

TP Complejidad

Nombre: Victor Ramirez

Ejercicio 1

Ejercicio 1

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

Entonces $T(n) = 6n^3$ y se tiene que cumplir que:

$$T(n) \leq c f(n) \text{ cuando } n \geq n_0 \quad \text{además}$$
$$6n^3 \leq cn^2 \rightarrow 6n \leq c \quad f(n) = O(g(n))$$

Donde podemos observar que no existe una constante c tal que $6n \leq c$.

Ejercicio 2

Ejercicio 2

El mejor caso para QuickSort es una lista (no necesariamente ordenada) donde se elige como pivote el elemento que divide la lista en 2 sublistas. $\Omega(n \log n)$

Ej:

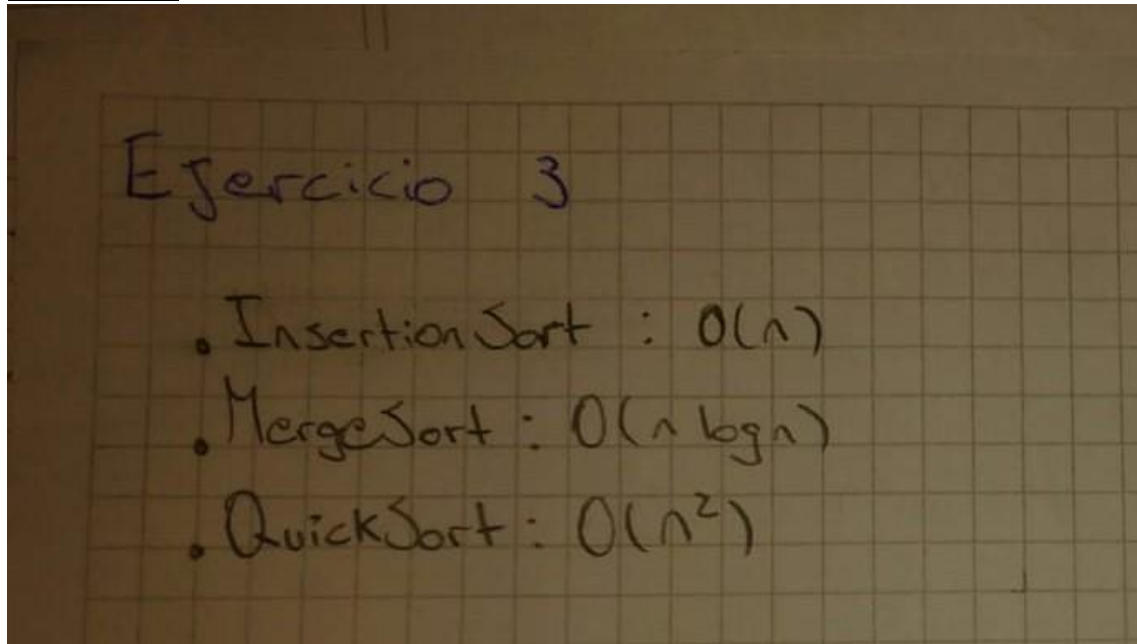
13 | 7 | 9 | 1 | 5 | 6 | 4 | 2 | 8 | 10

1 | 3 | 1 | 4 | 2 | 7 | 9 | 6 | 8 | 10

1 | 2 | 4 | 6 | 8 | 9 | 10

1 | 2 | 3 | 4 5 | 6 | 7 | 8 | 9 | 10

Ejercicio 3



Ejercicio 4

```
def ordenar_list(list):  
    medio = len(list)//2  
    aux = list[medio-1]  
    izq = list[:medio-1]  
    der = list[medio:]  
    #print(izq)  
    #print(der)  
    #print(aux)  
    rta = []  
  
    while izq != [] and der != []:  
        if izq != []:  
            rta.append(izq.pop())  
        if der != []:  
            rta.append(der.pop())  
  
    rta.insert(medio - 1 , aux)  
    #print(rta)  
    return rta
```

Ejercicio 5

```
def contiene_suma(A, n):
    A.sort() #O(n logn)
    left = 0
    right = len(A) - 1

    while left < right:
        suma = A[left] + A[right]
        if suma == n:
            return True
        elif suma < n:
            left += 1
        else:
            right -= 1
    return False

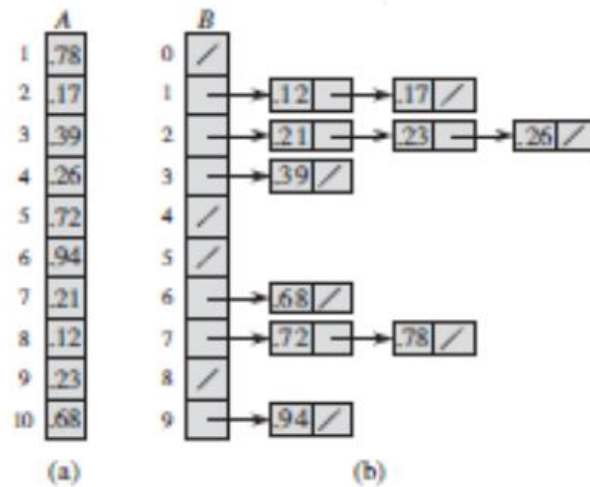
#En el peor de sus casos es O(nlogn)
list = [7,3,2,8,5,4,1,6,10,9]
print(contiene_suma(list , 20)) #Caso falso
print(contiene_suma(list,10)) #Caso Verdadero
|
```

Ejercicio 6

El algoritmo **Bucket sort** es un algoritmo de ordenamiento que divide el conjunto de elementos a ordenar en "buckets" o "cubetas", y luego ordena cada uno de estos buckets por separado utilizando otro algoritmo de ordenamiento, como Insertion sort, Merge sort o Quick sort. Finalmente, se concatenan todos los buckets para obtener la lista ordenada.

Ejemplo:

8.4 Bucket sort



$$\Omega(n)$$

$$O(n^2)$$

$$\Theta(n^2)$$

Si ordenamos las sublistas con Insertion Sort entonces tendremos el mejor caso, caso promedio y peor caso de esa forma. Pero esto puede variar, ya que si ordenamos las sublistas con Merge Sort el peor caso serias $O(n \log n)$.

Ejercicio 7

a) $T(n) = 2T(n/2) + n^4$

$a=2, b=2, f(n) = n^4 \quad \left\{ \log_b a = \log_2 2 = 1 \right.$

Caso 3: $f(n) = \Omega(n^{\log_b a + \epsilon})$, $\epsilon > 0$

$n^4 = n^{1+\epsilon}$, se cumple para $\epsilon=3$

Comprobamos la condición de regularidad

$$2f(n/2) \leq c f(n)$$

$$2 \cdot \frac{n^4}{2^4} \leq c n^4$$

$$2 \cdot \frac{n^4}{16} \leq c n^4$$

$$\frac{1}{8} n^4 \leq c n^4$$

Entonces $c = \frac{1}{8} < 1$ y por lo tanto $T(n) = \Theta(n^4)$

Entonces $C = \frac{1}{8}$ y por lo tanto $T(n) = O(n^2)$

b) $T(n) = 2T(n/10) + n$

$a=2, b=10/7, f(n)=n \mid \log_b a \approx 1.94$

Caso 1: $f(n) = O(n^{1.94-\epsilon})$, $\epsilon > 0$, se cumple para $\epsilon \approx 0.99$

Entonces: $T(n) = \Theta(n^{\log_{10/7} 2})$

c) $T(n) = 16T(n/4) + n^2$

$a=16, b=4, f(n)=n^2 \mid \log_4 16 = 2$

Caso 2: $f(n) = \Theta(n^{\log_b a})$

$f(n) = \Theta(n^2)$

Entonces: $T(n) = \Theta(n^2 \log n)$

$$d) T(n) = 7T(n/3) + n^2$$

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

$$\text{Entonces: } \log_b a = \log_3 7 = 1,77 < 2$$

$$\text{Por lo tanto: } T(n) = \Theta(n^2)$$

$$e) T(n) = 7T(n/2) + n^2$$

$$a = 7, b = 2, c = 2$$

$$\text{Ent: } \log_2 7 = 2,8 > 2$$

$$\text{Por lo tanto: } T(n) = \Theta(n^{\log_2 7})$$

$$F) T(n) = 2T(n/4) + n^{1/2}$$

$$a = 2, b = 4, c = \frac{1}{2}$$

$$\text{Ent: } \log_4 2 = \frac{1}{2} = c$$

$$\text{Por lo tanto: } T(n) = \Theta(n^{1/2} \log n)$$