Complejidad y análisis de algoritmos: ejercicios

Programación II

Segundo cuatrimestre 2016

Los siguientes ejercicios se entregarán resueltos *en papel*[1] el **1 de septiembre de 2016**:

- bien en el laboratorio 7128 de 20 a 21:45, en el horario de clase;
- bien en el casillero de los profesores Simó o Bertaccini *antes* de las 19 horas del día señalado (los casilleros están en el piso 1 del módulo 2B).

Los ejercicios se resolverán de manera *individual*.

1. Comparación de órdenes

Se tiene las siguientes funciones y se desea ordenarlas por su crecimiento asintótico.

Esto es, se pide ordenar las funciones en una lista tal que si $f(n)$ aparece justo antes de $g(n)$, entonces se cumple que $f(n) \in \mathcal{O}(g(n))$.

$$
\begin{aligned}
f_1(n) &= n^{2,5} \\
f_2(n) &= \sqrt{2n} \\
f_3(n) &= n + 10 \\
f_4(n) &= 10^n \\
f_5(n) &= 100^n \\
f_6(n) &= n^2 \log n
\end{aligned}
$$

Para realizar las demostraciones, se puede usar la definición de $\mathcal{O}$, o las ecuaciones del álgebra de órdenes.

Ayuda: considerar $\sqrt{x}$ como el polinomio fraccional $x^{\frac{1}{2}}$.

[1] Es indistinto que la resolución se entregue impresa o a mano. Se recomienda a mano por celeridad, y para practicar programación en papel de cara al parcial.

2. Tamaños máximos

Se tiene una computadora capaz de realizar 10^{10} operaciones por segundo.

Dados los siguientes algoritmos (representados por el número exacto de operaciones para un tamaño n):

(a) n^2

(b) n^3

(c) $100n^2$

(d) $n \log n$

(e) 2^n

(f) 2^{2^n}

¿cuál es el tamaño máximo de n para una hora de procesamiento?

3. Análisis de un algoritmo

A continuación se proporciona un algoritmo que calcula en una matriz las sumas parciales de un arreglo de longitud N:

```
Para i = 0, …, N-1:
  Para j = i, i+1, …, N-1:
    S[i][j] = SUMAR(A[i] → A[j])
```

Por ejemplo, dado el arreglo `A = [7, 20, 3, 13, 29, 12]`, el resultado es:

```
S = [[7, 27, 30, 43, 72, 84],
     [_, 20, 23, 36, 65, 77],
     [_,  _,  3, 16, 45, 57],
     [_,  _,  _, 13, 42, 54],
     [_,  _,  _,  _, 29, 41],
     [_,  _,  _,  _,  _, 12]]
```

(Como se puede ver, la posición `S[i][j]`, con $i \leq j$, representa la suma parcial de `A[i]` hasta `A[j]`.)

Se pide:

(a) Calcular el orden de complejidad del algoritmo.

(b) Transcribir el algoritmo a código Java válido:[2]

```
public static int[][] sumasParciales(int arreglo[]);
```

(c) Implementar (en Java) una segunda versión mejorada que reduzca el orden de complejidad original.

[2] Se puede usar 0 para sustituir los espacios marcados en S, arriba, con '_'.