

Ejercicio 1:

Demuestre que $6n^3 \neq O(n^2)$

$6n^3 \neq O(n^2) \Leftrightarrow$ No existe una constante positiva C y un valor n_0 tal que:

$$6n^3 \leq Cn^2, \forall n \geq n_0$$

Balances:

$$6n^3 = Cn^2 \quad \text{Dividimos por } n^2$$

$$6n = C$$

Esto resulta falso ya que No existe una constante C que cumpla con la igualdad.

$$\therefore 6n^3 \neq O(n^2)$$

Ejercicio 2: ¿Cómo sería un array de números (min 10) para el mejor caso de la estrategia de ordenamiento Quicksort(n)?

Para el mejor caso ordena seis elementos en un tiempo de $\Omega(n \log(n))$. Esto ocurre cuando están equilibradas por igual las dos partes de la partición a cada nivel de la recursión. Ej:

$$[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$$

Ejercicio 3: Tiempo de ejecución cuando todos los elementos del array tienen el mismo valor:

- Quicksort(A): $O(n \log(n))$
- Mergesort(A): $O(n \log(n))$
- Insertion-Sort(A): $O(n^2)$

Ejercicio 6:

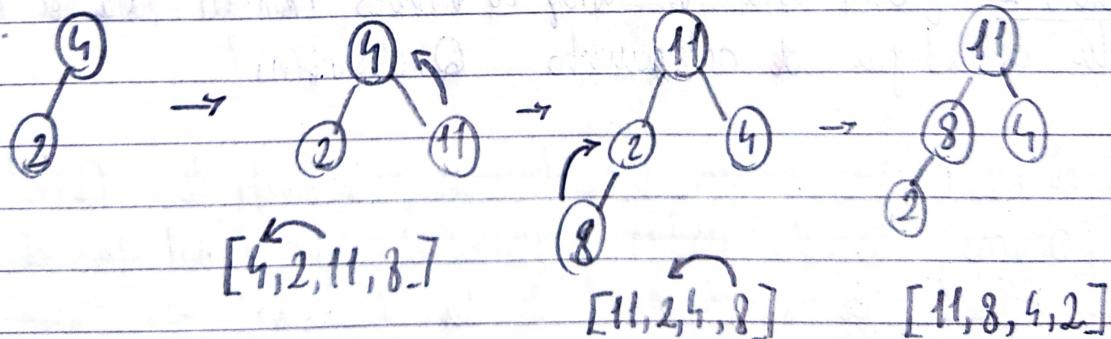
Heapsort, es un método de ordenamiento basado en Comparación. Usa Heap como estructura de datos.

Si Heap es un árbol binario. Este árbol binario tiene que ser completo. Si último nivel puede no estar lleno y los hijos deben reclinarse hacia un mismo lado.

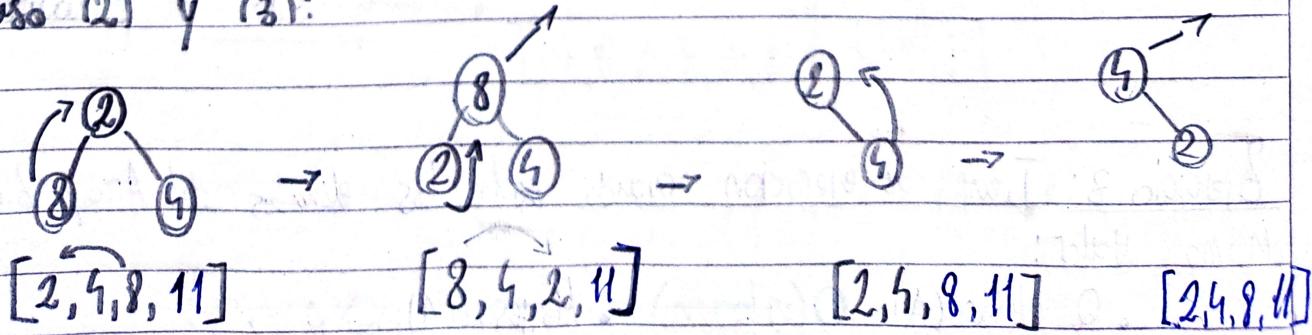
El algoritmo consiste en desenraizar todos los elementos en un Heap y luego extraer el nodo que queda como raíz en iteraciones sucesivas obteniendo el conjunto ordenado. Con los siguientes pasos:

1. Se construye el Heap a partir del arreglo original.
2. La raíz se coloca en el arreglo.
3. Si último elemento del Montículo se vuelve la raíz.
4. La nueva raíz se intercambia con el elemento de Mayor Valor de cada Nivel.
5. Tras el paso anterior la raíz vuelve a ser el mayor del Montículo.
6. Se repite el paso 2 hasta que quede arreglado.

Ej: El Arreglo $[4, 2, 11, 8]$ lo convertimos en un Heap.



Paso (2) y (3):



El tiempo promedio sería $O(n \log(n))$. Sin embargo, si por caso, se diera cuando el heap se encuentra desbalanceado, el tiempo de ejecución puede ser $O(n^2)$. En el mejor caso, también tiene tiempo de ejecución de $O(n \log(n))$.

Ejercicio 7:

$$a) T(n) = 2T\left(\frac{n}{2}\right) + n^4$$

$$a=2; b=2; f(n)=n^4 \rightarrow n^{\log_2(a)} = n^{\log_2(2)} = n$$

\therefore Como $f(n)$ tiene el mismo orden que $n^{\log_2(4)}$, $\Omega(n \log(n))$,

$$b) T(n) = 2T\left(\frac{n}{10}\right) + n$$

$$a=2; b=\frac{10}{4}; f(n)=n$$

$$n^{\log_2(a)} = n^{\log_{10/4}(2)} \approx n^{1.94}$$

$\therefore f(n) = n^{\log_b(a)-\epsilon}, \epsilon > 0 \rightarrow \Omega(n^{\log_{10/4}(2)} \cdot \lg(n)) = \Omega(n^{1.94} \lg(n))$

$$c) T(n) = 16T\left(\frac{n}{4}\right) + n^2$$

$$a=16; b=4; f(n)=n^2$$

$$n^{\log_b(a)} = n^{\log_4(16)} = n^2$$

$\therefore f(n) = n^{\log_b(a)} \rightarrow \Omega(n^2 \lg(n))$,

$$d) T(n) = 7T\left(\frac{n}{3}\right) + n^2$$

$$a=7; b=3; c=2$$

$\log_3(7) < c \rightarrow T(n) = \Omega(f(n)) = \Omega(n^2)$,

$$e) T(n) = 7T\left(\frac{n}{2}\right) + n^2; a=7; b=2; c=2$$

$\log_2(7) > c \rightarrow T(n) = \Omega(n^{2.8})$,

$$f) T(n) = 2T(n/4) + \sqrt{n}$$

$$a=2; b=4; c=1/2$$

$$\log_4(2) = C \rightarrow T(n) = \Omega(n^{1/2} \cdot \log(n)),$$