Analisis y diseño de algoritmos

Juan Gutiérrez

September 2019

1 Introducción

Definición 1.1. *Algoritmo: Procedimiento computacional bien definido que recibe un valor o conjunto de valores como entrada y produce un valor, o conjunto de valores como salida*

Herramienta para resolver un problema computacional

Ejemplo 1.1. *Problema de ordenación.*

Entrada: secuencia de n *números* $< a_1, a_2, \ldots a_n >$

Salida: Una permutacion $< a'_1, a'_2, \ldots a'_n >$ *de la secuencia de entrada, tal que* $a'_1 \leq a'_2 \leq \ldots \leq a'_n$

Instancia: caso particular del problema, por ejemplo, en el problema anterior, una instancia seria $< 31, 41, 59, 26, 41, 58 >$

Un algoritmo está *correcto* si para cada entrada, para con la correspondiente salida. En ese caso, decimos que el algoritmo *resuelve* el problema

Ejemplos prácticos: Genoma humano, internet, criptografia, programacion lineal (optimizacion)

Tecnicas usadas son similares a los siguientes problemas:

- Camino minimo
- Subsecuencia comun mas larga
- Ordenacion topologica
- Convex hull (Cerradura convexa)

Aprenderemos tecnicas. Sin embargo, estas tecnicas no funcionan en problemas NP-dificiles.

Eficiencia: tiempo de ejecución y memoria.

Insertion sort ($c_1 n^2$) vs Merge sort ($c_2 n \lg(n)$), $c_1 < c_2$. A medida que n crece, las constantes no son relevantes.

Ejemplo 1.2. *Computador A:* 10^{10} *inst/s, correr Insertion sort, cuyo tiempo de ejecucion es* $2n^2$ *(es decir, con entrada de tamaño* n*, ejecuta* $2n^2$ *instrucciones),*

Computador B:10^7 *inst/s, correr Merge sort sort, cuyo tiempo de ejecucion es* $50n \lg(n)$

Ordenamos $n = 10^7$ números.
A demora $\frac{2(10^7)^2 inst}{10^{10} inst/s} = 20000s = 5.55hs$
B demora $\frac{50(10^7)\lg 10^7 inst}{10^7 inst/s} = 1162.67s = 0.322hs$
Si ordenamos $n = 10^8$ números, A demora 23 días, pero B demora 4 hrs.

Ejercicio 1.1. *Suponga que en una misma computadora se correr insertion Sort y Merge Sort, donde insertion sort corre en $8n^2$ pasos con entrada de tamaño n y Merge Sort corre en $64n \lg n$ pasos. Para que valores de n, es mas eficiente el insertionSort?*

Obs $\lg x = \log_2 x$; $\log x = \log_{10} x$
Obs $\lg x = \log x / \log 2$
Borrador: queremos $8n^2 < 64n \lg n$, luego $n < 8 \lg n$

n	$8 \lg n$
1	0
2	8
3	12.67
10	26.57
20	34.57
30	39,..
40	42.57
43	43.4
44	43.67

Demostración: Si $n = 1$ entonces $8n^2 > 64n \lg n$. Supongamos entonces que $n \geq 2$. Note que $n/\lg n$ es una función creciente. Luego, para $n \leq 43$, tenemos que $\frac{n}{\lg n} \leq \frac{43}{\lg 43} = 7.92.. < 8$. Y para $n \geq 44$, tenemos que $\frac{n}{\lg n} \geq \frac{44}{\lg 44} = 8.05.. > 8$. Luego, $8n^2 < 64n \lg n$ si y solo si $2 \leq n \leq 43$.

Ejercicio 1.2. *Cual es el menor valor de n tal que un algoritmo con tiempo de ejecucion $100n^2$ es más rapido que uno con tiempo de ejecucion 2^n en la misma máquina.*

Borrador: queremos $100n^2 < 2^n$, luego $\lg 100n^2 < n$
Demostración: Note que $2^n/100n^2$ es una función creciente. Entonces, cuando $n \leq 14$, tenemos que $2^n/100n^2 \leq 2^{14}/(100 \cdot 14^2) < 1$. Además, cuando $n \geq 15$, tenemos que $2^n/100n^2 \geq 2^{15}/(100 \cdot 15^2) > 1$.

Ejercicio 1.3. *Para cada funcion $f(n)$ y tiempo t en la siguiente tabla, determine el mayor tamaño de n de un problema que puede ser resuelto en tiempo t, suponiendo que el algoritmo para resolver el problema toma $f(n)$ μs*

$\lg 100n^2$	n
6.64...	1
8.64 ..	2
9.813 ..	3
13.2877..	10
...	...
14.25..	14
14.457..	15

	1 second	1 minute	1 hour	1 day	1 month	1 year	1 century
$\lg n$							
$\sqrt{n}$							
n							
$n \lg n$							
n^2							
n^3							
2^n							
$n!$							

Recordar: $\log_a(b) = x$ si y solo si $b = a^x$
Recordar: $\log_a(x^y) = y \log_a(x)$
Recordar: $\log_a(xy) = \log_a(x) + \log_a(y)$
Para 1s.

- Borrador: Queremos que $\lg(n) \leq 10^6$, entonces se debe cumplir que $n \leq 2^{10^6}$.

  Demostración: Para $n \leq 2^{10^6}$, tenemos que $\lg n \leq \lg 2^{10^6} = 10^6$. Como $\lg n$ es una funcion creciente, entonces cuando $n > 2^{10^6}$, tenemos que $\lg n > \lg 2^{10^6} = 10^6$. Luego 2^{10^6} es el valor pedido.

- Borrador: Queremos que $\sqrt{n} \leq 10^6$, entonces se debe cumplir que $n \leq (10^6)^2 = 10^{12}$.

  Demostración: Para $n \leq 10^{12}$, tenemos que $\sqrt{n} \leq \sqrt{10^{12}} = 10^6$. Como

$\sqrt{n}$ es una funcion creciente, entonces cuando $n > 10^{12}$, tenemos que $\sqrt{n} > \sqrt{10^{12}} = 10^6$. Luego 10^{12} es el valor pedido.

- Borrador: Queremos que $n \lg n \leq 10^6$. Graficando o tabulando obtenemos $n = 6.24 \cdot 10^4$

  Demostración: Para $n \leq 6.24 \cdot 10^4$, tenemos que $\lg n \leq \lg(6.24 \cdot 10^4) = \lg(6.24) + \lg 10^4 = \lg(6.24) + 4 \lg 10 = 15.929258409$. Luego $n \lg n = 993985.724698947 \leq 10^6$. Falta: probar que para $n > 6.24 > 10^4$ se tiene que $n \lg n > 10^6$

2 Sumatorias

Cuando un algoritmo tiene una instrucción iterativa (for o while), podemos expresar su tiempo de ejecución como la suma de los tiempos en cada iteración. Es importante conocer conceptos de sumatorias y encontrar límites superiores para ellas (upper bounds)

2.1 Fórmulas y propriedades básicas

Dada una secuencia $a_1, a_2, \ldots, a_n$ de números, donde n es un entero no negativo, podemos escribir la suma finita $a_1 + a_2 + \cdots + a_n$ como

$$\sum_{k=1}^{n} a_k$$

.

Si $n = 0$ entonces el valor de la sumatoria es definida como 0.

Dada una secuencia infinita $a_1, a_2, \ldots$ de números, podemos escribir la suma infinita $a_1 + a_2 + \cdots$ como

$$\sum_{k=1}^{\infty} a_k = \lim_{n\to\infty} \sum_{k=1}^{n} a_k.$$

Si el límite no existe, la serie *diverge*, si no, esta *converge*.

Linearidad

Para cada número real c y cualesquier secuencias $a_1, a_2, \ldots, a_n$ y $b_1, b_2, \ldots, b_n$, tenemos que

$$\sum_{k=1}^{n} (ca_k + b_k) = c \sum_{k=1}^{n} a_k + \sum_{k=1}^{n} b_k$$

Series aritméticas

Son series en donde la resta de cada dos términos consecutivos es la misma. Ejemplo:

$$\sum_{k=1}^{n} k = 1 + 2 + \cdots + n = \frac{n(n+1)}{2}$$

Suma de cuadrados y cubos

$$\sum_{k=0}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$$

$$\sum_{k=0}^{n} k^3 = \frac{n^2(n+1)^2}{4} = (\frac{n(n+1)}{2})^2$$

Serie geométrica

Son series en donde la división de cada dos términos consecutivos es la misma.

Por ejemplo, para cada número real $x \neq 1$,

$$\sum_{k=0}^{n} x^k = 1 + x + x^2 + \cdots + x^n = \frac{x^{n+1} - 1}{x - 1}.$$

Obs: cuando $x = 2$, tenemos $1 + 2 + 2^2 + \ldots + 2^n = 2^{n+1} - 1$.

Tambien $2^n = 1 + 2 + 2^2 + \cdots + 2^{n-1} + 1$

Cuando $|x| < 1$, se cumple

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1 - x}.$$

De lo anterior, podemos concluir que (ejercicio)

$$\sum_{k=0}^{\infty} kx^k = \frac{x}{(1 - x)^2}.$$

Obs: suponemos que $0^0 = 1$.

Serie ármónica

Es la serie

$$H_n = \sum_{k=1}^{n} \frac{1}{k} = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n}$$

$H(n)$ es llamado *número armónico*. Se sabe que $H(n) = \ln n + c$ para una constante $c \approx 0.5772$.

Series telescópicas

Para cada secuencia $a_0, a_1, \ldots, a_n$,

$$\sum_{k=1}^{n}(a_k - a_{k-1}) = a_n - a_0.$$

También,

$$\sum_{k=0}^{n-1}(a_k - a_{k-1}) = a_0 - a_n.$$

Por ejemplo,

$$\sum_{k=1}^{n-1}\frac{1}{k(k+1)} = \sum_{k=1}^{n-1}(\frac{1}{k} - \frac{1}{k+1}) = 1 - \frac{1}{n}$$

Convirtiendo productorias a sumatorias

Podemos escribir el producto finito $a_1 a_2 \cdots a_n$ como $\prod_{k=1}^{n} a_k$ (cuando $n = 0$ el valor de producto es definido como 1).

Podemos hacer la siguiente conversión:

$$\lg(\prod_{k=1}^{n} a_k) = \sum_{k=1}^{n} \lg a_k.$$

2.2 Acotando sumatorias

Existen muchas técnicas para acotar sumatorias. Estas nos servirán al acotar las sumatorias que expresan los tiempos de ejecución de un algoritmo.

Inducción

Probaremos por inducción en n que $\sum_{k=1}^{n} k \leq \frac{1}{2}(n+1)^2$. Es fácil ver que la proiedad es cierta para $n = 1$ (caso base). Asumiremos por hipótesis de inducción que funciona para $n-1$. Osea, $\sum_{k=1}^{n-1} k \leq \frac{1}{2}(n-1+1)^2$.

Luego

$$\begin{aligned}
\sum_{k=1}^{n} &= \sum_{k=1}^{n-1} + n \\
&\leq \frac{1}{2}n^2 + n \text{ (por hipotesis de induccion)} \\
&\leq \frac{1}{2}n^2 + n + \frac{1}{2} \\
&= \frac{1}{2}(n+1)^2
\end{aligned}$$

Probaremos por inducción en n que $\sum_{k=0}^{n} 3^k \leq c3^n$ para alguna constante c. Es fácil ver que cuando $n = 0$, tenemos que $\sum_{k=0}^{n} 3^k = 1$. Luego para cualquier $c \geq 1$ la proposición es cierta. Por hipótesis de inducción, tenemos que existe c tal que $\sum_{k=0}^{n-1} 3^k \leq c3^{n-1}$. Luego, cuando $c \geq 3/2$, tenemos que

$$\begin{aligned} \sum_{k=0}^{n} 3^k &= \sum_{k=0}^{n-1} 3^k + 3^n \\ &\leq c3^{n-1} + 3^n \\ &= (\frac{1}{3} + \frac{1}{c})c3^n \\ &\leq c3^n. \end{aligned}$$

Portanto, basta tomar $c = 3/2$.

Acotando términos

Podemos acotar superiomente cada término de una serie, por ejemplo

$$\sum_{k=1}^{n} k \leq \sum_{k=1}^{n} n = n^2$$

En general, sea $a_{\text{max}} = \max_{1 \leq k \leq n} a_k$, tenemos que

$$\sum_{k=1}^{n} a_k \leq \sum_{k=1}^{n} a_{\text{max}} = n \cdot a_{\text{max}}$$

Supongamos que $a_{k+1}/a_k \leq r$ para todo $k \geq 0$, con $0 < r < 1$. Tenemos que

$$\begin{aligned} \sum_{k=0}^{n} a_k &\leq \sum_{k=0}^{\infty} a_0 r^k \\ &= a_0 \sum_{k=0}^{\infty} r^k \\ &= a_0 \cdot \frac{1}{1-r} \end{aligned}$$

Por ejemplo, acotaremos $\sum_{k=1}^{\infty}(k/3^k) = \sum_{k=0}^{\infty}((k+1)/3^{k+1})$. Tenemos $a_0 = 1/3$, $\frac{(k+2)/3^{k+2}}{(k+1)/3^{k+1}} = \frac{1}{3} \cdot \frac{k+2}{k+1} \leq \frac{2}{3} = r$.

Luego

$$\sum_{k=1}^{\infty}(k/3^k) \leq \frac{1}{3} \cdot \frac{1}{1-2/3} = 1$$

Dividiendo sumatorias

Acotaremos $\sum_{k=1}^{n} k$. Note que (asuma que n es par)

$$\begin{aligned}
\sum_{k=1}^{n} k &= \sum_{k=1}^{n/2} k + \sum_{k=n/2+1}^{n} k \\
&\geq \sum_{k=1}^{n/2} 0 + \sum_{k=n/2+1}^{n} (n/2) \\
&= (n/2)^2
\end{aligned}$$

Para cualquier constante $k_0 > 0$, tenemos que

$$\begin{aligned}
\sum_{k=0}^{n} a_k &= \sum_{k=0}^{k_0-1} a_k + \sum_{k=k_0}^{n} a_k \\
&= c + \sum_{k=k_0}^{n} a_k.
\end{aligned}$$

donde c es una constante.

Acotaremos $\sum_{k=0}^{\infty} k^2/2^k$. Note que cuando $k \geq 3$, $\frac{(k+1)^2/2^{k+1}}{k^2/2^k} = \frac{(k+1)^2}{2k^2} \leq \frac{8}{9}$. Luego

$$\begin{aligned}
\sum_{k=0}^{\infty} k^2/2^k &= \sum_{k=0}^{2} k^2/2^k + \sum_{k=3}^{\infty} k^2/2^k \\
&\leq \sum_{k=0}^{2} \frac{k^2}{2^k} + 9/8 \sum_{k=0}^{\infty} (8/9)^k \\
&= \sum_{k=0}^{2} \frac{k^2}{2^k} + 9/8 \cdot (1/(1-8/9)) \\
&= c
\end{aligned}$$

para alguna constante c.

Acotaremos $H_n = \sum_{k=1}^{n} \frac{1}{k}$. Note que

$$\begin{aligned}
\sum_{k=1}^{n} \frac{1}{k} &\leq \sum_{i=0}^{\lfloor \lg n \rfloor} \sum_{j=0}^{2^i-1} \frac{1}{2^i + j} \\
&\leq \sum_{i=0}^{\lfloor \lg n \rfloor} \sum_{j=0}^{2^i-1} \frac{1}{2^i} \\
&= \sum_{i=0}^{\lfloor \lg n \rfloor} 1 \\
&\leq \lg n + 1
\end{aligned}$$

3 Grafos

Un *grafo dirigido* es un par (V, E), donde V es un conjunto finito y E es una relación binaria en V. El conjunto V es llamado *conjunto de vértices* de G, y sus elementos son llamados *vértices*. El conjunto E es llamado *conjunto de aristas*.

Si E consiste en pares de vértices no ordenados, el grafo $G = (V, E)$ es denominado *grafo no dirigido*. Es decir, las aristas son conjuntos $\{u, v\}$, donde $u, v \in V$. Usamos la notación $(u, v) = (v, u)$.

Si (u, v) es una arista, decimos que (u, v)*incide* en u y en v. Además decimos que *sale* de u y *entra* en v. También decimos que v es *adyacente* a u y si G es dirigido escribimos $u \to v$.

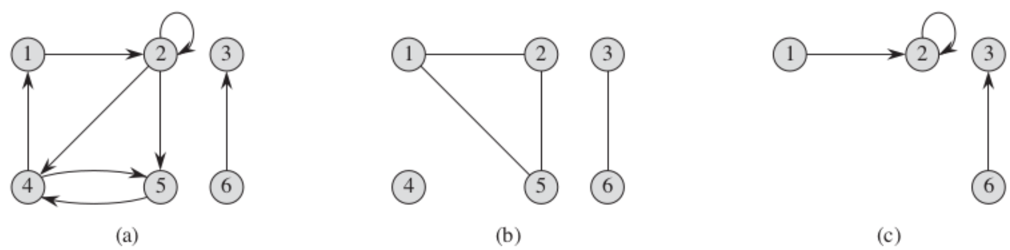

Figure B.2 Directed and undirected graphs. **(a)** A directed graph $G = (V, E)$, where $V = \{1, 2, 3, 4, 5, 6\}$ and $E = \{(1, 2), (2, 2), (2, 4), (2, 5), (4, 1), (4, 5), (5, 4), (6, 3)\}$. The edge $(2, 2)$ is a self-loop. **(b)** An undirected graph $G = (V, E)$, where $V = \{1, 2, 3, 4, 5, 6\}$ and $E = \{(1, 2), (1, 5), (2, 5), (3, 6)\}$. The vertex 4 is isolated. **(c)** The subgraph of the graph in part (a) induced by the vertex set $\{1, 2, 3, 6\}$.

Figure 1: Tomada del libro Cormen, Introduction to Algorithms

El *grado* de un vértice en un grafo no dirigido es el número de aristas incidentes a él. Un vértice con grado 0 es denominado*isolado*. En un grafo dirigido, el *grado de salida* de un vértice es el número de aristas que salen de él, y el *grado de entrada* es el número de vértices que entran a el.

Un *camino* de un vértice a un vértice u' en un grafo $G = (V, E)$ es una secuencia$< v_0, v_1, \ldots, v_k >$ de vértices tal que $u = v_0$, $u' = v_k$ y$(v_{i-1}, v_i) \in E$ para $i = 1, 2, \ldots, k$. La *longitud* de un camino es el número de aristas en el. Si existe camino de u a u', decimos que u' es *alcanzable* desde u. Un camino es *simple* si todos sus vértices son distintos.

En un grafo no dirigido, un camino $< v_0, v_1, \ldots, v_k >$ es un *ciclo* si $k > 0$, $v_0 = v_k$ y todas sus aristas son diferentes. Un ciclo es *simple* si $v_1, v_2, \ldots, v_k$ son diferentes entre sí.

Un grafo no dirigido es *conexo* si cada vértice es atingible desde cualquier otro vértice. Los *componentes* de un grafo no dirigido son los subgrafos maximales conexos.

Propiedad: En un grafo $G = (V, E)$, $\sum_{v \in V} d(v) = 2|E|$.

Propiedad: En un grafo conexo no dirigido $G = (V, E)$, $|E| \geq |V| - 1$.

4 Árboles

Un *árbol* es un grafo no dirigido, conexo y acíclico. Si dicho grafo es solo acíclico mas no necesariamente conexo, se le llama *bosque*.

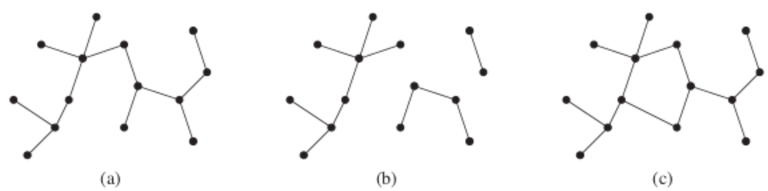

Figure B.4 **(a)** A free tree. **(b)** A forest. **(c)** A graph that contains a cycle and is therefore neither a tree nor a forest.

Figure 2: Tomada del libro Cormen, Introduction to Algorithms

Teorema 4.1. *Sea $G = (V, E)$ un grafo, los siguientes enunciados son equivalentes.*

1. *G es un árbol*
2. *Cualquiera dos vértices en G están conectados por un único camino simple*
3. *G es conexo, y si alguna arista es removida, el grafo resultante es desconexo*
4. *G es conexo y $|E| = |V| - 1$*
5. *G es acíclico y $|E| = |V| - 1$*
6. *G es acíclico, y si alguna arista es añadida a G, el grafo resultante contiene un ciclo.*

4.1 Árboles enraizados y árboles ordenados

Un *árbol enraizado* es un árbol con un vértice especial denominado *raíz*. Nos referimos a los vértices del árbol como *nodos*. Sea x un nodo en un árbol enraizado T con raíz r. Un *ancestro* de x es cualquier nodo en el único camino de r a x. Si y es un ancestro de x entonces x es un *descendiente* de y. Si y es un ancestro de x y $x \neq y$, entonces y es un *ancestro propio* de x yx es un descendiente propio de y. El subárbol enraizado en x es el árbol inducido por todos los descendientes de x.

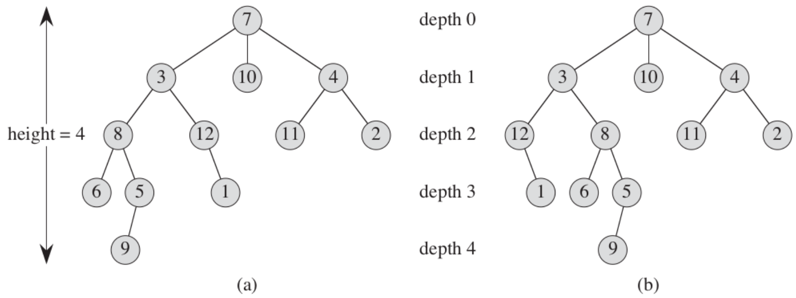

Figure 3: Tomada del libro Cormen, Introduction to Algorithms

El ancestro y adyacente a un nodo x es denominado *padre* de x, y x es denominado *hijo* de y. Si dos nodos tienen el mismo padre, son denominados *hermanos*. Un nodo sin hijos es denominado *hoja*. Un nodo que no es hoja es denominado *nodo interior*.

La longitud del camino de r hacia x es llamada la *profundidad* de x. Un *nivel* consiste en todos los nodos a igual profundidad. La *altura* de un nodo es el número de aristas en el camino máximo desde el nodo hacia alguna hoja. La altura de un árbol es la altura de su raíz.

4.2 Árboles binarios

Un *árbol binario* T es definido recursivamente según:

- si T no tiene nodos entonces es un árbol binario
- si T tiene nodos, está compuesto por un nodo *raíz*, un árbol binario llamado *subárbol izquierdo* y un árbol binario llamado *subárbol derecho*

El árbol binario sin nodos es llamado *nulo* o *vacío* y denotado por NIL. Si el subárbol izquierdo es no vacío, su raíz es llamada *hijo izquierdo*. Análogamente definimos *hijo izquierdo*.

Si cada nodo tiene exactamente dos hijos, un árbol binario es denominado *lleno*. Si todas las hojas tienen la misma profundidad, dicho árbol es denominado *completo*.

Propiedad: La altura de un árbol binario completo con k hojas es $\lg k$

Propiedad: La altura de un árbol binario completo con n vértices es $\lg{(n+1)} - 2$

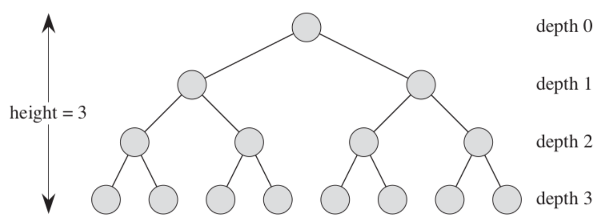

Figure B.8 A complete binary tree of height 3 with 8 leaves and 7 internal nodes.

Figure 4: Tomada del libro Cormen, Introduction to Algorithms