UNIVERSIDAD DE GRANADA

Departamento de Ciencias de la Computación e Inteligencia Artificial

◁ Ejercicio 1 ▷

Existen dos algoritmos, A y B, para resolver un determinado problema, con tiempos de ejecución $T_A(n) = 300 \cdot n^2$ segundos y $T_B(n) = 15 \cdot n^3$. ¿Cuál es el orden de eficiencia de cada algoritmo? ¿En qué casos es más eficiente utilizar un algoritmo u otro? Determine el tamaño del caso n_0 para el cual, a partir de ese tamaño de problema, uno de los algoritmos es mejor que el otro.

◁ Ejercicio 2 ▷

¿Es mejor un algoritmo A que resuelve un problema en $1000 \cdot n^2$ minutos u otro algoritmo B que resuelve el mismo problema en $10^{-6} \cdot n^3$ segundos? Razone la respuesta.

◁ Ejercicio 3 ▷

De las siguientes afirmaciones, indicar cuáles son ciertas y cuáles no:

- $n^2 \in \mathsf{O}(n^3)$
- $n^3 \in \mathsf{O}(n^2)$
- $2^{n+1} \in \mathsf{O}(2^n)$
- $(n+1)! \in \mathsf{O}(n!)$
- $3^n \in \mathsf{O}(2^n)$
- $\log(n) \in \mathsf{O}(n^{1/2})$
- $n^{1/2} \in \mathsf{O}(\log(n))$
- $n^2 \in \Omega(n^3)$
- $n^3 \in \Omega(n^2)$
- $2^{n+1} \in \Omega(2^n)$
- $(n+1)! \in \Omega(n!)$
- $3^n \in \Omega(2^n)$
- $\log(n) \in \Omega(n^{1/2})$
- $n^{1/2} \in \Omega(\log(n))$

◁ Ejercicio 4 ▷

Usando las relaciones $=$ y $\subset$, ordene los órdenes de complejidad de las siguientes funciones: $n \cdot \log(n)$, $n^2 \cdot \log(n)$, n^6, n^{1+a}, $(1+a)^n$, $(n^2 + 7 \cdot n + \log^3(n))^4$, $n^2/\log(n)$, 2^n. Se asume que a es una constante real tal que $0 < a < 1$.

◁ Ejercicio 5 ▷

Considere las siguientes funciones:

$$
\begin{aligned}
f_1(n) &= n^2 \\
f_3(n) &= \begin{cases} n & n \text{ impar} \\ n^3 & n \text{ par} \end{cases} \\
f_2(n) &= n^2 + 1000 \cdot n \\
f_4(n) &= \begin{cases} n & n \leq 100 \\ n^3 & n > 100 \end{cases}
\end{aligned}
$$

Para cada posible valor de i, j, indique si $f_i \in \mathsf{O}(f_j)$ y si $f_i \in \Omega(f_j)$.

◁ Ejercicio 6 ▷

Encuentre dos funciones $f(n)$ y $g(n)$ tales que $f \notin \mathsf{O}(g)$ y $g \notin \mathsf{O}(f)$.

◁ Ejercicio 7 ▷

Calcule el orden de eficiencia del peor y del mejor caso del siguiente algoritmo:

```
int Algoritmo_1(int *v, int n, int c) {
  int inf, sup, i;
  while (sup >= inf) {
    i = (inf + sup) / 2;
    if (v[i] == c){
      return i;
    }
    else if (c < v[i]) {
      sup = i - 1;
    }
    else {
      inf = i + 1;
    }
  }
  return 0;
}
```

◁ Ejercicio 8 ▷

Calcule el orden de eficiencia del peor y del mejor caso del siguiente algoritmo:

```
void Algoritmo_2(int n) {
  int i, j, k, s;
  s = 0;
  for (i = 1; i < n; i++) {
    for (j = i + 1; j <= n; j++) {
      for (k = 1; k <= j; k++) {
        s = s + 2;
      }
    }
  }
}
```

◁ Ejercicio 9 ▷

Calcule el orden de eficiencia de la siguiente función:

```
void Algoritmo_3(int num) {
  if (num < 10) {
    return num;
  }
  else {
    return (num \% 10) + Algoritmo_3(num / 10);
  }
}
```

- Modifique la función para eliminar la recursividad.
- Calcule la eficiencia de la función modificada y justifique en qué casos es más conveniente usar una u otra.

◁ Ejercicio 10 ▷

Calcule la eficiencia del peor caso del siguiente procedimiento, suponiendo que la llamada a la función `V` es O(n), donde `n = b - c`:

```
void Algoritmo_4(T *a, int b, int c) {
  if (b < c) {
    Algoritmo_4(a, b + 1, c);
    if (V(a, b, c)) {
      Algoritmo_4(a, b, c - 1);
    }
  }
}
```

◁ Ejercicio 11 ▷

Calcule el orden de eficiencia de la llamada a la función `Algoritmo_5(a,1,n)`, suponiendo que `n` es potencia de 3:

```
int Algoritmo_5(int *a, int prim, int ult) {
  int mitad, terc;
  if (prim >= ult) {
    return v[ult];
  }
  mitad = (prim + ult) / 2;
  terc = (ult - prim) / 3;
  return v[mitad] + Algoritmo_5(v, prim, prim + terc) + Algoritmo_5(a, ult - terc, ult);
}
```

◁ Ejercicio 12 ▷

Resuelva las siguientes ecuaciones e indique su orden de complejidad:

- $T(n) = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ 3 \cdot T(n-1) + 4 \cdot T(n-2) & n > 1 \end{cases}$

- $T(n) = \begin{cases} 1 & n = 1 \\ 8 & n = 2 \\ 4 \cdot T(n/2) + n^2 & n \geq 4, n \text{ potencia de 2} \end{cases}$

- $T(n) = \begin{cases} 1 & n = 1 \\ 3 \cdot T(n/2) + n \cdot \log_2(n) & n > 1, n \text{ potencia de 2} \end{cases}$

- $T(n) = \begin{cases} 1 & n = 1 \\ 5 \cdot T(n/2) + (n \cdot \log_2(n))^2 & n > 1, n \text{ potencia de 2} \end{cases}$

- $T(n) = \begin{cases} 9 \cdot n^2 - 15 \cdot n + 106 & n = 0, 1, 2 \\ T(n-1) + 2 \cdot T(n-2) - 2 \cdot T(n-3) & n > 2 \end{cases}$

- $T(n) = \begin{cases} 1 & n = 1 \\ 3/2 & n = 2 \\ (3/2) \cdot T(n/2) - (1/2) \cdot T(n/4) - (1/n) & n > 2 \end{cases}$

- $T(n) = \begin{cases} 1/3 & n = 1 \\ n \cdot T^2(n/2) & n > 1, n \text{ potencia de 2} \end{cases}$

- $T(n) = \begin{cases} 1 & n = 1 \\ 5 \cdot T(n/2) + (n \cdot \log_2(n))^2 & n > 1, n \text{ potencia de 2} \end{cases}$

- $T(n) = \begin{cases} 1 & n = 1, 2 \\ T(n/2) + 2 \cdot T(n/4) + n & n > 2, n \text{ potencia de 2} \end{cases}$

- $T(n) = \begin{cases} 1 & n = 2 \\ 2 \cdot T(\sqrt{n}) + \log_2(n) & n \geq 4 \end{cases}$

◁ Ejercicio 13 ▷

Resuelva la siguiente recurrencia:

$$T(n) = a \cdot T(n/b) + n^k$$

con $a \geq 1$, $b \geq 2$ y $k \geq 0$.

◁ Ejercicio 14 ▷

Calcule la eficiencia de los algoritmos cuyos tiempos de ejecución vienen dados por las siguientes ecuaciones de recurrencias:

- $T(n) = T(n/2) \cdot T^2(n/4)$
- $T(n) = 2 \cdot T(n-1) + 1$
- $T(n) = 3 \cdot T(n/2) + n$
- $T(n) = 2 \cdot T(n/2) + \log_2(n)$
- $T(n) = T(\sqrt{n}) + n$
- $T(n) = 2 \cdot T(n/2) + n \cdot \log(n)$
- $T(n) = 3 \cdot T(n/2) + 5 \cdot n + 3$
- $T(n) = 2 \cdot T(n/2) + \log_2(n)$
- $T(n) = 2 \cdot T(n/4) + n^{1/2}$
- $T(n) = 4 \cdot T(n/3) + n^2$
- $T(n) = 3 \cdot T(n/2) + 4 \cdot T(n/4) + n^2$