Examen de Algorithmique

Jeudi 19 juin 2008
Notes de cours autorisées

Exercice 1 : Rendre la monnaie

On souhaite construire un algorithme pour obtenir une somme S (S est un entier) à partir de $n \geqslant 1$ sortes de pièces de valeur $v_1, \ldots, v_n$. On suppose que pour chaque valeur v_i, le nombre de pièces disponibles est illimité. On suppose de plus que $v_n > v_{n-1} > \ldots > v_1$ et que $v_1 = 1$.

1. On considère l'algorithme suivant où les v_i sont stockées dans un tableau *Val*.
   Le tableau *Sol* est utilisé pour décrire une manière de composer une somme : *Sol*[i] donne le nombre de pièces de valeur v_i. La somme correspondante est donc : $\Sigma_{i=1}^{n} Sol[i] \cdot Val[i]$.

```
Procédure Somme(S, Val)
begin
    Sol := tableau d'entiers de taille n initialisé avec[0, ..., 0];
    j := n ;
    tant que S > 0 faire
        Sol[j] := S/Val[j] ;
        S := S%Val[j] ;
        j := j − 1;
    retourner Sol
end
```

   On rappelle que / désigne la division entière et % le reste de la division entière.

   (a) Montrer que cet algorithme termine. Montrer qu'il fournit une solution au problème (c'est-à-dire une manière de composer S). A quelle famille d'algorithme appartient-il ?

   (b) Montrer qu'il fournit une solution au problème initial.

   (c) Donner le résultat renvoyé dans le cas $S = 236$ et $v_1 = 1$, $v_2 = 3$, $v_3 = 50$ et $v_4 = 70$.

2. On veut maintenant trouver un algorithme qui renvoie le NOMBRE **minimal** de pièces pour composer la somme S (à partir des pièces de valeur $v_1, \ldots, v_n$).

   (a) Montrer que compter le nombre de pièces de la solution renvoyée par l'algorithme `Somme` n'est pas correct pour résoudre ce nouveau problème.

   (b) Proposer un algorithme qui résout le problème. On pourra utiliser et calculer le nombre minimal $N_{S',k}$ de pièces de valeur $v_1, \ldots, v_k$ nécessaire pour constituer la somme S', pour $S' \leqslant S$ et $k \leqslant n$.

   (c) Adapter votre algorithme pour calculer une solution optimale (c'est-à-dire un ensemble de pièces de somme S et de taille minimale).

3. Comparer la complexité de l'algorithme `Somme` avec celui établi à la question précédente.

4. Comment résoudre le problème lorsque le nombre de pièces de chaque catégorie $i = 1, \ldots, n$ est limité : étant donnés une somme S, un ensemble E de pièces composé de n_1 pièces de valeur v_1, n_2 pièces de valeur v_2, *etc.*, trouver le nombre minimal de pièces nécessaires pour réaliser la somme S et correspondant à une solution faisable avec E.

Exercice 2 : Arbre binaire de recherche

On considère des arbres binaires de recherche.

1. Que fait l'algorithme ci-dessous ? Et quelle est sa complexité ?

```
Procédure Truc(a)
begin
    si a non vide alors
        Truc(a.fd) ;
        print a.val ;
        Truc(a.fg) ;
end
```

NB : *fg*, *fg* et *val* représentent les champs habituels "fils gauche", "fils droit" et "valeur" des arbres binaires de recherche.

2. Donner un algorithme pour trouver la clé maximale stockée dans un arbre binaire de recherche. Evaluer sa complexité.

Exercice 3 : Algorithme sur les graphes

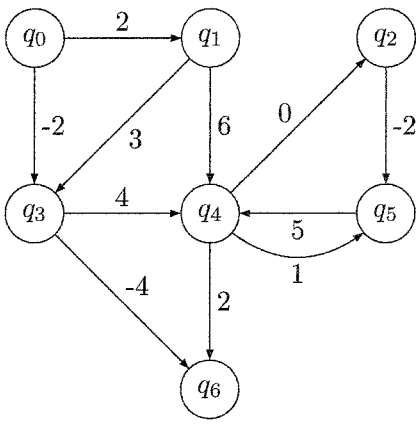
On considère l'algorithme suivant où $G = (S, A, w)$ est un graphe orienté valué avec $w : A \to \mathbb{R}$ et s est un sommet de G :

```
Procédure Algo-graphe(G, s)
begin
    d : tableau associant à chaque sommet une valeur dans ℝ ∪ {∞};
    pour v ∈ S faire
        d[v] := ∞
    d[s] := 0;
    pour i := 1 ... |S| − 1 faire
        pour tout (u, v) ∈ A faire
            Relax(u, v, d, w)
    retourner d
end
```

Avec la procédure `Relax` définie comme suit :

```
Procédure Relax(u, v, d, w)
begin
    si d[v] > d[u] + w(u, v) alors
        d[v] := d[u] + w(u, v)
end
```

1. Appliquer l'algorithme 3 au graphe ci-dessous et q_0.
2. Montrer que si G ne contient pas de cycle négatif atteignable depuis s, alors à la fin de l'algorithme on a : $\mathtt{d}[v] = \delta(s, v)$ pour tout sommet $v \in S$ où $\delta(s, v)$ désigne la distance d'un plus court chemin entre s et v.



3. Evaluer la complexité de l'algorithme `Algo-graphe`.
4. Montrer que si G contient un cycle négatif atteignable depuis s, alors il existe une arête $(u, v) \in A$ telle que, à la fin de l'algorithme, on a $\mathtt{d}[v] > \mathtt{d}[u] + w(u, v)$.
   Adapter l'algorithme `Algo-graphe` pour signaler la présence de cycle négatif.
5. Comparer cet algorithme avec l'algorithme de Dijkstra vu en cours.

Exercice 4 : Arbres couvrants minimaux

Que donnent les algorithmes de Prim et Kruskal dans le cas où le graphe G n'est pas connexe ?

Exercice 5 : Graphes

On considère un graphe orienté $G = (S, A)$ avec $S = \{x_1, \ldots, x_n\}$. On note M la matrice d'adjacence de G, c'est à dire la matrice booléenne $(\alpha_{ij})_{1 \leqslant i,j \leqslant n}$ telle que :

$$\alpha_{ij} \stackrel{def}{=} \begin{cases} \mathit{Vrai} & \text{si } (x_i, x_j) \in A \\ \mathit{Faux} & \text{sinon} \end{cases}$$

On définit la somme et le produit de matrices booléennes de manière standard : la multiplication est remplacée par la conjonction ($\wedge$), et l'addition par la disjonction ($\vee$). Ainsi soient A, B et C trois matrices booléennes $n \times n$ avec les coefficients a_{ij}, b_{ij} ou c_{ij}, on a :

- $C \stackrel{def}{=} A + B$ ssi $c_{ij} = a_{ij} \vee b_{ij}$;
- $C \stackrel{def}{=} A \cdot B$ ssi $c_{ij} = \bigvee_{k=1\ldots n} (a_{ik} \wedge b_{kj})$;

On note `Id` la matrice booléenne $(\gamma_{ij})_{1 \leqslant i,j \leqslant n}$ telle que $\gamma_{ii} \stackrel{def}{=} \mathit{Vrai}$ et $\gamma_{ij} \stackrel{def}{=} \mathit{Faux}$ si $i \neq j$.
On définit la suite de matrice $M^{(k)}$ pour $k = 0, \ldots, n-1$ de la manière suivante :

- $M^{(0)} \stackrel{def}{=} \mathtt{Id}$
- $M^{(k)} \stackrel{def}{=} M \cdot M^{(k-1)}$ pour $k = 1, \ldots, n-1$.

1. Montrer que $M^{(k)}$, pour $k = 0, \ldots n-1$, représente la matrice d'accessibilité en exactement k transitions de G.

2. Soit M^* la matrice booléenne correspondant à la fermeture réflexive et transitive de la relation induite par G : le coefficient γ_{ij} de M^* est *Vrai* si et seulement si il existe un chemin (de longueur quelconque) de x_i à x_j dans G.
   Montrer que $M^* = \sum_{i=0}^{n-1} M^{(i)}$
   Quelle est la complexité de l'algorithme qui calcule M^* à partir des $M^{(k)}$?
3. Adapter l'algorithme de Floyd pour calculer M^* plus efficacement.
   Donner sa complexité.