 2023 9:22 AM

To determine the cost of an algorithm let $T(n)$ be the number of ______ operations that the algorithm does for an input of size n. If $T(n) = 3n^2+2n+5$ we say the algorithm is quadratic in n.

For large n the term $3n^2$ dominates the cost of the alg.

Definition. Let $n \in \mathbb{N}$ and $f: \mathbb{N} \to \mathbb{R}$ and $g: \mathbb{N} \to \mathbb{R}$. We say g <u>dominates</u> f if $\exists\, c > 0$ and $\exists\, k \in \mathbb{N}$ s.t.

$$|f(n)| \le c\,|g(n)| \text{ for } n \ge k.$$

Define $O(g(n)) = \{ f(n) : g(n) \text{ dominates } f(n) \}$.

Examples $\quad g(n) \qquad\qquad |f(n)| \le c\,|g(n)| \ \forall n \ge k.$

$2n^2+5 \in O(n^2)$ because $2n^2+5 \le \boxed{7} \cdot n^2 \ \forall n \ge \boxed{1}$

$\to 3n+1 \in O(n^2)$ because $3n+1 \le \boxed{4}\, n^2 \ \forall n \ge \boxed{1}$

$n^3 \notin O(n^2)$ because $n^3 \not\le c \cdot n^2 \ \forall n > c$

So $O(n^2) = \{ an^2+bn+c,\ an+b,\ an^{1.5}+bn+c,\ a\,n\log n, \dots \}$

$$O(n) \subset O(n^2) \subset O(n^3)$$

Linear algs. Quadratic algs. Cubic algs.

$f, g \in \mathbb{Q}[x]$

$\deg f = \deg g = n$

$T(n) = \#\ +,-,\times,\div$ operations.

Let $\alpha \in \mathbb{Q}$

$f \pm g \qquad f \cdot g \qquad A \cdot B$ $n \times n$ matrices.

$f(\alpha) \qquad f \div g \qquad A \cdot x = b$

$\alpha f \qquad \gcd(f,g). \qquad$ Gaussian elim.

$f'(x)$

How do we show $\underline{O(f(n))} = \underline{O(g(n))}$? $\qquad O(3n^2+11) = O(n^2+2n)$?

We show $\boxed{O(f(n)) \subset O(g(n))}$ and $O(g(n)) \subset O(f(n))$.

$h(n)$

I claim if $f(n) \in O(g(n))$ then $O(f(n)) \subset O(g(n))$.

Proof. Suppose $f(n) \in O(g(n)) \Rightarrow |f(n)| \le c_1 |g(n)| \ \forall n \ge k_1$

Let $h(n) \in O(f(n)) \Rightarrow |h(n)| \le c_2 |f(n)| \ \forall n \ge k_2$

Proof. Suppose $f(n) \in O(g(n)) \Rightarrow |f(n)| \leq c_1 |g(n)| \ \forall n \geq k_1$

Let $h(n) \in O(f(n)) \Rightarrow |h(n)| \leq c_2 |f(n)| \ \forall n \geq k_2.$

$\Rightarrow |h(n)| \leq c_2 \left[c_1 |g(n)| \right] \ \forall n \geq \max(k_1, k_2).$

$\Rightarrow h(n) \in O(g(n)).$ $\quad$ true $c_1, c_2 \geq 1.$

Take $c \geq \max(c_2 c_1, 1).$

Exercise. For $a > 1$ and $b > 1$ show $O(\log_a n) = O(\log_b n)$.
$= O(\log n)$

Hint. $\log_a n = \dfrac{\ln n}{\ln a}.$

Properties of $O(g(n))$.

① $O(c \cdot f(n)) = O(f(n))$ for any constant $c > 0$.

E.g. $O(3n^2) = O(n^2)$. Proof Exercise.

② If $f(n) \in O(g(n))$ then $O(|f(n)| + |g(n)|) = O(g(n))$.

E.g. $O(\underbrace{3n^2}_{g(n)} + \underbrace{2n+11}_{f(n)}) = O(3n^2) = O(n^2).$

③ $f(n) \cdot O(g(n)) = O(f(n) \cdot g(n)).$

E.g. $2n \, O(n) = O(2n^2) = O(n^2).$

Often we want to add $O(f(n)) + O(g(n))$. E.g.

Algorithm foo

Step 1 ...... $\leq 2n\log_2 n + n \ \in O(n \log_2 n)$

Step 2 ...... $= 3n^2 + 2n - 1 \ \in O(n^2)$

Step 3 ..... $= 2n + 5 \ \in O(n).$

$$O(n \log_2 n) + O(n^2) + O(n) = O(n \log_2 n + n^2 + n)$$
$$= O(n^2).$$

Define $O(f(n)) + O(g(n)) = O(|f(n)| + |g(n)|).$

E.g. $O(1 + n^2) + O(n^2) = O(2n^2 + 1) = O(n^2).$

Example. Suppose $f, g \in \mathbb{Z}[x]$ of degree $n \geq 0$.

Suppose we want to compute $h(x) = f(x) \cdot g(x) \bmod x^{n+1}$

E.g. $(1 + 2x)(2 + 3x) \bmod x^2 = 2 + 4x + 3x$
$n = 1$ $\qquad = 2 + 7x.$

E.g. $(1+2x)(2+3x) \bmod x^2 = 2+4x+3x$
$\underline{n=1}$ $= 2+7x.$

Let $f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$
$g(x) = b_0 + b_1 x + b_2 x^2 + \cdots + b_n x^n$
$h(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_n x^n$

$$\begin{aligned}
c_0 &= a_0 b_0 \\
c_1 &= a_0 b_1 + a_1 b_0 \\
c_2 &= a_0 b_2 + a_1 b_1 + a_2 b_0 \\
&\vdots \\
c_{\underline{n}} &= a_0 b_n + a_1 b_{n-1} + \cdots + a_n b_0
\end{aligned}$$

Algorithm SeriesMult.
for $k = 0, 1, \ldots, n$ do
    $c_k = 0$
    for $i = 0, 1, \ldots, k$ do
        $c_k = c_k + a_i \cdot b_{k-i}$

$2(k+1)$ arith ops. $\in O(k)$.

Let $T(n)$ be the # of arithmetic operations in $\mathbb{Z}$.

$$T(n) = \sum_{k=0}^{n} 2(k+1) = 2\left(\sum_{k=0}^{n} k+1\right) = 2\left[\sum_{k=0}^{n} k + \sum_{k=0}^{n} 1\right]$$
$$= 2\,\frac{n\cdot(n+1)}{2} + n+1$$
$$= (n+1)(n+1) \in O(n^2).$$

$$T(n) = \sum_{k=0}^{n} O(k)$$
$$= O(0) + O(1) + \cdots + O(n)$$

By Def.
$$= O(0 + 1 + \cdots + n)$$
$$= O\left(\frac{n(n+1)}{2} = \frac{1}{2}n^2 + \frac{1}{2}n\right) = O\left(\frac{1}{2}n^2\right) = O(n^2).$$

Def. $\boxed{O(f_1(n)) + O(f_2(n)) + \cdots + O(f_m(n)) = O(|f_1(n)| + |f_2(n)| + \cdots + |f_m(n)|).}$

Question: Is the time complexity of algorithm SeriesMult $O(n^2)$? No. $f \in \mathbb{Z}[x]$. The time will also depend on the cost of the integer $+, \times$.

There are two main types of complexity.

Algebraic complexity: in $R[x]$ we count # ring ops in R: $+, -, \times$.

Bit complexity: count total # of bit operations.
(proportional to the time of the algorithm).

The polynomial evaluation problem.

Let $f \in \mathbb{Q}[x]$, $f \neq 0$, $n = \deg(f)$.

Let $\alpha \in \mathbb{Q}$. How can we compute $f(\alpha)$?

$$f = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n.$$
$$f = a_0 + a_1 \alpha + a_2 \alpha^2 + a\ \alpha^3 + \cdots a_n \alpha^n.$$

```
Pow(α, n)
  y = 1;
  for i = 1,2,3,...,n do
    y = α·y
  return y.
```

does n mults.

```
Eval1(f, α)
  y = a0
  for i = 1,2,...,n do
    y = ai · Pow(α, i) + y
  return y;
```

$1+i$

Let $T(n)$ be the # of mults in $\mathbb{Q}$ done.

$$T(n) = \sum_{i=1}^{n} (1+i) = \sum_{i=1}^{n} 1 + \sum_{i=1}^{n} i = n + \frac{n(n+1)}{2} = \frac{n(n+3)}{2}$$
$$= \frac{n^2}{2} + \frac{3}{2} n \in O(n^2).$$

```
Eval2(f, α).
  y = a0
  t = 1
  for i = 1,2,...,n do
    t = α·t
    y = y + ai·t
  return y.
```

$T(n) = 2 \cdot n \in O(n)$.

Algorithm Eval1 is $O(n^2)$ whereas alg. Eval2 is $O(n)$.

Horner's algorithm.

Horner(f, α) // $f = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} + a_n x^n$

Horner(f, α) // $f = a_0 + a_1 x + a_2 x + \cdots + a_{n-1} x^{n-1} + a_n x^n$

// $= a_0 + x(a_1 + x(a_2 + \cdots x(a_{n-1} + x(a_n)) \cdots))$

Start here

$y = a_n$

for $i = n-1, n-2, \ldots, 1, 0$ do

$y = \alpha \cdot y + a_i$

return y.

$T(n) = n. \in O(n).$