Concours Blanc : épreuve d'Informatique

PTSI A et B Lycée Eiffel

19 mai 2016

Durée : 2H.

Imprimantes (d'après Mines 2015).

Une entreprise est spécialisée dans la fabrication d'imprimantes. Nous allons nous intéresser dans ce sujet au traitement de données issues de tests de fonctionnement effectués sur ces imprimantes, sans chercher à décrire précisément ces tests. Le sujet est constitué de cinq parties indépendantes que le candidat pourra traiter dans le désordre s'il le souhaite. Au sein d'une partie, on pourra utiliser les résultats supposés des programmes des questions précédentes même si on n'a pas réussi à écrire les programmes correspondants.

Première partie : Extraction de données issues d'un fichier texte.

À l'issue d'une série de mesures effectuées lors d'un test sur une imprimante, les résultats sont stockés dans le fichier texte **mesures.txt** avec les conventions suivantes :

- chaque ligne du fichier est une chaine de caractères (sans séparateurs ni espaces) correspondant à une mesure différente.
- le premier caractère de chaque ligne est une lettre indiquant le type de mesure : U pour la tension du moteur, I pour le courant du moteur (les autres possibilités n'interviendront pas dans cet énoncé).
- les trois caractères suivants indiquent le nombre de données constituant la mesure (par exemple '012' pour indiquer une série de 12 valeurs).
- les données sont ensuite affichées successivement par blocs de quatre caractères : un caractère de signe + ou - suivi de trois caractères numériques.
- enfin, les quatre derniers caractères constituent un **checksum**, valeur numérique permettant de vérifier que les données ont été correctement transmises, et qui est égal à la somme des valeurs absolues des données.

Par exemple, si la première ligne du fichier est 'U005+012+004-023-002+0420083', cela signifie qu'on a mesuré la tension du moteur, qu'il y a cinq données sur cette ligne, dont les valeurs sont 12, 4, -23, -2 et 42 ; et enfin que la somme des valeurs absolues de ces cinq valeurs devrait être égale à 83 (ce qui est bien le cas ici).

1. Les mesures effectuées sont codées sur 10 bits en complément à 2. Quelle est l'intervalle des valeurs possibles pour nos mesures ?
2. Si on considère que les mesures effectuées pour la tension doivent correspondre à un intervalle allant de -5V à 5V, quelle est la résolution de la mesure (intervalle minimal entre deux mesures différentes de la tension) ? Une valeur exacte n'est pas nécessairement attendue, un ordre de grandeur suffira.
3. Quelle est la commande Python permettant de lire exactement k caractères d'un fichier texte ?
4. Écrire une fonction Python permettant d'ouvrir le fichier mesures.txt, d'en extraire la première ligne, et d'en recopier le contenu dans une liste au format suivant : ['U',[12,4,-23,-2,42],83] (on a repris ici l'exemple donné un peu plus haut).
5. Expliquer comment modifier la fonction précédente pour qu'elle ne stocke la ligne que si elle correspond à une mesure de courant (premier caractère 'I'), et uniquement si le checksum est correct.
6. On suppose créée une liste **listemesures** de listes de mesures de l'intensité (avec checksum correct) au format précédent (chaque élément de cette grosse liste ressemble donc à l'exemple de la question 4). Écrire un programme Python permettant de créer à partir de cette liste une nouvelle liste **mesurescourant** ne contenant que les données (sans le caractère initial ni le checksum) correspondant à toutes ces mesures. On ne distinguera pas les différentes séries de mesures : si par exemple on a une première ligne consituée d'une série de cinq mesures et une deuxième constituée d'une série de trois mesures, on veut juste obtenir la liste concaténée des huit mesures effectuées.
7. À l'aide du module matplotlib.pyplot, écrire un programme permettant de représenter graphiquement l'évolution du courant au cours du temps. On suppose que les mesures extraites ont été effectuées régulièrement avec un pas de temps de 2 milli-secondes, et que l'unité de mesure du courant est de 4 milli-ampères.

Deuxième partie : Analyse des mesures.

On suppose dans cette partie qu'on dispose de la liste **mesurescourant** de toutes les données concernant le courant du moteur, telle que décrite dans la partie précédente. Pour valider le fonctionnement de l'imprimante, on a besoin de calculer les deux valeurs suivantes :

- La valeur moyenne du courant $I_m = \dfrac{1}{t_f}\displaystyle\int_0^{t_f} I(t)\ dt$.
- L'écart-type du signal $I_{ec} = \sqrt{\dfrac{1}{t_f}\displaystyle\int_0^{t_f} (I(t) - I_m)^2}$.

1. Écrire un programme Python calculant une valeur approchée de I_m à l'aide de la méthode des trapèzes (le pas de temps étant bien sûr l'écart de temps correspondant à deux valeurs successives de la liste **mesurescourant**).
2. Écrire un programme Python calculant une valeur approchée de I_{ec} en utilisant le programme précédent.

Troisième partie : Bases de données.

On souhaite stocker les résultats de nos tests dans une base de données constituée des deux tables suivantes :

- une table **imprimantes** contenant les attributs **Num** (un numéro d'identifiant de l'imprimante), **n°serie** (chaine de caractères décrivant le modèle d'imprimante), et **dateprod** (date de production de l'imprimante).
- une table **tests** contenant les attributs **n°serie** (même attribut que dans la table imprimantes), **datetest** (date du teste), **Im**, **Iec** (valeurs obtenues pour les données décrites dans la deuxième partie de l'énoncé) et **fichiermes** (chaine de caractères donnant le nom du fichier de mesures correspondant).

1. Écrire une commande SQL permettant de créer la table **tests** (en donnant des types cohérents à chacun des attributs).
2. Écrire une commande SQL permettant d'obtenir la liste des numéros (attribut Num) des imprimantes ayant été testées le 19 mai 2016.
3. Écrire une commande SQL permettant d'obtenir la liste des numéros de série des imprimantes ayant une valeur de I_m comprise entre deux bornes I_{min} et I_{max} fixées.
4. Écrire une commande SQL permettant d'obtenir les numéros de série, la valeur de I_{ec} et le fichier de mesures des imprimantes ayant une valeur de I_{ec} strictement inférieure à la moyenne de toutes les valeurs de I_{ec}.

Quatrième partie : Simulation physique.

Les données obtenues lors d'un test effectué sur l'imprimante peuvent être modélisées par une fonction s vérifiant l'équation différentielle $\frac{ds}{dt} = -k\frac{s(t) - \sin(t)}{10}$, où k est un paramètre strictement positif.

1. Écrire une fonction Python effectuant une résolution approchée de cette équation par la méthode d'Euler. On impose les conditions suivantes : la résolution s'effectuera sur l'intervalle $[0, 10]$, avec un pas de temps de 0.1, et on a la condition initiale $s(0) = 0$.
2. On dispose de la liste **sortie** des données obtenues pour la fonction s lors du test réel sur un intervalle de temps de 10 secondes (la liste est donc constituée de 101 valeurs) et de la liste **euler** des valeurs théoriques prises par la fonction s (obtenues par exemple à l'aide du programme de la question précédente). Écrire une fonction Python permettant de vérifier la cohérence des résultats avec le modèle théorique en utilisant le critère suivant : il doit y avoir au maximum 10% d'écart relatif entre les valeurs correspondantes de chacune des deux listes.

Cinquième partie : Compression des données.

Pour effectuer le transfert des données de façon optimisée, on utilise en général des techniques de compression de données. Nous allons dans cette dernière partie décrire une méthode de codage en binaire des caractères contenus dans un texte qui est fréquemment utilisée dans le domaine de la compression de données (cette dernière partie est assez indépendante de nos histoires d'imprimantes). On suppose pour simplifier qu'on dispose d'un texte (chaine de caractères) ne contenant que des lettres minuscules et des espaces. On rappelle que la fonction Python **ord** donne le code ASCII d'un caractère (le code du a minuscule étant 97, celui du z minuscule $97 + 25 = 122$, et celui de l'espace 32).

1. Rappeler en quoi consiste le standard ASCII.
2. Écrire une commande Python permettant de créer une liste **frequence** contenant 27 éléments tous égaux à 0.

3. Écrire une fonction Python **freq(c)** prenant comme argument une chaine de caractères c, et remplissant la liste frequence avec le nombre d'apparitions de chacun des caractères dans le texte (on indiquera le nombre d'espaces dans frequence[0], celui de 'a' dans frequence[1] etc).
4. Écrire une fonction Python permettant de calculer la proportion d'espaces, ainsi que la proportion de voyelles, dans un texte (le 'y' sera considéré comme une voyelle).

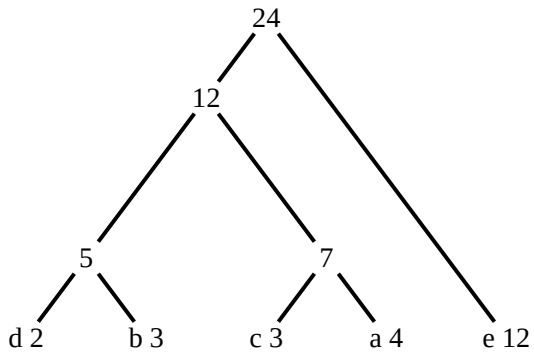
À partir de la liste des fréquences d'apparition des caractères d'un texte, on construit un arbre (appelé arbre de Huffman) de la façon suivante : on part de la liste des élements ayant une fréquence d'apparition non nulle, on regroupe à chaque étape les deux éléments ayant les fréquences les plus faibles pour créer un noeud dont la fréquence sera la somme de celles des deux éléments dont il est issu. On recommence jusqu'à n'avoir plus qu'un seul noeud. Ainsi, imaginons que les fréquence d'un texte soient données par la liste suivante (dans l'ordre croissant, c'est plus facile pour créer l'arbre) :

d 2 $\quad$ b 3 $\quad$ c 3 $\quad$ a 4 $\quad$ e 12

On obtient l'arbre à l'aide des étapes suivantes :

- on regroupe d et b pour créer un noeud de fréquence 5.
- on regroupe c et a pour créer un noeud de fréquence 7.
- on regroupe les deux noeuds qu'on vient de créer pour en faire un nouveau de fréquence 12.
- on regroupe ce dernier noeud avec le e, et l'abre est terminé.

L'arbre aura donc l'allure suivante :



Une fois cet arbre construit, on associe à chaque caractère un code binaire en procédant de la façon suivante : on part du sommet de l'arbre et on descend jusqu'au caractère, en écrivant un 0 à chaque fois qu'on prend à gauche et un 1 à chaque fois qu'on prend à droite si on croise un embranchement. Ainsi, le dans notre exemple, le a sera codé par la séquence 011 (gauche puis droite puis droite), le e sera simplement codé par 1.

1. Donner la liste des codes de tous les caractères de l'exemple donné.
2. Décoder (toujours avec le même exemple) le texte ayant le code suivant : 01001100110001 (ça n'est pas censé avoir un sens). Peut-on avoir deux décodages différents pour un même code ?
3. Si on part d'un texte contenant 27 caractères (les 26 lettres de l'alphabet et l'espace), combien de bits au minimum utilisera-t-on pour coder le caractère ayant le code le plus long ? Et combien au maximum ?
4. Écrire l'arbre de Huffman complet, puis donner la liste des codes correspondant à chaque caractère (n'oubliez pas l'espace) pour le texte suivant : 'le python est tellement trop chouette que je programme trois heures tous les soirs'