Diseno y Analisis de Algoritmos

Javier Liberman

Primavera 2015

Este apunte es realizado en base a las clases de Gonzalo Navarro

1 Cotas Inferiores de Problemas

Para calcular cotas inferiores de problemas, es decir, mostrar que un problema es dificil, se puede ocupar tres tecnicas: **Adversario, Reduccion o Teoria de la informacion.**

1.1 Tecnica del adversario

Se considera que el algoritmo es ejecutado contra un adversario que intenta que el algoritmo se desempene lo peor posible. El adversario no puede hacer 'trampa', o sea, no puede contradecir las indicaciones del problema.

1.1.1 Busqueda en un arreglo ordenado

La menor cantidad de accesos que se puede hacer es n. El adversario puede guardarse el elemento que se busca hasta el final.

1.1.2 Encontrar el minimo en $A[1, n]$

Tanto si se comparan secuencialmente, como si se comparan al estilo de un torneo de tenis, se realizan $n - 1$ comparaciones. Si se modela el problema como un grafo, donde cada nodo es un elemento, y una arista es una comparacion, resulta claro que se necesita al menos un grafo conexo. Y para conectar un grafo de n nodos se necesitan $n - 1$ aristas. Suponiendo una tupla (a, b, c) donde:

$$a = \text{Numero de elementos nunca comparados} \tag{1}$$

$$b = \text{Numero de elementos comparados alguna vez y siempre menores} \tag{2}$$

$$c = \text{Numero de elementos comparados alguna vez y alguna vez mayores} \tag{3}$$

Todo algorimo que desee responder quien es el mayor, parte y termina

$$(n, 0, 0) \to (0, 1, n - 1)$$

Aquellas opciones tachadas son las que el adversario elige evitar sin ser inconsistente. Si se sigue el camino $(a, a) \to (a, b)$ se obtiene el algoritmo de elegir el primero y luego iterar comparando sobre el resto. Si se sigue el camino (a, a) hasta que no queden mas a. Luego (b, b) hasta que no queden mas b, se obtiene el algoritmo del torneo de tenis.

Table 1: Tabla Adversario

	a	b	c
a	$(a-2, b+1, c)$	$(a-1, b, c+1)$	$(a-1, b+1, c), \cancel{(a-1, b, c+1)}$
b		$(a, b-1, c+1)$	$(a, b, c), \cancel{(a, b-1, c+1)}$
c			(a, b, c)

1.1.3 Encontrar el maximo y el minimo de $A[1, n]$

Ya se sabe que se puede encontrar el maximo con $n-1$ comparaciones. Y considerando que el arreglo es mas grande que uno, el maximo no es el minimo, luego quedan $n-1$ elementos y se puede encontrar el minimo con $n-1$ comparaciones. Dado que se construyo el algoritmo para encontrar el maximo y minimo con $2n-3$ comparaciones, se puede afirmar que existe una cota superior igual a $2n-3$

Se define la tupla (a, b, c, d) como:

$$a = \text{Numero de elementos nunca comparados} \tag{4}$$

$$b = \text{Numero de elementos comparados alguna vez y siempre menores} \tag{5}$$

$$c = \text{Numero de elementos comparados alguna vez y siempre mayores} \tag{6}$$

$$d = \text{Numero de elementos comparados alguna vez y alguna vez mayores y otra menores} \tag{7}$$

El algoritmo debe partir y terminar

$$(n, 0, 0, 0) \to (0, 1, 1, n-2)$$

Table 2: Tabla Adversario

	a	b	c	d
a	$(a-2, b+1, c+1, d)$	$(a-1, b, c, d+1), (a-1, b, c+1, d)$	$\cancel{(a-1, b, c, d+1)}, (a-1, b+1, c, d)$	$(a-1, b+1, c, d), (a-1, b, c+1, d)$
b		$(a, b-1, c, d+1)$	$(a, b, c, d), \cancel{(a, b-1, c-1, d+2)}$	$(a, b, c, d), \cancel{(a, b-1, c, d+1)}$
c			$(a, b, c-1, d+1)$	$(a, b, c, d), \cancel{(a, b, c-1, d+1)}$
d				(a, b, c, d)

Se deduce de la tabla que una cota inferior es $\lceil n/2 \rceil + n - 2$. Haciendo un torneo y luego entre los perdedores y otro torneo entre los ganadores se obtiene el algoritmo produciendo una cota superior. Luego como la cota inferior y superior obtenidas son iguales, se deduce que ambas cotas son lo mejor posible.

1.1.4 Maximo y segundo maximo

Esta calculado que lo mejor que se puede hacer es:

$$n - 1 + log_2(n)$$

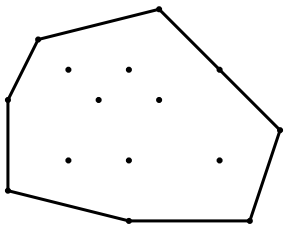
Y se realiza haciendo un torneo y luego un torneo entre los perdedores con el ganador. No es posible hacer una tabla para este algoritmo. La estrategia que se utiliza es la de la teoria de la informacion.

2 Reduccion

Si el problema A tiene una cota inferior de $f(n)$ y se puede reducir a el problema B en tiempo $o(f(n))$ entonces B tiene una cota inferior de $\Omega(f(n))$. Siempre se debe reducir el problema conocido al problema desconocido y **no al revez** (Para calcular cotas inferiores)

2.0.5 Capsula convexa - Convex hull

En el problema de la capsula convexa, se tiene un set de puntos en 2D y se debe encontrar la capsula que los contiene en sentido horario.

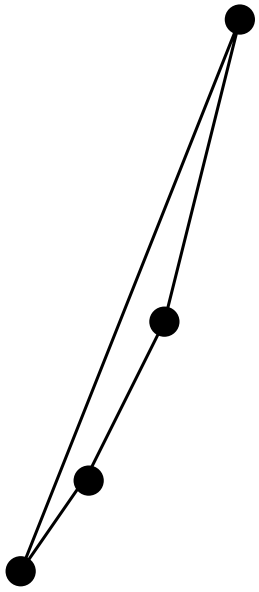


Sabiendo que ordenar es $\Omega(nlog(n))$ se puede aplicar la reduccion suponiendo que se tiene un arreglo:

$$a_1, a_2, a_3, ..., a_n$$

Se puede transformar en puntos 2D de la siguiente forma:

$$(a_1, a_1^2), (a_2, a_2^2), (a_3, a_3^2), ..., (a_n, a_n^2)$$



Es claro que se obtiene una capsula que contiene todos los puntos.

2.1 Colas de prioridad

Con un heap binario se tienen las siguientes operaciones y costos:

$$heapifyO(n) \tag{8}$$

$$insertO(log(n)) \tag{9}$$

$$extractMinO(log(n)) \tag{10}$$

Con un heap fibonacci se tiene:

$$heapifyO(n) \tag{11}$$

$$insertO(1) \tag{12}$$

$$extractMinO(log(n)) \tag{13}$$

Es posible tener una cola de prioridad que:

$$heapifyO(n) \tag{14}$$

$$insertO(log(n)) \tag{15}$$

$$extractMinO(1) \tag{16}$$

O una cola que haga:

$$heapifyO(1) \tag{17}$$

$$insertO(1) \tag{18}$$

$$extractMinO(1) \tag{19}$$

Ninguna de estas dos son posibles. Si fueran posibles, se podria ordenar en tiempo n. Bastaria con realizar heapify sobre el arreglo y luego ir extrallendo el minimo y se tendria el arreglo ordenado.

2.2 Problema de 3SUM

Dados n numeros $x_1, x_2, ..., x_n$ se pueden elegir a, b, c con repeticion permitida tal que $a+b+c=0$. Vale notar que no pueden haber ceros dentro de los n numeros, sino, resolver el problema seria trivial.

El problema tiene orden n^3 con fuerza bruta. Si se suman todos los pares y luego se recorre la lista para encontrar un inverso el problema tiene orden n^2

Mucho tiempo existio la conjetura que el orden de este problema era n^2. Gramlund & Pettie descubren que el orden de este problema es $\Omega(\frac{2}{3}\frac{n^2}{\frac{log(n)}{log(log(n))}})$

La nueva conjetura es que el orden del problema es $\Omega(n^2-\epsilon)$ para todo ϵ mayor que 0.

Se dice en demostraciones que un problema es 3SUM-hard para decir que es al menos tan dificil como 3SUM. Se asumia que se referia a que el problema era n^2. A pesar de que no era, las demostraciones no quedan invalidas.

2.3 3-colinear

Dados n puntos en el plano, existen 3 puntos que sean colineales?

Sea $x_1, ..., x_n$ un problema de 3SUM. Por cada x_i se crean 3 puntos.

$$(0, x_i) \tag{20}$$

$$(1, \frac{x_i}{2}) \tag{21}$$

$$(2, x_i) \tag{22}$$

Sean 3 puntos colineales: $(0, a), (1, b), (2, c)$
$b = \frac{a+b}{2}$

$$(0, a)x_i \tag{23}$$

$$(1, b)\frac{x_j}{2} \tag{24}$$

$$(2, c)x_k \tag{25}$$

$$\frac{-x_j}{2} = \frac{x_i + x_k}{2} \tag{26}$$

$$x_i + x_j + x_k = 0 \tag{27}$$

Se transforma 3SUM a 3-Colinear en tiempo lineal. Y con 3-Colinear resuelto, se obtiene la solucion de 3SUM inmediatamente. Considerando tambien que 3SUM no es posible resolverlo en menos de $\Omega(n)$ dado que por el adversario, se necesita ver al menos todo el input.
La reduccion solo funciona si el costo de la transformacion es una cota inferior conocida del otro problema.

3 Teoria de la Informacion

Es facil analogar el principio de la teoria de la informacion con el juego de las 20 preguntas. Si conozco 9 personas, no me pueden ganar siempre con 3 preguntas.
Para formalizar, se puede suponer que se tiene un arreglo ordenado desde 1 a n. Si solamente se realizan preguntas binarias se tiene que poder responder desde 1 hasta n. Es decir al menos se tienen que realizar $\lceil (log_2 n) \rceil$

3.1 Sort

Un algoritmo de ordenamiento haga lo que haga compara elementos de un arreglo, va haciendo intercambios y en algun momento termina. El arreglo contiene siempre los mismo numeros. Son los numeros del 1 al n reordenados. Si se tiene un arreglo con dos permutaciones π_1, π_2, cuando se le aplica la inversa de la permutacion se obtiene la identidad. Como hay $n!$ permutaciones distintas, se debe hacer $log_2(n)!$ preguntas binarias para elegir los cambios a hacer en el arreglo.