Projet Huffman

Frédéric Li Combeau, Kaan Doyurur

Présentation et fonctionnalités du projet :

Le but du projet est de compresser un texte avec le codage de Huffman.

Notre projet permet de :

- Charger et enregistrer un fichier à encoder ou à décoder et une clé d'encodage ou encoder un texte écrit directement sur l'interface
- Encoder ou décoder un fichier
- Afficher l'arbre correspondant au fichier encodé

Avec les fonctionnalités suivantes :

- Interface graphique moderne
- Enregistrement automatique lors de l'encodage ou du décodage
- Enregistrement dynamique selon le nom du fichier encodé ou décodé
- Paramètres d'accessibilités (Changement de la taille des composants, mode clair/sombre)

Structure du projet :

projet.py : Interface graphique principale du projet, fichier principal

Imports : fichier, affichage

App : Classe contenant l'interface graphique de Custom Tkinter

Init :

Super().__init__() de la classe customtkinter.CTk

Tous les widgets, button, frame etc.

Héritage : Hérite de la classe customtkinter.CTk, classe gérée par Custom Tkinter

fichiers.py : Fonctions liées aux fichiers, à l'interface graphique, opérations globales, partie qui lie l'interface graphique aux autres modules

Imports : encodage

Fichiers : Classe contenant les fonctions d'opérations sur des fichiers et appliquant les fonctions de codage afin d'obtenir le texte à encoder, la clé d'encodage et le texte encodé

Utilisation d'attributs protégés et du décorateur property

Init :

self._filename : nom du fichier

self._fichier : contenu du fichier

self._cle : clé d'encodage

self._texte_encode : texte encodé

Enregistrement : Enregistrement d'un dictionnaire sous la forme d'un str dans un fichier .key. La fonction eval est utilisée pour transformer le str en dictionnaire à nouveau

encodage.py : Fonctions liées à l'encodage

Imports : structure

Huffman : Classe contenant les fonctions de codage de Huffman

Init :

self.texte : texte à encoder

self.arbre : arbre crée par la classe ArbreB dans le module structure.py

Fonction decodage_char en dehors de la classe Huffman car il n'y a pas besoin d'en faire une méthode de cette classe

structure.py : Fonctions agissant sur l'arbre binaire

Sommet : Classe contenant les sommets d'un arbre binaire

Init :

self.valeur : valeur du sommet

self.etiquette :

self.gauche : Sous arbre gauche

self.droite : Sous arbre droit

self.code_binaire : code binaire du sommet

ArbreB : Classe contenant un arbre binaire

Init : self.racine : racine de l'arbre

Surcharge :

__lt__

__add__

affichage.py : Fonctions affichant l'arbre

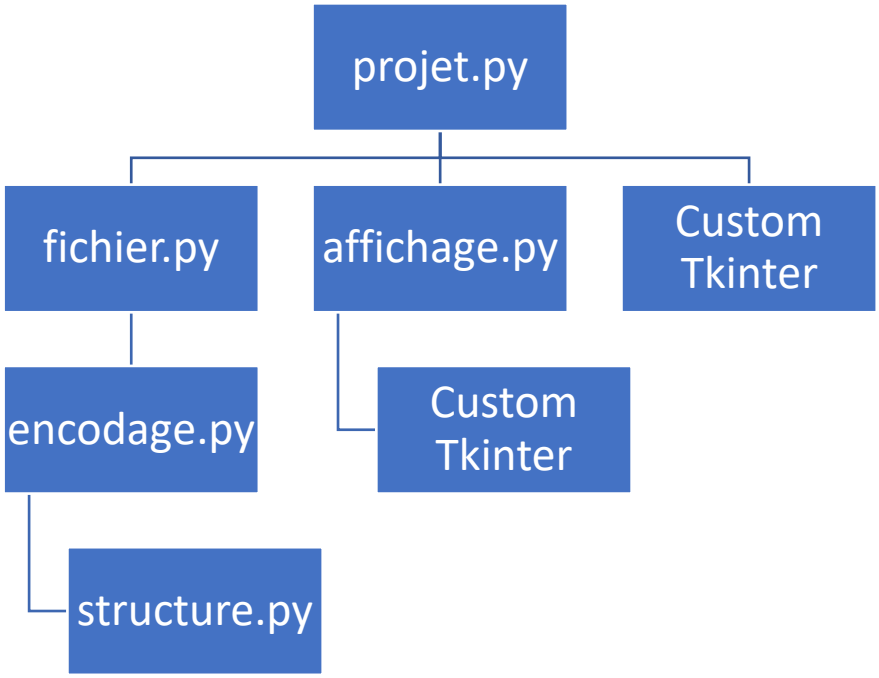
TopLevelWindow : Classe contenant les fonctions permettant l'affichage de l'arbre sur une interface graphique et permettant de se déplacer dans l'interface

Init :

Super().__init__() de la classe customtkinter.CTkTopLevel

Canvas pour afficher l'arbre

Héritage : Hérite de la classe customtkinter.CTkTopLevel, classe gérée par Custom Tkinter



Utilisation :

Lancer le fichier projet.py

Encodage :

- Charger un fichier ou écrire un texte à encoder directement dans l'interface
- Appuyer sur Encoder
- Le texte et la clé sont affichés et enregistrés automatiquement
- Vous pouvez également appuyer sur Afficher Arbre pour afficher l'arbre

Décodage :

- Charger un fichier à décoder ainsi que sa clé d'encodage

- Appuyer sur Décoder
- Le texte décodé est affiché et enregistré automatiquement

Utilitaire :

- Vous pouvez changer l'affichage avec les boutons en bas à gauche afin d'améliorer l'accessibilité

Fichiers fournis en exemple :

projet.py, structure.py, fichiers.py, encodage.py, affichage.py, programme_test.py (uniquement pour la soutenance)

Extrait de Les Misérables, Victor Hugo (version anglaise)

- Les Misérables.txt
- Les Misérables_cle.key
- Les Misérables_encode.txt
- Les Misérables_decode.txt (identique au fichier original)

Veuillez prévoir un léger freeze pour l'encodage et un freeze plus long pour le décodage, le fichier étant assez gros.

<u>Conclusion :</u>

L'encodage n'est pas encore tout à fait fini (ce qui est demandé par le projet l'est) car il faudrait convertir le codage binaire en bytes.

Les méthodes demandées dans la partie 1 ne sont pas toutes utiles, notamment la suppression de sommets.