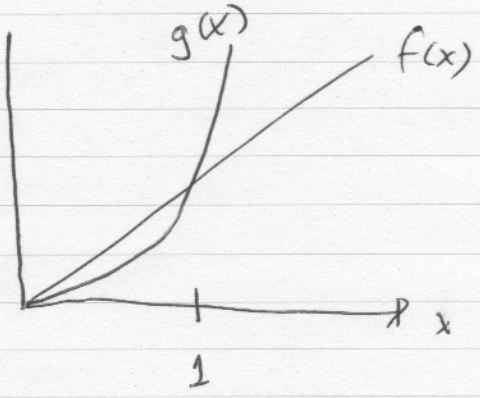
computational complexity of algorithms

- "Big O" notation

$f(x) = O(g(x))$
$f(x) \in O(g(x))$
} mean $|f(x)| \leq c|g(x)|$ for $x > k$.

c and k are "witnesses" (they are not unique).

e.g. $f(x) = x$, $g(x) = x^2$



$g(x) > f(x)$ for all $x > 1$.

$f(x) = O(g(x))$
with witnesses $c = 1$, $k = 1$.

witnesses not unique; also valid; $k = 100$, $c = 100$.

common ref funcs for $g(x)$

a) polynomials
b) summations
c) Factorials
d) log of factorials.

a) Big O for general polynomials.

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

$$\leq |a_n| x^n + |a_{n-1}| x^{n-1} + \cdots + |a_1| x + |a_0|$$

$$\leq |a_n| x^n + |a_{n-1}| x^n + \cdots + |a_1| x^n + |a_0| x^n$$

↑ for $x \geq 1$

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

$f(x) \leq cx^n$ where $c = (|a_n| + |a_{n-1}| + \cdots + |a_1| + |a_0|)$

$$\boxed{f(x) = O(x^n)} \text{ with } c = \sum_{i=0}^{n} |a_i|, \quad k = 1.$$

b) summation

$$f(n) = \sum_{i=1}^{n} i = 1 + 2 + 3 + \cdots + (n-1) + n = \frac{n(n+1)}{2} \leq cn^2$$

where $c = 1$ and $k = 1$.

$$\boxed{f(n) = O(n^2)}$$

c) Factorial

$$f(n) = n \cdot (n-1)(n-2) \cdots 2 \cdot 1$$

$$\leq n \cdot n \cdot n \cdot n \cdots n = n^n$$

$f(n) = O(n^n)$ where $c = 1$, $k = 1$.

d) log of a factorial

$$h(n) = \log n!$$

$$= \log [n \cdot (n-1) \cdot (n-2) \cdots 2 \cdot 1]$$

$$< \log (n^n) = n \log n$$

$$h(n) = \log n! = O(n \log n)$$

Example to show when Big-O does not hold.

e.g. Show n^2 is not $O(n)$

* show there are no witnesses, c, k, s.t. $n^2 \leq cn$ for $x > k$.

we would like $n^2 \leq cn$ then $n^2 = O(n)$

This requires $n \leq c$

But c is constant and $n \to \infty$

There is no constant c that is greater than all n.

$$\neg \exists_c \forall_{n \in R} (n \leq c)$$

Requires $c \to \infty$ and $k \to \infty$