5.5 Fusione

Un altro algoritmo interessante è quello che partendo da due array monodimensionali ordinati ne ricava un terzo, anch'esso ordinato. I due array possono essere di lunghezza qualsiasi e in generale non uguale. Il programma del Listato 5.3 richiede all'utente l'immissione della lunghezza di ognuna delle due sequenze e gli elementi che le compongono. Successivamente ordina le sequenze ed effettua la fusione (*merge*) di una nell'altra, memorizzando il risultato in un array a parte.

```
/* Fusione di due sequenze ordinate */

#include <stdio.h>
#define MAX_ELE 1000

main()
{
char vet1[MAX_ELE];      /* prima sequenza */
char vet2[MAX_ELE];      /* seconda sequenza */
char vet3[MAX_ELE*2];    /* merge */

int n;                   /* lunghezza prima sequenza */
int m;                   /* lunghezza seconda sequenza */

char aux;               /* variabile di appoggio per lo scambio */

int i, j, k, p, n1, m1;

do {
  printf("Lunghezza prima sequenza: ");
  scanf("%d", &n);
}
while(n<1 || n>MAX_ELE);

/* caricamento prima sequenza */
for(i = 0;i <= n-1; i++) {
  printf("vet1 %d° elemento: ",i+1);
  scanf("%1s", &vet1[i]);
}

do {
  printf("Lunghezza seconda sequenza: ");
  scanf("%d", &m);
}
while(m<1 || m>MAX_ELE);
```

```
/* caricamento seconda sequenza */
for(i=0; i<=m-1; i++) {
  printf("vet2 %d° elemento: ",i+1);
  scanf("%1s", &vet2[i]);
}

/* ordinamento prima sequenza */
p = n; n1 = n;
do {
  k = 0;
  for(i = 0; i < n1-1; i++) {
    if(vet1[i]> vet1[i+1]) {
      aux = vet1[i]; vet1[i] = vet1[i+1]; vet1[i+1] = aux;
      k = 1; p = i+1;
    }
  }
n1 = p;
}
while(k==1);

/* ordinamento seconda sequenza */
p = m; m1 = m;
do {
  k = 0;
  for(i=0; i<m1 - 1; i++) {
    if(vet2[i]>vet2[i+1]) {
      aux = vet2[i]; vet2[i] = vet2[i+1]; vet2[i+1] = aux;
      k = 1; p = i+1;
    }
  }
m1 = p;
}
while(k==1);

/* fusione delle due sequenze (merge) */
i = 0; j = 0; k = 0;
do {
  if(vet1[i]<=vet2[j])
    vet3[k++] = vet1[i++];
  else
    vet3[k++] = vet2[j++];
}
while(i<n && j<m);

if(i<n)
  for(; i<n; vet3[k++] = vet1[i++])
    ;
else
  for(; j<m; vet3[k++] = vet2[j++])
    ;

/* visualizzazione della fusione */
for(i=0; i<k; i++)
  printf("\n%c", vet3[i]);
}
```

Listato 5.3 Fusione di due array

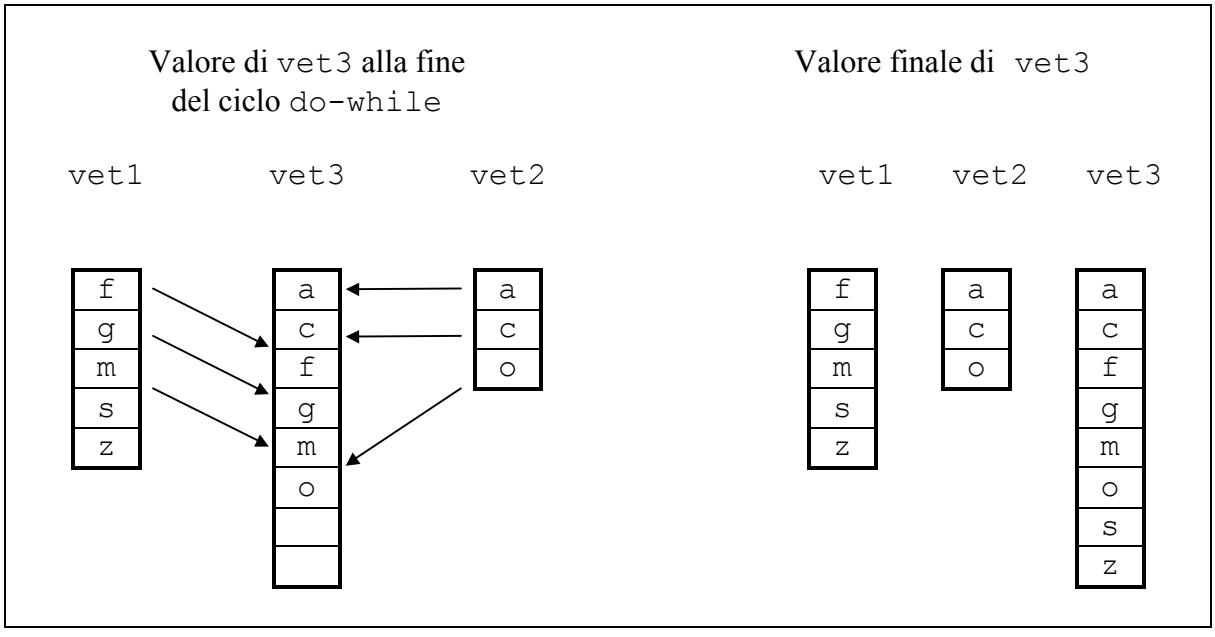

Figura 5.4 Risultato parziale e finale della fusione tra due vettori

In Figura 5.4 osserviamo il merge tra gli array ordinati `vet1` e `vet2` ordinati. L'operazione viene effettuata in due parti. La prima è data da:

```
i = 0; j = 0; k = 0;
     do {
       if(vet1[i]<=vet2[j])
         vet3[k++] = vet1[i++];
       else
         vet3[k++] = vet2[j++];
     }
     while(i<n && j<m);
```

Si controlla se l'*i*-esimo elemento di `vet1` è minore o uguale al *j*-esimo elemento di `vet2`, nel qual caso si aggiunge `vet1[i]` a `vet3` e si incrementa `i`. Nel caso contrario si aggiunge a `vet3` l'array `vet2[j]` e si incrementa `j`. In ogni caso si incrementa `k`, la variabile che indicizza `vet3`, perché si è aggiunto un elemento a `vet3`. Dal ciclo si esce quando `i` ha valore `n-1` o `j` ha valore `m-1`.

Si devono ancora aggiungere a `vet3` gli elementi di `vet1` (`j=m-1`) o di `vet2` (`i=n-1`) che non sono stati considerati. Nell'esempio precedente in `vet3` non ci sarebbero `s` e `z`. La seconda parte del merge ha proprio questo compito:

```
if(i<n)
       for(; i<n; vet3[k++] = vet1[i++])
         ;
     else
       for(; j<m; vet3[k++] = vet2[j++])
         ;
```