Licence 2 Informatique
Avril 2015

TD : algorithmes gloutons

1 Egypte

On appelle fraction égyptienne une fraction de la forme $\frac{1}{n}$ avec $n \in \mathbb{N}^*$.

Q1.1 Soient a et b deux entiers (premiers entre eux) tels que $a < b$. Donner l'expression de la fraction égyptienne la plus grande parmi les fractions égyptiennes strictement plus petites que $\frac{a}{b}$ et l'expression de leur différence.

Q1.2 On considère l'algorithme suivant :

```
1:  function EGYPTE(a, b)                    ▷ (a, b) est un couple d'entiers avec a < b
2:      c ← a
3:      a ← numer(c/b)
4:      b ← denom(c/b)
5:      if a = 1 then
6:          return b
7:      else
8:          c ← a − b mod a
9:          d ← b div a + 1
10:         return (d, EGYPTE(c, b × d))
11:     end if
12: end function
```

Quelle est la sortie de l'algorithme avec l'entrée $(a, b) = (2, 2n + 1)$ où $n > 0$?

Q1.3 Quel est le rôle de cet algorithme ? Démontrer.

Q1.4 L'algorithme glouton proposé donne-t-il une décomposition en somme de fractions égyptiennes avec le minimum de termes possibles ?

Corrigé

Q1.1

$$\frac{a}{b} = \frac{1}{\lfloor \frac{b}{a} \rfloor + 1} + \frac{a - \text{irem}(b, a)}{b \times \left(\lfloor \frac{b}{a} \rfloor + 1 \right)}$$

Q1.2

$$\frac{2}{2n+1} = \frac{1}{n+1} + \frac{1}{(n+1)(2n+1)}$$

Q1.3 L'algorithme écrit le quotient $\frac{a}{b}$ sous la forme d'une somme de fractions égyptiennes à dénominateurs strictement croissants.

(a) L'algorithme se termine.
a, b désignent ci-dessous le numérateur et le dénominateur après réduction de la fraction $\frac{a}{b}$.
A chaque étape,

i. Soit $a = 1$ (c'est à dire $\text{irem}(b, a) = 0$) et l'algorithme se termine.

ii. Soit $a \neq 1$. Dans ce cas, on a $\text{irem}(b, a) \neq 0$ (puisque, la fraction étant irréductible, on a b non multiple de a). On a donc $0 < \text{irem}(b, a) < a$ et $0 < a - \text{irem}(b, a) < a$. Les fractions $\frac{a-\text{irem}(b,a)}{b\times\left(\lfloor\frac{b}{a}\rfloor+1\right)}$ successives ont (après réduction) des numérateurs strictement décroissants compris entre 1 et a : l'algorithme se termine.

(b) La liste des dénominateurs renvoyée par l'algorithme est strictement croissante.
On a $2\frac{b}{a} > 2\lfloor\frac{b}{a}\rfloor$ pour $a \neq 1$ puisque $\frac{a}{b}$ est réduite et $2\lfloor\frac{b}{a}\rfloor \geqslant \left\lfloor\frac{b}{a}\right\rfloor + 1$ puisque $\lfloor\frac{b}{a}\rfloor \geqslant 1$. On a donc $2\frac{b}{a} > \lfloor\frac{b}{a}\rfloor + 1$ lorsque $a \neq 1$ (et aussi, de façon claire, lorsque $a = 1$ puisque $b > 1$ à toute étape).
On a donc :

$$\frac{a}{b} - \frac{1}{\lfloor\frac{b}{a}\rfloor+1} < \frac{1}{\lfloor\frac{b}{a}\rfloor+1}$$

En d'autres termes, la fraction $\frac{a-\text{irem}(b,a)}{b\times\left(\lfloor\frac{b}{a}\rfloor+1\right)}$ est strictement plus petite que la fraction égyptienne précédente. Les fractions égyptiennes obtenues dans la suite de la décomposition seront donc strictement plus petites que la fraction égyptienne $\frac{1}{\lfloor\frac{b}{a}\rfloor+1}$ et donc à dénominateurs strictement plus grands.

Q1.4 Le nombre de termes n'est pas minimal.
L'algorithme renvoie par exemple la décomposition suivante :

$$\frac{19}{20} = \frac{1}{2} + \frac{1}{3} + \frac{1}{9} + \frac{1}{180}$$

alors que l'on a :

$$\frac{19}{20} = \frac{10+5+4}{20} = \frac{1}{2} + \frac{1}{4} + \frac{1}{5}$$

Ou encore :

$$\frac{5}{121} = \frac{1}{25} + \frac{1}{757} + \frac{1}{763309} + \frac{1}{873960180913} + \frac{1}{1527612795642093418846225}$$

alors que :

$$\frac{5}{121} = \frac{1}{33} + \frac{1}{121} + \frac{1}{363}$$

2 Les épreuves dans le gymnase

Dans un gymnase doivent se dérouler une série d'épreuves. Les épreuves ne sont pas seulement caractérisées par leurs durées : chaque épreuve est caractérisée par une date de début d_i et une date de fin f_i. On souhaite "caser" le plus possible d'épreuves, deux épreuves ne pouvant avoir lieu en même temps (leurs intervalles de temps doivent être disjoints).

Glouton 1. On trie les épreuves par durée croissante, on choisit la plus courte, puis la plus courte parmi celles qui lui sont compatibles, puis ... Ce choix mène-t-il au déroulement d'un nombre d'épreuves maximal ?

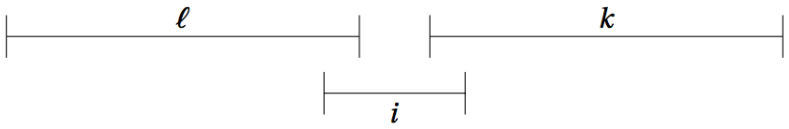
Glouton 2. On trie les événements par dates de commencement croissantes et on gloutonne : on choisit l'événement commençant le plus tôt, puis le plus tôt parmi les événements compatibles ... Même question.

Glouton 3. On trie cette fois les événements par nombre d'intersections croissant : on choisit d'abord celui qui intersecte le moins d'événements, puis... Même question.

Glouton 4. On trie les événements par dates de fin croissantes et on gloutonne : on choisit l'épreuve se terminant au plus tôt, puis l'épreuve se terminant au plus tôt parmi celles qui sont compatibles à la première... Même question.

Corrigé

Glouton 1. Non optimal. Contre-exemple :

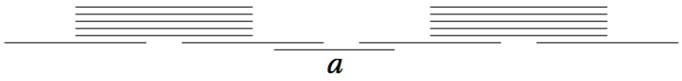


L'algorithme choisit i : une épreuve alors que deux sont possibles.

Glouton 2. Non optimal. Contre-exemple :

L'algorithme choisit une épreuve au lieu de plusieurs.

Glouton 3. Non optimal. Contre-exemple :



L'algorithme choisit a et trois événements seront choisis au total. Alors que l'on peut clairement en choisir 4.

Glouton 4. Optimal.

On note f la date de fin la plus petite.

Soit $\Gamma = \{f_1, f_2, \ldots, f_k\}$ un ensemble de dates de fin d'épreuves définissant une solution optimale avec $f_1 < f_2 < \cdots < f_k$.

Si $f \neq f_1$, on remplace une épreuve correspondant à f_1 par une épreuve correspondant à f, cela est possible puisque $f \leqslant f_1 < f_2 < \ldots$. L'ensemble $\Gamma' = \{f, f_2, \ldots, f_k\}$ est donc également optimal (même nombre d'épreuves).

On note ensuite f' la date de fin la plus petite parmi les dates de fin d'épreuves compatibles avec f (c'est à dire les épreuves de date de début $> f$).

Si $f_2 \neq f'$, on remplace une épreuve correspondant à f_2 par une épreuve correspondant à f', cela est possible puisque f' compatible avec f et $f' \leqslant f_2 < \ldots$. L'ensemble $\Gamma'' = \{f, f', \ldots, f_k\}$ est donc également optimal (même nombre d'épreuves).

...

3 Monnaie

Comment rendre une somme donné avec le minimum de pièces. Nous regarderons tout d'abord notre système monétaire : des pièces de 1, 2, 5, 10, 50, 100, 200.

Q3.1 Comment rendre 263 centimes d'euros ?

Q3.2 Trouvez un algorithme glouton pour ce problème.

Q3.3 Montrez que cet algorithme est optimal.

Q3.4 Qu'est ce qui se passe si les pièces ont comme valeur 1, 3, 4 ? Comment rendre la somme de 6 ?

Corrigé

Q3.1 Pour rendre la monnaie sur 263 centimes d'euros, on rend une pièce de 200, de 50, de 10, de 2 et de 1.

Q3.2 L'algorithme glouton pour ce problème est le suivant : on rend toujours la pièce de la plus grande valeur que l'on peut. On suppose que les valeurs des pièces sont stockées par ordre croissant dans un tableau `p` :

```
rendre_pieces(entier S):
-- rendu := 0
-- i := 0
-- r := tableau rempli de 0 de même taille que p
-- tant que (S-rendu != 0) faire
---- si p[i] > S-rendu alors
------ i := i+1
---- sinon alors
------ r[i] := r[i] + 1
------ rendu := rendu + p[i]
-- retourner r
```

Attention! Cet algorithme ne termine pas toujours, essayez de trouver quels sont les cas dans lesquels il ne termine pas.

Q3.3 Montrons que l'algorithme est optimal :

- Montrons d'abors la propriété de choix glouton : Supposons qu'il existe $s > 200$ pour lequel le rendu de monnaie optimal ne contient pas de pièce de 200. On peut avoir au maximum une pièce de 100, car sinon ou pourrait remplacer les deux pièces de 100 par une de 200, au maximum une pièce de 50, 4 pièces de 10, 1 pièce de 5 et 2 pièces de 2 (sinon on peut remplacer par 1 pièce de 5 et une pièce de 1) et 1 pièce de 1. La somme maximale que l'on peut donc obtenir de manière optimale sans pièce de 200 est $1+2\times 2+5+4\times 10+50+100=200$ qui est inférieure ou égale à 200 (et qui pourrait être remplacée par 200).
- Montrons la propriété de sous-structure optimale. Soit P une solution optimale pour une somme s et c le choix glouton, $P' = P \setminus \{c\}$ est une solution optimale pour $s-c$, car si il y a une meilleure solution P'' pour $s-c$, alors $P'' \cup \{c\}$ est une meilleure solution que P pour s.

Q3.4 Si les pièces ont comme valeurs 1,3,4, alors l'algorithme glouton va rendre la monnaie sous la forme d'une pièce de 4 et de deux pièces de 1, alors que rendre deux pièces de 3 aurait été optimal, il ne fournit donc pas la solution optimale.

4 Un problème d'ordonnancement

Étant donné un ensemble $E = \{1, 2, \ldots, n\}$ de taches unitaires (toutes ont pour durée une unité de temps) de dates échues $d_1, d_2, \ldots, d_n$ avec $1 \leq d_1 \leq d_2 \leq \ldots \leq d_n \leq n$ et de pénalités $w_1, w_2, \ldots, w_n$, on cherche un ordonnancement des taches $1, 2, ..., n$ qui minimise le cumul des pénalités pour non respect des dates échues.
Un ordonnancement est une permutation des taches $1, \ldots, n$. Un ensemble de taches F est indépendant s'il existe un ordonnancement des taches de F tel qu'aucune ne soit en retard sur sa date échue.

Q4.1 Montrez que le calcul d'un ordonnancement optimal équivaut à trouver un ensemble indépendant de taches de E dont la somme des pénalités soit le plus grand possible.

Q4.2 Soit i une tache de pénalité maximale. Existe-t-il un ensemble indépendant optimal qui contient i ?

Q4.3 En déduire une méthode glouton qui fournit une solution optimale.

Corrigé

Q4.1 Soit H un ordonnancement, on peut diviser H en deux sous-ensembles de tâches F et G, où F est le sous-ensemble des tâches effectuées avant l'échéance et est donc un ensemble indépendant et G contient le reste des tâches. Notons p_I la somme des pénalités d'un ensemble de tâches I. On a la relation suivante :

$$p_E = p_G + p_F$$

Le calcul d'un ordonnancement optimal revient à trouver un ordonnancement minimisant p_G, ce qui revient à trouver un ordonnancement maximisant p_F. Comme F est indépendant, le calcul d'un ordonnancement optimal revient à trouver un ensemble indépendant maximal de E.

Q4.2 Soit F un ensemble indépendant optimal, par définition, toute les tâches de F ont des pénalités inférieures ou égales à w_i, on peut donc remplacer n'importe quelle tâche de F dont la date d'échance est inférieure à d_i par i, et obtenir ainsi un ensemble indépendant dont la somme des pénalités est plus grande ou égale à celle de F.

Q4.3 Voici un algorithme glouton pour le problème :

```
ordonnancement(tableau d, tableau w)
-- L := tri des i par w[i] décroissants
-- F := vide
-- pour i=1 jusqu'à n faire :
---- si F + L[i] indépendant alors
------ F := F + L[i]
-- retourner F
```

Si F est ordonné par i croissants et plus par poids, alors vérifier que F est indépendant se fait en temps au plus n.

Montrons que cet algorithme est optimal, après avoir montré la propriété du choix glouton à la question 2, montrons maintenant la propriété de sous-structure optimale : Soit F une solution optimale, et i la tâche effectuée le plus tôt, alors $F' = F \setminus \{i\}$ est une solution optimale pour le problème $E' = \{j : d_j > t_i\}$ où t_i est la date à laquelle i est effectuée dans F : En effet, supposons que F' ne soit pas optimal pour E', et que la solution optimale soit G, alors $G \cup \{i\}$ serait une solution optimale pour E et F n'en serait pas une.

5 Le coût de la non panne sèche

Le professeur Bell conduit une voiture entre Amsterdam et Lisbonne sur l'autoroute E10. Son réservoir, quand il est plein, contient assez d'essence pour faire n kilomètres, et sa carte lui donne les distances entre les stations-service sur la route.

Q5.1 Donnez une méthode efficace grâce à laquelle Joseph Bell pourra déterminer les stations-service où il peut s'arrêter, sachant qu'il souhaite faire le moins d'arrêts possible.

Q5.2 Démontrez que votre stratégie est optimale.

Corrigé

Q5.1 La solution consiste à s'arrêter le plus tard possible, c'est-à-dire à la station la plus éloignée du dernier point d'arrêt parmi celles 'a moins de n kilomètres de ce même point.

Q5.2 *On procède de la même manière que pour prouver l'optimalité de l'algorithme glouton pour la location de voiture : on considère une solution $F = (x_1, x_2, ..., x_p)$ fournie par notre algorithme et une solution optimale $G = (y_1, y_2, ..., y_q)$, on a donc $p \geq q$ et on cherche à montrer que $p = q$ —ici les deux suites sont bien sûr des suites de stations-service, triées de celle la plus proche du point de départ à celle la plus éloignée.*

Soit k le plus petit entier tel que :

$$\forall i < k, x_i = y_i, \text{ et } x_k \neq y_k.$$

Par définition de notre algorithme, on a $x_k > y_k$. On construit à partir de la solution optimale $G = (y_1, y_2, ..., y_{k-1}, y_k, y_{k+1}, ..., y_q)$ la suite de stations-service $G' = (y_1, y_2, ..., y_{k-1}, x_k, y_{k+1}, ..., y_q)$. G' est une liste de même taille. Montrons que c'est aussi une solution —et donc une solution optimale. Il nous faut vérifier qu'il n'y a pas de risque de panne sèche c'est-à-dire que :

- *$x_k - y_{k-1} \leq n$. F est une solution, donc $x_k - x_{k-1} \leq n$. Par conséquent, $x_k - y_{k-1} = x_k - x_{k-1} \leq n$.*
- *$y_{k+1} - x_k \leq n$. G est une solution, donc $y_{k+1} - y_k \leq n$. D'où $y_{k+1} - x_k < y_{k+1} - y_k \leq n$. Si $y_{k+1} < x_k$, on peut en fait supprimer y_{k+1} de G' et obtenir une solution strictement meilleure ce qui contredirait l'optimalité de G.*

G' est donc bien une solution optimale de notre problème.

Nous avons donc construit à partir de G une solution G' qui contient un élément en commun de plus avec F. On itère jusqu'à ce que l'on ait une solution optimale contenant F. Alors $p \leq q$, ce qui prouve le résultat attendu.

6 Bipartition

Étant donné un ensemble de n nombres, répartissez les en 2 sous-ensembles tels que la somme des éléments du premier soit égal à la somme des éléments du second. Proposez des algorithmes gloutons pour résoudre ce problème. Appliquez les aux ensembles suivants :

$$\{2, 10, 3, 8, 5, 7, 9, 5, 3, 2\}$$

$$\{771, 121, 281, 854, 885, 734, 486, 1003, 83, 62\}$$

Corrigé

Un algorithme glouton possible est de trier l'ensemble par ordre décroissant et d'ajouter ensuite chaque nombre au sous-ensemble dont le somme est la plus petite. Cela donne respectivement pour les deux ensmbles de l'exemple :

- $E_1 = \{10, 7, 5, 3, 2\}$ et $E_2 = \{9, 8, 5, 3, 2\}$ les sommes sont les mêmes : 27.
- $E_1 = \{1003, 771, 486, 281, 83\}$ et $E_2 = \{885, 854, 734, 121, 62\}$ les sommes ne sont pas les mêmes mais sont proches : 2624 et 2656.

L'algorithme glouton n'est pas optimal mais est une 4/3-approximation polynômiale. Par ailleurs, ce problème est NP-complet.