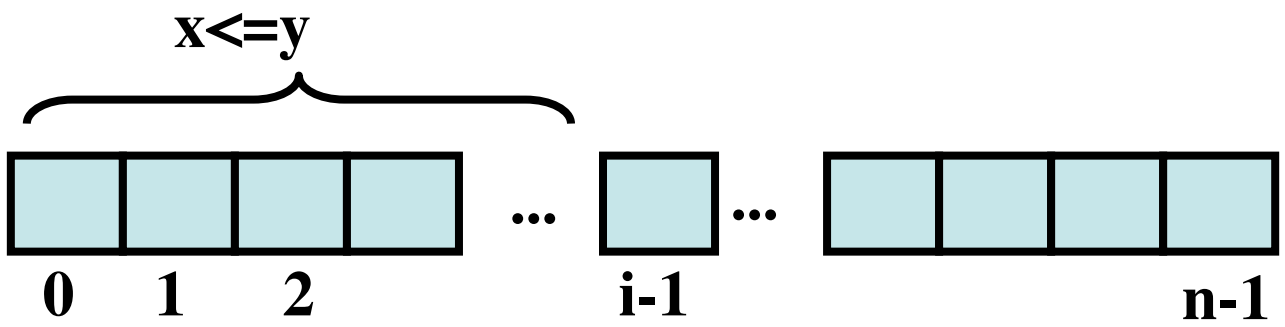
Ordinamento per inserimento (Insertion Sort)

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] */
```



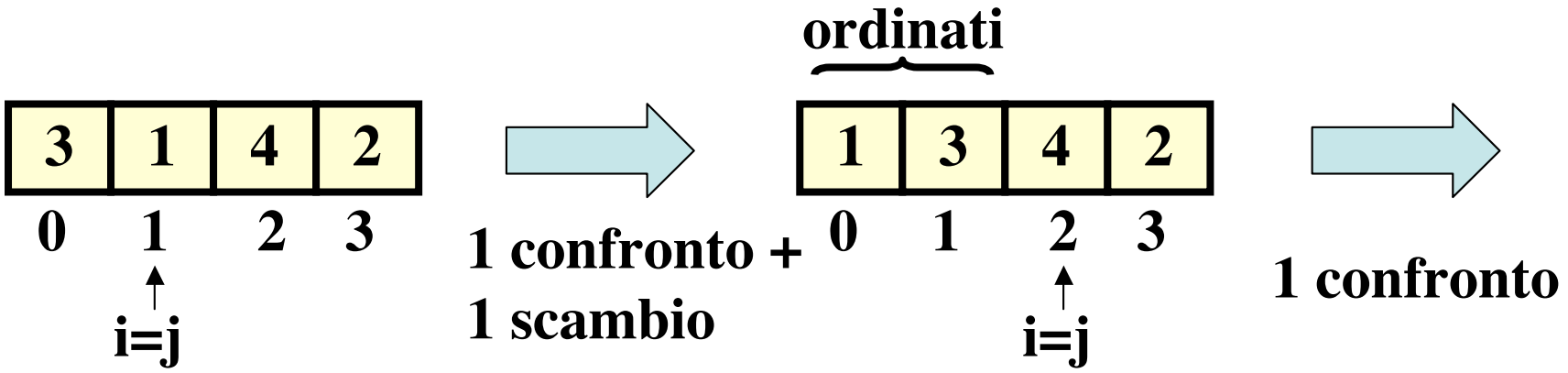
L'invariante di un ciclo è un'asserzione che è vera all'inizio di ogni esecuzione del ciclo

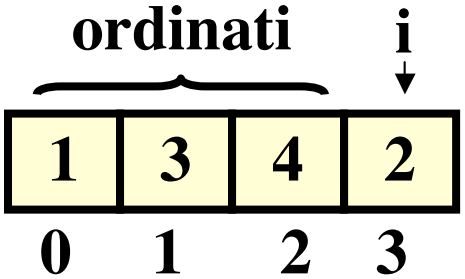
```
for (i = 1; i < n; i++)
/*invariante: A[0,…i-1] è già ordinato*/
   for (j = i; j > 0; j--)
      if (a[j] < a[j-1])  scambia(&a[j-1],&a[j]);
```

Si può migliorare?

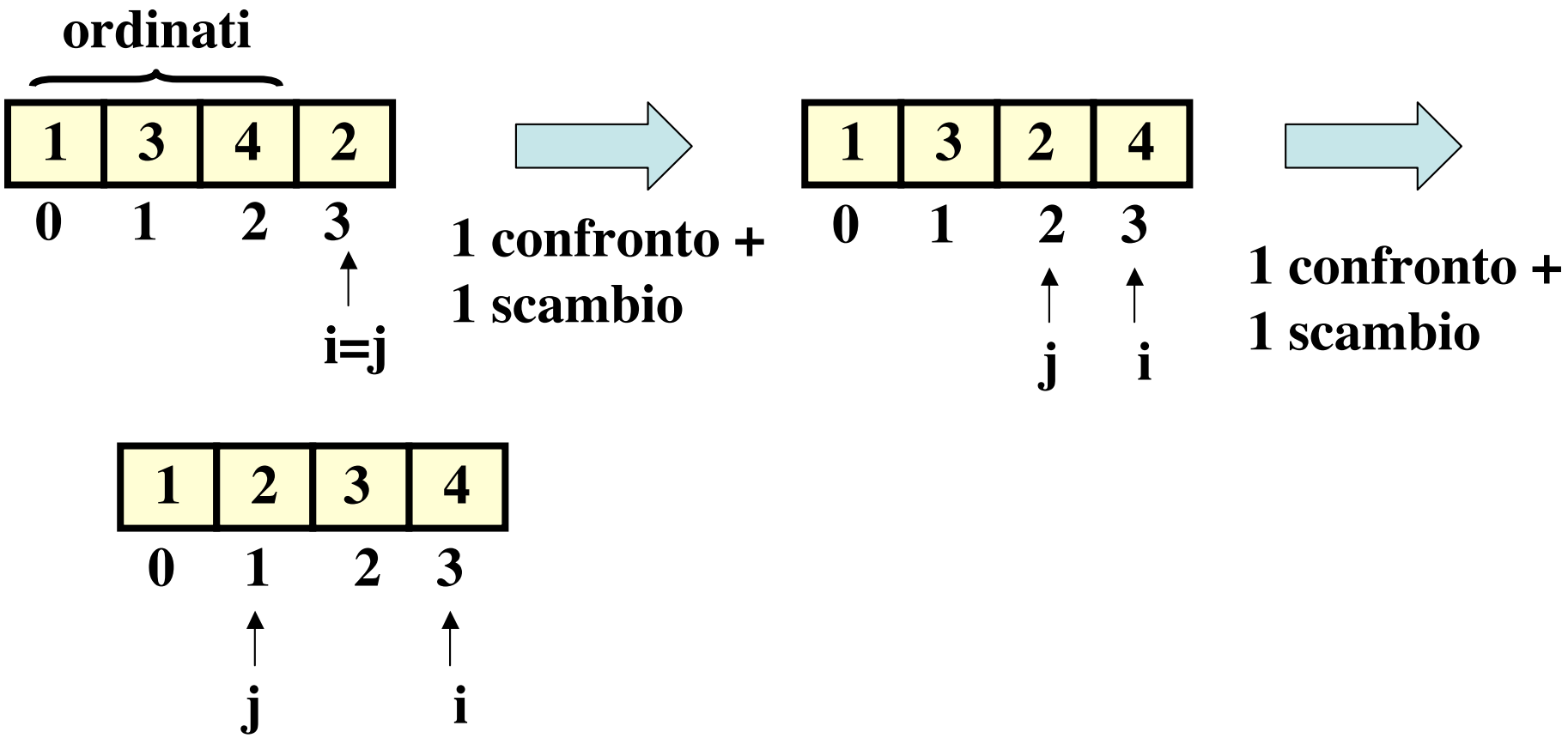
La precedente implementazione non teneva conto dell'invariante

```
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]);
```



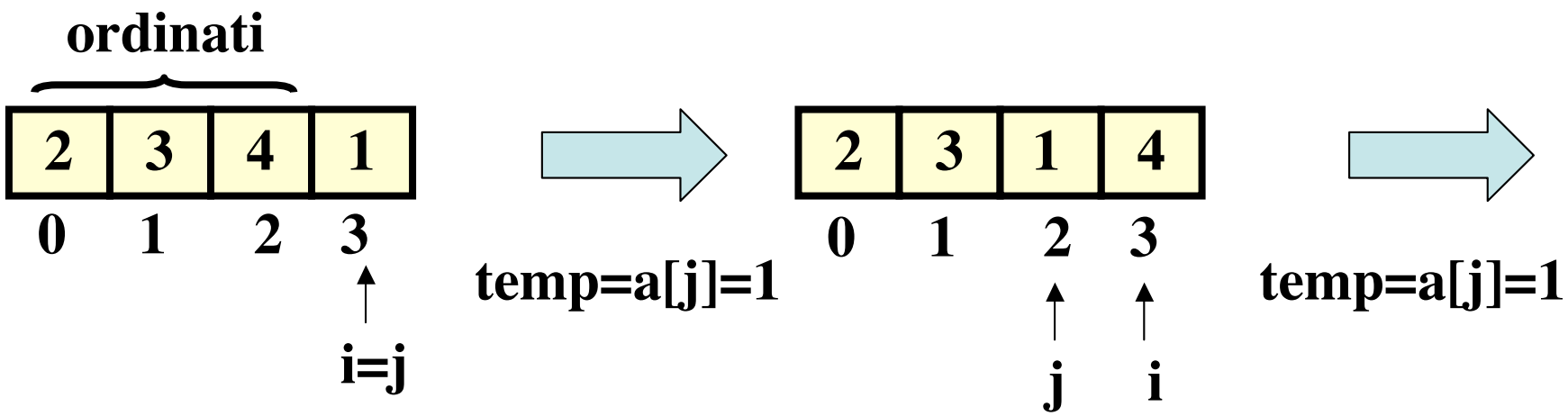


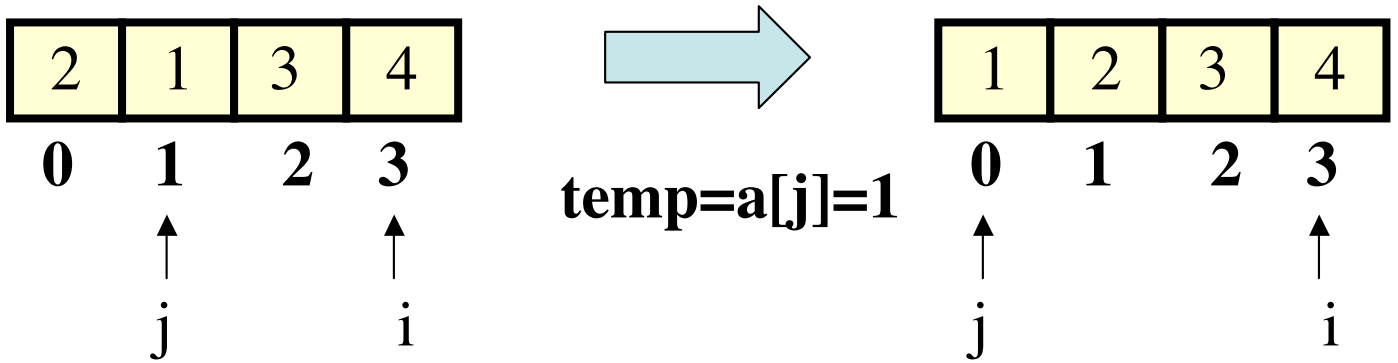
```
for (i = 1; i < n; i++)
/*invariante: A[0,…i-1] è già ordinato*/
  for(j = i; (j > 0)&& (a[j] < a[j-1]) ; j--)
  scambia(&a[j-1],&a[j]);
```



Code tuning: eliminazione chiamata

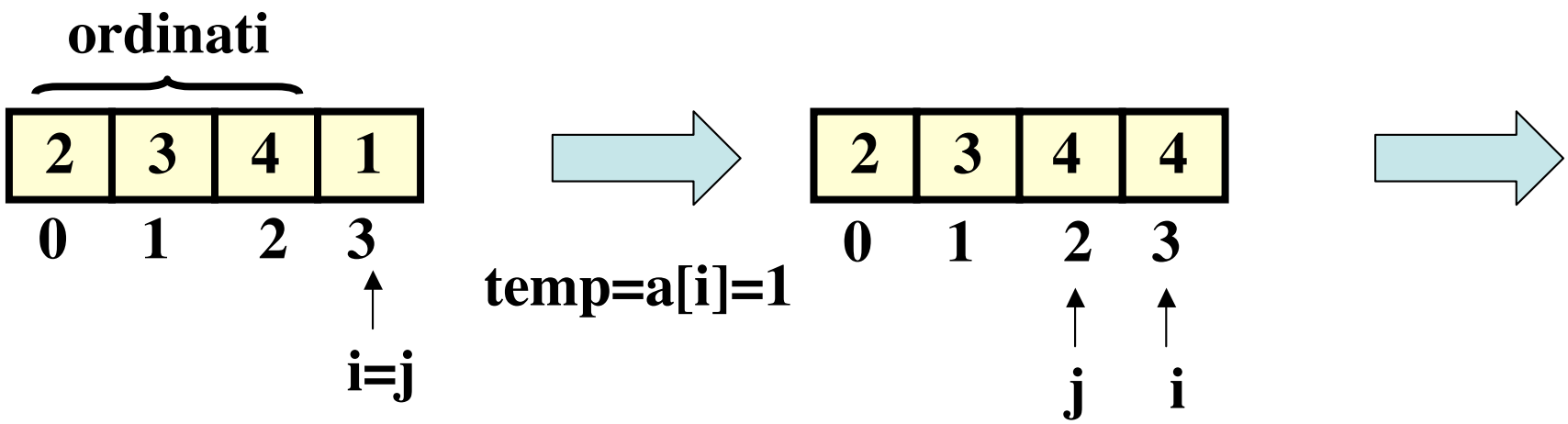
```
 for (i = 1; i < n; i++)
/*invariante: A[0,…i-1] è già ordinato*/
for (j = i; (j > 0) &&(a[j] < a[j-1]) ; j--)
      {temp = a[j];
      a[j] = a[j-1];
      a[j-1] = temp;}
```




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