Algorithmes Gloutons

Paradigme de conception
Propriété de choix glouton
Propriété de sous-structure optimale

Problème du choix d'activité
Compression de données et algorithme de Huffman

Auteur : Olivier Raynaud

Version de travail : rentrée 2020
Sensibilité :

Référence :

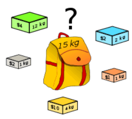
https://www.pinterest.fr/pin/495255290269224249/

Heuristique mentale

Problèmes au quotidien : des heuristiques mentales sont délcenchées pour résoudre de nombreuses situations de calcul dans notre quotidien. L'approche dite gloutonne s'impose très naturellement à notre esprit.

- Problème du rendu de la monnaie

- Problème du sac à dos



https://info.blaisepascal.fr/nsi-algorithmes-gloutons

wikipedia

Heuristique

Principe heuristique intuitive **:** Un **algorithme glouton** est un algorithme qui admet comme principe heuristique de faire un choix localement optimal à chaque étape du calcul afin de converger vers une solution globalement optimale.

[Wikipedia]

- Le choix dépend uniquement des choix passés mais en aucun cas des choix futurs, ni des solutions aux sous problèmes.

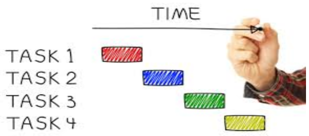
- *De très nombreuses applications*

Problème du choix d’activités

Problème d’optimisation

Problème du choix d’activité : 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.

- Trouver l’ensemble le plus grand possible d’activités compatibles entre elles.



https://researchleap.com/a-scenario-based-stochastic-time-cost-quality-trade-off-model-for-project-scheduling-problem/

Problème du choix d’activités

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 du choix d’activités

Algorithme glouton

```
Algorithm 2: Choix d’activité()
  Données: listeA : liste d’activités; i, j : entier;
  Résultat: res : liste d’activités;
  début
    l : liste; l.copie(listA)
    n ← l.taille;
    l.trier();
    res.inserer(l.contenu); l ← l.suivant;
    tant que (l) faire
      monActivite ← l.contenu;
      si (monActivite.debut ≥ res.dernier.fin) alors
        res.inserer(monActivite);
      l ← l.suivant;
    retourner res;
  fin
```

Problème du choix d'activités

❑ **Solution optimale**

Optimalité : L'algorithme Choix d'activités() calcule l'ensemble de taille maximale d'activités compatibles entre elles.

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 problème fait apparaître une propriété de sous structure optimale et que le même raisonnement peut être conduit sur S - {1}

Problème du choix d'activités

❑ **Démonstration**

La propriété de choix glouton : Montrer qu'il existe une solution optimale qui intègre le choix glouton.

Soit A une solution optimale ordonnée par horaire de fin croissante des activités, 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.

Problème du choix d'activités

❑ **Démonstration**

La propriété de sous structure optimale : 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.

Stratégie gloutonne

❑ Plan en 2 étapes essentielles

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

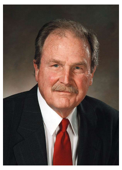
La propriété de sous structure optimale : Un problème fait apparaître une sous structure optimale si une solution optimale du problème contient la solution optimale des sous problèmes.

Codage de Huffman

❑ *Méthode de compression sans perte.*

Codage de Huffman : Le codage de Huffman est un algorithme de compression de données sans perte. Il applique un code longueur variable pour réprésenter un symbol du texte à transmettre.

[Wikipedia]



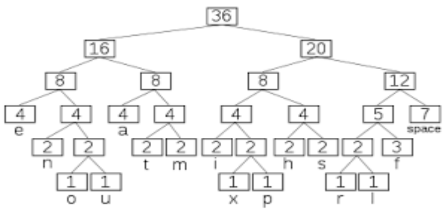
David Albert Huffman
1925-1999

➢ Un code de Huffman est optimal au sens de la plus courte longueur pour un codage par symbole et une distribution de probabilité connue.

Codage de Huffman

❑ Problème

Problème : Optimiser la taille du codage d'un fichier texte



Codage de Huffman

❑ Problème

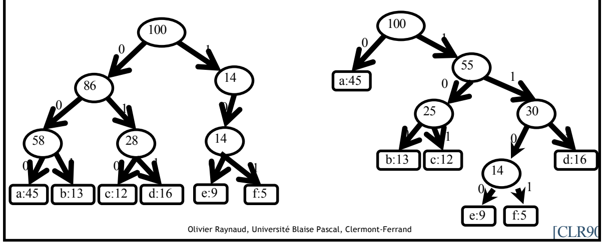
Problème : Optimiser la taille du codage d'un fichier texte

	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

Codage de Huffman

❑ Codage préfixe

Définition :
On appelle codage préfixe un codage ou aucun mot de code n'est le préfixe d'un autre mot de code.



Codage de Huffman

❑ Algorithme de Huffman

Algorithm 3: codage Huffman()

Données: $maListeFrequences$: liste de fréquences ;
Résultat: aB ;
début
- $monTas$: *tas* d'arbres binaires triés sur la fréquence ;
- $monTas \leftarrow nouveauTas(MaListeFrequence)$;
- **tant que** $(monTas.taille \geq 2)$ **faire**
  - $fGauche \leftarrow monTas.extraireMin()$;
  - $fDroit \leftarrow monTas.extraireMin()$;
  - $z : aB$;
  - $z \leftarrow nouveauAB(fg, fd)$;
  - $z.frequence \leftarrow fGauche.frequence + fDroit.frequence$;
  - $monTas.inserer(z)$;
- **retourner** $monTas.extraireMin$;

fin

Codage de Huffman

Modélisation

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.

Soit $B(T)$ le nombre de bits requis pour encoder le fichier. Nous avons :

$$B(T) = \sum_{c \in C} f(c).d_T(c)$$

Codage de Huffman

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.

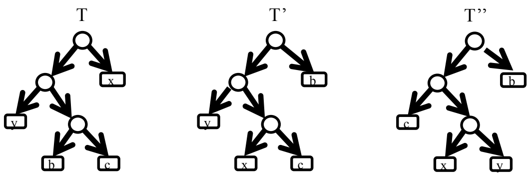
Il existe alors un codage préfixe optimal pour C dans lequel les mots de code pour **x** et **y** ont la même longueur et ne diffère que par le dernier bit.

[CLR90]

Codage de Huffman

Démonstration

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



Codage de Huffman

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

Sans perte de généralité, on supposera que :
$f(b) \leq f(c)$ et $f(x) \leq f(y)$ d'où $f(x) \leq f(b)$ et $f(y) \leq f(c)$;

$$
\begin{aligned}
B(T) - B(T') &= \sum_{c \in C} f(c).d_T(c) \; - \; \sum_{c \in C} f(c).d_{T'}(c) \\
&= f(x)d_T(x) + f(b)d_T(b) \; - \; (f(x)d_{T'}(x) + f(b)d_{T'}(b)) \\
&= f(x)d_T(x) + f(b)d_T(b) \; - \; (f(x)d_T(b) + f(b)d_T(x)) \\
&= (f(b) \; - \; f(x)) \; (d_T(b) \; - \; d_T(x)) \\
&\geq 0;
\end{aligned}
$$

De la même façon on pourra montrer que $B(T'')$ est inférieur à $B(T')$;
Ainsi nous avons $B(T'') \leq B(T)$; T'' est donc optimal ;

Codage de Huffman

❑ 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**}.

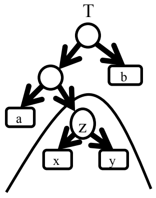
[CLR90]

Codage de Huffman

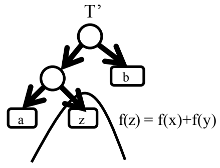
❑ Démonstration

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

$$B(T) = \sum_{c \in C} f(c).d_T(c)$$



$B(T) = A + (f(x)+ f(y)) (d_T(z)+ 1)$



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

Codage de Huffman

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

Pour tout caractère c dans $C \setminus \{x, y\}$, nous avons $f(c).d_T(c) = f(c).d_{T'}(c)$

D'autre part nous avons : $d_T(x) = d_T(y) = d_{T'}(z) + 1$

$$
\begin{aligned}
B(T) - B(T') &= f(x).d_T(x) + f(y).d_T(y) - f(z).d_{T'}(z); \\
&= (f(x) + f(y)).(d_{T'}(z) + 1) - (f(x) + f(y)).d_{T'}(z); \\
&= (f(x) + f(y)).d_{T'}(z) + (f(x) + f(y)) - (f(x) + f(y)).d_{T'}(z); \\
&= (f(x) + f(y))
\end{aligned}
$$

Nous obtenons : $B(T) = B(T') + f(x) + f(y)$

Pour résumer

- *Quelques points essentiels de l'exposé*
  - ✓ 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 permet de calculer la solution optimale aux problèmes pour lesquels les deux propriétés **« du choix glouton »** et de **« sous structure optimale »** sont vérifiées;
  - ✓ Dans cet exposé nous avons étudié deux problèmes :
    - *- Choix d'activité;*
    - *- Codage de Huffman;*

Bibliographie

- [HMU] **Introduction to Automata Theory, Langage and Computation** *J.E. Hopcroft, R. Motwani, J.D. Ullman* Edition Adison Wesley 2001;
- [CLM] **Introduction à l'algorithmique,** *T. Cormen, C. Leiserson, R. Rivest, Edition Dunod 2010;*
- [Wikipedia] **Multiples références dans le texte.**