Práctica IX: Complejidad Computacional

1. Para cada una de las funciones siguientes indique su orden de crecimiento con la notación O(f(n)) y ordene por velocidad de crecimiento (en sentido creciente)

   O(n^2) O(n) O(2^n) O(n^3)

   a. $6 + n^2$; $4 + n + 2\log(n)$; $n^2 + 3 * 2^n$; $5n^3$ — 4+n+2log(n) ; 6+ n^2 ; 5n^3 ; n^2 + 3*2^n

   O(n) O(2^n) O(1) o(n)

   b. $5n$; $n^2 + 5 * 2^n$; 9; $7*n$ — 9 ; 5n ; 7n ; n^2 + 5 * 2^n

   c. 2; $\log(n) + 2n$; $5n + 2^n$; n — 2 ; n ; log(n) + 2n ; 5n + 2^n

   O(1) O(n) O(2^n) O(n)

2. Para cada uno de los siguientes segmentos de algoritmos analice sus complejidades usando notación O grande

   a.

```
Leer(a)                          O(1)
n ← a*a                          O(1)
c ← 0                            O(1)
MIENTRAS (a>1) HACER             O(log(n))
      a ← a/2                    O(1)
      PARA i=1,n HACER           O(n)
      c ← c*2                    O(1)
Escribir(c)                      o(1)
```

   O(n*log(n))

   b.

```
k ← n                            O(1)
PARA i=1,n HACER                 O(n)
      b ← i                      O(1)
      PARA k=k,b (paso=-1) HACER O(n)
         SI a(k-1)>a(k) ENTONCES O(1)
               Aux ← a(k-1)      O(1)
               a(k-1) ← a(k)     O(1)
               a(k) ← Aux        O(1)
```

   O(n^2)

3. Suponiendo que se sabe que el tiempo de ejecución de un algoritmo pertenece a O(N.logN) y que el de otro algoritmo pertenece a $O(n^2)$, se pide:

   O(n*log(n)) tendra mejor tiempo que O (n^2)

   a. ¿Que se puede decir sobre el rendimiento relativo de estos algoritmos?
   b. Suponiendo que ambos algoritmos resuelvan el mismo problema, enumere en cuales caso implementaría uno y en cuales implementaría el otro.
   c. En el caso de que ambos algoritmos sean de ordenación interna, enumere situaciones donde sería valido usar el algoritmo cuadrático.

4. Sea la siguiente tabla:

$T(N) = C \cdot F(N)$

T(n) = 100 * n
T(n) = 1000
n = 10

T(n) = 100 * n
T(n) = 10000
n = 100

T(n) = 5n^2
T(n) = 10000
n = 44.72

T(n) = n^3 / 2
T(n) = 10000
n = 27.14

T(n) = 2^n
T(n) = 10000
n = ln(10000)/ln(2)

} Fijarse en la carpeta

T(n) Tiempo de ejecución proporcional a:	Tamaño máximo del problema para 10^3seg	Tamaño máximo del problema para 10^4seg	Incremento
100 n	10	100	90
$5n^2$	14	44.72	30.72
$n^3 / 2$	12	27.14	15.14
2^n	10	13.2877	3.2877

Se pide:

a) completar los datos que faltan.
b) saque alguna conclusión observando los respectivos incrementos.

5.

O(n)	N_i	100 veces más rápido	1000 veces más rápido
n	N_1	10^3 * N1	10^4 * N1
n^2	N_2		
n^3	N_3		
n^5	N_4		
2^n	N_5		
3^n	N_6		

Donde N_i es el tamaño máximo del problema que se puede resolver en una computadora dada. Se pide completar la tabla precedente en función de N_i , suponiendo que se corre el mismo algoritmo en una máquina 100 veces más veloz y la última columna en una máquina 1000 veces más veloz.

6.

	N_i / seg	N_i / minuto	N_i / hs.
n	1000	60000	3600000
n log n	140	642	203500
n^2	31	240	1860
n^3	10	39	153
2^n	9	15	20

Dada la tabla precedente se pide calcular el N_i para un minuto y para una hora.

7. Ídem al ejercicio 5 pero se pide calcular el Ni para una máquina 10 veces más rápida.

	N_i / seg	10 veces más rápido
n	1000	10000
n log n	140	1001
n^2	31	98
n^3	10	21
2^n	9	12

8. Para cada uno de los algoritmos vistos en la práctica anterior compare el gráfico obtenido con el O(n) correspondiente. (Analice teniendo en cuenta el mejor caso, el caso promedio y el peor caso).

9. Calcule la complejidad del algoritmo “calcular el factorial de n” con recursividad y sin recursividad. Realice mediciones de tiempo.

10. Realice una ficha técnica de su entorno de programación, indicando las características del mismo y cuanto tarda en realizar las siguientes operaciones:

    a. 100.000.000 de sumas.
    b. 100.000.000 de multiplicaciones
    c. 100.000.000 de divisiones
    d. 100.000.000 de comparaciones
    e. 100.000.000 de asignaciones

11. Determine si existe diferencia de tiempo, en su entorno de programación, en la ejecución de un algoritmo si cambia los ciclos For por While o do While.

12.

F(n)	10	20	30	40	50	60
N	1.000.000.000					
N^2	31.622					
N^3	1000					
N^5	63					
2^N	29					
N!	12 < n < 13					

Suponga que en la tabla anterior, f(n) se puede igualar a la cantidad de operaciones que realiza un algoritmo. Sabiendo que en una computadora determinada se realizan 100.000.000 de operaciones por segundo complete la tabla dada. Analice el resultado. Qué sucede si logramos una máquina un millón de veces más rápida?

13. Si se sabe que el algoritmo de selección tardó 10 segundos en ordenar 10000 elementos, cuántos elementos podría ordenar en el triple de tiempo?, cuánto tardaría en ordenar el triple de elementos?, cuánto tardaría en ordenar el triple de elementos en una máquina 3 veces más rápida?

14. Suponga que debe ordenar un vector por el método de Selección. Su entrada es de 2000 elementos y en su PC obtuvo un tiempo de 10 segundos. Si debiera ordenar un vector de 5000 entradas, cuál sería el tiempo estimado de ejecución? Cuál sería el tamaño máximo de vector que podría ordenar en 40 segundos?
    Resuelva este problema reemplazando el método de ordenamiento por cada uno de los métodos vistos.

15. Analice la complejidad computacional del algoritmo de búsqueda binaria. Compárelo con el de búsqueda lineal.

16. Se tiene un arreglo desordenado cuyo registro se compone de 10% clave y 90% datos, si se sabe que en una máquina determinada el algoritmo de burbujeo tardó 10 segundos en ordenar 4000 elementos, cuánto tardaría en ordenar 8000 elementos?, Cuánto tardaría el algoritmo de inserción en ordenar ambos vectores en la misma máquina?

17. El algoritmo de selección ordenó un vector de 10000 enteros en 5 seg. Cuál será el tamaño del vector que espera ordenar en 20 seg. ? Justifique. Qué sucede si ordena ambos vectores en una máquina 5 veces más rápida?

18. Utilizando el algoritmo de inserción se ordenó en un tiempo T1, N1 elementos. En un tiempo de dos T1 se ordenó un vector de dos N1 elementos. Explique las condiciones bajo las cuales pudieron hacerse las pruebas.

19. Utilizando el algoritmo de QuickSort se ordenó un vector de 1 millón de elementos en un tiempo T1. Cuántos elementos espera ordenar en el doble de tiempo?