Examen d'algorithmique

Mardi 20 juin 2017 15h30–17h45 / Aucun document autorisé

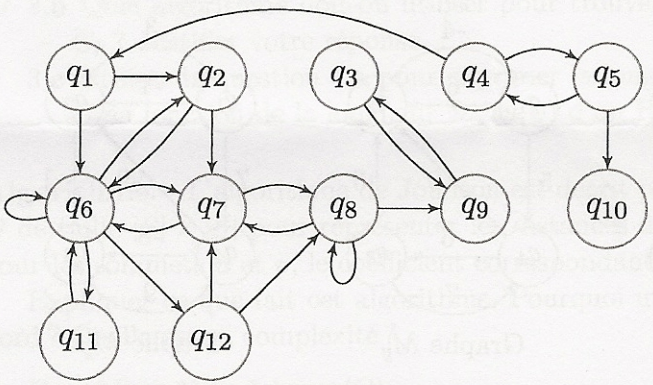
★★ L'annexe contient plusieurs algorithmes vus en cours. ★★

Exercice 1 : Recherche des composantes fortement connexes – (4 points)

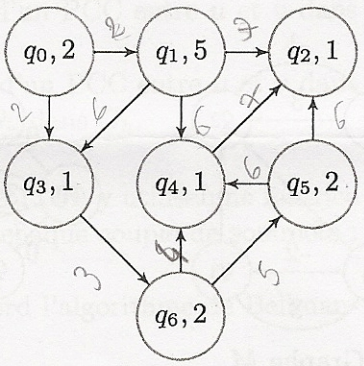
On rappelle l'algorithme vu en cours pour la recherche des composantes fortement connexes d'un graphe orienté $G = (S, A)$:

1. Appeler PPC(G) pour obtenir des dates f de chaque sommet ;
2. Construire G^t (le graphe transposé de G) ;
3. Appeler PPC(G^t) en énumérant les sommets par date f[−] décroissante ;
4. Retourner les arbres de parcours obtenus par PPC(G^t) : ce sont les CFC.

Appliquer l'algorithme au graphe de la figure (1a) en détaillant chaque étape : on donnera le résultat du premier parcours sous la forme des dates de début et de fin de traitement de chaque sommet (coloriage en gris et en noir), le graphe transposé, le résultat du parcours en profondeur du graphe transposé avec les dates et l'arborescence de parcours. On en déduira les composantes fortement connexes.



(a) Graphe pour l'exercice 1.



(b) Graphe pour la question 2 de l'exercice 2.

Exercice 2 : Problèmes et graphes – (6 points)

1. Tom a une chèvre, un loup et un chou. Il veut traverser une rivière avec une petite barque où il ne peut emporter qu'un seul passager (loup, chèvre ou chou) en plus de lui. Il ne peut pas laisser la chèvre seule avec le loup car il la mangerait, ni la chèvre seule avec le chou... Trouver une solution en le modélisant sous la forme d'un graphe et en appliquant un algorithme classique dessus.
2. On dispose d'un graphe orienté valué $G = (S, A, w)$ où la fonction $w : S \to \mathbb{R}^+$ associe une valeur réelle positive aux noeuds (et non aux arcs). La valeur $w(s)$ représente un coût à payer (ou une distance) lorsqu'on arrive dans le sommet s. Ainsi le chemin $s \to s' \to s''$ aura un coût égal à $w(s') + w(s'')$, et plus généralement un chemin $q_1 \to q_2 \to \ldots \to q_n$ aura un coût $\sum_{i=2\ldots n} w(q_i)$. Proposer une méthode pour trouver les coûts minimum pour atteindre chaque sommet depuis un sommet de départ s donné. Appliquer la sur le graphe de la figure (1b) depuis le sommet q_0.

Problème : Plus courts chemins – (10 points)

On s'intéresse au calcul des longueurs des plus courts chemins (PCC) dans un graphe orienté et valué $G = (S, A, w)$. Comme en cours, on note $\delta_G(x, y)$ la longueur d'un PCC entre x et y dans le graphe G.

Dans cet exercice, on va utiliser deux transformations de G :

- La première transformation est le graphe $G^+ = (S^+, A^+, w^+)$ tel que $S^+ = S \cup \{x_0\}$ où x_0 est un nouveau sommet, $A^+ = A \cup \{(x_0, u) \mid u \in S\}$ et : $w^+(u,v) = \begin{cases} w(u,v) & \text{si } (u,v) \in A \\ 0 & \text{si } u = x_0 \text{ et } v \in S \end{cases}$

G^+ contient donc les sommets et les arcs de G avec les mêmes poids, mais contient en plus un nouveau sommet x_0 relié à tous les autres par des arcs de poids nul.

- Et étant donnée une fonction f qui associe à chaque sommet de S une valeur dans $\mathbb{R}$ (c.-à-d. $f : S \to \mathbb{R}$), on définit la seconde transformation comme le graphe $G_f = (S, A, w_f)$ où $w_f(u,v) = w(u,v) + f(u) - f(v)$. **$G_f$ a donc les mêmes sommets et les mêmes arcs que G, seuls les poids des arcs changent.**

La figure 2 présente un graphe M, son graphe associé M^+ ainsi que deux exemples de graphe "M_f" : le graphe M_g pour la fonction g définie par : $g(q_1) = 2$, $g(q_2) = -1$, $g(q_3) = 4$ et $g(q_4) = 0$ et le graphe $M_{g'}$ pour $g'(q_1) = -1$, $g'(q_2) = 3$, $g'(q_3) = 1$ et $g'(q_4) = -5$.

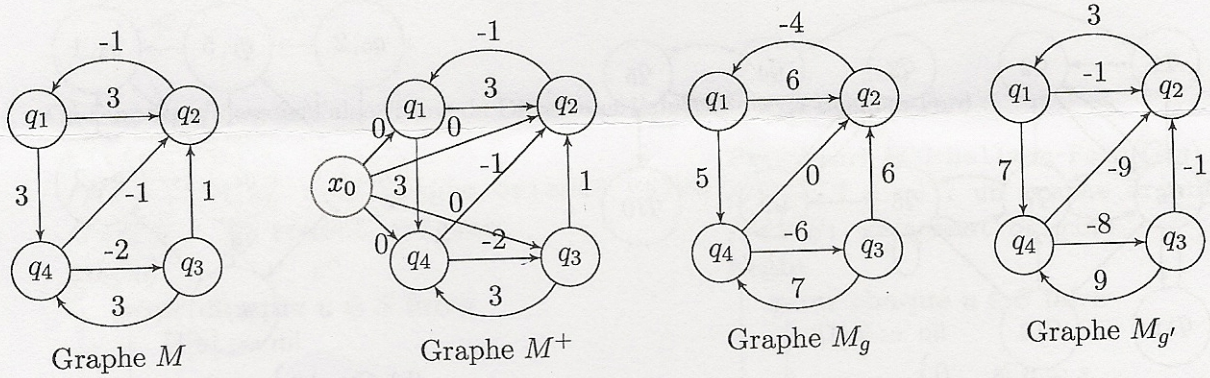


FIGURE 2 – Exemple d'un graphe M, sa transformation M^+ et deux transformations M_g et $M_{g'}$ pour les deux fonctions différentes g et g'.

On suppose donnés un graphe $G = (S, A, w)$ et une fonction $f : S \to \mathbb{R}$.

Questions de cours.

0.a Etant donnés deux sommets u et v, donner les conditions nécessaires et suffisantes pour qu'il existe un plus court chemin entre u et v dans G.

0.b Donner la complexité des algorithmes de Dijkstra et de Bellman-Ford. Quelle est la différence importante entre ces deux algorithmes ?

0.c Que signifie le booléen renvoyé par l'algorithme de Bellman-Ford ?

Propriétés.

1.a Soit $\rho = u \to \ldots \to v$ un chemin de u à v dans G (et dans G_f). Exprimer le poids[1] $w_f(\rho)$ du chemin ρ dans G_f en fonction de $w(\rho)$ (c'est-à-dire le poids du même chemin

1. la somme du poids de chaque arc du chemin...

dans G). En déduire que ρ est un plus court chemin entre u et v dans G **si et seulement si** ρ est un plus court chemin de u à v dans G_f.

1.b Montrer que G contient un circuit strictement négatif si et seulement si G_f contient un circuit strictement négatif.

1.c Montrer que G contient un circuit strictement négatif si et seulement si G^+ contient un circuit strictement négatif.

2. Pour le graphe M de la figure 2 :

2.a Donner les longueurs $\delta_{M^+}(x_0, q_1)$, $\delta_{M^+}(x_0, q_2)$, $\delta_{M^+}(x_0, q_3)$, et $\delta_{M^+}(x_0, q_4)$).

2.b Construire le graphe M_h où h est définie par : $h(q_i) \stackrel{\text{def}}{=} \delta_{M^+}(x_0, q_i)$ pour $i = 1, \ldots, 4$.

Pour les 3 questions suivantes (3.a, 3.b, et 3.c), on suppose donné un graphe G sans circuit strictement négatif.

On note $h : S \to \mathbb{R}$ la fonction donnant la longueur des PCC entre x_0 et les sommets de S dans G^+ : c'est-à-dire $h(u) = \delta_{G^+}(x_0, u)$ pour tout $u \in S$. Et on va utiliser cette fonction pour construire le graphe $G_h = (S, A, w_h)$ défini comme précédemment.

3.a Montrer que l'on a toujours $h(v) \leqslant w(u, v) + h(u)$. En déduire que l'on a $w_h(u, v) \geqslant 0$ pour tout $(u, v) \in A$.

3.b Quel algorithme doit-on utiliser pour trouver la longueur d'un PCC entre u et v dans G_h ? Justifier votre réponse.

3.c Utiliser la question 1.a pour exprimer la longueur $\delta_G(u, v)$ d'un PCC entre u et v dans G en fonction de la longueur $\delta_{G_h}(u, v)$ d'un PCC entre u et v dans G_h.

Algorithme. L'algorithme de Johnson est décrit par l'algorithme 1. On y utilise une matrice D de taille $|S| \times |S|$ pour représenter les distances des PCC entre chaque couple de sommets : pour les sommets u et v, le coefficient correspondant est noté d_{uv}.

Expliquer ce que fait cet algorithme. Pourquoi utilise-t-il d'abord l'algorithme de Bellman-Ford ? Quelle est sa complexité ?

Procédure `Algo-Johnson`(G)
//$G = (S, A, w)$: un graphe orienté, valué avec $w : A \to \mathbb{R}$.
begin
  $D :=$ `matrice`$(S \times S)$ telle que $d_{uv} = \infty$ pour tout $u, v \in S$
  $G^+ :=$ `Graphe`$(S \cup \{x_0\}, A \cup \{(x_0, u) \mid u \in S\}; \overline{w})$ avec $\overline{w}(u, v) = \begin{cases} w(u, v) & \text{si } u, v \in S \\ 0 & \text{si } u = x_0 \end{cases}$
  $(\text{res}, \text{d}, \Pi) :=$ `PCC-Bellman-Ford`(G^+, x_0)
  si res $== \perp$ **alors**
    Afficher("le graphe G contient un cycle strictement négatif.")
  sinon
    pour chaque $v \in S$ **faire** $h(v) := \text{d}(x_0, v)$
    pour chaque $(u, v) \in A$ **faire** $w_h(u, v) := w(u, v) + h(u) - h(v)$
    $G_h :=$ `Graphe`(S, A, w_h)
    pour chaque $u \in S$ **faire**
      $(\text{d}', \Pi) =$ `PCC-Dijkstra`(G_h, u)
      pour chaque $v \in S$ **faire** $d_{uv} := \text{d}'[v] + h(v) - h(u)$
  return D
end

Algorithme 1 : algorithme de Johnson

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 $(s, u) \in A$ **faire**
    si *Couleur*$[u]$ = *blanc* **alors**
      $\Pi[u] := s$;
      PP(G, u) ;
  Couleur$[s]$:= noir ;
  temps + + ;
  f$[s]$:= *temps* ;
end

Procédure PPC(G)
//$G = (S, A)$
begin
  pour chaque $x \in S$ **faire**
    Couleur$[x]$:= blanc ;
    $\Pi[x]$:= nil ;
  temps := 0 ;
  pour chaque $x \in S$ **faire**
    si *Couleur*$[x]$ = *blanc* **alors**
      PP(G, x) ;
end

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

Procédure PCC-Dijkstra(G, s)
//$G = (S, A, w)$: un graphe orienté valué
//$s \in S$: un sommet origine.
begin
  pour chaque $u \in S$ **faire**
    $\Pi[u]$:= nil
    $\mathsf{d}[u] := \begin{cases} 0 & \text{si } u = s \\ \infty & \text{sinon} \end{cases}$
  F := File(S, d)
  tant que $F \neq \emptyset$ **faire**
    u := Extraire-Min(F)
    pour chaque $(u, v) \in A$ **faire**
      si d$[v]$ > d$[u] + w(u, v)$ **alors**
        d$[v]$:= d$[u] + w(u, v)$
        $\Pi[v] := u$
        MaJ-F-Dijkstra(F, d, v)
  return (d, Π)
end

Procédure PCC-Bellman-Ford(G, s)
//$G = (S, A, w)$: un graphe orienté valué
//$s \in S$: un sommet origine
begin
  pour chaque $u \in S$ **faire**
    $\Pi[u]$:= nil
    $\mathsf{d}[u] := \begin{cases} 0 & \text{si } u = s \\ \infty & \text{sinon} \end{cases}$
  pour $i = 1$ *à* $|S| - 1$ **faire**
    pour chaque $(u, v) \in A$ **faire**
      si $d[v] > d[u] + w(u, v)$ **alors**
        d$[v]$:= d$[u] + w(u, v)$
        $\Pi[v] := u$
  pour chaque $(u, v) \in A$ **faire**
    si $d[v] > d[u] + w(u, v)$ **alors**
      return $(\perp, -, -)$
  return $(\top, d, \Pi)$
end