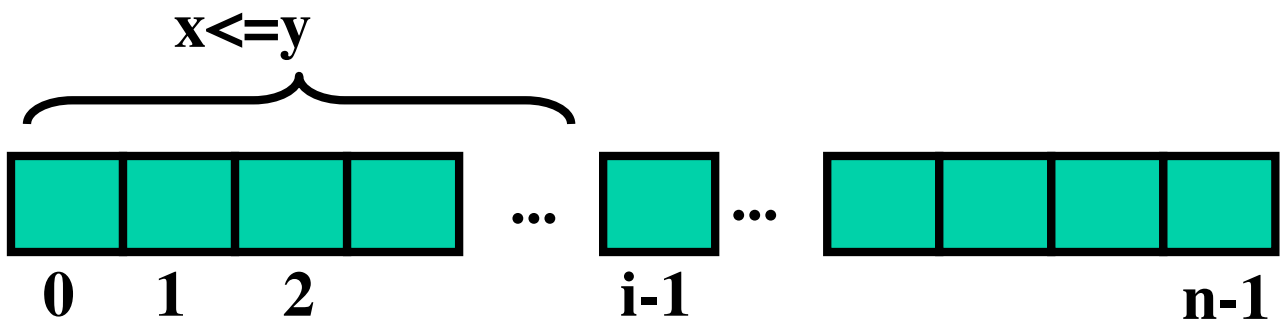
Ordinamento per inserimento (Insertion Sort)

Quello usato per ordinare le carte che un giocatore ha in mano:
ogni elemento viene inserito nel posto giusto tra quelli già considerati e ordinati

Per ordinare un vettore di interi a[0…n] in ordine crescente, partiamo con il primo elemento come un sottovettore ordinato e inseriamo gli elementi a[1],…,a[n] nell'ordine nel posto giusto tra quelli già considerati e ordinati:

```
for (i=1;i<n;i++)
/*invariante: a[0,…i-1] è già ordinato
   Obiettivo: inserire i, nel posto giusto tra A[0],…,A[i] */
```



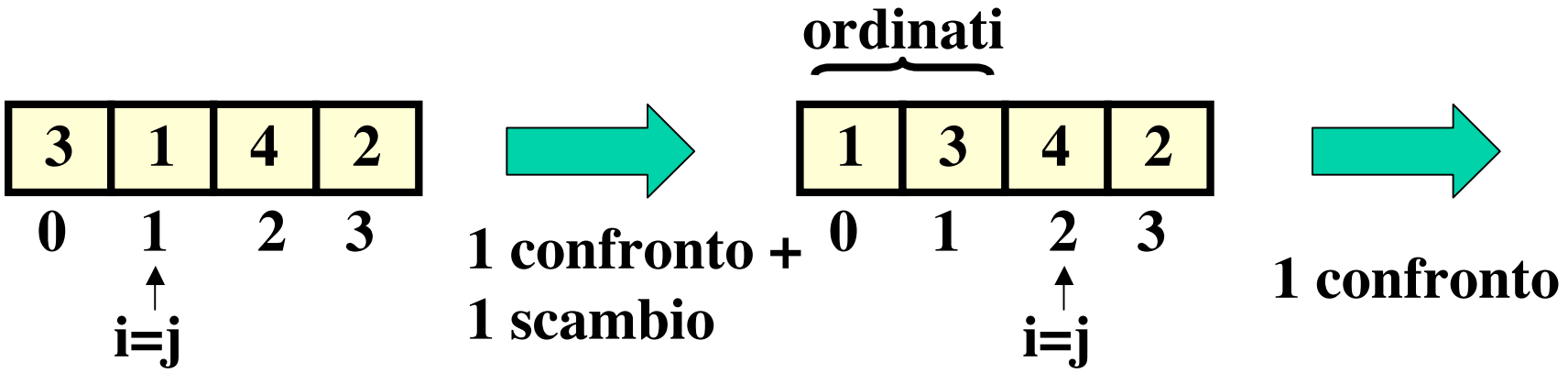
L'invariante di un ciclo è un'asserzione che è vera all'inizio di ogni esecuzione del ciclo

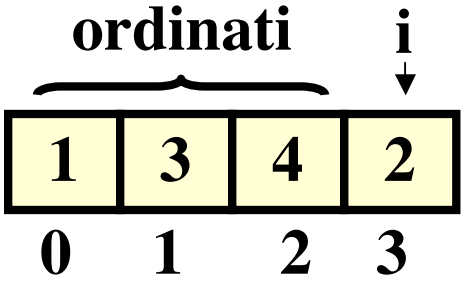
```
for (i = 1; i < n; i++)
/*invariante: A[0,…i-1] è già ordinato*/
   for (j = i; j > 0; j--)
      if (a[j] < a[j-1])  scambia(&a[j-1],&a[j]);
```

Si può migliorare?

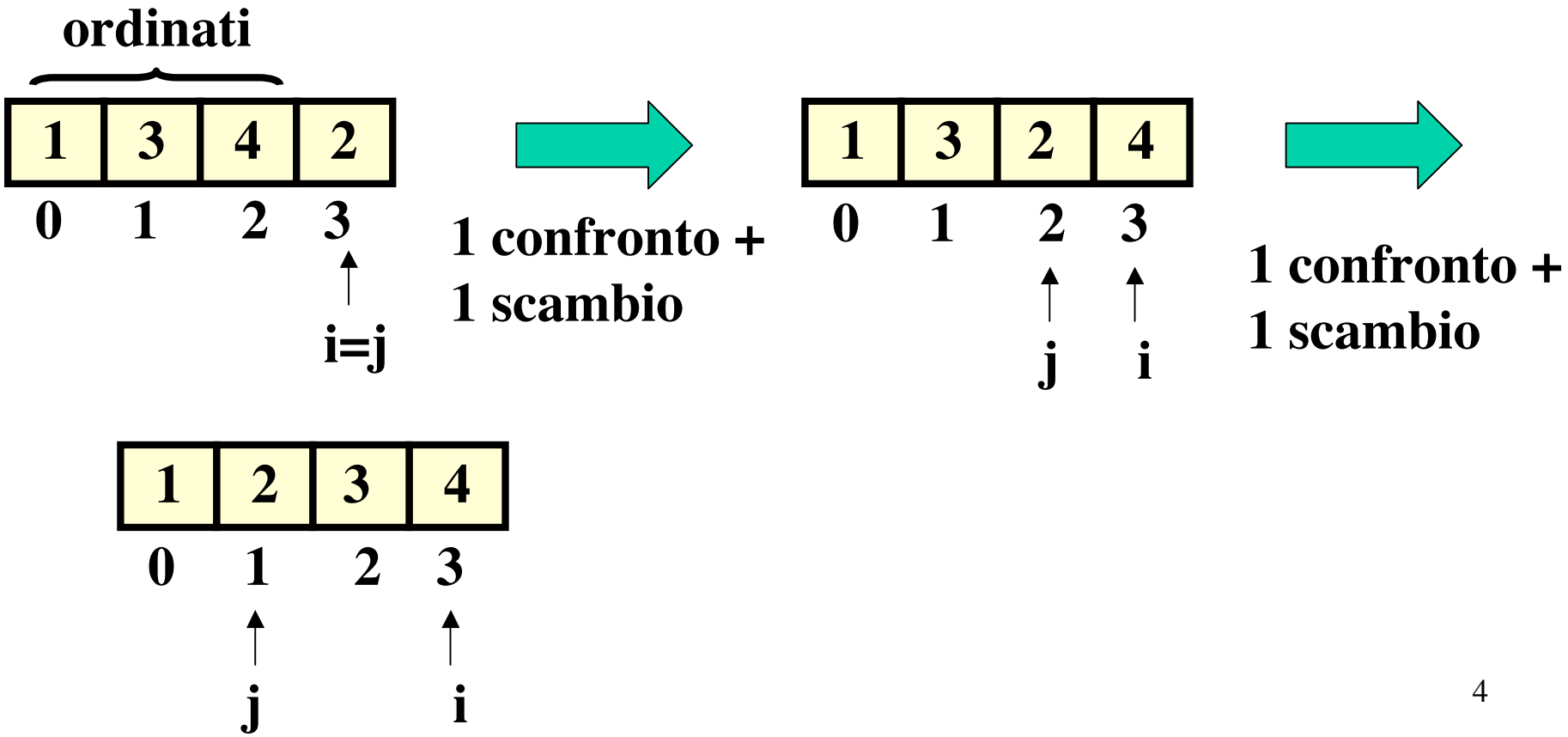
La precedente implementazione non teneva conto dell'invariante

```
for (i = 1; i < n; i++)
/*invariante: A[0,…i-1] è già ordinato*/
  for(j = i; (j > 0)&& (a[j] < a[j-1]) ; j--)
  scambia(&a[j-1],&a[j]);
```



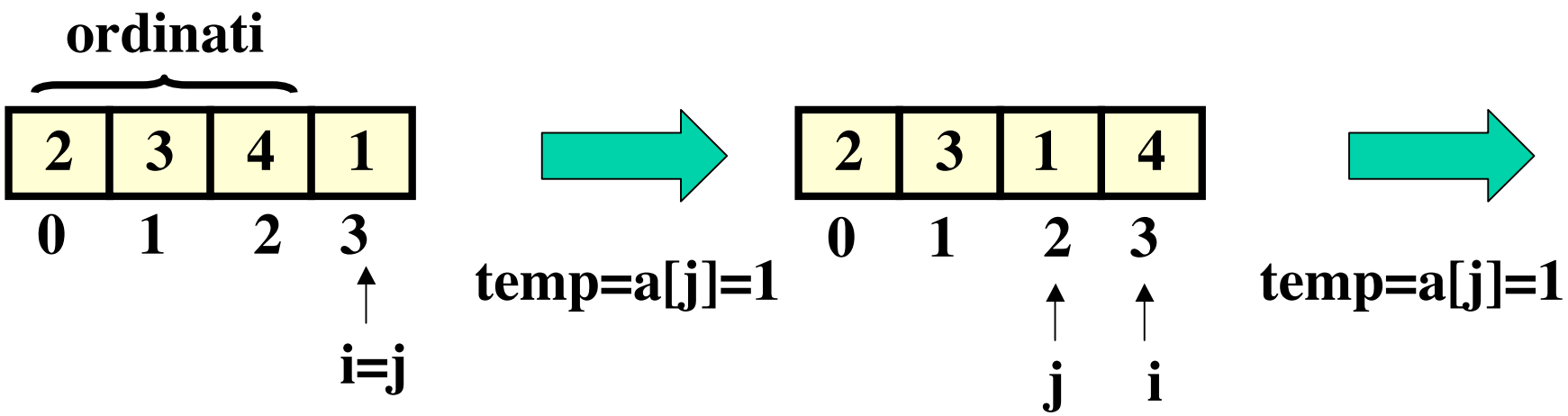


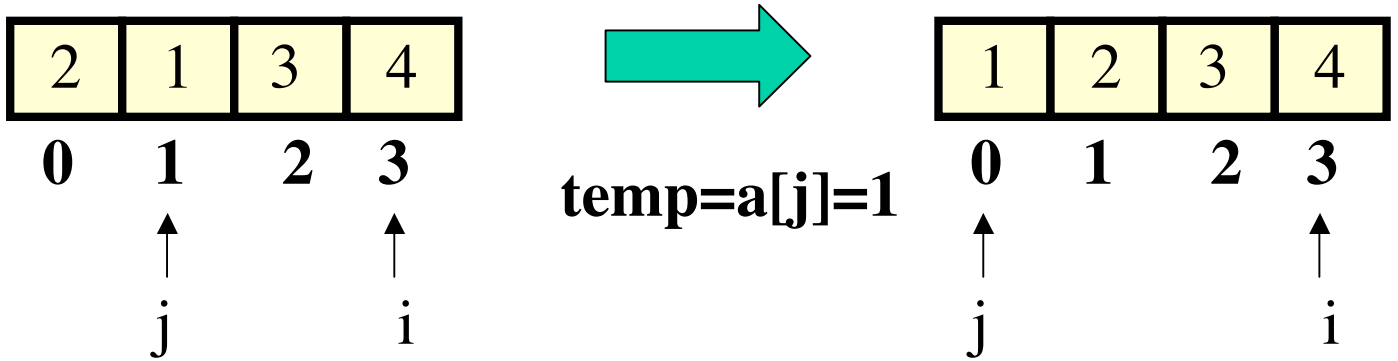
```
for (i = 1; i < n; i++)
/*invariante: A[0,…i-1] è già ordinato*/
  for(j = i; (j > 0)&& (a[j] < a[j-1]) ; j--)
  scambia(&a[j-1],&a[j]);
```



Code tuning: eliminazione chiamata

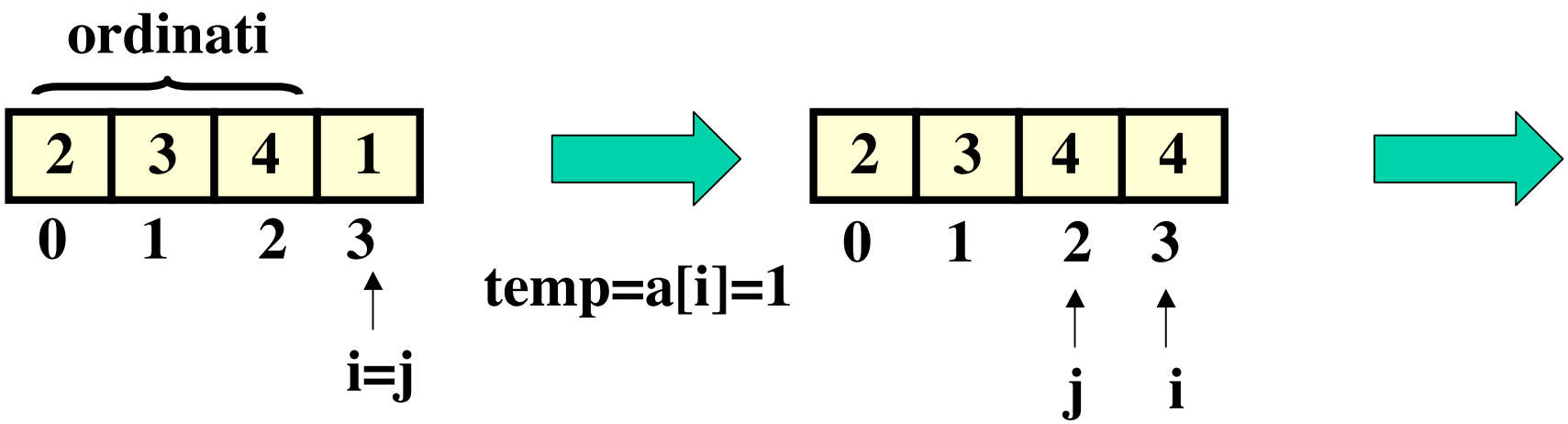
```
 for (i = 1; i < n; i++)
/*invariante: A[0,…i-1] è già ordinato*/
for (j = i; (j > 0) &&(a[j] < a[j-1]) ; j--)
      {temp = a[j];
      a[j] = a[j-1];
      a[j-1] = temp;}
```




2
1
3
4
0
1
2
3
j
i
temp=a[j]=1
1
2
3
4
0
1
2
3
j
i

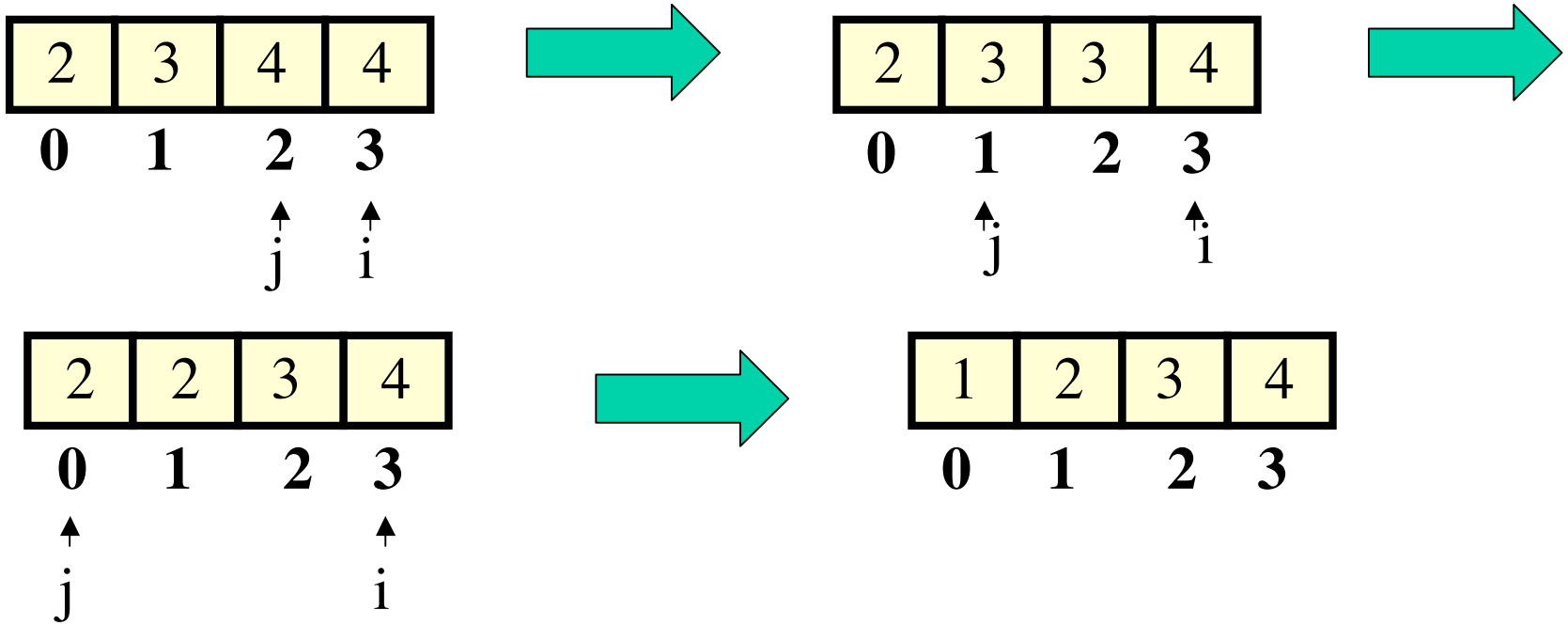
Code tuning: riduzione istruzioni del ciclo

```
 for (i = 1; i < n; i++)
/*invariante: A[0,…i-1] è già ordinato*/
{temp = a[i];
 for (j = i; (j > 0) &&(a[j-1] > temp) ; j--)
        a[j] = a[j-1];
 a[j] = temp;}
```



```
 for (i = 1; i < n; i++)
/*invariante: A[0,…i-1] è già ordinato*/
{temp = a[i];
 for (j = i; (j > 0) &&(a[j-1] > temp) ; j--)
      a[j] = a[j-1];
 a[j] = temp;}
```

temp=1



Abbiamo effettivamente fatto un guadagno apprezzabile?

```
Inserisci il numero di elementi del vettore (< 30000).
29000
Il tempo, in millisecondi, per insSort0 é 12520.0000
Il tempo, in millisecondi, per insSort0 é 10520.0000
Il tempo, in millisecondi, per insSort0 é 4760.0000
Il tempo, in millisecondi, per insSort0 é 2200.0000
```

nome file: confrTempiInsSort.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define NUM 4
#define SIZE 30000

typedef  void (*Ord)();
/*definizione di un tipo puntatore a funzione che  non restituisce un
valore */

 void insSort0(int* a, const int n);
 /* ordina in ordine crescente un vettore di interi
* postc : a[i] < a[i+1], per 0<=i<n.*/
```

nome file: confrTempiInsSort.c

```
void insSort1(int* a, const int n);
 /* ordina in ordine crescente un vettore di interi
* postc : a[i] < a[i+1], per 0<=i<n.*/

void insSort2(int* a, const int n);
 /* ordina in ordine crescente un vettore di interi
* postc : a[i] < a[i+1], per 0<=i<n.*/

 void insSort3(int* a, const int n);
 /* ordina in ordine crescente un vettore di interi
* postc : a[i] < a[i+1], per 0<=i<n.*/

void scambia(int* ,int * );
```

nome file: confrTempiInsSort.c

```
void scambia(int* x,int* y)
{ int temp;
 temp = *x;
 *x = *y;
 *y = temp;
 }

 void insSort0(int* a, const int n)
 {int i, j;
 for(i = 1; i < n; i++)
      for (j = i; j > 0; j--)
          if (a[j] < a[j-1])  scambia(&a[j-1],&a[j]);
 }

 void insSort1(int* a, const int n)
 {int i, j;
 for(i = 1; i < n; i++)
      for(j = i; (j > 0) &&(a[j] < a[j-1]) ; j--)
          scambia(&a[j-1],&a[j]);
}
```

nome file: confrTempiInsSort.c

```
void insSort2(int* a, const int n)
 {int i, j,temp;
 for(i = 1; i < n; i++)
     for(j = i; (j > 0) &&(a[j] < a[j-1]) ; j--)
                {temp = a[j];
                a[j] = a[j-1];
                a[j-1] = temp;}
  }

 void insSort3(int* a, const int n)
 {int i, j,temp;
 for(i = 1; i < n; i++)
  {temp = a[i];
                for(j = i; (j > 0) &&(a[j-1] > temp) ; j--)
                a[j] = a[j-1];
   a[j] = temp;}
}
```

nome file: confrTempiInsSort.c

```
/*definizione della funzione copiaVett */
main()
{/* si confrontano i tempi di esecuzione delle diverse versioni di
InsSort, calcolando i tempi in millisecondi */
int num, i;
int  vett[SIZE],vettApp[SIZE];
clock_t start,end;
double millisec;
Ord vettFun[NUM] = {insSort0,insSort1,insSort2,insSort3};
printf("Inserisci il numero di elementi del vettore (<= %d).\n",SIZE);
scanf("%d",&num);
srand(time(NULL)); /*inizializza la sequenza di numeri pseudocasuali */
for (i = 0;i < num;i++) vett[i] = rand() ; /*vett è un vettore pseudocasuale*/
copiaVett(vettApp,vett,num);
for (i= 0;i < NUM;i++)
            {start = clock();
             vettFun[i](vett,num);
            end = clock();
            millisec = (end-start)/(CLOCKS_PER_SEC /(double) 1000.0);
            printf("i millisecondi necessari per ordinare con InsSort%d sono%.4f\n",
            scelta,millisec);
            copiaVett(vett,vettApp,num); }
 return 0;}
```