Théorie de graphes – Introduction

Frédéric Guinand, Stefan Balev

Master I - Le Havre

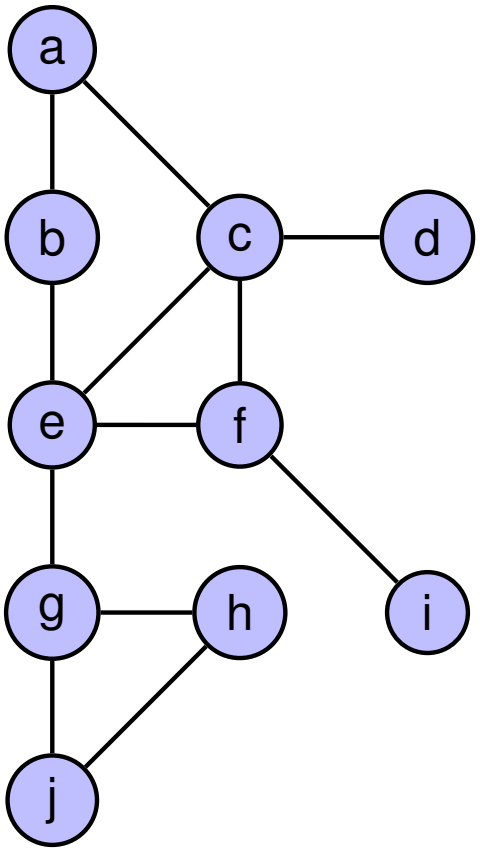
Représentation des graphes en machines

Représentation matricielle

- matrice d'adjacence
- matrice d'incidence sommets-arêtes
- matrice d'incidence sommets-arcs
- matrice laplacienne

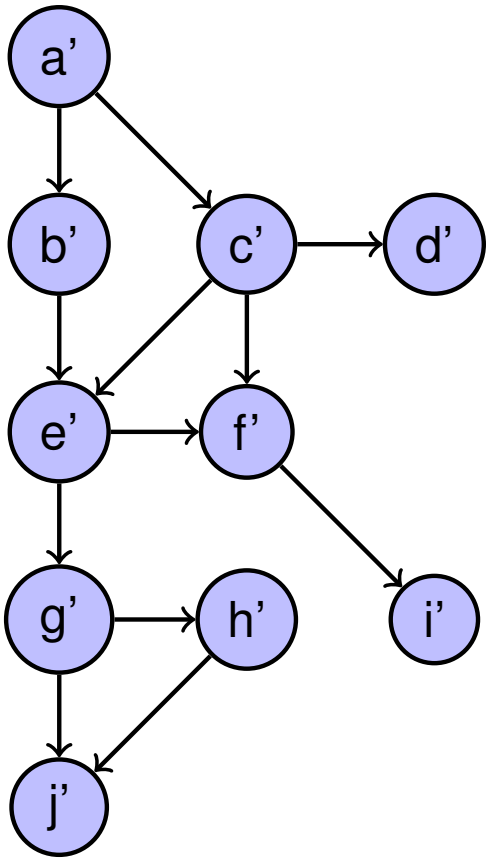
Théorie spectrale des graphes

Matrice d'adjacence



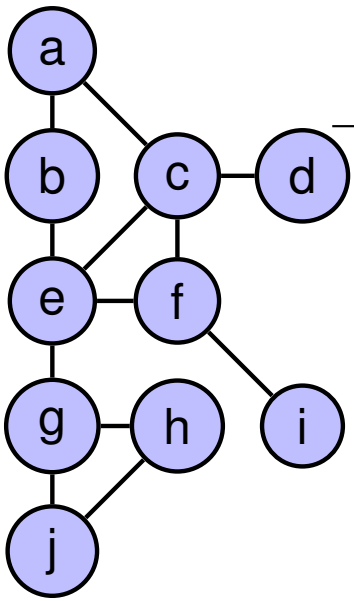
	a	b	c	d	e	f	g	h	i	j
a	0	1	1	0	0	0	0	0	0	0
b	1	0	0	0	1	0	0	0	0	0
c	1	0	0	1	1	1	0	0	0	0
d	0	0	1	0	0	0	0	0	0	0
e	0	1	0	0	0	1	1	0	0	0
f	0	0	1	0	1	0	0	0	1	0
g	0	0	0	0	1	0	0	1	0	1
h	0	0	0	0	0	0	1	0	0	1
i	0	0	0	0	0	1	0	0	0	0
j	0	0	0	0	0	0	0	1	0	0

Matrice d'adjacence



	a'	b'	c'	d'	e'	f'	g'	h'	i'	j'
a'	0	1	1	0	0	0	0	0	0	0
b'	0	0	0	0	1	0	0	0	0	0
c'	0	0	0	1	1	1	0	0	0	0
d'	0	0	0	0	0	0	0	0	0	0
e'	0	0	0	0	0	1	1	0	0	0
f'	0	0	0	0	0	0	0	0	1	0
g'	0	0	0	0	0	0	0	1	0	1
h'	0	0	0	0	0	0	0	0	0	1
i'	0	0	0	0	0	0	0	0	0	0
j'	0	0	0	0	0	0	0	0	0	0

Matrice d'incidence sommets-arêtes



	(ab)	(ac)	(be)	(cd)	(ce)	(cf)	(ef)	(eg)	(fi)	(gh)	(gj)	(hj)
a	1	1	0	0	0	0	0	0	0	0	0	0
b	1	0	1	0	0	0	0	0	0	0	0	0
c	0	1	0	1	1	1	0	0	0	0	0	0
d	0	0	0	1	0	0	0	0	0	0	0	0
e	0	0	1	0	1	0	1	1	0	0	0	0
f	0	0	0	0	0	1	1	0	1	0	0	0
g	0	0	0	0	0	0	0	1	0	1	1	0
h	0	0	0	0	0	0	0	0	0	1	0	1
i	0	0	0	0	0	0	0	0	1	0	0	0
j	0	0	0	0	0	0	0	0	0	0	1	1

Matrice laplacienne

	a	b	c	d	e	f	g	h	i	j
a	2	-1	-1	0	0	0	0	0	0	0
b	-1	2	0	0	-1	0	0	0	0	0
c	-1	0	4	-1	-1	-1	0	0	0	0
d	0	0	-1	1	0	0	0	0	0	0
e	0	-1	0	0	4	-1	-1	0	0	0
f	0	0	-1	0	-1	3	0	0	-1	0
g	0	0	0	0	-1	0	3	-1	0	-1
h	0	0	0	0	0	0	-1	2	0	-1
i	0	0	0	0	0	-1	0	0	1	0
j	0	0	0	0	0	0	-1	-1	0	2

Matrice laplacienne normalisée

- Les coefficients A_{ii} valent 1 si degré$(i) > 0$
- Les coefficients A_{ij} valent 0 si i et j ne sont pas adjacents.
- Les coefficients A_{ij} valent $\frac{-1}{\sqrt{\text{degré(i)} \times \text{degré(j)}}}$

Matrice laplacienne normalisée

	a	b	c	d	e	f	g	h	i	j
a	1	$\frac{-1}{2}$	$\frac{-1}{2\sqrt{2}}$	0	0	0	0	0	0	0
b	$\frac{-1}{2}$	1	0	0	$\frac{-1}{2\sqrt{2}}$	0	0	0	0	0
c	$\frac{-1}{2\sqrt{2}}$	0	1	$\frac{-1}{2}$	$\frac{-1}{4}$	$\frac{-1}{2\sqrt{3}}$	0	0	0	0
d	0	0	$\frac{-1}{2}$	1	0	0	0	0	0	0
e	0	$\frac{-1}{2\sqrt{2}}$	$\frac{-1}{4}$	0	1	$\frac{-1}{2\sqrt{3}}$	$\frac{-1}{2\sqrt{3}}$	0	0	0
f	0	0	$\frac{-1}{2\sqrt{3}}$	0	$\frac{-1}{2\sqrt{3}}$	1	0	0	$\frac{-1}{\sqrt{3}}$	0
g	0	0	0	0	$\frac{-1}{2\sqrt{3}}$	0	1	$\frac{-1}{\sqrt{6}}$	0	$\frac{-1}{\sqrt{6}}$
h	0	0	0	0	0	0	$\frac{-1}{\sqrt{6}}$	1	0	$\frac{-1}{2}$
i	0	0	0	0	0	$\frac{-1}{\sqrt{3}}$	0	0	1	0
j	0	0	0	0	0	0	$\frac{-1}{\sqrt{6}}$	$\frac{-1}{2}$	0	1

Limitations de la représentation matricielle

1. problème de mémoire

   $\Rightarrow$ graphe de 1 million de sommets $\rightarrow 10^{12}$ cellules

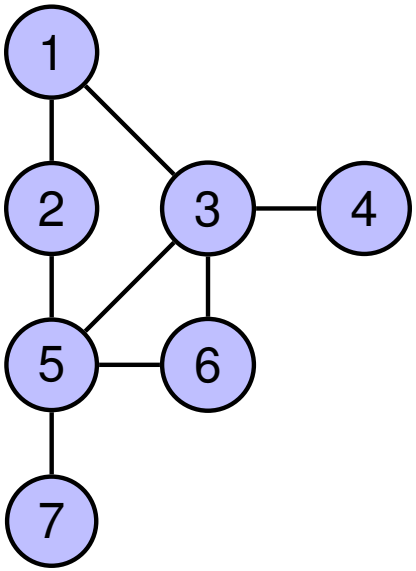
   $\Rightarrow$ 125 Go ! !

   mémoire utile (si degré moyen $\sim$ 1000) : 125 Mo ! !

   $\Rightarrow$ nécessité de coder les matrices différemment matrices creuses

   $\Rightarrow$ manipulation moins simple
2. structure inadaptée pour étudier l'évolution du graphe (retrait ou ajout de sommets problématique).

Représentation par tableaux

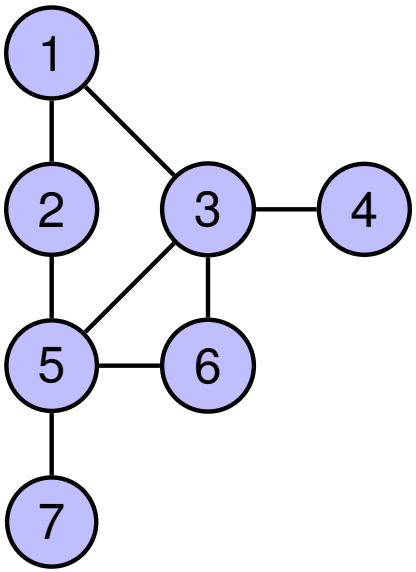


sommet	1	2	3	4	5	6	7
nombre de voisins	2	2	4	1	4	2	1
indice dans T_A	1	3	5	9	10	14	16

tableau T_A

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	3	1	5	1	4	5	6	3	2	3	6	7	3	5	5

Représentation par structure de voisinage



1 / next → 2 / next → 3 / fin

2 / next → 1 / next → 5 / fin

3 / next → 5 / next → 6 / next → 1 / next → 4 / fin

4 / next → 3 / fin

5 / next → 2 / next → 6 / next → 7 / next → 3 / fin

6 / next → 5 / next → 3 / fin

7 / next → 5 / fin

Approche objet

- Un graphe $\rightarrow$ des attributs (nom, orientation, tableau de sommets, tableau de liens, etc) et des méthodes (ajouter un sommet/une arête, supprimer un sommet/une arête, déterminer le degré moyen, dessiner la distribution des degrés, etc).
- Un sommet $\rightarrow$ des attributs (identifiant, attributs divers, couleur, etc) et des méthodes (voisinage, changer la couleur, la valeur, etc).
- Une arête $\rightarrow$ des attributs et des méthodes.

Bilan - Matrices

+ simple à manipuler : tableau à deux dimensions
+ outils mathématiques puissants disponibles
- lecture des voisinages en $O(n)$
- gourmand en mémoire pour les graphes peu denses
- inadapté à la prise en compte de l'évolution dans les graphes
- structures de données complémentaires si informations additionnelles (couleur, etc).

Bilan - Tableaux

+ simple à manipuler : tableau à deux dimensions
+ optimisation de l'espace utilisé
- inadapté à la prise en compte de l'évolution dans les graphes
- structures de données complémentaires si informations additionnelles (couleur, etc).

Bilan - Structures

+ voisinages directement accessibles
+ optimisation mémoire
+ adapté à l'évolution des graphes
+ possibilité d'intégrer des informations additionnelles (couleur, etc).
- délicat à manipuler
- chaque structure est dédiée à une application donnée $\Rightarrow$ faible réutilisabilité des codes.

Bilan - Objets

+ voisinages directement accessibles
+ optimisation de la mémoire
+ adapté à l'évolution des graphes
+ possibilité d'intégrer des informations additionnelles (couleur, etc).
+ simple à manipuler et à visualiser (java)
+ réutilisabilité plus importante
- verbeux

Algorithmes de parcours

Algorithmes de parcours

- exploration du graphe guidée par la structure
- basée sur les voisinages
- base nécessaire pour de nombreux problèmes (plus court chemin, routage, flot, connexité, diamètre, coloration, etc.)
- deux principes possibles :
  - parcours déterministe
  - parcours probabiliste (guidé par le hasard) $\rightarrow$ marche aléatoire (*random walk*)
- parcours profondeur d'abord : *Depth-First-Search*
- parcours largeur d'abord : *Breadth-First-Search*

Parcours

Un parcours d'un graphe $G = (V, E)$ à partir d'un sommet s est la génération d'une liste L de sommets, telle que :

- L contient tous les sommets de G atteignables à partir de s ;
- s est le premier élément de L ;
- chaque sommet de G est au plus une fois dans L ;
- pour chaque sommet $v \neq s$ dans L il existe un voisin de v avant lui dans L

Algorithmes de parcours

- principe
- algorithme
- complexité algorithmique
- correction de l'algorithme

Algorithme générique

marquer(s), ouvrir(s)
TantQue (il existe un sommet ouvert) **Faire**
    $v \leftarrow$ un sommet ouvert
    Si (tous les voisins de v sont marqués) **Alors**
        fermer(v)
    Sinon
        $u \leftarrow$ un voisin de v non-marqué
        marquer(u), ouvrir(u)
    finSi
finTantQue

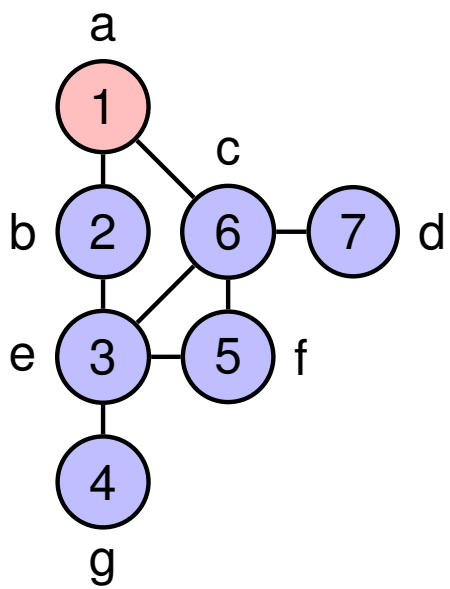
marquer(s) : ajouter s à la liste L
sommet ouvert : un sommet marqué, mais dont on n'est pas sûr que tous les voisins le soient

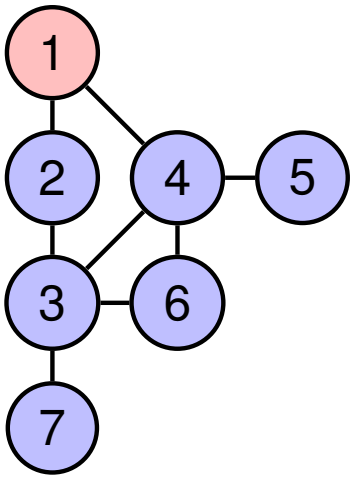
Correction de l'algorithme

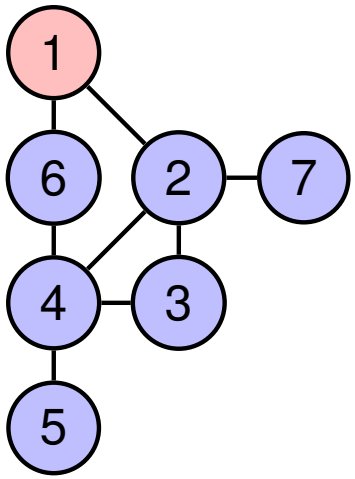
- Est-ce que l'algorithme se termine ?
- Est-ce que la liste construite correspond à la définition de parcours ?

DFS

Principe : prendre toujours le dernier sommet ouvert







Algorithme basé sur une pile

Pour $v \in V$ **Faire**
    marqué[v] $\leftarrow$ faux
finPour
P.empiler(s)
TantQue P $\neq \emptyset$ **Faire**
    $v \leftarrow$ P.dépiler()
    Si (non marqué[v]) **Alors**
        marqué[v] = vrai
        Pour $u \in$ voisins(v) **Faire**
            Si (non marqué[u]) **Alors**
                P.empiler(u)
            finSi
        finPour
    finSi
finTantQue

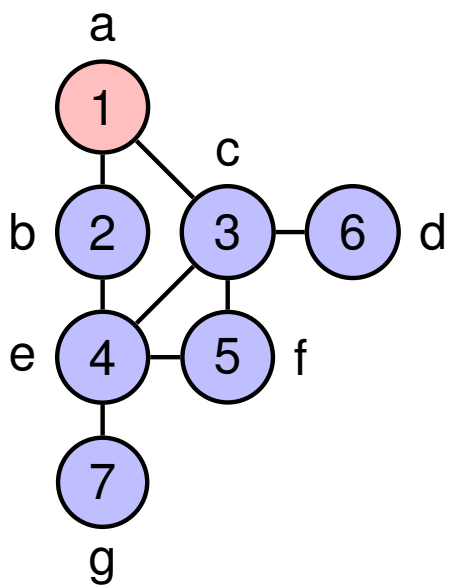
Complexité : $O(m)$

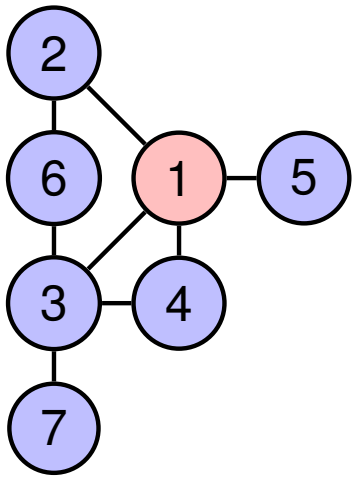
Algorithme récursif

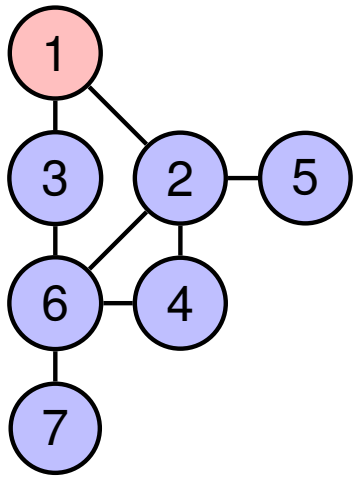
```
DFS(marqué, v)
debut
        marqué[v] = vrai
        Pour u ∈ voisins(v) Faire
                Si (non marqué[u]) Alors
                    DFS(marqué, u)
                finSi
        finPour
fin
```

BFS

Principe : prendre toujours le plus ancien sommet ouvert







Exercices

1. Sur la base de l'algorithme DFS, proposez un algorithme qui utilise une file
2. Calculer la complexité de ce nouvel algorithme
3. Justifiez la correction de cet algorithme
4. Proposez un algorithme qui calcule le plus court chemin entre un sommet s et tous les autres sommets dans un graphe G