Chapitre 1
Généralités sur les graphes / Notions fondamentales de la théorie des graphes

Présenté par :
Dr. H. BENKAOUHA
Bureau 222, Faculté d'Informatique, USTHB
haroun.benkaouha@usthb.edu.dz
haroun.benkaouha@gmail.com

Introduction

- Un graphe est défini par 2 ensembles :
  - Un ensemble de sommets noté X
  - Un ensemble de relations entre les sommets noté U ou E. Selon le type de cette relation.
- On distingue 2 grandes classes de graphes :
  - Graphes orientés si la relation est orientée
  - Graphes non orientés dans le cas contraire

Introduction – Graphe orienté

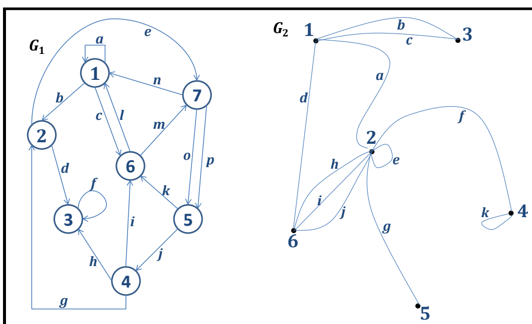
- $G=(X, U)$ est défini par 2 ensembles :
- $X = \{ x_1, x_2, ..., x_n \}$: Ensemble de sommets
  - n entier fini
  - $n \geq 1$,
  - chaque $x_i \in X$ est un sommet du graphe.
- $U = \{ u_1, u_2, ..., u_m \}$: Ensemble des arcs
  - $m \geq 0$ et fini,
  - Chaque $u_j \in U$ est une paire ordonnée de sommets
  - $u_j = (x, y)$.
  - x : extrémité initiale de u_j
  - y : extrémité terminale de u_j.
  - U peut être vide.

Introduction – Graphe non orienté

- $G=(X, E)$ est défini par 2 ensembles :
- $X = \{ x_1, x_2, ..., x_n \}$: Ensemble de sommets
  - n entier fini
  - $n \geq 1$,
  - chaque $x_i \in X$ est un sommet du graphe.
- $E = \{ e_1, e_2, ..., e_m \}$: Ensemble des arêtes
  - $m \geq 0$ et fini,
  - Chaque $e_j \in E$ est une paire non ordonnée de sommets
  - $e_j = \{x, y\} = \{y, x\}$.
  - x et y : extrémités de e_j
  - E peut être vide.

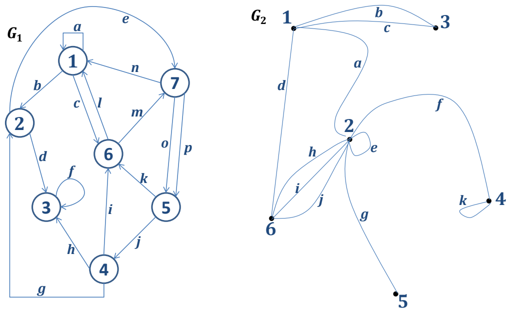
Introduction – Représentation

- Il n'y a pas vraiment de standard
- Généralement
  - sommet par un point ou un cercle
  - Un arc par une flèche
  - Une arête par un trait
  - La flèche et le trait peuvent être courbés
  - On peut rajouter des étiquettes



Graphe orienté :
$G_1=(X_1, U_1)$
$X_1=\{1, 2, ..., 7\}$
$U_1=\{a, b, ..., p\}$

Graphe non orienté :
$G_2=(X_2, E_2)$
$X_2=\{1, 2, ..., 6\}$
$E_2=\{a, b, ..., k\}$



a=(1,1), b=(1,2)
d=(2,3), e=(2,7)
g=(4,2), i=(4,6)
k=(5,6), p=(7,5)

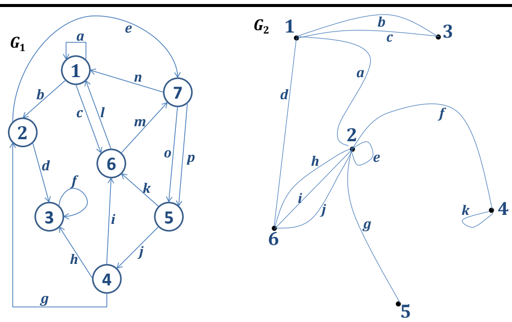
a={1,2}, b={1,3}
d={1,6}, e={2,2}
f={2,4}, g={2,5}
i={2,6}, k={4,4}

Enseignant : Dr. H. BENKAOUHA (Faculté d'Informatqiue - USTHB) 7

Définitions (1/5)

- Ordre du graphe = $|X|$ nombre de sommets
- Taille du graphe = $|U|$ ou $=|E|$ nombre d'arcs ou arêtes
- Extrémités confondues : boucle
- Mêmes extrémités : parallèles
- Graphe simple : pas de boucles, pas d'arcs (arêtes) parallèles

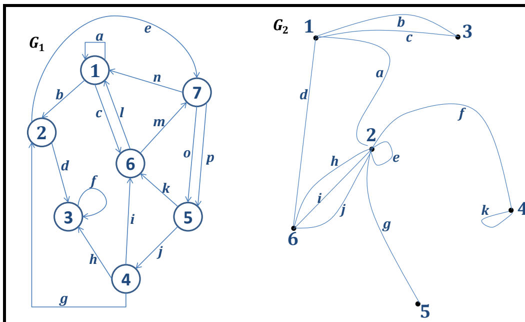
Enseignant : Dr. H. BENKAOUHA (Faculté d'Informatqiue - USTHB) 8



Graphe orienté : d'ordre 7 et de taille 16.

Graphe non orienté : d'ordre 6 et de taille 11.

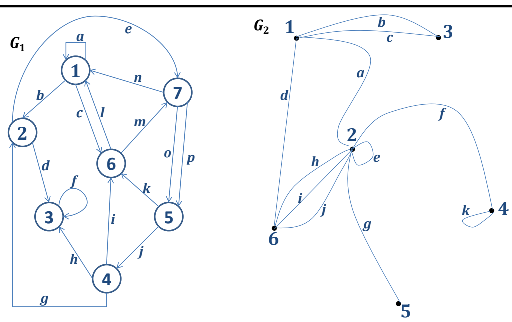
Enseignant : Dr. H. BENKAOUHA (Faculté d'Informatqiue - USTHB) 9



***a=(1,1)* et *f=(3,3)* sont des boucles**

***e={2,2}* et *k={4,4}* sont des boucles**

Enseignant : Dr. H. BENKAOUHA (Faculté d'Informatqiue - USTHB) 10



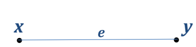
o // p
c // l

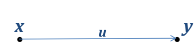
h // i // j
b // c

Enseignant : Dr. H. BENKAOUHA (Faculté d'Informatqiue - USTHB) 11

Définitions (2/5)

- Arc $u=(x, y)$ (Arête $e=\{x, y\}$)
  - x et y sont adjacents.
  - x et y incidents à u (resp. à e).
  - u (resp. e) est incident à x et y.
  - Uniquement l'arc u :
  - x prédécesseur de y.
  - y successeur de x
  - u est incident vers l'extérieur de x
  - u est incident vers intérieur de y

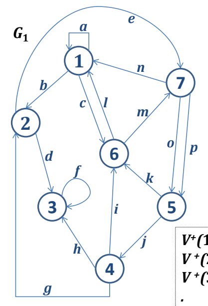




Enseignant : Dr. H. BENKAOUHA (Faculté d'Informatqiue - USTHB) 12

Définitions (3/5)

- G=(X, U), x ∈X, On a :
  - $\Gamma^+(x) = Succ(x) = \{ y \in X \,/\, (x, y) \in U\}$
  - Ensemble des successeurs du sommet *x*.
  - $\Gamma^-(x) = Pred(x) = \{ y \in X \,/\, (y, x) \in U\}$
  - Ensemble des prédécesseurs du sommet *x*.
- *G=(X, U)* (resp. *G=(X, E)*), Ens. voisins de $x \in X$:
  - $V(x) = \{y \in X\text{-}\{x\} \,/\, \{x, y\} \in E\}$ Cas non orienté
  - $V(x) = V^+(x) \cup V^-(x)$ Cas orienté
  - $V^+(x) = \{y \in X\text{-}\{x\} \,/\, (x, y) \in U\}$
  - $V^-(x) = \{y \in X\text{-}\{x\} \,/\, (y, x) \in U\}$.
  - $V^+(x)$ (resp. $V^-(x)$) est appelé ensemble des voisins externes (resp. internes) de *x*.



$\Gamma^+(1) = \{1, 2, 6\}$
$\Gamma^+(2) = \{3, 7\}$
$\Gamma^+(3) = \{3\}$
.
.
.
$\Gamma^+(7) = \{1, 5\}$

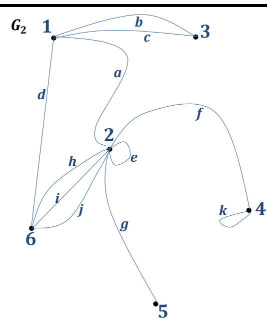
$\Gamma^-(1) = \{1, 6, 7\}$
$\Gamma^-(2) = \{1, 4\}$
$\Gamma^-(3) = \{2, 3, 4\}$
.
.
.
$\Gamma^-(7) = \{2, 6\}$

$V^+(1) = \{2, 6\}$
$V^+(2) = \{3, 7\}$
$V^+(3) = \{\} = \phi$
.
.
.
$V^+(7) = \{1, 5\}$

$V^-(1) = \{6, 7\}$
$V^-(2) = \{1, 4\}$
$V^-(3) = \{2, 4\}$
.
.
.
$V^-(7) = \{2, 6\}$

$V(1) = \{2, 6, 7\}$
$V(2) = \{1, 3, 4, 7\}$
$V(3) = \{2, 4\}$
.
.
.
$V(7) = \{1, 2, 5, 6\}$

$V(1) = \{2, 3, 6\}$
$V(2) = \{1, 4, 5, 6\}$
$V(3) = \{1\}$
$V(4) = \{2\}$
$V(5) = \{2\}$
$V(6) = \{1, 2\}$



h // i // j
b // c

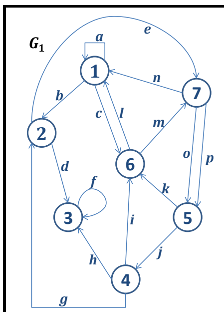
Définitions (4/5)

- Arcs (arêtes) adjacent(e)s : extrémité communes.
- Graphe orienté : 2 applications donnant l'extrémité initiale et terminale d'un arc donné:

$$I : U \rightarrow X \qquad\qquad T : U \rightarrow X$$
$$u=(x, y) \rightarrow x \qquad\qquad u=(x, y) \rightarrow y$$

Définitions (5/5)

- Multiplicité d'un arc (x_i, x_j)
  = la valeur m_{ij}
  = au nombre d'arcs qui relient x_i à x_j.
- Multiplicité d'un graphe *G*
  = $m(G)$ = maximum des m_{ij}.
- Si $m(G) = k$ on dit que *G* est un *k*-graphe



n* et *e* sont adjacents *(voir sommet 7)
h* et *j* sont adjacents *(voir sommet 4)

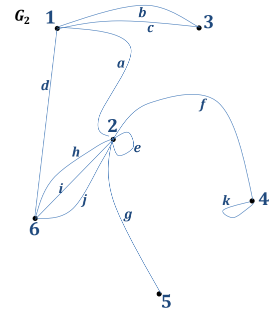
$I(b) = I(1, 2) = 1$
$T(b) = T(1, 2) = 2$
$I(p) = I(7, 5) = 7$
$T(p) = T(7, 5) = 5$

$m(1, 2) = 1$
$m(2, 1) = 0$
$m(1, 6) = 1$
$m(6, 1) = 1$
$m(7, 5) = 2$
...
$m(G_1) = 2$
➔ G_1 est un 2-graphe

Notion de degré – Général

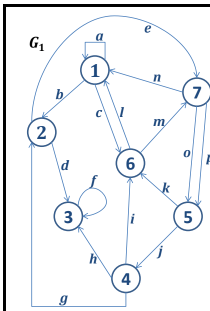
- *G = (X, E)* ou *G = (X, U)*
- $x \in X$, on peut lui associer une valeur entière positive ou nulle : $d_G(x)$,
- degré du sommet *x*.
- $d_G(x)$ = nombre de fois où *x* est extrémité d'un arc (resp. d'une arête).

Sommet *i*	$d_{G_2}(i)$
1	4
2	8
3	2
4	3
5	1
6	4



Notion de degré – Graphe orienté

- Soit *G* = (*X*, *U*) un graphe orienté.
  - demi-degré extérieur d'un sommet $x \in X$:
  - $d_G^+(x) = |\{ u \in U \,/\, I(u)=x \}|$.
  - demi-degré intérieur d'un sommet $x \in X$:
  - $d_G^-(x) = |\{ u \in U \,/\, T(u)=x \}|$.
  - $d_G(x) = d_G^+(x) + d_G^-(x)$



Sommet *i*	$d^+_{G_1}(i)$	$d^-_{G_1}(i)$	$d_{G_1}(i)$
1	3	3	6
2	2	2	4
3	1	3	4
4	3	1	4
5	2	2	4
6	2	3	5
7	3	2	5

Notion de degré – Remarques (1/3)

- Pour tout graphe, nous avons :
  - $d_G(x) \geq |V(x)|$.
  - Si *G* est simple Alors On a $d_G(x) = |V(x)|$.
- Pour tout graphe orienté, nous avons :
  - $d_G^+(x) \geq |V^+(x)|$ et $d_G^-(x) \geq |V^-(x)|$.
  - Si *G* est 1-graphe sans boucles Alors On a $d_G^+(x) = |V^+(x)|$ et $d_G^-(x) = |V^-(x)|$.

Notion de degré – Remarques (2/3)

- On appelle degré minimal d'un graphe *G*, le plus petit degré dans le graphe *G*.

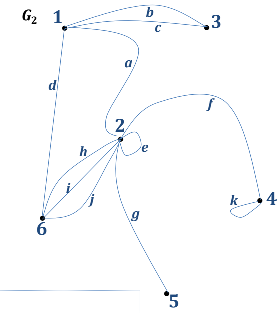
  $\delta(G) = \mathrm{Min}\ \{d_G(x)\}$
- On appelle degré maximal d'un graphe *G*, le plus grand degré dans le graphe *G*.

  $\Delta(G) = \mathrm{Max}\ \{d_G(x)\}$

Notion de degré – Remarques (3/3)

- Si $d_G(x) = 0$ Alors
  x est dit <u>sommet isolé</u>.
- Si $d_G(x) = 1$ Alors
  x est dit <u>sommet pendant</u>.
- Un arc (resp. Une arête) incident(e) à un sommet pendant est appelé(e) pendant(e).

Sommet i	$d_{G_2}(i)$
1	4
2	8
3	2
4	3
5	1
6	4



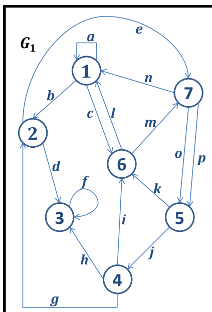
$\delta(G_2) = 1$
$\Delta(G_2) = 8$
5 sommet pendant
Ici, il n'y a pas de sommet isolé

Notion de degré – Formule des degrés

- **Cas non orienté :**
  Pour tout graphe : $G = (X, E)$, On a :
  $\sum d_G(x) = 2\,|E|$.
- **Cas orienté :**
  Pour tout graphe : $G = (X, U)$, On a :
  $\sum d_G(x) = 2\,|U|$ et $\sum d^+_G(x) = \sum d^-_G(x) = |U|$.
- **Conséquence :**
- Déduction ➔ nombre sommets degrés impairs toujours pair.

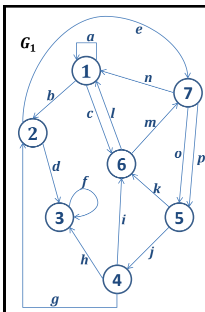
Représentation machine - Matrice d'adjacence (1/2)

- Tout graphe d'ordre n.
- Matrice M de n x n.
- Ligne i ➔ sommet i
- Colonne j ➔ sommet j
- M_{ij} : nombre d'arcs de i vers j.



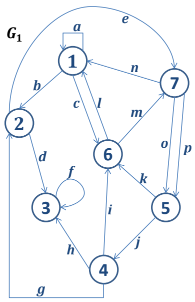
M

	1	2	3	4	5	6	7
1	1	1	0	0	0	1	0
2	0	0	1	0	0	0	1
3	0	0	1	0	0	0	0
4	0	1	1	0	0	1	0
5	0	0	0	1	0	1	0
6	1	0	0	0	0	0	1
7	1	0	0	0	2	0	0



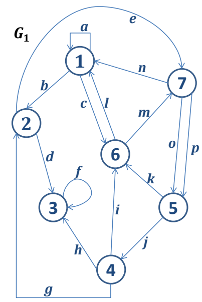
M

	1	2	3	4	5	6	7	$d^+_{G_1}(i)$
1	1	1	0	0	0	1	0	3
2	0	0	1	0	0	0	1	2
3	0	0	1	0	0	0	0	1
4	0	1	1	0	0	1	0	3
5	0	0	0	1	0	1	0	2
6	1	0	0	0	0	0	1	2
7	1	0	0	0	2	0	0	3



M

	1	2	3	4	5	6	7
1	1	1	0	0	0	1	0
2	0	0	1	0	0	0	1
3	0	0	1	0	0	0	0
4	0	1	1	0	0	1	0
5	0	0	0	1	0	1	0
6	1	0	0	0	0	0	1
7	1	0	0	0	2	0	0
$d^-_{G_1}(i)$	3	2	3	1	2	3	2

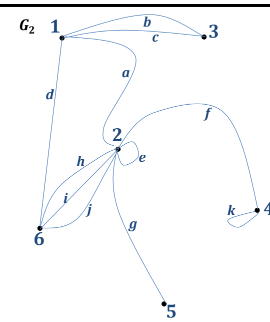


M

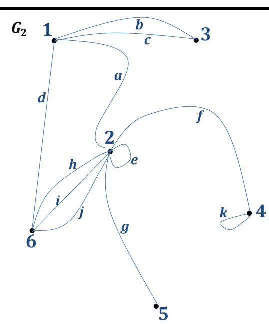
	1	2	3	4	5	6	7	$d^+_{G_1}(i)$
1	1	1	0	0	0	1	0	3
2	0	0	1	0	0	0	1	2
3	0	0	1	0	0	0	0	1
4	0	1	1	0	0	1	0	3
5	0	0	0	1	0	1	0	2
6	1	0	0	0	0	0	1	2
7	1	0	0	0	2	0	0	3
$d^-_{G_1}(i)$	3	2	3	1	2	3	2	

Représentation machine - Matrice d'adjacence (2/2)

- La somme d'une ligne $i = d_G^+(i)$
- La somme d'une colonne $j = d_G^-(j)$
- Si le graphe n'est pas orienté :
  - matrice symétrique,
  - boucle compte double.
  - Somme d'une ligne i = somme colonne $i = d_G(i)$



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



	1	2	3	4	5	6
1	0	1	2	0	0	1
2	1	2	0	1	1	3
3	2	0	0	0	0	0
4	0	1	0	2	0	0
5	0	1	0	0	0	0
6	1	3	0	0	0	0
$d_{G_2}(i)$	4	8	2	3	1	4

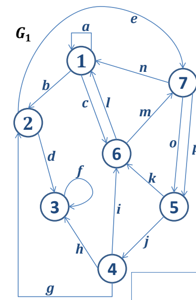
Représentation machine - Représentation par listes (1/2)

- Tout graphe d'ordre n et de taille m.
- 2 tableaux (vecteurs) *PS* et *LS.*
- *PS* : n+1 éléments
- *LS* : m éléments
- *PS*[*i*] : case contenant le 1er successeur de *i* dans *LS.*

Représentation machine - Représentation par listes (2/2)

- $PS[1]=1$ et $PS[n+1]=m+1$
- $PS[i]=PS[i-1]+d_G^+(i-1)$. Si i n'a pas de successeur : $PS[i]=PS[i+1]$
- Les successeurs d'un sommet i se trouvent entre la case n° $PS[i]$ et la case n° $PS[i+1]$-1 du tableau LS.

Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB) 37



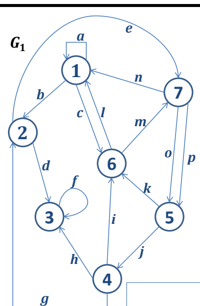
PS

1	2	3	4	5	6	7	8
1							

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

LS

Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB) 38



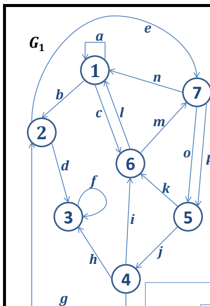
PS

1	2	3	4	5	6	7	8
1							

1	2	6													
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

LS

Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB) 39



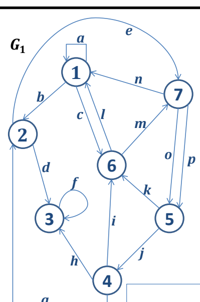
PS

1	2	3	4	5	6	7	8
1	4						

1	2	6													
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

LS

Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB) 40



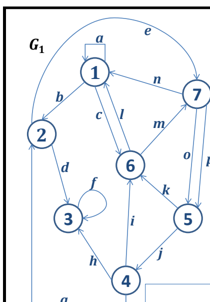
PS

1	2	3	4	5	6	7	8
1	4						

1	2	6	3	7											
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

LS

Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB) 41



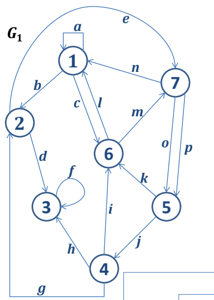
PS

1	2	3	4	5	6	7	8
1	4	6					

1	2	6	3	7											
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

LS

Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB) 42



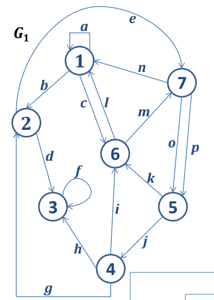
PS

1	2	3	4	5	6	7	8
1	4	6					

1	2	6	3	7	3										
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

LS

Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB) 43



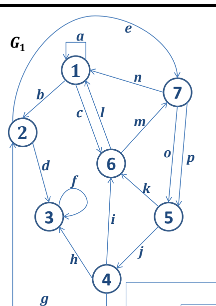
PS

1	2	3	4	5	6	7	8
1	4	6	7				

1	2	6	3	7	3										
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

LS

Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB) 44



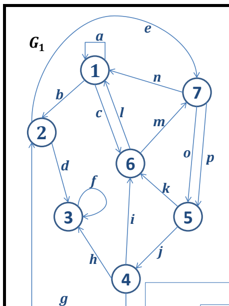
PS

1	2	3	4	5	6	7	8
1	4	6	7				

1	2	6	3	7	3	2	3	6							
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

LS

Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB) 45



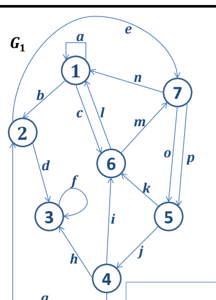
PS

1	2	3	4	5	6	7	8
1	4	6	7	10			

1	2	6	3	7	3	2	3	6							
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

LS

Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB) 46



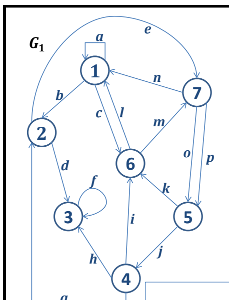
PS

1	2	3	4	5	6	7	8
1	4	6	7	10			

1	2	6	3	7	3	2	3	6	4	6					
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

LS

Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB) 47



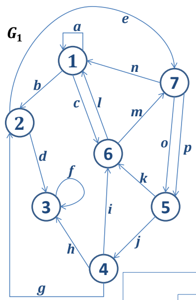
PS

1	2	3	4	5	6	7	8
1	4	6	7	10	12		

1	2	6	3	7	3	2	3	6	4	6					
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

LS

Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB) 48



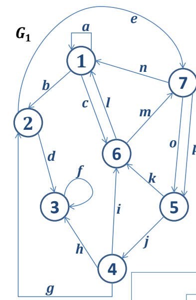
PS

1	2	3	4	5	6	7	8
1	4	6	7	10	12		

1	2	6	3	7	3	2	3	6	4	6	1	7			
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

LS

Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB) 49



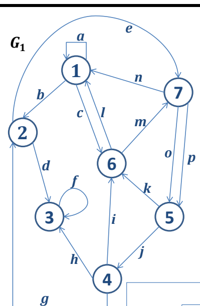
PS

1	2	3	4	5	6	7	8
1	4	6	7	10	12	14	

1	2	6	3	7	3	2	3	6	4	6	1	7			
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

LS

Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB) 50



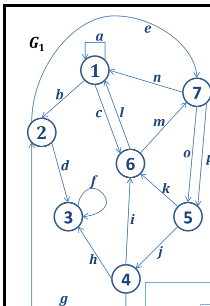
PS

1	2	3	4	5	6	7	8
1	4	6	7	10	12	14	

1	2	6	3	7	3	2	3	6	4	6	1	7	1	5	5
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

LS

Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB) 51



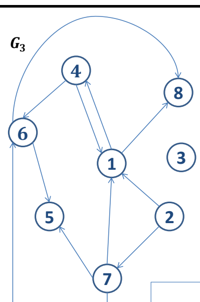
PS

1	2	3	4	5	6	7	8
1	4	6	7	10	12	14	17

1	2	6	3	7	3	2	3	6	4	6	1	7	1	5	5
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

LS

Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB) 52



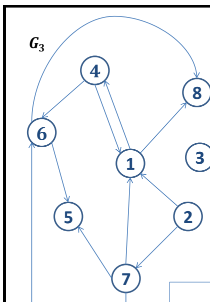
PS

1	2	3	4	5	6	7	8	9
1								

4	8									
1	2	3	4	5	6	7	8	9	10	11

LS

Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB) 53



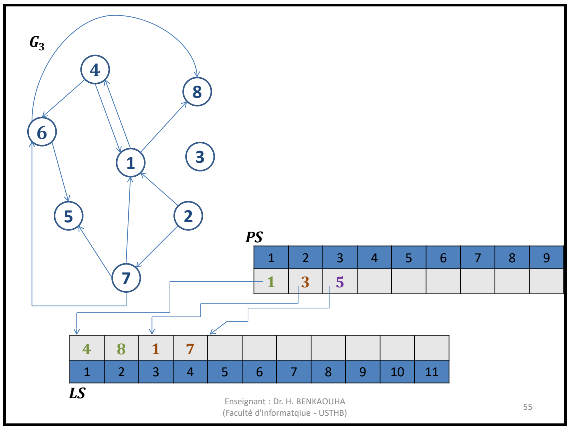
PS

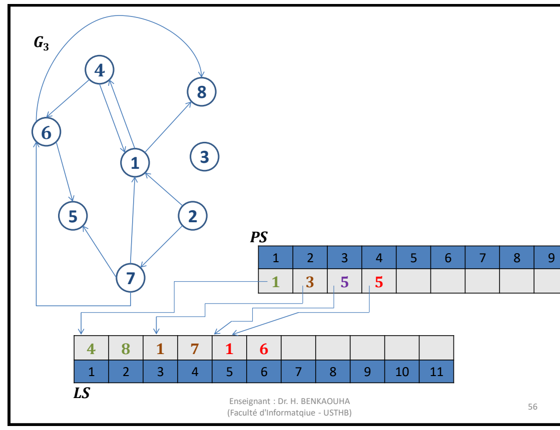
1	2	3	4	5	6	7	8	9
1	3							

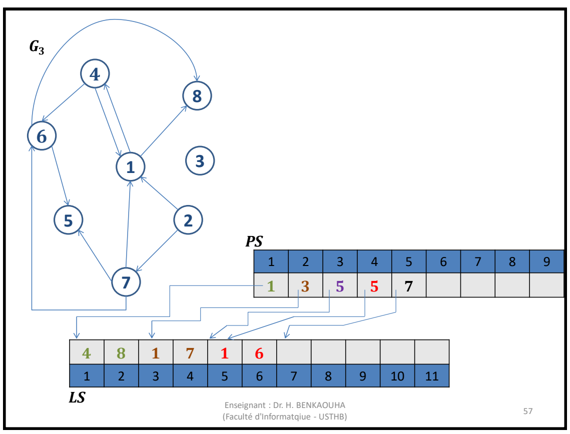
4	8	1	7							
1	2	3	4	5	6	7	8	9	10	11

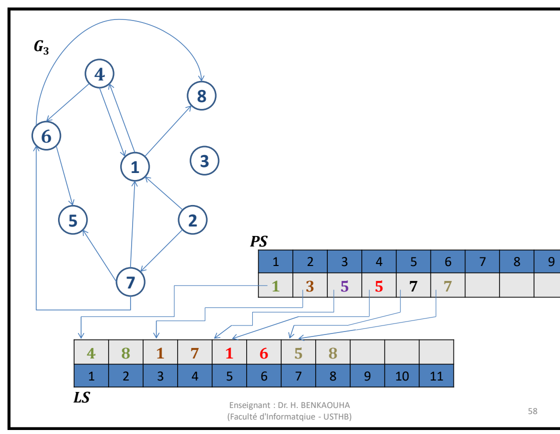
LS

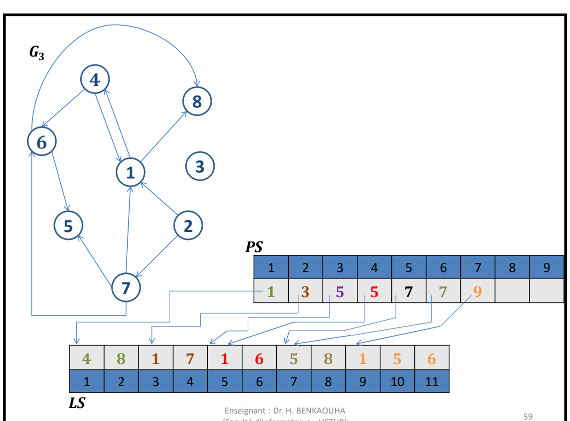
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB) 54

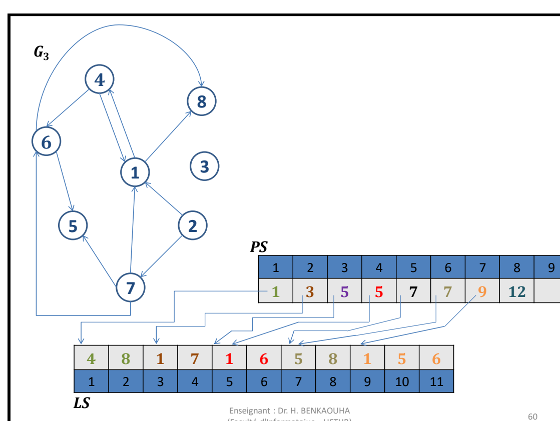

G3
4 8 6 1 3 5 2 7
PS
1 2 3 4 5 6 7 8 9
1 3 5
4 8 1 7
1 2 3 4 5 6 7 8 9 10 11
LS
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB)
55

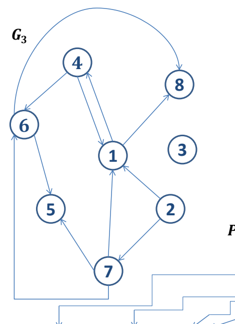

G3
4 8 6 1 3 5 2 7
PS
1 2 3 4 5 6 7 8 9
1 3 5 5
4 8 1 7 1 6
1 2 3 4 5 6 7 8 9 10 11
LS
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB)
56


G3
4 8 6 1 3 5 2 7
PS
1 2 3 4 5 6 7 8 9
1 3 5 5 7
4 8 1 7 1 6
1 2 3 4 5 6 7 8 9 10 11
LS
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB)
57


G3
4 8 6 1 3 5 2 7
PS
1 2 3 4 5 6 7 8 9
1 3 5 5 7 7
4 8 1 7 1 6 5 8
1 2 3 4 5 6 7 8 9 10 11
LS
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB)
58


G3
4 8 6 1 3 5 2 7
PS
1 2 3 4 5 6 7 8 9
1 3 5 5 7 7 9
4 8 1 7 1 6 5 8 1 5 6
1 2 3 4 5 6 7 8 9 10 11
LS
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB)
59


G3
4 8 6 1 3 5 2 7
PS
1 2 3 4 5 6 7 8 9
1 3 5 5 7 7 9 12
4 8 1 7 1 6 5 8 1 5 6
1 2 3 4 5 6 7 8 9 10 11
LS
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB)
60



PS

1	2	3	4	5	6	7	8	9
1	3	5	5	7	7	9	12	12

4	8	1	7	1	6	5	8	1	5	6
1	2	3	4	5	6	7	8	9	10	11

LS

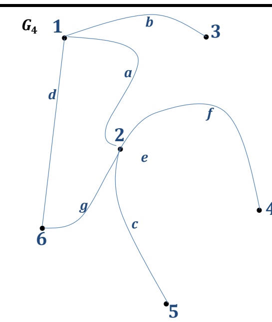
Algo : Dessiner le Graphe à partir des Listes PS et LS

```
Pour i de 1 à n
    dessiner_sommet(i)
Pour i de 1 à n
    j=PS[i]
    k=PS[i+1]-1
    Pour p de j  à k
        dessiner_arc(i,LS[p])
```


Propriétés des Graphes – Graphe simple

- Ni boucles,
- Ni arcs parallèles,
- Si G est simple, on a $d_G(x) = |V(x)|$.

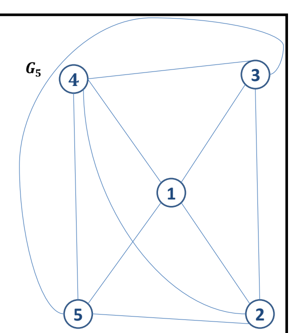
Exemple de graphe simple



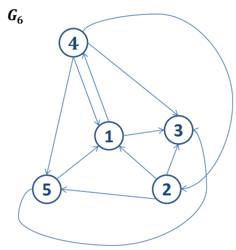
Propriétés des Graphes – Graphe complet

- Cas orienté :
  - G est complet **ssi** $\forall x \neq y \in X, (x, y) \notin U \Rightarrow (y, x) \in U$
- Cas non orienté :
  - G est complet **ssi** $\forall x \neq y \in X, \{x, y\} \in E$.
- Un graphe simple complet d'ordre n est noté K_n.

Graphe non-orienté complet. Il est aussi simple $\Rightarrow K_5$



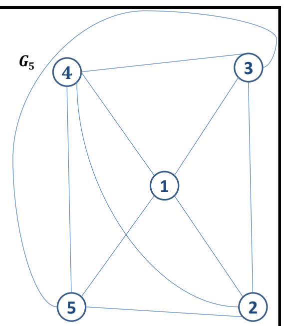
Graphe orienté complet : Le sens des arcs n'est pas important.



Propriétés des Graphes – Graphe régulier

- G est k-régulier : $\forall x$ sommet de G, on a $d_G(x) = k$.
- En d'autres termes, $\delta(G) = \Delta(G) = k$.
- Si $k = 0$, G est un graphe sans arêtes (sans arcs) appelé stable. G est constitué seulement de sommets isolés.
- Si $k = 1$, G est constitué d'arcs (arêtes) dispersé(e)s dans l'espace.

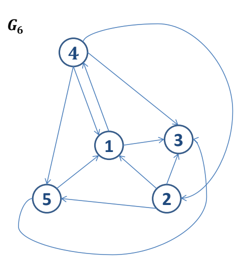
Graphe 4-régulier



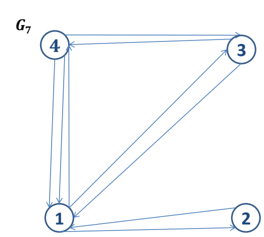
Propriétés des Graphes – Graphe symétrique

- Uniquement graphes orientés.
- G est symétrique **ssi**
  - $\forall x, y \in X, (x, y) \in U \Rightarrow (y, x) \in U$

Graphe orienté : Ce graphe n'est pas symétrique malgré qu'il y a deux arcs symétriques.



Graphe orienté : Ce graphe est symétrique.

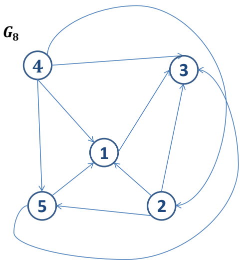


Propriétés des Graphes – Graphe antisymétrique

- Uniquement graphes orientés.
- G est antisymétrique **ssi**
  - $\forall x, y \in X, (x, y) \in U \Rightarrow (y, x) \notin U$

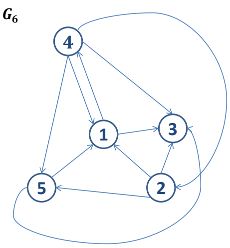
Graphe orienté : Ce graphe est anti-symétrique.

G_8



Graphe orienté : Ce graphe n'est pas anti-symétrique et il n'est pas symétrique.

G_6

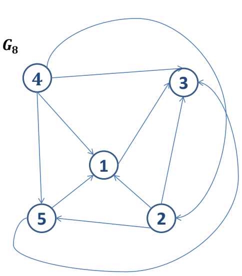


Propriétés des Graphes – Graphe transitif

- Uniquement graphes orientés.
- G est transitif **ssi**
  - $\forall x, y, z \in X, (x, y) \in U$ et $(y, z) \in U \Rightarrow (x, z) \in U$

Graphe orienté : Ce graphe est transitif.

G_8

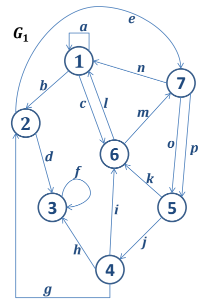


Propriétés des Graphes – Graphe biparti

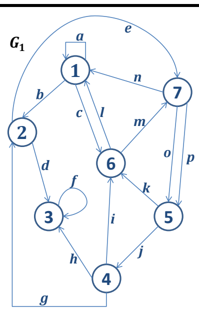
- G <u>biparti</u> ssi X admet une partition en 2 sous ensembles X_1 et X_2 avec $X_1 \cap X_2 = \varnothing$ et $X_1 \cup X_2 = X$.
- Cas orienté : $\forall (x, y) \in U \Rightarrow x \in X_1$ et $y \in X_2$
- Cas non orienté : $\forall \{x, y\} \in E$ $(x \in X_1$ et $y \in X_2)$ ou $(x \in X_2$ et $y \in X_1)$
- G <u>biparti complet</u> **ssi** G biparti et $\forall x \in X_1$ et $\forall y \in X_2 \Rightarrow (x, y) \in U$.
- Un graphe biparti complet et simple $G=(X_1 \cup X_2, U)$ (resp. $G=(X_1 \cup X_2, E)$)avec $|X_1|=p$ et $|X_2|=q$ est noté $\boldsymbol{K_{p,q}}$.

Graphes particuliers – Sous-graphe

- Un sous graphe de G engendré par l'ensemble de sommets A est le graphe :
- $G_A = (A, U_A)$ où $U_A = \{u \in U \ / \ I(u) \in A \text{ et } T(u) \in A\}$ dans le cas orienté.
- $G_A = (A, E_A)$ où $E_A = \{e = \{x, y\} \in E \ / \ x \in A \text{ et } y \in A \}$ dans le cas non orienté.
- Si on pose B=X-A, on note G_A aussi G-B

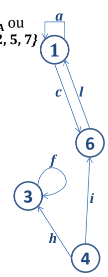


Le sous graphe de G_1 engendré par l'ensemble de sommets A={1, 3, 4, 6}
G_{1A} ou G_1 – {2, 5, 7}



Le sous graphe de G_1 engendré par l'ensemble de sommets A={1, 3, 4, 6}

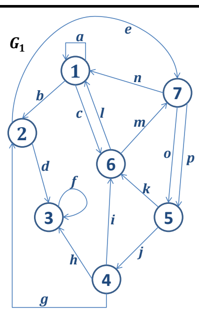
G_{1A} ou **G_1 – {2, 5, 7}**



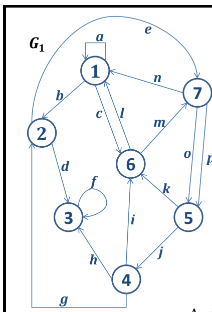
On supprime les arcs *b, d, e, g, j, k, l, m, n, o, p* car ils ont au moins une extrémité qui n'existe pas.

Graphes particuliers – Graphe partiel

- Un graphe partiel de G engendré par l'ensemble d'arcs (resp. d'arêtes) V est :
- Le graphe $G_V = (X, V)$.
- Si on pose W=U-V, on note G_V aussi G-W

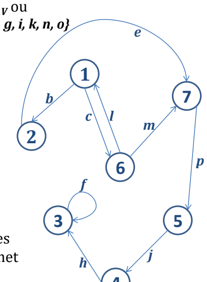


Le graphe partiel de G_1 engendré par l'ensemble des arcs V={b, c, e, f, h, j, l, m, p}
G_{1V} ou G_1 – {a, d, g, i, k, n, o}



Le graphe partiel de G_1 engendré par l'ensemble des arcs V={b, c, e, f, h, j, l, m, p}

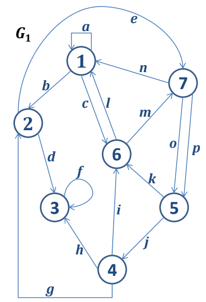
G_{1V} ou **G_1 – {a, d, g, i, k, n, o}**



Aucun sommet n'est supprimé dans les graphes partiels, même si le sommet est isolé.

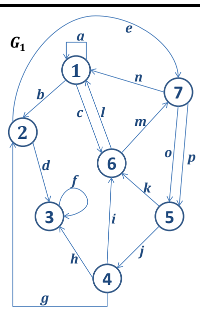
Graphes particuliers – Sous-graphe partiel

- Un sous graphe partiel de G engendré par l'ensemble de sommets A et l'ensemble d'arcs (resp. d'arêtes) V est le graphe $G_{A,V} = (A, V_A)$.
- V_A est l'ensemble d'arcs (resp. arêtes) qui ont leurs deux extrémités dans le sous ensemble V.

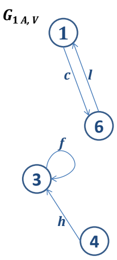


Le sous graphe partiel de G_1 engendré par l'ensemble de sommets $A=\{1, 3, 4, 6\}$ et l'ensemble des arcs $V=\{b, c, e, f, h, j, l, m, p\}$

$G_{1\ A,V}$

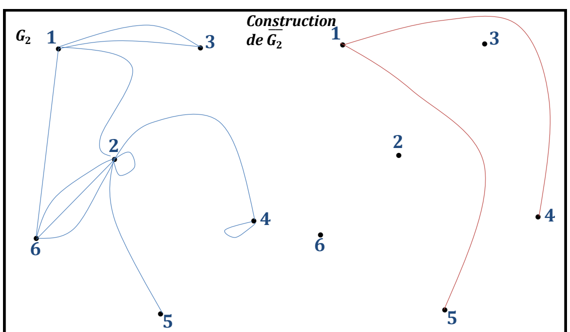


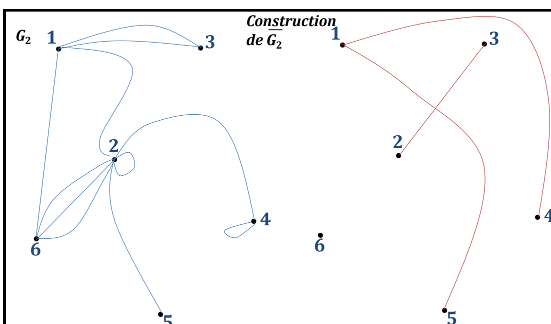
Le sous graphe partiel de G_1 engendré par l'ensemble de sommets $A=\{1, 3, 4, 6\}$ et l'ensemble des arcs $V=\{b, c, e, f, h, j, l, m, p\}$

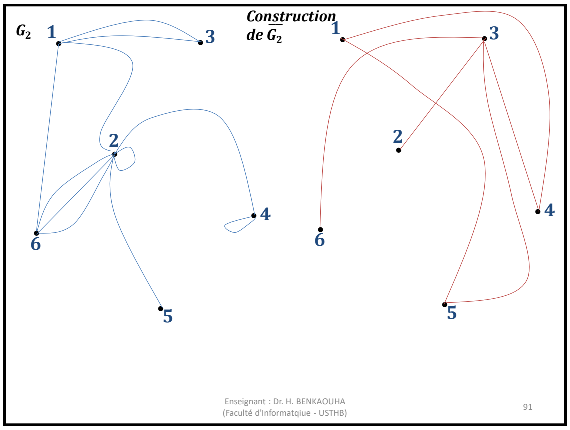


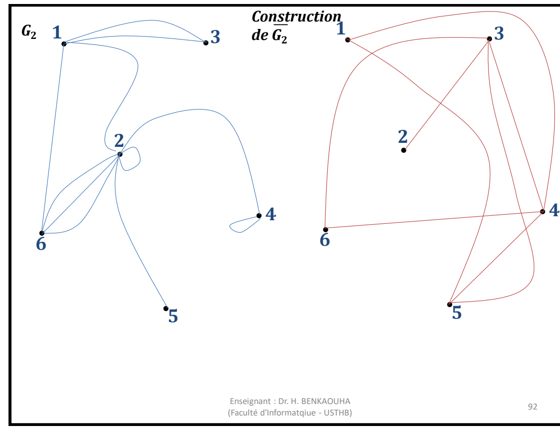
Graphes particuliers – Complément d'un graphe

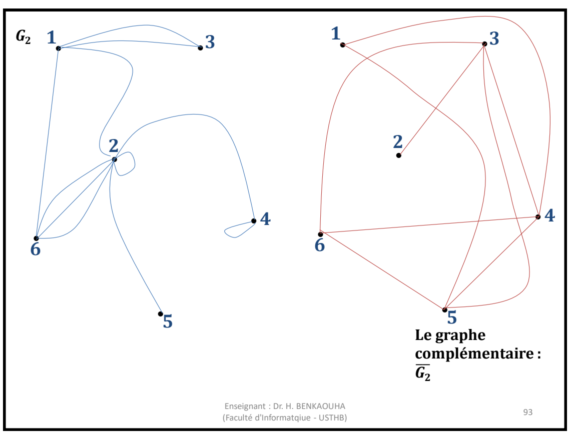
- Le graphe complémentaire de G est noté
  - $\overline{G} = (X, \overline{U})$ (resp. $\overline{G} = (X, \overline{E})$) où :
  - $\overline{U} = \{(x, y) \in X^2 \ / \ x \neq y$ et $(x, y) \notin U\}$
  - resp. $\overline{E} = \{ \{x, y\} \in X^2 \ / \ x \neq y$ et $\{x, y\} \notin E\}$

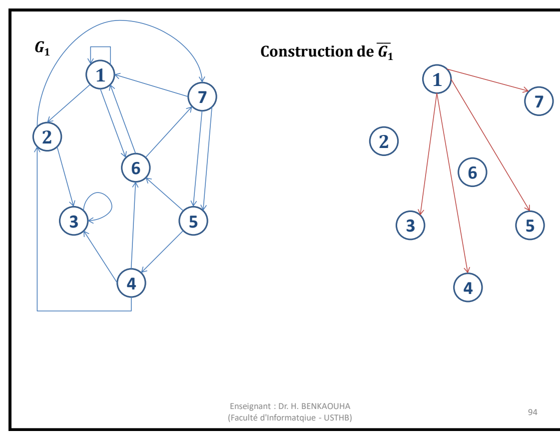


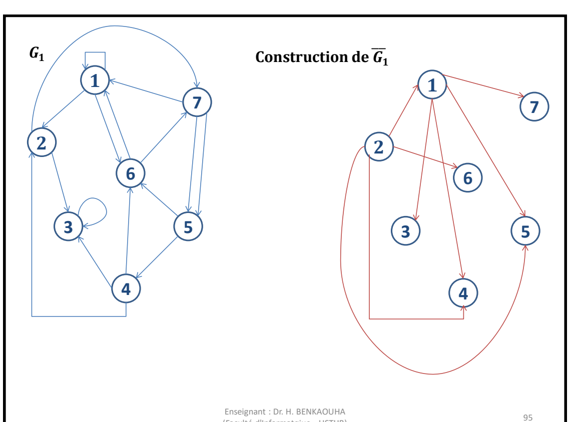


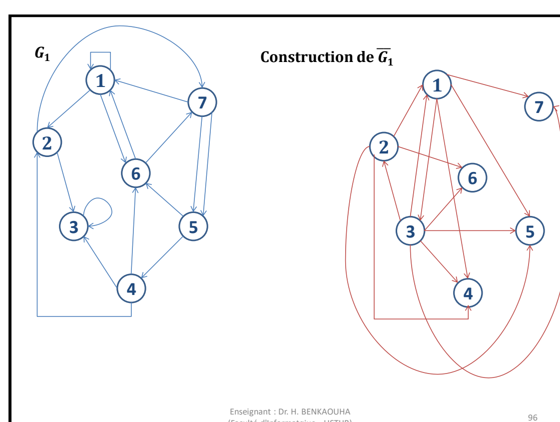

G_2
Construction de $\overline{G_2}$
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB)
91

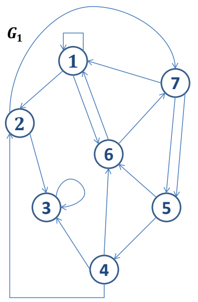

G_2
Construction de $\overline{G_2}$
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB)
92


G_2
Le graphe complémentaire : $\overline{G_2}$
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB)
93

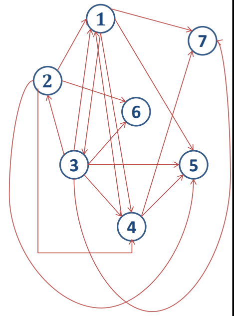

G_1
Construction de $\overline{G_1}$
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB)
94


G_1
Construction de $\overline{G_1}$
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB)
95

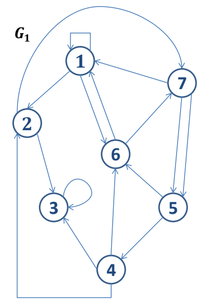

G_1
Construction de $\overline{G_1}$
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB)
96



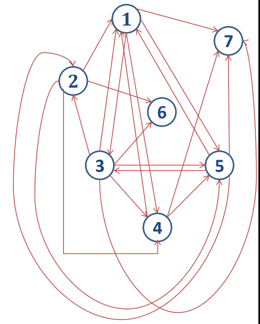
Construction de $\overline{G_1}$



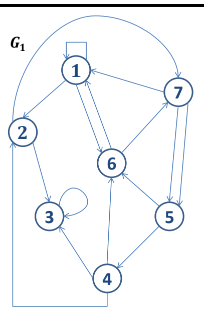
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB) 97



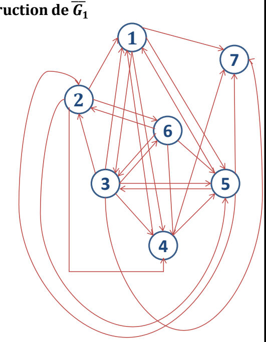
Construction de $\overline{G_1}$



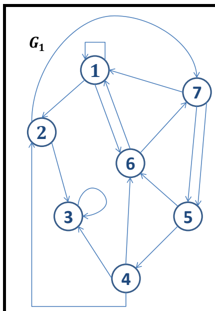
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB) 98



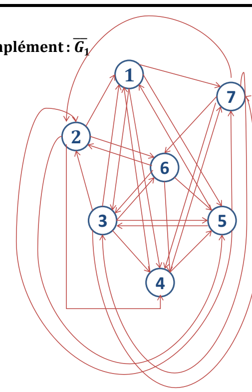
Construction de $\overline{G_1}$



Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB) 99



Le complément : $\overline{G_1}$

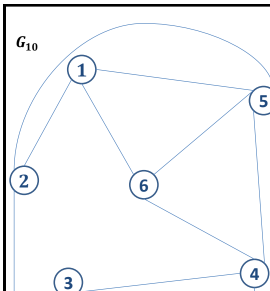


Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB) 100

Stable / Clique

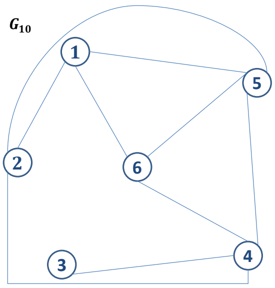
- Stable dans G :
  - Sous-ensemble de sommets $S \subseteq X$
  - Sous graphe engendré par S est formé de sommets isolés.
  - Chaque partition d'un graphe biparti forme un stable.
- Clique dans G :
  - Sous-ensemble de sommets $C \subseteq X$
  - Sous graphe engendré par C est complet.

Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB) 101



S_1={1, 4} ➔ Stable de 2 éléments.
S_2={2, 6} ➔ Stable de 2 éléments.
S_3={2, 3, 6} ➔ Stable de 3 éléments.

Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB) 102



C_1={3, 4} ➔ Clique de 2 éléments.
C_2={4, 5, 6} ➔ Clique de 3 éléments.
C_3={1, 2, 5} ➔ Clique de 3 éléments.

Coloration des sommets d'un graphe

- <u>*k*-coloration</u> de $G=(X, E)$: une application φ

$$\varphi : \quad X \rightarrow \{1, 2, ..., k\}$$
$$x \rightarrow \varphi(x)$$

  - Tel que $\forall y \neq x \in X$ si $\{x, y\} \in E$ Alors $\varphi(x) \neq \varphi(y)$.
  - 2 sommets adjacents : 2 couleurs différentes.
  - Tous les sommets doivent être coloriés.
- Une k-coloration partitionne X en k stables où tous les sommets du même stable ont la même couleur.

Coloration des sommets d'un graphe – Nombre chromatique

- <u>Nombre chromatique</u> de $G=(X, E)$:
  - Nombre **min.** de couleurs nécessaires pour colorier les sommets de G.
  - Ce nombre est noté $\chi(G)$.
  - $\Rightarrow \quad 1 \leq \chi(G) \leq n=|X|$.

Coloration des sommets d'un graphe – Problème de coloration

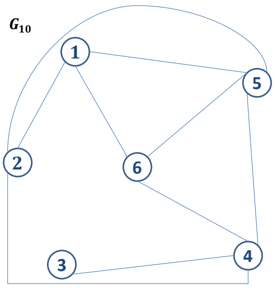
- Réaliser k-coloration d'1 graphe G.
- k doit être le plus proche possible de $\chi(G)$.
- L'<u>algorithme de ***Welsh & Powell***</u> est l'un des plus connus pour résoudre ce problème.

Coloration des sommets d'un graphe – Algorithme de Welsh et Powell

- Ordonner les sommets par de degrés
- Ordre décroissant : du plus grand au plus petit.
- $X = \{x_1, x_2, ..., x_n\}$ tel que $d_G(x_i) \geq d_G(x_{i+1})$
- Pour i de 1 à n :
  - Affecter à x_i la plus petite couleur possible distincte des couleurs de $V(x_i)$ colorés.

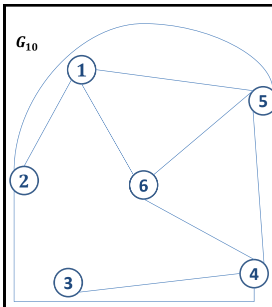
Coloration des sommets d'un graphe – Propositions

- $\forall$ $G=(X, E$: $\chi(G) \leq \Delta(G)+1$.
- $\forall$ $G=(X, E)$ complet K_n où $n \geq 2$ est l'ordre de G : $\chi(G) = \Delta(G)+1=n$.
- $\forall$ $G=(X, E)$ où $C \subseteq X$ est la plus grande clique dans G : $\chi(G) \geq |C|$.



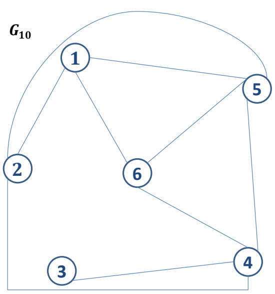
$i \in X$	$d_{G_{10}}(i)$	*Ordre*	φ(i)
1			
2			
3			
4			
5			
6			

Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB)
109



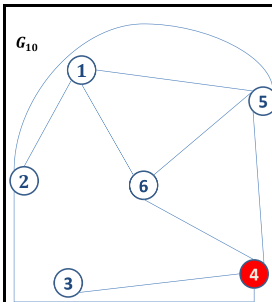
$i \in X$	$d_{G_{10}}(i)$	*Ordre*	φ(i)
1	3		
2	3		
3	1		
4	4		
5	4		
6	3		

Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB)
110



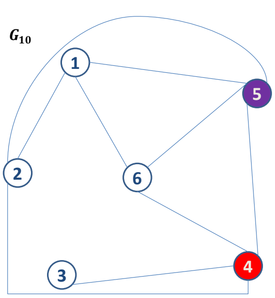
$i \in X$	$d_{G_{10}}(i)$	*Ordre*	φ(i)
1	3	3	
2	3	4	
3	1	6	
4	4	1	
5	4	2	
6	3	5	

Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB)
111



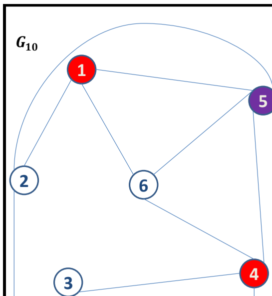
$i \in X$	$d_{G_{10}}(i)$	*Ordre*	φ(i)
1	3	3	
2	3	4	
3	1	6	
4	4	1	1
5	4	2	
6	3	5	

Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB)
112



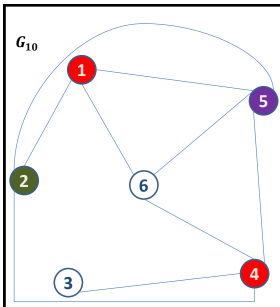
$i \in X$	$d_{G_{10}}(i)$	*Ordre*	φ(i)
1	3	3	
2	3	4	
3	1	6	
4	4	1	1
5	4	2	2
6	3	5	

Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB)
113

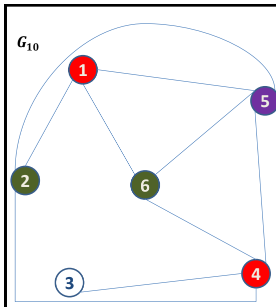


$i \in X$	$d_{G_{10}}(i)$	*Ordre*	φ(i)
1	3	3	1
2	3	4	
3	1	6	
4	4	1	1
5	4	2	2
6	3	5	

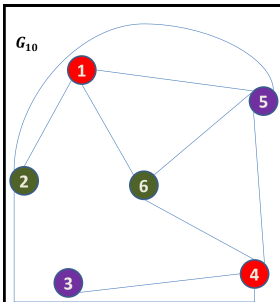
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB)
114



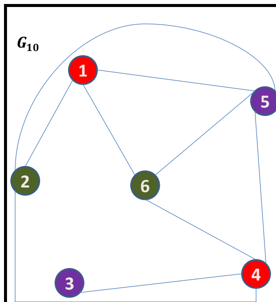
$i \in X$	$d_{G_{10}}(i)$	Ordre	φ(i)
1	3	3	1
2	3	4	3
3	1	6	
4	4	1	1
5	4	2	2
6	3	5	



$i \in X$	$d_{G_{10}}(i)$	Ordre	φ(i)
1	3	3	1
2	3	4	3
3	1	6	
4	4	1	1
5	4	2	2
6	3	5	3



$i \in X$	$d_{G_{10}}(i)$	Ordre	φ(i)
1	3	3	1
2	3	4	3
3	1	6	2
4	4	1	1
5	4	2	2
6	3	5	3



$i \in X$	$d_{G_{10}}(i)$	Ordre	φ(i)
1	3	3	1
2	3	4	3
3	1	6	2
4	4	1	1
5	4	2	2
6	3	5	3

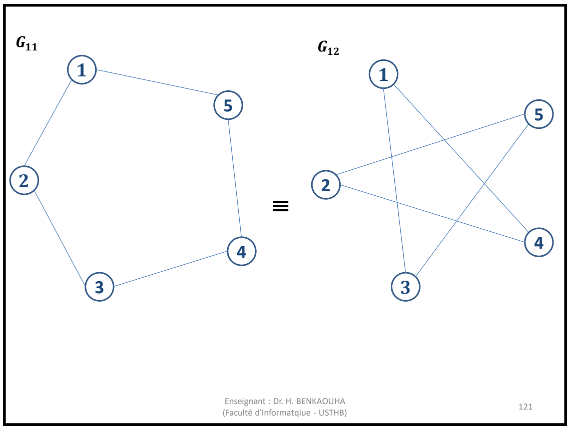
Nous avons obtenu une **3-*coloration***

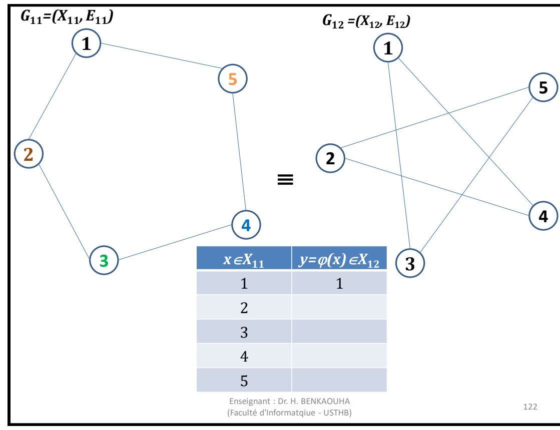
Isomorphisme (≡)

- 2 graphes orientés $G_1 = (X_1, U_1)$ et $G_2 = (X_2, U_2)$.
  - $G_1 \equiv G_2$ **ssi** $\exists f : X_1 \to X_2$ et $\exists g : U_1 \to U_2$ 2 bijections avec $\forall u \in U_1$, $u=(x, y) \Leftrightarrow g(u)=(f(x), f(y))$.
- 2 graphes non orientés $G_1 = (X_1, E_1)$ et $G_2 = (X_2, E_2)$.
  - $G_1 \equiv G_2$ **ssi** $\exists \varphi : X_1 \to X_2$ bijection avec $\forall x, y \in X_1$, $e=\{x, y\} \in E_1 \Rightarrow \{\varphi(x), \varphi(y)\} \in E_2$.

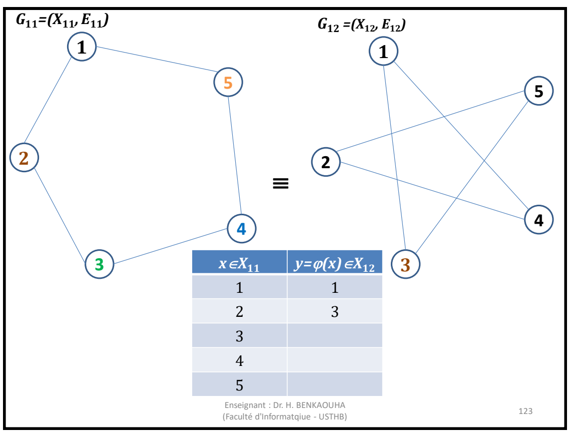
Isomorphisme (≡)

- $G_1 = (X_1, U_1) \equiv G_2 = (X_2, U_2)$ (resp. $G_1 = (X_1, E_1) \equiv G_2 = (X_2, E_2)$)
- Alors $|X_1| = |X_2|$ et $|U_1| = |U_2|$ (resp. $|E_1| = |E_2|$) et $\forall x \in X_1$ de degré $d_G(x)$, $\exists y \in X_2$ de degré $d_G(y) = d_G(x)$.
- La réciproque n'est pas toujours vraie. On peut trouver deux graphes non isomorphes ayant le même nombre de sommets et le même nombre d'arc (ou arêtes).

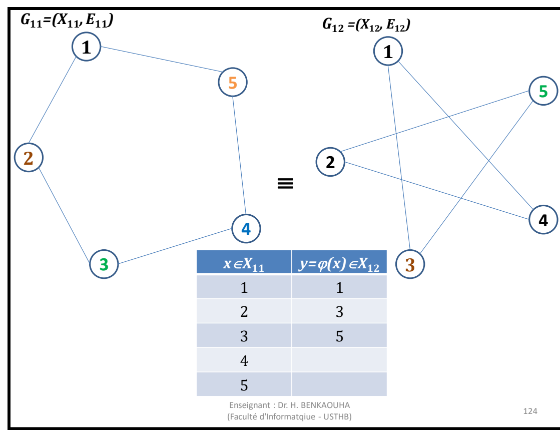




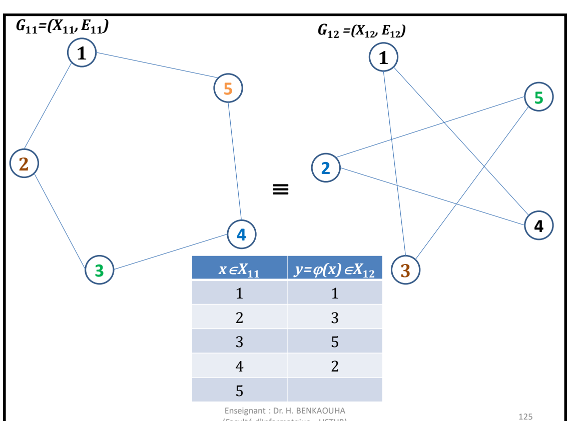
$x \in X_{11}$	$y=\varphi(x) \in X_{12}$
1	1
2	
3	
4	
5	



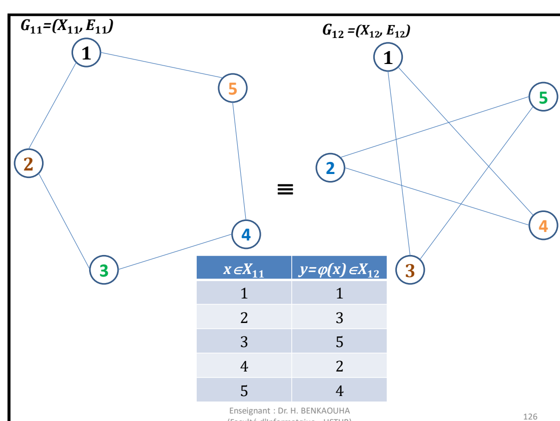
$x \in X_{11}$	$y=\varphi(x) \in X_{12}$
1	1
2	3
3	
4	
5	



$x \in X_{11}$	$y=\varphi(x) \in X_{12}$
1	1
2	3
3	5
4	
5	



$x \in X_{11}$	$y=\varphi(x) \in X_{12}$
1	1
2	3
3	5
4	2
5	



$x \in X_{11}$	$y=\varphi(x) \in X_{12}$
1	1
2	3
3	5
4	2
5	4

Chapitre 2
Cheminement dans les Graphes

Présenté par :
H. BENKAOUHA
Bureau 222, Département Informatique, USTHB
hbenkaouha@usthb.dz
haroun.benkaouha@gmail.com

Chaîne

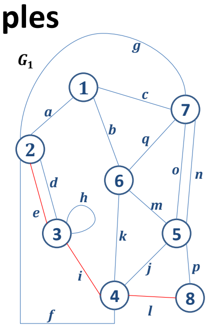
- Chaîne dans un graphe non orienté (resp. orienté) $G=(X, E)$ (resp. $G=(X, U)$):
- $\Rightarrow$Suite alternée de sommets et d'arêtes (resp. d'arcs) :
  - $\mu = x_0\ e_1\ x_1 \ldots x_{k\text{-}1}\ e_k\ x_k$
  - (resp. $\mu = x_0\ u_1\ x_1 \ldots x_{k\text{-}1}\ u_k\ x_k$)
- Tel que pour i de 1 à k,
  - $x_{i\text{-}1}$ et x_i sont extrémités de l'arête e_i (resp. de l'arc u_i).
- On dit que μ est une chaîne joignant les sommets x_0 et x_k de longueur k.

Chaîne - Remarque

- La notion de chaîne est une notion non orientée .
- Mais on peut l'appliquer sur les graphes orientés.
- Il suffit de ne pas prendre en considération le sens des arcs.
- C'est-à-dire, on peut prendre un arc dans le sens inverse.

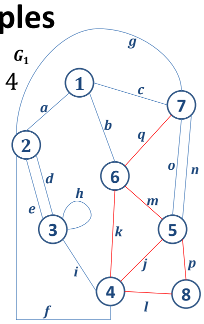
Chaîne - Exemples

- $\mu_1 = 8\ l\ 4\ i\ 3\ e\ 2$
  - Chaîne dans G_1
  - Joignant 8 et 2
  - De longueur 3



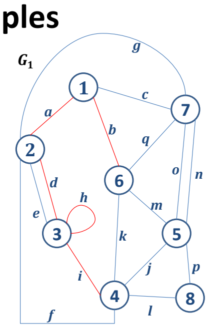
Chaîne - Exemples

- $\mu_2 = 7\ q\ 6\ m\ 5\ j\ 4\ k\ 6\ m\ 5\ p\ 8\ l\ 4$
  - Chaîne dans G_1
  - Joignant 7 et 4
  - De longueur 7



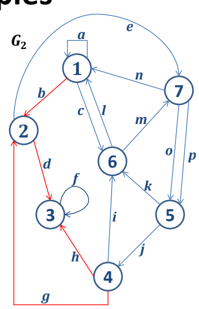
Chaîne - Exemples

- $\mu_3 = 6\ b\ 1\ a\ 2\ d\ 3\ h\ 3\ i\ 4$
  - Chaîne dans G_1
  - Joignant 6 et 4
  - De longueur 5



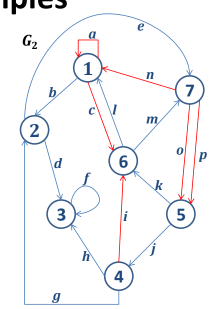
Chaîne - Exemples

- $\mu_4 = 4\ g\ 2\ d\ 3\ h\ 4\ g\ 2\ b\ 1$
  - Chaîne dans G_2
  - Joignant 4 et 1
  - De longueur 5



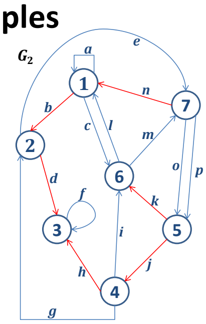
Chaîne - Exemples

- $\mu_5 = 7\ o\ 5\ p\ 7\ n\ 1\ a\ 1\ c\ 6\ i\ 4$
  - Chaîne dans G_2
  - Joignant 7 et 4
  - De longueur 6



Chaîne - Exemples

- $\mu_6 = 6\ k\ 5\ j\ 4\ h\ 3\ d\ 2\ b\ 1\ n\ 7$
  - Chaîne dans G_2
  - Joignant 6 et 7
  - De longueur 6

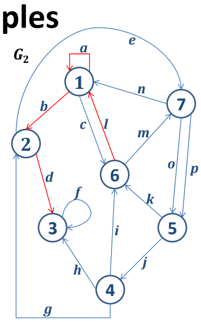


Chemin

- <u>Chemin</u> dans un graphe orienté $G=(X, U)$,
- $\Rightarrow$Suite alternée de sommets et d'arcs :
  - $\gamma = x_0\ u_1\ x_1\ \ldots\ x_{k-1}\ u_k\ x_k$
- Tel que pour i de 1 à k,
  - x_{i-1} est extrémité initiale de l'arc u_i
  - x_i est son extrémité terminale.
- On dit que γ est un chemin de x_0 vers x_k de longueur k.

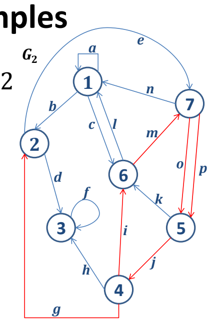
Chemin - Exemples

- $\gamma_1 = 6\ l\ 1\ a\ 1\ b\ 2\ d\ 3$
  - Chemin dans G_2
  - Allant de 6 vers 3
  - De longueur 4



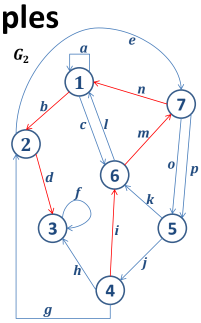
Chemin - Exemples

- $\gamma_2 = 7\ o\ 5\ j\ 4\ i\ 6\ m\ 7\ p\ 5\ j\ 4\ g\ 2$
  - Chemin dans G_2
  - Allant de 7 vers 2
  - De longueur 7



Chemin - Exemples

- $\gamma_3 = 4\ i\ 6\ m\ 7\ n\ 1\ b\ 2\ d\ 3$
  - Chemin dans G_2
  - Allant de 4 vers 3
  - De longueur 5



Propriétés Chaînes/Chemins

- Chaîne / Chemin simple
  - Si tous les arcs ou les arêtes les composant sont distincts.
- Chaîne / Chemin élémentaire
  - Si tous les sommets les composant sont distincts.

Propriétés Chaînes - Exemple

- $\mu_1 = 8\ l\ 4\ i\ 3\ e\ 2$
  - Simple et élémentaire
- $\mu_2 = 7\ q\ 6\ m\ 5\ j\ 4\ k\ 6\ m\ 5\ p\ 8\ l\ 4$
  - Non simple et non élémentaire
- $\mu_3 = 6\ b\ 1\ a\ 2\ d\ 3\ h\ 3\ i\ 4$
  - Simple mais non élémentaire

Propriétés Chaînes - Exemple

- $\mu_4 = 4\ g\ 2\ d\ 3\ h\ 4\ g\ 2\ b\ 1$
  - Non simple et non élémentaire
- $\mu_5 = 7\ o\ 5\ p\ 7\ n\ 1\ a\ 1\ c\ 6\ i\ 4$
  - Simple mais non élémentaire
- $\mu_6 = 6\ k\ 5\ j\ 4\ h\ 3\ d\ 2\ b\ 1\ n\ 7$
  - Simple et élémentaire

Propriétés Chemins - Exemple

- $\gamma_1 = 6\ l\ 1\ a\ 1\ b\ 2\ d\ 3$
  - Simple mais non élémentaire
- $\gamma_2 = 7\ o\ 5\ j\ 4\ i\ 6\ m\ 7\ p\ 5\ j\ 4\ g\ 2$
  - Non simple et non élémentaire
- $\gamma_3 = 4\ i\ 6\ m\ 7\ n\ 1\ b\ 2\ d\ 3$
  - Simple et élémentaire

Remarques (Chaînes/Chemins)

- Longueur d'1 chaîne (chemin) simple = nombre d'arêtes (arcs) formant cette chaîne (chemin).
- Si ∃ chemin d'1 sommet x vers 1 sommet y, on note : $x\ \alpha\ y$.
- Toute chaîne (ou chemin) élémentaire est aussi simple. L'inverse n'est pas toujours vrai.

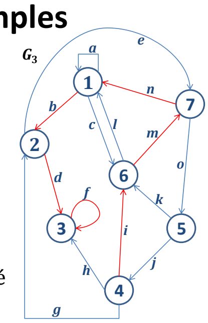
Remarques (Chaînes/Chemins)

- Dans un graphe simple, une chaîne ou un chemin peuvent être déterminés juste en énumérant la suite des sommets qui les composent.
- Un chemin peut être déterminé juste en énumérant la suite des sommets qui le composent si le graphe est un 1-graphe

H. BENKAOUHA 19

Chemin - Exemples

- $\gamma_4 = 4\ i\ 6\ m\ 7\ n\ 1\ b\ 2\ d\ 3\ f\ 3$
- $\gamma_4 = 4\ 6\ 7\ 1\ 2\ 3\ 3$
  - Chemin dans G_3
  - Allant de 4 vers 3
  - De longueur 5
  - Simple, non élémentaire
  - Il n'y a qu'une seule possibilité pour passer d'un sommet à un autre car c'est un 1-graphe



H. BENKAOUHA 20

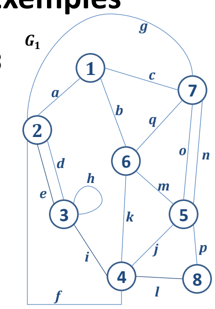
Chaîne fermée / Chemin fermé

- Un chemin dont les extrémités sont confondues est dit <u>chemin fermé</u>.
- Une chaîne dont les extrémités sont confondues est dite <u>chaîne fermée</u>.

H. BENKAOUHA 21

Chaîne fermée - Exemples

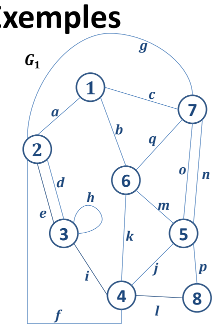
- $\mu_7 = 8\ l\ 4\ i\ 3\ e\ 2\ d\ 3\ i\ 4\ j\ 5\ p\ 8$
  - Chaîne fermée dans G_1
  - De longueur 7
  - Non simple, non élémentaire



H. BENKAOUHA 22

Chaîne fermée - Exemples

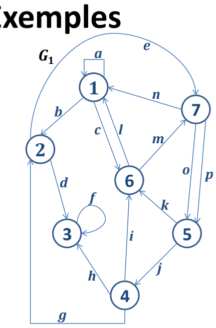
- $\mu_8 = 1\ b\ 6\ m\ 5\ o\ 7\ g\ 2\ a\ 1$
  - Chaîne fermée dans G_1
  - De longueur 5
  - Simple, non élémentaire



H. BENKAOUHA 23

Chemin fermé - Exemples

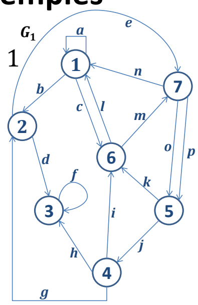
- $\gamma_5 = 7\ n\ 1\ c\ 6\ m\ 7\ n\ 1\ b\ 2\ e\ 7$
  - Chemin fermée dans G_1
  - De longueur 6
  - Non simple, non élémentaire



H. BENKAOUHA 24

Chemin fermé - Exemples

- $\gamma_6 = 1\ a\ 1\ b\ 2\ e\ 7\ o\ 5\ k\ 6\ m\ 7\ n\ 1$
  - Chemin fermée dans G_1
  - De longueur 7
  - Simple, non élémentaire

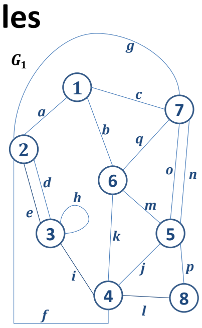


Cycle

- On appelle <u>cycle</u> dans un graphe non orienté (resp. orienté) $G=(X, E)$ (resp. $G=(X, U)$), toute chaîne fermée simple :
- $\mu = x_0\ e_1\ x_1 \ldots x_{k-1}\ e_k\ x_k$ (resp. $\mu = x_0\ u_1\ x_1 \ldots x_{k-1}\ u_k\ x_k$) Tel que $k > 0$, et $x_0 = x_k$.
- On dit que μ est un cycle de longueur k.

Cycle - Exemples

- $\mu_7 = 8\ l\ 4\ i\ 3\ e\ 2\ d\ 3\ i\ 4\ j\ 5\ p\ 8$
  - Chaîne fermée mais pas cycle
- $\mu_8 = 1\ b\ 6\ m\ 5\ o\ 7\ g\ 2\ a\ 1$
  - Cycle dans G_1
  - De longueur 5

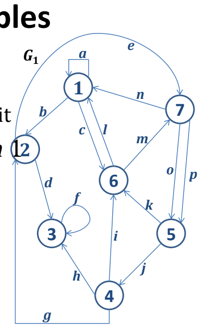


Circuit

- On appelle <u>circuit</u> dans un graphe orienté $G=(X, U)$, tout chemin fermé simple :
- $\gamma = x_0\ u_1\ x_1 \ldots x_{k-1}\ u_k\ x_k$ Tel que $k > 0$, et $x_0 = x_k$.
- On dit que μ est un circuit de longueur k.

Circuit - Exemples

- $\gamma_5 = 7\ n\ 1\ c\ 6\ m\ 7\ n\ 1\ b\ 2\ e\ 7$
  - Chemin fermée mais pas circuit
- $\gamma_6 = 1\ a\ 1\ b\ 2\ e\ 7\ o\ 5\ k\ 6\ m\ 7\ n\ 1$
  - Circuit dans G_1
  - De longueur 7



Cycle / Circuit élémentaire

- On dit qu'un cycle ou circuit est <u>élémentaire</u> si tous les sommets qui les composent sont distincts.
- On ne regarde pas la répétition due à la fermeture.
- Le cycle (resp. circuit) élémentaire est une chaîne (resp. un chemin) fermée non élémentaire.

Remarques (Cycles/Circuits) 1/2

- Longueur cycle ou circuit élémentaire = nombre de sommets formant ce cycle ou circuit.
- Dans un graphe simple, un cycle ou un circuit peuvent être déterminés juste en énumérant la suite des sommets qui les composent.

Remarques (Cycles/Circuits) 2/2

- Une boucle est un cycle élémentaire de longueur 1.
- Une boucle dans un graphe orienté est un circuit élémentaire de longueur 1.
- Tout cycle est aussi chaîne. Tout circuit est aussi chemin. Tout circuit est aussi cycle. Tout chemin est aussi chaîne.

Propositions

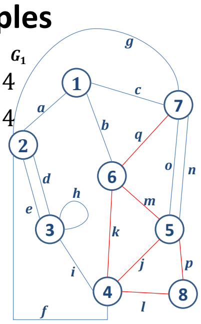
- Soit G=(X, E) un graphe non orienté. De toute chaîne joignant deux sommets x et $y \in X$, on peut extraire 1 chaîne élémentaire joignant x et y.
- Soit G=(X, U) un graphe orienté. De tout chemin allant du sommet $x \in X$ vers le sommet $y \in X$, on peut extraire 1 chemin élémentaire allant de x à y.
- Il suffit de supprimer les cycles (resp. circuits) intermédiaires

Propositions

- Soit G=(X, E) un graphe non orienté (resp. G=(X, U) un graphe orienté). De tout cycle (resp. circuit) passant par 1 arête $e \in E$ (resp. 1 arc $u \in U$), on peut extraire 1 cycle (resp. circuit) élémentaire passant par e (resp. u).
- Il suffit de supprimer les cycles (resp. circuits) intermédiaires qui ne passent pas par e (resp. par u)

Chaîne - Exemples

- $\mu_2 = 7\ q\ 6\ m\ 5\ j\ 4\ k\ 6\ m\ 5\ p\ 8\ l\ 4$
- $\mu_2 = 7\ q\ 6\ m$ $5\ j\ 4\ k\ 6\ m\ 5$ $p\ 8\ l\ 4$
- $\mu_2' = 7\ q\ 6\ m\ 5\ p\ 8\ l\ 4$



Existence d'un cycle

- Si G est un graphe vérifiant $\delta(G) \geq k \geq 2$ Alors G contient un cycle.
- Si de plus G est simple alors G admet un cycle élémentaire de longueur $\geq k$+1 et une chaine élémentaire de longueur $\geq k$.
- **Conséquence :**
  - Si $m \geq n$ (m étant le nombre d'arcs ou arêtes et n le nombre de sommets dans G) alors G admet un cycle.

Existence d'un circuit

- Si G est un graphe vérifiant $\delta^+(G) \geq k \geq 1$ (resp. $\delta^-(G) \geq k \geq 1$) Alors G contient un circuit.
- Si de plus G est simple alors G admet un circuit élémentaire de longueur $\geq k+1$ et un chemin élémentaire de longueur $\geq k$.

Matrice de fermeture transitive

- Soit $G=(X, U)$ un 1-graphe orienté d'ordre n.
- A partir de sa matrice d'adjacence M (doit être booléenne), on peut calculer la matrice de fermeture transitive de G
- Notée $\hat{M}$
- Chaque élément :

$$\hat{m}_{ij} = \begin{cases} 1 \ si \ \exists i \alpha j \\ 0 \ \ \text{sinon} \end{cases}$$

Calcul Matriciel Direct (1/2)

- On peut avoir la matrice de fermeture transitive par calcul matriciel comme suit :

$$\hat{M} = \bigvee_{l=1}^{n} M^{[l]}$$

- Où chaque matrice $M^{[l]}$ se calcule par récurrence (sur l) à travers le produit matriciel booléen comme suit :

$$\begin{cases} M^{[1]} = M \\ M^{[l+1]} = M^{[l]} * M \end{cases}$$

Calcul Matriciel Direct (2/2)

- Chaque élément de $M^{[l]}$:

$$m_{ij}^{[l]} = \bigvee_{k=1}^{n} (m_{ik}^{[l-1]} \wedge m_{kj})$$

  – où l varie de 2 à n.
- La matrice $M^{[l]}$ représente tous les chemins dans G de longueur l.

Algorithme de Warshall (1/2)

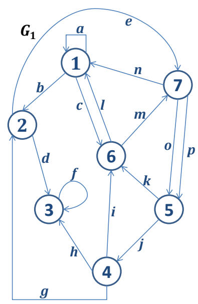
```
Algorithme Warshall
  Début
    Pour j de 1 à n Faire
        Pour i de 1 à n Faire
            Si M[i,j] = 1 Alors
              Pour k de 1 à n Faire
                M[i,k]  = M[i,k] ∨ M[j,k]
              Fait;
            fSi;
        Fait;
    Fait;
  Fin.
```


Algorithme de Warshall (2/2)

- Le calcul direct de $\hat{M}$ nécessite trop d'opérations matricielles.
- L'algorithme de Warshall permet un gain considérable en nombre d'opérations :
  n^2 tests et au plus n^3 opérations $\vee$,
- $\Rightarrow$ algorithme en $O(n^3)$

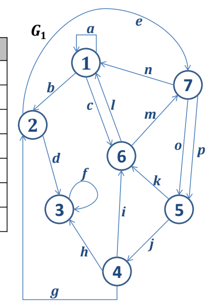
Exemple

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



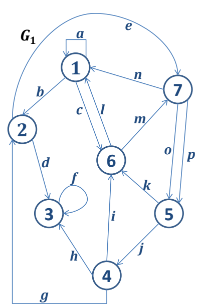
Exemple

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



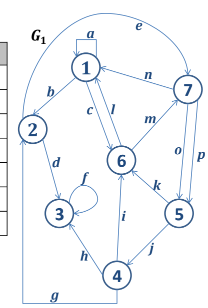
Exemple

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



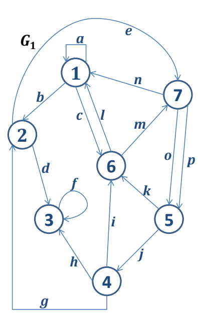
Exemple

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



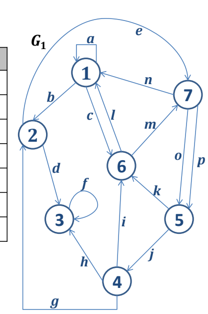
Exemple

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



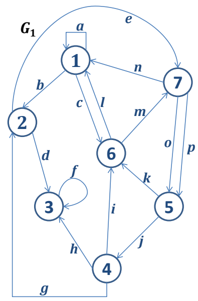
Exemple

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



Exemple

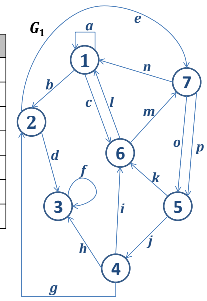
	1	2	3	4	5	6	7
1	1	1	1			1	1
2			1				1
3			1				
4		1	1			1	1
5				1		1	
6	1	1	1			1	1
7	1	1	1		1	1	1



H. BENKAOUHA 49

Exemple

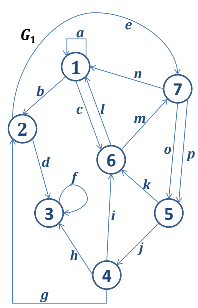
	1	2	3	4	5	6	7
1	1	1	1			1	1
2			1				1
3			1				
4		1	1			1	1
5				1		1	
6	1	1	1			1	1
7	1	1	1		1	1	1



H. BENKAOUHA 50

Exemple

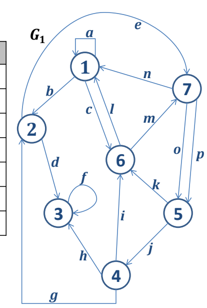
	1	2	3	4	5	6	7
1	1	1	1			1	1
2			1				1
3			1				
4		1	1			1	1
5		1	1	1		1	1
6	1	1	1			1	1
7	1	1	1		1	1	1



H. BENKAOUHA 51

Exemple

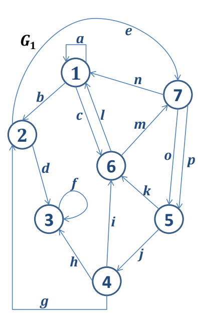
	1	2	3	4	5	6	7
1	1	1	1			1	1
2			1				1
3			1				
4		1	1			1	1
5		1	1	1		1	1
6	1	1	1			1	1
7	1	1	1		1	1	1



H. BENKAOUHA 52

Exemple

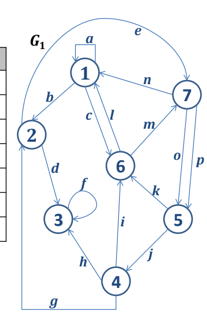
	1	2	3	4	5	6	7
1	1	1	1			1	1
2			1				1
3			1				
4		1	1			1	1
5		1	1	1		1	1
6	1	1	1			1	1
7	1	1	1	1	1	1	1



H. BENKAOUHA 53

Exemple

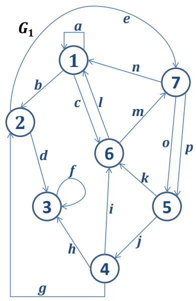
	1	2	3	4	5	6	7
1	1	1	1			1	1
2			1				1
3			1				
4		1	1			1	1
5		1	1	1		1	1
6	1	1	1			1	1
7	1	1	1	1	1	1	1



H. BENKAOUHA 54

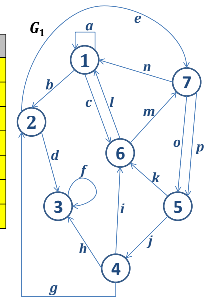
Exemple

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



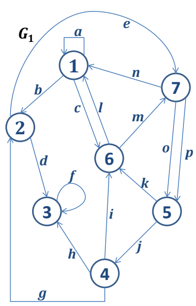
Exemple

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



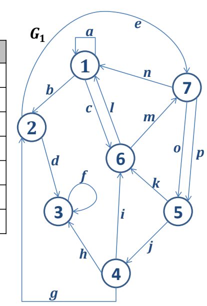
Exemple

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



Exemple

	1	2	3	4	5	6	7
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
3	0	0	1	0	0	0	0
4	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1



Exploration (Parcours) d'un graphe

- L'exploration d'un graphe est un parcours (via les arcs ou les arêtes)
- Permettant d'examiner de façon exhaustive (visiter) les sommets.
- L'exploration d'un graphe permet d'étudier une ou plusieurs propriétés du graphe tel que :
  - la connexité, la forte connexité, biparti, …

Algorithme d'exploration (1/2)

- Principe :
  - Consiste à déterminer l'ordre dans lequel seront visités les sommets.
  - Le parcours commence d'un sommet de départ *r* qu'on appelle *racine*
  - Il donne comme résultat une liste ordonnée de sommets où *r* apparaît en premier et les autres sommets apparaissent une seule fois.

Algorithme d'exploration (2/2)

```
P ← ∅ ;
L ← {r} ;
Tant que ((L ≠ ∅) et (P ≠ X))  Faire
  Choisir_extraire (i∈L) ;
  Pour (tout (i, j)∈U)   Faire
    Si (j∉P) Alors Ajouter j à L ; fSi;
  Fait;
  Pour (tout (j, i)∈U)   Faire
    Si (j∉P) Alors Ajouter j à L ; fSi;
  Fait;
  Ajouter i à la fin de P ;
Fait
```


Remarques : Algo. d'exploration

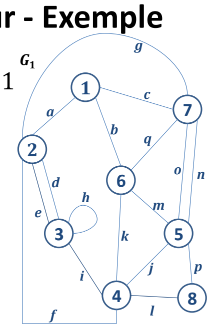
- Nous supposons que la fonction `choisir_extraire` existe et qui consiste à choisir de façon déterministe un sommet de *L* puis le supprime de *L*.
- Nous expliquons après les différentes implémentations de cette fonction.
- Il est possible d'appliquer cet algorithme sur un graphe non-orienté.

Exploration en largeur

- Consiste à parcourir le graphe
  - à partir du sommet de départ la racine (*r*)
  - puis ses voisins
  - puis les voisins des voisins non explorés
  - et ainsi de suite jusqu'à la fin.
- Nous pouvons utiliser l'algorithme d'exploration en déclarant
  - *L* comme une liste FIFO (premier arrivé, premier sorti)
  - La fonction `choisir_extraire` devient `defiler`
  - La fonction `Ajouter` devient `enfiler`.

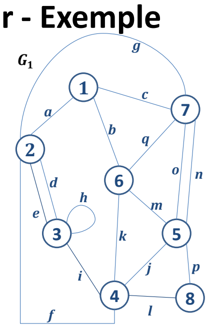
Exploration en largeur - Exemple

- Sommet de départ (racine) : 1
- File : 1
- Parcours :



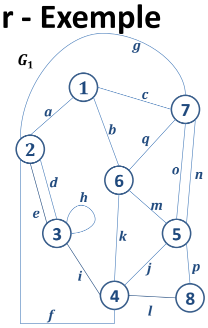
Exploration en largeur - Exemple

- File : 2 6 7
- Parcours : 1



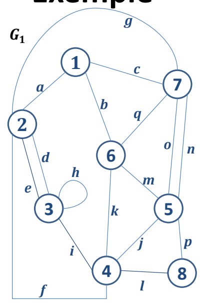
Exploration en largeur - Exemple

- File : 6 7 3 4 7
- Parcours : 1 2



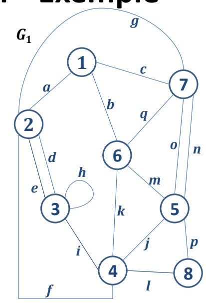
Exploration en largeur - Exemple

- File : 7 3 4 7 4 5 7
- Parcours : 1 2 6



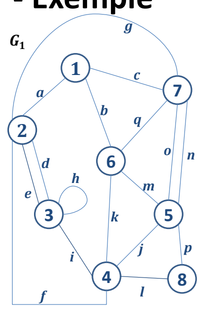
Exploration en largeur - Exemple

- File : 3 4 7 4 5 7 5
- Parcours : 1 2 6 7



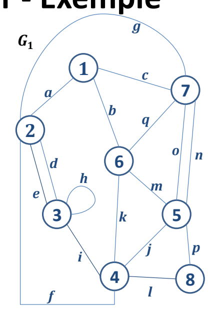
Exploration en largeur - Exemple

- File : 4 7 4 5 7 5 3 4
- Parcours : 1 2 6 7 3



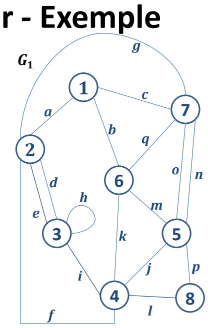
Exploration en largeur - Exemple

- File : 7 4 5 7 5 3 4 5 8
- Parcours : 1 2 6 7 3 4



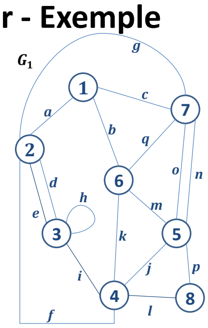
Exploration en largeur - Exemple

- File : 4 5 7 5 3 4 5 8
- Parcours : 1 2 6 7 3 4



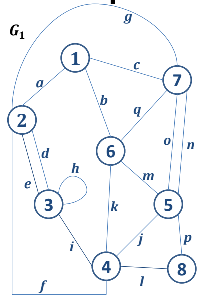
Exploration en largeur - Exemple

- File : 5 7 5 3 4 5 8
- Parcours : 1 2 6 7 3 4



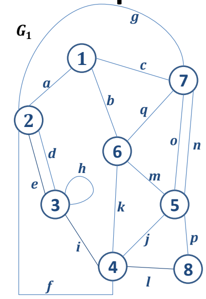
Exploration en largeur - Exemple

- File : 7 5 3 4 5 8 8
- Parcours : 1 2 6 7 3 4 5



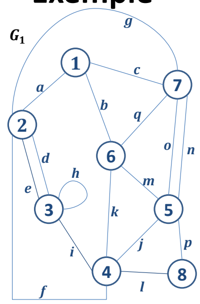
Exploration en largeur - Exemple

- File : 5 3 4 5 8 8
- Parcours : 1 2 6 7 3 4 5



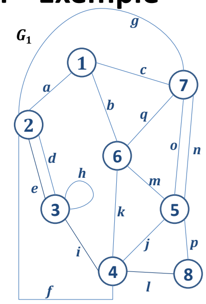
Exploration en largeur - Exemple

- File : 3 4 5 8 8
- Parcours : 1 2 6 7 3 4 5



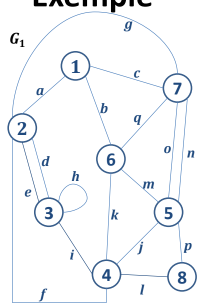
Exploration en largeur - Exemple

- File : 4 5 8 8
- Parcours : 1 2 6 7 3 4 5



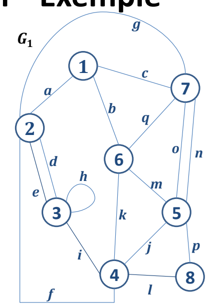
Exploration en largeur - Exemple

- File : 5 8 8
- Parcours : 1 2 6 7 3 4 5



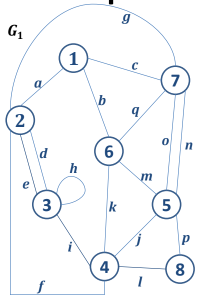
Exploration en largeur - Exemple

- File : 8 8
- Parcours : 1 2 6 7 3 4 5



Exploration en largeur - Exemple

- File : 8
- Parcours : 1 2 6 7 3 4 5 8
- Fin du parcours, tous les sommets sont dans le parcours.

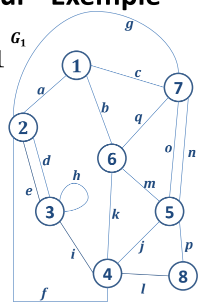


Exploration en profondeur

- Consiste à parcourir le graphe
  - à partir du sommet de départ la racine (r)
  - puis tracer une chaîne à partir de ce sommet
  - puis choisir un autre sommet (parmi les voisins des sommets de cette chaîne dans l'ordre) et faire de même jusqu'à la fin.
- Nous pouvons utiliser l'algorithme d'exploration en déclarant
  - L comme une pile (dernier arrivé, premier sorti)
  - La fonction `choisir_extraire` devient `depiler`
  - La fonction `Ajouter` devient `empiler`.

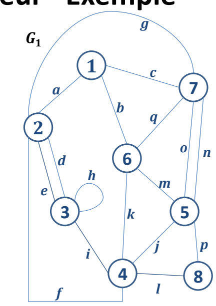
Exploration en profondeur - Exemple

- Sommet de départ (racine) : 1
- Pile : 1
- Parcours :



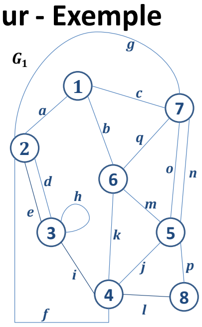
Exploration en profondeur - Exemple

- Pile : 2 6 7
- Parcours : 1



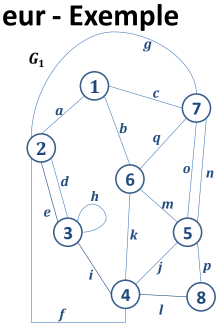
Exploration en profondeur - Exemple

- Pile : 2 6 2 5 6
- Parcours : 1 7



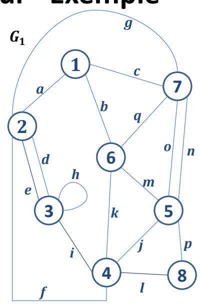
Exploration en profondeur - Exemple

- Pile : 2 6 2 5 4 5
- Parcours : 1 7 6



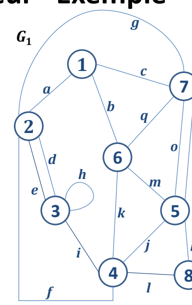
Exploration en profondeur - Exemple

- Pile : 2 6 2 5 4 4 8
- Parcours : 1 7 6 5



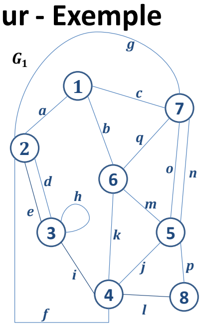
Exploration en profondeur - Exemple

- Pile : 2 6 2 5 4 4 4
- Parcours : 1 7 6 5 8



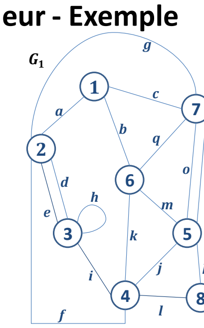
Exploration en profondeur - Exemple

- Pile : 2 6 2 5 4 4 2 3
- Parcours : 1 7 6 5 8 4



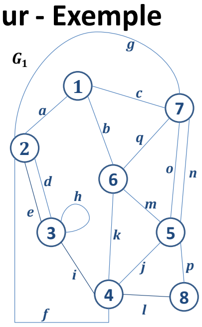
Exploration en profondeur - Exemple

- Pile : 2 6 2 5 4 4 2 2 3
- Parcours : 1 7 6 5 8 4 3



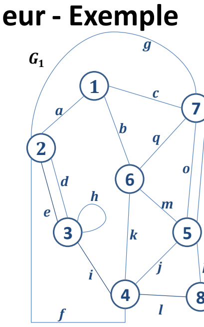
Exploration en profondeur - Exemple

- Pile : 2 6 2 5 4 4 2 2
- Parcours : 1 7 6 5 8 4 3



Exploration en profondeur - Exemple

- Pile : 2 6 2 5 4 4 2
- Parcours : 1 7 6 5 8 4 3 2
- Fin du parcours, tous les sommets sont dans le parcours.



Connexité

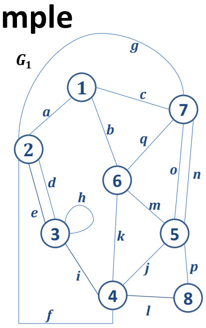
- Un graphe est dit connexe s'il existe
  - une chaîne joignant chaque paire de sommets x et y ($x \neq y$).
- Dessin :
  - On le voit comme une seule entité.

Composante Connexe (CC)

- Soit un graphe $G = (X, E)$ (resp. $G = (X, U)$) :
  - Le sous graphe engendré par un sommet isolé est considéré comme une composante connexe de G.
  - Si le sous graphe engendré par un ensemble de sommets $S \subseteq X$ (G_S) est connexe et tout sous graphe engendré par $S \cup \{x\}$ et $x \notin S$ n'est pas connexe Alors G_S est une composante connexe de G.
- Un graphe connexe contient une seule composante connexe.

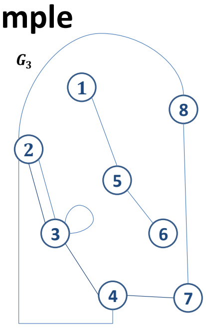
Connexité - Exemple

- Graphe connexe
- 1 CC = X



Connexité - Exemple

- Graphe non connexe
- 2 CC
  - $C_1 = \{1, 5, 6\}$
  - $C_2 = \{2, 3, 4, 7, 8\}$



Algorithme de connexité (1/2)

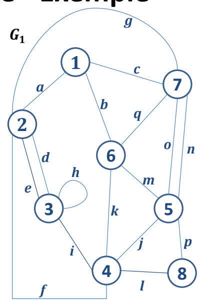
- Nous pouvons utiliser l'algorithme d'exploration afin de vérifier la connexité.
- Il s'agit juste de vérifier si la sortie $P = X$.
- Il existe aussi d'autres algorithmes permettant de vérifier la connexité.
- Le suivant utiliser les marquages.

Algorithme de connexité (2/2)

```
C ← {r} ;
Pour (tout i ∈ X)  Marque[i] ← faux ;
Tant que (∃ i∈C tel que Non(Marque[i]))
  Pour (tout (i, j)∈U)
    C ← C ∪ {j} ;
  Pour (tout (j, i)∈U)
    C ← C ∪ {j} ;
  Marque[i] ← vrai ;
Si C=X Alors Connexe ← Vrai ;
       Sinon Connexe ← Faux ;
```

Algorithme de Connexité - Exemple

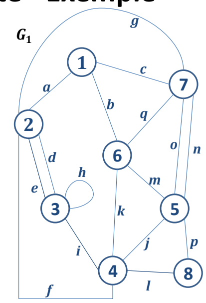
- Graphe connexe
- $C = \{\}$



H. BENKAOUHA 97

Algorithme de Connexité - Exemple

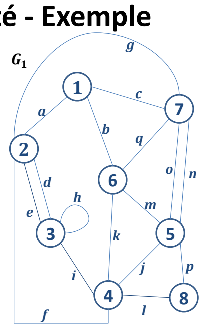
- Ajouter à C un sommet quelconque
- On choisit le 1
- $C = \{1\}$



H. BENKAOUHA 98

Algorithme de Connexité - Exemple

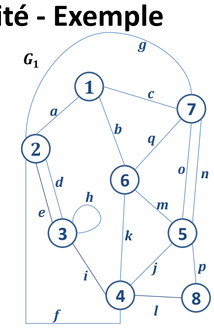
- Ajouter Voisins de 1
- $C = \{1, 2, 6, 7\}$



H. BENKAOUHA 99

Algorithme de Connexité - Exemple

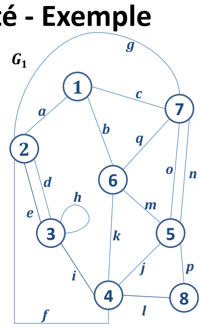
- Ajouter Voisins de 2
- $C = \{1, 2, 6, 7, 3, 4\}$



H. BENKAOUHA 100

Algorithme de Connexité - Exemple

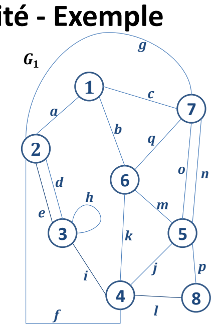
- Ajouter Voisins de 6
- $C = \{1, 2, 6, 7, 3, 4, 5\}$



H. BENKAOUHA 101

Algorithme de Connexité - Exemple

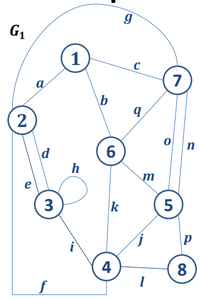
- Ajouter voisins de 4
- $C = \{1, 2, 6, 7, 3, 4, 5, 8\}$



H. BENKAOUHA 102

Algorithme de Connexité - Exemple

- Ajouter voisins de 4
- $C = X$
- Fin de l'Algorithme
- G_1 est connexe

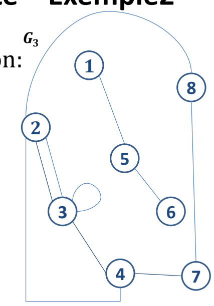


Algorithme de calcul des CC

```
k ← 1;
Tant que X ≠ ϕ
  Faire
    Choisir r ∈ X;
    C[k] ← Connexité (G, r);
// Ici, on considère que l'algorithme précédent ou
// l'algorithme d'exploration comme fonction qui a
// en entrée le graphe et un sommet de départ et
// retourne une CC C ou le parcoures P
    X ← X – C[k];
    k ← k + 1;
  Fait;
```

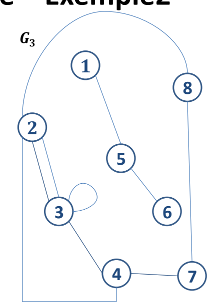

Algorithme de Connexité – Exemple2

- L'exploration comme fonction:
- File : 1
- Parcours :



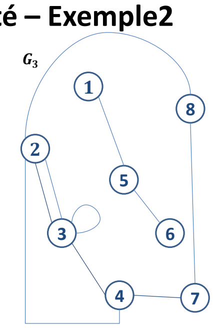
Algorithme de Connexité – Exemple2

- File : 5
- Parcours : 1



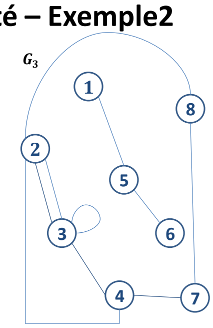
Algorithme de Connexité – Exemple2

- File : 6
- Parcours : 1 5



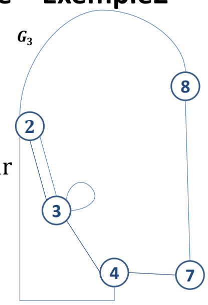
Algorithme de Connexité – Exemple2

- File :
- Parcours : 1 5 6
- Fin, car File vide
- G n'est pas connexe



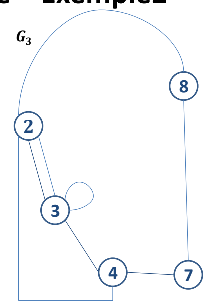
Algorithme de Connexité – Exemple2

- C_1 = { 1, 5, 6 }
- X = {2, 3, 4, 7, 8}
- On refait l'exploration à partir d'un sommet de X.



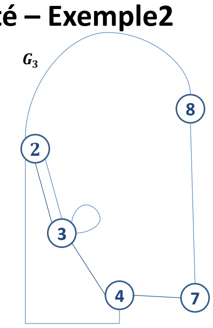
Algorithme de Connexité – Exemple2

- File : 2
- Parcours :



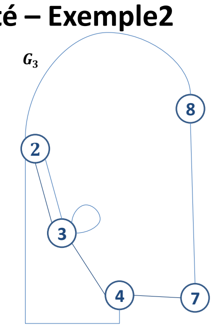
Algorithme de Connexité – Exemple2

- File : 3 4 8
- Parcours : 2



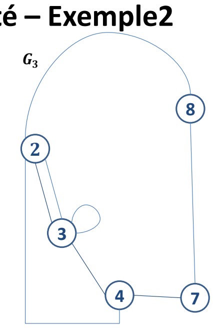
Algorithme de Connexité – Exemple2

- File : 4 8 3 4
- Parcours : 2 3



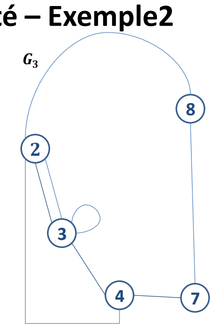
Algorithme de Connexité – Exemple2

- File : 8 3 4 7
- Parcours : 2 3 4



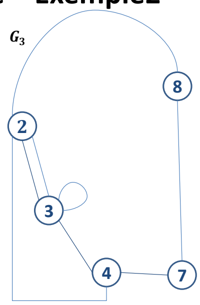
Algorithme de Connexité – Exemple2

- File : 3 4 7 7
- Parcours : 2 3 4 8



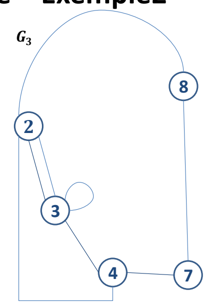
Algorithme de Connexité – Exemple2

- File : 4 7 7
- Parcours : 2 3 4 8



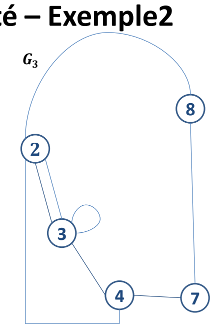
Algorithme de Connexité – Exemple2

- File : 7 7
- Parcours : 2 3 4 8



Algorithme de Connexité – Exemple2

- File : 7
- Parcours : 2 3 4 8 7
- Fin, car P contient tous les sommets restants dans X



Algorithme de Connexité – Exemple2

G_3

- $C_2 = \{2, 3, 4, 7, 8\}$
- $X = \{\}$
- On s'arrête car X est vide.
- On a 2 CC.

Forte Connexité

- Un graphe orienté $G=(X, U)$ est fortement connexe (f.c.) s'il existe entre chaque paire de sommets x et $y \in X$ ($x \neq y$) :
  - **un chemin de x à y ($x \,\alpha\, y$)**

et

  - **un chemin de y à x ($y \,\alpha\, x$)**.

Composante Fortement Connexe (CFC)

- Soit $G=(X, U)$ un graphe orienté :
  - Le sous graphe engendré par un sommet $x \in X$ tel que $d_G^+(x) = 0$ ou $d_G^-(x) = 0$ forme une composante fortement connexe de G.
  - Si le sous graphe engendré par un ensemble de sommets $S \subseteq X$ (G_S) est fortement connexe et le sous graphe engendré par $S \cup \{x\}$ et $x \notin S$ n'est pas fortement connexe Alors G_S est une composante fortement connexe de G.

Ascendants / Descendants

- Soit $G=(X, U)$ un graphe orienté,
- On définit pour chaque sommet $x \in X$,
- 2 ensembles :
  - L'ensemble des descendants de x :
    $D(x) = \{ y \in X \,/\, x \,\alpha\, y \}$
  - L'ensemble des ascendants de x :
    $A(x) = \{ y \in X \,/\, y \,\alpha\, x \}$

Algorithme de calcul des CFCs

```
D ← {r} ;
Pour (tout i ∈ X)   Marque[i] ← faux ;
Tant que ((∃ i ∈ D) et (Marque[i]=faux) )
   Marque[i] ← vrai ;
   Pour (tout (i, j) ∈ U )   D ← D ∪ {j} ;
A ← {r} ;
Pour (tout i ∈ X)   Marque[i] ← faux ;
Tant que ((∃ i ∈ A) et (Marque[i]=faux))
   Marque[i] ← vrai ;
   Pour (tout (j, i) ∈ U)   A ← A ∪ {j}
CFC  ← D ∩ A
```


Algorithme de calcul des CFCs à partir de la matrice de fermeture transitive

Tout sommet i ayant m_{ii}=0
seul dans une CFC
Les autres sommets
Sommets ayant :
lignes identiques
et
colonnes identiques
dans la même CFC

Exemple - CFC

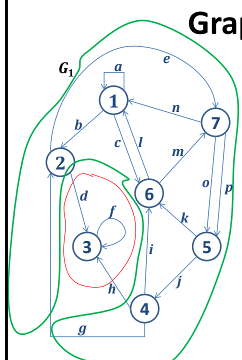
	1	2	3	4	5	6	7
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
3	0	0	1	0	0	0	0
4	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1

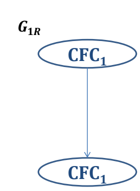
- Lignes identiques :
  L_1 = {1, 2, 4, 5, 6, 7, 8}
  L_2 = {3}
- Colonnes identiques :
  C_1 = {1, 2, 4, 5, 6, 7, 8}
  C_2 = {3}
- Les CFCs:
  CFC_1 = {1, 2, 4, 5, 6, 7, 8}
  CFC_2 = {3}

Graphe réduit

- A tout graphe orienté $G=(X, U)$ on associe le graphe simple $G_R=(X_R, U_R)$ appelé <u>graphe réduit</u> de G défini comme suit :
  - X_R = { A chaque c.f.c. de G correspond un sommet C_i }
  - $U_R = \{ (C_i, C_j) \,/\, i \neq j$ et $\exists\, x \in C_i$ et $\exists\, y \in C_j$ et $(x, y) \in U \}$
- Un graphe fortement connexe possède une seule C.F.C.
- Le graphe réduit d'un graphe ne possède pas de circuits.

Graphe réduit





Parcours Euleriens

- Un parcours Eulerien passe une fois et une seule fois par chaque arête (resp. arc) du graphe.
- Le parcours peut être une chaîne, un chemin, un cycle ou un circuit.
- Soit G un graphe contenant m arêtes (resp. m arcs) :
- Une chaîne simple, un chemin simple, un cycle ou un circuit de longueur m est appelé Eulérien.

H. BENKAOUHA 127

Théorème d'Euler (1766)

- Un multigraphe G admet une <u>chaîne Eulérienne</u>
- Si et seulement si
  - il est connexe (à des sommets isolés près)

  et
  - le nombre de sommets de degré impair est 0 ou 2.

H. BENKAOUHA 128

Théorème d'Euler (1766)

- **Conséquences**
  - Un graphe G admet une chaîne Eulérienne d'un sommet x à un sommet y ($x \neq y$) si et seulement si $d_G(x)$ et $d_G(y)$ sont impairs et $\forall z$ sommet de G ($z \neq x$ et $z \neq y$), on a $d_G(z)$ pair.
  - Un graphe G admet un cycle Eulérien si et seulement $\forall x$ sommet de G, on a $d_G(x)$ pair.

H. BENKAOUHA 129

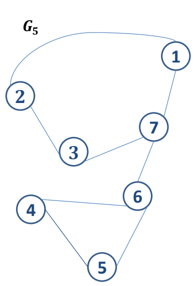
Détermination d'une chaîne Eulerienne

- Choisir sommet a de degré impair (Si pas de sommets de degrés impairs, choisir n'importe quel sommet).
  - On construit une chaîne à partir de a comme suit :
- A chaque étape k
  - On obtient une chaîne de longueur k
  - G_k correspond au graphe partiel de G engendré par l'ensemble des arêtes (resp. d'arcs) initial auquel on supprime ceux faisant partie de la chaîne.
- A chaque étape k, en arrivant à un sommet x,
  - Il faut éviter de prendre toute arête (resp. arc) qui est isthme dans G_k.
  - Sauf s'il s'agit de la seule et unique possibilité, on la prend.
- G_k graphe constitué de sommets isolés $\Rightarrow$ Fin.

H. BENKAOUHA 130

Exemple

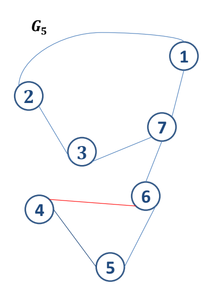
- Sommets de degrés impairs :
  - 6 et 7
- $\mu = 6$



H. BENKAOUHA 131

Exemple

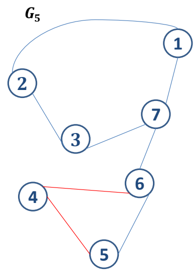
- $\mu = 6$
- On a {6,7} ou {6,4} ou {6,5}
- {6,7} déconnecte le graphe, on la prend pas
- On prend par exemple {6,4}
- $\mu = 6$ 4



H. BENKAOUHA 132

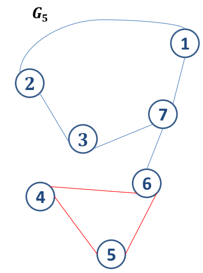
Exemple

- $\mu = 6\ 4$
- On a {4,5}
- On la prend
- $\mu = 6\ 4\ 5$



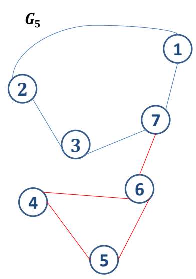
Exemple

- $\mu = 6\ 4\ 5$
- On a {5,6}
- On la prend
- $\mu = 6\ 4\ 5\ 6$



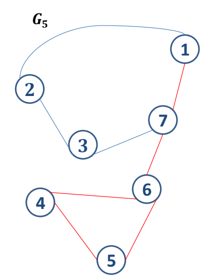
Exemple

- $\mu = 6\ 4\ 5\ 6$
- On a {6,7}
- On la prend
- $\mu = 6\ 4\ 5\ 6\ 7$



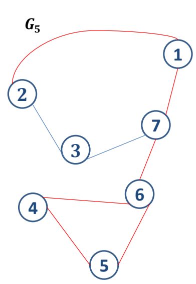
Exemple

- $\mu = 6\ 4\ 5\ 6\ 7$
- On a {7,1} ou {7,3}
- On prend par exemple {7,1}
- $\mu = 6\ 4\ 5\ 6\ 7\ 1$



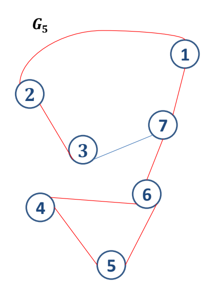
Exemple

- $\mu = 6\ 4\ 5\ 6\ 7\ 1$
- On a {1,2}
- On la prend
- $\mu = 6\ 4\ 5\ 6\ 7\ 1\ 2$



Exemple

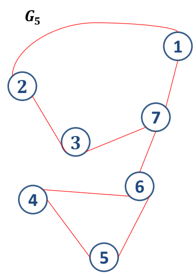
- $\mu = 6\ 4\ 5\ 6\ 7\ 1\ 2$
- On a {2,3}
- On la prend
- $\mu = 6\ 4\ 5\ 6\ 7\ 1\ 2\ 3$



Exemple

- μ = 6 4 5 6 7 1 2 3
- On a {3,7}
- On la prend
- $\mu = 6\ 4\ 5\ 6\ 7\ 1\ 2\ 3\ 7$

Chaîne Eulérienne



Circuit Eulérien

- **Proposition**
  - Un graphe *G=(X, U)* admet un circuit Eulérien Si et seulement si
  - Pour tout sommet *x*, on a $d^+_G(x) = d^-_G(x)$.
  - On dit que *G* est pseudo-symétrique.

Graphe Eulérien / semi-Eulérien

- *G* admet un cycle Eulérien

$\Rightarrow$ *G* est Eulérien.

- *G* admet une chaîne Eulérienne mais pas de cycle Eulérien

$\Rightarrow$ *G* est semi-Eulérien.

Parcours Hamiltonien (1/2)

- Un parcours Hamiltonien passe une fois et une seule fois par chaque sommet du graphe.
- Le parcours peut être une chaîne, un chemin, un cycle ou un circuit.

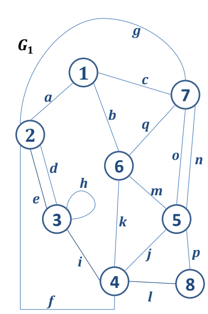
Parcours Hamiltonien (2/2)

- Soit *G* un graphe d'ordre *n* :
- Une chaîne (resp. un chemin) élémentaire de longueur *n*-1 est appelé <u>chaîne Hamiltonienne</u> (resp. <u>chemin Hamiltonien</u>).
- Un cycle (resp. circuit) élémentaire de longueur *n* est appelé <u>cycle</u> (resp. <u>circuit</u>) <u>Hamiltonien</u>.

Exemple

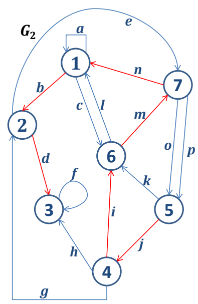
- 1 *a* 2 *e* 3 *i* 4 *k* 6 *q* 7 *o* 5 *p* 8
- Chaîne Hamiltonienne

- 1 *a* 2 *d* 3 *i* 4 *l* 8 *p* 5 *o* 6 *p* 7 *c* 1
- Cycle Hamiltonien



Exemple

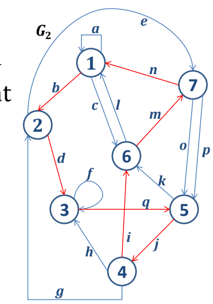
- 5 j 4 i 6 m 7 n 1 b 2 d 3
- Chemin Hamiltonien
- Il n'y a pas de circuit Hamiltonien car G n'est pas fortement connexe.



H. BENKAOUHA 145

Exemple

- Si on rajoute un arc (3, 5) au graphe précédent, on obtient Un circuit Hamiltonien
- 5 j 4 i 6 m 7 n 1 b 2 d 3 q 5



H. BENKAOUHA 146

Graphe Hamiltonien / semi-Hamiltonien

- Un graphe qui contient un cycle Hamiltonien ⇒ graphe Hamiltonien.
- Un graphe semi-Hamiltonien : contient une chaîne Hamiltonienne, mais pas de cycle Hamiltonien.
- Le plus petit graphe Hamiltonien d'ordre n est le graphe cycle (Graphe connexe non-orienté à n arêtes. Il est 2-régulier)

H. BENKAOUHA 147

Graphe Hamiltonien - Propositions

- Un graphe complet d'ordre $n \geq 3$ est Hamiltonien.
- Tout graphe tournoi (un graphe orienté simple et complet) d'ordre n, noté T_n contient un chemin Hamiltonien.
- Tout tournoi d'ordre n (T_n) fortement connexe contient un circuit Hamiltonien.

H. BENKAOUHA 148

Graphe Hamiltonien - Théorème

- Utilisé pour démontrer qu'un graphe n'est pas Hamiltonien (ne contient pas de cycle Hamiltonien).
- Si $G=(X,E)$ est un graphe Hamiltonien, alors pour tout ensemble de sommets $S \subset X$, on a :
  – $p(G_{X\text{-}S}) \leq |S|$
  – où $p(G_{X\text{-}S})$ est le nombre de composantes connexes du sous graphe de G induit par l'ensemble $X\text{-}S$

H. BENKAOUHA 149

Chapitre 3
Arbre de couverture optimal

Présenté par :
H. BENKAOUHA
Bureau 222, Faculté d'Informatique, USTHB
haroun.benkaouha@usthb.edu.dz
haroun.benkaouha@gmail.com

Définition d'un arbre

- Soit $G=(X,U)$ un graphe orienté d'ordre $n \geq 2$. G est un **arbre** si l'une des **six (6) propriétés** suivantes est vérifiée :

1. G est connexe et sans cycles.
2. G est sans cycles et admet $n - 1$ arcs ou arêtes.
3. G est connexe et admet $n - 1$ arcs ou arêtes.
4. G est sans cycle maximal (tout arc ou arête supplémentaire créé un cycle dans G).

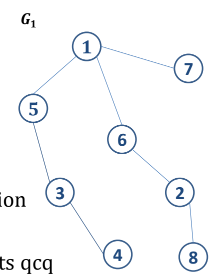
Définition d'un arbre

5. G est connexe minimal (la suppression d'un arc ou arête quelconque le rend non connexe).
6. Pour toute paire de sommets ($x, y \in X, x \neq y$) Il existe dans G une chaîne et une seule joignant x à y.

- Les 6 caractéristiques (propriétés) ci-dessus (propres aux arbres) sont équivalentes.

Arbre - Exemple

- Un arbre de :
  - 8 sommets
  - 7 arêtes
  - Connexe
  - Pas de cycles
  - Supprimer 1 arête : déconnexion
  - Rajouter 1 arête : cycle
  - 1 seule chaîne entre 2 sommets qcq

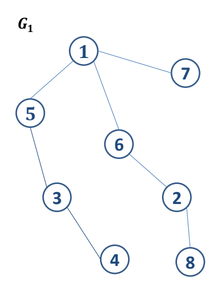


Codage de Prufer – algorithme de codage

```
P ← vide;
Tant que |X|>2
 Faire
  Choisir x dans X tel que dG(x)=1 et x
  minimal; // feuille de numéro minimal
  P ← P . Adjacent(x);
  // Rajouter le sommet adjacent à x dans la liste ordonnée P
  X ← X - {x}; // Supprimer le sommet x
  E ← E - {x, Adjacent(x)};
  // Supprimer l'arête incidente à x
 Fait
```

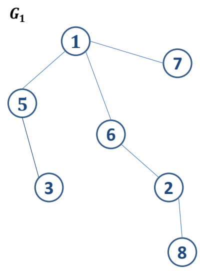

Prufer : Exemple Algorithme de codage

- Sommets de degré 1 :
  - 4 7 8
  - Min : 4
  - Voisin de 4 : 3
  - Rajouter 3 dans P
  - P : 3



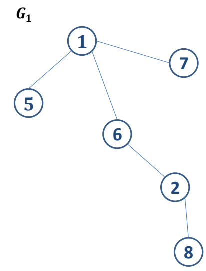
Prufer : Exemple Algorithme de codage

- Supprimer
  - le sommet 4
  - l'arête incidente à 4
- Sommets de degré 1 :
  - 3 7 8
  - Min : 3
  - Voisin de 3 : 5
  - Rajouter 5 dans P
  - P : 3 5



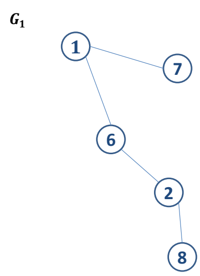
Prufer : Exemple Algorithme de codage

- Supprimer
  - le sommet 3
  - l'arête incidente à 3
- Sommets de degré 1 :
  - 5 7 8
  - Min : 5
  - Voisin de 5 : 1
  - Rajouter 1 dans P
  - P : 3 5 1



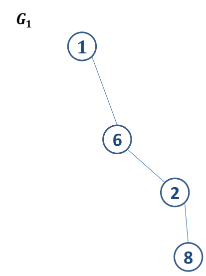
Prufer : Exemple Algorithme de codage

- Supprimer
  - le sommet 5
  - l'arête incidente à 5
- Sommets de degré 1 :
  - 7 8
  - Min : 7
  - Voisin de 7 : 1
  - Rajouter 1 dans P
  - P : 3 5 1 1



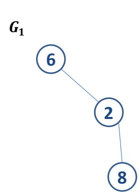
Prufer : Exemple Algorithme de codage

- Supprimer
  - le sommet 7
  - l'arête incidente à 7
- Sommets de degré 1 :
  - 1 8
  - Min : 1
  - Voisin de 1 : 6
  - Rajouter 6 dans P
  - P : 3 5 1 1 6



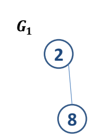
Prufer : Exemple Algorithme de codage

- Supprimer
  - le sommet 1
  - l'arête incidente à 1
- Sommets de degré 1 :
  - 6 8
  - Min : 6
  - Voisin de 6 : 2
  - Rajouter 2 dans P
  - P : 3 5 1 1 6 2



Prufer : Exemple Algorithme de codage

- Supprimer
  - le sommet 6
  - l'arête incidente à 6
- Il ne reste que 2 sommets
  - Fin du codage
  - P= 3 5 1 1 6 2



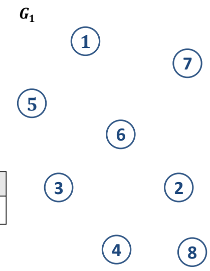
Codage de Prufer – algorithme de décodage

```
n ← longueur(P)+ 2;
X ← ∅;   E ← ∅;
Pour i de 1 à n
 X ← X ∪ {i};   D[i] ← 1;
Pour chaque valeur j de P
 D[i] ← D[i] + 1;
Pour chaque valeur j de P
  Chercher k tel que D[k]=1 et k minimal;
  E ← E ∪ {j,k};
  D[j] ← D[j]-1;   D[k] ← 0;
Relier les deux sommets ayant D[i]=1;
```


Prufer : Exemple Algorithme de décodage

- P= 3 5 1 1 6 2
- Longueur de P =6
- On prépare un graphe de 6+2=8 sommets isolés

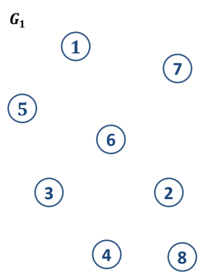
1	2	3	4	5	6	7	8
1	1	1	1	1	1	1	1



Prufer : Exemple Algorithme de décodage

- P= 3 5 1 1 6 2
- Mettre à jour le vecteur D selon P

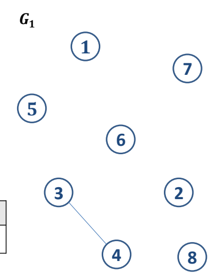
1	2	3	4	5	6	7	8
3	2	2	1	2	2	1	1



Prufer : Exemple Algorithme de décodage

- P= 3 5 1 1 6 2
  - Élément de P : 3
- D[i]=1 et i minimal
  - i=4
- Créer {3, 4}

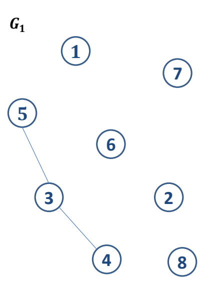
1	2	3	4	5	6	7	8
3	2	1	0	2	2	1	1



Prufer : Exemple Algorithme de décodage

- P= 3 5 1 1 6 2
  - Élément de P : 5
- D[i]=1 et i minimal
  - i=3
- Créer {3, 5}

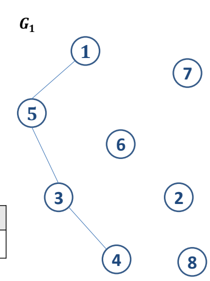
1	2	3	4	5	6	7	8
3	2	0	0	1	2	1	1



Prufer : Exemple Algorithme de décodage

- P= 3 5 1 1 6 2
  - Élément de P : 1
- D[i]=1 et i minimal
  - i=5
- Créer {1, 5}

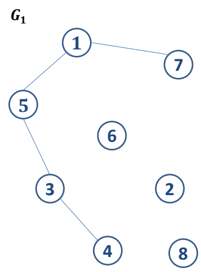
1	2	3	4	5	6	7	8
2	2	0	0	0	2	1	1



Prufer : Exemple Algorithme de décodage

- P= 3 5 1 1 6 2
  - Élément de P : 1
- D[i]=1 et i minimal
  - i=7
- Créer {1, 7}

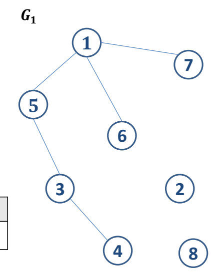
1	2	3	4	5	6	7	8
1	2	0	0	0	2	0	1



Prufer : Exemple Algorithme de décodage

- P= 3 5 1 1 6 2
  - Élément de P : 6
- D[i]=1 et i minimal
  - i=1
- Créer {1, 6}

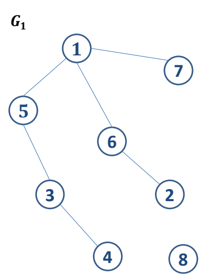
1	2	3	4	5	6	7	8
0	2	0	0	0	1	0	1



Prufer : Exemple Algorithme de décodage

- P= 3 5 1 1 6 2
  - Élément de P : 2
- D[i]=1 et i minimal
  - i=6
- Créer {2, 6}

1	2	3	4	5	6	7	8
0	1	0	0	0	0	0	1

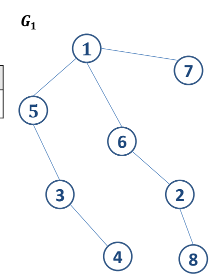


Prufer : Exemple Algorithme de décodage

- P= 3 5 1 1 6 2

1	2	3	4	5	6	7	8
0	1	0	0	0	0	0	1

- Il n'y a que 2 sommets ayant D[i]=1 : 2 et 8
- Créer {2, 8}
- Fin

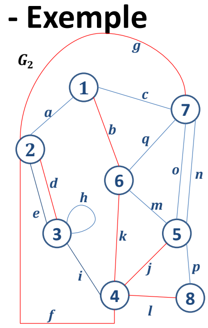


Arbre de couverture d'un graphe

- Soit $G=(X,U)$ un graphe orienté d'ordre $n \geq 2$.
- On appelle <u>arbre dans *G*</u> un sous-graphe partiel de *G*, <u>$H=(Y, V)$ connexe et sans cycles</u>.
- Un arbre est maximal dans *G* s'il contient le maximum possible de sommets de *G* (c'est-à-dire $Y=X$ si *G* est connexe).

Arbre de couverture - Exemple

- En rouge, un arbre de couverture de G_2
- Cet arbre est maximal
- Il y a tous les sommets de *G*



Graphe pondéré (valué)

- **Poids d'un arc**
  - Soit *G=(X,U)* un graphe orienté,
  - On définit $p : U \rightarrow \Re$ une application
  - associe pour chaque arc $u \in U$ de *G* une valeur réelle *p(u)*
  - appelée poids de l'arc *u*.
- Un tel graphe est appelé:
  - graphe pondéré, graphe valué ou réseau.
  - Noté *G=(X,U, p)*

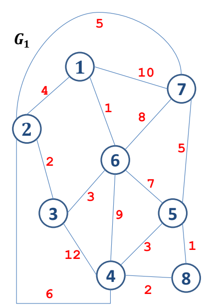
Identification du problème

- Soit *G=(X,U)* un graphe orienté connexe muni d'une application poids *p*.
- La recherche dans *G* d'un arbre de poids optimal revient à :
  - trouver un graphe partiel *G'=(X,U')* de *G* qui soit un arbre maximal
  - et pour lequel la somme des poids des arcs de *G'* soit optimale (maximale ou minimale selon la situation).

Algorithme de Kruksal

```
V = tri_poids_croissant(U,p);
H ← {V[1]};        i ← 1 ;        j ← 1;
Tant que (j < n-1)
 Faire
  i ← i+1; u ← V[i];
  Si (H ∪ {u} ne contient pas de cycle)
     Alors H ← H ∪ {u}; j ← j+1;
     // L'arc sélectionné ne doit pas créer de cycle
  fSi
 Fait
```


Algorithme de Kruksal - Exemple

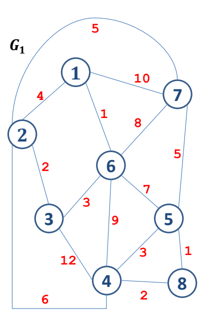


Algorithme de Kruksal - Exemple

- Classer les arêtes dans un ordre croissant des poids.

Arête	Poids
{1,6}	1
{5,8}	1
{2,3}	2
{4,8}	2
{3,6}	3
{4,5}	3
{1,2}	4
{2,7}	5

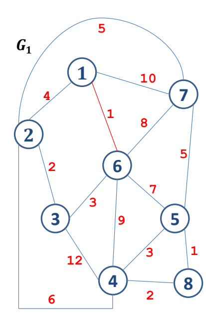
Arête	Poids
{5,7}	5
{2,4}	6
{5,6}	7
{6,7}	8
{4,6}	9
{1,7}	10
{3,4}	12



Algorithme de Kruksal - Exemple

Arête	Poids
{1,6}	1
{5,8}	1
{2,3}	2
{4,8}	2
{3,6}	3
{4,5}	3
{1,2}	4
{2,7}	5

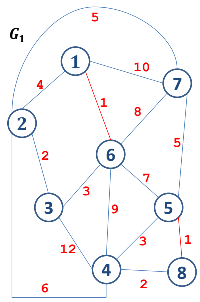
Arête	Poids
{5,7}	5
{2,4}	6
{5,6}	7
{6,7}	8
{4,6}	9
{1,7}	10
{3,4}	12



Algorithme de Kruksal - Exemple

Arête	Poids
{1,6}	1
{5,8}	1
{2,3}	2
{4,8}	2
{3,6}	3
{4,5}	3
{1,2}	4
{2,7}	5

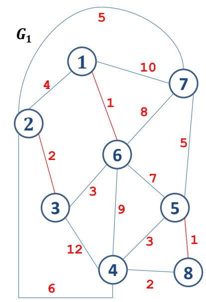
Arête	Poids
{5,7}	5
{2,4}	6
{5,6}	7
{6,7}	8
{4,6}	9
{1,7}	10
{3,4}	12



Algorithme de Kruksal - Exemple

Arête	Poids
{1,6}	1
{5,8}	1
{2,3}	2
{4,8}	2
{3,6}	3
{4,5}	3
{1,2}	4
{2,7}	5

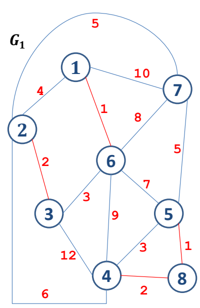
Arête	Poids
{5,7}	5
{2,4}	6
{5,6}	7
{6,7}	8
{4,6}	9
{1,7}	10
{3,4}	12



Algorithme de Kruksal - Exemple

Arête	Poids
{1,6}	1
{5,8}	1
{2,3}	2
{4,8}	2
{3,6}	3
{4,5}	3
{1,2}	4
{2,7}	5

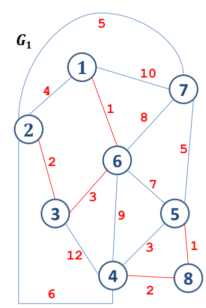
Arête	Poids
{5,7}	5
{2,4}	6
{5,6}	7
{6,7}	8
{4,6}	9
{1,7}	10
{3,4}	12



Algorithme de Kruksal - Exemple

Arête	Poids
{1,6}	1
{5,8}	1
{2,3}	2
{4,8}	2
{3,6}	3
{4,5}	3
{1,2}	4
{2,7}	5

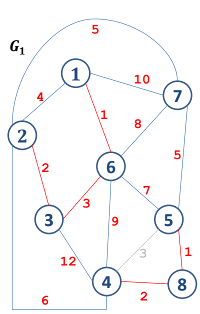
Arête	Poids
{5,7}	5
{2,4}	6
{5,6}	7
{6,7}	8
{4,6}	9
{1,7}	10
{3,4}	12



Algorithme de Kruksal - Exemple

Arête	Poids
{1,6}	1
{5,8}	1
{2,3}	2
{4,8}	2
{3,6}	3
{4,5}	3
{1,2}	4
{2,7}	5

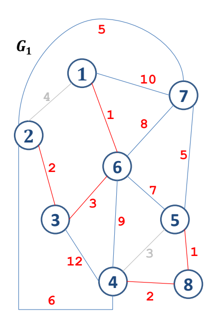
Arête	Poids
{5,7}	5
{2,4}	6
{5,6}	7
{6,7}	8
{4,6}	9
{1,7}	10
{3,4}	12



Algorithme de Kruksal - Exemple

Arête	Poids
{1,6}	1
{5,8}	1
{2,3}	2
{4,8}	2
{3,6}	3
{4,5}	3
{1,2}	4
{2,7}	5

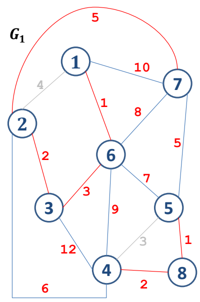
Arête	Poids
{5,7}	5
{2,4}	6
{5,6}	7
{6,7}	8
{4,6}	9
{1,7}	10
{3,4}	12



Algorithme de Kruksal - Exemple

Arête	Poids
{1,6}	1
{5,8}	1
{2,3}	2
{4,8}	2
{3,6}	3
{4,5}	3
{1,2}	4
{2,7}	5

Arête	Poids
{5,7}	5
{2,4}	6
{5,6}	7
{6,7}	8
{4,6}	9
{1,7}	10
{3,4}	12

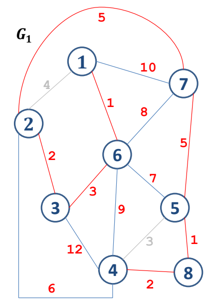


Algorithme de Kruksal - Exemple

- 7 arêtes sélectionnées pour 8 sommets : Fin Algo.

Arête	Poids
{1,6}	1
{5,8}	1
{2,3}	2
{4,8}	2
{3,6}	3
{4,5}	3
{1,2}	4
{2,7}	5

Arête	Poids
{5,7}	5
{2,4}	6
{5,6}	7
{6,7}	8
{4,6}	9
{1,7}	10
{3,4}	12

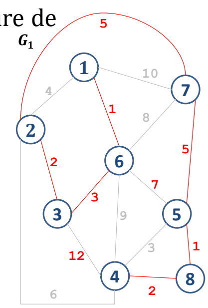


Algorithme de Kruksal - Exemple

- En rouge, l'arbre de couverture de poids minimal : Poids = 19.

Arête	Poids
{1,6}	1
{5,8}	1
{2,3}	2
{4,8}	2
{3,6}	3
{4,5}	3
{1,2}	4
{2,7}	5

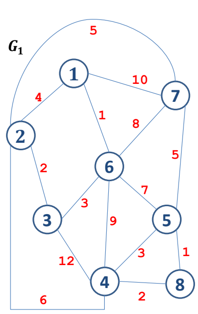
Arête	Poids
{5,7}	5
{2,4}	6
{5,6}	7
{6,7}	8
{4,6}	9
{1,7}	10
{3,4}	12



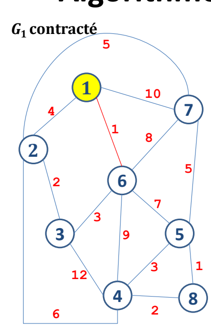
Algorithme de Prim

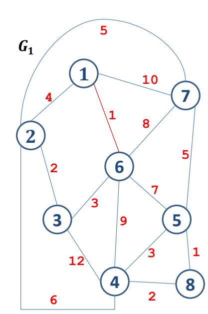
```
G ← R ;          H ← ∅;
Tant que |X|≥2
 Faire
  Choisir x dans X;
  u ← u(x);
  // choisir l'arc u de poids min ayant x comme
  extrémité
  H ← H ∪ {u};
  G ← Gu ; // Contracter G par rapport à l'arc u
Fait
```

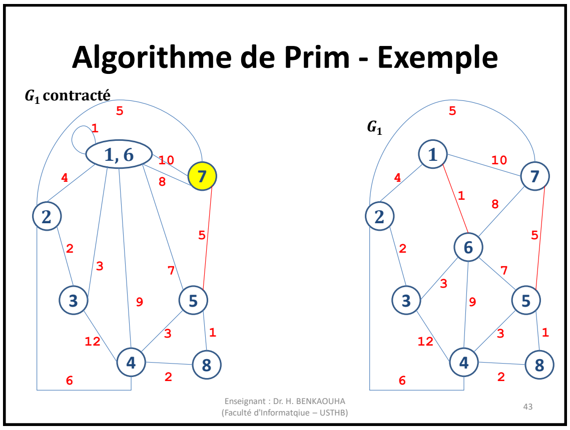

Algorithme de Prim - Exemple

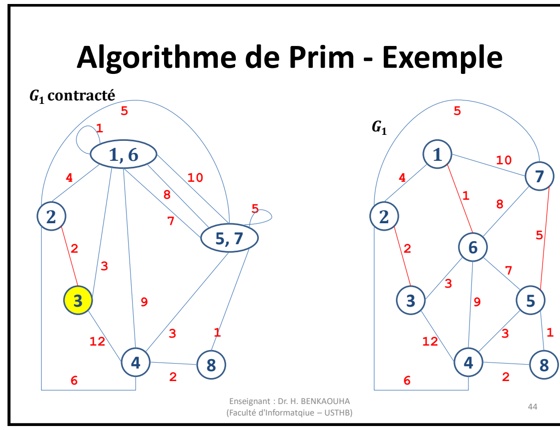


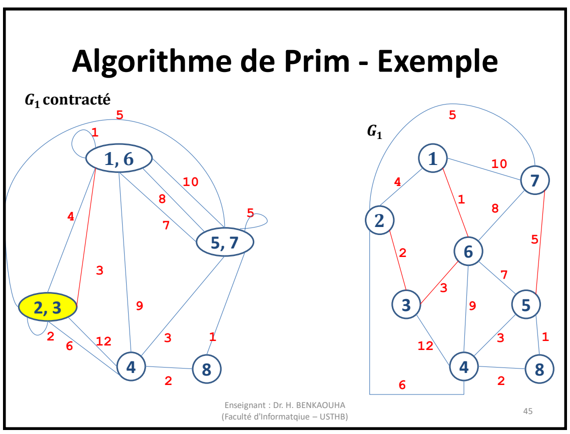
Algorithme de Prim - Exemple

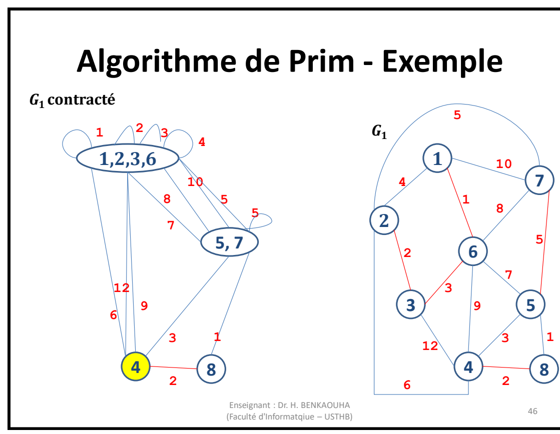


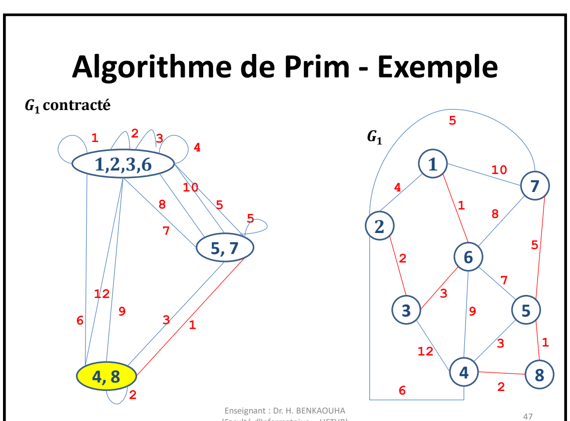


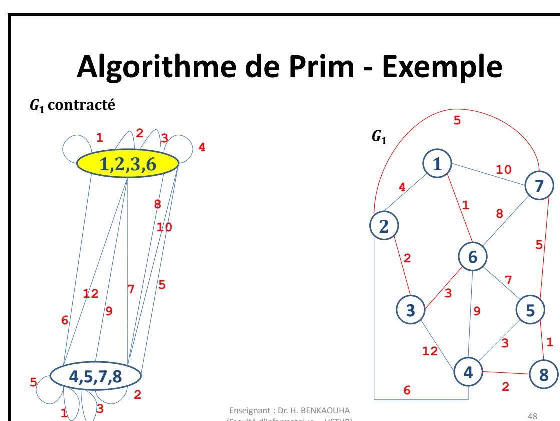

Algorithme de Prim - Exemple
G_1 contracté
G_1
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue – USTHB)
43

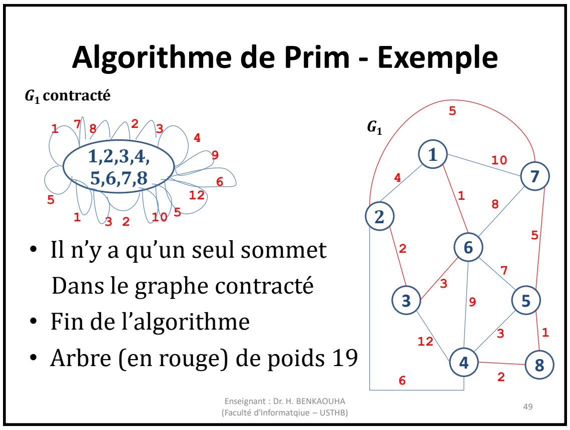

Algorithme de Prim - Exemple
G_1 contracté
G_1
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue – USTHB)
44


Algorithme de Prim - Exemple
G_1 contracté
G_1
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue – USTHB)
45


Algorithme de Prim - Exemple
G_1 contracté
G_1
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue – USTHB)
46


Algorithme de Prim - Exemple
G_1 contracté
G_1
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue – USTHB)
47


Algorithme de Prim - Exemple
G_1 contracté
G_1
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue – USTHB)
48


Algorithme de Prim - Exemple
G_1 contracté
1,2,3,4,
5,6,7,8
G_1
• Il n'y a qu'un seul sommet
Dans le graphe contracté
• Fin de l'algorithme
• Arbre (en rouge) de poids 19
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue – USTHB)
49

Chapitre 4
Problèmes de cheminement dans les graphes

Présenté par :
H. BENKAOUHA
Bureau 222, Faculté d'Informatique, USTHB
haroun.benkaouha@gmail.com
haroun.benkaouha@usthb.edu.dz

Graphe sans circuits (S.C.)

- Un graphe orienté G=(X, U) est dit S.C.
- Ssi toute composante fortement connexe est réduite à un sommet.
- Ssi il est isomorphe à son graphe réduit.
- Ssi tout chemin dans G est élémentaire.

Source et Puits

- Un sommet s est appelé <u>source</u> Ssi $d^-(s)=0$.
- Un sommet p est appelé <u>puits</u> Ssi $d^+(p)=0$.
- Tt graphe S.C. possède une source et un puits.
- Dans un graphe S.C. $G=(X, U)$ $\forall x \in X$,
  - L'<u>extrémité initiale</u> ($\boldsymbol{s} \in X$) du <u>plus long chemin</u> vers x est une <u>source</u>.
  - L'<u>extrémité terminale</u> ($\boldsymbol{p} \in X$) d'<u>un plus long chemin</u> commençant en x est un <u>puits</u>.

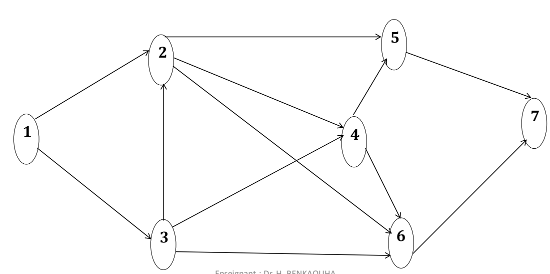
Niveau d'un sommet

- Soit $G=(X, U)$, un graphe orienté sans circuits. A tout $x \in X$, on associe un entier :
  - $v(x)$: **niveau** de x = la longueur max. d'un chemin élémentaire se terminant à x.
- On affecte par convention à une source s la valeur $v(s)$=0.
- On note par $\lambda(G)$ le plus grand niveau dans G.
- $\lambda(G)$ correspond au niveau d'un puits.

Partition en niveaux

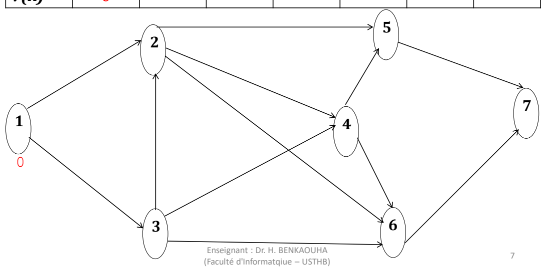
- Soit $G=(X,U)$ est un graphe S.C.
- L'ensemble des sommets X peut être partitionné au maximum en $\lambda(G)$+1 stables. Où
  - chaque sommet de niveau i sera placé dans le stable N_i.
  - Chaque stable N_i représente un niveau de G.
- $G=(X,U)$ est S.C. <u>Ssi</u>
  - X admet une partition $\{N_0 \cup N_1 \cup \ldots\ldots \cup N_p\}$ / $x \in N_i \Leftrightarrow v(x)=i$

Partitionnement en niveaux - Exemple



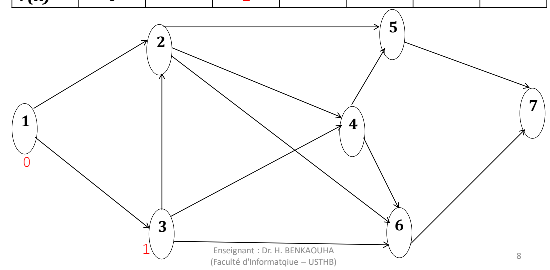
Partitionnement en niveaux - Exemple

x	1	2	3	4	5	6	7
v(x)	0						



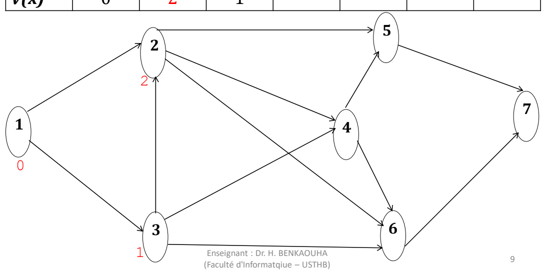
Partitionnement en niveaux - Exemple

x	1	2	3	4	5	6	7
v(x)	0		1				



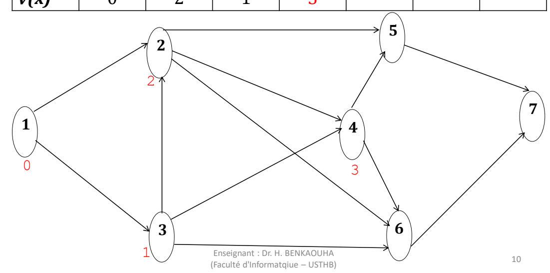
Partitionnement en niveaux - Exemple

x	1	2	3	4	5	6	7
v(x)	0	2	1				



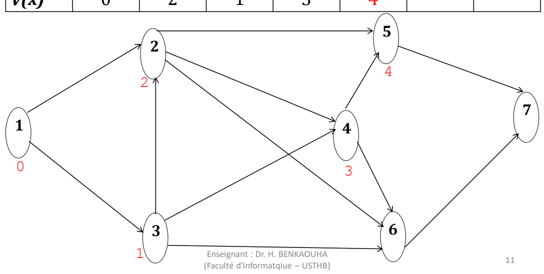
Partitionnement en niveaux - Exemple

x	1	2	3	4	5	6	7
v(x)	0	2	1	3			



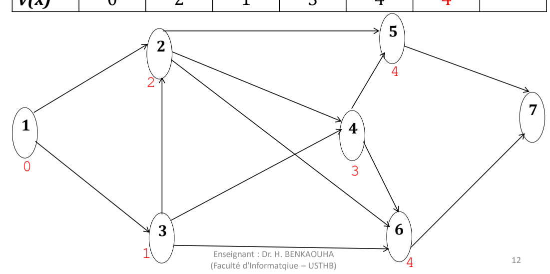
Partitionnement en niveaux - Exemple

x	1	2	3	4	5	6	7
v(x)	0	2	1	3	4		



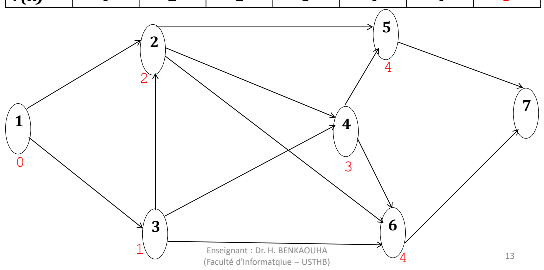
Partitionnement en niveaux - Exemple

x	1	2	3	4	5	6	7
v(x)	0	2	1	3	4	4	



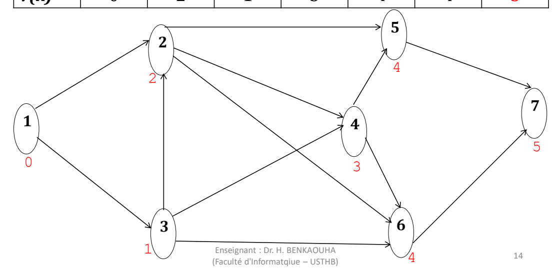
Partitionnement en niveaux - Exemple

x	1	2	3	4	5	6	7
$v(x)$	0	2	1	3	4	4	5



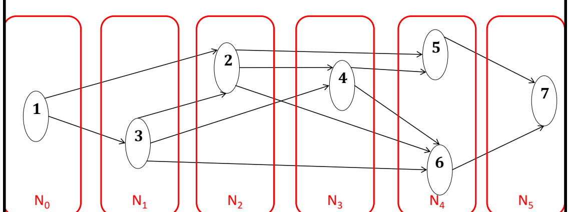
Partitionnement en niveaux - Exemple

x	1	2	3	4	5	6	7
$v(x)$	0	2	1	3	4	4	5



Partitionnement en niveaux - Exemple

x	1	2	3	4	5	6	7
$v(x)$	0	2	1	3	4	4	5

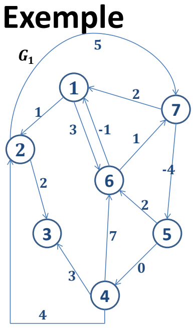


Poids d'un chemin

- On définit le poids d'un chemin γ comme la somme des poids des arcs de γ, $p(\gamma)=$. On l'appelle aussi distance.
- Un circuit γ est dit absorbant si son poids est négatif ($p(\gamma) < 0$).

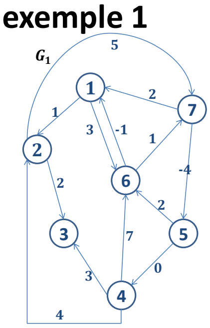
Poids d'un chemin - Exemple

- γ=1 6 7 5 4 2.
- $p(\gamma) = p(1, 6)+p(6, 7)+p(7, 5)$ $+p(5, 4)+p(4, 2)$ = 3+1-4+0+4 =4



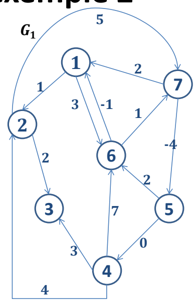
Circuit absorbant – exemple 1

- γ'=1 6 1.
- $p(\gamma) = p(1, 6)+p(6, 1)$ = 3-1 =2
- *Le circuit γ' n'est pas absorbant.*



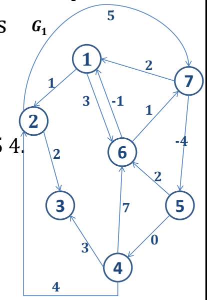
Circuit absorbant – exemple 2

- γ''=6 7 5 6.
- $p(\gamma)$ = 1-4+2 = -1
- *Le circuit γ'' est absorbant.*



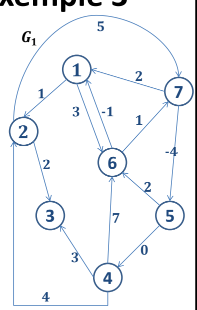
Circuit absorbant – exemple 3

- On cherche le chemin de poids minimal de 1 vers 4
  - γ_0=1 6 7 5 4.
  - $p(\gamma_0)$ = 3+1-4+0 = 0
  - γ_1=1 6 7 5 6 7 5 4=1 6(7 5 6)7 5 4.
  - $p(\gamma_1)$ = 3+1-4+2+1-4+0 = -1
  - γ_2=1 6 7 5 6 7 5 6 7 5 4 =1 6 (7 5 6)^2 7 5 4
  - $p(\gamma_2)$ = 3+(-1*2)+1-4+0 = -2
  - γ_k= 1 6 (7 5 6)^k 7 5 4.
  - $p(\gamma_k)$ = 3+(-1*k)+1-4+0 = -k



Circuit absorbant – exemple 3

- On cherche le chemin de poids minimal de 1 vers 4
- Le meilleur chemin est lorsque $-k$ est le plus petit possible
  $\Rightarrow -k \to -\infty \Rightarrow k \to +\infty$
  $\Rightarrow$Passer par le circuit un nombre infini de fois
  $\Rightarrow$ne jamais atteindre 4
  $\Rightarrow\Rightarrow$ Pas de solution



Chemin optimal

- A chaque sommet $x \in X$, on veut associer un chemin de poids optimal joignant la source du graphe $r \in X$ à x dans le réseau $R=(X, U, p)$
- Consiste à affecter à chaque sommet x d'un réseau $R=(X,U,p)$ une valeur $\pi(x)$ (appelée potentiel de x) qui représente le poids du chemin optimal reliant r à x.

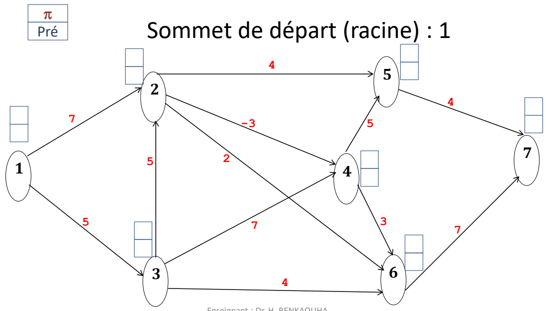
Algorithme de Bellman-Ford

- Condition :
  - Graphe sans circuits
  - Afin d'éviter les circuits absorbants

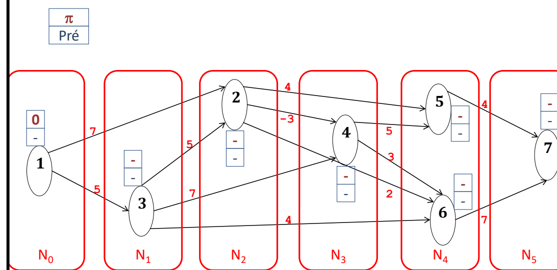
Algorithme de Bellman-Ford

```
S←{r};
Pour tout x∈X
  Faire
  π[x]← +∞ ;  Pré[x] ←NULL ;
  Fait
π[r]← 0;
Pour tout (x∈X-S) tel que (∀u∈U si T(u)=x on a I(u)∈S)
  Faire
  Pour tout ((y, x) ∈ U)
      Faire
      Si π[x] > π[y]+p[(y,x)]
           Alors π[x] ← π[y]+p[(y,x)]; Pré[x] ← y;
      fSi
      Fait
  S ← S ∪{x} ;
  Fait
```

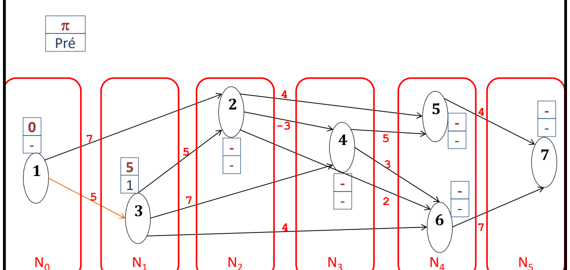
Bellman-Ford - Exemple



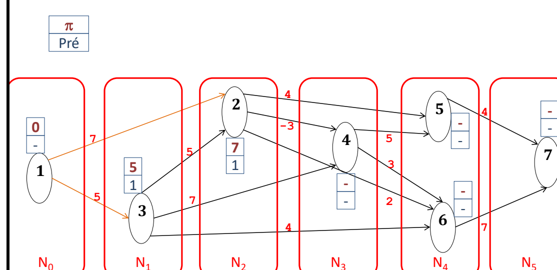
Bellman-Ford - Exemple



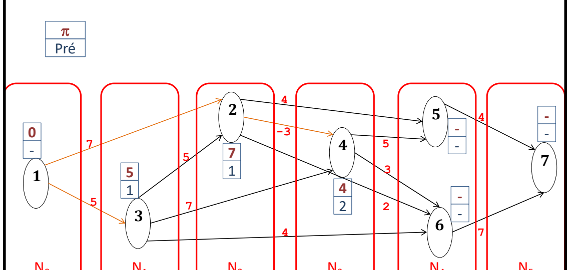
Bellman-Ford - Exemple



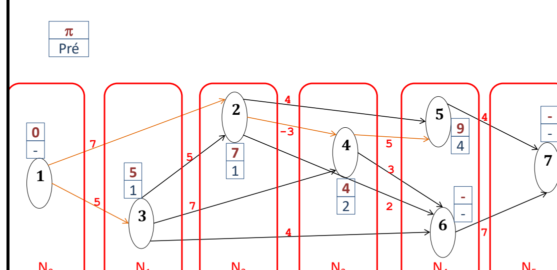
Bellman-Ford - Exemple



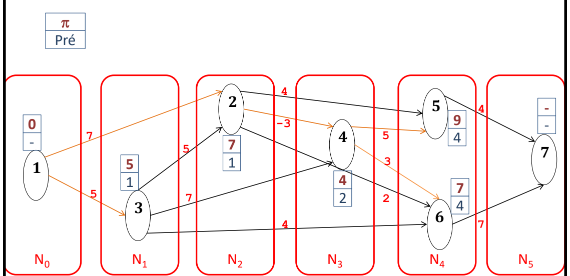
Bellman-Ford - Exemple



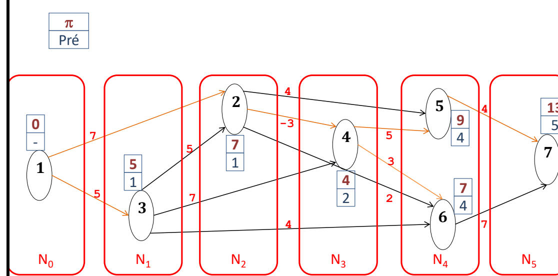
Bellman-Ford - Exemple



Bellman-Ford - Exemple



Bellman-Ford - Exemple



Bellman-Ford - Remarques

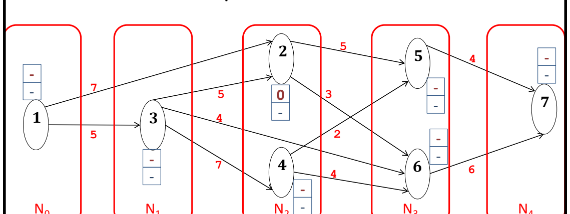
- L'algorithme retourne une arborescence optimale.
- Si le chemin optimal est le chemin de poids maximal, on change dans l'algorithme « *min* » par « *max* » et « *+∞* » par « *-∞* ».
- Si $\forall x \in X\ p(x)=1$, l'algorithme calculera le plus court chemin en nombre d'arcs.
- La complexité de l'algorithme est $o(n^2)$.

Bellman-Ford - Remarques

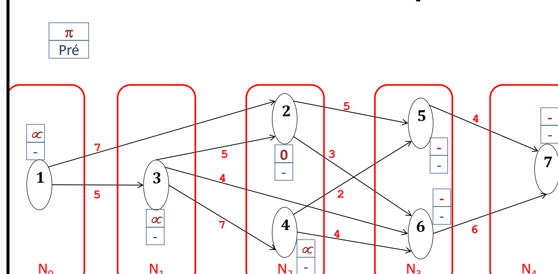
- Si le graphe contient plusieurs sources ou le sommet initial *r* n'est pas la source du graphe, il est exigé de :
  - partitionner le graphe en niveaux au préalable
  - mettre tous les sommets de niveau ≤ $v(r)$ dans *S*.
  - A chaque itération, faire le choix du prochain sommet celui qui n'est pas dans *S* ayant le plus petit niveau.

Bellman-Ford - Exemple

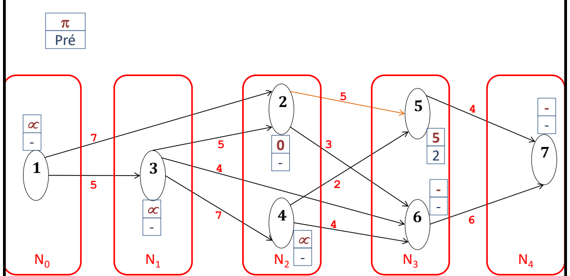
Départ : sommet 2



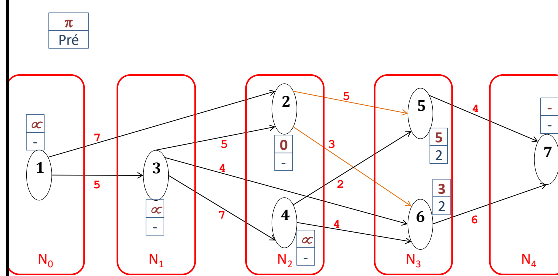
Bellman-Ford - Exemple



Bellman-Ford - Exemple

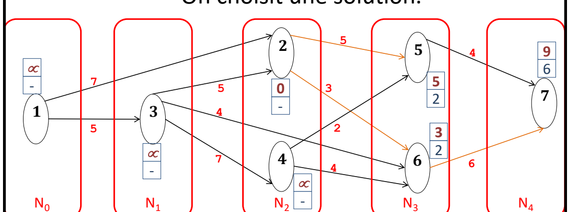


Bellman-Ford - Exemple



Bellman-Ford - Exemple

En cas d'égalité : Pas de règle,
La solution optimale n'est pas unique,
On choisit une solution.



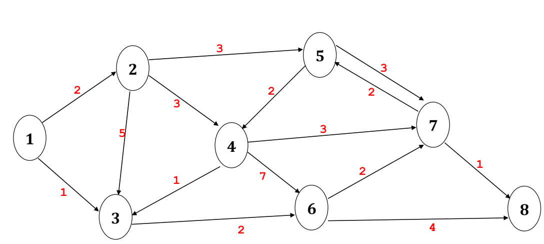
Algorithme de Dijkstra

- Condition :
  - Tous les poids positifs ou nuls
  - Pour éviter les circuits absorbants

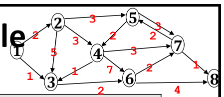
Algorithme de Dijkstra

```
s←{r};     k←1;     f[k]←r;
Pour tout x∈X faire π[x]← +∞ ; fait
π[r]←0 ;
Tant que (k<n) et (π[x]<+∞)
  Faire
  Pour tout u∈U / (I(u)=f[k]) et (T(u)∉S)
     Faire
     x = T(u);
     Si (π[x] > π[f[k]]+p[u]])
          Alors π(x) ← π[f[k]]+p[u]; Pré[x] ←
  f[k];
     fSi
     Fait
  x ← y / y ∈ X-S et π[y] minimal;
  k ← k+1; f[k] ← x; S ←S {x};
  Fait
```


Dijkstra - Exemple

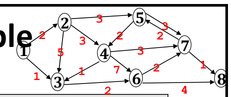


Dijkstra - Exemple



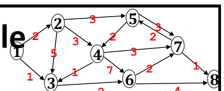
k	x	(x,y)	π								PRE							
			1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
			0	∞	∞	∞	∞	∞	∞	∞	-	-	-	-	-	-	-	-
1																		
2																		
3																		
4																		
5																		
6																		
7																		

Dijkstra - Exemple



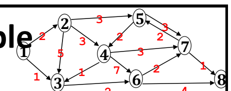
k	x	(x,y)	π								PRE							
			1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
			0	~~∞~~	~~∞~~	~~∞~~	~~∞~~	~~∞~~	~~∞~~	~~∞~~	-	-	-	-	-	-	-	-
1	1	(1,2) (1,3)		2 	 1							1 	 1					
2																		
3																		
4																		
5																		
6																		
7																		

Dijkstra - Exemple



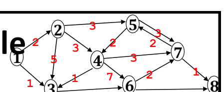
k	x	(x,y)	π								PRE							
			1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
			0	~~∞~~	~~∞~~	~~∞~~	~~∞~~	~~∞~~	~~∞~~	~~∞~~	-	-	-	-	-	-	-	-
1	1	(1,2) (1,3)		2 	 1							1 	 1					
2	3	(3,6)						3								3		
3																		
4																		
5																		
6																		
7																		

Dijkstra - Exemple



k	x	(x,y)	π								PRE							
			1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
			0	~~∞~~	~~∞~~	~~∞~~	~~∞~~	~~∞~~	~~∞~~	~~∞~~	-	-	-	-	-	-	-	-
1	1	(1,2) (1,3)		2 	 1							1 	 1					
2	3	(3,6)						3								3		
3	2	~~(2,3)~~ (2,4) (2,5)				 5 	 5							 2 	 2			
4																		
5																		
6																		
7																		

Dijkstra - Exemple

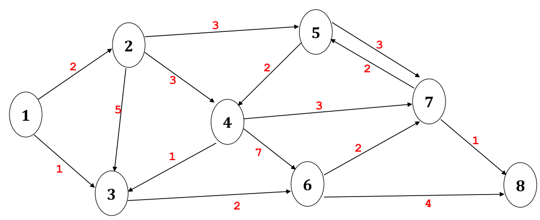


k	x	(x,y)	π								PRE							
			1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
			0	~~∞~~	~~∞~~	~~∞~~	~~∞~~	~~∞~~	~~∞~~	~~∞~~	-	-	-	-	-	-	-	-
1	1	(1,2) (1,3)		2 	 1							1 	 1					
2	3	(3,6)						3								3		
3	2	~~(2,3)~~ (2,4) (2,5)				 5 	 5							 2 	 2			
4	6	(6,7) (6,8)							5 	 7							6 	 6
5																		
6																		
7																		

Dijkstra - Exemple

k	x	(x,y)	π								PRE							
			1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
			0	~~∞~~	~~∞~~	~~∞~~	~~∞~~	~~∞~~	~~∞~~	~~∞~~	-	-	-	-	-	-	-	-
1	1	(1,2) (1,3)		2 	 1							1 	 1					
2	3	(3,6)						3								3		
3	2	~~(2,3)~~ (2,4) (2,5)				 5 	 5							 2 	 2			
4	6	(6,7) (6,8)							5 	 7							6 	 6
5	4	~~(4,3)~~ ~~(4,6)~~ (4,7)							 ~~8~~									
6																		
7																		

Dijkstra - Exemple



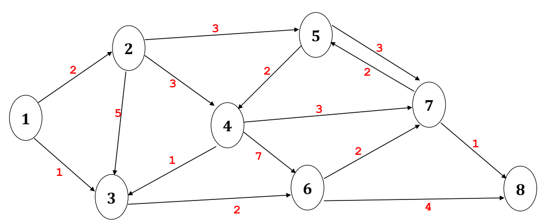
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue – USTHB)
49

Dijkstra - Exemple

k	x	(x,y)	π 1	2	3	4	5	6	7	8	PRE 1	2	3	4	5	6	7	8
			0	~~∞~~	~~∞~~	~~∞~~	~~∞~~	~~∞~~	~~∞~~	~~∞~~	-	-	-	-	-	-	-	-
1	1	(1,2) (1,3)		2	1							1	1					
2	3	(3,6)						3								3		
3	2	~~(2,3)~~ (2,4) (2,5)				5	5							2	2			
4	6	(6,7) (6,8)							5	7							6	6
5	4	~~(4,3)~~ ~~(4,6)~~ (4,7)							~~8~~									
6	5	~~(5,4)~~ (5,7)							~~8~~									
7																		

Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue – USTHB)
50

Dijkstra - Exemple



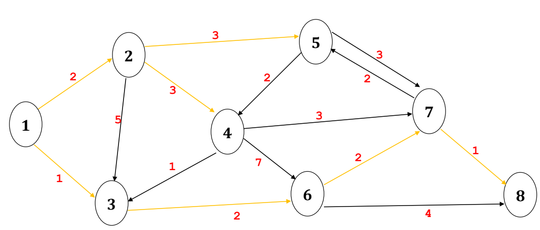
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue – USTHB)
51

Dijkstra - Exemple

k	x	(x,y)	π 1	2	3	4	5	6	7	8	PRE 1	2	3	4	5	6	7	8
			0	~~∞~~	~~∞~~	~~∞~~	~~∞~~	~~∞~~	~~∞~~	~~∞~~	-	-	-	-	-	-	-	-
1	1	(1,2) (1,3)		2	1							1	1					
2	3	(3,6)						3								3		
3	2	~~(2,3)~~ (2,4) (2,5)				5	5							2	2			
4	6	(6,7) (6,8)							5	~~7~~							6	~~6~~
5	4	~~(4,3)~~ ~~(4,6)~~ (4,7)							~~8~~									
6	5	~~(5,4)~~ (5,7)							~~8~~									
7	7	~~(7,5)~~ (7,8)								6								7

Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue – USTHB)
52

Dijkstra - Exemple



Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue – USTHB)
53

Dijkstra - Remarques

- On dit que l'algorithme retourne une arborescence optimale.
- La complexité de l'algorithme est $o(m^2)$ où m est le nombre d'arcs.
- Si le graphe est non orienté, on peut associer à chaque arête $\{x,y\}$ de poids p, deux (2) arcs (x,y) et (y,x) de même poids p, puis on applique l'algorithme de Dijkstra.

Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue – USTHB)
54

Chapitre 5
Problème d'ordonnancement

Présenté par :
H. BENKAOUHA
Bureau 222, Faculté d'Informatique, USTHB
haroun.benkaouha@usthb.edu.dz
haroun.benkaouha@gmail.com

Identification du problème

- L'examen d'un projet (industriel, administratif, informatique, …) comporte en général deux phases importantes:
  - Phase 1 : modélisation
  - Phase 2 : analyse et étude

Phase 1

- La division du projet en plusieurs tâches (ou étapes) élémentaires,
- L'étude des liaisons (contraintes logiques, chronologiques,…) et l'estimation de la durée de chaque tâche.
- Permet de construire d'un graphe orienté où :
  - les sommets représentent les tâches élémentaires
  - les arcs sont pondérés.

Phase 2

- Elle consiste à analyser et étudier le graphe obtenu en phase1.
- Parmi les résultats de cette phase :
  - la détermination d'un planning (ou ordre chronologique)
  - la recherche de la durée totale du projet dans le but de la minimiser.
- L'étude du graphe revient à déterminer un chemin de poids optimal.

Contraintes potentielles

- Certaines tâches sont liées entre elles par des contraintes dites de potentielles,
- Sont exprimées sous la forme $t_j - t_i \geq a_{ij}$
  - où t_i et t_j représentent les <u>dates de début au plus tôt</u> des tâches *i* et *j*.
  - Correspond à : «la tâche *j*, ne peut commencer qu'après que *i* aura consommée a_{ij} unités de temps depuis son début».

Graphe potentiel tâches

- Appelé aussi Méthode Potentiels Métra (MPM)
- Chaque tâche est représentée par un sommet
- chaque arc $(i,j) \in U$ est créé si on a une contrainte de potentiel entre *i* et *j* ($t_j - t_i \geq a_{ij}$) tel que son poids $p(i,j)=a_{ij}$.
- On introduit 2 <u>**tâches fictives**</u> *«D»* et *«F»* représentant le début et la fin du projet.

Graphe potentiel tâches

- Pour chaque tâche *i* n'ayant de tâche précédente, on rajoute l'arc (D,i). Le poids de l'arc est généralement 0 sauf s'il a été précisé dans l'énoncé un délai entre le début du projet et cette tâche.
- Pour chaque tâche *i*, on rajoute l'arc (i,F). $p(i,F)$= la durée de *i*.
- On peut optimiser le graphe en supprimant certains arcs (i,F) qui risquent de représenter des contraintes redondantes.

Dates au plus tôt

- La durée minimale de l'ordonnancement est donnée par la date au plus tôt (t_F) de la tâche fictive «fin projet».
- La date au plus tôt d'une tâche *j* est obtenue par la formule suivante :
- $t_j = \max_{i \in LP(j)} \{t_i + a_{ij}\}$, $j \neq 0$.
- $LP(j)$ est la liste des prédécesseurs du sommet *j*.
- $t_0 = t_D = 0$

Dates au plus tard

- t^*_i : <u>date au plus tard</u> de la tâche *i* calculée comme suit :
- $t^*_F = t_F$
- $t^*_i = \min_{j \in LS(i)} \{t^*_j - a_{ij}\}$ $i \neq F$=*«fin projet»*.
- $LS(i)$ est la liste des successeurs du sommet *i*.

Marge totale

- Le délai de retard pour une tâche *i* qui n'affecte pas la durée minimale du projet est noté M_i et est appelé **<u>marge totale</u> de la tâche *i***.
- $M_i = t^*_i - t_i$ où :
  - t_i : date au plus tôt de la tâche *i*
  - t^*_i : date au plus tard de la tâche *i*
- Une tâche *i* ayant une marge totale M_i= 0 et est appelé **<u>tâche critique</u>**.

Marge libre

- La **<u>marge libre</u> d'une tâche *i***,
  - notée m_i,
  - est le délai de retard d'une tâche *i*
  - sans affecter les dates de début au plus tôt des tâches postérieures.
- $m_i = \{t_j - t_i - a_{ij}\}$

Marge certaine

- La **<u>marge certaine</u> d'une tâche *i***,
  - notée μ_i,
  - est le délai de retard d'une tâche *i*,
  - quand les tâches antérieures commencent à leurs dates au plus tard
  - et les tâches postérieures à leurs dates plus tôt
- $\mu_i = \max \{ 0, \{t_j - a_{ij}\} - \{t^*_k + a_{ki}\}\}$

Exemple — Enoncé

Tâche *i*	Durée de *i*	Contraintes liées au début d'exécution de la tâche *i*
1	6	-
2	10	Ne peut débuter que 2 jours après le début des travaux.
3	20	Après la fin de 1 et 2.
4	7	Après la fin de 1.
5	8	Peut débuter 2 jours après le début de 2.
6	4	Après la fin de 5 et 10 jours au maximum après le début de 4.
7	10	Peut débuter après la fin de 4 et lorsque 3 est à moitié réalisé.
8	5	Ne doit pas dépasser 5 jours après la fin de 6 et la 5 doit être achevée.
9	12	-
10	20	Peut commencer 5 jours après la fin de 9.

Exemple — Modélisation (1/14)

Tâche *i*	Durée de *i*	Contraintes liées au début d'exécution de la tâche *i*
1	6	-

- 1 n'a pas de contraintes

  $\Rightarrow$ On lui rajoute une avec la tâche fictive de début des travaux qu'on note 0.
- 1 doit commencer après 0.

  $t_1 \geq t_0$

  $\Rightarrow$

  $t_1 - t_0 \geq 0$

Exemple — Modélisation (2/14)

Tâche *i*	Durée de *i*	Contraintes liées au début d'exécution de la tâche *i*
2	10	Ne peut débuter que 2 jours après le début des travaux.

- On a noté par 0 la tâche fictive de début des travaux.

  $t_2 \geq t_0 + 2$

  $\Rightarrow$

  $t_2 - t_0 \geq 2$

Exemple — Modélisation (3/14)

Tâche *i*	Durée de *i*	Contraintes liées au début d'exécution de la tâche *i*
1	6	-
2	10	Ne peut débuter que 2 jours après le début des travaux.
3	20	Après la fin de 1 et 2.

$t_3 \geq t_1 + \text{durée}(1)$

$\Rightarrow t_3 \geq t_1 + 6$

$\Rightarrow t_3 - t_1 \geq 6$

$t_3 \geq t_2 + \text{durée}(2)$

$\Rightarrow t_3 \geq t_2 + 10$

$\Rightarrow t_3 - t_2 \geq 10$

Exemple — Modélisation (4/14)

Tâche *i*	Durée de *i*	Contraintes liées au début d'exécution de la tâche *i*
1	6	-
4	7	Après la fin de 1.

$t_4 \geq t_1 + \text{durée}(1)$

$\Rightarrow$

$t_4 \geq t_1 + 6$

$\Rightarrow$

$t_4 - t_1 \geq 6$

Exemple — Modélisation (5/14)

Tâche *i*	Durée de *i*	Contraintes liées au début d'exécution de la tâche *i*
2	10	Ne peut débuter que 2 jours après le début des travaux.
5	8	Peut débuter 2 jours après le début de 2.

$t_5 \geq t_2 + 2$

$\Rightarrow$

$t_5 - t_2 \geq 2$

Exemple — Modélisation (6/14)

Tâche *i*	Durée de *i*	Contraintes liées au début d'exécution de la tâche *i*
4	7	Après la fin de 1.
5	8	Peut débuter 2 jours après le début de 2.
6	4	Après la fin de 5 et 10 jours au maximum après le début de 4.

$t_6 \geq t_5$+durée(5)

$\Rightarrow t_6 \geq t_5+8$

$\Rightarrow t_6 - t_5 \geq 8$

$t_6 \leq t_4+10$

$\Rightarrow t_6 - t_4 \leq 10$

$\Rightarrow t_4 - t_6 \geq -10$

Exemple — Modélisation (7/14)

Tâche *i*	Durée de *i*	Contraintes liées au début d'exécution de la tâche *i*
3	20	Après la fin de 1 et 2.
4	7	Après la fin de 1.
7	10	Peut débuter après la fin de 4 et lorsque 3 est à moitié réalisé.

$t_7 \geq t_4$+durée(4)

$\Rightarrow t_7 \geq t_4+7$

$\Rightarrow t_7 - t_4 \geq 7$

$t_7 \geq t_3$+(durée(3)/2)

$\Rightarrow t_7 \geq t_3+(20/2)$

$\Rightarrow t_7 - t_3 \geq 10$

Exemple — Modélisation (8/14)

Tâche *i*	Durée de *i*	Contraintes liées au début d'exécution de la tâche *i*
5	8	Peut débuter 2 jrs après le début de 2.
6	4	Après la fin de 5 et 10 jrs au maximum après le début de 4.
8	5	Ne doit pas dépasser 5 jrs après la fin de 6 et la 5 doit être achevée.

$t_8 \leq t_6$+durée(6)+5

$\Rightarrow t_8 \leq t_6+4+5$

$\Rightarrow t_6 - t_8 \geq -9$

$t_8 \geq t_5$+durée(5)

$\Rightarrow t_8 \geq t_5+8$

$\Rightarrow t_8 - t_5 \geq 8$

Exemple — Modélisation (9/14)

Tâche *i*	Durée de *i*	Contraintes liées au début d'exécution de la tâche *i*
9	12	-

- 9 n'a pas de contraintes

  $\Rightarrow$ On lui rajoute une avec la tâche fictive de début des travaux qu'on a noté 0.
- 9 doit commencer après 0.

  $t_9 \geq t_0$

  $\Rightarrow$

  $t_9 - t_0 \geq 0$

Exemple — Modélisation (10/14)

Tâche *i*	Durée de *i*	Contraintes liées au début d'exécution de la tâche *i*
9	12	-
10	20	Peut commencer 5 jours après la fin de 9.

$t_{10} \geq t_9$+durée(9)+5

$\Rightarrow$

$t_{10} \geq t_9+12+5$

$\Rightarrow$

$t_{10} - t_9 \geq 17$

Exemple — Modélisation (11/14)

- De chaque tâche *i* (*i* de 1 à 10, i.e. toutes les tâches sauf les tâches fictives)
- La contrainte *i* doit se terminer avant la fin du projet ou fin de projet après la fin de toutes les tâches
- On note par 11 la tâche fictive de fin du projet.

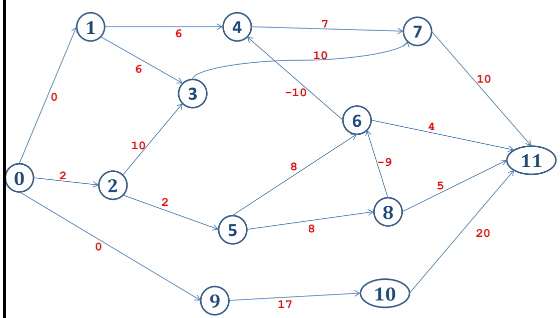
  $t_{11} \geq t_i$+durée(i), $\forall i$ de 1 à 10

Exemple — Modélisation (12/14)

- Le système de contraintes :

t_1-t_0≥ 0
t_2-t_0≥ 2
t_3-t_1≥ 6
t_3-t_2≥ 10
t_4-t_1≥ 6
t_5-t_2≥ 2
t_6-t_5≥ 8
t_4-t_6≥ -10
t_7-t_4≥ 7
t_7-t_3≥ 10
t_6-t_8≥ -9
t_8-t_5≥ 8
t_9-t_0≥ 0
t_{10}-t_9≥ 17
t_{11}≥ t_i+durée(i), i de 1 à 10

Exemple — Modélisation (13/14)

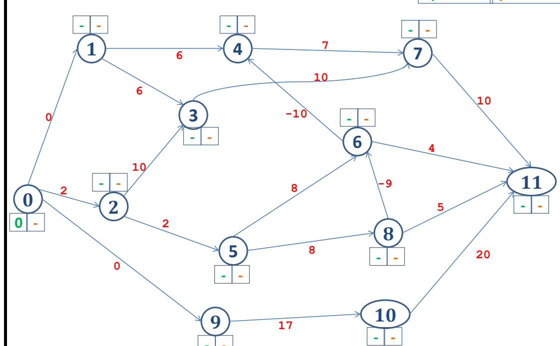


Exemple — Modélisation (14/14)

- Pour optimiser, nous avons éviter certains arcs vers la tâche fictive de fin de projet (11)
- Si ∃ γ chemin de i vers 11 et $p(\gamma)\geq durée(i)$ Alors
- Pas la peine de rajouter l'arc $(i, 11)$
- Cette étape n'est pas obligatoire

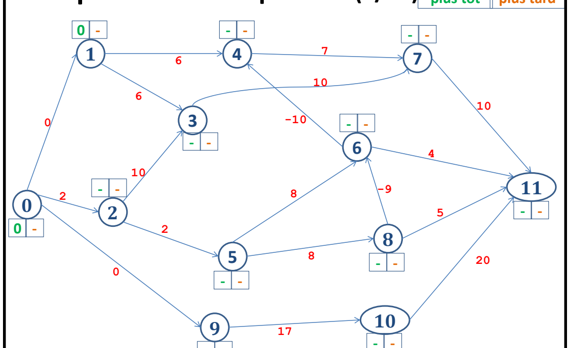
Exemple — Dates au plus tôt (1/13)

Date au plus tôt | Date au plus tard



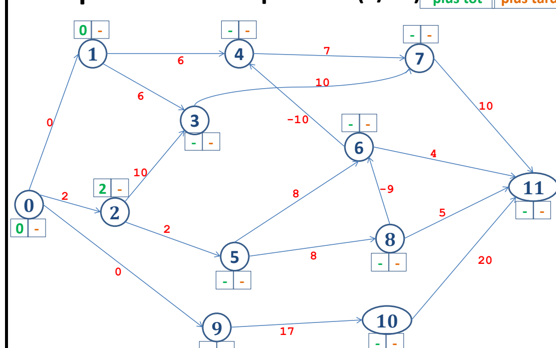
Exemple — Dates au plus tôt (2/13)

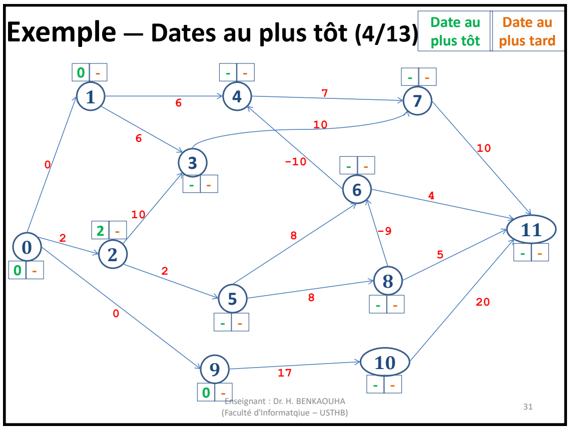
Date au plus tôt | Date au plus tard

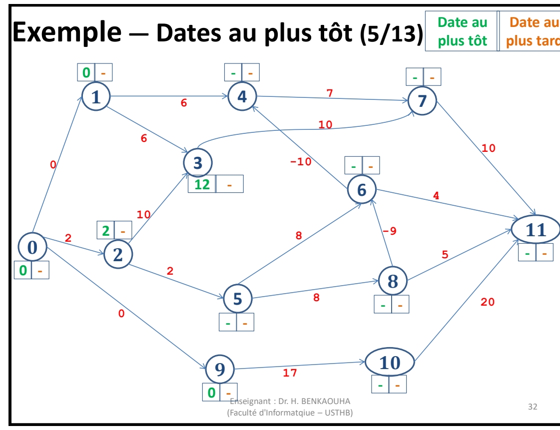


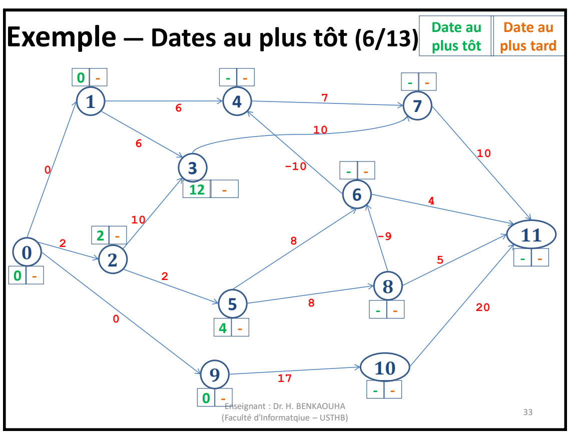
Exemple — Dates au plus tôt (3/13)

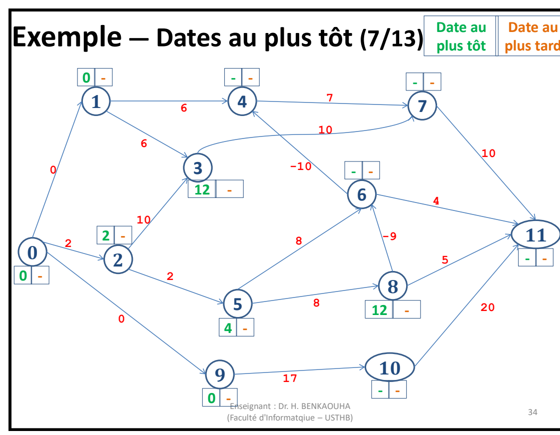
Date au plus tôt | Date au plus tard

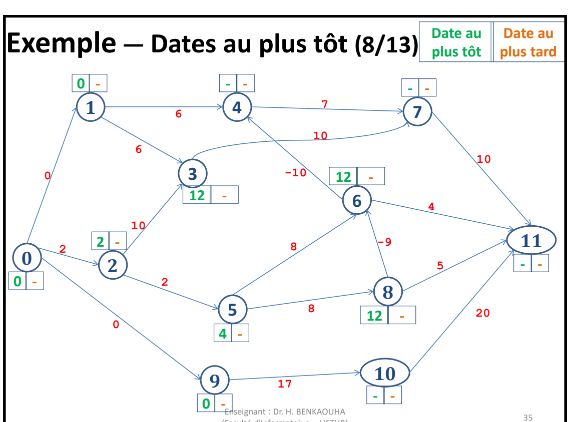


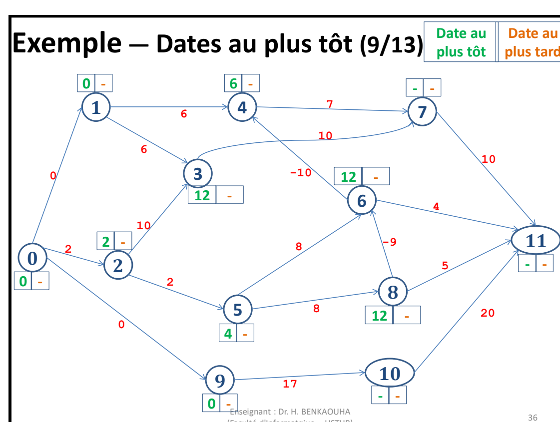

Exemple – Dates au plus tôt (4/13)
Date au plus tôt
Date au plus tard
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue – USTHB)
31


Exemple – Dates au plus tôt (5/13)
Date au plus tôt
Date au plus tard
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue – USTHB)
32


Exemple – Dates au plus tôt (6/13)
Date au plus tôt
Date au plus tard
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue – USTHB)
33

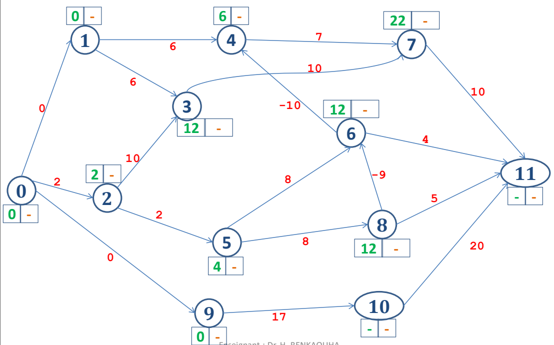

Exemple – Dates au plus tôt (7/13)
Date au plus tôt
Date au plus tard
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue – USTHB)
34


Exemple – Dates au plus tôt (8/13)
Date au plus tôt
Date au plus tard
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue – USTHB)
35


Exemple – Dates au plus tôt (9/13)
Date au plus tôt
Date au plus tard
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue – USTHB)
36

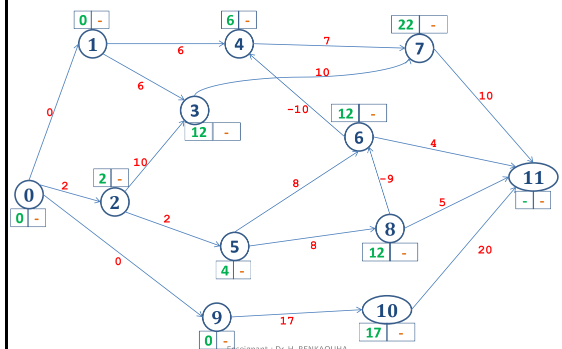
Exemple – Dates au plus tôt (10/13)

Date au plus tôt | Date au plus tard



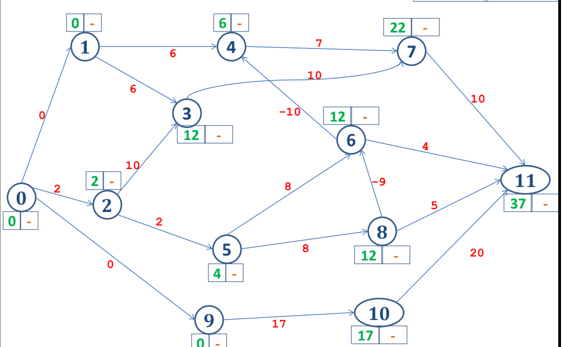
Exemple – Dates au plus tôt (11/13)

Date au plus tôt | Date au plus tard



Exemple – Dates au plus tôt (12/13)

Date au plus tôt | Date au plus tard



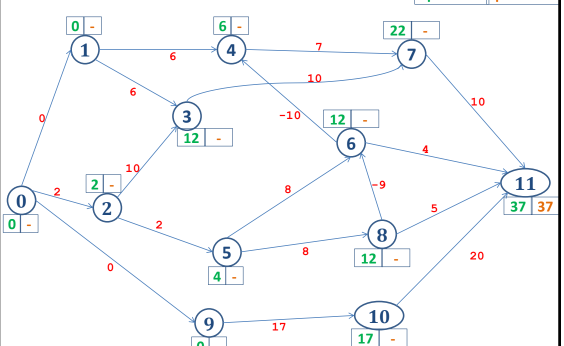
Exemple – Dates au plus tôt (13/13)

Tâches	1	2	3	4	5	6	7	8	9	10
Dates au plus tôt	0	2	12	6	4	12	22	12	0	17

- La durée optimale du projet est de **37** jours

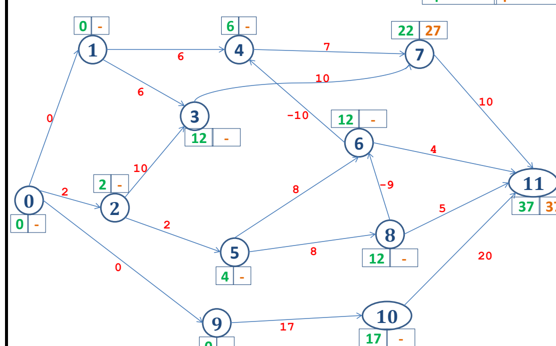
Exemple – Dates au plus tard (1/13)

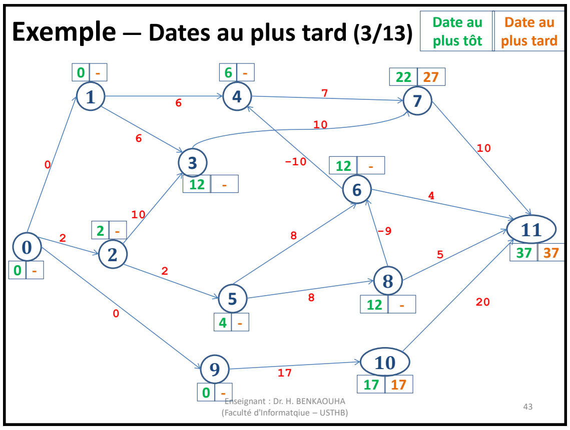
Date au plus tôt | Date au plus tard

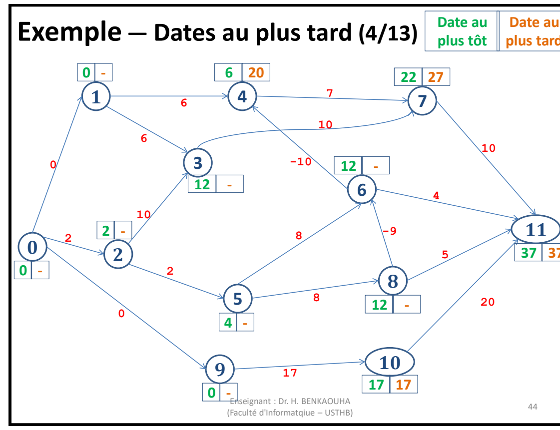


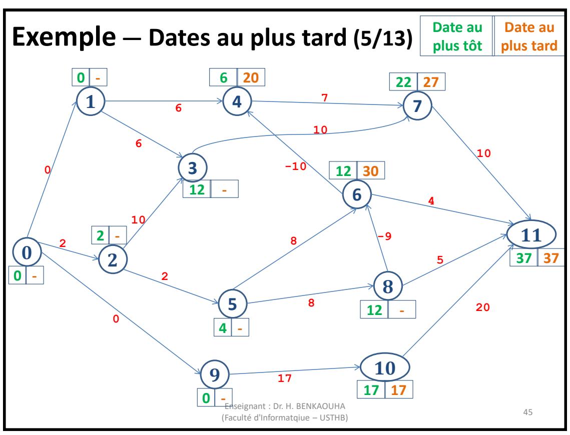
Exemple – Dates au plus tard (2/13)

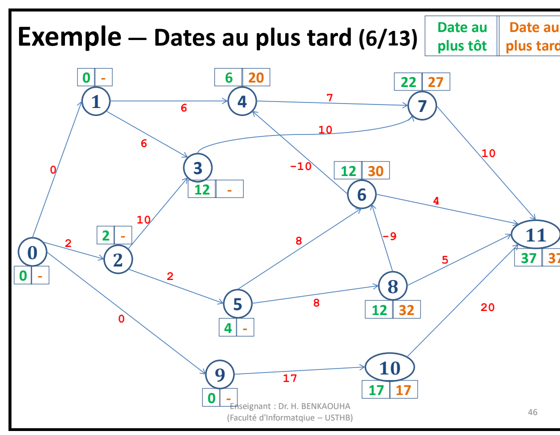
Date au plus tôt | Date au plus tard

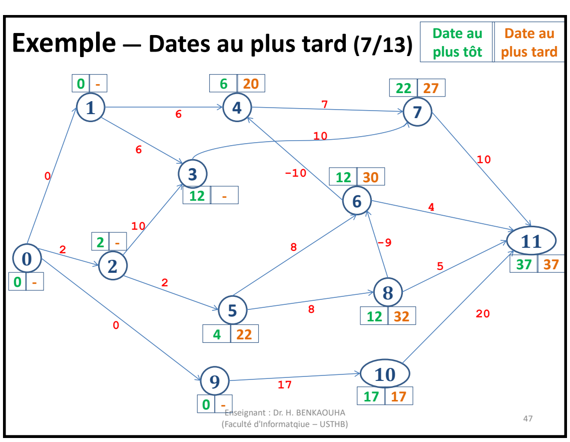


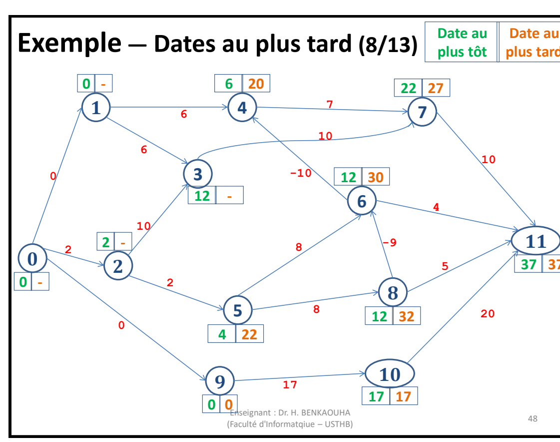

Exemple — Dates au plus tard (3/13)
Date au plus tôt
Date au plus tard
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue – USTHB)
43


Exemple — Dates au plus tard (4/13)
Date au plus tôt
Date au plus tard
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue – USTHB)
44


Exemple — Dates au plus tard (5/13)
Date au plus tôt
Date au plus tard
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue – USTHB)
45

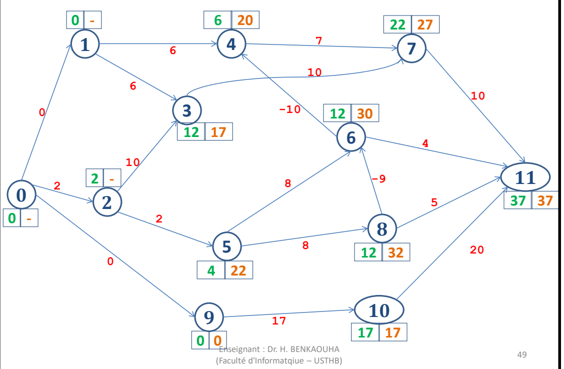

Exemple — Dates au plus tard (6/13)
Date au plus tôt
Date au plus tard
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue – USTHB)
46


Exemple — Dates au plus tard (7/13)
Date au plus tôt
Date au plus tard
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue – USTHB)
47


Exemple — Dates au plus tard (8/13)
Date au plus tôt
Date au plus tard
Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue – USTHB)
48

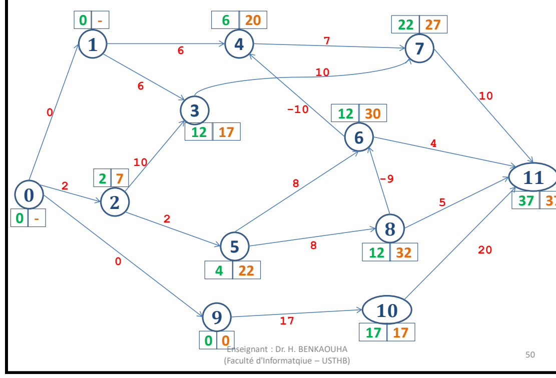
Exemple — Dates au plus tard (9/13)

Date au plus tôt | Date au plus tard



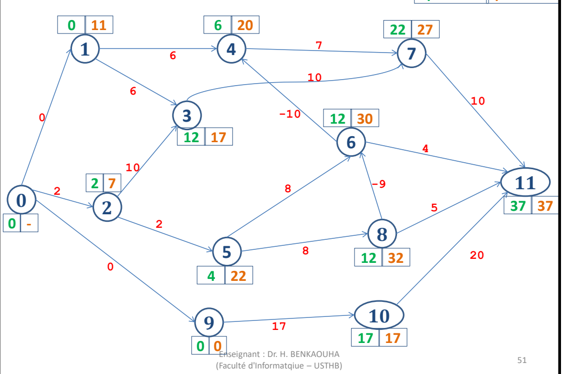
Exemple — Dates au plus tard (10/13)

Date au plus tôt | Date au plus tard



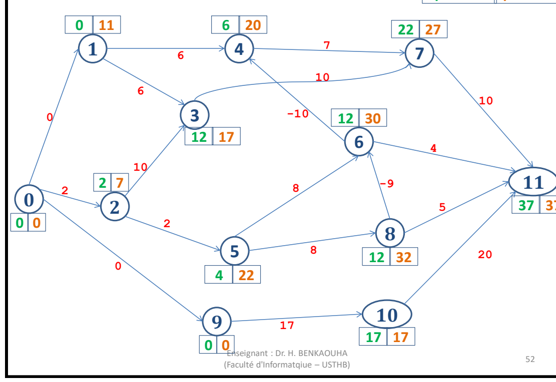
Exemple — Dates au plus tard (11/13)

Date au plus tôt | Date au plus tard



Exemple — Dates au plus tard (12/13)

Date au plus tôt | Date au plus tard



Exemple — Dates au plus tard (13/13)

Tâches	1	2	3	4	5	6	7	8	9	10
Dates au plus tôt	0	2	12	6	4	12	22	12	0	17
Dates au plus tard	11	7	17	20	22	30	27	32	0	17

Exemple — Marge/Tâches critiques/Chemin critique

Tâches	1	2	3	4	5	6	7	8	9	10
Dates au plus tôt	0	2	12	6	4	12	22	12	0	17
Dates au plus tard	11	7	17	20	22	30	27	32	0	17
Marge totale	11	5	5	14	18	18	5	10	0	0

- Les tâches critiques : **9** et **10**
- Le chemin critique : **0 9 10 11**

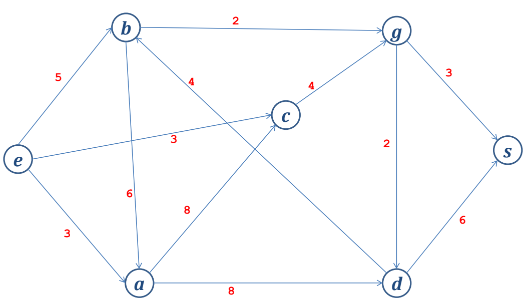
Chapitre 6
Les Flots

Présenté par :
H. BENKAOUHA
Bureau 222, Faculté d'Informatique, USTHB
haroun.benkaouha@usthb.edu.dz
haroun.benkaouha@gmail.com

Réseau de transport

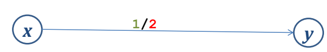
- Graphe orienté pondéré $G=(X, U, c)$
  - X : ensemble de sommets,
  - U : ensemble d'arcs
  - $c(u)$ poids de l'arc u, $\forall\ u \in U$
  - $c(u) \geq 0$ appelé **capacité de l'arc *u***.
- Il a 2 sommets particuliers :
  - e : **entrée du graphe** (source $d^-_G(e)=0$)
  - s : **sortie du graphe** (puits $d^+_G(s)=0$)

Réseau de transport : Exemple



Flux d'un arc

- Soit $G=(X, U, c)$ un réseau de transport.
- A chaque arc u, une autre valeur $f(u) \in R$.
- $f(u)$ est appelé le flux de l'arc u.
- $f(u)$ est dit réalisable ssi $0 \leq f(u) \leq c(u)$



- En vert la valeur du flot de l'arc (x, y) qui est réalisable.

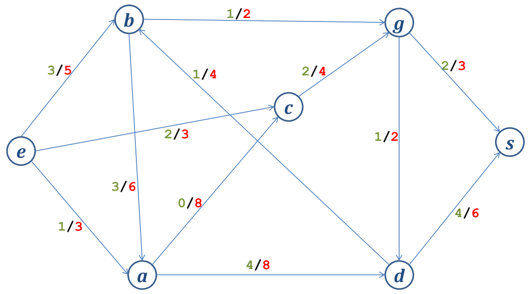
Flot dans un réseau de transport

- Soit $G=(X, U, c)$ un réseau de transport
- $|U|=m$.
- Un flot dans G :
  - vecteur f de m composantes $\in R$.
- La $j^{\text{ème}}$ composante correspond au :
  - flux de l'arc u_j : $f(u_j)$

Flot compatible

- Soit $G=(X, U, c)$ un réseau de transport
- Un flot f est dit <u>compatible</u> ssi
  - Chaque flux est réalisable : $\forall u \in U,\ 0 \leq f(u) \leq c(u)$
  - Loi de *Kirchhoff* : $\forall x \in X - \{e, s\}, \sum_{u\in\omega^+(\{x\})} f(u) - \sum_{u\in\omega^-(\{x\})} f(u) = 0$
    - $\omega^+(\{x\})$ est l'ensemble des arcs sortants de x
    - $\omega^-(\{x\})$ est l'ensemble des arcs entrants vers x
- On dit aussi <u>flot réalisable</u>

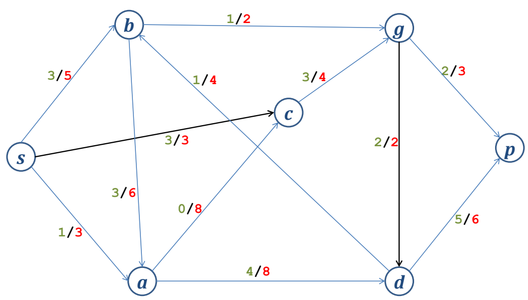
Flot compatible : Exemple



Arc saturé par un flot

- Soit $G=(X, U, c)$ un réseau de transport
- Soit un arc $u \in U$
- u est dit saturé par un flot compatible f ssi : $f(u)=c(u)$.

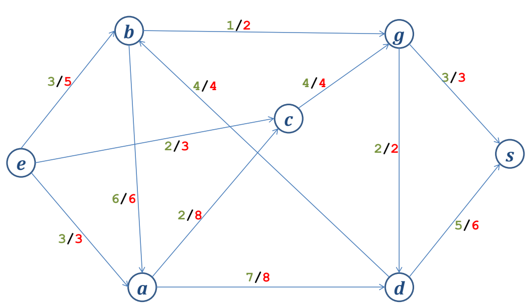
Arc saturé : Exemple



Flot complet

- Soit $G=(X, U, c)$ un réseau de transport.
- $e \in X$ est l'entrée de G
- $s \in X$ est la sortie de G
- Soit f un flot compatible dans G.
- f est dit complet ssi :
- Tout chemin de e vers s passe par un arc saturé.

Flot complet : Exemple

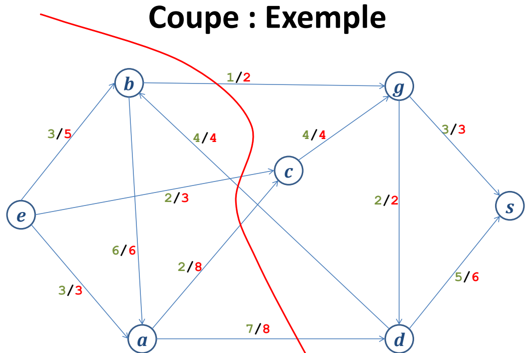


Coupe

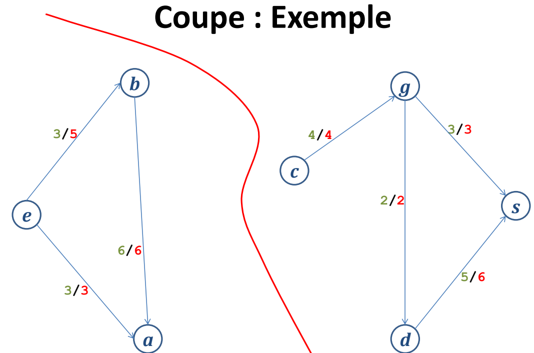
- Soit $G=(X, U, c)$ un réseau de transport.
- e et $s \in X$ respectivement : entrée et sortie de G.
- Un flot compatible f.
- Une coupe est un ensemble d'arcs C
  - Déconnectant e de s.

  $\Rightarrow$Aucun chemin de e vers s.

Coupe : Exemple



Coupe : Exemple



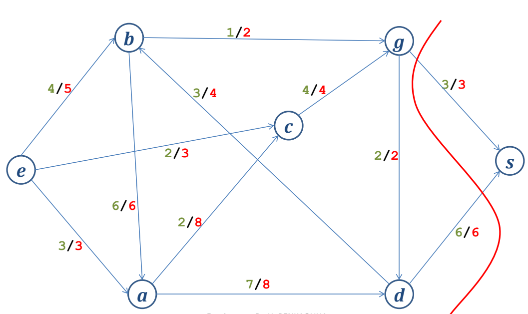
Flot maximal (Définition 1.)

- Soit *G=(X, U, c)* un réseau de transport.
- *e* et *s* ∈*X* respectivement : entrée et sortie de *G*.
- Un flot compatible *f*.
- *f* est dit <u>maximal</u> dans *G* si et seulement si
- $\sum_{x \in X} f(e,x) = \sum_{x \in X} f(x,s)$ est de valeur maximale.

Flot maximal (Définition 2.)

- Soit *G=(X, U, c)* un réseau de transport.
- *e* et *s* ∈*X* respectivement : entrée et sortie de *G*.
- Un flot compatible *f*.
- *f* est dit <u>maximal</u> dans *G* si et seulement si
- Il existe une coupe *C* dont tous les arcs sont saturés appelée **<u>coupe minimale</u>**.
- **Remarque :** Un flot maximal est complet

Flot maximal : Exemple



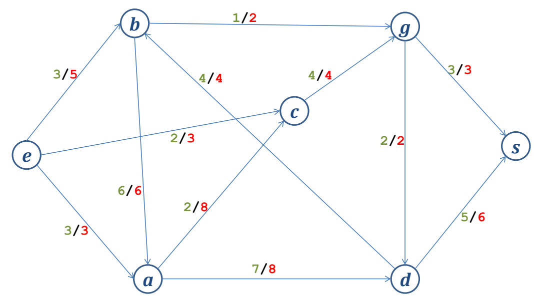
Graphe résiduel

- Soit *G=(X, U, c)* un réseau de transport.
- Un flot compatible *f*.
- On peut construire un réseau résiduel $G_f=(X, U_f, c_f)$ comme suit :
  - Pour tout arc $u=(x, y)$ de capacité $c(u)$ et flux $f(u)$ dans G, on a dans G_f :
  - Un arc (x, y) de poids $c(u)$-$f(u)$ ssi $c(u)$-$f(u)$>0
  - Un arc (y, x) de poids $f(u)$ ssi $f(u)$>0

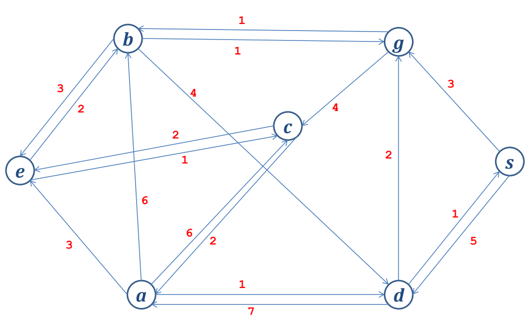
Graphe résiduel : Algorithme

```
Uf←∅ ;
Pour tout arc u=(x,y)∈U
    Faire
        Si (f(u)≠0)
            Alors       Uf ← Uf ∪ {(y,x)} ;
                        cf(y,x) ← f(u) ;
        fSi
        Si (c(u) - f(u)≠0)
            Alors       Uf ← Uf ∪ {(x,y)} ;
            cf(x,y) ← c(u) - f(u) ;
        fSi
    Fait
```


Graphe résiduel : Exemple (1/2)



Graphe résiduel : Exemple (2/2)

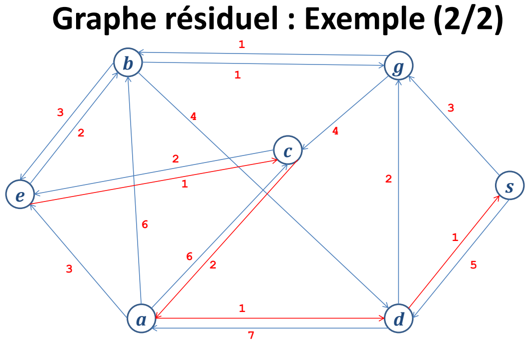


Chemin d'augmentation

- $G=(X, U, c)$ réseau de transport
- e et $s \in X$ respectivement : entrée et sortie de G.
- f un flot compatible
- $G_f=(X, U_f, c_f)$ le réseau résiduel de G associé à f.
- Tout chemin γ dans G_f de e vers s est un chemin d'augmentation de capacité $c(\gamma)$ où

$$c(\gamma) = \underset{u \in \gamma}{Min}(c_f(u))$$

Graphe résiduel : Exemple (2/2)



$\gamma = e\ c\ a\ d\ s$ est un chemin d'augmentation de capacité 1

Flot maximal (Re-Définition)

- $G=(X, U, c)$ réseau de transport
- e et $s \in X$ sont respectivement les entrée et sortie de G.
- f un flot compatible
- $G_f=(X, U_f, c_f)$ le réseau résiduel de G associé à f.
- f est <u>maximal</u>, s'il n'existe <u>aucun chemin</u> d'augmentation dans le réseau graphe G_f

Algorithme de Ford-Fulkerson

- **Principe :**
  1. Réseau de transport en entrée
  2. Un flot initial compatible (Sinon 0)
  3. Calculer le graphe résiduel
  4. Trouver un chemin d'augmentation γ de capacité c(γ) si pas de chemin FIN.
  5. Incrémenter les flux des arcs de G dans le même sens que γ de c(γ)
  6. Décrémenter les flux des arcs de G dans le même sens que γ de c(γ) et revenir à (3).

Algorithme de Ford-Fulkerson

```
Gf ← Graphe_Residuel(G, f)
Tant Que (∃γ chemin d'augmentation dans Gf)
    Faire
      Pour tout (u=(x,y)∈γ)
        Faire
          Si (u∈U)  Alors f(u) ← f(u)+c(γ)
                    Sinon f(y,x) ← f(y,x)-c(γ)
          fSi
        Fait
      Gf ← Graphe_Residuel(G, f)
    Fait
```


Remarque (1/2)

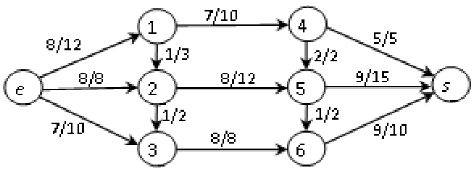
- Si le graphe initial contient plusieurs sources $(s_1, s_2, \ldots, s_p)$:
- On ajoute un nouveau sommet *e*
- *e* sera la nouvelle unique source (entrée du réseau).
- Pour chaque ancienne source s_k, on ajoute un arc (e, s_k) de capacité $c(e, s_k) = \Sigma_{x\in X}\, c(s_k, x)$

Remarque (2/2)

- Si le graphe initial contient plusieurs puits $(p_1, p_2, \ldots, p_q)$:
- On ajoute un nouveau sommet *s*
- *s* sera le nouveau unique puits (sortie du réseau).
- Pour chaque ancien puits p_k, on ajoute un arc (p_k, s) de capacité $c(p_k, s) = \Sigma_{x\in X}\, c(x, p_k)$

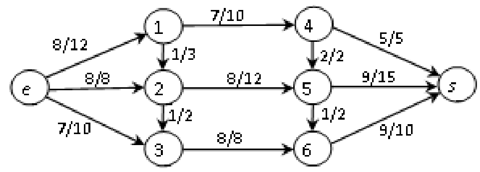
Exemple Général (1/10)

- On a le réseau de transport suivant. Sur chaque arc, nous avons le flux de l'arc / capacité de l'arc



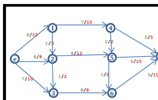
Exemple Général (2/10)

- Le flot est compatible (réalisable) :
  - Chaque flux est réalisable
  - Vérification loi de Kirchhoff à chaque sommet



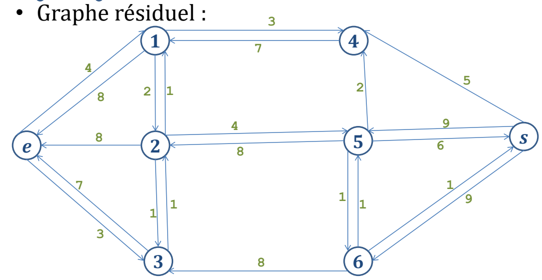
Exemple Général (3/10)

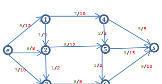
- Recherche du flot maximal :
  - On applique l'algorithme de Ford-Fulkerson
  - On démarre de ce flot compatible



Exemple Général (4/10)

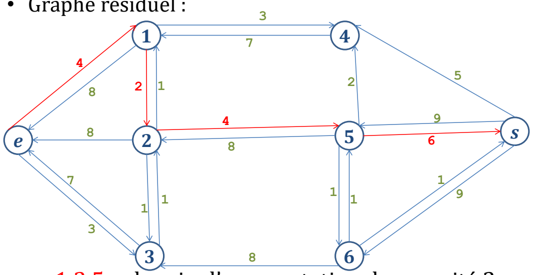
- Graphe résiduel :



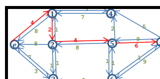


Exemple Général (4/10)

- Graphe résiduel :

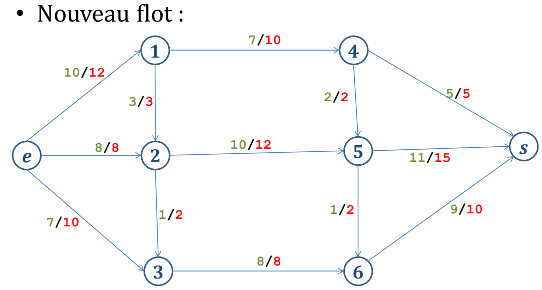


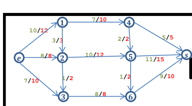
$\gamma = e\ 1\ 2\ 5\ s$ chemin d'augmentation de capacité 2.



Exemple Général (5/10)

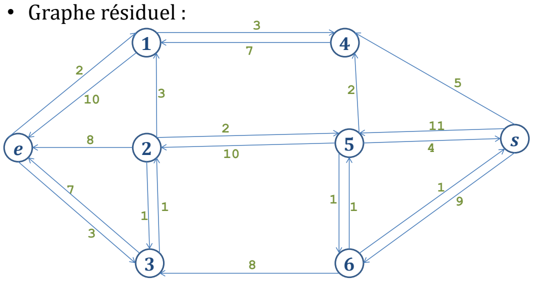
- Nouveau flot :

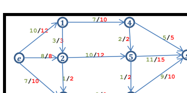




Exemple Général (6/10)

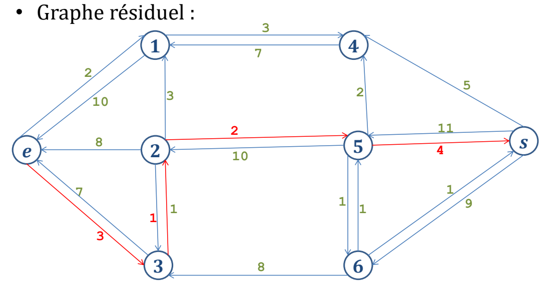
- Graphe résiduel :



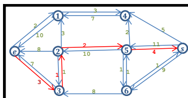


Exemple Général (6/10)

- Graphe résiduel :

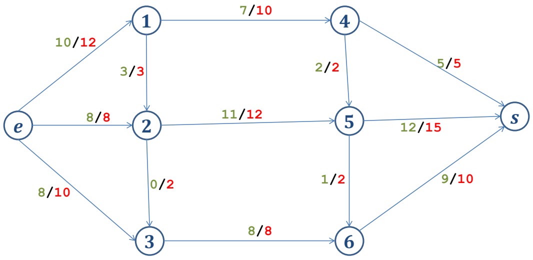


$\gamma = e\ 3\ 2\ 5\ s$ chemin d'augmentation de capacité 1.

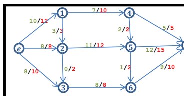


Exemple Général (7/10)

- Nouveau flot :

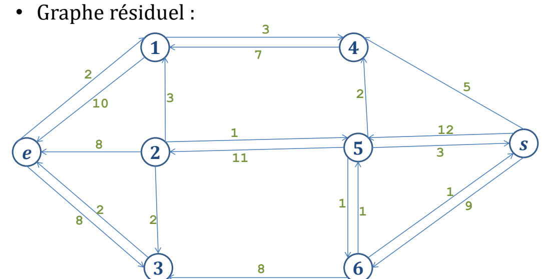


Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB) 37



Exemple Général (8/10)

- Graphe résiduel :

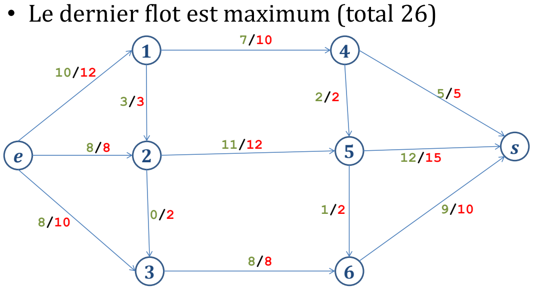


- Aucun chemin d'augmentation ⇒ Fin Algo.

Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB) 38

Exemple Général (9/10)

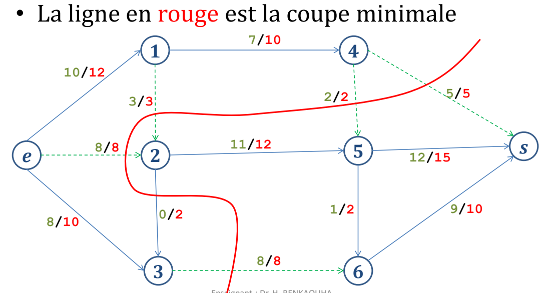
- Le dernier flot est maximum (total 26)



Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB) 39

Exemple Général (10/10)

- La ligne en rouge est la coupe minimale



Enseignant : Dr. H. BENKAOUHA
(Faculté d'Informatqiue - USTHB) 40