ALGORITMI E STRUTTURE DATI

Prof. Manuela Montangero

A.A. 2022/23

ALGORITMI DI ORDINAMENTO: SelectitonSort

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

PROBLEMA dell'ORDINAMENTO

INPUT:

Una sequenza di n valori $A = < a_0, a_1, \ldots, a_{n-1} >$ appartenenti ad un insieme totalmente ordinabili

OUTPUT:

La sequenza A ordinata in ordine non decrescente

COSTO COMPUTAZIONALE: contiamo i confronti tra valori di A

PROBLEMA dell'ORDINAMENTO

PRIMA IDEA:

Provare tutte le permutazioni dei valori in A e, per ogni permutazione, controllare se i valori sono ordinati.

Quanto costa?

- Per ogni permutazione $\Theta(n)$
- Ci sono $n!$ permutazioni

$$\Downarrow$$

$$\Theta(n \cdot n!)$$

😬

SelectionSort

SECONDA IDEA (**SelectionSort**):

- Cerca il minimo tra gli n elementi e scambialo con quello nella prima posizione
- Ordina i restanti $n-1$ elementi nello stesso modo
- Se c'e' un solo elemento da ordinare, hai finito

E' RICORSIVO!

```
SelectionSort(A,i,n)
 if i < n-1
  then
   k := indice dell'elemento minimo in A[i..n-1]
   inverti A[i] e A[k]
   SelectionSort(A,i+1,n)
```

E' una PROCEDURA!

Alla fine dell'esecuzione gli elementi in `A` sono ordinati

Chiamata Principale: **`SelectionSort(A,0,n)`**

SelectionSort

Scrivere una funzione per il seguente problema:

INPUT: array `A` di n elementi e un indice i, $0 \leq i \leq n-1$

OUTPUT: l'indice dell'elemento minimo in `A[i..n-1]`

```
IndiceDelMinimo(A,i,n)
k := i
for j = i+1 to n-1
 if A[j] < A[k]
  then
    k := j
return k
```

E' una funzione

Costo computazionale: esattamente

$$(n-1)-(i+1)+1 = n-1-i$$

confronti

$$\Rightarrow \Theta(n-i)$$

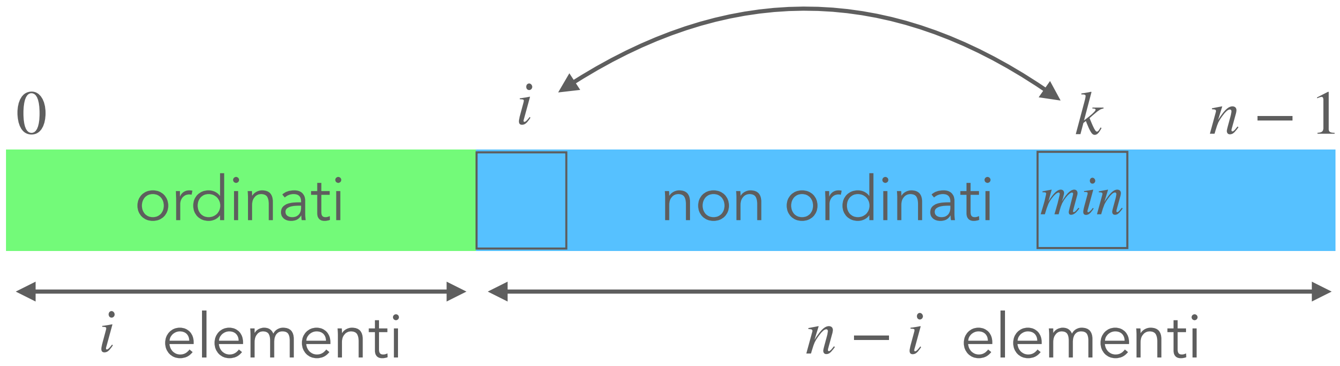
SelectionSort

Perché funziona?

Prima della chiamata generica su $A[i\,..\,n-1]$

Proprietà:

$$A[z] \leq A[t] \;\; \forall z \in [0..i-1], t \in [i\,..\,n-1] \quad (*)$$



Cosa facciamo durante la chiamata

SelectionSort

Perché funziona?

Prima della chiamata su $A[i+1..n-1]$

0 i $i+1$ $n-1$

sono ordinati? SI | non ordinati

$i+1$ elementi | $n-i-1$ elementi

- Fino all'indice $i-1$ non è cambiato niente
- A[i] è $\geq$ di tutti i valori in A[0..i-1] per (*)

La proprietà vale ancora? cioè è vero che

$$A[z] \leq A[t] \ \ \forall z \in [0..i], t \in [i+1..n-1] \ \ ?$$ SI

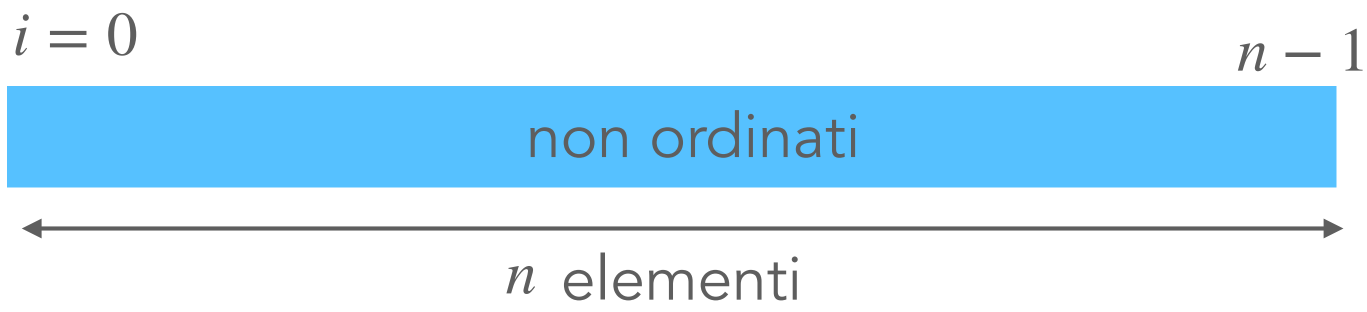
- Per gli intervalli di indici $[0..i-1]$ e $[i+1..n-1]$ non è cambiato niente
- in $A[i]$ abbiamo il minimo di $A[i..n-1]$, quindi è vero anche che

$$A[i] \leq A[t] \ \ \forall t \in A[i+1..n-1]$$

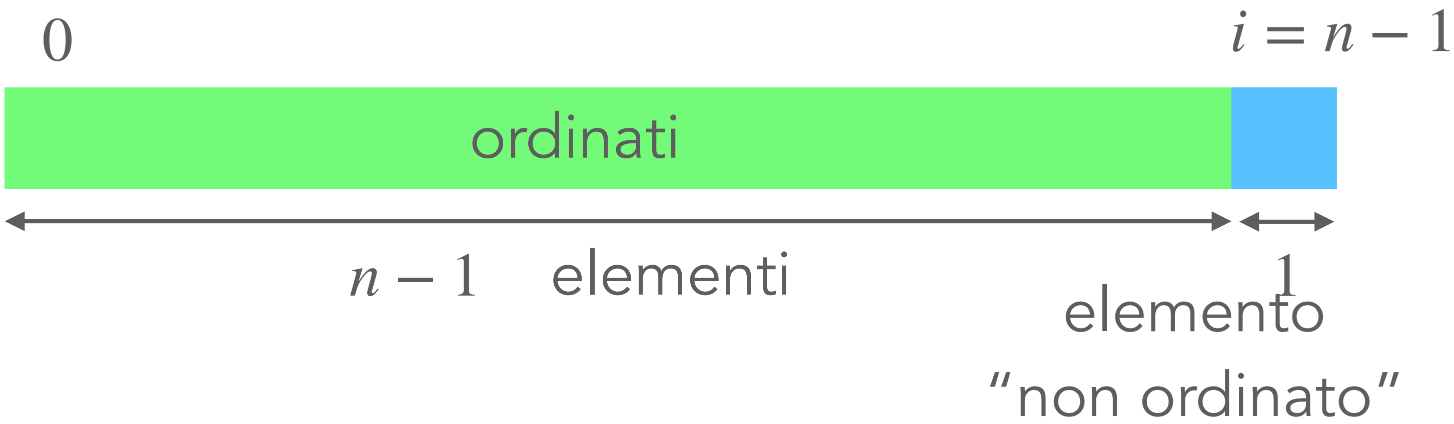
SelectionSort

Prima della prima chiamata

La porzione ordinata ha dimensione zero



Prima dell'ultima chiamata



Per la proprietà (*) $A[n-1]$ è il massimo e quindi è al posto giusto

SelectionSort

```
SelectionSort(A,i,n)
 if i < n-1
  then
   k := IndiceDelMinimo(A,i,n)
   inverti A[i] e A[k]
   SelectionSort(A,i+1,n)
```

Chiamata Principale:

`SelectionSort(A,0,n)`

Quanto costa?

- Il caso base si ha quando l'array ha un solo elemento da ordinare, $i = n - 1$, e non si devono eseguire confronti
- Ogni chiamata ricorsiva richiede: i confronti tra elementi di `A` necessari per trovare k, PIU' quelli della chiamata ricorsiva
- Dato `A` con i elementi, `IndiceDelMinimo` richiede $n - 1 - i$ operazioni
- La procedura ricorsiva viene invocata esattamente una volta per ogni i tale che $0 \leq i \leq n - 2$

SelectionSort

Quanto costa?

- Il caso base si ha quando l'array ha un solo elemento da ordinare, $i = n - 1$, e non si devono eseguire confronti
- Ogni chiamata ricorsiva richiede: i confronti tra elementi di `A` necessari per trovare k, PIU' quelli della chiamata ricorsiva
- Dato `A` con i elementi, `IndiceDelMinimo` richiede $n - 1 - i$ operazioni
- La procedura ricorsiva viene invocata esattamente una volta per ogni i tale che $0 \leq i \leq n - 2$

Sommando su tutte le chiamate ricorsive

$$T(n) = \sum_{i=0}^{n-2} (n - 1 - i) = \sum_{i=0}^{n-2} (n - 1) - \sum_{i=0}^{n-2} i = (n - 1) \sum_{i=0}^{n-2} 1 - \frac{(n-2)(n-1)}{2} =$$

$$= (n-1)(n-1) - \frac{(n-1)(n-2)}{2} = (n-1)\frac{2(n-1) - (n-2)}{2} = \frac{(n-1)n}{2} \in \Theta(n^2)$$

SelectionSort

Quanto costa? Scriviamo un'equazione di ricorrenza

- Il caso base si ha quando l'array ha un solo elemento da ordinare (ovvero il problema ha dimensione uno), e non si devono eseguire operazioni
- Ogni chiamata ricorsiva richiede: i confronti tra elementi di `A` necessari per trovare k, PIU' quelli della chiamata ricorsiva
- Dato il problema di dimensione n, `IndiceDelMinimo` richiede $n-1$ operazioni di confronto
- L'invocazione della procedura ricorsiva viene fatta su un sottoproblema di dimensione $n-1$

$$T(n) = \begin{cases} 0 & \text{se } n = 1 \\ T(n-1) + n - 1 & \text{altrimenti} \end{cases}$$

$$T(n) = T(n-1) + n - 1 = T(n-2) + (n-2) + (n-1) =$$

$$= T(n-3) + (n-3) + (n-2) + (n-1) = \ldots$$

$$\ldots = T(n-(n-1)) + 1 + 2 + \ldots (n-2) + (n-1) = \sum_{i=0}^{n-1} i = \frac{(n-1)n}{2} \in \Theta(n^2)$$