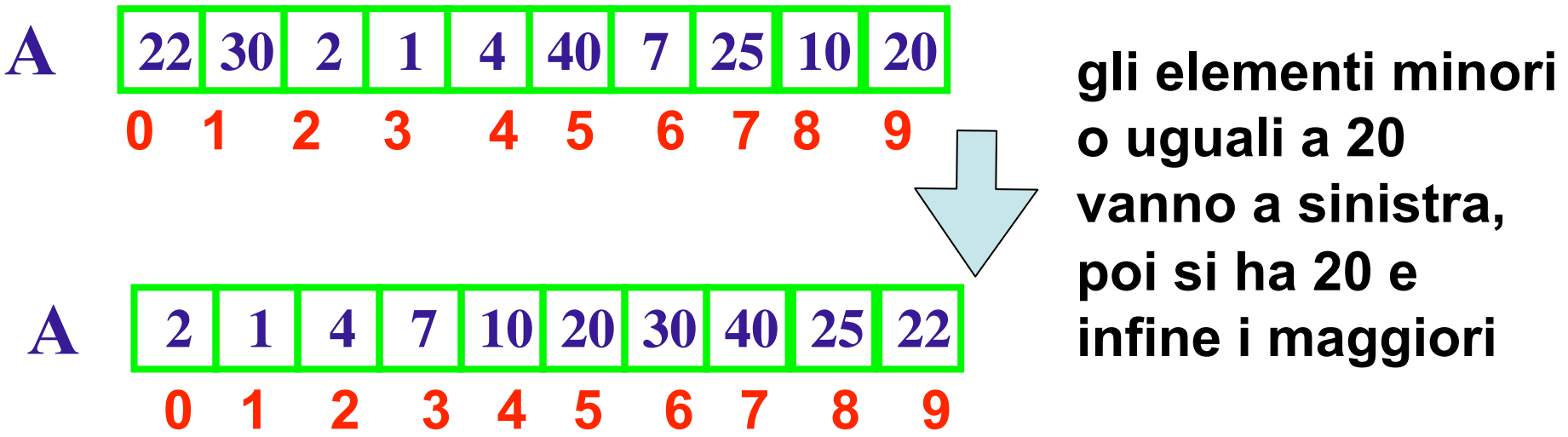
Partition

La funzione PARTIZIONE prende in input un array A non ordinato di interi e individua la posizione che dovrebbe assumere un elemento scelto, detto pivot, di A, se fosse ordinato. Gli elementi minori o uguali del pivot sono spostati in modo da precedere il pivot stesso e tutti quelli maggiori sono spostati in modo da seguirlo.

Partizione

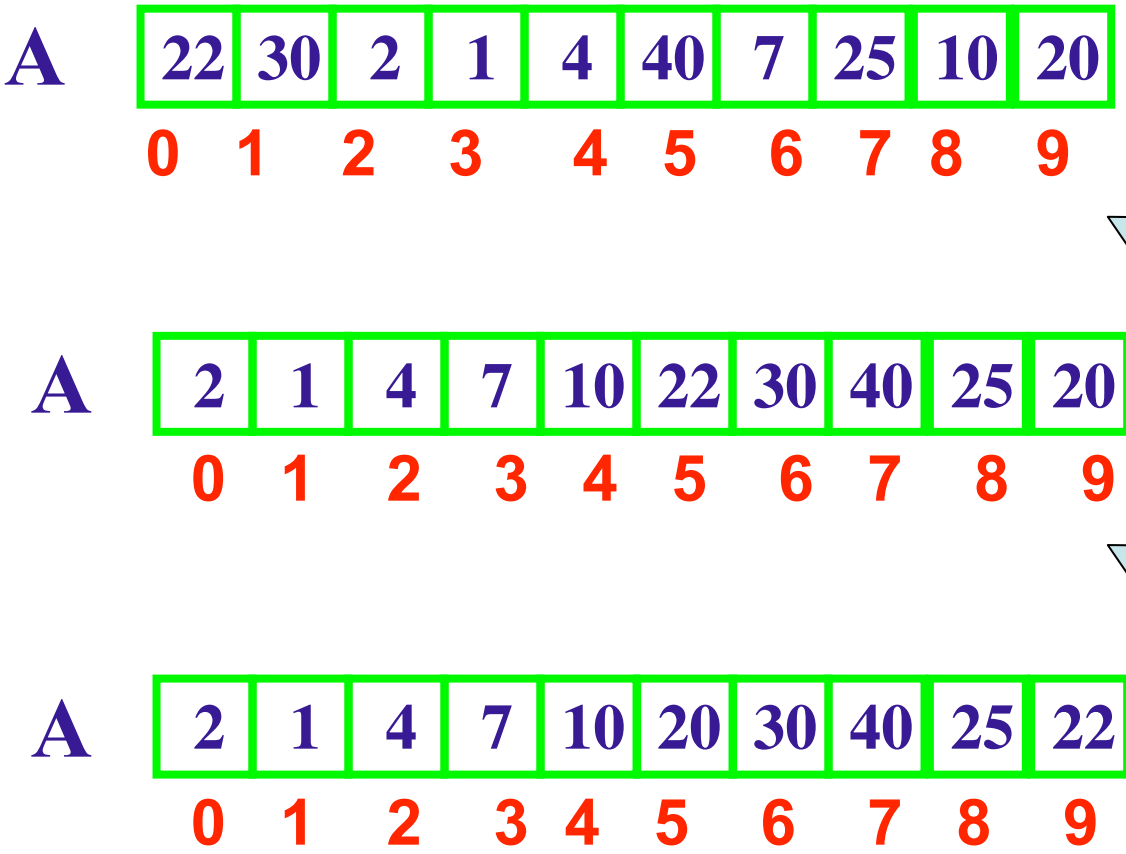
Dato un array A di interi individuato un elemento di A, chiamato pivot, vogliamo spostare tutti gli elementi minori o uguali al pivot in modo che precedano in A il pivot e che quest'ultimo preceda in A quelli più grandi del pivot. Esempio, il pivot è l'ultimo elemento di A:



gli elementi minori o uguali a 20 vanno a sinistra, poi si ha 20 e infine i maggiori

Partizione

L'obiettivo viene raggiunto in due passi, nel primo si spostano gi elementi minori o uguali al pivot in modo che precedano i maggiori. Nel secondo si scambia il pivot con il primo dei più grandi.



gli elementi minori o uguali a 20 vanno a sinistra, a seguire i maggiori

scambio del pivot con il primo dei più grandi

L’idea

L’idea è di mantenere un indice, i, che limita la porzione di array contenente gli elementi più piccoli del pivot e disporre gli elementi più grandi a seguire, cioè tra i e quello di scorrimento dell’array. Ad ogni passo se l’elemento è maggiore o uguale al pivot basta passare a esaminare l’elemento successivo, se invece è più piccolo possiamo sistemarlo scambiandolo con il primo dei più grandi, cioè quello di indice i+1.

Partizione(A,p,r)

Input: A è un array e 0≤p≤r≤ A.length

output: restituisce l'indice, i+1, gli elementi di indice da p a i sono minori di A[r], quelli di indice da i+2 a r sono maggiori.

```
pivot = A[r]
i = p-1   All'inizio non ci sono
              elementi ≤ pivot
j = p     e nemmeno > pivot
while j ≤ r-1 do
      if  A[j] ≤ pivot then
            i =  i+1
            scambia A[i] con A[j]
     j= j+1 end while
scambia A[i+1] con A[r]
i+1 è la posizione del pivot
return i+1
```

in ogni passo di esecuzione del ciclo
Gli elementi di indice tra p e i sono ≤ pivot
gli elementi di indice tra i+1 e j-1 sono
> pivot

L’idea

L’idea è di scorrere l’array sia da sinistra che da destra verso il centro. Si scorre da sinistra finché gli elementi sono minori o uguali al pivot e da destra finché gli elementi sono maggiori del pivot. Quando i due processi si fermano vuol dire che a sinistra ho un elemento maggiore del pivot e a destra un elemento minore, quindi bisogna scambiarli e far ripartire l’esame dell’array da sinistra e da destra verso il centro, come prima.
Il processo si conclude quando tutti gli elementi sono stati confrontati.

Partizione2(A,p,r)

Input: A è un array e 0≤p≤r≤ A.length

output: restituisce l'indice j, gli elementi di indice da p a j-1 sono minori di A[p], quelli di indice da j+1 a r sono maggiori.

```
pivot = A[p]
i = p+1  All'inizio non ci sono elementi ≤ pivot
j  = r e nemmeno > pivot!
while i≤j   do
        while  A[j] > pivot do j = j-1  qui si esce se A[j]≤pivot
        while  i≤j and A[i] ≤ pivot do i = i+1
                    qui si esce se  i>j o A[i]>pivot
         if (i < j) then
                scambia A[i] con A[j]
                 j = j-1
                 i = i+1
scambia il pivot con A[j]
return j
```

perché nel primo ciclo non c'è bisogno del test j≥p?