U.F.R SCIENCES ET TECHNIQUES
Département d'Informatique
B.P. 1155
64013 PAU CEDEX
Téléphone secrétariat : 05.59.40.79.64
Télécopie : 05.59.40.76.54

TYPE DE STRUCTURES DE GRAPHE

I- NOTIONS DE GRAPHE
II- TYPE GRAPHE ORIENTE
III- REPRESENTATION DE GRAPHE

I- NOTION DE GRAPHE

Beaucoup de données traitées dans la vie courantes sont des **structures relationnelles**.

1- Pourquoi les graphes ?

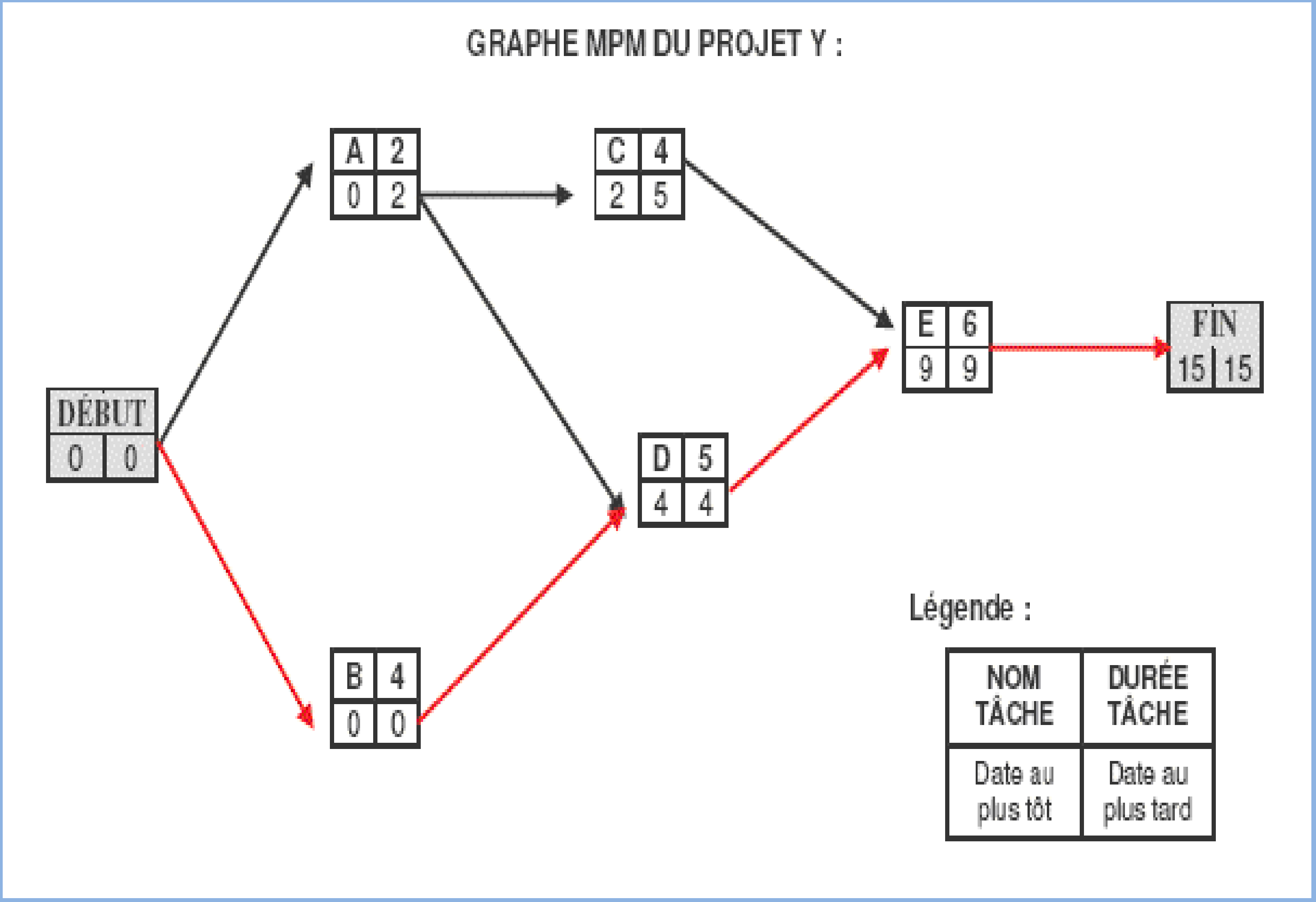
Dans un programme, les structures relationnelles sont modélisées à l'aide des **graphes**.

C’est le cas notamment des:

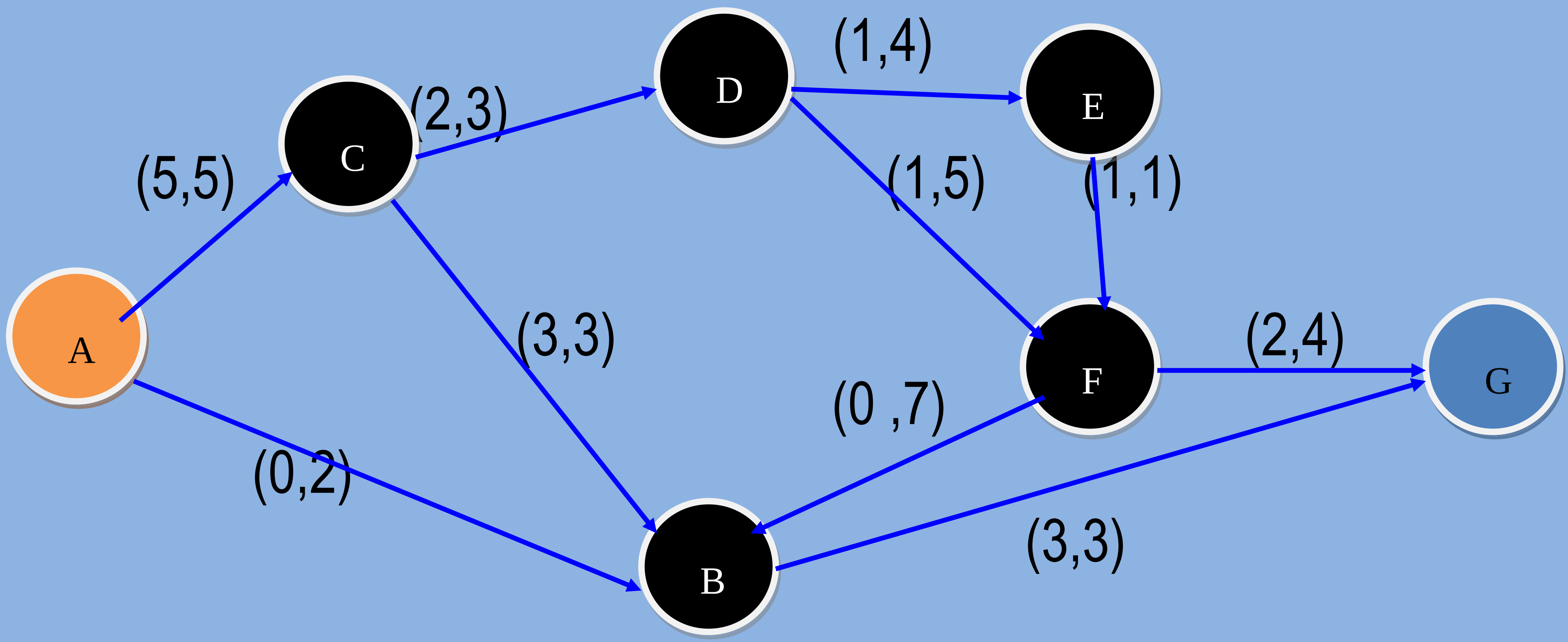
-les réseaux de communication,
-l’ordonnancement des tâches d’un robot,
- la maillage routier d’une région,
- les liaisons aériennes d’un pays,
-...

qui sont des **structures relationnelles.**

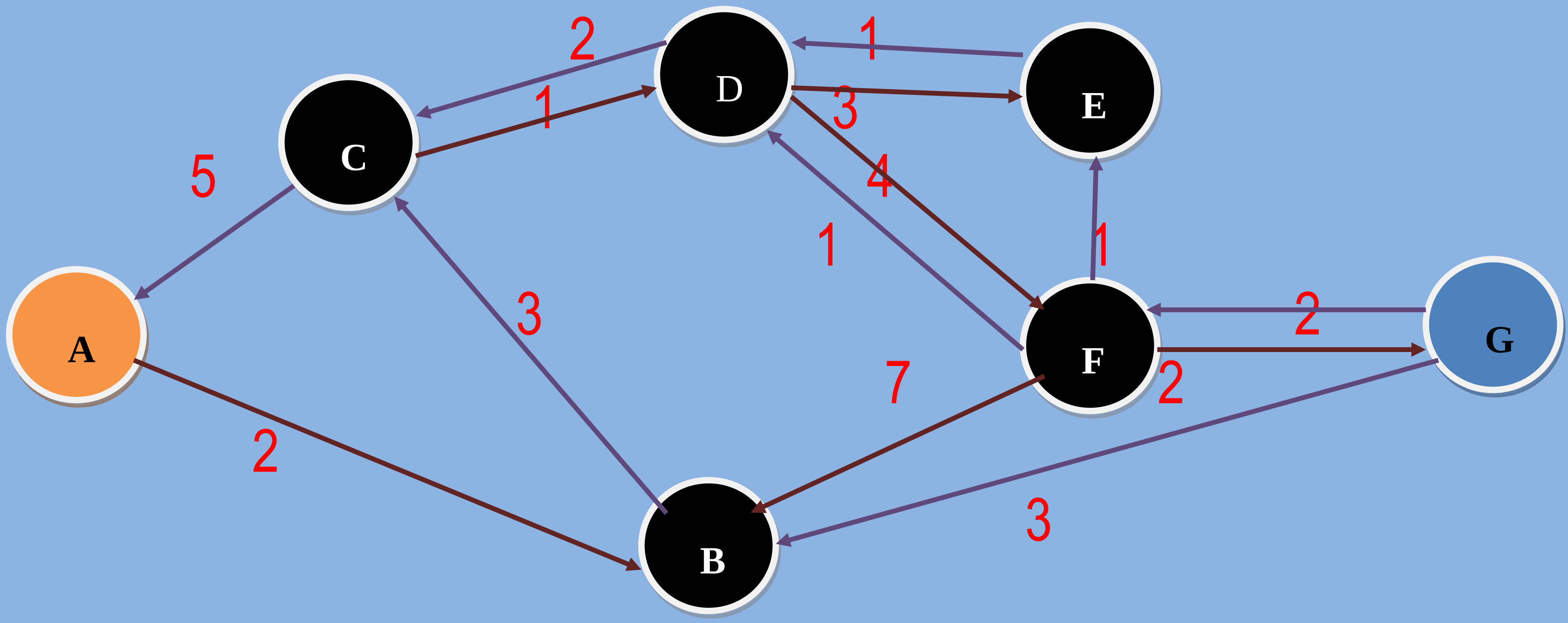
Graphe de planification d'un projet



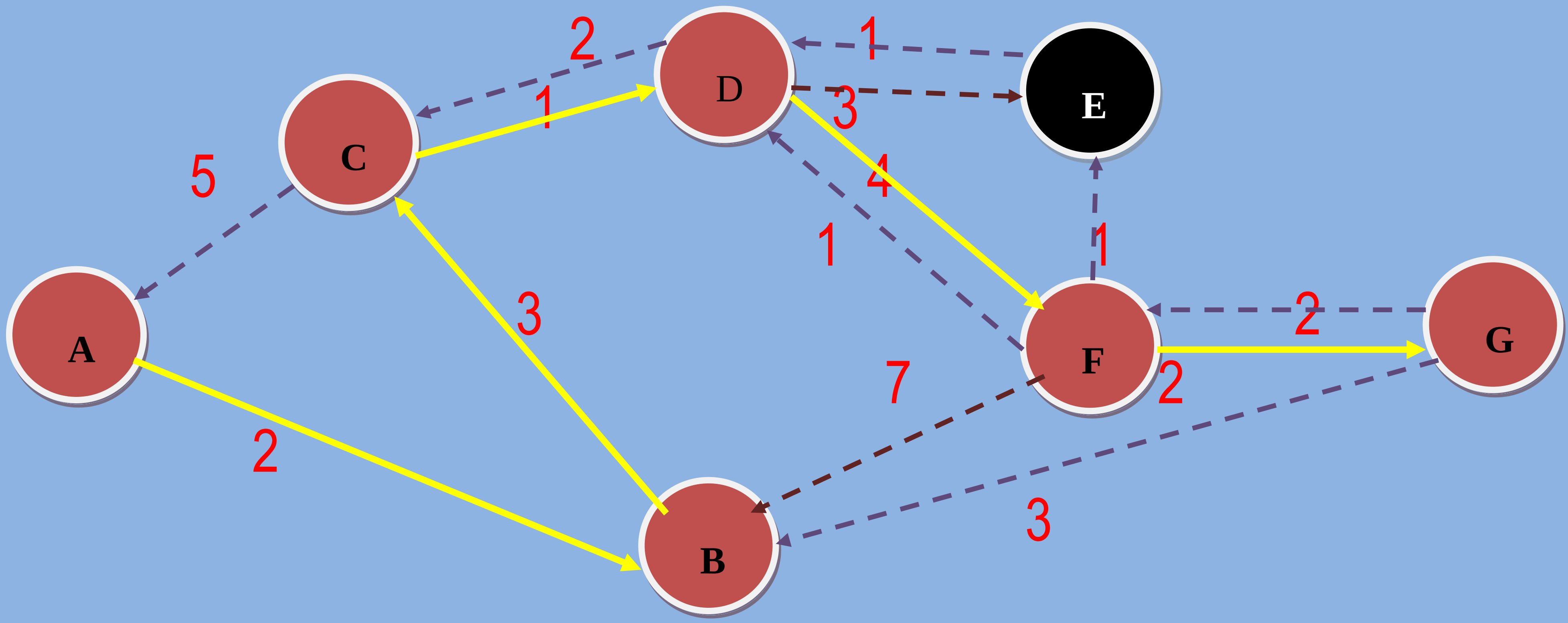
Circulation un flot dans un « réseau transport » .



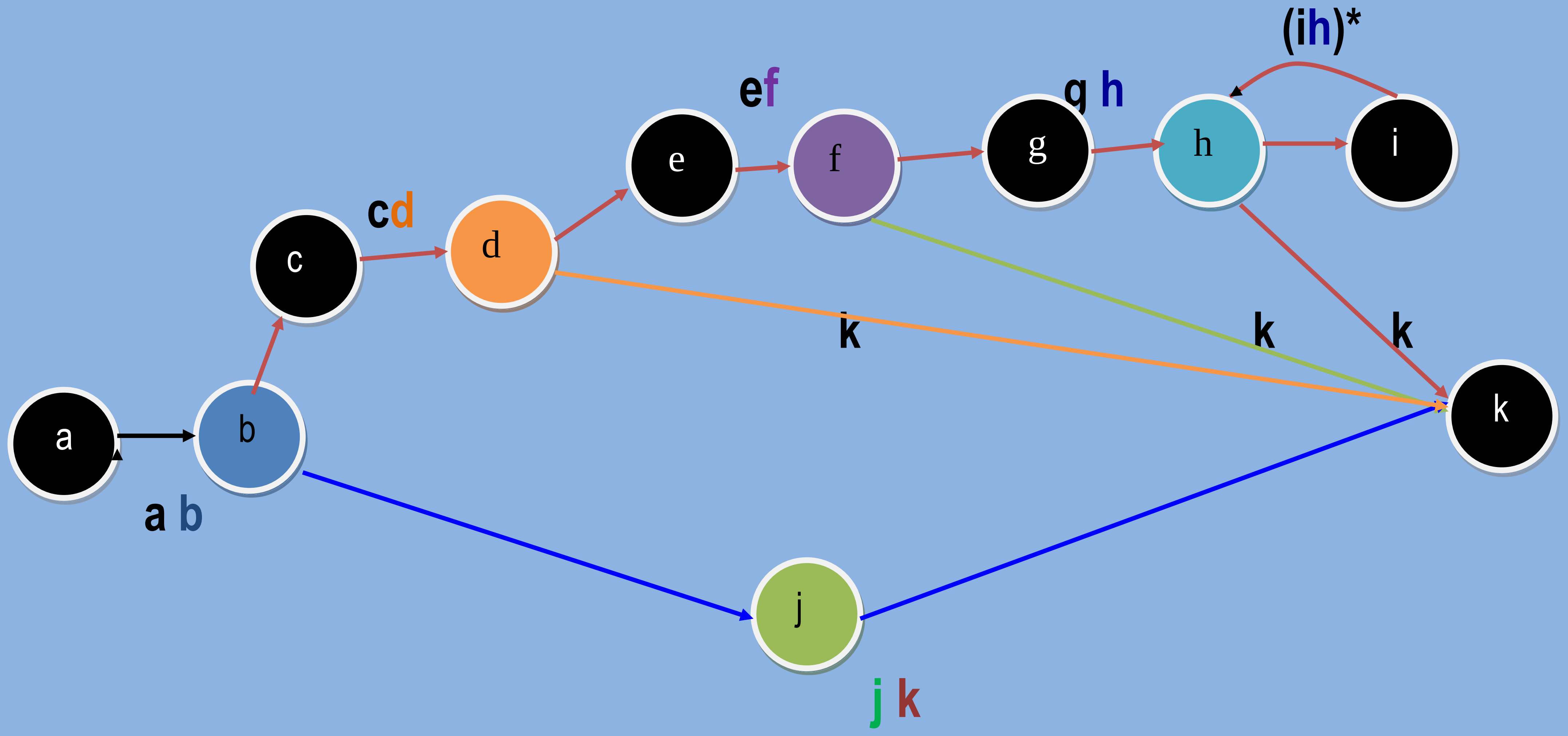
Le graphe d'écart correspondant au réseau



Augmentation du flot

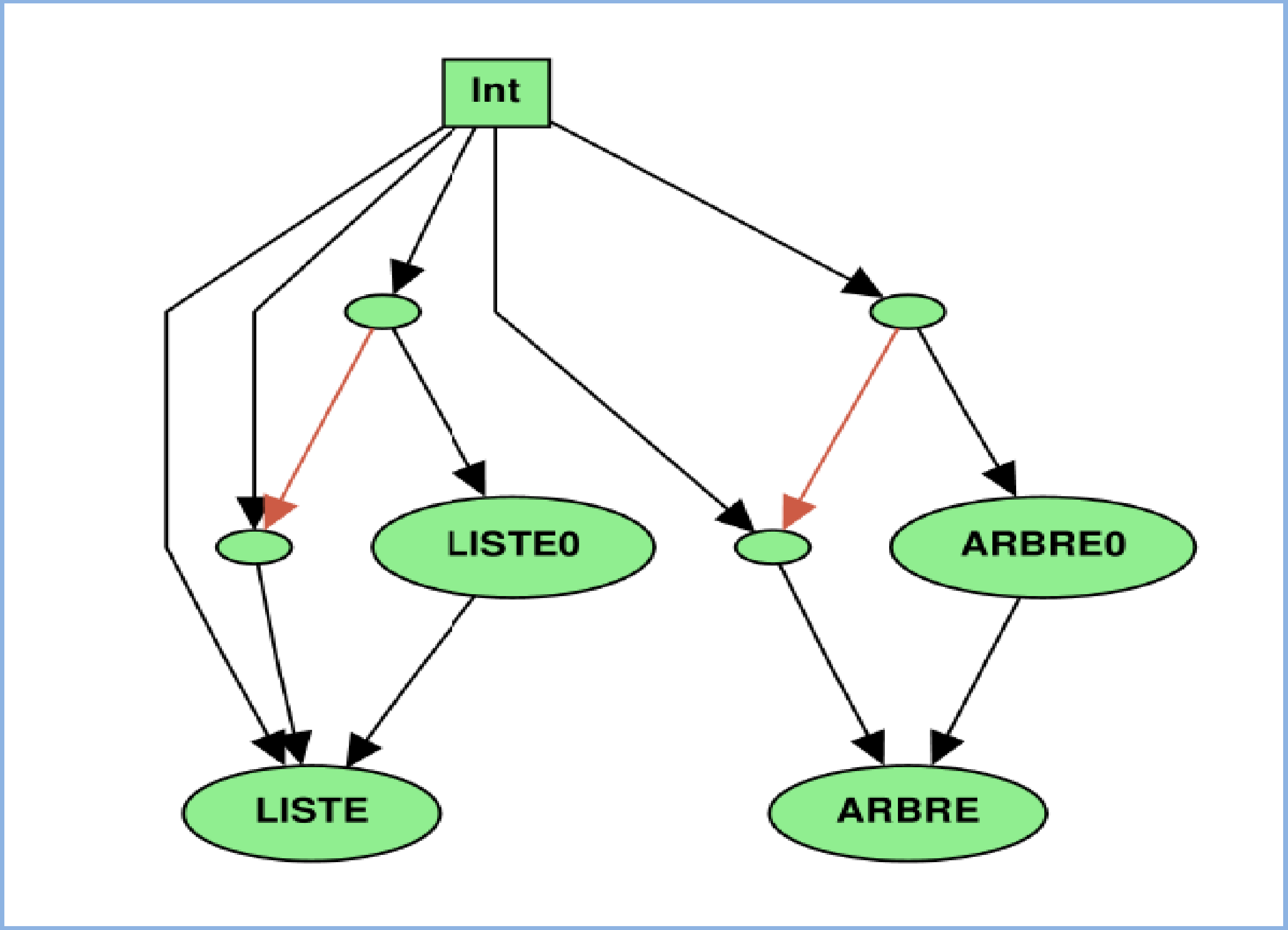


Graphe de contrôle d'un programme

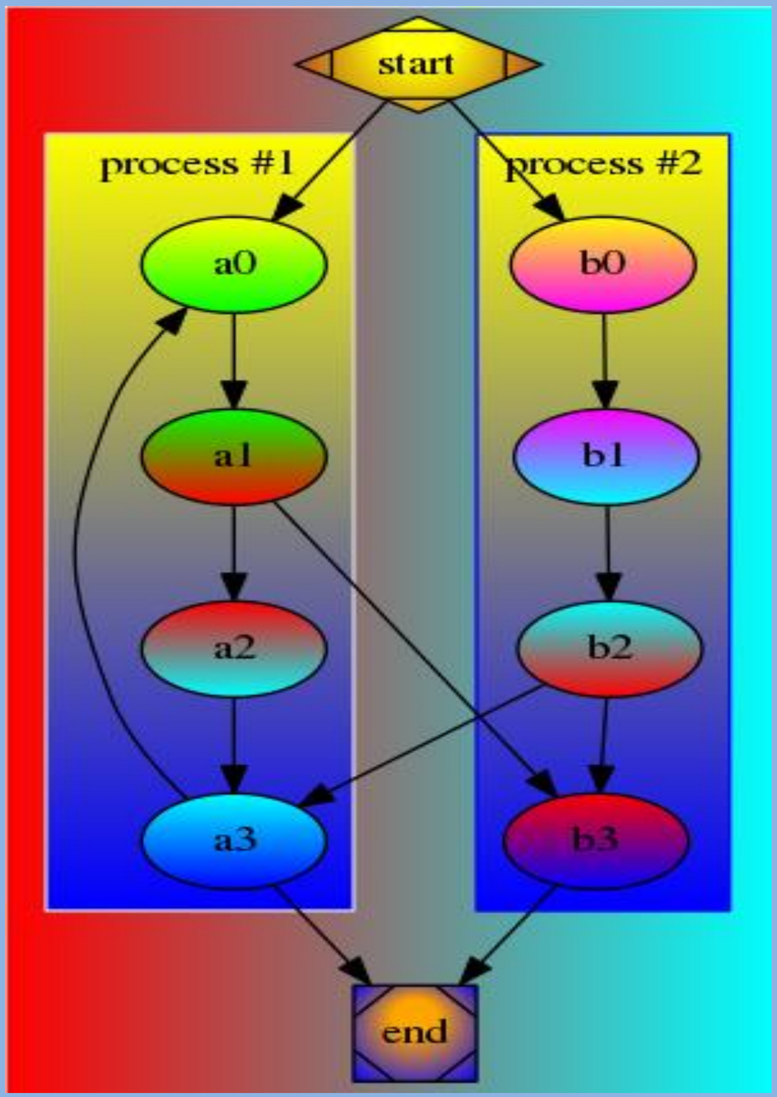


```
Unknown()
      begin
      read(b,c,x)                                                  a
      if b < c    then begin                                       b
                       d:=2*b ; f:=3*c                             c
                       if x>= 0    then begin                      d
                                      y:= x; e:= c                 e
                                      if(y=0)     then begin       f
                                                  a:=f-e           g
                                                  while d<a  begin h
                                                            d:=d+2 i
                                                            end    k
                                                  end              k
                                   end                             k
                  else begin
                       b:=b-1                                      j
                       end                                         k
      end                                                          k
      end
```

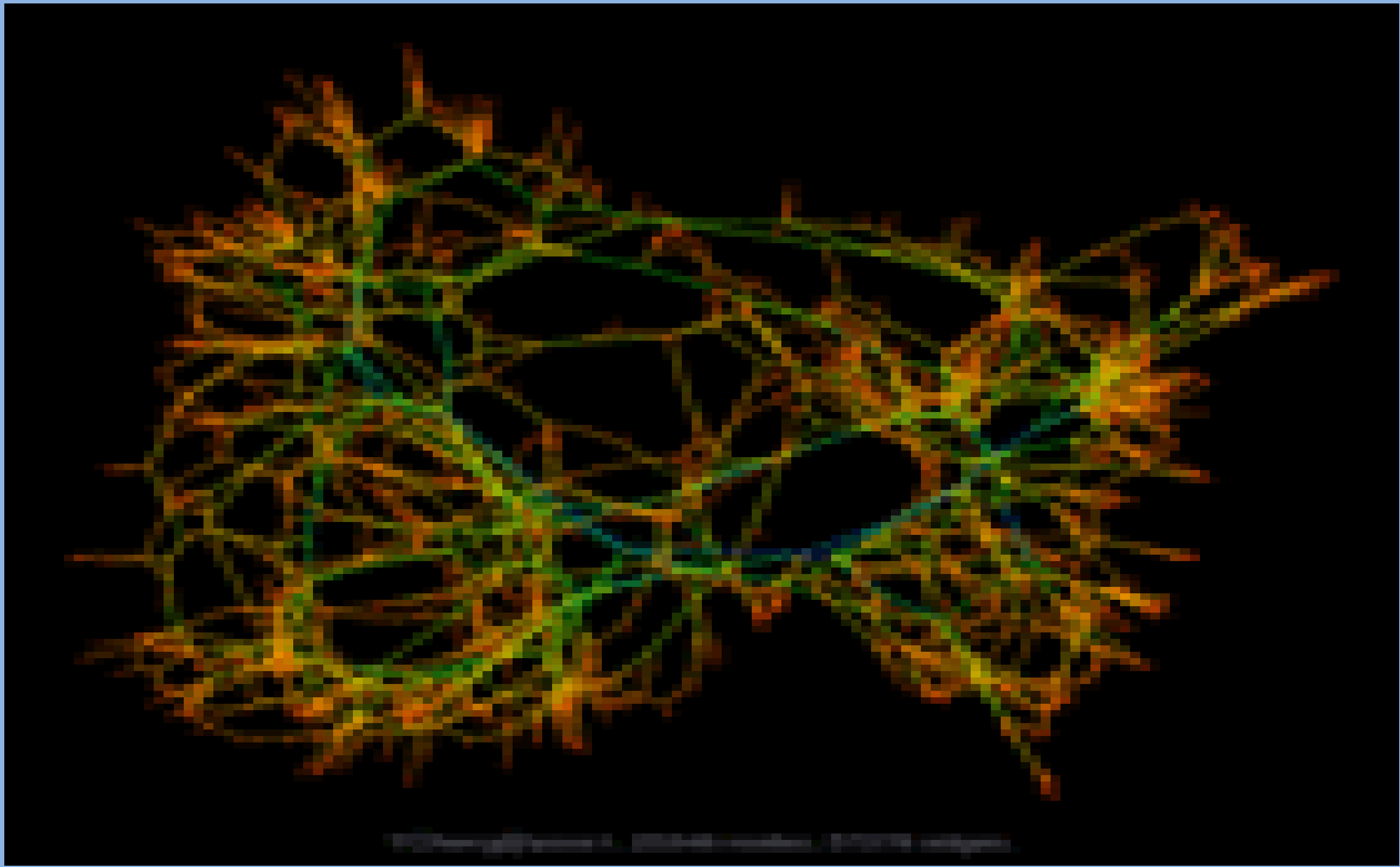
Graphe de preuves de Vinci



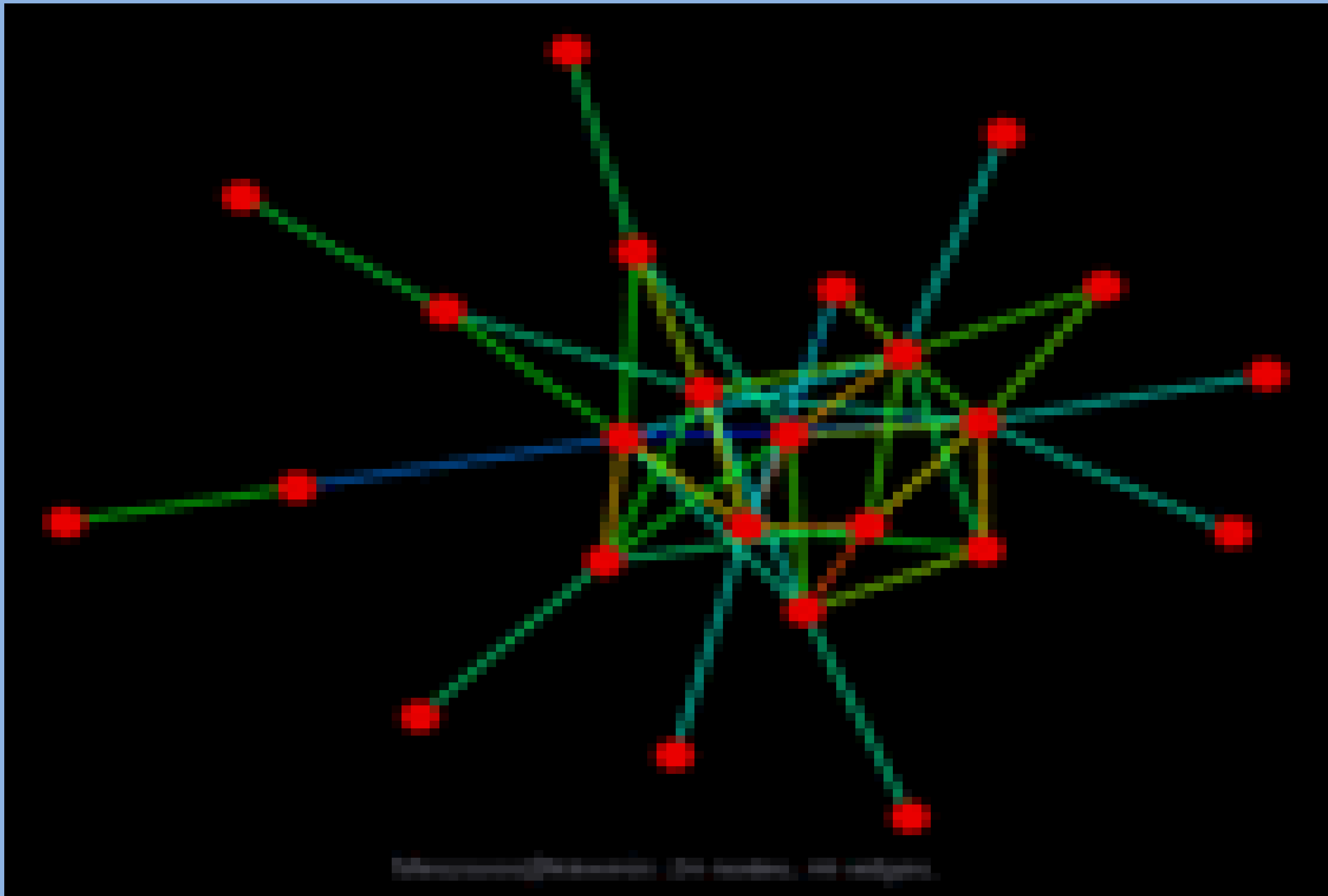
Graphe de processus concurrents



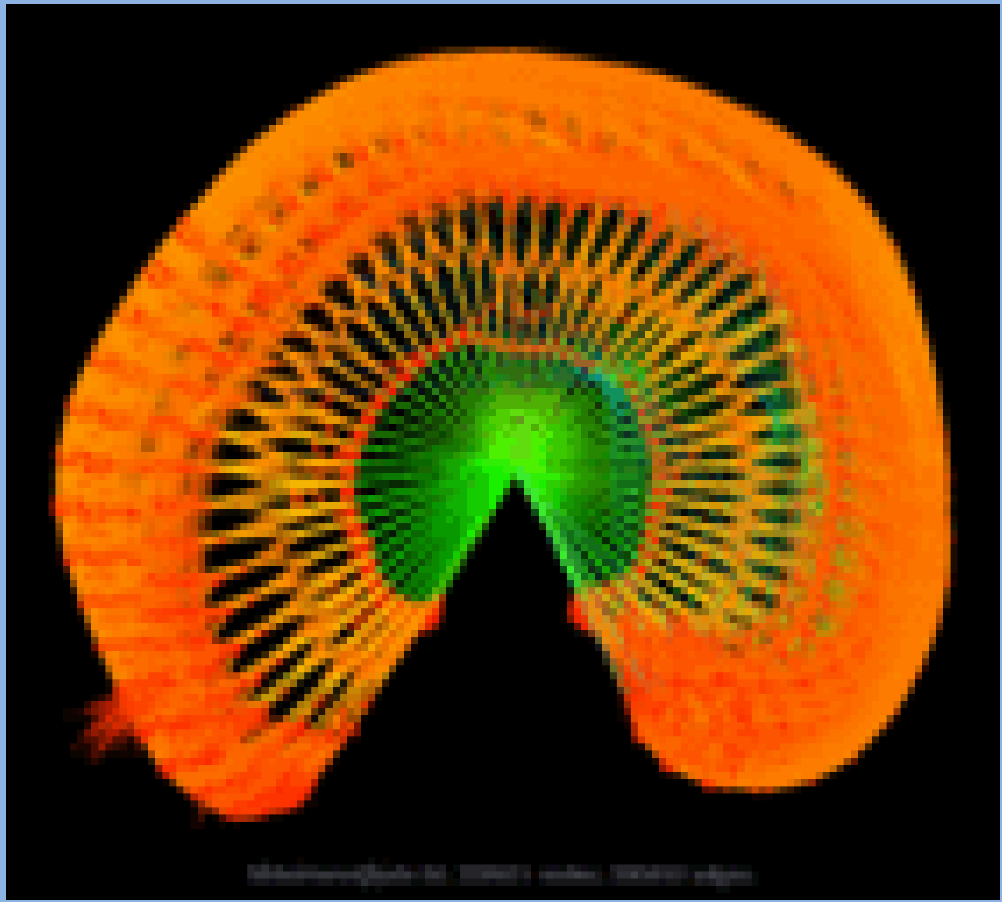
Graphe réseau de neurones



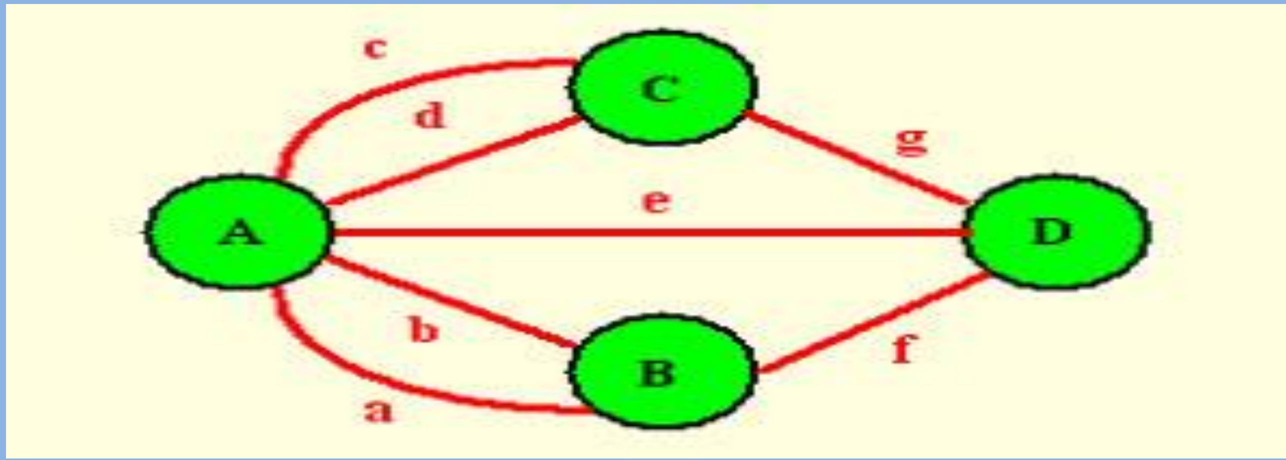
Graphe de trafic aérien



Graphe de numérisation d’une cellule



Modèle d’Euler des 7 ponts de Königsberg



2- Qu'est-ce qu'un graphe ?

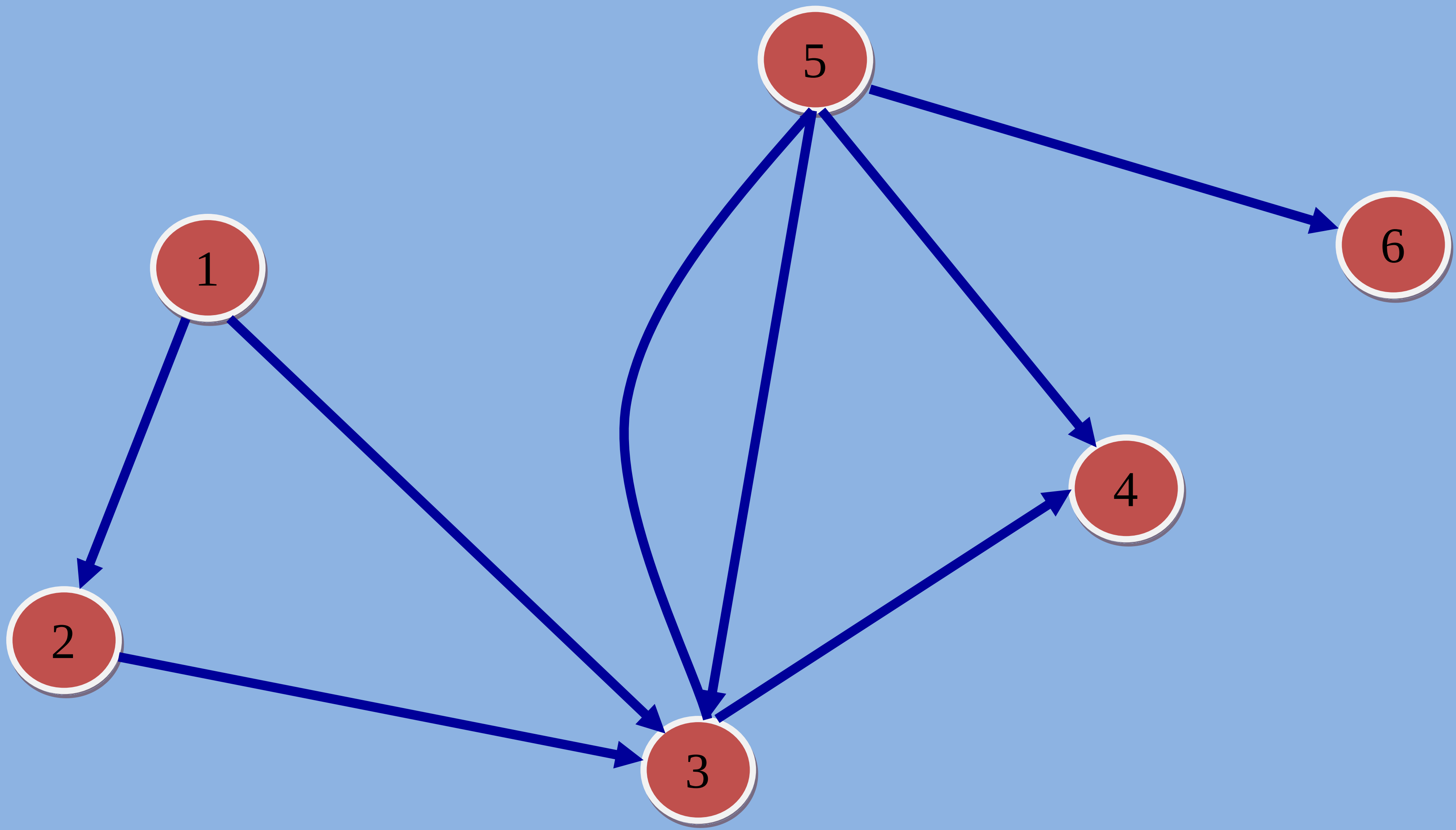
De façon formelle, on appelle graphe G :

- un ensemble S d'objets appelés **noeuds**,
- un ensemble A de **relations** entre ces nœuds.

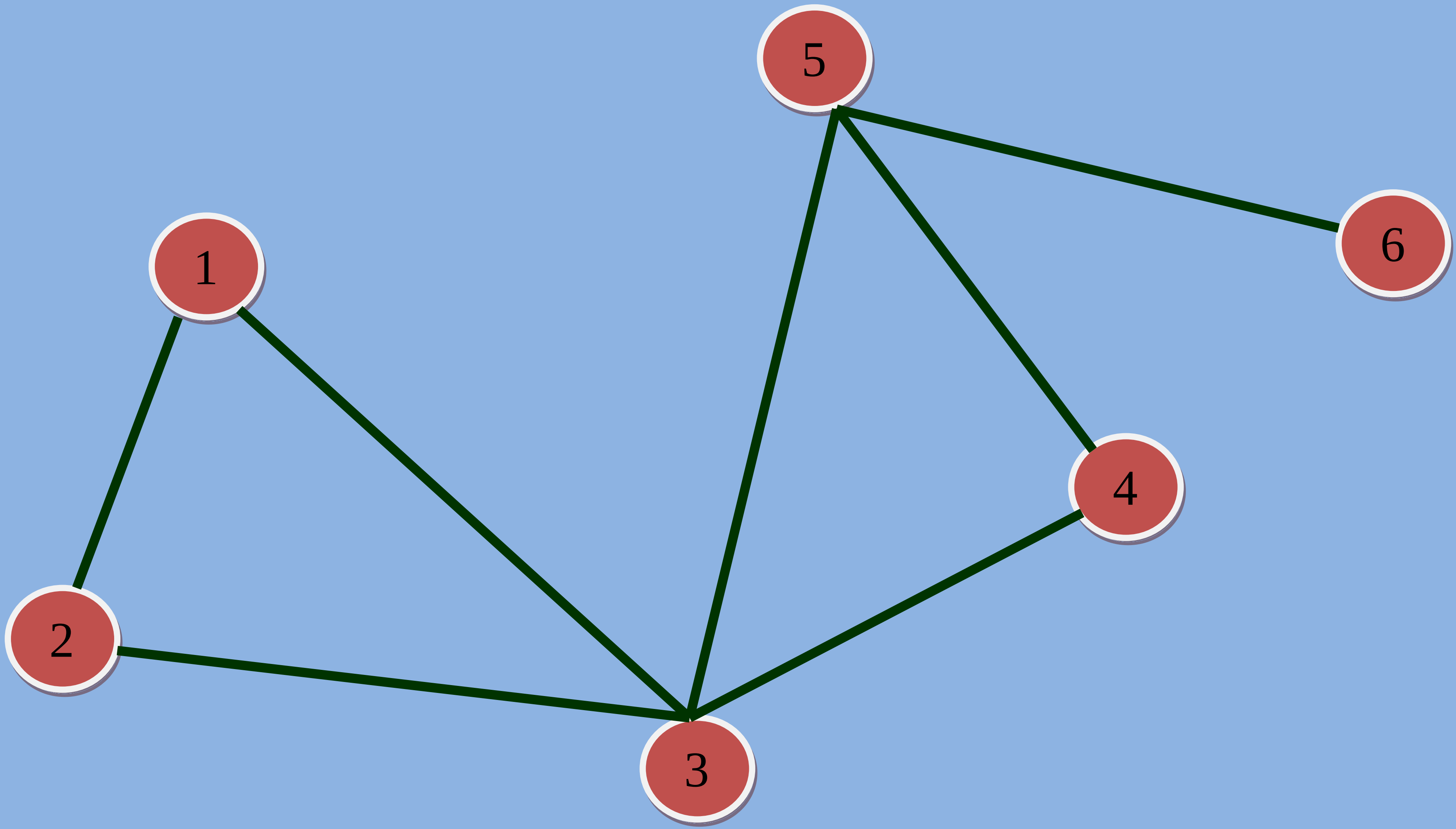
On note habituellement:

G= (S, A)

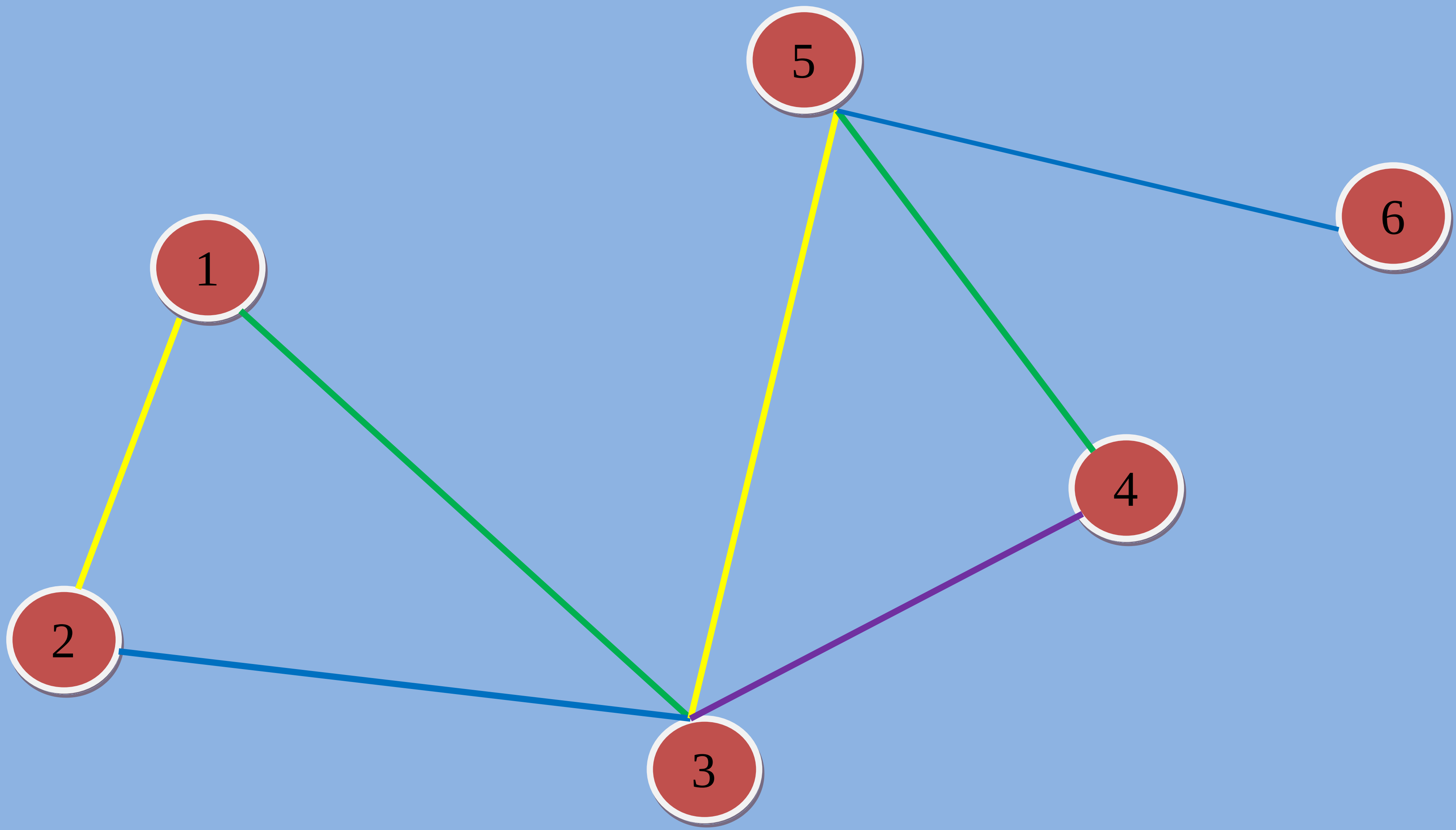
Graphe orienté



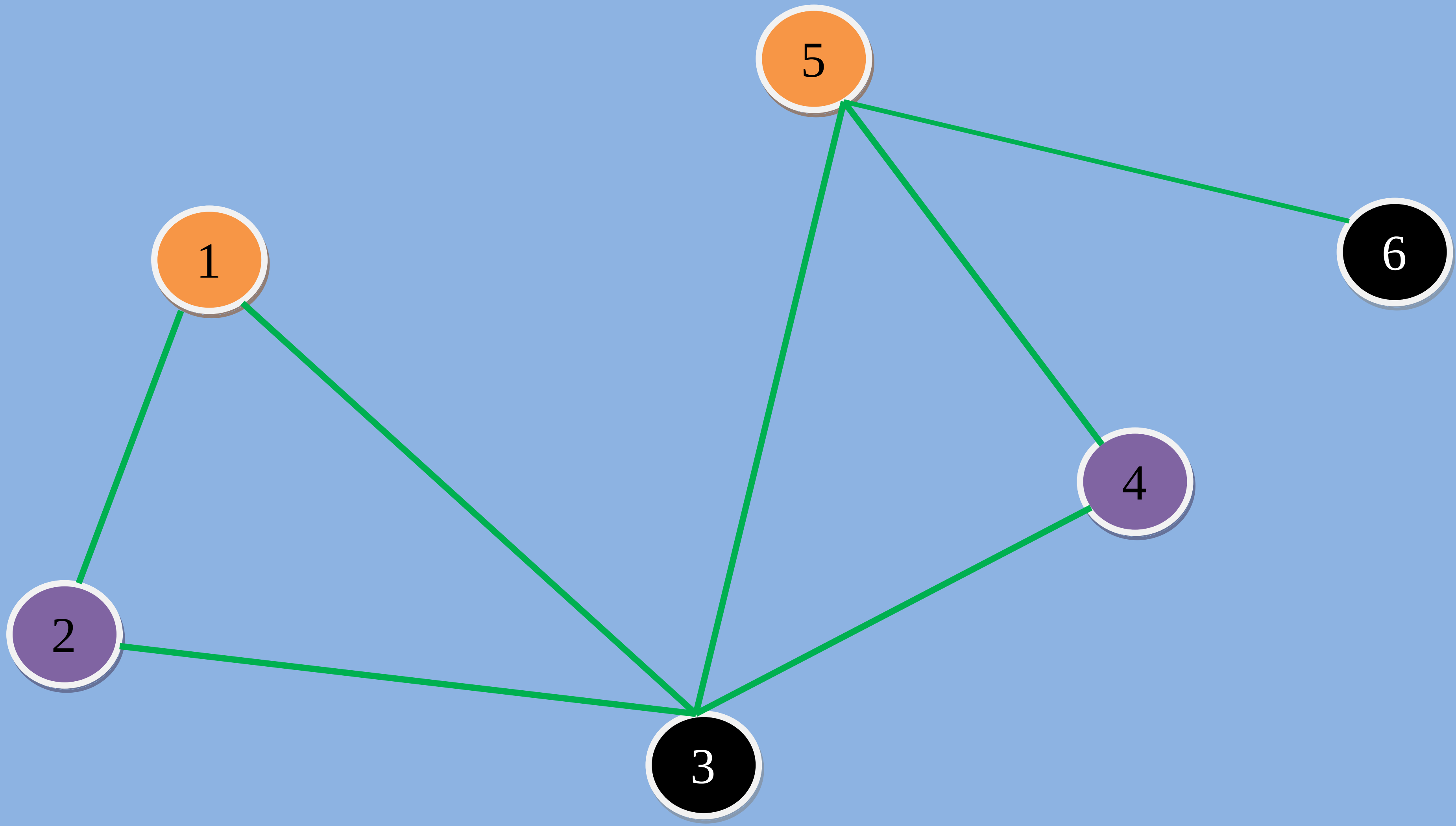
Graphe non orienté



Coloration des arêtes



Coloration des nœuds



Deux hypothèses se présentent:

1- les relations sont **symétriques** : on parle alors de **graphe non orienté**,

2- les relations ne sont **pas symétriques** : on parle alors de **graphe orienté**.

Graphe orienté

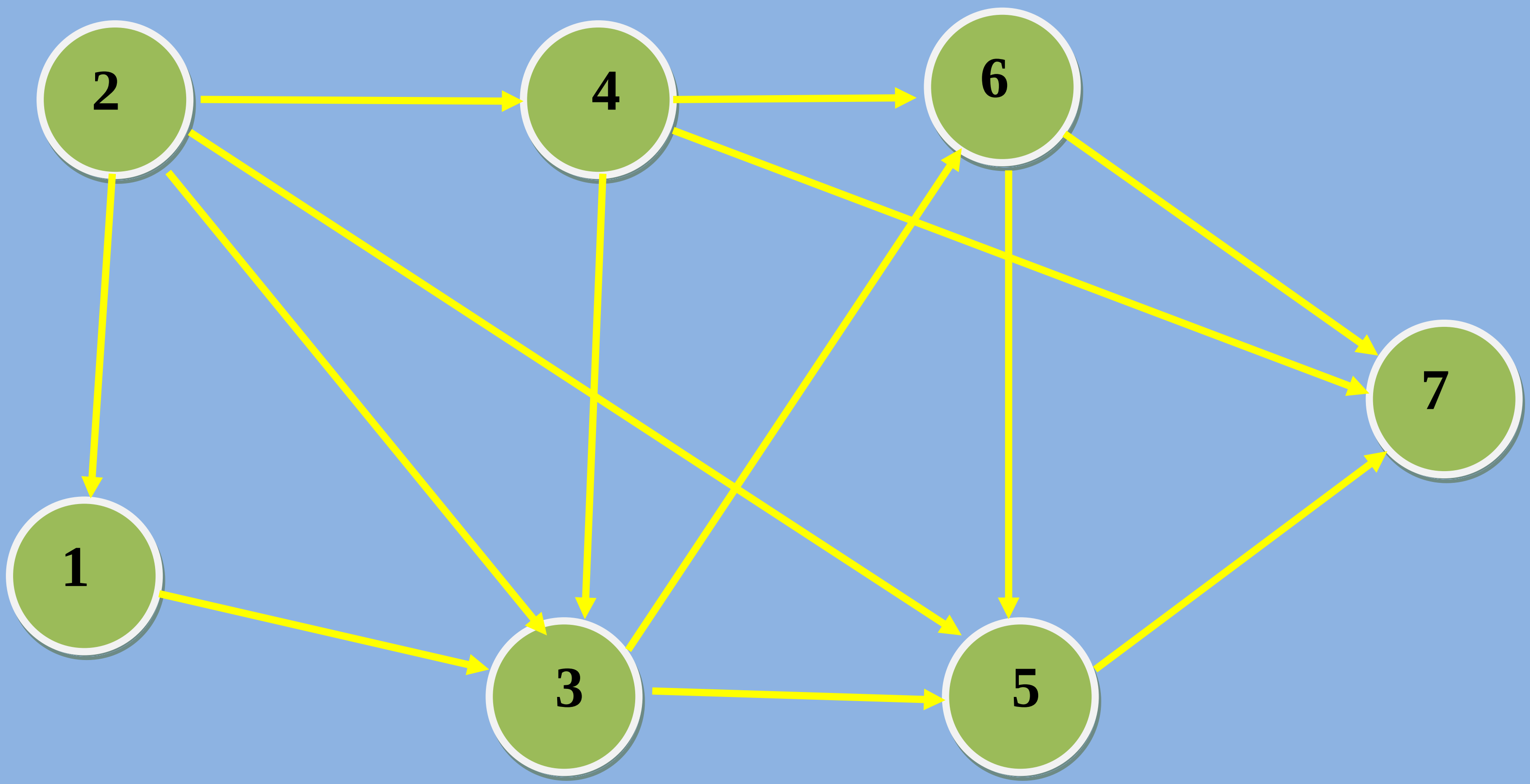
Un graphe orienté G est un couple :

G= (S,A)

Où :

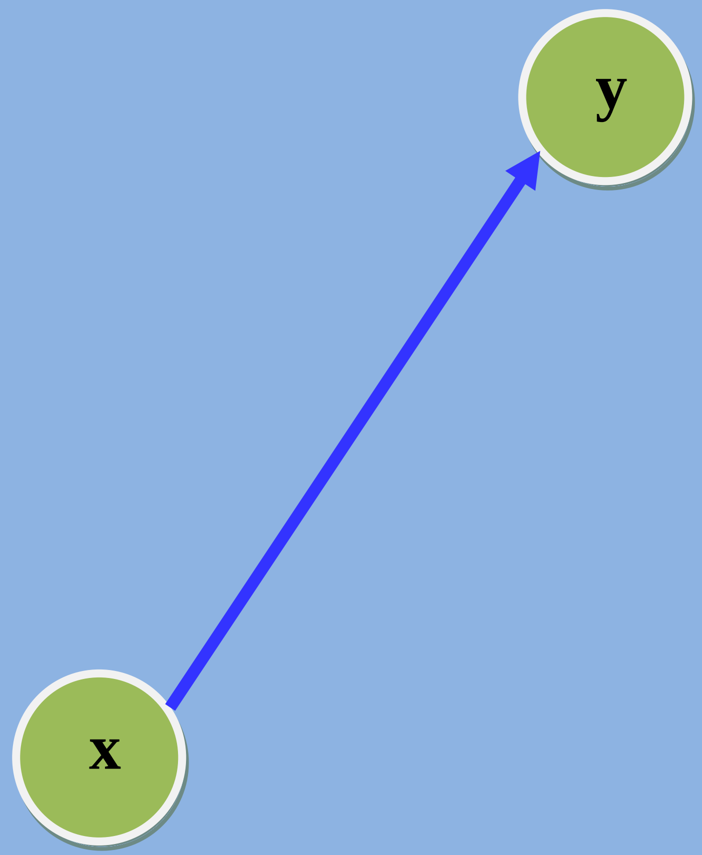
- S ensemble fini de **nœuds**,
- A, un ensemble fini de **paires ordonnées** de nœuds, appelées **arcs**.

Graphe orienté



On note x$\rightarrow$ y l'arc (x,y):

- **x** désigne l'**extrémité initiale**,
- **y** désigne l'**extrémité terminale**.

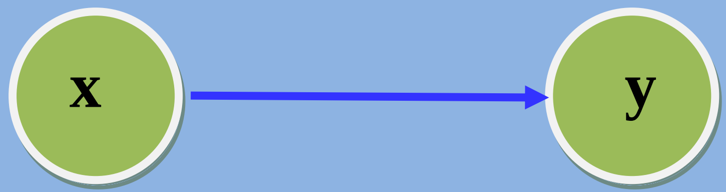


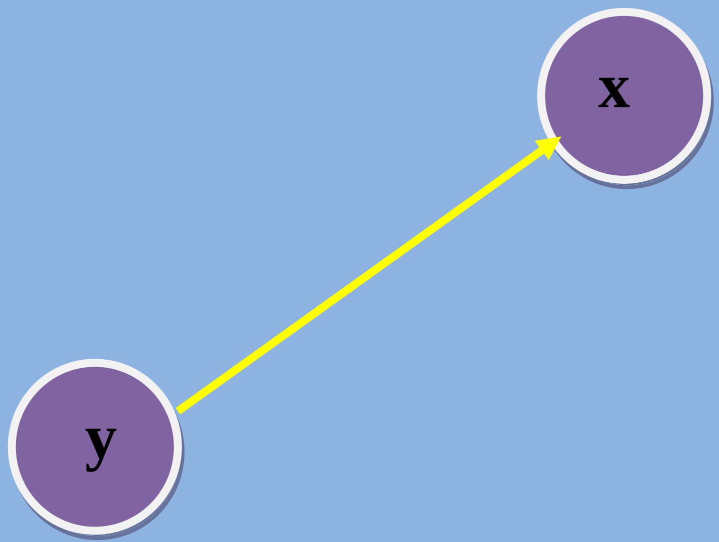
On dit que :

- **y** est le **successeur de** x,
- **x** est le **prédécesseur de** y.

Le nœud **y** est dit **adjacent** à **x** s'il existe un arc

$x \longrightarrow y$ ou $y \longrightarrow x$

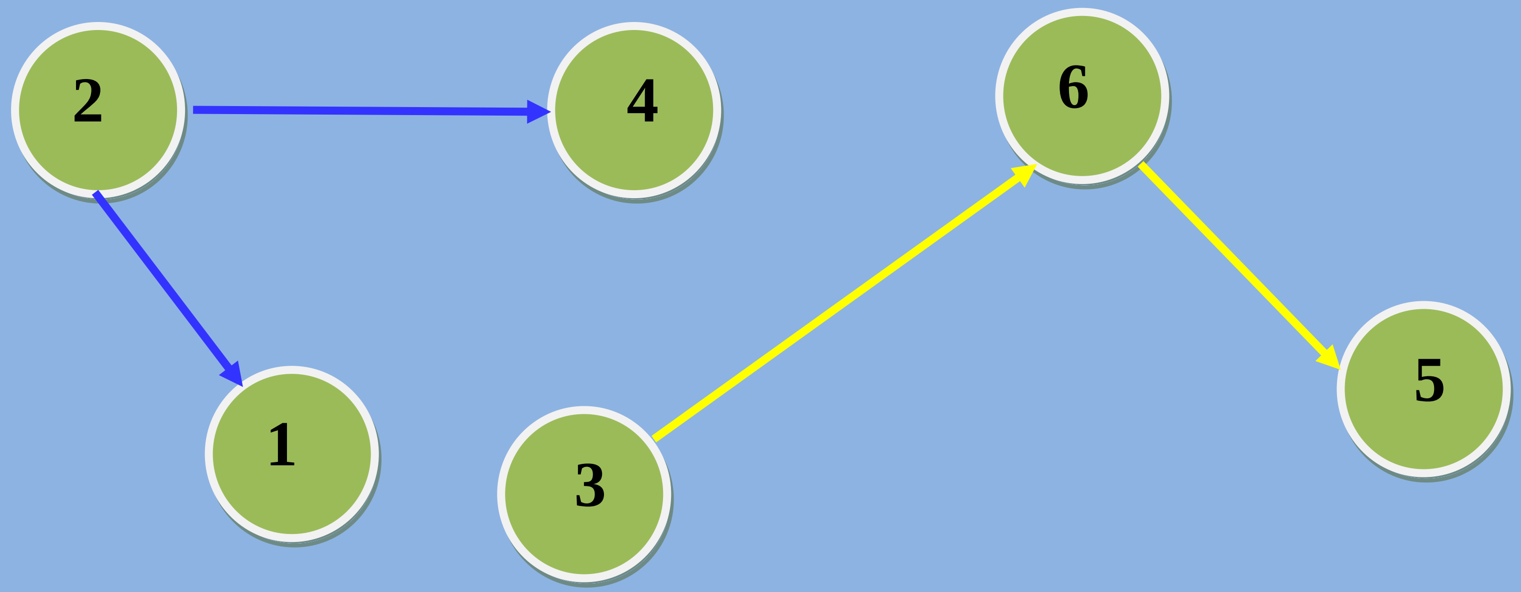




Relation entre arcs et nœuds

Arcs adjacents

Deux arcs sont **adjacents** s'ils ont au moins une **extrémité commune**.

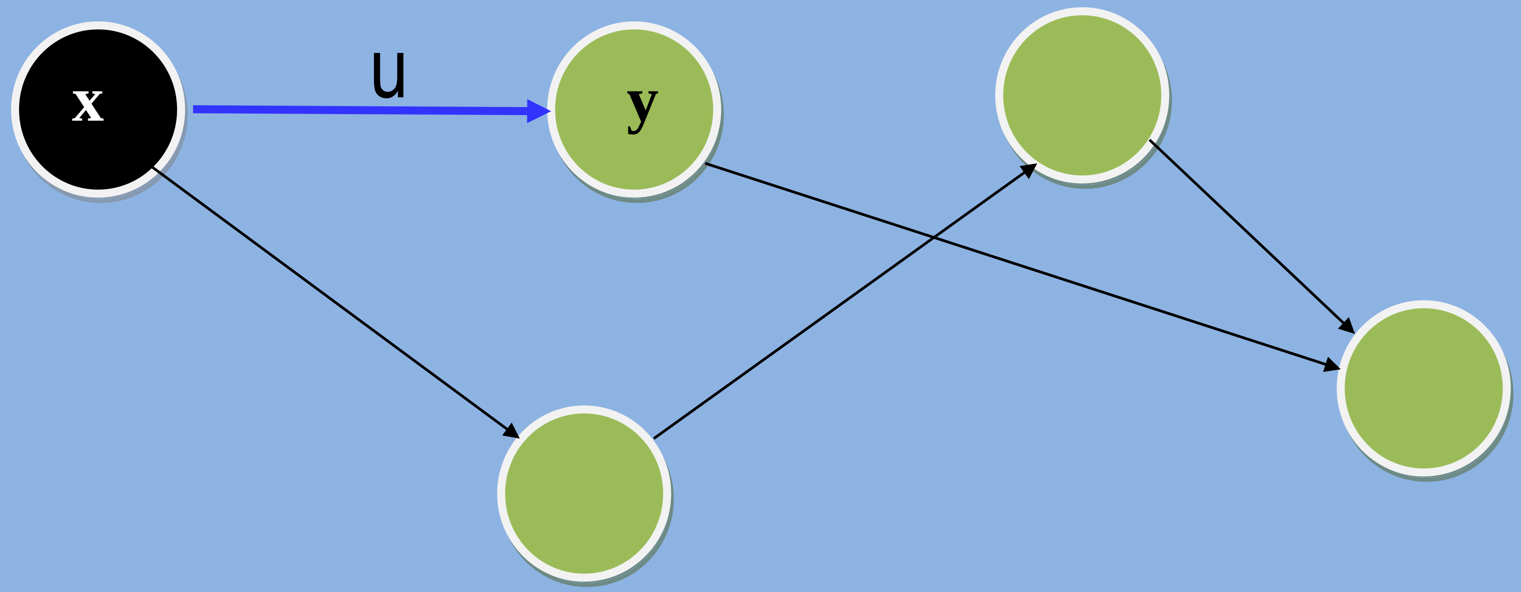


Incidence Arc/nœud

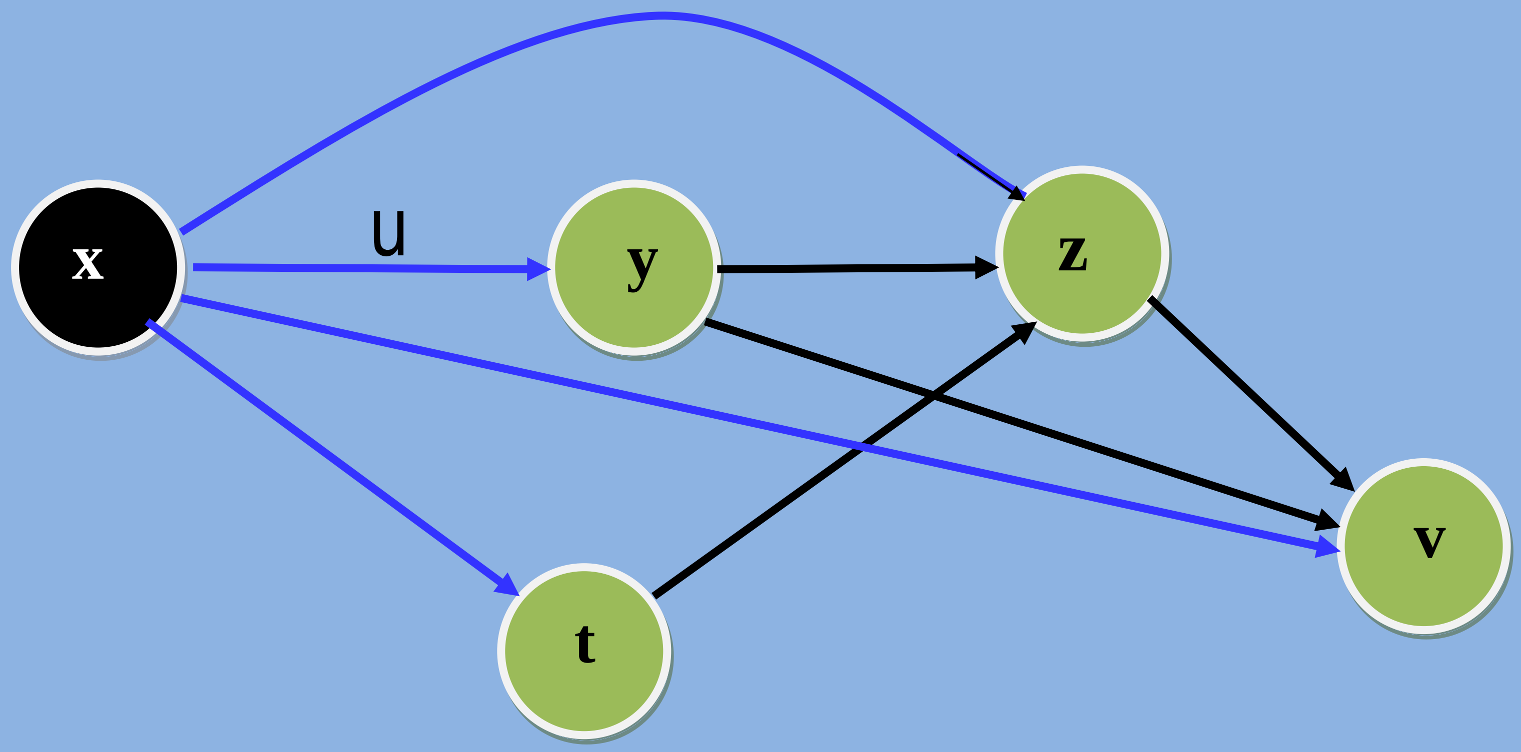
Si le nœud x est l'extrémité initiale d'un arc u :

$$u = x \rightarrow y$$

on dit que u est **incident à x vers l'extérieur**.



Le nombre d’arcs ayant leur extrémité initiale en x est appelé **demi-degré extérieur** ou **demi-degré positif**.



On note $\mathbf{d°^+(x)}$ le demi-degré positif de x.

$$d°^+(x) = 4$$

On définit de même les notions :

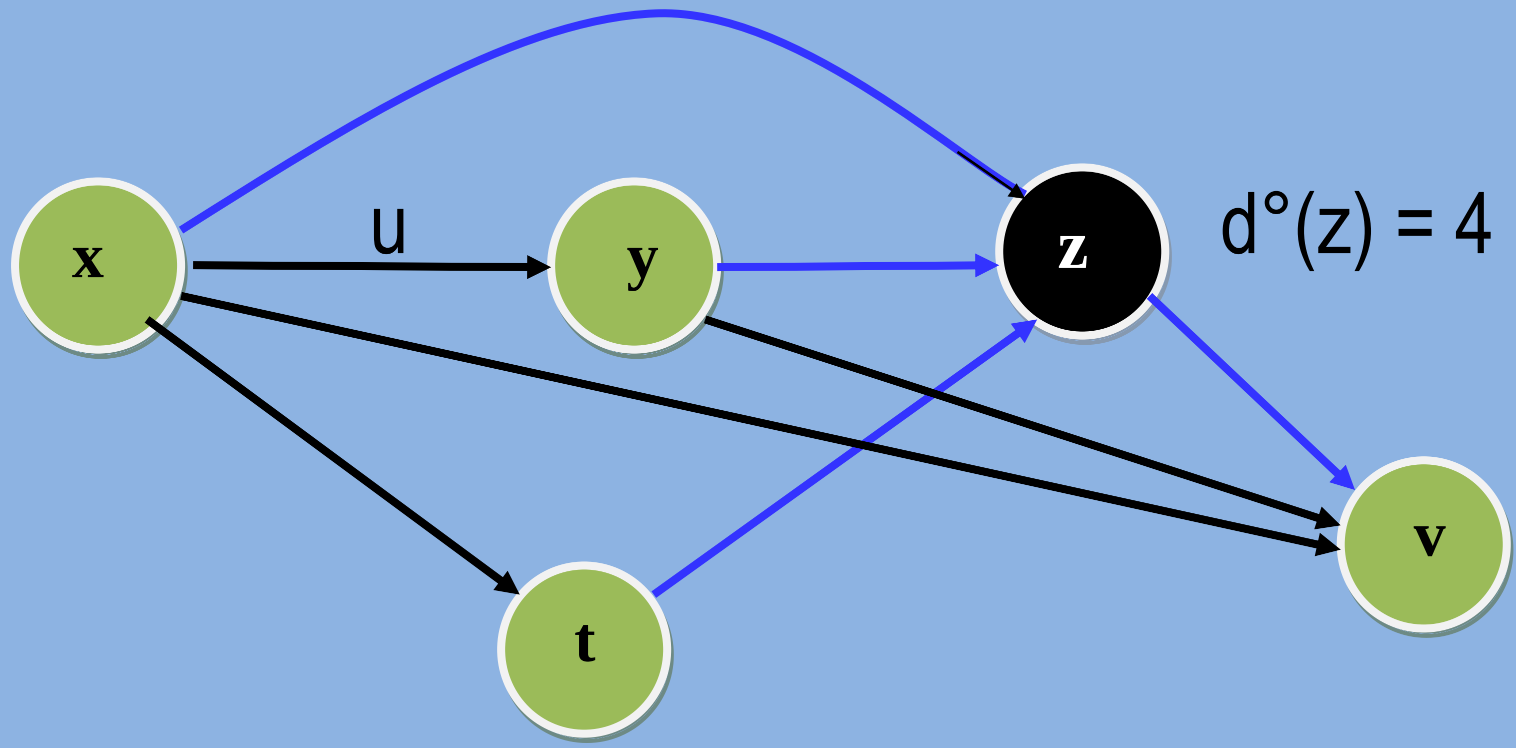
- d'arc **incident vers l'intérieur**,
- de demi-degré **négatif**, noté $\mathbf{d°^-(x)}$.

$$d°^-(v) = 3$$

Dans le cas d'un **graphe orienté**, on :

$$\forall \mathbf{z} \in S \quad \mathbf{d°(z)} = d°^{+}(z) + d°^{-}(z)$$

d°(z**)** est appelé **degré** du nœud **z.**



II- TYPE GRAPHE ORIENTE

Une spécification minimale peut être établie comme suit :

```
spec GRAPHE0[sort Noeud] [sort Arc] =

generated type Graphe::=  grapheVide |
                          addNoeud(Noeud; Graphe) |
                          addArc(Noeud; Noeud; Arc; Graphe) ?

%% avec trois générateurs: grapheVide, AddNoeud, AddArc
```

preds

estNoeudDe : Noeud × Graphe;
estArcDe : Arc × Graphe;

%% prédicats exprimant les propriétés pertinentes %%

ops

origine: Arc × Graphe →? Noeud;
extremite : Arc × Graphe →? Noeud

%% observateurs des arcs %%

$\forall$n, n1, s, t, s1, t1, s2, t2 : Noeud ;
e, e1, e2 : Arc;
g, g': Graphe

- **def addArc**(s, t, e, g) $\Leftrightarrow$ $\neg$ estArcDe(e,g)
- **def origine** (e,g) $\Leftrightarrow$ estArcDe(e,g)
- **def extremite**(e,g) $\Leftrightarrow$ estArcDe(e,g)

- $\neg$ **estNoeudDe**(n, grapheVide)
- **estNoeudDe** (n, addNoeud(n1, g) $\Leftrightarrow$ n = n1 $\vee$ estNoeudDe(n,g)
- **estNoeudDe** (n, addArc(s, t, e, g) $\Leftrightarrow$ n = s $\vee$ n = t
  $\vee$ estNoeudDe(n,g)

- $\neg$ **estArcDe**(e, grapheVide)
- **estArcDe**(e, addNoeud(n, g))$\Leftrightarrow$ estArcDe(e,g)
- **est ArcDe**(e1, addArc(s2, t2, e2, g) $\Leftrightarrow$ e1 = e2 $\vee$ estArcDe(e1,g)

- **origine**(e, addNoeud(n, g)) = origine (e, g)
- **origine** (e1, addArc(s, t, e2, g)) = s when e1 = e2
  else origine (e1,g)

- **extremite**(e, addNoeud(n, g)) = extremite(e,g)
- **extremite**(e1, addArc(s, t, e2, g)) = t when e1 = e2

  else extremite(e1,g)

- g = g' ⇔

  (∀n: Noeud • estNoeudDe(n,g) ⇔ estNoeudDe(n,g')) ∧

  (∀e: Arc • estArcDe(e,g) ⇔ estArcDe(e,g')) ∧

  (∀e: Arc • origine(e,g) = origine(e,g')) ∧

  (∀e: Arc • extremite(e,g) = extremite(e,g'))

end

La spécification précédente peut être enrichie par **extension**.

L'extension ajoute les opérations de suppression de nœuds et d'arcs dans un graphe:

suppNoeud : Noeud × Graphe → Graphe;

suppArc: Arc × Graphe → Graphe

spec GRAPHE[sort Noeud] [sort Arc] =

GRAPHE0 [sort Noeud] [sort Arc]

then

%% extension de la spécification GRAPHE %%

ops

suppNoeud : Noeud × Graphe ⟶ Graphe;

suppArc : Arc × Graphe ⟶ Graphe

$\forall$ n, n1, n2: Noeud ; e, e1, e2 :Arc; g, g': Graphe

- **suppNoeud**(n, **grapheVide**) = grapheVide
- **suppNoeud**(n, **addNoeud**(n1,g))=
  suppNoeud(n,g) when n = n1
  else addNoeud(n1,suppNoeud(n,g))
- **suppNoeud**(n, **addArc**(n1, n2, e, g)) =
  suppNoeud(n,g) when n= n1 $\vee$ n=n2
  else addArc(n1, n2, e, suppNoeud(n,g))

- **suppArc**(e, **grapheVide**) = grapheVide

- **suppArc** (e, **addNoeud**(n1,g)) = addNoeud(n1,suppArc(e,g))

- **suppArc** (e, **addArc**(n1, n2, e1, g)) =
  suppArc(e, g) when e = e1
  else addArc(n1,n2,e1,suppArc(e, g))

end

III- REPRESENTATION D'UN GRAPHE

Il existe deux classes de représentations pour les objets de type GRAPHE:

-la **matrice d'adjacence**,
-les **listes d'adjacence**.

1- Représentation par matrice d'adjacence

Soit un graphe orienté

$$G = (S,A) \quad \text{tel que} \quad |S| = n$$

La technique est centrée sur la **représentation des arcs** du graphe.

Elle permet de représenter G l'aide d'une matrice **M** appelée **matrice d'adjacence**.

Calcul de la matrice d'adjacence

Les lignes et les colonnes de la matrice M représentent les nœuds du graphe G.

Notons M_{ij} l'élément appartement à la ligne **i** et à la colonne **j** de la matrice M.

M est définie par :

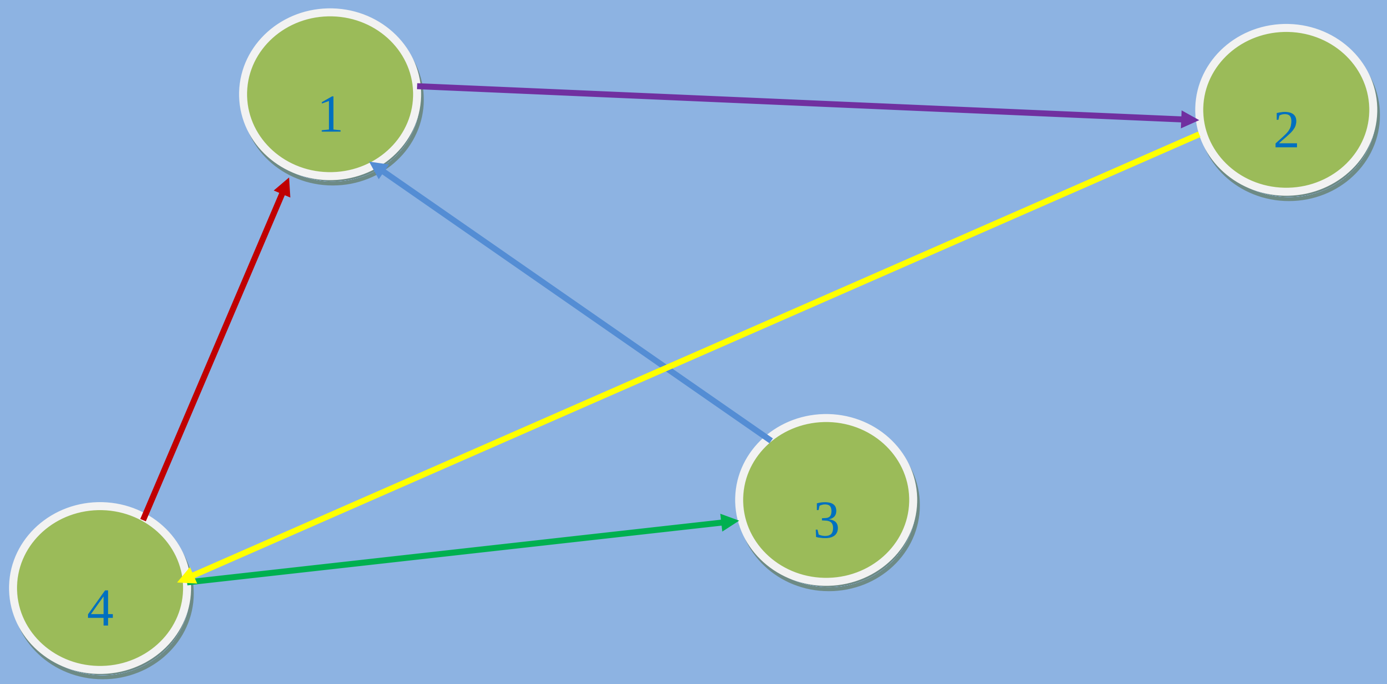
$$M_{ij} = \text{si } \mathbf{i} \rightarrow \mathbf{j} \in A \text{ alors } 1 \text{ sinon } 0$$

La matrice d’adjacences est une matrice **binaire carrée** d’ordre n.

Il n'y a que des zéros sur la diagonale.

La présence d’un 1 sur la diagonale indiquerait une **boucle** : ce qui est interdit par convention.

Le graphe suivant :



est représenté par la matrice :

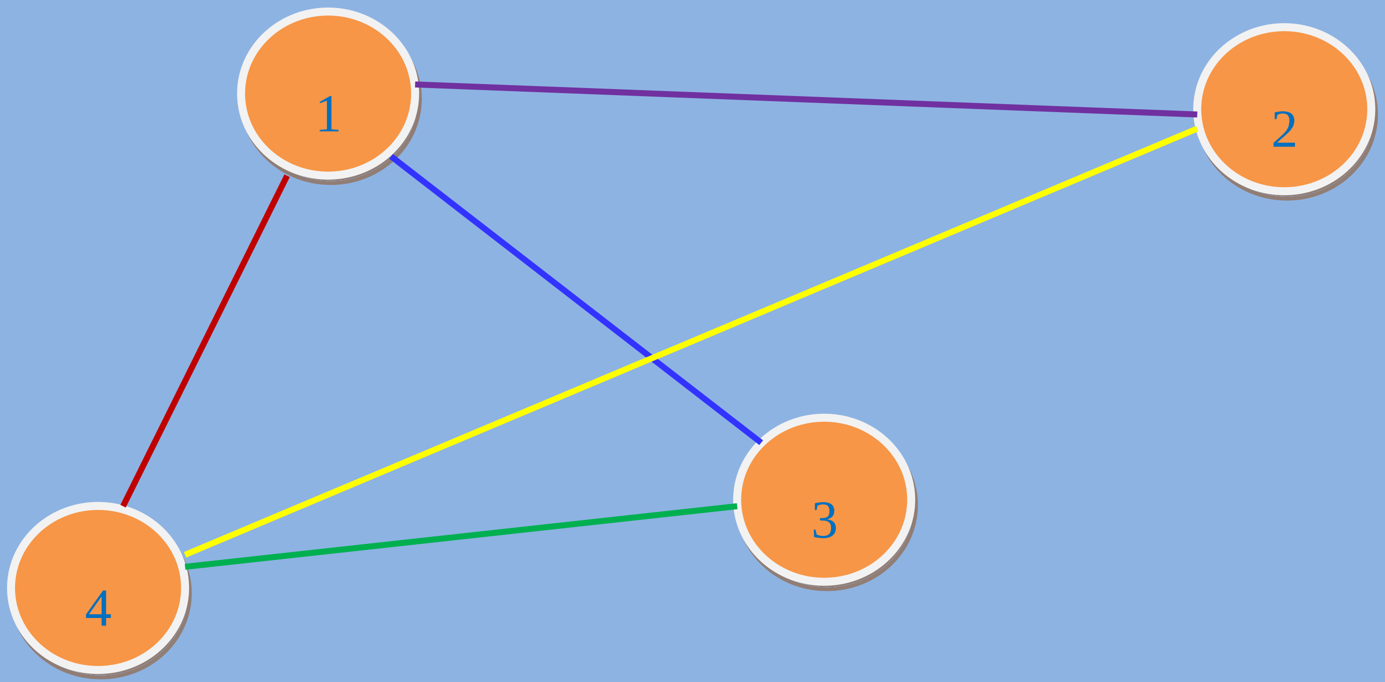
$$M = \begin{array}{cccc} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{array}$$

Remarque :

La matrice d'adjacence d'un graphe **non orienté** est symétrique:

$$M_{ij} = M_{ji}$$

Le graphe suivant:



est représenté par la matrice :

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

Avantages

1- La représentation matricielle est pratique pour **tester l'existence** d'un arc (arête).

2- Il est plus facile d'**ajouter** ou **retirer** un arc (arête).

3- Il est facile de **parcourir** les successeurs ou prédécesseurs d'un nœud.

Inconvénient

1- Il demande **n tests** pour détecter les successeurs ou prédécesseurs d'un nœud **s** quel que soit leur nombre.

2- Il en est de même du **calcul de** $d°^{+}$ **et** $d°^{-}$ de s.

3- La **consultation complète** de la matrice de dimension **n** requiert un **temps** d'ordre n^2.

4- La représentation matricielle exige un **espace mémoire** de O($\mathbf{n}^2$)

5- Cela interdit d'avoir des algorithmes d'ordre inférieur à $\mathbf{n}^2$ pour des graphes à **n** nœuds n'ayant que **peu d'arcs** (arêtes).

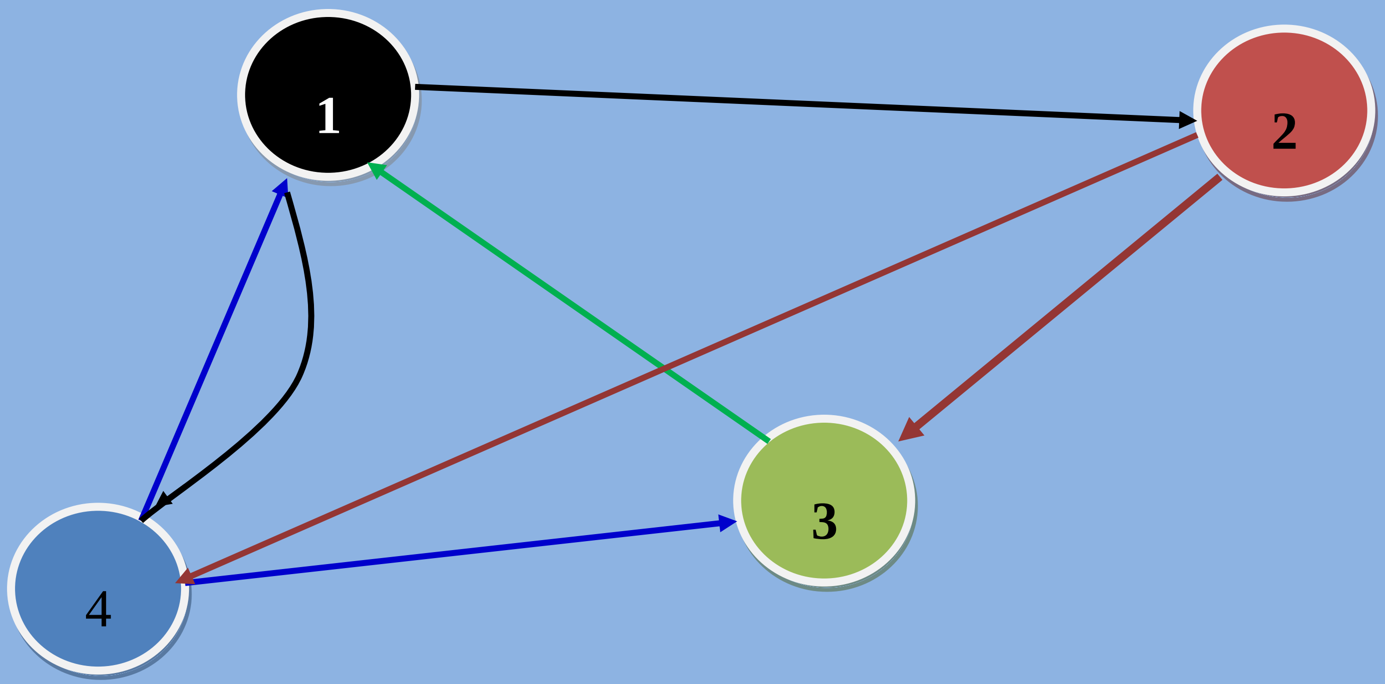
2- Représentation par liste d'adjacence

Cette technique est centrée sur la représentation des **nœuds**.

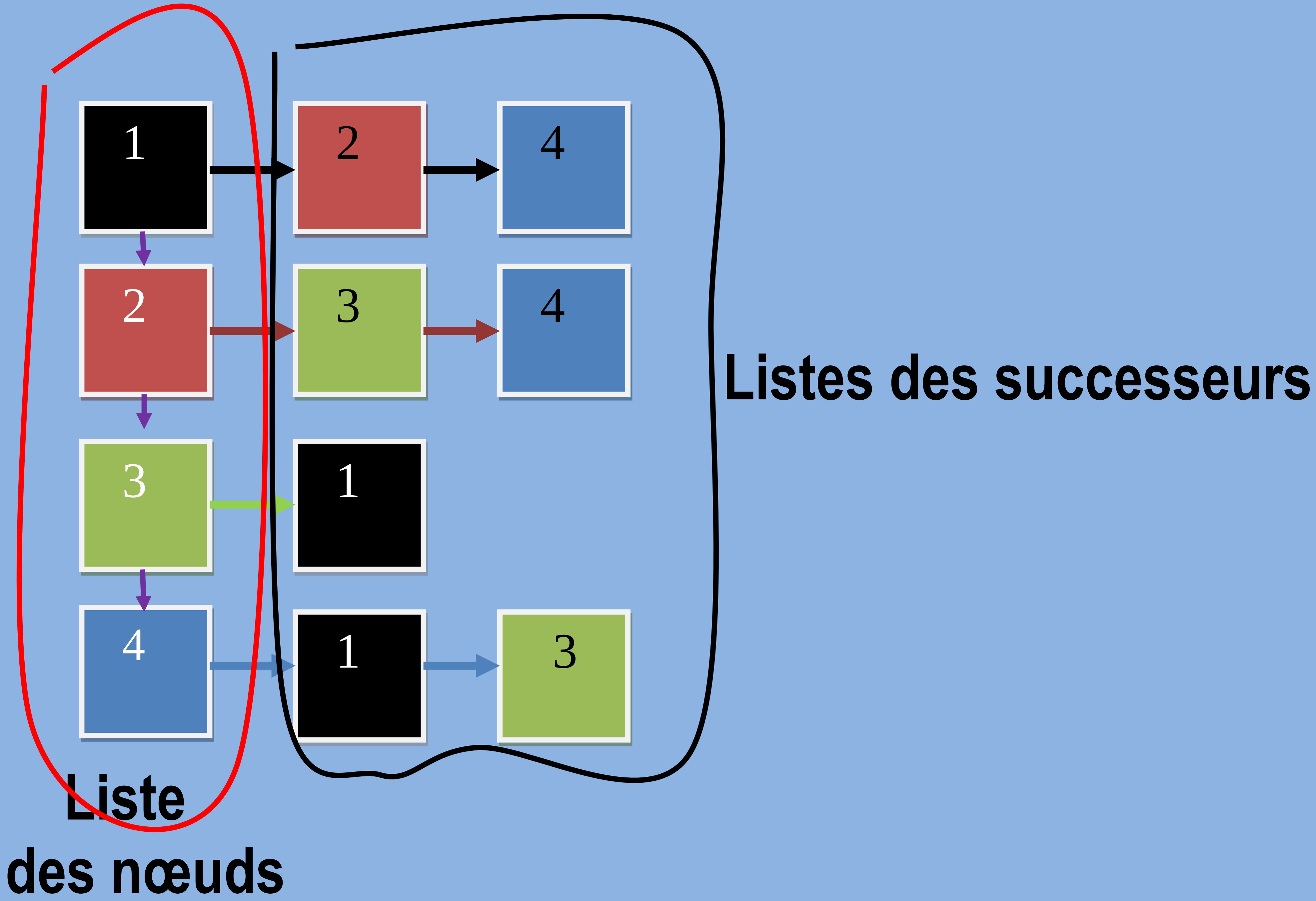
Elle consiste à :

- représenter l’ensemble des nœuds,
- associer à chaque nœud la liste de ses **successeurs** rangés dans un **ordre arbitraire**.

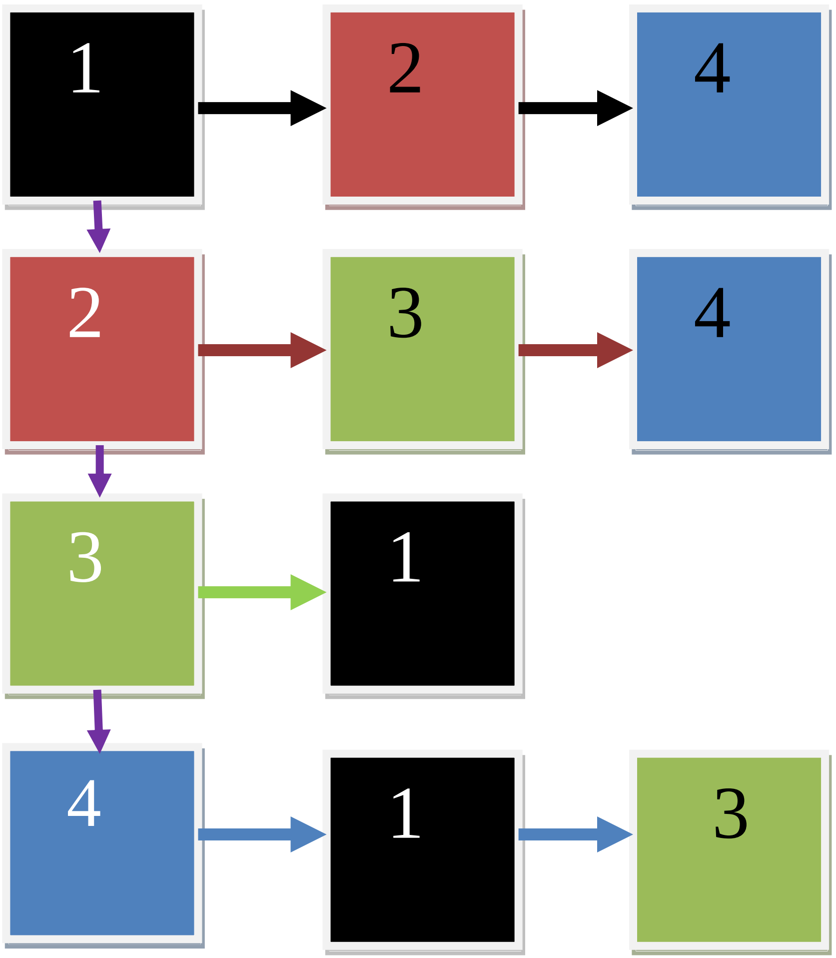
Le graphe suivant



est représenté par les listes suivantes :


1
2
4
2
3
4
3
1
4
1
3
Listes des successeurs
Liste
des nœuds

Représentation d'un graphe par **liste d'adjacence**



Les listes générées par cette représentation sont appelées **listes d’adjacence**.

Si le nombre de sommet n’évolue pas, les **listes des successeurs** sont accessibles :

- à partir d’un **tableau**,
- tableau qui contient, pour chaque nœud, un pointeur vers la **tête** de liste de ses successeurs.

2.1-Avantages

1- L'espace mémoire utilisé pour un graphe **orienté** avec **n** sommets et **p** arcs est en O(**n+p**).

2- Dans le cas d'un graphe **non orienté** avec **n** nœuds et **p** arêtes, l'espace mémoire utilisé est en O(**n+2p**).

3-Pour un traitement sur les **successeurs** d’un nœud **s**:

nombre de nœuds visités = $\mathbf{d°^{+}(s)}$

3- Un algorithme qui traite **tous** les arcs d’un graphe de **p** arcs peut donc être d’ordre **p**.

2.2- Inconvenients

1-Pour **tester** s'il existe un arc **x** $\rightarrow$**y** (arête **x-y**), la représentation exige :

- un temps d'ordre **n**
- dans le pire des cas.

Le pire des cas :

- la liste d'adjacence est de longueur **n**-1,
- **y** est en fin de liste

2- Il en va de même pour **ajouter** un arc ou une arête (avec test de non répétition).

3- Elle ne permet pas de calculer facilement les opérations relatives aux **prédécesseurs**:

d°⁻(s) , **ième_pred(**i,s,g**)**