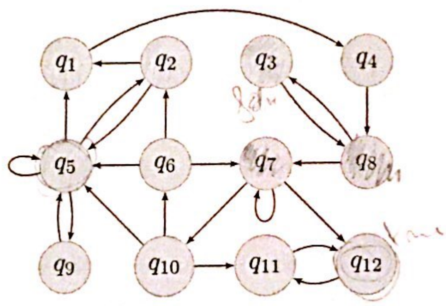
Examen d'algorithmique

Mardi 19 décembre 2017 8h30 – 11h30 / Aucun document autorisé

Mode d'emploi : Le barème est donné à titre indicatif. **La qualité de la rédaction sera très fortement prise en compte pour la note.** On peut toujours supposer une question résolue et passer à la suite. L'annexe contient plusieurs algorithmes vus en cours.

Exercice 1 : Autour des parcours – (3 points)

On considère un graphe orienté $G = (S, A)$ et deux sommets x et y. Écrire un algorithme qui affiche un chemin pour aller de x à y si il en existe un, et $\perp$ sinon. Donner sa complexité. Appliquer votre algorithme sur le graphe de la figure ci-contre avec $x = q_5$ et $y = q_{12}$.

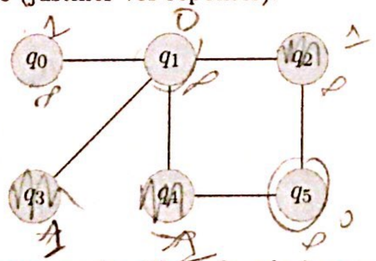


Exercice 2 : Autour des parcours – (3 points)

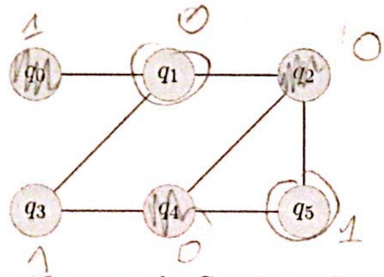
On considère un graphe non-orienté $G = (S, A)$. On cherche à savoir si il est possible de colorier les sommets de G en deux couleurs $\{0, 1\}$. On rappelle que dans ce cas, un coloriage est une fonction $\mathcal{C} : S \to \{0, 1\}$ telle que pour tous sommets u et v, si (u, v) est une arête de A on a $\mathcal{C}(u) \neq \mathcal{C}(v)$. Pour un graphe, être 2-coloriable correspond donc à être biparti.

1. Proposer un coloriage en deux couleurs pour les graphes de la figure ci-dessous si il en existe (justifier vos réponses).

G_1 :



G_2 :



2. Proposer un algorithme de coloriage en deux couleurs pour un graphe G, qui renvoie vrai (et un coloriage) lorsque c'est possible, et faux sinon. On pourra se baser sur le parcours en profondeur.

Exercice 3 : Flots (4 points)

On considère le réseau de transport R de la figure 1 :

1. Appliquer l'algorithme de Ford-Fulkerson pour trouver un flot maximal. On donnera les flots intermédiaires et les graphes des augmentations obtenus par l'algorithme (une feuille est préremplie, voir pages 5 et 6 du sujet). Attention : **on demande à ce que votre application de l'algorithme utilise au moins un arc arrière.**
2. En déduire une coupe de capacité minimale du réseau R.

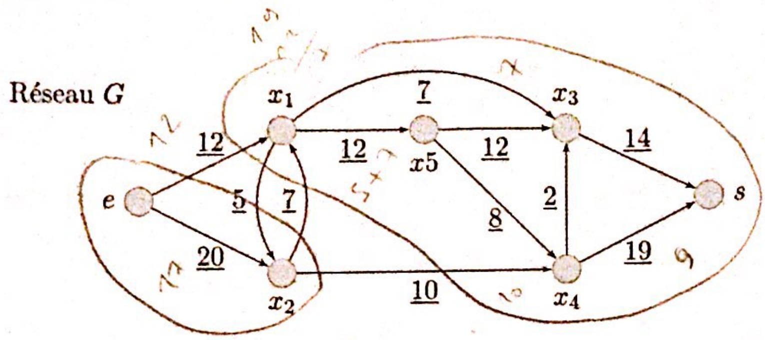


FIGURE 1 – Le réseau de transport R pour l'exercice 3.

Problème : Contraintes et plus courts chemins – (10 points)

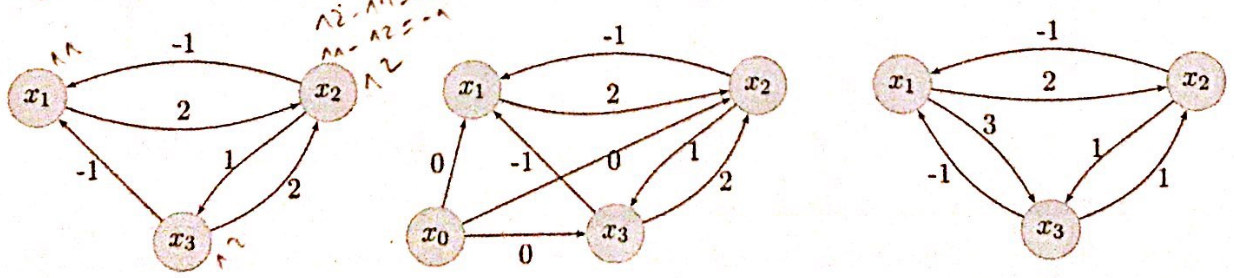
Soit $X = \{x_1, \ldots, x_n\}$ un ensemble de n variables réelles. On appelle <u>contrainte atomique</u> une inégalité sur une différence de variables, c'est à dire de la forme $x_j - x_i \leqslant c$ avec $i, j \in \{1, \ldots, n\}, i \neq j$ et $c \in \mathbf{Z}$. Une <u>contrainte convexe</u> C est un ensemble de contraintes atomiques, interprété comme une conjonction (toutes les contraintes doivent être vérifiées). On supposera que pour toute paire de variables x_i, x_j, il y a au plus une contrainte de la forme $x_j - x_i \leqslant c$. Étant donnée une contrainte convexe E, on note S_E l'ensemble des valuations pour X qui sont des solutions de E (i.e. qui vérifient toutes les contraintes atomiques de E).

Dans la suite, on utilise la contrainte E_0 définie par l'ensemble $\{x_2 - x_1 \leqslant 2, x_1 - x_2 \leqslant -1, x_1 - x_3 \leqslant -1, x_3 - x_2 \leqslant 1, x_2 - x_3 \leqslant 2\}$ pour $X_0 = \{x_1, x_2, x_3\}$ comme un exemple récurrent pour toutes les questions. Les valuations $v = (11, 12, 12)$ et $v' = (5, 6, 7)$ sont des solutions possibles pour E_0 (et donc $v, v' \in S_{E_0}$).

À toute contrainte convexe E, on associe un graphe orienté valué $G_E = (S, A, w)$ avec :

— $S = \{x_1, \ldots, x_n\}$,
— $A = \{(x_i, x_j) \mid \exists (x_j - x_i \leqslant c) \in E\}$,
— $w(x_i, x_j) = c$ ssi $\exists (x_j - x_i \leqslant c) \in E$.

Par exemple, la contrainte convexe E_0 pour X_0 est représentée par le graphe G_{E_0} de la Figure 2.



Graphe G_{E_0} Graphe $G^+_{E_0}$ Graphe $G^*_{E_0}$ (question 4)

FIGURE 2 – Graphes pour l'exemple E_0.

1. Pour chaque cas ci-dessous, donner le graphe de contraintes correspondant à la contrainte convexe associée à l'ensemble X :
   (a) $E_1 = \{x_1 - x_3 \leqslant 2; x_2 - x_3 \leqslant -1; x_3 - x_1 \leqslant 1; x_1 - x_2 \leqslant 1\}$ et $X_1 = \{x_1, x_2, x_3\}$.
   (b) $E_2 = \{x_1 - x_3 \leqslant 2; x_2 - x_3 \leqslant -1; x_3 - x_1 \leqslant -1; x_1 - x_2 \leqslant 1\}$ et $X_2 = \{x_1, x_2, x_3\}$.
   (c) $E_3 = \{x_1 - x_3 \leqslant 2; x_2 - x_3 \leqslant -1; x_3 - x_1 \leqslant 1; x_1 - x_2 \leqslant 1\}$ et $X_3 = \{x_1, x_2, x_3, x_4\}$.

2,1,3

distance
donnée

Est-ce que $v_1 = (12, 11, 12)$ est une solution pour E_1 ? Et $v_1' = (12, 15, 12)$? Pouvez-vous trouver une solution pour E_2 ?

2. On considère à présent le graphe $G_E^+ = (S^+, A^+, w^+)$ qui étend le graphe G_E de la manière suivante : $S^+ = S \cup \{x_0\}$, $A^+ = A \cup \{(x_0, x_i) \mid \text{pour tout } 0 < i \leqslant n\}$, et $w^+(u,v) = w(u,v)$ si $(u,v) \in A$ et $w^+(x_0, x_i) = 0$ pour tout $1 \leqslant i \leqslant n$. A la Figure 2, on donne le graphe $G_{E_0}^+$ pour l'exemple E_0 ci-dessus.

   (a) Construire le graphe $G_{E_1}^+$.

   (b) Calculer les distances d'un plus court chemin entre x_0 et les sommets x_1, x_2, x_3 dans le graphe $G_{E_1}^+$. On notera ces distances $\delta_{G_{E_1}^+}(x_0, x_1)$, $\delta_{G_{E_1}^+}(x_0, x_2)$, et $\delta_{G_{E_1}^+}(x_0, x_3)$.

   (c) Vérifier que la valuation $v = (\delta_{G_{E_1}^+}(x_0, x_1), \delta_{G_{E_1}^+}(x_0, x_2), \delta_{G_{E_1}^+}(x_0, x_3))$ est une solution de E_1.

   (d) Montrer (en utilisant les propriétés des plus courts chemins) que si il n'y a pas de circuit strictement négatif dans le graphe G_E^+ d'une contrainte convexe E quelconque, alors la valuation v qui associe à x_i (pour $i = 1, \ldots, n$) la valeur $\delta_{G_E^+}(x_0, x_i)$ est toujours une solution de E. C'est-à-dire que si $x_j - x_i \leqslant c$ est une contrainte atomique de E, alors on a $\delta_{G_E^+}(x_0, x_j) - \delta_{G_E^+}(x_0, x_i) \leqslant c$.

   (e) Montrer que si il y a un circuit strictement négatif dans G_E^+, alors il n'y a pas de solution pour E (et donc S_E est vide).

   (f) En déduire un algorithme qui étant donnée une contrainte convexe E pour un ensemble de variables X, renvoie une valuation solution pour E si il en existe une, ou la valuation vide $v = \emptyset$ sinon. Donner sa complexité.

3. Donner un algorithme qui teste, étant données deux contraintes convexes E et E', si il existe des solutions appartenant à l'intersection $S_E \cap S_{E'}$. Même question pour $S_E \cup S_{E'}$.

4. On souhaite normaliser les contraintes de manière à ce que deux contraintes ayant les mêmes ensembles solution aient la même représentation.

   Pour cela, on représente le graphe G_E sous la forme d'une matrice d'adjacence[1]. On note $G_E^* = (S, A^*, w^*)$ le graphe orienté pondéré obtenu à partir de G_E par application de l'algorithme de Floyd-Warshall (la fonction w^* correspond donc aux distances des plus courts chemins dans G_E). Par exemple, le graphe $G_{E_0}^*$ de l'exemple précédent est décrit Figure 2 et on donne ci-dessous la matrice de G_{E_0} et celle pour $G_{E_0}^*$:

$$M_{G_{E_0}} = \begin{pmatrix} 0 & 2 & \infty \\ -1 & 0 & 1 \\ -1 & 2 & 0 \end{pmatrix} \qquad M_{G_{E_0}^*} = \begin{pmatrix} 0 & 2 & 3 \\ -1 & 0 & 1 \\ -1 & 1 & 0 \end{pmatrix}$$

   — Expliquer (et justifier) à quoi correspond $w^*(x_i, x_j)$ dans le graphe G_E^*. On pourra prendre pour exemple le coefficient $w^*(x_3, x_2)$ dans $G_{E_0}^*$.

   — $\boxed{\text{Bonus :}}$ En déduire un algorithme pour tester si deux contraintes E et E' représentent le même ensemble de valuations. Donner sa complexité.

   — $\boxed{\text{Bonus :}}$ En déduire un algorithme pour tester si l'ensemble de valuations vérifiant une contrainte E est inclus dans celui représenté par une contrainte E'. Donner sa complexité.

1. avec des coefficients $\alpha_{i,j}$ définis par : $\alpha_{i,i} = 0$ pour tout i, et $\alpha_{i,j} = c$ si $w(x_i, x_j) = c$ et $\alpha_{i,j} = \infty$ si $(x_i, x_j) \notin A$.

Annexe

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

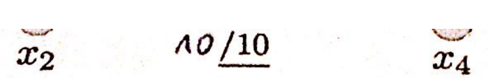
```
Procédure PP(G, s)
//G = (S, A)
begin
  Couleur[s] := gris;
  temps++;
  d[s] := temps;
  pour chaque (s, u) ∈ A faire
    si Couleur[u] = blanc alors
      Π[u] := s;
      PP(G, u);
  Couleur[s] := noir;
  temps++;
  f[s] := temps;
end
```

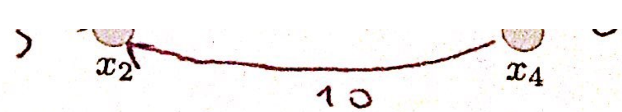
```
Procédure PPC(G)
//G = (S, A)
begin
  pour chaque x ∈ S faire
    Couleur[x] := blanc;
    Π[x] := nil;
  temps := 0;
  pour chaque x ∈ S faire
    si Couleur[x] = blanc alors
      PP(G, x);
end
```

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

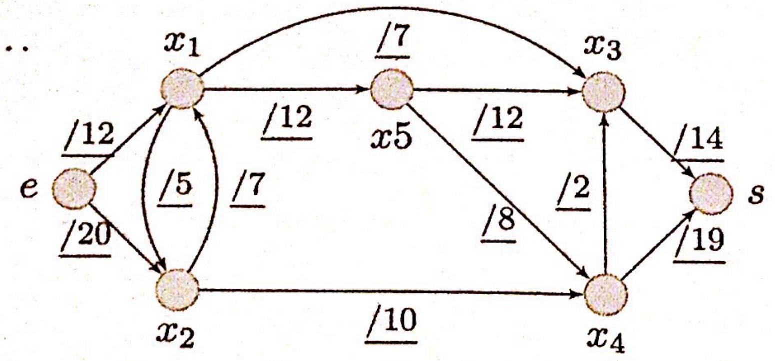
```
Procédure PCC-Floyd(G)
//G = (S, A, w) : orienté valué
//avec S = {x_1, ..., x_n}
//avec M = (α_ij)_{1≤i,j≤n} la
//matrice correspendant à A
begin
  //On initialise D avec M:
  pour i = 1...n faire
    pour j = 1...n faire
      d_ij := α_ij
      si α_ij ≠ ∞ alors π_ij := i
  pour k = 1...n faire
    pour i = 1...n faire
      pour j = 1...n faire
        si d_ij > d_ik + d_kj alors
          d_ij := d_ik + d_kj
          π_ij := π_kj
  return D, Π
end
```

```
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
```

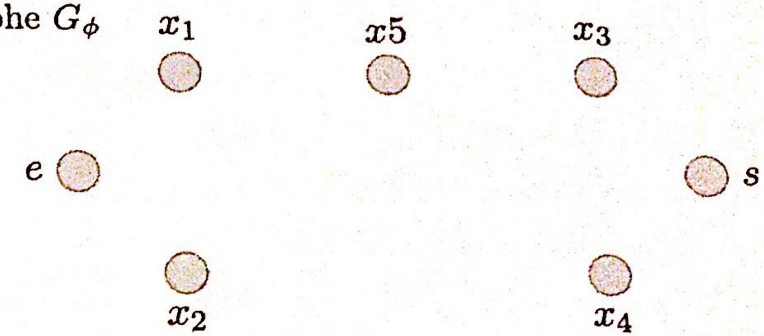




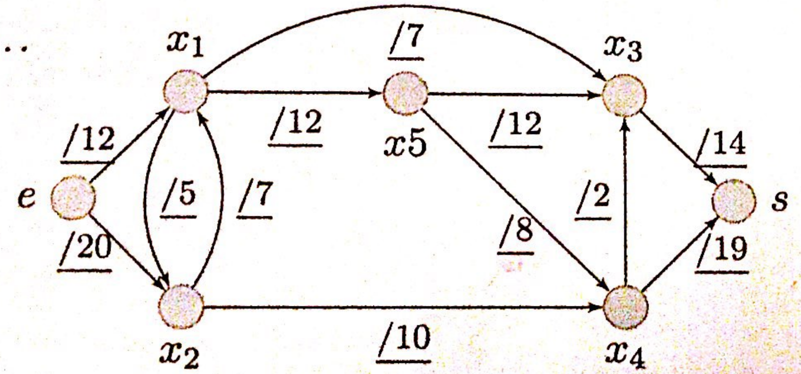
Flot ...



Graphe G_ϕ



Flot ...



Graphe G_ϕ