Análisis de Algoritmos y Estructuras de Datos

Tema 2: Análisis de la complejidad de los algoritmos

Mª Teresa García Horcajadas
José Fidel Argudo Argudo
Antonio García Domínguez
Francisco Palomo Lozano

Universidad de Cádiz

Versión 1.0

Índice

Tiempo y espacio algorítmicos

Comparación de algoritmos

- Compararemos la eficiencia de varios algoritmos para un mismo problema: ¿cuál usa menos recursos computacionales?
- En máquinas secuenciales, tenemos dos: tiempo y espacio
- Los recursos usados por los algoritmos son abstractos, y los usados por los programas son reales

Relación entre entradas y uso de recursos

- En general, depende de cada entrada $d \in D$
- A veces, basta relacionarlo con su tamaño $n = \|d\| \in \mathbb{N}$
- El tamaño depende de la representación interna de la entrada: p. ej. número de elementos en una lista o cifras de un número

Enfoques en el análisis del tiempo

Teórico: estudio formal sobre el algoritmo

1. Cálculo de la función $t(n)$ de tiempo del algoritmo
2. Estudio de $t(n)$ y su orden de crecimiento

Empírico: mediciones sobre el programa

1. Programación del algoritmo
2. Medida de tiempos reales del programa
3. Estudio cualitativo de los resultados

Híbrido: mezcla de los anteriores

1. Obtención de un modelo teórico del tiempo del algoritmo
2. Programación del algoritmo
3. Medida de los tiempos reales del programa
4. Ajuste del modelo teórico mediante regresión

Tiempo de un algoritmo

Operaciones críticas y operaciones elementales

- Una operación es elemental si se ejecuta en una máquina real en un tiempo acotado por una constante ($t_{op} \in \Theta(1)$)
- Una operación es crítica si se ejecuta al menos tantas veces como cualquier otra del algoritmo

Medición de $t(n)$ sobre una operación elemental y crítica

- Al ser crítica, el orden de todas las operaciones realizadas coincidirá con el de las veces que se ejecutó esta operación
- Si además es elemental, el orden del número de ejecuciones de esta operación coincidirá con el orden del tiempo real de todo programa que implemente el algoritmo
- Esto nos simplifica mucho el análisis

Caso peor y caso mejor

Dependencia entre entradas y tiempos

- Asumimos que para todo $d \in D$ con $n = \|d\|$ podemos usar $t(n)$ como $t(d)$
- Sin embargo, puede que $t(d)$ varíe para un mismo n
- Tampoco podemos estudiar todas las entradas: ¿qué hacer?

Solución: dividir las entradas con un tamaño n en clases.

Caso peor: entradas con tamaño n más costosas

$$t_{\text{máx}}(n) = \text{máx}\{t(d) \mid d \in D \land \|d\| = n\}$$

Caso mejor: entradas con tamaño n menos costosas

$$t_{\text{mín}}(n) = \text{mín}\{t(d) \mid d \in D \land \|d\| = n\}$$

Caso peor y caso mejor

Dependencia entre entradas y tiempos

- Asumimos que para todo $d \in D$ con $n = \|d\|$ podemos usar $t(n)$ como $t(d)$
- Sin embargo, puede que $t(d)$ varíe para un mismo n
- Tampoco podemos estudiar todas las entradas: ¿qué hacer?

Solución: dividir las entradas con un tamaño n en clases.

Caso peor: entradas con tamaño n más costosas

$$t_{\text{máx}}(n) = \text{máx}\{t(d) \mid d \in D \wedge \|d\| = n\}$$

Caso mejor: entradas con tamaño n menos costosas

$$t_{\text{mín}}(n) = \text{mín}\{t(d) \mid d \in D \wedge \|d\| = n\}$$

Caso peor y caso mejor

Dependencia entre entradas y tiempos

- Asumimos que para todo $d \in D$ con $n = \|d\|$ podemos usar $t(n)$ como $t(d)$
- Sin embargo, puede que $t(d)$ varíe para un mismo n
- Tampoco podemos estudiar todas las entradas: ¿qué hacer?

Solución: dividir las entradas con un tamaño n en clases.

Caso peor: entradas con tamaño n más costosas

$$t_{\text{máx}}(n) = \text{máx}\{t(d) \mid d \in D \wedge \|d\| = n\}$$

Caso mejor: entradas con tamaño n menos costosas

$$t_{\text{mín}}(n) = \text{mín}\{t(d) \mid d \in D \wedge \|d\| = n\}$$

Caso peor y caso mejor

Dependencia entre entradas y tiempos

- Asumimos que para todo $d \in D$ con $n = \|d\|$ podemos usar $t(n)$ como $t(d)$
- Sin embargo, puede que $t(d)$ varíe para un mismo n
- Tampoco podemos estudiar todas las entradas: ¿qué hacer?

Solución: dividir las entradas con un tamaño n en clases.

Caso peor: entradas con tamaño n más costosas

$$t_{\text{máx}}(n) = \text{máx}\{t(d) \mid d \in D \wedge \|d\| = n\}$$

Caso mejor: entradas con tamaño n menos costosas

$$t_{\text{mín}}(n) = \text{mín}\{t(d) \mid d \in D \wedge \|d\| = n\}$$

Caso promedio

Cuando hay una diferencia significativa entre el peor y el mejor caso, conviene estudiar el tiempo en el caso promedio:

$$\overline{t}(n) = \sum_{d \in D \wedge \|d\| = n} \mathcal{P}(d)t(d)$$

$\mathcal{P}(d)$ es la probabilidad de que d aparezca como entrada de tamaño n. Se cumple que:

$$t_{\text{mín}}(n) \leqslant \overline{t}(n) \leqslant t_{\text{máx}}(n)$$

Cuando no haya confusión, hablaremos simplemente de $t(n)$: el contexto determinará a qué caso nos estamos refiriendo.

Ejemplo

Si tras el análisis de un algoritmo nos informan únicamente de que $t(n) \in O(f(n))$, nos están proporcionando una cota superior para $t(n)$ y por lo tanto para $t_{\text{máx}}(n)$.

Análisis de las estructuras de control

Procedimiento general para calcular $t(n)$

Contar cuántas veces se ejecuta una operación crítica.

Importancia de las estructuras de control

- Hay que considerar cada una de sus componentes
- Si se modifica n, el análisis de las posteriores instrucciones tendrá que usar el nuevo valor
- Si se conserva n, podremos usar unas reglas generales

$$\begin{array}{ll} C_1 & \leftarrow t_{C_1}(n) \\ C_2 & \leftarrow t_{C_2}(n) \end{array}$$

Composición secuencial

$$t(n) = t_{C_1}(n) + t_{C_2}(n)$$

Estructuras condicionales

si B $\leftarrow t_B(n)$
  C_1 $\leftarrow t_{C_1}(n)$
si no
  C_2 $\leftarrow t_{C_2}(n)$

Caso peor

$$t(n) = t_B(n) + \text{máx}\{t_{C_1}(n), t_{C_2}(n)\}$$

Caso mejor

$$t(n) = t_B(n) + \text{mín}\{t_{C_1}(n), t_{C_2}(n)\}$$

Caso promedio

$$t(n) = t_B(n) + \mathcal{P}(B) t_{C_1}(n) + \mathcal{P}(\neg B) t_{C_2}(n)$$

Nota: las probabilidades pueden depender de n. En ese caso, tendríamos $\mathcal{P}(B, n)$ y $\mathcal{P}(\neg B, n)$.

Estructuras iterativas

Usaremos $v(n)$ como número total de vueltas, y la vuelta actual será i

mientras B $\leftarrow$ $t_B(n)$: no varía a cada vuelta
C $\leftarrow$ $t_C(n)$: no varía a cada vuelta

$$t(n) = (v(n)+1)t_B(n) + v(n)t_C(n)$$

mientras B $\leftarrow$ $t_B(n)$: no varía a cada vuelta
C $\leftarrow$ $t_C(i,n)$: varía a cada vuelta

$$t(n) = (v(n)+1)t_B(n) + \sum_{i=1}^{v(n)} t_C(i,n)$$

mientras B $\leftarrow$ $t_B(i,n)$: varía a cada vuelta
C $\leftarrow$ $t_C(i,n)$: varía a cada vuelta

$$t(n) = \sum_{i=1}^{v(n)+1} t_B(i,n) + \sum_{i=1}^{v(n)} t_C(i,n)$$

Mínimo de un vector $v \in V^n$ según $\langle V, \leqslant \rangle$

$mínimo(v, n) \rightarrow m$
$m \leftarrow v[1]$
desde $i \leftarrow 2$ hasta n
    si $v[i] < m$ $\leftarrow$ op. crítica
        $m \leftarrow v[i]$

Invariante

$$m = \min_{1 \leqslant \alpha < i} v[\alpha]$$

Análisis

- Operación seleccionada: comparaciones de elem. del vector
- $t(n)$ no depende del contenido de v: todos los casos coinciden
- En este caso, $t(n) = n - 1 \in \Theta(n)$

Inserción en orden

Dados $v[1] \leqslant \cdots \leqslant v[n-1]$, insertar $v[n]$ en el lugar para que v quede ordenado.

$$
\begin{array}{l}
\mathit{inserción}(v, n) \to v \\
\text{mientras } n > 1 \wedge v[n] < v[n-1] \\
\qquad v[n] \leftrightarrow v[n-1] \\
\qquad n \leftarrow n-1
\end{array}
$$

Análisis

- Operación crítica: comparaciones entre elementos del vector
- $\wedge$ opera en cortocircuito, evitando acceder a una posición no válida

Caso mejor: v[n] sigue en v[n]

- Sólo hacemos 1 comparación
- $t_{\text{mín}}(n) = 1 \in \Theta(1)$

Inserción en orden

Dados $v[1] \leqslant \cdots \leqslant v[n-1]$, insertar $v[n]$ en el lugar para que v quede ordenado.

$$
\begin{array}{l}
\textit{inserción}(v, n) \rightarrow v \\
\text{mientras } n > 1 \wedge v[n] < v[n-1] \\
\qquad v[n] \leftrightarrow v[n-1] \\
\qquad n \leftarrow n - 1
\end{array}
$$

Análisis

- Operación crítica: comparaciones entre elementos del vector
- $\wedge$ opera en cortocircuito, evitando acceder a una posición no válida

Caso peor: v[n] pasa a v[1] o v[2]

- Comparamos todos los pares consecutivos
- $t_{\text{máx}}(n) = n - 1 \in \Theta(n)$

Inserción en orden

Dados $v[1] \leqslant \cdots \leqslant v[n-1]$, insertar $v[n]$ en el lugar para que v quede ordenado.

$$
\begin{array}{l}
\textit{inserción}(v, n) \to v \\
\text{mientras } n > 1 \wedge v[n] < v[n-1] \\
\qquad v[n] \leftrightarrow v[n-1] \\
\qquad n \leftarrow n-1
\end{array}
$$

Análisis

- Operación crítica: comparaciones entre elementos del vector
- $\wedge$ opera en cortocircuito, evitando acceder a una posición no válida

Caso promedio

Hay que tener en cuenta las probabilidades de que v[n] acabe en cada una de las posiciones disponibles.

Inserción en orden: caso promedio

Si p es la posición final de $v[n]$ nos queda:

$$\overline{t}(n) = \sum_{\alpha=1}^{n} \mathcal{P}(p=\alpha)\, t(p=\alpha)$$

A falta de más información, supondremos que todas las posiciones son equiprobables: p está uniformemente distribuida en $[1, n]$.

$$\mathcal{P}(p=1) = \cdots = \mathcal{P}(p=n) = \frac{1}{n}$$

Pero, cuando $p = \alpha$, la operación crítica se ejecuta $n - \alpha + 1$ veces, salvo si $\alpha = 1$, ya que entonces se ejecuta una vez menos:

$$\begin{aligned}\overline{t}(n) &= \frac{1}{n}(n-1) + \frac{1}{n}\sum_{\alpha=2}^{n}(n-\alpha+1) \\ &= \frac{1}{n}\left(n-1+\frac{n(n-1)}{2}\right) = \frac{n+1}{2} - \frac{1}{n} \in \Theta(n)\end{aligned}$$

Inserción en orden: mejoras

inserción$(v, n) \rightarrow v$
$x \leftarrow v[n]$
mientras $n > 1 \wedge x < v[n-1]$
$\quad v[n] \leftarrow v[n-1]$
$\quad n \leftarrow n-1$
$v[n] \leftarrow x$

Mejoras

- Los intercambios se producen en cadena y pueden sustituirse por asignaciones simples: ahorramos 2/3 de las asignaciones.
- En la comparación $v[n] < v[n-1]$, el valor de $v[n]$ es siempre el mismo y puede sustituirse por una variable x que lo contenga: ahorramos un acceso por comparación.

La constante multiplicativa disminuye, pero el resultado principal no cambia: el orden no varía.

Referencias

- Baase, Sara & Van Gelder, Allen.
  Computer Algorithms. Introduction to Design and Analysis.
  Addison-Wesley. 2000. 3ª ed.
- Brassard, Gilles & Bratley, Paul.
  Fundamentos de Algoritmia.
  Prentice-Hall. 1997.
- Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L. & Stein, Clifford.
  Introduction to Algorithms.
  MIT Press. 2001. 2ª ed.
- Levitin, Anany V.
  Introduction to the design and analysis of algorithms.
  Addison-Wesley. 2003.