M1 Informatique – Graphes

Irena.Rusu@univ-nantes.fr
LS2N, bât. 34, bureau 303
tél. 02.51.12.58.16

- 12h de CM
- 9h20 de TD (deux groupes)
- Enseignement en distanciel (sur MADOC)

- Evaluation courte :
  - le lundi 7/10/2019 quelque part entre 9h30 et 10h50 (fournit la note de CC)
  - amphi G

- Examen : renseignement pris, ils auront lieu en janvier !

ni autres tablettes etc.

Réponses à une question que l'on me pose souvent :

- Note 1ère session :

  NoteFinale = 0,5*NoteCC+0,5*NoteExamen

- Note 2ème session :

  NoteFinale = 0,4*NoteCC+0,6*NoteExamen

- A. Aho, J. Hopcroft, J. Ullman,
  Structures de données et algorithmes
  InterEditions, 1987.
- Th. Cormen, Ch. Leiserson, R. Rivest
  Introduction à l'algorithmique
  Dunod, 1994.
- C. Froidevaux, M.C. Gaudel, M. Soria,
  Types de données et algorithmes
  Edisciences, 1994.
- C. A. Shaffer
  A Practical Introduction to Data Structures and Algorithm Analysis
  Prentice Hall, 1998.

Résoudre efficacement des problèmes faisant intervenir des graphes.

- Les graphes sont un outil de modélisation très puissant.

- Algorithmique des graphes bien développée
  - Des algorithmes existent pour beaucoup de problèmes
  - La difficulté des problèmes (existence ou non d’un algorithme, meilleure complexité, etc.) souvent déjà établie

- **Deux** points de vue:
  - Mémoire utilisée
  - Temps d'exécution (~ nombre d'opérations)

- Paramètres qui fournissent les unités de mesure:
  - La taille des données en entrée (peut être exprimée par plusieurs paramètres)
  - La forme des données (via divers paramètres)

Exemple. Tri à bulles d'un tableau de *n* éléments

Mémoire : *n* cases

Nombre d'opérations : $5(n-1)^2$

(une comparaison, 3 affectations, une incrémentation par case, sauf la 1ère)

- Très grossièrement (et théoriquement) :

« efficace » ~ polynomial (n^2, n^3, $n \log n$, n^{17}), où n = taille des données

«inefficace » ~ au-delà (2^n, 2^{2^n}, $n^{n^{n^n}}$)

- Pratiquement :
  - Des algorithmes polynomiaux avec un gros exposant ne seront pas efficaces
  - Certains algorithmes exponentiels seront efficaces sur une très grande majorité d'instances (et inefficaces sur très peu d'instances).

But suivi : trouver des algorithmes aussi efficaces que possible.

Comment s'y prendre ?

- Réfléchir avant d'écrire
- Chercher la meilleure modélisation du problème (problème de graphes, d'algorithmique du texte, de programmation linéaire …?)
- Chercher la meilleure méthode d'approche (traité de manière combinatoire, ou par la programmation linéaire ou …?)
  - … en faisant appel à son expérience (qui permettra d'avoir une idée a priori du degré de difficulté)
  - … et en s'approchant autant que possible de problèmes connus.
- Choisir la représentation des données la plus appropriée à la méthode choisie pour la résolution.
- Apprendre à bien calculer la complexité d'un algorithme.
- Ne jamais se contenter de la première version.

- Eléments de théorie des graphes
- Arbres et Arborescences
- Distanciel : composantes fortement connexes (sem. 39+40 !)
- Plus courts chemins
- Problèmes de couplage

M1 Informatique – Graphes
Eléments de théorie des graphes

Irena.Rusu@univ-nantes.fr
LS2N, bât. 34, bureau 303
tél. 02.51.12.58.16

- Graphes et sous-graphes
- Chemins, cycles, circuits
- Représentations en machine
- Autres applications

- Graphes et sous-graphes
- Chemins, cycles, circuits
- Représentations en machine
- Autres applications

Graphe orienté $G = (S, A)$

S ensemble (fini) de sommets

$A \subseteq S \times S$ ensemble d'arcs, *i.e.*, relation sur S

Graphe non orienté $G = (S, A)$

$A \subseteq S \times S$ t.q. (x,y) et (y,x) sont identiques par convention

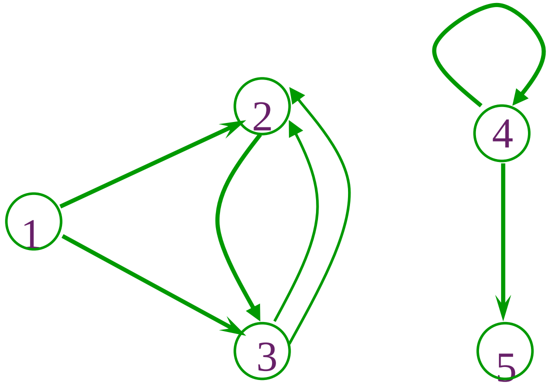
ensemble d'arêtes, relation symétrique

Graphe orienté $G = (S, A)$

S ensemble (fini) des sommets

$A \subseteq S \times S$ ensemble des arcs,

i.e., relation sur S



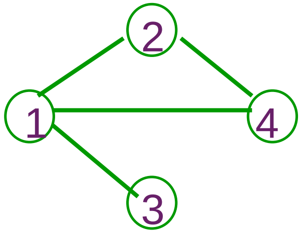
$S = \{ 1, 2, 3, 4, 5 \}$

$A = \{ (1, 2), (1, 3), (2, 3), (3, 2), (4, 4), (4, 5) \}$

Graphe non orienté $G = (S, A)$

$A \subseteq S \times S$ ensemble des arêtes,

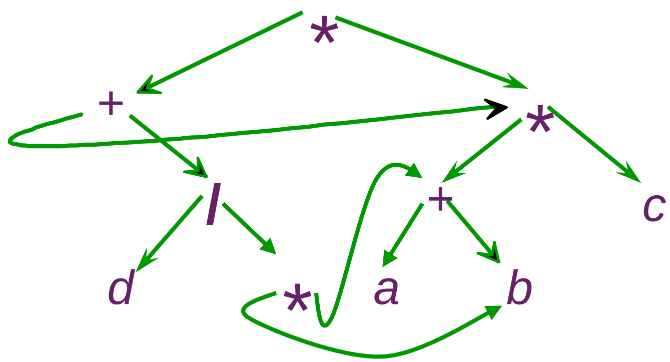
relation symétrique



$S = \{ 1, 2, 3, 4 \}$

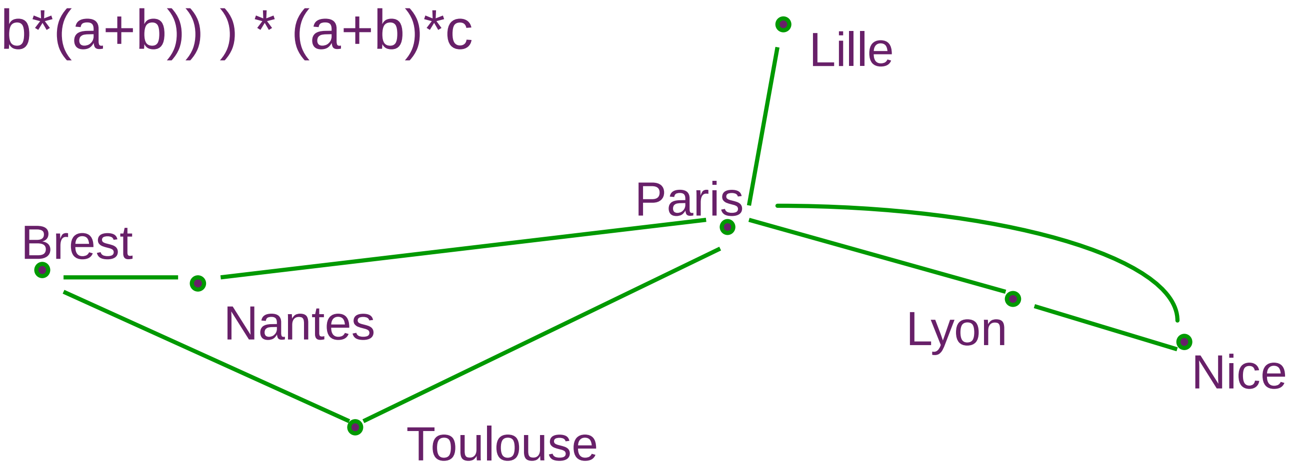
$A = \{ \{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 4\} \}$

Graphe acyclique d'une expression (DAG)



((a+b)*c+d/(b*(a+b))) * (a+b)*c

Réseau



```
01  static void
02  make_blank (struct line *blank, int count)
03  {
04    int i;
05    unsigned char *buffer;
06    struct field *fields;

07    blank->nfields = count;
08    blank->buf.size = blank->buf.length = count + 1;
09    blank->buf.buffer = (char*) xmalloc (blank->buf.size);
10    buffer = (unsigned char *) blank->buf.buffer;
11    blank->fields = fields =
        (struct field *) xmalloc (sizeof (struct field) * count);

12    for (i = 0; i < count; i++){
13        ...
14    }
15  }
```

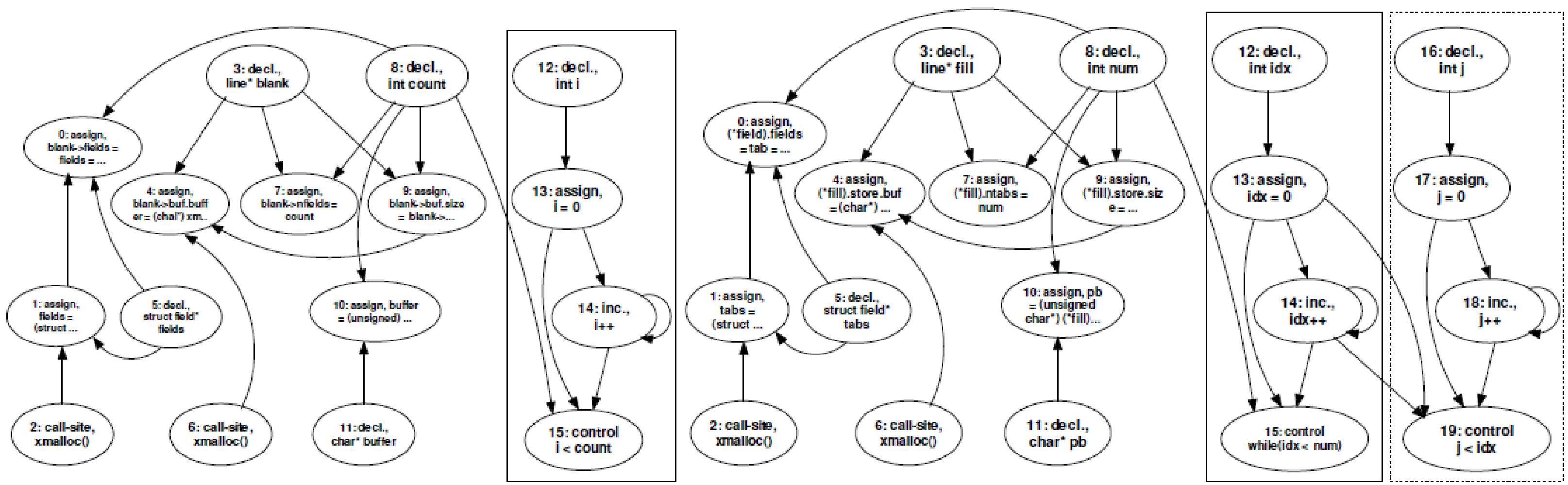
Original Code

```
01 static void
02 fill_content(int num, struct line* fill)
03 {
04    (*fill).store.size = fill->store.length = num + 1;
05    struct field *tabs;
06    (*fill).fields = tabs = (struct field *)
                  xmalloc (sizeof (struct field) * num);
07    (*fill).store.buffer = (char*) xmalloc (fill->store.size);
08    (*fill).ntabs = num;
09    unsigned char *pb;
10    pb = (unsigned char *) (*fill).store.buffer;

11    int idx = 0;
12    while(idx < num){ // fill in the storage
13      ...
14      for(int j = 0; j < idx; j++)
15        ...
16      idx++;
17    }
18 }
```

Plagiarized Code

Idée: on peut définir un graphe de dépendances qui change très peu



(a) PDG of the Original Code (b) PDG of the Plagiarized Code

©**GPLAG: Detection of Software Plagiarism by ProgramDependence Graph Analysis,** Chao Liu et al. *KDD'06,* 2006.

Graphe : $G = (S, A)$ orienté

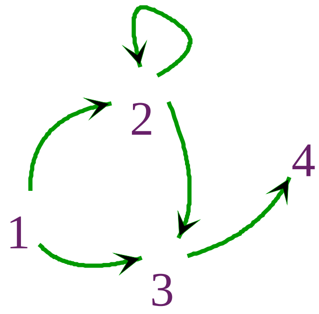
Arc : $(s, t) \in A$ t adjacent à s, t successeur de s,

s prédécesseur de t, s et t sont adjacents

Boucle : $(t, t) \in A$

Degré sortant de s : nombre d'arcs sortant de s

Degré entrant de s : nombre d'arcs entrant dans s



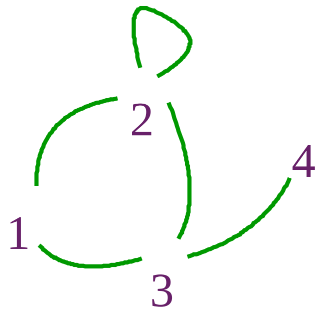
Graphe : $G = (S, A)$ non-orienté

Arête : $(s, t) \in A$ t adjacent à s, t voisin de s,

s voisin de t, s et t sont adjacents

Boucle : $(t, t) \in A$

Degré de s : nombre d'arêtes incidentes à s



Graphe simple : sans boucle ni arêtes(arcs) multiples

$G=(S,A)$ un graphe

Graphe partiel de G : graphe $G'=(S',A')$ tel que

- $S'=S$ (si inclusion, alors sous-graphe partiel)
- $A' \subseteq A$

Sous-graphe induit de G : graphe $G''=(S'',A'')$ tel que

- $S'' \subseteq S$
- $A''=A|S''$, l'ensemble des arêtes (arcs) de A ayant les deux extrémités dans S''

- Graphes et sous-graphes
- Chemins, cycles, circuits
- Représentations en machine
- Autres applications

G=(S,A) un graphe

Chemin : suite d'arêtes (arcs) $c = ((s_0,s_1), (s_1,s_2), \ldots, (s_{k-1},s_k))$

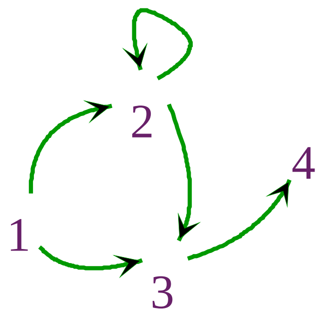
où les $(s_{i-1},s_i) \in A$

origine = s_0

extrémité = s_k

longueur = k

((1,2), (2,2), (2,3), (3,4))



Circuit (graphe orienté) ou cycle (graphe non-orienté) :

chemin dont l'origine et l'extrémité finale coïncident

Chemin (ou circuit, ou cycle) simple : s'il ne traverse pas deux fois la même arête (arc).

Chemin (ou circuit, ou cycle) élémentaire : s'il ne traverse pas deux fois le même sommet.

G=(S,A) graphe non-orienté

Distance *d(s,s')* entre deux sommets s,s' d'un graphe non-orienté : la longueur du plus court chemin joignant les deux sommets (+∞ s'il n'y a pas de chemin).

Diamètre : le maximum des distances entre deux de ses points.

Sommet isolé : distance + ∞ à tous les autres sommets

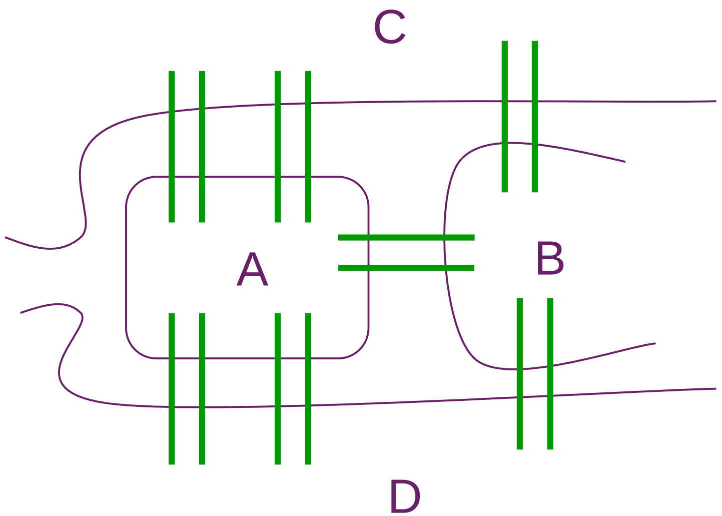
G=(S,A) un graphe

Chemin eulerien : qui contient toutes les arêtes de *G* une fois et une seule.

Chemin hamiltonien : qui contient tous les sommets de *G* une fois et une seule.

Remarque. Si un graphe non-orienté *G* contient deux chemins simples différents joignant deux sommets distincts, alors il contient un cycle simple.

Il y a sept ponts dans la ville de Königsberg. Un promeneur peut-il visiter la ville en traversant chaque pont une fois et une seule ?

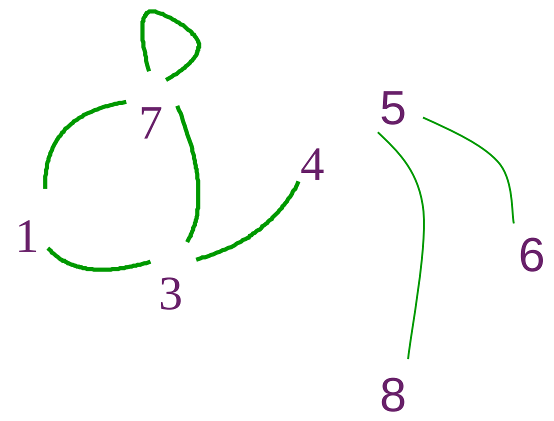


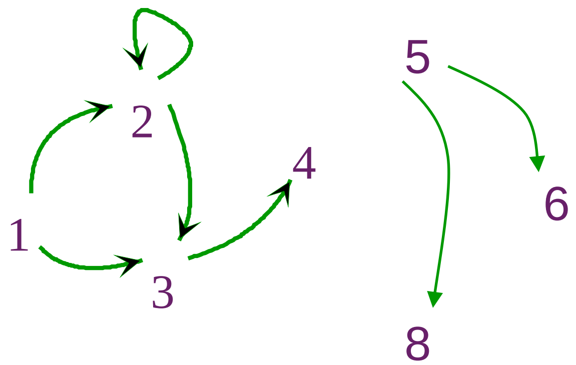
Il s’agit de trouver un chemin eulerien. Mais dans quel graphe ? Et dans quelles conditions un tel chemin existe ?

G=(S,A) graphe non-orienté

Graphe connexe : pour toute paire *(s,s')* de sommets distincts, il existe un chemin joignant s et s'.

Composante connexe de *G* : sous-graphe connexe maximal.





G=(S,A) graphe orienté

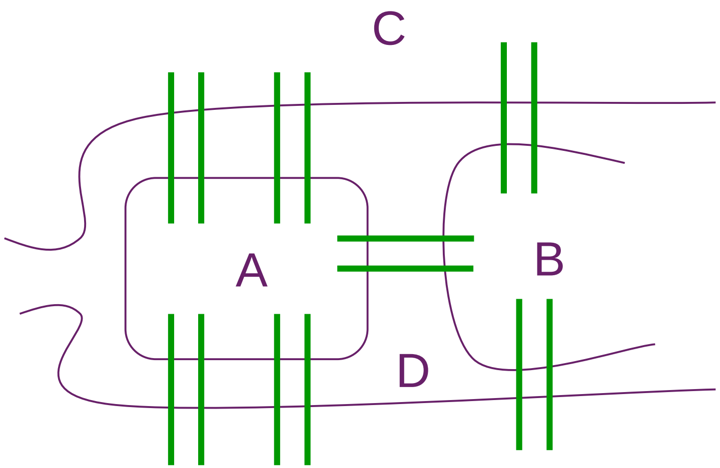
Graphe fortement connexe : pour toute paire (s,s') de sommets distincts, il existe un chemin allant de *s* à *s'.*

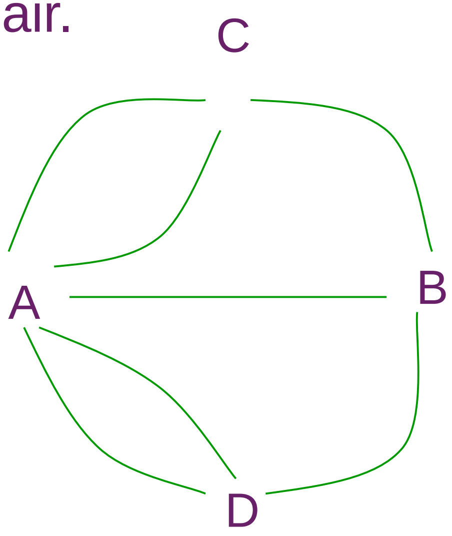
Composante fortement connexe de *G* : sous-graphe fortement connexe maximal.

G=(S,A) graphe non-orienté

Théorème. G sans sommets isolés possède un chemin éulerien si et seulement si :

- il est connexe
- il a 0 ou 2 sommets de degré impair.





- Graphes et sous-graphes
- Chemins, cycles, circuits
- Représentations en machine
- Autres applications

$G = (S, A)$ $S = \{1, 2, ..., n\}$

Matrice d'adjacence

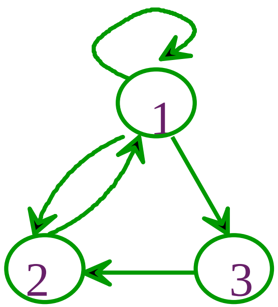
utilisation d'opérations matricielles

temps de traitement courant : quadratique

Listes de successeurs

réduit la taille si $|A| \ll |S|^2$

temps de traitement courant : $O(|S| + |A|)$

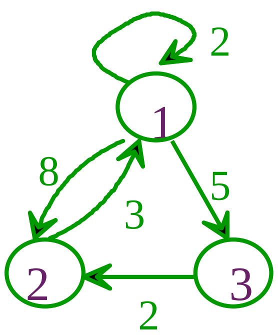


$S = \{ 1, 2, 3 \}$

$A = \{ (1,1), (1, 2), (1, 3), (2, 1), (3, 2)\}$

$$M = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

$M [i, j] = 1$ ssi j adjacent à i

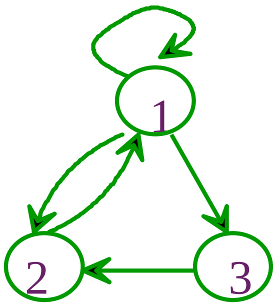


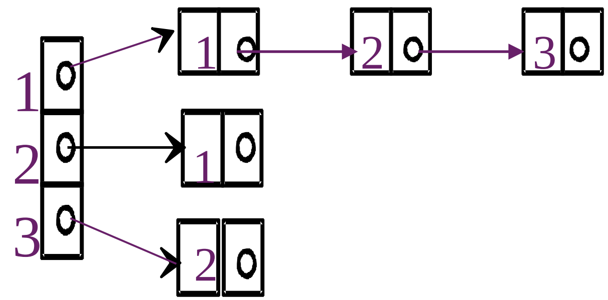
$$V = \begin{pmatrix} 2 & 8 & 5 \\ 3 & 0 & 0 \\ 0 & 2 & 0 \end{pmatrix}$$

Valuation : $v : A \longrightarrow X$

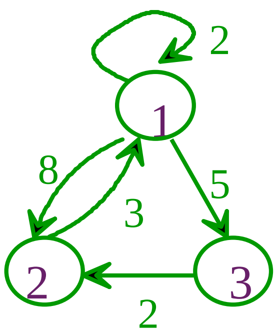
$S = \{\ 1, 2, 3\ \}$

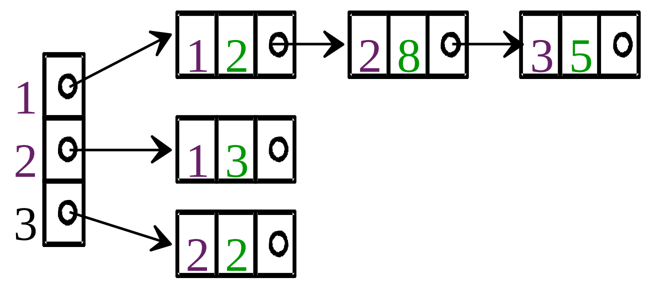
$A = \{\ (1,1), (1, 2), (1, 3), (2, 1), (3, 2)\}$





Listes des $A(s)$





Valuation : $v : A \longrightarrow X$

- Graphes et sous-graphes
- Chemins, cycles, circuits
- Représentations en machine
- Autres applications

Les faits :

- Livre disparu dans une bibliothèque, entre 14h et 16h
- Sept personnes ont passé plus ou moins de temps dans la bibliothèque dans ce créneau.

Le but : trouver le/la coupable

L'enquêteur : M. Subtil (le célèbre …)

Selon l'enquête de M. Subtil :

- chacune des sept personnes dit être entrée dans la bibliothèque une seule fois, et y être restée pendant un intervalle de temps plus ou moins long.

- Qui a vu qui selon les déclarations :
  - Antoine a vu Barbara, Christine et Elodie
  - Barbara a vu Antoine, Christine et Francois
  - Christine a vu Antoine, Barbara, Elodie et Gauthier
  - Elodie a vu Antoine, Christine, Francois et Gauthier
  - Francois a vu Barbara, Elodie et Herve
  - Gauthier a vu Christine, Elodie et Herve
  - Herve a vu Francois et Gauthier.

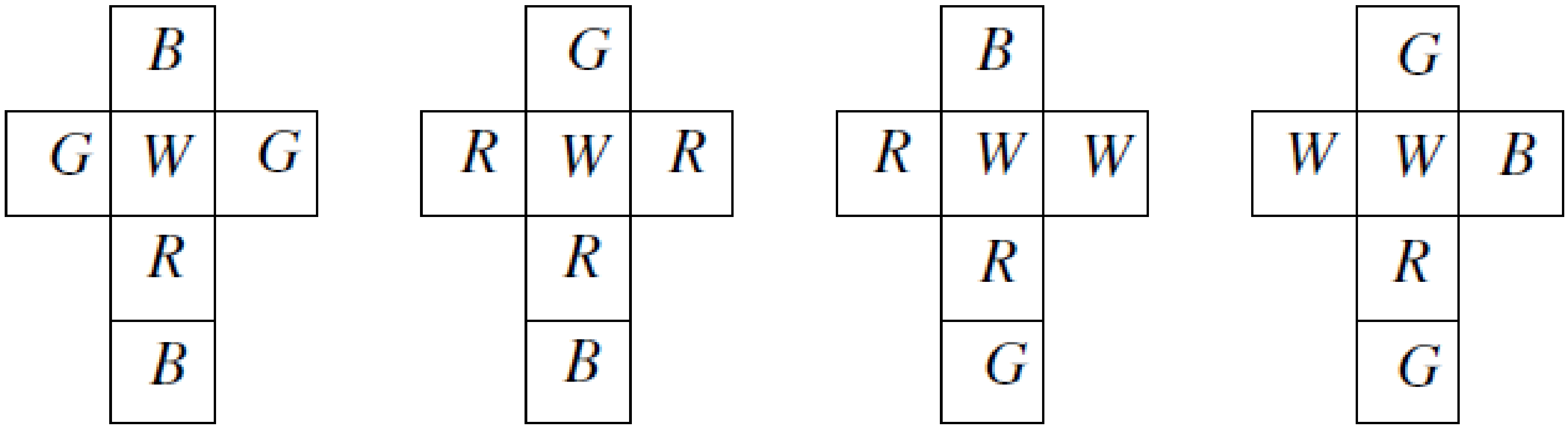
(Ces declarations étant concordantes, M. Subtil pense qu'il y a de grandes chances pour qu'elles soient vraies)

M. Subtil

- Dessine
- Réfléchit
- Elimine au fur et à mesure les suspects non-coupables
- Et garde un seul nom.

Lequel ?

- Soient 4 cubes avec les faces coloriées en 4 couleurs, dont les variantes dépliées sont les suivantes :



© Tom Davis, Graph Theory Problems and Solutions (figures de cette page et les suivantes)

Problème : Empiler ces quatre cubes de sorte que, lorsqu’on regarde chacune des quatre faces de la tour obtenue (avant, arrière, gauche, droite), chaque couleur apparait exactement une fois.

- C’est celle où on essaie toutes les possibilités

- 24 positions différentes pour chaque cube
- 4 cubes

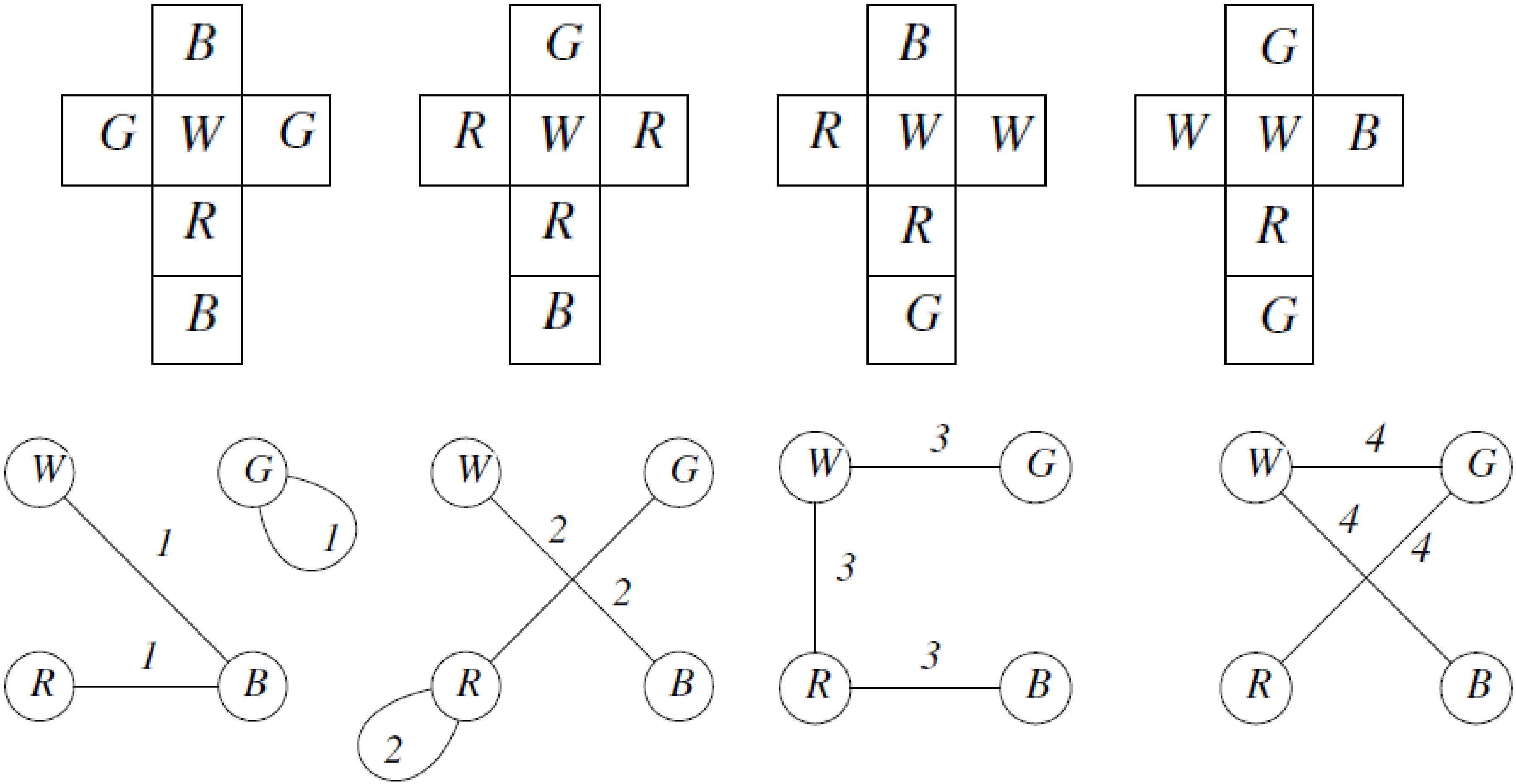
$\longrightarrow$ 24^4 c’est-à-dire 331776 configurations

… mais une seule solution !

- Trouver une solution partielle qui arrange correctement les faces, mais seulement 2 par 2 :
  1. On peut arranger les cubes de sorte que l'avant et l'arrière de la tour contiennent chaque couleur exactement une fois (peu importe ce qu'il arrive aux faces latérales)
  2. On peut faire la même chose pour les faces latérales (peu importe ce qu'il arrive aux faces avant et arrière)
  3. Les faces des cubes impliquées dans le 1er arrangement et dans le 2ème sont disjointes.

- Obtenir la solution finale en :
  - Arrangeant d'abord les cubes selon le 1er arrangement.
  - Tournant les cubes verticalement de sorte que les faces gauche et droite se retrouvent arrangées selon le 2ème arrangement.
  - Possible grâce à la condition 3 ci-dessus.

- Construire un graphe pour chaque cube, reliant les couleurs se retrouvant sur des faces opposées et en étiquetant les arêtes avec le numéro du cube.



Deux faces opposées du cube ont la configuration recherchée (chaque couleur exactement une fois sur chaque face)

si et seulement si

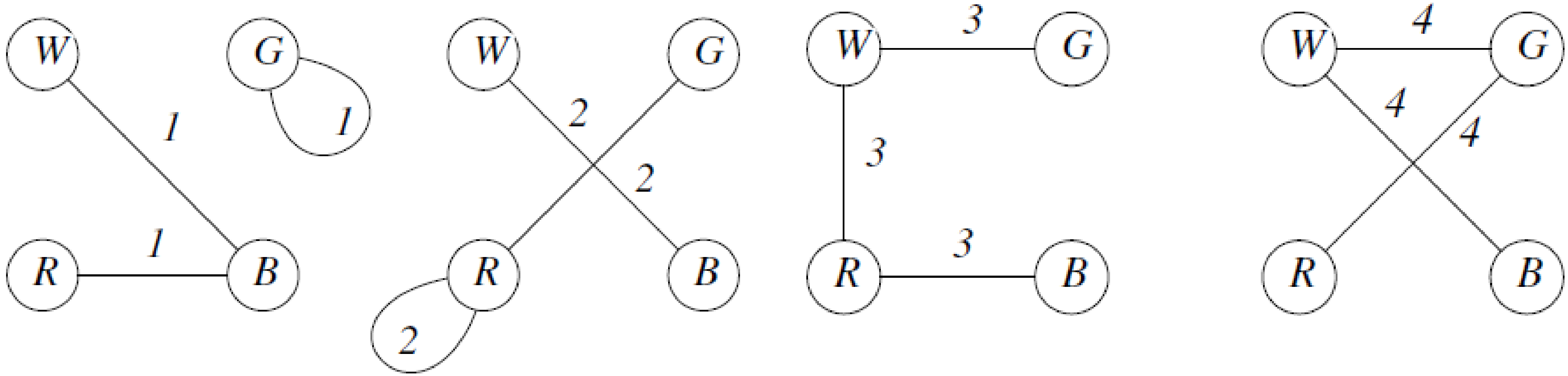
le graphe obtenu pour ces deux faces utilisant la définition de la page précédente est un ensemble de cycles dont les 4 arêtes ont des étiquettes distinctes.

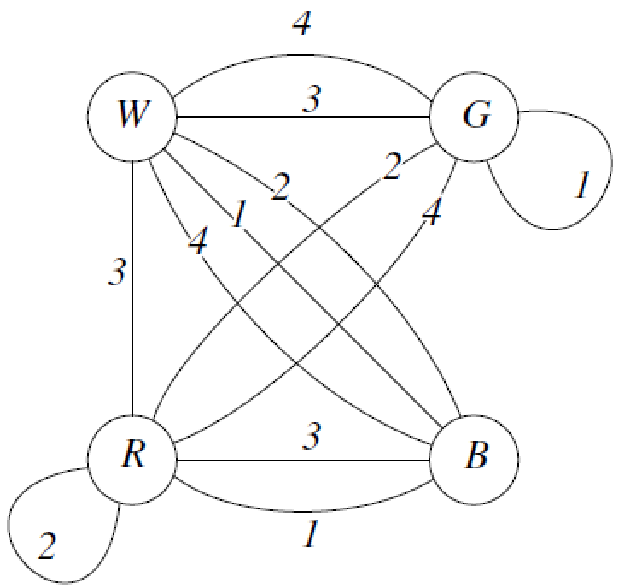
On peut avoir :

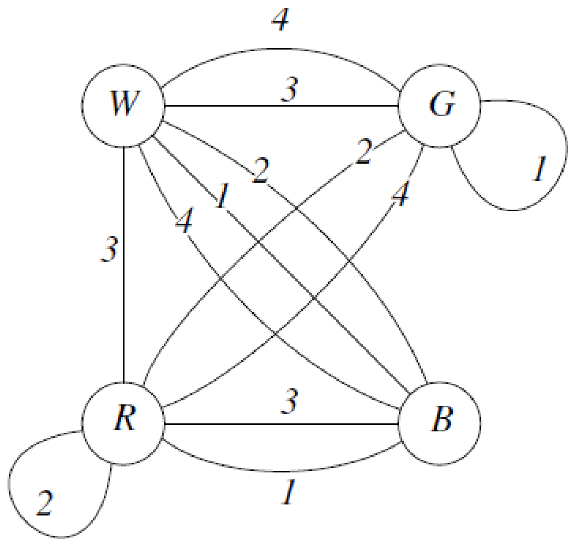
- 1 cycle de longueur 4, ou
- 1 boucle et un cycle de longueur 3, ou
- deux cycles de longueur 2.

Les conditions globales impliquent un traitement globalisé

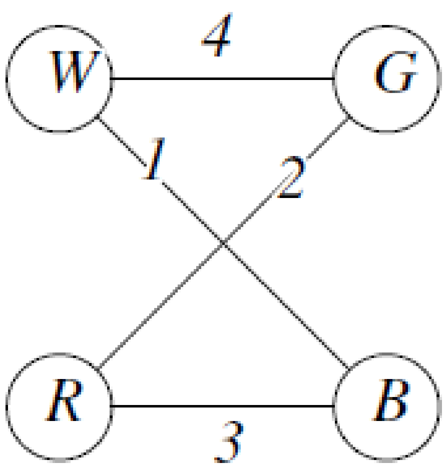
- Construire le multi-graphe qui est l’ « union » des quatre graphes

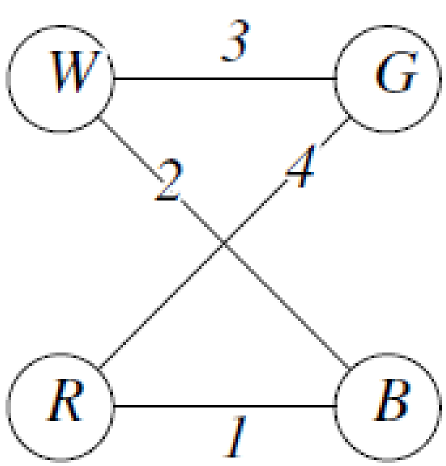




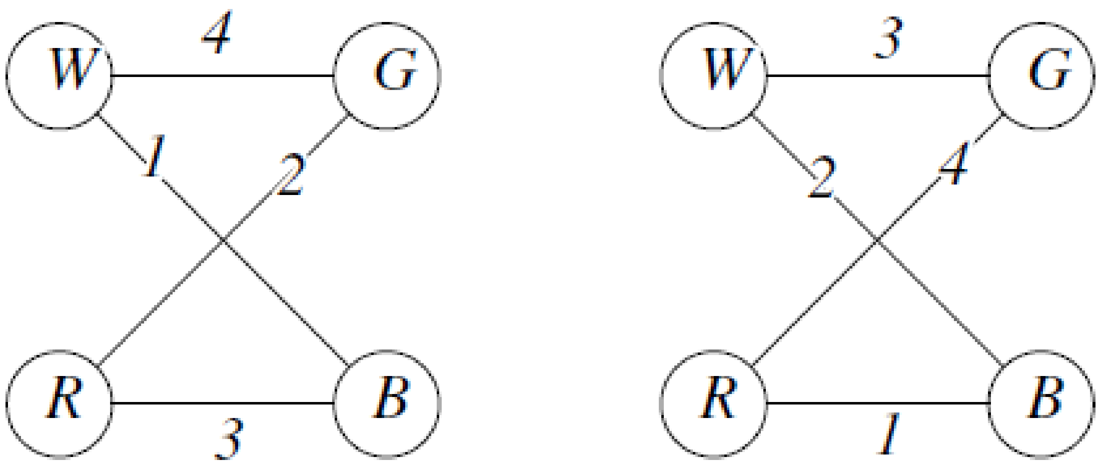


- Trouver deux sous-graphes partiels arête-disjoints qui sont des (ensembles de) cycles et qui ont quatre arêtes chacun, numérotées 1, 2, 3, 4 (cf. observation essentielle précédente):





- On peut construire les faces avant et arrière suivant le graphe de gauche, et (indépendamment) les faces latérales suivant le graphe de droite :



- Cela veut dire (à gauche) : cube 1 avec W avant et B arrière
  cube 2 avec R avant et G arrière
  cube 3 avec B avant et R arrière
  cube 4 avec G avant et W arrière

- Une fois cela fait, tourner verticalement le cube 1 en laissant W et B fixes, pour mettre R à gauche et B à droite etc. selon le 2ème cycle.

- C'est fait ! La preuve en images :

- Décodage : W remplacé par Rouge
  G remplacé par Vert
  B remplacé par Bleu
  R remplacé par Jaune

- Vu dans ce cours introductif:
  - Terminologie
  - Représentations en machine
  - Quelques applications, montrant que l'outil graphe peut être essentiel.

- Par la suite
  - (Sous)Graphes particuliers: arbres, arborescences
  - Problèmes particuliers sur les graphes
    - Plus courts chemins
    - Couplages