Projet Python : Cryptage et décryptage

Groupe TD1

Constituer de :

- Raphael DUPUY

- Elie KANGA

Ce projet est à effectuer pour l'unité d’enseignement IN407 - concept avancé de programmation. Nous en avons eu connaissance le vendredi 09 février 2023. Suite au cours magistral du même jour, nous avions déjà une idée du sujet et de comment l’appréhender. Celui-ci nous a été fourni le lundi 13 février 2023.

Nous aborderons ce compte rendu comme un journal de bord qui nous accompagnera tout au long de notre travail. vous pourrez donc y trouver les obstacles que nous allons rencontrer , les solutions qui y répondent et même celles qui n'auront pas été retenu dans la version finale de notre projet qui sera présentée lors du dernier TD. Vous pourrez donc mieux comprendre nos choix et l'architecture de notre programme.

Tout d'abord ce projet a pour but la création d'un programme qui aura comme tâche le cryptage et le décryptage d'un fichier .txt. Nous utiliserons le codage d'huffman, qui nous permettra d'obtenir une "clé de cryptage" unique pour chaque fichier. Pour cela nous aurons besoin d'une des structures de données les plus utilisées en informatique : les arbres, et plus précisément les arbres binaires. En effet le codage d'huffman est basé sur les arbres binaires mais nous aurons l’occasion d'en reparler plus tard quand nous nous pencherons sur la partie 2 du projet c'est-à-dire le cryptage.

Nous avons commencé la première partie

Nous abordons maintenant l’interface graphique du projet. Pour la construire nous avons décidé de nous servir de la bibliothèque TKINTER. Nous avons effectué une première version qui tenait en une fonction. Elle nous permettait d’instancier la fenêtre graphique et d’afficher l’arbre qui lui était donné en argument. La fenêtre graphique c’est à dire le root était composé d’un Canvas et d’un bouton. La taille du Canvas ne tenait pas compte des dimensions de l’arbre, aussi pour le parcourir et pouvoir l’afficher nous avons choisi un parcours en largeur

.

Cependant avec cette implémentation certes l’arbre était affiché mais il y avait plusieurs problèmes :

* Les sommets de l’arbre étaient illisibles soit n’apparaissant pas dans Screen soit étant entassés sur eux.
* Les branches de l’arbre se croisaient, ce qui le rendait encore moins lisible.

On a donc pensé à la deuxième version. Dans celle-ci nous avons créé un fichier int\_graph.py où nous définissons les fonctions nécessaires à la construction de notre interface graphique. Nous avons toujours la fenêtre graphique qui est instanciée avec une colonne qui contient les boutons et un Canvas qui lui cette fois- ci est construit après avoir calculer la hauteur de l’arbre pour que ses dimensions soient adéquates, et nous utilisons toujours un parcours en largeur. Avec la fonction creation\_arbre() nous plaçons ensuite la racine de l’arbre au centre de la largeur du Canvas nous vérifions les fils gauche et droit, tant que ceux-ci sont des objets de la classe ArbreB nous rappelons la fonction creation\_arbre() sur eux en divisant la largeur par deux. Chaque sous-arbre tient alors dans une moitié de l’arbre dont le sommet est la racine.