Análisis de Algoritmos y Estructuras de Datos

Tema 0: Introducción

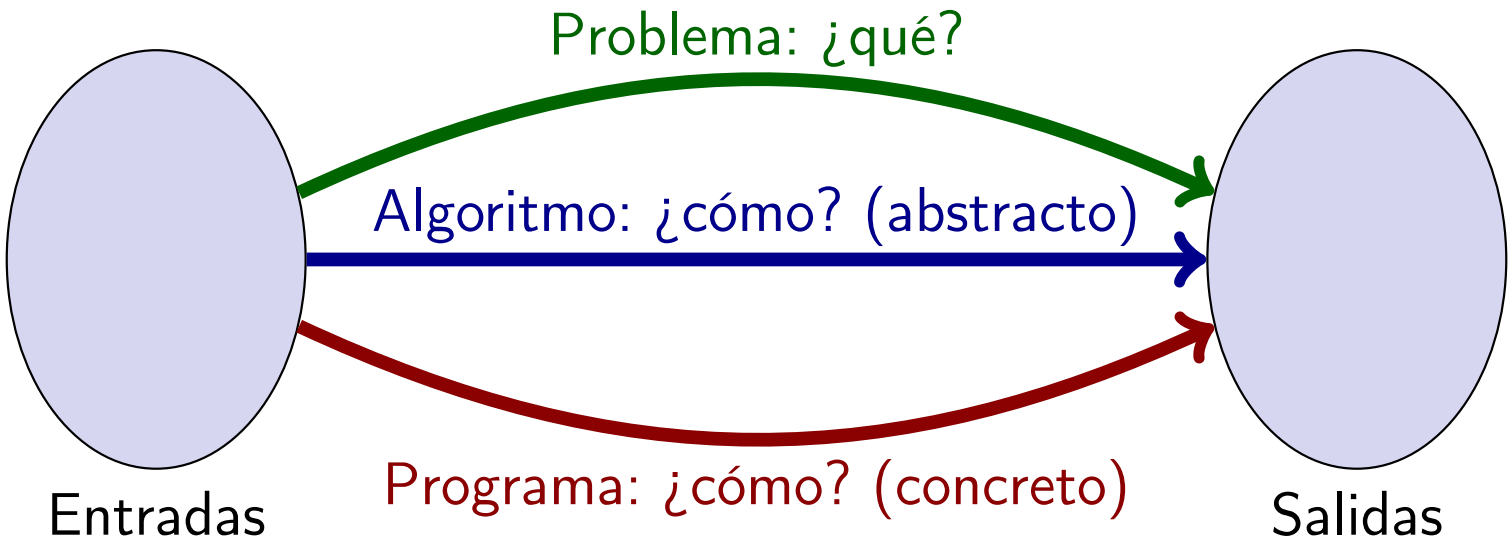
Mª Teresa García Horcajadas José Fidel Argudo Argudo
Antonio García Domínguez Francisco Palomo Lozano

UCA
Universidad de Cádiz

Versión 1.0

Índice

Conceptos básicos



Problema Especificación de una función de entradas en salidas.

Algoritmo Secuencia finita de instrucciones capaz de computar la función. Se ejecuta en una máquina abstracta.

Programa Expresión de un algoritmo en un lenguaje de programación. Se ejecuta en una máquina real.

Lenguaje de Especificación de Algoritmos (LEA)

Asignación normal y paralela

$x \leftarrow \text{expr}$

$(x_1, \ldots, x_n) \leftarrow (\text{expr}_1, \ldots, \text{expr}_n)$

Condición

si expr-lógica
  instrucciones$_1$
[si no
  instrucciones$_2$]

Iteración

mientras expr-lógica
  instrucciones

repite
  instrucciones
[mientras | hasta] expr-lógica

LEA: utilidad

- Describiremos los algoritmos de forma concisa mediante un lenguaje más abstracto: LEA.
- LEA tiene una semántica bien definida, a diferencia de otros lenguajes.
- Las variables son de un cierto tipo abstracto, con sus constantes, predicados y funciones habituales.

Corrección y eficiencia

Corrección algorítmica: verificación de algoritmos

¿Resuelve el algoritmo el problema?

Parada ¿Termina su ejecución?

Corrección parcial Si termina, ¿da el resultado esperado?

Eficiencia algorítmica: análisis de algoritmos

¿Aprovecha el algoritmo bien los recursos?
A nivel abstracto, hay dos fundamentales: espacio y tiempo.

Relación entre programa y algoritmo

Un programa no puede ser correcto ni eficiente si su algoritmo subyacente no lo es.

Ejemplo: multiplicación rusa

implementado como

Algoritmo

$mul(a, b) \to c$
$c \leftarrow 0$
mientras $a \neq 0$
    si $a \mod 2 \neq 0$
        $c \leftarrow c + b$
    $(a, b) \leftarrow (a \operatorname{div} 2, b \cdot 2)$

Programa (en C)

```
1  typedef unsigned natural;

3  natural mul(natural a, natural b)
4  {
5    natural c = 0;

7    while (a != 0) {
8      if (a % 2 != 0)
9        c = c + b;
10     a = a / 2; b = b * 2;
11   }
12   return c;
13 }
```

Ejemplo: multiplicación rusa de 22 y 19

- Entrada: ejemplar del problema
- Cálculo: ejecución del algoritmo
- Salida: solución del ejemplar

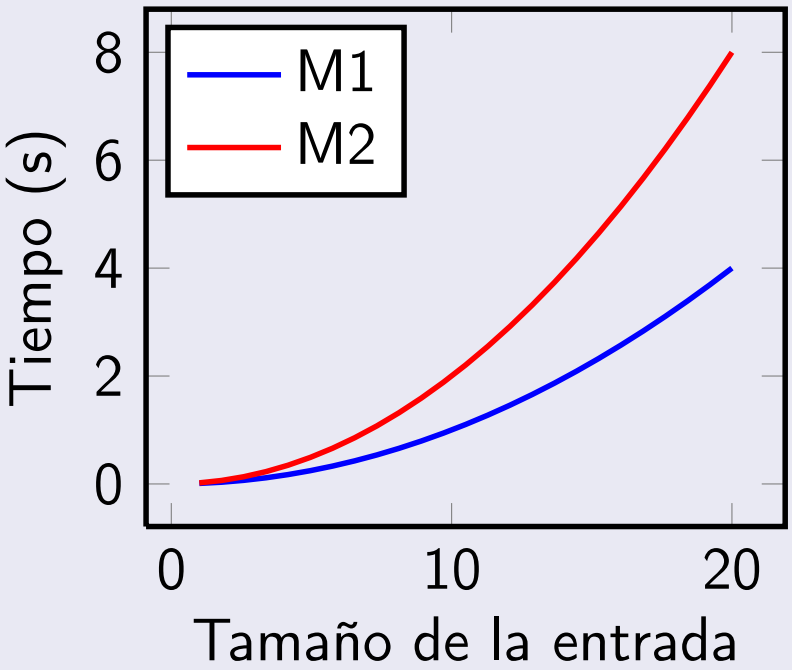
	a	b	c	
Entrada $\rightarrow$	22	19	0	
	11	38	0	
	5	76	38	
	2	152	114	
	1	304	114	
	0	608	418	$\leftarrow$ Salida

Principio de invarianza

Proposición

Dos programas correspondientes a un mismo algoritmo sólo difieren en eficiencia temporal en un factor constante.

Interpretación gráfica ideal



Interpretación intuitiva

Mejorar el programa o cambiar de máquina nunca reducirá su tiempo más que en un factor constante, siempre que:

1. No se altere el algoritmo subyacente.
2. No se altere radicalmente la arquitectura de la máquina.

Ejemplo: optimización del programa anterior

```
1   natural c = 0;

3   while (a != 0) {
4     if (a % 2 != 0)
5       c = c + b;
6     a = a / 2; b = b * 2;
7   }
8   return c;
```

```
1   register natural c = 0;

3   while (a) {
4     if (a & 1)
5       c += b;
6     a >>= 1; b <<= 1;
7   }
8   return c;
```

Por el principio de invarianza, los tiempos diferirán (aprox.) en un factor constante independiente de los números escogidos.

Definición informal de orden

Definición

Dos algoritmos de tiempos $t_1(n)$ y $t_2(n)$ tienen el mismo orden si existen dos constantes reales y positivas, c_1 y c_2, tales que:

$$c_1 \leqslant \frac{t_1(n)}{t_2(n)} \leqslant c_2$$

Advertencia

Esta definición es algo imprecisa: la formalizaremos con los conceptos de tiempo algorítmico y orden asintótico.

Referencias

- Brassard, Gilles y Bratley, Paul.
  Fundamentos de Algoritmia.
  Prentice-Hall. 1997.
- Levitin, Anany V.
  Introduction to the design and analysis of algorithms.
  Addison-Wesley. 2ª ed. 2007.
- Neapolitan, Richard y Naimipour, Kumarss.
  Foundations of algorithms.
  Jones and Bartlett. 4ª ed. 2009.
- Sedgewick, Robert y Wayne, Kevin.
  Algorithms.
  Addison-Wesley. 4ª ed. 2011.