UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria Informàtica

etsinf

Tema 2. Análisis

Programación (PRG)

Jorge González Mollá

Departamento de Sistemas Informáticos y Computación

Índice

Motivación

- Para un mismo problema hay diversas soluciones propuestas.
- En programación, este concepto se aplicaría a los algoritmos, es decir, varios algoritmos correctos para un mismo problema.
- ¿Cómo decidir cuál de todas esas propuestas es la mejor? Mediante un criterio objetivo ⇨ eficiencia.
- El algoritmo más eficiente es el que consume menos recursos.
- Los recursos básicos son la memoria RAM y el tiempo de CPU.

Eficiencia

La eficiencia de los algoritmos se expresa en términos de:

- Coste Temporal: Medida de la cantidad de tiempo que un algoritmo determinado tarda en ejecutarse.
- Coste Espacial: Medida de la cantidad de espacio en memoria que un algoritmo determinado requiere durante su ejecución.

Coste

El coste de un algoritmo dado depende de dos tipos de factores:

- Factores propios del algoritmo
  - Estrategia
  - Tipos de datos
- Factores circunstanciales del entorno de programación
  - Tipo de ordenador
  - Lenguaje de programación
  - Compilador
  - Carga del sistema
  - etc.

Coste

El coste de un algoritmo dado se puede estimar de dos maneras:

- Análisis teórico o a priori
  - Se estima en función de los factores propios del algoritmo
  - Es un análisis independiente del entorno de programación
- Análisis experimental o a posteriori
  - Se estima incluyendo también los factores circunstanciales, es decir, midiendo el número de segundos en tiempo real que un algoritmo determinado necesita para su ejecución; y la cantidad de bytes de memoria ocupados por el mismo.
  - Es un análisis bajo un entorno de programación específico y para un conjunto dado de datos de entrada (1 instancia).

Coste

1. Los dos tipos de análisis son importantes y muy necesarios, es decir, son complementarios.
2. Todo programador debería saber realizar un análisis teórico de sus algoritmos, con el objeto de evitar perder el tiempo programando código del que después se tiene que prescindir.
3. La eficiencia es un criterio objetivo que todo informático debería tener en cuenta durante el diseño de sus algoritmos.
4. Independientemente de cuál fuere su entorno de ejecución, un algoritmo siempre debería ser lo más eficiente posible.

Coste Temporal

Ejemplo: Dado un número entero n, calcular su cuadrado n^2

- Algoritmo A1:

```
m = n*n;
```

- Algoritmo A2:

```
m = 0;
for (int i = 0; i < n; i++) { m = m + n; }
```

- Algoritmo A3:

```
m = 0;
for (int i = 0; i < n; i++) {
      for (int j = 0; j < n; j++) { m++; }
}
```

Coste Temporal

El coste temporal de un algoritmo se estima en función del tiempo de ejecución de las operaciones de CPU más elementales

- A1:
```
m = n*n;
```
$T_{A1} = t_a + t_m$

- A2:
```
m = 0;
for (int i = 0; i < n; i++) { m = m + n; }
```
$T_{A2} = t_a + t_a + (n+1)\, t_c + n\, t_i + n\, t_a + n\, t_s$

- A3:
```
m = 0;
for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) { m++; }
}
```
$T_{A3} = t_a + t_a + (n+1)\, t_c + n\, t_i + n\, t_a + n\,(n+1)\, t_c + 2\, n^2\, t_i$

siendo t_a el tiempo de ejecución de una operación de asignación; t_m el de una operación de multiplicación; t_c el de una operación de comparación; t_i el de una operación de incremento; y t_s el de una operación de suma.

➢ Comparar los costes temporales de estos algoritmos a través de este análisis resulta bastante complejo, y además requiere un esfuerzo de conteo considerable.

Coste Temporal

- El primer paso sería independizar el coste temporal de los tiempos de ejecución de las operaciones de la CPU: asumir que todas las instrucciones de la CPU tardan lo mismo (1 U.T.).
  - A1: $T_{A1} = t_a + t_m = 1 + 1 = 2$ U.T. $\equiv k_1$
    Constante: no depende de n
  - A2: $T_{A2} = t_a + t_a + (n+1)\, t_c + n\, t_i + n\, t_a + n\, t_s = 4n + 3$ U.T. $\equiv k_2\, n + k_3$
    Dependencia lineal sobre n
  - A3: $T_{A3} = t_a + t_a + (n+1)\, t_c + n\, t_i + n\, t_a + n\,(n+1)\, t_c + 2\, n^2\, t_i =$
    $= 3n^2 + 4n + 3$ U.T. $\equiv k_4\, n^2 + k_5\, n + k_6$
    Dependencia cuadrática sobre n

Talla del problema

- El coste temporal (o espacial) de un algoritmo se define como:
  - Una función T(n), no decreciente, de la cantidad de tiempo (o de espacio en memoria) que necesita dicho algoritmo para su ejecución, en función del tamaño del problema (n).
- La talla o tamaño de un problema se define como:
  - Un subconjunto de parámetros de entrada del problema que representa una medida cuantitativa de su dificultad.

Y suele estar relacionada con el número de datos a procesar.

Talla del problema

- Ejemplo de problemas conocidos y sus tallas correspondientes

Problema	Talla del problema
Búsqueda de un elemento en un conjunto	Número de elementos en el conjunto
Multiplicación de matrices	Dimensión de las matrices
Cálculo del factorial de un número	Valor del número
Resolución de un sistema de ecuaciones lineales	Número de ecuaciones y/o incógnitas
Ordenación de un array	Número de elementos en el array

Pasos de Programa

- El coste temporal de un algoritmo A se expresará como: $T_A(n)$
- Aún así, todavía requiere un esfuerzo de conteo considerable.
- Siguiente asunción: una secuencia de operaciones simples independiente de la talla se ejecutará en 1 paso de programa.
  - A1:
    ```
    m = n*n;
    ```
    $T_{A1}(n) = 1$ p.p.
  - A2:
    ```
    m = 0;
    for (int i = 0; i < n; i++) { m = m + n; }
    ```
    $T_{A2}(n) = 1 + n*1 + 1 = n + 2$ p.p.
  - A3:
    ```
    m = 0;
    for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) { m++; }
    }
    ```
    $T_{A3}(n) = n(n + 2) + 2 = n^2 + 2n + 2$ p.p.

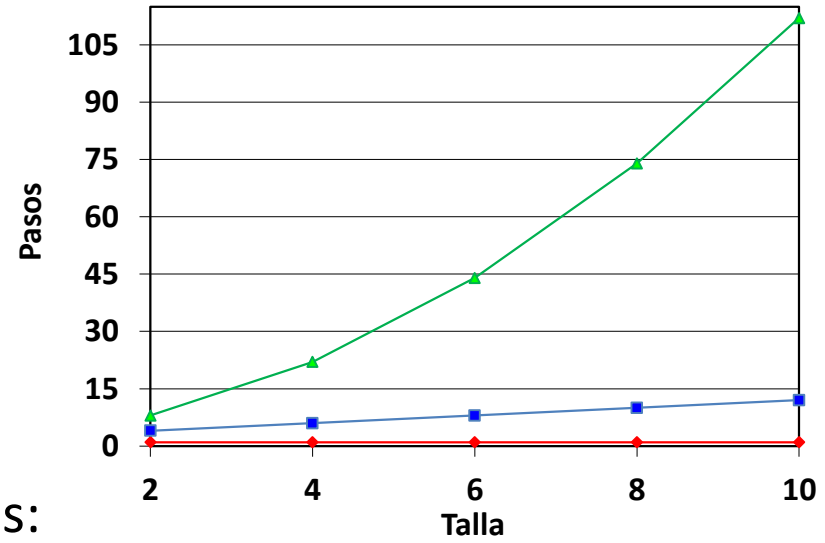
Funciones de Coste

- Comparar costes de algoritmos consiste en:
  - Comparar funciones no decrecientes, donde lo que nos interesa es su tasa de crecimiento.

$T_{A1}(n) = 1 \approx k$

$T_{A2}(n) = n + 2 \approx n$

$T_{A3}(n) = n^2 + 2n + 2 \approx n^2$



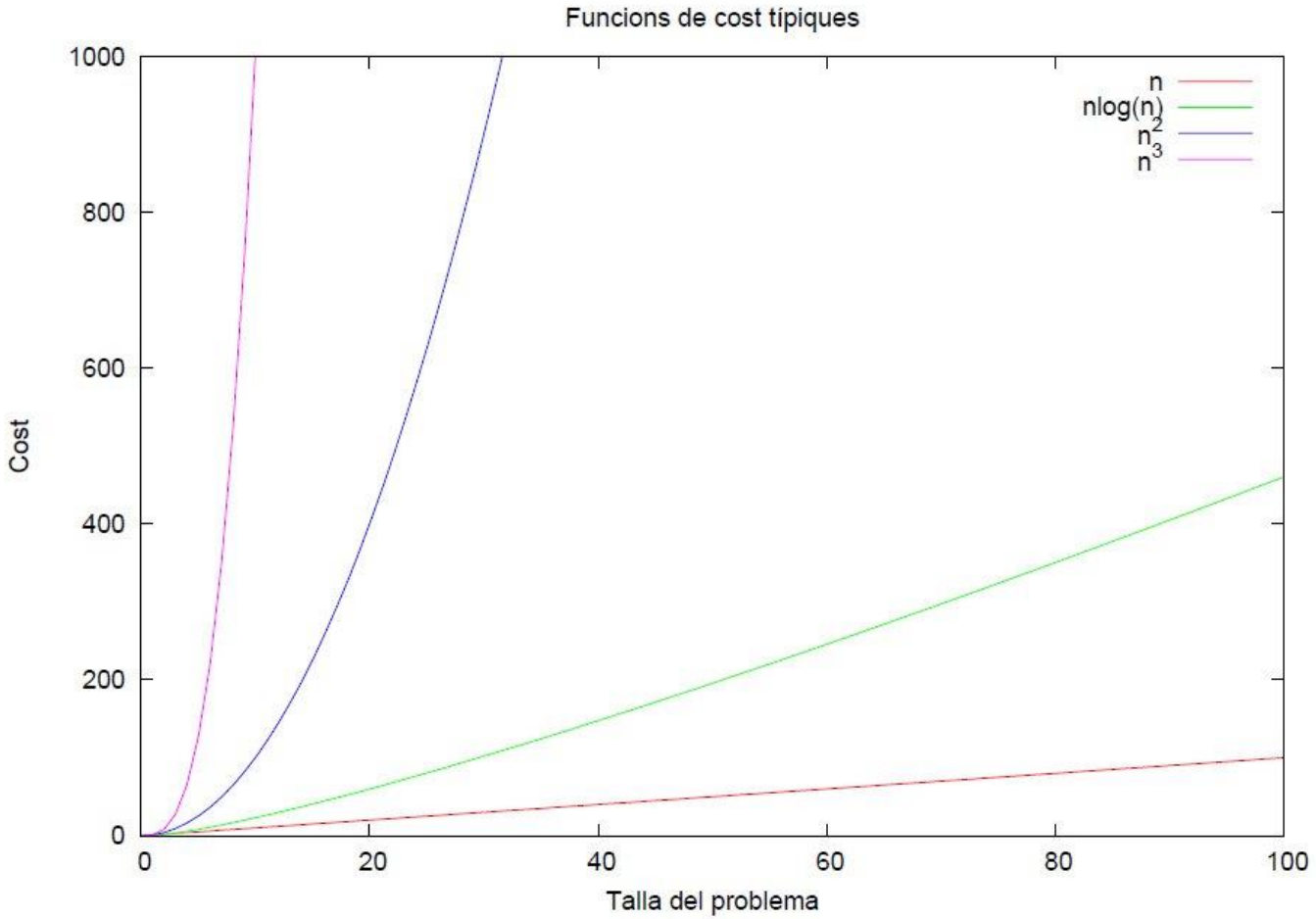
- Si T(n) es un polinomio, entonces:
  - El término de mayor grado del polinomio determina el aspecto de la curva de crecimiento.

Funciones de Coste

- Tabla con funciones de coste típicas en orden creciente

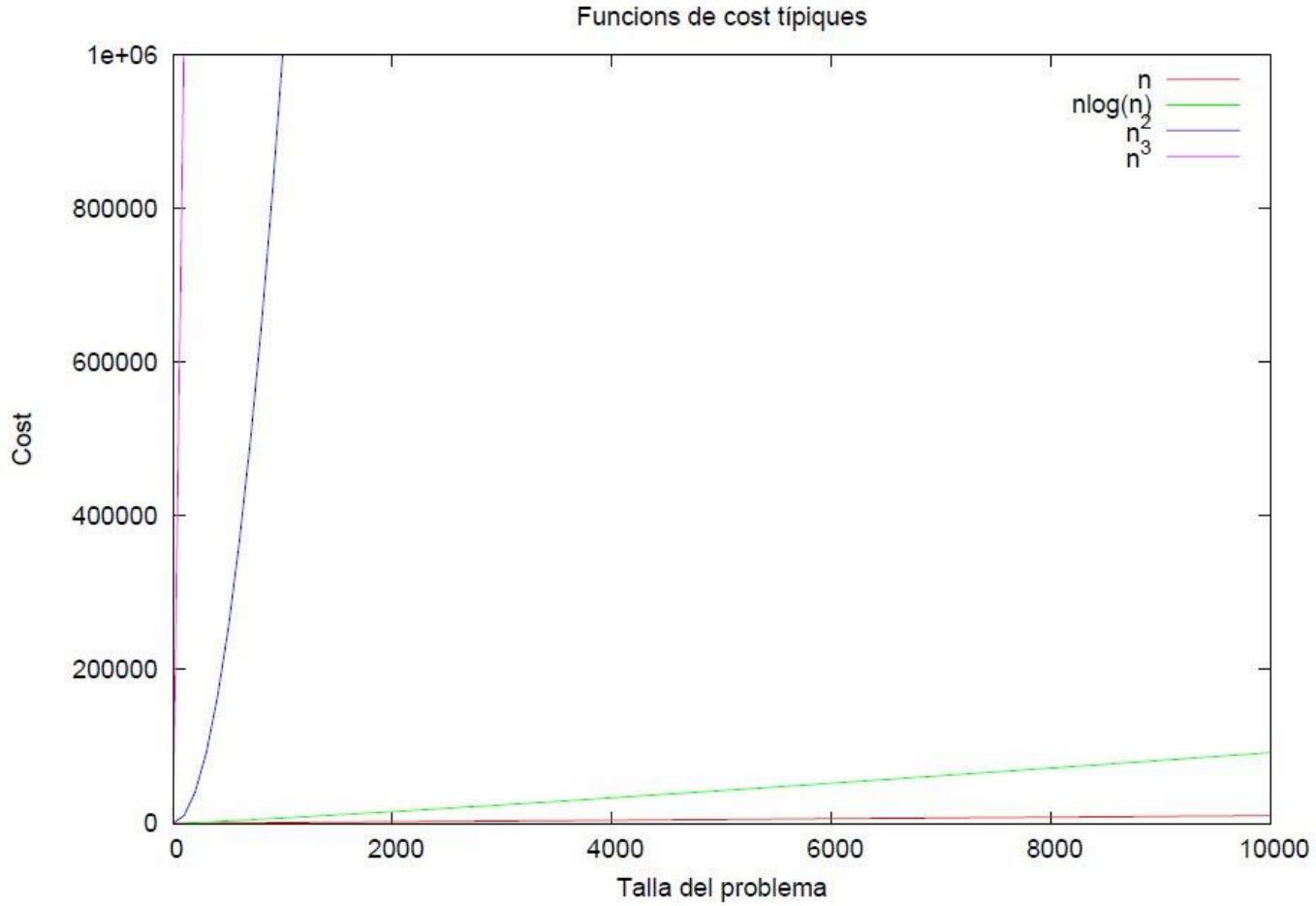
Función	Nombre
c	constante
$\log n$	logarítmica
$\log^2 n$	logarítmica al cuadrado
n	lineal
$n \log n$	n log n
n^2	cuadrática
n^3	cúbica
2^n	exponencial

Funciones de Coste



- Las funciones no lineales implican tiempos mayores que las funciones lineales.
- En general, los valores pequeños de n no suelen tener demasiada importancia. Para un valor de n=20, todos los algoritmos terminan en menos de 5 segundos. En la práctica, no existen muchas diferencias entre el mejor y el peor algoritmo.

Funciones de Coste



- Para valores de n suficientemente grandes, la función T(n) se encuentra completamente determinada por su término dominante. Por ejemplo, calculemos el valor de la función $T(n)=10n^3+n^2+40n+80$ para un $n=1000$: $T(1000)$ vale 10.001.040.080, del que 10.000.000.000 se debe al término $10n^3$.

Instrucción Crítica

- Contar pasos de programa sigue suponiendo mucho esfuerzo.

- Última asunción: tener en cuenta sólo el bloque o secuencia que contiene la instrucción crítica del algoritmo, lo que en la práctica podéis asumir como el cuerpo del bucle más interno.

  - A1:
    ```
    m = n*n;
    ```
    $T_{A1}(n) = 1$ i.c.
  - A2:
    ```
    m = 0;
    for (int i = 0; i < n; i++) { m = m + n; }
    ```
    $T_{A2}(n) = n*1 = n$ i.c.
  - A3:
    ```
    m = 0;
    for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) { m++; }
    }
    ```
    $T_{A3}(n) = n*n = n^2$ i.c.

Ejemplo: Coste Lineal

- **Ejemplo:** Supongamos que se está descargando un fichero de internet, de forma que hay un retraso inicial de 2 seg. para establecer la conexión y después la descarga se realiza a razón de 1.6 Mb/seg. Si el tamaño del fichero es de N Mb, el tiempo de descarga viene dado por: $T(n)=n/1.6 + 2$.
- La descarga de un fichero de 80 Mb requerirá 52 seg., la descarga de un fichero el doble de grande requiere del orden de 102 seg., casi el doble.
- Un algoritmo lineal se caracteriza porque el tiempo de ejecución es proporcional a la entrada del problema.
- El coste lineal es un buen coste, el coste logarítmico es el mejor, y el coste n log n también es bueno.
- El resto de costes ya limitan considerablemente la talla de los problemas que podemos resolver.