Fouille de graphes et réseaux sociaux

Raphaël Fournier-S'niehotta

CNAM Paris, fournier@cnam.fr

HTT-FOD
RCP216
2020-2021

le cnam

Intégrité académique

Le contexte académique requiert de l'honnêteté sur l'originalité du travail présenté et sur la réalité de l'autorat.

Il convient donc, sauf autorisations explicites :

- de ne pas recopier le travail d'une autre personne
- de ne pas colloraborer
- de ne pas faire passer le travail d'une personne pour celui d'une autre

La commission de discipline du CNAM peut prononcer des sanctions en cas de plagiat ou de fraude lors d'un examen, allant jusqu'à l'exclusion ferme de l'établissement, voire de tout établissement d'enseignement supérieur (cf. article R811-36 du Code de l'éducation).

Présentation de la partie

5 séances :

1. Introduction aux graphes et réseaux sociaux
2. Propriétés des graphes, modèles
3. Communautés et dynamique

4. Visualisation de données
5. Visualisation de graphes

En anglais :

- Social Network Analysis [and Mining] (SNA(M))
- DataViz

Plan

Introduction

Expérience de Milgram

Expérience de Milgram (1967)

Stanley Milgram (1933-1984), psychologue social américain. Connu notamment pour les expériences de soumission à l'autorité.

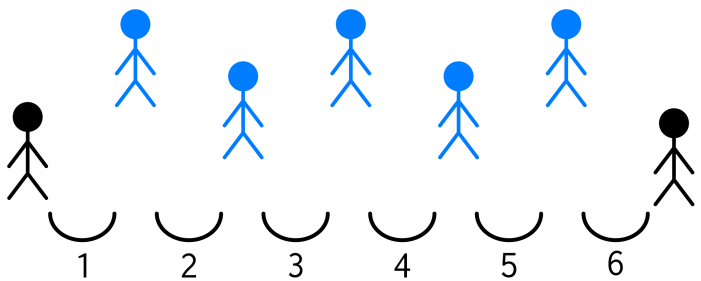
- Objectif de l'expérience : faire transiter une lettre de Omaha, NE à Boston, MA

Règle :

- une personne initie la chaîne
- transition de la main à la main à des personnes que l'on connaît, chacune étant supposée se rapprocher de la destination

Expérience de Milgram (1967)

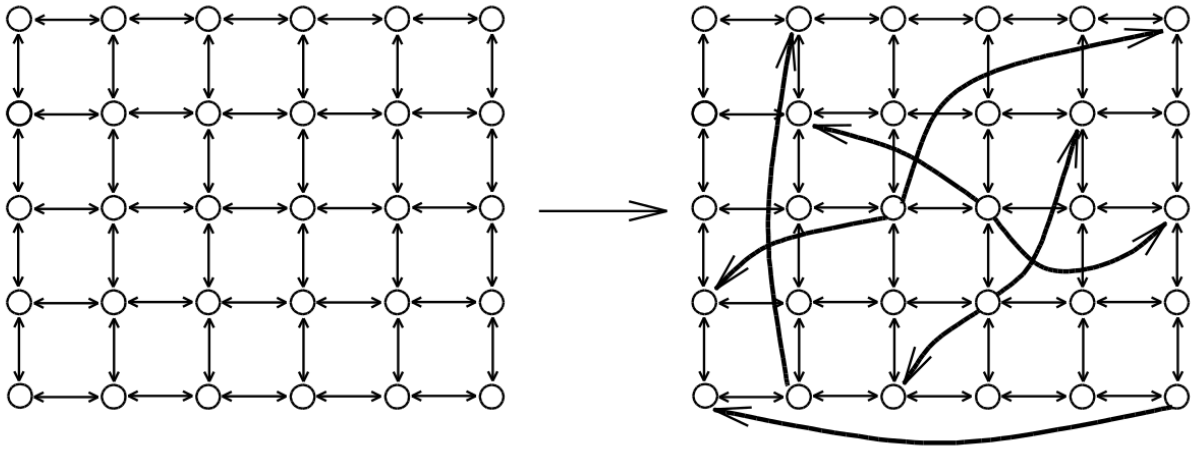
- Résultats
  - 64 lettres sur 296 arrivent
  - Chemins avec 5 intermédiaires en moyenne.
- Remarques :
  - Chemin interrompu $\neq$ Il n'existe pas de chemin.
  - Chemin de longueur x $\neq$ Il n'existe pas de chemin de longueur <x
- Conclusions :
  - Il existe des chemins courts.
  - Les intermédiaires arrivent à les trouver sans connaissance globale du réseau.



Expérience de Milgram : modélisation

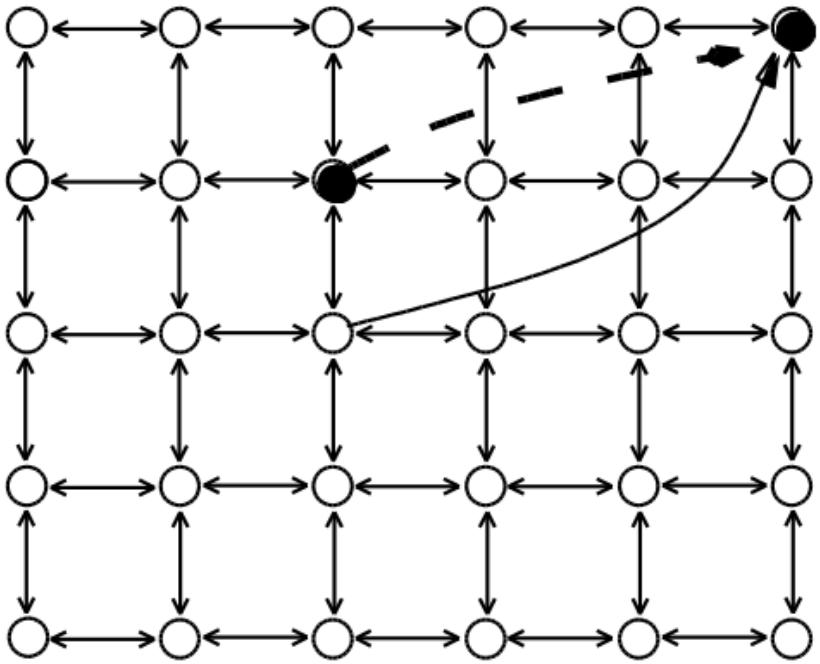
- Objectif : formaliser l'expérience de Milgram
- Travaux de D. Watts/S. Strogatz, puis de J. Kleinberg (vers 1998)

- Initialement une grille (amis proches).
- On ajoute q voisins quelconques à chaque sommet (amis lointains).



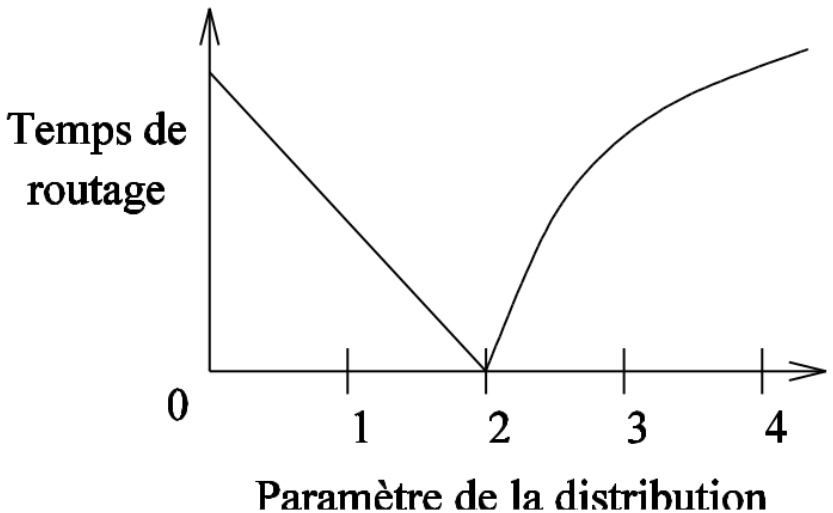
Expérience de Milgram : modélisation

- Un sommet connaît :
  - Sa position, celle de ses voisins, celle de la destination.
  - Il envoie le message à son voisin le plus proche de la destination.



Expérience de Milgram : modélisation

- Un seul lien supplémentaire pour chaque sommet u.
- La destination choisie avec une probabilité dépendant de sa distance à u.
- Dans la majorité des cas, pas de chemins courts
- **Mais :**

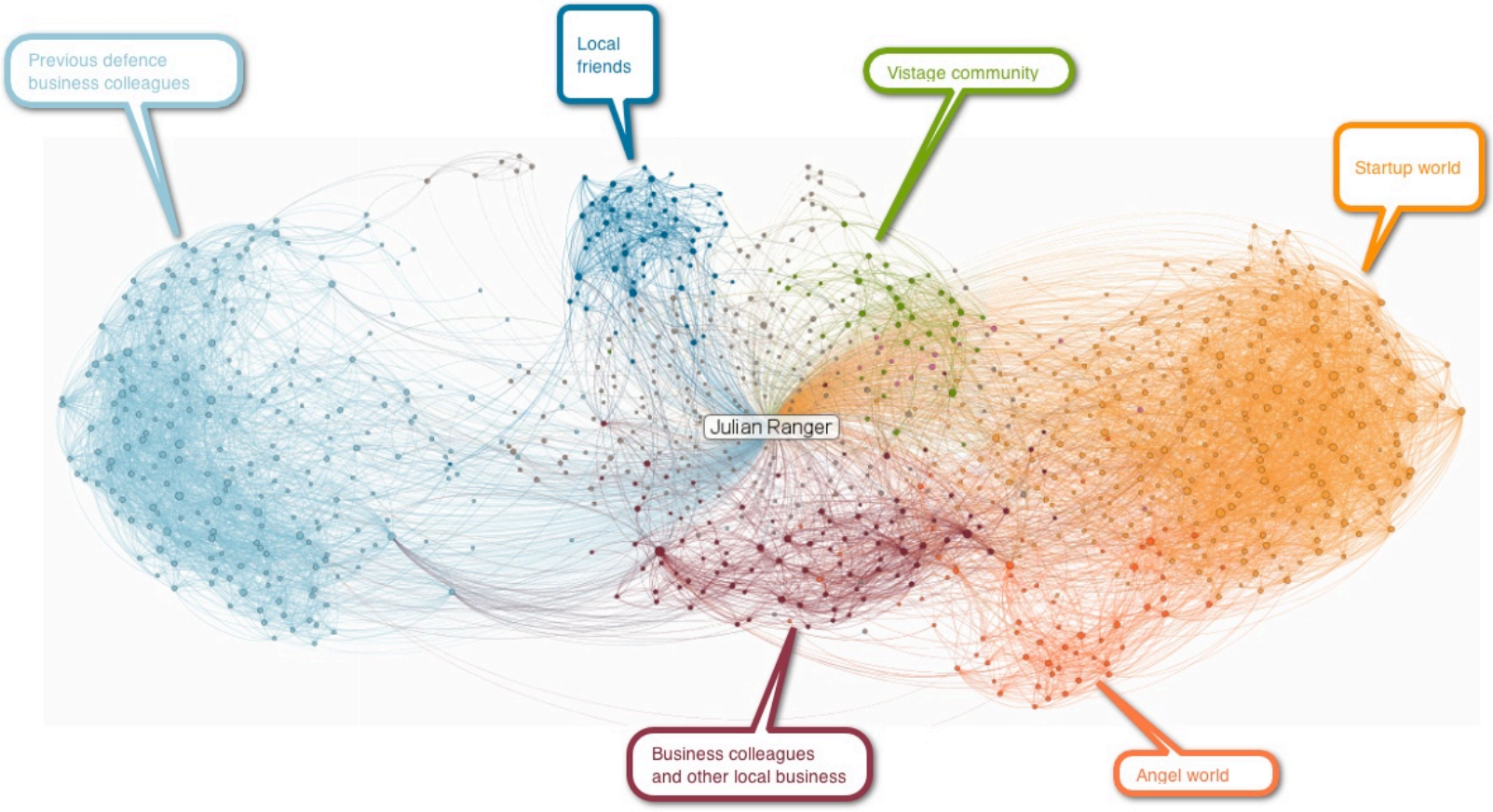


Petits mondes

- On parle de “petit monde” si tous les individus d’un réseau peuvent être connectés par un chemin court
- Des chercheurs ont mis en évidence ce phénomène dans des réseaux de domaines très variés (neurones de C. elegans, électricité aux USA)
- modèle
  - structure sous-jacente traduisant les liens entre la plupart des nœuds
  - quelques liens aléatoires expliquent le diamètre faible
  - spécialisation régionale avec transfert efficace d’information entre zones

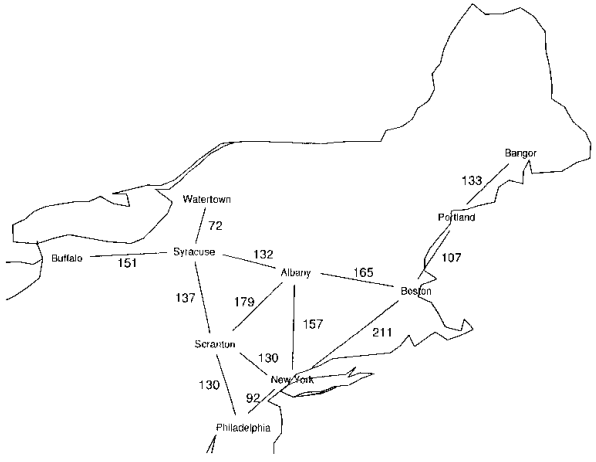
Exemples de réseaux/graphes

Réseau personnel : LinkedIn Maps



Réseau routier national

- Noeuds : les villes / communes
- Arêtes : (auto)routes
- Valuation possible : distance, ou temps de parcours
- Orientation possible



Des questions :

- quel est le plus court chemin passant par des villes données ?
- quel chemin traverse le moins de villes pour aller d'un point à un autre ?
- peut-on passer par toutes les villes sans passer deux fois par la même route ? (*voyageur de commerce*)

Individus : Kevin Bacon Game

Kevin Bacon (1958–), acteur américain, qui a joué dans plus de 75 films.

- Graphe d'acteurs
  - Deux acteurs sont reliés s'ils ont joué dans un même film.
  - Distance entre acteurs ?
    - http://oracleofbacon.org/
    - Distance entre Tom Cruise et Clint Eastwood ? 2 (acteur commun entre Space Cowboys et Eyes Wide Shut)
    - Distance entre Mickey Mouse et Omar Sy ? 4
- graphes produits à partir de http://www.imdb.com/interfaces
- calculs de plus courts chemins

Individus : possesseurs de fichiers P2P

- Propagation d'un fichier d'utilisateurs en utilisateurs
  - ♠ video
- Problèmes et biais de mesure
  - dynamicité du réseau
  - parcours non exhaustif et depuis une source

Les réseaux sociaux en ligne

- Blogs, wikis
- Microblogs
- Actualités sociales
- Bookmarks
- Partage de contenus multimédia
- Site de notations
- Q & A

Caractéristiques des réseaux en ligne

- Participation
  - les réseaux sociaux encouragent les contributions (contenu) et les retours (likes) d'un grand nombre de personnes (frontière public/media floue)
- Ouverture
  - réduction des barrières au partage d'information, au commentaire, à la réutilisation de contenus
- Conversation
  - si les médias traditionnels ont un modèle "un vers tous", les réseaux sociaux mettent l'accent sur des relations bilatérales
- Communauté
  - les réseaux sociaux favorisent la création et le développement de communautés (d'intérêts, d'opinion, etc.)
- Connexion
  - l'essor de ces réseaux repose sur une perméabilité aux contenus externes, à la facilité d'accès (liens) vers d'autres ressources et personnes

Autres types de réseaux étudiés

- informatique : pages Web, routeurs, P2P, etc.
- biologie : protéines, neurones cérébraux, etc.
- sciences sociales : amitiés, collaboration, contacts sexuels, etc.
- économie : échanges financiers
- histoire : mariages
- linguistique : synonymie, co-occurrence
- transports : réseau aérien, électrique

Propriétés et problématiques communes

Présentation du cours

Objectifs

Comprendre le comportement des entités qui interagissent dans le système étudié, et ce qui les gouverne

- On cherche donc :
  - quelle est la structure des graphes
  - quelle est l'évolution de cette structure
  - quels sont les phénomènes reposant sur l'existence de ce réseau

Graphes et fouille de données

D'un point de vue Data Mining, un réseau social (graphe), c'est :

- un jeu de données souvent très hétérogène
- multirelationnel et de grande taille
- les noeuds (sommets) sont les objets
- les arêtes sont les relations
- nœuds et arêtes peuvent avoir des attributs, rendant complexe l'analyse

Applications : domaines

- Informatique
  - Réseaux : routage, protocoles, sécurité
  - P2P : conception de systèmes, déviances
  - Web : indexation, moteurs de recherche
  - Dessin de graphes
- Sociologie :
  - Diffusion d'innovations, rumeurs
  - Identification de communautés
- Épidémiologie
  - Diffusion de virus, vaccination

Applications : informations

- Identification du rôle des individus :
  - Leader
  - Suiveur
  - Intermédiaire
- Identification de l'importance d'un groupe en analysant :
  - la taille
  - la cohésion
  - les profils
  - les relations internes et externes
- Repérer les doublons (même réseau)

Méthodologie

- Outils formels
  - Théorie des graphes
  - Analyse statistique
  - Modélisation probabiliste
- Études expérimentales
  - Simulation
  - Utilisation de données réelles
- Étudier des applications
  - Comprendre en profondeur certains réseaux
  - Extraction de concepts généraux

Ce cours

- Problématiques classées dans 4 grandes catégories :
  - Mesure
    - Comment mesurer les réseaux réels ?
  - Modélisation
    - A quoi ressemblent-ils ?
    - Peut-on créer des réseaux artificiels similaires ?
  - Analyse
    - quelles sont leur propriétés ?
  - Algorithmique
    - Comment calculer sur de grands graphes ?
- Détection de communautés (clustering)
- Visualisation

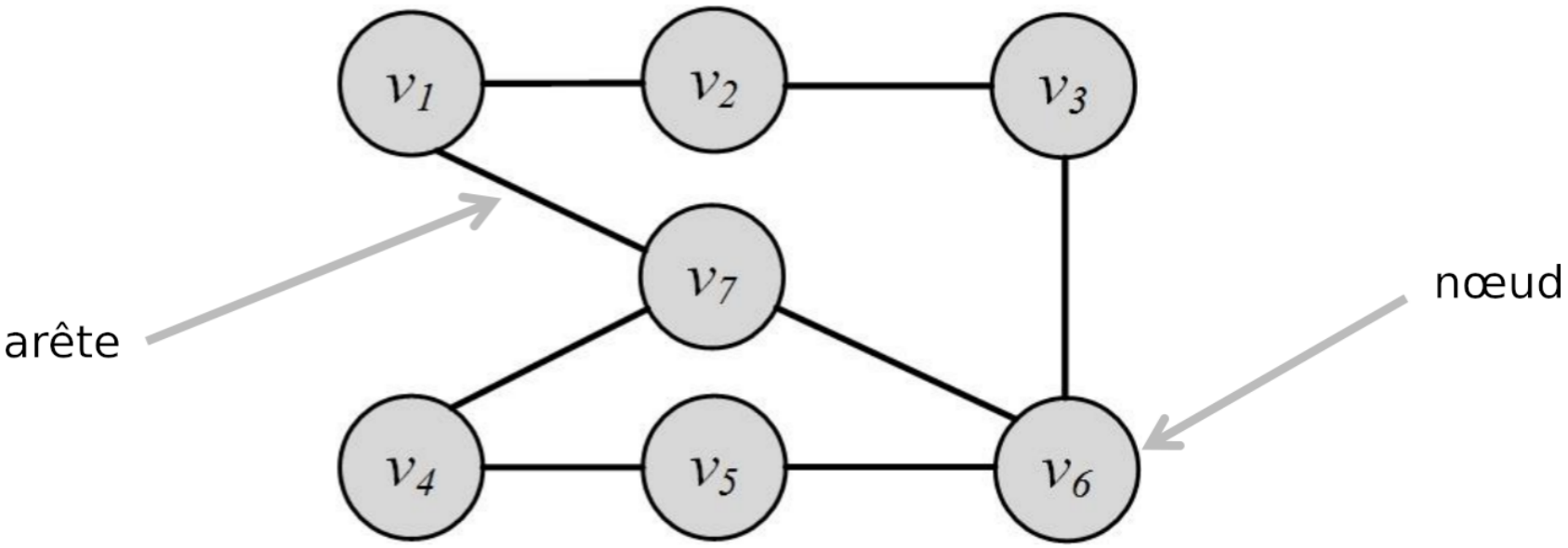
Éléments sur les graphes

Vocabulaire

Nœuds et arêtes

On assimile souvent "réseau" et "graphe". Un graphe est :

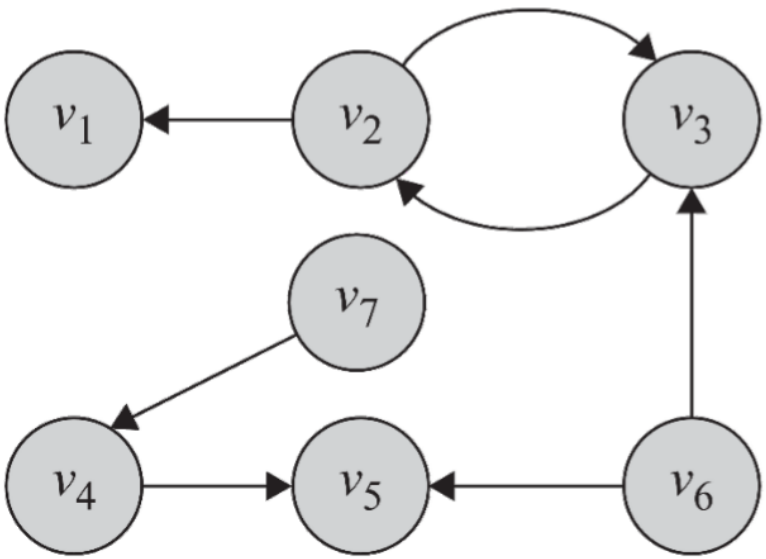
- un ensemble de sommets (aussi appelés nœuds, ou acteurs). On le note généralement V, de l'anglais vertex (vertices).
- un ensemble d'arêtes (liens). On le note généralement E, de l'anglais edge(s)
- $G = (V, E)$, |V| = n, |E| = m.



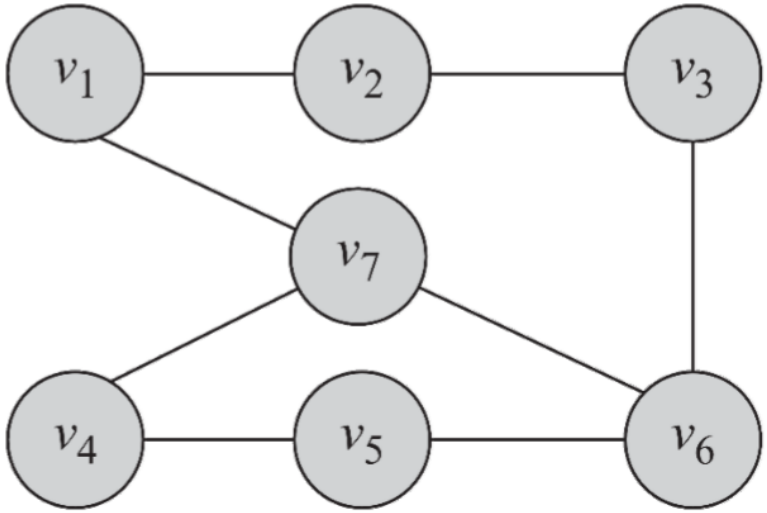
Arêtes (non-)dirigées

- une arête est une paire de sommets : $e = (v_1, v_2)$
- dans un graphe non orienté, $(v_1, v_2) = (v_2, v_1)$
- dans un graphe orienté, on parle d'arc

graphe orienté



graphe non orienté

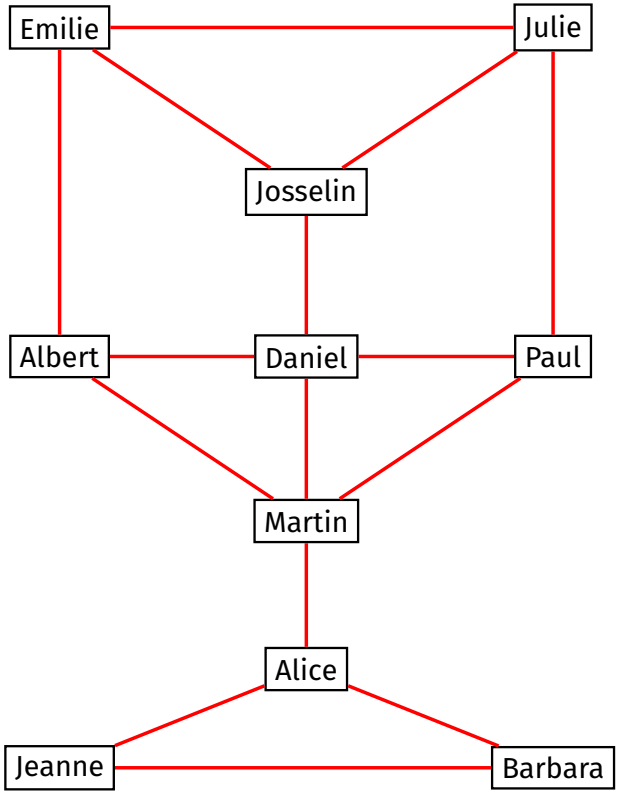


Degré, voisinage

- Pour un nœud v donné, dans un graphe non orienté, l'ensemble des nœuds auxquels il est connecté (relié) par une arête est appelé son voisinage. On le note $N(v)$, de l'anglais neighborhood.
- dans un graphe orienté, on parle des voisins entrants $N_{in}(v)$ et sortants $N_{out}(v)$
- le nombre de sommets connectés à un nœud est le degré de ce nœud : $d(v) = |N(v)|$. On parle de degré entrant et de degré sortant, dans un graphe orienté.
- on note parfois les degrés d, d_{in} et d_{out}

Distribution de degré

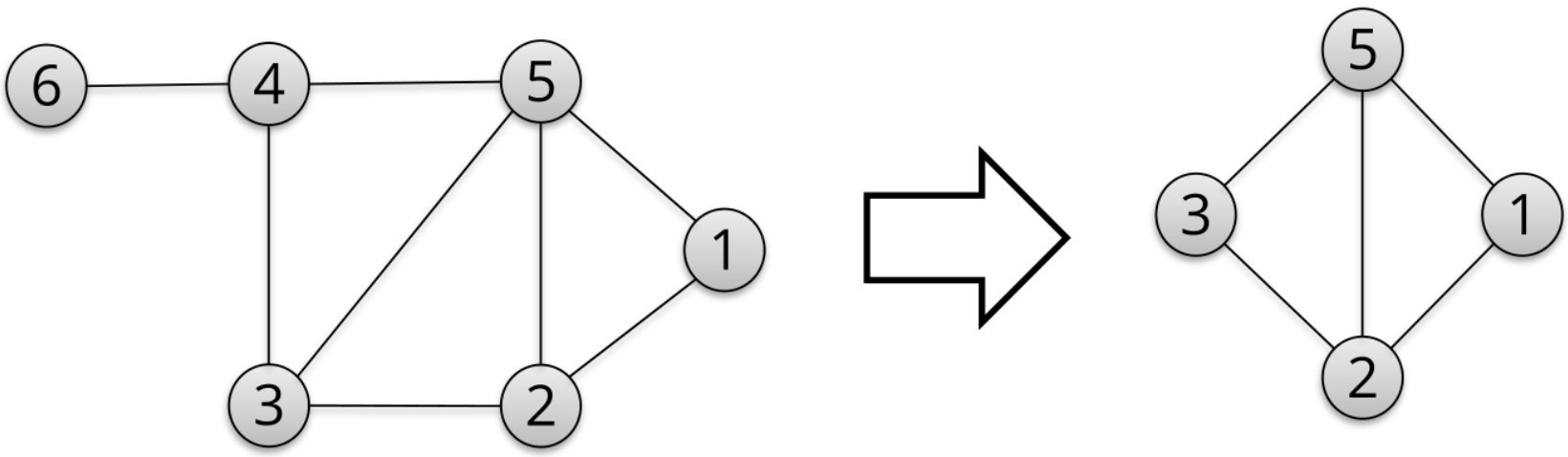
- La distribution des degrés est un concept important pour la caractérisation d'un graphe, surtout quand il est de grande taille
- $\pi(d) = \{d_1, d_2, ..., d_n\}$
  $p_d = \frac{n_d}{n}$, où n_d est le nombre de nœuds avec degré d et $\sum_{d=0}^{\infty} p_d = 1$



- $p_1 = 0$
- $p_2 = \frac{2}{9}$
- $p_3 = \frac{5}{9}$
- $p_4 = \frac{2}{9}$

Sous-graphe

- Un graphe $G' = (V', E')$ est un sous-graphe de $G = (V, E)$ si :
  - $V' \subseteq V$
  - $E' \subseteq (V' \times V') \cap E$



Représentation des graphes

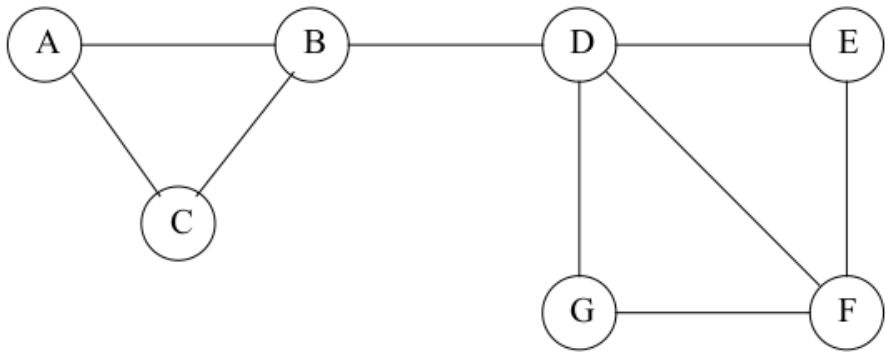
Il y a plusieurs représentations classiques possibles pour les graphes :

- liste d'adjacence (listes chaînées ou tableaux de longueurs variables)
  - efficace pour énumérer les successeurs d'un nœud, beaucoup moins les prédecesseurs
- matrice d'adjacence
  - attention au coût pour des graphes "creux"
  - représentations optimisées dans les langages/frameworks modernes
- liste d'arêtes

On cherche des représentations sans perte d'information, facilement manipulables par des machines, sur lesquelles on peut efficacement faire des calculs.

Matrices de graphes

- Matrice d'adjacence : $A_{ij} = 1$ si l'arête (v_i, v_j) existe, 0 sinon. Sur la diagonale, les boucles.



$$\begin{array}{c} \\ A \\ B \end{array} \begin{array}{c} \begin{array}{ccccccc} A & B & & & & & \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

  Celle-ci peut être très creuse !
- Matrice des degrés : $m_{ij} = d(v_j)$ pour $i = j$, 0 sinon.

$$\begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

- Matrices de Laplace (ou "laplacien(ne)") : L = D - A (attention : nombreuses variantes)

Liste d'adjacence, d'arêtes

Adjacence

- Pour chaque nœud, on conserve la liste des noœuds auxquels il est connecté
- La liste est généralement triée par ordre de (identifiants de) nœud

Liste d'arêtes

- Chaque élément de la liste est une arête (u, v)

Types de graphes

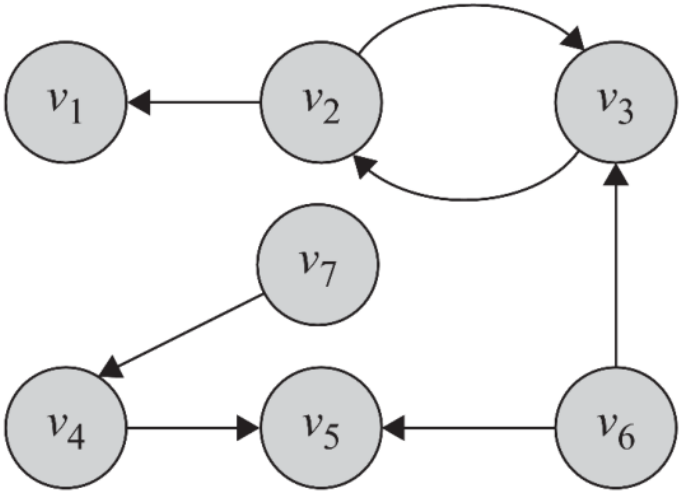
Graphe vide

- Un graphe nul est un graphe dans lequel l'ensemble des sommets est vide.
- N'ayant pas de sommet, le graphe n'a pas d'arête

$$G(V, E), V = E = \emptyset$$

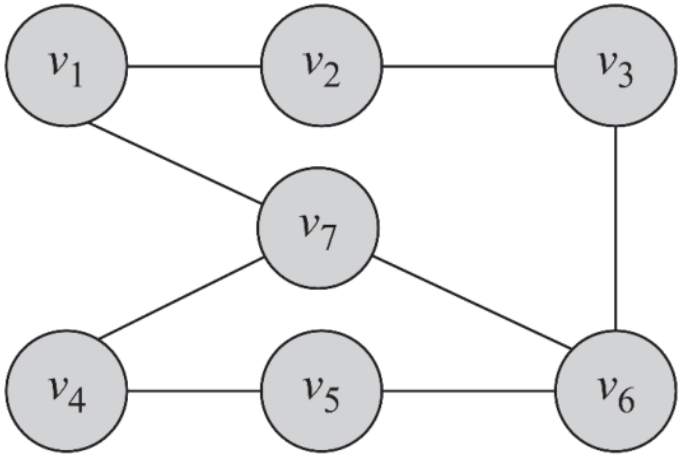
- Un graphe vide est un graphe sans arête. Seul l'ensemble E est vide.
- Un graphe nul est un graphe vide.

Graphes orientés



- la matrice d'adjacence d'un graphe orienté est en général non symétrique :

$$A \neq A^T$$

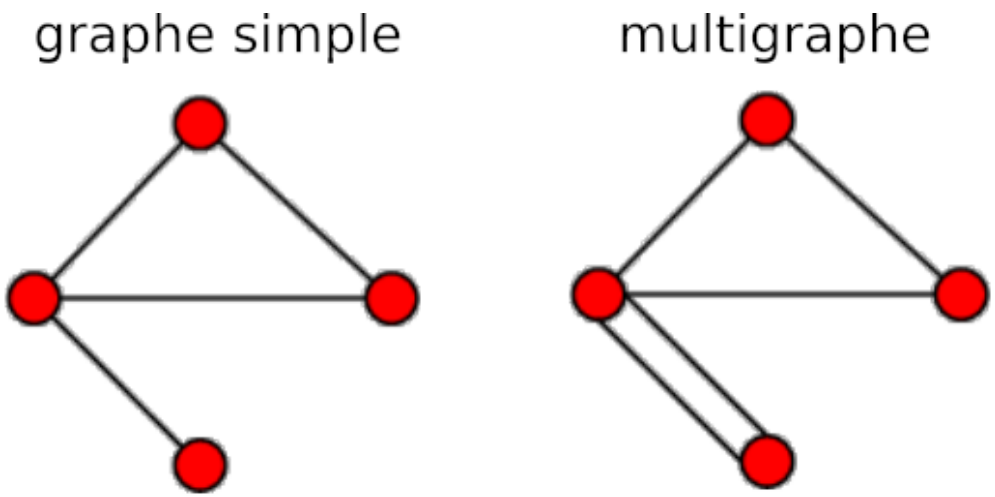


- la matrice d'adjacence d'un graphe non orienté est symétrique :

$$A = A^T$$

Graphes simples, multigraphes

- Les graphes simples sont des graphes tels qu'il existe au plus une arête entre deux sommets
- les multigraphes sont les graphes où il est possible d'avoir plusieurs arêtes entre 2 sommets
- la matrice d'adjacence de multigraphes peut comporter des valeurs plus grandes que 1, pour indiquer la présence de plusieurs arêtes



Graphes valués, signés

Graphe valué

- Un graphe est valué lorsque ses arêtes sont associées à des poids
- Par exemple : graphe de transport routier, où les poids sont les distances en kilomètres.

$$A_{ij} = \begin{cases} w_{ij}, & w_{ij} \in \mathbb{R} \\ 0, & \text{pas de lien entre } v_i \text{ et } v_j. \end{cases} \quad (1)$$

Graphe signé

- Un graphe est signé si les poids sont binaires (0-1, -1/1, +/-)
- On l'utilise pour représenter des amitiés/inimitiés

Connexité

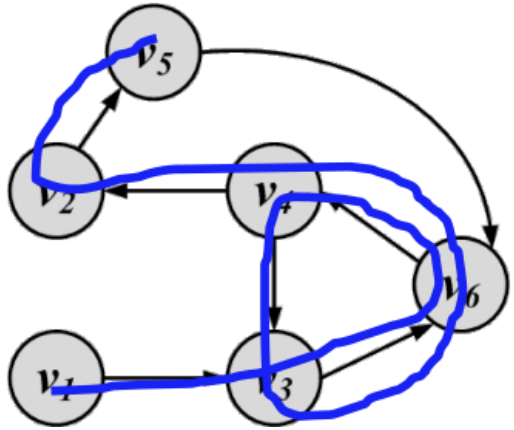
Adjacence, incidence

- Deux nœuds sont adjacents s'ils sont reliés par une arête
- Deux arêtes sont dites incidentes si elles partagent une extrémité
- Dans un graphe orienté, il faut que les directions d'arêtes soient concordantes pour qu'on parle d'arêtes incidentes
- On traverse une arête quand on peut partir de l'une de ses extrémités, se déplacer le long de celle-ci et s'arrêter à l'autre extrémité

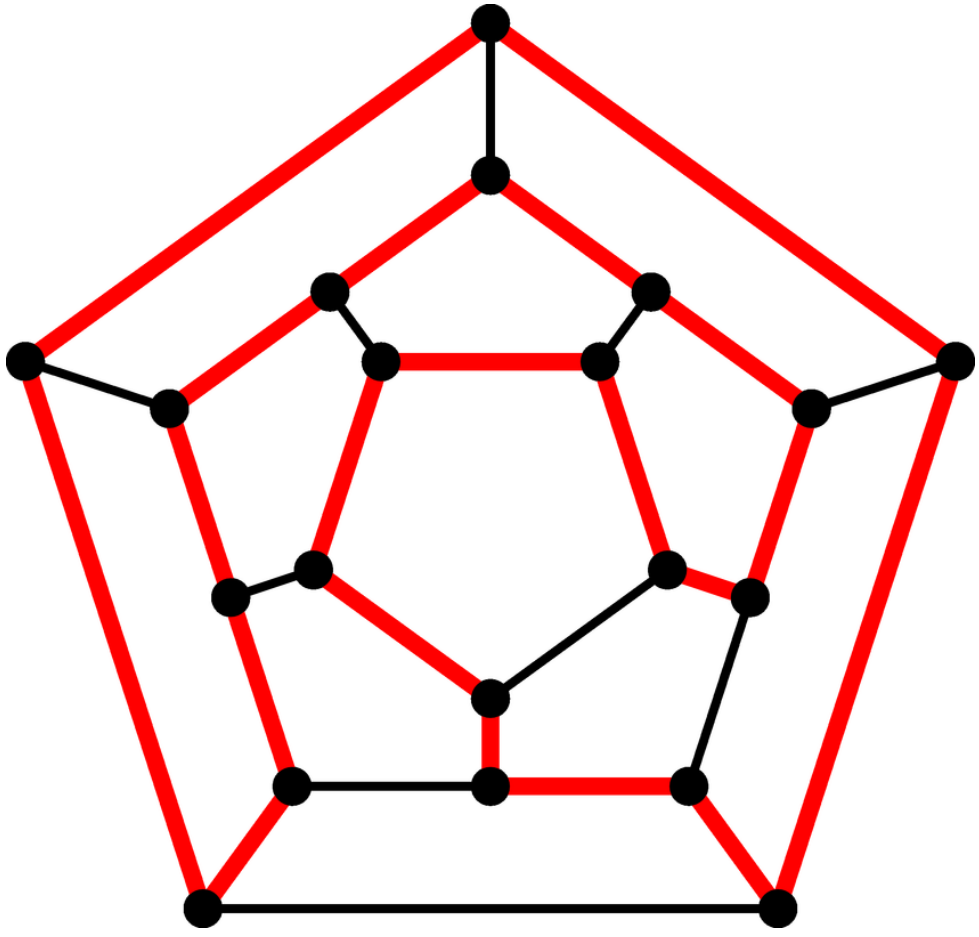
Chaîne

- Une chaîne est une suite d'arêtes incidentes traversées l'une après l'autre.
- Une chaîne peut être fermée (si elle se termine là où elle avait commencé, c'est un *cycle*) ou ouverte (dans le cas contraire)
- Une chaîne peut être représentée :
  - comme une suite de nœuds : $v_1, v_2, \ldots, v_n$
  - comme une suite d'arêtes : $e_1, e_2, \ldots, e_n$
- la longueur d'une chaîne est le nombre d'arêtes traversées.
- une chaîne élémentaire ne passe pas 2 fois par le même sommet
- une chaîne simple ne passe jamais 2 fois par la même arête

- chaîne de longueur 8
- chaîne = walk
- chemin : pour les graphes orientés



Exemples



Cycle hamiltonien, qui passe une fois et une seule par tous les sommets du graphe

Marche aléatoire

- Une marche aléatoire est une chaîne où le nœud suivant est choisi aléatoirement parmi les voisins du nœud courant.
- le poids d'une arête peut être utilisé pour définir la probabilité de l'emprunter
- pour toutes les arêtes qui commencent sur le nœud v_i, on a :

$$\sum_x w_{i,x} = 1, \qquad \forall i, j, w_{i,j} \geq 0$$

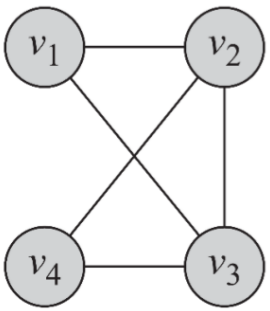
- Chaîne de Markov, PageRank

Connexité

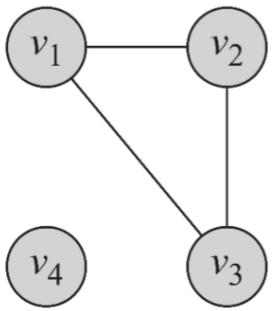
- On dit qu'un nœud v_i est connecté à un nœud v_j si :
  - v_i et v_j sont adjacents
  - il existe un chemin de v_i à v_j
- un graphe est connexe s'il existe un chemin pour toute paire de sommets
- dans un graphe orienté :
  - le graphe est fortement connexe s'il existe un chemin direct pour chaque paire de sommets
  - le graphe est faiblement connexe s'il existe un chemin pour chaque paire de sommets, sans tenir compte des orientations d'arêtes

Connexité : exemple

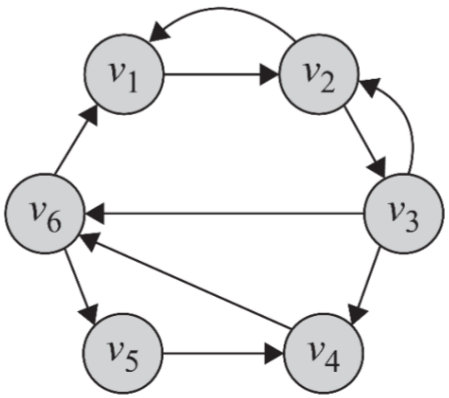
graphe connexe



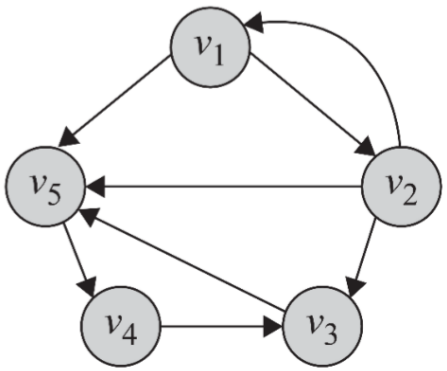
graphe non connexe



graphe fortement connexe

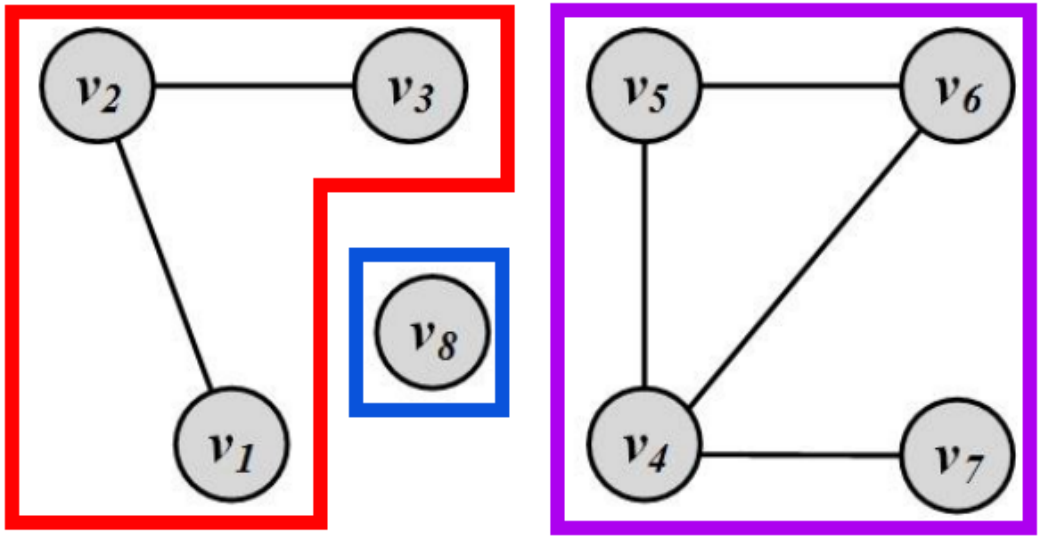


graphe faiblement connexe



Composante connexe

- Une composante connexe est un sous-graphe connexe, c'est-à-dire un sous-ensemble de sommets tel qu'il existe un chemin entre toute paire de sommets.
- dans un graphe orienté, on a :
  - des composantes fortement connexes
  - des composantes faiblement connexes
- 3 composantes connexes :



Plus court chemin

- le plus court chemin entre deux sommets est le chemin qui a la longueur la plus courte. On note cette longueur l.
- on peut généraliser le concept de voisinage à l'aide des plus courts chemins : un voisinage à distance k est l'ensemble des sommets qui sont à une distance inférieure à k d'un nœud donné.
- le diamètre d'un graphe est le plus long des plus courts chemins entre toutes les paires de sommets du graphe :

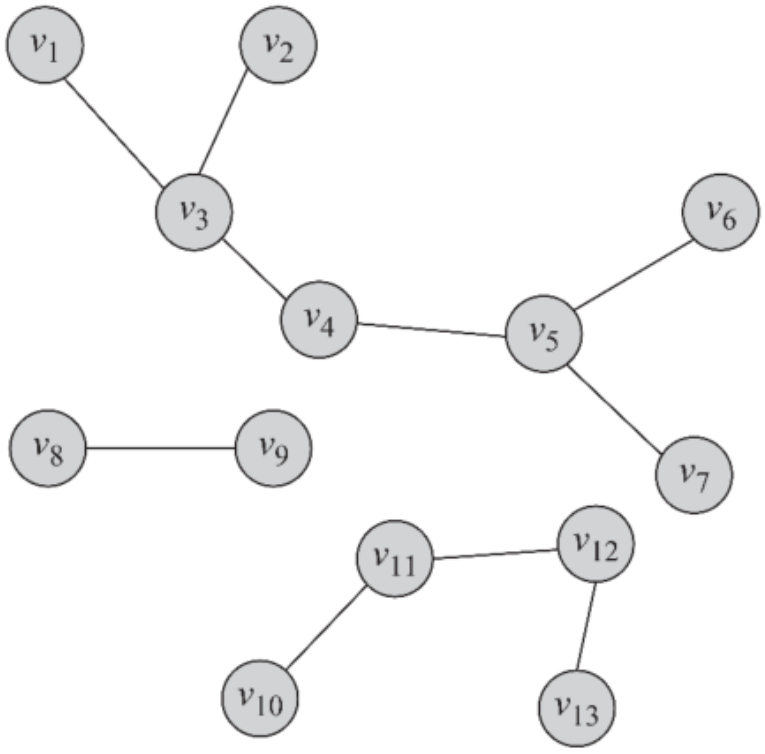
$$\text{diam}_G = \max_{(u,v)\in V\times V} l_{uv}$$

Graphes remarquables

Arbres et forêts

- Un arbre est un graphe acyclique connexe
- Dans un arbre il y a exactement un chemin pour chaque paire de sommets
- $|V| = |E| + 1$

- Un ensemble d'arbres (déconnectés) est appelé une forêt

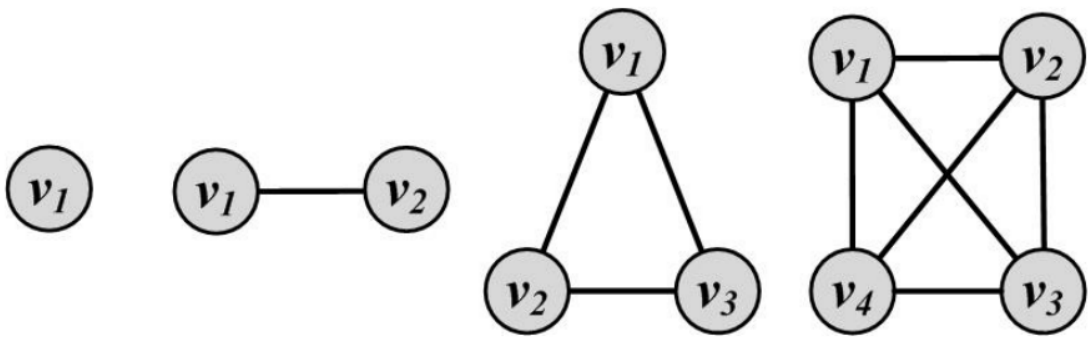


Arbre couvrant

- Pour un graphe connexe, l'arbre couvrant est un sous-graphe du graphe et un arbre incluant tous les nœuds du graphe
- Il peut exister plusieurs arbres couvrants pour un graphe
- Dans un graphe valué, le poids de l'arbre couvrant est le poids des arêtes contenues dans l'arbre
- il existe un arbre couvrant de poids minimal. Si on considère un réseau où un ensemble d'objets doivent être reliés entre eux (réseau électrique et habitations), cet arbre est la façon de construire un tel réseau en minimisant un coût représenté par le poids des arêtes (ex.: longueur totale de câble utilisée pour construire un réseau électrique)
- il existe de nombreux algorithmes pour construire un tel arbre (hors cours)

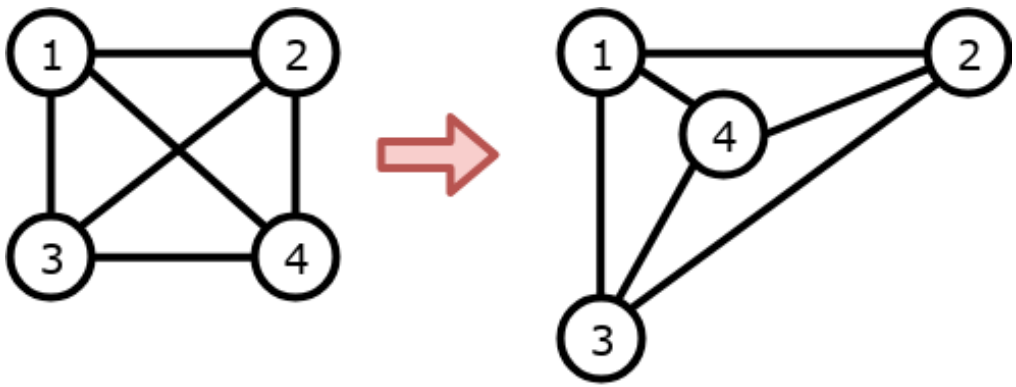
Graphe complet

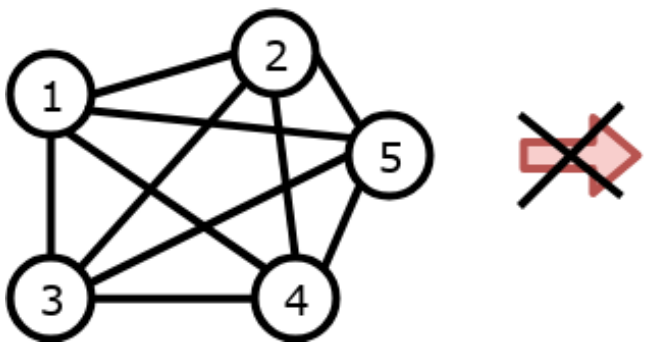
- Un graphe complet est un graphe dans lequel toutes les arêtes possibles existent
- $|E| = \binom{|V|}{2} = \frac{n(n-1)}{2}$



Planarité

- Un graphe est dit planaire s'il peut être dessiné sans chevauchement d'arêtes.
- Un graphe est non-planaire si **toutes** ses représentations ont **au moins un** chevauchement d'arête

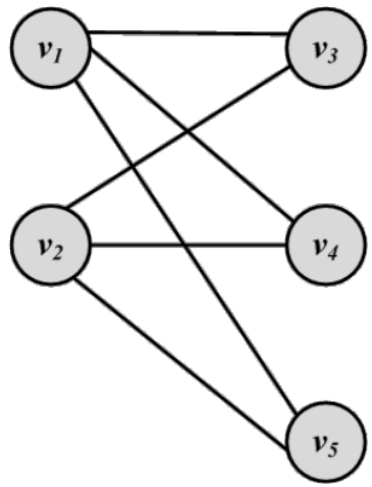




Graphe biparti

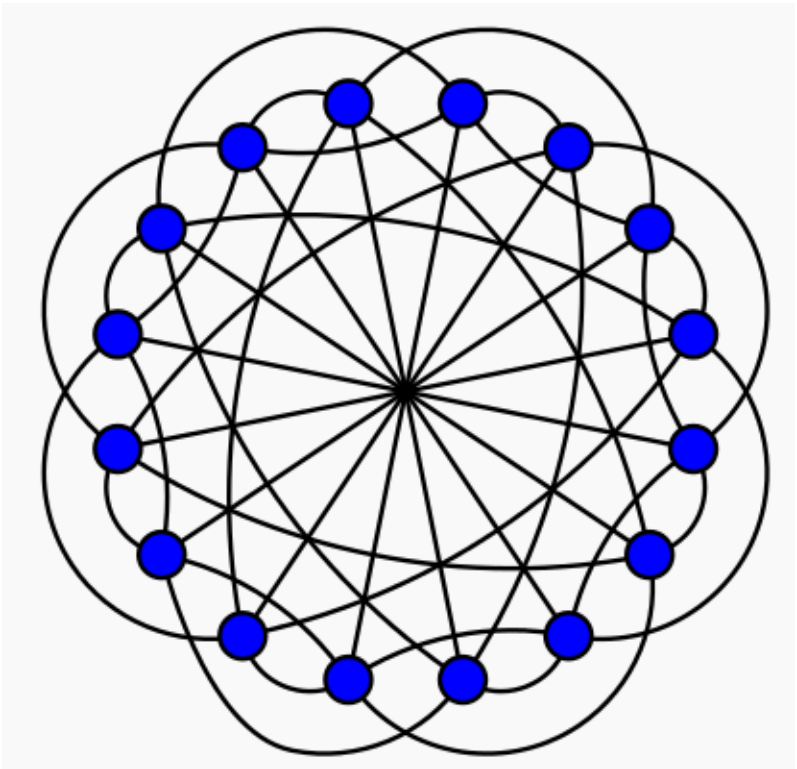
- Un graphe est dit biparti si l'on peut partitionner l'ensemble V de ses sommets en 2 ensembles tels que, pour toutes les arêtes, une extrémité soit dans l'un des ensembles et l'autre extrémité dans l'autre ensemble.

$$\begin{cases} V = V_L \cup V_R, \\ V_L \cap V_R = \emptyset, \\ E \subset V_L \times V_R \end{cases}$$



Graphe régulier

- Dans un graphe régulier, tous les sommets ont le même degré
- Le graphe peut être connexe ou non connexe
- dans un graphe k-régulier, tous les nœuds ont pour degré k
- les graphes complets sont des exempls de graphes réguliers
- le graphe de Clebsch est 5-régulier :



Algorithmes

Besoin d'algorithmes spécifiques

- Gros problème = taille :
  - Internet = Millions de sommets (routeurs)
  - Facebook = 2 milliards d'utilisateurs actifs
  - Web = Google indexe plus de 40 milliards de pages
- il est non trivial de
  - stocker le graphe en mémoire
  - faire des calculs sur le graphe

Exemples

- Compter les triangles d'un graphe (clustering) :
  - naïvement O(n*n*n)
  - $O(m * n^{(1/a)})$ si distribution des degrés en loi de puissance d'exposant a
- Diamètre :
  - complexité théorique : O(nm)
  - approximation en O(m)
- Problèmes NP-complets
- Beaucoup de problèmes spécifiques aux graphes réels (détection de communautés). Approximation (non prouvée) linéaire.

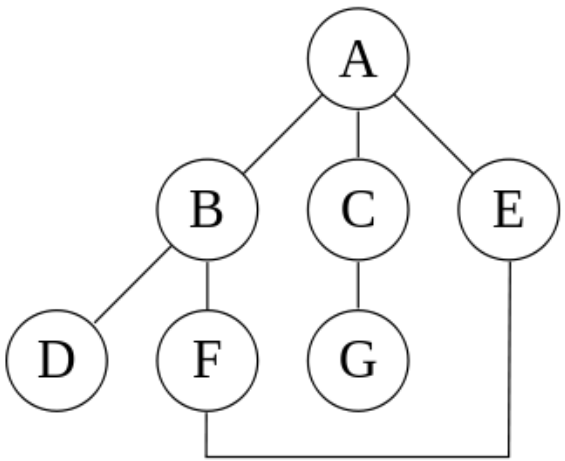
Parcours de graphes

- On cherche une stratégie pour fouiller un réseau social et calculer l'âge moyen des utilisateurs
- On part d'une personne
- On veut ensuite atteindre ses amis, puis les amis de ces amis, puis ...
- On doit garantir que :
  - tous les utilisateurs sont visités
  - chaque utilisateur n'est pas visité plus d'une fois

- Deux grandes techniques
  - le parcours en profondeur (DFS)
  - le parcours en largeur (BFS)

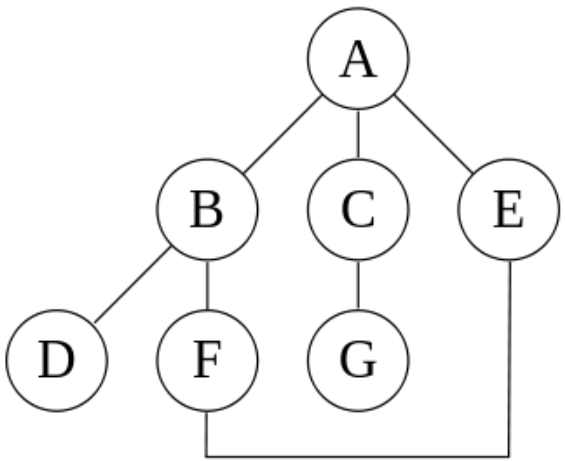
Parcours en profondeur

- En anglais, DFS, pour Depth First Search
- progresse à partir d'un sommet S en s'appelant récursivement pour chaque sommet voisin de S.
- pour chaque sommet, on prend le premier sommet voisin, on explore tous ses voisins (non marqués) avant de revenir au "père"
- Ordre de visite : A, B, D, F, E, C, G
- s'implémente avec une pile (LIFO)



Parcours en largeur

- En anglais, BFS, pour Breadth First Search
- pour chaque sommet, on repère tous ses voisins, on stocke ceux qui ne sont pas marqués dans une file (queue)
- Ordre de visite : A, B, C, E, D, F, G
- s'implémente avec une file (FIFO)
- on obtient les plus courts chemins à la racine



Plus courts chemins

- Dans un graphe connexe, il est fréquent que plusieurs chemins existent pour relier deux nœuds pris au hasard
- Dans de nombreux cas, on a besoin du plus court chemin

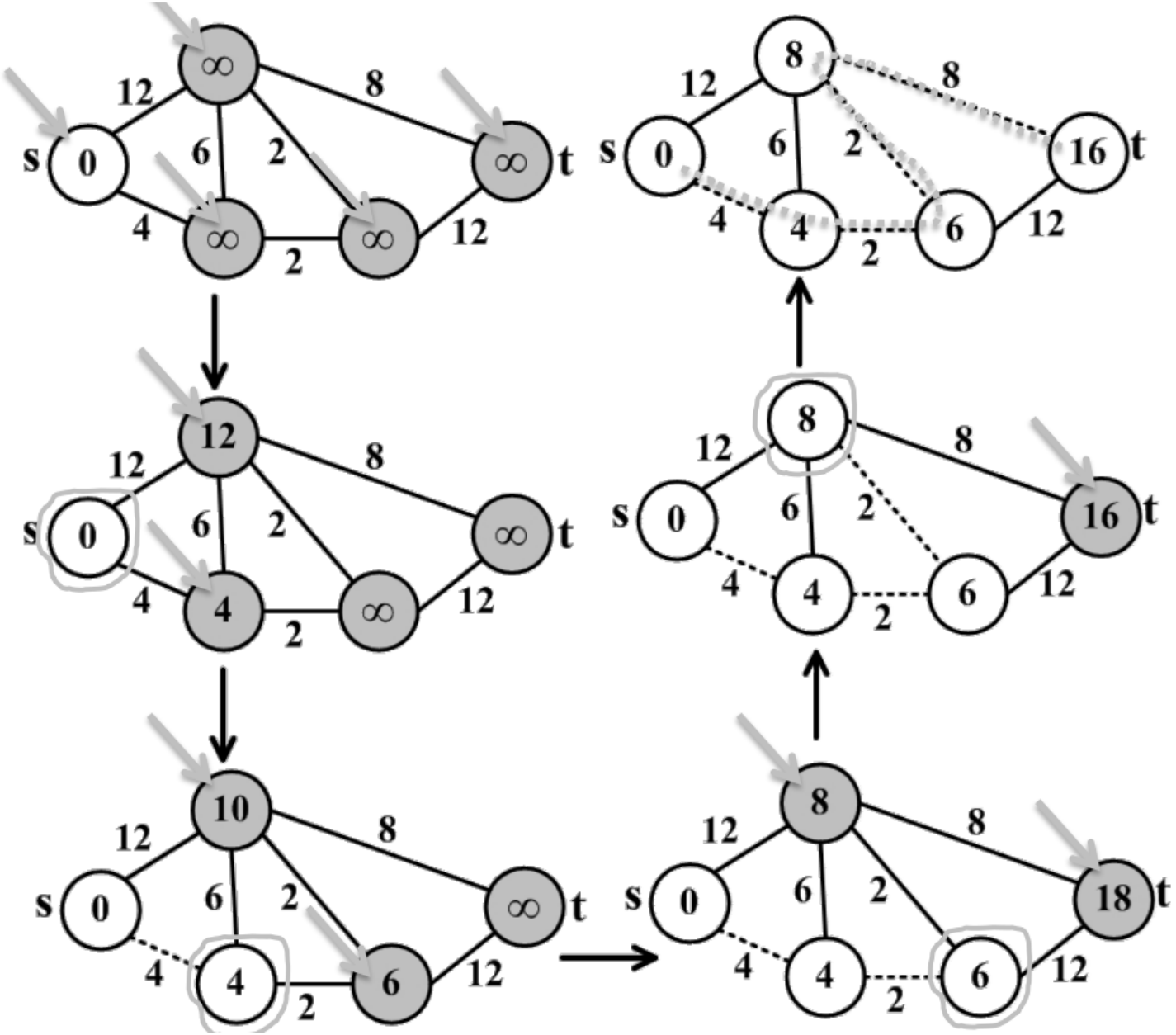
L'algorithme de Dijkstra (Edsger Dijkstra, 1930–2002) :

- conçu pour les graphes valués (sans arêtes négatives)
- trouve les plus courts chemins d'un nœud source s vers tous les autres sommets
- trouve les plus courts chemins et leurs longueur

Algorithme de Dijkstra

1. Initialisation
   - On assigne la valeur 0 à la source, l'infini à tous les autres
   - On marque tous les nœuds comme non-visités
   - Le nœud source est marqué courant
2. Pour le nœud courant, on examine les voisins non-visités et on calcule leurs distances "possibles" (distance actuelle + poids de l'arête)
   - Si la distance possible est plus petite que celle du voisin, alors la distance du voisin = distance possible
3. Après avoir examiné tous les voisins du nœud courant, on marque le nœud courant comme visité et on l'enlève de l'ensemble des non-visités
4. Si le nœud destination a été marqué comme visité ou si la plus petite distance possible dans les nœud de l'ensemble non-visités est l'infini, alors on s'arrête.
5. On marque le nœud non visité avec la plus petite distance possible comme le prochain nœud courant, et l'on repart à l'étape 2

Exemple



Algorithme de Dijkstra : remarques

- L'algorithme dépend de la source utilisée
- Donc pour trouver tous les plus courts chemins pour toutes les paires:
  - on peut exécuter l'algorithme n fois
  - ou utiliser d'autres algorithmes comme celui de Floyd-Warshall
- Si l'on a seulement besoin du calcul du plus court chemin entre s et un nœud destination d, on peut arrêter l'algorithme une fois que le plus court chemin vers d a été trouvé

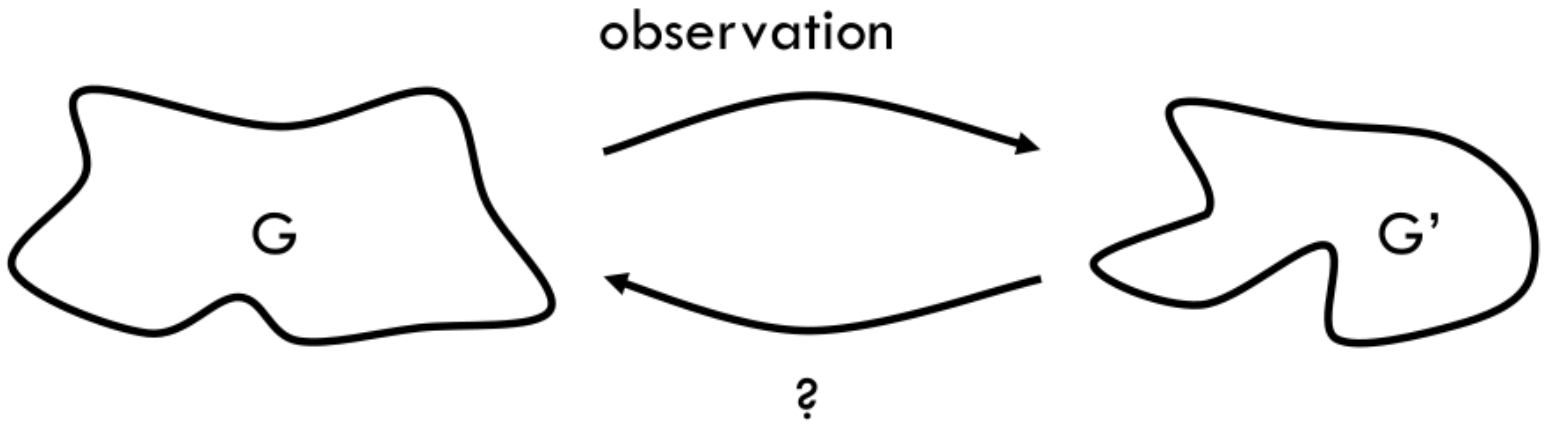
Mesure : collecte de données

Métrologie

- En général : impossibilité d'étudier l'objet réel, seulement une mesure
- Questions :
  - qui a fait la mesure ?
  - quelle proportion a été mesurée ?
  - combien de temps la mesure a-t-elle duré ?
  - quelles étaient les contraintes / biais ?
  - la mesure peut-elle être reproduite ?

Métrologie

- Étude du biais introduit par l'observation
- Que dire de l'objet réel à partir de l'observation ?
- Nouveaux protocoles de mesures, etc.



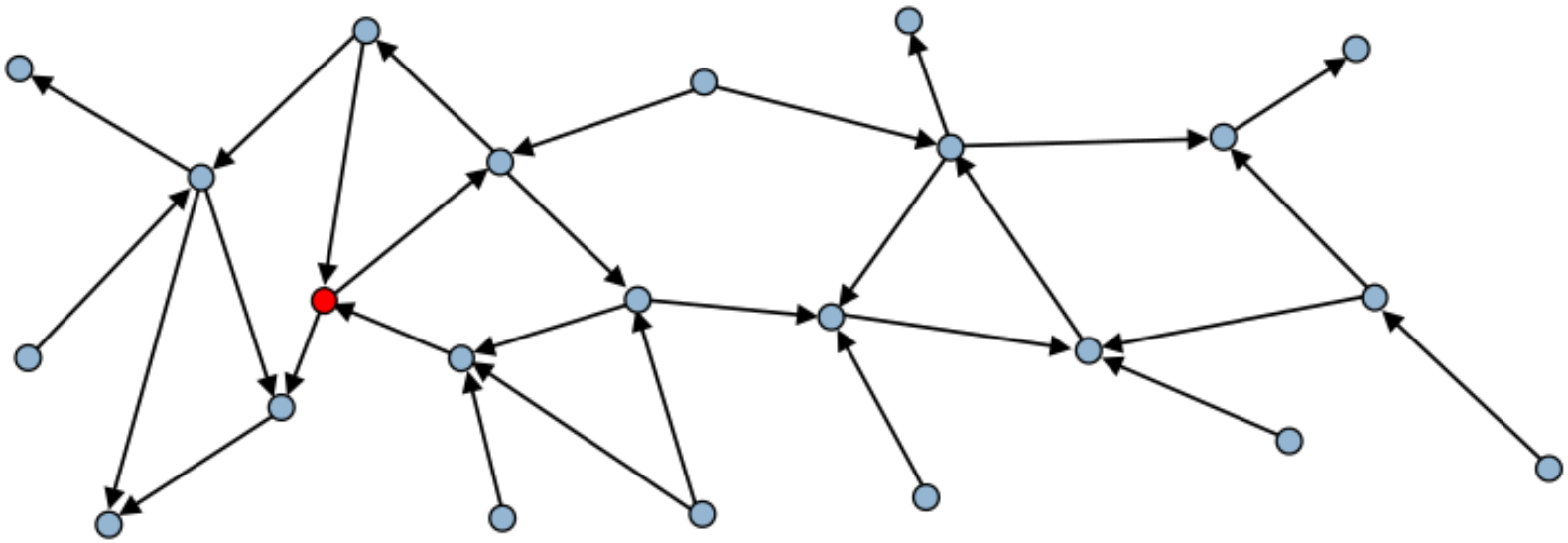
- Évaluer la représentativité des "cartes"

Une approche

- On simule la mesure sur un graphe aléatoire
- Modélisation du processus de mesure :
  - Internet : traceroute = chemins courts
  - Web : crawl = parcours en largeur
- Modélisation du réseau :
  - Graphes aléatoires
  - Respect des degrés, du clustering, etc.

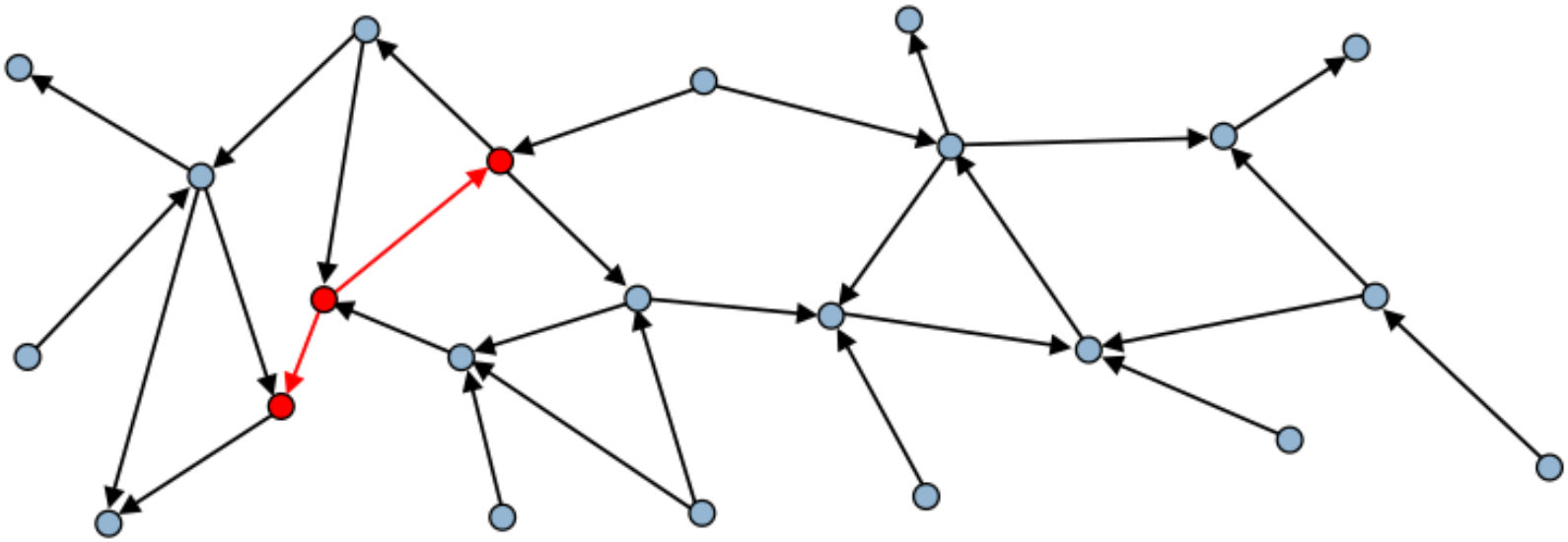
Mesure de l'Internet

- Processus de mesure par parcours en largeur depuis plusieurs sources
- Réseau : orienté, non connexe, dynamique



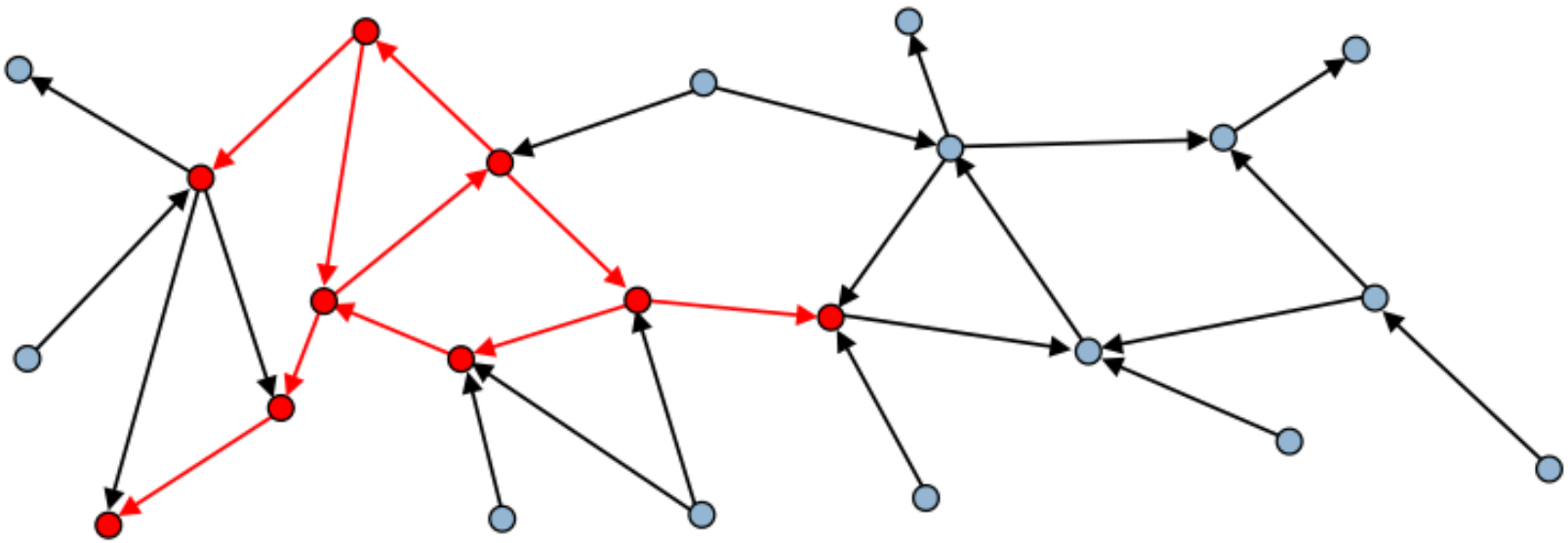
Mesure de l'Internet

- Processus de mesure par parcours en largeur depuis plusieurs sources
- Réseau : orienté, non connexe, dynamique



Mesure de l'Internet

- Processus de mesure par parcours en largeur depuis plusieurs sources
- Réseau : orienté, non connexe, dynamique

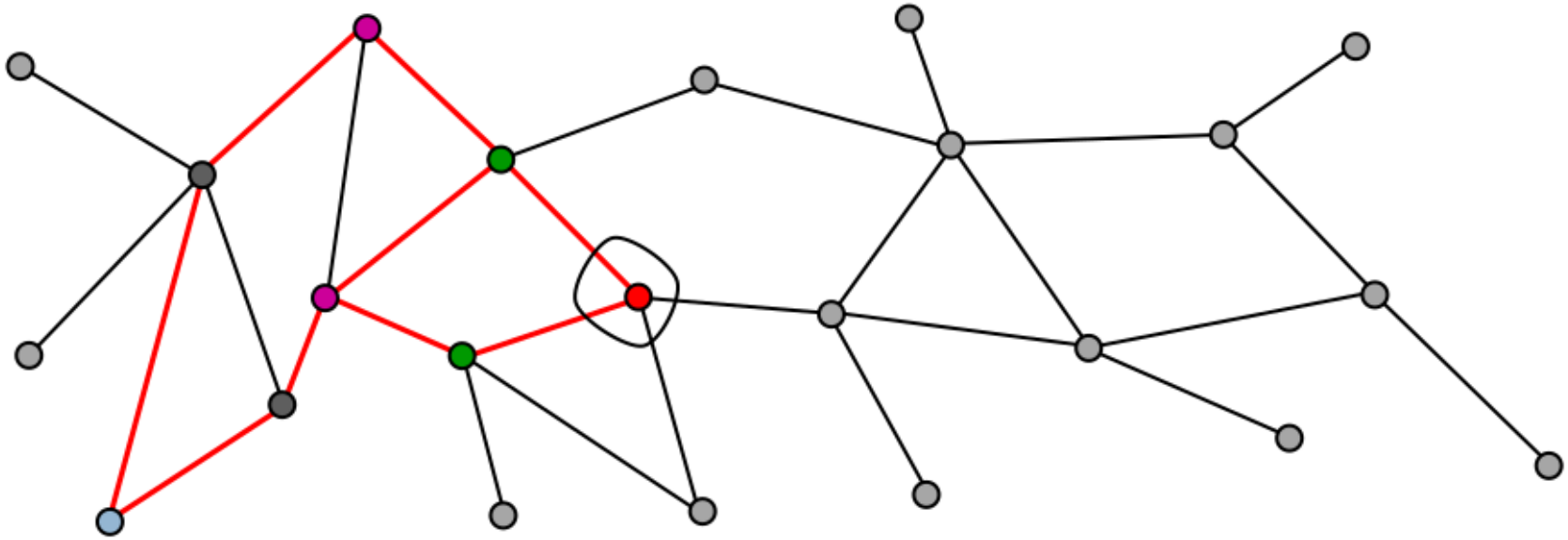


Questions

- Influence sur le résultat de :
  - Nombre de sources et destinations
  - Propriétés du réseau
  - Localisation des sources et destinations
- Modélisation :
  - Traceroute = plus courts chemins (un ou tous)
  - Graphe = graphe aléatoire (modèle à choisir)

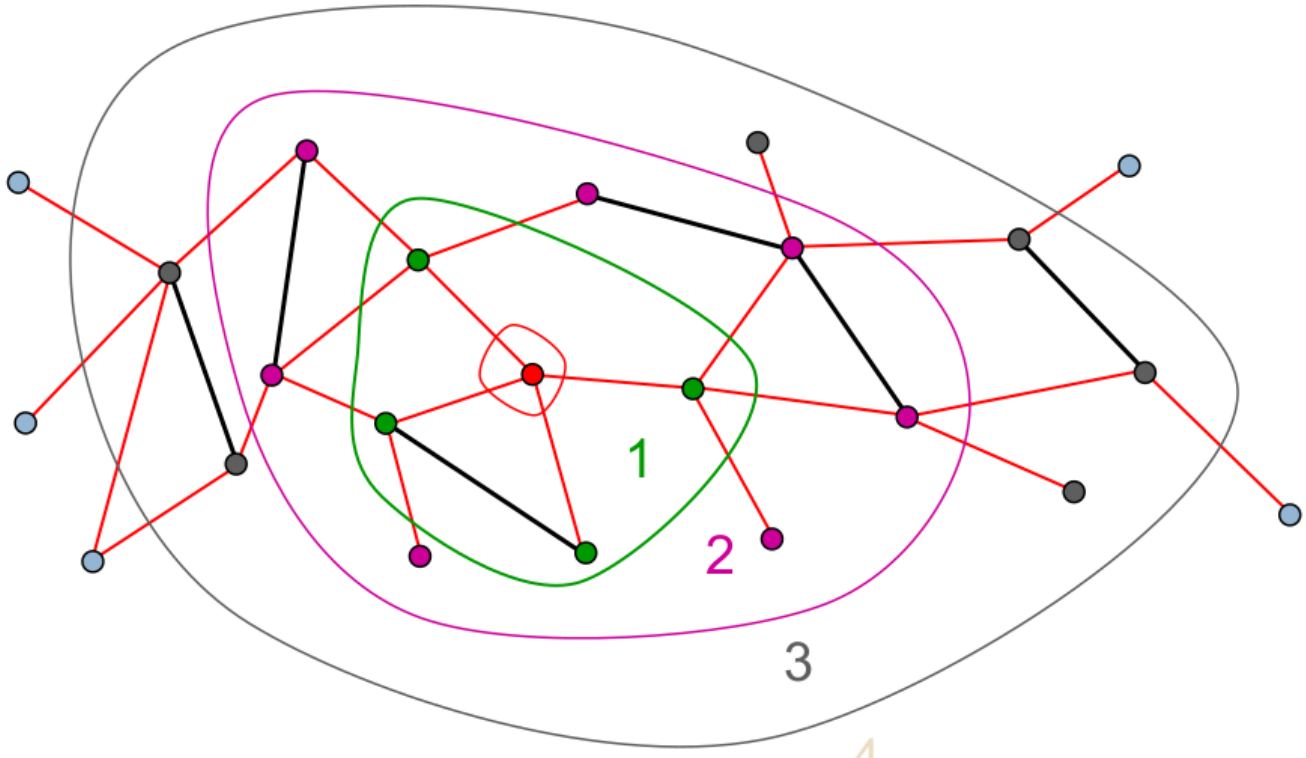
Que voit-on ?

- D'une source vers tout le monde
  - liens rouges découverts (sur plus courts chemins)
  - on répète pour les autres destinations
  - liens noirs invisibles



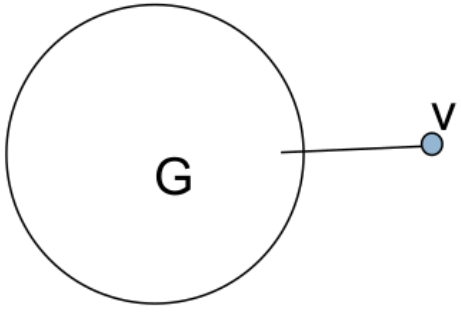
Que voit-on ?

- D'une source vers tout le monde
  - liens rouges découverts (sur plus courts chemins)
  - on répète pour les autres destinations
  - liens noirs invisibles

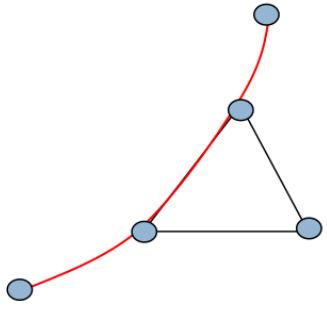


Zones dures à mesurer

- Sommet de degré 1 : uniquement visible si source ou destination

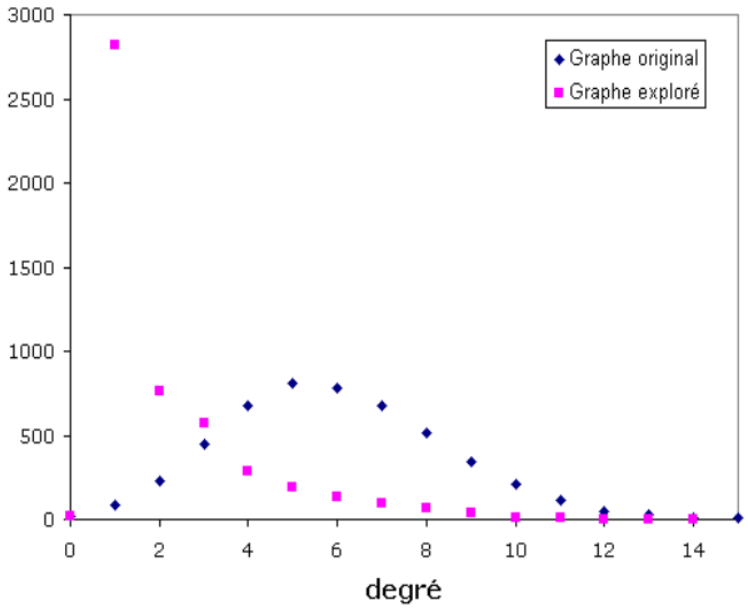


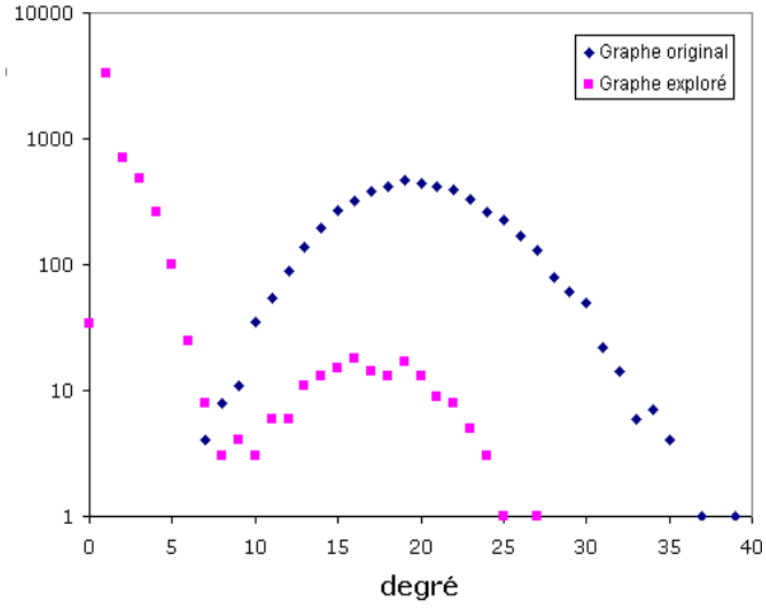
- graphe complet : visiter tous les liens



Distribution de degrés

- différences entre original et mesuré
  - beaucoup de sommets de faible degré
  - peu de sommets de fort degré
- mauvaise estimation de la propriété réelle





Outils

Démarche sur un problème réel

- définir les sommets (page web, compte FB/TW, etc.)
- définir les arêtes (appels 5x/mois, 2x/mois ?)
- identifier les hubs (traiter séparément, enlever)
- identifier les mesures pertinentes (PageRank, degree, triangle, LCC)
- construire la source de données
- écrire le code de traitement
- analyser
- recommencer !

Frameworks et bibliothèques

- Pregel : Google, 2010. Passage de messages entre nœuds. Diverses implémentations sur Hadoop.
- GraphLab : projet de CMU, 2009. GraphX dans Spark
- Gelly (dans Flink)

- Neo4j : base de données “graphe” open source.
- FlockDB (Twitter), AllegroGraph, GraphDB, ...

Logiciels

- Gephi
- Linkurious
- SocNetV
- Pajek
- Tulip
- Guess

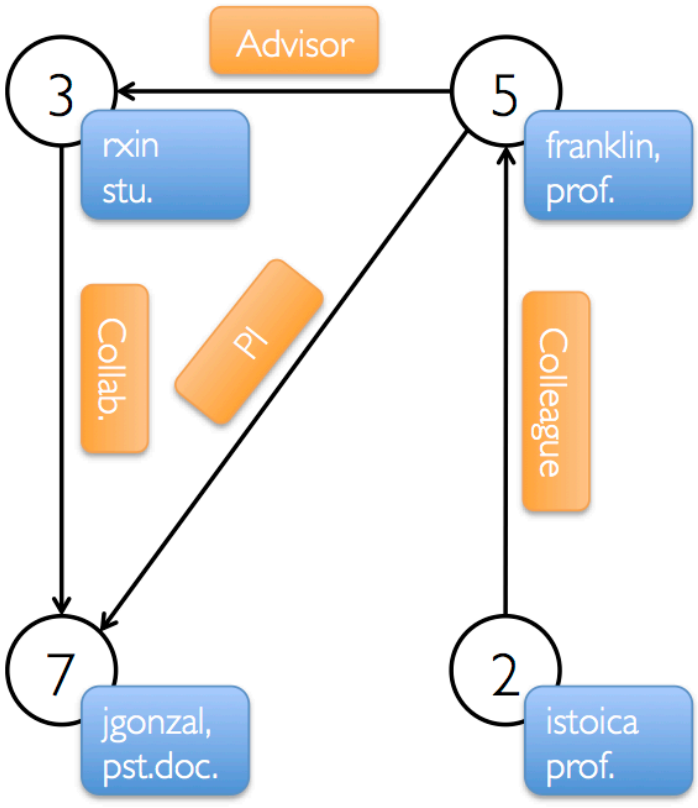
- Awesome Network Analysis https://github.com/briatte/awesome-network-analysis
- Network Repository http://networkrepository.com/index.php

GraphX

- librairie de Spark pour gérer les calculs sur les graphes
- en particulier, le parallèlisme
- introduit une abstraction Graph (au-dessus de RDD) :
  - un multigraphe orienté, avec des propriétés attachées à chaque sommet et chaque arête
  - facilite les cas où il y a plusieurs arêtes entre des noeuds
- https://spark.apache.org/docs/latest/graphx-programming-guide.html

GraphX

Property Graph



Vertex Table

Id	Property (V)
3	(rxin, student)
7	(jgonzal, postdoc)
5	(franklin, professor)
2	(istoica, professor)

Edge Table

SrcId	DstId	Property (E)
3	7	Collaborator
5	3	Advisor
2	5	Colleague
5	7	PI

GraphX

```scala
val sc: SparkContext
// Create an RDD for the vertices
val users: RDD[(VertexId, (String, String))] =
  sc.parallelize(Array((3L, ("rxin", "student")), (7L, ("jgonzal", "postdoc")),
                       (5L, ("franklin", "prof")), (2L, ("istoica", "prof"))))
// Create an RDD for edges
val relationships: RDD[Edge[String]] =
  sc.parallelize(Array(Edge(3L, 7L, "collab"),    Edge(5L, 3L, "advisor"),
                       Edge(2L, 5L, "colleague"), Edge(5L, 7L, "pi")))
// Define a default user in case there are relationship with missing user
val defaultUser = ("John Doe", "Missing")
// Build the initial Graph
val graph = Graph(users, relationships, defaultUser)
```

GraphX : opérateurs

```scala
val graph: Graph[(String, String), String]
// Use the implicit GraphOps.inDegrees operator
val inDegrees: VertexRDD[Int] = graph.inDegrees
```

D'autres opérateurs :

- numEdges/numVertices
- collectNeighbors
- subgraph
- connectedComponents
- triangleCount

Conclusion

Références

Ce cours repose sur les travaux et documents suivants :

- le livre *Social Media Mining* de R. Zafarani, M. A. Abbasi, and H. Liu. Cambridge Univ. Press, 2014. Livre et slides gratuits disponibles sur `http://socialmediamining.info`
- l'équipe ComplexNetworks du LIP6 (Sorbonne Université, `http://www.complexnetworks.fr`), en particulier les cours de Jean-Loup Guillaume (PR, U. de La Rochelle) et de Clémence Magnien (DR CNRS)
- le livre *Mining Massive datasets* (`http://www.mmds.org`), de Jure Leskovec, Anand Rajaraman, Jeff Ullman