

Informatica 1

Corso di Laurea Triennale in Matematica

Gianluca Rossi

`gianluca.rossi@uniroma2.it`

Dipartimento di Matematica
Università di Roma "Tor Vergata"

a-1: Algoritmi di Ordinamento



Ordinare gli elementi di un vettore

Ordinamento

Data una sequenza di elementi sui quali è definita una relazione d'ordine, si vuole modificare l'ordine degli elementi della sequenza in modo che questi risultino ordinati (p.e. in modo non decrescente).



Insertion Sort

- Sia x_0, x_1, \dots, x_{n-1} gli elementi da ordinare;
- Si scandisce la sequenza dal primo all'ultimo elemento;
- Quando si considera x_i assumiamo che tutti gli elementi da x_0 a x_{i-1} sono ordinati;
- | | | | |
|----------------------------|-------------------------|-------|---------|
| $\dots x_j \leq x_i \dots$ | $\dots x_j > x_i \dots$ | x_i | \dots |
|----------------------------|-------------------------|-------|---------|
- Si posiziona x_i in posizione corretta relativamente agli elementi già analizzati.



Il codice

```
#include<stdio.h>
```

```
void insertionsort();
```

```
int v[] = {1,3,2,6,5,3,1,7,8,9,0};
```

```
int n;
```

```
main(){
```

```
    int i;
```

```
    n = sizeof(v)/sizeof(int);
```

```
    insertionsort();
```

```
    for(i=0; i<n; i++)
```

```
        printf("%d,  ", v[i]);
```

```
    printf("\n");
```

```
}
```

```
void insertionsort(){
```

```
    int i,j,t;
```

```
    for(i=1; i<n; i++){
```

```
        /* si pone v[i] nella posizione  
           corretta tra v[0] e v[i] */
```

```
        j = i-1;
```

```
        while(j>=0 && v[j+1] < v[j]){
```

```
            /*scambio tra v[j+1] e v[j]*/
```

```
            t = v[j];
```

```
            v[j] = v[j+1];
```

```
            v[j+1] = t;
```

```
            j--;
```

```
        }
```

```
    }
```

```
}
```



Quanto costa tutto ciò?

Quante operazioni occorrono per ordinare n elementi?

Se il vettore è ordinato.

- Per ogni i : 1 sottrazione, 1 assegnazione, 1 somma, 2 test.
- Assumendo che ogni operazione elementare costa 1, allora il costo complessivo è $c \cdot n$ dove c è una opportuna costante.

Se il vettore è ordinato al contrario.

- Per ogni i : 1 sottrazione, 1 assegnazione. Per tutti gli $i - 1$ elementi precedenti un numero costante di operazioni elementari all'interno del corpo del **while**.

$$\sum_{i=1}^{n-1} (2 + (i - 1)c) \approx d \cdot n^2$$

Per una opportuna costante d .



Analisi della complessità degli algoritmi: upper-bound

- Per semplicità si ignorano le costanti utilizzando soltanto gli ordini di grandezza.
- $f, g : \mathbb{N}^+ \rightarrow \mathbb{N}^+$, $f \in O(g)$ se $f(n) \leq cg(n)$ a partire da un $n \geq n_0$.
- Se $T(n)$ è il numero di passi che esegue **selectionsort** per ordinare n elementi allora $T \in O(n^2)$
- Diciamo che **selectionsort** ha *complessità* temporale $O(n^2)$.



Analisi della complessità degli algoritmi: lower-bound

- $f, g : \mathbb{N}^+ \rightarrow \mathbb{N}^+$, $f \in \Omega(g)$ se $f(n) \geq cg(n)$ a partire da un $n \geq n_0$.
- Se $T(n)$ è il numero di passi che esegue **selectionsort** per ordinare n elementi allora $T \in \Omega(n)$
- Qualsiasi algoritmo di ordinamento deve leggere l'input.
- Il numero di passi che deve seguire un qualsiasi algoritmo di ordinamento è $\Omega(n)$.

