Graphes

Objectifs

- **Etudier une nouvelle structure de données non linéaire, plus générale, où chaque élément peut posséder plusieurs prédécesseurs et plusieurs successeurs :**
  - **Terminologie**
  - **Type Abstrait de Données Graphe**
  - **Représentation et implémentation**
  - **Parcours d'un graphe**

Notion de Graphes

- **Les graphes sont l'une des structures de données les plus utilisées en informatique :**
  - Les algorithmes permettant de les manipuler constituent les fondements de l'informatique
  - Il existe des centaines de problèmes informatiques qui sont définis en termes de graphes
- **Les graphes servent généralement à modéliser des problèmes en termes de relations ou de connexions entre des objets**
- **Les objets sont représentés par des sommets**
- **Les relations ou connexions sont représentées par des arcs reliant les sommets**
- **Les graphes peuvent être orientés (les arcs vont d'un sommet à l'autre dans un sens précis) ou non orientés (les arcs n'ont pas de sens)**

3

Exemples

- **Dans une carte de liaisons aériennes, les villes sont des sommets du graphe et l'existence d'une liaison aérienne entre deux villes est la relation du graphe**
- **Dans le graphe du flot de contrôle d'un programme, les boîtes (instructions ou tests) sont les sommets, et les flèches indiquent les enchaînements possibles entre celles-ci**
- **Dans une entreprise où certaines tâches doivent être exécutées avant d'autres, on peut schématiser l'ordonnancement des tâches par un graphe où les sommets sont les tâches et où il existe un arc entre deux tâches t_i et t_j seulement si t_i doit être terminée juste avant d'exécuter t_j**

4

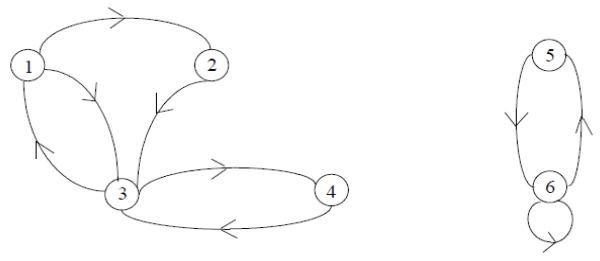
Graphe Orienté (Définitions)

- **Un graphe orienté *G* est un couple (*S*,*A*), où :**
  - ***S* est un ensemble fini d'éléments appelés sommets** (***vertex*** en anglais, au pluriel ***vertices***)
  - ***A* est un ensemble fini de paires (ordonnées) de sommets, appelées arcs (*arc* en anglais)**
- **On écrit *G* = (*S*,*A*) pour représenter le graphe**
- **Un graphe orienté est dit complet si quels que soient deux sommets distincts, il existe un arc les reliant dans un sens ou dans l'autre**

5

Graphe Orienté (Exemple 1)

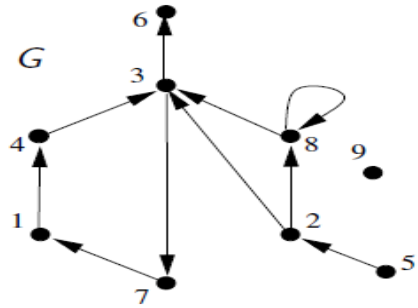
- Soient *S* ={1,2,3,4,5,6} et *A*={(1;2),(1;3),(2;3),(3;1),(3;4),(4;3),(5;6),(6;5),(6;6)}
- (*S*, *A*) est un graphe orienté qui peut être représenté par :



6

Graphe Orienté (Exemple 2)

- Soit le graphe orienté G=(S,A) où
  - S = {1,2,3,4,5,6,7,8,9} et
  - A={(1,4);(2,3);(2,8);(3,6);(3,7);(4,3);(5,2);(7,1);(8,3);(8,8)}



7

Graphe Orienté (Terminologie) (1)

- Soit $G = (S, A)$ un graphe orienté. Si $X = (a,b) \in A$, on dit que :
  - a est **adjacent** à b
  - a est **un prédécesseur** de b.
  - b est **un successeur** de a.
  - a est **l'origine** de l'arc X.
  - b est **l'extrémité** de l'arc X.
  - X est **incident** au sommet a et au sommet b.
  - De plus, si $a = b$, on dit que X est **une boucle**.

8

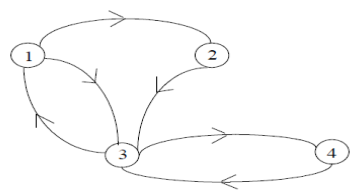
Graphe Orienté (Terminologie) (2)

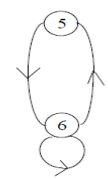
- On appelle **chemin** d'un graphe orienté *G* une suite (finie) d'arcs de *G* telle que l'extrémité d'un arc est toujours confondue avec l'origine du suivant.
- L'origine du premier arc de la suite est appelé **origine du chemin**.
- L'extrémité du dernier arc de la suite est appelé **extrémité du chemin**
- La **longueur d'un chemin** est le nombre d'arcs qui le composent
- Un chemin est dit **simple** si tous les arcs qui le composent sont différents.
- Un chemin est dit **élémentaire** si tous les sommets qui le composent sont différents.
- On appelle **circuit** tout chemin dont l'origine et l'extrémité sont confondues.

9

Exemple

- En reprenant **l'exemple 1**, on a :
  - {(1,3);(3,1);(1,2)} est un chemin simple non élémentaire d'origine 1 et d'extrémité 2.
  - {(1,2);(2,3);(3,4)} est un chemin simple et élémentaire. Ce chemin est de longueur 3
  - {(2,3);(3,4);(4,3);(3,1);(1,2)} est un circuit simple et non élémentaire
  - {(6,6)}





10

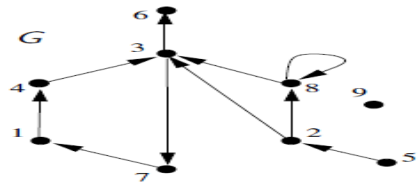
Graphe Orienté (Terminologie) (3)

- Soient u et v deux sommets d'un graphe orienté. On dit que :
  - v est **un descendant** de u s'il existe un chemin allant u à v
  - v est **un ascendant** de u s'il existe un chemin allant v à u.
  - Un sommet v tel qu'il n'existe aucun chemin de u à v dans G est dit **inaccessible** (ou **non atteignable**) à partir de u.
  - Un sommet est dit **isolé** s'il n'est accessible par aucun autre sommet du graphe
- Il est à noter que les sommets d'un circuit sont tous ascendants et descendants les uns des autres

11

Exemple

- En reprenant **l'exemple 2** :
  - On considère le chemin {(5,2);(2,8);(8,3);(3,6)}.
  - Le sommet 6 est descendant du sommet 5 mais l'inverse n'est pas vrai.
  - Le sommet 5 est ascendant du sommet 6
  - Le sommet 2 est inaccessible depuis le sommet 3.
  - Le sommet 9 est **isolé** du reste du graphe



12

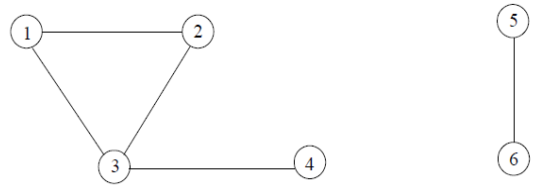
Graphe Non Orienté (Définitions)

- **Un graphe (simple) non orienté *G* est un couple (*S*,*A*), où :**
  - *S* est un ensemble fini de **sommets**.
  - *A* est un ensemble fini de paires (non ordonnées) de sommets de *S*, appelées **arêtes** (**edge** en anglais)
- **On écrit *G* = (*S*,*A*) pour représenter le graphe**
- **Un graphe non orienté est dit complet si quels que soient deux sommets distincts, il existe une arête les reliant**

13

Graphe Non Orienté (Exemple 1)

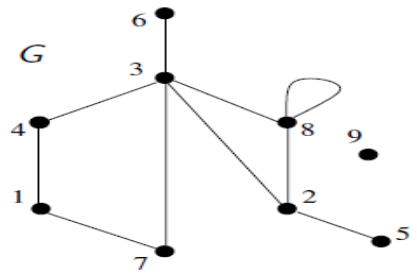
- Soient *S* ={1,2,3,4,5,6} et *A*={(1;2),(1;3),(2;3),(3;4),(5;6)}
- (*S*, *A*) est un graphe non orienté qui peut être représenté par :



14

Graphe Non Orienté (Exemple 2)

- Soit le graphe non orienté G=(S,A) où
  - S = {1,2,3,4,5,6,7,8,9} et
  - A={(1,4);(1,7);(2,3);(2,5);(2,8);(3,4);(3,6); (3,7);(3,8);(8,8)}



15

Graphe Non Orienté (Définitions)

- Soit $G = (S, A)$ un graphe non orienté.
  - Si $X = \{a,b\} \in A$, on dit que a et b sont **voisins**.
  - On appelle **chaîne** de G une suite (finie) d'arêtes de G telle que 2 arêtes consécutives dans la suite ont un sommet commun.
  - Un **cycle** est une chaîne dont l'origine et l'extrémité sont confondues.
  - Une chaîne est dite **élémentaire** si elle ne contient pas plusieurs fois le même sommet
  - La **longueur** d'une chaîne est le nombre d'arêtes qui la composent.

16

Graphe Connexe/Fortement Connexe

- Un graphe non orienté *G*= (*S*, *A*) est dit **connexe** si et seulement si, pour toute paire de sommets distincts {*x,y*} de *S*, il existe une chaîne entre les sommets *x* et *y*.
- Un graphe orienté *G*= (*S*, *A*) est dit **fortement connexe** si et seulement si, pour toute paire de sommets distincts {*x,y*} de *S*, il existe un chemin de *x* à *y* et un chemin de *y à x.*

17

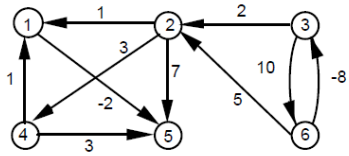
Notion de Graphe Valué

- Dans de nombreuses applications, il est naturel d'associer **une valeur** (on dit aussi **un coût** ou **un poids**) aux arcs ou aux arêtes du graphe.
- **Un graphe valué** (ou **pondéré**), orienté (resp. non orienté) est un triplet (*S,A,C*) où *S* est un ensemble fini de sommets, *A* un ensemble fini d'arcs (resp. d'arêtes) et *C* une fonction de A à valeurs réelles appelée **fonction coût**
- Ainsi, on pourra traiter des problèmes tels que la recherche du plus court chemin entre deux sommets d'un graphe

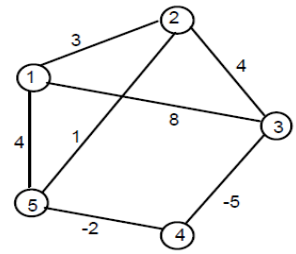
18

Exemples de Graphes Valués

- **Exemple de graphe orienté valué :**



- **Exemple de graphe non orienté valué :**



19

Distance et Diamètre

- **La distance entre deux sommets** d'un graphe est la plus petite longueur des chaînes, ou des chemins, reliant ces deux sommets.
- **Le diamètre d'un graphe** est la plus longue des distances entre deux sommets.

20

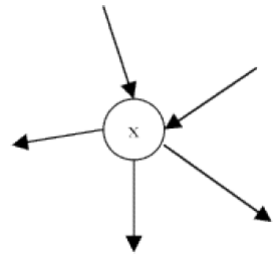
Notion de Degré

- Dans un graphe *orienté*, si X=(u, v) est un arc, on dit que **X est incident à u vers l'extérieur**. Le nombre d'arcs ayant leur extrémité initiale en u, se note $\mathbf{d^{o+}(u)}$ et s'appelle le ***demi-degré extérieur*** **de u**. c'est le nombre de successeurs de u.
- On définit de même les notions d'**arc incident vers l'intérieur** et de **demi-degré intérieur** qui est noté $\mathbf{d^{o-}(u)}$. le nombre de prédécesseurs de u.
- Dans un graphe orienté (resp. non orienté), on appelle **degré d'un sommet**, et on note $\mathbf{d^{o}(u)}$, le nombre d'arcs (resp. d'arêtes) dont u est une extrémité.
- Dans le cas d'un graphe orienté, on a $\mathbf{d^{o}(u) = d^{o+}(u) + d^{o-}(u)}$, pour tout sommet u. C'est le nombre de sommets adjacents à u.
- Un sommet de degré **1** (resp. **0**) est dit **sommet pendant** (resp. **isolé**)
- Un graphe est dit **régulier** si les degrés de tous ses sommets sont égaux

21

Exemple

- Dans le graphe suivant :
  - $\mathbf{d^{\circ+}(x) = 3, d^{\circ-}(x) = 2}$ **et** $\mathbf{d(x) = 5}$



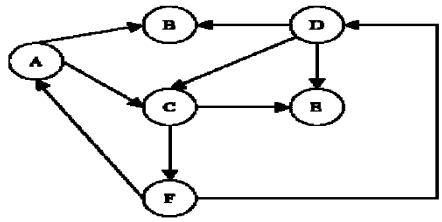
22

Sous-Graphe et Graphe partiel

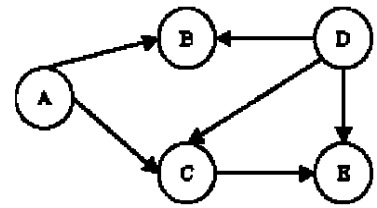
- Soit *G=(S,A)* un graphe. **Le sous-graphe de *G* engendré par *S'*** (un sous-ensemble de *S*) est le graphe *G'* dont les sommets sont les éléments de *S'* et dont les arcs (resp. les arêtes) sont les arcs (resp. les arêtes) de *G* ayant leurs deux extrémités dans *S'*. Autrement dit, on ignore les sommets de *S**S'* ainsi que les arcs ayant au moins une extrémité dans *S**S'*.
- Soit *G=(S,A)* un graphe. **Le graphe partiel de *G* engendré par *A'*** (un sous-ensemble de *A*) est le graphe *G' =(S,A')* dont les sommets sont les éléments de *S* et dont les arcs (resp. les arêtes) sont ceux de *A'*. Autrement dit, on élimine de G les arcs (resp. arêtes) de *A**A'*.

23

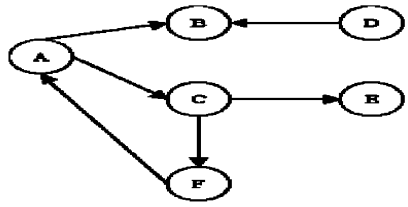
Exemples



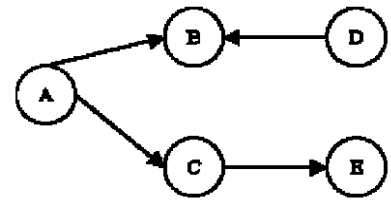
Un graphe G



Un sous-graphe de G



Un graphe partiel de G



Un sous-graphe partiel de G

24

Composantes Connexes d'un Graphe Non Orienté

- On définit la relation :

  v est accessible à partir de u si et seulement si il existe un chemin de longueur k ≥ 0 d'origine u et d'extrémité v.

- C'est une relation d'équivalence :
  - elle est *réflexive* car k = 0 est admis; elle est *symétrique* car le graphe est non orienté; elle est *transitive*, car on "concatène" les chemins.

- Par définition, **les *composantes connexes*** d'un graphe *non orienté* G sont les classes d'équivalence pour la relation: **« être accessible à partir de »**.
  - **D'une autre manière, on appelle composante connexe un sous-graphe connexe maximal.**

25

Composantes Fortement Connexes d'un Graphe Orienté

- Pour un graphe orienté, la relation "*être accessible à partir de*" est toujours réflexive et transitive, mais elle n'est plus symétrique. On considère alors sa symétrisée :

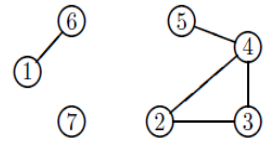
  v et u sont mutuellement accessibles si et seulement si il existe un chemin (de longueur k ≥ 0) d'origine u et d'extrémité v et un chemin (de longueur l ≥ 0) d'origine v et d'extrémité u.

- Par définition, **les *composantes fortement connexes*** d'un graphe *orienté* sont les classes d'équivalence de G pour la relation : **« être mutuellement accessibles »**.
  - **D'une autre manière, on appelle composante fortement connexe un sous-graphe fortement connexe maximal.**

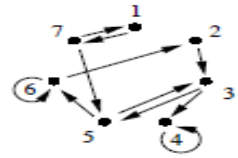
26

Exemples

- Le graphe suivant a trois composantes connexes : {1,6}, {7} et {2,3,4,5}



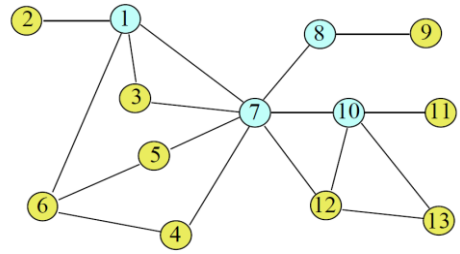
- Le graphe suivant a trois composantes fortement connexes : {1,7}, {2,3,5,6} et {4}



27

Point d'Articulation d'un Graphe

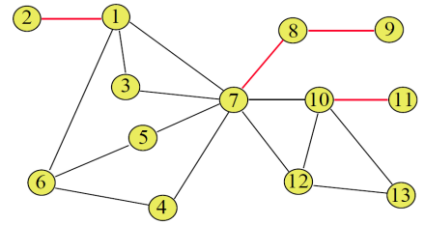
- C'est un sommet d'un graphe, qui, si on le supprime, déconnecte le graphe
- Dans le graphe suivant, les sommets 1, 7, 8 et 10 sont des points d'articulation



28

Pont d'un Graphe

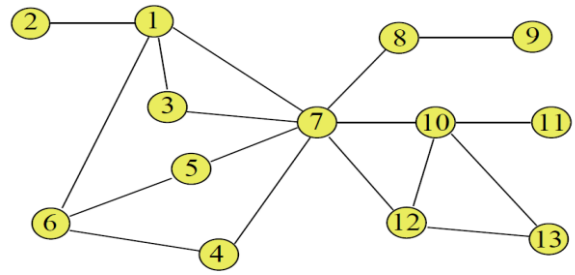
- C'est une arête d'un graphe, qui, si on la supprime, déconnecte le graphe
- Dans le graphe suivant, les arcs (1,2), (7,8), (8,9) et (10,11) sont des ponts



29

Graphe Bi-Connexe

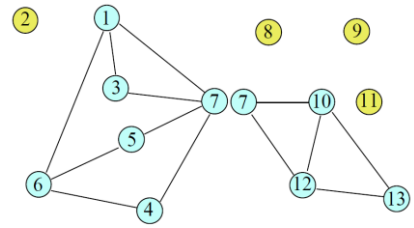
- Un graphe connexe sans point d'articulation est dit **bi-connexe**
- Le graphe suivant n'est pas bi-connexe



30

Composantes Bi-Connexes

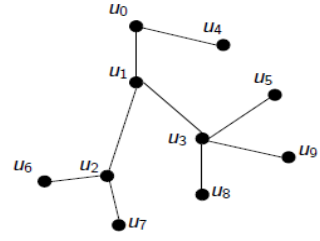
- Un graphe peut ne pas être bi-connexe mais contenir des **composantes bi-connexes**
- Dans une composante bi-connexe, il existe un circuit entre deux sommets quelconques



31

Notion d'Arbre

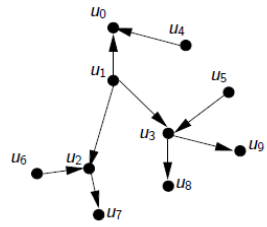
- Un graphe non orienté où tous les sommets sont accessibles les uns des autres est dit connexe.
- On appelle **arbre** un graphe non orienté connexe et sans cycle.



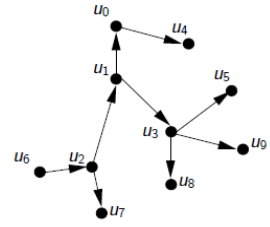
32

Notion d'Arborescence

- Lorsqu'on oriente les arêtes d'un arbre, le graphe obtenu est appelé **une arborescence**.
- Dans une arborescence, on appelle racine un sommet pour lequel tous les autres sommets sont accessibles (il n'existe pas toujours de racine).



Arborescence sans racine

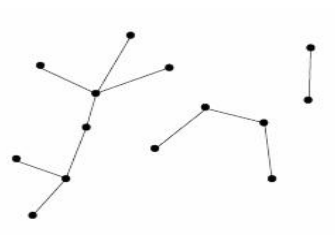


u_6 est racine.

33

Notion de Forêt

- **Une *forêt*** est un graphe non orienté (resp. orienté) dont chaque composante connexe (resp. fortement connexe) est un **arbre** (resp. **une arborescence**).
- **Exemple de forêt (graphe acyclique) :**



34

Type Abstrait Graphe

- Parfois, le graphe est donné une fois pour toutes. Les opérations intéressantes sont :
  - **test d'existence d'un arc** (**d'une arête** entre deux sommets)
  - **test d'existence d'un sommet** parmi les successeurs d'un autre sommet
  - **énumération des successeurs d'un sommet**. Pour ce faire, il faut connaître **le demi-degré extérieur** de tout sommet et **le ième successeur** d'un sommet
  - **…**
- Le plus souvent, le graphe est évolutif ; on veut donc lui appliquer les opérations :
  - **ajout et suppression d'un sommet**
  - **ajout et suppression d'un arc**
  - …
- **Deux types abstraits :**
  - un pour les graphes orientés
  - un autre pour les graphes non orientés
- Ces deux types abstraits utilisent **le type Sommet** :
  - pour distinguer les sommets d'un graphe, **on les étiquette**, soit par des chaînes de caractères, soit par des numéros (ce qui va être utilisé dans la suite)

35

Type Abstrait Sommet

Type Sommet {on étiquette un sommet par un numéro}

Utilise Entier

Opérations

créer	**:**	**Entier → Sommet**
modifier	**:**	**Sommet x Entier → Sommet**
numéro	**:**	**Sommet → Entier**

Axiomes

numéro(som(i)) = i, pour tout entier i

36

Spécification des graphes orientés (Conventions)

- Quand on ajoute un sommet, celui-ci est isolé (il n'a aucun arc incident) ;
- Quand on ajoute un arc, si les sommets adjacents à cet arc n'appartiennent pas au graphe, on les ajoute ;
- Quand on retire un arc, les sommets adjacents ne sont pas retirés ;
- Quand on retire un sommet, tous les arcs incidents sont supprimés.

37

Type Abstrait Graphe (Orienté) (1)

```
Type Graphe {cas orienté}
Utilise Sommet, Entier, Booléen
Opérations
  graphe_vide      : → Graphe
  ajouter_sommet   : Sommet x Graphe → Graphe
  ajouter_arc      : Sommet x Sommet x Graphe → Graphe
  est_sommet       : Sommet x Graphe → Booléen
  est_arc          : Sommet x Sommet x Graphe → Booléen
  d°+              : Sommet x Graphe → Entier
  ième_succ        : Entier x Sommet x Graphe → Sommet
  d°-              : Sommet x Graphe → Entier
  ième_pred        : Entier x Sommet x Graphe → Sommet
  supprimer_sommet: Sommet x Graphe → Graphe
  supprimer_arc    : Sommet x Sommet → Graphe
```

38

Type Abstrait Graphe (Orienté) (2)

Préconditions

ajouter_sommet(s,g) ***est-défini-ssi*** **est_sommet(s,g) = faux**

ajouter_arc(s,s',g) ***est-défini-ssi*** **s ≠ s' ET est_arc(s,s',g) = faux**

$d^{\circ+}$(s,g) ***est-défini-ssi*** **est_sommet(s,g) = vrai**

$d^{\circ-}$(s,g) ***est-défini-ssi*** **est_sommet(s,g) = vrai**

ième_suc(i,s,g) ***est-défini-ssi*** **est_sommet(s,g) = vrai**
ET (i ≤ $d^{\circ+}$(s,g)) = vrai

supprimer_sommet(s,g) ***est-défini-ssi*** **est_sommet(s,g) = vrai**

supprimer_arc(s,s'g) ***est-défini-ssi*** **est_arc(s,s',g) = vrai**

39

Type Abstrait Graphe (Orienté) (3)

Axiomes {pour est_sommet}

est_sommet(s,graphe_vide()) = faux

si s = s' alors est_sommet(s,ajouter_sommet(s',g)) = faux

si s ≠ s' alors est_sommet(s,ajouter_sommet(s',g)) = vrai

si s = s' OU s = s'' alors est_sommet(s,ajouter_arc(s',s'',g)) = faux

si s ≠ s' ET s ≠ s'' alors
est_sommet(s,ajouter_sommet(s'',g)) = est_sommet(s,g)

40

Type Abstrait Graphe (Orienté) (4) (Opérations Auxiliaires)

```
premsucc          : Sommet x Graphe → Sommet

succsuivant       : Sommet x Sommet x Graphe → Sommet

coût              : Sommet x Sommet x Graphe → Réel
ajouter_arc_valué : Sommet x Sommet x Réel x Graphe → Graphe

nb_sommets        : Graphe → Entier

nb_arcs           : Graphe → Entier
```

41

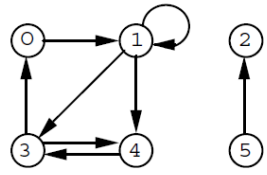
Représentations des Graphes

- **Deux implémentations classiques :**
  - **Par matrice d'adjacence**
  - **Par liste d'adjacence**
- **D'autres implémentations efficaces pour certains algorithmes :**
  - **Matrice d'incidence**
  - **Liste des arcs**
  - **…**

42

Représentation par Matrice d'Adjacence (1)

- **Correspond au cas où l'ensemble de sommets du graphe n'évolue pas**
- **On représente l'ensemble des arcs par un tableau de booléens**
- **Le graphe est représenté par une matrice carrée de booléens, dite matrice d'adjacence, de dimension n si le graphe a n sommets**

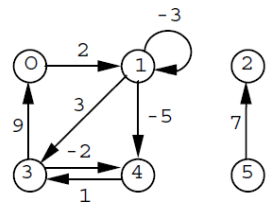


	0	1	2	3	4	5
0	0	1	0	0	0	0
1	0	1	0	1	1	0
2	0	0	0	0	0	0
3	1	0	0	0	1	0
4	0	0	0	1	0	0
5	0	0	1	0	0	0

43

Représentation par Matrice d'Adjacence (2)

- **Dans le cas où le graphe est non orienté, la matrice est *symétrique***
- **Dans le cas où le graphe est valué, on utilise une matrice où :**
  - **l'élément d'indices i et j a pour valeur le poids de l'arc/arête du sommet i au sommet j, si cet arc/arête existe,**
  - **et sinon une valeur dont on sait qu'elle ne peut être un poids: par exemple, le plus grand entier utilisable si les poids sont des entiers bornés supérieurement.**



	0	1	2	3	4	5
0	0	2	0	0	0	0
1	0	-3	0	3	-5	0
2	0	0	0	0	0	0
3	9	0	0	0	-2	0
4	0	0	0	1	0	0
5	0	0	7	0	0	0

44

Représentation par Matrice d'Adjacence (3)

- **Avantages :**
  - tester l'existence d'un arc (ou d'une arête) entre deux sommets: on accède directement à l'élément de la matrice (en un temps constant).
  - il est facile d'ajouter ou de retirer un arc ou une arête
  - il facile de parcourir tous les successeurs ou prédécesseurs d'un sommet.
- **Inconvénients :**
  - n tests quel que soit le nombre de successeurs de i. Il en est de même du calcul de d^{o+} ou de d^{o-}.
  - une consultation complète de la matrice requiert un temps d'ordre n^2
  - exige un espace mémoire de **$O(n^2)$** si le graphe a n sommets, quel que soit le nombre d'arcs ou d'arêtes du graphe.
- Pour remédier à cet inconvénient, on préfère souvent utiliser une représentation appelée "**par listes d'adjacence**".
- Cette représentation convient pour les petits graphes et lorsque l'accès aux successeurs, et surtout aux prédécesseurs, est important

45

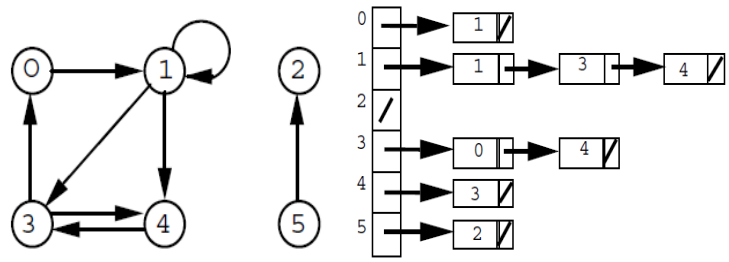
Implémentation en C d'un Graphe par Matrice d'Adjacence

```
#define N_MAX 20
typedef struct {
  int g[N_MAX][N_MAX];
  int n;
} GrapheM;
```

46

Représentation par Liste d'Adjacence (1)

- **C'est un tableau de listes chaînées :**
  - **La dimension du tableau est de *n (nombre de sommets)***
  - **Chaque sommet du tableau contient une liste chaînée de sommets qui lui sont adjacents (liste de ses successeurs)**



47

Représentation par Liste d'Adjacence (2)

- **Avantages :**
  - l'espace mémoire utilisé est, pour un graphe orienté avec n sommets et m arcs, en **O(n+m)**.
  - dans le cas d'un graphe non orienté avec m arêtes, l'espace mémoire est en **O(n+2m)**.
  - pour faire un traitement sur les successeurs d'un sommet s, le nombre de sommets parcourus est exactement le nombre de successeurs de s, soit $d^{\circ +}(s)$.
- **Inconvénients :**
  - exige, dans le pire de cas, un temps d'ordre n pour tester s'il existe un arc (resp. une arête) entre un sommet donné x et un sommet y (cas où la liste d'adjacence est de longueur n-1 et où y est en fin de liste) ou pour l'ajout d'un arc ou d'une arête (avec test de non répétition).
  - ne permet pas de calculer facilement les opérations relatives aux prédécesseurs ($d^{\circ -}$ et ième_pred).
- Représentation convenable pour les grands graphes :
  - utilisation de moins d'espace mémoire et parcours rapide des successeurs d'un sommet

48