Codes de Huffman et Shannon-Fano

Dans le monde numérique, la transmission et le stockage de données volumineux constituent un enjeu majeur. La compression de données s'avère alors une solution essentielle pour réduire la taille des fichiers sans perte d'information, optimisant ainsi l'espace de stockage et la bande passante lors de la transmission.

Parmi les techniques de compression sans perte les plus populaires, les codes de Huffman et Shannon-Fano se distinguent par leur efficacité et leur simplicité. Ils exploitent les statistiques des données pour construire des codes préfixes, où les symboles les plus fréquents sont représentés par des codes plus courts.

1. Principes des codes de Huffman et Shannon-Fano

1. Analyse statistique :

* Déterminer la fréquence d'apparition de chaque symbole (lettre, caractère, etc.) dans les données à compresser.
* Stocker ces fréquences dans une table de probabilités.

2. Construction du code :

a) Code de Huffman:

* Arbre de Huffman:
* Créer un arbre binaire en regroupant les symboles par paires selon leurs fréquences croissantes.
* Les symboles les plus fréquents se situent aux niveaux inférieurs de l'arbre.
* Les codes sont construits en suivant les chemins depuis la racine vers les feuilles, en utilisant "0" pour les branches à gauche et "1" pour les branches à droite.
* Propriété : Le code de Huffman est optimal, c'est-à-dire qu'il minimise la longueur moyenne des codes, optimisant la compression.

b) Code de Shannon-Fano:

* Division récursive :
* Diviser les symboles en deux groupes selon la médiane de leurs fréquences.
* Affecter "0" au groupe de gauche et "1" au groupe de droite.
* Répéter la division sur chaque groupe jusqu'à ce que chaque symbole ait son propre code.
* Exemple :

Symboles : A, B, C, D avec fréquences respectives 4, 3, 2, 1

Codes : A = 0, B = 10, C = 110, D = 111

* Propriété : Le code de Shannon-Fano n'est pas nécessairement optimal, mais il est généralement proche de l'optimalité.

1. Algorithmes
2. Codage de Shannon-Fano

Utilisé dans les années cinquante, le code de Shannon-Fano est le premier code à avoir exploité la redondance d'une source. L'algorithme de Shannon-Fano consiste à faire en sorte que les éléments binaires composant les mots code apportent une quantité d'information moyenne le plus grand possible.

– 1ère étape : On classe les symboles source (par exemple sur une colonne) par ordre de probabilité décroissante (par exemple de haut en bas),

– 2ème étape : On divise l'ensemble des symboles en deux sous-ensembles de telle sorte que les probabilités cumulées des éléments constituant chacun des deux sous-ensembles soient les plus proches. On attribue l'élément binaire "1" (resp. "0") aux éléments du sous ensemble situé en haut.

– 3ème étape : On procède comme à la première étape sur tous les sous-ensembles comportant au moins deux éléments. On s'arrête lorsque tous les sous-ensembles ne comportent plus qu'un élément.

Algorithme :

N : nombre de symboles utilisés

f (i) : fréquence (occurrence) du symbole numéro i

T : table des fréquences f(i)

– Evaluer les occurrences f(i)

– Dresser la table T telle que f(1)>=f(2)>=….>=f(N)

– Scinder T en 2 sous-tables donnant les 2 fréquences les plus proches

– Itérer pour chaque sous-table jusqu’à card(Tj)=1 pour toute sous-table Tj.

– Attribuer dans l’arborescence les bits 0 et 1 aux 2 sous-tables de même niveau.

– Construire les codes binaires

1. Codage de Huffman

Contrairement au code de Shannon-Fano, la construction d'un code de Huffman s'effectue de façon ascendante (des feuilles vers la racine de l'arbre). Sa mise en œuvre est décrite par l'algorithme :

– 1ère étape: On classe (par exemple sur une ligne) les symboles source par ordre de probabilité croissante (par exemple de la gauche vers la droite).

– 2ème étape: On relie les deux symboles de probabilités les plus faibles à l'aide de deux arêtes. On obtient alors un nœud que l'on appelle "super-symbole" et dont le "poids" est obtenu en effectuant la somme des deux probabilités des deux symboles. Le supersymbole remplace les deux symboles qui sont maintenant éliminés de la liste.

– 3ème étape: On procède comme à l'étape précédente en considérant les super-symboles comme des symboles. On s'arrête lorsqu'on est arrivé à la racine de l'arbre.

Algorithme :

● N : nombre de symboles utilisés dans le message

● f(i) : fréquence (occurrence) du symbole numéro i

● T : table des occurrences f(i)

p, s, k: variables locales

Evaluer les occurrences f(i)

Dresser la table T telle que f(1)>=f(2)>=….>=f(N)

p=N

Tant que card(T)>=2 faire

s=f(p) + f(p-1) (les 2 plus petites fréquences)

k= min(i | s >=f(i) )

si s=f(k) alors

k=k+1

Retirer f(p) et f(p-1) de T

Insérer s dans T à la position k

p=p-1

Affecter aux antécédents de k le bit 0

Construire les codes binaires

1. Exemples
2. Code de Huffman

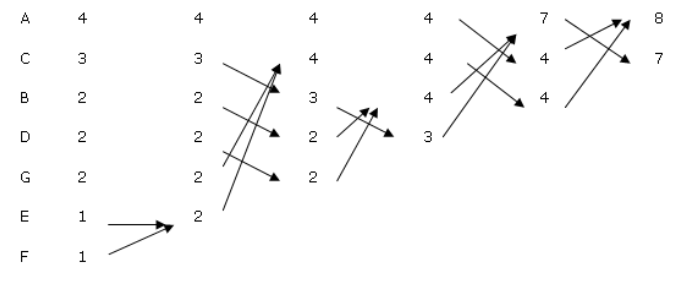
Dans cet exercice, nous allons effectuer les différentes étapes de l’algorithme de compactage à la main : choix des symboles, établissement de la table des fréquences et des codes pour le compactage selon la méthode de Huffman. La chaîne à traiter est : « BACFGABDDACEACG ».

1) Les symboles utilisés sont les lettres : A, B, C, D, E, F, G

2) La table des fréquences est donnée ci-dessous. Les symboles sont écrits dans la table par ordre décroissant de fréquence d'apparition. BACFGABDDACEACG

Table T :A 4 C 3 B 2 D 2 G 2 E 1 F 1

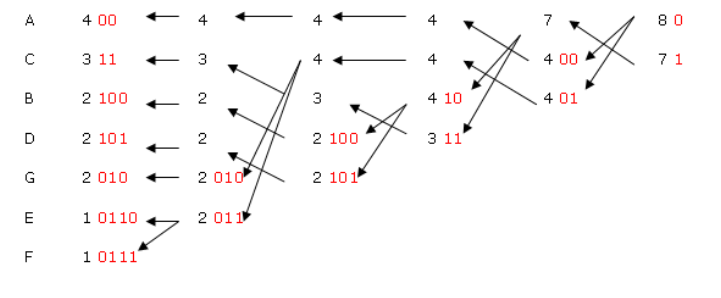
3) On regroupe les symboles de plus faible fréquence en sommant leurs fréquences et on place le nombre ainsi calculé dans une nouvelle table où les deux symboles initiaux ont été enlevés. On obtient l’arborescence suivante:



4) On utilise l'arborescence de Huffman pour construire les codes binaires associés aux symboles.

On attribue à la plus haute fréquence de la dernière table la valeur 0, 1 à l'autre fréquence.

Ensuite on code par retour arrière de la manière suivante : à chaque bifurcation dans une table on attribue à l'antécédent de la plus haute fréquence la valeur 0, à l'antécédent de la plus faible fréquence la valeur 1. On obtient ainsi l'arbre décoré suivant :



Pour obtenir les codes, il suffit de parcourir l'arbre depuis le symbole étudié jusqu'à la fin du chemin en empilant les symboles binaires rencontrés sur le chemin: A: 00 C: 11 B: 100 D: 101

G: 010 E: 0110 F: 0111

1. Codage de Shannon-Fano

Chaîne à traiter : ACBBCDECECBCEEECCCCABCECBDBDBD

1) Les symboles utilisés sont les lettres A, B, C, D et E.

2) On commence comme pour l’algorithme de Huffman à classer les symboles par ordre décroissant de fréquence, puis le processus de scission de la table commence. Les scissions de tables sont toujours réalisées de la même manière: C: 11 B: 7 E: 6 D: 4 A: 2

1ère scission : 11, 7 d’une part et 6, 4, 2 d’autre part. C’est la scission qui donne la meilleure répartition des fréquences. Chaque sous-table est à nouveau scindée : 11, 7 en 2 tables 11 et 7,

6, 4, 2 en 2 tables : 6 d’une part et 4, 2 d’autre part, enfin la dernière sous-table 4,2 est scindée en 4 et 2.

3) Il faut affecter le bit 0 ou 1 en descendant l’arborescence depuis la racine. On choisit par exemple 0 pour la sous-table de fréquence la plus élevée à chaque fois. Les codes associés sont donc :

C (11) : 00 | B (7) : 01 | E (6) : 10 | D (4) : 110 | A (2) : 111

1. Avantages et inconvénients
2. Avantages communs :

* Compression de données sans perte: les données originales peuvent être parfaitement reconstituées à partir du code compressé.
* Simplicité d'implémentation: les algorithmes de codage et de décodage sont relativement simples à comprendre et à mettre en œuvre.
* Efficacité: les deux codes peuvent réduire considérablement la taille des données, en particulier pour les sources avec une entropie élevée.

1. Avantages du code de Huffman :

* Optimalité: le code de Huffman génère toujours le codage le plus court possible pour une source donnée, en moyenne.
* Flexibilité: le code de Huffman peut générer des codes de longueur variable, ce qui peut être utile pour certains types de données.

1. Inconvénients du code de Huffman :

* Complexité de l'algorithme: l'algorithme de construction du code de Huffman est légèrement plus complexe que celui de Shannon-Fano.
* Sensibilité aux modifications de la source: un changement mineur dans la distribution des symboles peut entraîner une modification importante du code de Huffman.

1. Avantages du code de Shannon-Fano :

* Simplicité de l'algorithme: l'algorithme de construction du code de Shannon-Fano est plus simple que celui de Huffman.
* Rapidité d'exécution: le codage et le décodage de Shannon-Fano sont généralement plus rapides que ceux de Huffman.

1. Inconvénients du code de Shannon-Fano :

* Non-optimalité: le code de Shannon-Fano n'est pas toujours optimal et peut générer des codes légèrement plus longs que Huffman.
* Rigidité: le code de Shannon-Fano génère des codes de longueur fixe, ce qui peut être moins efficace pour certains types de données.

1. Domaines d’applications

Les codes de Huffman et de Shannon-Fano sont utilisés dans divers domaines pour compresser des données sans perte. Voici quelques exemples :

Compression de fichiers:

• Fichiers texte: Les codes de Huffman et de Shannon-Fano sont souvent utilisés pour compresser des fichiers texte, tels que des documents, des livres électroniques et des scripts.

• Fichiers d'images: Les codes de Huffman et de Shannon-Fano peuvent être utilisés pour compresser des fichiers d'images, en particulier les images bitmap et les images en niveaux de gris.

• Fichiers audio: Les codes de Huffman et de Shannon-Fano peuvent être utilisés pour compresser des fichiers audio, en particulier les formats sans perte tels que FLAC et ALAC.

Réseaux et communications:

• Transmission de données: Les codes de Huffman et de Shannon-Fano peuvent être utilisés pour compresser des données avant de les transmettre sur un réseau, ce qui permet de réduire la bande passante nécessaire.

• Stockage de données: Les codes de Huffman et de Shannon-Fano peuvent être utilisés pour compresser des données avant de les stocker, ce qui permet de réduire l'espace de stockage nécessaire.

Autres domaines:

• Compression de logiciels: Les codes de Huffman et de Shannon-Fano peuvent être utilisés pour compresser des logiciels, ce qui permet de réduire la taille des fichiers exécutables et des bibliothèques.

• Bases de données: Les codes de Huffman et de Shannon-Fano peuvent être utilisés pour compresser des données dans les bases de données, ce qui permet d'améliorer les performances et de réduire l'espace de stockage nécessaire.

Conclusion:

Les codes de Huffman et Shannon-Fano constituent des outils fondamentaux pour la compression de données sans perte. Bien que le code de Huffman soit optimal, le code de Shannon-Fano offre un bon compromis entre efficacité et simplicité. La compression de données joue un rôle crucial dans l'optimisation de l'espace de stockage et de la transmission d'informations dans le monde numérique.