ESERCIZIO 5

Sia A un vettore di interi distinti di dimensione n. L'ingegnere L.B. ci fornisce un algoritmo QuickSelect(A, p, q, r), dove p, q, r sono tali che $1 \leq p \leq r \leq q \leq n$. Dopo l'esecuzione di QuickSelect(A, p, q, r) valgono le seguenti condizioni su A:

- A[r] contiene l'elemento che si troverebbe in posizione r in A se il sottovettore A[p..q] fosse ordinato;
- A[p..q] è partizionato rispetto al pivot A[r], precisamente:
  - $A[i] < A[r]$ per ogni i tale che $p \leq i < r$
  - $A[r] < A[j]$ per ogni j tale che $r < j \leq q$

L.B. ci dice che QuickSelect(A, p, q, r) ha complessità O(log m), dove $m = q - p + 1$.

1. Si utilizzi QuickSelect per costruire un algoritmo efficiente per ordinare A. Si scriva lo pseudo-codice e si calcoli la complessità dell'algoritmo proposto.
2. Può esistere un algoritmo QuickSelect con le proprietà sopra descritte? Motivare la risposta.

Risposte:

1. La complessità dell'algoritmo è data dal costo di QuickSelect moltiplicato per il numero di volte che viene richiamato ricorsivamente. Poiché la dimensione del problema viene ridotta di almeno un fattore costante ad ogni chiamata, l'algoritmo ha una complessità di O(n log n) nel caso peggiore, dove n è la dimensione del vettore A. QuickSelect non garantisce la stabilità dell'ordinamento.

```
QuickSelect(A, p, q, r)
    if p == q
        return A[p]

    x = scegliPivot(A, p, q)    // sceglie il pivot

    if r == x
        return A[i]
    else if se r < x
        return QuickSelect(A, p, x - 1, r)
    else
        return QuickSelect(A, x + 1, q, r)

scegliPivot(A, p, q)
    x = sceglie valore casuale tra p e q
    return x
```

2. Sì, esistono varianti di QuickSelect che hanno una complessità di O(log m), dove m = q - p + 1. Queste varianti si basano sull'utilizzo di un'euristica che sceglie il pivot in modo efficiente, ad esempio scegliendo un pivot casuale.

Un esempio di algoritmo QuickSelect con complessità O(log m) è il seguente:

```
QuickSelect(A, p, q, r)
    se p == q
        return A[p]
    r = scegliPivot(A, p, q)    // sceglie il pivot in modo efficiente
    swap([A[r], A[q]])          // scambia il pivot con l'ultimo elemento
    i = p - 1
    per j = p fino a q - 1
        se A[j] < A[q]
            i = i + 1
            swap([A[i], A[j]])
    i = i + 1
    swap([A[i], A[q]])
    if r == i
        return A[i]
    else if se r < i
        return QuickSelect(A, p, i - 1, r)
    else
        return QuickSelect(A, i + 1, q, r)

scegliPivot(A, p, q)
    med = sceglie valore casuale tra p e q
    return med
```

L'algoritmo QuickSelect sopra ha una complessità di O(log m) nel caso peggiore, dove m = q - p + 1. Questo perché, scegliendo il pivot in modo efficiente, il sottovettore considerato ad ogni chiamata ricorsiva viene dimezzato, garantendo quindi una complessità logaritmica.