Algorithmique
TD n° 7 : Algorithmes gloutons

Exercice 1 : Chansons (examen 23-24, session 2)

On veut enregistrer n chansons de durée $d_1, d_2, ...d_n$ sur un support sans accès direct (p. ex. une bande magnétique) : on veut savoir dans quel ordre enregistrer les chansons afin de minimiser le temps de « rembobinage » nécessaire pour y accéder depuis le début de la bande.

1. Donner un algorithme pour vérifier que la bande de capacité C (NB : C est une durée) peut stocker toutes les chansons.
2. On considère 4 chansons de durées $d_1 = 2$, $d_2 = 8$, $d_3 = 14$ et $d_4 = 7$. Calculer le temps moyen de rembobinage pour accéder aux chansons lorsqu'on les place dans l'ordre $3, 1, 2, 4$ (la chanson 3, puis la 1, *etc.*). Et si on place les chansons dans l'ordre $4, 2, 1, 3$?
3. Etant donnée une fonction $o : \{1, \ldots, n\} \to \{1, \ldots, n\}$ définissant un ordre des chansons (la chanson $o(1)$, puis la chanson $o(2)$, etc.) donner l'expression du temps de rembobinage moyen (c-à-d la grandeur à minimiser par l'algorithme).
4. Proposer un algorithme glouton pour résoudre le problème. Justifier votre algorithme.

Exercice 2 : Stable d'un arbre

Etant donné un graphe non-orienté $G = (S, A)$, on appelle un *stable* (ou un ensemble indépendant) un sous-ensemble de sommets $E \subseteq S$ tel que pour tous sommets x et y dans E, il n'y a pas d'arête reliant x à y (c'est-à-dire, deux sommets adjacents ne peuvent pas se trouver ensemble dans E). Rechercher un stable de taille maximum est un problème difficile (NP-complet). Mais c'est plus simple lorsqu'on s'intéresse à des arbres...

1. Montrer que pour toute feuille x d'une forêt F (un ensemble d'arbres disjoints), il existe un stable de taille maximum pour F qui contient x.
2. En déduire un algorithme qui construit un stable de taille maximum pour une forêt F.
3. Affiner l'algorithme pour calculer un stable de taille maximum pour un arbre A à partir de sa racine r en supposant que pour chaque noeud x on dispose sa liste de successeurs immédiats (ses fils) avec $\mathsf{succ}(x)$. Quelle est la complexité de l'algorithme ?

Exercice 3 : fractions égyptiennes

La façon égyptienne d'écrire des fractions $\frac{a}{b}$ avec $a, b \in \mathbb{N}$ et $a < b$ consiste à les écrire comme une somme de *fractions unitaires* (de la forme $\frac{1}{d}$) *distinctes*. Par exemple

$$\frac{2}{3} = \frac{1}{3} + \frac{1}{4} + \frac{1}{12}$$

Le but de cet exercice est de minimiser le nombre de fractions utilisées dans la somme.
Comment peut-on écrire $\frac{5}{6}$?
On considère les deux algorithmes suivants, dus tous les deux à Fibonacci (13ème siècle)

Algorithme 1

1. écrire $\frac{a}{b}$ comme la somme de a termes $\frac{1}{b}$;
2. tant qu'il reste des termes identiques dans la somme, en prendre un de plus petit dénominateur et le décomposer grâce à l'identité

$$\frac{1}{b} = \frac{1}{b+1} + \frac{1}{b(b+1)}$$

Ainsi pour l'exemple $\frac{2}{3}$ on commence par écrire $\frac{1}{3} + \frac{1}{3}$ puis l'on décompose le deuxième $\frac{1}{3}$ en $\frac{1}{3+1} + \frac{1}{3(3+1)}$ ce qui donne la décomposition précédente.

Algorithme 2 (glouton)

1. partir avec la somme d'une fraction $\frac{a}{b}$ avec $a < b$;
2. tant qu'elle n'est pas unitaire, appliquer à la dernière fraction de la somme l'identité

$$\frac{a}{b} = \frac{1}{\lceil b/a \rceil} + \frac{a - (b \bmod a)}{b\lceil b/a \rceil}$$

Par exemple $\frac{7}{15} = \frac{1}{3} + \frac{6}{45} = \frac{1}{3} + \frac{2}{15} = \frac{1}{3} + \frac{1}{8} + \frac{1}{120}$

Pour chacun de ces deux algorithmes,

1. Appliquer l'algorithme sur les exemples $\frac{2}{5}$ et $\frac{2}{11}$;
2. (Commencer à) appliquer l'algorithme 1 sur l'exemple $\frac{5}{121}$;
3. montrer qu'il termine en produisant une décomposition correcte ;
4. si elle est optimale (en nombre de termes), le prouver ; sinon, exhiber un contre-exemple ;
5. peut-on donner une borne de la longueur de la suite ainsi produite ?

Exercice 4 : algorithme de Borůvka

Il s'agit d'un algorithme d'arbre couvrant de poids minimum écrit en 1926 par Otakar Borůvka pour concevoir un réseau de distribution électrique pour la Moravie du Sud. Il travaille sur des multi-graphes $G = (V, E)$: boucles autorisées, plusieurs arêtes possibles entre deux sommets donnés. Chaque arête a un poids strictement positif. On suppose que G est connexe.

Procédure `Borůvka(`G`)`

- $F \leftarrow \emptyset$
- **tant que** G *a plusieurs sommets* **faire**
  - supprimer les boucles de G
  - **pour** *toute paire de sommets* x *et* y **faire**
    - **si** *plusieurs arêtes entre* x *et* y **alors**
      - ne garder que la plus légère
  - **pour** *tout sommet* x **faire**
    - $e_x \leftarrow \{x, y\}$ l'arête ***minimale*** parmi les arêtes touchant x *(parmi les arêtes de poids minimal, celle avec* y *minimal)*
    - $F \leftarrow F \cup \{e_x\}$
  - **pour** *toute arête* e_x *choisie à l'étape précédente* **faire**
    - contracter e_x *(ses deux extrémités deviennent un seul sommet)*
- **retourner** F

1. Faire tourner l'algorithme sur l'exemple à droite.
2. Quelle est sa complexité ? En particulier, donner une borne pour le nombre de tours de la boucle externe, et réfléchir à l'implémentation des opérations des boucles *pour*.
3. À quoi la règle « y minimal » sert-elle ? Qu'est-ce qui se passe, si on a par exemple un graphe avec 3 sommets tous connectés avec une arête de même valeur ?
4. Montrer la correction de cet algorithme.
5. On dit qu'un algorithme est *parallélisable* si son temps d'exécution peut être divisé par k en utilisant k processeurs. Cet algorithme est-il parallélisable ?