

Complejidad

Estructuras de Datos II

Erwin Meza Vega

Introducción

- No es suficiente que los algoritmos funcionen “correctamente”, deben ser eficientes
- Eficiencia: se mide en términos de espacio (memoria utilizada) y tiempo (de ejecución)

Tiempo de ejecución de un algoritmo

- Depende de diversos parámetros
 - Número de datos a procesar
 - Cantidad de código
 - Complejidad del algoritmo
- Se puede obtener:
 - Medida teórica (a priori): Funciones que acotan superior o inferiormente el tiempo de ejecución
 - Medida real (a posteriori): Medir el tiempo de ejecución del algoritmo
 - Ambas medidas se realizan para un número determinado de datos

Medición de tiempo

- Tamaño de entrada: Número de componentes (datos) sobre los cuales se va a ejecutar el algoritmo
 - Dimensión del vector a procesar, orden de las matrices a las cuales se les realizarán operaciones

$T(n)$: Tiempo de ejecución del algoritmo para una entrada de tamaño n

- Número de instrucciones ejecutadas por un computador ideal.

Principio de invarianza

Dado un algoritmo y dos implementaciones
suyas I_1 e I_2 , que tardan $T_1(n)$ y $T_2(n)$,
existe una constante real $c > 0$ y un
número natural n_0 tales que
para todo $n \geq n_0$ se verifica que
 $T_1(n) \leq cT_2(n)$.

- El tiempo de ejecución de los dos algoritmos difiere sólo en una constante multiplicativa

Factores a tener en cuenta

- El comportamiento de un algoritmo puede cambiar para diferentes entradas (por ejemplo los métodos de ordenación)
 - Se estudian tres casos: mejor, peor y medio.
- Mejor caso: Traza del algoritmo en la que se ejecutan menos instrucciones
- Peor caso: Traza del algoritmo en la que se ejecutan más instrucciones

Operaciones elementales (OE)

- Aquellas operaciones que realiza el computador que se encuentran acotadas por alguna constante
 - Operaciones aritméticas básicas
 - Asignación de variables
 - Llamadas a procedimientos y retornos
 - Comparaciones lógicas
 - Acceso a estructuras de datos como vectores y matrices
- Todas estas operaciones

Ejemplo

```
CONST n =...; (* num. maximo de elementos de un vector *);  
TYPE vector = ARRAY [1..n] OF INTEGER;
```

```
PROCEDURE Buscar(VAR a:vector;c:INTEGER):CARDINAL;  
  VAR j:CARDINAL;  
  BEGIN  
    j:=1;  
    WHILE (a[j]<c) AND (j<n) DO  
      j:=j+1  
    END;  
    IF a[j]=c THEN  
      RETURN j  
    ELSE RETURN 0  
  END  
END Buscar;
```

```
(* 1 *) 1 OE  
(* 2 *) 4 OE  
(* 3 *) 2 OE  
(* 4 *)  
(* 5 *) 2 OE  
(* 6 *) 1 OE  
(* 7 *) 1 OE  
(* 8 *)
```


Análisis

- Caso mejor: (1) y la mitad de (2), (5) y (7)
 $T(n) = 1 + 2 + 3 = 6$
- Caso peor: (1), bucle (n-1), (5) y (7)
 - Cada iteración del bucle: (2) y (3)

$$T(n) = 1 + \left(\left(\sum_{i=1}^{n-1} (4 + 2) \right) + 4 \right) + 2 + 1 = 6n + 2.$$

Caso medio

- El bucle se ejecutará un número de veces entre 0 y $n-1$. Si cada iteración tiene igual probabilidad de suceder,

$$\sum_{i=0}^{n-1} i \frac{1}{n} = \frac{n-1}{2}$$

- Por lo tanto

$$T(n) = 1 + \left(\left(\sum_{i=1}^{(n-1)/2} (4+2) \right) + 2 \right) + 2 + 1 = 3n + 3$$

Reglas para el cálculo de OE

- El tiempo de ejecución de una secuencia consecutiva de instrucciones se calcula sumando los tiempos de ejecución de cada una de las instrucciones.
- El tiempo de ejecución de la sentencia “CASE C OF $v_1:S_1|v_2:S_2|\dots|v_n:S_n$ END;” es
$$T = T(C) + \max\{T(S_1), T(S_2), \dots, T(S_n)\}$$
- El tiempo de ejecución de la sentencia “IF C THEN S_1 ELSE S_2 END;” es
$$T = T(C) + \max\{T(S_1), T(S_2)\}.$$

Reglas para el cálculo de OE

- El tiempo de ejecución de un bucle de sentencias “WHILE C DO S END;” es

$$T = T(C) + (n^{\circ} \text{ iteraciones}) * (T(S) + T(C)).$$

Obsérvese que tanto $T(C)$ como $T(S)$ pueden variar en cada iteración, y por tanto habrá que tenerlo en cuenta para su cálculo.

- Para calcular el tiempo de ejecución del resto de sentencias iterativas (*FOR*, *REPEAT*, *LOOP*) basta expresarlas como un bucle *WHILE*.

Por ejemplo el tiempo de ejecución del bucle:

```
FOR i:=1 TO n DO
```

```
S
```

```
END;
```

puede ser calculado a partir del bucle equivalente:

```
i:=1;
```

```
WHILE i<=n DO
```

```
S; INC(i)
```

```
END;
```

Reglas para el cálculo de OE

- El tiempo de ejecución de una llamada a un procedimiento o función

$F(P_1, P_2, \dots, P_n)$ es 1 (por la llamada), más el tiempo de evaluación de los parámetros P_1, P_2, \dots, P_n , más el tiempo que tarde en ejecutarse F

$$T = 1 + T(P1) + T(P2) + \dots + T(Pn) + T(F).$$

- El tiempo de ejecución de las llamadas a procedimientos recursivos va a dar lugar a ecuaciones en recurrencia.

Cota superior. Notación O

- Sea $f: \mathbf{N} \rightarrow [0, \infty)$. Se define el conjunto de funciones de orden O (Omicron) de f como:

$$O(f) = \{g: \mathbf{N} \rightarrow [0, \infty) \mid \exists c \in \mathbf{R}, c > 0, \exists n_0 \in \mathbf{N} \\ \bullet g(n) \leq cf(n) \ \forall n \geq n_0\}.$$

Diremos que una función $t: \mathbf{N} \rightarrow [0, \infty)$ es de orden O de f si $t \in O(f)$.

Indica que t está acotada superiormente por algún múltiplo de f .

Propiedades de O

- 1. Para cualquier función f se tiene que $f \in O(f)$.
- 2. $f \in O(g) \Rightarrow O(f) \subset O(g)$.
- 3. $O(f) = O(g) \Leftrightarrow f \in O(g)$ y $g \in O(f)$.
- 4. Si $f \in O(g)$ y $g \in O(h) \Rightarrow f \in O(h)$.
- 5. Si $f \in O(g)$ y $f \in O(h) \Rightarrow f \in O(\min(g, h))$.
- 6. Regla de la suma: Si $f_1 \in O(g)$ y $f_2 \in O(h) \Rightarrow f_1 + f_2 \in O(\max(g, h))$.
- 7. Regla del producto:
Si $f_1 \in O(g)$ y $f_2 \in O(h) \Rightarrow f_1 \cdot f_2 \in O(g \cdot h)$.

Propiedades de O

8. Si existe $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = k$, dependiendo de los valores que tome k obtenemos:
- a) Si $k \neq 0$ y $k < \infty$ entonces $O(f) = O(g)$.
 - b) Si $k = 0$ entonces $f \in O(g)$, es decir, $O(f) \subset O(g)$, pero sin embargo se verifica que $g \notin O(f)$.

Cota Inferior. Notación Ω

- Sea $f: \mathbf{N} \rightarrow [0, \infty)$. Se define el conjunto de funciones de orden Ω (Omega) de f como:
$$\Omega(f) = \{g: \mathbf{N} \rightarrow [0, \infty) \mid \exists c \in \mathbf{R}, c > 0, \exists n_0 \in \mathbf{N} \bullet g(n) \geq cf(n) \ \forall n \geq n_0\}.$$
 - Diremos que una función $t: \mathbf{N} \rightarrow [0, \infty)$ es de orden Ω de f si $t \in \Omega(f)$.
- $t \in \Omega(f)$ indica que t está acotada inferiormente por algún múltiplo de f

Propiedades de Ω

1. Para cualquier función f se tiene que $f \in \Omega(f)$.
2. $f \in \Omega(g) \Rightarrow \Omega(f) \subset \Omega(g)$.
3. $\Omega(f) = \Omega(g) \Leftrightarrow f \in \Omega(g) \text{ y } g \in \Omega(f)$.
4. Si $f \in \Omega(g)$ y $g \in \Omega(h) \Rightarrow f \in \Omega(h)$.
5. Si $f \in \Omega(g)$ y $f \in \Omega(h) \Rightarrow f \in \Omega(\max(g, h))$.
6. Regla de la suma: Si $f_1 \in \Omega(g)$ y $f_2 \in \Omega(h) \Rightarrow f_1 + f_2 \in \Omega(g + h)$.

Propiedades de Ω

7. Regla del producto: Si $f_1 \in \Omega(g)$ y $f_2 \in \Omega(h) \Rightarrow f_1 \cdot f_2 \in \Omega(g \cdot h)$.

8. Si existe $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = k$, dependiendo de los valores que tome k obtenemos:

a) Si $k \neq 0$ y $k < \infty$ entonces $\Omega(f) = \Omega(g)$.

b) Si $k = 0$ entonces $g \in \Omega(f)$, es decir, $\Omega(g) \subset \Omega(f)$, pero sin embargo $f \notin \Omega(g)$.

Orden Exacto. Notación Θ

- Sea $f: \mathbf{N} \rightarrow [0, \infty)$. Se define el conjunto de funciones de orden Θ (Theta) de f como:

$$\Theta(f) = O(f) \cap \Omega(f)$$

o como:

$$\Theta(f) = \{g: \mathbf{N} \rightarrow [0, \infty) \mid \exists c, d \in \mathbf{R}, c, d > 0, \exists n_0 \in \mathbf{N} \cdot cf(n) \leq g(n) \leq df(n) \forall n \geq n_0\}.$$

Diremos que una función $t: \mathbf{N} \rightarrow [0, \infty)$ es de orden Θ de f si $t \in \Theta(f)$.

Propiedades de Θ

1. Para cualquier función f se tiene que $f \in \Theta(f)$.
2. $f \in \Theta(g) \Rightarrow \Theta(f) = \Theta(g)$.
3. $\Theta(f) = \Theta(g) \Leftrightarrow f \in \Theta(g) \text{ y } g \in \Theta(f)$.
4. Si $f \in \Theta(g)$ y $g \in \Theta(h) \Rightarrow f \in \Theta(h)$.
5. Regla de la suma: Si $f_1 \in \Theta(g)$ y $f_2 \in \Theta(h) \Rightarrow f_1 + f_2 \in \Theta(\max(g, h))$.
6. Regla del producto: Si $f_1 \in \Theta(g)$ y $f_2 \in \Theta(h) \Rightarrow f_1 \cdot f_2 \in \Theta(g \cdot h)$.

Propiedades de Θ

7. Si existe $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = k$, dependiendo de los valores que tome k obtenemos:
- a) Si $k \neq 0$ y $k < \infty$ entonces $\Theta(f) = \Theta(g)$.
 - b) Si $k = 0$ los órdenes exactos de f y g son distintos.

Ejemplos

De las siguientes afirmaciones, indicar cuales son ciertas y cuales no:

(i) $n^2 \in O(n^3)$

(ii) $n^3 \in O(n^2)$

(iii) $2^{n+1} \in O(2^n)$

(iv) $(n+1)! \in O(n!)$

(v) $f(n) \in O(n) \Rightarrow 2^{f(n)} \in O(2^n)$

(vi) $3^n \in O(2^n)$

(vii) $\log n \in O(n^{1/2})$

(viii) $n^{1/2} \in O(\log n)$

(ix) $n^2 \in \Omega(n^3)$

(x) $n^3 \in \Omega(n^2)$

(xi) $2^{n+1} \in \Omega(2^n)$

(xii) $(n+1)! \in \Omega(n!)$

(xiii) $f(n) \in \Omega(n) \Rightarrow 2^{f(n)} \in \Omega(2^n)$

(xiv) $3^n \in \Omega(2^n)$

(xv) $\log n \in \Omega(n^{1/2})$

(xvi) $n^{1/2} \in \Omega(\log n)$