

Andrea Lazzarotto

Classe 3AI

**Relazione sugli algoritmi di ordinamento
studiati e realizzati in laboratorio**

Introduzione

Durante le ultime lezioni teoriche e pratiche di Informatica, si è lavorato sulla realizzazione di algoritmi di ordinamento. Gli algoritmi utilizzano delle metodologie operative differenti tra loro ma restituiscono il medesimo risultato.

Il processo di ordinamento consiste nell'operare degli spostamenti agli elementi di un vettore allo scopo di disporli secondo un ordine stabilito a partire dal più piccolo al più grande. Questo criterio è di immediata comprensione soprattutto per i numeri, i quali si possono disporre idealmente su una retta orientata per comprendere l'ordine che devono assumere.

A tale scopo sono stati scritti e successivamente implementati in linguaggio di programmazione tre tipi di algoritmi di ordinamento. Verranno ora analizzate i loro metodi di operare e la loro efficienza.

Ordinamento semplice

Questo algoritmo utilizza una strategia molto semplice per ordinare gli elementi. A partire dal primo, lo confronta con i successivi effettuando uno scambio ogni volta che ne trova uno di valore inferiore. Poi passa al secondo confrontando i seguenti, e così via. Ad esempio, partendo dal seguente array contenente numeri casuali la dinamica è la seguente:

9	7	2	1	1	1	1	1	1	1
7	9	9	9	7	6	2	2	2	2
2	2	7	7	9	9	9	7	6	6
6	6	6	6	6	7	7	9	9	7
1	1	1	2	2	2	6	6	7	9

Ordinamento per selezione

L'algoritmo utilizza un metodo simile al precedente, limitando però gli scambi al numero in assoluto più piccolo. Per fare questo il programma deve ad ogni giro memorizzare l'indice dell'elemento più piccolo per poi sapere cosa scambiare. In questo caso l'ordinamento si svolge così:

9	1	1	1
7	7	2	2
2	2	7	6
6	6	6	7
1	9	9	9

Risulta immediatamente visibile il minor numero di scambi effettuati dall'algoritmo.

Ordinamento per inserzione

Al contrario dei due algoritmi finora trattati, questo non effettua degli scambi ma delle inserzioni. Ovvero viene cercato il posto relativo che un elemento deve assumere rispetto ai suoi precedenti. In questo caso, non avendo senso parlare di scambi, vengono contati ai fini dell'analisi i singoli spostamenti effettuati. Lo svolgimento è questo:

9	7	2	2	1
7	9	7	6	2
2	2	9	7	6
6	6	6	9	7
1	1	1	1	9

Anche in questo caso si nota un minore numero di passaggi rispetto all'algoritmo semplice.

Descrizione delle prove effettuate

Per verificare l'efficienza e la funzionalità degli algoritmi, i medesimi sono stati sottoposti a numerose prove in cui venivano testati con array di dimensioni pari a 10, 100, 1000 e 5000 elementi; utilizzando numeri casuali da 0 a 5000. Sono poi stati affrontati dei casi particolari in cui i numeri sono stati inseriti manualmente.

Sono stati presi in considerazioni gli scambi e gli spostamenti rispettivamente per i primi due ed il terzo algoritmo. Non si è ritenuto necessario fare diverse prove per analizzare i confronti eseguiti tra gli elementi, dato che in ogni configurazione devono essere fatti tutti per assicurarsi che l'ordine venga effettuato nel modo corretto.

Ovviamente i risultati sono da ritenersi indicativi in quanto i numeri casuali variano di volta in volta e quindi non rappresentano un certo tipo di configurazione media sempre uguale.

Tabella dei risultati

Per prima cosa è bene osservare la tabella delle prove realizzate.

Elementi dell'array:	10	10	10	100	100	100	1000	1000	1000	5000	5000	5000
Prova 1	9	8	46	2431	96	2601	240134	990	249829	4577480	4990	6265471
Prova 2	25	8	33	2442	96	2639	227035	990	244906	4645673	4985	6324617
Prova 3	20	6	28	2536	98	2769	237618	989	252041	4587087	4994	6271003
Prova 4	22	8	28	2369	95	2649	227404	994	248380	4658864	4994	6200682
Prova 5	23	9	42	2534	94	2527	234456	993	252252	4656656	4991	6213038
Prova 6	34	8	27	2288	96	2696	240756	990	254111	4596476	4992	6196516
Prova 7	22	8	46	2749	97	2840	235493	995	261108	4563624	4989	6341817
Prova 8	31	5	48	2352	95	2782	236908	993	254569	4569736	4990	6251641
Prova 9	17	9	28	2521	93	2603	228785	994	250752	4581941	4986	6211080
Prova 10	14	7	42	2802	95	2411	245293	992	256728	4565858	4991	6151767
Media:	21,7	7,6	36,8	2502,4	95,5	2651,7	235388,2	992	252467,6	4600339,5	4990,2	6242763,2

Legenda: ■ Ordinamento semplice ■ Ordinamento per selezione ■ Ordinamento per inserzione

Come si può vedere, nel primo caso la crescita del numero degli scambi è esponenziale, mentre nel secondo caso gli scambi si mantengono sempre minori del numero di elementi. Nel terzo caso gli spostamenti sono leggermente superiori agli scambi del primo. Si sarebbe quindi portati a pensare che il terzo sia il peggiore in assoluto. Tuttavia è da considerare che ogni scambio per i primi due corrisponde a tre spostamenti, quindi il primo algoritmo esegue quasi tre volte gli spostamenti del terzo.

È altresì vero che il terzo algoritmo esegue meno confronti del secondo. Infatti, l'algoritmo di selezione ne deve fare il doppio.

Casi particolari

La parte più significativa dell'analisi è da ricercarsi nei casi particolari di vettori già ordinati e vettori ordinati inversamente.

Nel primo caso in qualsiasi prova e con qualsiasi algoritmo, non viene effettuato alcuno scambio. Quindi la differenza consiste nei confronti.

Nel secondo caso, il numero di scambi che viene svolto si differenzia a seconda dell'algoritmo. Con il metodo semplice, si può facilmente verificare che il numero di scambi è:

$$S = \frac{n \cdot (n-1)}{2}$$

Infatti, al primo giro vengono effettuati $n-1$ scambi, al secondo $n-2$ e così via fino ad 1 . Pertanto si applica la formula di Gauss per la sommatoria dei numeri da 1 a $n-1$. È il caso peggiore per questo algoritmo.

Per quanto riguarda il metodo di selezione, data la natura dell'algoritmo, vengono invertiti i valori a due a due e perciò il numero di scambi è dato dalla divisione intera:

$$S = n \text{ div } 2$$

Questo invece non è il caso peggiore, in quanto si nota nelle prove casuali che si sono verificati più scambi. Osservando i dati relativi ai numeri casuali, si può intuitivamente pensare che il numero massimo di scambi sia:

$$S = n-1$$

Infine, nell'ultimo metodo, il numero degli spostamenti si calcola come segue. Dato che al primo giro (partendo però dal secondo elemento) vengono fatti 3 spostamenti, al secondo 4, al terzo 5 e così via fino all' $(n-2)$ esimo giro in cui vengono fatti $n+1$ spostamenti; si può utilizzare sempre la sommatoria di Gauss. Si nota che si tratta della sommatoria dei numeri da 1 a $n+1$, sottratti l' 1 ed il 2 . Pertanto si ha:

$$S = \frac{(n+2) \cdot (n+1)}{2} - 3$$

Anche qui vale l'osservazione che gli scambi effettuati nei primi due algoritmi corrispondono ognuno a tre spostamenti, e quindi i primi due risultati devono essere triplicati per un equo

paragone.

I confronti effettuati

In modo analogamente semplice, si può calcolare il numero di confronti che ogni algoritmo fa in qualsiasi situazione.

Nell'ordinamento semplice, vengono sempre effettuati i confronti pari alla sommatoria da 1 a $n-1$:

$$C = \frac{n \cdot (n-1)}{2}$$

Nell'ordinamento di selezione oltre ai confronti tra l'elemento che deve essere scambiato, vanno considerati anche i confronti con l'elemento che al momento sembra essere quello con cui va scambiato. Pertanto si svolgono esattamente il doppio di confronti:

$$C = n \cdot (n-1)$$

In quello di inserzione i confronti sono i medesimi del primo, in caso l'array sia già ordinato:

$$C = \frac{n \cdot (n-1)}{2}$$

E nel caso che il vettore sia ordinato all'inverso:

$$C = n-1$$

Conclusione

Confrontando i risultati a cui si è giunti tramite le varie prove e l'analisi matematica per determinare il numero di operazioni (spostamenti e confronti) si può quindi stabilire in modo indicativo quale algoritmo è migliore nei diversi casi.

Nel caso di array già ordinati, evidentemente l'efficienza dipende dal numero di confronti. In tal caso il primo ed il terzo metodo sono a pari merito.

Se invece il vettore è ordinato all'inverso il secondo algoritmo effettua molti meno scambi. È anche vero però che il terzo algoritmo effettua assai meno confronti. Perciò il secondo è leggermente migliore.

Nelle prove casuali gli algoritmi si ritrovano approssimativamente nella stessa situazione del caso in cui gli array siano già ordinati. Tuttavia, nonostante il terzo faccia più scambi, in genere effettua meno confronti.

Per questi motivi, dato che gli array con elementi in ordine casuale sono i più frequenti, appare ragionevole ritenere più efficiente l'ordinamento per inserzione.