LICENCE D'INFORMATIQUE

Sorbonne Université

# 3I003 – Algorithmique

## Cours 3 : Introduction aux graphes

Année 2018-2019

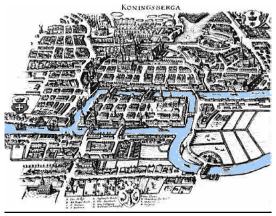
Responsables et chargés de cours

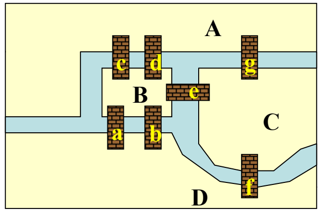
Fanny Pascual

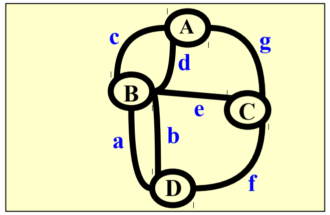
Olivier Spanjaard

# 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.






Représentation des ponts : arêtes

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

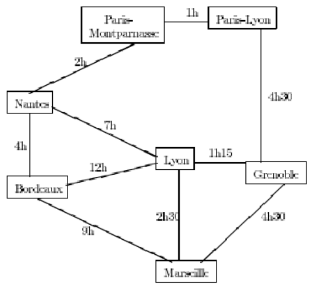
**Un tel cycle est appelé un cycle eulérien.**



# 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.

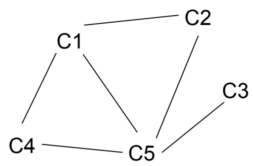


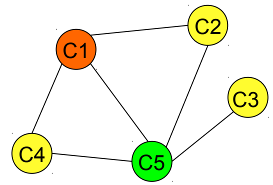
# Emploi du temps

- 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 !) :

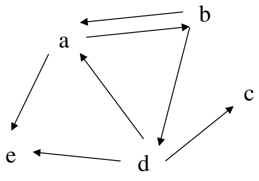
- Web : itinéraires dans Google maps, étude du « graphe du web »...
- GPS : logiciels de détermination d'itinéraires
- Gestion de production : optimisation de l'ordonnancement
- Compagnies aériennes : planification des vols, itinéraires, plannings du personnel aérien...
- Telecoms : affectation de fréquence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



# Graphe orienté

Un graphe $G=(S,A)$ est formé:
- d'un ensemble fini $S=\{s_1,s_2,\dots,s_n\}$ de $n$ « sommets »;
- d'un ensemble fini $A \subset S\times S$ de $m$ couples de sommets appelés « arcs ».

Si $a=(s_i,s_j)$ est un arc :
- $s_i$ est son extrémité initiale;
- $s_j$ est son extrémité terminale.

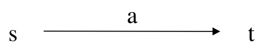



Exemple :
n=5 ; S={a,b,c,d,e}
A={(a,b),(b,a),(b,d),(d,a),(a,e),(d,e),(d,c)}
G=(S,A)



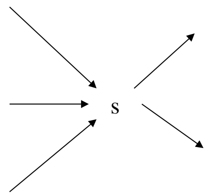
# Vocabulaire sur les graphes orientés

Prédécesseur d'un sommet t :
sommet s tel (s,t) est un arc.




Successeur d'un sommet s :
sommet t tel (s,t) est un arc.

Demi-degré extérieur d'un sommet s :
$d^+(s)$ = nombre de successeurs de s.




$d^-(s)=3$, $d^+(s)=2$

Demi-degré intérieur d'un sommet s :
$d^-(s)$ = nombre de prédécesseurs de s.

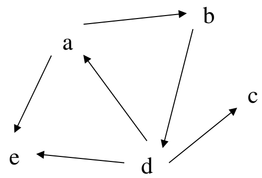
Degré d'un sommet s : $d(s) = d^+(s) + d^-(s)$



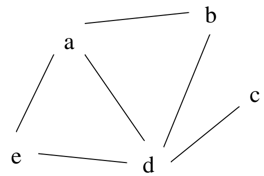
# Graphe non orienté

Si l'orientation des flèches n'est pas pertinente pour le modèle, on supprime l'orientation des arcs.

On obtient alors un graphe non orienté.
Les liaisons entre sommets s'appellent alors des arêtes.




graphe G



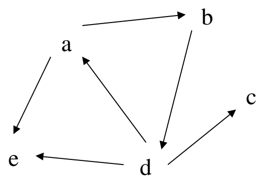

graphe non orienté associé à G

# Graphe partiel

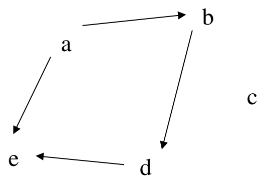
Soit G=(S,A) un graphe.

Si A' ⊂ A, alors G'=(S, A') est un graphe partiel de G.

Remarque : il existe $2^m$ graphes partiels d'un graphe comportant m arcs.




graphe G




un graphe partiel de G



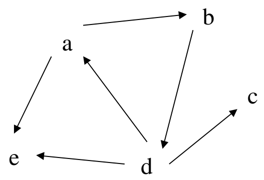
# Sous-graphe induit

Si T ⊂ S, alors G(T)= (T, A(T))

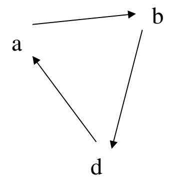
où : $A(T)=\{(s_i,s_j) \in A \mid s_i \in T \text{ et } s_j \in T\}$

est le sous-graphe de G induit par T.

Remarque : il existe $2^n$ sous-graphes induits d'un graphe comportant n sommets.




graphe G




sous-graphe de G induit par T={a,b,d}



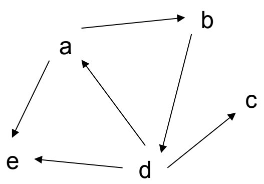
# Représentations des graphes

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 | | | | | |

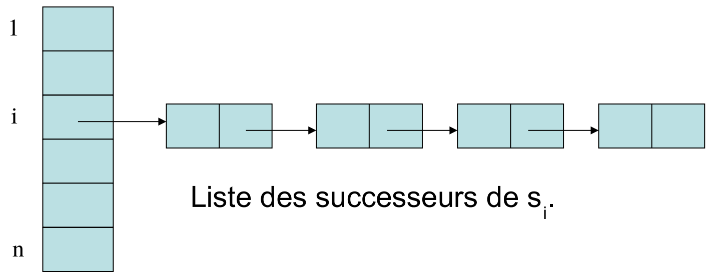
B

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 $a=(s_i,s_j)$ : $O(d^+(s_i))$

# Chaîne-Cycle-Chemin-Circuit

Une chaîne c de longueur p est une liste du type :

$c = (s_0,s_1,s_2,....,s_{p-1},s_p)$ si $p > 0$ ;

$c = (s_0)$ si $p = 0$ ;

telle que pour tout $k = 1,2,...,p$ les sommets $s_{k-1}$ et $s_k$ sont les deux extrémités de l'arc $a_k$.

Soit $c=(s_0,s_1,s_2,....,s_{p-1},s_p)$ une chaîne.

Si $a_k=(s_{k-1},s_k)$, alors $a_k$ est direct pour c ;

Si $a_k=(s_k,s_{k-1})$, alors $a_k$ est inverse pour c.

Une chaîne c est élémentaire si les sommets empruntés par c sont distincts.



Une chaîne $c=(s_0,s_1,s_2,....,s_{p-1},s_p)$ est un cycle si $s_0 = s_p$.

Un cycle est élémentaire si :

- soit $p = 0$ ;
- soit $p > 0$ et $(s_0,s_1,s_2,....,s_{p-1})$ est une chaîne élémentaire.

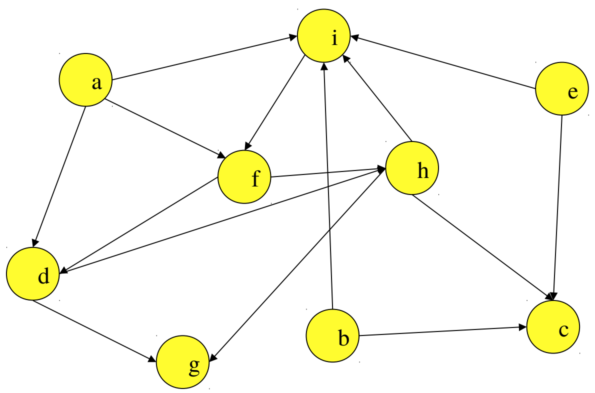
Une chaîne $c=(s_0,s_1,s_2,....,s_{p-1},s_p)$ est un chemin si tous les arcs empruntés sont directs. On note alors $c=(s_0,s_1,s_2,....,s_{p-1},s_p)$

Un chemin $c=(s_0,s_1,\dots,s_p)$ est un circuit si $s_0 = s_p$.

Un circuit est élémentaire si :

- soit $p = 0$ ;
- soit $p > 0$ et $(s_0,s_1,s_2,s_3,....,s_{p-1})$ est un chemin élémentaire.






graphe G

c = (a,i,h,c,b,i,f) est une chaîne (non élémentaire) de longueur 6 de a à f.

c=(a,i,f,h,i) est un chemin (non élémentaire) de longueur 4.

c=(i,f,h,i) est un circuit élémentaire de longueur 3.



# Connexité simple

Soit G=(S,A) un graphe.

Le sommet s est (simplement) relié au sommet t (notation s – t) s'il existe une chaîne de G de s à t.

La relation – est:

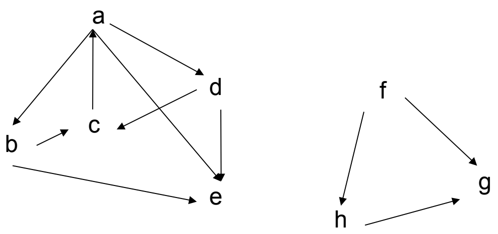
- réflexive (chaîne de longueur nulle)
- symétrique (chaîne miroir)
- transitive (concaténation des 2 chaînes)

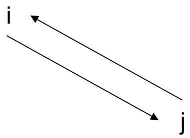
Les classes d'équivalence sont les composantes connexes de G.

Un graphe G est dit « connexe » s'il ne possède qu'une seule composante connexe.

# Composantes connexes

Composante connexe = sous-graphe induit maximal connexe.




G comporte 3 composantes connexes.



# Connexité forte

Soit G=(S,A) un graphe orienté.

Le sommet s est (fortement) relié au sommet t (notation $s \leftrightarrow t$) s'il existe dans G un chemin de s à t et un chemin de t à s.

La relation $\leftrightarrow$ est :

- réflexive (chemin de longueur nulle)
- symétrique (définition)
- transitive (concaténation des 2 chemins dans chaque sens)

Les classes d'équivalence de $\leftrightarrow$ sont les composantes fortement connexes de G.

Un graphe G est dit « fortement connexe » s'il ne possède qu'une seule composante fortement connexe.



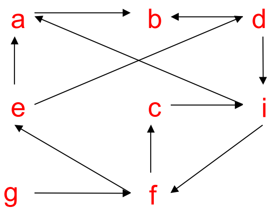
Graphe réduit d'un graphe G=(S,A).

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

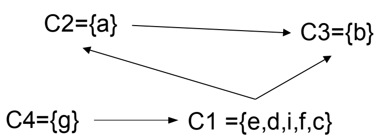
Arcs : $(C_i,C_j)$ est un arc si

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.

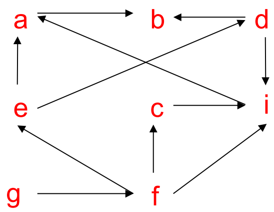
Preuve :
S'il existait un circuit dans le graphe réduit, les composantes fortement connexes du circuit appartiendraient à une même composante fortement connexe. Contradiction.



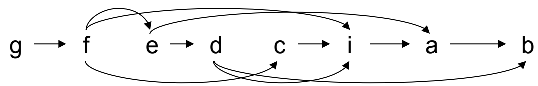
# Graphes particuliers

## Graphe sans circuit

Propriété (liste topologique des sommets) :
Soit G=(S,A) un graphe sans circuit.
Il existe une liste $(s_1, s_2, \ldots, s_n)$ 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-1arêtes

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

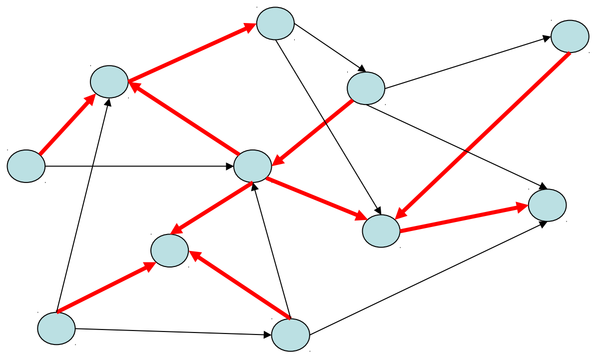
$D_4$: graphe connexe, qui devient non connexe par suppression d'une arête quelconque

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



# 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.



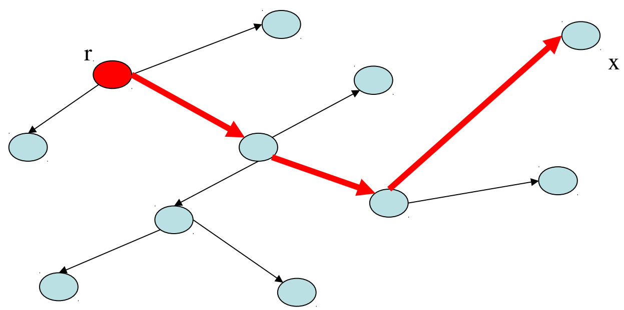
# Des arbres particuliers : les arborescences

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.

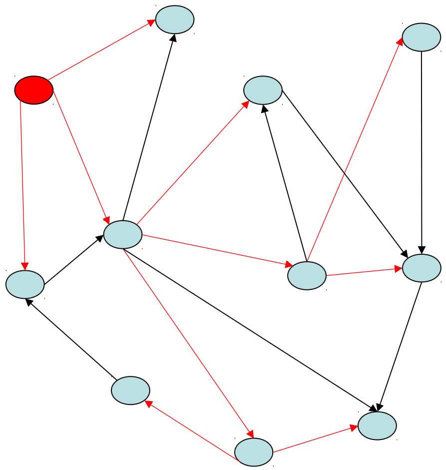
Une arborescence



## 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.



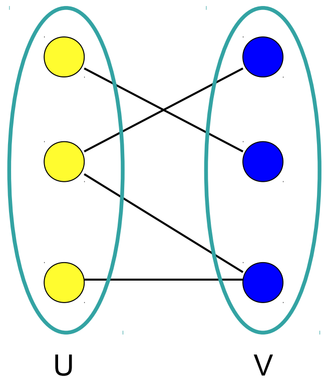
Propriété :
Le graphe G=(S,A) possède une arborescence couvrante si et seulement si G possède une racine.

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**

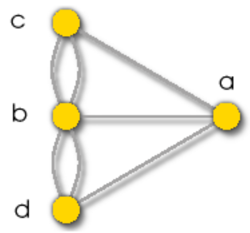
*Condition nécessaire.* Considérons un sommet x du cycle eulérien. Lors du parcours du cycle, à chaque fois que nous passons par x, nous y arrivons et nous en repartons par 2 arêtes non encore parcourues. Le sommet x est donc de degré pair.

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



# Retour sur les ponts de Koenigsberg

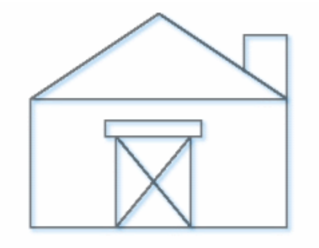
Théorème. Un graphe **non-orienté** est eulérien ssi il est connexe et tous ses sommets sont de degré pair.




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 ?

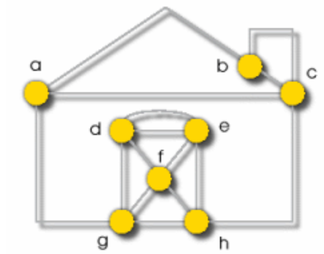




# Chaîne eulérienne

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.

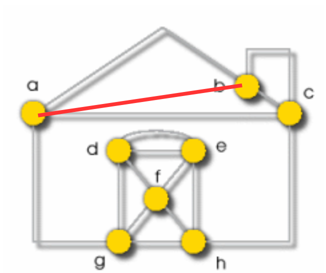


# Chaîne eulérienne

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.



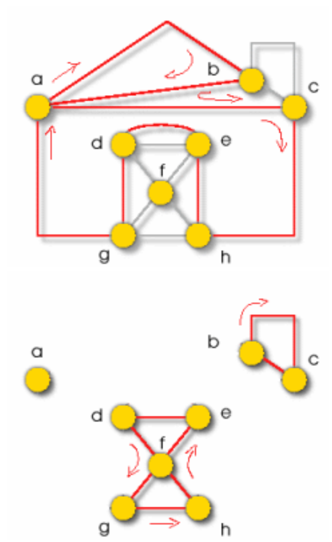



# Algorithme

```
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 contenant 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 déconnecté lors de l'arrivée à ce sommet.

- 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 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
```

Il y a n appels récursifs au plus. Au cours de ces appels récursifs, chacune des m arêtes est visitée au plus une fois (puisqu'une arête est supprimée dès qu'elle est visitée). Avec une représentation par liste d'adjacence, la complexité est donc en O(n+m). Si le graphe est supposé connexe, on a m >= n-1, et donc la complexité est en O(m).

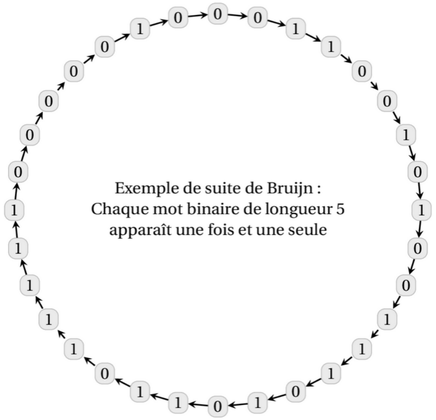


# Un tour de cartes

- 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

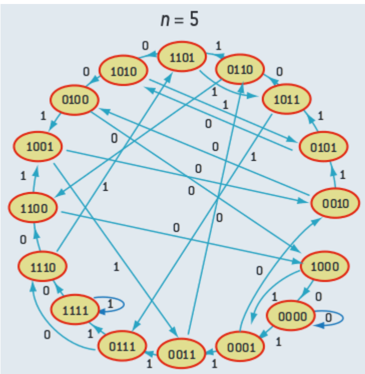
Si vous classez un jeu de 32 cartes dans l'ordre 8♦ 9♥ r♥ 7♥ 10♦ v♣ d♥ 9♦ v♠ v♥ 9♣ 7♣ d♦ 10♥ 9♠ r♣ d♣ r♠ 10♣ r♦ 8♥ a♦ 8♠ d♠ 7♦7♠ a♠ 8♣ v♦ 10♠ a♥ a♣, alors la seule connaissance de la couleur (noir ou rouge) de cinq cartes consécutives vous permet de savoir quelles sont ces cartes. C'est la clé du tour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

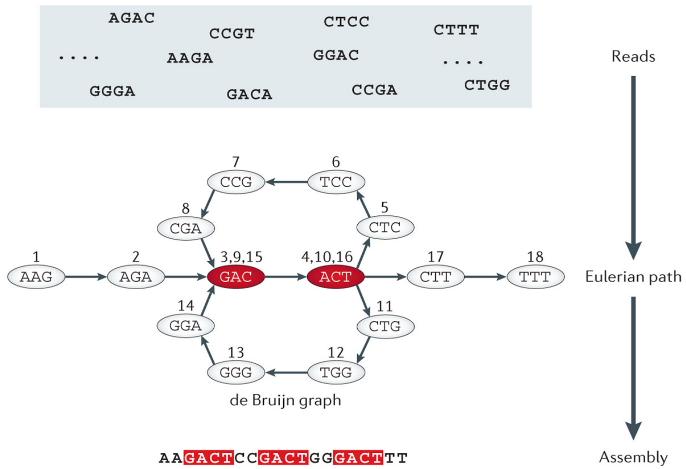
Pour un sommet donné, il y a autant d'arcs sortants que de mots de longueur n-1 dont les n-2 premiers caractères sont communs, soit |A| arcs sortants.
Pour un sommet donné, il y a autant d'arcs entrants que de mots de longueur n-1 dont les n-2 derniers caractères sont communs, soit |A| arcs entrants.

**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.




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

Nature Reviews | Genetics