Vettore range

# esercizio vettore range:

Leggere il primo numero di un vettore casuale, e suddividere il vettore spostando alla fine, in modo non ordinato, tutti i valori maggiori ad esso, e tutti quelli minori all’inizio, controllando ogni volta se il valore da spostare si trova già in una posizione adatta (minimizzare gli spostamenti), e solo eventualmente spostarlo.

## Analisi:

Leggo il primo valore, lo salvo come valore bandiera e lo sposto alla fine, ma leggo se questo alla fine è già maggiore o minore a questo, nel caso sia minore, faccio proprio uno swap e lo sposto all’inizio, nel caso invece sia maggiore e quindi andrebbe bene, passo al penultimo valore e verifico la stessa condizione, e così via fino a quando non riesco a swappare il valore.

Successivamente, saltando la lettura del primo valore e del numero di valori finali che sono già stati verificati come in una posizione valida, leggo quelli successivi e continuo con le stesse condizioni.

Una volta che tutti i valori che non soddisfano le condizioni sono stati spostati, finisco.

### codice:

Partition(A, p, r)  
x <- A[p]  
i <- p – 1  
j <- r + 1  
 while true  
 do repeat j <- j – 1  
 until A[j] <= x  
 repeat i <- i + 1  
 until A[i] >= x  
 if i < j  
 then scambia A[i] <-> A[j]  
 else return j

Decodificato in poche parole:

/\*\*  
 \* A e' il vettore.  
 \* inizio e' la posizione iniziale o fin dove e' ordinato.  
 \* nNumeri e' il numero di numeri, ossia la fine o fin dove ordinare.  
 \*  
 \* La condizione di fine resta nel fatto che quando questi due si incontrano (diventano uguali), allora si ha finito  
 \* e si ritorna il valore della posizione in cui si incontrano.  
 \* \*/

# Quicksort:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 18 | 7 | 13 | 7 | 4 | 3 | 12 | 10 | 17 | 1 | 2 | 8 | 9 | 15 | 6 | 41 |
| 6 | 7 | 13 | 7 | 4 | 3 | 12 | 10 | 17 | 1 | 2 | 8 | 9 | 15 | 18 | 41 |
| 2 | 1 | 3 | 4 | 7 | 13 | 12 | 10 | 17 | 7 | 6 | 8 | 9 | 15 | 18 | 41 |
| 1 | 2 | 3 | 4 | 6 | 13 | 12 | 10 | 17 | 7 | 7 | 8 | 9 | 15 | 18 | 41 |
| 1 | 2 | 3 | 4 | 6 | 7 | 7 | 10 | 17 | 12 | 13 | 8 | 9 | 15 | 18 | 41 |
| 1 | 2 | 3 | 4 | 6 | 7 | 7 | 9 | 8 | 12 | 13 | 17 | 10 | 15 | 18 | 41 |
| 1 | 2 | 3 | 4 | 6 | 7 | 7 | 8 | 9 | 10 | 13 | 17 | 12 | 15 | 18 | 41 |
| 1 | 2 | 3 | 4 | 6 | 7 | 7 | 8 | 9 | 10 | 12 | 17 | 13 | 15 | 18 | 41 |
| 1 | 2 | 3 | 4 | 6 | 7 | 7 | 8 | 9 | 10 | 12 | 15 | 13 | 17 | 18 | 41 |
| 1 | 2 | 3 | 4 | 6 | 7 | 7 | 8 | 9 | 10 | 12 | 13 | 15 | 17 | 18 | 41 |

Partition(A,p,r)  
x = A[p]  
i = p -1  
j = r + 1  
while true  
 do repeat j = j – 1  
 until a[j] <= x  
 repeat i = i + 1  
 until A[i] >= x  
 if i < j  
 then swap A[i], A[j]  
 else return j;