EXERCICE 5 (~~4 points~~)

Cet exercice porte sur la programmation en général.

Étant donné un tableau non vide de nombres entiers relatifs, on appelle sous-séquence une suite non vide d'éléments voisins de ce tableau. On cherche dans cet exercice à déterminer la plus grande somme possible obtenue en additionnant les éléments d'une sous-séquence.

Par exemple, pour le tableau ci-dessous, la somme maximale vaut 18.
Elle est obtenue en additionnant les éléments de la sous-séquence encadrée en gras ci-dessous (6 ; 8 ; -6 ; 10).

-8	-4	**6**	**8**	**-6**	**10**	-4	-4

1. **a.** Quelle est la solution du problème si les éléments du tableau sont tous positifs ?

   b. Quelle est la solution du problème si tous les éléments sont négatifs ?

2. Dans cette question, on examine toutes les sous-séquences possibles.

   a. Écrire le code Python d'une fonction `somme_sous_sequence(lst, i, j)` qui prend en argument une liste et deux entiers `i`, `j` et renvoie la somme de la sous-séquence délimitée par les indices `i` et `j` (inclus).

   b. La fonction `pgsp` ci-dessous permet de déterminer la plus grande des sommes obtenues en additionnant les éléments de toutes les sous-séquences possibles du tableau `lst`.

```python
def pgsp(lst):
    n = len(lst)
    somme_max = lst[0]
    for i in range(n):
        for j in range(i, n):
            s = somme_sous_sequence(lst, i, j)
            if s > somme_max :
                somme_max = s
    return somme_max
```

   Parmi les quatre choix suivants, quel est le nombre de comparaisons effectuées par cette fonction si le tableau `lst` passé en paramètre contient 10 éléments ?

   10   55   100   1055

c. Recopier et modifier la fonction `pgsp` pour qu'elle renvoie un tuple contenant la somme maximale et les indices qui délimitent la sous-séquence correspondant à cette somme maximale.

3. On considère dans cette question une approche plus élaborée. Son principe consiste, pour toutes les valeurs possibles de l'indice i, à déterminer la somme maximale $S(i)$ des sous-séquences qui se terminent à l'indice i.
En désignant par lst$[i]$ l'élément de lst d'indice i, on peut vérifier que

- $S(0) =$ lst$[0]$
- et pour $i \geq 1$:

  $S(i) =$ lst$[i]$ si $S(i-1) \leq 0$;

  $S(i) =$ lst$[i] + S(i-1)$ si $S(i-1) > 0$.

a. Recopier et compléter le tableau ci-dessous avec les valeurs de $S(i)$ pour la liste considérée en exemple.

i	0	1	2	3	4	5	6	7
lst$[i]$	-8	-4	6	8	-6	10	-4	-4
$S(i)$								

b. La solution au problème étant la plus grande valeur des $S(i)$, on demande de compléter la fonction `pgsp2` ci-dessous, de sorte que la variable `sommes_max` contienne la liste des valeurs $S(i)$.

```
def pgsp2(lst):
    sommes_max = [lst[0]]
    for i in range(1, len(lst)):

        # à compléter


    return max(sommes_max)
```

c. En quoi la solution obtenue par cette approche est-elle plus avantageuse que celle de la question **2.b.** ?