Azzolini Riccardo 2019-04-01

Insertion sort e bubble sort

1 Insertion sort

```
public class Insertion {
    public static void sort(Comparable[] a) {
        int N = a.length;
        for (int i = 1; i < N; i++) {
            int j = i;
            while (j > 0 && less(a[j], a[j - 1])) {
                exch(a, j, j - 1);
                j--;
            }
        }
    }
}
```

L'algoritmo **insertion sort** (**ordinamento per inserzione**) suddivide il vettore in una porzione già ordinata, dall'inizio fino alla posizione `i - 1`, e una ancora da ordinare. A ogni iterazione del ciclo `for` esterno, l'elemento in posizione `i` viene *inserito* (da cui il nome dell'algoritmo) nella parte ordinata, mediante degli scambi che lo fanno scorrere a sinistra fino alla posizione corretta.

Quest'algoritmo è:

- *stabile*, perché due elementi uguali non vengono mai scambiati;
- *adattivo*, perché il ciclo `while` interno ha un numero di iterazioni che dipende dai dati in input (a differenza del *selection sort*, basato su due cicli `for` annidati con un numero fisso di iterazioni).

1.1 Complessità

- Nel caso migliore, un vettore ordinato, l'algoritmo esegue

$$\sum_{i=1}^{n-1} 1 = n - 1 = \Theta(n)$$

  confronti e 0 scambi.

- Nel caso peggiore, se il vettore è ordinato al contrario, il numero di confronti è uguale a quello di scambi:

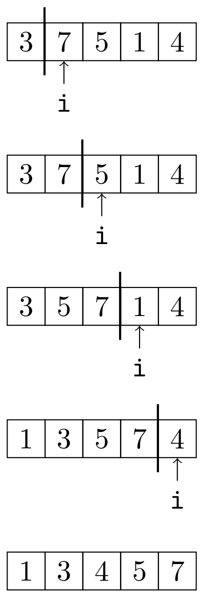
$$\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2} \sim \frac{n^2}{2} = \Theta(n^2)$$

- Nel caso medio, si eseguono la metà dei confronti e degli scambi, quindi il numero di operazioni è

$$\sim \frac{n^2}{4} = \Theta(n^2)$$

Di conseguenza, il numero complessivo di operazioni è $O(n^2)$ in ogni caso (ma non sempre $\Theta(n^2)$, a differenza del *selection sort*).

1.2 Esempio



2 Bubble sort

```java
public class Bubble {
    public static void sort(Comparable[] a) {
        int N = a.length;
        for (int i = 0; i < N; i++) {
```

```
            for (int j = N - 1; j > i; j--) {
                if (less(a[j], a[j - 1])) exch(a, j, j - 1);
            }
        }
    }
}
```

L'algoritmo **bubble sort** (**ordinamento a bolle**) prende il nome da un'analogia con il comportamento delle bolle in un liquido: quelle più piccole, corrispondenti agli elementi minori, "risalgono" più rapidamente verso l'inizio del vettore.

Infatti, il ciclo `for` esterno determina dove si fermano le "bolle" (all'indice `i`), mentre quello interno parte dalla fine del vettore e esegue degli scambi in modo da far risalire man mano la bolla più piccola, ordinando così un elemento per ogni iterazione del ciclo esterno.

È un algoritmo:

- *stabile*, perché non vengono mai scambiati elementi uguali;
- *non adattivo*, perché il numero di iterazioni dei cicli, e quindi di confronti, è fisso; in particolare, l'esecuzione continua anche quando il vettore diventa ordinato.

2.1 Complessità

Il numero di confronti è

$$\sum_{i=0}^{n-1} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-1} (n - i - 1) = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2} = \Theta(n^2)$$

in ogni caso.

2.2 Esempio

0	3	1	1	1	1	1	1
1	4	3	2	2	2	2	2
2	6	4	3	3	3	3	3
3	1	6	4	4	4	4	4
4	2	2	6	5	5	5	5
5	5	5	5	6	6	6	6

confronti inutili

2.3 Versione adattiva

```
public class Bubble {
    public static void sort(Comparable[] a) {
        int N = a.length;
        for (int i = 0; i < N; i++) {
            boolean scambio = false;
            for (int j = N - 1; j > i; j--) {
                if (less(a[j], a[j - 1])) {
                    exch(a, j, j - 1);
                    scambio = true;
                }
            }
            if (!scambio) break;
        }
    }
}
```

L'unica differenza rispetto alla versione non adattiva è che, se il ciclo interno non esegue scambi, cioè se il vettore è ordinato, l'algoritmo termina immediatamente.

Perciò, il numero di confronti si riduce, nel caso migliore, a

$$\sum_{j=1}^{n-1} 1 = n - 1 = \Theta(n)$$

3 Casi particolari

In certi casi particolari, in base al numero di inversioni, gli algoritmi elementari adattivi hanno un comportamento lineare.

3.1 Insertion sort

Teorema: Insertion sort effettua $\Theta(n)$ confronti e scambi per ordinare una sequenza con $\Theta(n)$ inversioni.

Dimostrazione: Ogni volta che un elemento viene traslato a destra di una posizione (nel ciclo interno, mediante un confronto e uno scambio), il numero di inversioni associate a esso diminuisce di 1.

Di conseguenza, il numero di inversioni $N_{inv}(X)$ è esattamente uguale al numero di scambi, che a sua volta è asintoticamente uguale al numero di confronti.[1] $\square$

3.2 Insertion sort e bubble sort

Teorema: Insertion sort e bubble sort (adattivo) effettuano $\Theta(n)$ confronti e scambi per ordinare una sequenza i cui elementi hanno un numero di inversioni associate limitato da una costante c.

Dimostrazione:

- Per insertion sort, è sufficiente osservare che il numero complessivo di inversioni è minore o uguale a $c\,n = \Theta(n)$ e applicare il teorema precedente.
- Per bubble sort, si sceglie di calcolare il numero di inversioni associate contando gli elementi minori a destra. Allora, la "bolla" che risale (essendo l'elemento più piccolo trovato a partire da destra) ha sempre 0 inversioni, mentre ogni elemento che "sprofonda" sotto la bolla rimane con 1 inversione in meno.

  Perciò, a ogni iterazione del ciclo esterno il numero di inversioni associate a ciascun elemento si riduce di 1 (se non è già 0), e allora serviranno al massimo c iterazioni.

  Siccome la prima iterazione ha sempre costo $\Theta(n)$, e le successive $O(n)$, il numero di operazioni è complessivamente $\Theta(n) + (c-1)O(n) = \Theta(n)$. $\square$

[1] Per ognuna delle $n-1$ iterazione del ciclo esterno, può essere eseguito al massimo un confronto con esito `false`, cioè non seguito da uno scambio, dato che esso causa la terminazione del ciclo `while` interno. In totale, quindi, i confronti senza scambi sono $O(n)$, e allora il numero di confronti è $N_{inv}(X) + O(n) = \Theta(n) + O(n) = \Theta(n)$.