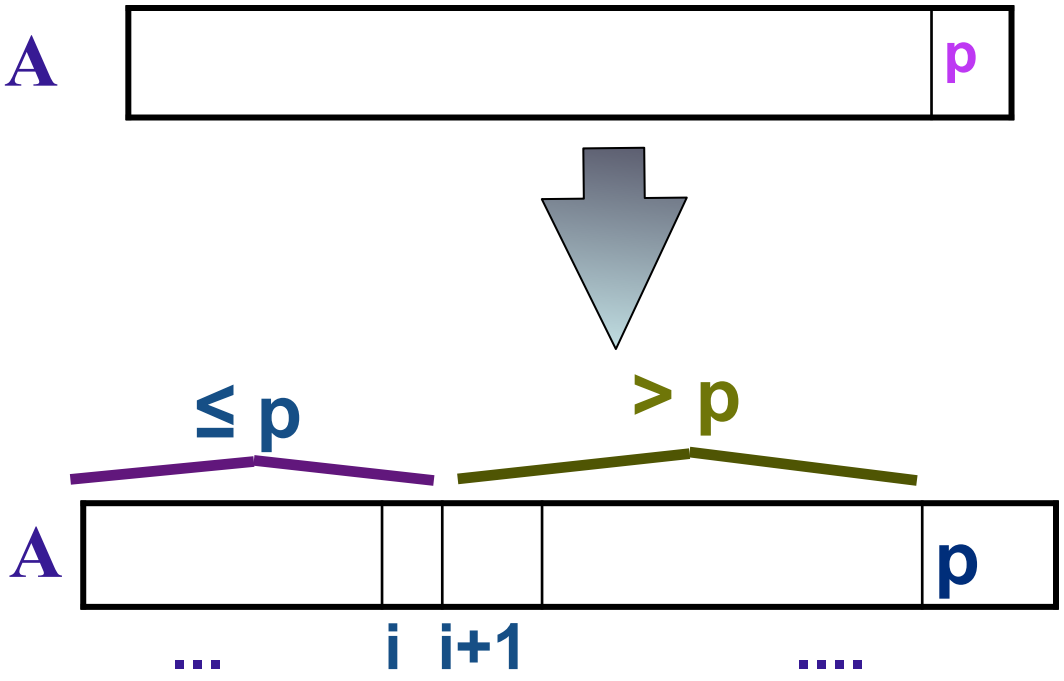
In questa lezione

Descriveremo la funzione PARTIZIONE che prende in input un array non ordinato di interi e individua la posizione che dovrebbe assumere un elemento scelto di un array di interi, se fosse ordinato.

Partizione di un array

Preso un elemento della lista, p, si vuole dividere la lista in modo che i più piccoli dell'elemento preso precedano i più grandi. L'elemento scelto viene chiamato pivot.

il confine tra le due parti è la posizione che il pivot dovrebbe assumere se la lista fosse ordinata.



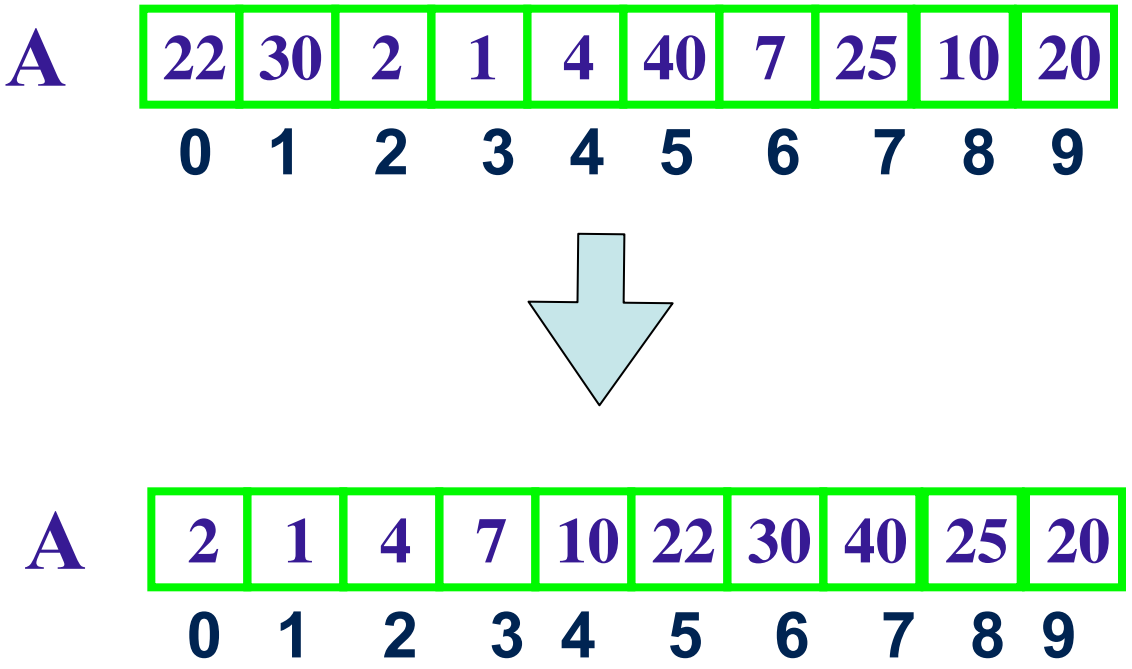
p è il pivot, è l'ultimo elemento.

Si vuole progettare un ciclo che al termine assicuri che

per ogni 0≤h≤i A[h] ≤ pivot e i<k<n-1 A[k] > pivot.

Partizione: esempio

Bisogna trovare l'indice che individua la posizione "giusta" per il pivot, cioè dividere gli altri elementi tra minori o uguali e maggiori del pivot.



5 è l'indice giusto per 20: gli elementi di indice 0 fino a 4 sono più piccoli e quelli da 5 a 8 sono più grandi

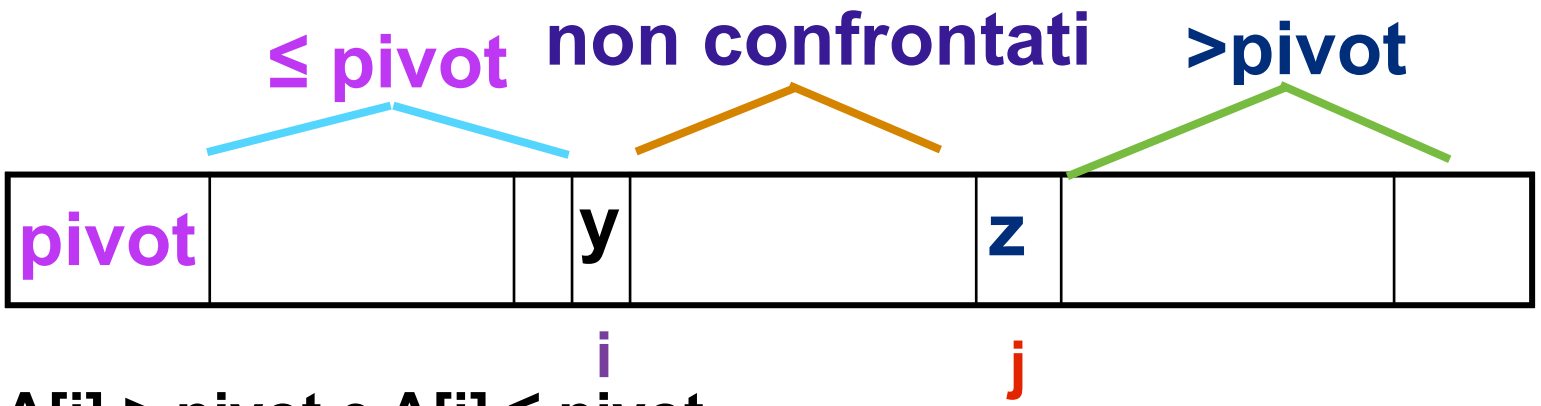
La partizione di un array: l'algoritmo di Hoare

Si potrebbe scorrere l'array da sinistra verso destra e da destra verso sinistra.

Da sinistra verso destra si procede fintanto che gli elementi sono minori o uguali al pivot, da destra verso sinistra fino a che sono maggiori del pivot. Questa volta il pivot è il primo elemento.

Quando i due cicli terminano i relativi indici di scorrimento indicano rispettivamente un elemento maggiore del pivot e un elemento minore del pivot

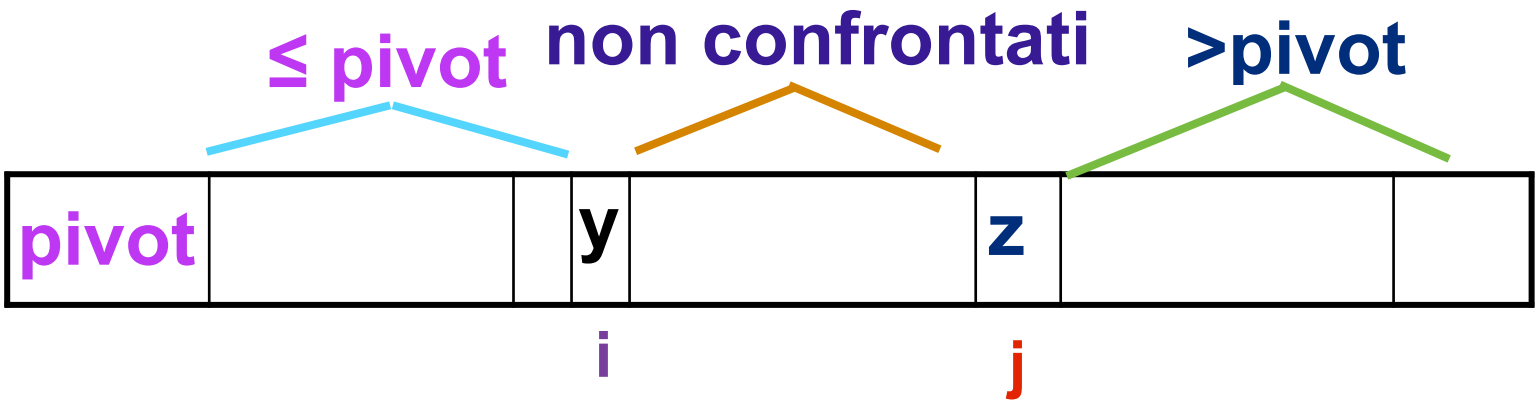
Come pivot prendiamo il primo elemento. Quindi in un passo intermedio:



A[i] > pivot e A[j] ≤ pivot

Partizione: verso l'algoritmo

L'obiettivo del ciclo di partizione è: per ogni 0≤h≤i A[h] ≤ pivot e i<k<n A[k] > pivot.

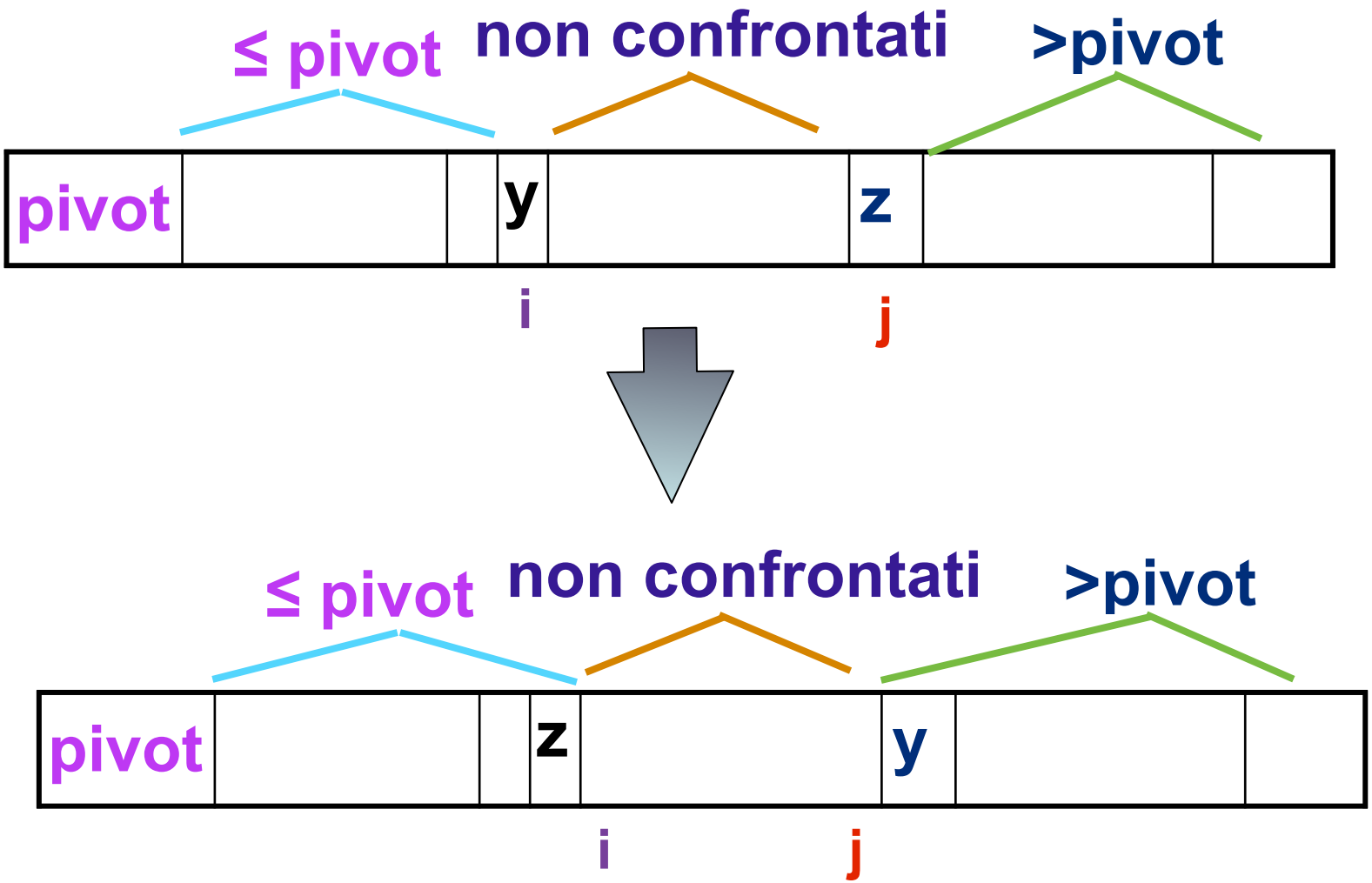


A[i] > pivot e A[j] ≤ pivot

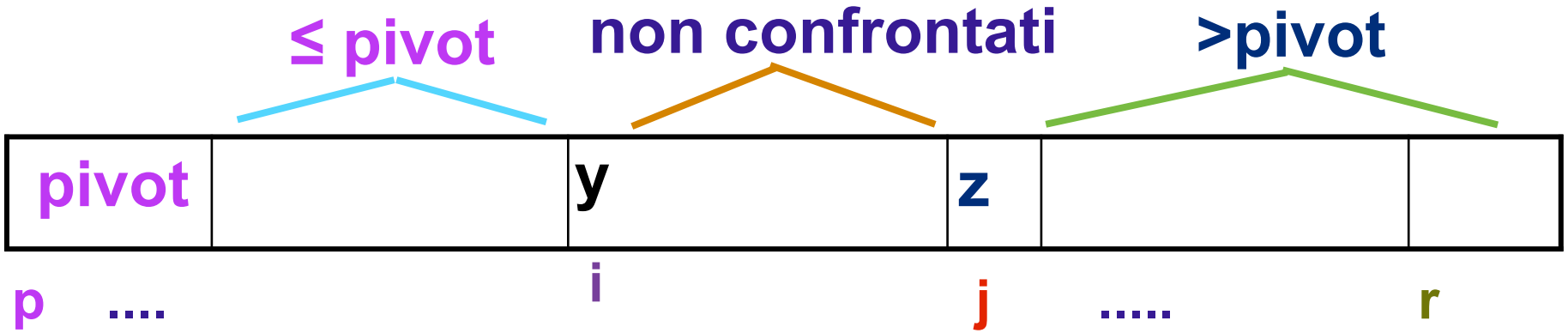
Qual'è la mossa da fare?

Lo scambio tra y e z e l'incremento di entrambi gli indici

Partizione: un passo



Verso lo pseudocodice



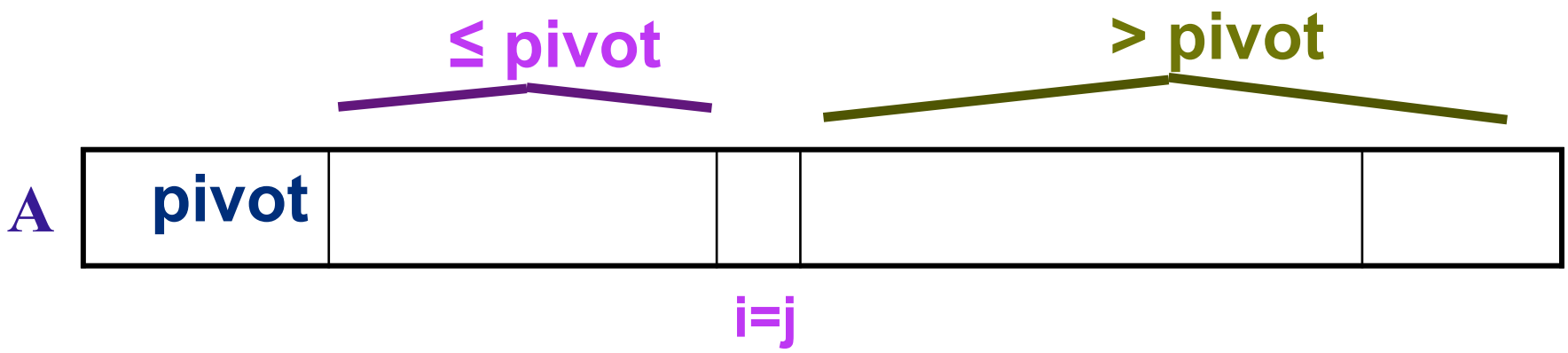
istruzioni dentro il ciclo

finchè A[i] ≤ pivot do i++ Da questo ciclo si esce con A[i]>pivot

finchè A[j] > pivot do j- - Da questo ciclo si esce con A[j]≤pivot

scambia A[i] con A[j]

definire l'uscita dal ciclo



l'ultimo elemento da confrontare sarà per i = j.

Poi se A[i] ≤ pivot si incrementa i, quindi i>j,

A[j] ≤ pivot e A[j+1] > pivot

altrimenti A[i] > pivot e allora si decrementa j, si ha ancora i>j,

A[j] ≤ pivot e A[j+1] > pivot

Dunque j è la posizione finale per il pivot.

Lo pseudocodice

```
Partizione(A)
 n = len(A)
 pivot = A[0]
 i = 1
 j  = n-1
while i≤j   do
     while  A[j] > pivot do j- -
     while  A[i] ≤ pivot and i≤j do i++
          if (i<j) scambia A[i] con A[j]
                   j- -
                   i++
 return j
```

Perché non è necessario il confronto di j con 0?

Lo pseudocodice

```
Partizione(A)
 n = len(A)
 pivot = A[0]
 i = 1
 j  = n-1
while i≤j   do
     while  A[j] > pivot do j- -
     while  A[i] ≤ pivot and i≤j do i++
          if (i<j) scambia A[i] con A[j]
                   j- -
                   i++
 return j
```

Complessità?