TP4 : Diviser pour régner

Le but de ce TP est d'implanter des algorithmes de type « diviser pour régner » vus en cours. Bien penser à tester vos fonctions avec des tableaux de différents types (aléatoires, croissants, décroissants) et de tailles diverses (de très petit à très grand). **Le fichier à rendre est** `TrisRang.cpp`**.**

On commence par implanter le tri fusion étudié en cours.

1. Compléter la fonction `void fusion(int n1, int n2, int* T1, int* T2, int* T)` qui fusionne les tableaux *triés* `T1` et `T2` de tailles respectives `n1` et `n2` dans le tableau `T`. *On suppose que* `T` *a été correctement alloué à la bonne taille précédemment. Il n'est pas nécessaire de vérifier que les tableaux* `T1` *et* `T2` *sont bien triés.*
2. Compléter la fonction `void trifusion(int n, int* T)` qui trie le tableau `T` en le modifiant. *Il faut déclarer et allouer deux tableaux* `T1` *et* `T2` *qui sont fusionnés à la fin de l'algorithme dans* `T`. *Bien penser à les supprimer en fin de fonction !*

On implante maintenant le calcul de rang, en utilisant un pivot. Le choix de pivot peut être fixé (premier élément du tableau) ou aléatoire. Ce choix est décrit par un booléen (`true` signifiant aléatoire, `false` fixé) et implanté dans la fonction `int pivot(int n, int* T, bool b)` qui est fournie.

3. Compléter la fonction `int rang(int k, int n, int* T, bool b)` qui renvoie le $k^{\text{ème}}$ plus petit élément du tableau `T` de taille `n`.
   Test. Comparer les temps de calcul en fonction du choix de pivot : voit-on une différence sur un tableau aléatoire ? sur un tableau initialement trié (ordre croissant ou décroissant) ?

On implante enfin l'algorithme de tri rapide (non décrit en cours). C'est un algorithme de type « diviser pour régner » basé sur la même idée que le calcul de rang : *(i)* on choisit un pivot p ; *(ii)* on construit les deux tableaux T_{inf} et T_{sup} à l'aide de p ; *(iii)* on trie par appels récursifs ces deux tableaux ; *(iv)* on reconstruit T trié en concaténant trois tableaux : T_{inf} trié, un tableau de n_{eq} fois le pivot p et T_{sup} trié.

4. Compléter la fonction `void trirapide(int n, int* T, bool b)` qui implante cet algorithme (où `b` code toujours le type de pivot, aléatoire ou fixé).
   Test. Comparer les temps calculs selon le type de pivot utilisé, et selon le type de tableau en entrée. Comparer également avec l'algorithme du tri fusion.