Insertion sort

Emanuele Vivoli
5979383
emanuele.vivoli@stud.unifi.it

Marzo 2017

Indice

1 Introduzione **2**
1.1 Ordinare un mazzo di carte 2
1.2 Ordinare un array numerico 2

2 Algoritmo **3**
2.1 Pseudocodice 3
2.2 Analisi 3
2.3 Invariante di ciclo 3
2.4 Esempio grafico 4

1 Introduzione

L' *Insertion Sort* é un algoritmo relativamete semplice per ordinare un array e molto efficente per ordinare un piccolo numero di elementi. É un algoritmo chiamato *in place*, cioé ordina l'array senza doverne creare un altro 'di appoggio', quindi risparmia memoria. Non é molto diverso dal modo in cui un essere umano, spesso, ordina un mazzo di carte nella propria mano.

1.1 Ordinare un mazzo di carte

- Si inizia con la mano vuota e le carte capovolte sul tavolo
- Poi si prende una carta alla volta dal tavolo e si inserisce nella giusta posizione
- Per trovare la giusta posizione per una carta, la confrontiamo con le altre carte nella mano, da destra verso sinistra. Ogni carta piú grande verrá spostata verso destra in modo da fare posto alla carta da inserire.

N.B. : Ogni volta che sollevo una carta dal mazzo, le carte nella mano sono ordinate sempre in ordine crescente.

1.2 Ordinare un array numerico

- La sequenza é logicamente suddivisa in due parti: una ordinata ed una non ordinata (all'inizio la sottosequenza ordinata sará formata solamente dal primo valore)
- Viene preso un valore alla volta dalla sottosequenza non ordinata e si inserisce nella sottosequenza ordinata
- Per trovare la giusta posizione del valore appena prelevato dalla sottosequenza non ordinata, confrontiamo il valore con gli altri presenti in quella ordinata, da destra verso sinistra. Ogni valore piú grande verrá spostato verso destra in modo da fare posto al valore da inserire.

2 Algoritmo

Input L'algoritmo prede in input una sequanza numerica od un array.

Output L'algoritmo restituisce la sequenza/array avuto in input, ordinato in ordine crescente.

2.1 Pseudocodice

Algorithm 1 Insertion Sort

```
for j ←2 to [A].length do
    key ← A[j]
    i ← j − 1
    while i > 0 and A[i] > key do
        A[i + 1] ← Aij]
        i ← i − 1
    end while
    A[i + 1] ← key
end for
```

2.2 Analisi

Caso ottimo la sequenza di partenza sia giá ordinata.

In questo caso l'algoritmo ha tempo di esecuzione lineare, ossia $\Theta(n)$.

Infatti, in questo caso, in ogni iterazione il primo elemento della sottosequenza non ordinata viene confrontato solo con l'ultimo della sottosequenza ordinata.

Caso pessimo la sequenza di partenza sia ordinata al contrario.

In questo caso, ogni iterazione dovrá scorrere e spostare ogni elemento della sottosequenza ordinata prima di poter inserire il primo elemento della sottosequenza non ordinata.

Pertanto, in questo caso l'algoritmo di *Insertion Sort* ha complessitá temporale quadratica, ossia $\Theta(n^2)$.

Caso medio anch'esso ha complessitá quadratica, il che lo rende impraticabile per ordinare sequenze grandi.

2.3 Invariante di ciclo

Prima di ogni iterazione il sotto-array $A[1 \ldots i]$ é ordinato.

Ad ogni iterazione si prende l'elemento $A[j]$ e lo si pone al suo posto (in ordine crescente) nel sotto-array ordinato.

Si prosegue cosí con il sotto-array non ordinato $A[j + 1 \ldots n - 1]$.

2.4 Esempio grafico

Legenda : **VERDE:** Array ordinato, **ROSSO:** Array non ordinato