Examen d'algorithmique

Mardi 8 novembre 2016 14h30–16h30 / Aucun document autorisé

Mode d'emploi : Le barème est donné à titre indicatif. **La qualité de la rédaction des algorithmes et des explications sera très fortement prise en compte pour la note.** On peut toujours supposer une question résolue et passer à la suite. L'annexe contient l'algorithme de parcours en profondeur et l'algorithme de Bellman-Ford.

Exercice 1 : Autour du parcours en profondeur - 6 points

1. Appliquer l'algorithme de parcours en profondeur sur le graphe G (donc PPC(G)) de la Figure 1 : donner les dates d et f obtenues pour chaque sommet et la table Π.
2. Modifier l'algorithme pour qu'il s'arrête lorsqu'il détecte un circuit, et affiche le circuit trouvé. Justifier votre algorithme. Donner sa complexité.

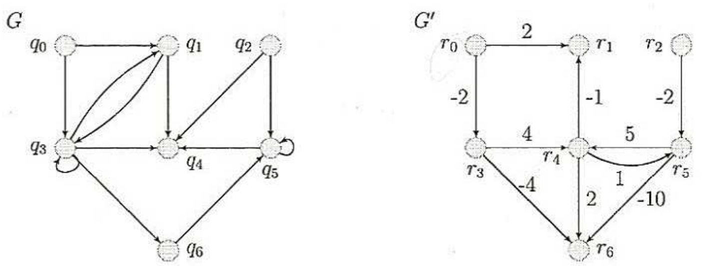

FIGURE 1 – Graphes G et G' pour les exercices 1 et 2.

Exercice 2 : Autour de l'algorithme de Bellman-Ford - 6 points

1. Appliquer l'algorithme de Bellman-Ford sur le graphe orienté valué G' de la Figure 1 à partir du sommet $\overset{r_0}{q_0}$ (donc PCC-Bellman-Ford(G, r_0)) : Donner la valeur d obtenue pour chaque sommet après chaque itération de la boucle principale. Donner également la table Π à la fin de l'algorithme.
2. Que fait cet algorithme ? Justifier votre réponse.
3. Montrer qu'après l'itération i, $\mathsf{d}[x] = \cancel{\delta(s,x)}$ si il existe un PCC de s à x avec au plus i transitions. $\delta(r_0, x)$

Université Paris Diderot L3 Informatique Année 2016–2017

Exercice 3 : Relation d'accessibilité – 8 points

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 :

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

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

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

On note Id la matrice booléenne « identité » définie par $(\gamma_{ij})_{1 \leqslant i,j \leqslant n}$ telle que $\gamma_{ii} \overset{\text{def}}{=} \texttt{Vrai}$ et $\gamma_{ij} \overset{\text{def}}{=} \texttt{Faux}$ si $i \neq j$.

On définit la suite de matrices $M^{(k)}$ pour $k = 0, \ldots, n-1$ de la manière suivante :

- $M^{(0)} \overset{\text{def}}{=} \texttt{Id}$
- $M^{(k)} \overset{\text{def}}{=} M \cdot M^{(k-1)}$ 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.
   (a) Montrer que $M^* = M^{(0)} + M^{(1)} + M^{(2)} + \ldots + M^{(n-1)}$
   (b) Quelle est la complexité de l'algorithme qui calcule M^* à partir des $M^{(k)}$?
3. Adapter l'algorithme de Floyd-Warshall pour calculer M^* plus efficacement. Justifier votre algorithme.
   Donner sa complexité.

Annexe

On rappelle ci-dessous l'algorithme de parcours en profondeur : la procédure de base PP et la procédure principale PPC.

```
Procédure PP(G, s)
//G = (S, A)
begin
    Couleur[s] := gris;
    temps++;
    d[s] := temps;
    pour chaque (s, u) ∈ A faire
        si Couleur[u] = blanc alors
            Π[u] := s;
            PP(G, u);
    Couleur[s] := noir;
    temps++;
    f[s] := temps;
end
```

```
Procédure PPC(G)
//G = (S, A)
begin
    pour chaque x ∈ S faire
        Couleur[x] := blanc;
        Π[x] := nil;
    temps := 0;
    pour chaque x ∈ S faire
        si Couleur[x] = blanc alors
            PP(G, x);
end
```

On rappelle ci-dessous l'algorithme de Bellman-Ford :

```
Procédure PCC-Bellman-Ford(G, s)
//G = (S, A, w) : un graphe orienté valué
//s ∈ S : un sommet origine.
begin
    pour chaque u ∈ S faire
        Π[u] := nil
        d[u] := { 0   si u = s
                { ∞   sinon
    pour i = 1 à |S| − 1 faire
        pour chaque (u, v) ∈ A faire
            si d[v] > d[u] + w(u, v) alors
                d[v] := d[u] + w(u, v)
                Π[v] := u
    pour chaque (u, v) ∈ A faire
        si d[v] > d[u] + w(u, v) alors
            return (⊥, −, −)
    return (⊤, d, Π)
end
```