Chapitre 4

Algorithme Glouton

Directement extrait de [CLR90]

Olivier Raynaud, Université Blaise Pascal, Clermont-Ferrand

Le choix glouton

Principe : les algorithmes gloutons sont des algorithmes pour lesquels à chaque itération on fixe la valeur d 'une ou plusieurs variables décrivant le problème sans remettre en cause les choix antérieurs.

Précisément Le principe consiste à faire localement le choix qui semble le meilleur, pour se ramener ensuite à la résolution d'un sous problème identique.

Le problème du choix d'activité

Description : soit un ensemble S de n activités concurrentes qui souhaitent utiliser une ressource commune qui ne peut être allouée que pour une activité à la fois.
Chaque activité possède un horaire de début d_i et de fin f_i.

Question: Trouver l'ensemble le plus grand possible d'activités compatibles entre elles.

Le problème du choix d'activité

Exemple d'instance

Considérons les activités suivantes:

a1: [5-9]   a5: [5-7]   a9:[8-11]
a2: [2-13]   a6: [3-8]   a10:[3-5]
a3: [0-6]   a7: [12-14]   a11[8-12]
a4: [1-4]   a8: [6-10]

Question: Trouver l'ensemble le plus grand possible d'activités compatibles entre elles.

Problème « choix d'activité »

Algorithme choixActivité();
Données : s : liste d'activités;
i,j : entier;
Résultat : A : liste d'activités;
Début
n ← s.longueur(); A ← {1}; j ← 1;
pour (i ← 2 à n) faire
si $(d_i >= f_j)$ alors
A ← A union i;
j ← i;
fin pour
Retourner A.
Fin

Optimalité

Théorème : l'algorithme choixActivité() calcule l'ensemble de taille maximum d'activités compatibles.

Elément de démonstration :

- Montrer qu'il existe une solution optimale qui intègre le choix glouton (c'est-à-dire l'activité 1);
- Montrer que le même raisonnement peut être conduit sur S - {1}

Démonstration

Montrer qu'il existe une solution optimale qui intègre le choix glouton :

Soit A une solution optimale ordonnées par horaire de fin croissante, soit k la première activité de A :

Si k=1, *A est optimale et intègre le choix glouton;*
Sinon
soit B = A - {k} + {1} avec $f_1 <= f_k$

Les activités de B sont disjointes et comme B possède autant d'activités que A, B est optimale.

Démonstration

Montrer que si A est une solution optimale sur S, alors A'=A-{1} est une solution optimale sur S'={i dans S : $d_i >= f_1$}

Par l'absurde :
si A'n'est pas optimale,
alors il existe une solution B' pour S' contenant plus d'activité que A'.

B' union {1} est alors une solution pour S contenant plus d'activités que A.

Ce qui contredit l'hypothèse que A était optimale sur S.

Elément de stratégie gloutonne

La propriété du choix glouton : on peut arriver à une solution globalement optimale en effectuant un choix localement optimal.

La propriété de sous structure optimale: un problème fait apparaître une sous structure optimale si une solution optimale du problème contient la solution optimale des sous problèmes.

Validation de la démarche gloutonne

Techniquement :

- Etudier une solution globalement optimale puis montrer que cette solution peut être modifiée pour qu'un choix glouton soit effectué à la première étape;
- Enfin, pour montrer qu'un choix glouton nous ramène à l'étude d'un problème similaire mais plus petit, il suffit de s'assurer qu'une solution optimale fait bien apparaitre des sous structures optimales.

Codage de Huffman

Problème

Minimiser la taille du codage global d'un fichier texte.

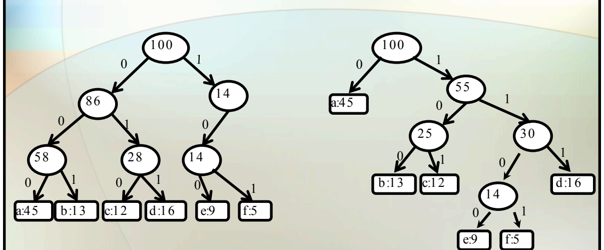
	A	B	c	d	e	f
Fréquence (en millier)	45	13	12	16	9	5
Code de longueur fixe	000	001	010	011	100	101
Code de longueur variable	0	101	100	111	1100	1101

[CLR90]

Codage de Huffman

Définition

Nous appelons codage préfixe un codage ou aucun mot de code n'est aussi préfixe d'un autre mot de code.



[CLR90]

Problème « codage d'Huffman »

Algorithme Huffman();
Données : C : chaîne de caractères;
Résultat: arbre binaire;

```
Début
n ← C.taille;  F ← C;
pour ( i ← 1 à n-1) faire
        z ← nouveauNoeud();
        x ← F.extraireMin(); z.gauche ← x;
        y ← F.extraireMin(); z.droite ← y;
        f(z) ← f(x) + f(y);
     F.inserer(z);
fin pour
Retourner F.extraireMin;
Fin
```


Codage de Huffman

Définition

Soit un alphabet *C* et un caractère *c*, *f(c)* est la fréquence de *c* dans le fichier et $d_T(c)$ la profondeur de la feuille c dans l 'arbre T.

Le nombre de bits requis pour encoder un fichier vaut :

$$B(T) = \Sigma_{c \text{ dans } C} f(c) \cdot d_T(c)$$

Propriété du choix glouton

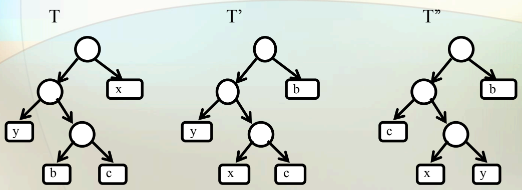
Lemme : Soit C un alphabet et f une fréquence d 'apparition sur C. Soient x et y de C ayant les fréquences les plus basses.

Il existe alors un codage préfixe optimal pour C dans lequel les mots de code pour x et y ont la même longueur et ne diffère que par le dernier bit.

[CLR90]

Démonstration

Montrer qu'il existe une solution optimale qui intègre le choix glouton réunissant *x* et *y* :



Chap. 4 : Algorithmes gloutons

Démonstration

Pourquoi le cout de l'arbre n'est pas dégradé?

- On supposera $f(b)<=f(c)$ et $f(x)<=f(y)$ d'où $f(x)<=f(b)$ et $f(y)<=f(c)$
- $B(T) - B(T') = \sum_{c \text{ dans } C} F(c) \,.\, d_T(c) - \sum_{c \text{ dans } C} F(c) \,.\, d_{T'}(c)$
  $= f(x)d_T(x) + f(b)d_T(b) - (f(x)d_{T'}(x) + f(b)d_{T'}(b))$
  $= f(x)d_T(x) + f(b)d_T(b) - (f(x)d_T(b) + f(b)d_T(x))$
  $= (f(b) - f(x))\,(d_T(b) - d_T(x)) >= 0;$
- De la même façon l'on pourra montrer que B(T') – B(T") est supérieur à 0;

Ainsi nous avons : B(T'') <= B(T); T'' est optimal;

Chap. 4 : Algorithmes gloutons

Propriété de sous structure optimale

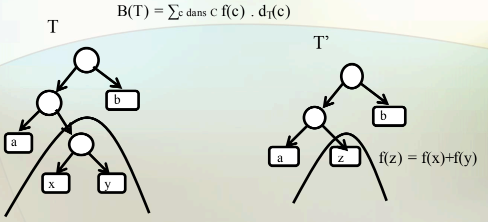
Lemme : Soit T un arbre binaire représentant un codage préfixe optimal pour C, soient 2 caractères x et y quelconques qui apparaissent comme feuille sœurs dans T, et soit z leur père.

Alors, en considérant z de fréquence f(x) + f(y), l 'arbre T '=T-{x,y} représente un codage préfixe optimal pour l'alphabet C '= C - {x,y} U {z}.

Démonstration

Montrer que la procédure « Huffman » calcul un codage optimal sur l'alphabet C / {x,y} u z;

$B(T) = \sum_{c \text{ dans } C} f(c) \,.\, d_T(c)$



$B(T) = A + (f(x)+f(y))\ (d_T(z)+1)$ $\quad$ $B(T') = A + (f(x)+f(y))\ d_T(z)$

Démonstration

Montrons que le cout B(T) de T peut être exprimé en fonction du cout B(T') de l'arbre T'.

- Rappel : $B(T) = \sum_{c \text{ dans } C} f(c) \,.\, d_T(c)$
- Pour tout c dans C / {x,y}, on a $f(c)d_T(c) = f(c)d_{T'}(c)$
- D'autre part $d_T(x) = d_T(y) = d_{T'}(z) + 1;$
- $f(x)d_T(x) + f(y)d_T(y) = (f(x) + f(y))\,.\,(d_{T'}(z) + 1)$
  $= f(z)\, d_{T'}(z) + f(x) + f(y)$

D'où $B(T) = B(T') + f(x) + f(y)$;

Algorithme glouton générique

Données : G
Résultat R

Begin
ensemble W={}
tant que (W ≠ G)
  choisir (d dans G \ W)
  R = R + traiter (d)
  W = W union {d}
fin tant que
Retourner R
End

Pour résumer

- Le paradigme « glouton » consiste à faire un choix local qui maximise un critère donné à un instant donné. Ce choix ne sera jamais remis en cause.

- Le paradigme glouton est adapté aux problèmes pour lesquels les deux propriétés « **du choix glouton** » et de « **sous structure optimale** » sont vérifiées;

- Dans ce chapitre nous avons étudié deux problèmes :
  - *Choix d'activité;*
  - *Codage de Huffman;*