Clase 15 - Algoritmos y complejidad

IIC1001 - Algoritmos y Sistemas Computacionales

Cristian Ruz – `cruz@uc.cl`

Lunes 27-Mayo-2024

Departamento de Ciencia de la Computación
Escuela de Ingeniería
Pontificia Universidad Católica de Chile

Contenidos

Contenidos

I2, Miércoles 29-Mayo 2024, 17:30

- Multiplexores / demultiplexores (sumadores, restadores, etc)
- ALU (instrucciones, registros)
- Instrucciones en computador (ejecución, flujo)
- Sistemas operativos (procesos, memoria)
- Complejidad algorítmica

Contenidos

Complejidad algorítmica

Algorithm 4 Selection Sort

for $i = 1$ to $n - 1$ **do**
    $min = i$
    for $j = i + 1$ to n **do**
        // Find the index of the i^{th} smallest element
        if $A[j] < A[min]$ **then**
            $min = j$
        end if
    end for
    Swap $A[min]$ and $A[i]$
end for

Complejidad algorítmica

```
INSERTION-SORT(A)
for j = 2 to A.length
    key = A[j]
    // Insert A[j] into the sorted sequence A[1 .. j - 1].
    i = j - 1
    while i > 0 and A[i] > key
        A[i + 1] = A[i]
        i = i - 1
    A[i + 1] = key
```

Complejidad algorítmica

BUBBLESORT(A)

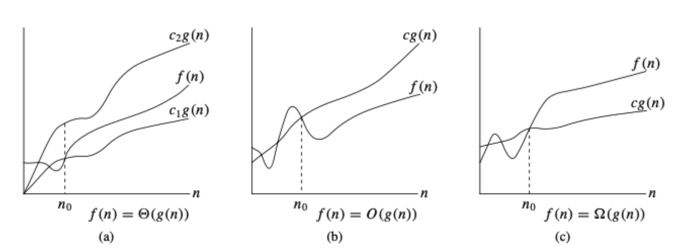
for $i \leftarrow 1$ **to** $\mathit{length}[A]$
    do for $j \leftarrow \mathit{length}[A]$ **downto** $i + 1$
        do if $A[j] < A[j-1]$
            then exchange $A[j] \leftrightarrow A[j-1]$

INSERTION-SORT(A)	*cost*	*times*
for $j \leftarrow 2$ **to** *length*$[A]$	c_1	n
do *key* $\leftarrow A[j]$	c_2	$n-1$
$\triangleright$ Insert $A[j]$ into the sorted sequence $A[1\,..\,j-1]$.	0	$n-1$
$i \leftarrow j-1$	c_4	$n-1$
while $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
do $A[i+1] \leftarrow A[i]$	c_6	$\sum_{j=2}^{n}(t_j - 1)$
$i \leftarrow i-1$	c_7	$\sum_{j=2}^{n}(t_j - 1)$
$A[i+1] \leftarrow key$	c_8	$n-1$

Notación asintótica nos permite establecer el comportamiento de un algoritmo "en el largo plazo"

$$\Theta(g(n)) = \{f(n) : \text{ there exist positive constants } c_1, c_2, \text{ and } n_0 \text{ such that } 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ for all } n \geq n_0\} .$$ [1]

Notación asintótica nos permite establecer el comportamiento de un algoritmo "en el largo plazo"

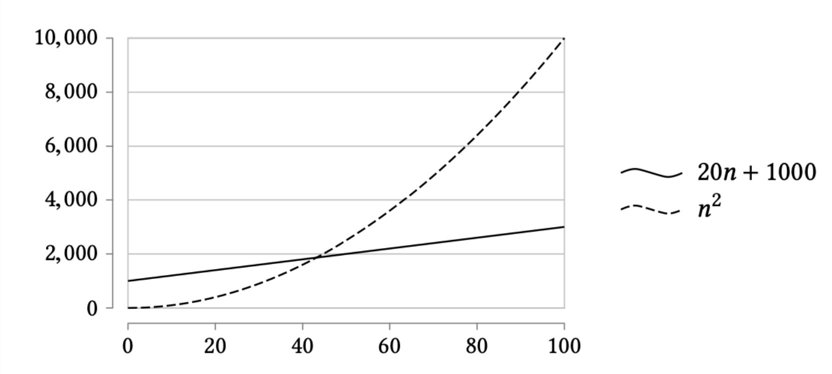


$$\Theta(g(n)) = \{f(n) : \text{ there exist positive constants } c_1, c_2, \text{ and } n_0 \text{ such that } 0 \le c_1 g(n) \le f(n) \le c_2 g(n) \text{ for all } n \ge n_0\}\ .^1$$

$$O(g(n)) = \{f(n) : \text{ there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \le f(n) \le cg(n) \text{ for all } n \ge n_0\}\ .$$

$$\Omega(g(n)) = \{f(n) : \text{ there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \le cg(n) \le f(n) \text{ for all } n \ge n_0\}\ .$$

Complejidad algorítmica



Complejidad algorítmica

Growth of functions.

Function	Input size				
	1	10	100	1000	1,000,000
$\lg(n)$	0	3.32	6.64	9.97	19.93
n	1	10	100	1000	1,000,000
$n\ln(n)$	0	33.22	664.39	9965.78	1.9×10^7
n^2	1	100	10,000	1,000,000	10^{12}
n^3	1	1000	1,000,000	10^9	10^{18}
2^n	2	1024	1.3×10^{30}	10^{301}	$10^{10^{5.5}}$
$n!$	1	3,628,800	9.33×10^{157}	4×10^{2567}	$10^{10^{6.7}}$

Complejidad algorítmica