Algo L3 Info
Travaux dirigés, séance 10.1

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Compression, algorithme de Huffman

On dispose d'un texte écrit avec un alphabet $\mathcal{A} = \{A, B, C, D, E, F, G, H\}$. Les proportions de ces symboles dans le texte sont données en pourcentage dans le tableau suivant :

Symbole	A	B	C	D	E	F	G	H
Proportion (%)	2	10	4	18	12	16	32	6

Codage de longueur fixe
Donner la taille du codage de longueur fixe nécessaire pour coder cet alphabet $\mathcal{A}$.
Pourquoi n'est-ce probablement pas le codage le plus court en moyenne pour cet alphabet ?
Réponse 0 :

L'alphabet $\mathcal{A}$ possède 8 lettres, il faut donc au moins $\log_2(8) = 3$ bits pour coder ces 8 symboles (code de longueur fixe). On calcule l'entropie associée à cette distribution de probabilité.

$Symbole$	$\%$	$Proba$	$-\log_2(p_i)$	$-\log_2(p_i)p_i$
A	2	$0,02$	$5,64$	$0,11$
B	10	$0,10$	$3,32$	$0,33$
C	4	$0,04$	$4,64$	$0,18$
D	18	$0,18$	$2,47$	$0,44$
E	12	$0,12$	$3,05$	$0,36$
F	16	$0,16$	$2,64$	$0,42$
G	32	$0,32$	$1,64$	$0,52$
H	6	$0,06$	$4,05$	$0,24$
$s =$	100		$H(p) =$	$2,63$

On trouve que $H(p) = 2,63$ c'est à dire qu'il faut au minimum et en moyenne 2.63 bits pour coder un symbole de l'alphabet alors qu'il en faudrait 3 pour un code de longueur fixe. On peut donc gagner à avoir un code de longueur variable qui compressera le texte.

Codage de longueur variable
Construire avec l'algorithme de Huffman un codage de longueur variable pour $\mathcal{A}$. Donner l'arbre de codage correspondant et le tableau des codes. Calculer la longueur moyenne du codage et commenter le résultat en une phrase. **Réponse 0 :**

Après avoir fait tourner Huffman (on peut avoir plusieurs solutions) on obtient

$Symbole$	$\%$	$Proba$	$code$	$longueur$	$longueur\ \ theorique$
A	2	$0,02$	01000	5	$5,64$
B	10	$0,10$	011	3	$3,32$
C	4	$0,04$	01001	5	$4,64$
D	18	$0,18$	00	2	$2,47$
E	12	$0,12$	100	3	$3,05$
F	16	$0,16$	101	3	$2,64$
G	32	$0,32$	11	2	$1,64$
H	6	$0,06$	0101	4	$4,05$
$s =$	100		$L(h) =$	$2,68$	$H(p) = 2,63$

On retrouve bien $H(p) \leqslant L(h) \leqslant H(p) + 1$. Ici l'algorithme produit un résultat très proche de la valeur minimale théorique, il sera difficile de faire mieux

Comptage (cas général)
Quel est le nombre minimal et le nombre maximal de nœuds internes dans un arbre de codage construit avec l'algorithme de Huffman pour un alphabet de n lettres ?
Vous justifierez la réponse et le calcul avec soin.

Algorithme de décodage (cas général)
Etant donné un arbre de codage binaire, écrire un algorithme de décodage qui lit un texte codé et affiche le texte décodé. (Les opérateurs de base sur le type arbre de codage sont donnés au verso de cette feuille).
Calculer la complexité de cet algorithme, en fonction de la longueur du texte codé et/ou de toute mesure qui vous semblera utile à propos de l'arbre de codage.

Réponse 0 :

Il faut lire successivement les symboles du texte codé et parcourir en même temps l'arbre en suivant les branches indiquées par les bits. De plus, dès qu'on arrive à une feuille, on écrit le symbole correspondant.

Tableau des codes (cas général)
Etant donné un arbre de codage binaire, écrire un algorithme qui construit la table qui associe les codes aux symboles.
Calculer et commenter la complexité de cet algorithme.

Réponse 0 :

Il suffit de parcourir l'arbre, de mémoriser les chemins parcourus sous forme d'une suite de bits, et d'afficher ces suites au niveau des feuilles. C'est à dire que l'on garde le chemin de la racine au noeud courant. (voir les algos du TD correspondant)

Opérateurs sur le type Arbre de codage

```
EstFeuille
paramètres : un Arbre de codage A (donnée)
valeur de retour : un booléen
```

Algorithme de décodage

```
Arbre code, code_ courant ;
Symbole s ;
{Code contient l'arbre de codage construit auparavant par l'algorithme de Huffman}
code_courant = code ;
s=lire_symbole ;
while s !=fin_de_texte do
   if s="0" then
      code_courant = FGauche(code_courant) ;
   else
      code_courant = FDroit(code_courant) ;
   end if
   if Estfeuille(code_courant) then
      Ecrit(Symbole(code_courant )) ;
      code_courant = code ;
   end if
   s=lire_symbole ;
end while
if code_courant != code then
   Message d'erreur : la fin du texte n'est pas décodée
end if
```

```
description : vaut vrai ssi A est réduit à une feuille
effets de bord : aucun

Symbole
paramètres : un Arbre de codage A (donnée)
valeur de retour : un symbole de l'alphabet de départ
description : A doit être réduit à une feuille,
                        renvoie le symbole associé à la feuille
effets de bord : aucun

FDroit
paramètres : un Arbre de codage A (donnée)
valeur de retour : un Arbre de codage
description : A est un arbre non réduit à une feuille,
                        renvoie le fils droit associé à la racine de A
effets de bord : aucun

FGauche
paramètres : un Arbre de codage A (donnée)
valeur de retour : un Arbre de codage
description :  A est un arbre non réduit à une feuille,
                         renvoie le fils gauche associé à la racine de A
effets de bord : aucun
```

POUR LA PROCHAINE FOIS

1. Révisez si nécessaire la notion de graphe, et le vocabulaire associé : sommet, arête, arc, orientation, connexité, cycle, circuit, chemin, sous-graphe, composante connexe, composante fortement connexe ...
2. On peut définir un arbre (non orienté) comme étant un graphe connexe sans cycle. Cherchez 6 autres propriétés permettant de caractériser un arbre : un graphe $T = (X, R)$ (X est l'ensemble des sommets, R est l'ensemble des arêtes) est un arbre si et seulement si ...