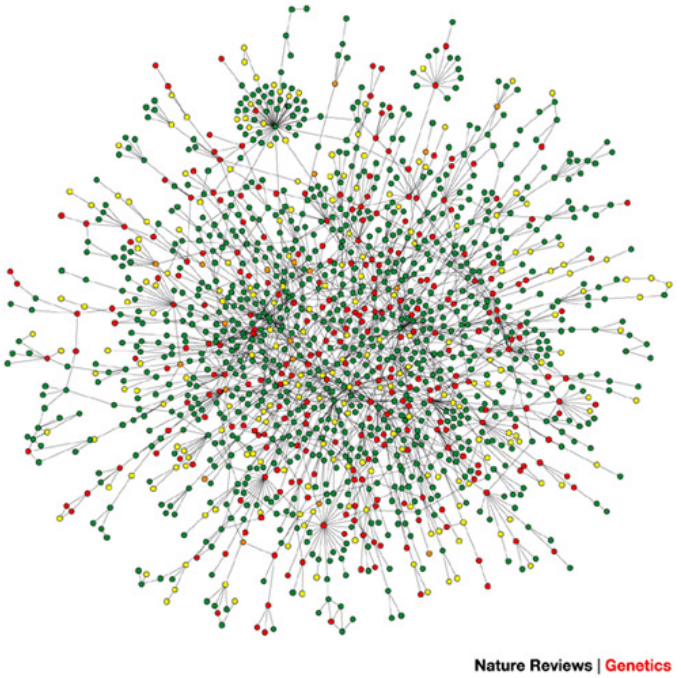
Graphes

Graphes non-orientés

Déf. Un **graphe non-orienté** est representé par un couple (V, E) où $E \subseteq \binom{V}{2}$ (paires non-ordonnées). V est l'ensemble des sommets et E est l'ensemble des arêtes.



Barabási & Oltvai *Nature Rev Genet* 5 :101, 2004

Graphes orientés

Déf. Un **graphe orienté** est representé par un couple (V, E) où $E \subseteq V \times V$ (paires ordonnées). V est l'ensemble des nœuds ou sommets, et E est l'ensemble des arcs.

Terminologie

Graphe non-orienté

L'arête $\{u, v\}$ est dénotée par uv.
Si $uv \in E$, alors v est adjacent à u.
L'arête $uv \in E$ est incidente aux sommets u et v.
Le degré de $u \in V$ est le nombre d'arêtes qui y sont incidentes.

Le graphe non-orienté $G = (V, E)$ avec $E = \binom{V}{2}$ est un **graphe complet**, dénoté par K_n où $n = |V|$.

Graphe orienté

L'arc (u, v) est dénotée par uv : l'arc part de u et arrive à v.
Si $uv \in E$, alors v est adjacent à u.
Le degré sortant de $u \in V$ est le nombre d'arcs qui y partent ; le degré rentrant est le nombre d'arcs qui y arrivent.

Chemins

Un **chemin** de longueur ℓ est une séquence $v_0, v_1, \ldots, v_\ell$ où $v_{i-1}v_i \in E$ pour tout $i = 1, \ldots, \ell$.

($\ell = 0$ est OK : chemin sans arêtes/arcs.)

Si $v_0 = v_\ell$, alors le chemin forme un **cycle**.

(Plus précisement, les sommets initial et final ne sont pas distingués dans le cycle.)

Le chemin $v_0 \cdots v_\ell$ est **élémentaire** ssi $v_1, \ldots, v_\ell$ sont distincts. Si $v_0 = v_\ell$, alors le chemin forme un **cycle élémentaire**.

Un graphe sans cycle est dit **acyclique**.

Un graphe non-orienté est **connexe** si chaque paire de sommets est relié par un chemin.

Un graphe non-orienté connexe acyclique est un **arbre**.

Sous-graphes

Le graphe $G' = (V', E')$ est un **sous-graphe** de $G = (V, E)$ ssi $V' \subseteq V$ et $E' \subseteq E$.

Étant donné un sous-ensemble de sommets $V' \subseteq V$, le sous-graphe de G **engendré** par V' est le graphe $G' = (V', E')$ avec $E' = \left\{uv \in E \colon u, v \in V'\right\}$.

Pondération

Graphe ponderé : chaque arc (ou arête) possède un **poids** ou coût associé, défini par la fonction de pondération $c\colon E \mapsto \mathbb{R}$.

Poids d'un sous-graphe : somme de poids des arcs dans le sous-graphe

Poids d'un chemin : somme de poids des arcs dans le chemin

Questions intéressantes

Stocker les graphes dans le mémoire d'un ordinateur

Parcours d'un graphe

Vérifier si le graphe est connexe

Plus court chemins

Arbres couvrants

Comment stocker le graphe ?

Matrice d'adjacence : matrice $V \times V$, $A[u, v]$ donne le poids de uv ($\pm\infty$ ou NaN pour arcs non-existants), ou valeurs booléennes pour noter juste présence

Listes d'adjacence : liste $\mathsf{Adj}[u]$ pour chaque sommet u qui stocke l'ensemble $\{v : uv \in E\}$ ou l'ensemble des couples $\{\langle v, c(u, v)\rangle : uv \in E\}$.

Usage de mémoire : dépend de la densité du graphe $\frac{|E|}{|V|^2}$.

Déterminer si $uv \in E$ ou $c(u, v)$: rapide avec la matrice mais plus lente avec les listes d'adjacence

Parcours

Technique essentielle : marquage/coloriage «jamais vu», «déjà vu»

```
     Algo PARCOURS-PROFONDEUR(u)
P1   // prévisite de u
P2   marquer u // «déjà vu»
P3   pour tout v adjacent à u
P4       si v n'est pas marqué alors PARCOURS-PROFONDEUR(v)
P5   // post-visite de u
```

(Généralisation du parcours préfixe ou postfixe sur les arbres.)

Parcours en largeur

Utiliser une queue Q

```
     Algo PARCOURS-LARGEUR(s)
L1   Q.enqueue(s)
L2   tandis que Q n'est pas vide
L3       u ← Q.dequeue()
L4       marquer u // «déjà vu»
L5       pour tout v adjacent à u faire
L6           si v n'est pas marqué alors Q.enqueue(v)
```

Temps de calcul : $O(|E|)$ avec listes d'adjacence ou $O(|V|^2)$ avec matrice d'adjacence