Graphe -
Modélisation informatique

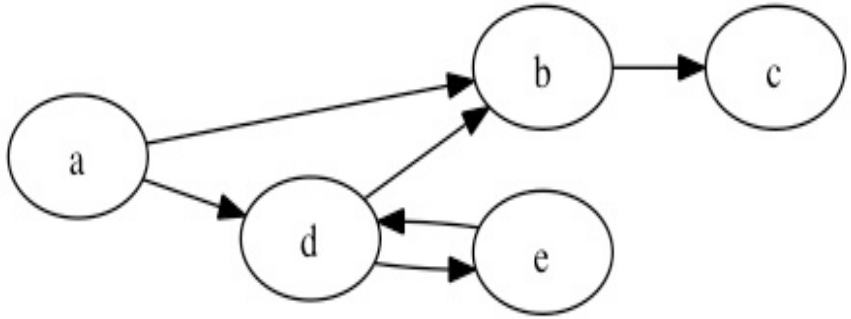
Institut National des Sciences Appliquées – Rouen
Département Architecture des Systèmes d'Information
michel.mainguenaud@insa-rouen.fr

Propriétés

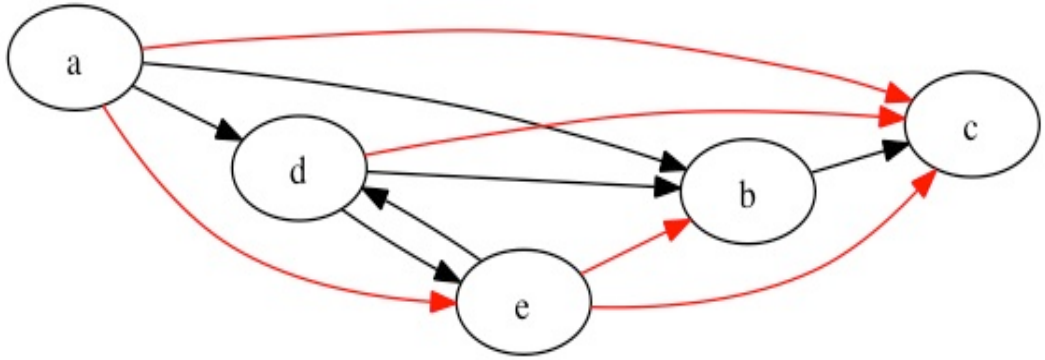
- Multiplicité d'une paire x,y :
  - $m^+(x, y)$: Nombre d'arcs de G ayant x comme extrémité initiale et y comme extrémité terminale.
  - $m^-(x, y) = m^+(y, x)$
  - $m(x, y) = m^+(x, y) + m^-(x, y)$
  - Si x = y alors m (x, y) est égal à 2 fois le nombre de boucles attachées au noeud x

Fermeture Transitive

- Construire un nouveau graphe G* à partir de G
- Arc (i, j) ⇔ Il existe un chemin de i à j dans G



G



G*

Représentations informatique

- Matrices
  - Matrice adjacence : 1-graphe (booléenne ou valuée) vs. p-graphe
    - Carrée : ligne : noeud, colonne : noeud
    - $A_{ij} = m^{+}(x_i, x_j)$ ou Vrai ou valuation
  - Matrice d'incidence :
    - noeuds-arcs (-1 -extrémité-, 0, 1 -origine-)
    - ligne : noeud, colonne : arc
- Listes : (matrices creuses)
  - Listes des noeuds successeurs / prédécesseurs
  - Listes des arcs (cocycles)

Représentations informatique (2)

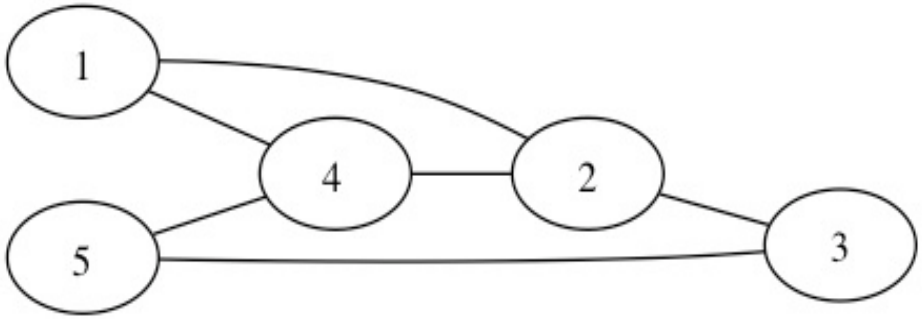
- Matrice : Que des problèmes
  - Espace mémoire important
  - Temps passé à faire des tests (complexité des algorithmes)
  - Pas de flexibilité
  - Pas de multi-graphe ou augmentation de la dimension
- Listes
  - Type abstrait avec primitives de manipulations
    - Flexibilité (insertion / suppression)
    - Emplacement mémoire raisonnable

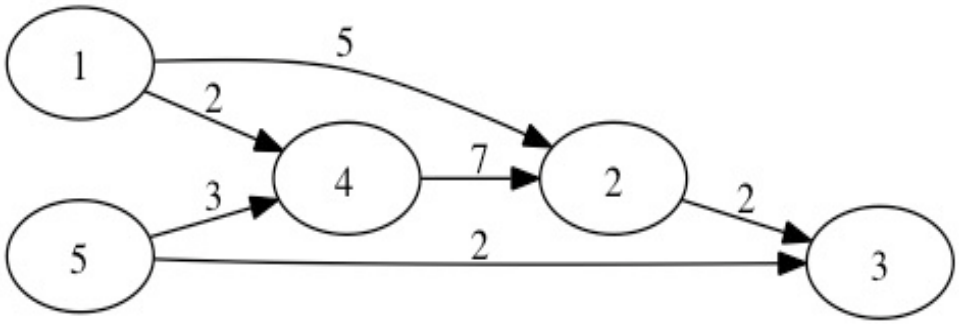
Représentations informatique (3)

- Matrice d'adjacence (non) valuée / (non) orientée
- Matrice d'incidence noeuds / arcs
- Liste des successeurs / prédécesseurs / voisins / noeuds / arcs / arêtes
- Liste des cocycles orientés / non orientés

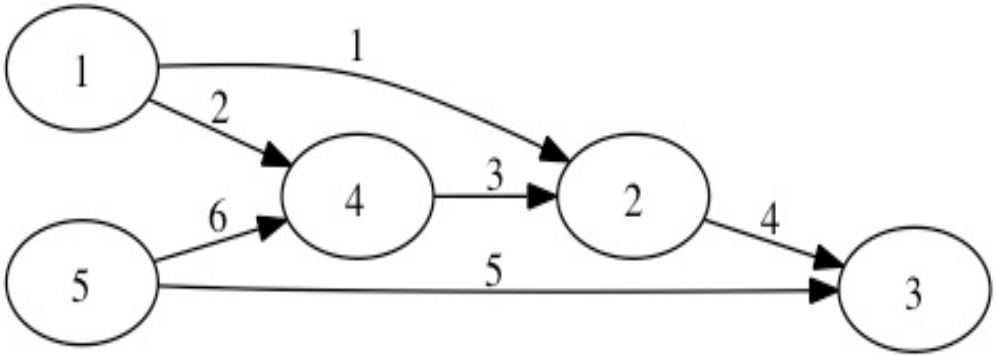
=> Problème passage d'une représentation à une autre

Exemple : Graphes





Fonction de coût



Numérotation des arcs

Booléenne

- 1-graphe, Non orienté ⇔2 arcs ⇒ Matrice symétrique

	1	2	3	4	5
1	0	1	0	1	0
2	1	0	1	1	0
3	0	1	0	0	1
4	1	1	0	0	1
5	0	0	1	1	0

Valuation ⇒ Fonction de coût

- Arc inexistant : 0 ou infini => problème de représentation

	1	2	3	4	5
1	0	5	0	2	0
2	0	0	2	0	0
3	0	0	0	0	0
4	0	7	0	0	0
5	0	0	2	3	0

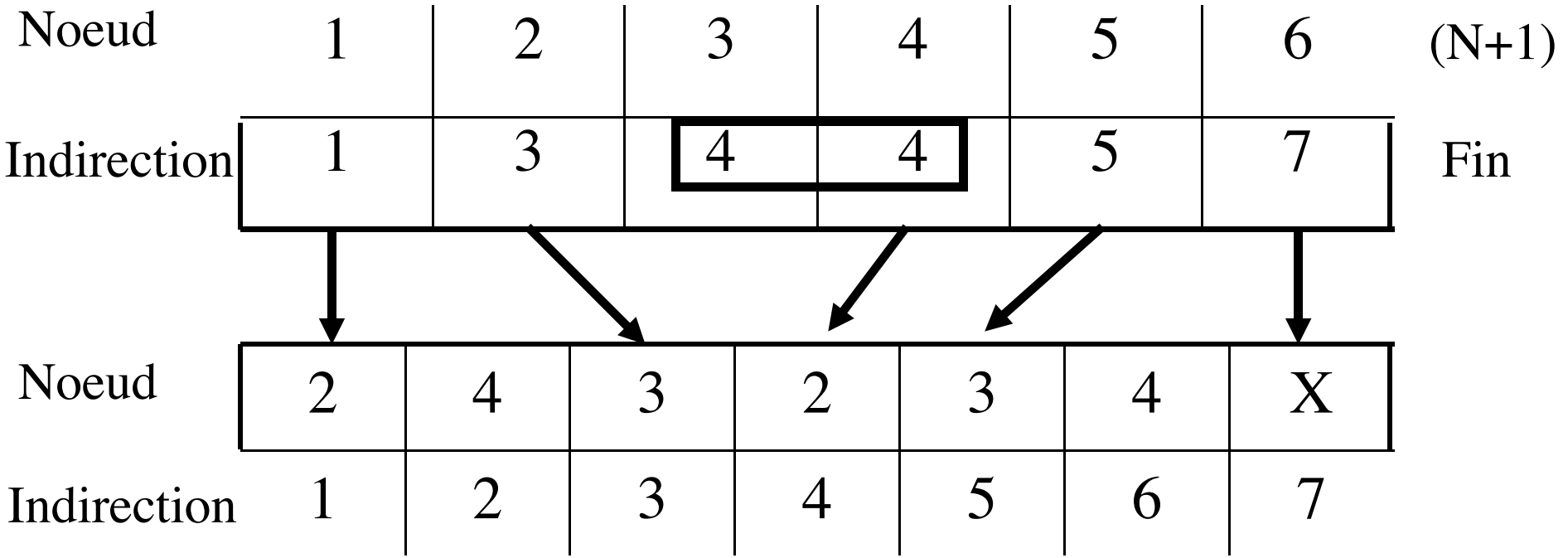
Noeuds-arcs

- p-graphe, pas de boucle
- Colonne : arc, ligne : noeud, origine : + destination : -

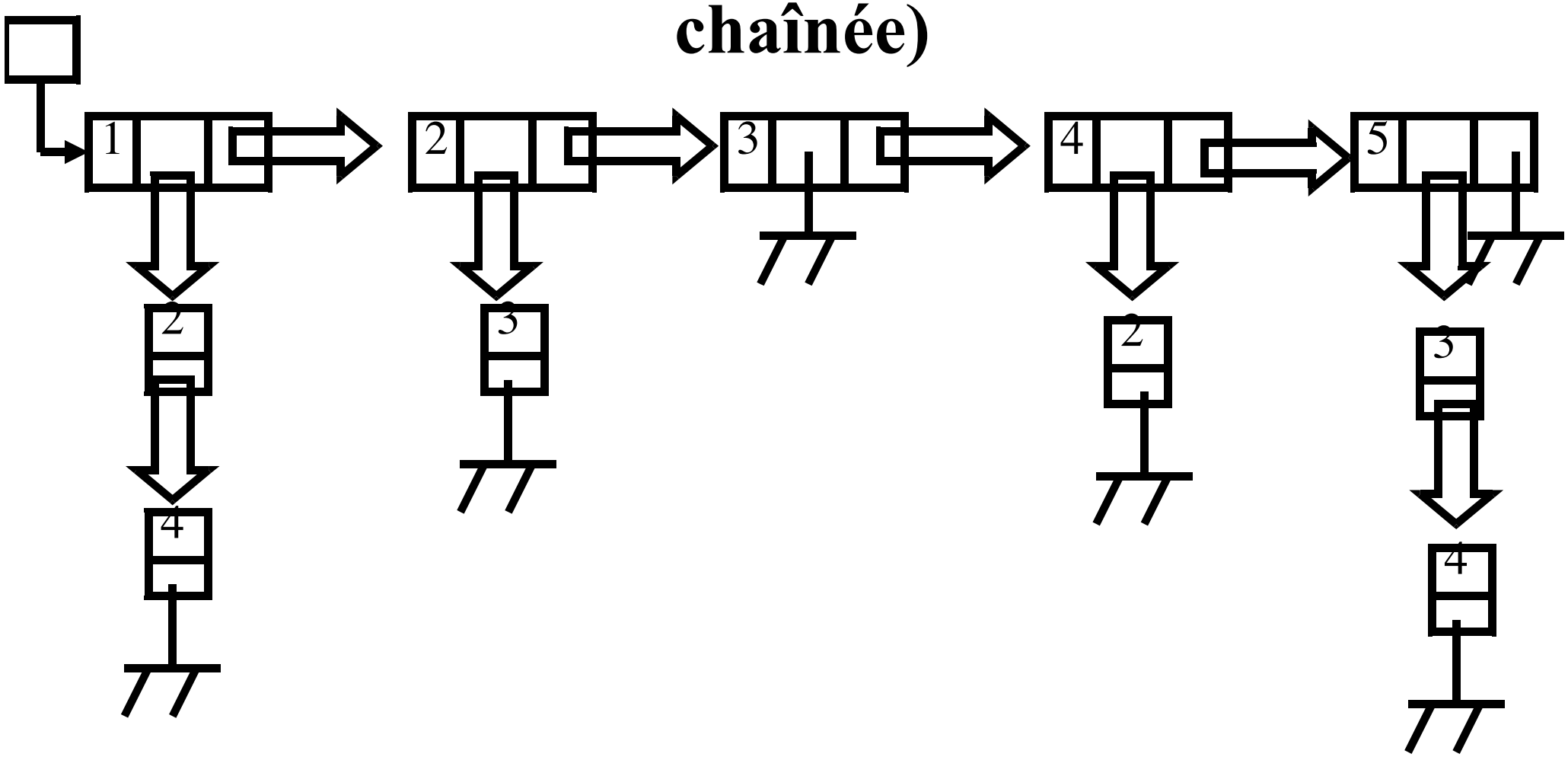
	1	2	3	4	5	6
1	+1	+1	0	0	0	0
2	-1	0	-1	+1	0	0
3	0	0	0	-1	-1	0
4	0	-1	+1	0	0	-1
5	0	0	0	0	+1	+1

Successeurs

- Tableau : Nœud : tableau [1.. N+1] d'indirections sur les successeurs ([1.. M+1])
- Principe : idem pour les voisins (non orienté) mais symétrique => 2M + 1 ou pour les prédécesseurs

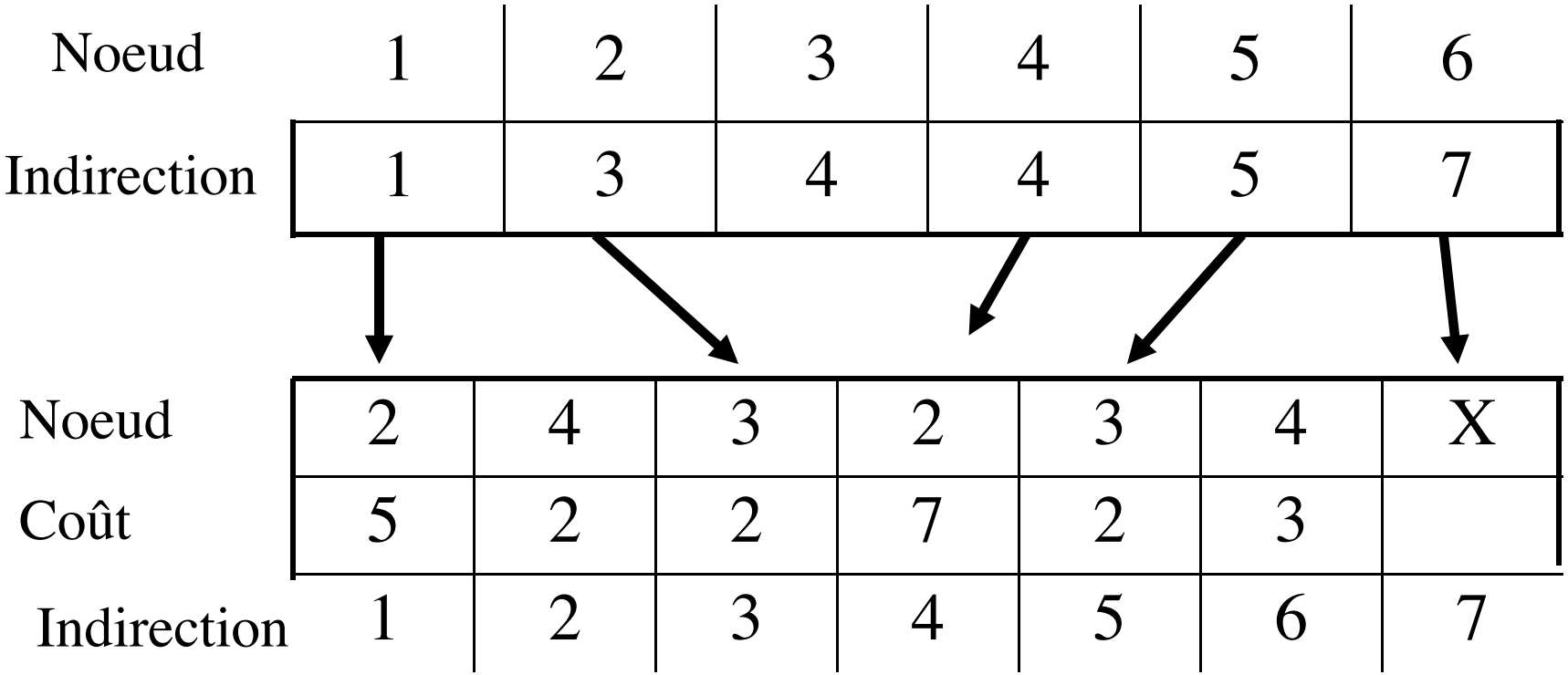


Successeurs (version liste chaînée)



Orienté Valué

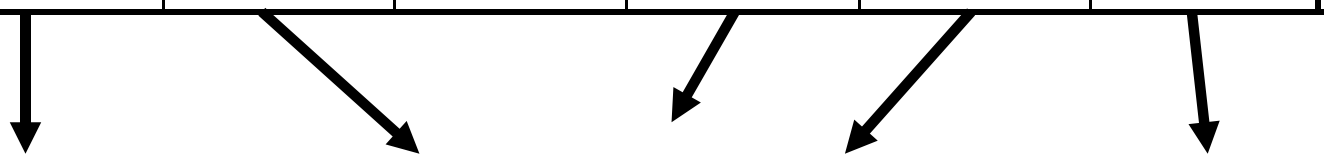
- Idem successeurs + valuation (liste chaînée : multi-graphe)



Cocycle

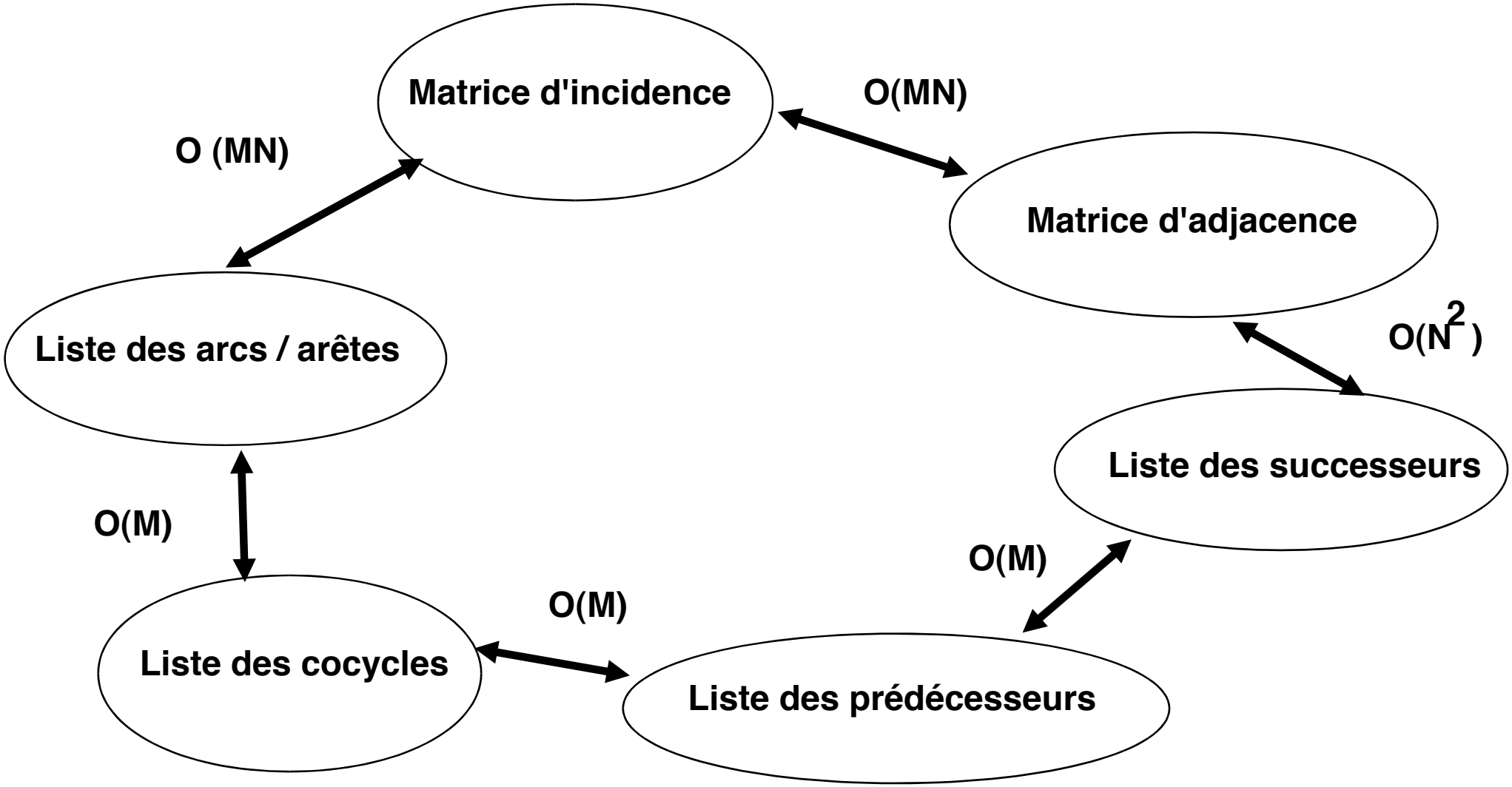
- Idem liste des successeurs mais avec les arcs => p-graphes avec boucles

	1	2	3	4	5	6
Noeud	1	3	4	4	5	7



	1	2	3	4	5	6	7
Arc	1	2	4	3	5	6	X
Extrémité	2	4	3	2	3	4	X
Indirection	1	2	3	4	5	6	7

Passages


Matrice d'incidence
O(MN)
O (MN)
Matrice d'adjacence
Liste des arcs / arêtes
O(N2)
Liste des successeurs
O(M)
O(M)
O(M)
Liste des cocycles
Liste des prédécesseurs

Problématique Base de données

- Fonction d'étiquettage (noeud, arc)
  - Le graphe ne tient pas en MC
  - Opérateurs de manipulation - chemin –
- Modélisation du graphe (1 ou plusieurs niveaux d'abstraction)
  - Passage des informations aux différents niveaux
  - Gestion des informations alphanumériques

Problèmes conventionnels

- Chemin eulérien : chemin qui emprunte une fois et une seule chaque arc
- Chemin hamiltonien : chemin qui emprunte une fois et une seule chaque noeud
- Stable : Sous-ensemble, S, de noeuds d'un graphe / deux noeuds de S ne sont pas adjacents (Γ)
- Clique : Sous graphe complet
- Iso/(homéo)morphisme de sous-graphes
- ...

Problèmes conventionnels (2)

- Domaine de la recherche opérationnelle (parcours / contraintes)
- Opérateurs de manipulation : Attention / pb NP-Complet
- Bases de données déductives / documentaires
- Bases de données spatiales (transports)

Calculs

- Pour étudier la complexité d'un problème d'optimisation -> Définir le modèle de résolution
  - Machine de Turing
  - Random Access Memory
  - Lambda-calcul
- Exemple : Voyageur de commerce
  - Soit y un entier, existe-t-il un circuit hamiltonien dont la longueur est inférieure ou égale à y ?

Problème de décision

- Divisé en deux classes :
  - Problèmes indécidables
    - Impossible d'écrire un algorithme pour le résoudre (il n'y a pas assez d'hypothèses)
    - Exemple : age du capitaine connaissant la longueur du bateau
  - Problèmes décidables
    - Hypothèses, Données, Question
    - Réponse : oui / non (il existe un algorithme)
    - Amélioration :
      - Problèmes calculables : donne une solution
      - Hiérarchie dans la difficulté à donner une solution

Problème indécidable

- Arrêt d'un programme

Supposons que arret (P) existe et renvoie Vrai si P s'arrête et Faux si P ne s'arrête pas

Procedure marrant (P)

    tant que arret (P)

    ftq

fproc

Si P se termine, marrant boucle

Si P ne se termine pas marrant se termine

Appel : marrant (marrant)

si marrant se termine c'est que marrant ne se termine pas

si marrant ne se termine pas c'est que marrant se termine

P / NP (1)

- Classe P (Problèmes « faciles »):
  - Il existe un algorithme polynomial qui permet de trouver la réponse à toute instance du problème
  - Différence entre la théorie et la pratique : une complexité > n^3 se révèle peu réaliste si n est grand
  - Les constantes ne comptent pas 3n $\Leftrightarrow$ n

P / NP (2)

- Classe Non-Déterministes Polynomiaux (« difficile »)
  - Il existe un algorithme polynomial qui permet de reconnaître une instance positive
  - Il est peu probable qu'un algorithme polynomial puisse être construit pour donner une solution
  - On dispose d'un algorithme polynomial non déterministe (dans un automate, à partir d'un état et d'une lettre, deux états différents sont accessibles)
  - Exemple : voyageur de commerce : on donne une séquence d'arcs, il répond oui/non à la propriété « circuit hamiltonien de longueur inférieure ou égale à y donné »

Réduction

- Un problème P1 est réductible à un problème P2
  - Il existe une fonction polynomiale qui transforme chaque énoncé de P1 en un autre énoncé P2
  - La réponse pour P1 est oui si et seulement si la réponse pour P2 est oui
  - Nécessite de montrer que P1 est NP et qu'il existe P2 connu être NP-complet et P1 est réductible à P2
  - Un problème est NP-difficile si le problème de décision correspondant est NP-complet
  - Exemple : voyageur de commerce

NP-complet

- Problème NP-complet au sens faible
  - Il est NP-complet
  - Il existe un algorithme pseudo-polynomial (peut s'exprimer comme un polynôme des paramètres du problème) pour le résoudre
- Problème NP-complet au sens fort
  - Il appartient à la classe NP
  - Il reste NP-complet même si on limite la taille des paramètres du problème

Heuristique

- Trouver une solution approchée du problèmes sans savoir s'il s'agit de la meilleure solution
- Permet de rendre « raisonnable » la résolution d'un problème
- La solution peut être utilisée comme initialisation d'une méthode exacte (plus couteuse)
- Spécifique à un problème
  - Méta-heuristique pour des méthodes plus générales (recuit-simulé, algorithmes génétiques, ...)

Quelques problèmes NP-Complets

- Voyageur de commerce (circuit hamiltonien)
- Clique maximum (sous-graphe complet)
- Isomorphisme de sous-graphes
- Stable maximal
- Support de taille < K
- Sac à dos
- ...

Complexité

- Fermeture transitive
  - Simple : Dijkstra : $O(V^2)$
  - Méthode matricielle
    - Base : $N^3 \log_2 N$
    - Floyd-Warshall : N^3
  - Méthode de séquence
    - Ford : $2 N^3$
    - Yen : $N^3 / 6$
  - …. En gros du N^3

Complexité (2)

- Gestion des flux
  - Ford-Fulkerson : N^4 (Log N + ½ Log C)
    C : capacité moyenne des arcs du graphe
  - Edmonds-Karp : N^4
  - Dinic-Kazanov : …. N^2 (principe différent de la chaîne améliorante)