Complejidad Temporal

Paolo Rosso
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica Valencia, España

14 de Agosto de 2001
Dpto. Informática
Universidad Técnica Federico Santa María, Valparaíso, Chile

Contenido

- Introducción
- Estudio de la Complejidad (temporal)
- Ejemplo de la Computación Numérica
- Ejercicios propuestos
- El paradigma de la Programación Paralela

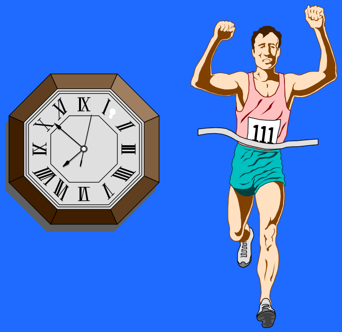
Introducción

- Tamaño/talla del problema
- Complejidad (temporal) de los algoritmos (que resuelven el problema)
- Notación asintótica: Θ, Ω, O
- *Análisis a priori y a posteriori*
- *Análisis de la complejidad en FLOPS (FLOating Point operationS)*

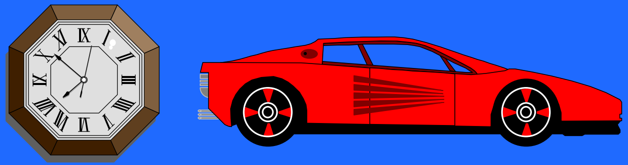
Introducción

Talla del problema: x metros

Algoritmo a pie (x:entero)



Algoritmo en coche (x:entero)



Introducción

Problema: n*n **Talla del Problema: n**

Algoritmo A1: **Función A1 (n:entero) devuelve entero;**
var m: entero finvar;
m:=n*n; devuelve m
finfunción;

Algoritmo A2: **Función A2 (n: entero) devuelve entero;**
var m, i: entero finvar;
m:=0; para i:=1 hasta n hacer m:=m+n finpara;
devuelve m
finfunción;

Algoritmo A3: **Función A3 (n: entero) devuelve entero;**
var m, i, j: entero finvar;
m:=0; para i:=1 hasta n hacer
para j:=1 hasta n hacer m:=m+1 finpara
finpara;
devuelve m
finfunción;

Introducción

		$\underline{t_{as} \quad t_{co} \quad t_{op}}$
A1:	m:=n*n;	$t_{as}+t_{op}$
A2:	m:=0;	t_{as}
	para i:=1 to n do	$t_{as}+n\,t_{co}$
	m:=m+n;	$n\,(t_{as}+t_{op})$
A3:	m:=0;	t_{as}
	para i:=1 to n do	$t_{as}+n\,t_{co}$
	para j:=1 to n do	$n\,t_{as}+n^2\,t_{co}$
	m:=m+1;	$n^2\,(t_{as}+t_{op})$

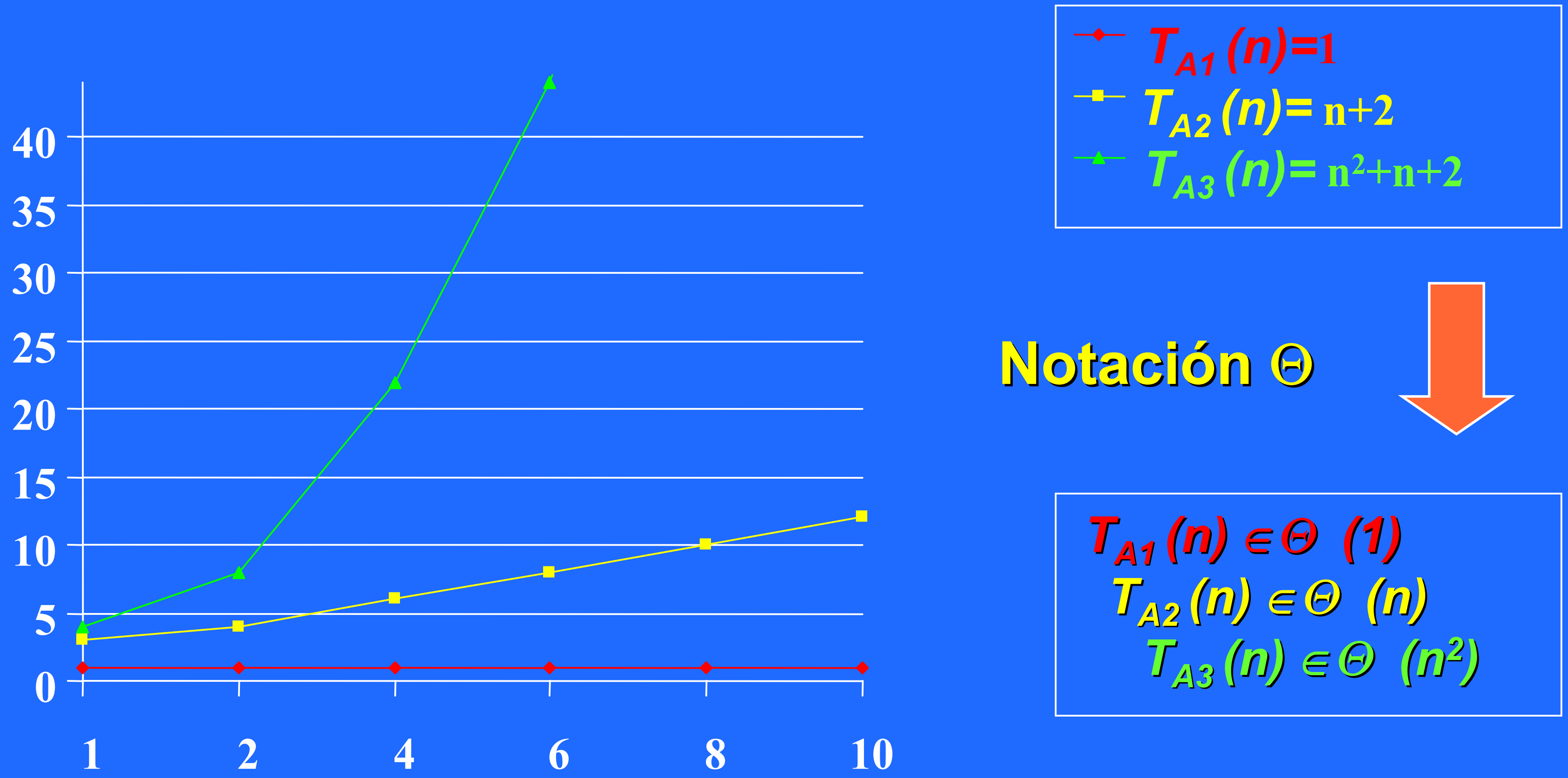
$T_{A1}=t_{op}+t_{as} = k_1 \Rightarrow T_{A1}(n)= 1$ *paso*

$T_{A2}=n(t_{op}+t_{co})+(n+2)\,t_{as}= k_2*n+k_3 \Rightarrow T_{A2}(n)= n+2$ *pasos*

$T_{A3}=n^2t_{op}+(n^2+n+2)\,t_{as}+(n^2+n)\,t_{co} = k_4*n^2+k_5*n+k_6 \Rightarrow$

$T_{A3}(n)= n^2+n+2$ *pasos*

Introducción



Notación Θ

$T_{A1}(n) \in \Theta(1)$

$T_{A2}(n) \in \Theta(n)$

$T_{A3}(n) \in \Theta(n^2)$

Introducción

Problema: Búsqueda elemento en un vector de n elementos

Talla: n

```
función Secuencial (v:vector[1..n] de enteros; x:entero)  devulve lógico;
var enc:lógico; i:entero finvar;
        i:=1; enc:=falso;
        mientras ((i<=n) ∧ not enc) entonces
                si v[i]=x entonces enc:=cierto {instrucción critica}
                sino i:=i+1 finsi;
        devuelve enc
finfunción;
```

$T(n) \in \Omega(1)$
$T(n) \in O(n)$

Introducción

```
{Vector Ordenado Ascendentemente}
función Binaria (v:vector [1..n] de enteros; x: entero) devuelve lógico;
var  izq, der, med: entero; enc:lógico finvar;
        izq:=1; der:=n; enc:=falso;
        repetir
                med:=(izq+der) div 2;
                si v[med] =x entonces enc:=cierto {instrucción critica}
                sino      si v[med] > x entonces der:=med - 1
                          sino izq:=med + 1
                          finsi
                finsi
        hasta enc ∨ (izq>der);
        devuelve enc
finfunción;
```

$T(n) \in \Omega(1)$

$T(n) \in O(\log_2 n)$

Introducción

Notación O

Definición: Se dice que:

g es como mucho del orden de f, o que

g crece más lentamente o igual que f, o que

f es una cota superior de g,

sii $\exists\ c>0,\ \exists\ n_0 \in N:\ g(n) \leq c\ f(n)\ \forall\ n \geq n_0$

f, g, f_1, f_2, y f_3 serán funciones de N en R^+

O(f(n)) = {funciones que son como mucho del orden de f(n)}

$3n \in O(n^2)$: $c=3,\ n_0=0\ \ g(n) = 3n \leq c \cdot f(n) = 3n^2\ \ \forall n \geq 0$

$3n \in O(n)$: $c=3,\ n_0=0\ \ g(n) = 3n \leq 3n = c \cdot f(n)\ \ \forall n \geq 0$

$n \in O(n)$: $c=1,\ n_0=0\ \ g(n) = n \leq n = c \cdot f(n)\ \ \forall n \geq 0$

Introducción

Notación Ω

Definición: Se dice que:

g es como mínimo del orden de f, o que

g crece más rapidamente o igual que f, o que

f es una cota inferior de g,

sii $\exists\, c>0, \exists\, n_0 \in N: c\, f(n) \leq g(n) \quad \forall n \geq n_0$

$\Omega\,(f(n))$ *={funciones que son como mínimo del orden de f(n)}*

$3n \in \Omega(n)$: $c=1, n_0=0$ $c \cdot f(n) = 1 \cdot n \leq 3n = g(n)\ \forall n \geq 0$

$n^2 \in \Omega(3n)$: $c=1, n_0=3$ $c \cdot f(n) = 1 \cdot 3n \leq n^2 = g(n)\ \forall n \geq 3$

$n^2 \in \Omega(n)$: $c=1, n_0=0$ $c \cdot f(n) = 1 \cdot n \leq n^2 = g(n)\ \forall n \geq 0$

$n^2 \in \Omega(n^2)$: $c=1, n_0=0$ $c \cdot f(n) = 1 \cdot n^2 \leq n^2 = g(n)\ \forall n \geq 0$

Introducción

Notación Θ

<u>**Definición**</u>: Se dice que:

g es del orden de f, o que

g crece más igual de rapidamente que f, o que

f es una cota inferior e superior de g,

sii $\exists\ c,d > 0, \exists\ n_0 \in N: c\, g(n) \leq f(n) \leq d\, g(n)\ \ \forall\ n \geq n_0$

$\Theta(f(n))$ = {funciones que son del orden de f(n)}

$3n \in \Theta(n)$: $3n \in O(n)$, $3n \in \Omega(n)$

$d=3,\ n_0=0\ \ f(n) = 3n \leq 3n = d \cdot g(n)\ \ \forall n \geq 0$

$c=1,\ n_0=0\ \ c \cdot g(n) = 1 \cdot n \leq 3n = f(n)\ \forall n \geq 0$

Introducción CNU ⟵ ALG

- Se estudian algoritmos para implementar métodos numéricos:

  Ejemplos
  - Sistemas lineales
  - Interpolación
  - Integración, etc.
- Estos problemas se caracterizan por:
  - Elevada complejidad temporal ($O(n^3)$)
  - Elevado tamaño de los datos
  - Se requiere una solución en un tiempo limitado
  - (ejemplo: Predicción Meteorológica)
- Las estrategias de diseño de los algoritmos numéricos son las mismas que las estrategias utilizadas en ALG

Estudio de la Complejidad

Tiempos de ejecución en una máquina capaz de ejecutar 10^9 pasos de programa por segundo (1000 MIPS)

	n (Talla)					
Pasos	1	8	32	10^3	10^6	10^9
$\log_2 n$	< 1 ns	3 ns	5 ns	10 ns	20 ns	30 ns
n	1 ns	8 ns	32 ns	1 µs	1 ms	1 s
$n\log_2 n$	< 1 ns	24 ns	160 ns	10 µs	20 ms	30 s
n^2	1 ns	64 ns	1 µs	1 ms	17 min	38 años
n^3	1 ns	512 ns	33 µs	1 s	38 años	> 10^{10} años
2^n	2 ns	256 ns	4.3 s	> 10^{291} años	> 10^{300K} años	>10^{300M} años
n!	1 ns	40 µs	> 10^{18} años	> 10^{2K} años	> 10^{5M} años	>10^{9000M} años
n^n	1 ns	17 ms	> 10^{31} años	>10^{3K} años	> 10^{6M} años	>10^{9000M} años

Estudio de la Complejidad

Dos tipos de análisis:

- **A priori** (asintótico)
  - Independiente de la máquina y del lenguaje utilizado, así como de la carga de la misma
  - Dependiente del diseño del algoritmo y de los datos

    Dos tipos:
    - ALG: Instrucción crítica (MIPS)
    - CNU: Número de FLOPS
- **A posteriori** (técnica de temporización)

  Dependiente de la máquina utilizada, así como de la carga de la misma

Estudio de la Complejidad

Situación Real

Se está trabajando en una empresa y el jefe plantea alguna de las dos situaciones siguientes:

A. La empresa dispone de una máquina y hay que averiguar si un algoritmo se ejecutará en menos de un tiempo dado

B. Hay que averiguar qué máquina le interesa comprar a la empresa para que un algoritmo se ejecute en menos de un tiempo dado

¿Qué tipo de análisis de complejidad interesa utilizar para cada situación?

Ejemplo de Computación Numérica

Cálculo del Valor Numérico de un Polinomio

Descripción del problema

Dado el polinomio

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$$

se quiere calcular el valor de $P_n(x)$ en el punto x_0, es decir,

$$P_n(x_0) = a_n x_0^n + a_{n-1} x_0^{n-1} + \dots + a_1 x_0 + a_0$$

Ejemplo de Computación Numérica
Cálculo del Valor Numérico de un Polinomio

Diseño de la estructura de datos

- Es necesario definir una estructura de datos que permita representar al polinomio $P_n(x)$
- Lo más adecuado es utilizar un vector donde se almacenen los *n+1* coeficientes: a_0, a_1, a_2,..., a_n

```
tipo
        polinomio=vector [0..n] de real;
fintipo
...
```

Ejemplo de Computación Numérica
Cálculo del Valor Numérico de un Polinomio

Algoritmo 1: Iterativo, coste cuadrático con la talla

```
funcion polIt(e/s p:polinomio; x:real) devuelve real;

var s:real; i:entero; finvar
s:=0;
para i:=0 hasta n
        s:=s+p[i]*potencia(x,i);
finpara
devuelve s;
finfuncion
```

```
funcion potencia(base:real;exp:entero)
                                devuelve real;
var j: entero; res: real; finvar
        res:=1;
        para j:=1 hasta exp
                res:=res*base;
        finpara
        devuelve res;
finfuncion
```

Ejemplo de Computación Numérica
Cálculo del Valor Numérico de un Polinomio

Algoritmo 1: Análisis del Coste

$$t_{potencia}(\exp) \in \Theta(\exp)$$

$$t_{polIt}(n) = \sum_{i=0}^{n} t_{potencia}(i) = \sum_{i=0}^{n} \Theta(i) \Rightarrow$$

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2} \in \Theta(n^2)$$

Flops

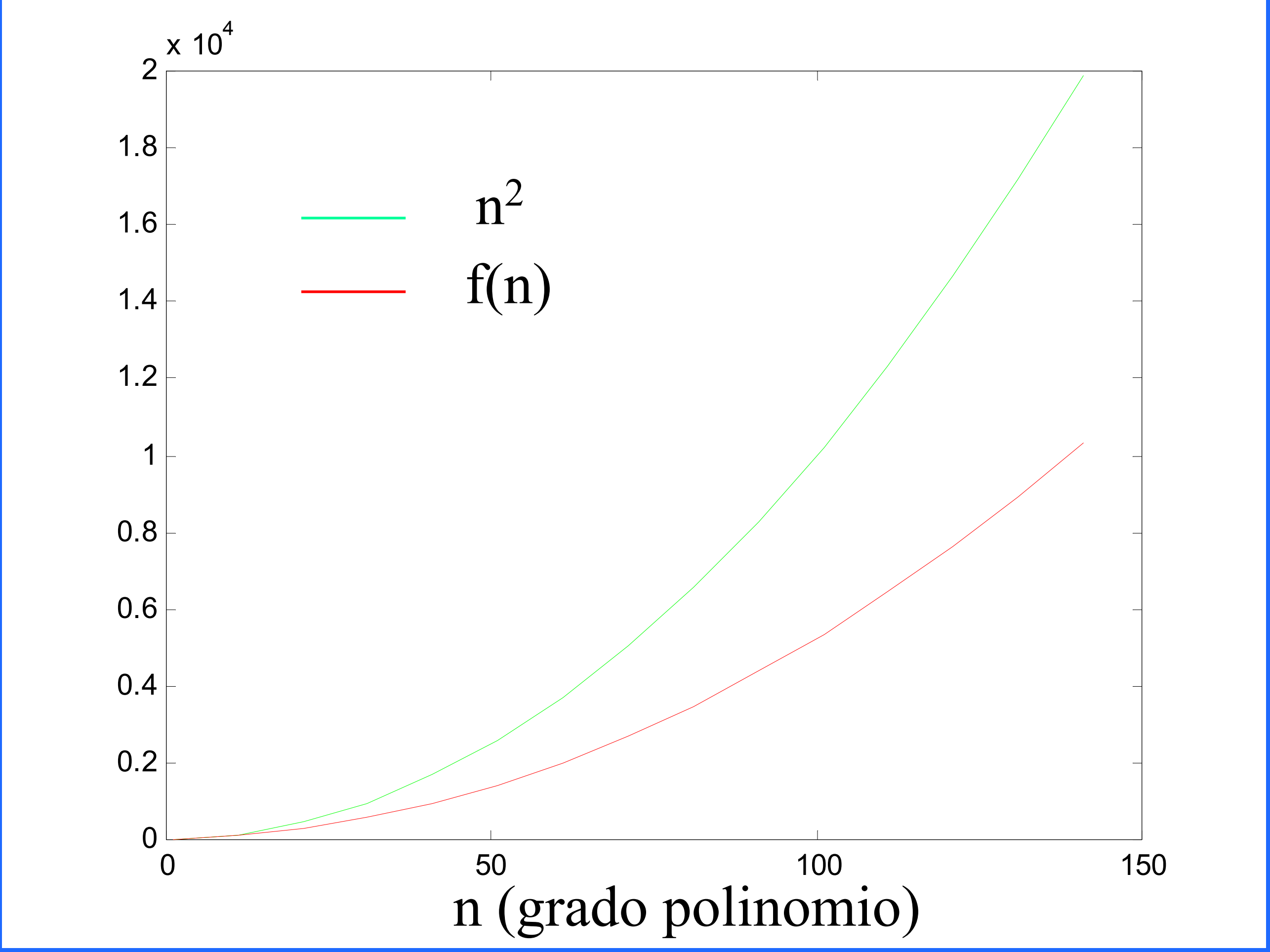
$$f_{potencia}(\exp) \equiv \exp$$

$$f_{polIt}(n) = \sum_{i=0}^{n} 2 + f_{potencia}(i) = \sum_{i=0}^{n} (2+i) = 2n + \frac{n(n+1)}{2} \approx \frac{n^2}{2} \in \Theta(n^2)$$

Ejemplo de Computación Numérica

Cálculo del Valor Numérico de un Polinomio

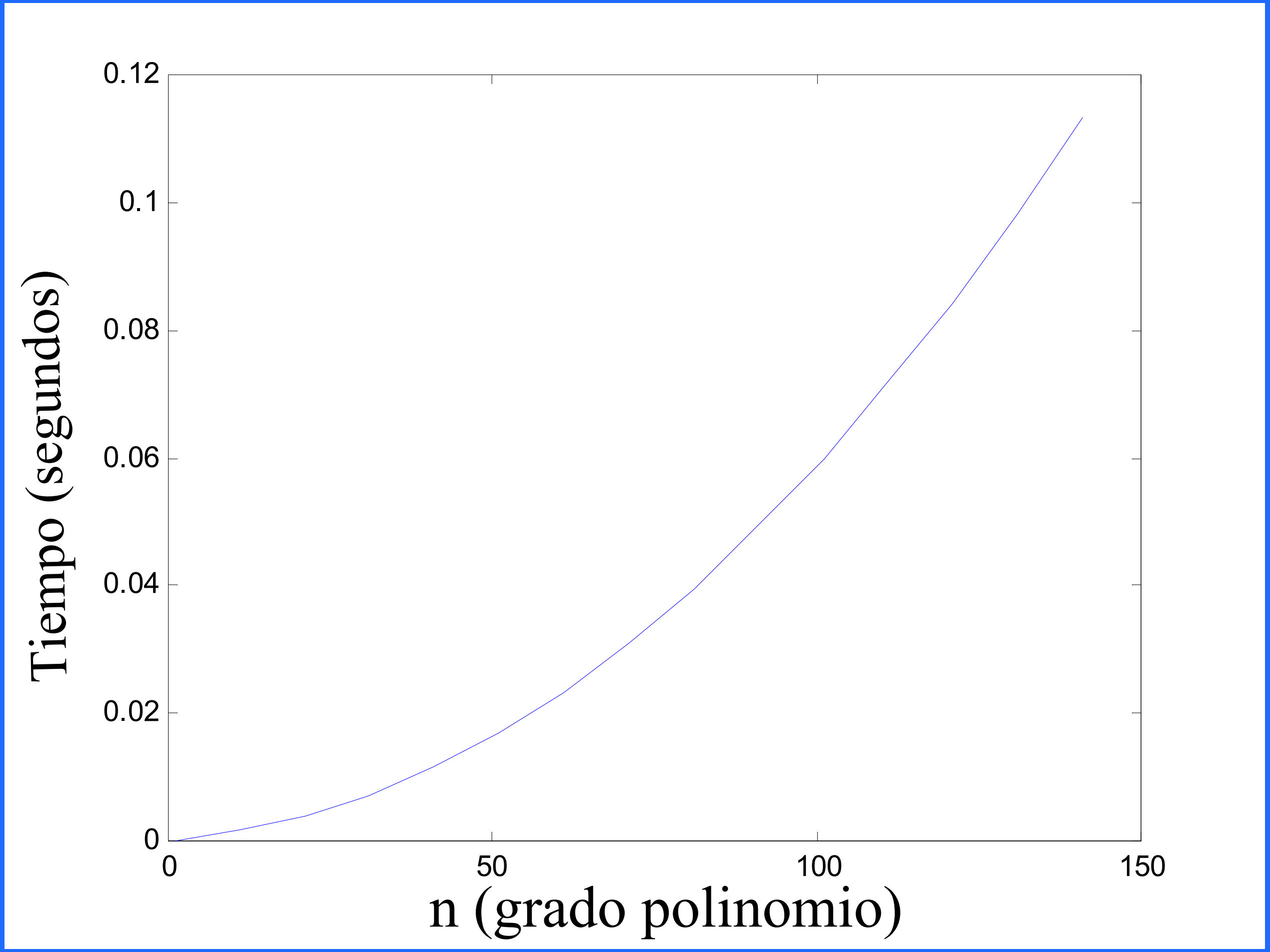
Análisis del Coste del Algoritmo 1 (A priori)



Ejemplo de Computación Numérica

Cálculo del Valor Numérico de un Polinomio

Análisis del Coste del Algoritmo 1 (A posteriori)



Ejemplo de Computación Numérica

Cálculo del Valor Numérico de un Polinomio

Ejemplo: $n = 4$

$$P_4(x_0) = a_4 x_0^4 + a_3 x_0^3 + a_2 x_0^2 + a_1 x_0 + a_0 =$$

$$(a_4 x_0^3 + a_3 x_0^2 + a_2 x_0 + a_1) x_0 + a_0 =$$

$$((a_4 x_0^2 + a_3 x_0 + a_2) x_0 + a_1) x_0 + a_0 =$$

$$(((\overbrace{a_4}^{s^{(4)}} x_0 + a_3) x_0 + a_2) x_0 + a_1) x_0 + a_0$$

$$s^{(3)} = s^{(4)} * x_0 + a_3$$

$$s^{(2)} = s^{(3)} * x_0 + a_2$$

$$s^{(1)} = s^{(2)} * x_0 + a_1$$

$$s^{(0)} = s^{(1)} * x_0 + a_0$$

⇨ Método de la multiplicación encajada

Ejemplo de Computación Numérica

Cálculo del Valor Numérico de un Polinomio

Algoritmo 2: Iterativo, coste lineal con la talla

```
funcion polIt2 (e/s p:polinomio; x:real) devuelve real;

var s:real, i:entero finvar
        s:=p[n];
        para i:=n-1 decreciendo hasta 0
                s:=s*x+p[i];
        finpara
        devuelve s;
finfuncion
```

Ejemplo de Computación Numérica

Cálculo del Valor Numérico de un Polinomio

Algoritmo 2: Análisis del Coste

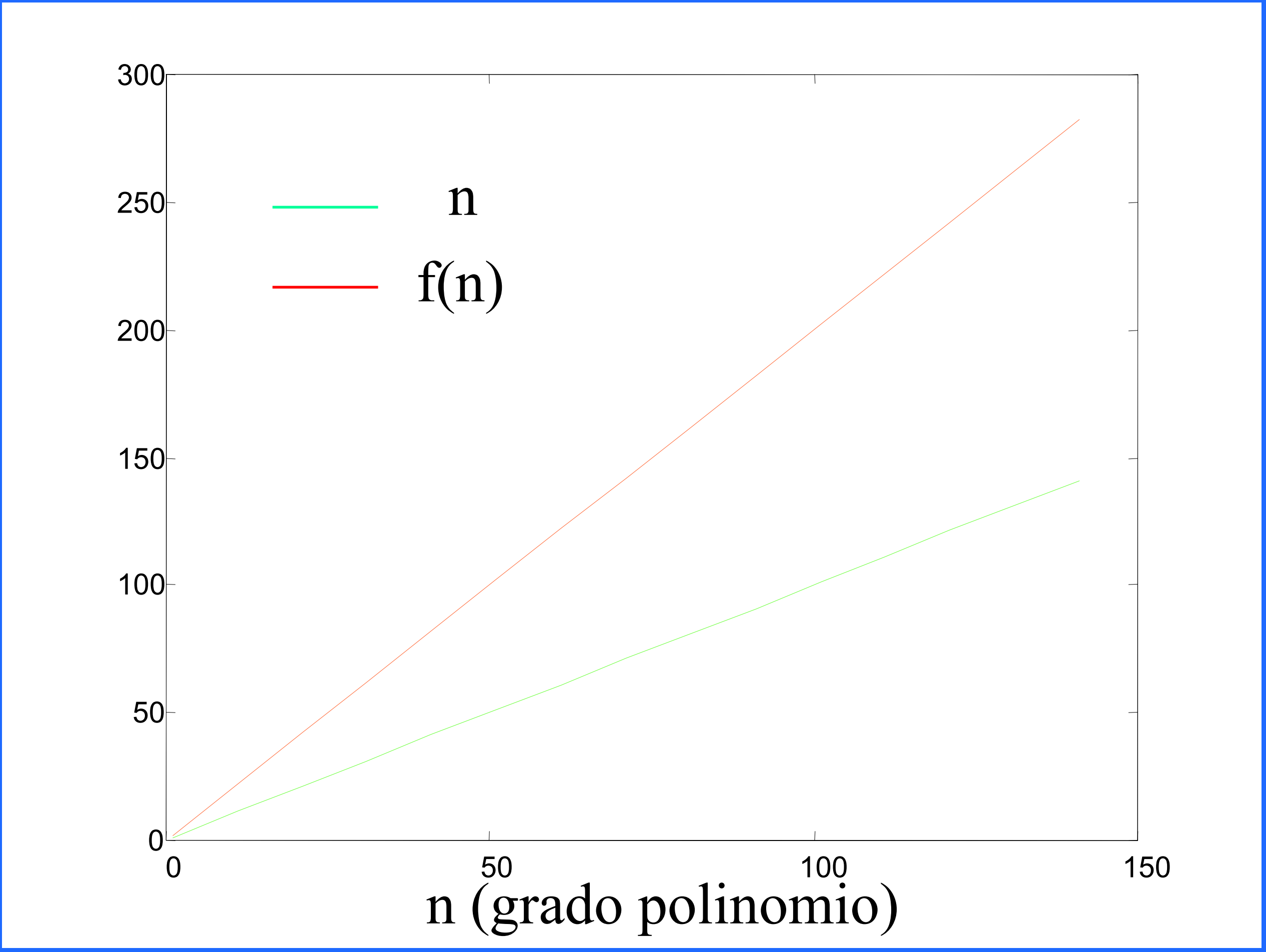
$$t_{polIt2}(n) \; : \; \sum_{i=1}^{n} 1 = n \in \Theta(n)$$

Flops

$$f_{polIt2}(n) \; : \; \sum_{i=1}^{n} 2 = 2n \in \Theta(n)$$

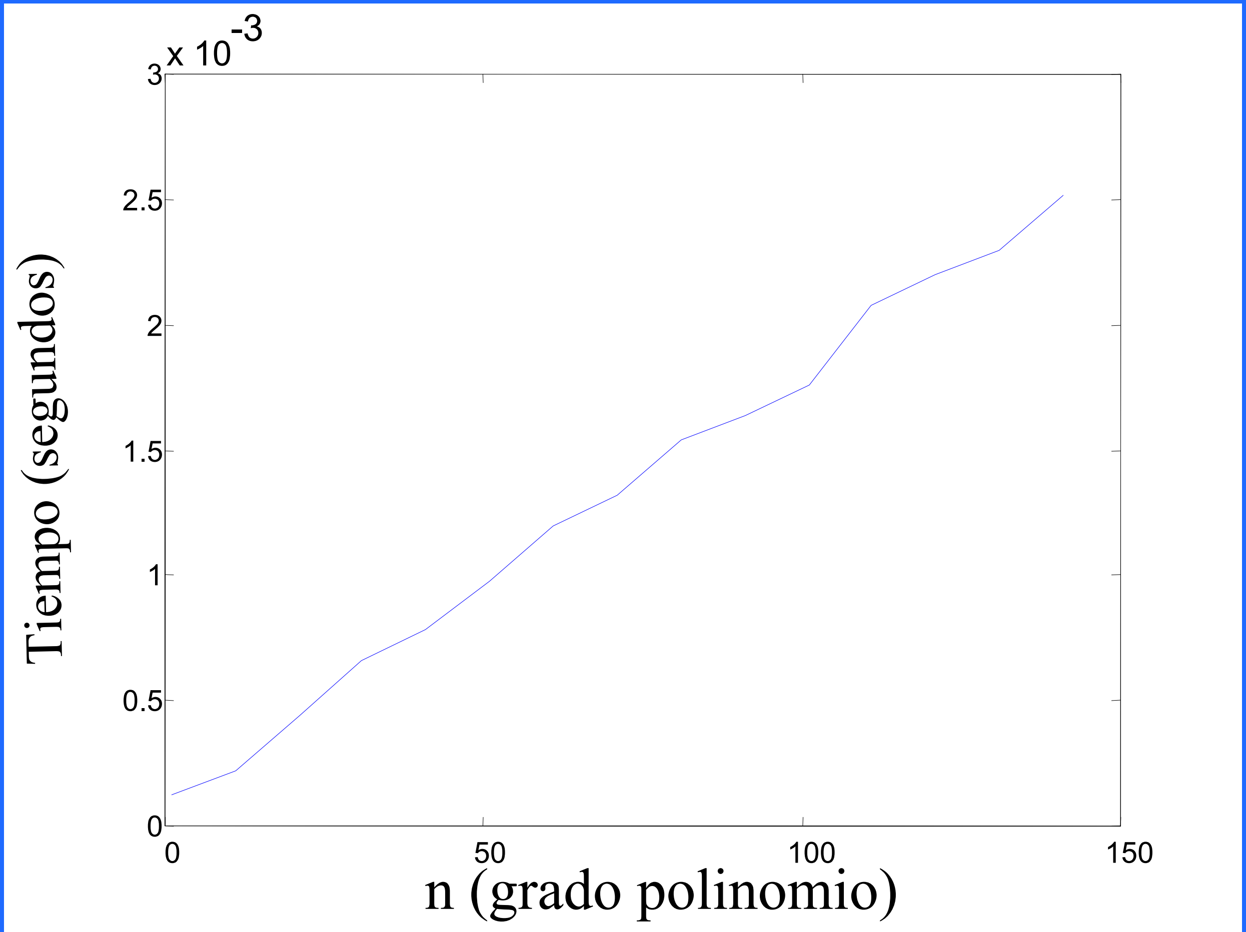
Ejemplo de Computación Numérica
Cálculo del Valor Numérico de un Polinomio

Análisis del Coste del Algoritmo 2 (a priori)



Ejemplo de Computación Numérica
Cálculo del Valor Numérico de un Polinomio

Análisis del Coste del Algoritmo 2 (a posteriori)



Ejemplo de Computación Numérica
Cálculo del Valor Numérico de un Polinomio

Algoritmo 3: Recursivo, coste lineal con la talla

```
funcion polRe (e/s p:polinomio; x:real; i: entero) devuelve real;
var s: real; finvar
        si i=n    entonces s:=p[n]
                  sino s:=p[i]+x*polRe(p,x,i+1)
        finsi
        devuelve s
finfuncion
```

(Llamada inicial con i=0)

Talla: $m \approx n-i$

$$t(m) = \begin{cases} k_1 & m=0 \\ k_2 + t(m-1) & m>0 \end{cases}$$

Ejemplo de Computación Numérica

Cálculo del Valor Numérico de un Polinomio

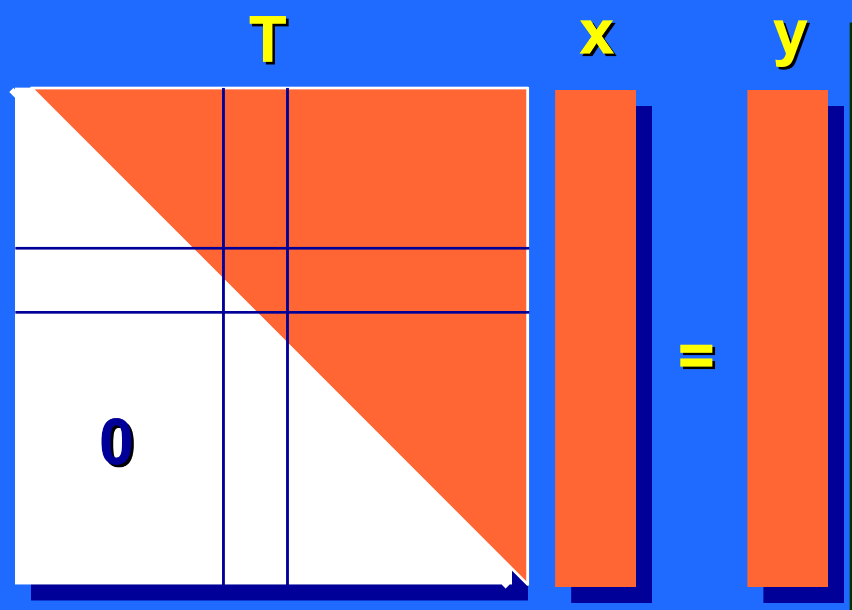
Algoritmo 3': Recursivo

```
tipo
        polinomio_ex=vector [0..NMax] de real;
fintipo


funcion polRe (e/s p:polinomio_ex; x:real; i, n: entero) devuelve real;
                    {ídem}
finfuncion
```

Ejercicios Propuestos

Producto de una Matriz Cuadrada por un Vector




```
proc mv2(T:vector[1..n,1..n] de
reales; x:vector[1..n] de reales;
salida y: vector[1..n] de reales);
var i, j: entero finvar
para i:=1 hasta n hacer
        y[i]:=0;
        para j:=i hasta n hacer
                y[i]:=y[i] + T[i,j]
                * x[j]
        finpara
finpara
finproc
```

$$t_{mv2}(n) = \sum_{i=1}^{n}\sum_{j=i}^{n}\Theta(1) \Rightarrow \sum_{i=1}^{n}\sum_{j=i}^{n}1 = \sum_{i=1}^{n}(n-i+1) = n^2 - \sum_{i=1}^{n} i + n = \frac{n^2}{2} + \frac{n}{2} \Rightarrow t_{mv2}(n) \in \Theta(n^2)$$

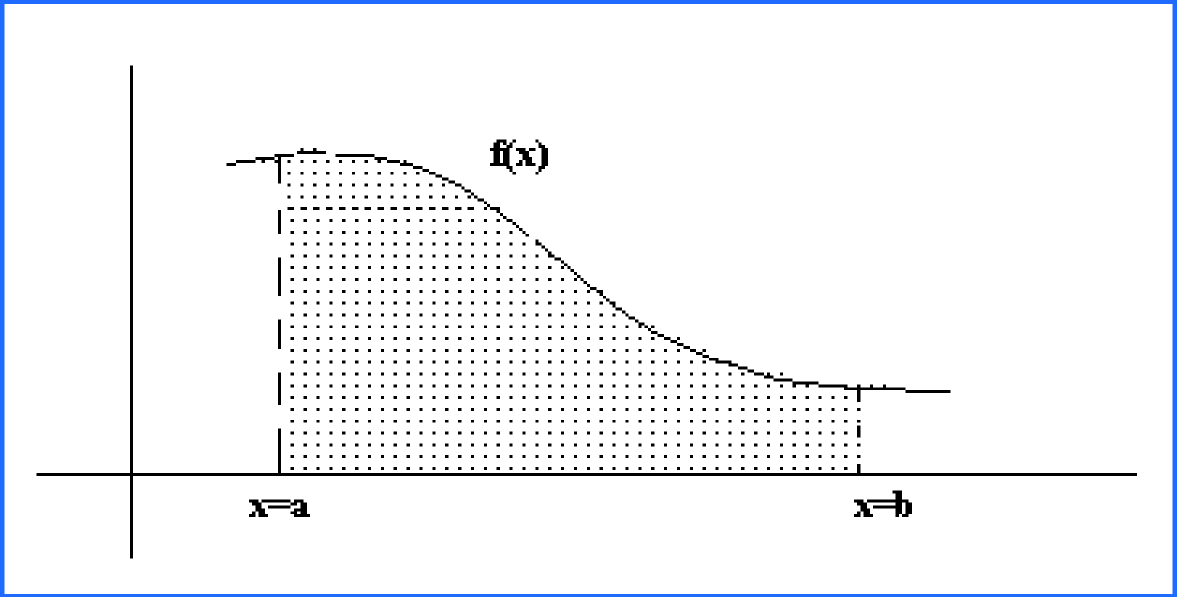
$$f_{mv2}(n) = \sum_{i=1}^{n}\sum_{j=i}^{n}2 = \sum_{i=1}^{n}2(n-i+1) = n^2 + n \Rightarrow f_{mv2}(n) \approx n^2$$

Ejercicios Propuestos

Cálculo de la Integral con el Método de los Rectángulos

Descripción del problema

- Dada la función f(x), se pretende calcular la integral de dicha función en el intervalo [a,b]
- Esta se define como el área dibujada en la figura

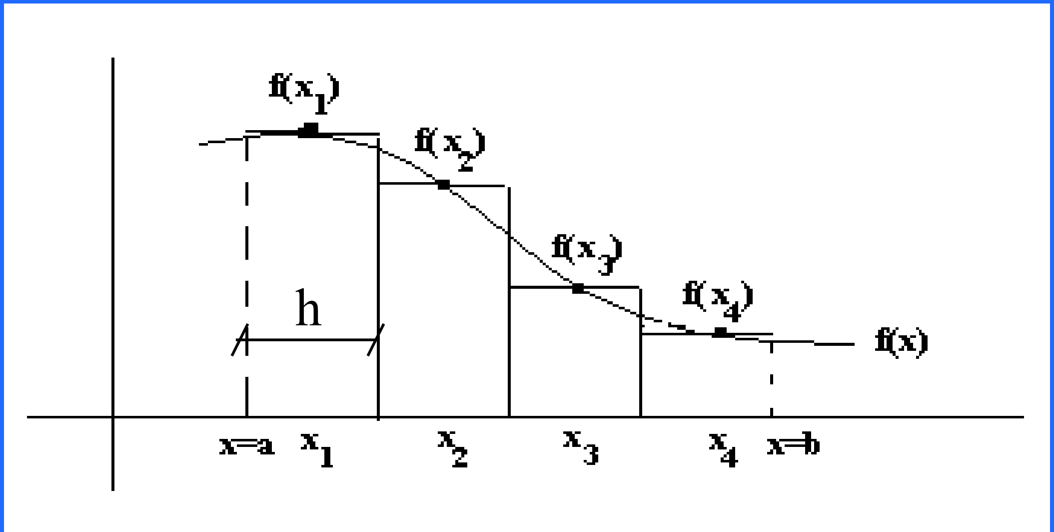


Ejercicios Propuestos

Cálculo de la Integral con el Método de los Rectángulos

Método a utilizar: Rectángulos

- Consiste en dividir el intervalo [a,b] en N intervalos de tamaño h=(b-a)/N. En el ejemplo de la figura N=4



- La integral se aproxima mediante la suma de los rectángulos de la figura
- Si se llama x_i al punto medio de cada intervalo i, entonces el área del rectángulo i será $h*f(x_i)$, i=1, 2, 3 y 4

Ejercicios Propuestos

Cálculo de la Integral con el Método de los Rectángulos

Se pide

a) Implementa un algoritmo que implemente el método de los rectángulos utilizando un esquema iterativo. Supondremos que la función $f(x)$ es un polinomio de grado n, es decir $f(x)=P_n(x)$. **Entradas**: P_n, h, a, b

b) Implementa un algoritmo que implemente el método de los rectángulos utilizando un esquema recursivo. Supondremos que la función $f(x)$ es un polinomio de grado n, es decir $f(x)=P_n(x)$. **Entradas**: P_n, h, I, J (Llamada inicial con I=a y J=b)

c) Analiza asintóticamente el coste temporal de ambos algoritmos

Ejercicios Propuestos

Cálculo de la Integral con el Método de los Rectángulos

$f(x)=a_m x^m + a_{m-1}x^{m-1} + \ldots + a_1 x + a_0$

Talla subproblema: m

Complejidad algoritmo: $t(m) \in \Theta(m)$

Talla problema: n, m

n intervalos (e.g. n=4) de tamaño $h=(b-a)/n$

$t(n,m) \in \Theta(n*m)$

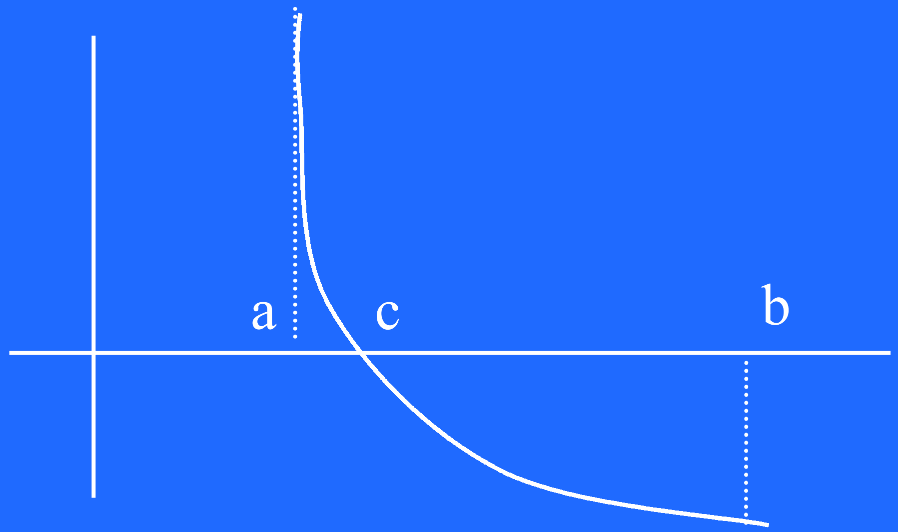
si $n >> m \approx t(n) \in \Theta(n)$

Ejercicios Propuestos

Cálculo de Raíces Mediante el Método de Bisección

Descripción del problema

Dada la función f(x), se pretende calcular una raíz de dicha función en el intervalo [a,b]. Es decir, calcular el punto c donde f(c)=0



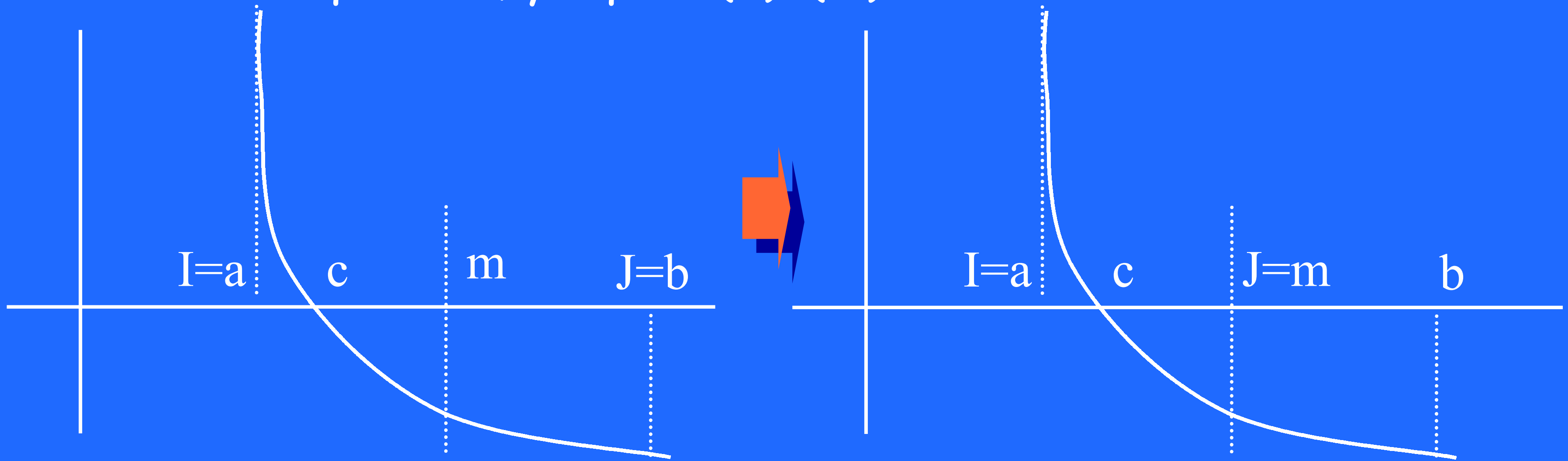
Para asegurar que la función f(x) tiene, al menos, una raíz en el intervalo [a,b] debe cumplirse que f(a)f(b)<0 (Teorema de Bolzano)

Ejercicios Propuestos

Cálculo de Raíces Mediante el Método de Bisección

Método a utilizar: Bisección

- El método de Bisección divide el intervalo inicial [a,b]=[I,J] (inicialmente I=a y J=b) en dos intervalos de igual tamaño [I,m] y [m,J]
- De esos dos intervalos se selecciona aquel en el que se satisface el teorema de Bolzano. En el caso de la figura el intervalo izquierdo, ya que $f(I)f(m) < 0$



Ejercicios Propuestos

Cálculo de Raíces Mediante el Método de Bisección

Se pide

a) Implementa un algoritmo que implemente el método de Bisección utilizando un esquema iterativo. Supondremos que la función $f(x)$ es un polinomio de grado n, es decir $f(x)=P_n(x)$. **Entradas**: P_n, a, b y tol (tolerancia)

b) Implementa un algoritmo que implemente el método de Bisección utilizando un esquema recursivo. Supondremos que la función $f(x)$ es un polinomio de grado n, es decir $f(x)=P_n(x)$. **Entradas**: P_n, I, J (llamada inicial I=a y J=b) y tol (tolerancia)

c) Analiza asintóticamente el coste temporal de ambos algoritmos

Ejercicios Propuestos

Cálculo de Raíces Mediante el Método de Bisección

Talla problema: n,p

$n = (b-a)/\varepsilon$

p: $f(x)=a_p x^p + a_{p-1}x^{p-1} + \ldots + a_1 x + a_0$

$t(n,p) \in \Omega(p) \quad t(n,p) \in O(p * \log_2 n)$

si $n >> p \approx t(n) \in \Omega(1)$ $t(p) \in O(\log_2 n)$

Programación Paralela

Multicomputadores con memoria distribuida