LEZIONE PRECEDENTE

PROBLEMI

RICERCA DI UN ELMENTO IN UN ARRAY

ALGORITMI

- RICERCA LINEARE
- RICERCA BINARIA

L'ORDINAMENTO DI UN ARRAY

- ORDINAMENTO PER SELEZIONE
- QUICK SORT

QUANTO È BUONO UN ALGORITMO?

- VELOCITÀ DI ESECUZIONE
- OCCUPAZIONE DI MEMORIA

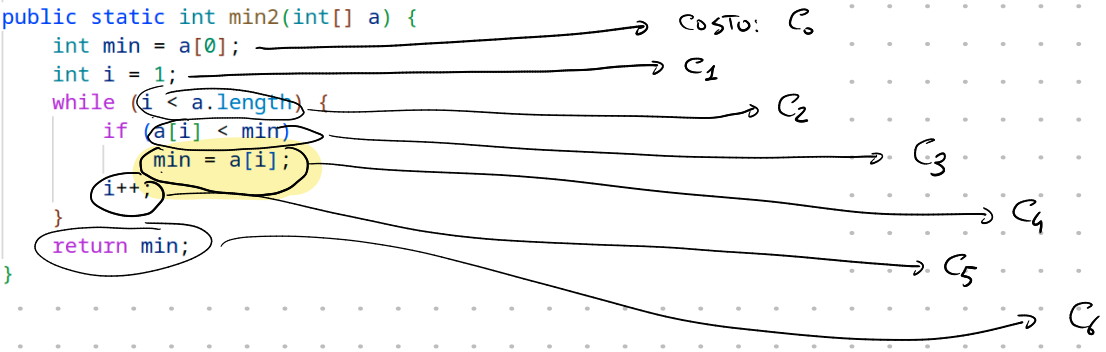
```
10      /**
11       * Restituisce il minimo del vettore a.
12       */
13      public static int min1(int[] a) {
14          int min = a[0];
15          for (int i = 1; i < a.length; i++) {
16              if (a[i] < min)
17                  min = a[i];
18          }
19          return min;
20      }
21
```

Possiamo supporre che a abbia almeno un elemento

ISTRUZIONI ELEMENTARI:

- ASSEGNAMENTI
- CONTROLLO DELLE CONDIZIONI
- INVOCAZIONE DI METODI
- RETURN

```
22      /**
23       * Restituisce il minimo del vettore a. Variante col ciclo while.
24       */
25      public static int min2(int[] a) {
26          int min = a[0];
27          int i = 1;
28          while (i < a.length) {
29              if (a[i] < min)
30                  min = a[i];
31              i++;
32          }
33          return min;
34      }
```



riga 26: $1 \cdot c_0$

riga 27: $1 \cdot c_1$

riga 28: lunghezza di a $\cdot c_2$

$\parallel$

$c_2 \cdot (n)$

la. uso per indicare la lunghezza di a

la prima volta che la riga 28 viene eseguita $i = 1$

poi 2, 3, 4 (e a questo punto la condizione è falsa e si esce dal ciclo)

riga 28: $c_3 \cdot (n-1)$

riga 30: $c_4 \cdot (n-1)$

riga 31: $c_5 \cdot (n-1)$

riga 33: $c_6 \cdot 1$

In generale dipende dai valori di a, ma per semplificare ci mettiamo nel CASO PESSIMO e supponiamo che la condizione dell'if abbia sempre successo

Costo totale dell'algoritmo:

$$c_0 + c_1 + c_2 \cdot n + c_3 (n-1) + c_4 (n-1) + c_5 (n-1) + c_6$$

$\parallel$

$$c_0 + c_1 + c_2 \cdot n + c_3 \cdot n - c_3 + c_4 \cdot n - c_4 + c_5 \cdot n - c_5 + c_6$$

$\parallel$

$$\boxed{(c_2 + c_3 + c_4 + c_5) \cdot n + (c_0 + c_1 - c_3 - c_4 - c_5 + c_6)}$$

Siccome le costanti c_i dipendono da caratteristiche del computer dove il programma è eseguito e non dell'algoritmo in se, decidiamo di porle tutte uguali ad 1. Vuol dire che quello che misuriamo come costo globale è il numero di operazioni elementari che facciamo.

Costo: $4n$ ← lunghezza dell'array di input

↑ dimensione dell'input

In generale, il costo di un algoritmo è una funzione a valori positivi della dimensione dell'input.

```
36      /**
37       * Restituisce la posizione di x nel vettore a, o -1 se x non si
38       * volte, viene restituita la prima posizione in cui compare.
39       */
40      public static int linearSearch(int[] a, int x) {
41          for (int i = 0; i < a.length; i++) {
42              if (a[i] == x)
43                  return i;
44          }
45          return -1;
46      }
```

0 .. a.length (n+1)

n = lunghezza di a

riga 41 i=0 : 1

riga 41: i < a.length : n+1

riga 41 : i++ : n

riga 42: n

riga 43: 0

riga 45: 1

TOT: $3n + 3$

↑ CASO PESSIMO

ATTENZIONE. In generale il ciclo for potrebbe essere eseguito meno di n volte, se x viene trovato nell'array a. Stiamo qui considerando il caso pessimo, in cui x non si trova nell'array a.

Proviamo a rifare l'analisi nel CASO OTTIMO, in cui x è il primo elemento di a.

riga 41 i=0 : 1

riga 41 i<a.length: 1

riga 41 i++ : 0

riga 42: 1

riga 43: 1

riga 45: 0

TOT: 4

Si può provare a fare una analisi di complessità nel caso medio.

Un esempio grossolano:

1) Se x (posizione) lo troviamo subito $\Rightarrow$ costo $= 4$

2) Se x è in posizione 1 $\Rightarrow$ costo $= 3 \cdot 1 + 4$

3) Se x è in posizione i $\Rightarrow$ costo $= 3 \cdot i + 4$

4) Se x non si trova nel vettore $\Rightarrow$ costo $= 3 \cdot n + 4$

Assumendo che le probabilità che x si trovi in posizione $0, 1, 2, \dots n-1$ o che non ci sia effettivamente sono uguali, il costo medio è la media di questi valori

$$TOT = \frac{4 + (3 \cdot 1 + 4) + (3 \cdot 2 + 4) + \dots + (3 \cdot (n-1) + 4) + (3n + 3)}{n+1}$$

- 4: x si trova in posizione 0
- $(3 \cdot 1 + 4)$: x si trova in posizione 1
- $(3 \cdot 2 + 4)$: x si trova in posizione 2
- $(3 \cdot (n-1) + 4)$: x si trova in ultima posizione
- $(3n + 3)$: x non si trova in v
- $n+1$: numero di casi

$$\simeq \frac{3}{2} n + 4$$

Poiché i calcoli sono complessi e il punto di partenza discutibile (probabilità dei casi uguale) il caso medio si studia raramente.

PARENTESI MATEMATICA

Siano date due funzioni $f, g : \mathbb{N} \to \mathbb{N}$ (oppure $\mathbb{N} \to \mathbb{R}$, $\mathbb{R} \to \mathbb{R}$, ...) Cosa succede se l'input di f e g crescono.

Esempio

$f(n) = n + 3$ $\quad \lim_{n \to +\infty} f(n) = +\infty$

$g(n) = n^2 + 1$ $\quad \lim_{n \to +\infty} g(n) = +\infty$

Quale cresce più rapidamente? Consideriamo

$$\frac{f(n)}{g(n)} = \frac{n+3}{n^2+1}$$ e in particolare il suo limite per $n \to +\infty$

$$\lim_{n\to+\infty} \frac{f(n)}{g(n)} = \lim_{n\to+\infty} \frac{n+3}{n^2+1} = 0$$

Definizione (O-grande)

Date due funzioni $f, g: \mathbb{N} \to \mathbb{N}$, se $\lim_{n\to+\infty} \frac{f(n)}{g(n)} = K \in \mathbb{R}$, diciamo che $f(n)$ è un O-grande di $g(n)$, e lo scriviamo come

$f(n) \in O(g(n))$ oppure $f(n) = O(g(n))$.

Esempi

$n^3 \in O(n^4)$ $\quad$ $n^3 \in O(3n^3)$

$n^3 \in O\left(\frac{1}{10} n^3\right)$ $\quad$ $n^3 \notin O(n^2)$ — $\lim_{n\to+\infty} \frac{n^3}{n^2} = +\infty$

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

Definizione (Ω-grande)

Date due funzioni $f, g: \mathbb{N} \to \mathbb{N}$, se $\lim_{n\to+\infty} \frac{f(n)}{g(n)} = \begin{cases} +\infty \text{ oppure} \\ K \in \mathbb{R}, K \neq 0 \end{cases}$, diciamo che $f(n)$ è un Ω-grande di $g(n)$, e lo scriviamo come

$f(n) \in \Omega(g(n))$ oppure $f(n) = \Omega(g(n))$.

Esempi

$n^4 \in \Omega(n^3)$ $\quad$ $n^3 \in \Omega(n^3)$ $\quad$ $n^3 \in \Omega(5n^3)$

$n^2 \notin \Omega(n^3)$

Notare che $n^3 \in O(2n^3)$, $n^3 \in \Omega(2n^3)$. Possiamo anche dire che $n^3 \in \Theta(2n^3)$.

↑ Theta

Riassunto

$f \in O(g)$: f più piccola o uguale a g

$f \in \Omega(g)$: f più grande o uguale a g

$f \in \Theta(g)$: f uguale a g

} come ordine di infinito

Tornando al risultato di prima

Costo della ricerca lineare → $3n+3$ nel caso pessimo

→ 4 nel caso ottimo

A noi interessa solo l'ordine di grandezza. Diremo quindi che:

Costo (ASINTOTICO) della ricerca lineare → $\Theta(n)$ nel caso pessimo

→ $\Theta(1)$ nel caso ottimo

ORDINAMENTO

	CASO PESSIMO	CASO MEDIO
SELECTION SORT	$\Theta(n^2)$	$\Theta(n^2)$
→ QUICK SORT	$\Theta(n^2)$	$\Theta(n \log n)$
MERGE SORT	$\Theta(n \log n)$	$\Theta(n \log n)$