

Pontificia Universidad Católica de Chile
Departamento de Ciencia de la Computación
IIC2283 - Diseño y Análisis de Algoritmos

Ayudantía 3 - Cotas Inferiores y Dividir y Conquistar

Agosto de 2025

Nicolás Buzeta, Paula Grune

Pregunta 1

Dado n tornillos y n tuercas, donde cada tornillo le hace a solo a una tuerca, encuentra los pares correspondientes. Los tornillos y tuercas no se pueden comparar directamente y solo puedes revisar si una tuerca es muy grande, chica, o igual a cualquier tornillo. Responde las siguientes preguntas:

a) Encuentra una cota inferior a la cantidad de comparaciones, para encontrar todos los pares correspondientes.

b) Ahora supone que solo queremos encontrar la mayoría de los pares correspondientes. Demuestra que esto tiene la misma cota inferior asintótica que encontrar todos los pares.

Solución

a) Vamos a modelar este problema como un árbol de decisión. Dado esto, necesitamos contar cuantas salidas, es decir hojas en nuestro árbol, pueden haber.

Para simplificar el problema, podemos imaginar que ponemos todos los tornillos en una mesa. Ahora podemos pensar que los tornillos se quedan fijos y solo tenemos que ver donde va cada tuerca. Dado esto, es trivial ver que hay $n!$ formas de asignar las tuercas. Dado esto, podemos concluir que nuestro árbol de decisión debe por lo menos $n!$ hojas:

$$L \geq n!$$

Como dice en el enunciado, se puede revisar si una tuerca es muy grande, chica, o igual a un tornillo. Dado esto, sabemos que cada nodo de nuestro árbol tiene 3 ramas. Dado esto, podemos decir que la altura del árbol debe cumplir:

$$\begin{aligned}
h &\geq \log_3(L) \\
&\geq \log_3(n!) \\
&\geq \frac{\ln(n!)}{\ln(3)} \\
&\geq \frac{\sqrt{2\pi n\left(\frac{n}{e}^n\right)}}{\ln(3)} && \text{Por Stirling's} \\
&\geq \frac{n\ln(n) - n\ln(e) + \frac{\ln(n)}{2} + \frac{\ln(2\pi)}{2}}{\ln(3)} \\
&\geq \Omega(n\ln(n))
\end{aligned}$$

Por lo que una cota inferior para la cantidad de comparaciones es $\Omega n\ln(n)$

b) Nuevamente debemos contar la cantidad de salidas que puede tener nuestro algoritmo.

$$\begin{aligned}
L &= \binom{n}{\frac{n}{2}} \cdot \binom{n}{\frac{n}{2}} \cdot \frac{n}{2}! \\
&= \left(\frac{n!}{\frac{n}{2}!\frac{n}{2}!}\right)^2 \cdot \frac{n}{2}! \\
&= \frac{(n!)^2}{\left(\left(\frac{n}{2}\right)!\right)^3}
\end{aligned}$$

$$
\begin{aligned}
\log L &= \log\left(\frac{(n!)^2}{\left(\left(\frac{n}{2}\right)!\right)^3}\right) \\
&= \log\left((n!)^2\right) - \log\left(\left(\left(\frac{n}{2}\right)!\right)^3\right) \\
&= 2\log\left(n!\right) - 3\log\left(\left(\frac{n}{2}\right)!\right) \\
&\geq 2\log\left(\frac{n}{e}\right)^n - 3\log\left(\left(\frac{n/2}{e}\right)^{n/2}\right) \\
&= 2n\log\left(\frac{n}{e}\right) - \frac{3n}{2}\log\left(\left(\frac{n/2}{e}\right)\right) \\
&= 2n\log\left(\frac{n}{e}\right) - \frac{3n}{2}\log\left(\left(\frac{n/2}{e}\right)\right) \\
&\geq \Omega(n\ln(n))
\end{aligned}
$$

Pregunta 2

Suponga una generalización del problema de merge, en donde uno quiere mezclar k arreglos ordenados de n/k elementos cada uno, para producir un arreglo ordenado que contiene los n elementos de dichos arreglos. Puede asumir que n es divisible por k. Muestre que $\Omega(n \log k)$ comparaciones es una cota inferior para dicho problema.

Hint 1: Demuestre que hay $\frac{n!}{((n/k)!)^k}$ permutaciones distintas, resultantes de mezclar k arreglos de tamaño n/k. Puede servirle usar el coeficiente multinomial, cuya definición es:

$$\binom{m}{v_1, v_2, \ldots, v_\ell} = \frac{m!}{v_1! \cdot v_2! \cdot \cdots \cdot v_\ell!}$$

que cuenta la cantidad de permutaciones de m elementos que hay tal que v_1 de esos elementos son de un cierto tipo, v_2 de otro tipo, y así siguiendo.

Solución

Esta es una generalización del problema visto en clase. Ahí se encontró una cota inferior para juntar dos arreglos ordenados. Ahora queremos encontrar una cota inferior para mezclar k arreglos, donde cada uno tiene $\frac{n}{k}$ elementos.

Para responder la pregunta, vamos a usar arboles de decisión tal como la respuesta anterior. Acá podemos pensar que cada nodo del árbol corresponde a una comparación de un par de elementos de arreglos distintos. Asumiendo, para simplificar el análisis, que todos los elementos son distintos, esto significa que cada nodo puede tener 2 ramas (elemento 1 es mayor o menos que elemento 2). Finalmente, sabemos que cada hoja de este árbol corresponde a una permutacion de n elementos, correspondiendo al arreglo mezclado.

Primero es clave ver que $n!$ no es una cota muy buena ya que los arreglos que nos entregan ya vienen ordenados. Dado lo anterior es mas facil analizar el problema desde otro punto de vista. En vez de pensar en como podemos formar todos estos arreglos, partimos con un salida cualquiera. Sabemos que esta salida va a tener n elementos. De estos n elementos, sabemos que hay $\frac{n}{k}$ elementos son del arreglo 1, $\frac{n}{k}$ elementos son del arreglo 2, etc. También es importante reconocer que no nos importa el orden de estos, ya que el orden viene dado por los arreglos originales (ya que viene ordenadas entre si). Dado esto, el problema se reduce a 'Cuantas formas hay de asignar $\frac{n}{k}$ elementos a 1 clase, $\frac{n}{k}$ a otra clase, etc'. Acá es donde entra el coeficiente multinomial:

$$\binom{m}{v_1, v_2, ..., v_3} = \frac{m!}{v_1! \cdot v_2! \cdots v_l!}$$

Usando esta ecuación, podemos contar la cantidad de formas que podemos asignar nuestros n elementos a k grupos:

$$\begin{aligned}\binom{n}{\frac{n}{k}, \frac{n}{k}, ..., \frac{n}{k}} &= \frac{n!}{\frac{n}{k}! \cdot \frac{n}{k}! \cdots \frac{n}{k}!} \\ &= \frac{n!}{\left(\frac{n}{k}!\right)^k}\end{aligned}$$

Por lo que la altura del árbol debe ser:

$$\begin{aligned}h &\geq \log\left(\frac{n!}{\left(\frac{n}{k}!\right)^k}\right) \\ &= \log(n!) - \log\left(\left(\frac{n}{k}!\right)^k\right) \\ &\geq \log(n!) - \log\left(\left(\left(\frac{n}{k}\right)^{\frac{n}{k}}\right)^k\right) \\ &= \log(n!) - \log\left(\left(\frac{n}{k}\right)^n\right) \\ &= \log(n!) - n\log\left(\frac{n}{k}\right) \\ &= \log(n!) - n(\log(n) - \log(k)) \\ &= \log(n!) - n\log(n) + n\log(k) \\ &\geq \log\left(\left(\frac{n}{e}\right)^n\right) - n\log(n) + n\log(k) \\ &= n\log\left(\frac{n}{e}\right) - n\log(n) + n\log(k) \\ &= n\log(n) - n\log(e) - n\log(n) + n\log(k) \\ &= -n\log(e) + n\log(k) \\ &= \Omega(n\log(k))\end{aligned}$$

En conclusión, $h \geq \Omega(n \log k)$ es una cota inferior.

Pregunta 3

Dado un arreglo $A[1..n]$ de valores enteros, se quiere encontrar el máximo valor $A[i] - A[j]$ para algún par de índices i y j del arreglo tal que $1 \leq j < i \leq n$. Considere los siguientes ejemplos:

- Si $A = (22, 5, 8, 10, -3, 1)$, la máxima diferencia es $A[4] - A[2] = 10 - 5 = 5$.
- Si $A = (15, 45, 9, 30, 18, 40, 35)$, la máxima diferencia es $A[6] - A[3] = 40 - 9 = 31$.
- Si $A = (22, 5, 8, -10, -3, 1)$, la máxima diferencia es $A[6] - A[4] = 1 + 10 = 11$.

Asuma que el resultado es 0 si el arreglo tiene un único elemento.

a) Diseñe (y escriba) un algoritmo de tipo dividir y conquistar que resuelva este problema. Puede usar pseudocódigo o cualquier lenguaje de programación. Asumiendo que n es una potencia de 2 , el tiempo de ejecución de su algoritmo (medido en cantidad de comparaciones entre elementos del arreglo A) debe respetar la ecuación de recurrencia:

$$T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + n - 2, & n > 1 \\ 0, & n = 1 \end{cases}$$

No se van a considerar soluciones que no sean del tipo dividir y conquistar.

b) Indique el tiempo de ejecución de su algoritmo usando notación asintótica.

Solución

a)

```python
def max_diff(arr: list[int]):
    n = len(arr)
    assert n >= 2

    if n == 2:
        return arr[1] - arr[0]

    arr_1 = arr[: n // 2]
    arr_2 = arr[n // 2 :]

    max_diff_1 = max_diff(arr_1)
    max_diff_2 = max_diff(arr_2)

    # Diferencia maxima puede ser max_diff_1, max_diff_2
    # o puede estar entre los dos arreglos
    min_arr_1 = min(arr_1)  # n / 2 - 1 comparaciones
    max_arr_2 = max(arr_2)  # n / 2 - 1 comparaciones

    max_diff_both = max_arr_2 - min_arr_1

    return max(max_diff_1, max_diff_2, max_diff_both)
```

b) Usamos el teorema maestro:

$$\log_b(a) = \log_2(2) = 1$$

y

$$f(n) = n - 2$$

Podemos ver que esto cumple con:

$$f(n) \in \Theta(n^{\log_b(a)} \cdot \log_2^k(n)) con k = 1$$

Por lo que:

$$T(n) \in \Theta(n \log n)$$