Examen d'algorithmique

Lundi 7 janvier 2019 8h30 – 11h30 / Aucun document autorisé

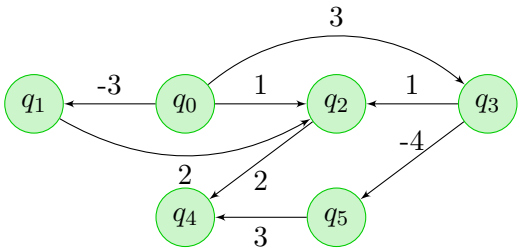
Mode d'emploi : Le barème est donné à titre indicatif. **La qualité de la rédaction sera très fortement prise en compte pour la note.** On peut toujours supposer une question résolue et passer à la suite. On peut utiliser sans les décrire les algorithmes du poly (l'annexe contient le parcours en profondeur).

Exercice 1 : PCC et graphes acyclique – (3 points)

On considère un graphe orienté valué $G = (S, A, w)$ avec $w : A \to \mathbf{R}$ **sans circuit**.

Proposer un algorithme efficace pour trouver les distances des PCC depuis un sommet q_0 donné (on supposera qu'aucun arc arrive en q_0, c'est-à-dire $\deg^+(q_0) = 0$). Justifier votre algorithme.

Appliquer votre algorithme sur le graphe ci-contre.



Indices : on vise un algorithme de complexité en $O(|S| + |A|)$. On pourra utiliser un tri topologique pour énumérer les sommets accessibles depuis s de manière à ce que l'énumération $q_0\, q_1\, q_2 \ldots q_n$ vérifie : $i < j \;\Rightarrow\; (q_j, q_i) \notin A$.

Exercice 2 : Graphe et représentation matricielle – (3 points)

On considère un graphe orienté $G = (S, A)$ **représenté sous la forme d'une matrice booléenne d'adjacence** (*i.e.* le coefficient $\alpha_{i,j}$ vaut `Vrai` ssi $(q_i, q_j) \in A$).
Un sommet p est un *puits* de G si (1) pour chaque $q \in S$ différent de p, on a : $(q, p) \in A$ et $(p, q) \notin A$; et (2) $(p, p) \notin A$. Donc p n'a aucun arc sortant et tous les autres sommets ont un arc qui mène à p.

1. Peut-il y avoir plusieurs plusieurs puits dans un graphe ?
2. Ecrire une fonction `Test-puits`(p) qui teste si p est un puits.
3. En déduire un algorithme qui détecte la présence d'un puits dans G. Evaluer sa complexité.
4. Proposer un autre algorithme plus efficace (en $O(|S|)$).

Exercice 3 : PCC et routage – (4 points)

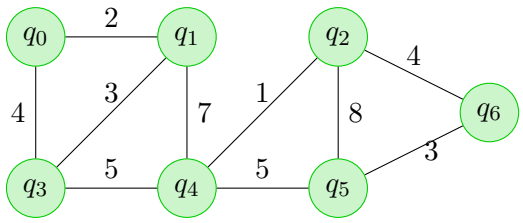
On considère un problème de routage dans un réseau fixe décrit par un graphe non-orienté valué $G = (S, A, w)$ avec $w : A \to \mathbb{N}$ où $w(u, v)$ représente la longueur de l'arc (u, v). Chaque noeud représente un routeur qui doit dispatcher les messages qu'il reçoit vers le bon destinataire : lorsque le noeud x reçoit un message pour le noeud y, alors si $x = y$ c'est fini, et sinon, il doit l'envoyer à l'un de ses voisins z afin que z transmette à son tour le message à un voisin, *etc.* Bien sûr on souhaite utiliser des PCC : si x envoie le message pour y à son voisin z, c'est qu'il existe un PCC reliant x à y dont la première arête est (x, z).

Proposer un algorithme qui construit pour chaque noeud x de G, une table des distances $\mathsf{d}_x[-]$ donnant les distances des PCC de x aux autres noeuds (ainsi $\mathsf{d}_x[y]$ est égal à la distance d'un PCC de x à y) et une table de routage $\mathsf{r}_x[-]$ indiquant à quel voisin de x il faut envoyer les messages selon leur destinataire ($\mathsf{r}_x[y] = z$ dans notre exemple). Donner la complexité de votre algorithme.

Exercice 4 : Arbres couvrants minimaux – (7 points)

Dans cet exercice, on considère des graphes **non orientés** et **valués** $G = (S, A, w)$ et on s'intéresse aux arbres couvrants minimaux (ACM).

Partie 1 : recherche d'un ACM Dans cette partie, on veut calculer un ACM de G **en partant de l'ensemble d'arêtes initial A et en enlevant à chaque étape une arête jusqu'à obtenir un ACM.**



1. Donner un ACM du graphe G décrit à la figure ci-contre.
2. Ecrire une fonction `Test-Chemin`(G, x, y) qui étant donnés un graphe G et deux sommets x et y retourne *Vrai* si et seulement si il existe un chemin de x à y.
3. Etant donnés un graphe $G = (S, A, w)$ et une arête (x, y) de A, on note $G\backslash(x, y)$ le graphe G privé de (x, y), c'est-à-dire $(S, A\backslash\{(x, y)\}, w)$.
   Montrer que si G est connexe, alors on a : $G\backslash(x, y)$ est connexe **si et seulement si** il existe un chemin de x à y dans $G\backslash(x, y)$.
4. Déduire des deux questions précédentes un algorithme qui étant donné un graphe $G = (S, A, w)$ **connexe**, calcule un ACM en partant de A et en enlevant des arêtes.
   Expliquer votre algorithme et justifier sa correction. Quelle est sa complexité ? Appliquer le sur le graphe de la question 1.
5. Comparer la complexité d'un d'algorithme qui enlève des arêtes de A (pour obtenir un ACM) et un algorithme qui en ajoute depuis l'ensemble vide (comme Prim ou Kruskal) ?

Partie 2 : unicité d'un ACM On étudie ici certaines propriétés des ACM.

1. Donner un exemple de graphe admettant plusieurs *arbres couvrants minimaux différents.*
2. Soit $T \subseteq A$ un arbre couvrant minimal de G tel que l'arête reliant deux sommets x et y de S appartienne à T. On définit le graphe G' comme le graphe G dans lequel les sommets x et y ont été fusionnés en un nouveau sommet z.
   Formellement, G' est le graphe (S', A', w') avec $S' = (S\backslash\{x, y\}) \cup \{z\}$ et les arêtes A' sont celles de $A\backslash\{(x, y)\}$ où les sommets x et y sont remplacés par z. La fonction w' renvoie le même poids que w. Et si il existe $(x, u) \in A$ et $(y, u) \in A$, on ne garde dans G' qu'une seule arête (z, u) et son poids est $\min(w(x, u), w(y, u))$.
   (a) Choisir deux sommets du graphe exemple de la question précédente et dessiner le graphe G' correspondant.
   (b) Montrer que l'ensemble d'arêtes T' contenant les arêtes de $T\backslash\{(x, y)\}$ où les sommets x et y sont remplacés par z, est un arbre couvrant minimal du graphe G'.
3. Montrer que si la fonction poids w de $G = (S, A, w)$ est telle que le poids de chaque arête de A est unique, alors il existe un unique arbre couvrant minimal pour G.
   Indices : on pourra utiliser le résultat précédent, faire une récurrence sur $|S|$ et montrer que l'arête de poids minimale est toujours présente dans un ACM de G...

Exercice 5 : Parcours – (3 points)

On considère un graphe orienté $G = (S, A)$. Ecrire un algorithme $\mathsf{CheminPair}(G, x, y)$ qui renvoie vrai si et seulement si il existe un chemin de longueur paire de x à y. (ici la longueur désigne le nombre d'arcs du chemin). NB : On considère des chemins quelconques.

Donner la complexité de votre algorithme.

Appliquer votre algorithme sur les trois graphes de la figure 1 pour les sommets x et y.

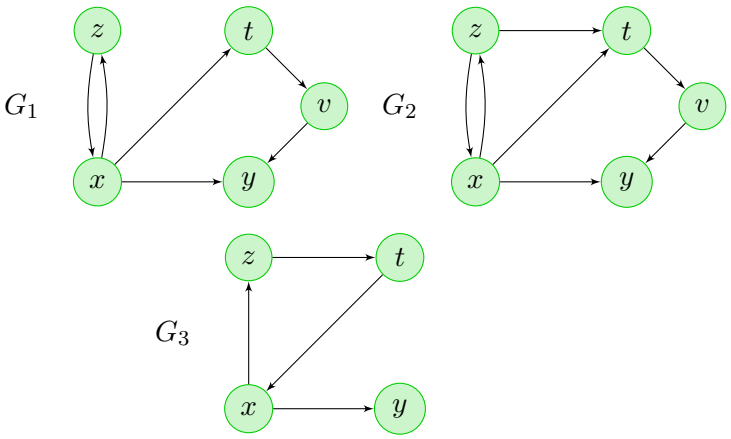


FIGURE 1 – Graphe pour l'exercice 5.

Exercice 6 : Flots (4 points)

On considère le réseau de transport R de la figure 2 :

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.

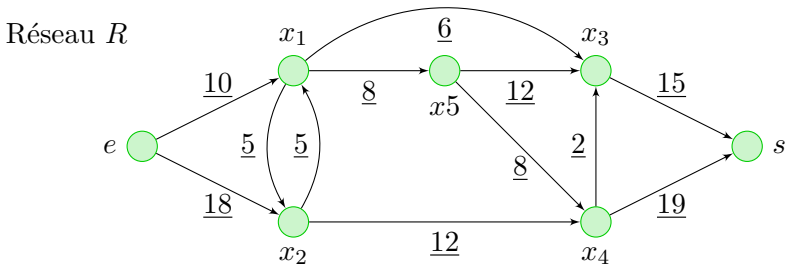


FIGURE 2 – Le réseau de transport R pour l'exercice 6.

Annexe

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

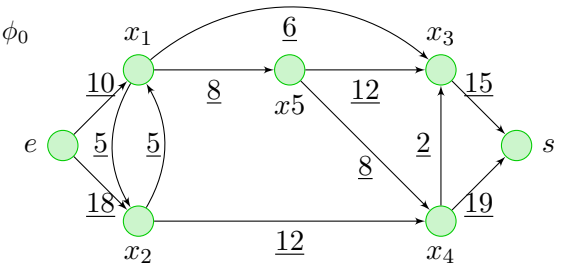
```
Procédure PP(G, s)
//G = (S, A)
begin
    Couleur[s] := gris;
    temps + +;
    d[s] := temps;
    pour chaque (s, u) ∈ A faire
        si Couleur[u] = blanc alors
            Π[u] := s ;
            PP(G, u) ;
    Couleur[s] := noir ;
    temps + + ;
    f[s] := temps ;
end
```

```
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
```

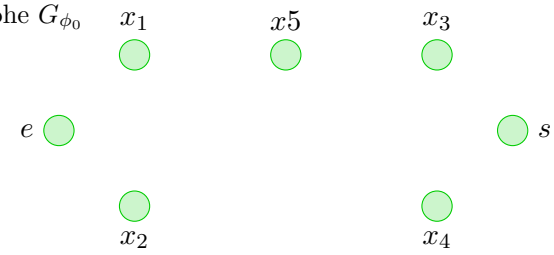
A détacher et à remettre avec la copie.

Numéro d'étudiant :

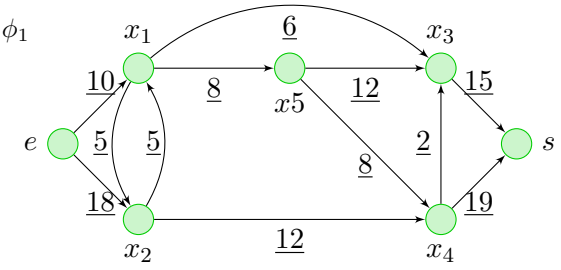
Flot ϕ_0



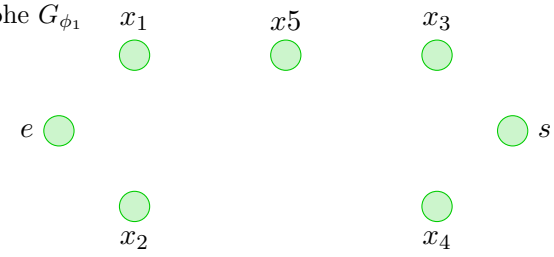
Graphe G_{ϕ_0}



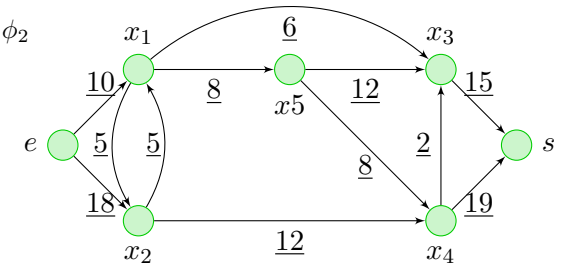
Flot ϕ_1



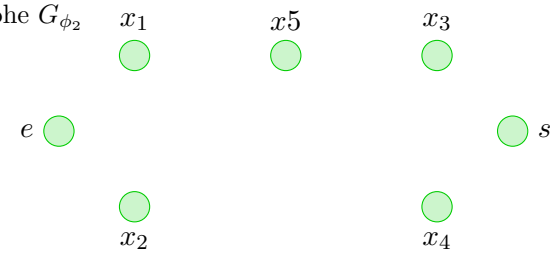
Graphe G_{ϕ_1}



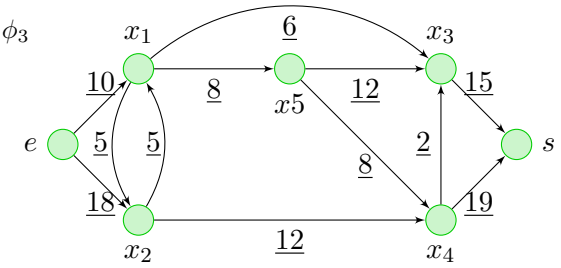
Flot ϕ_2



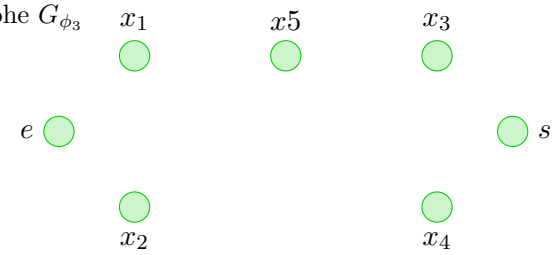
Graphe G_{ϕ_2}



Flot ϕ_3

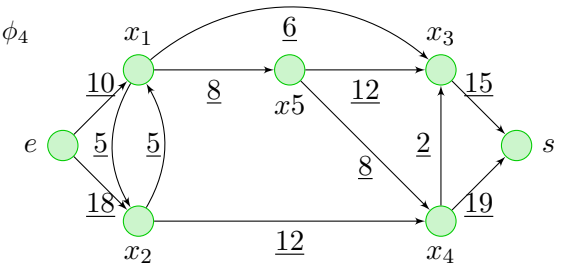


Graphe G_{ϕ_3}

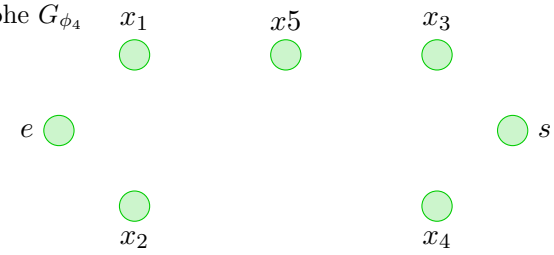


A détacher et à remettre avec la copie.

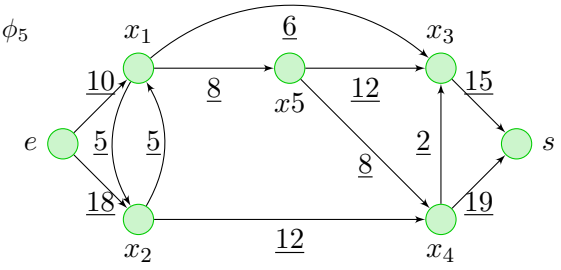
Flot ϕ_4



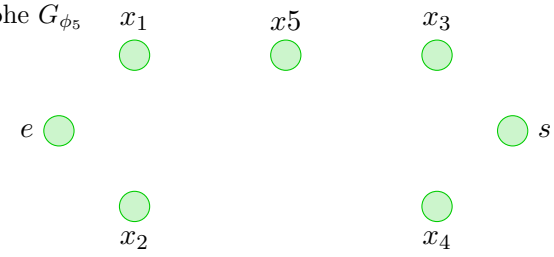
Graphe G_{ϕ_4}



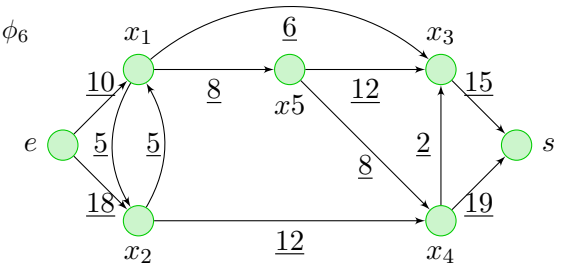
Flot ϕ_5



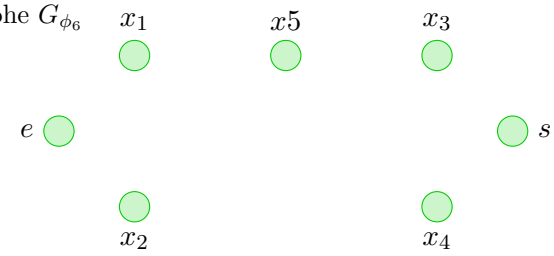
Graphe G_{ϕ_5}



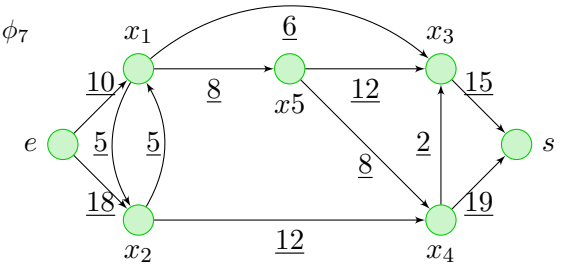
Flot ϕ_6



Graphe G_{ϕ_6}



Flot ϕ_7



Graphe G_{ϕ_7}