k-ième élément :

Problème :

Le problème du k-ième élément est celui de retrouver, dans un tableau de taille n, l'élément qui serait à la position k < n si le tableau était trié. Sachant que un tri peut au mieux être effectué en O(n·log(n)), il serait tentant de croire que la solution au problème du k-ième élément ne peut se résoudre qu'en O(n·log(n)).

Approche algorithmique :

Il est possible de retrouver le k-ième élément dans un tableau sans avoir à trier ce dernier. Pour cela, il faut utiliser une approche similaire à celle du tri rapide (QuickSort). Voici le pseudo-code :

```
kiemeRecursif( T[0..n-1] : tableau, k : indice recherché )

    Si  taille du tableau >= limite,
            On choisi un pivot dans T[0..n-1] par la méthode median3 ;
            On crée deux partitions (gauche et droite) contenant respectivement
                les éléments inférieurs et supérieurs au pivot ;
            m ← indice du pivot;
            Si m = k,
                Sortir;
            Sinon Si k < m,
                kiemeRecursif( T[0..m-1], k);
            Sinon
                kiemeRecursif( T[m+1..n-1], k);
            Fin Si
    Sinon
            insertionSort(T[0..n-1]);
    Fin Si

Fin kiemeRecursif;
```

À la fin de l'exécution de l'algorithme, le k-ième élément se trouve à l'emplacement T[k].

Analyse de complexité :

L'algorithme kiemeRecursif présenté ici a un pire cas en $O(n^2)$, tout comme QuickSort. Cependant, il a un cas moyen en O(n).

Pour nous en convaincre, considérons le cas où chacune des partitions a une taille moitié de celle du tableau en paramètre. L'appel récursif fait en sorte que l'on ne considère, à chaque itération, que la moitié du vecteur précédent. Le temps de générer deux partitions (gauche et droite) pour un tableau de taille n était O(n), nous aurons au total :

$$T(n) = n + n/2 + n/4 + n/8 + \ldots. = n\cdot(1+1/2+1/4+1/8+\ldots).$$

Sachant que :

$\lim_{n\to\infty} \sum_{i=1}^{n} \frac{1}{2^i} = 1$, nous aurons T(n) ≈ n(1 + 1) = 2n.

Ce qui nous donne une complexité en O(n) dans ce cas idéal.

Il est possible de prouver que le cas moyen de l'exécution du kiemeRecursif a une complexité asymptotique O(n), mais cela dépasse le cadre de ce cours (intervention des notions de combinatoire et de probabilités). Le cas idéal d'un partitionnement continuellement égal devrait néanmoins donner une bonne intuition sur la question.