L3/Informatique-MIAGE **2016-2017**

Algorithmique des graphes

6 janvier 2017

Notes du cours et TD autorisées Aucun document extérieur n'est autorisé Calculatrices autorisées

Certaines de vos réponses doivent être données sur les feuilles du sujet que vous allez rendre à l'enseignant surveillant. Veillez à ne pas détacher les feuilles.

Indiquez votre numéro d'anonymat en haut à gauche.

Concepteurs : R. Andonov, S. François, Y. Mocquard

Ces exercices sont consacrés au thème de la *fermeture transitive* des graphes orientés. Il s'agit de proposer et d'analyser diverses algorithmes qui mènent à la résolution de ce problème, avec comme objectif d'améliorer la complexité dans certains cas particuliers.

Il sera aussi question de composantes *connexes* ou *fortement connexes* toujours dans le cadre de graphes orientés.

1 Rappel

Étant donné un graphe orienté $G = (V, E)$ avec $n = |V|$ sommets et $m = |E|$ arcs, les définitions et les propositions suivantes sont en vigueur :

1. La *fermeture transitive* de G est le graphe $G^+ = (V, E^+)$ tel que $(x, y) \in E^+$ si, et seulement si, il existe un chemin de G allant de x à y.
2. Un graphe est *transitif* si la relation binaire qui lui est associée est transitive. En d'autres termes $\forall x, y, z \in V : (x, y) \in E$ et $(y, z) \in E$ alors $(x, z) \in E$.
   (a) Un graphe est transitif si, et seulement si, tout chemin de longueur deux est sous-tendu par un arc.
   (b) Un graphe est transitif si, et seulement si, tout chemin est sous-tendu par un arc.
3. Soit la relation binaire définie sur l'ensemble de sommets V ayant la propriété "il existe au moins un chemin de x vers y et au moins un chemin de y vers x". Elle est réflexive, symétrique et transitive : donc c'est une équivalence. Les classes obtenues par cette relation d'équivalence se nomment les *composantes fortement connexes* du graphe.

2 Recherche de composantes fortement connexes

Il s'agit de trouver les composantes fortement connexes du graphe de la FIGURE 1.

1. Utiliser les FIGURES 1 et 2 pour trouver les composantes fortement connexes. Les valeurs *pre* et *post* (ou *in* et *out*) devront être clairement indiquées.
2. Dessiner le graphe réduit qui en découle dans l'espace de la FIGURE 3 au bas de cette feuille.

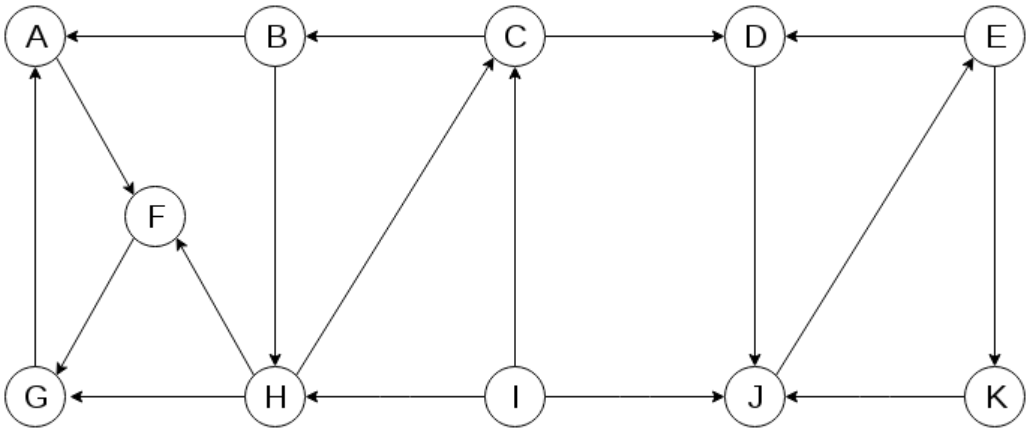


Figure 1: Graphe d'origine

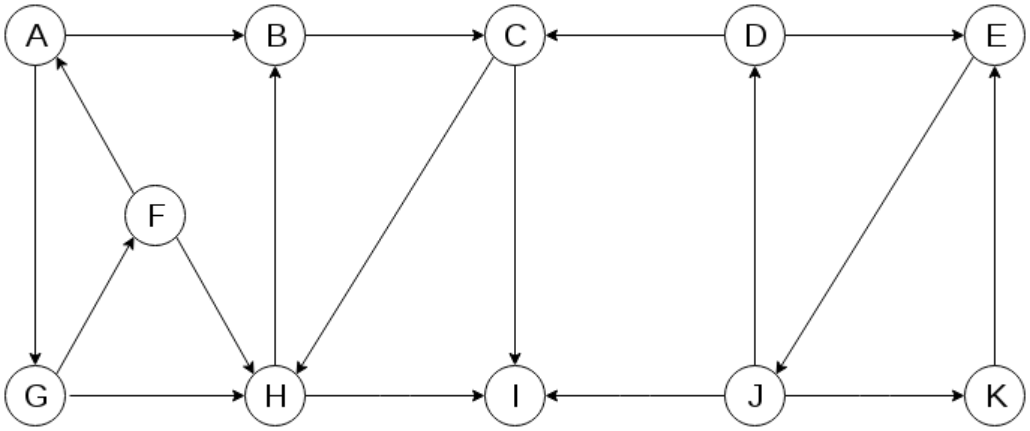


Figure 2: Graphe inverse

3. Combien d'arcs au minimum doit-on ajouter pour que le graphe d'origine soit fortement connexe ? Dessinez ces arcs en pointillés ou d'une autre couleur, sur le graphe réduit et sur le graphe d'origine.

Figure 3: Graphe réduit

Sol :

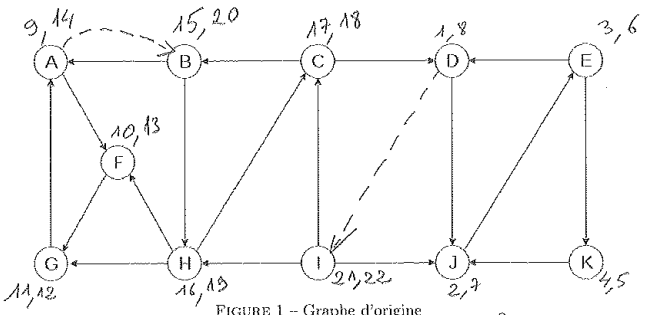


FIGURE 1 – Graphe d'origine

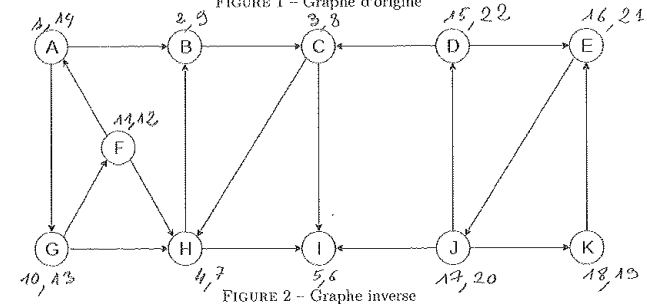


FIGURE 2 – Graphe inverse

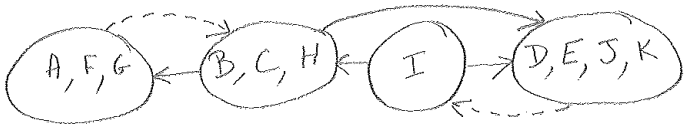


FIGURE 3 – Graphe réduit

2

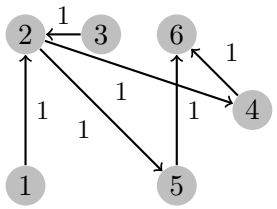
Figure 4: Solution les composantes fortement connexes

–Fin sol

3 Fermeture transitive

On associe la fermeture transitive $G^+ = (V, E^+)$ avec une matrice $T^{n \times n}$ telle que

$$T[i,j] = \begin{cases} 0 & \text{si } i \neq j \text{ et } (i,j) \notin E^+ \\ 1 & \text{si } i = j \text{ ou } (i,j) \in E^+ \end{cases} \quad (1)$$



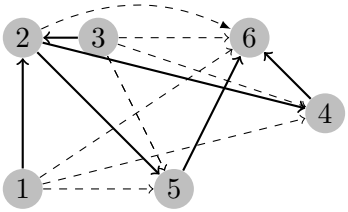
$A =$

	1	2	3	4	5	6
1	0	1	0	0	0	0
2	0	0	0	1	1	0
3	0	1	0	0	0	0
4	0	0	0	0	0	1
5	0	0	0	0	0	1
6	0	0	0	0	0	0

Figure 5: Le graphe G et sa matrice d'adjacence A.

Q 1. Trouver à la main et dessiner la fermeture transitive du graphe de la FIGURE 5 et donner la matrice associée T. Les arcs ajoutés seront illustrés en pointillé ou d'une autre couleur.

Sol :



$T =$

	1	2	3	4	5	6
1	1	1	0	1	1	1
2	0	1	0	1	1	1
3	0	1	1	1	1	1
4	0	0	0	1	0	1
5	0	0	0	0	1	1
6	0	0	0	0	0	1

Figure 6: La fermeture transitive du graphe G et sa matrice T.

–Fin sol

3.1 Cas géneral

On peut calculer la fermeture transitive en affectant à chaque arc de E un poids égal à 1, et en exécutant l'algorithme suivant qui calcule dans le tableau $dist(i, j, n)$ la distance la plus courte entre chaque couple de sommets i et j du graphe G. S'il existe un chemin de i à j, on obtient $dist(i, j, n) \leq n$ et on pose $T[i, j] = 1$. Sinon, on obtient $dist(i, j, n) = \infty$ et on pose $T[i, j] = 0$. Pour s'accorder avec la définition (1) on pose en plus $T[i, i] = 1$.
Les sommets sont numérotés de 1 à n.

Algorithm 1 Function fermetureTransitive(G,l)

Require: $G = (V, E)$ with edge weights $\{l_e = 1 : e \in E\}$
Ensure: $dist(i, j, n)$ = length of of the shortest path from i to j for all $i, j, \in V$

```
1:  for i ∈ V do
2:    for j ∈ V do
3:      if i = j then
4:        dist(i, j, 0) ← 0
5:      else if (i, j) ∈ E then
6:        dist(i, j, 0) ← l(i, j)
7:      else
8:        dist(i, j, 0) ← ∞
9:      end if
10:   end for
11: end for
12: for k = 1 to n do
13:   for i ∈ V do
14:     for j ∈ V do
15:       dist(i, j, k) ← min{dist(i, j, k − 1), dist(i, k, k − 1) + dist(k, j, k − 1)}
16:     end for
17:   end for
18: end for
19: for i ∈ V do
20:   for j ∈ V do
21:     if dist(i, j, n) ≤ n then
22:       T[i, j] ← 1
23:     else
24:       T[i, j] ← 0
25:     end if
26:   end for
27: end for
28: return T
```

Q 2. Donner la complexité de l'algorithme (1). Argumenter la réponse donnée.

Sol : Algorithme (1) est en $O(n^3)$ en raison des trois boucles sur les lignes 12, 13 et 14.
–**Fin sol**

3.2 Cas particulier : DAG (Directed Acyclic Graph)

Nous faisons l'hypothèse que nous ne considérons ici que des graphes connexes. On constate que le graphe de la FIGURE 5 est un graphe orienté sans circuit (ou DAG). L'objectif de cet exercice est de mettre au point un algorithme plus rapide que l'algorithme (1) en vous inspirant du calcul du plus court chemin dans un DAG que vous avez vu en cours. Le squelette de ce nouvel algorithme vous est donné (algorithme (3)) et les trois premières questions ci-dessous vous aideront à le comprendre afin de le completer.

Q 3. Enumérer les algorithmes vus en cours qui permettent de trier les sommets d'un graphe DAG dans l'ordre topologique(linéaire) et donner leur complexité.

Sol : Les algorithmes (6) et (7) du support du cours. L'algorithme (6) est basé sur l'ordre décroissant des valeurs $post(v)$ qui sont calculés avec un parcours en profondeur. L'algorithme (7) consiste à éliminer itérativement les sommets sources. Les deux algorithmes sont de complexité $O(|V| + |E|)$.

–**Fin sol**

Q 4. Appliquer l'algorithme de votre choix pour linéariser le graphe de la FIGURE 5. Détailler les itérations.

Sol :

On applique l'algorithme (6) qui est basé sur l'ordre décroissant des valeurs $post(v)$ qui sont calculés avec un parcours en profondeur.

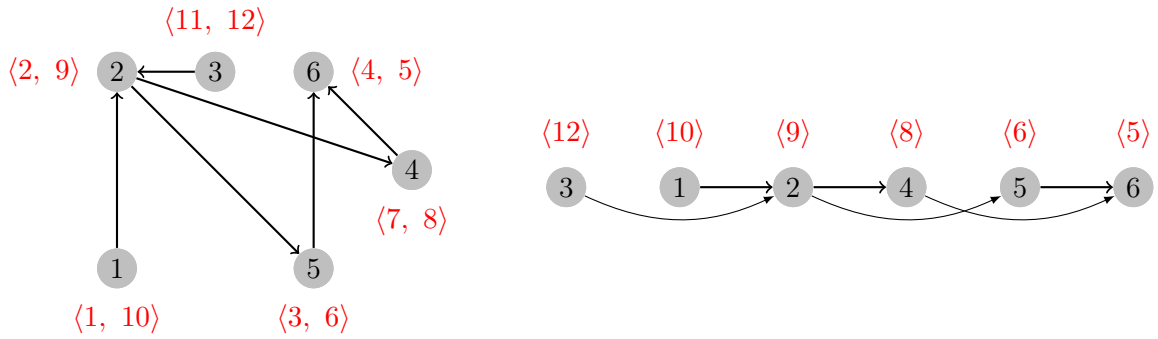


Figure 7: **Gauche:** Les valeurs $\langle pre,\ post\rangle$ obtenues avec un parcours en profondeur du graphe G. **Droite:** Le graphe G est linéarisé suivant l'ordre décroissant des valeurs $\langle post\rangle$.

–**Fin sol**

Algorithm 2 Function fermetureTransitive_DAG(G,l)

Require: DAG G=(V,E) with edge weights $\{l_e : e \in E\}$
Ensure: The matrix T defined as in equation (1).

1: linearize G
2: **for all** [ ] in linearized order **do**
3: $V \leftarrow V \setminus \{s\}$
4: $dist(s) \leftarrow 0$
5: **for all** $j \in V$ **do**
6: $dist(j) \leftarrow \infty$
7: **end for**
8: **for all** $j \in V$, in linearized order **do**
9: **for all** edges $(i, j) \in E$ **do**
10: $dist(j) = \min\{dist(j), dist(i) + l(i, j)\}$
11: **end for**
12: **if** [ ] **then**
13: [ ]
14: **end if**
15: **end for**
16: **end for**
17: **return** T

Q 5. Pour un sommet $s \in V$ fixé, quelles valeurs calcule-t-on dans le tableau *dist*?

Sol :

Pour un $s \in V$ fixé, *dist(j)* est la longueur du plus court chemin de s à j.

–**Fin sol**

Q 6. Compléter le code de l'algorithme (3) afin que la matrice T satisfasse l'égalité suivante quels que soient i et $j \in V$:

$$T[i,j] = \begin{cases} 0 & \text{si } i \neq j \text{ et } (i,j) \notin E^+ \\ 1 & \text{si } i = j \text{ ou } (i,j) \in E^+ \end{cases}$$

Sol :

Algorithm 3 Function fermetureTransitive_DAG(G,l)

Require: DAG G=(V,E) with edge weights $\{l_e : e \in E\}$
Ensure: The matrix T defined as in equation (1).

```
1:  linearize G
2:  for all  s ∈ V in linearized order  do
3:     V ← V \ {s}
4:     dist(s) ← 0
5:     for all  j ∈ V  do
6:        dist(j) ← ∞
7:     end for
8:     for all  j ∈ V, in linearized order  do
9:        for all edges (i, j) ∈ E do
10:          dist(j) = min{dist(j), dist(i) + l(i, j)}
11:       end for
12:       if  dist(j) < ∞ then
13:          T[s, j] ← 1 otherwise T[s, j] ← 0
14:       end if
15:    end for
16: end for
17: return  T
```

–**Fin sol**

Q 7. Evaluer la complexité de l'algorithme (3) en fonction de n et m. Justifier.

Sol : La boucle **for all** $s \in V$ de la ligne 2 s'execute $n = |V|$ fois.

Les deux boucles **for all** des lignes (8) et (9) donnent $\sum_{j \in V} d^-(j) = O(|V| + |E|)$.

En total: $O(|V|(|V| + |E|)) = O(n(n+m))$.

–**Fin sol**

Q 8. Evaluer la complexité de l'algorithme (3) dans le pire et dans le meilleur des cas en fonction de n uniquement. Justifier.

Sol : Le pire des cas correspond au cas d'un graphe dense où $m \approx n^2$. Dans ce cas la complexité de l'algorithme (3) est en $O(n^3)$.

Le meilleur des cas correspond au cas d'un graphe peu dense(creux) où $m \approx n$. Dans ce cas la complexité de l'algorithme (3) est en $O(n^2)$.

–**Fin sol**

Q 9. Conclure en comparant les algorithmes (1) et (3) selon les cas.

Sol : Les deux algorithmes se comportent pareil dans le cas des graphes denses. L'algorithme (3) est plus performant que l'algorithme (1) dans le cas d'un graphe creux.

–**Fin sol**

4 Plus longue sous-séquence décroissante : modélisation, résolution

Il s'agit dans cet exercice de trouver la plus longue sous-séquence **décroissante** d'une séquence de nombres $S = a_1, a_2, \ldots, a_n$.

4.1 Approche DAG:

L'espace des solutions sera présenté par le graphe $G = (V, E)$ qui est un graphe orienté sans circuit (DAG) où à chaque nombre a_i on associera un sommet $i \in V$ et on ajoutera un arc (i, j) si $i < j$ et $a_i > a_j$. On établit ainsi une bijection entre les chemins dans G et les sous-séquences décroissantes dans S. L'objectif est de trouver le plus long chemin (PLC) dans ce graphe.

Q 10. Dessiner le graphe associé à l'instance $S = 5, 11, 9, 12, 7, 10, 3$.
Sol :

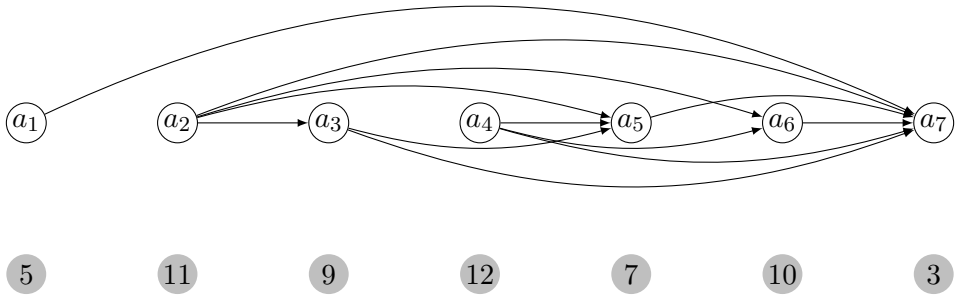


Figure 8: Le graphe G associé à l'instance $S = 5, 11, 9, 12, 7, 10, 3$.

–**Fin sol**

4.2 Approche PL:

Il s'agit de modéliser le même problème dans le contexte de la programmation linéaire.

Q 11. Introduire les variables du problèmes.
Sol : A chaque arc (a_i, a_j) on associe la variable $x_{i,j}$ telle que

$$x_{i,j} = \begin{cases} 1 & \text{si l'arc (i,j) fait partie du plus long chemin} \\ 0 & \text{sinon} \end{cases}$$

–**Fin sol**

Q 12. Donner la matrice d'incidence sommet/arc associée à l'instance ci-dessus.
Sol :

$A =$

	$(1,7)$	$(2,3)$	$(2,5)$	$(2,6)$	$(2,7)$	$(3,5)$	$(3,7)$	$(4,5)$	$(4,6)$	$(4,7)$	$(5,7)$	$(6,7)$
a_1	1	0	0	0	0	0	0	0	0	0	0	0
a_2	0	1	1	1	1	0	0	0	0	0	0	0
a_3	0	−1	0	0	0	1	1	0	0	0	0	0
a_4	0	0	0	0	0	0	0	1	1	1	0	0
a_5	0	0	−1	0	0	−1	0	−1	0	0	1	0
a_6	0	0	0	−1	0	0	0	0	−1	0	0	1
a_7	−1	0	0	0	−1	0	−1	0	0	−1	−1	−1

–Fin sol

Q 13. Décrire les contraintes linéaires et la fonction objectif.

Sol : On ajoute au graphe de la figure (8) deux sommets artificiels: un sommet source s et un sommet puits t. La source s sera connectée aux deux sommets sans prédécesseurs (a_1 et a_2), tandis que le puits t sera connecté à l'unique sommet sans succcésseurs a_t. Le graphe ainsi obtenu est présenté ci-dessous.

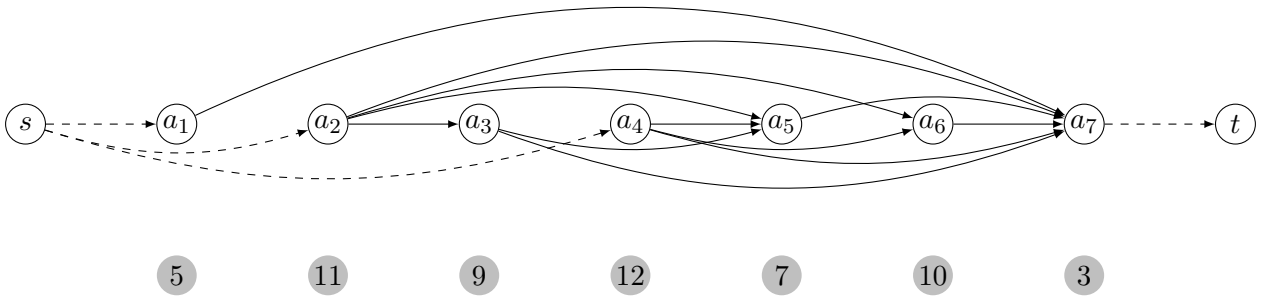


Figure 9: Un sommet source s et un sommet puits t sont ajoutés au graphe de la figure (8) et connectés de manière appropriée. La nouvelle matrice A' est donnée ci-dessous.

	$(1,7)$	$(2,3)$	$(2,5)$	$(2,6)$	$(2,7)$	$(3,5)$	$(3,7)$	$(4,5)$	$(4,6)$	$(4,7)$	$(5,7)$	$(6,7)$	$(s,1)$	$(s,2)$	$(s,4)$	$(7,t)$
s	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0
t	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	−1
a_1	1	0	0	0	0	0	0	0	0	0	0	0	−1	0	0	0
a_2	0	1	1	1	1	0	0	0	0	0	0	0	0	−1	0	0
a_3	0	−1	0	0	0	1	1	0	0	0	0	0	0	0	0	0
a_4	0	0	0	0	0	0	0	1	1	1	0	0	0	0	−1	0
a_5	0	0	−1	0	0	−1	0	−1	0	0	1	0	0	0	0	0
a_6	0	0	0	−1	0	0	0	0	−1	0	0	1	0	0	0	0
a_7	−1	0	0	0	−1	0	−1	0	0	−1	−1	−1	0	0	0	1

Notons b le vecteur $[1, -1, 0, \ldots, 0]^T$. Le programme linéaire est alors:

$$\max\{ \sum_{(i,j)\in E} x_{i,j} \mid A'x = b,\ x_{i,j} \in \{0,1\}\ (i,j) \in E\}$$

où $E = \{(s,1),(s,2),(s,4),(1,7),(2,3),(2,5),(2,6),(2,7),(3,5),(3,7),(4,5),(4,6),(4,7),(5,7),(6,7),(7,t)\}$.

–Fin sol