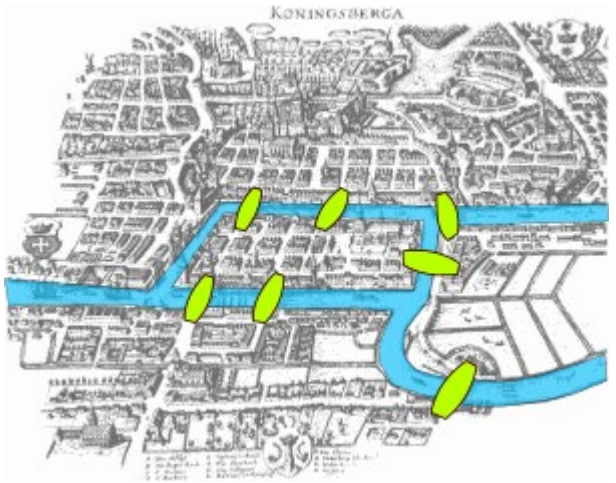
LES GRAPHES

Martine.Leonard@univ-rouen.fr
Master BioInforMatique

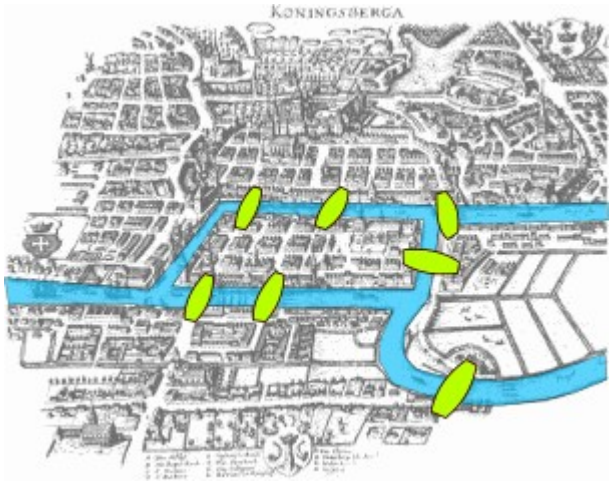
Naissance des graphes

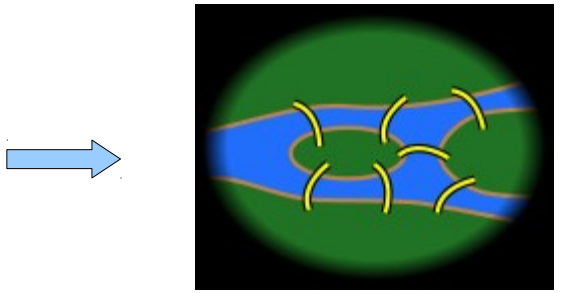


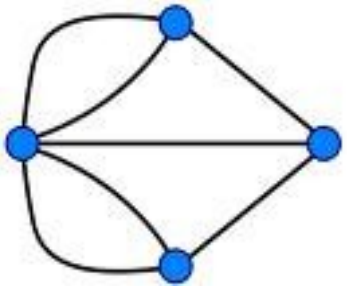
Problème posé par Euler en 1735

« Comment parcourir une fois et une seule les 7 ponts de la ville de Koenigsberg ? »

Naissance de la modélisation à l'aide des graphes







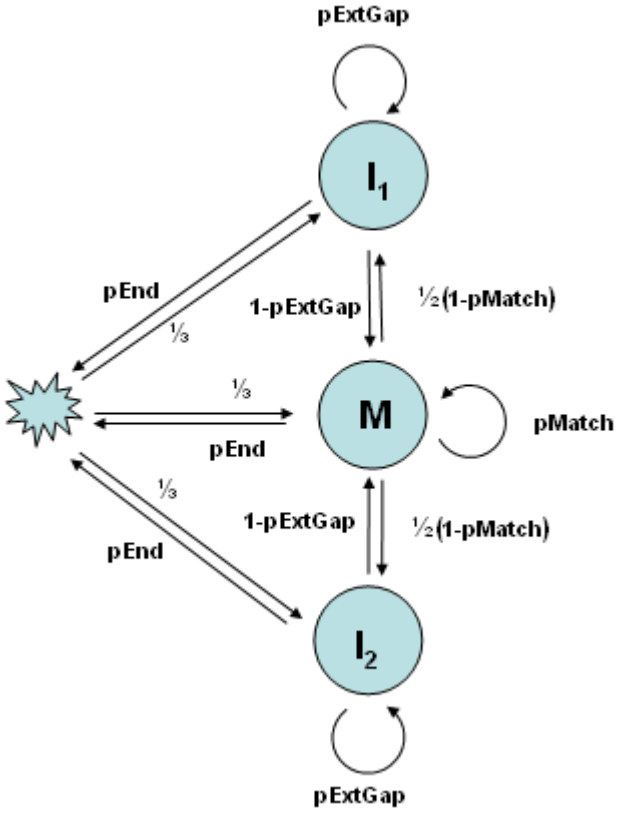
http://fr.wikipedia.org/wiki/Problème_des_sept_ponts_de_Königsberg

Réponse

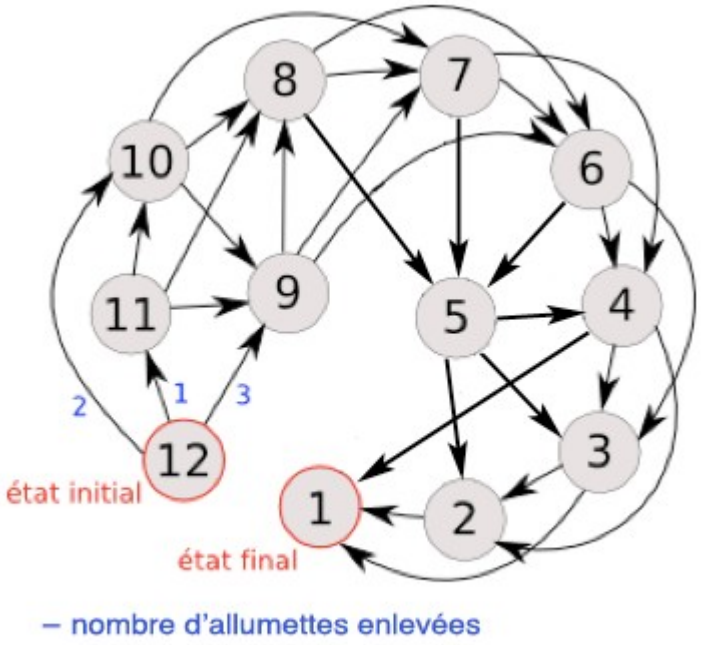
« Solutio problematis ad geometriam situs pertinentis »

Les graphes: définition (1)

Structure mathématique utilisée pour modéliser une **relation binaire** entre des objets ou des concepts



Modèle de Markov



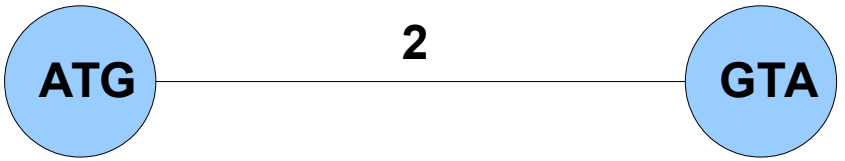
Jeu de Nim

Les graphes: définition (2)

Les objets sont représentés par des points appelés **sommets** ou **nœuds**.

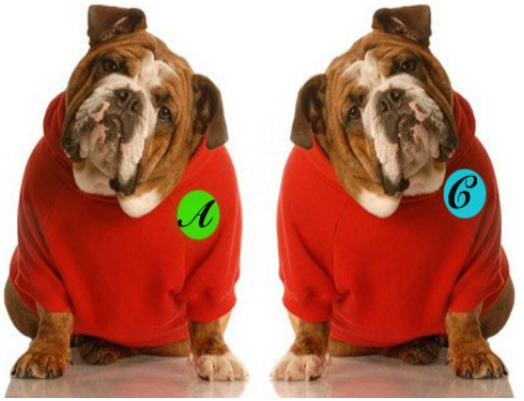
Les relations sont représentées par des lignes reliant les sommets et appelées **arêtes** ou **arcs**.

Une information peut être attachée aussi bien aux sommets qu'aux arêtes (**graphe étiqueté, valué, pondéré**).



Les graphes: définition (3)

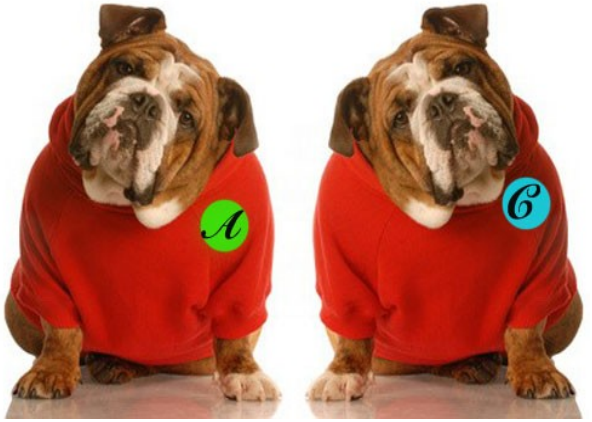
Deux types de graphes



Les graphes: définition (4)

Les **graphes non orientés**

la relation qui lie deux sommets est **symétrique**



« est de la même taille que »

Les graphes: définition (5)

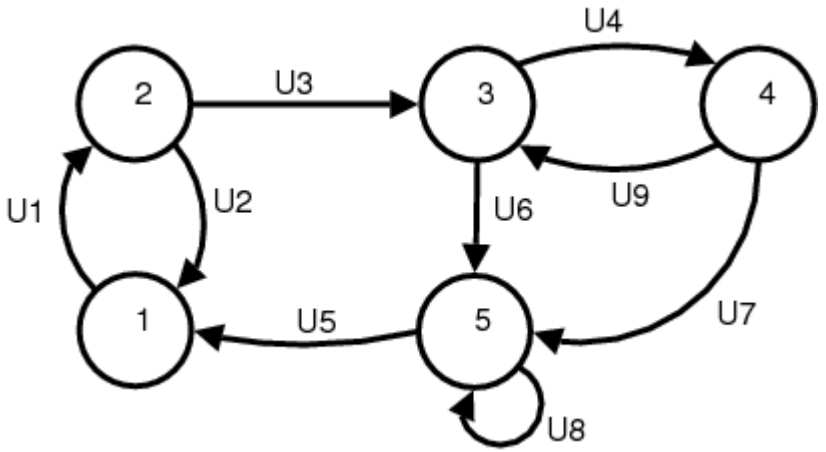
Les **graphes orientés**
la relation qui lie deux sommets **n'est pas symétrique**

« est plus petit que »

Les graphes orientés (1)

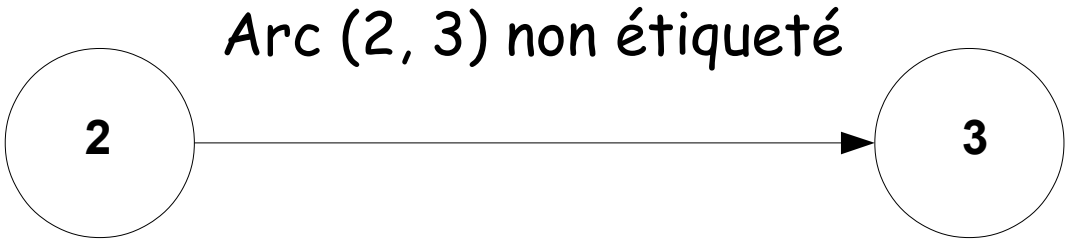
Un **graphe orienté G** est un couple **(S, A)** où

- **S** est l'ensemble fini non vide des **sommets** ou **nœuds** de G.
- **A** est un ensemble fini de couples (ordonnés) de S appelés **arcs** ou **flèches**.
- (x, y) est un arc qui relie le sommet x au sommet y.
- (x, v, y) est un arc **valué** par v reliant x à y.



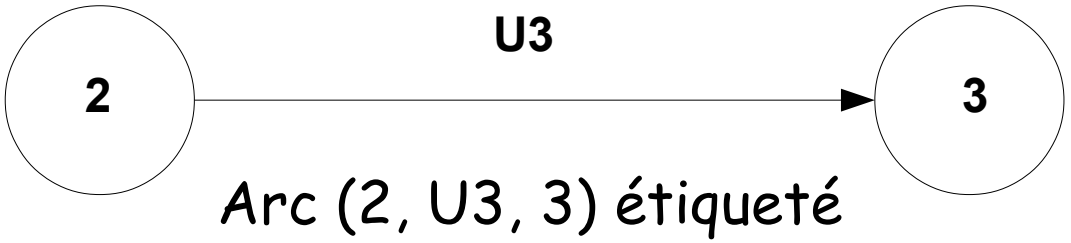
Graphe orienté valué
S={1, 2, 3, 4, 5}
A={(1, U1, 2), (2, U2, 1), (2, U3, 3), (3, U4, 4), (3, U6, 5), (4, U9, 3), (4, U7, 5), (5, U5, 1), (5, U8, 5)}
(5, U8, 5) est une **boucle**.

Les graphes orientés (2)



extrémité initiale
ou **origine**,
étiquette 2

extrémité terminale
ou **extrémité**,
étiquette 3



U3 est l'**étiquette** ou la **valeur** ou le **poids** de l'arc (2, U3, 3).

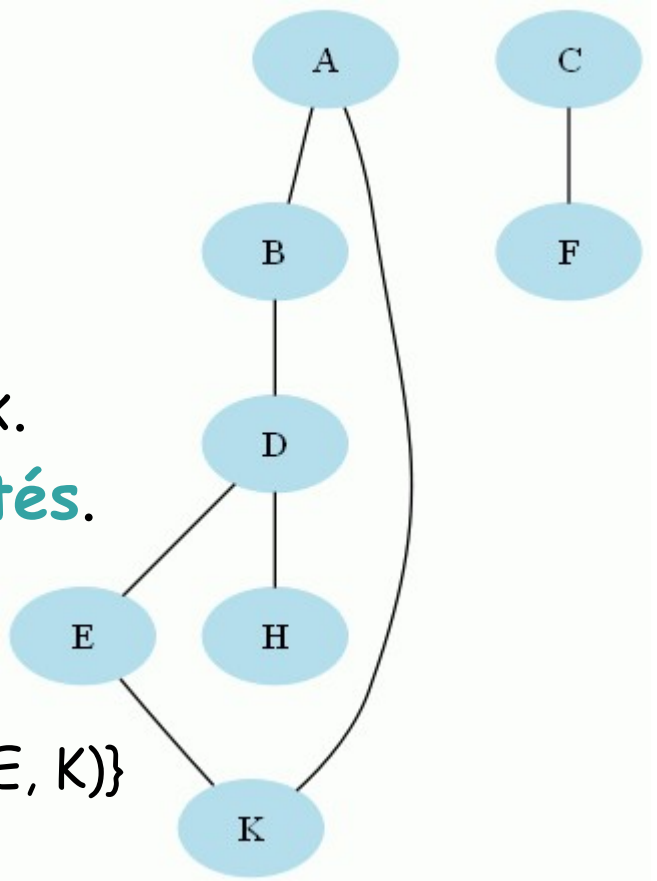
Les graphes non orientés

Un **graphe non orienté G** est un couple **(S, A)** où

- **S** est l'ensemble fini non vide des **sommets** ou **nœuds** de G.
- **A** est un ensemble fini de paires (non ordonnées) de S appelées **arêtes**.

(x, y) relie les sommets x et y entre eux.
(x, y) = (y, x) et x et y sont les **extrémités**.

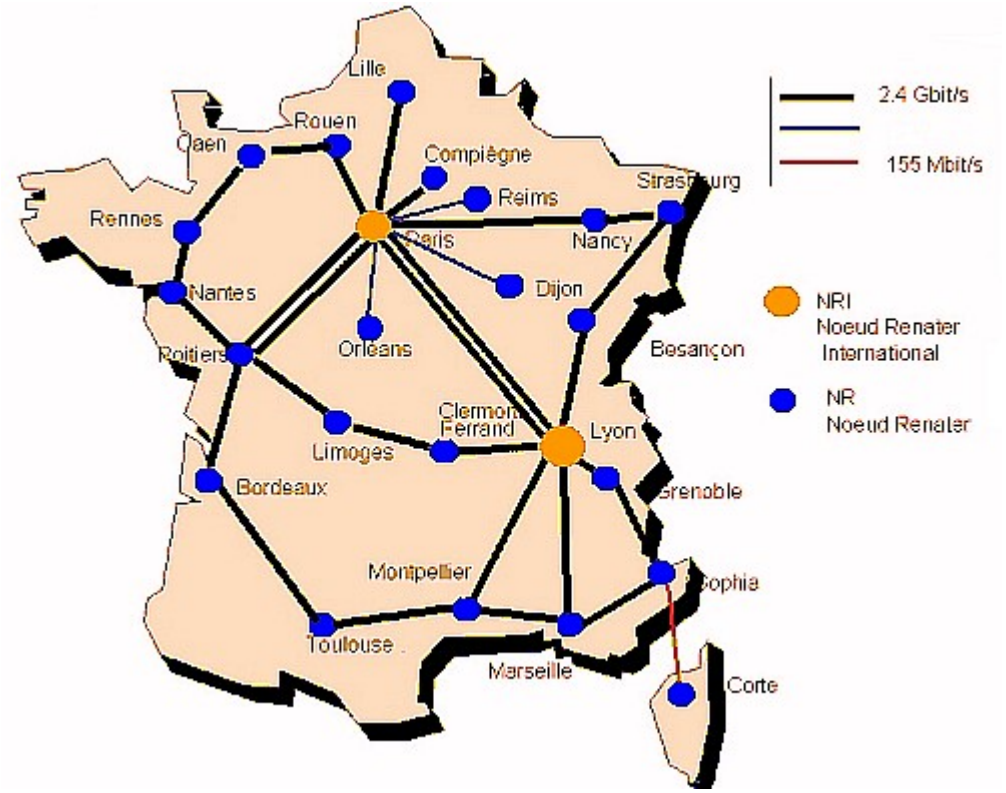
S = {A, B, C, D, E, F, H, K}
A = {(A, B), (A, K), (B, D), (C, F), (D, E), (D, H), (E, K)}



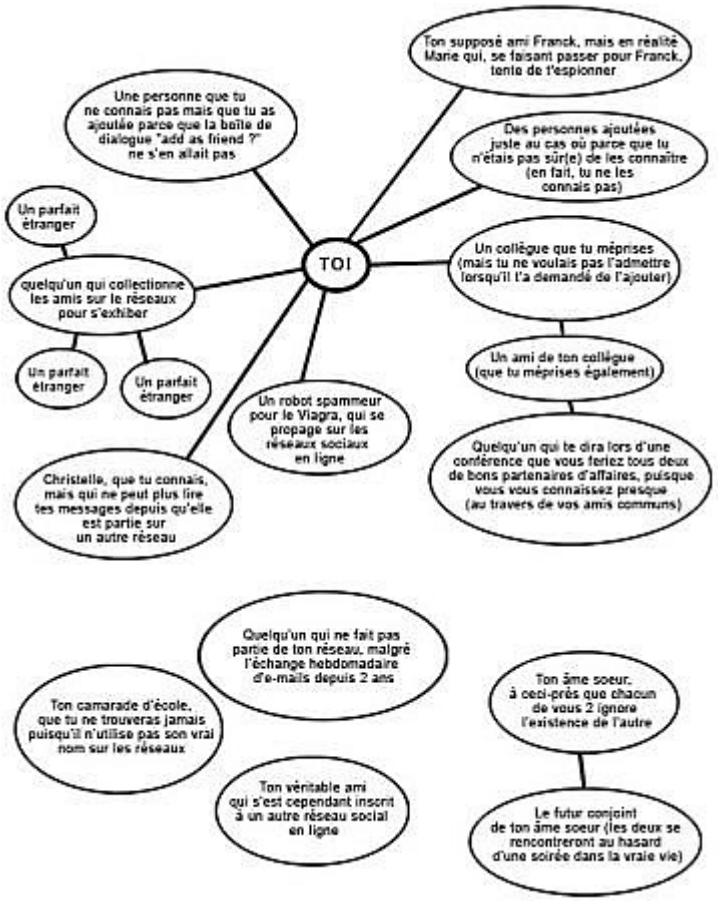
Les graphes : domaines d'application (1)

Modélisation des **réseaux de communication**

routes, réseaux informatiques, réseaux sociaux,…

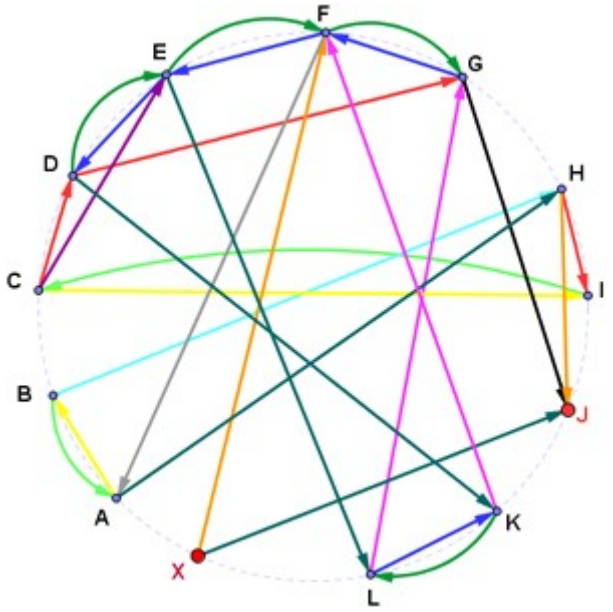


Réseau RENATER



Les graphes : domaines d'application (2)

Jeux, ordonnancement des tâches,...



Jeu Wyx

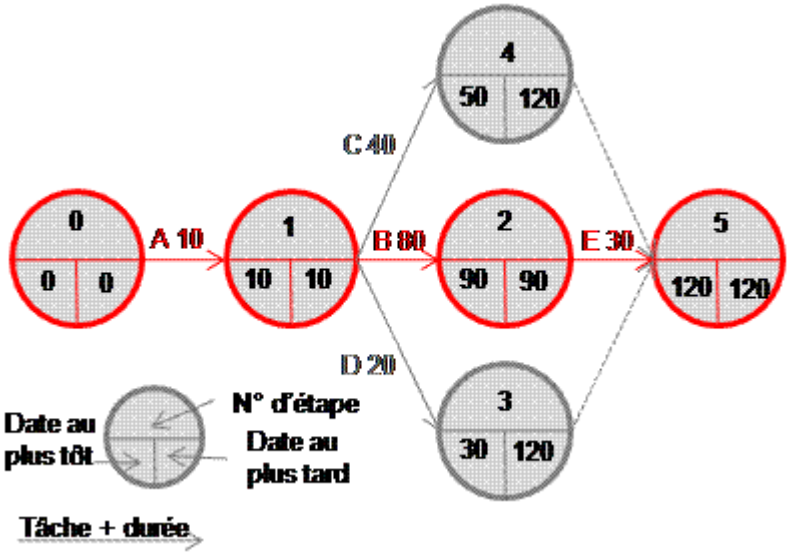
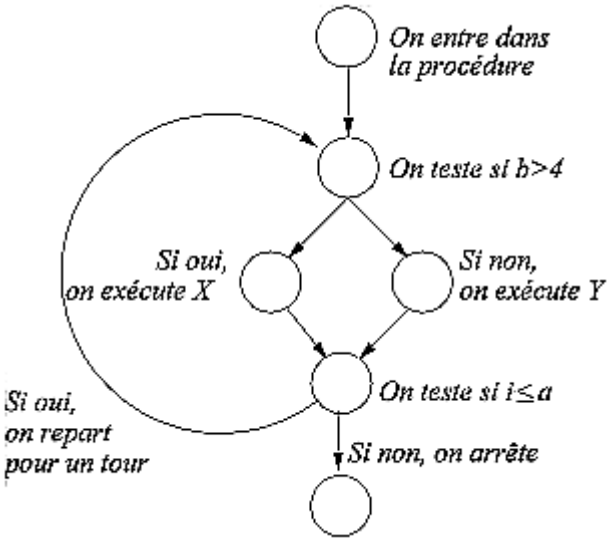


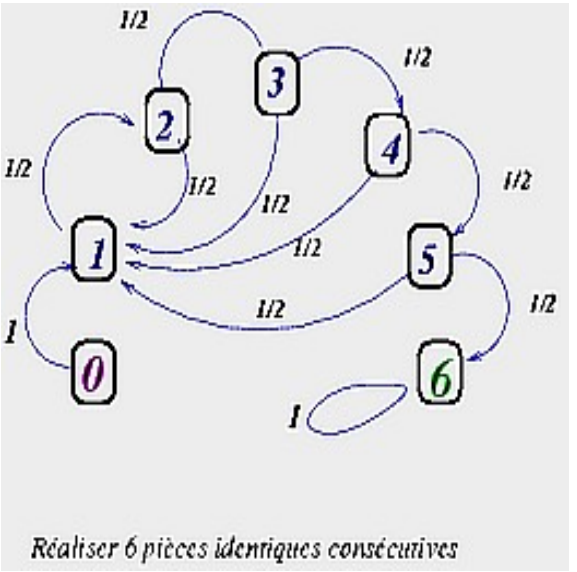
Diagramme Pert

Les graphes : domaines d'application (3)

processus évolutifs

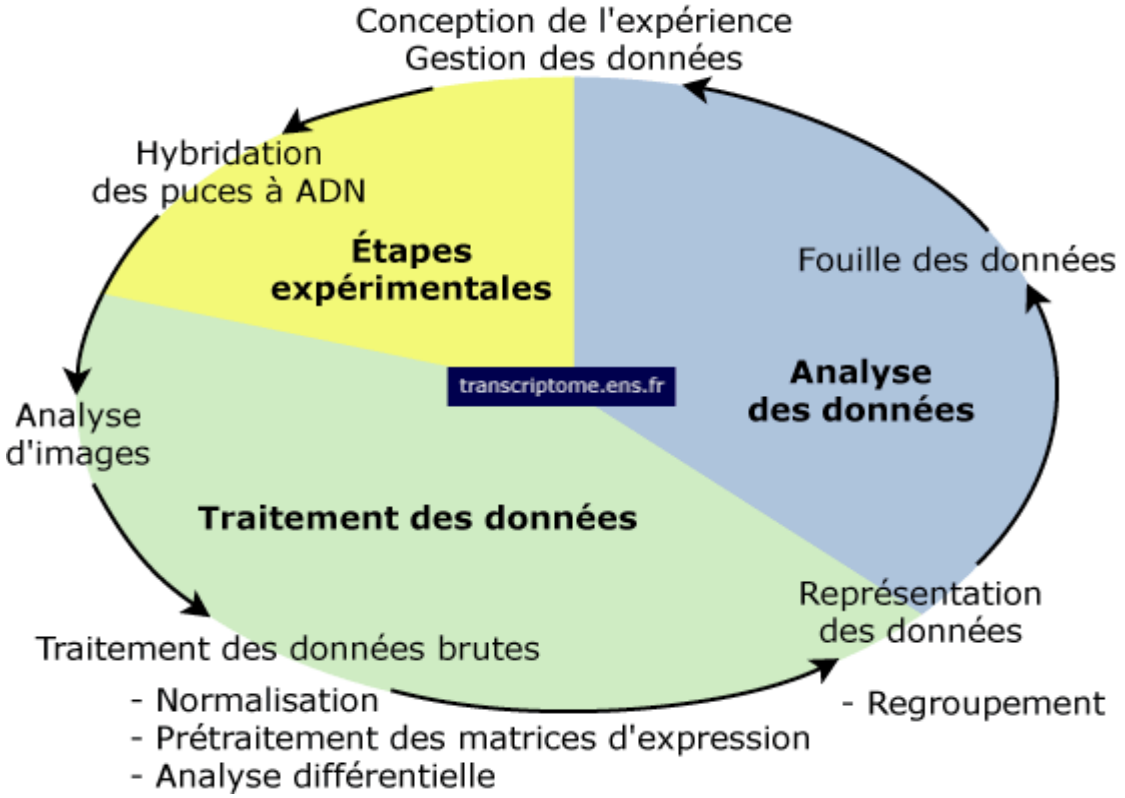
automates d'états finis, chaînes de Markov, organigrammes…





Les graphes et la bioinformatique (1)

Modéliser une expérimentation

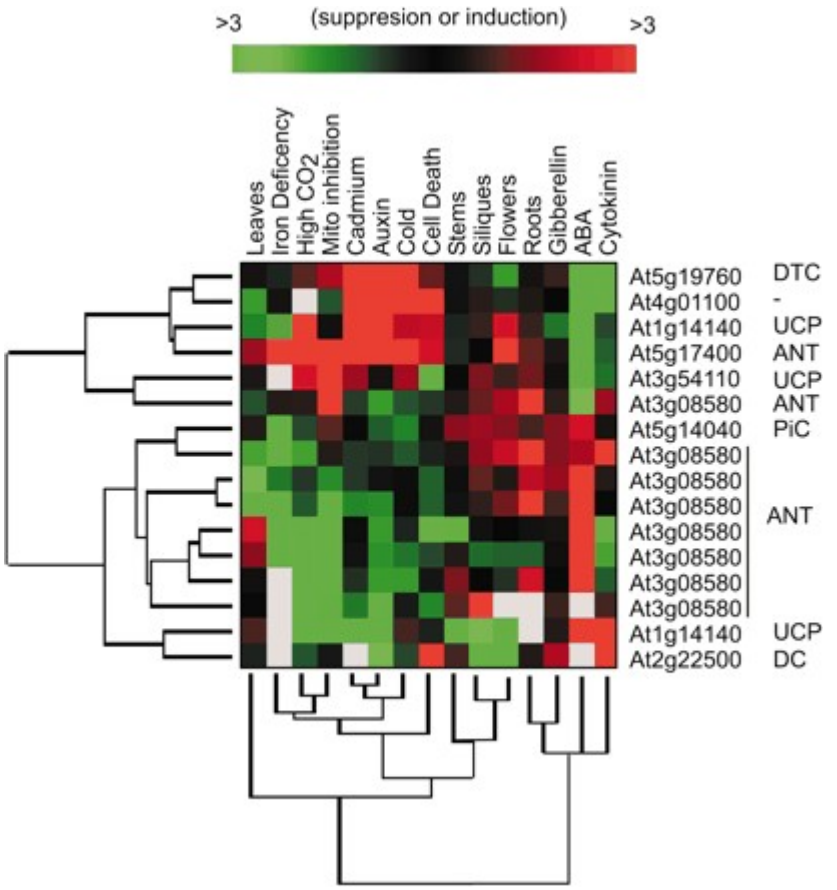


Source : "L'analyse des résultats de puces à ADN" - ENS

Les graphes et la bioinformatique (2)

Représenter les résultats d'une analyse

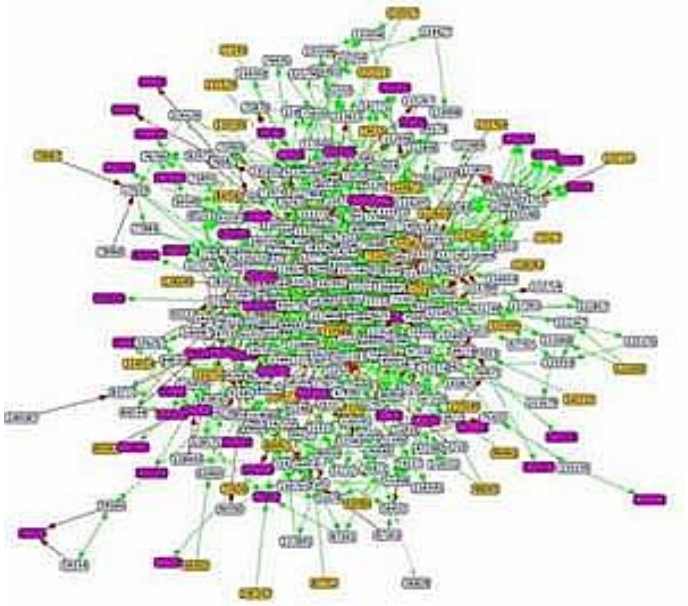
Différence de profils d'expression des gènes mitochondriaux de la famille des transporteurs de Arabidopsis thaliana.



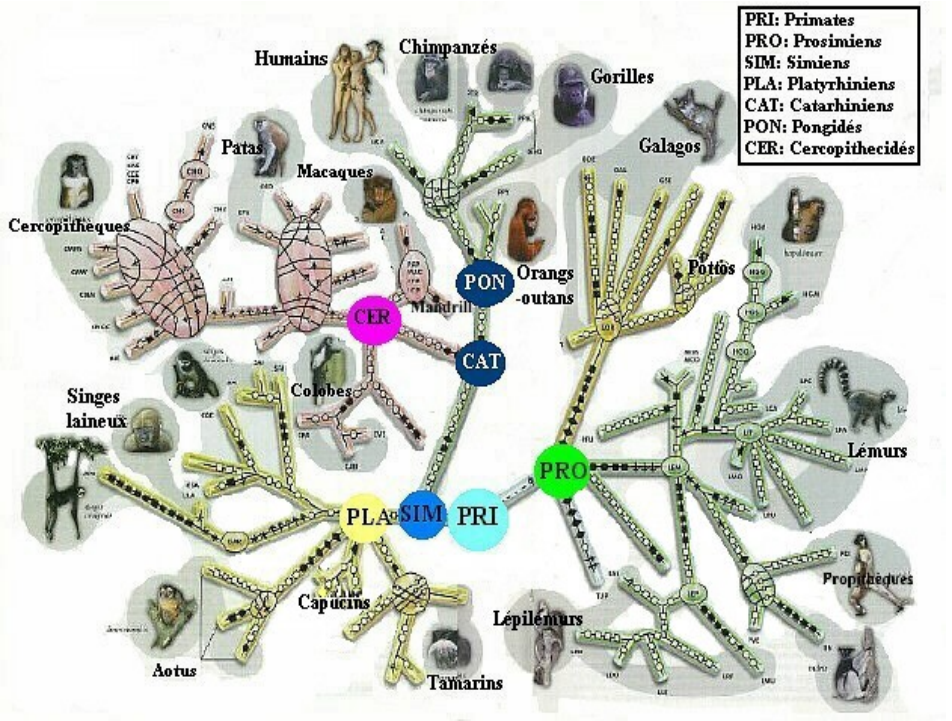
Source : Millar & Heazlewood (2003) "Genomic and Proteomic Analysis of Mitochondrial Carrier Proteins in Arabidopsis" Plant Physiol. 131, 443 - 453

Les graphes et la bioinformatique (3)

Représenter des relations entre des données



Réseau d'interactions moléculaires
© HELIOS BIOSCIENCES



Arbre phylogénétique

Les graphes : un peu de vocabulaire (0)

Mise en garde

Les graphes, c'est de l'histoire ancienne.
Alors, le vocabulaire est très riche, parfois redondant voire contradictoire selon les auteurs et les domaines d'application, donc

méfiance !!

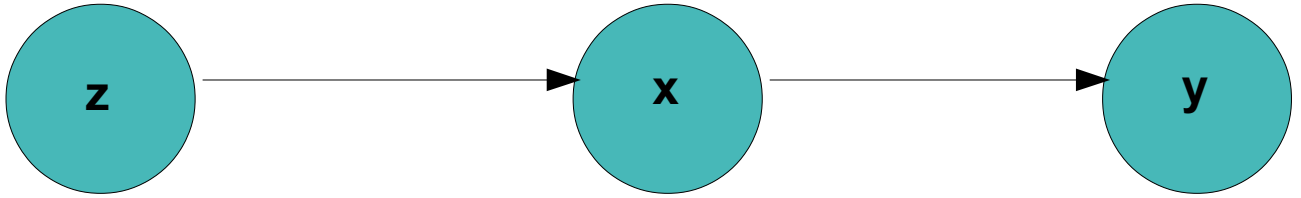
Les graphes : un peu de vocabulaire (1)

Deux sommets sont **adjacents** si il existe un arc ou une arête qui les relient.

Si G est un graphe orienté,

l'ensemble des **successeurs** d'un sommet x est l'ensemble des sommets y tels qu'il existe un arc (x, y),

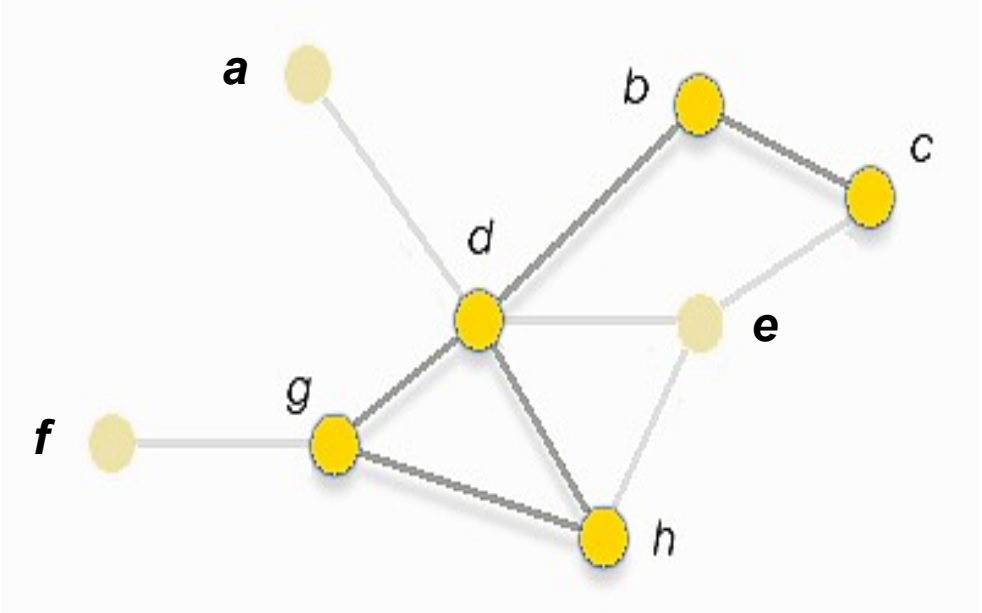
l'ensemble des **prédécesseurs** d'un sommet x est l'ensemble des sommets z tels qu'il existe un arc (z, x).



Les arcs (z, x) et (x, y) sont **consécutifs**.

Les graphes : un peu de vocabulaire (2)

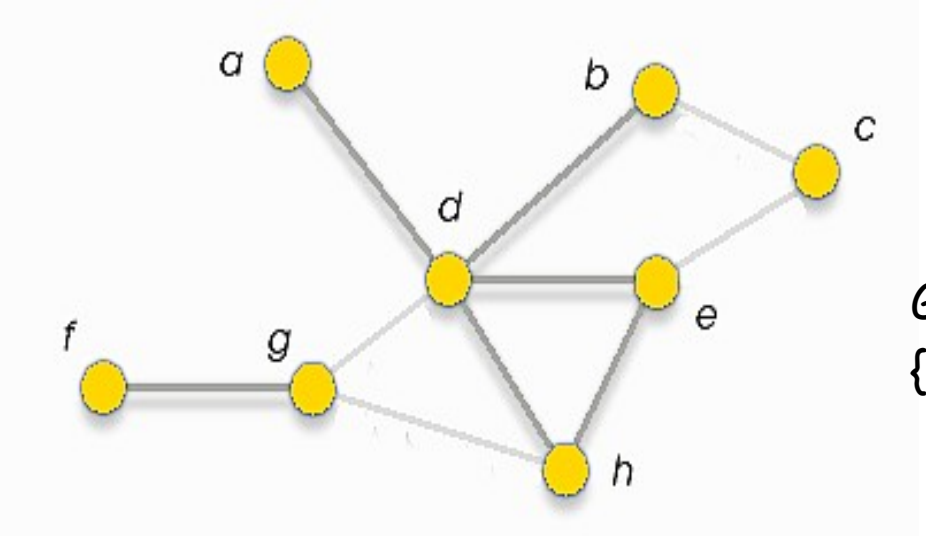
Un **sous-graphe** de G = (S, A) engendré par une partie X de S est le graphe dont les sommets sont les éléments de X et les arcs (ou arêtes) ceux de A dont les extrémités sont dans X.



Sous-graphe engendré par {b, c, d, g, h}

Les graphes : un peu de vocabulaire (3)

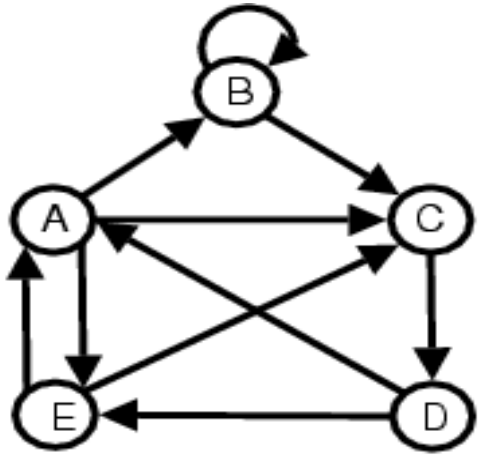
Un **graphe partiel** de G = (S, A) engendré par une partie H de A est le graphe G'=(S, H).



Graphe partiel engendré par
{(a, d), (b, d), (d, e), (e, h), (h, d), (f, g)}

Représentation des graphes par matrice d'adjacence (1)

Matrice booléenne



	A	B	C	D	E
A	0	1	1	0	1
B	0	1	1	0	0
C	0	0	0	1	0
D	1	0	0	0	1
E	1	0	1	0	0

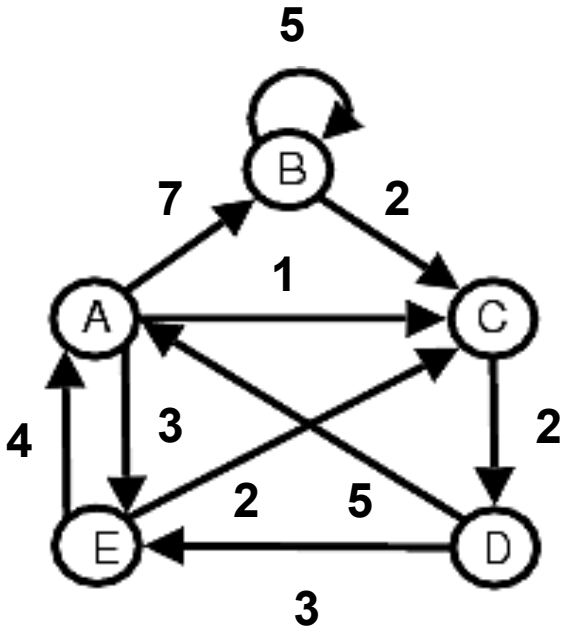
Inconvénient : Nécessite un espace mémoire proportionnel à $|S|^2$ quelque soit le nombre d'arêtes.

Avantage : L'espace occupé par chaque arête est faible.
On sait en **temps constant** si 2 sommets sont reliés.

⇒ devient intéressant quand le graphe est **dense** (nombre d'arêtes de l'ordre de $|S|^2$) et que l'algorithme nécessite fréquemment de savoir si 2 sommets sont reliés.

Représentation des graphes par matrice d'adjacence (2)

Matrice valuée

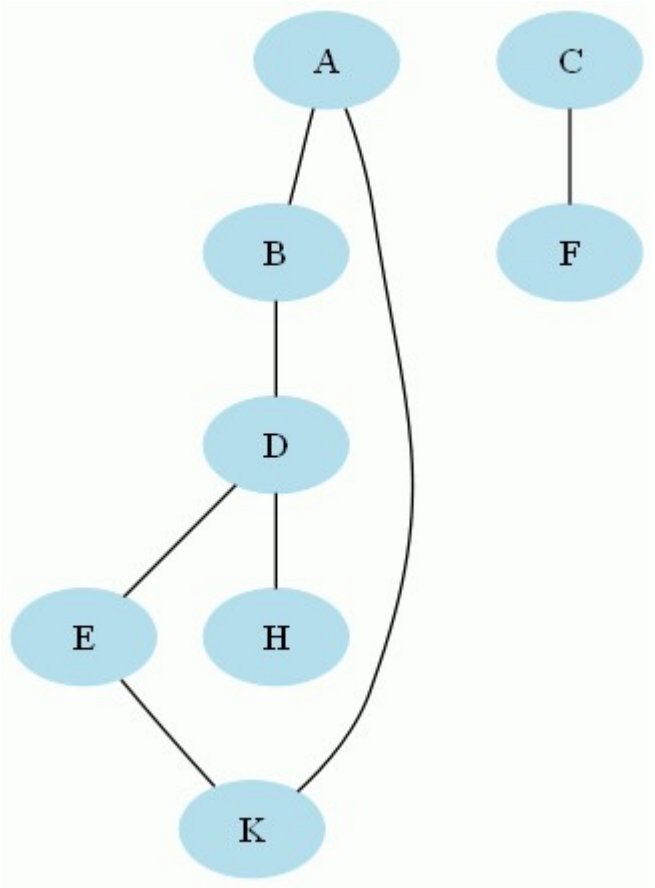


	A	B	C	D	E
A	0	7	1	0	3
B	0	5	2	0	0
C	0	0	0	2	0
D	5	0	0	0	3
E	4	0	2	0	0

Remarques : L'espace occupé par chaque arête dépend du type des valeurs des arêtes.
Lorsque 2 sommets ne sont pas reliés, on met dans la matrice une valeur adaptée à l'algorithme (0, ∞, ...).

Représentation des graphes par matrice d'adjacence (3)

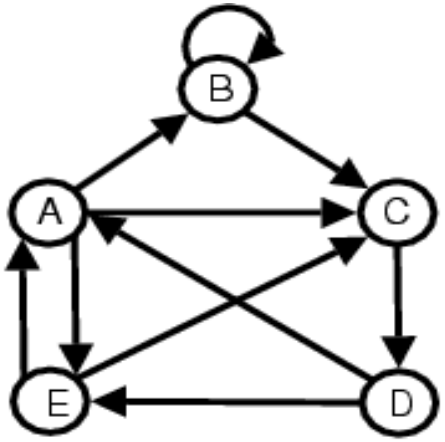
Cas des graphes non orientés



	A	B	C	D	E	F	H	K
A	0	1	0	0	0	0	0	1
B	1	0	0	1	0	0	0	0
C	0	0	0	0	0	1	0	0
D	0	1	0	0	1	0	1	0
E	0	0	0	1	0	0	0	1
F	0	0	1	0	0	0	0	0
H	0	0	0	1	0	0	0	0
K	1	0	0	0	1	0	0	0

Remarque : la matrice est symétrique.

Représentation des graphes par listes d'adjacence (1)



A : (B, C, E)
B : (B, C)
C : (D)
D : (A, E)
E : (A, C)

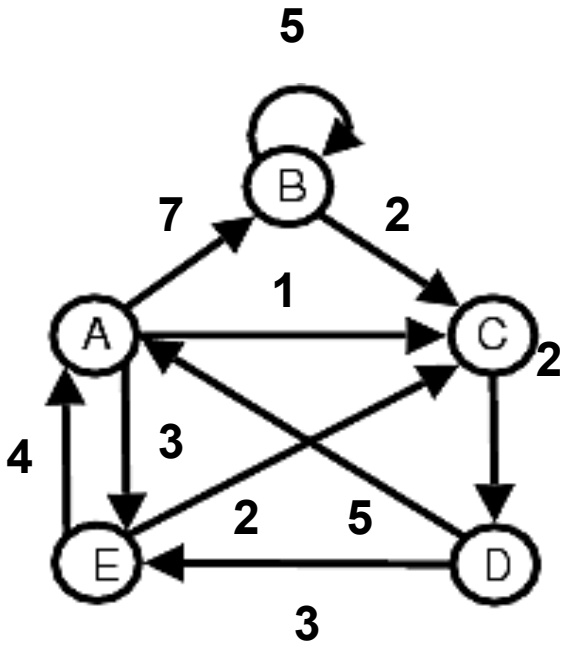
Inconvénient : L'espace occupé par chaque arête est relativement élevé. Tester si 2 sommets sont reliés peut coûter cher.

Avantage : L'espace occupé par le graphe est proportionnel au nombre d'arêtes du graphe (intéressant quand le nombre d'arêtes est inférieur à $|S|^2$).

Remarque : L'ordre dans lequel sont rangés les éléments de la liste est important.

Représentation des graphes par listes d'adjacence (2)

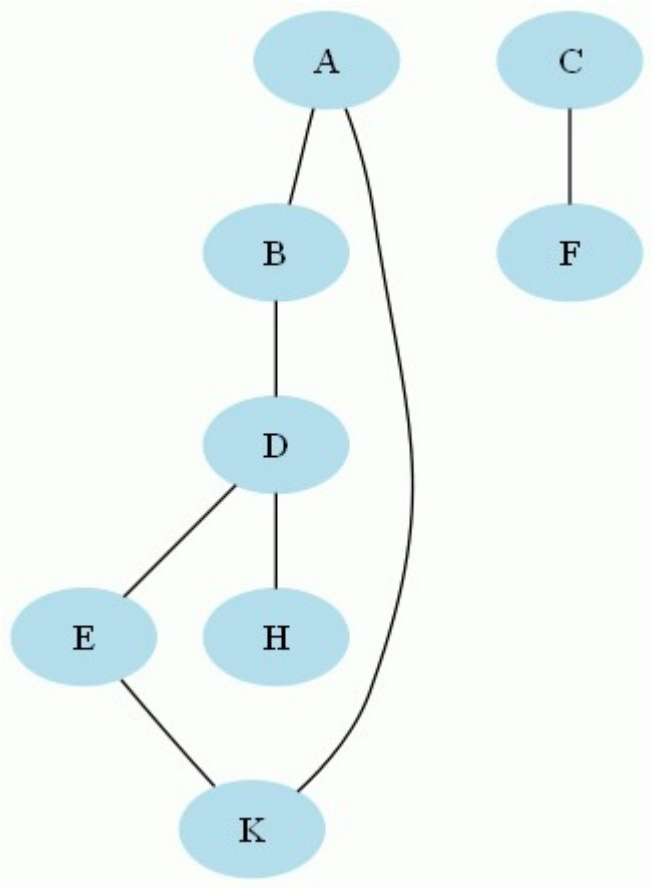
Cas des graphes valués



A : ((B, 7), (C, 1), (E, 3))
B : ((B, 5), (C, 2))
C : ((D, 2))
D : ((A, 5), (E, 3))
E : ((A, 4), (C, 2))

Représentation des graphes par listes d'adjacence (3)

Cas des graphes non orientés



A : (B, K)
B : (A, D)
C : (F)
D : (B, E, H)
E : (D, K)
F : (C)
H : (D)
K : (A, E)

Remarque : chaque arête apparait 2 fois

Représentation des graphes : résumé

Soit G un graphe ayant **n** sommets et **m** arcs (ou arêtes).

Représentation par matrice d'adjacence

Stockage : complexité en espace de l'ordre de n^2

Existence d'une arête entre 2 sommets : **temps constant**

Représentation par listes d'adjacence

Stockage : complexité en espace de l'ordre de **n+m**

Existence d'une arête entre 2 sommets : complexité en temps de l'ordre de **n dans le pire des cas**

Conclusion

Le choix de la représentation dépend de l'algorithme.

Remarque : dans certains cas, on utilisera des listes de prédécesseurs.

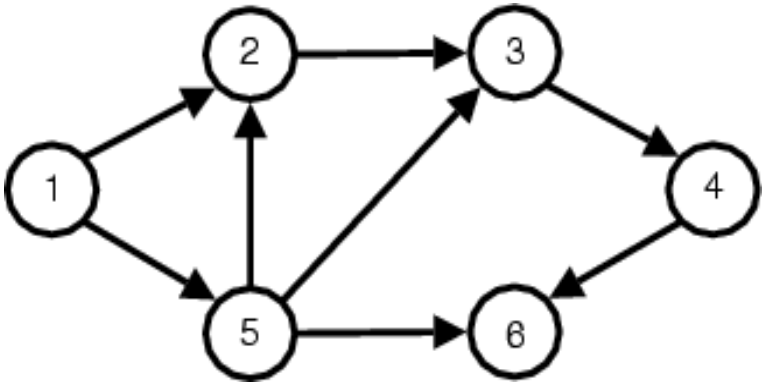
Représentation des graphes

Exercices (1)

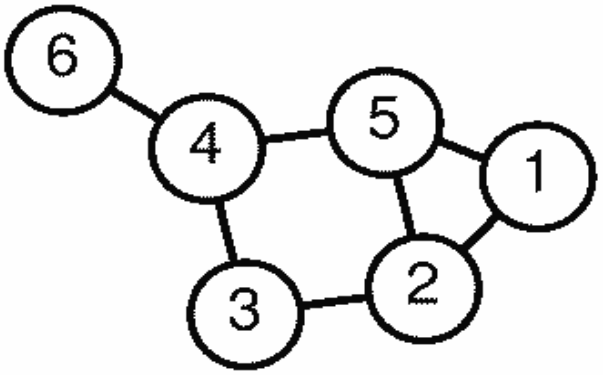
Exercice 1 : Échauffement

On considère les graphes G et G' ci-dessous. Donnez leurs propriétés, leur représentation par matrice d'adjacence et par listes d'adjacence.

G :



G':



Représentation des graphes

Exercices (2)

Exercice 2 : Revenons à la bioinformatique

On considère l'ensemble de fragments d'ADN S = {*ACA*, *AGCA*, *CAC*, *CAGCAC*, *CCA*, *CGA*, *CGAC*, *GACA*} et les 3 relations binaires R_1, R_2 et R_3 :

Soient u et v deux éléments de S,

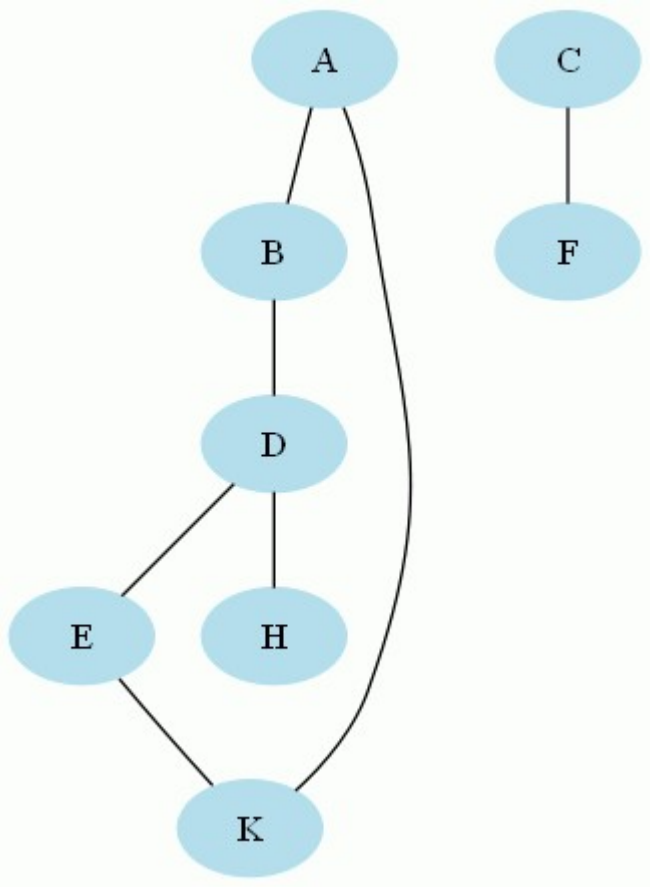
- u R_1 v si on peut obtenir v à partir de u par la suppression d'un ou plusieurs nucléotides.

La distance de Hamming entre deux fragments de même longueur est égale au nombre de positions sur lesquelles les fragments diffèrent. Par exemple, $d_H(AAC, ACG) = 2$.

- u R_2 v si $0 < d_H(u, v) \leq 2$, pour u et v ayant même longueur.
- u R_3 v si il existe un suffixe de u (une fin de u) non vide différent de u qui est un préfixe (un début) non vide de v différent de v.

Dessinez les graphes représentant les relations R_1 et R_2 sur S. Donnez la matrice d'adjacence du graphe de R_3 sur S.

Promenons-nous dans les graphes (1)
Chaines



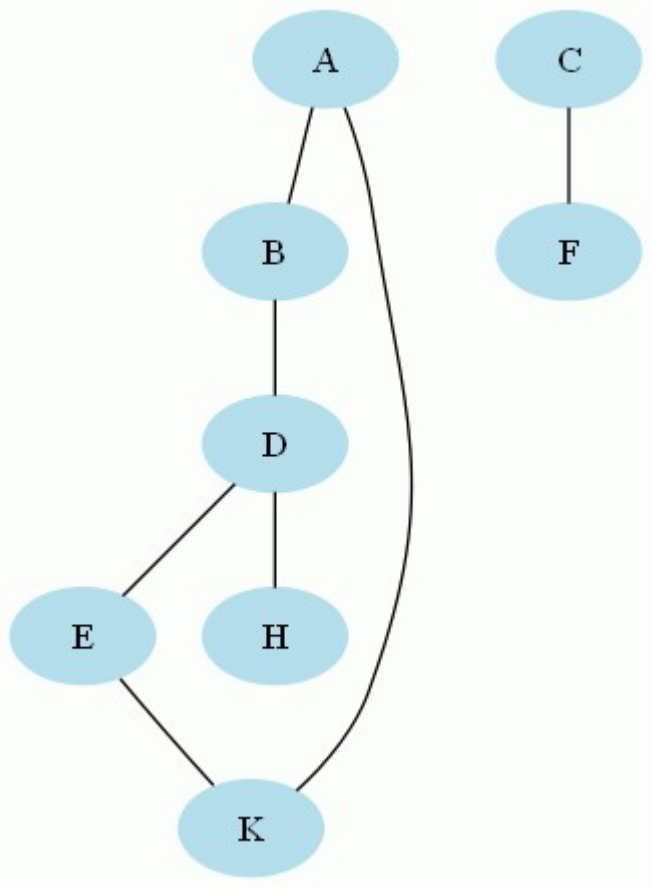
Dans un **graphe non orienté**, une **chaine** reliant un sommet x à un sommet y est une suite finie d'arêtes consécutives reliant x à y. x et y sont les **extrémités** de la chaine.

La **longueur** de la chaine est le nombre d'arêtes qui la composent.

Une chaine qui ne passe pas deux fois par le même sommet est dite **élémentaire**.
Une chaine qui n'utilise pas deux fois la même arête est dite **simple**.

((A, K), (K, E), (E, D), (D, H)) est une chaine élémentaire et simple de longueur 4.
((A, K), (K, E), (E, D), (D, B), (B, A)) est une chaine simple qui n'est pas élémentaire.
((A, K), (K, E), (E, D), (D, B), (B, A), (A, K)) est une chaine qui n'est ni simple ni élémentaire.

Promenons-nous dans les graphes (2)
Cycles

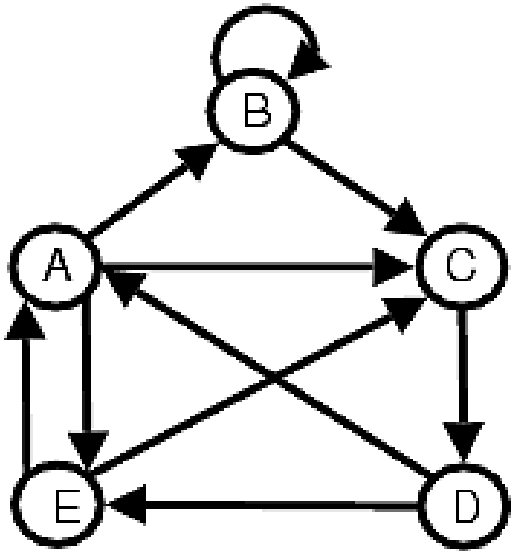


Dans un graphe non orienté, un **cycle** est une chaine dont les extrémités coïncident.
Dans un **cycle simple** toutes les arêtes sont distinctes.
Dans un **cycle élémentaire** tous les sommets sont distincts (seules les extrémités de la chaine coïncident).

((A, K), (K, E), (E, D), (D, B), (B, A)) est un cycle simple et élémentaire.
((C, F), (F, C)) est un cycle élémentaire qui n'est pas simple .

Remarque : on peut considérer la notion de chaine et de cycle dans un graphe orienté. Dans le « parcours » d'une chaine, on ignore alors le sens des arcs et on peut donc suivre un arc « à l'envers ».

Promenons-nous dans les graphes (3)
Chemins



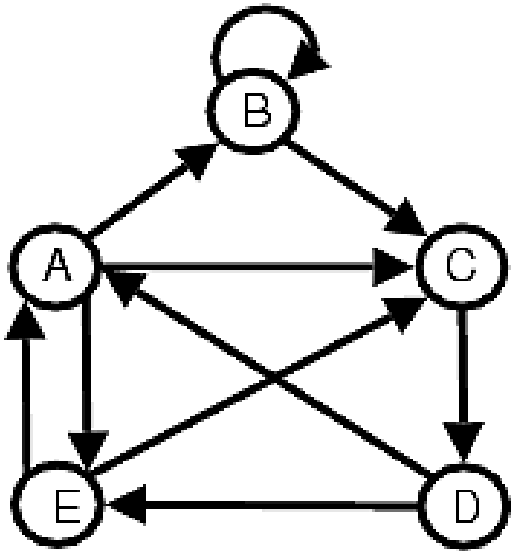
Dans un graphe orienté, un **chemin** d'**origine** x et d'**extrémité** y est une suite finie d'arcs consécutifs reliant x à y.

Un chemin **élémentaire** est un chemin ne passant pas deux fois par un même sommet.

Un chemin **simple** est un chemin ne passant pas deux fois par un même arc.

((A, B), (B, C), (C, D)) est un chemin simple et élémentaire.
((A, B), (B, B), (B, C)) est un chemin simple qui n'est pas élémentaire.
((E, A), (A, E), (E, A)) est un chemin ni simple ni élémentaire.

Promenons-nous dans les graphes (4)
Circuits



Dans un graphe orienté, un **circuit** est un chemin dans lequel l'origine et l'extrémité coïncident.

On définit les circuits **élémentaires** et les circuits **simples** de manière similaire aux définitions sur les cycles.

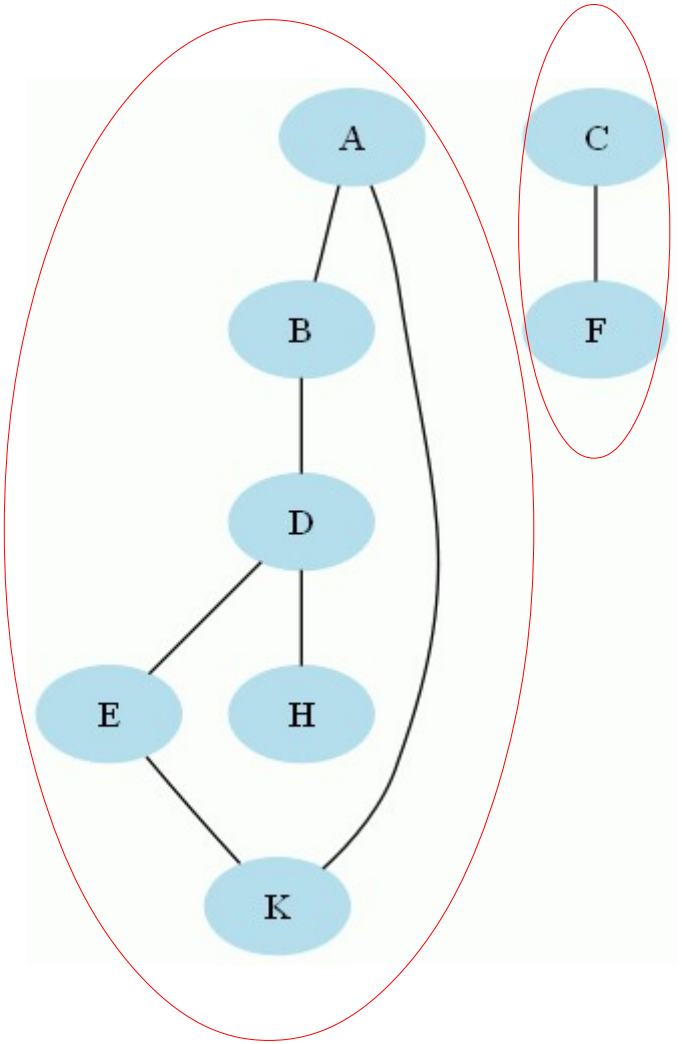
((A,E), (E, A)) et ((B, B)) sont des circuits simples et élémentaires.
((E, A), (A, C), (C, D), (D, A), (A, E)) est un circuit simple mais pas élémentaire.

Les graphes : connexité

Un graphe est **connexe** si quels que soient 2 sommets x et y, il existe au moins une chaine qui relie x et y.

Une **composante connexe** d'un graphe G est un sous-graphe connexe maximal de G, c'est-à-dire que si on ajoute n'importe quel sommet dans le sous-graphe, il n'est plus connexe.

Le graphe ci-contre n'est pas connexe.
Il possède 2 composantes connexes entourées en rouge.



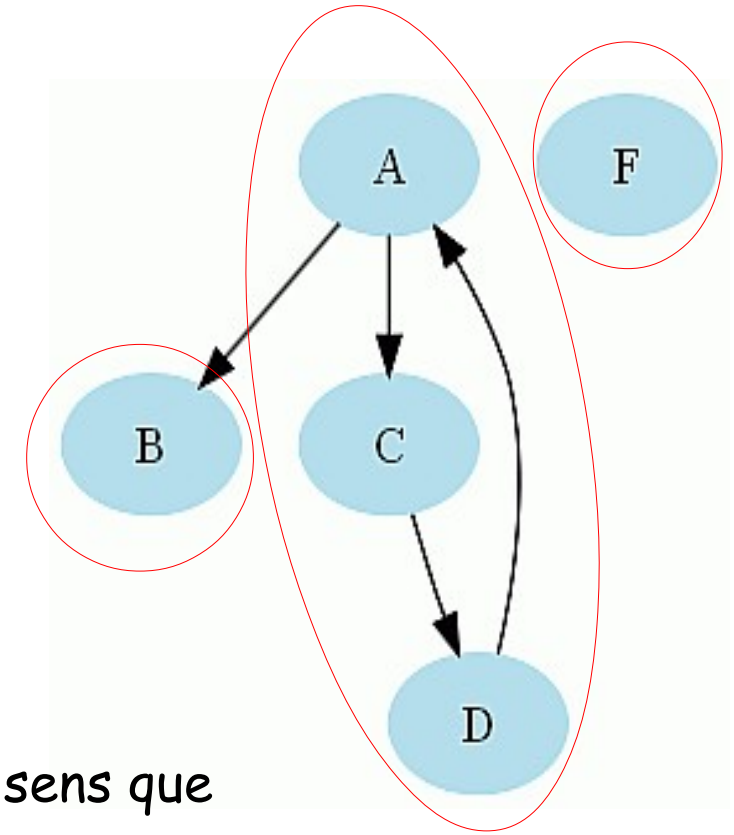
Remarque : la notion de connexité est donc définie sur les graphes orientés ou non orientés. Dans le cas des graphes orientés, on ne considère pas le sens des arcs.

Les graphes : forte connexité

Un graphe orienté est **fortement connexe** si pour 2 sommets quelconques x et y, il existe un chemin allant de x à y et un chemin allant de y à x.

Une **composante fortement connexe** d'un graphe orienté est un sous-graphe fortement connexe maximal.

Le graphe ci-contre n'est ni connexe ni fortement connexe. Ses composantes fortement connexes sont entourées en rouge.



Remarque : le notion de forte connexité n'a de sens que sur les graphes orientés.

Chaines, connexité et compagnie

Exercices (1)

Exercice 3 : Échauffement

soit G le graphe défini par les listes d'adjacence ci-dessous.

A : (C, D, G)	B : (C, E)
C : (A)	D : (G)
E : (F)	F : (E, G)
G : (H)	H : (D)

Ce graphe est-il orienté ou non ? Dessinez-le. Donnez, si ils existent, une chaine simple entre le sommet D et le sommet B, une chaine élémentaire entre D et B, un cycle élémentaire, un chemin simple entre B et H, un chemin élémentaire entre B et H, un circuit élémentaire. Donnez ses composantes connexes et ses composantes fortement connexes. Ce graphe est-il connexe, fortement connexe ?

Chaines, connexité et compagnie

Exercices (2)

Exercice 4 : Revenons à la bioinformatique

Reprenez les graphes R_1 et R_2 obtenus dans l'exercice 2. Donnez pour chacun d'eux ses composantes connexes et (quand cela a un sens) ses composantes fortement connexes.

Chaine eulérienne, hamiltonienne, ...

Une chaine est **eulérienne** si elle contient une fois et une seule chaque arête (ou arc) du graphe. Si une chaine eulérienne est un cycle, on a un **cycle eulérien**.

Une chaine est **hamiltonienne** si elle passe une fois et une seule par chaque sommet du graphe. Si une chaine hamiltonienne est un cycle, on a un **cycle hamiltonien**.

Il existe une définition similaire pour les **chemins eulériens** et les **circuits eulériens** et pour les **chemins hamiltoniens** et les **circuits hamiltoniens**.

Chaine eulérienne, hamiltonienne, ...

Exercices

Exercice 5 : Considérez à nouveau le graphe obtenu avec la relation R_3. Nous allons réaliser une simplification du graphe en supprimant tous les sommets correspondants à des fragments qui sont inclus dans d'autres fragments ainsi que tous les arcs qui sont reliés à ces sommets. Les boucles sont interdites. Les arcs sont maintenant pondérés par la longueur du plus long chevauchement. Dessinez le graphe obtenu. Donnez les composantes connexes et fortement connexes. Ces composantes ont-elles un intérêt biologique particulier ? Le poids d'un chemin est égal à la somme des poids des arcs qui le composent. Existe-t-il dans ce graphe des chemins eulériens, des chemins hamiltoniens ? Si de tels chemins existent, quel est leur poids ? Ces chemins ont-ils un intérêt biologique particulier ?

Exercice 6 : et si on reparlait d'Euler ...

Le problème posé par Euler en 1735 a-t-il une solution ?

Réponse de l'exercice 6

On appelle **degré** d'un sommet le nombre d'arêtes ou d'arcs qui sortent de ce sommet.

Le théorème d'Euler

(i) Un graphe connexe admet un cycle eulérien si et seulement si il ne possède aucun sommet de degré impair.

(ii) Un graphe connexe admet une chaine eulérienne si et seulement si il possède 0 ou 2 sommets de degré impair.

Conclusion ?

Algorithmes sur les graphes (1)

Il existe de nombreux algorithmes sur les graphes. Voici une liste non exhaustive des problèmes que l'on peut se poser sur les graphes et de quelques algorithmes qui répondent à ces questions.

Le parcours d'un graphe

Problème : comment visiter tous les sommets d'un graphe en passant une fois et une seule par chaque sommet ?
Réponses : parcours en profondeur d'abord, parcours en largeur d'abord

La fermeture réflexive et transitive d'un graphe orienté

Problème : soient 2 sommets x et y quelconques d'un graphe, existe-t-il un chemin allant de x à y ?
Réponses : algorithme de Warshall

Algorithmes sur les graphes (2)

Les plus courts chemins entre 2 sommets

Problème : soit x et y 2 sommets d'un graphe, combien coûte au minimum un chemin allant de x à y ?

Réponses : algorithme de Floyd , algorithme de Dijkstra lorsque la source (sommet de départ) est unique, algorithme de Bellman dans un graphe orienté sans circuit.

Arbre recouvrant d'un graphe non orienté

Problème : On considère G un graphe connexe non orienté valué. On appelle arbre recouvrant de G un arbre dont les sommets sont les sommets de G. On recherche pour G un arbre recouvrant de coût minimum.

Exemple d'application : Construction à moindre coût d'un réseau de communication permettant de relier plusieurs sommets.

Réponses : algorithme de Kruskal, algorithme de Prim

Algorithmes sur les graphes (3)

Le problème du voyageur de commerce

Problème : déterminer un plus court chemin passant par tous les sommets d'un graphe.
Exemple d'application : étant donné un ensemble de villes séparées par des distances données, trouver le plus court chemin qui relie toutes les villes.
Réponse : problème NP-complet !

Cela signifie qu'on ne connaît pas de méthode de résolution permettant d'obtenir des solutions exactes en un temps raisonnable pour de grandes valeurs (grand nombre de villes). On devra donc souvent se contenter de solutions approchées. On se retrouve en effet face à une explosion combinatoire : le nombre de chemins possibles passant par 69 villes est déjà un nombre à 100 chiffres!!

Une stratégie possible est d'utiliser un **algorithme glouton**.

Les algorithmes gloutons

Un algorithme glouton est un algorithme qui suit le principe de faire, étape par étape, un choix **optimum local**, dans l'espoir d'obtenir un résultat **optimum global**. Dans les cas où l'algorithme ne fournit pas systématiquement la solution optimale, on parle d'**heuristique gloutonne**.

Par exemple, dans le problème du rendu de monnaie (donner une somme avec le moins possible de pièces), l'algorithme consistant à répéter le choix de la pièce de plus grande valeur qui ne dépasse pas la somme restante est un algorithme glouton. Suivant le système de pièces, l'algorithme glouton est optimal ou pas. Dans le système de pièces européen (en centimes : 1, 2, 5, 10, 20, 50, 100, 200), où l'algorithme glouton donne la somme suivante pour 37 : 20 + 10 + 5 + 2, on peut montrer que l'algorithme glouton donne toujours une solution optimale. Par contre, dans le système de pièces (1, 3, 4), l'algorithme glouton n'est pas optimal, comme le montre l'exemple simple suivant. Il donne pour 6 : 4 + 1 + 1, alors que 3 + 3 est optimal.

Une solution au problème du voyageur de commerce : algorithme des plus proches voisins

Cet algorithme est simple et rapide, mais il n'est efficace que pour un nombre de villes assez réduit. Il est souvent utilisé pour constituer rapidement un trajet acceptable servant de base à un autre algorithme. Pour un nombre de villes important, on obtient généralement une solution plus longue de 50% ou moins au dessus de la solution optimale.

Principe :

Le trajet est initialement vide (pas de villes visitées).
On part d'une ville au hasard, que l'on met dans la liste des villes visitées.
On recherche la ville la plus proche que l'on ajoute dans la liste des villes visitées.
On recherche la ville la plus proche de cette nouvelle ville. Si la ville est déjà dans la liste des villes visitées, on prend la deuxième ville plus proche, la troisième si besoin, et ainsi de suite...
Une manière d'accélerer fortement le calcul de ces solutions est de calculer auparavant le tableau des villes les plus proches (pour chaque ville, le tableau contient la liste des autres villes triées dans l'ordre de la plus proche à la plus éloignée).

Exploration d'un graphe
Parcours en profondeur (1)

Problème : Soit *G* un graphe orienté. On veut visiter tous les sommets de *G* une fois et une seule.

Une première technique consiste à suivre un chemin dans le graphe aussi longtemps que possible avant de passer au suivant : c'est le **parcours en profondeur d'abord.**

Le marquage des sommets

Pour éviter de visiter plusieurs fois un même sommet et de boucler indéfiniment dans un cycle, on associe à chaque sommet du graphe une marque qui précise son état à un moment donné. Le sommet peut être « à voir » quand on ne l'a pas encore visité ou « vu » lorsque la visite a été effectuée. La procédure de parcours consiste à faire passer tous les sommets de « à voir » à « vu ».

Exploration d'un graphe
Parcours en profondeur (2)

```
Algorithme parcoursEnProfondeur
(G : graphe)
début
    pour chaque sommet s de S  faire
        marque[s] ← A_VOIR
    fpour
    pour chaque sommet s de S faire
        si marque[s] = A_VOIR :
            visiteEnProfondeur(s)
        fsi
    fpour
fin
```

```
Algorithme visiteEnProfondeur
(s : Sommet)
début
    marque[s] ← VU
    Traitement(s)
    pour chaque t adjacent à s  faire
        si marque[t] = A_VOIR :
            visiteEnProfondeur(t)
        fsi
    fpour
fin
```

Remarque : l'algorithme visiteEnProfondeur explore de manière systématique les sommets adjacents à un sommet s. Il est donc plus judicieux d'utiliser une représentation par listes d'adjacence que par matrice d'adjacence.

Exploration d'un graphe
Parcours en profondeur (3)

On considère le graphe donné par les listes d'adjacence suivantes :

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

Rappel : l'ordre dans lequel sont stockés les sommets dans les listes d'adjacence est important, de même que l'ordre dans lequel les sommets sont considérés dans la procédure parcoursEnProfondeur. L'ordre utilisé ici est l'ordre alphabétique.

Code couleur : A appel dans parcoursEnProfondeur
A appel dans visiteEnProfondeur
A marque de A mise à VU

Exploration d'un graphe
Parcours en profondeur (3)

On considère le graphe donné par les listes d'adjacence suivantes :

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	sommet
AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	A

Exploration d'un graphe
Parcours en profondeur (3)

On considère le graphe donné par les listes d'adjacence suivantes :

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	sommet
AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	A
V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	B

Exploration d'un graphe
Parcours en profondeur (3)

On considère le graphe donné par les listes d'adjacence suivantes :

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	sommet
AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	A
V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	B
V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	E

Exploration d'un graphe
Parcours en profondeur (3)

On considère le graphe donné par les listes d'adjacence suivantes :

A : (B,C,D)
B : (E,F)
C : (G,H,I)
D : (B,F)
E : (A,C)
F : (G,I)
G : (E)
H : (F,G)
I : (D)
J : (K,L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	sommet
AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	A
V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	B
V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	E
V	V	AV	AV	V	AV	AV	AV	AV	AV	AV	AV	C

Exploration d'un graphe
Parcours en profondeur (3)

On considère le graphe donné par les listes d'adjacence suivantes :

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	sommet
AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	A
V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	B
V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	E
V	V	AV	AV	V	AV	AV	AV	AV	AV	AV	AV	C
V	V	V	AV	V	AV	AV	AV	AV	AV	AV	AV	G

Exploration d'un graphe
Parcours en profondeur (3)

On considère le graphe donné par les listes d'adjacence suivantes :

A : (B,C,D)
B : (E,F)
C : (G,H,I)
D : (B,F)
E : (A,C)
F : (G,I)
G : (E)
H : (F,G)
I : (D)
J : (K,L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	sommet
AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	A
V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	B
V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	E
V	V	AV	AV	V	AV	AV	AV	AV	AV	AV	AV	C
V	V	V	AV	V	AV	AV	AV	AV	AV	AV	AV	G
V	V	V	AV	V	AV	V	AV	AV	AV	AV	AV	H

Exploration d'un graphe
Parcours en profondeur (3)

On considère le graphe donné par les listes d'adjacence suivantes :

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	sommet
AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	A
V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	B
V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	E
V	V	AV	AV	V	AV	AV	AV	AV	AV	AV	AV	C
V	V	V	AV	V	AV	AV	AV	AV	AV	AV	AV	G
V	V	V	AV	V	AV	V	AV	AV	AV	AV	AV	H
V	V	V	AV	V	AV	V	V	AV	AV	AV	AV	F

Exploration d'un graphe
Parcours en profondeur (3)

On considère le graphe donné par les listes d'adjacence suivantes :

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	sommet
AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	A
V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	B
V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	E
V	V	AV	AV	V	AV	AV	AV	AV	AV	AV	AV	C
V	V	V	AV	V	AV	AV	AV	AV	AV	AV	AV	G
V	V	V	AV	V	AV	V	AV	AV	AV	AV	AV	H
V	V	V	AV	V	AV	V	V	AV	AV	AV	AV	F
V	V	V	AV	V	V	V	V	AV	AV	AV	AV	I

Exploration d'un graphe
Parcours en profondeur (3)

On considère le graphe donné par les listes d'adjacence suivantes :

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	sommet
AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	A
V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	B
V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	E
V	V	AV	AV	V	AV	AV	AV	AV	AV	AV	AV	C
V	V	V	AV	V	AV	AV	AV	AV	AV	AV	AV	G
V	V	V	AV	V	AV	V	AV	AV	AV	AV	AV	H
V	V	V	AV	V	AV	V	V	AV	AV	AV	AV	F
V	V	V	AV	V	V	V	V	AV	AV	AV	AV	I
V	V	V	AV	V	V	V	V	V	AV	AV	AV	D

Exploration d'un graphe
Parcours en profondeur (3)

On considère le graphe donné par les listes d'adjacence suivantes :

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	sommet
AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	A
V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	B
V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	E
V	V	AV	AV	V	AV	AV	AV	AV	AV	AV	AV	C
V	V	V	AV	V	AV	AV	AV	AV	AV	AV	AV	G
V	V	V	AV	V	AV	V	AV	AV	AV	AV	AV	H
V	V	V	AV	V	AV	V	V	AV	AV	AV	AV	F
V	V	V	AV	V	V	V	V	AV	AV	AV	AV	I
V	V	V	AV	V	V	V	V	V	AV	AV	AV	D
V	V	V	V	V	V	V	V	V	AV	AV	AV	J

Exploration d'un graphe
Parcours en profondeur (3)

On considère le graphe donné par les listes d'adjacence suivantes :

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	sommet
AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	A
V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	B
V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	E
V	V	AV	AV	V	AV	AV	AV	AV	AV	AV	AV	C
V	V	V	AV	V	AV	AV	AV	AV	AV	AV	AV	G
V	V	V	AV	V	AV	V	AV	AV	AV	AV	AV	H
V	V	V	AV	V	AV	V	V	AV	AV	AV	AV	F
V	V	V	AV	V	V	V	V	AV	AV	AV	AV	I
V	V	V	AV	V	V	V	V	V	AV	AV	AV	D
V	V	V	V	V	V	V	V	V	AV	AV	AV	J
V	V	V	V	V	V	V	V	V	V	AV	AV	K

Exploration d'un graphe
Parcours en profondeur (3)

On considère le graphe donné par les listes d'adjacence suivantes :

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	sommet
AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	A
V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	B
V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	E
V	V	AV	AV	V	AV	AV	AV	AV	AV	AV	AV	C
V	V	V	AV	V	AV	AV	AV	AV	AV	AV	AV	G
V	V	V	AV	V	AV	V	AV	AV	AV	AV	AV	H
V	V	V	AV	V	AV	V	V	AV	AV	AV	AV	F
V	V	V	AV	V	V	V	V	AV	AV	AV	AV	I
V	V	V	AV	V	V	V	V	V	AV	AV	AV	D
V	V	V	V	V	V	V	V	V	AV	AV	AV	J
V	V	V	V	V	V	V	V	V	V	AV	AV	K
V	V	V	V	V	V	V	V	V	V	V	AV	L

Exploration d'un graphe
Parcours en profondeur (3)

On considère le graphe donné par les listes d'adjacence suivantes :

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	sommet
AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	A
V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	B
V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	E
V	V	AV	AV	V	AV	AV	AV	AV	AV	AV	AV	C
V	V	V	AV	V	AV	AV	AV	AV	AV	AV	AV	G
V	V	V	AV	V	AV	V	AV	AV	AV	AV	AV	H
V	V	V	AV	V	AV	V	V	AV	AV	AV	AV	F
V	V	V	AV	V	V	V	V	AV	AV	AV	AV	I
V	V	V	AV	V	V	V	V	V	AV	AV	AV	D
V	V	V	V	V	V	V	V	V	AV	AV	AV	J
V	V	V	V	V	V	V	V	V	V	AV	AV	K
V	V	V	V	V	V	V	V	V	V	V	AV	L
V	V	V	V	V	V	V	V	V	V	V	V	

Exploration d'un graphe
Parcours en largeur (1)

Une autre technique consiste lors de la visite d'un sommet à visiter ensuite tous les sommets qui lui sont adjacents : c'est le **parcours en largeur d'abord**.

Pour cela, on peut utiliser une **file d'attente** dans laquelle on met tous les sommets adjacents au sommet en cours de visite.

Exploration d'un graphe
Parcours en largeur (2)

```
Algorithme parcoursEnLargeur(G : graphe)

début
    f ← emptyQueue( )
    pour chaque sommet s de S  faire
        marque[s] ← A_VOIR
    fpour
    pour chaque sommet s de S faire
        si marque[s] = A_VOIR :
            visiteEnLargeur(s)
        fsi
    fpour
fin
```

Exploration d'un graphe
Parcours en largeur (3)

```
Procédure visiteEnLargeur(s : Sommet)
début
    marque[s] ← VU
    enQueue(s, f)
    Tant que non isEmptyQueue(f) faire
        x ← getHead(f)
        deQueue(f)
        Traitement(x)
        pour chaque sommet t adjacent à x faire
            si marque[t] = A_VOIR alors
                marque[t] ← VU
                enQueue(t, f)
            fsi
        fpour
    ftq
fin
```

Exploration d'un graphe
Parcours en largeur (4)

A : (B,C,D)
B : (E,F)
C : (G,H,I)
D : (B,F)
E : (A,C)
F : (G,I)
G : (E)
H : (F,G)
I : (D)
J : (K,L)
K : ( )
L : (K)

Comme dans le cas du parcours en profondeur, il est plus judicieux de représenter le graphe à l'aide de listes d'adjacence.

Code couleur : A dans appel parcoursEnLargeur
A dans appel visiteEnLargeur
A marque de A mise à VU

Exploration d'un graphe
Parcours en largeur (4)

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	file
AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	

Exploration d'un graphe
Parcours en largeur (4)

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	file
AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	
V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	A

Exploration d'un graphe
Parcours en largeur (4)

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	file
AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	
V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	A
V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	B

Exploration d'un graphe
Parcours en largeur (4)

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	file
AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	
V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	A
V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	B
V	V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	B, C

Exploration d'un graphe
Parcours en largeur (4)

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	file
AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	
V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	A
V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	B
V	V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	B, C
V	V	V	V	AV	AV	AV	AV	AV	AV	AV	AV	B, C, D

Exploration d'un graphe
Parcours en largeur (4)

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	file
AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	
V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	A
V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	B
V	V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	B, C
V	V	V	V	AV	AV	AV	AV	AV	AV	AV	AV	B, C, D
V	V	V	V	V	AV	AV	AV	AV	AV	AV	AV	C, D, E

Exploration d'un graphe
Parcours en largeur (4)

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	file
AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	
V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	A
V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	B
V	V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	B, C
V	V	V	V	AV	AV	AV	AV	AV	AV	AV	AV	B, C, D
V	V	V	V	V	AV	AV	AV	AV	AV	AV	AV	C, D, E
V	V	V	V	V	V	AV	AV	AV	AV	AV	AV	C, D, E, F

Exploration d'un graphe
Parcours en largeur (4)

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	file
AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	
V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	A
V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	B
V	V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	B, C
V	V	V	V	AV	AV	AV	AV	AV	AV	AV	AV	B, C, D
V	V	V	V	V	AV	AV	AV	AV	AV	AV	AV	C, D, E
V	V	V	V	V	V	AV	AV	AV	AV	AV	AV	C, D, E, F
V	V	V	V	V	V	V	AV	AV	AV	AV	AV	D, E, F, G

Exploration d'un graphe
Parcours en largeur (4)

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	file
AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	
V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	A
V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	B
V	V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	B, C
V	V	V	V	AV	AV	AV	AV	AV	AV	AV	AV	B, C, D
V	V	V	V	V	AV	AV	AV	AV	AV	AV	AV	C, D, E
V	V	V	V	V	V	AV	AV	AV	AV	AV	AV	C, D, E, F
V	V	V	V	V	V	V	AV	AV	AV	AV	AV	D, E, F, G
V	V	V	V	V	V	V	V	AV	AV	AV	AV	D, E, F, G, H

Exploration d'un graphe
Parcours en largeur (4)

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	file
AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	
V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	A
V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	B
V	V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	B, C
V	V	V	V	AV	AV	AV	AV	AV	AV	AV	AV	B, C, D
V	V	V	V	V	AV	AV	AV	AV	AV	AV	AV	C, D, E
V	V	V	V	V	V	AV	AV	AV	AV	AV	AV	C, D, E, F
V	V	V	V	V	V	V	AV	AV	AV	AV	AV	D, E, F, G
V	V	V	V	V	V	V	V	AV	AV	AV	AV	D, E, F, G, H
V	V	V	V	V	V	V	V	V	AV	AV	AV	D, E, F, G, H, I

Exploration d'un graphe
Parcours en largeur (4)

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	file
AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	
V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	A
V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	B
V	V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	B, C
V	V	V	V	AV	AV	AV	AV	AV	AV	AV	AV	B, C, D
V	V	V	V	V	AV	AV	AV	AV	AV	AV	AV	C, D, E
V	V	V	V	V	V	AV	AV	AV	AV	AV	AV	C, D, E, F
V	V	V	V	V	V	V	AV	AV	AV	AV	AV	D, E, F, G
V	V	V	V	V	V	V	V	AV	AV	AV	AV	D, E, F, G, H
V	V	V	V	V	V	V	V	V	AV	AV	AV	D, E, F, G, H, I
V	V	V	V	V	V	V	V	V	AV	AV	AV	E, F, G, H, I

Exploration d'un graphe
Parcours en largeur (4)

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	file
AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	
V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	A
V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	B
V	V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	B, C
V	V	V	V	AV	AV	AV	AV	AV	AV	AV	AV	B, C, D
V	V	V	V	V	AV	AV	AV	AV	AV	AV	AV	C, D, E
V	V	V	V	V	V	AV	AV	AV	AV	AV	AV	C, D, E, F
V	V	V	V	V	V	V	AV	AV	AV	AV	AV	D, E, F, G
V	V	V	V	V	V	V	V	AV	AV	AV	AV	D, E, F, G, H
V	V	V	V	V	V	V	V	V	AV	AV	AV	D, E, F, G, H, I
V	V	V	V	V	V	V	V	V	AV	AV	AV	E, F, G, H, I
V	V	V	V	V	V	V	V	V	AV	AV	AV	F, G, H, I

Exploration d'un graphe
Parcours en largeur (4)

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	file
AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	
V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	A
V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	B
V	V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	B, C
V	V	V	V	AV	AV	AV	AV	AV	AV	AV	AV	B, C, D
V	V	V	V	V	AV	AV	AV	AV	AV	AV	AV	C, D, E
V	V	V	V	V	V	AV	AV	AV	AV	AV	AV	C, D, E, F
V	V	V	V	V	V	V	AV	AV	AV	AV	AV	D, E, F, G
V	V	V	V	V	V	V	V	AV	AV	AV	AV	D, E, F, G, H
V	V	V	V	V	V	V	V	V	AV	AV	AV	D, E, F, G, H, I
V	V	V	V	V	V	V	V	V	AV	AV	AV	E, F, G, H, I
V	V	V	V	V	V	V	V	V	AV	AV	AV	F, G, H, I
V	V	V	V	V	V	V	V	V	AV	AV	AV	G, H, I

Exploration d'un graphe
Parcours en largeur (4)

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	file
AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	
V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	A
V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	B
V	V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	B, C
V	V	V	V	AV	AV	AV	AV	AV	AV	AV	AV	B, C, D
V	V	V	V	V	AV	AV	AV	AV	AV	AV	AV	C, D, E
V	V	V	V	V	V	AV	AV	AV	AV	AV	AV	C, D, E, F
V	V	V	V	V	V	V	AV	AV	AV	AV	AV	D, E, F, G
V	V	V	V	V	V	V	V	AV	AV	AV	AV	D, E, F, G, H
V	V	V	V	V	V	V	V	V	AV	AV	AV	D, E, F, G, H, I
V	V	V	V	V	V	V	V	V	AV	AV	AV	E, F, G, H, I
V	V	V	V	V	V	V	V	V	AV	AV	AV	F, G, H, I
V	V	V	V	V	V	V	V	V	AV	AV	AV	G, H, I
V	V	V	V	V	V	V	V	V	AV	AV	AV	H, I

Exploration d'un graphe
Parcours en largeur (4)

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	file
AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	
V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	A
V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	B
V	V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	B, C
V	V	V	V	AV	AV	AV	AV	AV	AV	AV	AV	B, C, D
V	V	V	V	V	AV	AV	AV	AV	AV	AV	AV	C, D, E
V	V	V	V	V	V	AV	AV	AV	AV	AV	AV	C, D, E, F
V	V	V	V	V	V	V	AV	AV	AV	AV	AV	D, E, F, G
V	V	V	V	V	V	V	V	AV	AV	AV	AV	D, E, F, G, H
V	V	V	V	V	V	V	V	V	AV	AV	AV	D, E, F, G, H, I
V	V	V	V	V	V	V	V	V	AV	AV	AV	E, F, G, H, I
V	V	V	V	V	V	V	V	V	AV	AV	AV	F, G, H, I
V	V	V	V	V	V	V	V	V	AV	AV	AV	G, H, I
V	V	V	V	V	V	V	V	V	AV	AV	AV	H, I
V	V	V	V	V	V	V	V	V	AV	AV	AV	I

Exploration d'un graphe
Parcours en largeur (4)

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	file
AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	
V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	A
V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	AV	B
V	V	V	AV	AV	AV	AV	AV	AV	AV	AV	AV	B, C
V	V	V	V	AV	AV	AV	AV	AV	AV	AV	AV	B, C, D
V	V	V	V	V	AV	AV	AV	AV	AV	AV	AV	C, D, E
V	V	V	V	V	V	AV	AV	AV	AV	AV	AV	C, D, E, F
V	V	V	V	V	V	V	AV	AV	AV	AV	AV	D, E, F, G
V	V	V	V	V	V	V	V	AV	AV	AV	AV	D, E, F, G, H
V	V	V	V	V	V	V	V	V	AV	AV	AV	D, E, F, G, H, I
V	V	V	V	V	V	V	V	V	AV	AV	AV	E, F, G, H, I
V	V	V	V	V	V	V	V	V	AV	AV	AV	F, G, H, I
V	V	V	V	V	V	V	V	V	AV	AV	AV	G, H, I
V	V	V	V	V	V	V	V	V	AV	AV	AV	H, I
V	V	V	V	V	V	V	V	V	AV	AV	AV	I
V	V	V	V	V	V	V	V	V	AV	AV	AV	

Exploration d'un graphe
Parcours en largeur (4)

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	file
V	V	V	V	V	V	V	V	V	V	AV	AV	J

Exploration d'un graphe
Parcours en largeur (4)

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	file
V	V	V	V	V	V	V	V	V	V	AV	AV	J
V	V	V	V	V	V	V	V	V	V	V	AV	K

Exploration d'un graphe
Parcours en largeur (4)

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	file
V	V	V	V	V	V	V	V	V	V	AV	AV	J
V	V	V	V	V	V	V	V	V	V	V	AV	K
V	V	V	V	V	V	V	V	V	V	V	V	K, L

Exploration d'un graphe
Parcours en largeur (4)

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	file
V	V	V	V	V	V	V	V	V	V	AV	AV	J
V	V	V	V	V	V	V	V	V	V	V	AV	K
V	V	V	V	V	V	V	V	V	V	V	V	K, L
V	V	V	V	V	V	V	V	V	V	V	V	L

Exploration d'un graphe
Parcours en largeur (4)

A : (B, C, D)
B : (E, F)
C : (G, H, I)
D : (B, F)
E : (A, C)
F : (G, I)
G : (E)
H : (F, G)
I : (D)
J : (K, L)
K : ( )
L : (K)

A	B	C	D	E	F	G	H	I	J	K	L	file
V	V	V	V	V	V	V	V	V	V	AV	AV	J
V	V	V	V	V	V	V	V	V	V	V	AV	K
V	V	V	V	V	V	V	V	V	V	V	V	K, L
V	V	V	V	V	V	V	V	V	V	V	V	L
V	V	V	V	V	V	V	V	V	V	V	V	

Exploration d'un graphe

Exercices

Exercice 7 : on considère le graphe donné par les listes d'adjacence suivantes:

1 : (2, 5, 10, 11)
2 : (1, 2, 4)
3 : (4, 6, 8)
4 : ( )
5 : (3, 4, 7, 9)
6 : (3, 8)
7 : (6)
8 : (2, 10, 11)
9 : (1, 4, 7)
10 : ( )
11 : (2, 7)

Réalisez un parcours en profondeur d'abord et un parcours en largeur d'abord. On utilisera l'ordre sur les entiers comme ordre de parcours.

Utiliser les graphes pour résoudre un problème classique de Bioinformatique :

l'assemblage *de novo*

Les graphes de de Bruijn

Graphes de de Bruijn
décrits par Camille Flye Sainte-Marie en 1894
puis par de Bruijn et Irving J. Good en 1946

Des logiciels comme Velvet, AbySS ou Euler utilisent les graphes de de Bruijn dans le problème de l'assemblage de courtes séquences pour construire des contigs.

Les k-mers

Un **k-mer** d'une séquence s est un facteur de s de longueur k.

Exercice 8 : soit s = GGCGATTCA, calculez tous les 4-mers de s. Combien y en a-t-il ? Peut-on généraliser ?

Exercice 9 : on considère maintenant un ensemble de fragments F = { TACAGT, CAGTC, AGTCA, CAGA } issus du séquencage la séquence s = TACAGTCAGA.
Combien peut-on construire de 3-mers à partir des éléments de F ?
Combien de 3-mers distincts ?
Quelle est la plus grande valeur de k telle que les k-mers de F sont exactement les k-mers des fragments de F ?

Les k-mers, solution

Un **k-mer** d'une séquence s est un facteur de s de longueur k.

Exercice 8 : soit s = GGCGATTCA, calculez tous les 4-mers de s.
Combien y en a-t-il ? 6
Peut-on généraliser ? Une séquence de longueur n a n–k+1 k-mers.

Exercice 9 : on considère maintenant un ensemble de fragments F = { TACAGT, CAGTC, AGTCA, CAGA } issus du séquencage la séquence s = TACAGTCAGA.
Combien peut-on construire de 3-mers à partir des éléments de F ?
(6-3+1) + (5-3+1) * 2 + (4-3+1) = 4 + 6 + 2 = 12
Combien de 3-mers distincts ? 7, CAG et AGT apparaissent 3 fois et GTC 2 fois.
Quelle est la plus grande valeur de k telle que les k-mers de s sont exactement les k-mers des fragments de F ?
La plus grande valeur de k pourrait être 4 à cause du fragment CAGA mais il manque alors le 4-mer TCAG. Pour k = 3, les k-mers sont identiques.

Graphes de de Bruijn et assemblage (1)

On considère un ensemble de reads R et un entier k fixé.

Le **graphe de de Bruijn** pour R et k est le couple (S, A) où

- Les sommets de S sont tous les k-mers apparaissant dans les reads.

- Pour tout (k + 1) - mer w apparaissant dans un read, il existe un arc du sommet u vers le sommet v où u est le préfixe de longueur k de w et v le suffixe de longueur k.

Graphes de de Bruijn et assemblage (2)

Exercice 10 : construire le graphe de de Bruijn pour l'ensemble de reads R = { ACTG, CTGC, TGCT } et k = 3.

Exercice 11 : que se passe-t-il lorsque certains reads apparaissent plusieurs fois comme dans le multi-ensemble R = { ACTG, ACTG, CTGC, CTGC, CTGC, TGCT, TGCT } ?

Exercice 12 : que se passe-t-il lorsque certains reads comportent des erreurs de séquençage comme dans R = { ACTG, CTGC, CTGA, TGCT } ?

Exercice 13 : que se passe-t-il lorsque la séquence d'origine comporte de courtes répétitions comme dans R = { ACTG, CTGC, TGCT, GCTG, CTGA, TGAC } ?

Graphes de de Bruijn et construction des contigs

Le principe de la construction de contigs à partir d'un graphe de reads est de trouver les plus longs chemins simples du graphe qui ne partagent pas de sommet.

Algorithme de construction des contigs

Tant qu'il existe des sommets non sélectionnés
    Sélectionner un sommet s.
    Si ce sommet a un unique successeur, que ce successeur a au maximum un prédécesseur et que ce successeur n'a jamais été sélectionné, le sélectionner et recommencer cette étape à partir de ce sommet.
    Recommencer, cette fois dans l'autre sens (avec les prédécesseurs), à partir de s.

Exercice 14 : soit R = {TAC, ACA, CAG, AGT, GTC, AGA, TCA}. Construisez le graphe de de Bruijn pour R et k = 2. Combien produit-on de contigs à partir de ce graphe ?

Graphes de de Bruijn et construction des contigs

Exercice 14 : soit R = {TAC, ACA, CAG, AGT, GTC, AGA, TCA}. Construisez le graphe de de Bruijn pour R et k = 2. Combien produit-on de contigs à partir de ce graphe ?

4 si on part de CA : (CAG, TAC, GTC, GA)

5 si on part de TC ou de TA : (GTC, TAC, CA, AG, GA)

Le problème de la plus courte sur-séquence commune (SSC)

Le problème de la plus courte sur-séquence commune (SSC) d'un ensemble de fragments est une modélisation du problème de l'assemblage.

Soit F une collection de fragments d'ADN. Il s'agit de trouver une séquence S de longueur minimale telle que tout fragment f de F soit un facteur de S. Par exemple, si F = {CTA, ACT, AGT}, la séquence S = ACTAGT est la plus courte sur-séquence commune de F.

On supposera, sans perte de généralité, qu'aucun fragment de F n'est lui-même un facteur d'un autre fragment de F. En effet, si u est facteur d'un autre fragment de F, alors une solution pour $F - \{ u \}$ est également une solution pour F.

Les *recouvrements* (overlaps) entre les fragments de F jouent évidemment un rôle majeur dans la solution de ce problème.

SSC

Question 1

Proposez une structure de graphe pondéré pour représenter une collection *F* et les recouvrements entre les fragments de *F*.
Dans ce graphe pondéré, comment s'exprime le problème de la recherche d'une plus courte SSC ? Connaissez-vous un problème apparenté ?

Question 2

Proposez un algorithme glouton *G* pour trouver une « courte » SSC.

Question 3

Dans certains cas, on est « presque sûr » que les recouvrements pertinents sont de taille supérieure à un certain seuil *t*. On peut alors retirer du graphe des recouvrements les arcs de poids inférieur à *t*.
Écrivez un algorithme construisant le graphe des recouvrements à partir d'une collection de fragments.

Diapos supplémentaires

Primitives files d'attentes

Queues

Interface

emptyQueue() : Queue returns an empty queue

getHead(q: Queue) : Dreturns the element at the head of queue q, $q$should not be empty

enqueue(x: D, q: Queue) inserts element $x$at the end of queue q

dequeue(q: Queue) deletes element at the head of queue q, q should not be empty

isEmptyQueue(q: Queue) : boolean tests if queue $q$is empty