Legrand Amandine
Ribas Justine

Projet semestre 1 CPI1

Rapport : Algorithme de Huffman

Sommaire :

I- Résumé du problème

Pour ce projet, nous devions réaliser un programme permettant de compresser un fichier texte. Pour cela il fallait se référer à l'algorithme de Huffman. Autrement dit, nous codions les caractères les plus fréquents avec un petit nombre de bits et inversement afin d'optimiser l'espace de stockage.
Il était également demandé de réaliser le programme permettant la décompression du fichier.

II- Description du programme

Notre programme propose à l'utilisateur deux options possibles : la compression (-c) ou la décompression (-d).Le détail de l'utilisation des commandes se trouve dans le README. Nous avons voulu essayer de faire une compression réelle mais nous n'avons malheureusement pas abouti.
Pour réaliser la compression nous avons procédé à plusieurs étapes :

- L'ouverture d'un fichier et la lecture de chaque caractère
- La création d'une table de fréquence
- La création de la table de Huffman
- La création d'un nouveau fichier et la traduction du texte en binaire selon la table de Huffman

Pour réaliser la décompression nous avons fait :

- L'ouverture du fichier compressé
- La traduction du binaire en caractère selon la table de Huffman jointe en entête

III- Détails des choix pour l'implémentation

Pour l'implémentation de ce programme nous avons dû faire différents choix, notamment pour la création des tables de fréquence et de code.

En effet pour créer la **table de fréquence** nous avons déjà créé un tableau d'entier de taille 256 pour pouvoir contenir les fréquences de tous les caractères potentiels. Nous avons créé dans un deuxième temps seulement le tableau de fréquence utilisé dans le reste du programme. Ce tableau définitif est un tableau de type structuré "s_freqCar" et contient les valeurs du premier tableau excepté les valeurs dont la fréquence est nulle. Autrement dit nous avons un tableau contenant les informations relatives aux caractères présents dans le fichier à compresser uniquement. Grâce à la structure "s_freqCar" nous pouvons stocker dans le tableau une liste de caractères (qui est concrètement un tableau d'entier), la taille de cette liste ainsi que la fréquence relative à cette liste. Initialement chaque case correspond à un seul caractère, la liste est donc un tableau à une case (tailleListe=1).
Cependant lors de la création de la table de code nous allons modifier notre tableau de fréquence. Une fois que notre programme trouve les deux fréquences minimum, il modifie les informations relatives à une des deux cases du tableau (ici la case de min1) : il crée une nouvelle liste contenant les caractères de min1 ainsi que ceux de min2, il additionne alors la taille des deux listes ainsi que leurs fréquences. Ensuite on décale toutes les cases à partir le la case de min2 pour la supprimer en quelques sortes, ainsi on diminue la taille du tableau.
On répète cette opération jusqu'à ce que la taille du tableau soit égale à 1.

Pour créer la **table de code**, nous avons initialisé un tableau du type structuré "s_codeCar" de taille 256. Dans cette structure nous avons le code sous forme de tableau d'entier (où on stockera des 0 ou des 1), ainsi que la taille de ce tableau. A chaque modification du tableau de fréquence on associe un 1 ou un 0 au code des deux min.

Pour cela on crée un nouveaux tableau d'entiers plus grand par lequel on remplace l'ancien. Une fois la table de Huffman entièrement créée, on traduit les tableaux de code par des entiers qu'on stocke dans un tableau de 8 cases pour obtenir un octet.

Pour créer le **fichier compressé,** on écrit d'abord le nombres de caractères différents en entête, puis la table de Huffman sous la forme : code Ascii - taille code - code Huffman. Ensuite, on vient lire dans le fichier à compresser caractère par caractère et on écrit la suite textuelle de 0 et de 1 correspondante.

Pour la **décompression**, on vient lire la table Huffman en entête puis on la "reconstitue" sous forme de tableau. Le nombre de caractère au tout début du fichier compressé nous indique la fin de la table de Huffman. Ensuite nous avons la suite textuelle de 0 et 1 correspondant au texte traduit. Nous allons donc lire le premier caractère. Si celui-ci correspond à un code de la table Huffman, on écrit son caractère correspondant dans le fichier décompressé. Sinon on prend le caractère suivant et on recherche à nouveau, et ainsi de suite. A la fin, la totalité de la suite textuelle a été traduite et le fichier décompressé est identique au fichier de base.

IV- Avantages et limites de cette solution

Notre programme fonctionne mais il n'est malheureusement pas très optimisé dû au manque de temps pour modifier notre logique de départ. En effet, pour la création de la table Huffman, nous stockons le code sous forme de tableaux de 0 et de 1. Cependant créer et utiliser des tableaux utilise beaucoup de mémoire. De plus, nous nous sommes rendu compte plus tard que pour la compression réelle et la décompression, il était préférable d'avoir seulement un entier et non un tableau d'entiers. Ainsi la logique de notre code peut paraître ambigüe et nous utilisons beaucoup de mémoire.

Par manque de temps également nous n'avons pas pu finir la compression réelle.

V- Contribution de chaque élève au projet

Pour réaliser ce projet nous avons d'abord réfléchi à la logique ensemble. Ensuite nous nous sommes réparti les tâches.

Justine s'est occupée de créer le tableau de fréquences, de toute la partie décompression et de la mise en commun / mise en forme du projet.

Amandine a codé toutes les fonctions relatives à la compression, c'est à dire la création de la table de Huffman ainsi que la création du fichier compressé.

VI- Bilan personnel

Conclusion Justine :

J'ai beaucoup aimé travailler sur ce projet. J'ai beaucoup appris d'une part sur les nouvelles techniques de code utilisées mais aussi d'autre par sur le travail en équipe. En effet il n'est pas toujours évident de coder à plusieurs car il faut que l'ensemble des travaux mis en commun compile. Ensuite il est presque impossible d'aborder le problème avec la même logique. Il a donc fallu trouver des compromis pour que nous travaillons toutes les deux avec une logique qui nous correspondait. Enfin j'ai réalisé que le manque de temps pouvait être un vrai problème pour un aussi long projet.
Pour conclure, grâce à ce projet j'ai acquis une bonne expérience en ce qui concerne le travail en groupe.

Conclusion Amandine :

Le projet m'a aidée pour comprendre certains aspects du langage C (structures, écriture et lecture dans un fichier) et c'est intéressant de faire pour la première fois un long programme qui nécessite beaucoup de fonctions. Mais c'était compliqué de se mettre d'accord sur certains points, et je n'ai pas eu l'impression de pouvoir maîtriser totalement le sujet.

Pour conclure, je pense que cela m'a fait progresser.