El problema de Ordenar

¿Qué tan difícil es ordenar un arreglo con n datos?

¿Cambia la respuesta si sabemos algo de los datos?

Según sea el caso, puede convenir usar diferentes algoritmos

Instancias de ordenación

¿Qué tan rápido podemos ordenar un arreglo si…

los datos ya están ordenados?

Instancias de ordenación

¿Qué tan rápido podemos ordenar un arreglo si…

los datos ya están ordenados, pero al revés?

Instancias de ordenación

¿Qué tan rápido podemos ordenar un arreglo si…

los datos están *casi* ordenados?

Instancias de ordenación

¿Qué tan rápido podemos ordenar un arreglo si…

los datos están separados en dos secuencias ordenadas?

Merge

Para dos secuencias ordenadas, A y B

1. Sea C una secuencia ordenada, inicialmente vacía
2. Sean a y b el primer elemento de A y B respectivamente
3. Extraer de su respectiva secuencia el menor entre a y b
4. Insertar ese elemento extraído al final de C
5. Si quedan elementos en ambos A y B, volver a 2.
6. Concatenar C con la secuencia que aún tenga elementos

Merge

¿Como demostramos que Merge es correcto?

¿Cuál es su complejidad?

Finitud

En cada paso el algoritmo extrae un elemento de A o B y se pone en C

Cuando una de las secuencias se vacía, se toma todo lo de la otra secuencia y se pone en C

En total se hacen $|A| + |B|$ pasos, y como tanto A como B son finitos, el algoritmo es finito

Correctitud

PD: Luego de insertar el último elemento en C, esta está ordenada

Por **inducción** sobre las inserciones en C

Caso Base: Luego de la primera inserción, C tiene un solo elemento x_1, por lo que está ordenada.

Hipótesis Inductiva: Luego de la i-ésima inserción del elemento x_i, C está ordenada.

Ahora toca la siguiente inserción.

- Si quedan elementos en A y en B, sea x_{i+1} el más pequeño entre las cabezas de A y de B.

- Si solo quedan elementos en una de las dos secuencias, sea x_{i+1} la cabeza de esta.

Se elimina x_{i+1} de su respectiva secuencia y se inserta al final de C.

$x_i \leq x_{i+1}$. De no ser así x_{i+1} habría salido antes, ó A y B no habrían estado ordenadas.

Como C estaba ordenada, $x_1 \leq x_2 \leq \cdots \leq x_i$, y como $x_i \leq x_{i+1}$, entonces C está ordenada.

Por lo tanto, luego de insertar el último elemento x_n, C está ordenada.

Merge

¿Podremos usar **merge** para ordenar una secuencia arbitraria?

Si de algún modo podemos crear dos secuencias ordenadas…

…podemos combinarlas, ordenando la secuencia completa

MergeSort

Para una secuencia A

1. Si A tiene un solo elemento, terminar en este paso
2. Dividir la secuencia en dos mitades iguales
3. Ordenar cada mitad recursivamente usando MergeSort
4. Combinar las mitades usando Merge

MergeSort

¿Cómo demostramos que MergeSort es correcto?

¿Cuál es su complejidad?

La situación antes descrita se expresa con la recurrencia

$$T(n) = \begin{cases} 1 & n = 1 \\ n + T\left(\left\lceil \frac{n-1}{2} \right\rceil\right) + T\left(\left\lfloor \frac{n-1}{2} \right\rfloor\right) & n > 1 \end{cases}$$

Podemos simplificar la expresión acotándolo por arriba:

$$T\left(\left\lceil \frac{n-1}{2} \right\rceil\right) \geq T\left(\left\lfloor \frac{n-1}{2} \right\rfloor\right)$$

Por lo tanto queda simplificado a

$$T(n) \leq \begin{cases} 1 & n = 1 \\ n + 2 \cdot T\left(\left\lceil \frac{n-1}{2} \right\rceil\right) & n > 1 \end{cases}$$

Además, podemos seguir acotándolo:

$$T\left(\left\lceil \frac{n-1}{2} \right\rceil\right) < T\left(\frac{n}{2}\right)$$

Por lo tanto queda simplificado a

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

Vamos a resolver esta recurrencia para un 2^k tal que

$$n \leq 2^k < 2n$$

$$T(n) \leq T\left(2^k\right) \leq 2^k + 2 \cdot T\left(2^{k-1}\right)$$

$$= 2^k + 2 \cdot \left[2^{k-1} + 2 \cdot T\left(2^{k-2}\right)\right]$$

$$= 2^k + 2^k + 2^2 \cdot T\left(2^{k-2}\right)$$

$$= 2^k + 2^k + 2^2 \cdot \left[2^{k-2} + 2 \cdot T\left(2^{k-3}\right)\right]$$

$$= 2^k + 2^k + 2^k + 2^3 \cdot T\left(2^{k-3}\right)$$

$$\vdots$$

$$= i \cdot 2^k + 2^i \cdot T\left(2^{k-i}\right)$$

Cuando $i = k$, $T\left(2^{k-i}\right) = 1$ por el caso base, por lo tanto

$$T(n) \leq k \cdot 2^k + 2^k$$

Tenemos entonces que

$$T(n) \leq k \cdot 2^k + 2^k$$

Por construcción de k

$$2^k < 2n$$

Por lo tanto

$$T(n) \leq k \cdot 2^k + 2^k < \log(2n) \cdot 2n + 2n$$

Finalmente

$$T(n) \in O(n \cdot \log n)$$