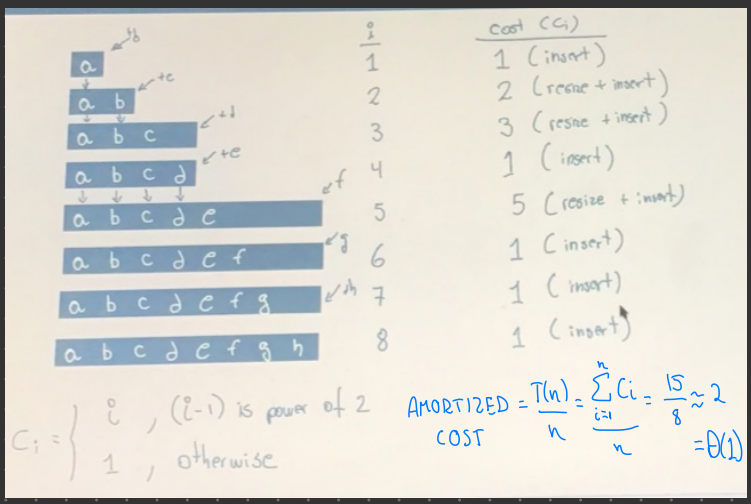
Aggregate Method

Steps:
1) Calculate the total cost of all n operations as $T(n)$
2) Calculate average cost of each operation as $T(n)/n$

$$\sum \left(\begin{matrix} \text{CHEAP} \\ \text{OPS} \end{matrix} + \begin{matrix} \text{EXPENSIVE} \\ \text{OPS} \end{matrix} \right) \leq 2n \approx O(n)$$



Account Method (Propóner costo amortizado)

Intuition: Low cost and frequent operations are charged more than high cost and less frequent operations.
c_i: Actual cost
$\hat{c}_i$: Charged cost

$$\sum_{i=1}^{n} c_i \leq \sum_{i=1}^{n} \hat{c}_i$$ → balance shouldn't be negative

Ejercicio

$\hat{c}_i = \$3.00$ { $1.00 for insert
{ $2.00 stored for later use (resize)

	$\hat{c}_i$	Credit	Cost	Balance
1	3	3	1	2
1 2	3	5	1+1	3
1 2 3	3	6	2+1	3
1 2 3 4	3	6	1	5
1 2 3 4 5	3	8	4+1	3
1 2 3 4 5 6	3	6	1	5
1 2 3 4 5 6 7	3	8	1	7
1 2 3 4 5 6 7 8	3	10	1	9
1 2 3 4 5 6 7 8 ...	3	12	8+1	3

$\hat{c}_i = 3 \approx \Theta(1)$

Potential Method (proponer cuenta bancaria)

$\phi : \{D_i\} \to \mathbb{R}$ defines the state of a data structure D.

- $\phi(D_0) = 0$
- $\phi(D_i) \geq 0, \forall i$

Amortized Cost $\hat{c}_i$ with respect to ϕ

$\hat{c}_i = c_i + \Delta(\phi_i)$

$\hat{c}_i = c_i + \phi(D_i) - \phi(D_{i-1})$

- $\Delta\phi_i > 0 \Rightarrow \hat{c}_i > c_i$: save energy for later
- $\Delta\phi_i < 0 \Rightarrow \hat{c}_i < c_i$: release energy to afford operation

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} (c_i + \phi(D_i) - \phi(D_{i-1})) \to \sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \underbrace{\phi(D_n)}_{\geq 0} - \underbrace{\phi(D_0)}_{0}$$

Ejercicio

$\phi(D_i) = 2i - \text{size}$ ← capacity i $\qquad \phi(D_i) = 2i - 2^{\lceil \log i \rceil}$

$\hat{c}_i = c_i + \phi(D_i) - \phi(D_{i-1})$

$\hat{c}_i = c_i + (2i - \text{size}) - (2(i-1) - \text{size})$

Case 1: // Insertion

$\hat{c}_i = 1 + 2i - \text{size} - 2i + 2 + \text{size}$

$\hat{c}_i = 3$

Case 2: // Expansion

$\hat{c}_i = (i+1) + (2i - 2i) - (2(i-1) - i)$ (2i: after, 2(i-1): before)

$\hat{c}_i = 3$

$\phi(D_i) = 2i - 2^{\lceil \log i \rceil}$

Case 1: Insertion

$\hat{c}_i = c_i + (2i - 2^{\lceil \log i \rceil}) - (2(i-1) - 2^{\lceil \log i \rceil})$

$\hat{c}_i = c_i + (2i - 2^{\lceil \log i \rceil}) - (2(i-1) - 2^{\lceil \log i-1 \rceil})$

$\hat{c}_i = c_i + 2i - 2^{\lceil \log i \rceil} - 2i + 2 + 2^{\lceil \log i-1 \rceil}$

Case 2: Expansion

$\hat{c}_i = c_i +$

Aggregate Analysis

Cost of MultiPop: $\min(s, k)$

Any sequence of n Push, Pop and MultiPop operations on an initially empty stack can cost at most $O(n)$.

The number of Pop operations is at most the number of Push operations, which is at most n.

For any value of n, any sequence of n Push, Pop and MultiPop operations takes a total of $O(n)$ time.

The average cost of an operation is $O(n)/n = O(1)$

In aggregate analysis, we assign the amortized cost of each operation to the average cost.

We showed a worst case bound of $O(n)$ on a sequence of n operations.

Accounting Method

Actual Costs:
- Push 1
- Pop 1
- MultiPop $\min(s, k)$

Amortized Costs
- Push 2
- Pop 0
- MultiPop 0

When we push we are left with a credit of \$1.00.

When we pop, we use the dollar left by the push of the same object.

Since each plate on the stack has \$1.00 on it, and the stack has a nonnegative number of plates, we have ensured that the amount of credit is always nonnegative. Thus, for any sequence of n operations the total amortized cost is an upper bound on the total cost. Amortized cost is $O(n)$, so is the actual cost.

Potential Method

$\hat{c}_i$: Actual cost of ith operation

D_i: The data structure that results after applying the ith operation to data structure D_{i-1}.

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1})$$

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} (c_i + \Delta\Phi_i)$$

$$= \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0)$$

Stack Operations

Φ: # of items in the stack

$\Phi(D_0) = 0$

$\Phi(D_i) \geq 0 \geq \Phi(D_0)$

If the ith operation is a Push

$$\phi(D_i) - \phi(D_{i-1}) = (s+1) - s$$
$$= 1$$

$$\hat{c}_i = c_i + \Delta\phi_i$$
$$\hat{c}_i = 1 + 1 = 2$$

If the ith operation is a MultiPop

$$\phi(D_i) - \phi(D_{i-1}) = (s-K) - (s)$$
$$= -K$$

$$\hat{c}_i = c_i + \Delta\phi_i$$
$$\hat{c}_i = k - K$$
$$\hat{c}_i = 0$$

If the ith operation is a Pop

$$\phi(D_i) - \phi(D_{i-1}) = (s-1) - (s)$$
$$= -1$$

$$\hat{c}_i = c_i + \phi(D_i) - \phi(D_{i-1})$$
$$= 1 - 1 = 0$$

The amortized cost of each operation is $O(1)$, and thus the total amortized cost of a sequence of n operations is $O(n)$. Since $\phi(D_i) \geq \phi(D_0)$, the total amortized cost of n operations is an upper bound on the actual cost.

17.3-6

Show how to implement a queue with two ordinary stacks (Exercise 10.1-6) so that the amortized cost of each ENQUEUE and each DEQUEUE operation is $O(1)$.

ϕ: # NUMBER OF ITEMS

$\phi(D_0) = 0$

$\phi(D_i) \geq 0 \geq \phi(D_0)$

IF THE OPERATION IS AN ENQUEUE

$$\phi(D_i) - \phi(D_{i-1}) = (s+1) - (s) = 1$$

$$\hat{c}_i = c_i + \Delta\phi_i = 1 + 1 = 2$$

IF THE OPERATION IS A DEQUE

$$\phi(D_i) - \phi(D_{i-1}) = (s-1) - s = -1$$

$$\hat{c}_i = c_i + \Delta\phi_i = 1 - 1 = 0$$

17.3-3

Consider an ordinary binary min-heap data structure with n elements supporting the instructions INSERT and EXTRACT-MIN in $O(\lg n)$ worst-case time. Give a potential function Φ such that the amortized cost of INSERT is $O(\lg n)$ and the amortized cost of EXTRACT-MIN is $O(1)$, and show that it works.

$\phi: \sum_{i=1}^{n} \lg i$ WHERE n IS THE SIZE OF THE MIN-HEAP

THERE IS STILL A COST OF $O(\lg n)$ TO INSERT, SINCE ONLY $\lg n$ WAS SPENT TO INCREASE THE SIZE OF THE HEAP BY ONE.

AMORTIZED COST OF EXTRACT-MIN IS O, AS ALL ITS ACTUAL COST IS COMPENSATED

D_i	ith OPERATION	$\hat{c}_i - c_i$
°	INSERT	~~$\lg 0 - \lg 0$~~ $= 0$
°°	INSERT	$\lg 1 - \lg 1 = 0$
°°°	INSERT	$\lg 2 + \lg 1 - \lg 2 = \lg 1$
°°°°	INSERT	$\lg 3 + \lg 2 + \lg 1 - \lg(3) = \lg 2 + \lg 1$
°°°°°	INSERT	$\lg 4 + \lg 3 + \lg 2 + \lg 1 - \lg 4 = \lg 3 + \lg 2 + \lg 1$

17.1-3

Suppose we perform a sequence of n operations on a data structure in which the ith operation costs i if i is an exact power of 2, and 1 otherwise. Use aggregate analysis to determine the amortized cost per operation.

$T(n) = \sum c_i$

$$\sum_{i=1}^{n} c_i = \sum_{i=1}^{\lg n} 2^i + \sum 1 \quad \text{(} i \le n \text{ IS NOT A POWER OF TWO)}$$

$$\le \sum_{i=1}^{\lg n} 2i + n$$

$$= 2^{\lg n + 1} - 1 + n$$

$$= 2 \cdot 2^{\lg n} + n$$

$$= 2 \cdot n^{\lg 2} + n - 1$$

$$= 3n - 1 \in O(n)$$

AMORTIZED COST PER OPERATION: $\frac{T(n)}{n} = \frac{O(n)}{n} = O(1)$

Master Method

$T(n) = a\,T(n/b) + f(n)$, $a \geq 1$, $b > 1$, $f(n) = O(n^k \log^p n)$

① if $\log_b a > k$ THEN $O(n^{\log_b a})$

② if $\log_b a = k$
- if $p > -1$ THEN $O(n^k \log^{p+1} n)$
- if $p = -1$ THEN $O(n^k \log\log n)$
- if $p < -1$ THEN $O(n^k)$

③ if $\log_b a < k$
- if $p \geq 0$ $O(n^k \log^p n)$
- if $p < 0$ $O(n^k)$

Esercicios

- $T(n) = 4T(n/3) + n^1$
  $\log_3 4 > 1$ $\Theta(n^{\log_3 4})$
- $T(n) = 4T(n/2) + n^2$
  $\log_2 4 = 2$ $\Theta(n^2 \log n)$
- $T(n) = T(n/2) + \sqrt{n}$
  $\log_2 1^{0} < 1/2$ $\Theta(n^{1/2})$
- $T(n) = 7T(\lceil n/2 \rceil) + \Theta(n^2)$
  $\log_2 7 > 2$ $\Theta(n^{\log_2 7})$
- $T(n) = 7T(n/2) + n^2$ $\Theta(n^{\log_2 7})$
- $T(n) = 2T(n/3) + n^3$ $\Theta(n^3)$
- $T(n) = 3T(n/2) + n\log n$
  $\log_2 3 > 1$ $\Theta(n^{\log_2 3})$
- $T(n) = 3T(\lceil n/2 \rceil) + \Theta(n)$
  $\log_2 3 > 1$ $\Theta(n^{\log_2 3})$
- $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + \Theta(n \log n)$
  $\log_2 2 = 1$ $\Theta(n \log^2 n)$

Theorem 4.1 (Master theorem)
Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

$$T(n) = aT(n/b) + f(n),$$

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. ■

- $T(n) = 3T(n/4) + n\lg n$

  $n^{\log_4 3} = O(n^{0.793})$

  $n\lg n = \Omega(n^{\log_4 3 + \epsilon})$, WHERE $\epsilon = 0.2$

  $af(n/b) \leq cf(n)$
  $3(n/4)\lg(n/4) \leq 3/4\, n\lg n$

  $T(n) = n\lg n$

- $T(n) = 2T(n/2) + n\lg n$

  $n^{\log_2 2} = n \qquad f(n) = n\lg n$

  $f(n)/n^{\log_b a} = (n\lg n)/n = \lg n \rightarrow$ ASYMPTOTICALLY LESS THAN n^{ϵ}

  $f(n) = \Omega(n^{\log_b a + \epsilon})$?

  $\lim_{n\to\infty} \frac{n\lg n}{n^{1+\epsilon}} = \frac{\cancel{n}\lg n}{\cancel{n}\, n^{\epsilon}} = \frac{\lg n}{n^{\epsilon}} \xrightarrow{\text{L'HOPITEL}} \frac{1/n}{\epsilon n^{\epsilon-1}} = \frac{n^{-1}}{\epsilon n^{\epsilon} n^{-1}} = \frac{1}{\epsilon n^{\epsilon}} = 0$

  $f(n) \neq \Omega(n^{\log_b a + \epsilon})$

- $T(n) = 3T(n/4) + n\lg n$

  $n^{\log_4 3}$

  $f(n) = \Omega(n^{\log_4 3 + \epsilon})$?

  $\lim_{n\to\infty} \frac{n\lg n}{n^{\log_4 3 + \epsilon}} = \frac{n\lg n}{\cancel{n^{0.8}}\, n^{\epsilon}} = n^{0.2-\epsilon}\lg n = \infty$

  $f(n) = \Omega(n^{\log_4 3 + \epsilon})$

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

- $T(n) = 2T(n/2) + \Theta(n)$

  $T(n \lg n)$
- $T(n) = 8T(n/2) + \Theta(n^2)$

  $n^{\log 8} = n^3$

  $f(n) = O(n^{3-1})$

  $T(n) = \Theta(n^3)$
- $T(n) = 7T(n/2) + \Theta(n^2)$

  $n^{\log 7}$

  $f(n) = O(n^{\log_b a - \epsilon})$

  $T(n) = \Theta(n^{\lg 7})$

Substitution Method

- $T(n) = T(\lceil n/2 \rceil) + 1$, $T(n) = O(\log n)$

  $T(k) \leq c\log k$, $k < n$

  $T(n) \leq c\log(\lceil n/2 \rceil) + 1$
  $\leq c\log(n/2 + 1) + 1$
  $\leq c\log((n+2)/2) + 1$
  $\leq c\log(n+2) - c\log 2 + 1$
  $\leq c\log(n+2) - c + 1$ → DOESN'T CLOSE INDUCTION

  $T(k) \leq c\log(k-2)$, $k < n$

  $T(n) \leq c\log(\lceil n/2 \rceil - 2) + 1$
  $\leq c\log(n/2 + 1 - 2) + 1$
  $\leq c\log(n/2 - 1) + 1$
  $\leq c\log((n-2)/2) + 1$
  $\leq c\log(n-2) - \underbrace{(c-1)}_{\geq 0,\ c \geq 1}$

- $T(n) = T(n-1) + n$

  $T(k) \leq ck^2$, $k < n$

  $T(n) \leq c(n-1)^2 + n$
  $\leq cn^2 - 2cn + c + n$
  $\leq cn^2 - (2cn - c - n)$
  ≥ 0 $\quad 2cn - c - n \geq 0$, $c \geq 1$

- $T(n) = T(n-2) + \log n$

  $T(k) \leq ck\log(k)$

  $\log(n)$
  $\log(n-2)$
  $\log(n-4)$
  $\vdots$
  $\log(2)$

  $\log n + \log(n-2) + \cdots + \log(2) \leq \log(n) + \log(n-1) + \log(n-2) + \cdots \log(2) + \log(1)$
  $= \log(n(n-1)(n-2)(n-3)\cdots(2)(1))$
  $= \log n!$

  $\log(1) + \log(2) + \cdots + \log(n) \leq n\log n$

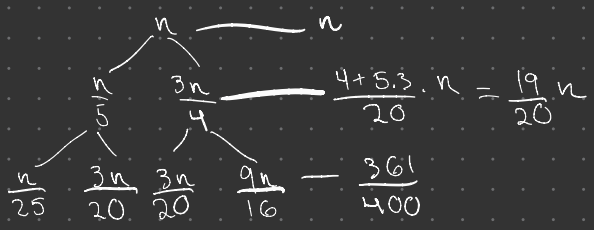
  ENCADENANDO EXPRESIONES, $T(n) = O(n\log n)$

- $T(n) = 2T(\lfloor n/2 \rfloor + 17) + n$

  $T(k) \leq ck \log k$

  $T(n) \leq 2c(\lfloor n/2 \rfloor + 17)\log(\lfloor n/2 \rfloor + 17) + n$
  $\leq 2c(n/2 + 17)\log(n/2 + 17) + n$
  $\leq 2c\left(\frac{n+34}{2}\right)\log(n/2 + 17) + n$
  $\leq (cn + 34c)\log(n/2 + n/4) + n$
  $\leq (cn + 34c)\log(n/(4/3)) + n$
  $\leq cn\log n - cn\log 4/3 + 34c\log n - 34c\log 4/3 + n$
  $\leq cn\log n - cn\log 4/3 + kn - 34c\log 4/3 + n$
  $- (cn\log 4/3 - kn + 34c\log 4/3 - n)$
  $- (n(c\log 4/3 - k - 1)$
  $\geq 0 \qquad c \geq (k+1)/\log 4/3$

- $T(n) = T(n/5) + T(3n/4) + \Theta(n)$



$$n \cdot \sum_{i=0}^{\infty} \left(\frac{19}{20}\right)^i = n \cdot \frac{1}{1 - 19/20} = n \cdot \frac{1}{1/20} = 20n$$

$T(k) \leq ck, k < n$ $\qquad$ $T(k) \leq 20ck$

$T(n) \leq cn/5 + 3cn/4 + cn$ $\qquad$ $T(n) \leq 4cn + 15cn + cn$
$\leq \frac{39}{20} cn$ $\qquad$ $\leq 20cn$

- $T(n) = 3T(\sqrt{n}) + \log n$

  $\log n = m$
  $n = 2^m$
  $\sqrt{n} = 2^{m/2}$

  $T(2^m) = 3T(2^{m/2}) + m$

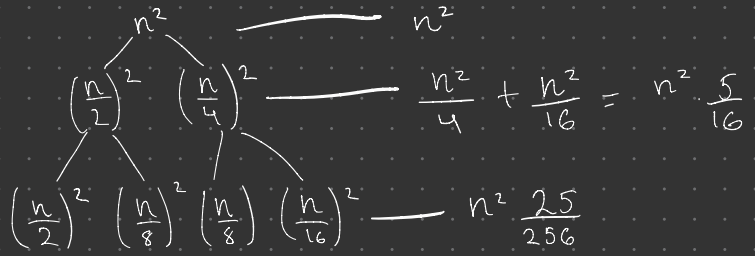
  $Q(k) = T(2^k)$

  $T(2^m) = Q(m) = 3Q(m/2) + m$

  $Q(m) = \Theta(m^{\log 3}) \rightarrow T(n) = \Theta(\log n^{\log 3})$
  $T(2^m) = T(n)$ $\qquad$ $\log n$

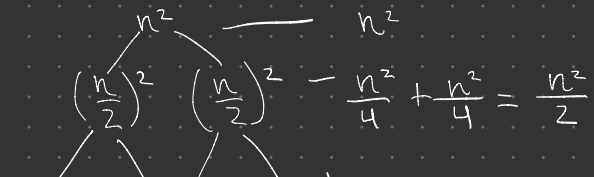
Recursion Tree

- $T(n) = T(n/2) + T(n/4) + \theta(n^2)$



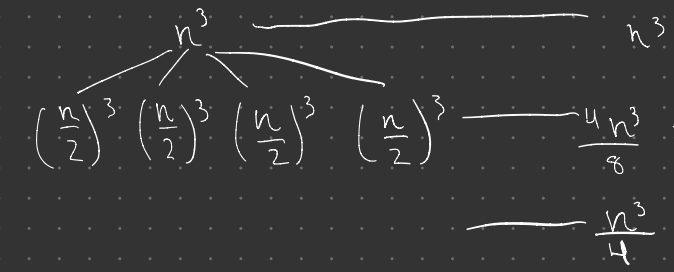
$$n^2 \sum_{k=0}^{\infty} \left(\frac{5}{16}\right)^k = n^2 \frac{1}{1 - 5/16} = n^2 \frac{1}{11/16} = \frac{16}{11} = O(n^2)$$

- $T(n) = 2T(n/2) + n^2$



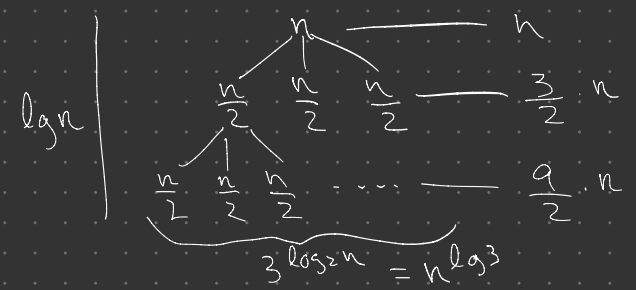
$$n^2 \sum_{k=0}^{\infty} \left(\frac{1}{2}\right)^k = n^2 \cdot \frac{1}{1 - 1/2} = 2n^2. \quad T(n) = O(n^2)$$

- $T(n) = 4T(n/2) + n^3$



$$n^3 \sum_{k=0}^{\infty} \left(\frac{1}{2}\right)^k = n^3 \cdot \frac{1}{1 - 1/2} = 2 \cdot n^3. \quad T(n) = O(n^3)$$

- $T(n) = 3T(\lfloor n/2 \rfloor) + n$



$T(k) = ck^{\log 3}$

$T(n) \leq 3 \cdot c(\lfloor n/2 \rfloor)^{\log 3} + n$
$\leq 3 \cdot c(n/2)^{\log 3} + n$
$\leq 3 \cdot c \frac{n^{\log 3}}{2^{\log 3}} + n$
$\leq 3 \cdot c \frac{n^{\log 3}}{3} + n$
$\leq cn^{\log 3} + n$

$T(k) = ck^{\log 3} - dn$

$T(n) \leq 3[(c(n/2)^{\log 3}) - d(n/2)] + n$
$\leq 3\left[\frac{cn^{\log 3}}{3} - \frac{dn}{2}\right] + n$
$\leq cn^{\log 3} - 3/2 \cdot dn + n$

$cn^{\log 3} - 3/2 \cdot dn + n \leq cn^{\log 3}$
$n \leq 3/2 \cdot dn$
$2/3 \leq d$

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

  $\lim_{n\to\infty} \frac{T(n)}{T(n)} = 1, \quad c>0 \Rightarrow T(n) = \Theta(T(n))$

- $T(n) = \Theta(g(n))$ AND $g(n) = \Theta(h(n))$ IMPLIES $T(n) = \Theta(h(n))$

  $\lim_{n\to\infty} \frac{T(n)}{g(n)} = c_1 > 0 \quad \lim_{n\to\infty} \frac{g(n)}{h(n)} = c_2 > 0$

  $\frac{T(n)}{g(n)} \cdot \frac{g(n)}{h(n)} = \frac{T(n)}{h(n)} = c_1 \cdot c_2 > 0 \rightarrow T(n) = \Theta(h(n))$

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

  $\lim_{n\to\infty} \frac{T(n)}{f(n)} = c \quad \lim_{n\to\infty} \frac{f(n)}{T(n)} = \frac{1}{c}$

- $f(n) \neq O(g(n))$ AND $g(n) \neq O(f(n))$

  $\lim_{n\to\infty} \frac{f(n)}{g(n)} \nearrow \infty \searrow c \qquad \lim_{n\to\infty} \frac{g(n)}{f(n)} \nearrow \infty \searrow c$ SOMETIMES TRUE

- $f(n) = \Omega(g(n))$ AND $f(n) = o(g(n))$

  $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty \qquad \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$ NEVER TRUE

- $f(n) = \Theta(f(n/2))$

  $\lim_{n\to\infty} \frac{f(n)}{f(n/2)} = \infty$ NEVER TRUE

- $f(n) = O(g(n)) \Rightarrow g(n) = \Omega(f(n))$

  $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \qquad \lim_{n\to\infty} \frac{g(n)}{f(n)} = \infty$ ALWAYS TRUE

- CONSIDER $T(n) = 4T(n/2) + f(n)$ WHERE $f(n) = n^2 \log n$, SHOW IF $f(n)$ IS POLYNOMIALLY GREATER THAN $n^{\log_b a}$. IF THAT IS TRUE, SOLVING THE M.T. ELSE PROPOSE AN UPPER BOUND AND A LOWER BOUND.

  $n^{\log_b a} = n^2$

  WE WANT TO PROVE $f(n) = \Omega(n^2)$

  $\lim_{n\to\infty} \frac{n^2 \log n}{n^2} = \infty \Rightarrow f(n) = \Omega(n^2)$. $f(n)$ IS LARGER THAN n^2

  $f(n) = \Omega(n^{\log_a b + \epsilon})$?

$\lim_{n\to\infty} \frac{n^2 \log n}{n^{2+\epsilon}} = \lim_{n\to\infty} n^{2-2-\epsilon} \log n = \lim_{n\to\infty} \frac{\log n}{n^\epsilon} \xrightarrow{\text{L'HOPITEL}} \frac{1/n}{\epsilon n^{\epsilon-1}} = \frac{1}{\epsilon n^\epsilon} = 0 = O(1)$

$f(n) \neq \Omega(n^{\log_b a + \epsilon})$

- $T(n) = 4T(n/2) + n^2/\log n$

$n^{\log_a b} = n^{\log_2 4} = n^2$

$\lim_{n\to\infty} \frac{f(n)}{n^{\log_b a}} = \frac{n^2/\log n}{n^2} = \frac{1}{\log n} = 0 \quad f(n) = O(n^{\log_b a}),\ f(n) = o(n^{\log_b a})$

$f(n)$ IS POLYNOMIALLY SMALLER THAN $n^{\log_b a}$?

$f(n) = O(n^{\log_b a - \epsilon})$

$\lim_{n\to\infty} \frac{f(n)}{n^{\log_b a - \epsilon}},\ \epsilon > 0 \Rightarrow \lim_{n\to\infty} \frac{n^2/\log n}{n^{2-\epsilon}} = \frac{1/\log n}{n^{-\epsilon}} = \frac{n^\epsilon}{\log n}$

$\lim_{n\to\infty} \frac{n^\epsilon}{\log n} \xrightarrow{\text{L'HOPITEL}} \frac{\epsilon n^{\epsilon-1}}{1/n} = \epsilon n^{\epsilon-1+1} = \epsilon n^\epsilon \Rightarrow \epsilon \lim_{n\to\infty} n^\epsilon = \infty$

$f(n) = \Omega(n^{\log_b a - \epsilon})$