Strutture dati Astratte in C++ Parte II

A CURA DEL TEAM INFORMATICA OPENSOURCE

SITO WEB HTTPS://WWW.INFORMATICAOPENSOUCE.IT

Algoritmi notevoli per gli array

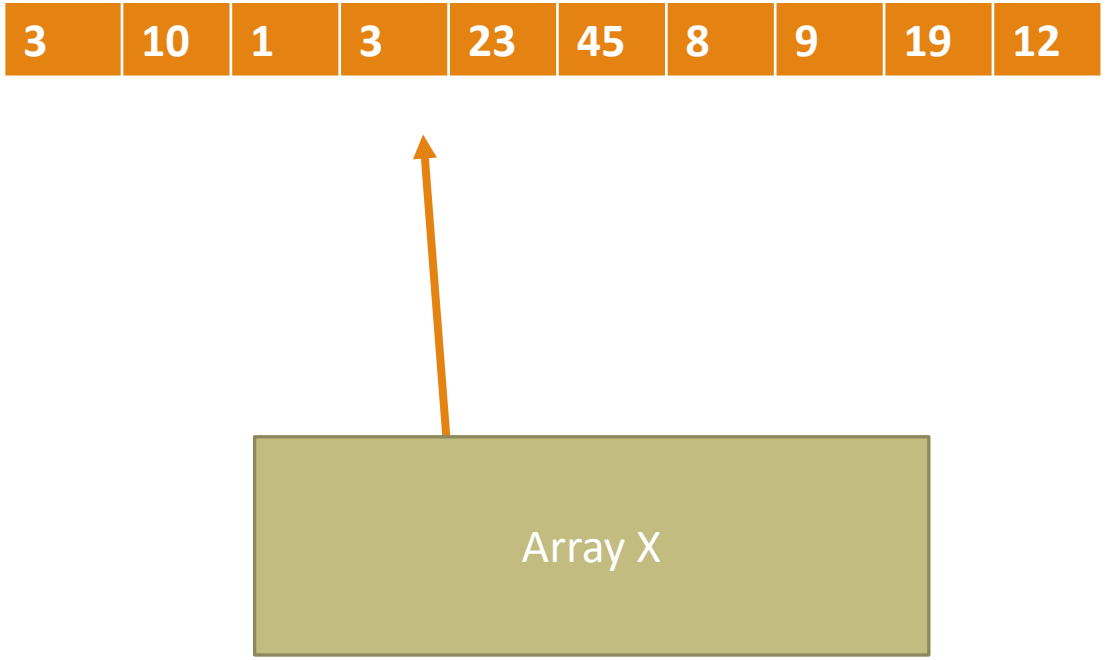
Ordinamento per selezione e bolle

Ricerca Dicotomica

Compattamento

Fusione di due array

Ordinamento per selezione



Ordinamento crescente o decrescente

I dati dell'arrray possono essere numerici o alfabetici

L'array ha dimensione N

L'ordinamento è eseguito con un doppio ciclo il primo fissa l'elemento, il secondo confrona lo stesso con tutti i successivi.
In caso di ordinamento crescente se l'elemento di riferimetno p minore di quello puntato dal secondo ciclo avvien lo scambio.

Ordinamento per selezione

3	10	1	3	23	45	8	9	19	12

1	10	3	3	23	45	8	9	19	12

1	3	10	3	23	45	8	9	19	12

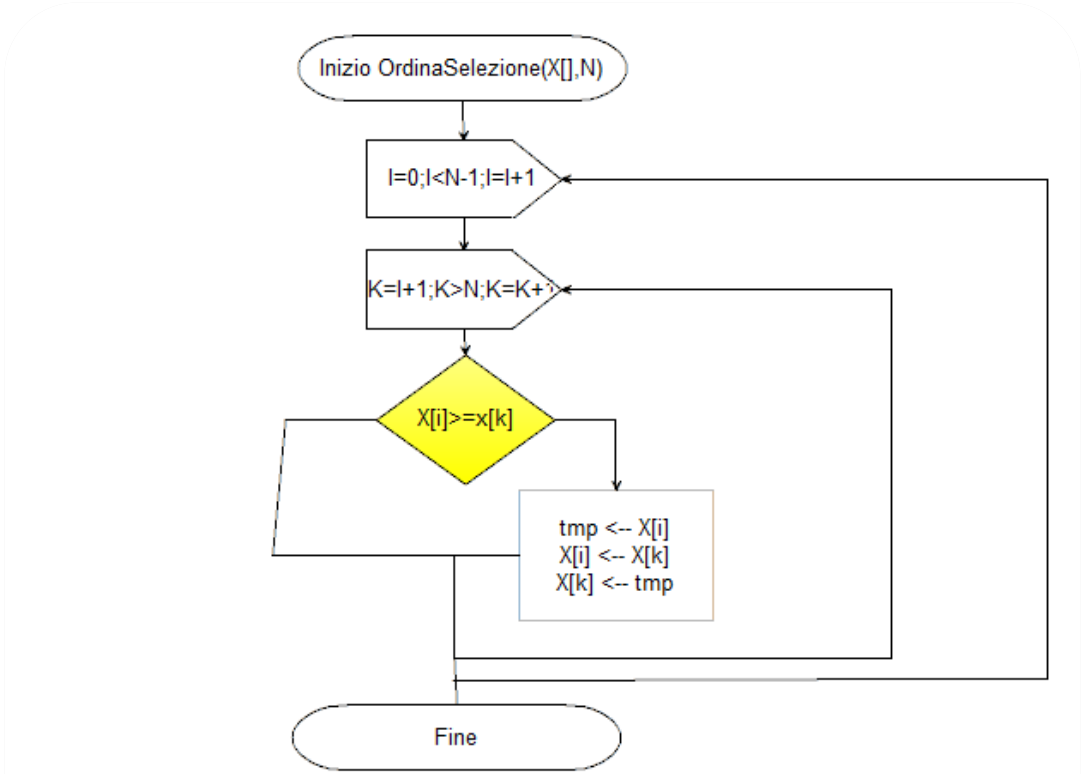
1	3	3	10	23	45	8	9	19	12

1	3	3	8	23	45	10	9	19	12

Nella prima passata I=0; e K varia da I+1 a N.
Avviene lo scambio fra X[i=0] e X[K=2].
L'indice K è incrementato fino a N-1 e poi viene reimpostato dal nuovo valore dell'indice I=1.

In questa seconda tornata di confronti non si scambia nulla poiché X[i=1]=3 ed è l'elemento più piccolo.
L'indice I diventa I=2 e in questa tornata si scambiano X[i=2] e X[k=3] e successivamente X[K=6].

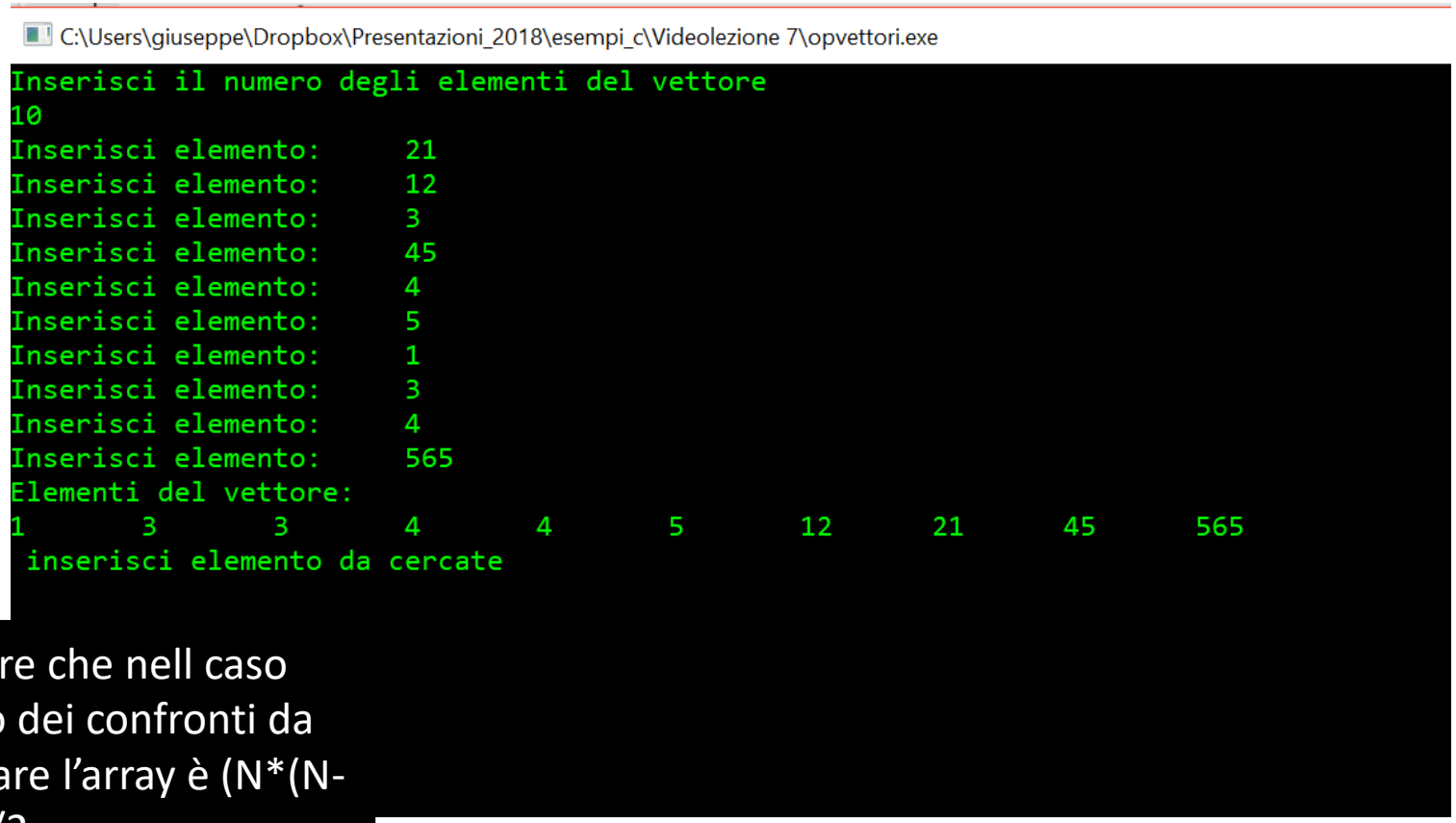
Algoritmo dell’ordinamento per selezione



Codifica in C++

```
void ordinaselezione (int x[],int l)
{
    int i,k,tmp;
    for (i=0;i<l-1;i++)
        for (k=i+1;k<n;k++)
            if (x[i]>=x[k])
                {
                    tmp=x[i];
                    x[i]=x[k];
                    x[k]=tmp;
                }
}
```

Esecuzione del programma



E’ facile dimostrare che nell caso peggiore il numero dei confronti da effettuare per ordinare l’array è (N*(N-1))/2

Algoritmo Bubble Sort o ordinamento a bolle

L’algoritmo per selezione è più semplice e intuitivo, però non è efficiente

E’ possibile pensare ad una soluzione alternativa, operando un confronto fra elementi adiacenti del vettore.

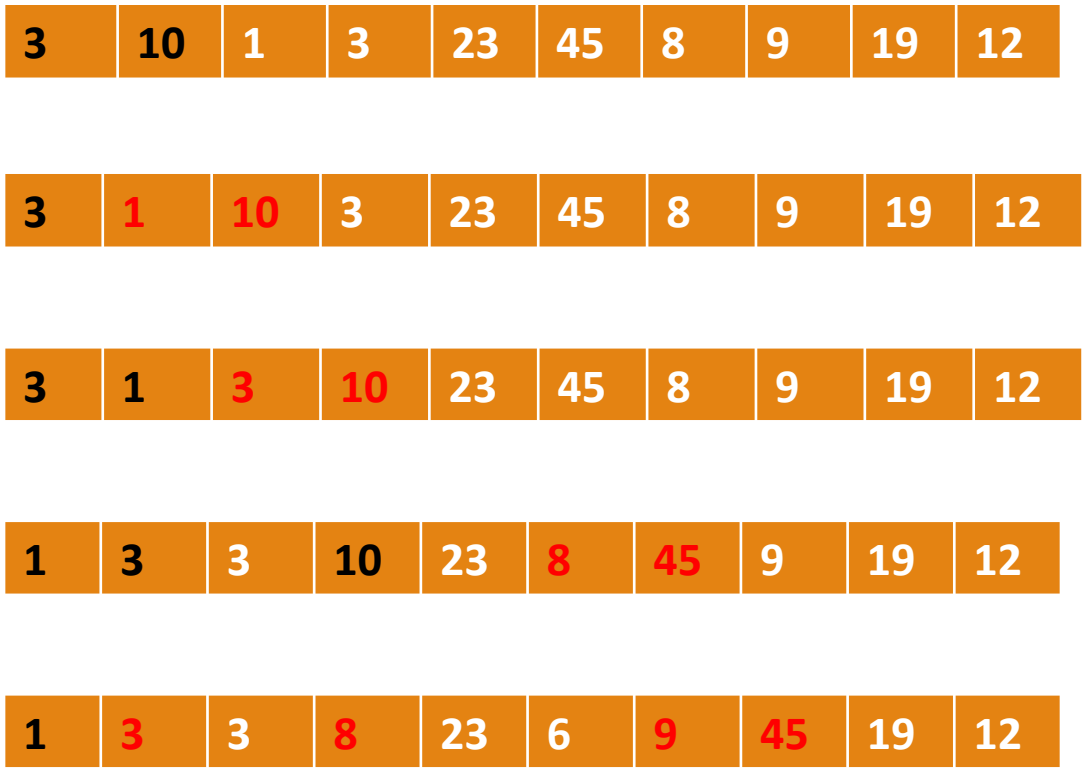
In questo modo ad esempio nell’ordinamento crescente ogni volta l’elemento x[i] è maggiore o uguale all’elemento x[j+1] avviene lo scambio

E’ utilizzato un flag che assume valore 0 per indicare che il vettore o array non ci sono stati scambi e il valore 1 se invece nella scansione dell’array è avvenuto almeno uno scambio

E’ un algoritmo ottimizzato che nel caso peggiore svolge N^2 scambi

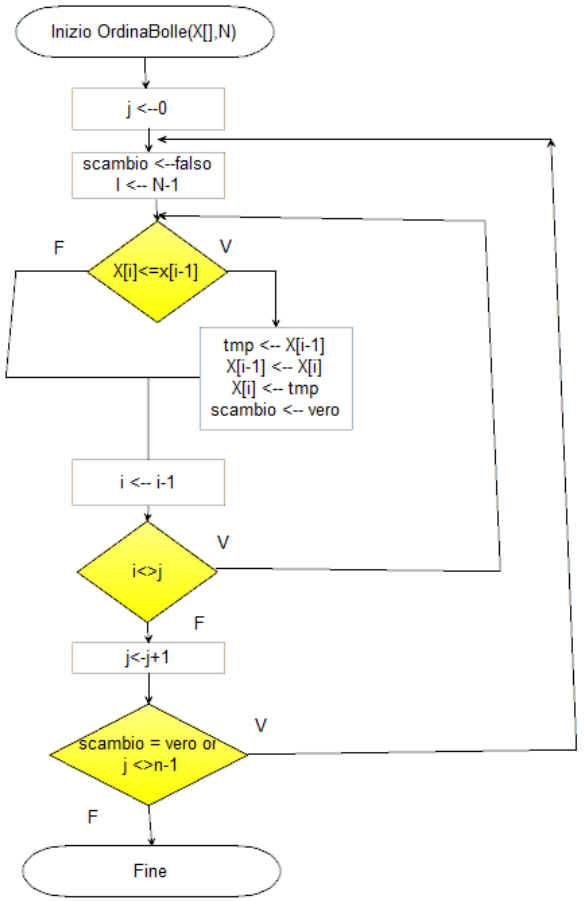
Sono necessari01 due cicli condizionali

Ordinamento a bolle



In questo esempio gli elementi sono scambiati quando si verifica la condizione che l'elemento di posto i è maggiore uguale dell'elemento di posto i+1.
Ogni volta che si verifica lo scambio il flag viene posto a zero e il ciclo continua l'esecuzione.

Algoritmo Bubble Sort Ottimizzato



Codifica in C della funzione

```
void ordinabolle(int x[],int l)
{
    int i,tmp,j;
    bool scambio;
    j=0;
    do {
        scambio=false;
        i=l-1;
        do {
            if (x[i] <=x[i-1])
                {
                    tmp=x[i-1];
                    x[i-1]=x[i];
                    x[i]=tmp;
                    scambio = true;
                }
                i--;}
        while (i!=j);
        j++;
        }
    while (scambio==true && j!=l-1);
    }
```

Esecuzione del programma

```
Inserisci il numero degli elementi del vettore
10
Inserisci elemento:     3
Inserisci elemento:     4
Inserisci elemento:     5
Inserisci elemento:     6
Inserisci elemento:     7
Inserisci elemento:     9
Inserisci elemento:     -3
Inserisci elemento:     -4
Inserisci elemento:     1
Inserisci elemento:     23
Elementi del vettore:
-4      -3      1       3       4       5       6       7       9       23
```

Ricerca dicotomica o binaria

L’array è ordinato

L’utente inserisci l’elemento incognito da cercare

L’array è diviso in due parti, le posizioni estreme sono chiamate inf e sup mentre la posizione centrale è chiamata centro

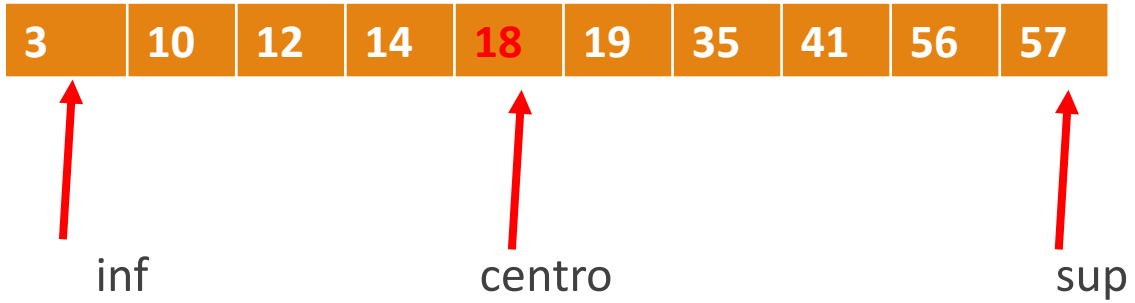
Se l’elemento è minore di quello puntato dalla variabile centro significa che l’elemento incognito è nel sotto array di sinistra, altrimenti in quello di destra

Il sotto array viene nuovamente suddiviso e sono riposizionate le variabili inf, sup, centro

L’elemento incognito viene confrontato con quello di centro e nel caso viene identificato una variabile flag termina il ciclo

Ricerca dicotomica

Array di partenza ordinato:

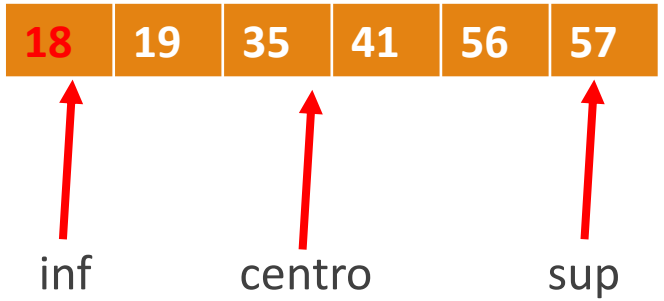


Esempio Elemento incognito y=19
inf=0 sup=9
centro=(inf+sup)/2

- Confronto x[centro] con y se y è minore allora il sotto vettore è quello di sinistra altrimenti quello di destra.
  Prima di fare questo si verifica se l'elemento centrale è proprio uguale a y in caso affermativo la ricerca termina e la variabile flag diventa vera
- Tale procedimento è iterato fino alla inf non diventa superiore a sup. Nel momento in cui inf è maggiore a sup vuol dire che la ricerca termina e che l'elemento non è stato trovato

Ricerca dicotomica seconda scansione

Array di partenza ordinato:

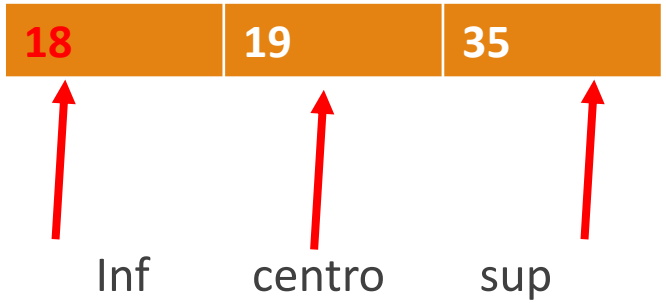


Esempio Elemento incognito y=19
inf=5 sup=9
centro=(inf+sup)/2

- Confronto x[centro] con y se y è minore allora il sotto vettore è quello di sinistra altrimenti quello di destra.
  Prima di fare questo si verifica se l'elemento centrale è proprio uguale a y in caso affermativo la ricerca termina e la variabile flag diventa vera
- In questo caso x[centro] è superiore a y e il sotto vettore da considerare è quello di sinistra

Ricerca dicotomica terza scansione

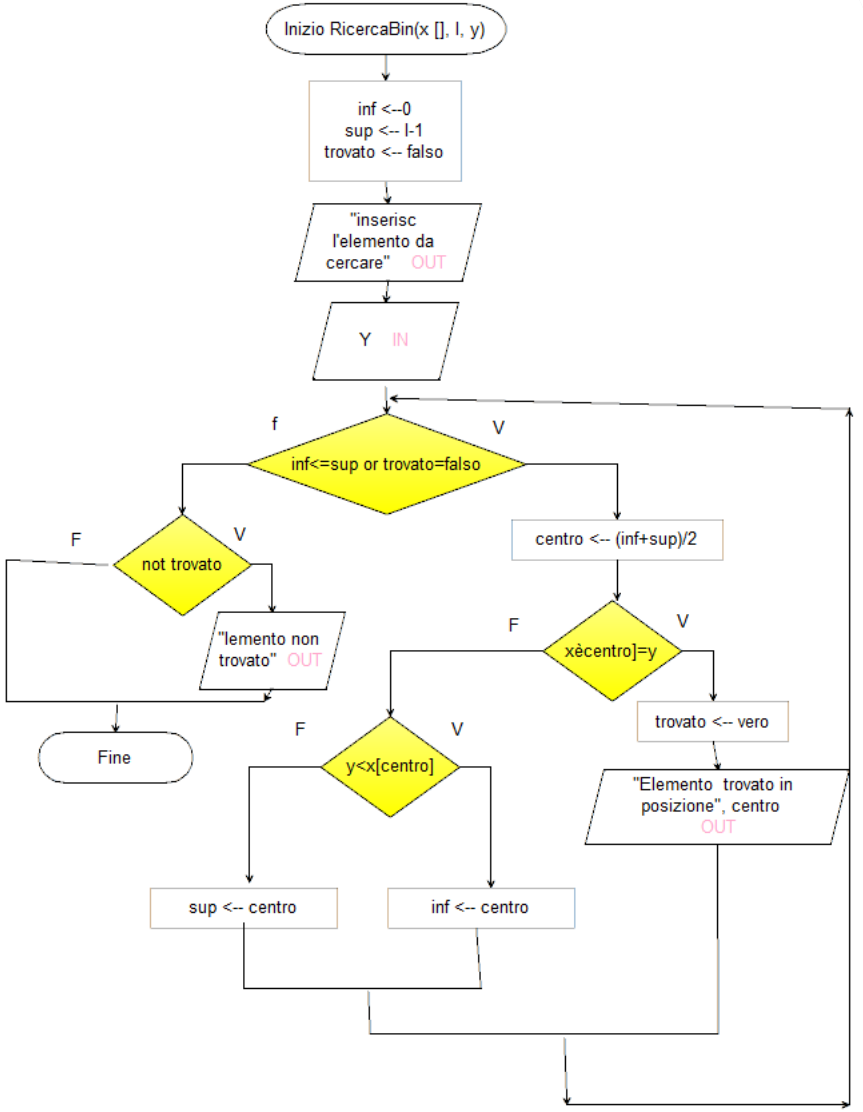
Array di partenza ordinato:



Esempio Elemento incognito y=19
inf=5 sup=7
centro=(inf+sup)/2

- In quest'ultimo caso x[centro] è proprio l'elemento y incognito e quindi il flag diventa vero e il ciclo termina
- Se l'elemento non fosse stato trovato il ciclo alla prossima suddivisione dell'array termina poiché inf non è minore uguale a sup

Algoritmo di ricerca dicotomica



Codifica in C++ della ricerca dicotomica

```
void ricercabin (int x[], int &y, int l)
{
    int inf,sup,centro;
    bool trovato;
    cout <<"\n Inserisci l'eleento da ricercare nell'array \t";
    cin >> y;
    inf=0;
    sup=l-1;
    trovato=false;
    while (inf <=sup and trovato==false)
    {
        centro = (inf+sup)/2;
        if (x[centro]==y)
        {
            trovato=true;
            cout <<"\nElemento trovato in posizione \t "<<centro;
        }
        else
            if (y< x[centro])
                sup=centro;
            else
                inf=centro;
    }
    if (!trovato)
        cout <<"\n Elemento non trovato nel vettore\n";
}
```

Esecuzione della ricerca dicotomica

```
Inserisci il numero degli elementi del vettore
10
Inserisci elemento:     3
Inserisci elemento:     4
Inserisci elemento:     5
Inserisci elemento:     6
Inserisci elemento:     7
Inserisci elemento:     9
Inserisci elemento:     -3
Inserisci elemento:     -4
Inserisci elemento:     1
Inserisci elemento:     23
Elementi del vettore:
-4      -3      1       3       4       5       6       7       9       23
 Inserisci l'eleento da ricercare nell'array    5
Elemento trovato in posizione    5
```

Compattamento Array – Il Problema

L’array contiene valori non più utili

La cancellazione fisica prevede di eliminare gli elementi segnati con un criterio e di compattare il vettore facendo rimanere solo gli elementi necessari

L’algoritmo di compattamento è strutturato ricopiando gli elementi che devono permanere nell’array in nuovo vettore di dimensione pari al numero degli elementi rimanenti nel vettore origine.

Gli elementi da eliminare sono selezionati mediante un algoritmo di ricerca eventualmente dicotomica su array ordinate.

Gli elementi da cancellare sono scelti dall’utente oppure segnati con valori non nel contesto

Ad esempio su un array che contiene i prezzi di alcuni prodotti la cancellazione logica può essere eseguita cambiando i prezzi dei prodotti in negativo. Questa tecnica permette di ripristinare in caso di errore i valori cancellati.

Una volta compattato l’array i valori sono eliminati definitivamente e non recuperabili.

Compattamento di un array

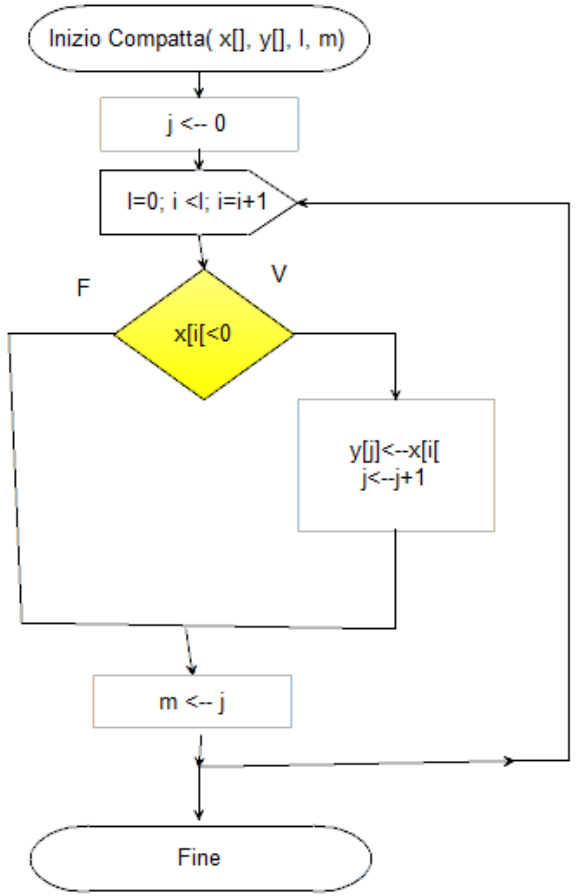
3	-10	12	-14	18	19	35	-41	56	57

poiché sono fuori contesto

La procedura prevede un algoritmo di ricerca sequenziale che ricopia ogni valore in contesto (in questo caso positivo) nel nuovo vettore che avrà dimensione al massimo pari al primo vettore

E’ possibile definire anche strutture dinamiche più idonee

Algoritmo risolutivo



Codifica in C++

```
void compatta (int x[],int y[],int l,int &m)
{
    int j,i;
    j=0;
    for (i=0;i<l;i++)
        if (x[i]>=0)
            {
                y[j]=x[i];
                j++;
            }
            m=j;
}
```

Fusione di due array

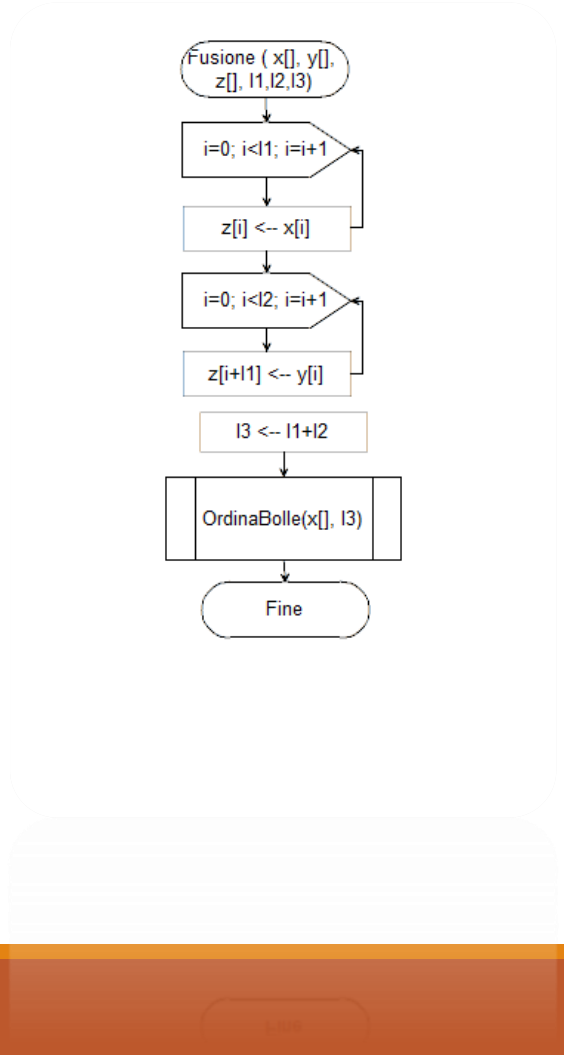
Due array ordinati fusi in un unico array

Le dimensioni dei due array sono diverse fra loro

Sequenza operativa

- Copia degli elementi dei due array in un terzo array
- Ordinamento a bolle dell'array risultatane
- L'algoritmo è molto semplice

Algoritmo di Fusione



Codifica in C++ dell'algoritmo

```
void fusione(int x[],int y[],int z[],int l1,int l2,int &l3)
{
    int i;
    for (i=0;i<l1;i++)
        z[i]=x[i];
    for (i=0;i<l2;i++)
        z[i+l1] = y[i];
        l3=l1+l2;
    ordinabolle(z,l3);
}
```

Fine Unità Didattica

Revisione 2021

A cura del Team informatica Opensource

Sito Web
https://www.informaticaopensource.it

Sito web per i corsi on line

https://www.corsi-on-line.it