Algoritmi di Ricerca elementi in unvettore

Prof. Francesco Accarino
IIS Altiero Spinelli Sesto San Giovanni

Algoritmi classici

- In ambito informatico alcuni problemi si presentano con elevata frequenza in più ambiti e sono stati ampiamente studiati
  - Ricerca di un elemento in un vettore
  - Ricerca del minimo e del massimo
  - Ordinamento
- Gli algoritmi impiegabili in questi casi sono numerosi. I più noti, di seguito presentati, vengono spesso impiegati anche come termini di paragone per valutare le prestazioni di nuove soluzione proposte

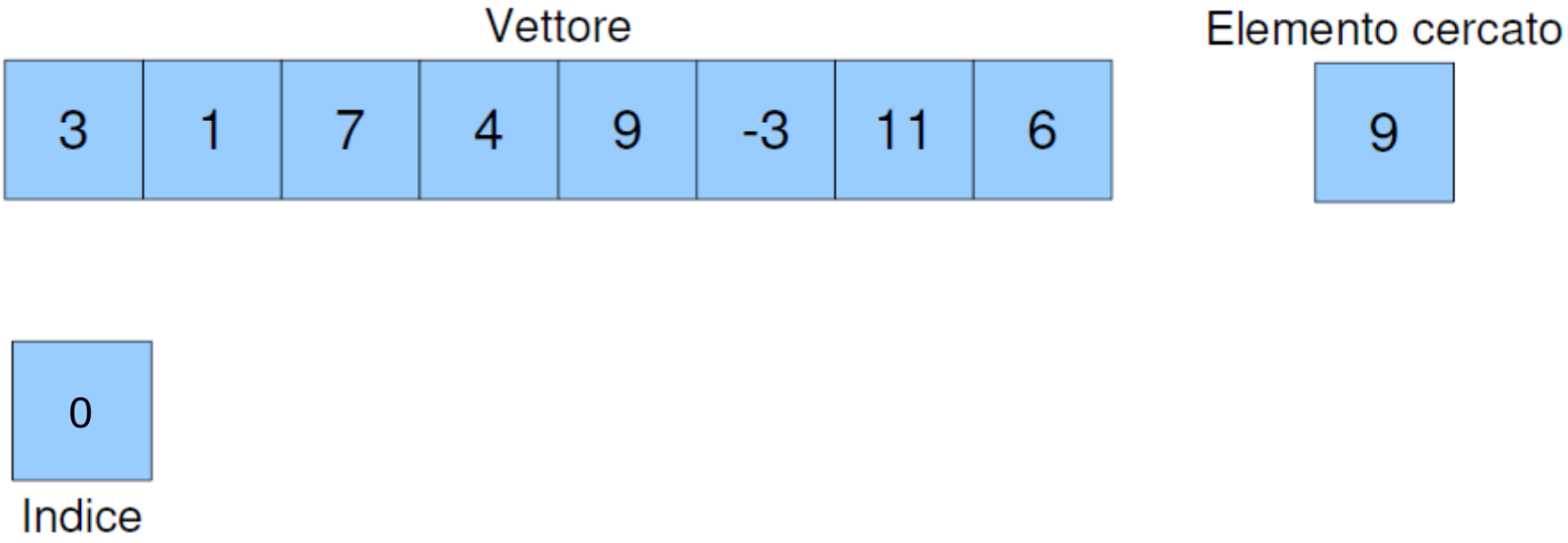
Algoritmi di ricerca di un elemento

- Il problema della ricerca di un elemento in un vettore si presenta frequentemente:
  - Occorre verificare se l'elemento appartiene al vettore
  - Ad un elemento (o alla sua posizione) sono associate informazioni supplementari

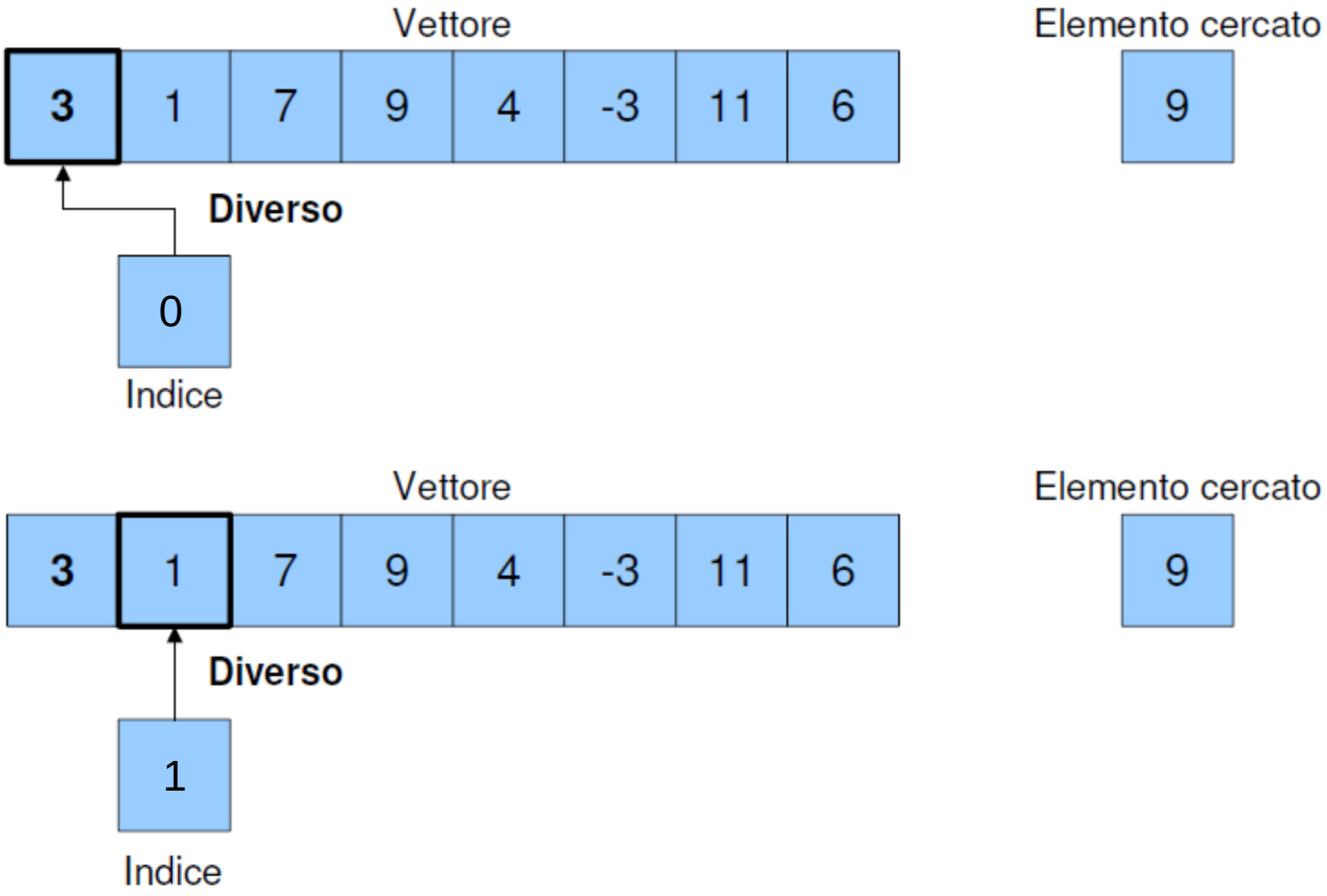
- Esistono tre algoritmi "standard" per la risoluzione di questo problema
  - Ricerca Lineare o Sequenziale
  - Ricerca dicotomica
  - Ricerca Completa

Ricerca Lineare o Sequenziale

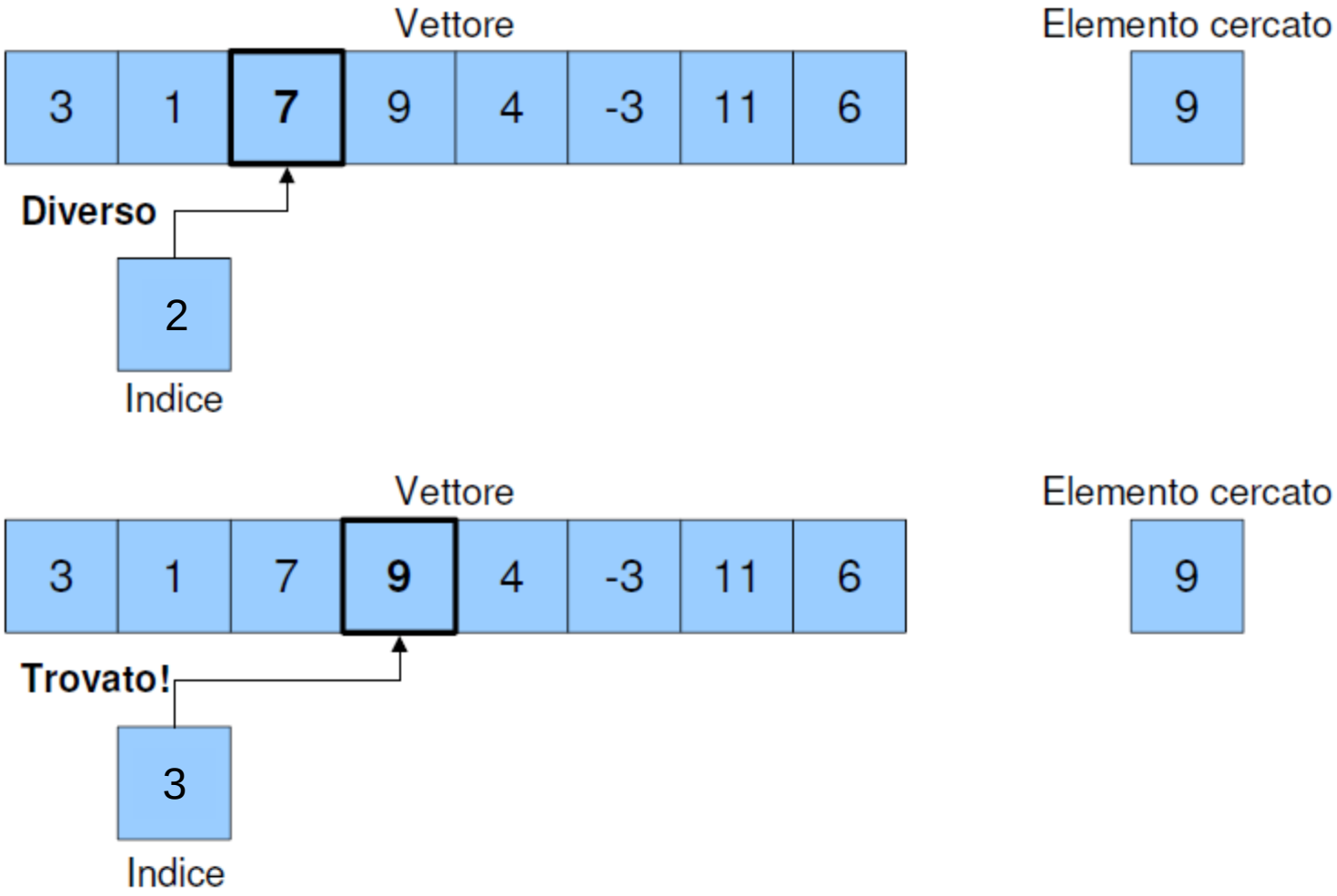
- L'idea di fondo è semplice: Si scorre l'intero vettore dalla prima posizione e si confronta ogni elemento con quello ricercato



Ricerca Lineare o Sequenziale



Ricerca Lineare o Sequenziale



Ricerca Lineare o Sequenziale

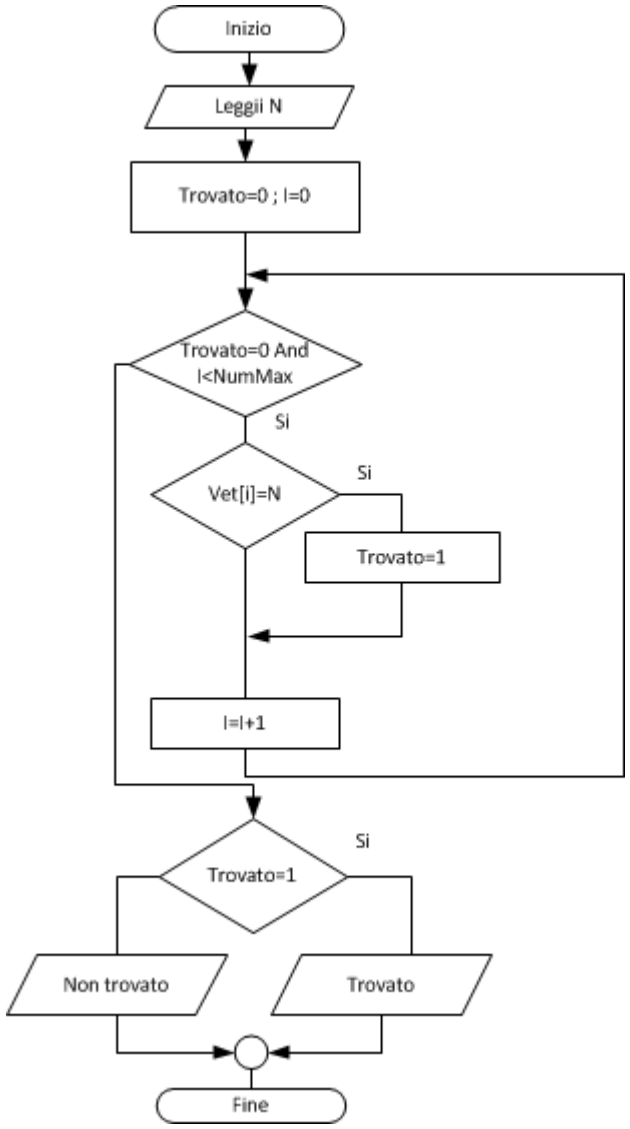
- L'algoritmo di ricerca sequenziale funziona senza richiedere particolari ipotesi sull'ordinamento dei dati

- Mediamente occorre scandire metà vettore per trovare l'elemento cercato (se c'è) infatti è possibile trovare l'elemento al primo tentativo o dopo N tentativi quindi in media: $\frac{N+1}{2}$ (casi favorevoli 1, N fratto i casi possibili)

- se non c'è, occorre fare tutti gli N tentativi
- Si dice che il tempo di esecuzione cresce linearmente al variare di N

Flowchart e Codifica in C

```
#include “stdio.h”
#include “conio.h”
#define NumMax 20
Void Main(){
Int vet[NumMax],i,N;
Printf(“inserisci il numero da cercare”);
Scanf(“%d”,&N);
Char Trovato=0;
i=0;
While(Trovato==0&&i<NumMax){
If(vet[i]==N
   Trovato=1;
i++;
}
If(Trovato==1)
   Printf(“trovato il numero %d”,N);
Else
 Printf(“non trovato il numero %d”,N);
}
```



Ricerca binaria

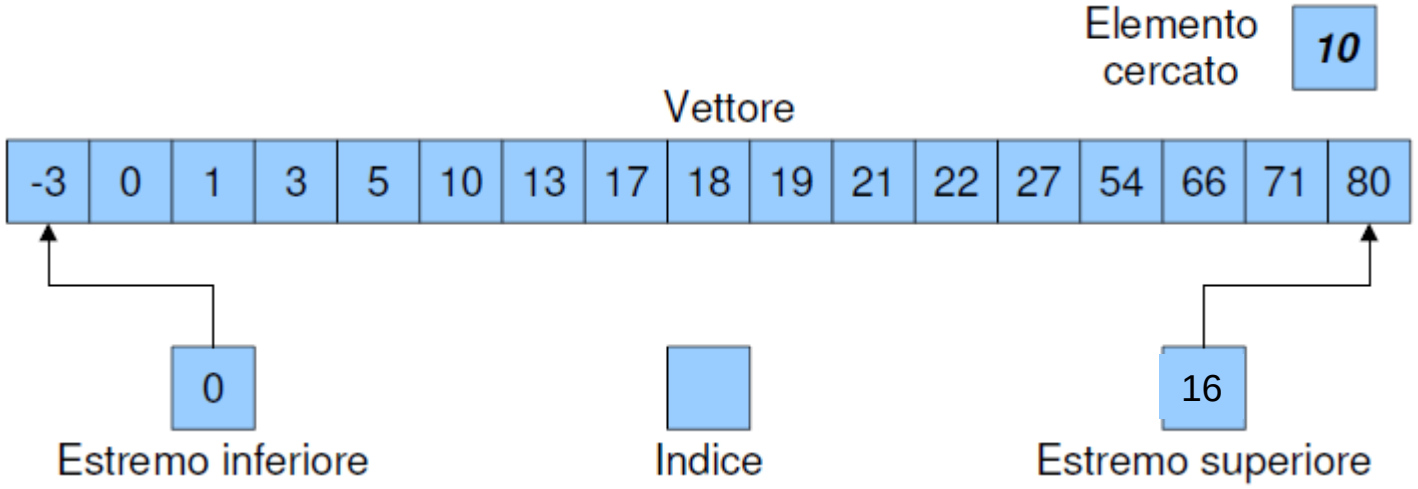
- Se il vettore è ordinato, allora è possibile utilizzare approcci più efficienti rispetto alla ricerca sequenziale
- La ricerca binaria prevede l'osservazione dell'elemento al centro del vettore e dei due estremi.
- Se uno di questi è l'elemento cercato termina, altrimenti scarta tutta una metà del vettore a seconda che l'elemento centrale sia maggiore o minore di quello cercato
- Il processo si ripete fino a trovare l'elemento cercato o a scartarli tutti

Ricerca binaria

- L'approccio è lo stesso adottato per cercare una parola nel dizionario o un nome nella rubrica telefonica
  - Es. cerco il numero di Rossi Mario
  - Apro circa a metà, sulla lettera N
  - Vado avanti di diverse pagine
  - Arrivo alla lettera S
  - Torno indietro a Ra…
  - Vado avanti di poco, arrivando a Rov..
  - Ci sono quasi, torno indietro di una pagina
  - etc…

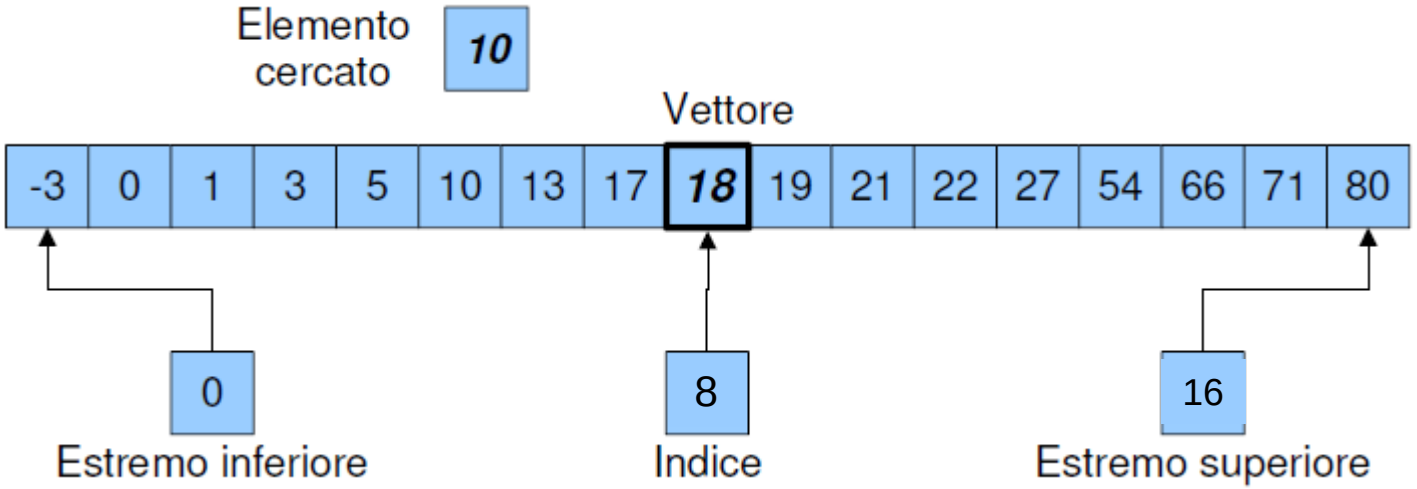
Ricerca binaria

- Rispetto all'algoritmo sequenziale, la ricerca binaria usa due ulteriori indici per individuare gli estremi della porzione del vettore non ancora esclusa



Ricerca binaria

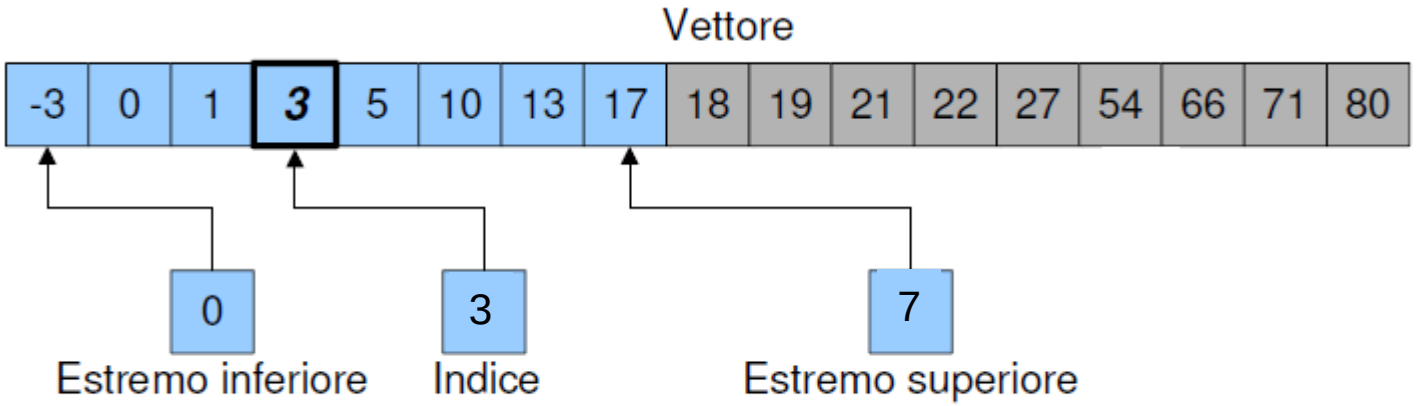
- L'indice iniziale è sempre a metà tra gli indici degli estremi



- L'elemento cercato (10) è minore di 18, per cui si esclude la seconda metà del vettore

Ricerca binaria

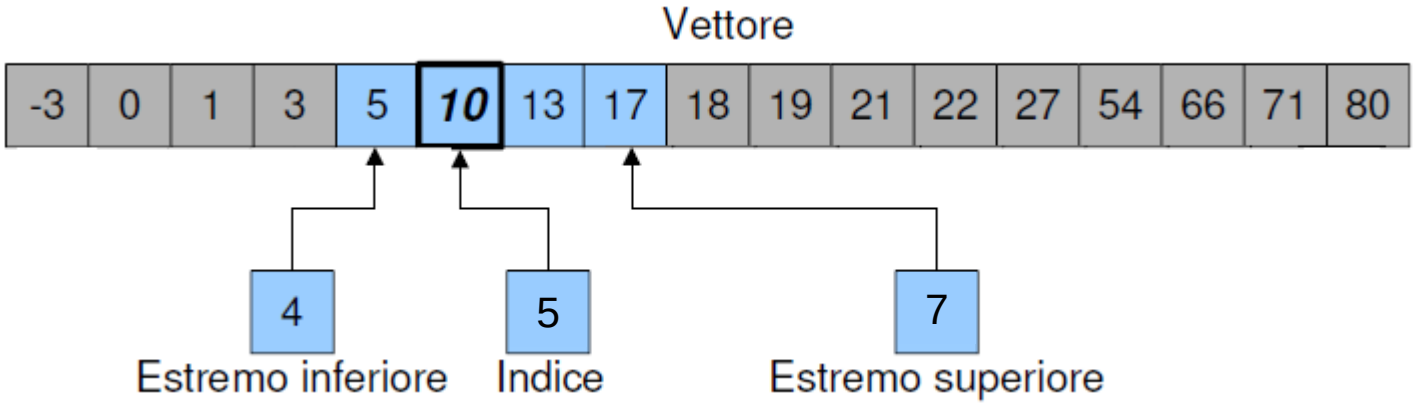
- La metà tra 0 e 7 è 3,5: poiché gli indici devono essere interi, si sceglie tra 3 e 4



- Questa volta l'elemento centrale è precedente a quello cercato, per cui si prosegue nella ricerca escludendo la metà di sinistra

Ricerca binaria

- Come nel caso precedente, si tronca il valore dell'indice all'intero inferiore



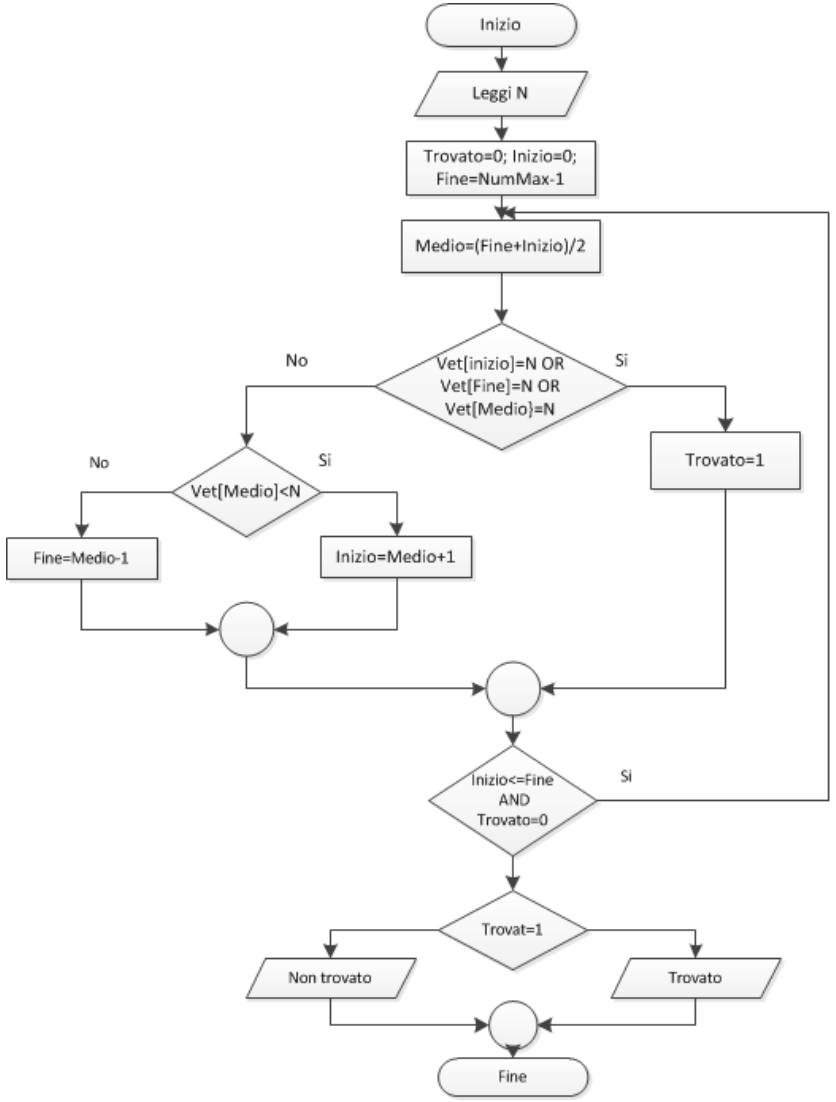
- L'elemento cercato, viene quindi trovato. La risposta fornita dall'algoritmo è **5, cioè la** posizione corrispondente al valore cercato

Ricerca binaria

- L'algoritmo di ricerca binaria dimezza la dimensione dello *spazio di ricerca ad ogni* passo Il tempo necessario all'esecuzione dell'algoritmo è dunque proporzionale al logaritmo di N
- Visto che *logN cresce più lentamente di N,* la ricerca binaria è più efficiente di quella sequenziale (ma richiede l'ipotesi supplementare di ordinamento dei dati)
- Nel caso peggiore l'algoritmo termina quando la dimensione dello spazio di ricerca diventa 1

Flowchart e Codifica in C

```
#include “stdio.h”
#include “conio.h”
#define NumMax 20
Void Main(){
Int vet[NumMax],i,N,Inizio,Fine,Medio;
Printf(“inserisci il numero da cercare”);
Scanf(“%d”,&N);
Char Trovato=0;
Inizio=0;
Do{
Medio=(Fine+Inizio)/2;
If(vet[Inizio]==N||Vet[Medio]==N||Vet[Fine]==N)
   Trovato=1;
else
}
If(Vet[medio<=N)
   Fine=Medio+1;
Else
Inizio=Medio-1;
}while(Inizio<=Fine&&Trovato==0);
If(Trovato==1)
   Printf(“trovato il numero %d”,N);
Else
 Printf(“non trovato il numero %d”,N);
}
}
```



Ricerca Completa

La ricerca completa cerca il numero di occorrenze di un dato elemento in un vettore: ad esempio il numero di 4 nel vettore seguente

3	4	3	5	6	4	7	4	8	6	9

Una ricerca completa quindi, deve richiedere l'elemento da cercare

e contare il numero di occorrenze trovate

```
#include <stdio.h>
int main(){
  int voti[10]={3,4,3,5,6,4,7,4,8,6,9};
  int i,voto,numvoto=0;
  printf("inserisci il voto da cercare");
  scanf("%d",&voto);
  for(i=0;i<n;i++)
  {
    if(v[i]==voto)
       numvoto++;
  }
  printf("il numero di occorrenze di %d e' %d",voto,numvoto);
  return 0;
}
```

Ricerca del minimo

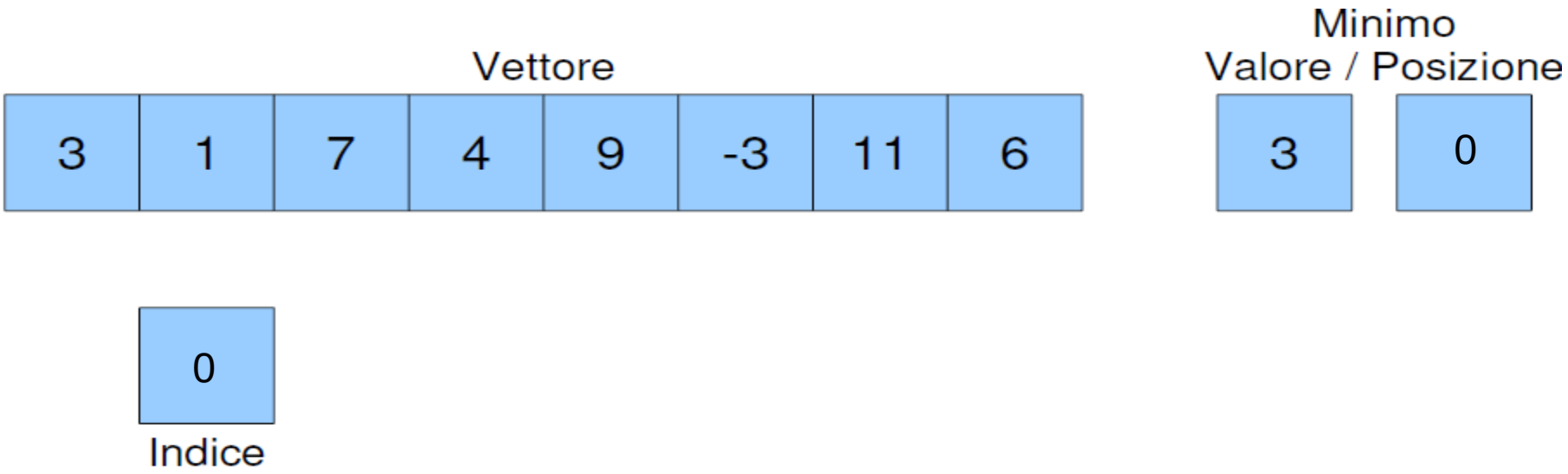
- Un secondo problema riguarda la ricerca del valore minimo (o massimo) all'interno di un vettore non ordinato
  - Naturalmente il problema è banale se il vettore è ordinato!
- Dato che i problemi di ricerca del minimo e di ricerca del massimo sono del tutto analoghi, di seguito si farà riferimento esclusivamente alla ricerca del minimo

Ricerca del minimo

- Per risolvere il problema vengono utilizzate due variabili di supporto, contenenti:
  - il valore minimo trovato sinora
  - la posizione (indice) di tale valore
- L'algoritmo scorre l'intero vettore e confronta ciascun elemento col minimo contenuto nella variabile di supporto
  - Se l'elemento nel vettore è inferiore a quello nella variabile di appoggio, allora sostituisce la variabile di supporto con l'elemento considerato

Ricerca del minimo

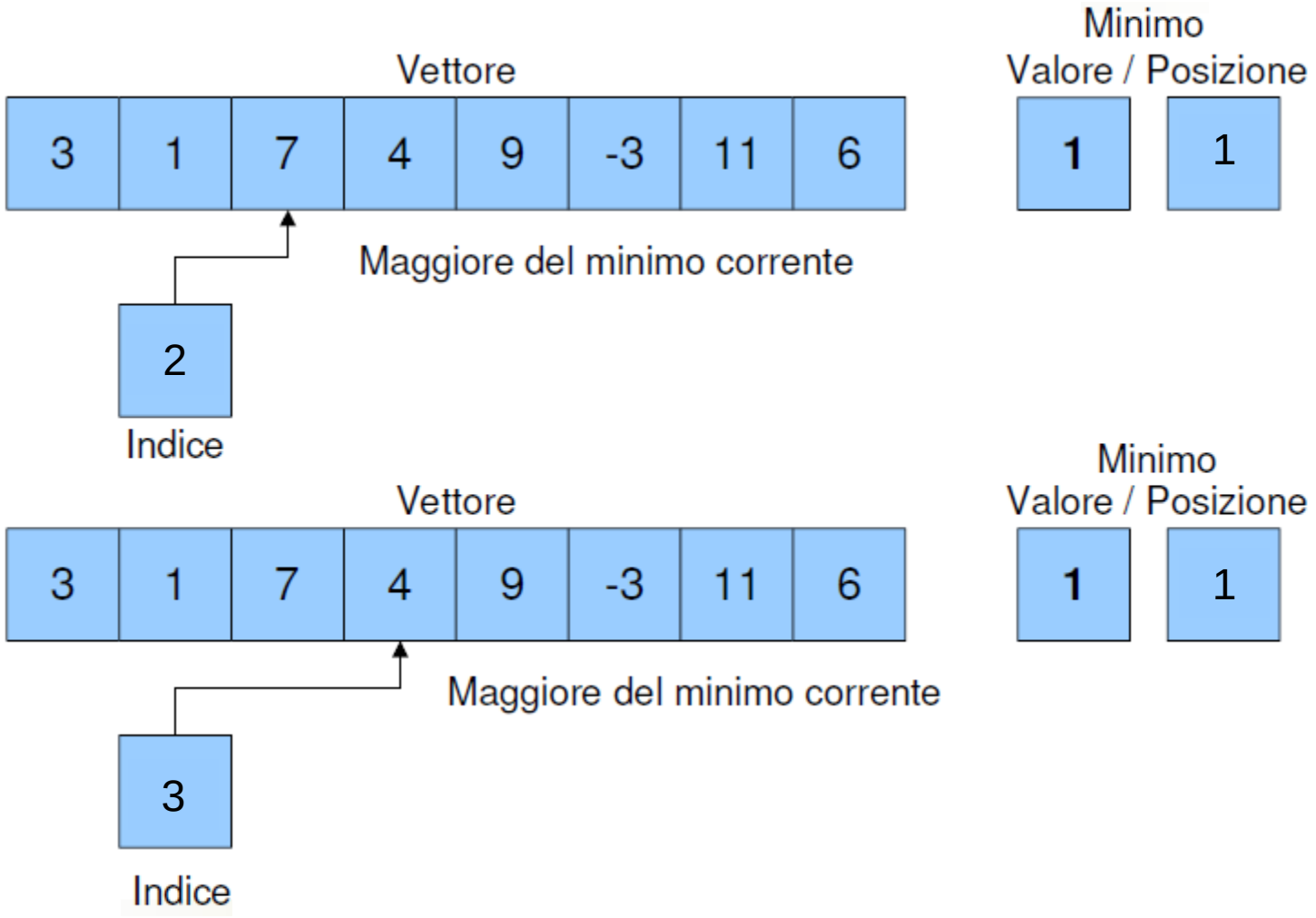
- Si inizializzano le variabili di supporto con la posizione e il valore del primo elemento



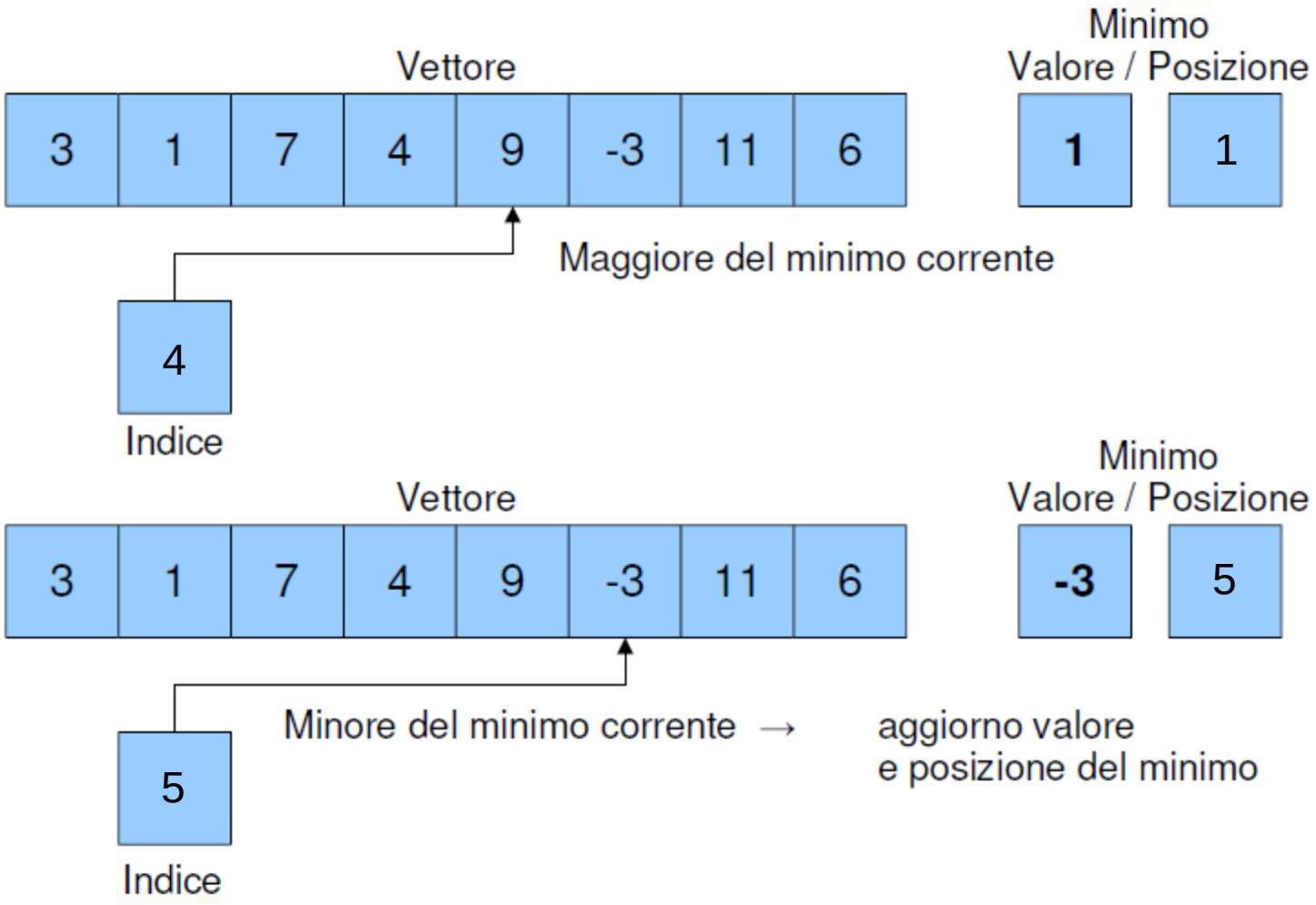
- In questo modo il minimo temporaneo è non inferiore al minimo del vettore

Ricerca del minimo

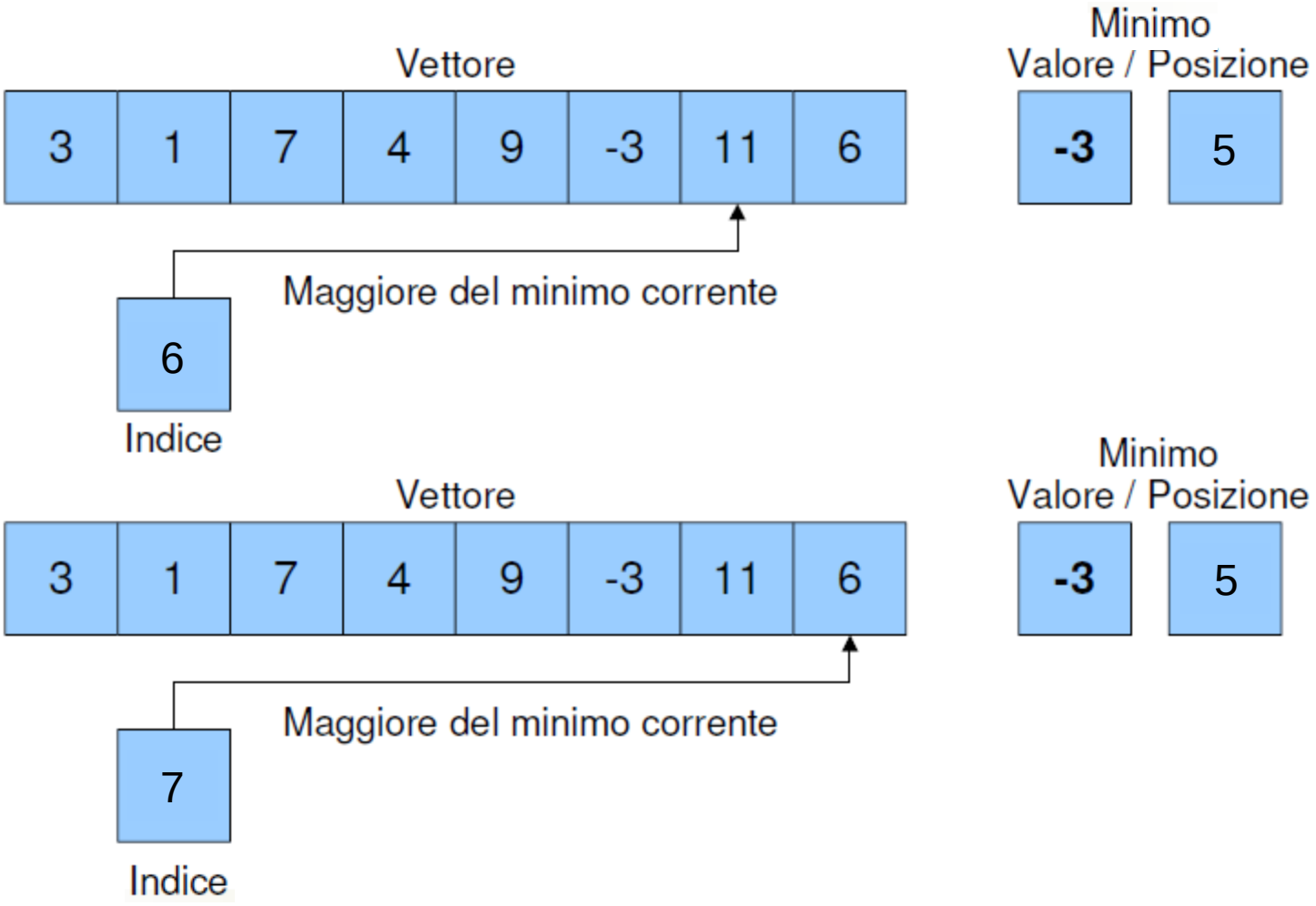
Ricerca del minimo



Ricerca del minimo



Ricerca del minimo



Ricerca del minimo

- Al termine dell'algoritmo le variabili di supporto contengono il valore e la posizione del minimo
  - Se fossero presenti più minimi uguali sarebbe possibile decidere quale tenere in considerazione
- Data la lunghezza N del vettore, è necessario effettuare N confronti
  - Il tempo necessario al completamento dell'esecuzione è proporzionale alla dimensione del vettore

Flowchart e Codifica in C

```
#include “stdio.h”
#include “conio.h”
#define NumMax 20
Void Main(){
Int vet[NumMax],i,min,pos;
min=vet[0];
pos=0;
for(i=0; i<NumMax;i++)
  if(vet[i]<min){
      min=vet[i];
      pos=i;
    }
printf(“il minimo è %d e si trova in posizione %d”, min, pos);
}
```