Structure de données
Les arbres (exercices)

Ce TD propose d'étudier une méthode de compression de données inventée par David Albert Huffman en 1952, qui permet de réduire la longueur du codage d'un alphabet et qui repose sur la création d'un arbre binaire.

1 Découverte de l'algorithme : Les marmottes font dodo

Activité faite ensemble, d'après le travail de Marie DUFLOT-KREMER, maître de conférences en informatique à l'Université de Lorraine, au Loria, Laboratoire lorrain de Recherche en Informatique et ses Applications.

2 Codage de Huffman :

On appelle alphabet l'ensemble des symboles (caractères) composant la donnée de départ à compresser. Dans la suite, nous utiliserons un alphabet composé seulement des 8 lettres **`A, B, C, D, E, F, G et H.`**

2.1 Codage

On cherche à coder chaque lettre de cet alphabet par une séquence de chiffres binaires.

Question 1. Combien de bits sont nécessaires pour coder chacune des 8 lettres de l'alphabet ?

Question 2. Quelle est la longueur en octets d'un message de 1 000 caractères construit sur cet alphabet ?

Question 3. Proposer un code de taille fixe pour chaque caractère de l'alphabet de 8 lettres.

Question 4. On considère maintenant le codage suivant, la longueur du code de chaque caractère étant variable.

Lettre	A	B	C	D	E	F	G	H
Code	10	001	000	1100	01	1101	1110	1111

Ce type de code est dit préfixe, ce qui signifie qu'aucun code n'est le préfixe d'un autre (le code de A est 10 et aucun autre code ne commence par 10, le code de B est 001 et aucun autre code ne commence par 001). Cette propriété permet de séparer les caractères de manière non ambiguë.

En utilisant la table précédente, donner le code du message : **`CACHE`**.

Question 5. Quel est le message correspondant au code **`001101100111001`**.

Question 6. Dans un texte, chacun des 8 caractères a un nombre d'apparitions différent. Cela est résumé dans le tableau suivant, construit à partir d'un texte de 1 000 caractères.

Lettre	A	B	C	D	E	F	G	H
Nombre	240	140	160	51	280	49	45	35

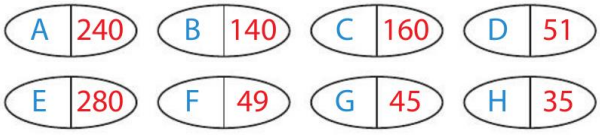
Question 7. En utilisant le code de taille fixe proposé à la **question 3**, quelle est la longueur en bits du message contenant les 1 000 caractères énumérés dans le tableau précédent ?

Question 8. En utilisant le code de la question 4, quelle est la longueur du même message en bits ?

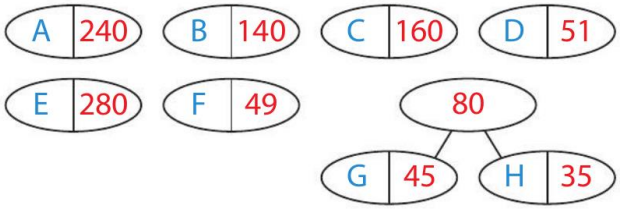
2.2 Code de Huffman

L'objectif du codage de Huffman est de trouver le codage à la question 4 qui minimise la taille en nombre de bits du message codé en se basant sur le nombre d'apparition de chaque caractère, un caractère qui apparaît souvent aura un code plutôt court (comme la marmotte qui se réveille souvent est placée près de la sortie).

Pour déterminer le code optimal, on considère 8 arbres, chacun réduit à une racine, contenant le symbole et son nombre d'apparitions.



Puis on fusionne les deux arbres contenant les plus petits nombres d'apparitions (valeur inscrite sur la racine), et on affecte à ce nouvel arbre la somme des nombres d'apparitions de ses deux sous-arbres. Lors de la fusion des deux arbres, le choix de mettre l'un ou l'autre à gauche n'a pas d'importance. Nous choisissons ici de mettre le plus fréquent à gauche (s'il y a un cas d'égalité, nous faisons un choix arbitraire).



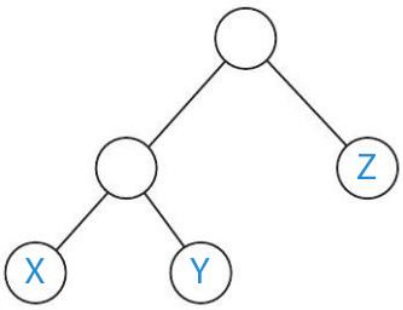
On recommence jusqu'à ce qu'il n'y ait plus qu'un seul arbre.

Question 9. Combien d'étapes (combien de fusions d'arbres) sont nécessaires pour que cette algorithme se termine ?

Question 10. En suivant l'algorithme précédent, construire l'arbre de Huffman.

Le code à affecter à chaque lettre est déterminé par sa position dans l'arbre. Précisément, le code d'un symbole de l'alphabet décrit le chemin de la racine à la feuille qui le contient : un 0 indique qu'on descend par le fils gauche et un 1 indique qu'on descend par le fils droit.

Dans le cas de l'arbre ci-contre, le code de X est 00 (deux fois à gauche), le code de Y est 01, et celui de Z est 1.
Sur chaque arête de l'arbre construit à la question 10, inscrire 0 ou 1 selon que l'arête joint un fils gauche ou un fils droit.



Question 11. Quel est le code de F ?

3 Programmation

Le code (incomplet) fourni dans le fichier **`huffman.py`**, à partir d'un fichier nommé **`texte.txt`**, de construire l'arbre de Huffman puis un dictionnaire qui associe à chaque caractère du fichier d'entrée son code sous forme d'une séquence de bits (liste de 0 et de 1)

- Compléter ce code en vous appuyant sur les commentaires et les docstring contenu dans le fichier.
- Tester votre code pour des exemples simples.
- En vous inspirant du travail précédant sur les arbres, faire une représentation graphique de l'arbre de Huffman à l'aide de la bibliothèque **`networkx`**.
- ….