Ejercicios Análisis de Eficiencia. Luis Quesada Torres

Aviso:

El objetivo de los siguientes ejercicios y cuestiones no es aprender a resolver mecánicamente preguntas de examen, sino comprender los contenidos del temario y aprender a realizar razonamientos idealmente por encima de las expectativas del profesor. De esta forma será menos el temario que deberás memorizar, y toda la asignatura parecerá conexa.

Por ello, algunas preguntas pueden parecer **complejas** y **con trampa**. Recuerda que no debes desanimarte, el objetivo final no es que las respondas correctamente (porque yo no te voy a evaluar), sino que intentes encontrar la respuesta y te topes con todas las dudas que pudieran surgirte durante el examen. **En clase resolveremos todos los que queráis.**

1. Ordena de menor a mayor los siguientes órdenes de complejidad:
   - $\log_2 n$
   - $n*\log_2 n$
   - 1
   - n!
   - n^3
   - 2^n
   - n^2
   - n
   - 3^n
   - $n^2*\log_2 n$

2. Si para resolver un problema disponemos de un algoritmo A con un orden de complejidad $O(n^3)$, y otro algoritmo B con un orden de complejidad $O(n^2)$:
   a) ¿Es mejor utilizar en cualquier caso el algoritmo B? ¿Por qué?
   b) Si el tamaño del problema fuera enorme, ¿Es mejor utilizar el algoritmo B? ¿Por qué?
   c) ¿Podría dar mejor resultado el algoritmo A en algún problema enorme? ¿Por qué?

3. Suponemos que las sumas, multiplicaciones, restas y divisiones son operaciones elementales:
   a) ¿Podría, en un problema, una operación suma no ser una operación elemental?
   b) ¿Qué debería ocurrir para ello?
   c) ¿Qué orden de complejidad tendría esa operación suma?
   d) ¿Podría una operación multiplicación no ser una operación elemental?
   e) ¿Qué orden de complejidad tendría esa operación multiplicación?

4. a) ¿Cuál sería el orden de complejidad de esta función?

```
1: void vacia(int v[],int n) {
2:   int i;
3:   for (i = 0;i < n;i++)
4:     v[i] = 0;
5: }
```

   b) ¿Cuál sería el orden de complejidad del siguiente código (sup. n=tamaño problema)?

```
1:   a = 0;
2:   b = a*2;
3:   vacia(v,n);
```

   c) ¿Cuál sería el orden de complejidad del siguiente código?

```
1:   a = 0;
2:   b = a*2;
3:   vacia(v,5);
```

5. a) ¿Qué cota de la eficiencia obtenemos usando la notación O Mayúscula?
   b) ¿Qué cota de la eficiencia obtenemos usando la notación Ω (Omega)?
   c) ¿Por qué en las sentencias condicionales, para la notación Ω (Omega), cogemos también el peor caso posible?

6. ¿Qué tendríamos que considerar para realizar el estudio de eficiencia del mejor caso, en lugar del peor caso?

7. ¿Qué tendríamos que considerar para realizar el estudio de eficiencia del caso medio, en lugar del peor caso?

8. a) Calcula el orden de complejidad del siguiente código:

```
1:   for (i = 0;i < n;i++) {
2:     x = x+1;
3:   }
```

   b) ¿Podrías sustituir ese código por otro con orden de complejidad constante?

9. a) Calcula el orden de complejidad del siguiente código:

```
1:   for (i = 0;i < 2*n;i++) {
2:     x = x+1;
3:   }
```

   b) ¿Podrías sustituir ese código por otro con orden de complejidad constante?

10. a) Calcula el orden de complejidad del siguiente código:

```
1:   i = 0;
2:   while (i < 2*n) {
3:     x = x+1;
4:     i = i+2;
5:   }
```

   b) ¿Podrías sustituir ese código por otro con orden de complejidad constante?

11. Calcula el orden de complejidad del siguiente código:

```
1:   for (i = 0;i < n;i++) {
2:     for (j = 0;j < n;j++) {
3:       cout << i << " " << j << endl;
4:     }
5:   }
```

12. Calcula el orden de complejidad del siguiente código (hace lo mismo que el anterior):

```
1:   for (x = 0;x < n*n;i++) {
2:     i = x/n;
3:     j = x%n;
4:     cout << i << " " << j << endl;
5:   }
```

13. Calcula el orden de complejidad del siguiente código:

```
1:   i = n;
2:   while (i >= 2) {
3:     x = x+1;
4:     i = i/2; //se redondea hacia abajo.
5:   }
```

14. Calcula el orden de complejidad del siguiente código (hace lo mismo que el anterior):

```
1:   for (i = n;i >= 2;i/=2) {
2:     x = x+1;
3:   }
```

15. Calcula el orden de complejidad del siguiente código:

```
1:   for (i = n;i >= 2;i/=2) {
2:     for (j = 0;j < n;j++) {
3:       x = x+1;
4:     }
5:   }
```

16. Calcula el orden de complejidad del siguiente código:

```
1:   for (i = 0;i < n;i++)
2:     for (j = i;j < n;j++)
3:       x = x+1;
```

17. Calcula el orden de complejidad del siguiente código:

```
1:   for (i = 0;i < n;i++) {
2:     if (entrada == true) {
3:       for (j = 0;j < n;j++)
4:         x += i;
5:     }
6:   }
```

18. Calcula el orden de complejidad del siguiente código:

```
1:   for (i = 0;i < n;i++) {
2:     if (i%2) {
3:       for (j = i;j < n;j++)
4:         x *= i;
5:       for (j = 1;j < i;j++)
6:         y *= j;
7:     }
8:   }
```

19. Encuentra el orden de complejidad (notación Θ (Theta)) para el siguiente código:

```
1:   i = 2;
2:   while (i < n) {
3:     i = i*i;
4:     x = x+1;
5:   }
```

20. Demostrar que 17 es O(1).

21. Demostrar que n(n-1)/2 es $O(n^2)$.

22. Demostrar que max(n^3,$10n^2$) es $O(n^3)$.

23. Ordena de menor a mayor los siguientes órdenes de complejidad:
    - n
    - $\sqrt{n}$
    - n^3+1
    - n^2
    - $n\log_2(n^2)$
    - $3^{\log2(n)}$
    - 3^n
    - 2^n
    - 2^n+3^{n-1}
    - 20000
    - n+100
    - $n2^n$

24. ¿Verdadero o falso? Si $T_1(n)$ es $O(f(n))$ y $T_2(n)$ es $O(f(n))$...
    a) $T_1(n)+T_2(n)$ es $O(f(n))$
    b) $T_1(n)$ es $O(f^2(n))$
    c) $T_1(n)/T_2(n)$ es $O(1)$

25. Indicar para cada par distinto i y j si $f_i(n)$ es $O(f_j(n))$ y si $f_i(n)$ es $\Omega(f_j(n))$

    $f_1(n) = n^2$

    $f_2(n) = n^2 + 1000n$

    $f_3(n) = \begin{cases} n & \text{si n es impar} \\ n^3 & \text{si n es par} \end{cases}$

    $f_4(n) = \begin{cases} n & \text{si n <= 100} \\ n^3 & \text{si n > 100} \end{cases}$

26. Pon un ejemplo de código cuyo orden de complejidad sea:

    a) n
    b) $n\log_2(n)$
    c) n^2
    d) n en el mejor caso, n^2 en el peor.

27. Si T(n) = 4^(n+1) , ¿es O(4^n)?