Universidad de Chile
Departamento de Ciencias de la Computación
CC4102 - Diseño y Análisis de Algoritmos

Auxiliar 2 - "Cotas Inferiores y Memoria Externa"

Profesor: Gonzalo Navarro
Auxiliar: ~~Manuel~~ Ariel Cáceres Reyes

21 de Agosto del 2017

Cotas inferiores

P1. Considere el problema de mezclar 2 listas ordenadas de igual tamaño, $X[1,\dots n], Y[1,\dots n]$ de modo que el resultado final este también ordenado. En este problema estaremos interesados en las preguntas del estilo: ¿$X[i] < Y[j]$?

a) Muestre que el problema es $\mathcal{O}(2n-1)$

b) Muestre, usando árbol de decisión, que el problema es $\Omega\left(2n - \frac{1}{2}\log n\right)$

c) Muestre, usando adversario, que el problema no puede ser resuelto en menos de $2n-1$ comparaciones

Memoria Externa

P2. **Mayoría**

Considere una lista L de $N \gg M$ elementos en memoria externa. Diseñe un algoritmo eficiente para encontrar un elemento que tenga la mayoría absoluta (es decir, que aparezca más de $N/2$ veces), y reportar su frecuencia. Si no existe tal elemento debe reportarse `no`.

P3. **Skyline**

Se nos entrega un conjunto P de N puntos en el plano en *posición general*; es decir, ningún par de puntos comparte el mismo valor en alguna de sus coordenadas. Considere que $N \gg M$. Dado un punto p en el plano , llamaremos $p[i]$ a su i-ésima coordenada. Un punto p_1 domina a otro punto p_2 (denotado como $p_1 \prec p_2$) si se cumple $p_1[i] < p_2[i], \forall i = 1, 2$.
Se nos pide calcular el *skyline* de P , denotado como $SKY(P)$, que incluye todos los puntos de P que no son dominados por ningún otro:

$$SKY(P) = \{p \in P |\ \nexists p' \in P, p' \prec p\}$$

a) Resuelva el problema usando $\mathcal{O}(n \log_m n)$ operaciones de I/O.

b) **Propuesto:** Resuelva el problema, ahora para puntos en el espacio. Llegue a la misma cota para la cantidad de operaciones de I/O que en el caso anterior.

"No man has a good enough memory to be a successful liar"
Abraham Lincoln

Soluciones

P1.

Teorema 1. 💎
Todo árbol binario con n hojas tiene altura $h \geq \log_2 n$

a) El algoritmo de "merge" ocupado en "MergeSort" realiza en el peor caso $2n-1$ comparaciones (una por cada elemento que pone en la lista de los ordenados). ⚡

b) En total tenemos $2n$ elementos, pero no todas las $(2n)!$ permutaciones son posibles outputs del algoritmo (pues deben respetar que los órdenes en las listas originales se mantengan).

¿Cuántos outputs posibles hay entonces? 🤔
Para contarlos notemos que basta elegir las posiciones en las cuales se encuentran los elementos de una de las listas (si ya están escogidos el resto tiene su posición determinada) lo que se puede hacer de $\binom{2n}{n}$ formas distintas, siendo por tanto este el número de outputs posibles.
Si ponemos estos outputs en las hojas de un árbol de decisión, tenemos que por el teorema 💎su altura será $\geq \log \binom{2n}{n}$.

Usando ahora la aproximación de Stirling $(n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n)$ llegamos a que $\log \binom{2n}{n} \sim 2n - \frac{1}{2}\log n - \frac{1}{2}\log \pi$, luego el problema es $\Omega(2n - \frac{1}{2}\log n)$

c) Por $\Rightarrow\Leftarrow$ existe un algoritmo A que realiza $< 2n-1$ comparaciones.

El adversario 😈construye el siguiente input:

- $X \rightarrow$ lista cuyos elementos son $x_i = 2i - 1$
- $Y \rightarrow$ lista cuyos elementos son $y_i = 2i$

Si ejecutamos el algoritmo A sobre el input anterior encontraremos un i^* tal que x_{i^*} no fue comparado con ambos y_{i^*-1} e y_{i^*} (si todos los i lo cumplieran se habrían hecho $2n-1$ comparaciones (al menos)).
Supongamos sin mucha pérdida de generalidad que solo no fue comparado con y_{i^*}, entonces en el siguiente input:

- $X \rightarrow$ lista cuyos elementos son $x_i = 2i - 1$, excepto $x_{i^*} = 2i^*$
- $Y \rightarrow$ lista cuyos elementos son $y_i = 2i$, excepto $y_{i^*} = 2i^* - 1$

El algoritmo recibe exactamente las mismas respuestas a las mismas preguntas que se hizo sobre el input anterior (pues el orden relativo de los elementos no se altera y no se compara x_{i^*} con y_{i^*}). Por lo tanto tengo 2 inputs válidos de listas para los cuales A retorna el mismo resultado, pero no lo es, pues en el primero $x_{i^*} < y_{i^*}$ y en el segundo $x_{i^*} < y_{i^*}$ lo que es una $\Rightarrow\Leftarrow$.

"No man has a good enough memory to be a successful liar"
Abraham Lincoln

P2. Mayoría

El siguiente algoritmo que escanea el input 2 veces por bloques (por lo que cuesta $\mathcal{O}(n)$ operaciones de I/O) resuelve el problema:

```
i ← 0
for x ∈ L do
    if i == 0 then
        m ← x
        i ← 1
    end
    else
        if x == m then
            ++i
        end
        else
            --i
        end
    end
end
```

```
j ← 0
for x ∈ L do
    if x == m then
        ++j
    end
end
if j > N/2 then
    return m, j
end
else
    return no
end
```

🎩🎩. Veamos que el primer ciclo presenta a m como un candidato a ser mayoría absoluta y el segundo ciclo verifica que este elemento realmente lo sea.
Si no existe mayoría absoluta el algoritmo responde correctamente *no* pues el segundo ciclo desecha el candidato m presentado por el primero.
Si existe un elemento K que es mayoría absoluta, en este caso imaginemos un índice l que no es parte del algoritmo pero se actualiza a medida que este avanza.l aumenta en 1 si el elemento escaneado es igual a K y disminuye en 1 si es distinto a K. Observemos que cuando $l > 0$

se cumple que el i del algoritmo es > 0 y el $m = K$ (pues significa que se han visto más Ks que otro símbolo). Finalmente cuando se terminan las iteraciones $l > 0$, pues K es mayoría absoluta, y por lo tanto $m = K$, el algoritmo lo postula como candidato y luego lo corrobora.

P3. Skyline

a) El siguiente algoritmo computa el SKY:

```
SKY ← ∅
y_min ← ∞
Sort_x(P)
for i = 1...N do
    (x, y) ← p_i
    if y < y_min then
        y_min ← y
        SKY.add(p_i)
    end
end
```

El mayor costo del algoritmo se realiza cuando se ordena por la coordenada x, por lo que el número de llamadas de I/O es $\mathcal{O}(n \log_m n)$. Veamos que cuando un punto es dominado por otro p se cumplirá que no es agregado a SKY, pues antes de analizarlo (dado que están ordenados por x) se debió haber seteado y_{min} como el y del punto o como algo menor que esto, por lo que no cumple la condición del **if**. Por otro lado, si el punto no es dominado, ningún punto menor que el en la coordenada x debería ser menor también en la coordenada y, por lo que a l ser procesado cumple la condición del **if** y es agregado al SKY.