ITC – TD n°5
Problème du sous-tableau maximal

On s'intéresse au problème du sous-tableau maximal, qui consiste à trouver un sous-tableau dont la somme des éléments est maximale : par exemple, le tableau `t1 = [3, -1, -4, 3, -2, 5, 3, -3, 1]` a pour sous-tableau maximal `[3, -2, 5, 4]`, compris entre les indices `[3 :7]`, pour une valeur de 9.

1 – Quel est nécessairement le sous-tableau maximal si le tableau ne contient que des entiers positifs ?

I. Une approche brutale

On propose une premiére approche : on peut tester tous les sous-tableaux possibles afin de déterminer le sous-tableau cherché.

2 – Écrire une fonction `sstab_partiel(tab, i)` qui prend en paramètres une liste et un indice valide de la liste, qui calcule l'indice j du tableau tel que `tab[i :j]` soit le sous-tableau maximal débutant en i, et renvoie j et la somme comprise dans ce sous-tableau. Par exemple,

```
In [1]: sstab_partiel(t1, 4)
Out[1]: 7, 6
```

La complexité de cette fonction doit être $\mathcal{O}(n)$, avec n la taille du tableau.

3 – Écrire une fonction `sstab_brute(tab)` qui utilise la fonction précédente pour calculer le sous-tableau maximal par test exhaustif ; elle renvoie les indices i, j et la somme comprise dans le sous-tableau, par exemple

```
In [1]: sstab_brute(t1)
Out[1]: 3, 7, 9
```

Quelle est la complexité de cette fonction ?

II. Diviser-pour-régner

On cherche un algorithme plus subtil pour déterminer le sous-tableau maximal : on coupe le tableau en deux par son milieu m et on cherche :

- un sous-tableau maximal à gauche de m ;
- un sous-tableau maximal à droite de m ;
- un sous-tableau maximal qui chevauche m ;

puis on les compare : celui qui totalise la plus grande somme est gardé comme le sous-tableau maximal recherché. Sur l'exemple `[3, -5, 2, 3, -7, 4, 2]`, on obtient :

- un sous-tableau maximal à gauche `[3]`, indices [0 : 1], valeur 3 ;
- un sous-tableau maximal à droite `[4, 2]`, indices [5 : 7], valeur 6 ;
- un sous-tableau maximal qui chevauche `[2, 3]`, indices [2 : 4], valeur 5 ;

et le sous-tableau maximal est donc `[4, 2]`, obtenu aux indices [5 : 7], de valeur 6.

L'avantage de cette méthode est qu'on peut obtenir les sous-tableaux de gauche et droite récursivement. Nous allons donc commencer par déterminer le sous-tableau maximal qui chevauche le milieu.

4 – Écrire une fonction `sstab_gauche(tab, j)` , qui s'inspire de `sstab_partiel(tab, i)` pour chercher le sous-tableau maximal finissant par l'indice au milieu du tableau.

5 – En utilisant `sstab_partiel(tab, i)` et `sstab_gauche(tab, j)` pour écrire une fonction `sstab_milieu(tab)` qui calcule le sous-tableau maximal contenant l'indice milieu m ; elle renvoie les indices et la valeur totale du sous-tableau obtenu. Justifiez que cette fonction est en $\mathcal{O}(n)$.

6 – En déduire une fonction récursive `sstab_rec(tab)` qui utilise l'algorithme proposé pour déterminer le sous-tableau maximal.

7 – Montrer que la fonction obtenue a pour complexité $\mathcal{O}(n \ln(n))$.