$f_1(n) \in O(g_1(n))$

$f_2(n) \in O(g_2(n))$

$$f_1(n) \cdot f_2(n) \in O(g_1(n) \cdot g_2(n))$$

FibonacciMultiply($X[0\,..\,\cancel{m-1}\ n-1], Y[0\,..\,n-1]$):

$O(n^2)$

    $hold \leftarrow 0$

    for $k \leftarrow 0$ to $\cancel{n+m-1}\ 2n-1$ $\leftarrow 2n \in O(n)$ times

$O(n)$

        for all i and j such that $i+j=k$ $\leftarrow \leq 2n \in O(n)$ times

            $hold \leftarrow hold + X[i] \cdot Y[j] \in O(1)$

        $Z[k] \leftarrow hold \bmod 10$ $\leftarrow O(1)$

        $hold \leftarrow \lfloor hold/10 \rfloor$ $\leftarrow O(1)$

$O(n+1+1) \in O(n)$

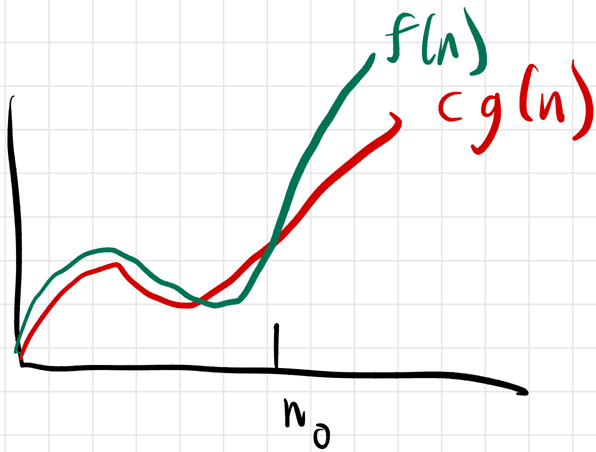
    return $Z[0\,..\,\cancel{m+n-1}\ 2n-1]$

Total: $O(1+n^2+1) \in O(n^2)$

O: loose upper bound

Ω: loose lower bound

$\Omega(g(n)) = \{f(n)$: exist pos. constants c, n_0 s.t. $0 \le cg(n) \le f(n)$ for all $n \ge n_0\}$



Fib.Mult has $\ge n/2 \in \Omega(n)$ values of k s.t. inner loop runs $\ge n/2 \in \Omega(n)$

times.

$\Rightarrow$ runs in $\Omega(n^2)$ time

Big-Theta: $\Theta(g(n)) = O(g(n)) \cap \Omega(g(n))$

FibMult takes $\Theta(n^2)$ time

"asymptotically tight bound"

little o-oh

$o(g(n)) = \{ f(n) :$ for all pos. constants c, there exists n_0 s.t. $0 \le f(n) < c\,g(n)$ for all $n \ge n_0 \}$

little omega ω: tight lower bound

$\omega(g(n)) = \{ f(n) \mid \forall\, c > 0\ \exists\, n_0 > 0$ s.t. $0 \le c\,g(n) < f(n)\ \forall\, n \ge n_0 \}$

$5n^2 = O(n^2)$

$20n^2 + O(n) = O(n^2)$

// for all choices of functions in each bit of asymptotic notation on the left, there exist choices of function for right to make (in)equality true.

FibMult

$$O(1) + O(n)\cdot(O(1) + O(n) + O(1))$$
$$+ O(1)$$
$$= O(n^2)$$

$$5n^2 + 1000n = 5n^2 + O(n)$$
$$= O(n^2)$$
$$= O(2^n)$$
$$\leq O(2^{2^n})$$

$f(n)$ is polynomially bounded if

$f(n) = O(n^k)$ for some constant k.

$$n^{k_1} = o(n^{k_2}) \text{ iff } k_1 < k_2$$

most running times from this class

exponential functions

$n^k = o(a^n)$ for any constants k & $a > 1$.

$a^n = o(c^n)$ for any $c > a > 1$.

polylogarithmically bounded:

$(\log_b n)^\ell = o(n^k)$ for any constants $b > 1$, ℓ, & $k > 0$.

$\lg n := \log_2 n$

$\ln n := \log_e n$

$\log n \overset{?}{:=} \log_{10} n$

$$\log_b^{\ell} n := (\log_b n)^{\ell}$$

$$\log_b n = \frac{\log_a n}{\log_a b} = \Theta(\log n)$$

(a + b constants)

$$\Rightarrow O(n \lg n) = O(n \log n)$$

WARNING:

$$5^{\log_3 n} \in o(5^{\lg n})$$

For running times

$O(n^2)$ is better than

$25 \cdot O(n^2)$

or $O(25n^2)$

$$O(n^2) + O(n) = O(n^2)$$

$$O(5^{\log_3 n}) = O(n^{\log_3 5})$$

A **reduction** from problem X to problem Y means an algorithm for X that uses an algorithm for Y as a "black-box" or subroutine.

Alg for X must be correct for any alg for Y.

(running time might care)

Math uses simple theorems called lemmas. Big important™ proof may reduce to already proven lemmas.

★ theorem:

Let n be a pos. integer.

A <u>divisor</u> is a pos. integer p s.t. n/p is an integer.

n is <u>prime</u> if it has <u>exactly</u> two divisors, n & 1.

n is composite if it has > 2 divisors.

Thm: Every integer $n > 1$ has a prime divisor.

Proof (?): Suppose there is an $n > 1$ with no prime divisor.

n is its own divisor, so n is not prime

So there is a divisor d s.t. $1 < d < n$.

By assumption, d is not prime.

So $\exists d_1$ is a divisor of d

s.t. $1 < d_1 < d$.

$n/d_1 = (n/d) \cdot (d/d_1)$ is an integer so d' divides n.

So d_1 is not prime...

Try 2: Assume Thm is wrong

Let $n > 1$ be the smallest

counter example.

n is not prime

So there is a divisor

$1 < d < n$.

By assumption, d has

a prime divisor $1 < p \leq d$.

$(n/p) = (n/d) \cdot (d/p)$ is an

integer, so p is a prime (!)

divisor of n. $\perp$

Proof: Let $n>1$. Assume all k s.t. $1<k<n$ has a prime divisor.

Suppose n is prime.

n is its own prime divisor!

Suppose n is not prime.

There is a divisor $1<d<n$.

By assumption, d has a prime divisor p.

p is a prime divisor of n.

In all cases, n has a prime divisor.

$\square$

Proof by induction.