LICENCE D'INFORMATIQUE

Sorbonne Université

LU3IN003 – Algorithmique

Cours 4 : Rappels sur les graphes

Année 2023-2024

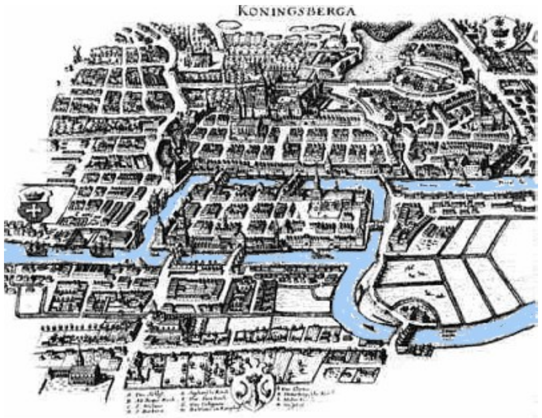
Responsables et chargés de cours

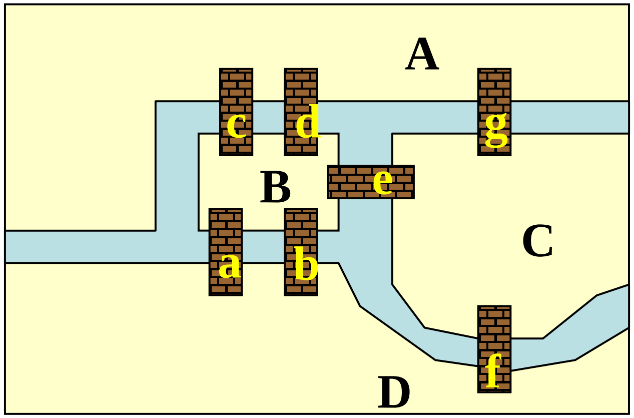
Fanny Pascual

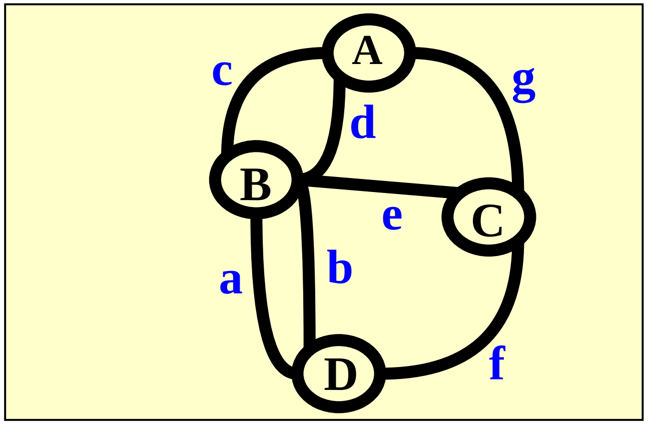
Olivier Spanjaard

Les ponts de Koenigsberg (Euler, 1736)

Trouver une promenade qui permet de passer une fois et une seule sur chaque pont en revenant au point de départ (***cf. LU2IN003 !***).







Représentation des ponts : arêtes

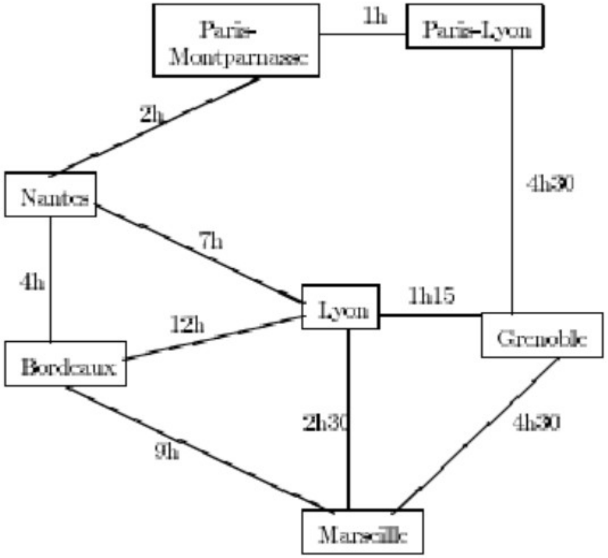
Problème posé : Existe-t-il un cycle passant par A empruntant une fois et une seule chaque arête ?

Un tel cycle est appelé un cycle eulérien.

Choix d'un itinéraire

Connaissant la durée des trajets suivants, comment faire pour aller le plus rapidement de Bordeaux vers Grenoble ?

Bordeaux → Nantes 4 h
Bordeaux → Marseille 9 h
Bordeaux → Lyon 12 h
Nantes → Paris-Montparnasse 2 h
Nantes → Lyon 7 h
Paris Montparnasse → Paris Lyon 1 h (en autobus)
Paris-Lyon → Grenoble 4 h 30
Marseille → Lyon 2 h 30
Marseille → Grenoble 4 h 30
Lyon → Grenoble 1 h 15



Cela revient à déterminer un plus court chemin de Bordeaux vers Grenoble dans le graphe de droite.

Emploi du temps

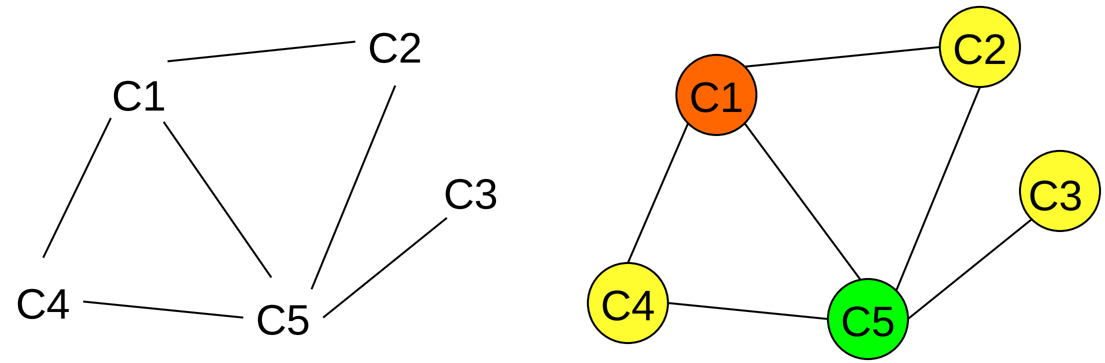
- Ensemble de cours à planifier C1, C2, … Cn.
- Tous les cours ont la même durée (1 heure) et sont indivisibles.
- Certains cours ne peuvent pas s’exécuter simultanément.

Quel est le nombre d’heures minimum nécessaire pour la planification de tous les cours ?

Graphe non orienté

Sommets Cours

Arêtes Disjonction



Problème posé : Quel est le nombre de couleurs d’une coloration du graphe de cardinalité minimale ?

Nombre d’heures min = nombre chromatique du graphe

Applications des graphes

L'algorithmique des graphes, et plus généralement l'optimisation combinatoire, a de très nombreuses applications (liste très loin d'être exhaustive !) :

- Web : itinéraires dans Google maps, étude du « graphe du web »...
- GPS : logiciels de détermination d'itinéraires
- Gestion de production : optimisation de l'ordonnancement
- Compagnies aériennes : planification des vols, itinéraires, plannings du personnel aérien...
- Telecoms : affectation de fréquence en téléphonie mobile, organisation des réseaux...
- SNCF : optimisation des horaires des trains, emplois du temps...
- Armée : planification stratégique
- Finance : optimisation de portefeuille

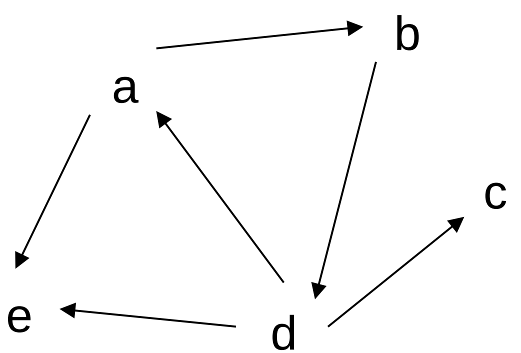
Représentations des graphes

Matrice B=(b_{st}) booléenne d’adjacence:

indices des lignes = sommets de G,

indices des colonnes = sommets de G,

b_{st}=1 si a=(s,t) $\in$ A, 0 sinon.



graphe G

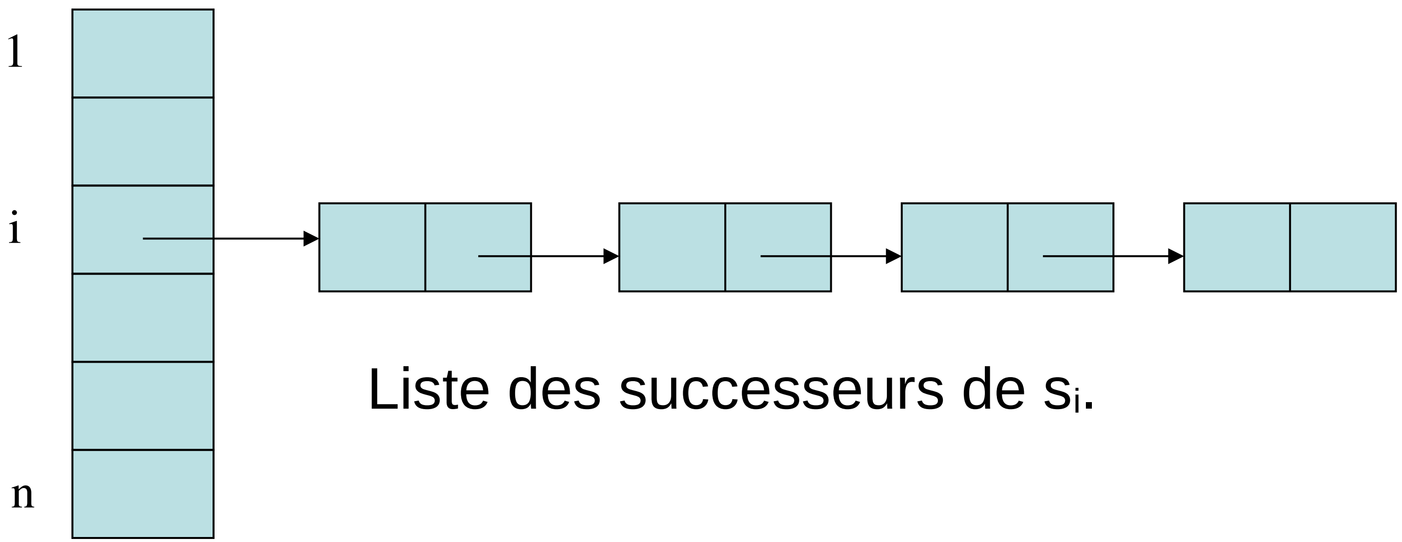
	a	b	c	d	e
a		1			1
b				1	
c					
d	1		1		1
e					

B

Inconvénient (si G non dense) : place mémoire : O(n^2)

Avantage test d’existence de l’arc (s_i,s_j) : O(1)

2ème représentation : tableau des listes de successeurs

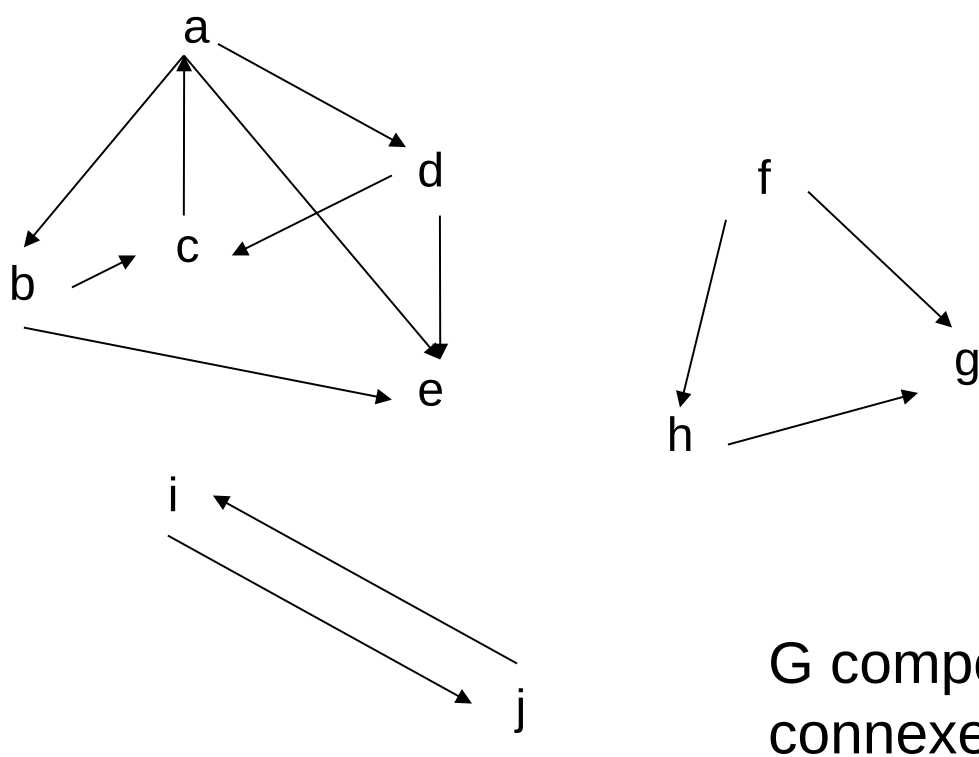


Avantage : place mémoire : O(n+m)

Inconvénient : test d'existence d'un arc a =(s_i,s_j) : $O(d^+(s_i))$

Composantes connexes

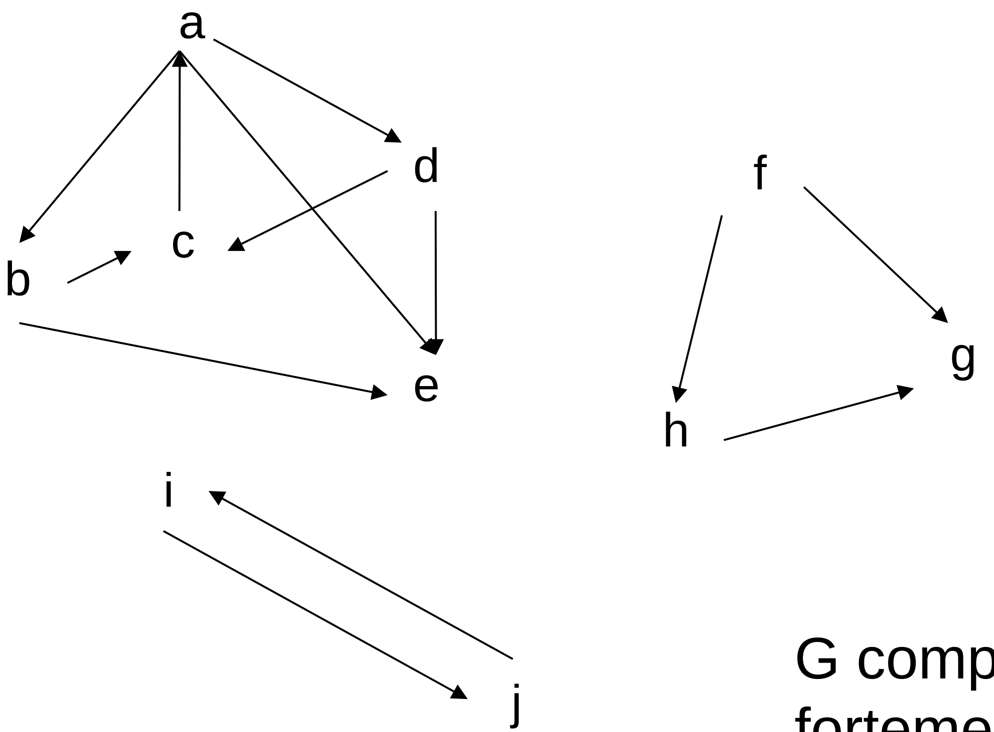
Composante connexe = sous-graphe induit maximal connexe.



G comporte 3 composantes connexes.

Composantes fortement connexes

Composante connexe = sous-graphe induit maximal ***fortement*** connexe.



G comporte 7 composantes fortement connexes : {a,c,d}, {b}, {e}, {f}, {g}, {h}, {i,j}.

Graphe réduit d'un graphe G=(S,A).

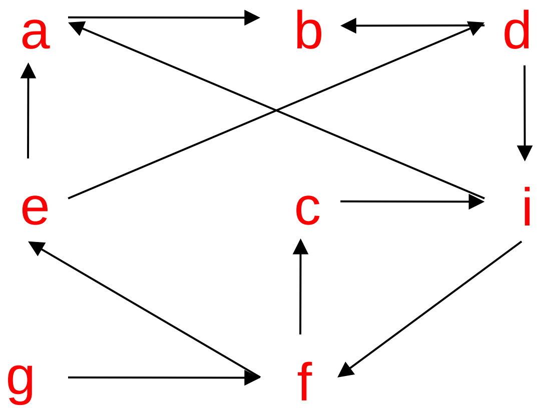
Sommets : composantes fortement connexes $\{C_1, C_2, \ldots, C_p\}$ de G

Arcs : (C_i,C_j) est un arc si

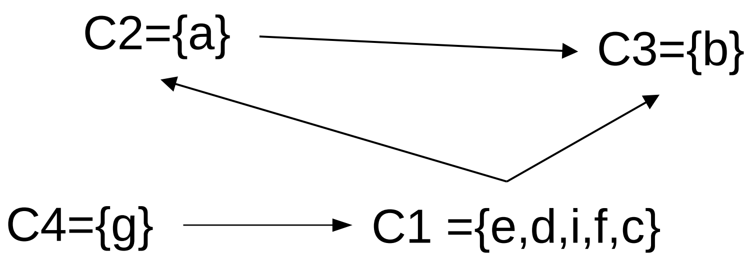
a) $i \neq j$ et

b) il existe $a \in A$ tel que $a^- \in C_i$ et $a^+ \in C_j$

graphe G :



graphe réduit de G :



Propriété : Un graphe réduit est sans circuit.

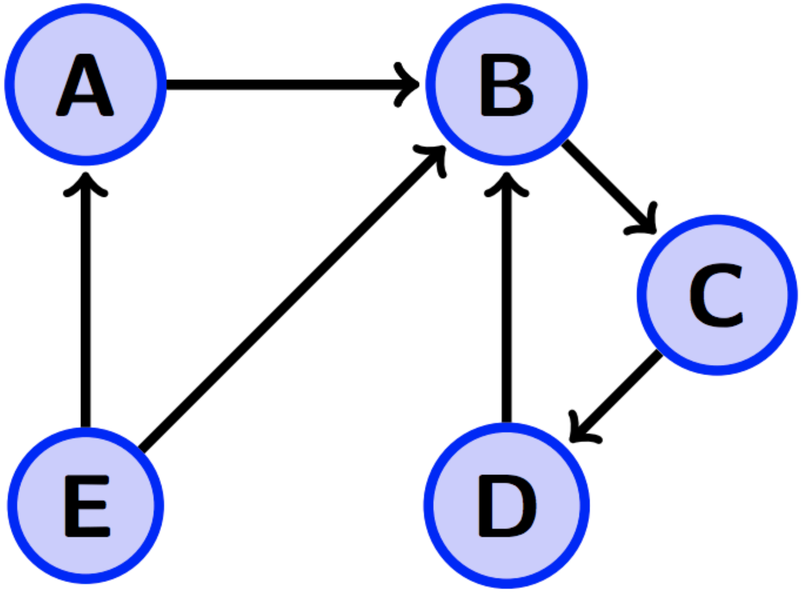
Preuve :

S'il existait un circuit dans le graphe réduit, les composantes fortement connexes du circuit appartiendraient à une même composante fortement connexe. Contradiction.

Quiz : Composantes fortement connexes

Quel est le nombre de composantes fortement connexes dans le graphe ?

A) 1
B) 2
C) 3
D) 5



Quiz : Graphe réduit

Soit G un graphe fortement connexe et soit G_R le graphe réduit de G. Laquelle de ces propositions est fausse ?

A) G_R ne contient pas de circuit.

B) Le nombre de sommets de G_R dépend du nombre de sommets de G.

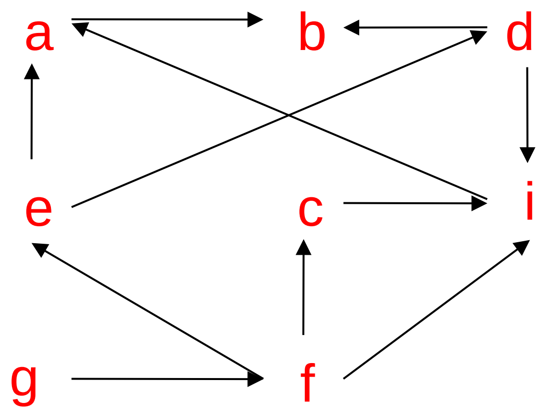
C) G_R contient un seul sommet.

D) G_R a autant de composantes fortement connexes que de sommets.

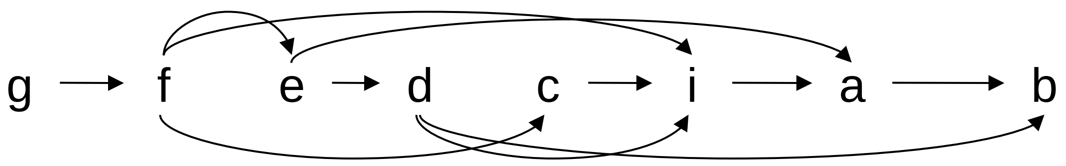
Graphes particuliers

Graphe sans circuit

Propriété (liste topologique des sommets) :
Soit G=(S,A) un graphe sans circuit.
Il existe une liste (s_1, s_2, … , s_n) des sommets de G telle que pour tout arc (s_i,s_j) : i < j



graphe G sans circuit



liste topologique des sommets de G :
(g,f,e,d,c,i,a,b)

Arbre

Remarque :

Le sous-graphe induit par chaque composante connexe d’un graphe sans cycle est connexe et sans cycle* (arbre).

* sans cycle élémentaire de taille supérieure ou égale à 3.

Définitions équivalentes d’un arbre:

D_0: graphe connexe sans cycle

D_1: graphe connexe à n-1 arêtes

D_2: graphe sans cycle à n-1 arêtes

D_3: graphe t.q. il existe une chaîne unique entre toute paire de sommets

D_4: graphe connexe, qui devient non connexe par suppression d’une arête quelconque

D_5: graphe sans cycle, création d’un cycle unique par ajout d’une arête quelconque

Quiz : Choix d'un algorithme

Soit A et B deux algorithmes résolvant le même problème pour un graphe non-orienté G. A est en O(n+m) et B est en O(n log n). Soit G_1 un arbre et G_2 un graphe complet.

Quel est le meilleur choix parmi les suivants ?

A) Utiliser A sur G_1 et utiliser B sur G_2.

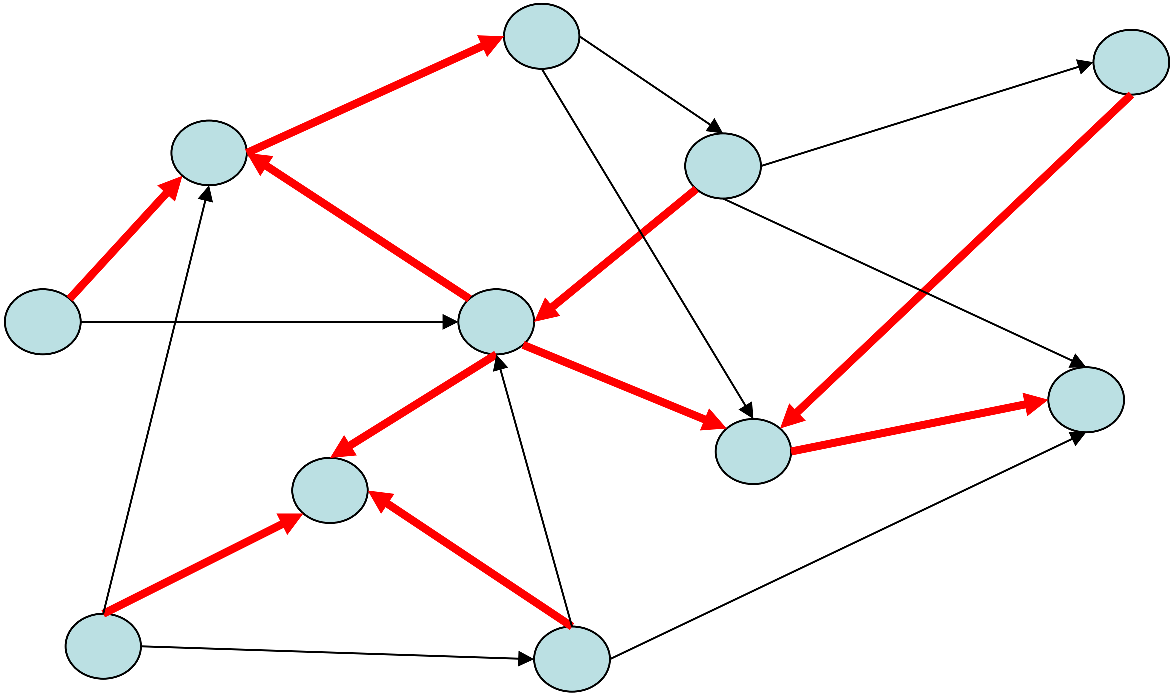
B) Utiliser B sur G_1 et utiliser A sur G_2.

C) Utiliser A sur G_1 et utiliser A sur G_2.

D) Utiliser B sur G_1 et utiliser B sur G_2.

Arbre couvrant

Soit G=(S,A) un graphe.
Un arbre couvrant de G est un graphe partiel de G qui est un arbre.



Arcs rouges :
arbre couvrant H

Arcs noirs :
co-arbre de H

Propriété 1:

Un graphe G possède un arbre couvrant si et seulement si il est connexe.

Preuve :

Si G possède un arbre couvrant, alors G est connexe.

Si G est connexe, on construit un graphe partiel par l'algorithme suivant :

```
H:=G;
Tant qu'il existe un cycle dans H faire
    Supprimer de H une arête quelconque du cycle
Fin Tant que;
```

Lors de la terminaison, le graphe partiel H est connexe et sans cycle. C'est un arbre couvrant de G.

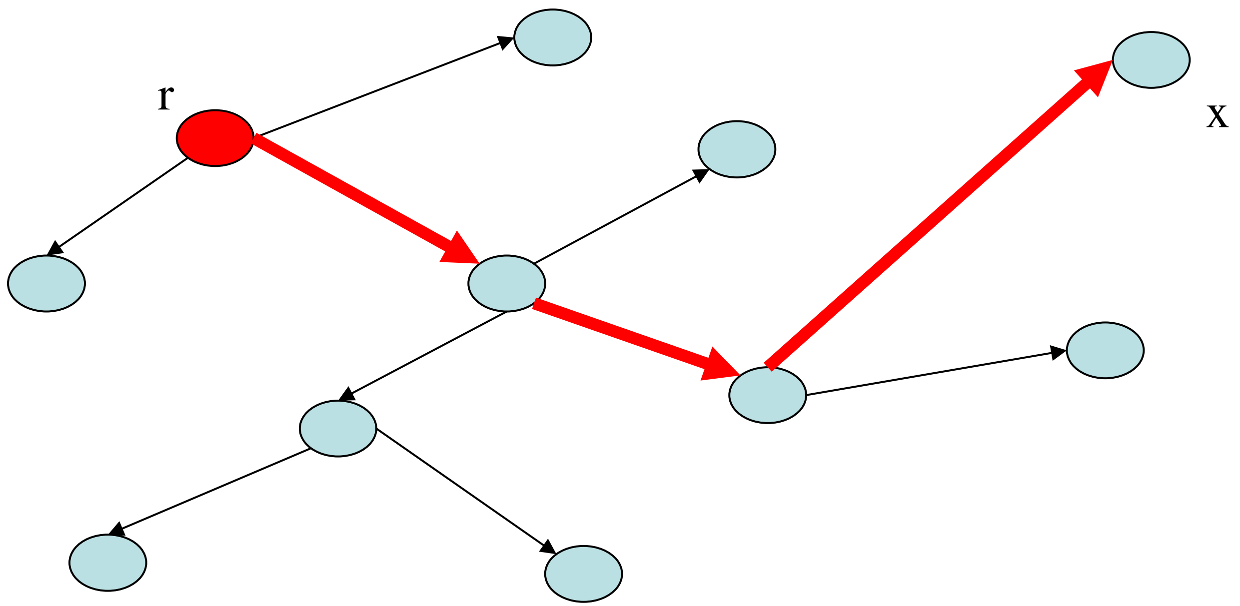
Des arbres particuliers : les arborescences

Arborescence:

Arbre tel que :

- un sommet r est distingué (la racine)
- pour tout sommet x de l'arbre, la chaîne de r à x est un chemin.

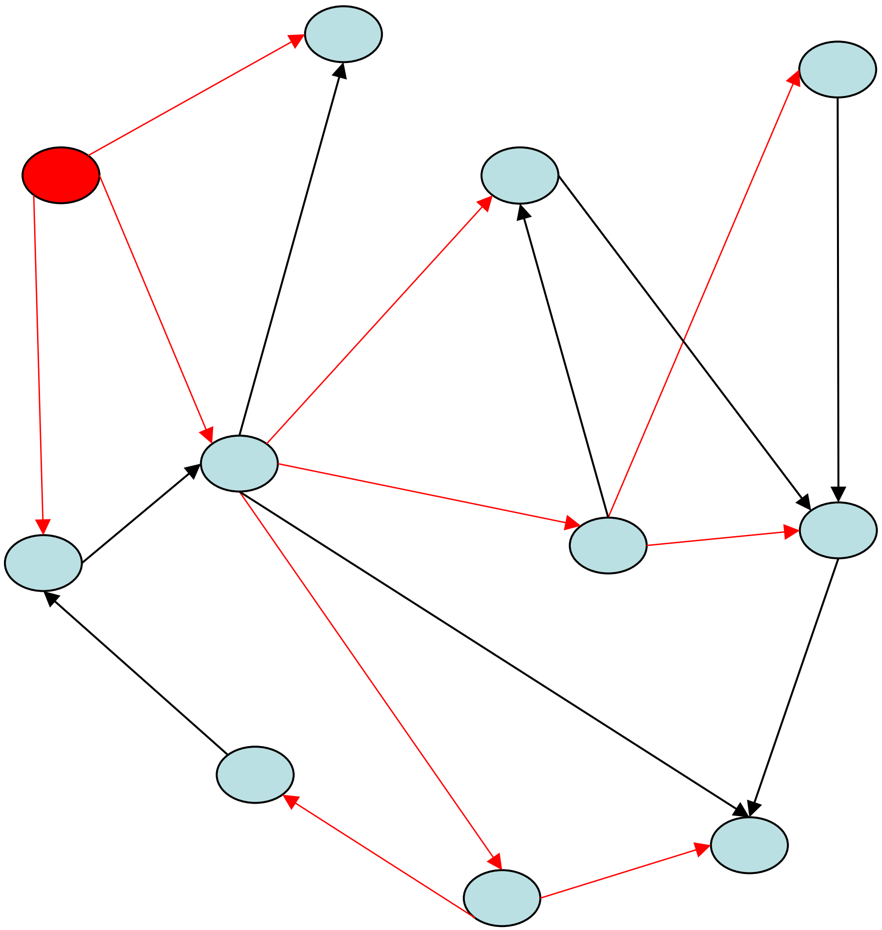
Une arborescence



Arborescence couvrante

- Une arborescence couvrante de G est un graphe partiel de G qui est une arborescence

- Le sommet s est une racine de G si pour tout sommet x de G, il existe un chemin de s à x.

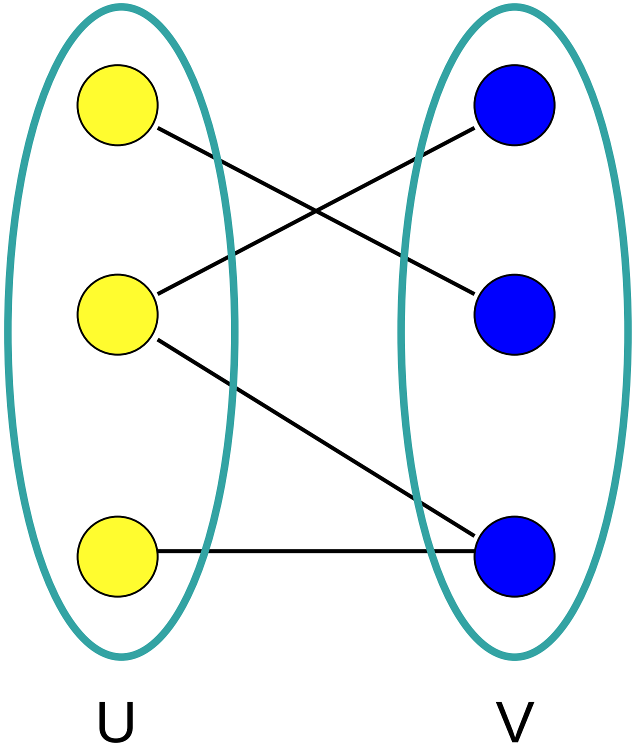


Propriété :
Le graphe G=(S,A) possède une arborescence couvrante si et seulement si G possède une racine.

Preuve : analogue à celle de l'existence d'un arbre couvrant.

Graphe biparti

Définition. Un graphe est dit biparti si il existe une partition de son ensemble de sommets en deux sous-ensembles U et V telle que chaque arête ait une extrémité dans U et l'autre dans V.



Algorithme de reconnaissance de graphe biparti vu en TD.

Graphe Eulérien

Définition. Un cycle eulérien est un cycle passant une et une seule fois par chaque arête du graphe. Un graphe est dit Eulérien si il admet un cycle eulérien.

Théorème. Un graphe **non-orienté** est Eulérien ssi il est connexe et tous ses sommets sont de degré pair.

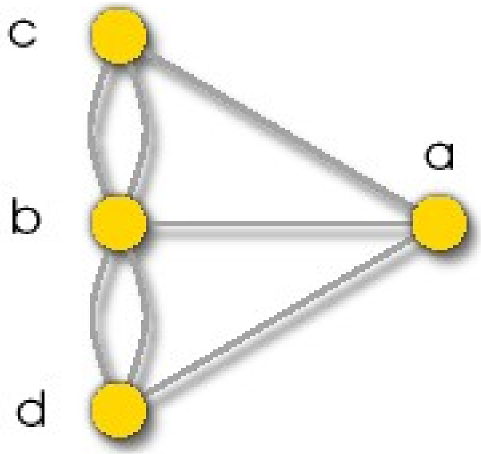
Preuve

Condition nécessaire. Considérons un sommet x du cycle eulérien. Lors du parcours du cycle, à chaque fois que nous passons par x, nous y arrivons et nous en repartons par 2 arêtes non encore parcourues. Le sommet x est donc de degré pair.

Condition suffisante. Preuve constructive par l'algorithme donné dans la suite.

Retour sur les ponts de Koenigsberg

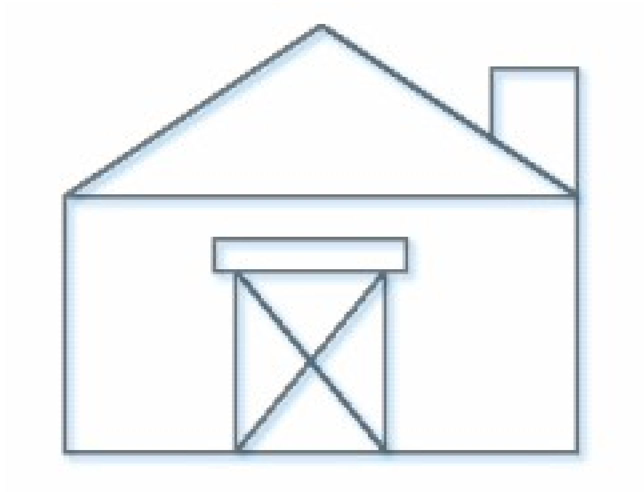
Théorème. Un graphe **non-orienté** est eulérien ssi il est connexe et tous ses sommets sont de degré pair.



Les sommets étant de degré impair, le graphe n'est pas Eulérien, et il n'existe donc pas de promenade passant une fois et une seule par chaque pont.

Un problème voisin

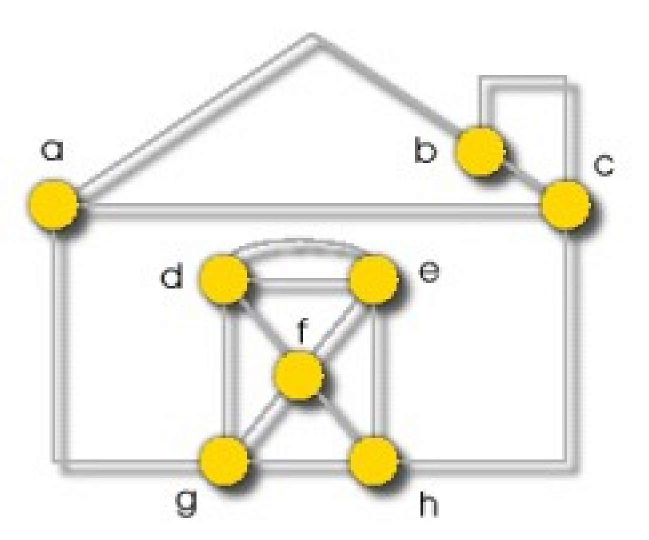
Question Est-il possible de dessiner cette maison sans lever le crayon, et bien sûr sans repasser par le même trait ?



Chaîne eulérienne

Définition. Une chaîne eulérienne est une chaîne passant une et une seule fois par chaque arête du graphe.

Le problème précédent revient à tester l'existence d'une chaîne eulérienne dans le graphe non-orienté suivant.



Théorème. Un graphe **non-orienté** admet une chaîne eulérienne ssi il est connexe et le nombre de sommets de degré impair est 0 ou 2.

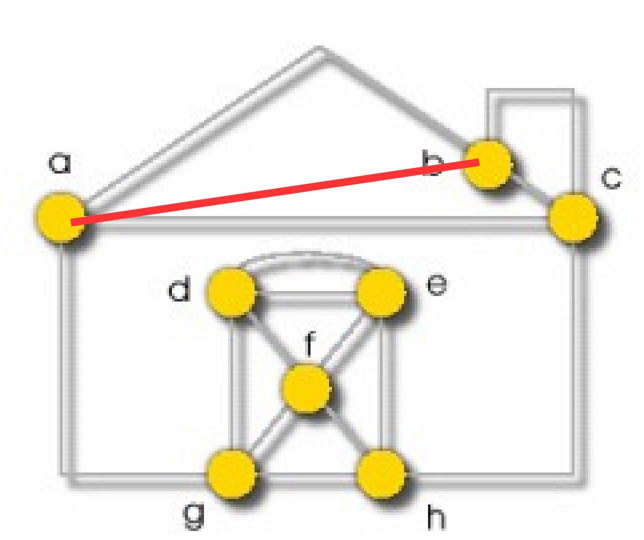
Seuls a et b sont de degré impair, donc il existe une chaîne eulérienne.

Chaîne eulérienne

OK, mais comment tracer le dessin en pratique ? (autrement dit, déterminer une chaîne eulérienne)

La recherche d'une chaîne eulérienne revient à la recherche d'un cycle eulérien :

- Si tous les sommets sont de degré pair, on recherche un cycle eulérien ;
- Si deux sommets sont de degré impair, on ajoute une arête entre ces deux sommets et on est ramené au cas précédent.

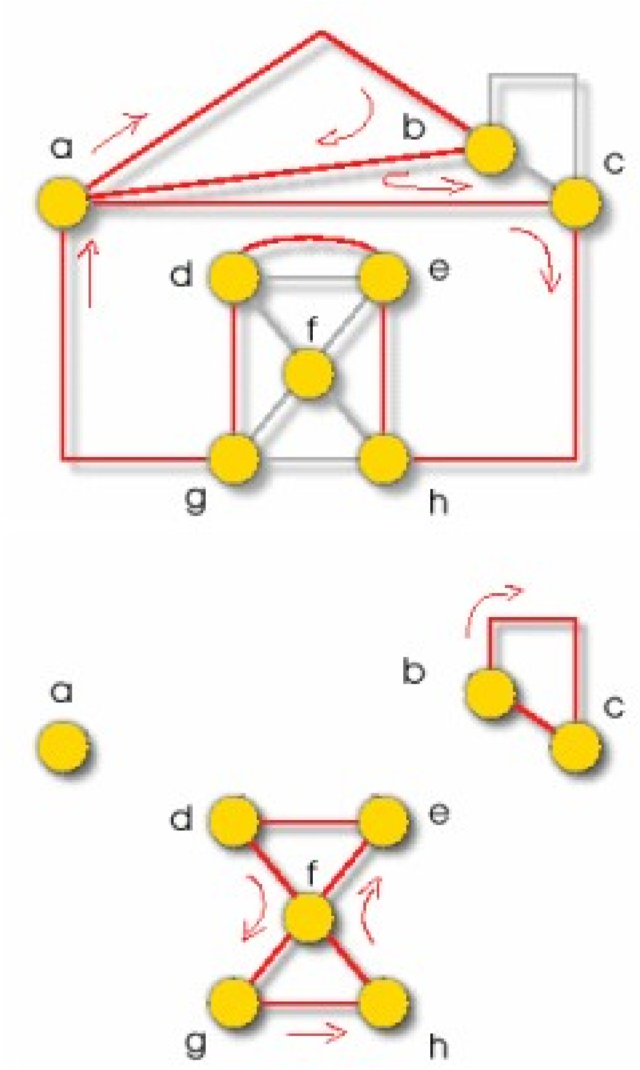


Algorithme

`ALGORITHME` Euler
`ENTREES` $G=(V,E)$ un graphe dont tous les sommets sont de degré pair, x un sommet de V
`SORTIE` ϕ un cycle eulérien sur la composante connexe de x

```
φ : LISTE des sommets du cycle dans l'ordre de parcours
Initialiser φ := (x)
// Base de la récursivité : x est isolé
Si x est un sommet isolé
Alors
      Retourner φ
Sinon         // On construit un cycle contenant x
     Initialiser y := x
     Tant Que y n'est pas un sommet isolé
          Choisir z l'un de ses voisins
          Supprimer l'arête (y,z) ; y := z
          φ ← y    // on ajoute le sommet au cycle
     Fin TantQue
     // Appel récursif sur chacun des k sommets du cycle φ en concaténant les réponses
     Retourner Euler(G, φ (1))○...○ Euler(G, φ (k))
Fin Si
```

Exemple



La première phase construit par exemple le cycle (a,b,a,c,h,e,d,g,a) en partant du sommet a.

Récursivement l'algorithme est appelé sur chacun des sommets du cycle :

- Le sommet a étant isolé, l'algorithme retourne immédiatement (a).
- Pour le sommet b, l'algorithme construit récursivement le cycle (b,c,b).
- Le sommet c étant maintenant isolé, l'algorithme retourne (c).
- Pour le sommet h, l'algorithme construit le cycle (h,f,e,d,f,g,h).
- Les sommets restant à visiter sur le cycle, (e,d,g,a), sont désormais tous isolés.

Le cycle eulérien retourné est **(a,b,c,b,a,c,h,f,e,d,f,g,h,e,d,g,a)**.

Preuve

```
ALGORITHME Euler
ENTREES G=(V,E) un graphe dont tous les sommets sont de degré pair, x un sommet de V
SORTIE  ϕ  un cycle eulérien sur la composante connexe de x

ϕ : LISTE des sommets du cycle dans l'ordre de parcours
Initialiser  ϕ := (x)
// Base de la récursivité : x est isolé
Si  x est un sommet isolé
Alors
       Retourner ϕ
Sinon        // On construit un cycle contenant x
      Initialiser y := x
      Tant Que  y n'est pas un sommet isolé
            Choisir z l'un de ses voisins
            Supprimer l'arête (y,z) ; y := z
             ϕ ← y    // on ajoute le sommet au cycle
      Fin TantQue
      // Appel récursif sur chacun des k sommets du cycle ϕ  en concaténant les réponses
      Retourner Euler(G, ϕ (1))○...○ Euler(G, ϕ (k))
Fin Si
```

- Tout d'abord remarquons que la première phase de l'algorithme construit bien un cycle de x à x. En effet chaque fois que nous arrivons et repartons d'un sommet dans notre marche, nous supprimons 2 de ses arêtes incidentes. Tous les sommets étant de degré pair, seul le sommet de départ x peut être isolé en entrée de boucle tant que.

- Le fait que l'algorithme construit un cycle eulérien peut alors se montrer par induction sur le nombre d'arêtes du graphe. Les arêtes du graphe étant supprimées au fur et à mesure de la construction, elles apparaissent bien au plus une fois dans le cycle. Par connexité de G, elles apparaissent au moins une fois. Elles apparaissent donc exactement une fois dans le cycle final.

Complexité

ALGORITHME Euler
ENTREES *G=(V,E)* un graphe dont tous les sommets sont de degré pair, *x* un sommet de *V*
SORTIE ϕ un cycle eulérien sur la composante connexe de *x*

```
ϕ : LISTE des sommets du cycle dans l'ordre de parcours
Initialiser ϕ := (x)
// Base de la récursivité : x est isolé
Si x est un sommet isolé
Alors
     Retourner ϕ
Sinon        // On construit un cycle contenant x
    Initialiser y := x
    Tant Que y n'est pas un sommet isolé
        Choisir z l'un de ses voisins
        Supprimer l'arête (y,z) ; y := z
        ϕ ← y    // on ajoute le sommet au cycle
    Fin TantQue
    // Appel récursif sur chacun des k sommets du cycle ϕ en concaténant les réponses
    Retourner Euler(G, ϕ (1))∘...∘ Euler(G, ϕ (k))
Fin Si
```

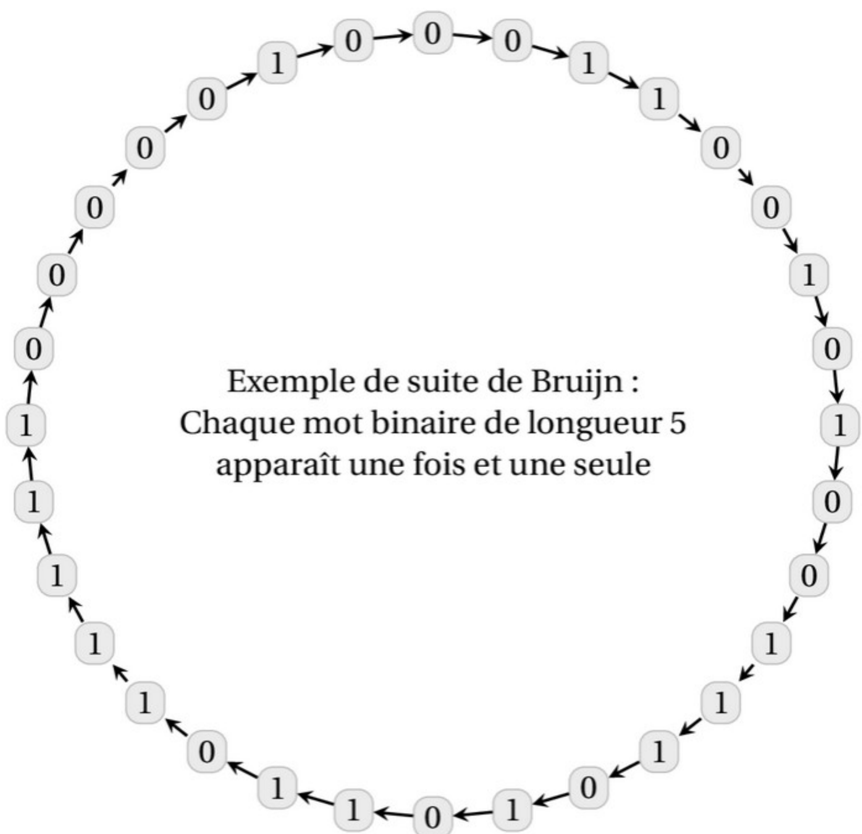
Au cours des différents appels récursifs, la boucle Tant Que est lancée n fois au plus puisque le sommet x devient isolé au terme de la boucle. De plus, le corps de la boucle Tant Que est exécuté au plus m fois (une fois pour chaque arête car l'arête est supprimée dès qu'elle est visitée). Avec une représentation par listes d'adjacence, les opérations « Choisir » et « Supprimer » sont en O(1). La complexité est alors en O(n+m). Si le graphe est connexe, on a m ≥ n-1, et donc la complexité est en O(m).

Un tour de cartes

- Un jeu de 32 cartes.
- Un spectateur coupe le jeu, prend la carte du dessus (qu'il consulte secrètement), passe le jeu à son voisin de droite, qui prend la carte du dessus, etc.
- Quand 5 cartes ont été tirées, on s'arrête.
- Le magicien demande aux personnes ayant tirée une carte noire de se lever et de se concentrer sur leur carte (toujours secrète). Le premier et le troisième spectateur se lèvent et se concentrent.
- Le magicien indique alors sans se tromper les cartes qui ont été tirées par les spectateurs : 10♠ a♥ a♣ 8♦ 9♥

Les suites de Nicolaas de Bruijn

Une suite de de Bruijn pour les mots de longueur ***n*** sur un alphabet ***A*** est une suite cyclique dans laquelle apparaît une fois et une seule chaque mot de longueur ***n*** sur l'alphabet ***A***. Une telle suite comporte nécessairement autant d'éléments que de mots de longueur ***n***, autrement dit $|A|^n$ éléments.



Ce qui donne sur l'alphabet {R,N} :

RRRRR-RRRRN-RRRNR-RRNRR-
RNRRR-NRRRN-RRRNN-RRNNR-
RNNRR-NNRRN-NRRNR-RRNRN-
RNRNR-NRNRR-RNRRN-NRRNN-
RRNNN-RNNNR-NNNRN-NNRNR-
NRNRN-RNRNN-NRNNR-RNNRN-
NNRNN-NRNNN-RNNNN-NNNNN-
NNNNR-NNNRR-NNRRR-NRRRR

R = 0
N = 1

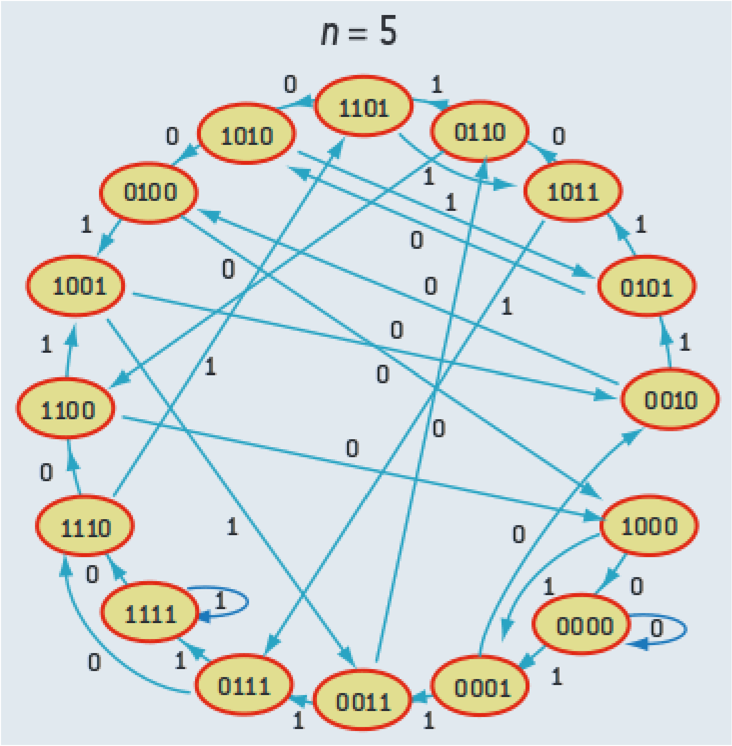
Le tableau du magicien

Si vous classez un jeu de 32 cartes dans l'ordre 8♦ 9♥ r♥ 7♥ 10♦ v♣ d♥ 9♦ v♠ v♥ 9♣ 7♣ d♦ 10♥ 9♠ r♣ d♣ r♠ 10♣ r♦ 8♥ a♦ 8♠ d♠ 7♦7♠ a♠ 8♣ v♦ 10♠ a♥ a♣, alors la seule connaissance de la couleur (noir ou rouge) de cinq cartes consécutives vous permet de savoir quelles sont ces cartes. C'est la clé du tour présenté dans l'article.

Le tableau ci-dessous permet la réalisation pratique du tour. En fonction du quintuplet de N ou R (les 32 possibilités sont indiquées dans la colonne de gauche), il vous indique les cinq cartes dont il s'agit.

NNNNN	9♠	r♣	d♣	r♠	10♣
NNNNR	r♣	d♣	r♠	10♣	r♦
NNNRN	7♠	a♠	8♣	v♦	10♠
NNNRR	d♣	r♠	10♣	r♦	8♥
NNRNN	8♠	d♠	7♦	7♠	a♠
NNRNR	a♠	8♣	v♦	10♠	a♥
NNRRN	9♣	7♣	d♦	10♥	9♠
NNRRR	r♠	10♣	r♦	8♥	a♦
NRNNN	d♠	7♦	7♠	a♠	8♣
NRNNR	v♠	v♥	9♣	7♣	d♦
NRNRN	8♣	v♦	10♠	a♥	a♣
NRNRR	10♠	a♥	a♣	8♦	9♥
NRRNN	7♣	d♦	10♥	9♠	r♣
NRRNR	v♣	d♥	9♦	v♠	v♥
NRRRN	10♣	r♦	8♥	a♦	8♠

Graphes de de Bruijn



Un sommet : un mot de longueur *n-1*.

On trace un arc entre deux sommets si les *n-2* derniers caractères du mot initial correspondent aux *n-2* premiers caractères du mot terminal. L'arc est étiqueté par le dernier caractère du mot terminal.

Un circuit eulérien dans ce graphe correspond à une suite de de Bruijn.

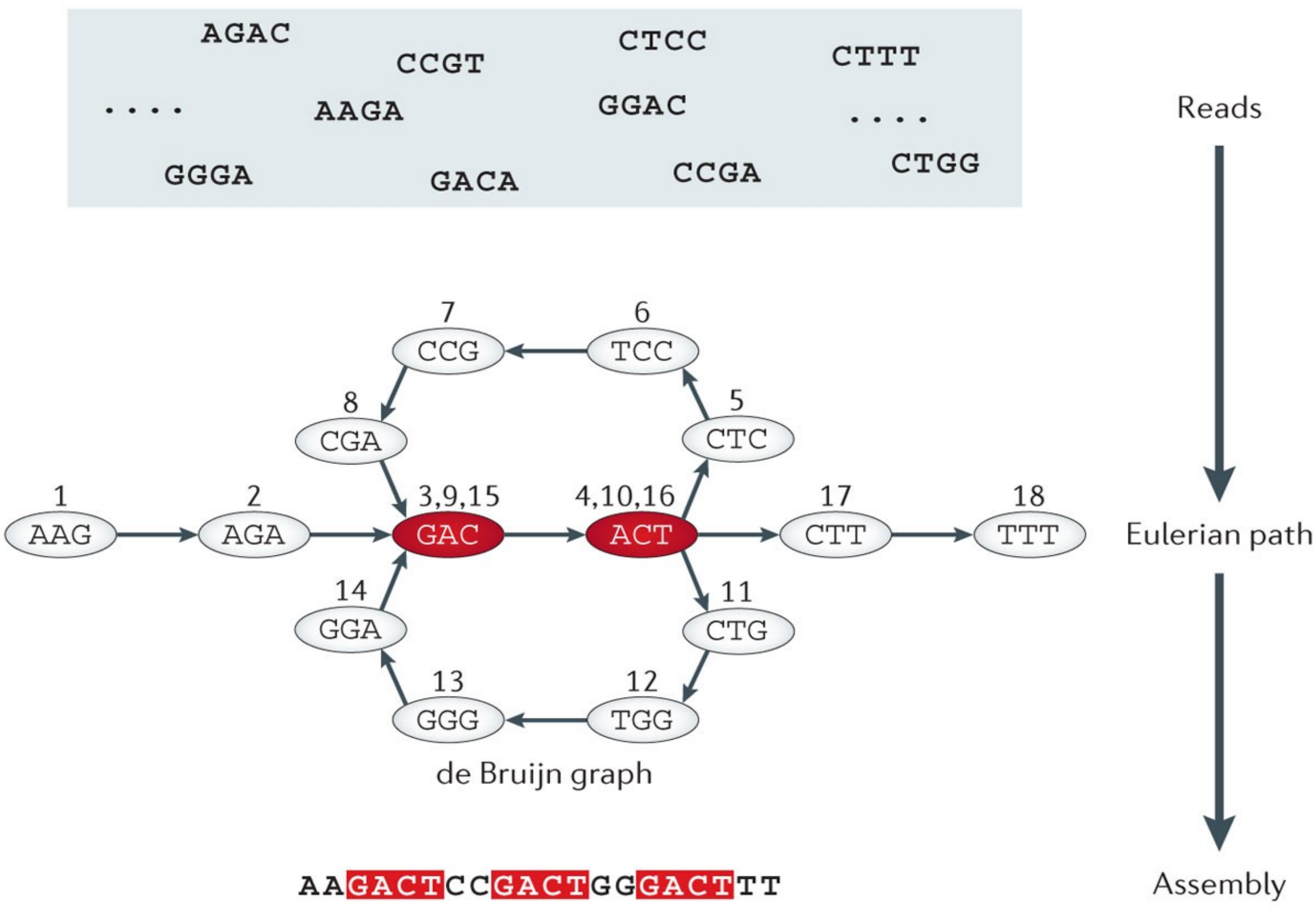
Théorème. Un graphe **orienté** est eulérien ssi il est connexe et pour tout sommet *s* on vérifie $d^+(s) = d^-(s)$.

Pour un sommet donné, il y a autant d'arcs sortants que de mots de longueur n-1 dont les n-2 premiers caractères sont communs, soit |A| arcs sortants.
Pour un sommet donné, il y a autant d'arcs entrants que de mots de longueur n-1 dont les n-2 derniers caractères sont communs, soit |A| arcs entrants.

Les graphes de de Bruijn sont eulériens, dont il existe une suite de Bruijn pour tout alphabet *A* et pour tout *n* !

Assemblage des génomes

- Les séquenceurs d'ADN produisent de nombreuses petites séquences extraites d'une longue séquence de quatre lettres A,G,C,T (codant un gène, un chromosome, etc.).
- Les petites séquences ont des parties communes qui déterminent leur assemblage correct.
- Une petite séquence peut parfois s'assembler avec plusieurs, et on est donc face au problème suivant : assembler les petites séquences afin de ne déterminer qu'une seule grande séquence.



Nature Reviews | Genetics

La méthode est fondée sur la recherche d'un chemin eulérien dans le graphe des petites séquences.

Quiz : Chaîne et cycle Eulérien

Le graphe ci-dessous comporte :

A) Un cycle Eulérien.
B) Un chaîne Eulérienne.
C) Les deux.
D) Aucun des deux.