

Algoritmo di ordinamento

1. Introduzione e requisiti del problema
2. Specifica
3. Progetto della soluzione
4. Codifica

1. Introduzione e requisiti del problema

Requisiti del problema

Scrivere un programma che dato in ingresso un vettore di interi stampi a video lo stesso vettore ordinato in senso crescente.

In questa esercitazione si affronterà il problema *dell'ordinamento di un vettore di interi* attraverso l'uso di diversi algoritmi di ordinamento. Gli algoritmi presentati verranno poi confrontati fra di loro in termini di complessità e tempi di esecuzione.

Il problema di ordinare un insieme di elementi è un problema frequente e ampiamente affrontato nel mondo dell'informatica. Esistono diversi algoritmi di ordinamento.

In questa esercitazione verranno proposti i seguenti algoritmi:

- *algoritmo di ordinamento per inserimento*
- *algoritmo di ordinamento per selezione*
- *algoritmo di ordinamento per scambio*

Casi di test

caso 1:

Input: 9 7 15 31 8 2 10 6 12 18
Output: 2 6 7 8 9 10 12 15 18 31

caso 2:

Input: 3 5 7 14 23 30 41 45 60 90
Output: 3 5 7 14 23 30 41 45 60 90

Il programma richiede di leggere, ordinare e stampare un vettore di interi.

Le funzioni di lettura e di stampa del vettore sono lasciate come esercizio per lo studente.

Il problema dell'ordinamento di insiemi di dati è un classico problema di programmazione, con molti riscontri nella pratica.

Ad esempio la ricerca di un elemento di un vettore ordinato è molto più veloce rispetto al caso di un vettore non ordinato.

Algoritmo di scambio


Algoritmo semplice, ma poco efficiente per ordinare un insieme di dati.

Un vettore è ordinato se per ogni coppia di elementi di un array si verifica la seguente condizione:

$$\text{vett}[i] \leq \text{vett}[i+1]$$

Esempio

9 7 15 31 8 2 10 6 12 18



Il vettore non è ordinato

Algoritmo

```
for(scandire l'array)
{
    if(elemento corrente > elemento successivo)
        inverti elementi
}
```

Algoritmo

```
for(scandire l'array)
{
    if(elemento corrente > elemento successivo)
        inverti elementi
}
```

Algoritmo

```
for(scandire l'array)
{
    if(elemento corrente > elemento successivo)
        inverti elementi
}
```

Esempio

9 7 15 31 8 2 10 6 12 18

Esempio

7 9 15 31 8 2 10 6 12 18

Esempio

7 9 15 31 8 2 10 6 12 18

Esempio

7 9 15 31 8 2 10 6 12 18



Esempio

7 9 15 31 8 2 10 6 12 18

Esempio

7 9 15 8 31 2 10 6 12 18

Esempio

7 9 15 8 31 2 10 6 12 18

Esempio

7 9 15 8 2 31 10 6 12 18

Esempio

7 9 15 8 2 31 10 6 12 18



Esempio

7 9 15 8 2 10 31 6 12 18

Esempio

7 9 15 8 2 10 31 6 12 18

Esempio

7 9 15 8 2 10 6 31 12 18

Esempio

7 9 15 8 2 10 6 31 12 18

Esempio

7 9 15 8 2 10 6 12 31 18

Esempio

7 9 15 8 2 10 6 12 31 18

Esempio

7 9 15 8 2 10 6 12 18 31

Esempio

7 9 15 8 2 10 6 12 18 31



Il vettore non è ancora ordinato

Esempio

9 7 15 31 8 2 10 6 12 18

Esempio

7 9 15 31 8 2 10 6 12 18

Esempio

7 9 15 8 31 2 10 6 12 18

Esempio

7 9 15 8 2 31 10 6 12 18

Esempio

7 9 15 8 2 10 31 6 12 18

Esempio

7 9 15 8 2 10 6 31 12 18

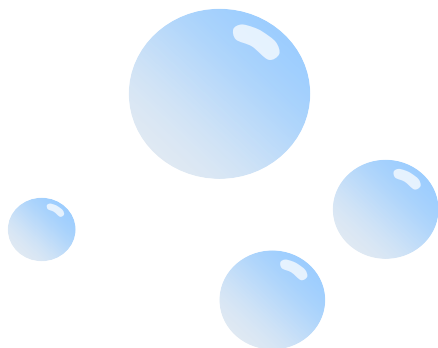
Esempio

7 9 15 8 2 10 6 12 31 18



Esempio

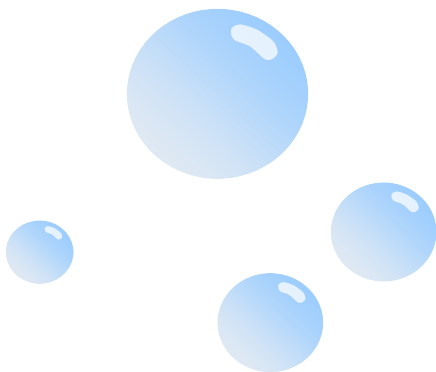
7 9 15 8 2 10 6 12 18 31





Esempio

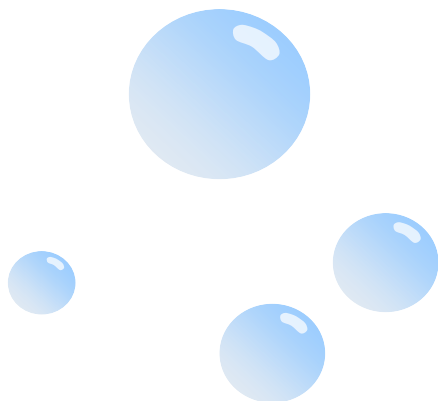
7 9 15 8 2 10 6 12 18 31





Esempio

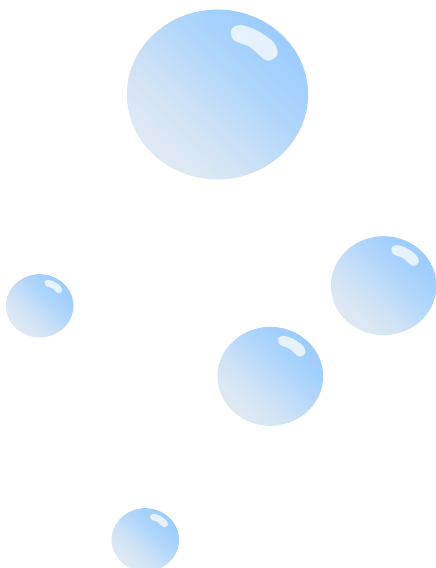
7 9 15 8 2 10 6 12 18 31





Esempio

7 9 15 8 2 10 6 12 18 31



Esempio

7 9 15 8 2 10 6 12 18 31

bubble sort

Esempio

7 9 15 8 2 10 6 12 18 31

Esempio

7 9 15 8 2 10 6 12 18 31

Esempio

7 9 15 8 2 10 6 12 18 31

Esempio

7 9 8 15 2 10 6 12 18 31

Esempio

7 9 8 15 2 10 6 12 18 31

Esempio

7 9 8 2 15 10 6 12 18 31

Esempio

7 9 8 2 15 10 6 12 18 31

Esempio

7 9 8 2 10 15 6 12 18 31

Esempio

7 9 8 2 10 15 6 12 18 31

Esempio

7 9 8 2 10 6 15 12 18 31

Esempio

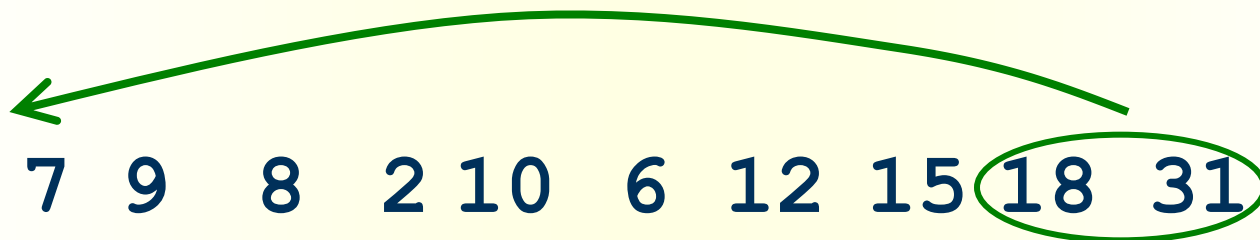
7 9 8 2 10 6 15 12 18 31

Esempio

7 9 8 2 10 6 12 15 18 31

Esempio

7 9 8 2 10 6 12 15 18 31

Esempio

Variabile=FALSE, TRUE

Algoritmo

```
ordinato=FALSE
while (ordinato==FALSE)
{
    ordinato=TRUE
    for (scandisco l'array)
        if (valore corrente > valore successivo)
        {
            ordinato=FALSE
            tmp=valore successivo
            valore successivo=valore corrente
            valore corrente=tmp
        }
}
```

Algoritmo di ordinamento per Inserimento

Approccio **diverso** dall'algoritmo per scambio.

L'algoritmo per inserimento si basa sulla costruzione di una sottosequenza ordinata in cui, via via che si procede nella lettura del vettore, si inserisce nella corretta posizione l'elemento correntemente letto.

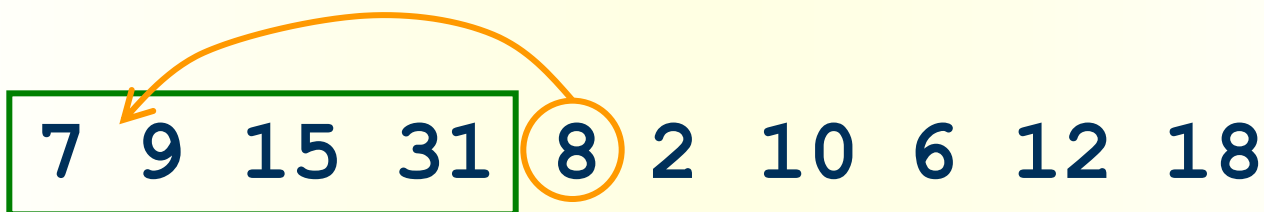
Esempio

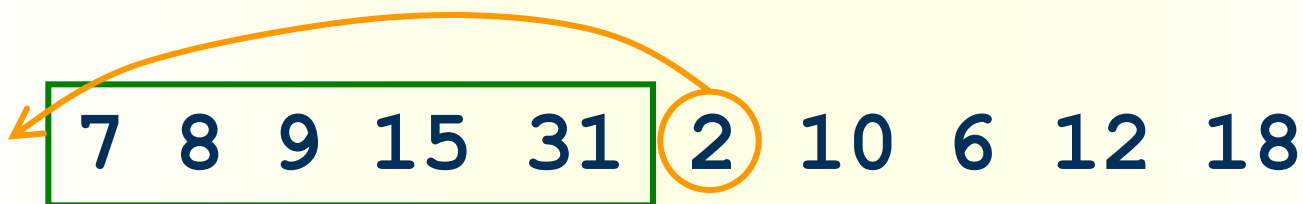
Esempio

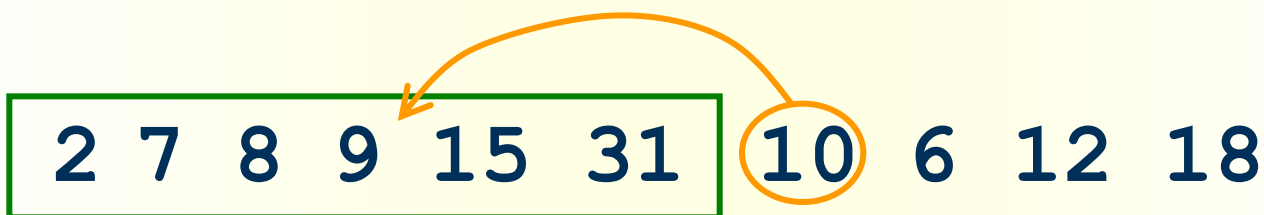
7 9 15 31 8 2 10 6 12 18

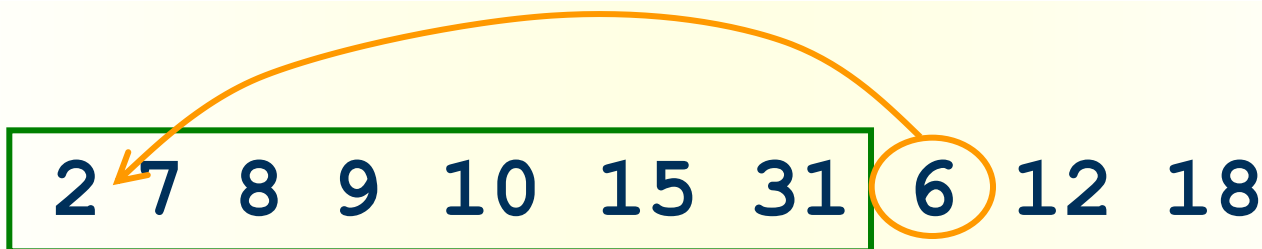
Esempio

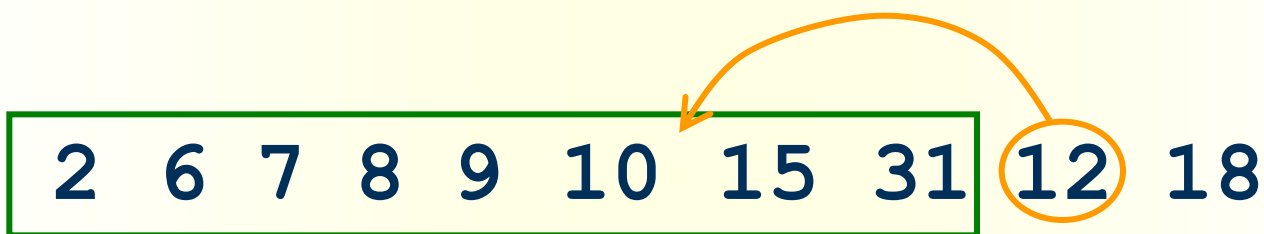
7 9 15 31 8 2 10 6 12 18

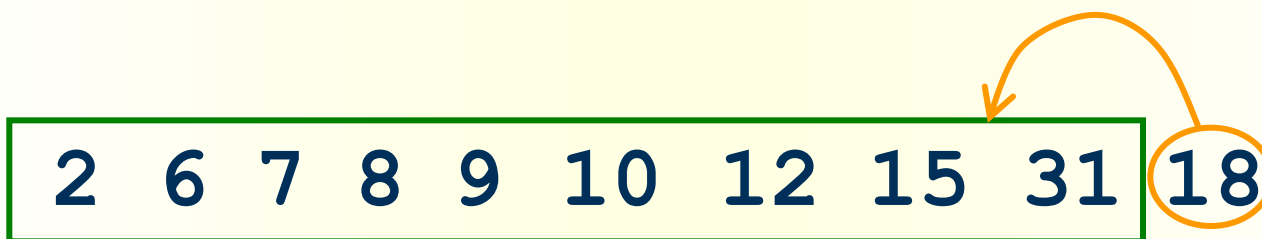
Esempio

Esempio

Esempio

Esempio

Esempio

Esempio

Esempio

2	6	7	8	9	10	12	15	18	31
---	---	---	---	---	----	----	----	----	----

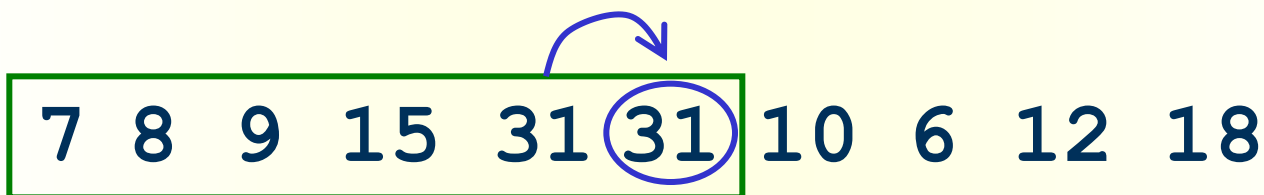
Esempio

7 8 9 15 31 2 10 6 12 18

Esempio

7 8 9 15 31 2 10 6 12 18

Cur=2

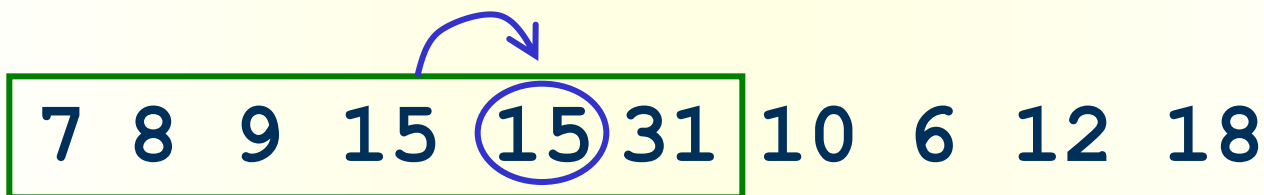
Esempio

Cur=2

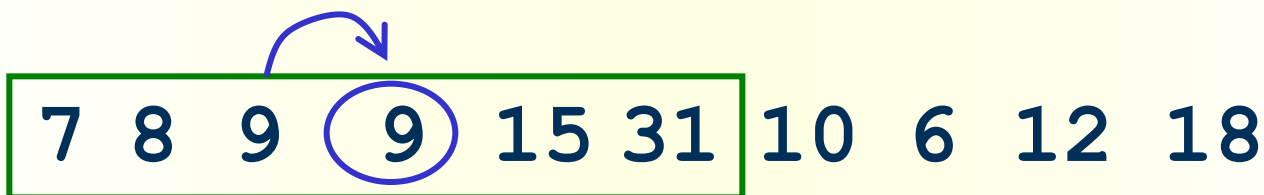
Esempio

7 8 9 15 31 31 10 6 12 18

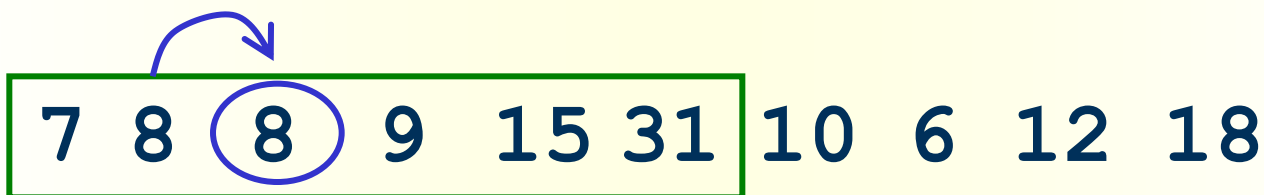
Cur=2

Esempio

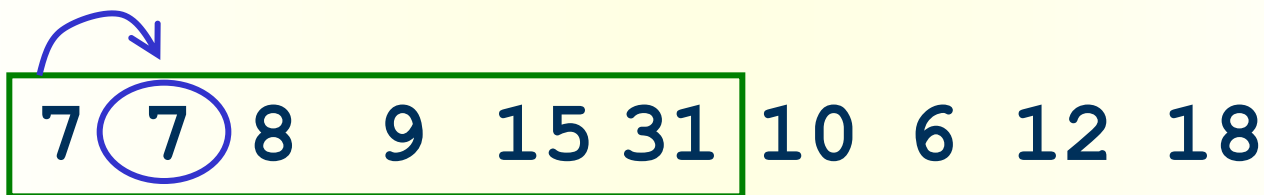
Cur=2

Esempio

Cur=2

Esempio

Cur=2

Esempio

Cur=2

Esempio

2 7 8 9 15 31 10 6 12 18

Cur=2

Algoritmo

```
for(scandisci l'array)
{
    curr=elemento corrente
    while((non sono in testa al vettore)&&
        (curr<elemento in esame))
    {
        elemento in esame=elemento precedente
        a quello in esame
    }
    elemento in esame=curr
}
```

Algoritmo

```
for(scandisci l'array)
{
    curr=elemento corrente
    while((non sono in testa al vettore)&&
        (curr<elemento in esame))
    {
        elemento in esame=elemento precedente
        a quello in esame
    }
    elemento in esame=curr
}
```

Algoritmo

```
for(scandisci l'array)
{
    curr=elemento corrente
    while((non sono in testa al vettore)&&
        (curr<elemento in esame))
    {
        elemento in esame=elemento precedente
        a quello in esame
    }
    elemento in esame=curr
}
```

Algoritmo

```
for(scandisci l'array)
{
    curr=elemento corrente
    while((non sono in testa al vettore)&&
          (curr<elemento in esame))
    {
        elemento in esame=elemento precedente
        a quello in esame
    }
    elemento in esame=curr
}
```

Algoritmo

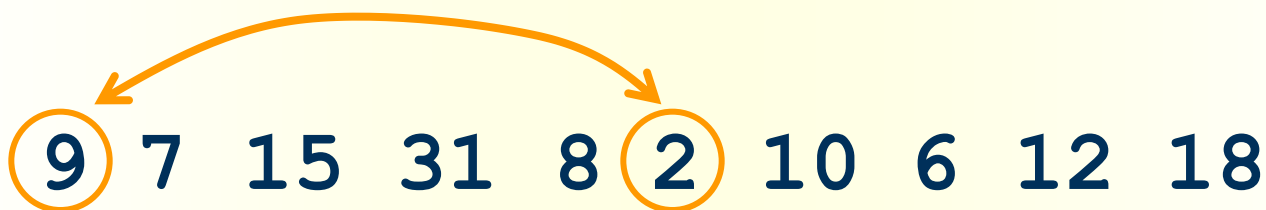
```
for(scandisci l'array)
{
    curr=elemento corrente
    while((non sono in testa al vettore)&&
        (curr<elemento in esame))
    {
        elemento in esame=elemento precedente
        a quello in esame
    }
    elemento in esame=curr
}
```


Algoritmo

```
for(scandisci l'array)
{
    curr=elemento corrente
    while((non sono in testa al vettore)&&
        (curr<elemento in esame))
    {
        elemento in esame=elemento precedente
        a quello in esame
    }
    elemento in esame=curr
}
```

Algoritmo di selezione

L'algoritmo di selezione si basa sulla costruzione dell'array ordinato, tramite il riconoscimento e il posizionamento dei valori.

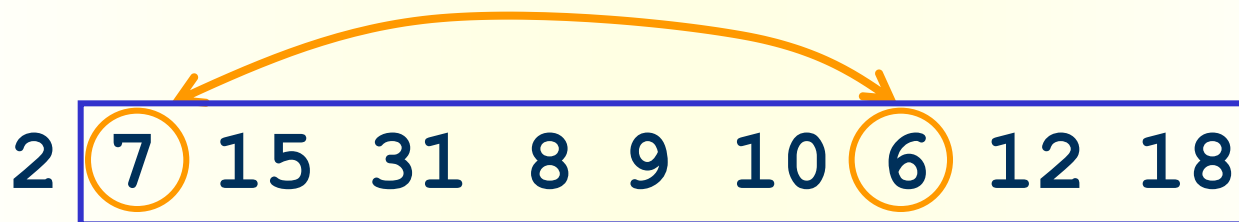
Esempio

Valore minimo= 2

Esempio

② 7 15 31 8 ⑨ 10 6 12 18

Valore minimo= 2

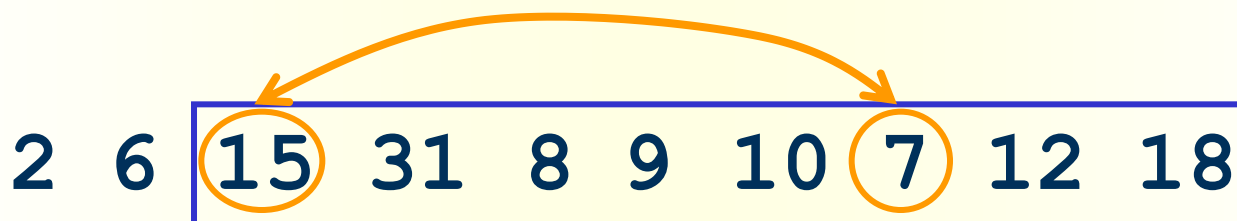
Esempio

Valore minimo= 6

Esempio

2 ⑥ 15 31 8 9 10 ⑦ 12 18

Valore minimo= 6

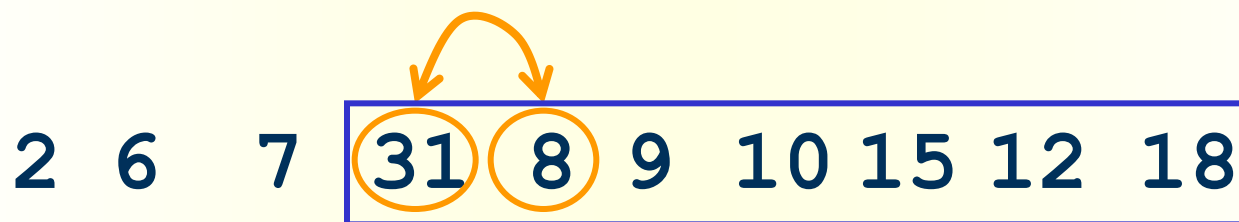
Esempio

Valore minimo= 7

Esempio

2 6 7 31 8 9 10 15 12 18

Valore minimo= 7

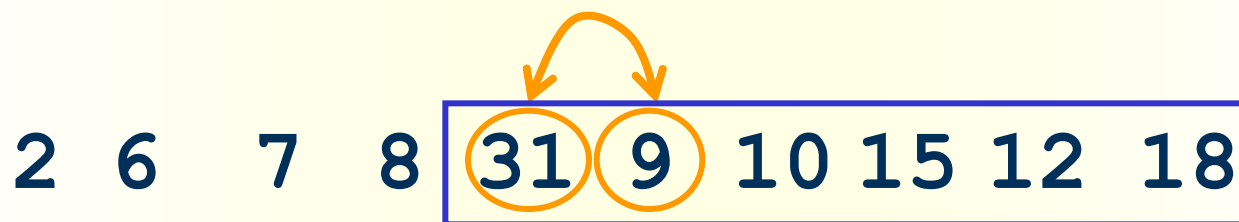
Esempio

Valore minimo= 8

Esempio

2 6 7 8 31 9 10 15 12 18

Valore minimo= 8

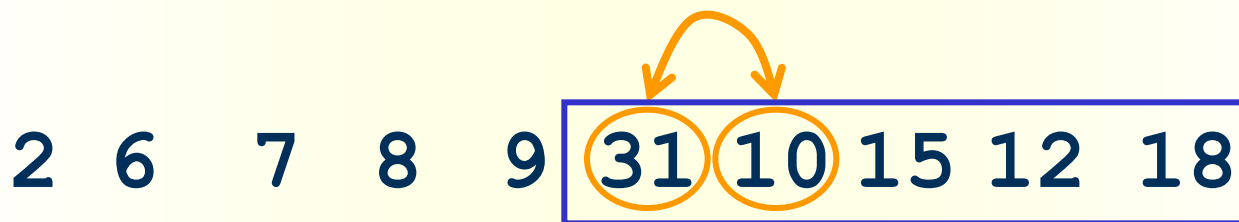
Esempio

Valore minimo= 9

Esempio

2 6 7 8 9 31 10 15 12 18

Valore minimo= 9

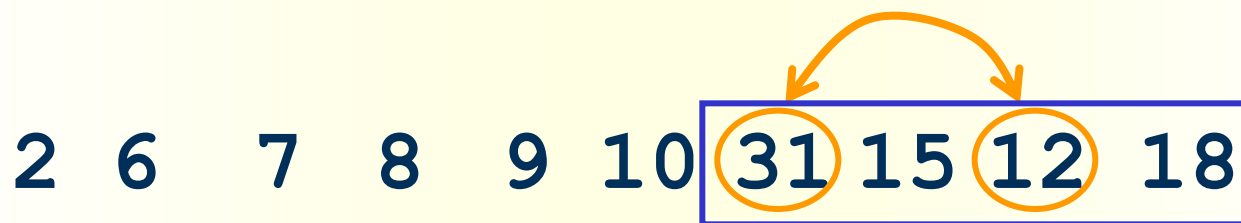
Esempio

Valore minimo=10

Esempio

2 6 7 8 9 10 31 15 12 18

Valore minimo=10

Esempio

Valore minimo=12

Esempio

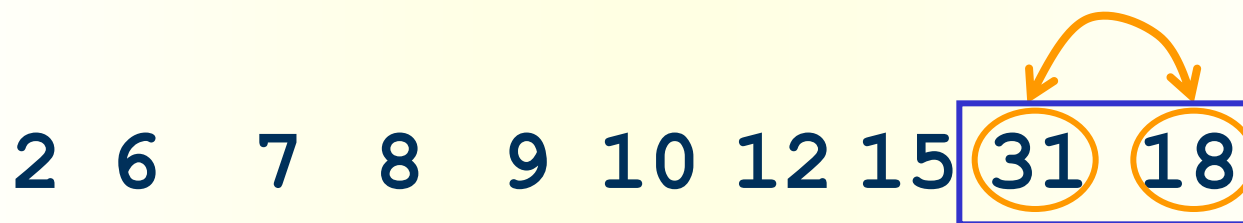
2 6 7 8 9 10 12 15 31 18

Valore minimo=12

Esempio

2 6 7 8 9 10 12 15 31 18

Valore minimo=15 (è in ordine)

Esempio

Valore minimo=18

Esempio

2 6 7 8 9 10 12 15 18 31

Valore minimo=18

Algoritmo

```
for(tutto l'array)
{
    for(sottosequenza non ordinata){
        trova il valore minimo}
    scambia il valore minimo con il primo elemento della
        sottosequenza
}
```

Algoritmo

```
for(tutto l'array)
{
    for(sottosequenza non ordinata){
        trova il valore minimo}
    scambia il valore minimo con il primo elemento della
        sottosequenza
}
```

Algoritmo

```
for(tutto l'array)
{
    for(sottosequenza non ordinata){
        trova il valore minimo}
    scambia il valore minimo con il primo elemento della
        sottosequenza
}
```

Algoritmo

```
for(tutto l'array)
{
  for(sottosequenza non ordinata){
    trova il valore minimo}
    scambia il valore minimo con il primo elemento della
      sottosequenza
  }
```

Confronto tra algoritmi

Per confrontare gli algoritmi è necessario definire dei criteri generali di confronto.

Confronto tra algoritmi

criteri di riferimento:

- velocità di ordinamento in un caso generale
- valore delle prestazioni nei casi più o meno favorevoli

La tipologia di dati di ingresso può fortemente influenzare le prestazioni degli algoritmi.

Confronto tra algoritmi

Alcuni algoritmi hanno dei comportamenti ottimi nel caso di dati di ingresso medi ma pessimi negli altri.

La scelta dell'algoritmo di ordinamento è una scelta di progetto importante.

Confronto tra algoritmi

Criterio aggiuntivo

➤ Comportamento naturale

Un algoritmo si comporta in maniera naturale se interviene poco quando il vettore è abbastanza ordinato.

Confronto tra algoritmi

La definizione di dato "medio" o meno "favorevole" si ottiene tramite applicazioni di metodi statistici.

La velocità di un algoritmo è calcolata in base al numero di scambi e di confronti che avvengono fra gli elementi dell'array.

Confronto tra algoritmi

Algoritmo di scambio

il numero di confronti dipende solo dalla lunghezza dell'array.

numero di confronti=

$$\frac{1}{2}(n^2 - n)$$

Confronto tra algoritmi

Algoritmo di scambio

numero di scambi in base alla tipologia
di dati=

Vettore ordinato → 

Vettore semiordinato → $\frac{3}{4} (n^2 - n)$

Vettore non ordinato → $\frac{3}{2} (n^2 - n)$

Confronto tra algoritmi

Algoritmo di scambio

numero di scambi in base alla tipologia
di dati=

Vettore ordinato



0

Vettore semiordinato



$$\frac{3}{4} (n^2 - n)$$

Vettore non ordinato



$$\frac{3}{2} (n^2 - n)$$

Confronto tra algoritmi

Algoritmo di scambio

numero di scambi in base alla tipologia
di dati=

Vettore ordinato → 0

Vettore semiordinato → $\frac{3}{4} (n^2 - n)$

Vettore non ordinato → $\frac{3}{2} (n^2 - n)$

Confronto tra algoritmi

Algoritmo di scambio

L'efficienza dell'algoritmo di scambio dipende dal quadrato del numero degli elementi dell'array, perché si può considerare il termine di grado più alto.

Confronto tra algoritmi

Algoritmo di selezione

numero di confronti=

$$\frac{1}{2}(n^2 - n)$$

Confronto tra algoritmi

Algoritmo di selezione

numero di scambi=

Vettore ordinato → $3(n-1)$ Vettore semiordinato → $n(\log(n) + y)$ Vettore non ordinato → $\frac{n^2}{4} + 3(n-1)$

Confronto tra algoritmi

Algoritmo di selezione

numero di scambi=

Vettore ordinato $\rightarrow 3(n-1)$ Vettore semiordinato $\rightarrow n(\log(n) + y)$ Vettore non ordinato $\rightarrow \frac{n^2}{4} + 3(n-1)$

Confronto tra algoritmi

Algoritmo di selezione

numero di scambi=

Vettore ordinato $\rightarrow 3(n-1)$ Vettore semiordinato $\rightarrow n(\log(n) + y)$ Vettore non ordinato $\rightarrow \frac{n^2}{4} + 3(n-1)$

Confronto tra algoritmi

Algoritmo per inserimento

il numero di confronti è in funzione del livello di ordinamento del vettore.

numero di confronti=

Vettore ordinato → $n-1$

Vettore semiordinato → $\frac{1}{4} (n^2 - n)$

Vettore non ordinato → $\frac{1}{2} (n^2 - n)$

Confronto tra algoritmi

Algoritmo per inserimento

il numero di confronti è in funzione del livello di ordinamento del vettore.

numero di confronti=

Vettore ordinato → $n-1$

Vettore semiordinato → $\frac{1}{4} (n^2 - n)$

Vettore non ordinato → $\frac{1}{2} (n^2 - n)$

Confronto tra algoritmi

Algoritmo per inserimento

il numero di confronti è in funzione del livello di ordinamento del vettore.

numero di confronti=

Vettore ordinato → $n-1$

Vettore semiordinato → $\frac{1}{4} (n^2 - n)$

Vettore non ordinato → $\frac{1}{2} (n^2 - n)$

Confronto tra algoritmi

Algoritmo per inserimento

numero di scambi=

Vettore ordinato → $2(n-1)$

Vettore semiordinato → $\frac{1}{4}(n^2-n)$

Vettore non ordinato → $\frac{1}{2}(n^2-n)$

Confronto tra algoritmi

Algoritmo per inserimento

numero di scambi=

Vettore ordinato $\rightarrow 2(n-1)$

Vettore semiordinato $\rightarrow \frac{1}{4}(n^2-n)$

Vettore non ordinato $\rightarrow \frac{1}{2}(n^2-n)$

Confronto tra algoritmi

Algoritmo per inserimento

numero di scambi=

Vettore ordinato $\rightarrow 2(n-1)$

Vettore semiordinato $\rightarrow \frac{1}{4}(n^2-n)$

Vettore non ordinato $\rightarrow \frac{1}{2}(n^2-n)$

Confronto tra algoritmi

In generale i tre algoritmi hanno delle prestazioni proporzionali al quadrato del numero degli elementi di un vettore.

Il migliore dei tre è l'algoritmo **per inserimento** perché ha un comportamento naturale ovvero interviene poco se l'array è quasi ordinato.