Concours Blanc : corrigé

PTSI A et B Lycée Eiffel

19 mai 2016

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

1. Le complément à deux sert à coder des entiers relatifs, ici 10 bits vont permettre de coder les entiers de l'intervalle $[-2^9, 2^9 - 1]$, soit $[-512, 511]$.
2. Puisqu'on a environ 1 000 valeurs pour un intervalle de 10 V, la résolution est d'à peu près 0.01 V.
3. Il s'agit de la commande **read(k)**.
4. Plutôt qu'une fonction, on va écrire un morceau de programme :

```
f=open('fichier.txt','r')
c=f.readline()
l=[]
l.append(c[0])
n=int(c[1 :4])
l2=[]
for i in range(1,n+1) :
    s=c[4*n]
    if s=='+' :
        l2.append(int(c[4*n+1 :4*n+4]))
    else :
        l2.append(-int(c[4*n+1 :4*n+4]))
l.append(l2)
l.append(int(c[4*n+4 :]))
f.close()
```

5. Il faut rajouter une première condition :

```
if c[0]=='I'
```

   et une deuxième qui sera plus facile à vérifier une fois la liste l constituée :

```
if l[2]==sum(abs(i) for i in l[1])
```

6. Là, c'est simple, il s'agit juste de concaténer les sous-listes se trouvant en deuxième position de toutes les listes de mesures :

```
mesurescourant=[]
for i in listemesures :
    mesurescourant=mesurescourant+i[1]
```

7. Il faut créer la liste des valeurs du temps avant de tracer la courbe, on prendra les milliampères et les millisecondes comme unités respectives des deux axes :

```
import matplotlib.pyplot as plt
temps=range(0,len(mesurescourant),2)
mesurescourant=[4*i for i in mesurescourant]
plt.plot(temps,mesurescourant)
plt.show()
```

Deuxième partie : Analyse des mesures.

1. Par rapport aux notations du cours, le pas de temps h est donc égal à 2, et on dispose déjà de la liste des valeurs prises par la fonction aux temps correspondants (on suppose que les valeurs ont été multipliées par 4 comme nous l'avons fait dans la partie précédente pour obtenir les valeurs correctes en milliampères). Il suffit donc, comme le prévoit l'algorithme de la méthode des trapèzes, d'additionner toutes ces valeurs, en mettant un facteur $\frac{1}{2}$ sur les valeurs extrêmes, avant de diviser le tout par la longueur de l'intervalle de temps :

```
def moy(l)
    I=(l[0]+l[-1])/2
    for i in range(1,len(l)-1) :
        I=I+l[i]
    return (I/(2*(len(l)-1)))
print(moy(mesurescourant))
```

2. On va créer une nouvelle liste contenant les carrés des écarts à la moyenne puis appliquer le programme précédent pour calculer l'intégrale :

```
Im=moy(mesurescourant)
ecarts=[(i-Im)**2 for i in mesurescourant]
print(sqrt(moy(ecarts)))
```

Troisième partie : Bases de données.

1. CREATE TABLE tets (idtest INTEGER PRIMARY KEY, n°serie VARCHAR(20), datetest DATE, Im FLOAT, Iec FLOAT, fichiermes VARCHAR(30))
2. SELECT Num FROM imprimantes, tests WHERE imprimantes.n°serie=tests.n°serie AND datetest=2016-05-19
3. SELECT n°serie FROM teste WHERE Im BETWEEN Imin AND Imax
4. SELECT n°serie, Iec, fichiermes FROM tests WHERE Iec < (SELECT AVG(Iec) FROM tests)

Quatrième partie : Simulation physique.

1. On peut reprendre le programme vu en cours en remplaçant simplement la fonction décrivant l'équation par celle de l'énoncé, et en donnant immédiatement les bonnes conditions initiales. On suppose que le module matplotlib.pyplot est resté importé depuis le programme de la première partie.

```
from math import sin
```

```
t,s,h=0,0,0.1
temps,sortie=[],[]
for i in range(101) :
    s=s-0.1*k*(s-sin(t))/10
    t=t+0.1
    temps.append(t)
    sortie.append(s)
plt.plot(temps,sortie)
```

2. Il suffit donc de comparer élément par élément les valeurs prises et de renvoyer un message dès qu'il y a un écart relatif de plus de 10%. Tant qu'à faire, on peut essayer d'arrêter les comparaisons dès qu'on en a une qui donne un écart trop grand :

```
b=True
w=len(sortie)
n=0
while b==True and n<w :
    if abs((euler[i]-sortie[i])/euler[i]) >0.1 :
        b=False
if b==True :
    return ('Les résultats sont cohérents')
else :
    return ('Les résultats ne sont pas cohérents')
```

Cinquième partie : Compression des données.

1. Le standard ASCII (pour **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange) est une norme de codage des caractères initialement destinée à coder les 128 caractères (lettres, signes de ponctuation, etc) les plus utilisés aux Etats-Unis.
2. Pour faire très bête : 27*[0]
3. Il faut faire attention à ne pas faire bugguer le programme si le caractère croisé n'est ni un espace ni une lettre. On peut aussi décider de compter les lettres majuscules (pour les accents, par contre, ça va être difficile).

```
def freq(c) :
frequence=27*[0]
for i in c :
    o=ord(i.lower())
    if o==32
        frequence[0]+=1
    elif 96<o and o<123 :
        frequence[o-96]+=1
return frequence
```

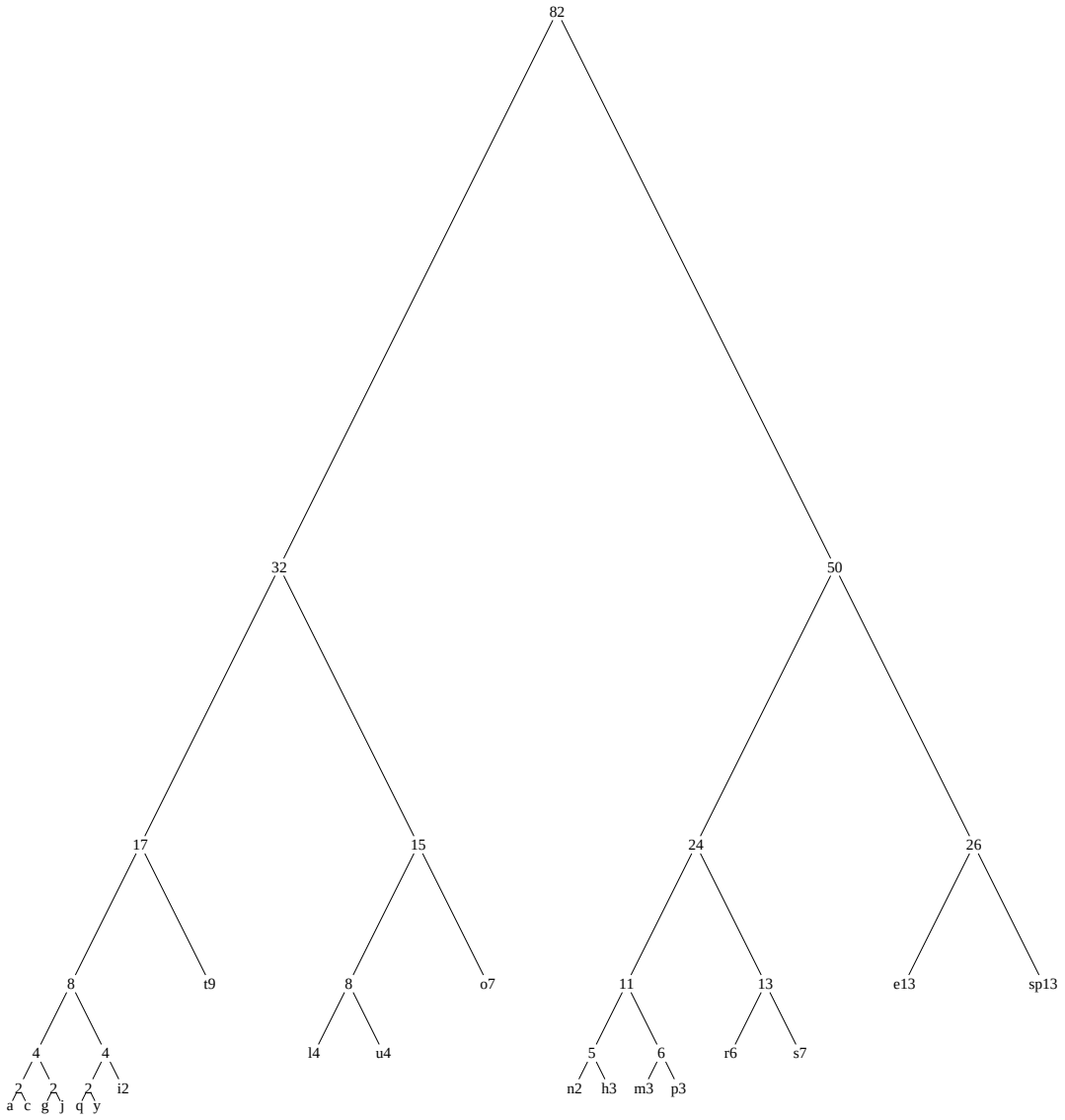
4. Pour la proportion d'espaces,, il suffit de calculer freq(c)[0]/len(c). Pour la proportion de voyelles, en mode un peu brutal, on calcule (freq(c)[1]+freq(c)[5]+freq(c)[9]+freq(c)[15]+freq(c)[21]+freq(c)

1. Le 'a' a pour code 011, le 'b' a pour code 001, le 'c' a pour code 010, le 'd' a pour code 000 et le 'e' a pour code 1.

2. On décode le passionnant texte 'cabede'. Le décodage est unique (on passe à la lettre suivante quand on atteint le bout d'une branche de l'arbre, il ne peut pas y avoir d'ambiguïté), par contre on ne peut pas savoir facilement la longueur du texte à partir de son code.
3. Il faut au minimum 27 caractères, ce qui nécessite 5 bits d'information (puisque $2^5 = 32 > 27$). Au pire, on utilise un bit pour le premier caractère, deux pour le deuxième, etc, jusqu'à 26 pour les deux derniers caractères.
4. Après un laborieux travail de comptage, on obtient les fréquences suivantes (82 caractères au total) :

espace	a	c	e	g	h	i	j	l	m	n	o	p	q	r	s	t	u	y
13	1	1	13	1	3	2	1	4	3	2	7	3	1	6	7	9	4	1

Il ne reste plus qu'à construire l'arbre sans tricher, ce qui donne l'horreur suivante :



On peut alors terminer avec la liste des codes :

- espace : 111
- e : 110
- s : 1011
- r : 1010

- p : 10011
- m : 10010
- h : 10001
- n : 10000
- o : 011
- u : 0101
- l : 0100
- t : 001
- i : 00011
- y : 000101
- q : 000100
- j : 000011
- g : 000010
- c : 000001
- a : 000000

Et il est vraiment temps d'aller se coucher...