Exercice 1

On considére des codes de longueur fixe.

1. Pour $X = \{a, b, c\}$ on note $P(X) = \{\{\}, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$. Proposer un encodage en binaire de l'ensemble $P(X)$ tel qu'on puisse reconstruire l'ensemble à partir de son encodage.
2. Soit $X = \{a, b, c, d, e\}$. Combien y a-t-il de mots possibles de longueur n ? Combien y a-t-il de mots possibles de longueur $< n$?
3. On pose $X = \{0, 1, \ldots, 9, A, B, \ldots, F\}$ et $Y = \{0, 1\}$. Quelle doit être au minimum la taille des mots d'un codage de taille fixe de l'ensemble X par des mots sur Y ? Combien y a-t-il de tels codages de taille fixe minimale ? Donner le nom d'un tel code de taille fixe de X sur Y bien connu. Dans ce code, quelle est l'écriture de ABC ? Quel mot est codé par 00111010 ?

Exercice 2

On considère des codes de longueur variable.

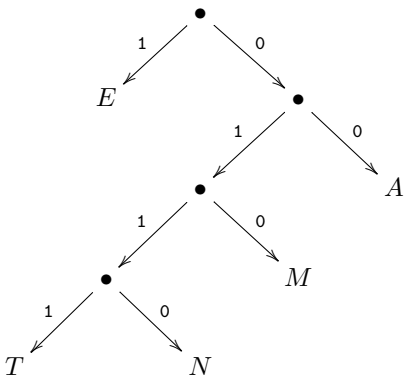
1. Considérons $X = \{a, b, c, d, e\}, Y = \{0, 1\}$ et le codage suivant :

$$\tau(a) = 0 \quad \tau(b) = 10 \quad \tau(c) = 01 \quad \tau(d) = 110 \quad \tau(e) = 1011.$$

   Encoder *babc* et *bcac*. Quel est le problème ?
2. Un code préfixe (respectivement, suffixe) est un ensemble de mots tel qu'aucun codage d'une lettre (de l'alphabet de départ) n'est le préfixe (respectivement, suffixe) du codage d'une autre lettre (de l'alphabet de départ).
   (a) Soit $\sigma(a) = 101000, \sigma(b) = 01, \sigma(c) = 1010$. Est-ce que τ est un code préfixe ?
   (b) Le code σ est-il suffixe ?
   (c) Tout code préfixe est uniquement déchiffrable. Est-ce que la réciproque est vraie ?
   (d) Que penser de $\{01, 100, 1101, 0111\}$? de $\{01, 101, 110, 1110, 0100\}$? de $\{0, 11, 01110, 10101\}$?

Exercice 3

1. En utilisant l'arbre de Huffman suivant, déchiffrer `0 0 0 1 1 0 0 1 1 1 1 0 1 1 0 0 1 1 0 1`.



2. Cet arbre a été construit à partir des fréquences de lettres suivantes :

$$f(E) = 40\%; \quad f(A) = 30\%; \quad f(M) = 20\%; \quad f(N) = 5\%; \quad f(T) = 5\%.$$

   Calculer la taille moyenne d'une lettre. Est-ce bénéfique par rapport à un code de taille fixe ?

3. Construire l'arbre de Huffman donnant un code optimal pour l'alphabet $\{r, s, t, u, v, w\}$ avec :

$$f(r) = 30\%; \quad f(s) = 25\%; \quad f(t) = 15\%; \quad f(u) = 15\%; \quad f(v) = 10\%; \quad f(w) = 5\%.$$

Exercice 4

On considère une source qui émet continûment des `0` avec une probabilité $p_0 = \frac{1}{4}$ et des `1` avec une probabilité $p_1 = \frac{3}{4}$. On désire utiliser la méthode de Huffman pour compresser l'information reçue. Pour cela on regroupe par blocs de deux bits : `00`, `01`, `10` et `11`. On calcule la probabilité que chaque bloc apparaisse et on applique la méthode de Huffman.

1. Calculer le taux de compression τ_2.
2. On regroupe maintenant par blocs de 3 bits : calculer le taux de compression τ_3.

Exercice 5

1. Le codage RLE remplace une suite de `0` et de `1` par une suite de longueurs : on remplace chaque plage de `0` (ou de `1`) par sa longueur. Utiliser cet encodage sur la séquence suivante :

   `0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0`

2. Utiliser la méthode de Huffman sur la suite de longueurs obtenue. Quelle est la séquence obtenue ? Quel est le taux de compression ?

Exercice 1

On s'intéresse au test suivant en utilisant le standard IEEE 754 sur 16 bits (1 bit de signe, 5 bit pour l'exposant et 10 bits pour la mantisse) : `( 0.9 - 0.8 ) == ( 0.8 - 0.7 )`.

1. Quelle est la valeur binaire de `0.9` ?
2. Écrire en binaire `0.8f` et `0.7`.
3. Calculer alors `0.9 - 0.8` et `0.8 - 0.7` en utilisant leurs représentations binaires.

Vérifier la réponse sur la machine en utilisant le standard IEEE 754 sur 32 bits.

Dans les exercices suivants, utilisez la fonction Java suivante, qui nous permet de sélectionner le caractère en position i d'une chaîne de caractères.

```
char charAt(String s, int i) {
    return s.charAt(i);
}
```

Exercice 2

Écrire le code ASCII en hexadécimal du message `Ceci est du code Ascii` sachant que le code décimal de `' '` est 32, de `'A'` est 65, et de `'a'` est 97. Vérifier la réponse sur la machine.

Exercice 3

Soit $n > 1$ un entier pair et soit $u(n)$ le mot binaire $u_1 u_2 \cdots u_n$ avec $u_k =$ `1` si k est un nombre premier et `0` sinon.

1. Écrire une fonction Java `chainepremier` qui prend un entier pair $n > 1$ et renvoie la chaîne de caractères représentant $u(n)$. Par exemple, `chainepremier(10)` renvoie `0110101000` parce que les nombres premiers plus petits que 10 sont 2, 3, 5 et 7.
2. On découpe le mot $u(n)$ en blocs de taille 2, de la forme $u_{2k-1}u_{2k}$ pour $1 \leq k \leq n/2$. Écrire une fonction `occurrences` qui prend un entier pair $n > 1$ et affiche le nombre d'occurrences de chacun de ces blocs de taille 2 dans $u(n)$. Par exemple, `occurrences(10)` affiche `00:1,01:1` et `10:3`.

En utilisant la sortie de `occurrences(100)`, construire un arbre de Huffman sur les blocs de taille 2 et calculer la taille du mot $u(100)$ une fois compressé par la méthode de Huffman.

Exercice 4

Implémenter une fonction Java `rle` qui prend une chaîne de caractères `0` et `1` et renvoie une chaîne de caractères représentant son encodage RLE en décimal. Par exemple, `rle("000111100000111111000000")` affiche `3 4 5 6 7`. En utilisant la sortie de `rle("000000111111000000000000011110000000000000")`, construire l'encodage RLE en binaire.