$f(n): N \to R^+$

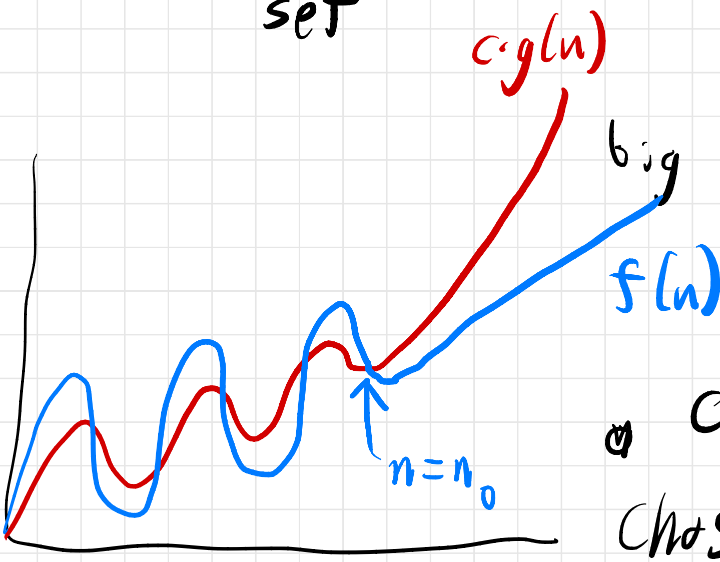
$g(n): N \to R^+$

then it does not exceed a constant multiple of g

$O(g(n)) = \{f(n) :$ there exist positive constants c and n_0 such that $0 \le f(n) \le cg(n)$ for all $n \ge n_0\}.$

set

when n is big enough "to care"



c & n_0 ~~differ~~ chosen for each f(n)

$1000000000n^2 + 10^{100} n = O(n^2)$

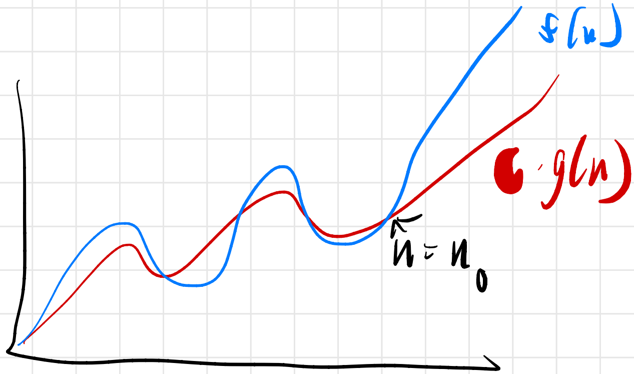
$256n = O(n)$

$256n = O(n^2)$ "a loose upper bound"

$$\Omega(g(n)) = \{f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \le c\,g(n) \le f(n) \text{ for all } n \ge n_0\}.$$

"big Omega"

"loose lower bound"



"big-Theta"

$$\Theta(g) = O(g) \cap \Omega(g)$$

$$c_1 g(n) \le f(n) \le c_2 g(n)$$
$$\forall n \ge n_0$$

* $f(n) \in \Theta(g)$, then $g(n)$ is an asymptotically tight bound on $f(n)$

little-oh $f(n) \in o(g(n))$

informally, $f(n) \in O(g(n))$

but $f(n) \notin \Omega(g(n))$

"strict upper bound"

$f(n) \in \omega(g(n))$

"little omega"

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

but $f(n) \notin O(g(n))$

$f(n) \in O(g(n)) \quad g(n) \in O(h(n))$

$\Rightarrow f(n) \in O(h(n))$

$$c \cdot f_1(n) = O(f_1(n)) \text{ for any positive constant } c, \tag{3.1}$$

$$f_1(n) + f_2(n) = O(g_1(n) + g_2(n)), \tag{3.2}$$

$$f_1(n) \cdot f_2(n) = O(g_1(n) \cdot g_2(n)), \text{ and} \tag{3.3}$$

$$f_1(n) + f_2(n) = O(\max\{g_1(n), g_2(n)\}). \tag{3.4}$$

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

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

$f(n) = O(g(n))$

$\forall f_2(n) \in O(1)$

$+O(1)$

$25n^2 + O(n) = \Theta(n^2)$

for all $f_1(n) \in O(n)$

$\exists f_3(n) \in \Theta(n^2)$

$\log^{\ell'} n =$

$$o(\log^{\ell} n) = o(n^k) = o(n^{k'}) = o(c^n) = o(d^n)$$

polylog

polynomial

exponentials

for any constants $\ell > \ell'$

$k' > k > 0$

$d > c > 1$

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

$\lg n := \log_2 n$

$\ln n := \log_e n$

$\log_a n = \Theta(\log_c n)$ for all constants a, c

```
FIBONACCIMULTIPLY(X[0 .. m-1], Y[0 .. n-1]):
  hold ← 0
  for k ← 0 to n + m − 1
      for all i and j such that i + j = k
          hold ← hold + X[i] · Y[j]
      Z[k] ← hold mod 10
      hold ← ⌊hold/10⌋
  return Z[0..m + n − 1]
```

O(n) iters

O(n) iterations

O(1) per

O(1) per

O(n) per

Suppose $m = n$.

$O(n) \cdot O(n) = O(n^2)$ total run time

$\Theta(n^2)$ actually

```
MERGESORT(A[1 .. n]):
  if n > 1
        m ← ⌊n/2⌋
        MERGESORT(A[1 .. m])          ⟨⟨Recurse!⟩⟩
        MERGESORT(A[m + 1 .. n])      ⟨⟨Recurse!⟩⟩
        MERGE(A[1 .. n], m)
```

```
MERGE(A[1 .. n], m):
  i ← 1;  j ← m + 1
  for k ← 1 to n
        if j > n
              B[k] ← A[i];  i ← i + 1
        else if i > m
              B[k] ← A[j];  j ← j + 1
        else if A[i] < A[j]
              B[k] ← A[i];  i ← i + 1
        else
              B[k] ← A[j];  j ← j + 1

  for k ← 1 to n
        A[k] ← B[k]
```

for divide-and-conquer, use a recurrence

$T(n) :=$ worst-case time for mergesorting $A[1..n]$.

$$T(\Theta(1)) = \Theta(1)$$

$$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + O(n)$$

$$\approx 2 \cdot T(n/2) + O(n)$$

$$= ?$$

recursion trees

rooted tree

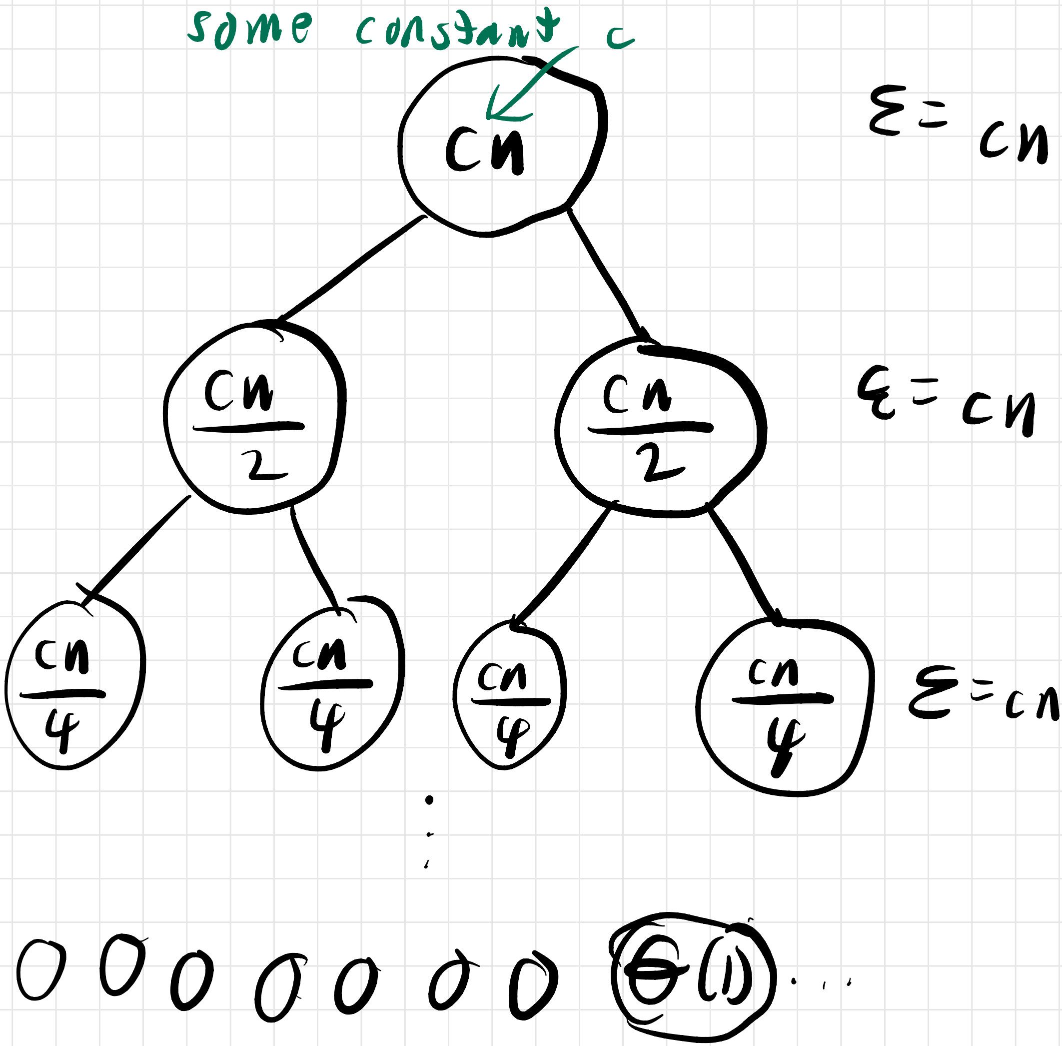
nodes: individual recursive subproblems found during execution

root: top level call

children of a node: direct recursive calls

value on a node: contribution to sum/amount of work done by that subproblem

(does not include recursive calls)



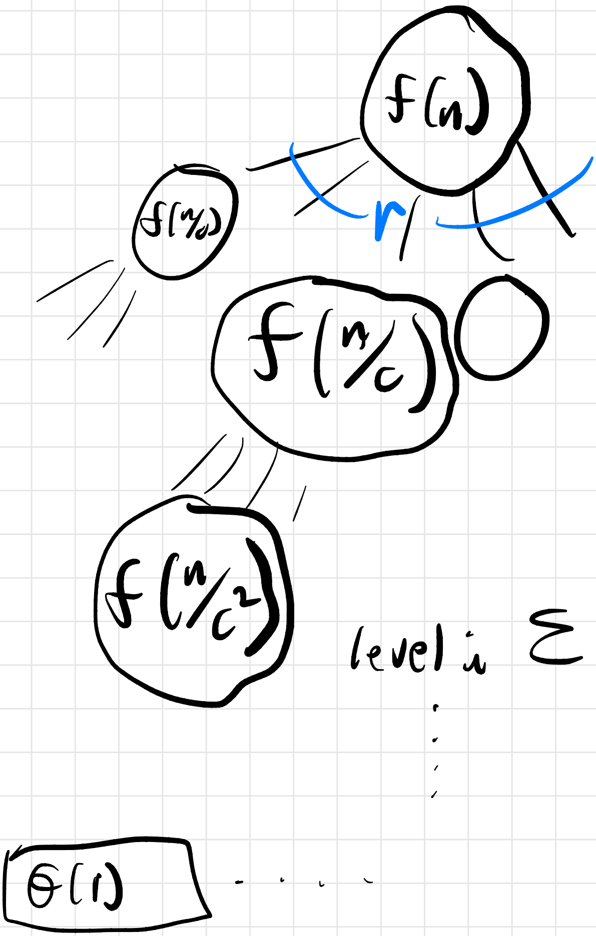
$T(n) = \Sigma$ of node values

Σ = sum

lg n levels

$T(n) \le cn \lg n = \Theta(n \log n)$

often $T(n) = \overset{\text{r recursive calls}}{r} \cdot T(\underbrace{n/c}_{n/c \text{ subproblem size}}) + \underbrace{f(n)}_{\text{work}}$



$\Sigma = f(n)$

$\Sigma = r \cdot f(n/c)$

$\Sigma = r^2 \cdot f(n/c^2)$

level i $\Sigma = r^i \cdot f(n/c^i)$

Three common cases

Decreasing: decay exponentially

i.e. $r \cdot f(n/c) = k \cdot f(n)$ where $k<1$

$$T(n) = \Theta(f(n))$$

Equal: $r \cdot f(n/c) = f(n)$

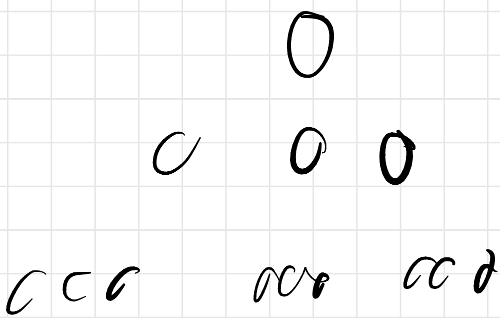
$$T(n) = f(n) \cdot \#\text{levels}$$
$$= \Theta(f(n) \log_c n)$$
$$= \Theta(f(n) \log n)$$

Increasing: grows exponentially

$$T(n) = \#\text{ leaves}$$
$$= \Theta(r^{\log_c n}) = \Theta(n^{\log_c r})$$

Compare to (but don't memorize) Master method [CLRS]

$$T(n) = 3T(n/2) + n$$



$n \quad n$

$3 \cdot n/2 \quad \frac{3}{2} n$

$9 \cdot n/4 \quad \left(\frac{3}{2}\right)^2 n$

$\vdots$

$$T(n) = \theta\left(n^{\log_2 3}\right)$$