Mines Informatique MP-PC-PSI 2018 — Corrigé

Ce corrigé est proposé par Jean-Julien Fleck (professeur en CPGE) ; il a été relu par Virgile Andreani (ENS Ulm) et Guillaume Batog (professeur en CPGE).

Le sujet s'inspire d'un problème physique réel (mesures de houle en divers points du globe) et se découpe en cinq grandes parties qui visent chacune à balayer une partie du programme d'informatique commune.

- La première partie s'intéresse au stockage des données récoltées par les instruments de mesure. Elle permet de vérifier que les candidats ont des notions concernant la place occupée en mémoire par un caractère et sur la lecture d'un fichier de données.
- La deuxième partie permet de rentrer plus avant dans le traitement des données en demandant d'implémenter quelques algorithmes simples qui sont au programme de prépa : calcul d'une moyenne, intégration à partir d'une liste de valeurs, recherche d'une particularité dans une liste, calcul de propriétés à partir des spécifications de l'énoncé, etc.
- La troisième partie, après l'implémentation d'une recherche de maximum sur une liste de listes, s'intéresse plus particulièrement au programme de deuxième année avec une petite incursion dans les algorithmes de tri, demandant par exemple de compléter une implémentation préremplie du tri rapide, puis d'écrire entièrement une implémentation d'un tri par insertion. Une analyse de programme est aussi demandée pour y détecter un calcul inefficace et demander de l'améliorer.
- La quatrième partie concerne les bases de données. Il s'agit de survoler rapidement toute une page de présentation de la base de données considérée pour pouvoir écrire les trois requêtes SQL demandées. Là encore, il ne s'agit pas de comprendre *in extenso* tous les tenants et les aboutissants de la base, mais d'identifier les points clés afin de répondre rapidement à la question posée.
- La cinquième partie présente et demande d'implémenter l'algorithme de Cooley-Tukey de transformée de Fourier discrète en imposant une approche récursive (au programme de seconde année). C'est néanmoins tout à fait abordable en première année.

Le sujet est globalement équilibré. Il comporte de nombreuses questions simples de programmation et explore l'ensemble du programme des deux années (avec notamment des questions sur les tris et une implémentation récursive pour le programme de seconde année) mais peut être abordé en grande partie en première année.

Comme chaque année, l'épreuve des Mines nécessite d'être efficace vu sa durée limitée (seulement 1h30). Les questions ne sont pas particulièrement difficiles et il faut savoir aller à l'essentiel pour pouvoir toutes les traiter, quitte à en laisser certaines de côté si elles semblent demander trop de temps de réflexion.

Téléchargé gratuitement sur Doc-Solus.fr .

Indications

Partie I

4 L'ouverture d'un descripteur de fichier se fait à l'aide de la commande `open`. La lecture des lignes dans un tableau se fait à l'aide de `readlines`. Ne pas oublier la conversion en flottant à l'aide de la fonction `float` qui gère naturellement les signes et les caractères de fin de ligne.

Partie II

7 La méthode des trapèzes sur un découpage $(x_i)_{i\in[\![0\,;n]\!]}$ en n intervalles revient à approximer

$$\int_{x_0}^{x_n} f(x)\,\mathrm{d}x \approx \sum_{i=0}^{n-1}\left(\frac{f(x_i)+f(x_{i+1})}{2}\right)\times(x_{i+1}-x_i)$$

La moyenne d'une fonction η sur un intervalle de temps t_{tot} se définit comme

$$\langle\eta(t)\rangle = \frac{1}{t_{\text{tot}}}\int_0^{t_{\text{tot}}}\eta(t)\,\mathrm{d}t$$

8 Penser à calculer la moyenne en dehors de la boucle pour éviter de refaire de nombreuses fois le même calcul.

9 Le calcul de la moyenne est déjà de complexité linéaire : il faut que celle-ci soit fournie en argument de la fonction si on veut espérer atteindre une complexité en $\mathrm{O}(1)$ dans le meilleur des cas.

10 Attention, la définition des indices qu'il faut stocker dans la liste des successeurs n'est plus celle des questions 8 et 9 : il y a un décalage d'une position supplémentaire.

11 Penser à utiliser la fonction précédente pour calculer la liste des successeurs et utiliser le *slicing* pour découper en tranches correspondant aux vagues.

12 Utiliser de même la fonction définie à la question précédente pour obtenir les différentes vagues. Attention au fait que la hauteur maximale se calcule à cheval sur deux vagues (maximum de l'une auquel on soustrait le minimum de la suivante). Attention aussi au fait qu'il faut calculer le maximum précédant le premier PND.

Partie III

13 La structure particulière de la liste passée en argument impose d'implémenter la recherche de maximum à la main.

14 Ne pas essayer de comprendre ce que fait le code en détail, il suffit de trouver « par homogénéité » la forme attendue pour le pivot.

16 Penser à la manière dont on trie ses cartes en propageant la place libre jusqu'à atteindre le bon emplacement.

17 Recalculer de nombreuses fois la même chose ne sert à rien : penser à calculer la moyenne une unique fois avant de rentrer dans la boucle.

Téléchargé gratuitement sur Doc-Solus.fr .

Partie IV

19 La deuxième requête peut se faire en comptant le nombre d'apparition d'une bouée dans la table `Tempete`, ledit nombre devant être égal à 0 si la bouée n'a pas connu de tempête. On peut aussi utiliser une différence ensembliste à l'aide de `EXCEPT` ou `MINUS`.

La troisième requête se fait assez simplement avec une jointure et une unique fonction d'agrégation.

Partie V

20 On peut estimer la complexité en procédant d'une manière proche du tri fusion : on a $k = \log_2(\mathrm{N})$ étages de divisions par 2 avec une complexité linéaire à chaque étage.

Téléchargé gratuitement sur Doc-Solus.fr .

I. Stockage interne des données

1 Hormis la première ligne qu'on demande d'ignorer, chaque ligne est constituée de 8 caractères (5 chiffres, un signe, un point et le caractère de fin de ligne comme le signale l'énoncé) donc occupe 8 octets. 20 minutes correspondent à 1 200 s. En outre, comme il y a deux mesures par seconde, cela correspond à 2 400 mesures au total, soit une taille de $8 \times 2\,400 = 19\,200$ octets $= 19{,}2$ ko.

2 La campagne de mesure fait état d'un fichier récolté toutes les demi-heures pendant 15 jours. Il y a donc $15 \times 24 \times 2 = 720$ fois les informations précédentes collectées, ce qui représente donc une taille totale de 13 824 ko, soit 13,8 Mo.

Une carte mémoire de 1 Go est largement suffisante.

3 Ôter un chiffre revient à passer chaque ligne de 8 à seulement 7 octets, d'où une réduction de $1/8 \approx 12\%$ de la taille totale du fichier.

4 La lecture d'un fichier texte peut se faire de la manière suivante

```
with open('donnees.txt') as f:# Ouverture du descripteur de fichier
    L = f.readlines()         # Lecture effective du fichier
    liste_niveaux = []        # Définition du conteneur
    for i in range(1,len(L)): # On saute la première ligne de texte
        liste_niveaux.append(float(L[i])) # et on convertit en flottants
```

Bien sûr, on peut aussi demander à Numpy de faire tout le travail à notre place

```
import numpy as np # Importation de la bibliothèque
# On saute la première ligne et on convertit au vol en liste car np.loadtxt
# renvoie normalement un np.array
liste_niveaux = list(np.loadtxt('donnees.txt',skiprows=1))
```

La lecture de données numériques dans un fichier est tellement utile en pratique (par exemple en TIPE) qu'il faut que ce soit maîtrisé par les candidats. Ici, l'usage de la fonction `float` n'est pas rappelé, mais on peut partir du principe qu'elle a été conçue de manière à ce que tous les nombres représentés de manière « usuelle » puissent se convertir facilement depuis une représentation en chaîne de caractères. En particulier, le `+` n'est pas gênant en début de nombre, ni le caractère de fin de ligne (ou tout autre espace qui pourrait être présent tout à droite ou tout à gauche de la chaîne de caractères). En revanche, `float` n'est pas capable d'effectuer des opérations de calcul comme « `float("2+2")` », il ne faut tout de même pas trop lui en demander.

La construction avec `with` permet de s'assurer que le descripteur de fichier sera fermé quoi qu'il arrive et que le fichier ne soit pas corrompu (voir par exemple `https://tinyurl.com/with-open-python`), mais la structure suivante sera acceptable de la même manière.

```
f = open('donnees.txt')   # Ouverture du descripteur de fichier
L = f.readlines()         # Lecture effective du fichier
liste_niveaux = []        # Définition du conteneur
for i in range(1,len(L)): # On saute la première ligne de texte
    liste_niveaux.append(float(L[i])) # et on convertit en flottants
f.close()                 # Fermeture du descripteur de fichier
```

Téléchargé gratuitement sur Doc-Solus.fr .