Sage

Sage est un logiciel libre pour les mathématiques pures et appliqués, algèbre, calcule symbolique, théorie des nombres, théorie des codes, cryptographie, calcule numérique, algèbre commutative, théorie des groupes, combinatoire, théorie des graphes, et algèbre linéaire.

Les binaires et la source pour Sage (sous Linux, MacOSX, et Windows), et documentation sur l'usage se trouve sur le site :

`http://www.sagemath.org`

Dans les salles d'informatique, nous utiliserons le serveur publique de Sage, qui se trouve à

`http://www.sagenb.org`

Il faut créer un compte avant de commencer.

Semestre 2 **Exercices pour TP 1 (semaine 3)** 2010–2011

1. Pour avoir les fonctions spéciales pour ce cours, il faut lire leurs définitions sur le site du cours :

```
MAT16 = "http://iml.univ-mrs.fr/~kohel/tch/MAT16/"
exec(open(get_remote_file(MAT16 + "MAT16.sage")).read())
```

Il est possible de créer les ensembles des mots $\{\mathtt{A}, \mathtt{B}, \ldots, \mathtt{Z}\}^*$ et $\{0,1\}^*$:

```
S = AlphabeticStrings() # = {A,B,...,Z}^*
B = BinaryStrings() # = {0,1}^*
m = S("AAACDCBDCEAADADACEDAEADCABEDADDCECAAAAAD")
print m
```

```
AAACDCBDCEAADADACEDAEADCABEDADDCECAAAAAD
```

Nous pouvons créer un espace de probabilité avec les probabiités suivantes :

$$p(\mathtt{A}) = p(\mathtt{B}) = 0.125,\ p(\mathtt{C}) = 0.250,\ \text{et}\ p(\mathtt{D}) = 0.500,$$

comme espace de probabilités. La manière la plus simple de créer cet espace est en tant que fréquences des caractères dans le mot `ABCCDDDD` :

```
s = S("ABCCDDDD")
X = s.frequency_distribution()
print X
```

```
Discrete probability space defined by {A: 0.125000000000000,
C: 0.250000000000000, B: 0.125000000000000, D: 0.500000000000000}
```

```
Alph = [ S(c) for c in 'ABCD' ]
print "Valeurs:"
for c in Alph:
    print (c,X(c))
```

```
Valeurs:
(A, 0.125000000000000)
(B, 0.125000000000000)
(C, 0.250000000000000)
(D, 0.500000000000000)
```

On peut calculer l'entropie de cet espace :

```
X.entropy()
```

```
1.75000000000000
```

et vérifier que c'est en accord avec sa définition :

```
prob = [ X(S(c)) for c in "ABCD" ]
sum([ -p*log(p,2) for p in prob ])
```

```
1.75000000000000
```

2. Soit $X = \{\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}, \mathtt{E}, \mathtt{F}, \mathtt{G}, \mathtt{H}\}$ l'espace de probabilité avec les probabilités suivantes :

x	A	B	C	D	E	F	G	H
$p(x)$	0.25	0.25	0.20	0.14	0.10	0.04	0.01	0.01

Cet espace de probabilité peut être créé comme ci-dessous :

```
Alph = [ S.gen(i) for i in range(8) ]
# La liste des probabilites
prob = [ 0.25, 0.25, 0.20, 0.14, 0.10, 0.04, 0.01, 0.01 ]
Prob = {} # un dictionaire de caractere -> probabilite:
for i in range(8):
   Prob[Alph[i]] = prob[i]
X = DiscreteProbabilitySpace(Alph, Prob)
```

et une variable aléatoire pour les longueurs comme ci-dessous :

```
long = [ 1, 2, 3, 4, 4, 5, 6, 6 ] # liste des longeurs:
Long = {} # un dictionaire caractere -> longeur:
for i in range(8):
   Long[Alph[i]] = long[i]
Y = DiscreteRandomVariable(X,Long)
```

a. Calculer l'entropie $H(X)$.

b. Contruire l'arbre de codage associé à un codage de Huffman C pour X, et donner les codes $C(x)$ pour chaque élément x.

c. Trouver l'espérance mathématique de la fonction longueur de $C(x)$, et vérifier le théorème *Codage source* de Shannon. Comparer les longueurs de votre codage et les valeurs $-p(x)\log_2(p(x))$.

d. Comparer les longueurs de mots dans X et leurs codages dans $\{0,1\}^*$.

L'espérance se trouve avec la fonction `expectation` :

```
Y.expectation()
```

et les codages peuvent être réalisés en Sage :

```
ABCD = [ S(x) for x in "ABCD" ]
UVWX = [ B("0"), B("10"), B("110"), B("111") ]
C = Codage(ABCD,UVWX)
C.code(S("ABCCDDDD"))
```

```
01011011011111111111
```

3. Soit $\mathcal{A} = \{\mathtt{A}, \mathtt{B}, \mathtt{C}\}$, et supposons que les diagrammes suivent les probabilités :

		y		
	$p(xy)$	A	B	C
x	A	0	4/15	1/15
	B	8/27	8/27	0
	C	1/27	4/135	1/135

a. Calculer les probabilités $p(x)$ pour chaque lettre x dans $\mathcal{A}$.

b. Déterminer $H(\mathcal{A}^2)$, $H(\mathcal{A})$ et vérifier $H(\mathcal{A}^2) \leq 2H(\mathcal{A})$.

c. Trouver des codages $C_1 : \mathcal{A} \longrightarrow \{0,1\}^*$, et $C_2 : \mathcal{A}^2 \longrightarrow \{0,1\}^*$, qui sont uniquement decodables, avec l'espérance des longueurs bornée par $H(\mathcal{A}) + 1$ et $H(\mathcal{A}^2) + 1$.

d. Trouver les longueurs des codes $C_1(M)$ et $C_2(M)$ où M est

ABBABABABABABABBBABBBBBABABABABABBBACACABBABBBBABBABACBBBABA.