5.4 Ricerca binaria

Quando l'array risulta ordinato la ricerca di un valore al suo interno può avvenire mediante criteri particolari, uno dei quali è la ricerca detta *binaria* o *dicotomica*.

```
/* Ricerca binaria */

#include <stdio.h>

main()
{
```

```
char vet[6];           /* array contenente i caratteri immessi */
int i,n,k,p;
char aux;              /* variabile di appoggio per lo scambio */
char ele;              /* elemento da ricercare */
int basso, alto, pos;  /* usati per la ricerca binaria */

/* Immissione caratteri */
n = 6;
for(i=0;i<=n-1; i++) {
  printf("vet %d° elemento: ", i+1);
  scanf("%1s", &vet[i]);
}

/* ordinamento ottimizzato */
p = n;
do {
  k = 0;
  for(i=0; i<n-1; i++) {
    if(vet[i]>vet[i+1]) {
      aux = vet[i]; vet[i] = vet[i+1]; vet[i+1] = aux;
      k = 1; p = i+1;
    }
  }
n = p;
}
while(k==1);

printf("\nElemento da ricercare: ");
scanf("%1s", &ele);

/* ricerca binaria */
n = 6;
alto = 0; basso = n-1; pos = -1;
do {
  i = (alto+basso)/2;
  if(vet[i]==ele)  pos = i;
  else
    if(vet[i]<ele)
      alto = i+1;
    else
      basso = i-1;
}
while(alto<=basso && pos==-1);

if(pos != -1)
  printf("\nElemento %c presente in posizione %d\n",ele,pos);
else
  printf("\nElemento non presente! %d\n", pos);
}
```

Listato 5.2 Programma completo di immissione, ordinamento e ricerca

Nel Listato 5.2, dopo l'immissione dei valori del vettore, il loro ordinamento con bubblesort e l'accettazione dell'elemento da cercare, abbiamo i comandi della ricerca binaria vera e propria:

```
/* ricerca binaria */
     alto = 0; basso = n-1; pos = -1;
     do {
```

```
    i = (alto+basso)/2;
    if(vet[i]==ele) pos=i;
    else
      if(vet[i]<ele)
        alto = i+1;
      else
        basso = i-1;
  }
  while(alto<=basso && pos==-1);
```

Si confronta il valore da ricercare, che è memorizzato nella variabile `ele`, con l'elemento intermedio dell'array. L'indice `i` di tale elemento lo si calcola sommando l'indice inferiore dell'array (`0`), memorizzato nella variabile `alto`, con quello superiore (`n-1`), memorizzato nella variabile `basso`, e dividendolo per due. Essendo l'array ordinato si possono presentare tre casi:

1. 1. `vet[i]` è uguale a `ele`, la ricerca è finita positivamente, si memorizza l'indice dell'elemento in `pos` e il ciclo di ricerca ha termine;

2. 2. `vet[i]` è minore di `ele`, la ricerca continua tra i valori maggiori di `vet[i]` che sono memorizzati negli elementi con indice compreso tra `i+1` e `basso`, per cui si assegna ad `alto` il valore `i+1`. Se non si sono già esaminati tutti gli elementi del vettore (`alto` non è minore o uguale a `basso`) la ricerca continua assegnando ancora una volta a `i` il valore `(basso+alto)/2`;

3. 3. `vet[i]` è maggiore di `ele`, la ricerca continua tra i valori minori di `vet[i]` che sono memorizzati negli elementi con indice compreso tra `alto` e `i-1`, per cui si assegna a `basso` il valore `i-1`. Se non si sono già esaminati tutti gli elementi del vettore (`alto` non è minore di `basso`) la ricerca continua assegnando ancora una volta a `i` il valore `(basso+alto)/2`.

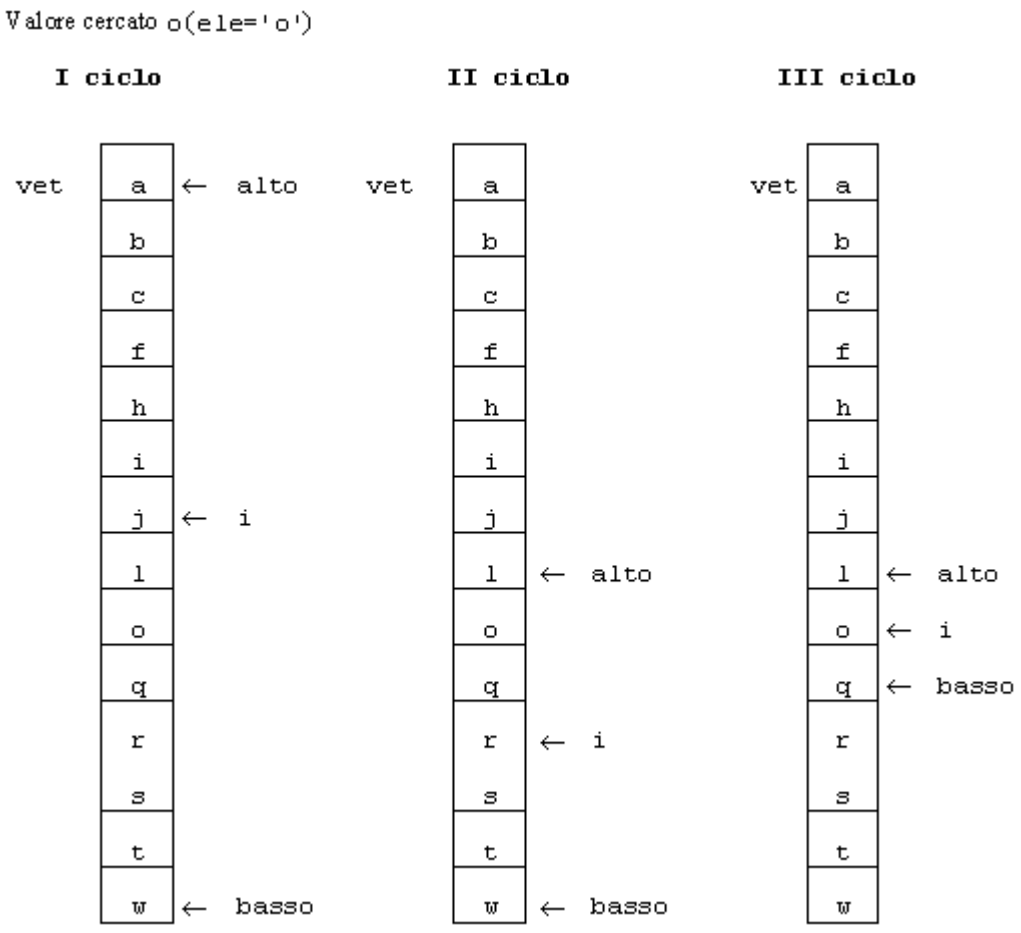

Figura 5.3 Esempio di ricerca binaria di `ele='o'`.

Nella Figura 5.3 si osserva il mutare dei valori di `alto`, `basso` e `i` fino al reperimento del valore desiderato ("`o`"). Il numero di cicli e corrispondenti confronti effettuati è risultato uguale a tre ▬, mentre se avessimo utilizzato la ricerca sequenziale avremmo avuto nove iterazioni. La ricerca sequenziale esegue nel caso più fortunato – quello in cui l'elemento cercato è proprio il primo – un unico confronto; nel caso più sfortunato – quello in cui l'elemento cercato è invece l'ultimo – esegue n confronti. Si ha quindi che la ricerca sequenziale effettua in media $(n+1)/2$ confronti.
La ricerca binaria offre delle prestazioni indubbiamente migliori: al massimo esegue un numero di confronti pari al logaritmo in base due di n. Questo implica che nel caso in cui n sia uguale a 1000 per la ricerca sequenziale si hanno in media 500 confronti, per quella binaria al massimo 10. Poiché, come per l'ordinamento, il tempo impiegato per eseguire il programma è direttamente proporzionale al numero dei confronti effettuati, è chiaro come la ricerca binaria abbia tempi di risposta mediamente molto migliori della ricerca sequenziale ▬.
Osserviamo, tuttavia, che mentre si può effettuare la ricerca sequenziale su qualsiasi vettore, per la ricerca binaria è necessario disporre di un vettore ordinato ▬, così che non sempre risulta possibile applicare tale algoritmo ▬.