Titolo unità didattica: Ordinamento di array, parte I [11]

Titolo modulo : Algoritmo di ordinamento per selezione [02-T]

Variante dell'approccio incrementale all'ordinamento

Argomenti trattati:

- ✓ ordinamento per selezione di minimo
- ✓ ordinamento per selezione di massimo
- ✓ analisi dell'efficienza

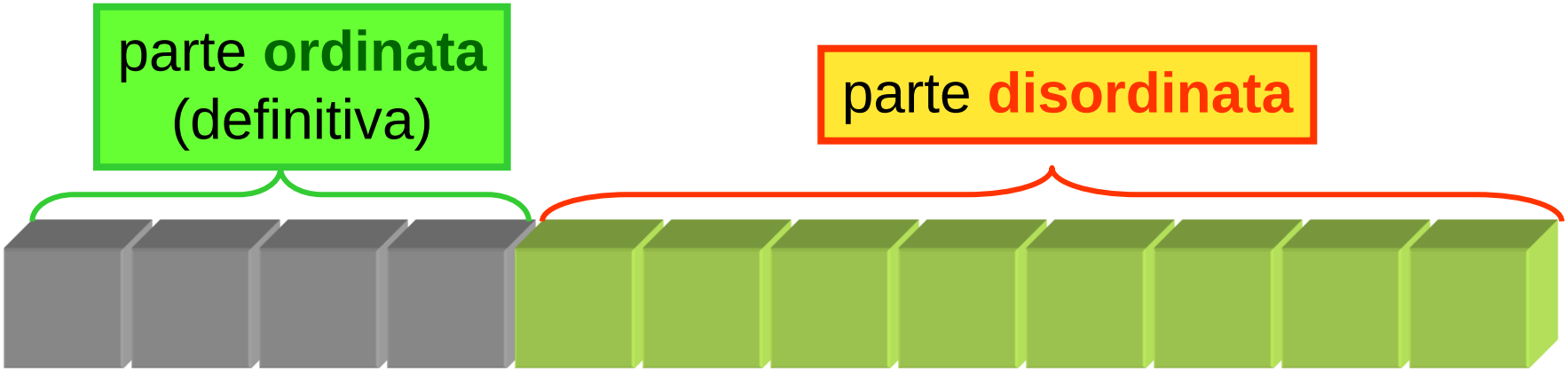
Prerequisiti richiesti: P1-07-02-T

problema: (**sorting**)
ordinamento (senso crescente) dei valori di un array 1D

dati di input: l'array `a` (da ordinare), il size `n`

dato di output: l'array `a` (ordinato) (*in place*)

idea: costruzione incrementale dell'array ordinato **finale**

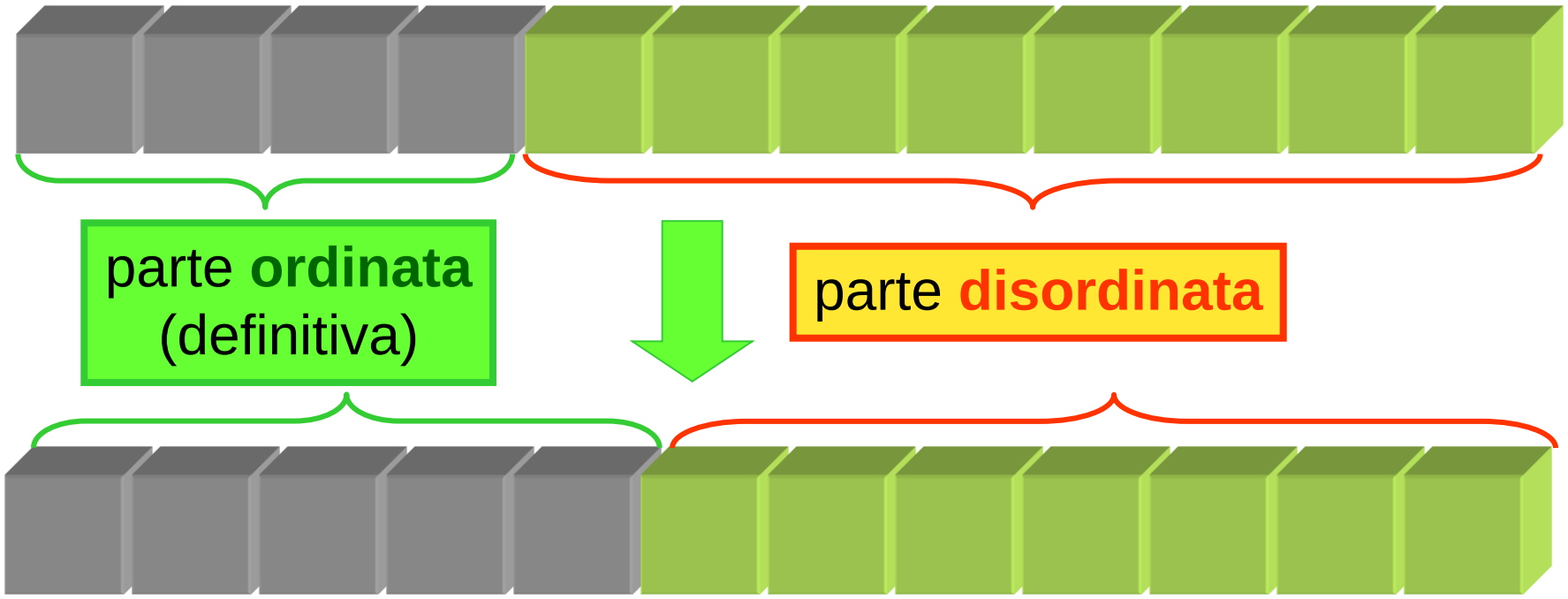


problema:
ordinamento dei valori di un array 1D (**sort**)

dati di input: l'array a (da ordinare), il size n

dato di output: l'array a (ordinato) (*in place*)

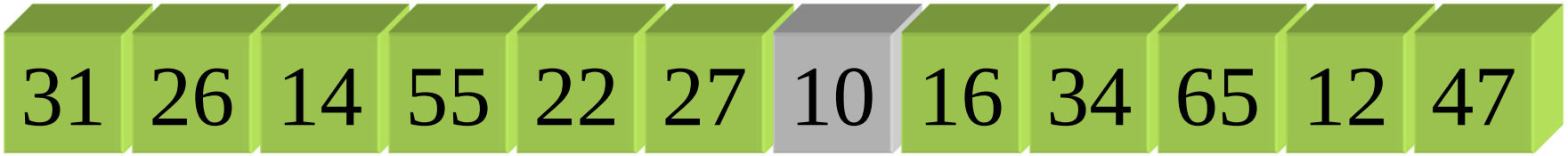
idea: costruzione incrementale dell'array ordinato **finale**



i indice di inizio porzione

n-1 indice di fine porzione

```
for (i=0; i < n-1; i++) {
      determinare l'elemento minino
      della porzione i..n-1 dell'array
      metterlo al primo posto della porzione
}
```



i indice di inizio porzione

n-1 indice di fine porzione

```
for (i=0; i < n-1; i++) {
    determinare l'elemento minino
    della porzione i..n-1 dell'array
    metterlo al primo posto della porzione
}
```

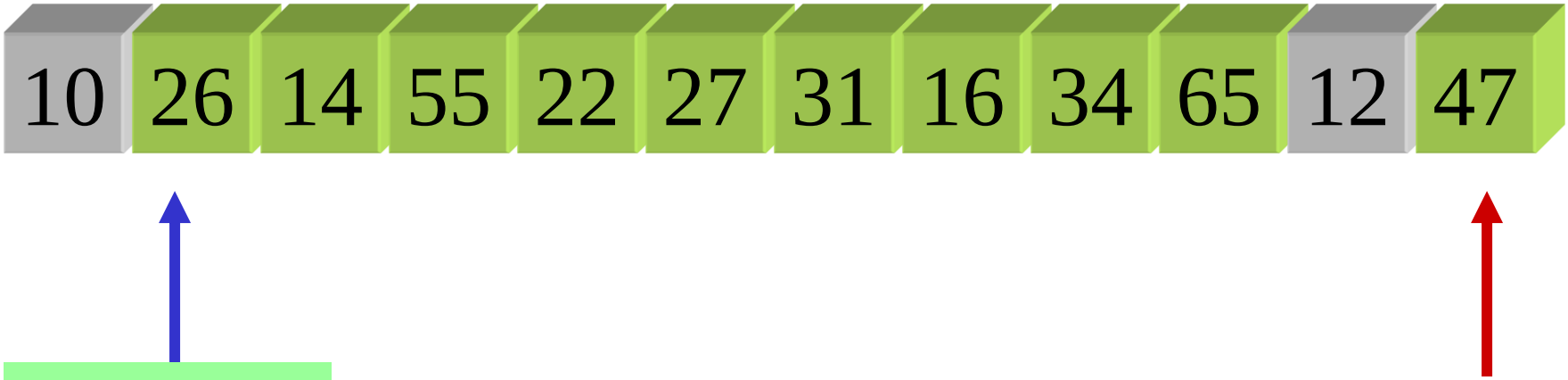
i=0

i indice di inizio porzione

n-1 indice di fine porzione

```
for (i=0; i < n-1; i++) {
      determinare l'elemento minino
      della porzione i..n-1 dell'array
      metterlo al primo posto della porzione
}
```

i=0




```
for (i=0; i < n-1; i++) {
    determinare l'elemento minino
    della porzione i..n-1 dell'array
    metterlo al primo posto della porzione
}
```

i=1

i indice di inizio porzione

n-1 indice di fine porzione

```
for (i=0; i < n-1; i++) {
      determinare l'elemento minino
      della porzione i..n-1 dell'array
      metterlo al primo posto della porzione
}
```

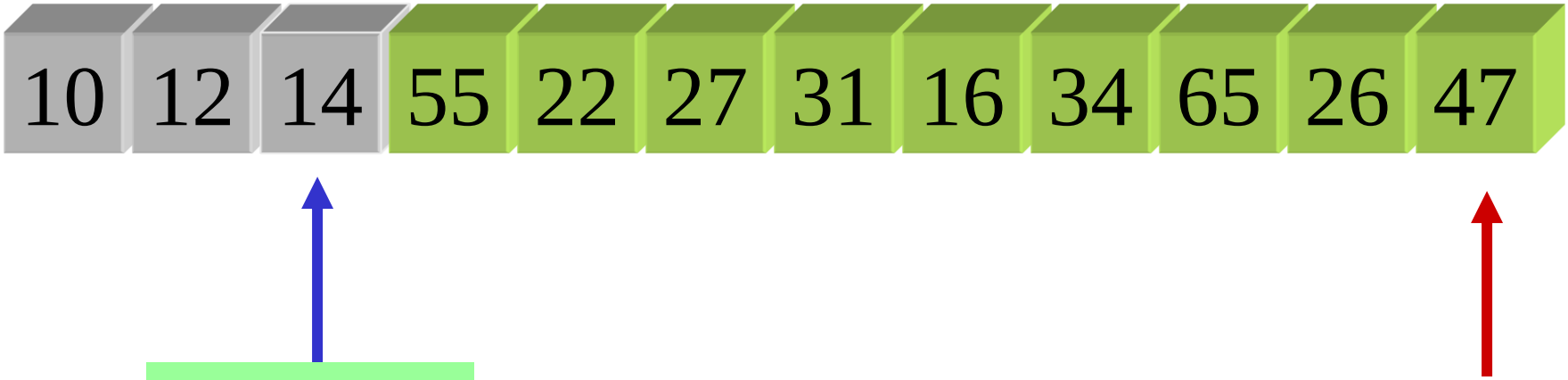
i=1

i indice di inizio porzione

n-1 indice di fine porzione

```
for (i=0; i < n-1; i++) {
    determinare l'elemento minino
    della porzione i..n-1 dell'array
    metterlo al primo posto della porzione
}
```

i=2




```
for (i=0; i < n-1; i++) {
    determinare l'elemento minino
    della porzione i..n-1 dell'array
    metterlo al primo posto della porzione
}
```

i=2

i indice di inizio porzione

n-1 indice di fine porzione

```
for (i=0; i < n-1; i++) {
      determinare l'elemento minino
      della porzione i..n-1 dell'array
      metterlo al primo posto della porzione
}
```

i=2

i indice di inizio porzione

n-1 indice di fine porzione

```
for (i=0; i < n-1; i++) {
    determinare l'elemento minino
    della porzione i..n-1 dell'array
    metterlo al primo posto della porzione
}
```

i=3

i indice di inizio porzione

n-1 indice di fine porzione

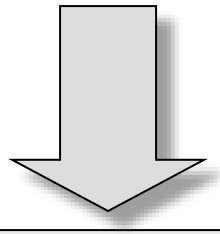
```
for (i=0; i < n-1; i++) {
      determinare l'elemento minino
      della porzione i..n-1 dell'array
      metterlo al primo posto della porzione
}
```

i=3

costrutto ripetitivo: 2 cicli **for** innestati

operazione ripetuta (al generico passo **i**, ciclo **for** esterno)**:**

- ✓ determinare il **minimo** elemento della porzione disordinata dell'array (**i..n-1**)
  [richiede un ciclo **for** **]**;
- ✓ spostare il minimo al **primo** posto della porzione



algoritmo di ordinamento **per selezione** di **minimo** (**selection sort**)

```
void ord_sel_min (char a[], int n) {
 int i, k, indice_min;
 for (i=0; i < n-1; i++) {
     indice_min = i ;
     for (k=i+1; k<n; k++) {
        if (a[k] < a[indice_min] )
           { indice_min = k ; }
      }
     scambiare_c(&a[i], &a[indice_min]);
  }
}
```

```
for (i=0;i<n-1;i++)
   for (k=i+1;k<n;k++)
         …
```

i	confronti
0	n-1 +
1	n-2 +
2	n-3 +
…	
n-3	2 +
n-2	1
	n(n-1)/2

```
void ord_sel_min (char a[], int n) {
 int i, k, indice_min;
 for (i=0; i < n-1; i++) {
     indice_min = i ;
     for (k=i+1; k<n; k++) {
        if (a[k] < a[indice_min] )
            { indice_min = k ; }
     }
    scambiare_c(&a[i], &a[indice_min]);
  }
}
```

n*(n-1)/2
confronti
(tra elementi dell'array)

n-1
scambi
(tra elementi dell'array)

```
void ord_sel_min (char a[], int n) {
 int i, k, indice_min;
 for (i=0; i < n-1; i++) {
     indice_min = i ;
     for (k=i+1; k<n; k++) {
        if (a[k] < a[indice_min] )
            { indice_min = k ; }
     }
    scambiare_c(&a[i], &a[indice_min]);
  }
}
```

determinazione dell'indice dell'elemento minimo porzione i..(n-1)

```
void ord_sel_min (char a[], int n) {
 int i, k, indice_min;
 for (i=0; i < n-1; i++) {
     indice_min = i ;
     for (k=i+1; k<n; k++) {
        if (a[k] < a[indice_min] )
           { indice_min = k ; }
     }
    scambiare_c(&a[i], &a[indice_min]);
  }
}
```

usare una function min_ind_a

int min_ind_a (char b[],int nb)

determina l'indice dell'elemento minimo dell'array **b** di size **nb**

```
void ord_sel_min (char a[], int n) {
 int i, k, indice_min;
 for (i=0; i < n-1; i++) {
    scambiare_c(&a[i], &a[indice_min      ]);
  }
}
```

usare una function min_ind_a

int min_ind_a (char b[],int nb)

determina l'indice dell'elemento minimo dell'array **b** di size **nb**

```
void ord_sel_min (char a[], int n) {
 int i, k, indice_min;
 for (i=0; i < n-1; i++) {

 indice_min= min_ind_a(a+i,n-i);

    scambiare_c(&a[i], &a[indice_min      ]);
  }
}
```

usare una function min_ind_a

int min_ind_a (char b[],int nb)

determina l'indice dell'elemento minimo dell'array **b** di size **nb**

applicare **min_ind_a** alla porzione **i..n-1** di **a**, che ha size **n-i**

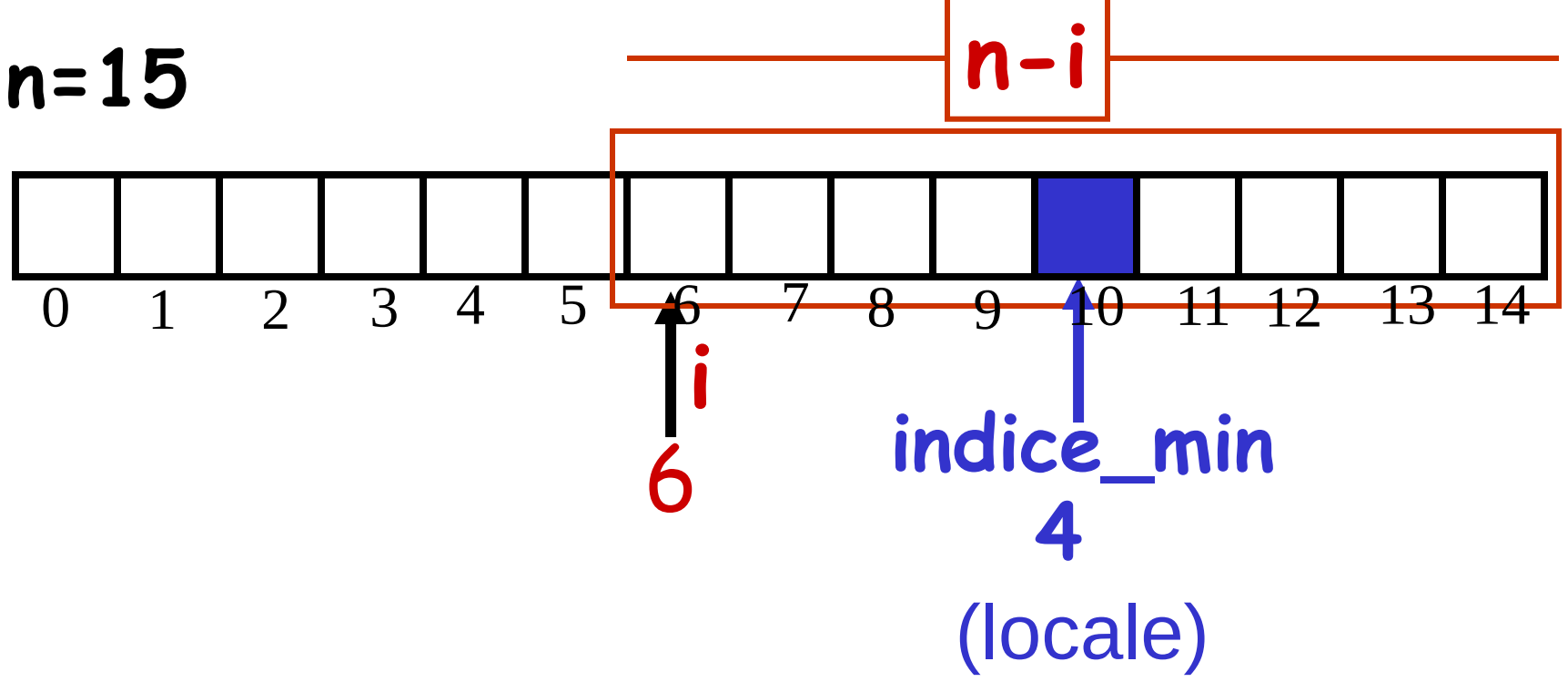
versione 2

```
void ord_sel_min (char a[], int n) {
 int indice_min, i;
 for (i=0; i < n-1; i++) {
     indice_min = min_ind_a(a+i, n-i) ;
     scambiare_c(&a[i], &a[indice_min+i]) ;
  }
}
```

indirizzo di inizio della porzione disordinata

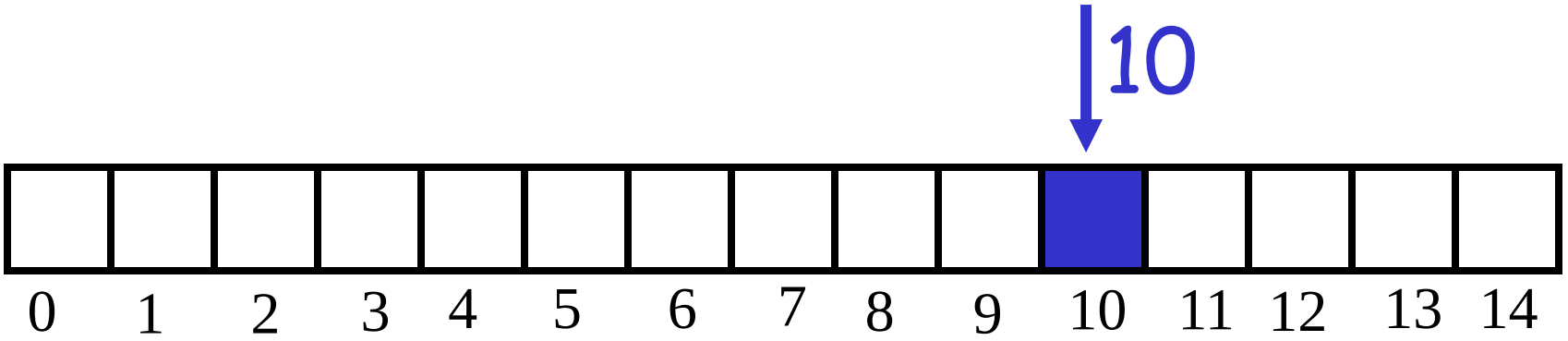
size della porzione disordinata

spiazzamento dell'indice locale



spiazzamento dell'indice locale

indice_min + i = 4+6=10



algoritmo di ordinamento **per selezione** (**selection sort**)

$T(n) = n*(n-1)/2$

confronti

(tra elementi dell'array)

$T(n) = O(n^2)$

confronti

$T(n) = n-1$

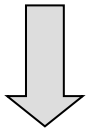
scambi

(tra elementi dell'array)

$T(n) = O(n)$

scambi

array già ordinato



complessità di tempo invariata

algoritmo di ordinamento
per selezione
(**selection sort**)

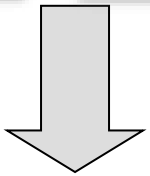
$T(n) = n*(n-1)/2 = O(n^2)$
confronti
(tra elementi dell'array)

$T(n) = n-1 = O(n)$
scambi
(tra elementi dell'array)

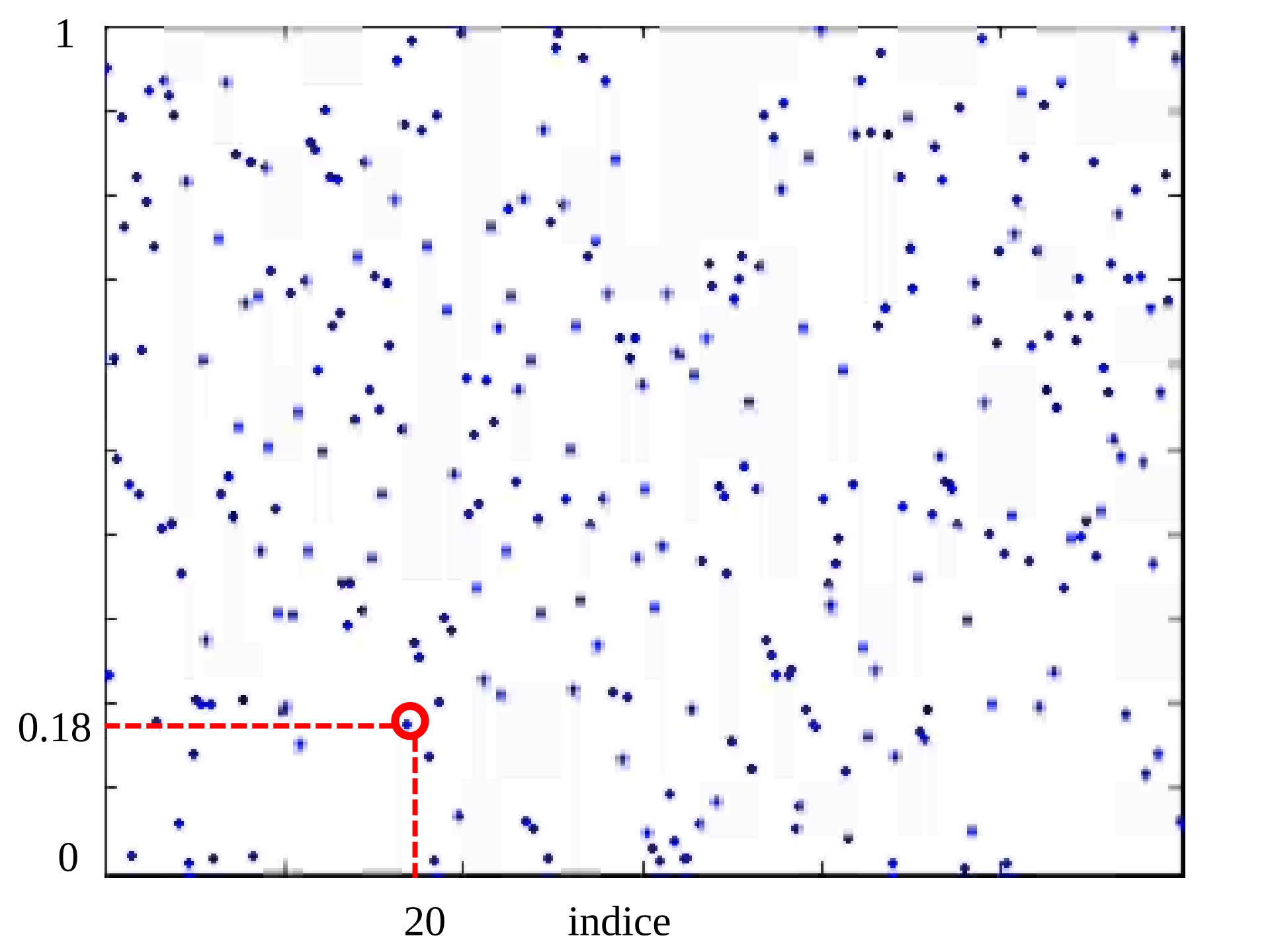
algoritmo di ordinamento
per inserimento
(**insertion sort**)

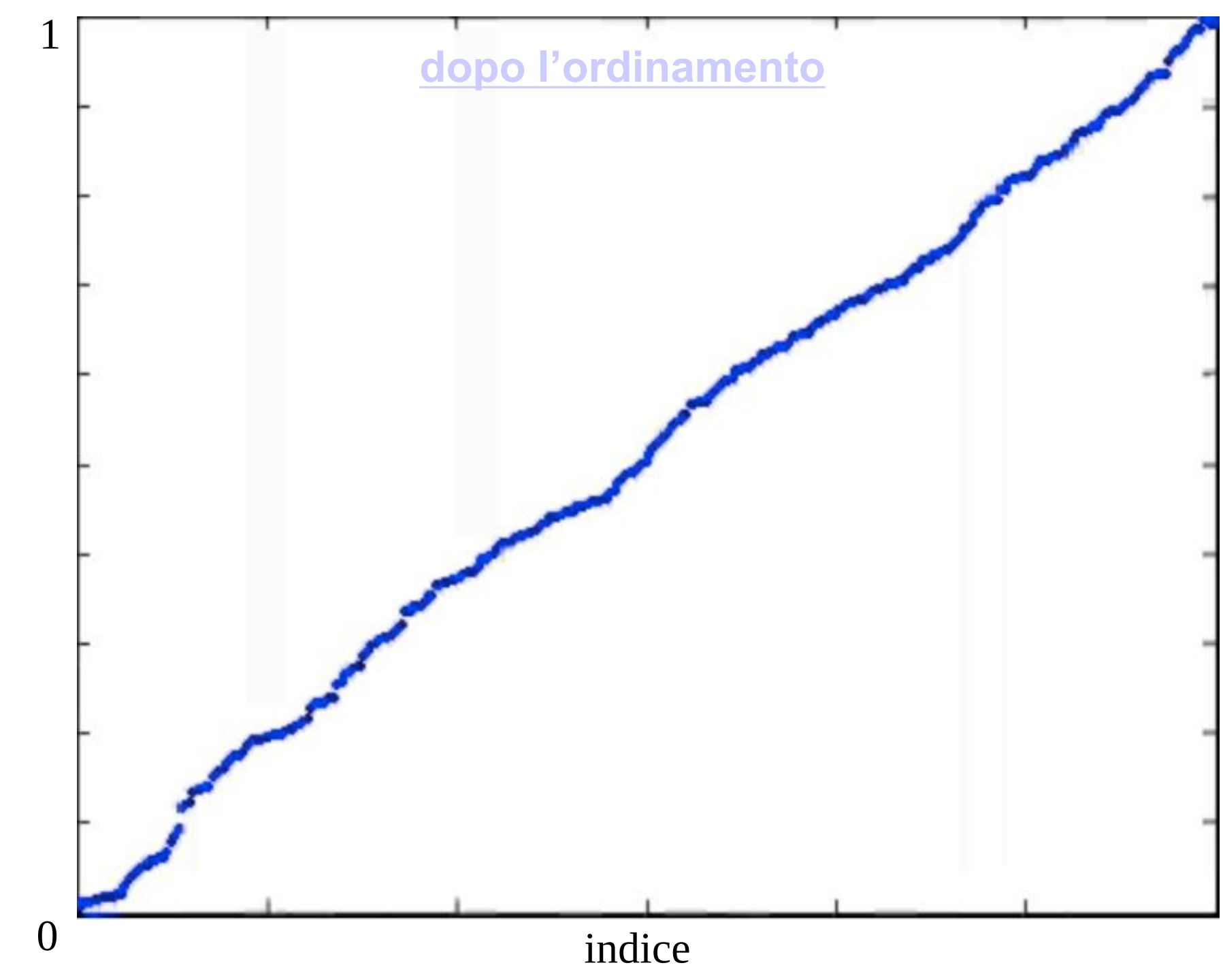
$T(n) = n*(n-1)/2 = O(n^2)$
(tra elementi dell'array)
al più

$T(n) = n*(n-1)/2 = O(n^2)$
scambi (spostamenti)
(tra elementi dell'array)
al più



algoritmi a
complessità di tempo **quadratica**
(confronti)


1
0.18
0
20
indice

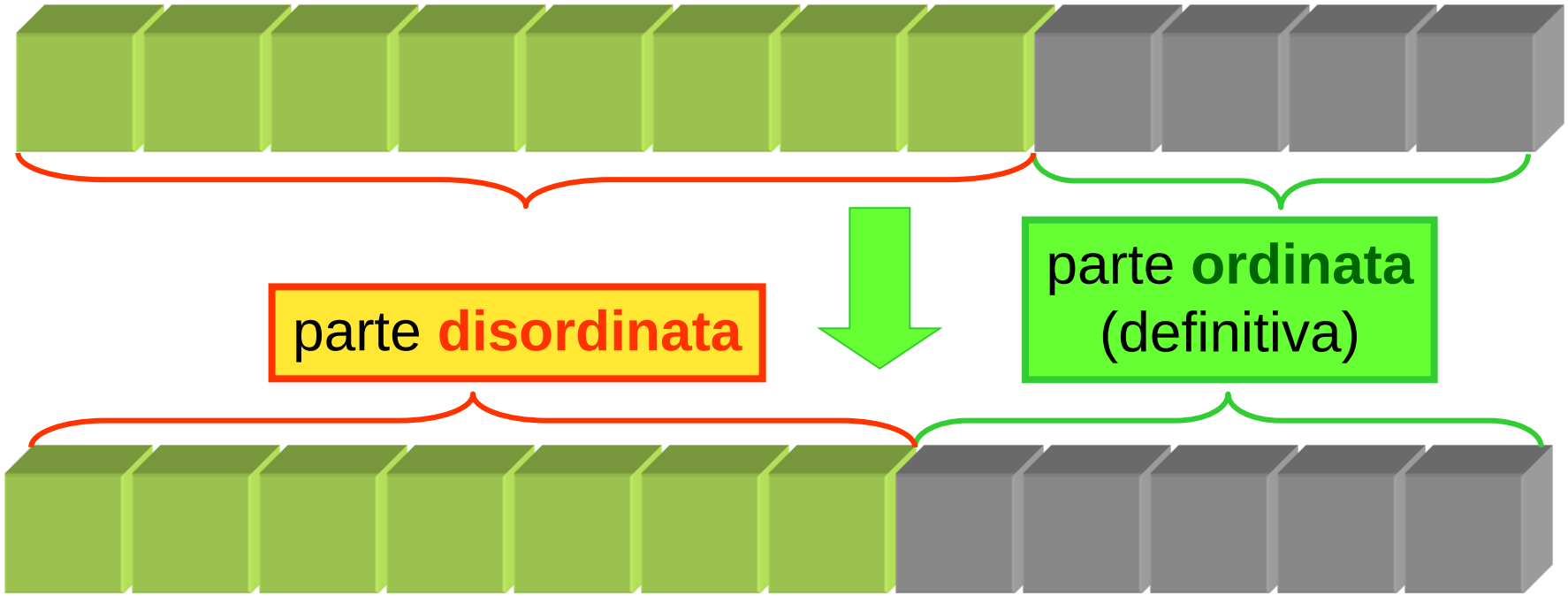

1
dopo l’ordinamento
0
indice

algoritmo di ordinamento **per selezione** di **minimo**

VARIANTE

VARIANTE

algoritmo di ordinamento **per selezione** di **massimo**



0 indice di inizio porzione

i indice di fine porzione

```
for (i=n-1; i >=1; i--) {
      determinare l'elemento massimo
      della porzione 0..i dell'array
      metterlo all'ultimo posto della porzione
}
```

0 indice di inizio porzione

i indice di fine porzione

```
for (i=n-1; i >=1; i--) {
      determinare l'elemento massimo
      della porzione 0..i dell'array
      metterlo all'ultimo posto della porzione
}
```

i=n-1

0 indice di inizio porzione

i indice di fine porzione

```
for (i=n-1; i >=1; i--) {
      determinare l'elemento massimo
      della porzione 0..i dell'array
      metterlo all'ultimo posto della porzione
}
```

i=n-1

0 indice di inizio porzione

i indice di fine porzione

```
for (i=n-1; i >=1; i--) {
    determinare l'elemento massimo
    della porzione 0..i dell'array
    metterlo all'ultimo posto della porzione
}
```

i=n-2

0 indice di inizio porzione

i indice di fine porzione

```
for (i=n-1; i >=1; i--) {
    determinare l'elemento massimo
    della porzione 0..i dell'array
    metterlo all'ultimo posto della porzione
}
```

i=n-2

0 indice di inizio porzione

i indice di fine porzione

```
for (i=n-1; i >=1; i--) {
      determinare l'elemento massimo
      della porzione 0..i dell'array
      metterlo all'ultimo posto della porzione
}
```

i=n-2

0 indice di inizio porzione

i indice di fine porzione

```
for (i=n-1; i >=1; i--) {
```

determinare l'elemento massimo della porzione 0..i dell'array
metterlo all'ultimo posto della porzione

```
}
```

i=n-3

0 indice di inizio porzione

i indice di fine porzione

```
for (i=n-1; i >=1; i--) {
      determinare l'elemento massimo
      della porzione 0..i dell'array
      metterlo all'ultimo posto della porzione
}
```

i=n-3

0 indice di inizio porzione

i indice di fine porzione

```
for (i=n-1; i >=1; i--) {
      determinare l'elemento massimo
      della porzione 0..i dell'array
      metterlo all'ultimo posto della porzione
}
```

i=n-3

0 indice di inizio porzione

i indice di fine porzione

```
for (i=n-1; i >=1; i--) {
      determinare l'elemento massimo
      della porzione 0..i dell'array
      metterlo all'ultimo posto della porzione
}
```

i=n-4

```
void ord_sel_max (char a[], int n) {
 int indice_max, i, k ;
 for (i=n-1; i > 0; i--) {
     indice_max = 0 ;
     for (k=1; k <= i; k++) {
        if (a[k] > a[indice_max] )
            { indice_max = k ; }
     }
   scambiare_c(&a[i], &a[indice_max]) ;
  }
}
```

n*(n-1)/2
confronti
(tra elementi dell'array)

n-1
scambi
(tra elementi dell'array)

```
void ord_sel_max (char a[], int n) {
 int indice_max, i, k ;
 for (i=n-1; i > 0; i--) {
     indice_max = 0 ;
     for (k=1; k <= i; k++) {
        if (a[k] > a[indice_max])
           { indice_max = k ; }
     }
     scambiare_c(&a[i], &a[indice_max]) ;
  }
}
```

determinazione dell'indice dell'elemento massimo porzione 0..i

```
void ord_sel_max (char a[], int n) {
 int indice_max, i, k ;
 for (i=n-1; i > 0; i--) {
     indice_max = 0 ;
     for (k=1; k <= i; k++) {
        if (a[k] > a[indice_max])
            { indice_max = k ; }
     }
     scambiare_c(&a[i], &a[indice_max]) ;
  }
}
```

usare una function max_ind_a

int max_ind_a (char b[],int nb)

determina l'indice dell'elemento massimo dell'array **b** di size **nb**

```
void ord_sel_max (char a[], int n) {
 int indice_max, i, k ;
 for (i=n-1; i > 0; i--) {

 indice_max= max_ind_a(a, i+1);

    scambiare_c(&a[i], &a[indice_max]) ;
  }
}
```

usare una function max_ind_a

int max_ind_a (char b[],int nb)

determina l'indice dell'elemento massimo dell'array **b** di size **nb**

applicare **max_ind_a** alla porzione **0..i** di **a**, che ha size **i+1**

versione 2

```
void ord_sel_max (char a[], int n) {
 int indice_max, i;
 for (i=n-1; i > 0; i--) {
     indice_max = max_ind_a(a, i+1) ;
     scambiare_c(&a[i], &a[indice_max]) ;
  }
}
```

indirizzo di inizio della porzione disordinata

size della porzione disordinata