Chapitre E.2 : La méthode "diviser pour régner"

L'algorithme de recherche dichotomique, vu en première, permet la recherche d'un élément dans un tableau triée avec une complexité logarithmique. Il consiste à séparer le tableau en deux à chaque étape et à chercher l'élément dans l'un des deux sous-tableaux.
L'approche algorithmique diviser pour régner est une généralisation de la méthode dichotomique.

I. La méthode "diviser pour régner"

Les grandes étapes d'un algorithme suivant le principe diviser pour régner sont :

- **DIVISER** : On découpe le problème à résoudre en plusieurs sous-problème.
- **REGNER** : On résout chacun des sous-problème.
- **COMBINER** : On construit la solution du problème de départ à partir des résultat de chaque sous-problème.

Pour résoudre les sous-problèmes, on les divise à nouveau en sous-problème. Ce qui implique l'utilisation de fonctions récursive .
Par conséquent, il est important de bien traiter le(s) cas de base

II. Exemple : Le tri fusion

Le tri fusion est une méthode de tri qui utilise la méthode diviser pour régner.
Les étapes du tri fusion sont :

- **DIVISER** : Découper le tableau à trié en deux sous-tableaux de taille (environ) égale.
- **REGNER** : Trier les deux sous-tableaux en suivant le même principe.
  Cas de base : un tableau de taille 1 est déjà trié.
- **COMBINER** : Fusionner les deux sous-tableaux triés pour produire le tableau trié complet.

Pour écrire l'algorithme du tri fusion, nous aurons besoin d'écrire deux fonctions, `tri_fusion` et `fusion`.

Fonction `tri_fusion`

```
fonction tri_fusion(t) :
    Si la taille de t est inférieure ou égale à 1
        On renvoie le tableau
    Sinon
        t1 est un tableau contenant la première moitié des valeurs de t
        t2 est un tableau contenant la seconde moitié des valeurs de t
        On renvoie la fusion(tri_fusion(t1), tri_fusion(t2))
```

Fonction `fusion`

```
fonction fusion(t1,t2) :
    Si t1 est vide :
        On renvoie t2
    Sinon si t2 est vide :
        On renvoie t1
    Sinon si le premier élément de t1 < premier élément de t2 :
         On renvoie [premier element de t1] + fusion(t1 ôter de son premier élément,
t2)
    Sinon
           On renvoie [premier element de t2] + fusion (t1, t2 ôter de son premier
élément)
```

III. Complexité du tri fusion.

En première, vous avez vu que le tri par insertion et tri par sélection ont tous les deux une complexité $O(n^2)$. Qu'en est-il pour le tri-fusion ?

Le calcul rigoureux de la complexité de cet algorithme sort du cadre du programme de terminal cours. Mais, en remarquant que la première phase (DIVISER) consiste à "couper" les tableaux en deux plusieurs fois de suite, intuitivement, on peut dire qu'un logarithme base 2 doit intervenir. La deuxième phase consiste à faire des comparaisons entre les premiers éléments de chaque tableau à fusionner, on peut donc supposer que pour un tableau de n éléments, on aura n comparaisons. En combinant ces 2 constatations on peut donc dire que la complexité du tri-fusion est $O(n \times \log2(n))$.

Néanmoins, malgré cette rapidité d'exécution, cet algorithme, en raison des découpages répétés, nécessite de l'espace mémoire supplémentaire qui peut être un facteur critique lorsque le jeu de données à trier est conséquent