Devoir surveillé n°1

Durée : 2H. Calculatrices interdites. **Les candidats sont invités à porter une attention particulière à la rédaction et la propreté : les copies illisibles ou mal présentées seront pénalisées.**
Lisez l'énoncé attentivement et en entier **AVANT** de commencer chaque exercice.

Dans ce sujet, on dira que l'ordre de grandeur du nombre d'opérations nécessaires à l'exécution d'une fonction prenant en argument une liste de longueur n est $f(n)$ lorsque ce nombre d'opération est un $O(f(n))$, c'est à dire est inférieur à $k \times f(n)$ où k est une constante qu'on ne cherchera pas à préciser. On compte dans l'exercice 1 uniquement les opérations arithmétiques entre flottants : $+, \times, -, /$

Exercice 1 (Quelques algorithmes)
Les fonctions définies ici seront réutilisées dans le deuxième exercice. Vous pourrez admettre que chaque fonction a été implémentée avec succès.

Nous allons chercher, dans cet exercice, déterminer pour une liste `L` (contenant des nombres) quel est le maximum des sommes d'éléments consécutifs de `L`. Plus précisément, on cherche la plus grande valeur des sommes des tranches non vides de L de la forme `L[g: d]`, c'est à dire des sommes de la forme $\sum_{k=g}^{d-1} \mathrm{L}[k]$ où $0 \leqslant g < d \leqslant n$ (en notant n la longueur de `L`). On note S(L) la valeur de cette somme : $\mathrm{S(L)} = \max\{\sum_{k=g}^{d-1} \mathrm{L}[k];\ 0 \leqslant g < d \leqslant \texttt{len(L)}\}$
Par exemple $\mathrm{S}([-4, 8, -1, -3, 5, -2])$ vaut 9 et est obtenu pour $g = 1$ et $d = 5$.
L'utilisation de la fonction `sum` de python est interdite ici.

1. Écrire une fonction `max(a, b)` prenant comme paramètres deux flottants (ou entiers) et renvoyant le plus grand de ces deux flottants.
   La fonction `max` ne peut plus être appliquée à des listes ou des vecteurs numpy à partir de maintenant, mais seulement à deux nombres.
2. Nous admettons que la fonction suivante calcule S(L) pour une liste L de nombres.

```
1  def som_max1(L):
2      n = len(L)
3      s_max = L[0]
4      for g in range(n):
5          for d in range(g + 1, n + 1):
6              s = t[g]
7              for k in range(g + 1, d):
8                  s = s + t[k]
9          s_max = max(s_max, s)
10     return s_max
```

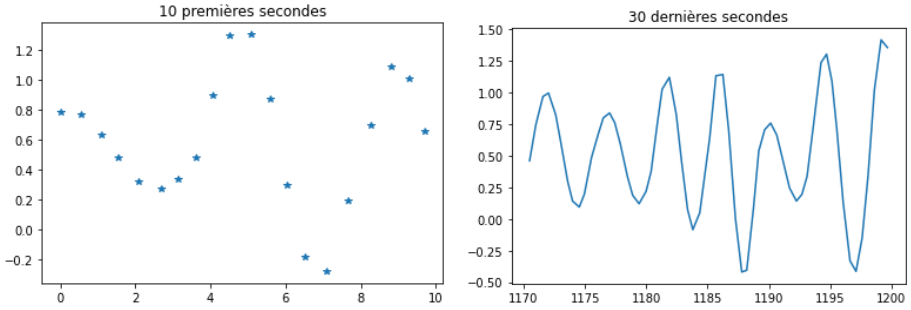
   (a) Expliquer rapidement ce que calculent les lignes numérotées 6 à 8.
   (b) Quelle est l'utilité de la ligne 9 ?
   (c) En notant, comme dans le code, n la longueur de `L`, préciser quelles sont les valeurs prises par `g`, et pour une valeur fixée de `g` quelles sont les valeurs prises par `d`.
   On en déduit que le nombre d'opérations effectuées par `som_max1`, noté $\mathrm{T}_1(n)$, est de l'ordre de grandeur de n^3 (ce qui est admis ici).

3. On cherche à améliorer la rapidité du calcul de S(L) en améliorant l'algorithme précédent.
   (a) Écrire une fonction `som_max_depuis(g, L)` qui calcule, pour un entier `g` fixé, le maximum des sommes d'éléments consécutifs de `L` et dont le premier terme est `L[g]`. La valeur de retour doit donc être $\max\{\sum_{k=g}^{d-1} \mathrm{L}[k];\ g < d \leqslant n\}$ (on a fixé g).
   Pour écrire cette fonction, on utilisera au maximum une boucle `for` qui met à jour à chaque itération une variable que l'on pourra nommer `s_max` au fur et à mesure du calcul de somme.
   (b) Montrer que le nombre d'opérations nécessaires au calcul `som_max_depuis(g, L)` est égal à $n - g - 1$.
   (c) Déduire de la fonction `som_max_depuis` une fonction `som_max2(L)` qui calcule S(L).
   (d) Montrer que l'ordre de grandeur de $\mathrm{T}_2(n)$ (défini comme pour `som_max1`) est cette fois n^2.

4. On remarque (et on admet) que pour une liste `L` de longueur $n \geqslant 2$, S(L) est le plus grands parmi les $n+1$ nombres suivants : $\sum_{k=0}^{d-1} \mathrm{L}[k]$ pour $d \in [\![1, n]\!]$ et S(L[1 : n]) (où `L[1:n]` est la liste des éléments de `L` dont les indices sont les nombres de $[\![1, n-1]\!]$).
   Déduire de cette remarque une fonction **récursive** `som_max3(L)` calculant S(L).

Exercice 2

Dans cet exercice, on s'intéresse à la construction d'une digue [1], et en vue de dimensionner cette digue on s'intéresse à divers paramètres de la houle.
Une bouée munit de capteurs réalise plusieurs campagnes de mesures : elle mesure à intervalle de temps irréguliers la hauteur de la mer, et donc la hauteur des vagues (le point de référence est arbitraire). Un exemple de représentation graphique des données obtenues est :



1. On recueille les données de la bouée par un fichier de texte au format csv. Un extrait de ce que pourrait être un tel fichier est :

```
46.92;+0.669
47.41;+0.158
47.83;-0.219
48.39;-0.452
48.79;-0.364
49.37;+0.077
```

   Les données représentées sont : l'instant de la mesure (en secondes) depuis le début de l'enregistrement ainsi que la hauteur mesurée (en mètres). A titre d'exemple, une telle ligne peut être manipulée comme :

```
>>> s = '46.92;+0.669'
>>> s.split(';')
['46.92', '+0.669']
```

   On souhaite extraire en python les données des mesures d'un enregistrement, stockées dans le fichier `donnees.csv`, et construire ainsi deux listes : la liste des instants des mesures ainsi que la liste des hauteurs correspondantes. Pour cela on propose la structure suivante :

```
1  f = open('donnees.csv')
2  lignes = f.readlines()
3  T = []
4  H = []
5
6  f.close()
```

   Proposez un code python à insérer à la ligne 5 (ce code peut comporter plusieurs lignes !) pour permettre d'obtenir, après évaluation, la liste T des instants de mesure ainsi que la liste H des hauteurs mesurées.
   Indication : Pour convertir une chaîne en nombre, on peut utiliser la fonction `float` qui prend comme argument un chaîne (entre autre) et retourne le nombre flottant correspondant.
2. En supposant construit les listes `T` et `H` et qu'on a exécuté le code

```
1  import matplotlib.pyplot as plt
```

   donner le code python permettant d'afficher le graphique d'évolution de la hauteur de la bouée en fonction du temps.
3. La digue doit pouvoir résister à une succession de hautes vagues. On cherche donc à extraire la pire (au sens que l'on définira à la question 3f) séquence de vagues successives qui a été observée.
   (a) On souhaite tout d'abord estimer les hauteurs maximales de houle successives (correspondant au passage de vagues successives sous la bouée). Pour une liste (de nombres) `L` de longueur n et pour $i \in [\![1, n-2]\!]$, à quel condition la valeur `L[i]` est-elle un maximum local ?

1. imaginez une ville de bord de mer menacée par la montée des eaux

(b) En déduire une fonction `maxi_locaux(L)` qui prend comme argument une liste `L` et retourne la liste des **indices** des maxima locaux de `L`

(c) Donner un code python (pas nécessairement une fonction) permettant de créer `hauteurs_max` et `instants_max`, les listes des hauteurs maximales successives de la houle (les maxima locaux de `H`) et des instants correspondants.

(d) La digue est au minimum conçue pour résister à la moyenne des vagues. Écrire une fonction `moyenne(L)` qui prend comme argument une liste de nombre non vide `L` et retourne la moyenne de ses éléments.

(e) Pour essayer d'estimer l'intensité maximale moyenne, on normalise la liste `hauteurs_max` en soustrayant la hauteur moyenne des vagues à chaque élément. Attention, il ne s'agit pas de la hauteur moyenne de l'eau au cours du temps, mais de la hauteur moyenne atteinte par la crête des vagues : une petite vague aura une hauteur normalisée négative.

Donner une suite d'instructions python permettant de créer `hauteurs_norm` , la liste des hauteurs normalisées.

(f) On calcule ensuite `S(hauteurs_norm)` (voir l'exercice 1) pour obtenir la pire suite de vague supérieures à la moyenne. Pour notre exemple numérique, l'enregistrement dure 20 minutes et on obtient la valeur voulue en 0.1s par l'utilisation de `som_max1` et en 2×10^{-3}s par l'utilisation de `som_max2`.

Donner une estimation des temps de calculs pour chacune de ces deux fonctions pour les données provenant d'un enregistrement d'une heure.

4. Une deuxième donnée intéressante est la vitesse à laquelle l'eau monte au passage d'une vague. Nous allons tenter de l'estimer.

(a) Grâce aux listes `H` et `T`, donner une estimation de $v(t_i)$, la vitesse instantanée (verticale) de la bouée à l'instant t_i (l'instant de la $i^{\text{ème}}$ mesure). Quelle approximation utilisez-vous ?

(b) En déduire une suite d'instruction python permettant de créer la liste `V` des vitesses de la bouée au cours de l'enregistrement. En notant n la longueur commune de `H` et `T`, quelle est la longueur de `V` ?

(c) Expliquer en quelques phrases comment trouver l'accélération maximale de la bouée.

(d) On souhaite maintenant trouver les 10 vagues les plus violentes de notre enregistrement La violence d'une vague est ici mesurée par la valeur absolue de la vitesse instantanée de la bouée (et une vague peut très bien se trouver deux fois dans notre classement : une fois à la montée et une fois à la descente). On considère qu'on a remplacé les vitesses instantanées par leurs valeurs absolues dans `V`

Donner le nom d'un tri que nous pourrions utiliser ainsi que l'ordre de grandeur du nombre d'opérations effectuées pour trier la liste des vitesses en fonction de la longueur de `V` (que l'on pourra noter m).

(e) Comme on ne souhaite trouver que les 10 plus grandes valeurs, on peut penser à un autre algorithme :

- Trouver le plus grand élément de `V` et le placer en première position.
- Recommencer avec les éléments à partir de l'indice 1 (en deuxième position) jusqu'à ce que les 10 plus grands éléments de `V` soient dans l'ordre les 10 premiers éléments de `V`.

Écrire une fonction `top_10(L)` qui prend comme argument une liste `L` et place aux 10 premières positions de `L` ses 10 plus grands éléments, dans l'ordre.

(f) Exprimer, en fonction de n (la longueur de `L`) le nombre de comparaisons effectuées par le calcul de `top_10(L)`. En particulier, on montrera que l'ordre de grandeur est n.

(g) Afin de savoir s'il est plus rapide d'utiliser le tri python ou notre fonction `top_10`, on effectue quelques expériences numériques. On admet que le tri python s'exécute en un temps proportionnel à $n\ln(n)$, alors que `top_10` s'exécute en un temps proportionnel à n.

Pour des listes de longueur n = 1000, la fonction `top_10` est 3 fois plus lente que la fonction de tri de python. Trouver une expression de k tel que la fonction `top_10` devient plus rapide que le tri python pour les listes de longueur kn.

Indication : on exprimera le temps mis par python pour trier une liste de longueur kn en factorisant par le temps mis pour trier une liste de longueur n.

5. La dernière donnée que nous allons calculer est la quantité totale d'eau dépassant le niveau moyen mesuré et qui est passée sous la bouée pendant l'enregistrement.

(a) Expliquer rapidement quelles sont les valeurs contenues dans la liste `H2` construite par le code suivant.

```
1  H2 = []
2  moy = moyenne(H)
3  for elt in H:
4      if elt >= moy:
5          H2.append(elt - moy)
6      else:
7          H2.append(0)
```

(b) Il reste maintenant à calculer l'aire sous la courbe que l'on peut tracer par la commande `plt.plot(T, H2)`.

Rappeler rapidement le principe du calcul d'une intégrale par la méthode des trapèzes et appliquer cette méthode pour obtenir l'aire cherchée.

(c) Préciser l'unité de la quantité calculée précédemment. Ceci ne correspond pas à un volume sur une période donnée. Par quelle quantité faudrait-il multiplier pour estimer la quantité d'eau atteignant la digue (dans la durée de l'enregistrement) au dessus du niveau moyen ?