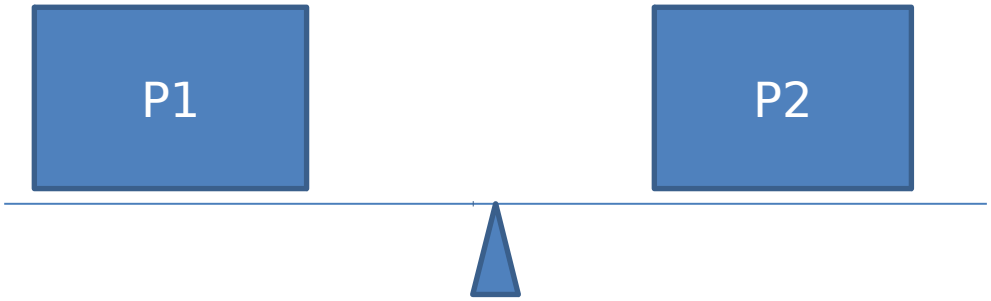
Análisis de Algoritmos

Programación Imperativa Modular

jueves, 21 de febrero de 2013

Panorama

- Tenemos dos programas P1 y P2



  - ¿cuál es mejor?
  - Depende de:
  - Tiempo de ejecución?
  - Mayor gasto/consumo de memoria?

Panorama

- Otros factores:
  - Velocidad del computador
  - El compilador

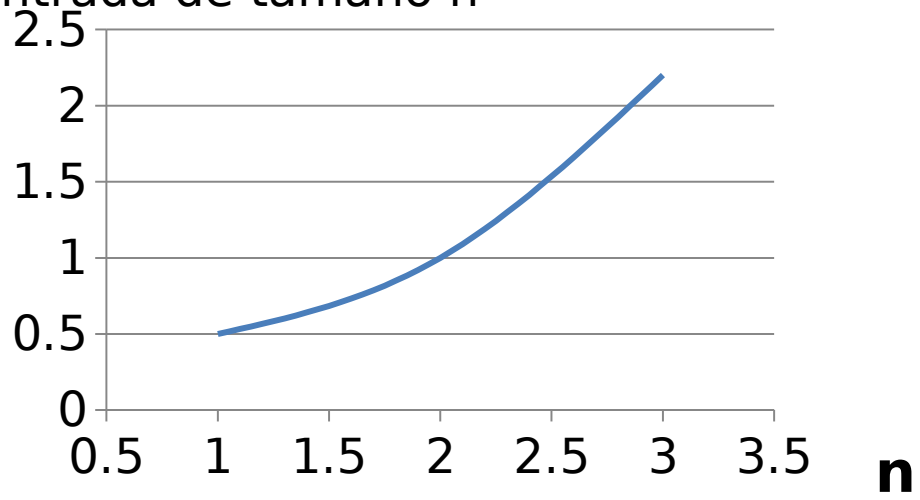
Panorama

- Si no hay programación, algoritmos:
  - Cuál es el mejor?
  - No hay tiempo de ejecución
  - No hay gasto de memoria
- Otros factores:
  - La estructura del algoritmo
  - Cantidad de datos – tamaño datos
  - Ej.: ordenar 5 datos vs ordenar 1000 datos
  - Calcular el factorial de 5 vs calcular el factorial de 500

Tiempo

- Denominamos

$T_A(n)$ → tiempo empleado por el algoritmo A para procesar entrada de tamaño n



Tiempo

- Ejemplo: Decir si un número E está en un arreglo de N elementos.

- Si cada comparación gasta t seg, cuánto gasta el algoritmo en tiempo?
  - Si la entrada es V = [5,6,7,8,9,10] , E = 5, $T_A(6) = ?$
  - Si la entrada es V = [5,6,7,8,9,10] , E = 9, $T_A(6) = ?$

- Tomamos el peor de los casos

Complejidad

- Consiste en encontrar una función f(n) que permita calcular el tiempo de un algoritmo en el peor de los casos, **acotar tiempo de ejecución.**

Complejidad

- Reglas:
  - Si $T_A(n)$ *está en* $O(k\, f(n))$ $\Rightarrow$ $T_A(n)$ *está en* $O(f(n))$

    porque $O(k) = 1$

  Ej.: tiempo de una asignación es constante, independiente del tamaño de los datos

  tiempo de una comparación

  tiempo de una instrucción de E/S

Complejidad

- Reglas:
  - Si tenemos dos algoritmos A y B que se ejecutan uno detrás de otro

$T_A(n)$ *está en* $O(f_1(n))$

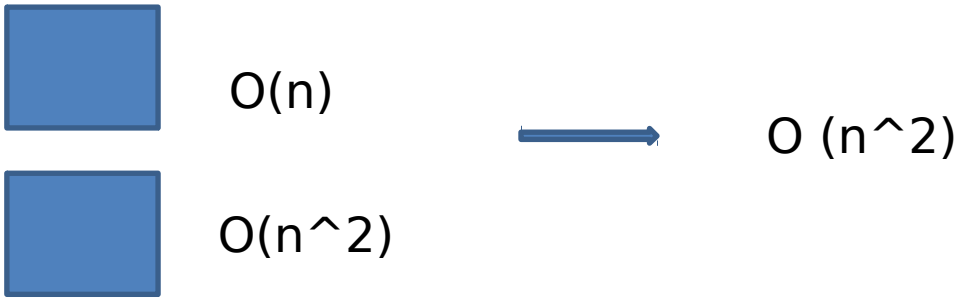
$T_B(n)$ *está en* $O(f_2(n))$

El tiempo total

$T_T(n) = T_A(n) + T_B(n)$ *está en* $O(\text{máx}(f_1(n), f_2(n))$

Complejidad

- Reglas:
  - Si tenemos dos algoritmos A y B que se ejecutan uno detrás de otro



El tiempo total

$$T_T(n) = T_A(n) + T_B(n) \text{ } \textit{está en} \text{ } O(\text{ máx } (\text{ } f_1(n) \text{ , } f_2(n) \text{ }))$$

Complejidad

- Reglas:
  - Si tenemos un condicional no repetitivo

si <cond1> -> <instruc1>
elseif <cond2> -> <instruc2>
elseif <cond3> -> <instruc3>
....
else <instruc-n>
endif

Gasta en el peor de los casos

$T_A(n)$ = T<cond1> + T<cond2> + T<condn> +

máx (T<instruc1>, T<instruc2>, ..,T<instruc-n>)

Complejidad

- Reglas:
  - Si tenemos un condicional repetitivo

haga si <cond> -> <instruc>

finhaga

Gasta en el peor de los casos

$T_A(n)$ = (T<cond1> + T<instruc>) * N-iteraciones