Tiempos de ejecución

(Agregar el principio de invariancia)

- el peor escenario ← utilizado en aplicaciones críticas
- el mejor escenario ← utilizado nunca jaja
- el escenario promedio ← utilizado cuando hay múltiples ejecuciones en distintas instancias

Para el escenario promedio, en un algoritmo que requieren n elementos, se deben tomar en consideración $n!$ combinaciones posibles.

El mal viaje de esta medición es que, en casos dónde buscamos ordenar una lista, si está parcialmente ordenada no te sirve cómo vara para medir que pasa, pues toma el promedio de las $n!$ combinaciones posibles, y acá no hay ningúna distribución normal, sino que, justamente, puede haber una mayoría de listas parcialmente ordenadas. Bueno, para más data repasar media-mediana-moda y campana de Gauss.

Operaciones elementales

- suma
- resta
- multiplicación
- división
- resto
- comparación
- asignación

En general, se considera con un costo unitario a la suma, resta, la multiplicación, división, módulo, comparaciones y asignaciones. Igual esto depende de qué estemos haciendo, el sistema en el que estamos trabajando, el lenguaje, etc, pero en teoría valen 1, y dsp se usa la razón para determinar si hay alguna complejidad oculta

Orden de operaciones.

Por el principio de invariancia, sabemos que dos implementaciones de un mismo algoritmos difieren en tiempo de ejecución por una constante $t(n) \leq c \cdot t(n)$. Bajo el mismo concepto, podemos determinar el orden de algoritmo, en relación con una funcion $t : \mathbb{N} \to \mathbb{R}^{\geq 0}$

Por ejemplo, tenemos un algoritmo con $t(n) = 24n^2 + 6n + \frac{1}{2}$. Sabemos que, para un n_0 lo suficientemente grande, existe un $n \geq n_0$ tal que

$$\begin{aligned} t(n) &= 24n^2 + 6n + \frac{1}{2} \\ &\leq 24n^2 + 6n^2 + \frac{1}{2}n^2 \\ &\leq \frac{61}{2}n^2 \end{aligned}$$

luego, nuestro algoritmo pertenece al orden de n^2, por el principio de invariancia, pues $24n^2 + 6n + \frac{1}{2} \leq c \cdot n^2$, en particular $c = \frac{61}{2}$, pero este número puede cambiar si elegimos un n_0 distinto.

Big $O(f(n))$

(clarificar la *treshhold rule* ??)

Decimos que Big $O(f(n))$ es el conjunto de todas las funciones que cumplen $t : \mathbb{N} \to \mathbb{R}^{\geq 0} | t(n) \leq c \cdot t(n)$ para cualquier implementación del algoritmo.

$$O(f(n)) = \{t : \mathbb{N} \to \mathbb{R}^{\geq 0} |\ (\exists c \in \mathbb{R}^+)(\forall n \in \mathbb{N})[t(n) \leq c \cdot t(n)]\}$$

Notemos que, además de esto, cómo $O(f(n))$ es el análisis asintótico de una función $\mathbb{N} \to \mathbb{R}^+$, podemos agregar un umbral, un 'límite' a partir del cual una función sea mayor que otra, y por lo tanto, también pertenezca a $O(f(n))$. Por ejemplo

$n^3 - 3n^2 - n - 8 \in O(n^3)$, aunque si $n \leq 3 \Rightarrow t(n) < 0$. Podemos expandir nuestra definición para incluir este comportamiento

$$O(f(n)) = \{t : \mathbb{N} \to \mathbb{R}^{\geq 0} |\ (\exists c \in \mathbb{R}^+)(\exists n_0 \in \mathbb{N})(\forall n > n_0)[t(n) \leq c \cdot t(n)]\}$$

Prove that $t(n) \in O(t(n))$
Completar, pag 102. Creo que no es muy necesario 🧐

Pero yo lo haría de la siguiente forma (a chequear esto eh, que me lo estoy inventando)
1. Probar la existencia de un n_0 tal que $t(n) > 0$. Este n_0 es nuestro candidato
2. Luego, ver qué función elemental mínima cumple que $t(n) < f(n), \forall n > n_0$.

En este último paso, podemos reafirmar nuestro n_0 candidato, o encontrar otro distinto.

Maximum Rule
Sean $f, g : \mathbb{N} \to \mathbb{R}^{\geq 0}$, entonces $O(f(n) + g(n)) = \max\{f(n), g(n)\}$.

Además, sean $p, q : \mathbb{N} \to \mathbb{R}^{\geq 0}\ |\ p(n) = f(n) + g(n) \land q(n) = \max\{f(n), g(n)\}$. Entonces, si $t(n) \in O(p(n)) \Rightarrow t(n) \in O(q(n))$

Ejemplos

$$O(n^3 + n^2 + n) = O(n^3) \tag{1}$$

$$O(n^2 - n^2 + n) \neq O(n^2) \tag{2}$$

$$\begin{aligned} O(n^3 \log(n) + n^3 - n^2 + 3) &= O(\max\{n^3 \log(n), (n^3 - n^2), 3\}) \\ &= O(n^3 \log(n)) \end{aligned} \tag{3}$$

Notemos que (2) no funca, porque no cumple la definición de la regla máxima, pues $f(n) = -n^2 \notin f : \mathbb{N} \to \mathbb{R}^{\geq 0}$

En (3) vemos un "workaround" para trabajar esta situación, de forma tal que podamos agrupar todas nuestras funciones en algo de la forma $f : \mathbb{N} \to \mathbb{R}^{\geq 0}$

$$\begin{aligned} O(t(n)) &= O(11n^3 \log(n) + n^3 \log(n) - 5n^2 + \log^2(n) + 36) \\ &= O(\max\{11n^3 \log(n), n^3 \log(n) - 5n^2, \log^2(n) + 36\}) \\ &= O(n^3 \log(n)) \end{aligned} \tag{4}$$

Sobre los logaritmos en la notación asintótica
No nos interesa su base, pues

$$\log_a(n) = \log_a(b) \times \log_b(n), \forall a, b, n > 0 \land a, b \neq 1$$

Además, cómo

$$\log_a(b) > 0 \forall a, b > 1$$

Por lo tanto, sabemos que $\log_a(n)$ y $\log_b(n)$ se diferencian sólo por un factor positivo

$$\log_a(n) = k \log_b(n), \text{con } k > 0$$

No entendí ni madres, pero la conclusión se parece mucho al principio de invariancia. Luego, no nos importa la base que escojamos 👽.

Esto no vale para $\log_{f(n)}(n)$, ni para $k^{\log(n)}$, ni ninguna variante rara. Tampoco para $a, b < 1$

Transitividad
Sean $f, g, h : \mathbb{N} \to \mathbb{R}^+$, si
- $f(n) \in O(g(n))$
- $g(n) \in O(g(n))$

Entonces $f(n) \in O(g(n))$

dice que también es "reflexiva" ¡ma perché!

Demostrar que no pertecene
Se hace por contradicción, no lo voy a estudiar de no ser necesario. En el primer capítulo hay un resúmen de cómo hacer estas demos.

Limmit Rule
1. $\lim_{n\to\infty} \frac{f(n)}{g(n)} \in \mathbb{R}^+ \Rightarrow f(n) \in O(g(n)) \land g(n) \in O(f(n))$
2. $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Rightarrow f(n) \in O(g(n)) \land g(n) \notin O(f(n))$
3. $\lim_{n\to\infty} \frac{f(n)}{g(n)} = +\infty \Rightarrow f(n) \notin O(g(n)) \land g(n) \in O(f(n))$

Notación Omega

$$\Omega(f(n)) = \{t : \mathbb{N} \to \mathbb{R}^{\geq 0} \mid (\exists d \in \mathbb{R}^+)(\forall n \in \mathbb{N})[t(n) \geq df(n)]\}$$

Duality rule

$$t(n) \in O(f(n)) \Leftrightarrow f(n) \in \Omega(t(n))$$

Particularidades
No entendí ni madres, en el libro página 86 explican algo, pero tiene toda la pinta de que se entiende mejor con ejercicios y ejemplos concretos. Por ahora, hay que pensarlo cómo "el límite inferior", o mejor dícho, "el mejor de los casos" de una ejecución.

Notación Theta

$$\Theta(f(n)) = O(f(n)) \cap \Omega(f(n))$$

Es decir

$$\Theta(f(n)) = \{t : \mathbb{N} \to \mathbb{R}^{\geq 0} \mid (\exists d, c \in \mathbb{R}^+)(\forall n \in \mathbb{N})[df(n) \leq t(n) \leq cf(n)]\}$$

Limit Rule
1. $\lim_{n\to\infty} \frac{f(n)}{g(n)} \in \mathbb{R}^+ \Rightarrow f(n) \in \Theta(f(n))$
2. $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Rightarrow f(n) \in O(g(n)) \land g(n) \notin \Theta(f(n))$
3. $\lim_{n\to\infty} \frac{f(n)}{g(n)} = +\infty \Rightarrow f(n) \in \Omega(g(n)) \land g(n) \notin \Theta(f(n))$