16/01/2020

Problemi di ordinamento

Siano $R_1, R_2, \dots, R_N$ dei "record" (tipicamente un insieme di record viene chiamato file) e supponiamo che a ogni record sia associata una chiave k_i : cioè a R_i è associata la chiave k_i.
Vogliamo inoltre che l'insieme dei k_i : $\{k_1, k_2, \dots, k_N\}$ abbia un ordinamento totale cioè ~~vale~~ devono valere due proprietà

i. Per ogni coppia k_i k_j deve essere $k_i < k_j$ oppure $k_i > k_j$ oppure $k_i = k_j$;

ii. Se $k_i < k_j$ e $k_j < k_h$ allora $k_i < k_h$.

Il problema di ordinamento di questi record è quello di trovare una permutazione $p(1), \dots, p(N)$ di $1, \dots, N$ tale che

$$k_{p(1)} < k_{p(2)} < \dots < k_{p(N)}$$

Esempio

Voglio ordinarli e so che $k_3 < k_2 < k_1$

L'ordinamento è $R_3 \quad R_2 \quad R_1$ e la permutazione $p(1), p(2), p(3)$ che mi risolve questo problema è

$$p(1) = 3, \quad p(2) = 2, \quad p(3) = 1$$

Quali possono essere dei metodi per risolvere questo tipo di problemi

1. Metodo per scambio (exchange sort) → bubble sort e altri.

2. Metodo per inserimento (insert sort) Si considera un record alla volta e si inserisce nel posto giusto

3. Metodo di selezione (selection sort) Si cerca il più piccolo elemento, poi il secondo più piccolo e così via

4. Metodo per conteggio (enumeration sort) Si considera un record e si conta quanti sono quelli più piccoli di lui

5. Metodo "ad hoc"

Conteggio per comparazione

Algoritmo C Prende una serie di record $R_1, \dots, R_N$ con chiavi $K_1, \dots, K_N$ e mantiene un array COUNT in modo tale che al termine dell'algoritmo in COUNT[j] si trova il numero di record con chiave più piccola della chiave K_j.

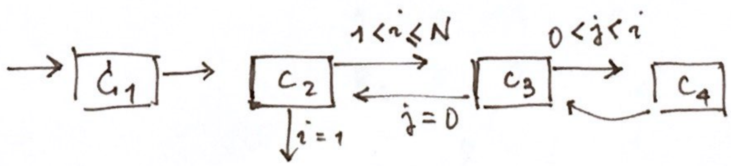
C1. Inizializzo a zero gli elementi di COUNT.

C2. Loop sull'indice i: ripetere lo step C3 con $i = N, N-1, \dots, 2$

C3. Loop sull'indice j: ripetere lo step C4 $j = i-1, i-2, \dots, 1$

C4. Confronta K_i con K_j: se $K_i < K_j$ incrementa di 1 COUNT[j] altrimenti aumenta di uno COUNT[i].

Diagramma di flusso



```
for j = 1 ... N
    COUNT[j] ← 0                                   N
end for

for i = N, ... 2
    for j = i-1, ..., 1
        if K_i < K_j
            COUNT[j] ++   ⎫
        else              ⎬   ~ N^2
            COUNT[i] ++   ⎭
    end for
end for
```

Proviamo ad eseguirlo con

K_1	K_2	K_3	K_4
2	3	1	4

COUNT $\boxed{\cancel{0}\,|\,\cancel{0}\,|\,0\,|\,0}$

$1 \quad \cancel{1}\ 2 \qquad \cancel{1}\ \cancel{2}\ 3$

$i = \cancel{4}\ \cancel{3}\ \cancel{2}\ 1$

$j = \cancel{3}\ \cancel{2}\ \cancel{2}\ \cancel{1}\ \cancel{1}\ \cancel{1}\ 0\ 0\ 0$

COUNT : $\boxed{1\,|\,2\,|\,0\,|\,3}$

1	2	3	4
K_3	K_1	K_2	K_4
1	2	3	4

<u>Complessità</u>

Analisi asintotica

inizializzazione	$O(N)$
confronto	$O(N^2)$
~~aggiornamento dei di~~ COUNT	~~$O(\ldots)$~~

$$O(N) + O(N^2) = O(N^2)$$

Definisco $A =$ # ~~vol~~ di confronti $(K_i \overset{?}{<} K_j)$

$B =$ # di volte che $K_i > K_j$ quando $i < j$

Quanto vale A ?

Quando $i = N$ ho $N-1$ possibili salti di j $(= N-2, N-3, \dots 1)$

" $i = N-1$ h $N-2$ " " " j $(= N-3, \dots$

" $\vdots$

$i = 2$ ho 1 " " " j $(= 1)$

Quindi

$$A = (N-1) + (N-2) + \cdots + 1 \overset{\text{Gauss}}{=} \frac{N(N-1)}{2}$$

Per B l'analisi è un pochino più complessa perché questo numero dipende dall'ordinamento dei record ~~da~~ sui quali utilizzo l'algoritmo.

Intanto chiediamoci

$\max B = ?$

$\min B = ?$

Iniziamo con $\min B$.

~~Sulle~~ Sui record già ordinati ho $B = 0$, poiché $B \geq 0$

$\Rightarrow \min B = 0$

Vediamo il max B

Il numero massimo di inversioni si ha quando i record sono ordinati in modo tale che

$$K_1 > K_2 > K_3 > \dots > K_N$$

In questo caso il # di inversioni è uguale al numero di modi in cui posso prendere due coppie dei numeri della permutazione e quindi

$$B = \binom{N}{2} = \frac{N(N-1)}{2}$$

Assumendo che # $K_1, \dots, K_N$ siano presi da una distribuzione (di ordinamenti) uniforme quanto vale

$$\text{mean } B = ?$$

Si dimostra che

$$\text{mean } B = \frac{N(N-1)}{4}$$