Codage d'Huffman

Le problème

obj: étant donné un texte sur un alphabet ∑,
le ***coder*** succinctement en binaire.

On cherche une fonction ϕ : ∑ → {0,1}*
Codage du texte = codage de chaque lettre

Le problème

Exemple:
« l'algorithmique c'est cool »

a	c	e	g	h	i	l	m	o	q	r	s	t	u	'	_
0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

01101110000001100011100010100101110001000111010110011101001011110000111100010101111001111000110001000011 0

Codage à longueur fixe… facile mais pas tjs concis !

Le problème

obj: étant donné un texte sur un alphabet ∑,
le coder ***succinctement*** en binaire.

succinctement = en tenant compte de la **fréquence** de chaque lettre dans le texte à coder !

Le problème

obj: étant donné un texte sur un alphabet Σ, le ***coder*** succinctement en binaire.

- données: un alphabet Σ et une fonction de «fréquence» $f : \Sigma \to N$
- résultat: un codage $\phi : \Sigma \to \{0,1\}^*$ tel que $\sum_{a \in \Sigma} f(a).|\phi(a)|$ soit minimal.

Exemple

Σ :	a	b	c	d	e	f
f:	45	13	12	16	9	5

Solution 1:

a	b	c	d	e	f
000	001	010	011	100	101

$\sum f(a).|\phi(a)| = 45x3+13x3... = 300$

Solution 2:

a	b	c	d	e	f
0	101	100	111	1101	1100

$\sum f(a).|\phi(a)| = 45x1+13x3... = 224$

Codage...

Ici on se limite aux codes préfixes, *ie*:
aucun $\phi(x)$ n'est préfixe d'un $\phi(y)$

Propriété:
Il existe un code préfixe optimal.

Avantage:
Le décodage est très simple !

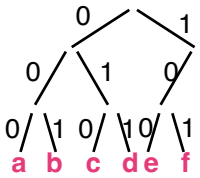
Exemple de décodage

a	b	c	d	e	f
0	101	100	111	1101	1100

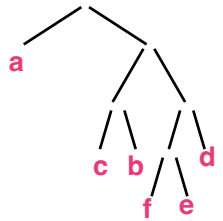
1 0 1 | 0 | 1 0 0 | 1 1 1 | 1 1 0 1 | 0 | 1 0 1
b a c d e a b

Codes préfixes et arbres binaires

a	b	c	d	e	f
000	001	010	011	100	101



a	b	c	d	e	f
0	101	100	111	1101	1100



Codes préfixes et arbres binaires

Un code préfixe optimal est toujours représenté sous la forme d'un arbre binaire localement complet.

Coût d'un arbre T selon f:

$$B_f(T) = \sum_{a \in F(T)} f(a) \cdot prof_T(a)$$

F(T) : feuilles de T

$prof_T(a)$: profondeur du noeud a dans T

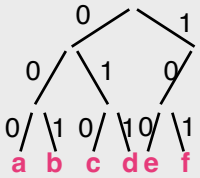
- On écrira B(T) lorsque f est fixée par le contexte.
- On étend B() aux codages: $B(\phi) = \sum f(a).|\phi(a)|$

Codes préfixes et arbres binaires

∑ :	a	b	c	d	e	f
f:	45	13	12	16	9	5

a	b	c	d	e	f
000	001	010	011	100	101

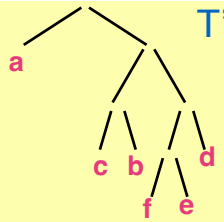
B(T)=(45+13+12+16+9+5)*3= 300



T

a	b	c	d	e	f
0	101	100	111	1101	1100

B(T')=45*1+13*3+12*3+16*3+9*4+5*4= 224



T'

Arbres

On considère les primitives suivantes sur les arbres:

- feuille(a) = crée une feuille étiquetée par «a»
- arbre(t_1,t_2) = crée un arbre avec t_1 comme fils gauche et t_2 comme fils droit
- fg(t) = retourne le fils gauche
- fd(t) : retourne le fils droit

Algorithme d'Huffman

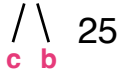
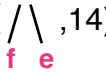
On utilise une file de priorité pour stocker des arbres (correspondant à des morceaux de codes pour des sous-ensembles de ∑) avec comme clé la somme des fréquences de des lettres du sous-ensemble.

```
n := |∑|
FP := FilePriorité( { (feuille(a),f(a))  | a ∈ ∑} )
Pour i = 1 à n-1:
    (t1,f1) := ExtraireMin(FP)
    (t2,f2) := ExtraireMin(FP)
    Ajouter(FP,(arbre(t1,t2),f1+f2))
(T,f) :=ExtraireMin(FP)
retourner T
```

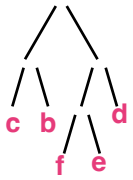
Exemple

∑ :	a	b	c	d	e	f
f:	45	13	12	16	9	5

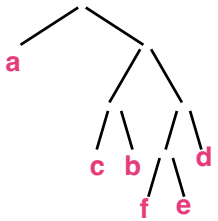
(f,5) et (e,9):  14 + (a,45) (b,13) (c,12) (d,16)

(c,12) et (b,13):  25 + (a,45) (d,16) ( ,14)

( ,14) et (d,16):  30 + (a,45) ( ,25)

( ,25) et ( ,30):  55 + (a,45)

Exemple

(a,45) + ( ,55) → 

Algorithme d'Huffman

Théorème:
L'algorithme d'Huffman donne un code préfixe optimal.

Algorithme d'Huffman

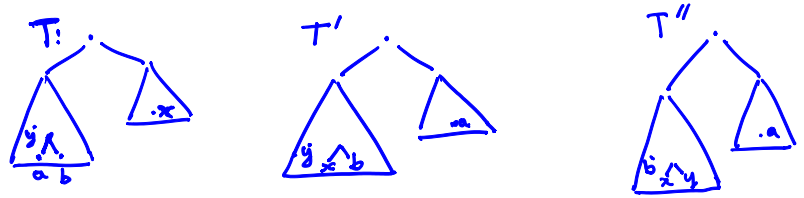
Lemme 1:
Étant donnés (Σ,f) et $x,y \in \Sigma$ telles que x et y aient des fréquences minimales, alors il existe un code préfixe optimal ϕ avec $\phi(x)=w.0$ et $\phi(y)=w.1$

c'est-à-dire un code où x et y ont le même père dans l'arbre binaire associé à ϕ.

Preuve:
Considérons un code optimal et supposons que x et y n'ont pas le même père dans l'arbre associé…

Lemme 1- preuve

- soit T l'arbre associé à un code optimal.
- soit a,b deux feuilles de T, de même père et situées à la prof. max dans T



$B(T')=B(T)-f(x).pf_T(x)-f(a).pf_T(a)+f(x).pf_T(a)+f(a).pf_T(x)$
$B(T')=B(T)+f(x).(pf_T(a)-pf_T(x)) - f(a)(pf_T(a)-pf_T(x))$
$B(T')=B(T)+ (f(x)-f(a)).(pf_T(a)-pf_T(x))$
$\leq 0 \quad \geq 0$
$\Rightarrow B(T') \leq B(T)$

hyp: $f(x) \leq f(a)$
$f(y) \leq f(b)$
$pf_T(a) \geq pf_T(x)$

$B(T'') \leq B(T') \leq B(T)$: T'' optimal !

Algorithme d'Huffman

Lemme 2:
Soit T un arbre binaire représentant un code préfixe optimal pour (Σ,f).
Soient x et y deux feuilles avec le même père z dans T.

Soit $T' = T \setminus \{x,y\}$ où on associe à z une nouvelle lettre z_{new} ,
Alors T' représente un code préfixe optimal pour (Σ',f')

avec: $\Sigma' = \Sigma \setminus \{x,y\} \cup \{z_{new}\}$ et
$f' = f_{|\Sigma \setminus \{x,y\}}$ et $f'(z_{new})=f(x)+f(y)$

Lemme 2 - preuve

$pf_T(x)=pf_T(y)=pf_{T'}(z_{new})+1$

$B(T) = \sum f(a).pf_T(a)$

$B(T') = B(T) - f(x).pf_T(x) - f(y).pf_T(y) + (f(x)+f(y)).pf_{T'}(z_{new})$
$B(T') = B(T) - (f(x)+f(y))\,(pf_T(x) - pf_{T'}(z_{new}))$
$B(T') = B(T) - f(x) - f(y)$

Si T' n'est pas optimal, il existe A' optimal pour $(\sum', f')$
Et remplacer z par (x,y) dans A' donne A tq
$B(A) = B(A') + f(x) + f(y) < B(T') + f(x) + f(y) = B(T)$!

On a donc trouvé un A meilleur que T !

Théorème:
L'algorithme d'Huffman donne un code préfixe optimal.

Preuve: par induction sur $|\sum|$
- $|\sum| = 2$: ok
- $|\sum| = n+1$

Soit ϕ_{algo} le code renvoyé par l'algo et ϕ_{opt} un code optimal.
Soient x et y les deux lettres choisies par l'algo avec des priorités min.
Par le Lemme 1, on peut supposer $\phi_{opt}(x)=w.0$ et $\phi_{opt}(y)=w.1$

Par le Lemme 2, on sait que ϕ' définie par:
$\phi'(u) = \phi_{opt}(u)$ si $u \in \sum \setminus \{x,y,z_{new}\}$ ET $\phi'(z_{new}) = w$
est optimal pour $(\sum', f')$ [...] !
Et de plus: $B(\phi') = B(\phi_{opt}) - f(x) - f(y)$

De son coté, l'algorithme calcule aussi un code ϕ'_{algo} pour $(\sum', f')$ et par **hypothèse d'induction**, il est optimal, donc: $B(\phi'_{algo}) = B(\phi')$
Et on a: $B(\phi'_{algo}) = B(\phi_{algo}) - f(x) - f(y)$ (car $\phi_{algo}(x)=w'.0$ et $\phi_{algo}(y)=w'.1$)
D'où: $B(\phi_{algo}) = B(\phi_{opt})$