Computational complexity of algorithms.

Consider two functions

$$f: \mathbb{R} \to \mathbb{R}$$
$$g: \mathbb{R} \to \mathbb{R}$$

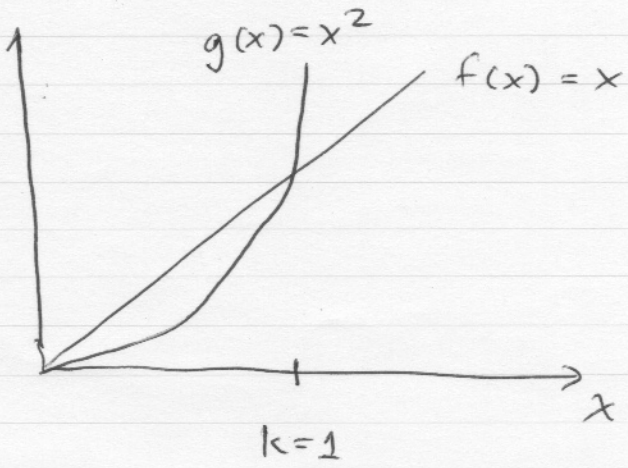
"Big O"

$$f(x) = O(g(x)) \text{ if } |f(x)| \leq c|g(x)| \text{ for } x > k.$$

(Func $|g(x)|$ multiplied by a constant c is always greater than or equal to $|f(x)|$ once x sufficiently large.)

The constants c & k are the "witnesses" to the relationship.

(They are not unique.
If a c_{min}, k_{min} pair exists all $c > c_{min}$ and $k > k_{min}$ are also valid.



$|f(x)| \leq |g(x)|$ for $x > 1$.

$c = 1$, $k = 1$.

also valid e.g.
$c = 10$, $k = 1$
$c = 3$, $k = 2$
etc.

Recall polynomial fun'cs:

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} + a_n x^n = \sum_{i=0}^{n} a_i x^i$$

$$\leq (|a_0| + |a_1| + \cdots |a_n|) x^n$$

$$|f(x)| \leq c\,|g(x)| \quad \text{for} \quad x > 1$$

$$\text{where} \quad c = \sum_{i=0}^{n} |a_i| \quad , \quad g(x) = x^n$$

For a polynomial $f(x) = O(x^n)$

also valid $f(x) \in O(x^n)$

all func's with $O(x^n)$ are in that set $O(x^n)$

(All polynomials with x^n largest power belong to $O(x^n)$)

Hierarchy of complexity classes:

If $f(x) \in O(g(x))$ and $g(x) \in O(h(x))$

then $f(x) \in O(h(x))$

e.g. $f(x) \in O(x^2) \in O(x^3) \in O(x^4) \ldots$

The witnesses are not unique.

e.g. $f(x) = 2x^2 + 100$

Find c, k to show $f(x) = O(x^2)$

Method 1: Formula for polynomial

$$c = \sum_{i=0}^{n} |a_i| = a_0 + a_2 = 102, \quad k > 1.$$

Method 2: (trial & error); Educated guess.

$f(x) = 2x^2 + 100$ Educated guess $g(x) = 3x^2$

set $c = 3$

Find the corresponding k.

$|3x^2| \geq |2x^2 + 100|$ for all $x > k$.

$\Rightarrow x \geq \sqrt{100} = 10.$

$f(x) \in O(x^2)$ { Method 1, $c = 102$, $k = 1$
Method 2, $c = 3$, $k = 10$.

In general:

Addition $f(x) \longrightarrow f(x) + 100$ shift k. (x-axis)

Multiplication $f(x) \longrightarrow 10 f(x)$ shift c. (y-axis)

Recall a hierarchy

$$f(x) \in O(x^2) \in O(x^3) \in O(x^4), \cdots.$$

Polynomial complexity $O(x^n)$.

- Nested loops:

```
for (j in 1 to n)
    for (k in 1 to m)
        execute code.
    end for
end for
```

- $\sum_{j=1}^{n} \sum_{k=1}^{m} (a_k + b_j)$ "Double sum"

- Takes $n \cdot m$ loops to execute.

- Matrix operations

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

"Big Ω"

$$f(x) \in \Omega(g(x))$$

if $|f(x)| \geq c|g(x)|$ for $x > k$., $c, k > 0$.

$|f(x)|$ is bounded from below by $c|g(x)|$

Polynomials:

$$f(x) = \sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

$$\geq a_n x^n$$

$$|f(x)| \geq |a_n| x^n$$

$f(x) \in \Omega(x^n)$ where $c = |a_n|$, $k = 1$.

Recall we showed $f(x) \in O(x^n)$ also!

In detail

$f(x) \geq c_1|g(x)|$ where $c_1 = |a_n|$, $k_1 = 1$

$f(x) \leq c_2|g(x)|$ where $c_2 = \sum_{i=0}^{n} |a_i|$, $k_2 = 1$.

* x^n bounds $f(x)$ from below and from above!

Thus $f(x) = \Theta(g(x))$ ← "Big theta"

In summary

"Big O" $f(x) \in O(g(x))$ $\exists c,k$ const $\forall x>k$ $|f(x)| \leq c|g(x)|$

"Big Ω" $f(x) \in \Omega(g(x))$ $\exists c,k$ const $\forall x>k$ $|f(x)| \geq c|g(x)|$

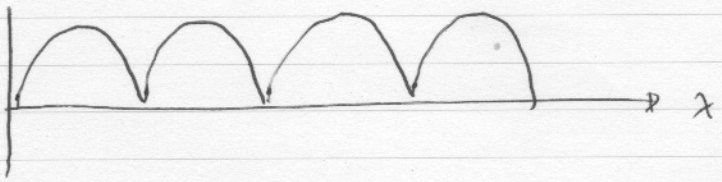
Omega Ω

"Big Θ" means $f(x) \in O(g(x))$ and $f(x) \in \Omega(g(x))$

(tight bound).

Some unusual functions

$|\sin x|$



$|\sin(x)| \in O(1)$

$|\sin(x)| \in \Omega(0)$

Common complexity classes

- Polynomial complexity $\Theta(n^b)$ (class P)
- Intractable computations for classes above P; 2^n, $n!$, etc.
- Unique class NP (nondeterministic polynomial).
  - No polynomial algorithm for execution
  - Given a potential outcome can verify in polynomial time.

→ Factoring ← is a great example.

a) Hard to find the factors of a number n.

b) Easy to verify if two numbers $ab = n$.

The proof $P \neq NP$ is a central challenge in C.S. & a Millenium prize ($1 mil).

New topic : Proof by induction

Proof technique used to prove propositions $P(n)$ about $n \in \mathbb{Z}^+, \mathbb{N}$

Three steps:

Assume $P(n)$ (Inductive hypothesis)

Prove $P(n) \rightarrow P(n+1)$ Inductive step

Prove $P(1) = T$ Basis step

$\therefore \forall_n P(n)$

$$[P(n) \rightarrow P(n+1) \wedge P(1)] \longrightarrow \forall_n P(n)$$

Typical useful for

- Summations
- Inequalities
- Division
- Sets
- Algoritms

Example proof by induction, of a sum.

First need $P(n)$ e.g. $P(n) := \sum_{i=0}^{n} i = \frac{n(n+1)}{2}$

Book (Rosen) example.

Instead consider the sum of the first n odd integers.

$P(n) := \sum_{i=1}^{n} (2i-1)$ what is the closed-form eqn?

$n=1$, $P(1) = 1$

$n=2$, $P(2) = 1+3 = 4$

$n=3$, $P(3) = 1+3+5 = 9$

$n=4$, $P(4) = 1+3+5+7 = 16$

$n=5$, $P(5) = 1+3+5+7+9 = 25$

⋮

Inductive Hypothesis: $P(n) := \sum_{i=1}^{n} (2i-1) = n^2$

Inductive step prove $P(n) \rightarrow P(n+1)$

Direct proof.

$$P(n+1) = \sum_{i=1}^{n+1} (2i-1) = \underbrace{\sum_{i=1}^{n} (2i-1)}_{P(n)} + (2(n+1)-1)$$

$$= P(n) + (2(n+1)-1)$$

$$= n^2 + 2n + 2 - 1 = n^2 + 2n + 1$$

$$= (n+1)^2 \;//$$

Proved given $P(n)$ then $P(n) \rightarrow P(n+1)$

Base case, show $P(1)$ is true

$$P(1) := \sum_{i=1}^{1} (2i-1) = 1 = (1)^2 \quad \checkmark$$

In summary

1) assumed $P(n)$
2) Direct proof $P(n) \rightarrow P(n+1)$
3) Base case $P(1) = \top$

$\therefore \forall_n P(n)$