

Algorithmique des graphes

David Pichardie

28 Mars 2018

Bilan du CM3

- Le parcours en largeur calcule les plus courts chemins d'un graphe non-pondéré
- Le parcours en profondeur permet de classer les arcs selon 4 types
- Le parcours en profondeur permet de détecter les cycles dans un graphe non-orienté, en temps linéaire
- Tri topologique
 - grâce à un parcours en profondeur
 - ou en utilisant l'algorithme de Kahn

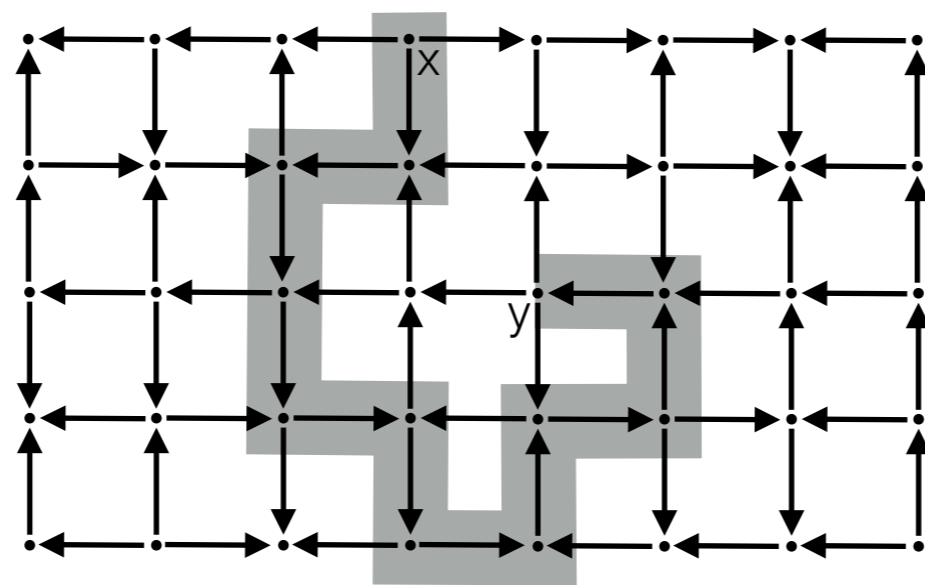
Fermeture transitive

Vocabulaire

Définition

Dans un graphe, orienté ou non, un sommet y est dit *accessible* depuis un sommet x si il existe un chemin de x à y .

Exemple



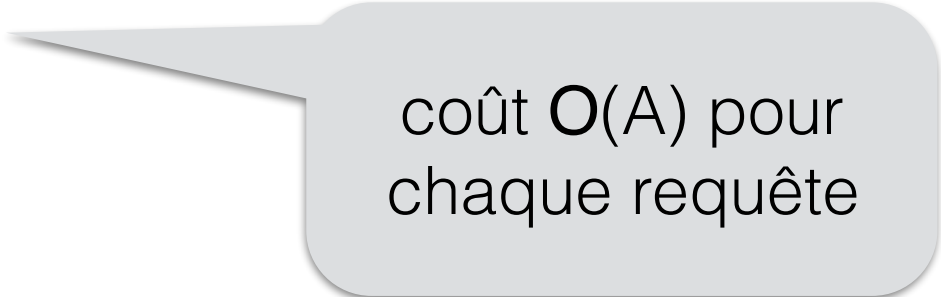
y est accessible depuis x

Accessibilité

On peut décider si y est accessible depuis x grâce à un parcours en profondeur démarrant en x .

```
VISITE(i,y) =  
  si i=y alors renvoie vraie  
  VU[i] <- vraie  
  pour tout j ∈ Adj[i]  
    si non VU[j] et VISITE(j,y) renvoie  
vraie  
  renvoie faux
```

```
ACCESSIBLE?(x,y) =  
  VU <- [faux, ..., faux]  
  renvoie VISITE(x,y)
```



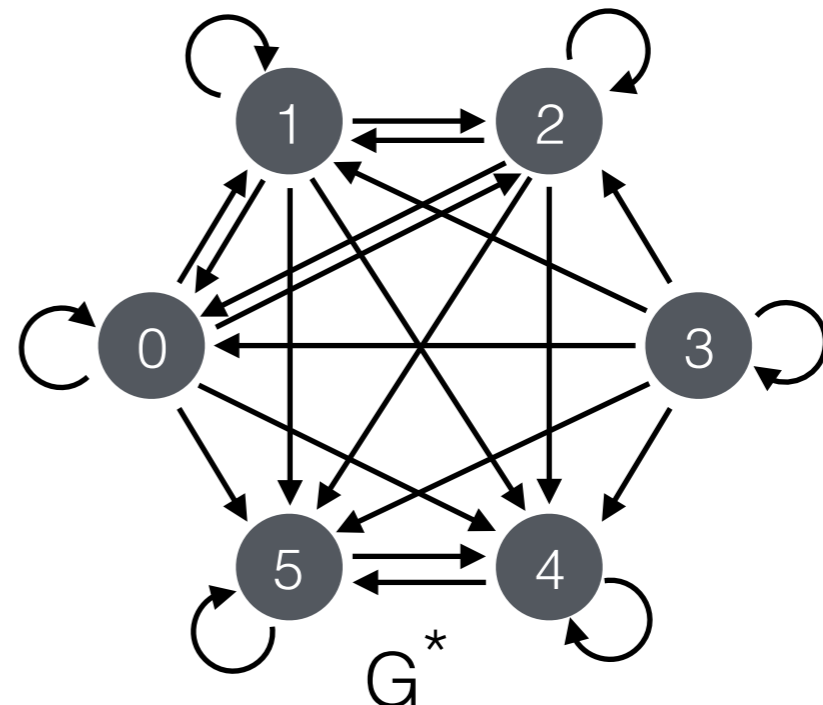
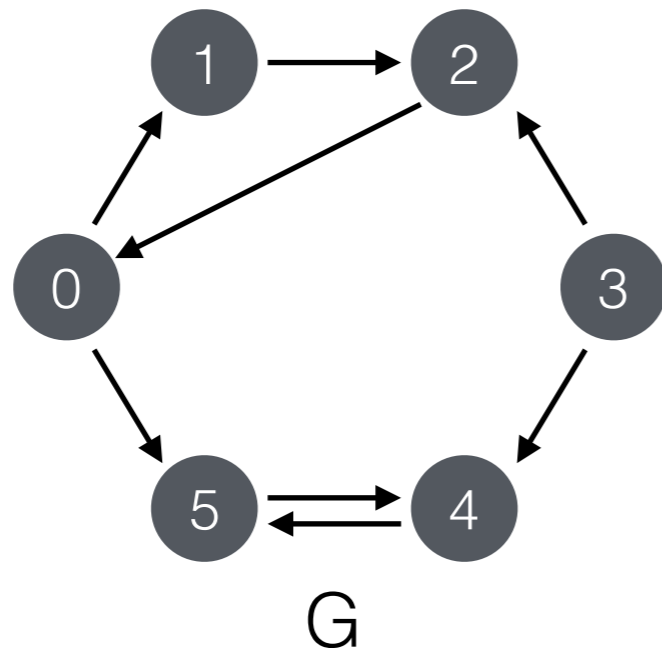
coût $O(A)$ pour
chaque requête

Vocabulaire

Définition

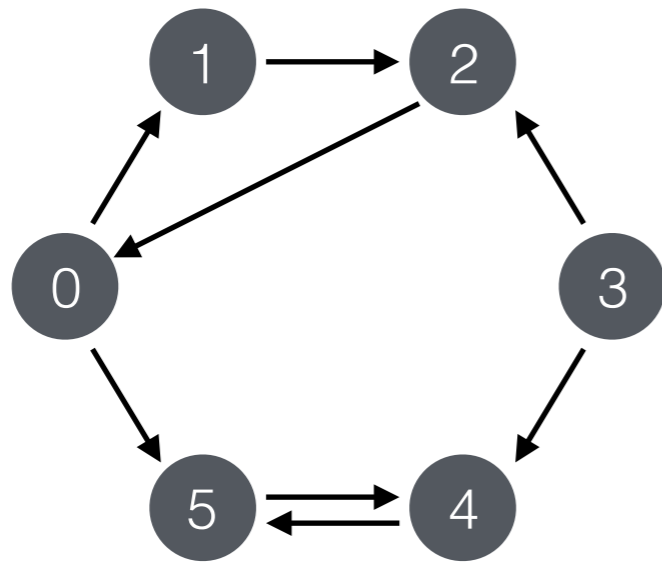
La *fermeture transitive* d'un graphe $G=(S,A)$ est un graphe G^* avec les même sommets S mais dans lequel il existe un arc entre x et y si et seulement si il y a un chemin de x à y dans G .

Exemple

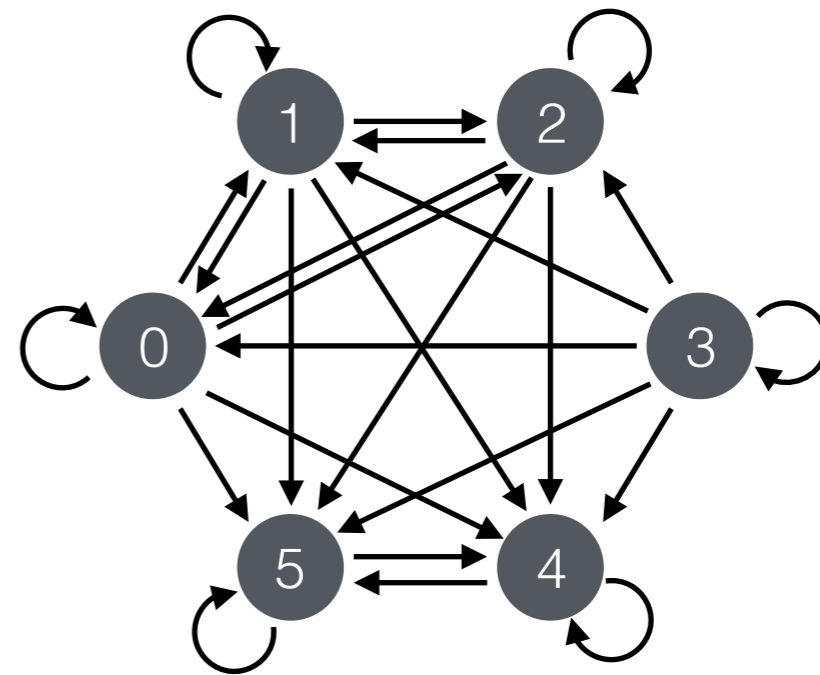


Remarque

La fermeture transitive d'un graphe est souvent dense : on représente donc la fermeture avec une matrice d'adjacence



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



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

Application

Si on connaît G^* , on peut résoudre les requêtes d'accessibilité en temps constant

`ACCESSIBLE?(Astar, x, y) =`
`renvoie (x, y) ∈ Astar`

Calcul de la fermeture transitive sur les graphes *sparses*

Avec $|S|$ parcours en profondeur (un pour chaque ligne)

on parcourt les accessibles depuis l

```
VISITE(l,i) =  
  A*[l,i] <- vraie  
  pour tout j ∈ Adj[i]  
    si non A*[l,j] alors VISITE(l,j)
```

```
FERMETURE_TRANSITIVE() =  
  A* <- [[faux, ..., faux], ..., [faux, ..., faux]]  
  pour tout i ∈ S  
    VISITE(i,i)  
  renvoie A*
```

on lance une visite avec $VU[] = A*[i][]$

coût $O(AS)$

Calcul de la fermeture transitive sur les graphes denses

Avec des calculs matriciels !

Rappel :

PRODUIT(A, B) =

C ← [[0, ..., 0], ..., [0, ..., 0]]

pour tout $i \in S$

pour tout $j \in S$

pour tout $k \in S$

C[i, j] ← C[i, j] + A[i, k] * B[k, j]

on calcule

$$C[i, j] = \sum_k A[i, k] \times B[k, j]$$

Calcul de la fermeture transitive sur les graphes denses

Produits de matrices à valeurs booléennes

```
PRODUIT_BOOL(A,B) =  
  C <- [[faux, ..., faux], ..., [faux, ..., faux]]  
  pour tout i ∈ S  
    pour tout j ∈ S  
      pour tout k ∈ S  
        C[i,j] <- C[i,j] || A[i,k]&&B[k,j]
```

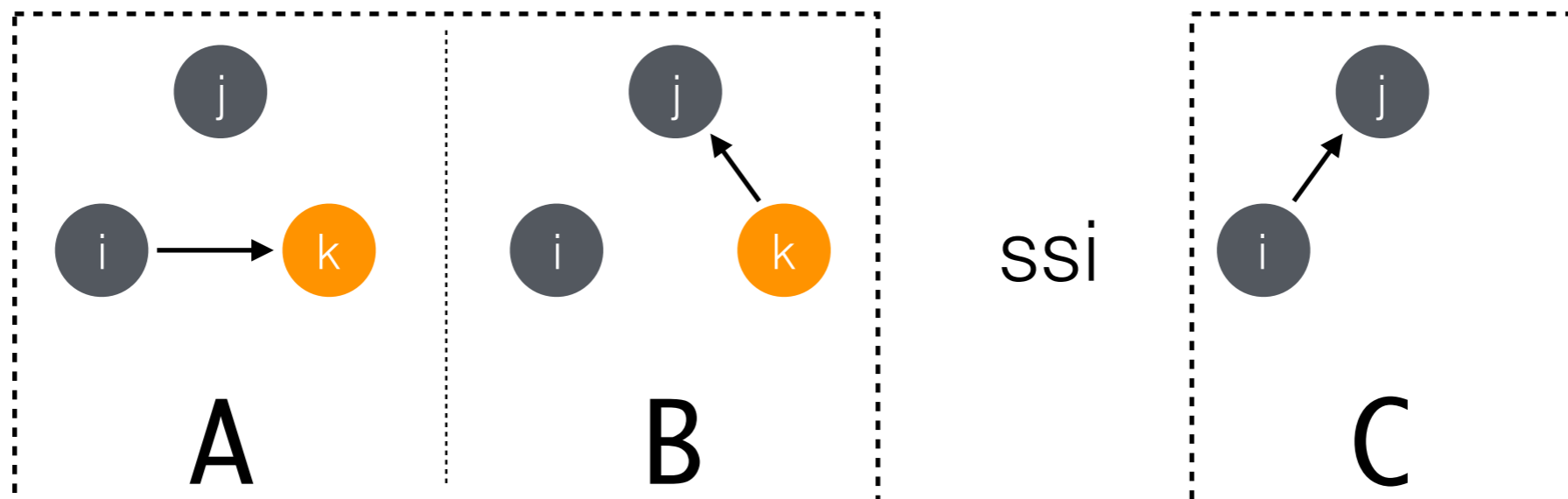
il existe k tel que
A[i,k]=vraie et B[k,j]=vraie

Produit de matrices d'adjacences

Si C est un graphe dont la matrice d'adjacence est égale au produit des matrices d'adjacence des graphes A et B , alors

il existe un arc (i,j) dans C
si et seulement si

il existe un sommet k tel que (i,k) est un arc de A et (k,j) est un arc de B .



Calcul de la fermeture transitive sur les graphes denses

Si A est la matrice d'adjacence d'un graphe G

- si G n'a pas d'auto-boucle,
pour tout sommets i, j ,
 $A^k[i, j]$ = vraie si et seulement si existe un chemin simple à k sommets entre i et j
- si G a des auto-boucle en chaque sommet,
pour tout sommets i, j ,
 $A^k[i, j]$ = vraie si et seulement si existe un chemin simple avec **au plus** k sommets entre i et j

Calcul de la fermeture transitive sur les graphes denses

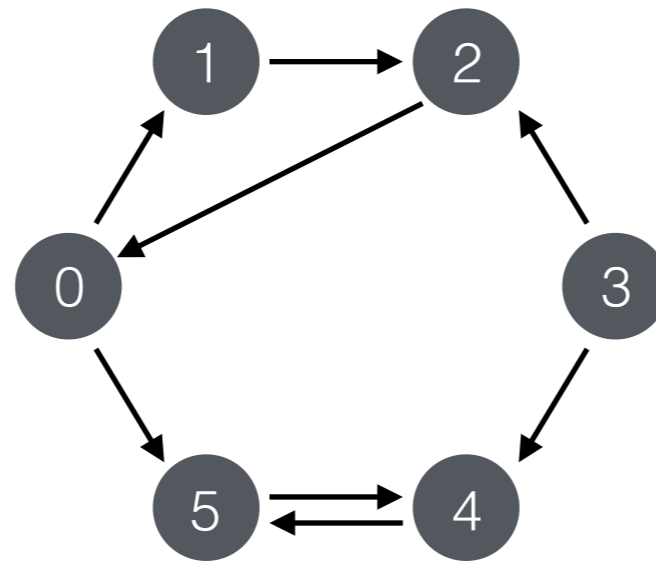
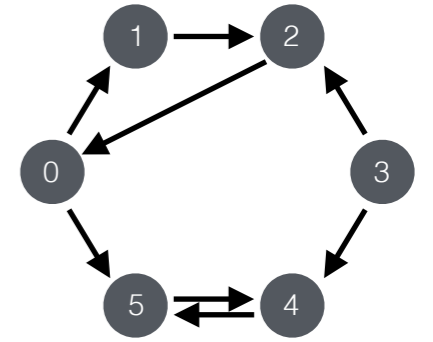
Puisque tous les chemins simples de G passent par au plus $|S|$ sommets, il suffit

- d'ajouter des auto-boucles en chaque sommet
- de calculer la puissance $|S|^{\text{ième}}$ de la matrice obtenue

$$A^* = (Id + A)^{|S|}$$

coût $O(S^4)$

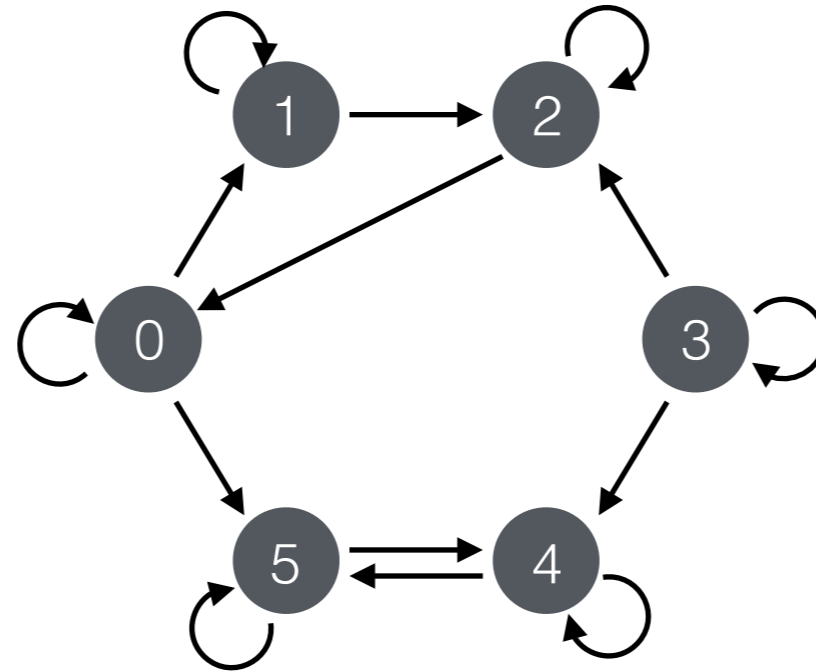
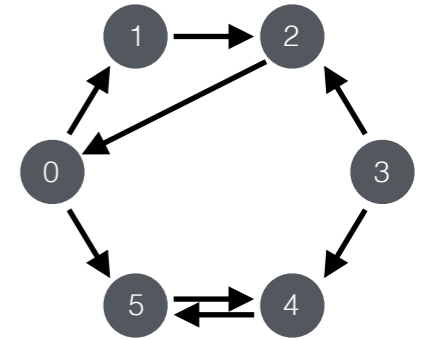
Exemple



A

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

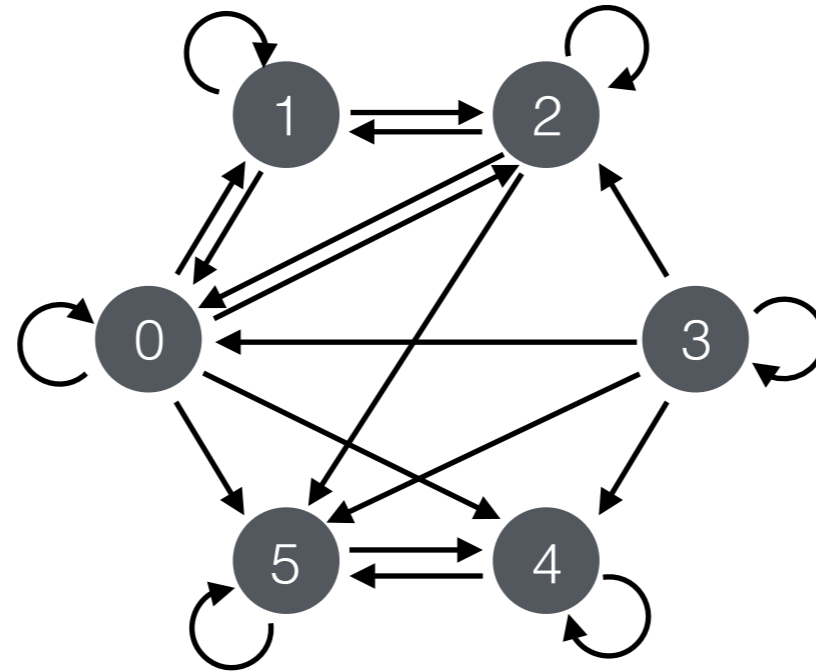
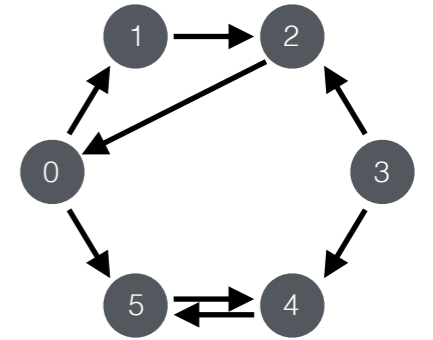
Exemple



$Id+A$

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

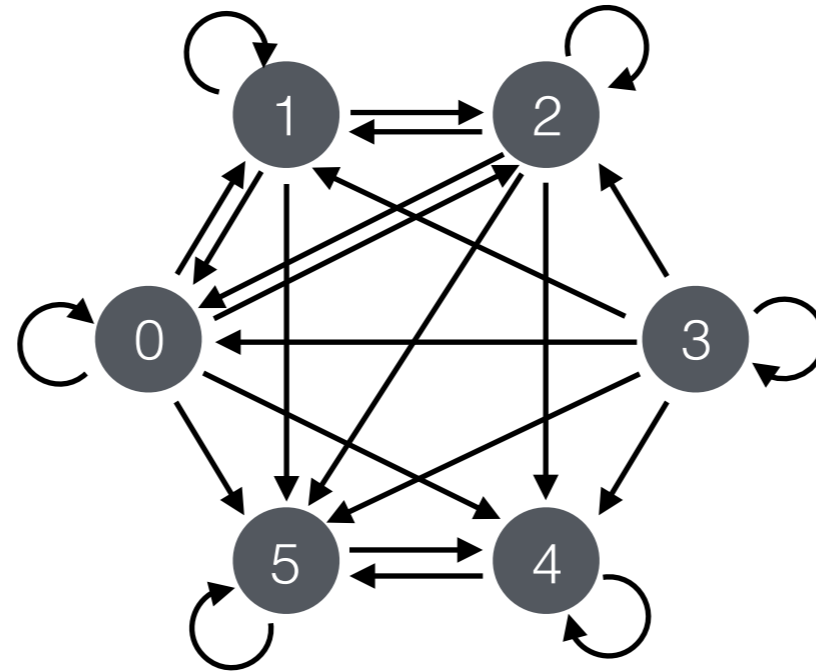
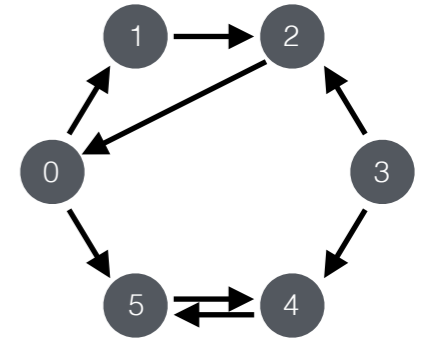
Exemple



$$(Id+A)^2$$

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

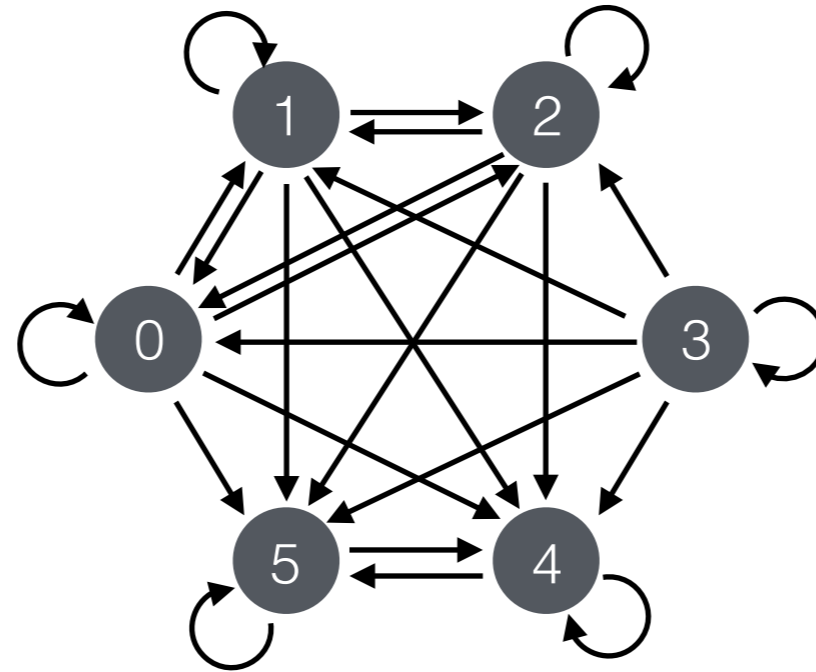
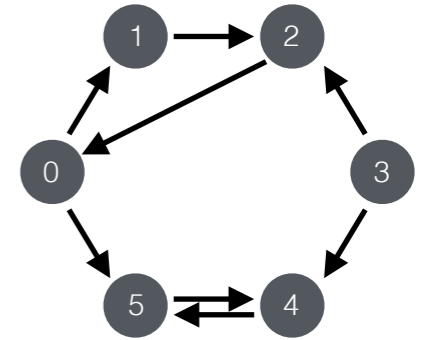
Exemple



$$(Id+A)^3$$

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

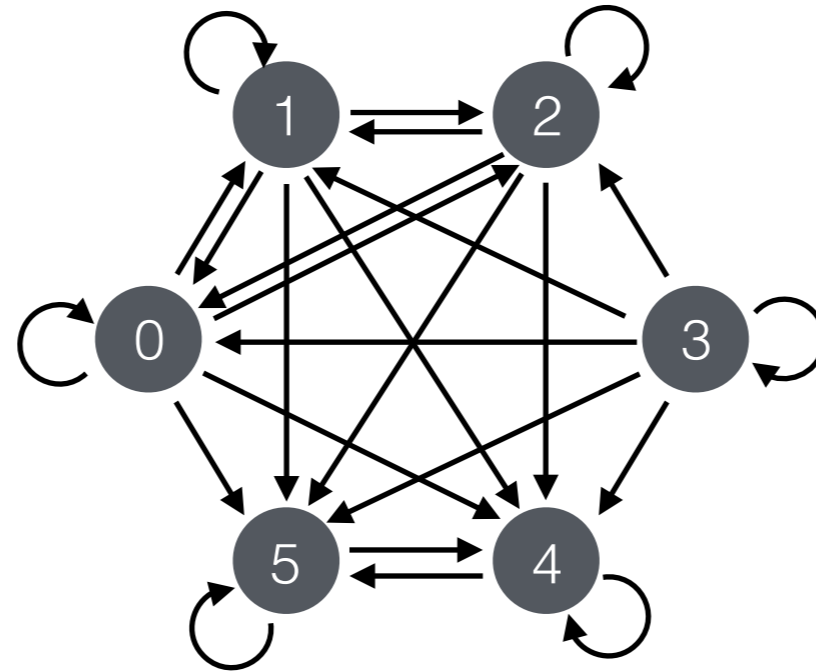
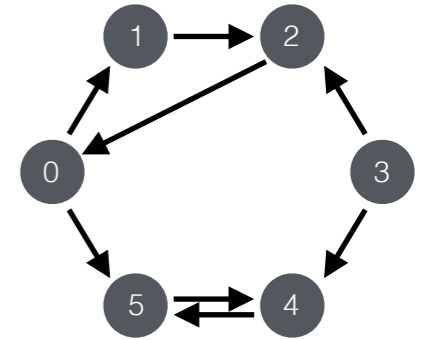
Exemple



$$(Id+A)^4$$

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

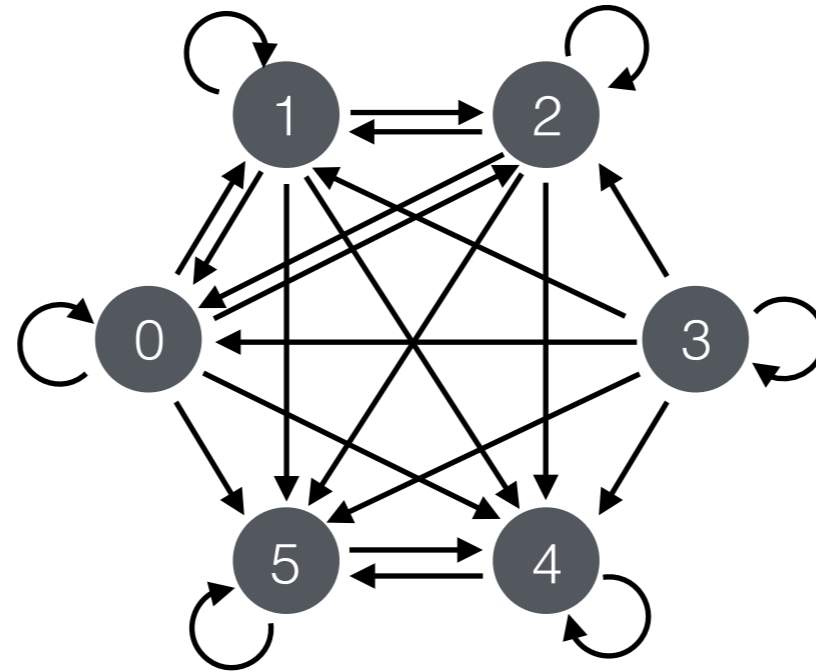
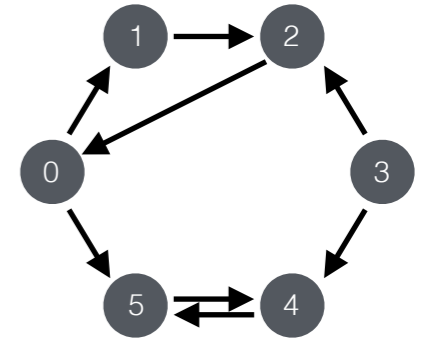
Exemple



$$(Id+A)^5$$

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

Exemple



$$(Id+A)^6$$

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

Algorithme de Warshall

Il est possible d'obtenir une complexité en $O(S^3)$

```
WARSHALL(A) =  
  A* ← copie(A)  
  pour tout k ∈ S  
    pour tout i ∈ S  
      pour tout j ∈ S  
        A*[i,j] ← A*[i,j] || (A*[i,k] && A*[k,j])  
  renvoie A*
```

Nous expliquerons cet algorithme dans un prochain cours

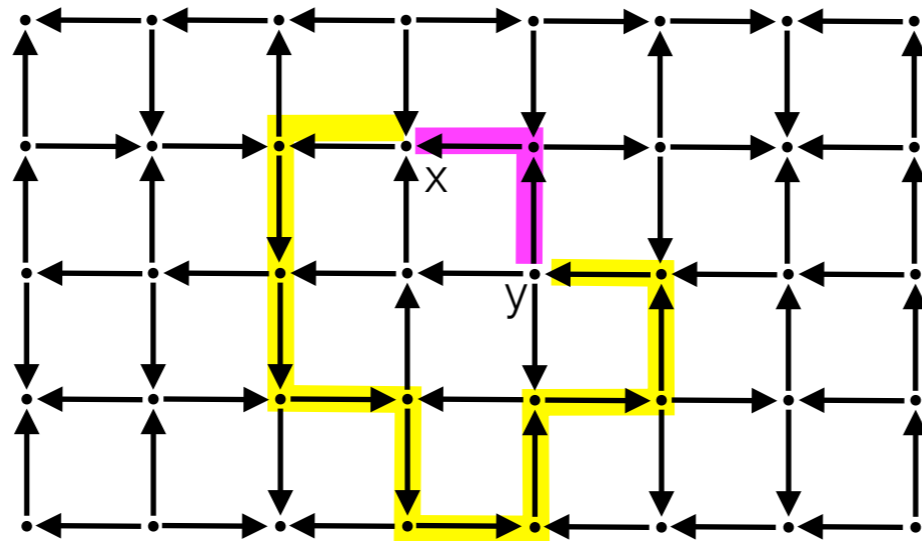
Composantes fortement
connexes

Vocabulaire

Définition

Dans un graphe orienté, deux sommets x et y sont fortement connectés si il existe un chemin de x à y et un chemin de y à x .

Exemple



x et y sont fortement connectés

Propriétés

Le connectivité forte est une relation d'équivalence

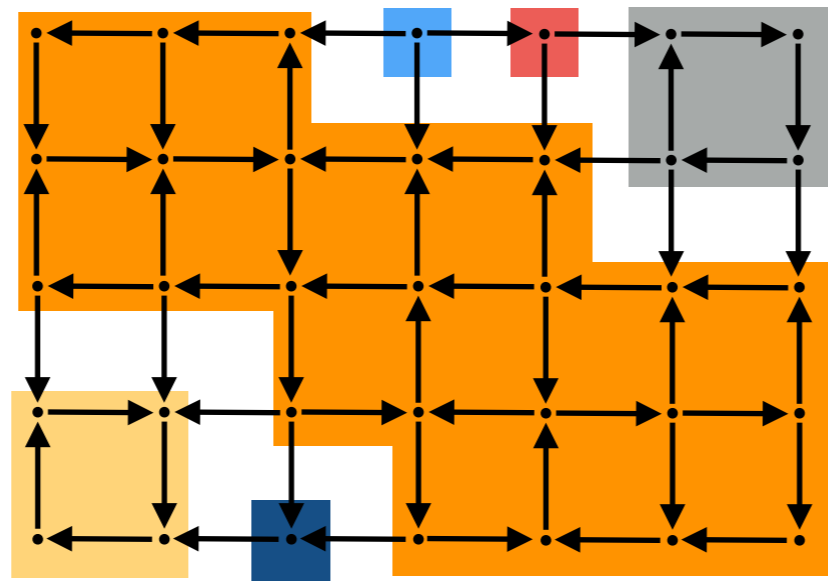
- x est fortement connecté à x (relation réflexive)
- si x est fortement connecté à y , alors y est fortement connecté à x (relation symétrique)
- si x est fortement connecté à y , et si y est fortement connecté à z , alors x est fortement connecté à z (relation transitive)

Vocabulaire

Définition

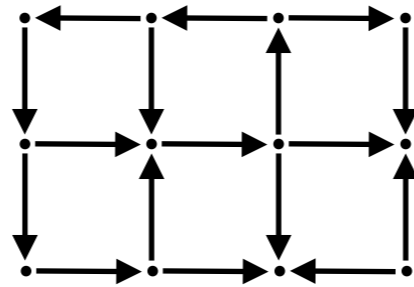
Dans un graphe orienté, une composante fortement connexe est un ensemble maximal de sommets fortement connectés.

Exemple



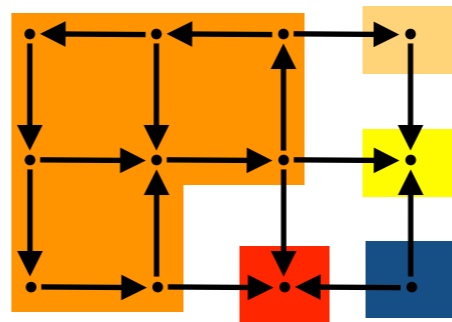
6 composantes fortement connexes

Exercice



Calculer les composantes fortement connexes (cfc)

Exercice



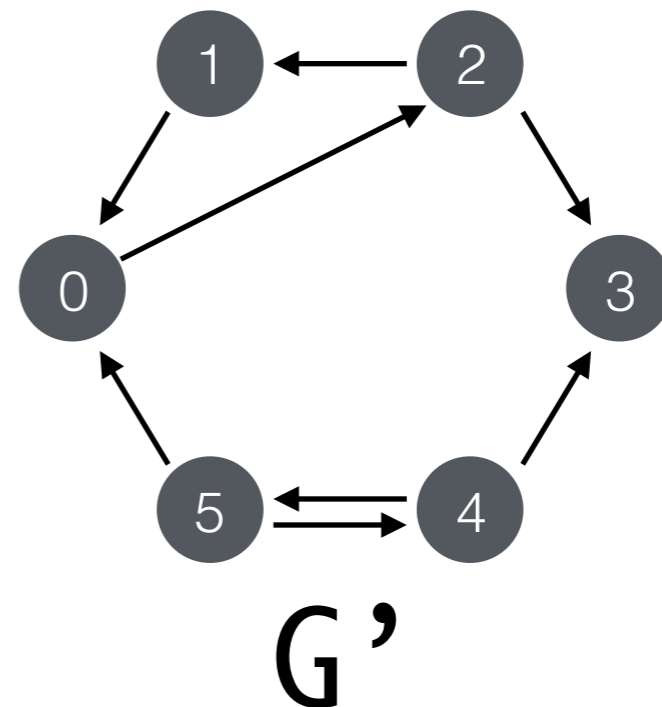
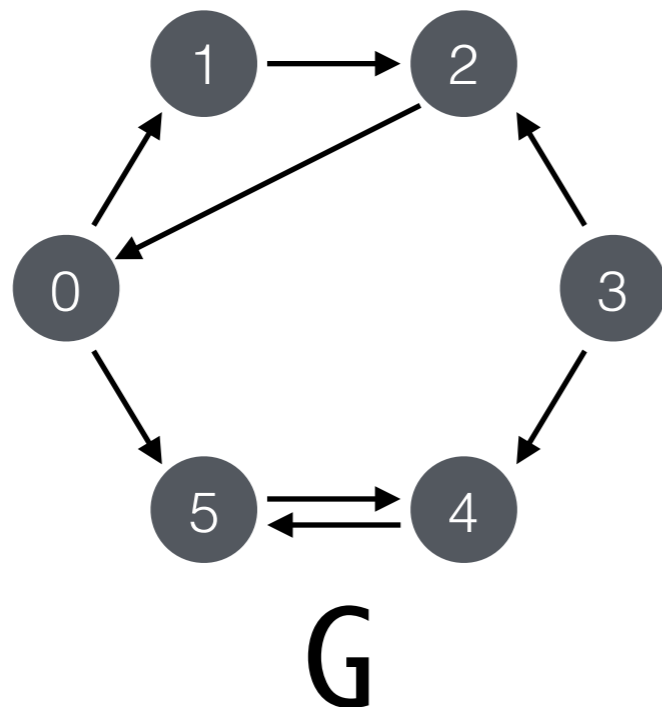
Calculer les composantes fortement connexes (cfc)

Vocabulaire

Définition

Le graphe *inverse* d'un graphe orienté $G=(S,A)$ est le graphe $G'=(S,A')$ possédant les même sommets mais où chaque arc $(i,j) \in A'$ si et seulement $(j,i) \in A$.

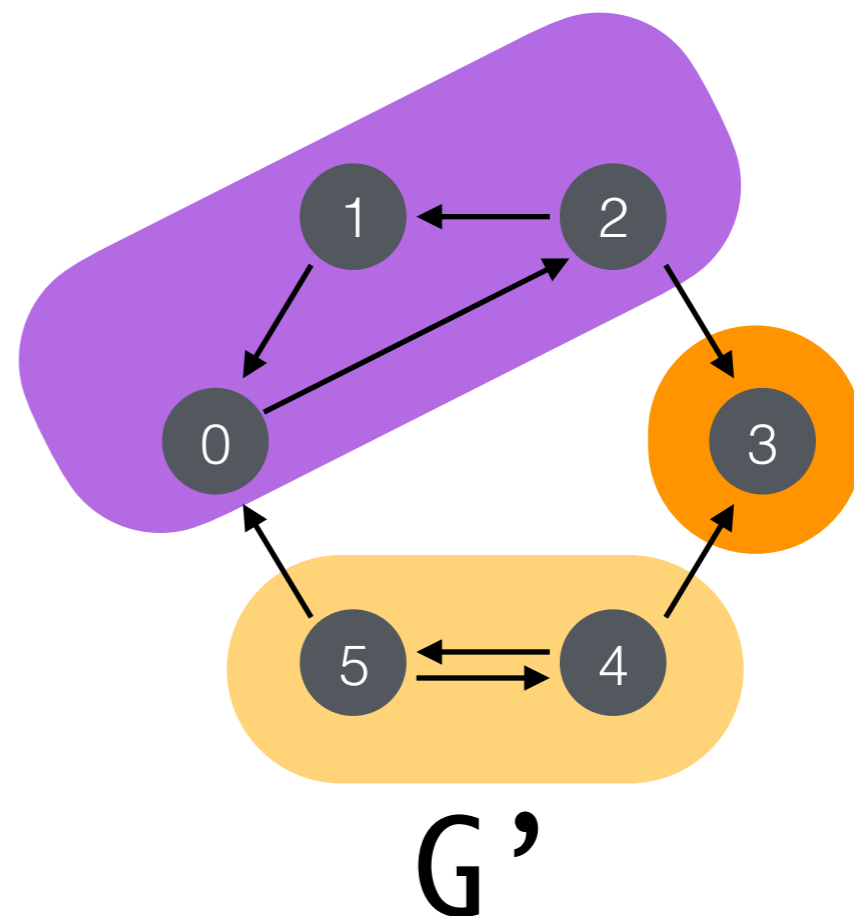
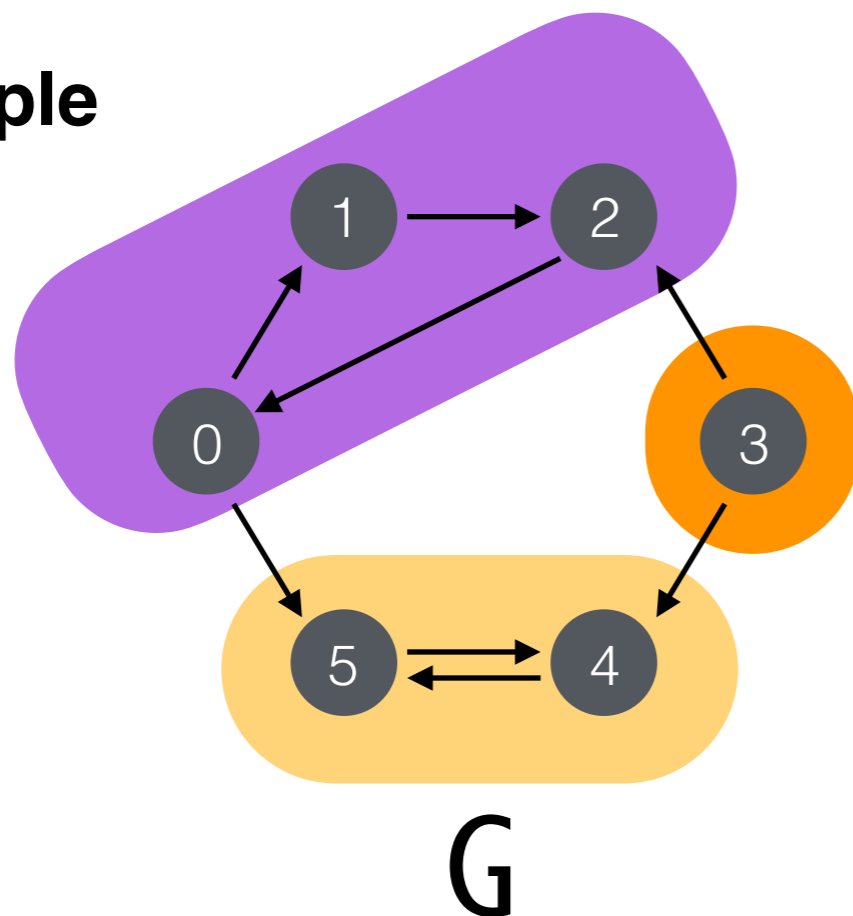
Exemple



Propriétés

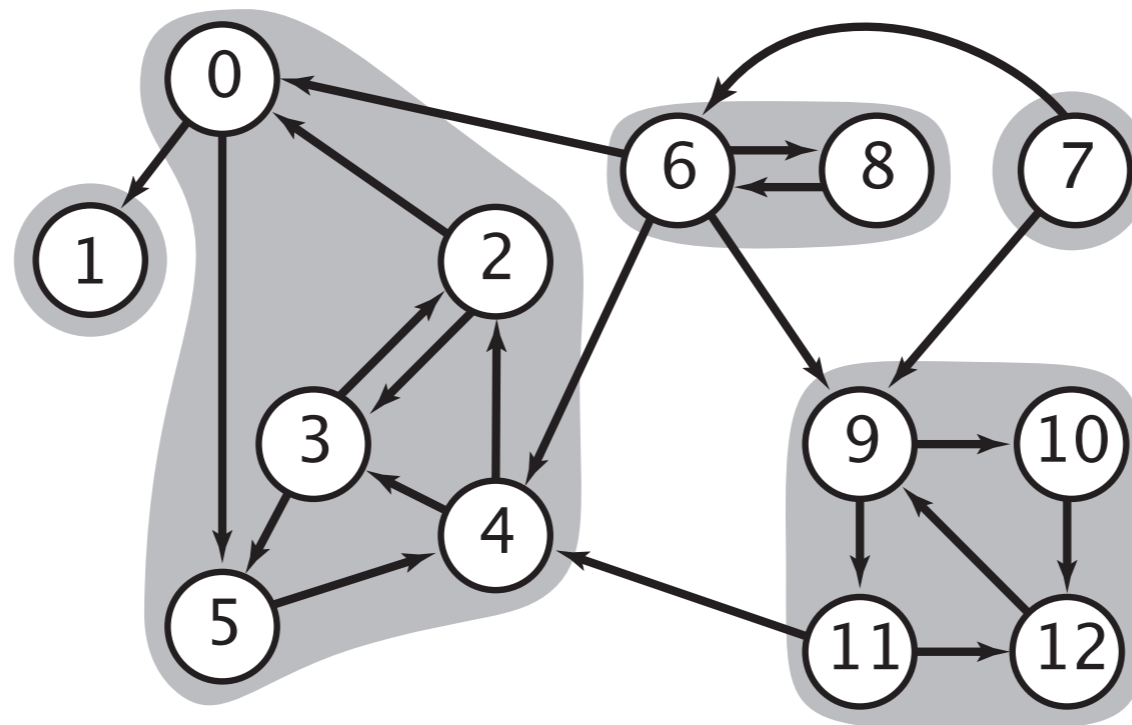
Les composantes fortement connexes d'un graphe G sont les mêmes que celles de son graphe inverse G'

Exemple



Propriétés

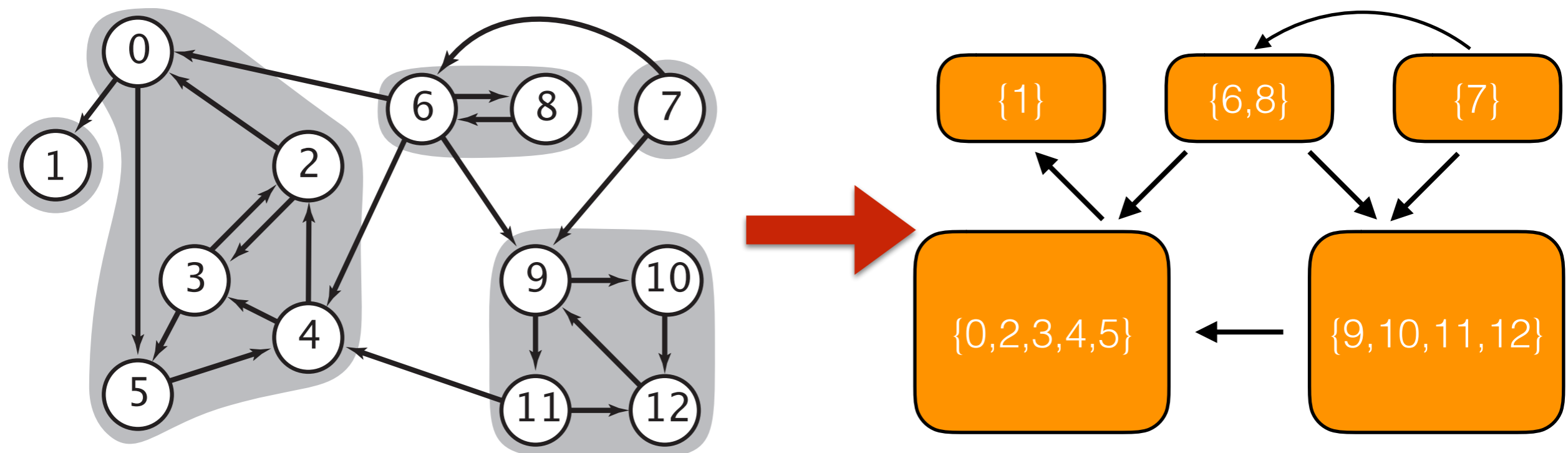
- Les composantes fortement connexes sont les classes d'équivalence de la relation d'équivalence associée
- Les classes d'équivalence forment une partition des sommets



Graphe quotient

On peut alors construire un nouveau graphe où

- les sommets sont les classe d'équivalence
- les arcs relient les classes pour lesquelles il existait au moins un arc dans le graphe initial



Propriétés

- le graphe quotient est un graphe orienté **acyclique**
- il donne une « vue d'avion » du graphe

Calcul des composantes fortement connexes : histoire

- années 60 : un problème classique
 - mais sans solution efficace (polynomial)
 - la complexité du problème n'est pas connue
- 1972 : algorithme linéaire proposé par Tarjan
 - une simple modification du parcours en profondeur
 - difficile à comprendre
- années 80 : algorithme linéaire de Kosaraju
 - plus simple à comprendre (2 parcours)
 - Kosaraju avait oublié ses notes de cours et a inventé l'algorithme pour préparer son cours !

Algorithme de Kosaraju

1. parcours en profondeur sur G pour calculer FIN (ordre suffixe)
2. calcul de l'inverse G' de G
3. parcours en profondeur sur G' , mais en itérant la boucle principale par ordre de FIN décroissant
4. les composantes fortement connexes sont les arbres de la forêt du 2e parcours

coût global $O(A+S)$

Exemple

