ALGORITHMES POLYNOMIAUX DE BASE POUR LES GRAPHES

Notions de base

- Un problème est une « question » qui cherche une réponse et qui comporte un ensemble de paramètres.
  - description des paramètres;
  - propriétés que la réponse ou la *solution,* doit satisfaire;
  - *Une instance du problème est obtenue quand on fixe les paramètres à des valeurs données.*
- Algorithme : un ensemble d'instructions décrivant comment réaliser une tâche ou résoudre un problème.
- Programme : la réalisation informatique d'un algorithme

Complexité des algorithmes (I)

- Il peut exister plusieurs algorithmes pour le même problème…
- Plusieurs questions se posent :
  - Lequel choisir ?
  - Comment les comparer ?
  - Comment mesurer leur efficacité ?
  - Quelles mesures, *temps*, espace (mémoire)…?

Complexité des algorithmes (II)

- Le temps d'exécution du programme dépend :
  - des données du problème, de leur taille,
  - de la qualité du code…,
  - de la puissance de l'ordinateur,
  - de l'efficacité de l'algorithme,
  - etc.
- On procède alors par analyse de l'algorithme.
  - On cherche une grandeur *n* pour quantifier les données d'entrée.
  - On calcule les performances seulement en fonction de *n*.
  - On évalue le nombre d'opérations élémentaires, *(on appel opération élémentaire toute instruction d'un langage informatique).*

Complexité des algorithmes (III)

- Chaque algorithme est composé d'une phase de lecture de données et d'une phase de calcul.
  - Le paramètre de complexité d'un algorithme est la taille d'une donnée (nous utilisons le codage binaire)

Définition de mesure de calcul de complexité :

Soit un entier $n>0$ et $T(n)$ le temps d'exécution d'un algorithme en fonction de la taille n des données, on dira que $T(n)$ est en $O(f(n))$ si et seulement si $\exists n_0$ et une constante c tels que :

$\forall n \geq n_0$, on a que $T(n) \leq c\, f(n)$

Si f(n) est un polynôme alors l'algorithme est de complexité polynomiale.

De façon similaire :

- T(n) est dite en $\Omega(f(n))$ s'il existe deux constantes n_0 et c tel que : $\forall n \geq n_0$, on a que $T(n) \geq c\, f(n)$.
- T(n) est dite en $\Theta(f(n))$ si T(n) est à la fois en $O(f(n))$ et en $\Omega(f(n))$.

Complexité des algorithmes (IV) exemples

Étudier la complexité au pire cas

Soit N nombres a_1, a_2, .., a_n appartenant à {1,..., K}.
L'algorithme MIN détermine leur minimum.
Complexité O(n).

Algorithme MIN
début
pour i=1 à n lire a_i;
B:=a_1, j:=1;
pour i=2 à n faire :
Si B=1 alors i:=n;
sinon: si a_i < B alors
début
j:=i;
B:=a_I;
fin;
écrire : le plus petit nombre est a_j;
fin.

Algorithme de recherche dichotomique

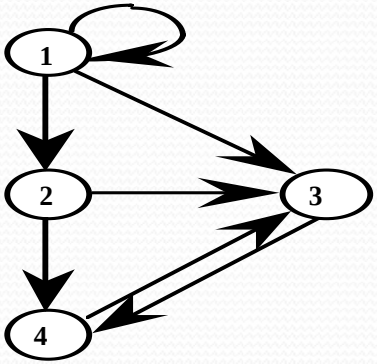
```
static int rechercheDichotomique(int[] t, int x)
  {
     int m, g, d, cmp;
     g = 0; d = N-1;
     do{
        m = (g+d)/2;
        if(t[m] == x)
           return m;
        else if(t[m] < x)
           d = m-1;
        else
           g = m+1;
     } while(g <= d);
     return -1;
```

Complexité des algorithmes

- O(1) temps constant
- O(log n) logarithmique
- O(n) linéaire
- O(n log n) p.ex. algorithme de tris
- $O(n^2)$ quadratique, polynomial
- $O(n^3)$ cubique, polynomial
- $O(2^n)$ exponentiel

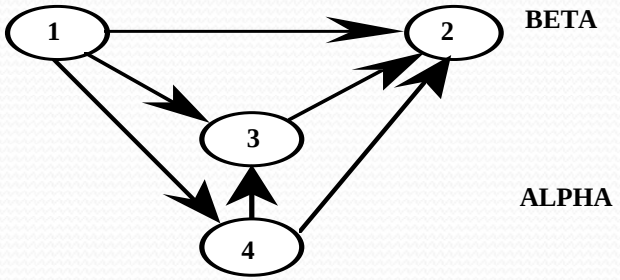
Codage d'un graphe (rappel)

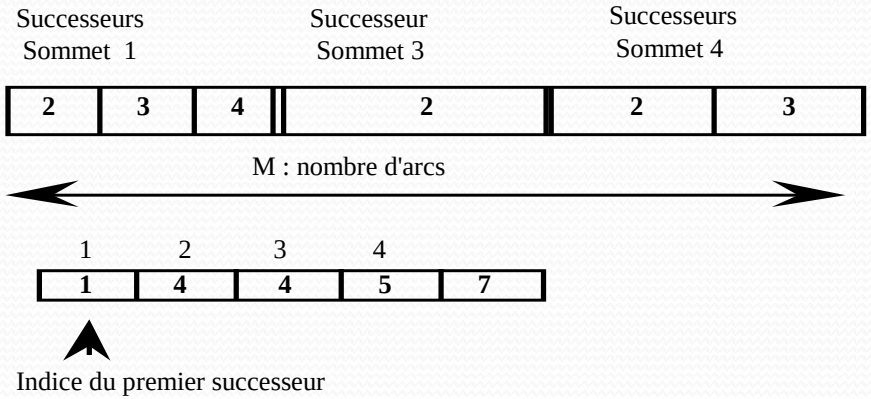
- Matrice associée



$$\begin{vmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{vmatrix}$$

- File des successeurs





Bonne numérotation

- **Bonne numérotation :** On dit que numéro(x) (application de X -> [1..n]) est une bonne numérotation si, pour tout arc (x,y) appartenant à U, numéro(x) est strictement plus petit que numéro(y).

- **Proposition 1**
  - Une condition nécessaire et suffisante pour qu'un graphe soit sans circuit est que tout sous-ensemble de sommets A non vide admette au moins un élément dont tous les prédécesseurs sont dans le complémentaire de A. C'est-à-dire le sous-graphe G_A a au moins un sommet sans prédécesseur.
- **Proposition 2 :**
  - Une condition nécessaire et suffisante pour qu'un graphe soit sans circuit est qu'il existe une bonne numérotation.

ALGORITHME NUMEROTER

(i) **Initialisation**

Lire N, M, ALPHA, BETA.{ALPHA est un tableau de dimension N+1 et BETA, de dimension M}

(ii) **Calcul du nombre de prédécesseurs de chaque sommet**

```
Pour I = 1  à N  faire NOMBRE(I) = 0
Pour K = 1 à M  faire
Début
    I = BETA(K); NOMBRE(I) = NOMBRE(I) + 1
Fin
```

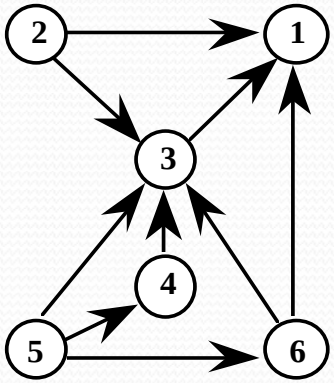
(iii) **Initialisation de la pile des sommets sans prédécesseurs**

```
SOMMET = 0
Pour I = 1 à N  faire
Début
    Si NOMBRE(I) = 0 faire
    Début
            SOMMET = SOMMET + 1
            PILE(SOMMET) = I
    Fin
Fin
```

(iv) **Numérotation des sommets**

```
Pour J = 1 à N
Début
  I = PILE(SOMMET);
  SOMMET = SOMMET - 1
  NUMERO(I) = J
  Pour H = ALPHA(I)  à  ALPHA(I+1) - 1
  Début
      L = BETA(H);
      NOMBRE(L) = NOMBRE(L) - 1
      Si NOMBRE(L) = 0  alors
      Début
          SOMMET = SOMMET + 1
           PILE(SOMMET) = L
      Fin
   Fin
Fin
```

Application de l'algorithme



Iteration 1

BETA

1	3	1	3	3	4	6	1	3
2		3	4		5		6	

ALPHA

1	1	3	4	5	8	10

NOMBRE

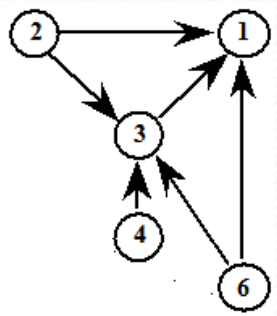
1	2	3	4	5	6
3	0	4	1	0	1

PILE

5
2

← SOMMET = 2

Iteration 2



NOMBRE

1	2	3	4	5	6
3	0	3	0	0	0

PILE

6
4
2

←

ALGORITHME NUMEROTER

ALGORITHME NUMEROTATION

[Version Récursive]

[NUMERO est un tableau commun]

INDICE = 1

NUMEROTER [G = (X, U), INDICE]

Begin

Déterminer un sommet j sans prédécesseur

NUMERO(j) = INDICE

Si INDICE plus petit à N faire

NUMEROTER [G - j, INDICE +1]

End.

LE PROBLEME DU COUPLAGE MAXIMAL

DEFINITIONS

THEOREME DE BERGE

ALGORITHME POUR UN GRAPHE BIPARTI

COMPLEXITE

PREUVE

LE PROBLEME DU COUPLAGE MAXIMAL

DEFINITIONS

COUPLAGE

DEUX APPLICATIONS

Équipe de pilotes

Mariage

SOMMET SATURE, SOMMET INSATURE

COUPLAGE MAXIMAL

COUPLAGE MAXIMAL DE COUT MINIMAL

CHAINE ALTERNEE

CHAINE AMELIORANTE

Chaîne alternée de longueur impaire

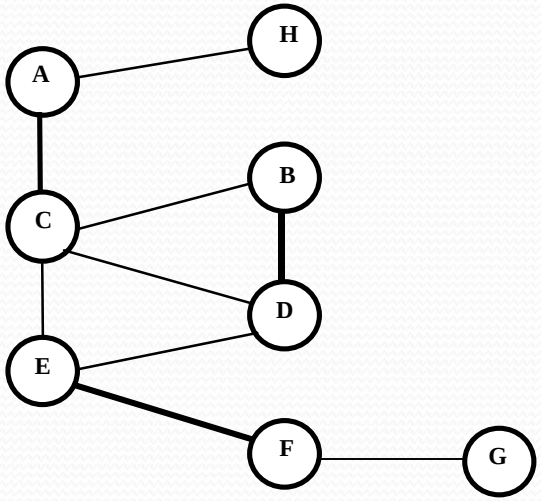
Couplage maximal

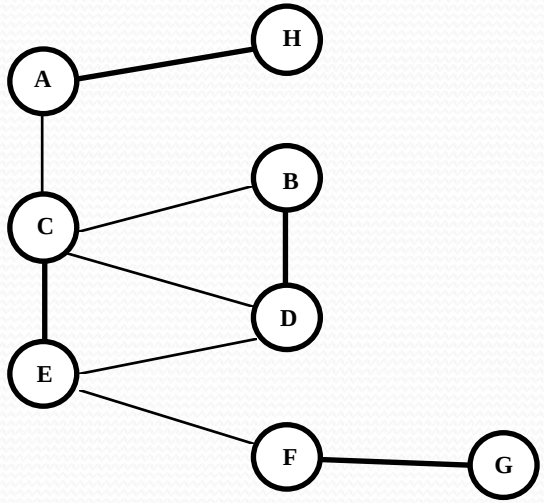
Couplage : Un couplage **C** est un ensemble d'arêtes (resp. d'arcs) ne contenant pas deux arêtes (resp. arcs) adjacentes à un même sommet. Les arcs du couplage seront dits épais les autres minces.

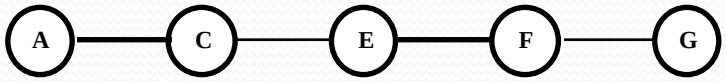
Sommet saturé, insaturé : Pour un couplage **C** un sommet est dit saturé si un arc du couplage est adjacent à ce sommet sinon il est dit insaturé.

Chaîne alternée, améliorante : Une chaîne alternée d'un couplage **C** est une chaîne du graphe dont les arcs sont alternativement dans le couplage et hors du couplage. Une chaîne améliorante est une chaîne alternée reliant deux sommets insaturés.

Des exemples illustratifs







Théorème de BERGE

UN COUPLAGE EST DE CARDINAL MAXIMAL SI ET SEULEMENT SI LE GRAPHE NE CONTIENT PAS DE CHAINE AMELIORANTE

Principe de la preuve :

CN - l'existence d'une chaine améliorante permettrait d'améliorer le couplage…

CS – notons C et C' respectivement un couplage sans chaine améliorante et un couplage max.

Posons D = C-C∩C' et D'=C'-C∩C';

Construisons H=(X, D∪D'), ayant des sommets de degrés 0, 1 ou 2.

Toute chaine est alors de longueur paire.

On en déduit que C et C' sont de même cardinalité.

Algorithme COUPLAGE MAX

Début.

Etape 0 : Initialisation

Lire les données - un graphe biparti G=(X,Y,U) ainsi qu'un couplage initial C (éventuellement C est vide);

Etape 1 : Recherche d'une chaine améliorante et amélioration du couplage

Marquer * tout sommet insaturé x dans X ;

Tant qu'il existe un sommet i marqué non examiné faire:

Début

(a) Si i = x ∈ X faire

Pour chaque arête [x, y] ∉ C incidente au sommet x faire:

Si y n'est pas marqué alors marquer y par x;

(b) Sinon, (i = y ∈ Y) faire :

Si y est insaturé, (il est extrémité d'une chaîne améliorante):

Améliorer le couplage C en inversant la chaîne améliorante obtenue;

Effacer les marques;

Marquer * tout sommet insaturé x dans X ;

Sinon

Déterminer l'unique arête [x, y] ∈ C;

Marquer x par y;

Fin (Tant que.)

Etape 2 : Fin .

Ecrire le couplage maximal C.

Fin

Algorithme COUPLAGE MAX
structure de données

STRUCTURE DE DONNEES

TAB(x,y) :

- 0 pas d'arête
- 1 une arête hors couplage
- 2 une arête dans le couplage

SATURE

MARQUE

FILE ou PILE

CHAINE

Algorithme COUPLAGE MAX
COMPLEXITE DE L'ALGORITHME

Étape 0 : $O(N^2)$ (lecture d'une matrice).

Le coût de l'étape 1 « sans amélioration de couplage » est $O(N^2)$, et le nombre maximal d'améliorations est de N/2. Donc sur l'ensemble de l'algorithme cette étape coûtera $O(N^3)$.

La complexité de l'étape 2 est O(N).

Donc la complexité de l'algorithme est $O(N^3)$.

Algorithme COUPLAGE MAX
preuve de l'algorithme

SCHEMA DE L'ALGORITHME

ALGORITHME COUPLAGE MAX

INITIALISER LE COUPLAGE

TANT QU'IL EXISTE UNE CHAINE AMELIORANTE CHERCHER UNE TELLE CHAINE AMELIORER LE COUPLAGE

RECHERCHE D'UNE CHAINE AMELIORANTE

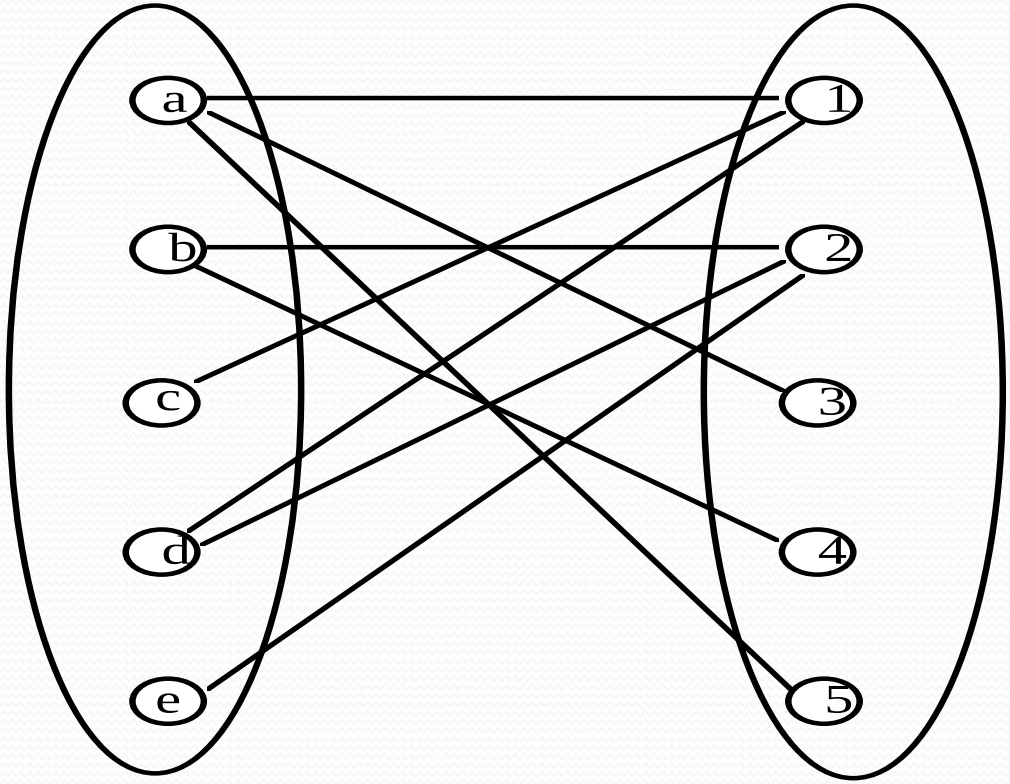
MARQUER TOUT SOMMET INSATURE de X

TANT QU'IL EXISTE UN SOMMET i MARQUE ET NON EXAMINE :

SI i APPARTIENT A Y ET EST INSATURE, ON A TROUVE UNE CHAINE AMELIORANTE

SINON EXAMINER i

Un exemple



Algorithme COUPLAGE MAX
principe de la preuve de l'algorithme

PROPRIETE 1

DANS UN GRAPHE BIPARTI G=(X,Y,U), UNE CHAÎNE DE LONGUEUR IMPAIRE RELIE UN SOMMET DE X A UN SOMMET DE Y .

DANS UN GRAPHE BIPARTI G=(X,Y,U) UNE CHAÎNE DE LONGUEUR PAIRE RELIE :

SOIT UN SOMMET DE X A UN SOMMET DE X .

SOIT UN SOMMET DE Y A UN SOMMET DE Y

PROPRIETE 2

DANS UN GRAPHE BIPARTI G=(X,Y,U), UNE CHAÎNE AMELIORANTE RELIE UN SOMMET DE X A UN SOMMET DE Y

PROPRIETE 3

UN SOMMET MARQUE EST EXTREMITE D'UNE CHAINE ALTERNEE ISSUE D'UN SOMMET INSATURE DE X

Algorithme COUPLAGE MAX
principe de la preuve de l'algorithme

PREUVE DE LALGORITHME

ON NE MARQUE QUE LES SOMMETS SATISFAISANT LA PROPRIETE 3 (démonstration par récurrence)

l'hypothèse de récurrence : si y dans Y (resp. x dans X) est un des p premiers sommets marqués il est extrémité d'une chaîne alternée issue d'un sommet x1 dans X insaturé de longueur impaire (resp. paire) dont la dernière arête est mince (resp. épaisse).

$\Rightarrow$ prouver ensuite pour p+1;

A LA FIN DE L'ALGORITHME, ON AURA MARQUE TOUS LES SOMMETS AYANT CETTE PROPRIETE 3 (démonstration par l'absurde)

Soit M l'ensemble des sommets marqués quand on arrive à l'étape 2 et montrons que M est alors exactement l'ensemble des sommets extrémités d'une chaîne alternée issue d'un sommet x1 dans X insaturé.

D'après la propriété démontrée par récurrence tous les sommets de M ont cette propriété. Il faut montrer que réciproquement un sommet ayant cette propriété est dans M.