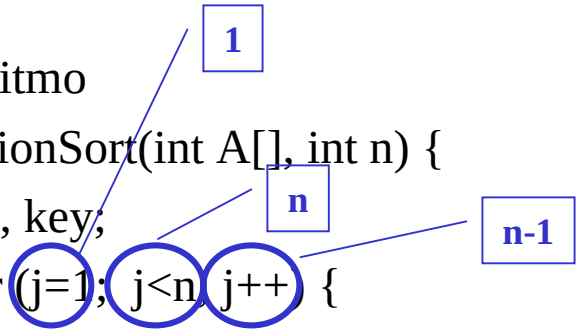
Análisis de algoritmos
Generalidades

Agustín J. González

1er. Sem. 2002.

Análisis de Algoritmos

Análisis de algoritmos consiste en predecir los recursos que el algoritmo requiere




```
Algoritmo
InsertionSort(int A[], int n) {
int j, i, key;
    for (j=1; j<n; j++) {
     key = A[j];
     /* inserta  A[j] en la secuencia ordenada
       A[0...j-1]      */
     i = j-1;
        while( i>=0  && A[i] > key ){
            A[i+1] = A[i];
             i --;
        }
       A[i+1]=key;
   }
}
```

Costo	Veces
c0	1
c1	n
c2	n-1
0	
c4	n-1
c5	Suma desde j=1 hasta n-1 de tj
c6	Suma desde j=1 hasta n-1 de (tj - 1)
c7	Suma desde j=1 hasta n-1 de (tj -1)
0	n-1
c8	n-1
0	1
0	1

Costo de Insertion-Sort

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=1}^{n-1} t_j + c_6 \sum_{j=1}^{n-1} (t_j - 1) + c_7 \sum_{j=2}^{n} (t_j - 1) + c_8(n-1)$$

- ¿Cuál es el mejor caso? El arreglo está ya ordenado. ¿cuánto tiempo toma?
- ¿Cuál es el peor caso? El arreglo está ordenado pero en orden inverso.

Mejor caso :

$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5(n-1) + c_8(n-1)$

$= an + b$

Peor caso

$T(n) = ... = an^2 + bn + c$

Casos: peor, promedio, mejor

- Peor caso de tiempo de ejecución es el límite superior para el tiempo de ejecución considerando cualquier entrada. El algoritmo no supera este valor en ningún caso.
- El mejor caso tiene definición análoga a la de peor caso.
- Caso promedio: la idea es obtener el valor esperado para el tiempo. Normalmente se obtiene suponiendo todas las entradas posibles con igual probabilidad.

Crecimiento de Funciones

- Normalmente estamos interesados en el comportamiento de los algoritmos para grandes entradas (muchos datos de entrada).
- Notación asintótica: para una función dada g(n) denotamos por Θ(g(n)) al conjunto de funciones:

$$\Theta(g(n)) = \{f(n): \quad \exists c_1 > 0, c_2 > 0, n_o > 0: \quad 0 \le c_1 g(n) \le f(n) \le c_2 g(n) \quad \forall n \ge n_o\}$$

- Aún cuando Θ(g(n)) es un conjunto, se acostumbra a notar
- f(n) = Θ(g(n))
- Ejemplo: demostrar que $f(n)=an^2+bn+c$ es $\Theta(n^2)$ con a,b,c>0

Notación O, Ω, o, ω

$$O(g(n)) = \{f(n): \quad \exists c > 0, n_o > 0: \quad 0 \le f(n) \le cg(n) \quad \forall n \ge n_o\}$$

$$\Omega(g(n)) = \{f(n): \quad \exists c > 0, n_o > 0: \quad 0 \le cg(n) \le f(n) \quad \forall n \ge n_o\}$$

$$o(g(n)) = \{f(n): \quad \forall c > 0, \exists n_o > 0: \quad 0 \le f(n) < cg(n) \quad \forall n \ge n_o\}$$

$$Ojo: \quad f(n) = o(g(n)) \quad \Rightarrow \quad \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

$$\omega(g(n)) = \{f(n): \quad \forall c > 0, \exists n_o > 0: \quad 0 \le cg(n) < f(n) \quad \forall n \ge n_o\}$$

$$Ojo: \quad f(n) = \omega(g(n)) \quad \Rightarrow \quad \lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$$

Analogía que puede ayudar

- $f(n) = O(g(n)) \sim a \leq b$
- $f(n) = o(g(n)) \sim a < b$
- $f(n) = \Theta(g(n)) \sim a = b$
- $f(n) = \omega(g(n)) \sim a > b$
- $f(n) = \Omega(g(n)) \sim a \geq b$

Ejercicio

- Probar que:
- $f(n)=\Theta(g(n))$ y $g(n)=\Theta(h(n)) \Rightarrow f(n)=\Theta(h(n))$
- Si $f(n)>0$ y $g(n)>0$, probar que $\max(f(n),g(n)) = \Theta(f(n)+g(n))$
- Ordenar las siguientes funciones por orden de crecimiento

$$\lg(\lg n) \qquad 2^{\lg n} \qquad (\sqrt{2})^{\lg n} \qquad n^2 \qquad n! \qquad (\lg n)! \qquad 1$$

$$(2/3)^n \qquad n^3 \qquad \lg^2 n \qquad \lg(n!) \qquad 2^{2^n} \qquad \ln(\ln n)$$