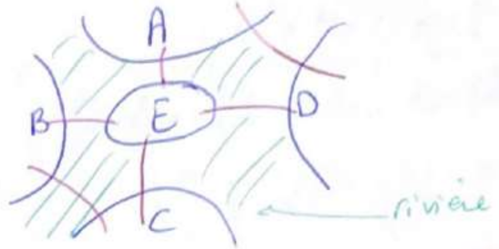
Chapitre 3 :
Theorie des graphes:



Les graphe est un outil d'utilisation qui permet de modéliser pleusiteurs situations comme par exemple

- Les villes et les ponts : ↳ lien entre les villes.
- Les villes et les autoroutes (rouge)
- Les réseaux informatiques (des pcs et des cables)
- Les réseaux de télécommunications (les antennes
- Les entrepôts et les réseaux ferroviaires
  مخازن مواقع تخزين — شبكة سكك حديدية

- un graphe est composée de
* ensemble des élements qu'on l'appelle noeud ou sommet

$X = \{A, B, C, D, E\}$ ensemble des elements (noeuds, sommets)
* ensemble des relations entre les elements qui est formée par les couples d'elements ex couple (A, B) (il ya relation entre A, B)

$U = \{(A, E), (A, D), (B, E), (B, C), (E, D), (E, C)\}$

On distingue deux types des graphes, Les graphes orientées et les graphes non orientées, les relations pour un graphe orientée s'appelle arc (x, y) origine destination (flèche)

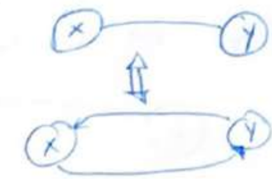
Un arc (x, y) c'est une relation qui a pour origine x vers y.

y : successeur de x ↳ parent de y
X: prédecesseur de y ↳ descendant de x

Sortie un graphe non orientée, les relation s'appellent des arrêtes (arc non orientée)

tang. dans un graphe non orientée, les relations sont symétriques $(x,y)=(y,x)$

une arrête (x,y) est équivalent à deux arc

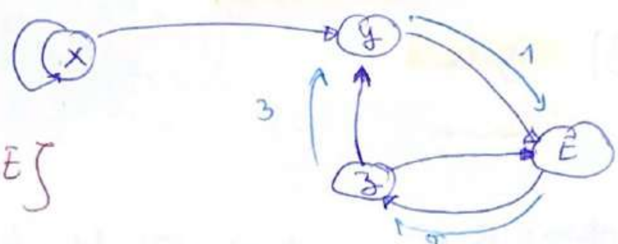


Les graphes orientées

$\vec{G}=(X,\vec{U})$: orientée

$G=(X,U)$: non orientée

exemple



$X=\{x,y,z,E\}$

$\vec{U}=\{(x,x),(x,y),(y,E),(E,z),(z,E),(z,y)\}$

$\vec{G}=(X,\vec{U})$

. à un graphe non orientée on peut lui associe un graphe orientée (العكس خاطئ)

* Vocabulaire pour graphe orientée :

- Chemin : un Chemin est une succession consécutif d'arc

exp Le chemin de x vers z : $(x,y),(y,E),(E,z)$

- Le chemin de x et E $(x,y),(y,E)$

- circuit : un circuit est un chemin qui se ferme sur lui même

① ,② ③ circuit de y vers y

② ,③ ,① " " E vers E

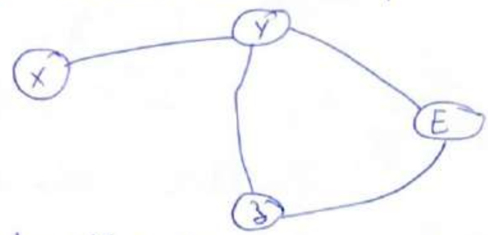
. boucle : un circuit de longueur 1 : est une boucle : x

(nbre d'arc)

- Vocabulaire pour graphe non orientée :

Chaine : est une succession consécutif des arêtes

cycle : est une chaîne qui se ferme sur lui même



$(x,y)(y,E)(E,z)$ chaîne de x vers z

$(y,E)(E,z)(z,y)$: cycle

vocabulaire des relations :

soit $u = (x,y)$

on dit x est adjascent à u

on dit que y est adjascent à u

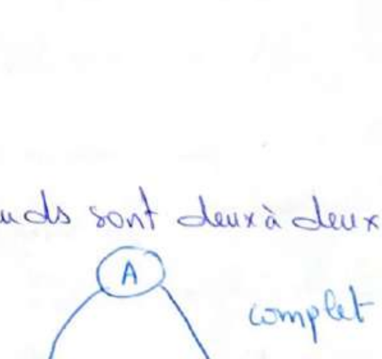
x et y adjascent à u

u est adjascent à x et y.

X et y sont adjascent

X et y sont voisins

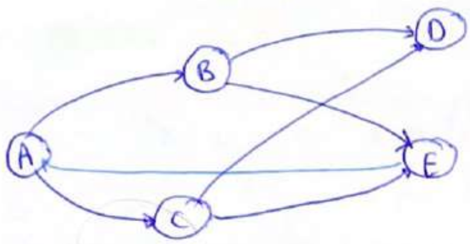
Un graphe est dit **complet** si tous les noeuds sont deux à deux adjascent. (L'exemple précédent non complet)



- Un graphe complet si $(x,y) \notin U \Rightarrow (y,x) \in U$

Définition demi-degrée

Considérons le graphe suivant



$\Gamma^{+}(x) = \{ y \in X \,/\, u = (x,y) \in U \}$: successeur de x

$= \{ y \in X \,/\, u = (y, x) \in U \}$ prédécesseur de x

S.P

$\Gamma^{+}(A) = \{B, C\}$ $\Gamma^{+}(E) = \phi$ $\Gamma^{+}(E) =$

$\Gamma^{-}(E) = \{B, C\}$

$\Gamma^{-}(A) = \phi$ entrée du graphe

$\Gamma^{+}(D) = \phi$ sortie du graphe.

$d^{+}(x) = \# \{\Gamma^{+}(x)\}$

demi degrée extérieur

$d^{+}(A) = 2$ $d^{+}(B) = 2$

$d^{+}(E) = 0$

$d^{+}(D) = 0$

$d^{-}(x) = \# \{\Gamma^{-}(x)\}$ Cardinal

demi degrée intérieur

$d^{-}(C) = 1$, $d^{-}(A) = 0$, $d^{-}(E) = 2$ $d^{-}(D) = 2$

$d(x) = d^{+}(x) + d^{-}(x)$: degré de x

$d(A) = 2$, $d(B) = 3$, $d(D) = 2$, $d(C) = 3$

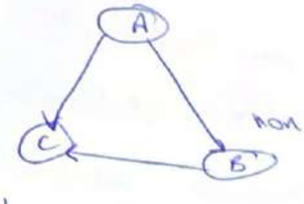
- graphe régulier :

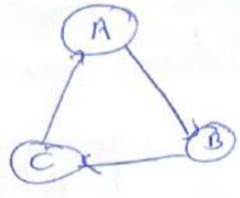
Un graphe est dit régulier si toutes ses noeuds ont le même degrée

- Connexité et forte connexité :

- On définit la connexité par une relation entre deux noeuds de la manière suivante, on dit que x et y ont une relation de connexité s'il existe une chaîne entre x et y ou $x = y$.
- on définit la forte connexité entre deux noeuds par une ... de la manière suivante : x et

x et y ont une forte connexité s'il existe une chemin de x vers y et un chemin de y vers x





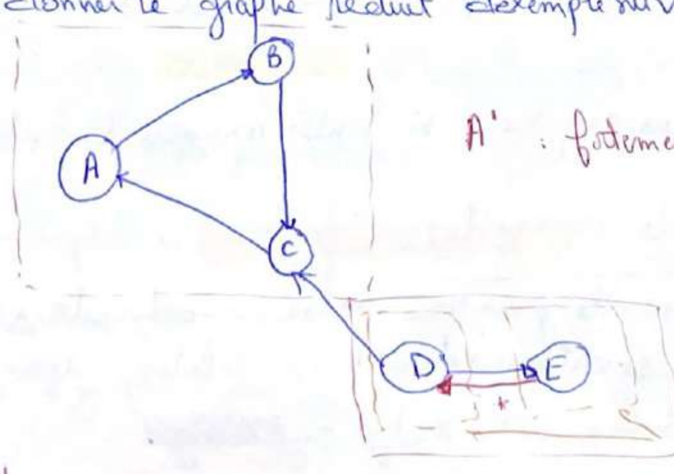
forte connexité

Composante connexe : On appelle composante connexe un ensemble de nœud d'un graphe G qui ont chemin qui relie 2 à 2. De plus toute nœud en dehors de cette composante n'a aucun relation d'adjacence avec les nœuds de la composante.
Un graphe est dit connexe si tous les nœuds forme une seule composante connexe.

graphe réduit : On appelle graphe réduit du graphe G., le graphe G' pour lequelle, chaque nœud est associée a une composante fortement connexe du G. De plus un arc relie un nœud x' à un nœud y' dans le graphe G' s'il existe un arc qui relie x a y dans G où x ∈ à la composante fortement connexe du G associée a x' (respectivement y à y')

exemple donner le graphe réduit de l'exemple suivant

exp ① :

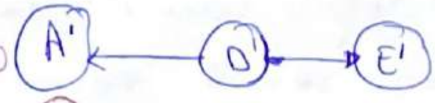


A' : fortement connexe

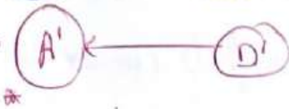
fortement connexe

Le graphe réduit :

exp①

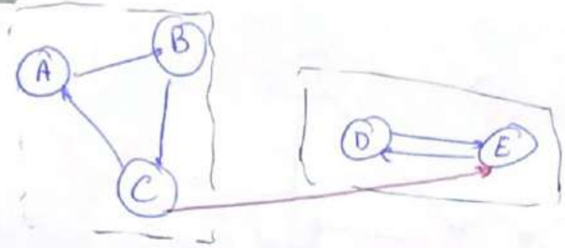


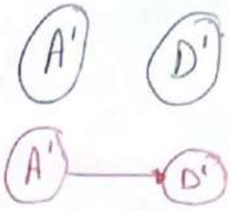
exp②



Sortie

tan





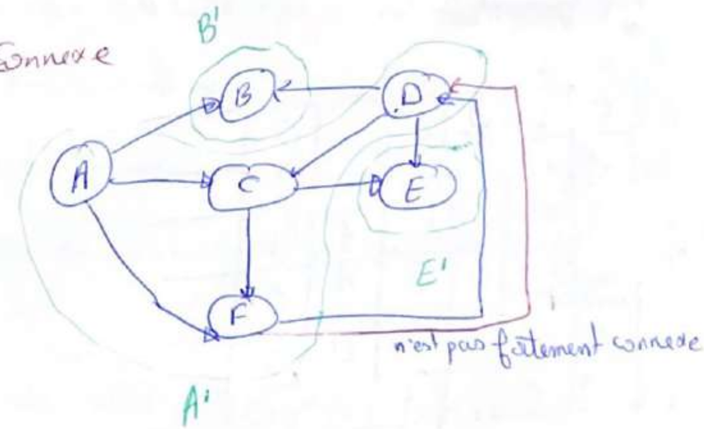
Sous graphe : soit $\begin{cases} \vec{G} = X\vec{U} \\ G = XU \end{cases}$, on appelle sous graphe de G un graphe $\begin{cases} \vec{G'} = X\vec{U'} \\ G' = XU' \end{cases}$ avec U' est obtenu à partir de U en supprimant qlq noeud de X et les arc qui sont adjacent à ces noeuds.

Graphe partielle : On appelle graphe partielle d'un graphe G un graphe G'' obtenu en supprimant quelque relation de U

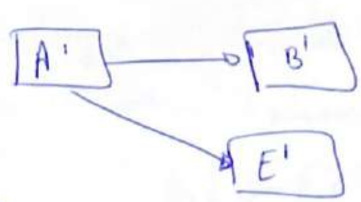
• Graphe départ : soit $\vec{G} = X\vec{U}$ un graphe, X ensemble des noeux
U : " des relation

• Un sous graphe partielle de G c'est un graphe partielle d'un sous graphe de G

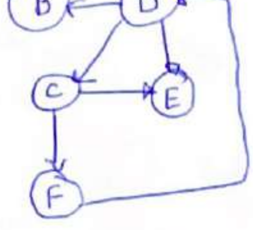
donner les composante fortement Connexe
- le graphe reduit
- sous graphe
- graphe partielle
- sous graphe partiell



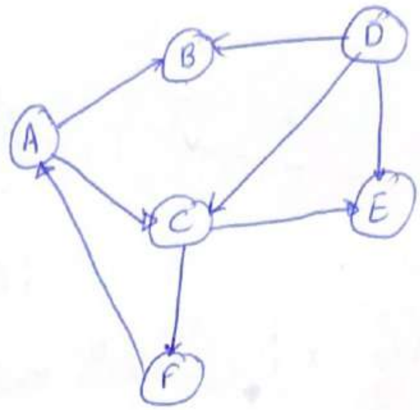
* graphe réduit :



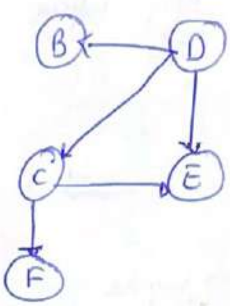
* Sous graphe



* graphe partielle :



* sous graphe partielle :



* representation informatique d'un graphe :

* Le matrice d'adjescence :

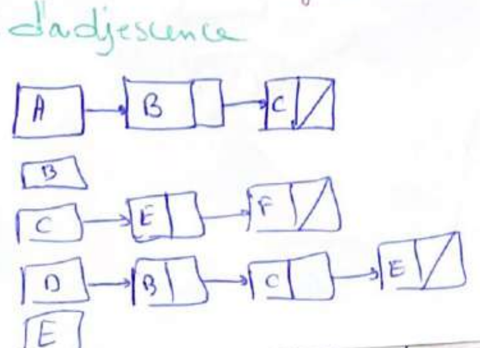
exp précedent

M =

	A	B	C	D	E	F
A	0	1	1	0	0	0
B	0	0	0	0	0	0
C	0	0	0	0	1	1
D	0	1	1	0	1	0
E	0	0	0	0	0	0
F	1	0	0	2 / 1	0	0

voir graphe si ilya 2 flèche F→D

* Liste d'adjescence



Premier algorithme :

détermination de la composante fortement connexe /connexe contenant un noeud donné

a partir d'un graphe $X\vec{u}$, donner l'algorithme qui peuvent détermin composante fortement connexe contenant a donnée.

Algorithme composante - connexe - noeud

Entrée $G = (X, U)$, $\{a\} \in X$.

Sortie $C \subset X$ ensemble des noeud

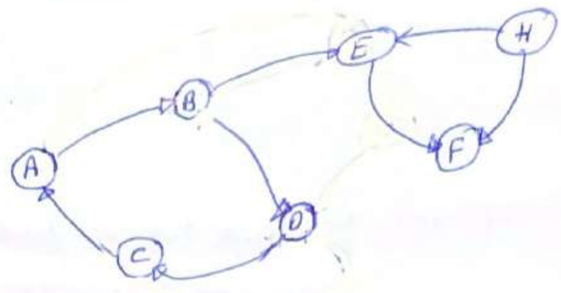
Début

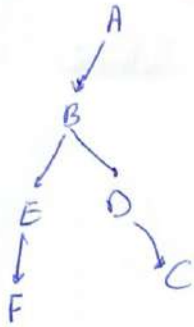
Cette algorithme recherche la composante fortement connexe d'un graphe G contenant un noeud A. l'idée de cette algo est de parcourir le graphe à partir du noeud A dans le sens directe (cad les arc) et de créer un ensemble des noeuds parcouru. la même chose est effetuée dans le sens indirecte (les arc en sens inversée) et de créer un deuxième ensemble des noeuds parcouru

La 1er ensemble regroupe les noeuds accessible à partir de A et la 2ème ensemble regroupe les noeuds a partir desquelle on peut atteindre A

L'intersection des ensemble ① et ② permet d'obtenir la composante fortement connexe contenant A

exemple





$\Rightarrow C_1 = \{A, B, C, D, E, F\}$

$C_2 = \{A, B, C, D\}$

$\Rightarrow C_1 \cap C_2 = \{A, B, C, D\}$

Algorithme composante-connexe-noeud:

Entrées $G = (X, U)$, $\{a\}$

Sortie $C \subset X$

Début

$\forall x \in X$

$[\text{examiner}(x) \leftarrow \text{faux}]$

$C_1 = \{a\}$

tanque $\exists x \in C_1 \mid \text{examiner}(x) == \text{faux}$

examiner$(x) \leftarrow$ vrai

pourtout $u = (x, y) \in U \mid y \notin C_1$

$C_1 = C_1 \cup \{y\}$

Fintanque

$\forall x \in X$

[examiner$(x) \leftarrow$ faux

$C_2 = \{a\}$

tanque $\exists x \in C_2$ | examiner $(x) ==$ faux

examiner$(x) \leftarrow$ vrai

pour tout $u \in (y, x) \in U \mid y \notin C_2$

$C_2 = C_2 \cup \{y\}$

Fintanque

$C = C_1 \cap C_2$

Retourner C.

Fin.

Algo composante - connexe

Entrée : $G = (X, U)$

Sortié : $C_1, C_2, \ldots C_n$

Début :

$i = 1$

tanque $\exists x \in X$

$C_i \leftarrow$ composante - connexe - nœud $((X, U), \{x\})$

$i \leftarrow i + 1$

$X \leftarrow X \setminus C_i$ (X - C_i privé de)

Fintanque

Retourner $C_1 - C_n$

Fin.

Exemple précedent

$X = \{A, B, C, D, E, F, H\}$

$C_1 = \{A, B, C, D\} \Rightarrow X = \{E, F, H\}$

$C_2 = \{E\} \Rightarrow X = \{F, H\}$

$C_3 = \{F\} \Rightarrow X = \{H\}$

$C_4 = \{H\} \Rightarrow X = \phi$

Les arbres

Considérons un graphe $\vec{G} = X\vec{U}$ avec n : nombre des éléments = card(X)
m : nombre des relations = card(U)

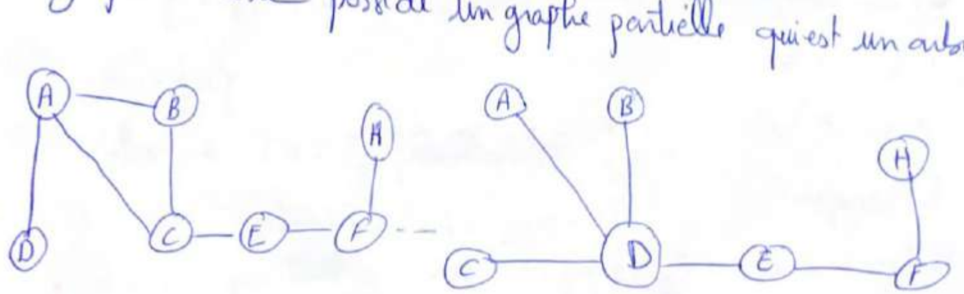
Théorème : • Si G est connexe $\Rightarrow$ $m \geq n - 1$
• Si G est sans cycle $\Rightarrow$ $m \leq n - 1$

Déf d'arbre

Un arbre est un graphe connexe sans cycle, il a donc n - 1 relations
Onpeut dire qu'un arbre est un graphe qui connecte tous les nœuds entre eux avec le minimum des relations

Rque

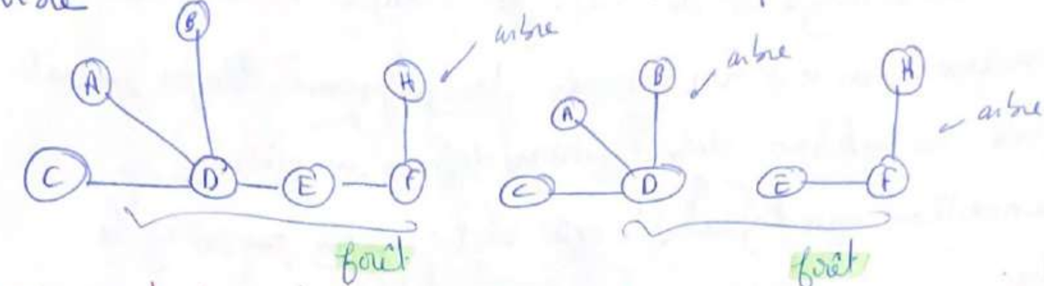
- L'ajout dau moindre relation supplémentaire dans un arbre crée un cycle
- Un graphe connexe possède un graphe partielle qui est un arbre



(exp) 8 graphe partiel = arbre

un seul composant connexe

On appelle forêt un graphe dont chaque composante connexe est un arbre



Racine et antiracine : (graphe orienté)

- Un noeud a d'un graphe G est une racine de G s'il existe un chemin joignant a à chaque noeud de G ; exp: A, B, C, D sont des racines
- Un noeud a d'un graphe G est une antiracine de G s'il existe un chemin joignant chaque noeud de G au noeud a

→ F : sont des antiracine.

- arborescence et antiarborescence
- Un graphe G est une arborescence de racine a si G est un arbre et si a est une racine
- Un graphe G est une antiarborescence de antiracine a si G est un arbre et si a antiracine

algorithme :

Dans cette partie on va associe à chaque relation une valeur (poids, longueur, débit, largeur --)

imaginons, on cherche a installer un réseau électrique qui connecte les maisons, d'où l'intérêt de construire un arbre (toutes les maisons sont interconnecté avec minimum des relations) en utilisant une longueur minimale

Algo de Kruskal

L'idée de cet algo. est de rangée les relations dans l'ordre crois des valeurs qui sont lui associés dans un premier temps, ensuite d'ajouter ces relations dans l'ordre prédéfinie un par un a condition que l'ajout de cette relation ne crée pas un cycle

algo :

Entrée $G = (X, U)$

Sortie $G' = (X, U')$

Début : $U' \leftarrow \phi$

Trie de $(U_1 \dots U_m)$ (ordre croissant) arbre coût minimal.

Pour i de 1 à m

si $(X, U' \cup \{u_i\})$ n'a pas de cycle.

$U' \leftarrow U' \cup \{u_i\}$.

Fin si

Fin pour

Retourner $G' = (X, U')$

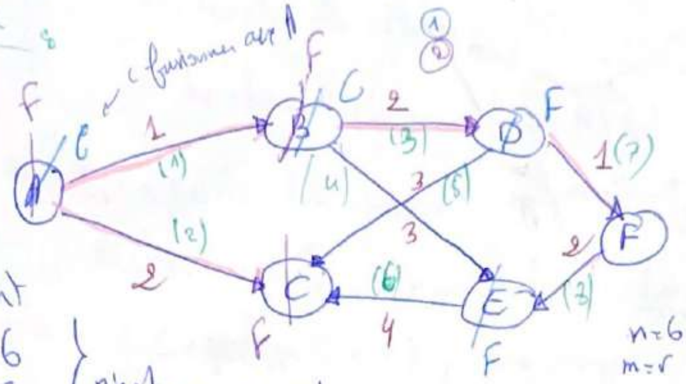
Fin

Méthode de Prim :

La méthode de Prim consiste a sélectionne un nœud x d'une manière arbitraire et de le diffusionne avec le nœud y pour former un seul nœud y est choisi telque l'arc (x, y) ou l'arc (y, x) est le coût minimal. ~~Toute nœud~~ tout arc adjascent à x ou y sera adjascent au nouvelle nœud obtenu par la fusion de x ou y

…pète ce processus jusqu'à que tous les nœuds forment un seul …end ou jusqu'à obtenir $(n-1)$ relations (10)

Exemple :



depart
$n=6$
$m=8$ } n'est pas un arbre

$n=6$ } arbre
$m=5$
rose

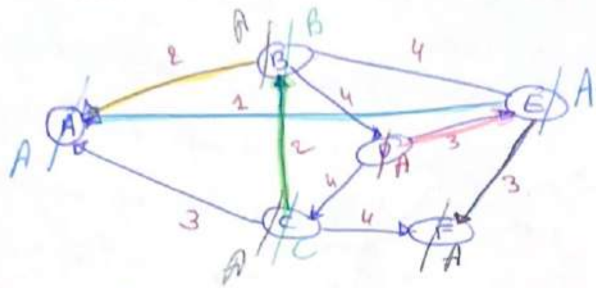
On cherche l'arbre a cout minimale, il faut garder uniquement "6 - 1 = 5" relations

Algo Prim

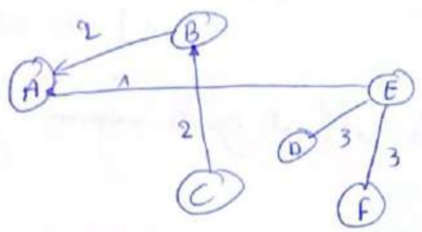
Entrées $G=(X,U)$
Sorties $G'=(X,U')$

Début
$U' \leftarrow \phi$
tanque $|U'| < n-1$ faire
 selectionner $x \in X$ au hasard
 Choisir $u \in U \mid u=(x,y)$ ou $u=(y,x)$, $y \neq x$.
 $cout(u) = \min \{ cout(v) \mid v=(x,y)$ ou $v=(y,x)$, $x \neq y \}$
 fusionner x et y
fin tanque
Fin.

Exemple



$n = 6$ } est une arbre
$m = 5$



: final

Chemin plus court dans un graphe

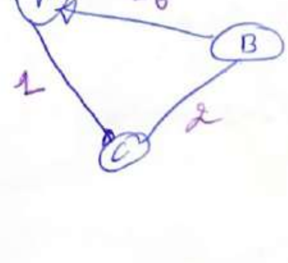
quelque déf :

réseau : un réseau est un graphe $G = (X, U)$ dans lequel on associe a chaque relation u, une mesure D_u qu'on appelle longueur

Le réseau sera notée $R = (X, U, d)$

Longueur : La longueur d'un chemin (d'une chaîne, circuit ou d'un cycle) est la somme du longueur de chaque arc qui le compose. par convention un chemin (une chaîne, un circuit ou un cycle) qui ne contient pas d'arc et de longueur nulle

circuit absorbant : un circuit est dit absorbant si sa longueur est négatif



$((A, C), (C, B), (B, A))$ est un circuit absorbant

$L(\_\_\_) = -3$

Sorti problème de plus court chemin est formée par 3 catégories de problème

tc Catégorie A : recherche de chemin plus court reliant deux nœuds données

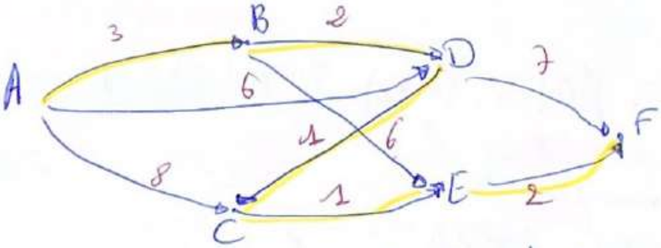
Catégorie B : se chercher les chemins les plus court reliant un nœud donnée a tous les autres nœuds des graphes.

Catégorie C : se chercher les chemins reliant chaque couple des nœuds de graphe (C enveloppe (B et A), B enveloppe A)

Théorème : Toute partie (sous-chemin) d'un chemin optimale est-elle même optimale

$\left\{ A \longrightarrow B \longrightarrow C \right.$ ($A \to C$ optimal donc $A \to B$ optimale)

Algo de Bellman (A)



$D_x(y)$: longueur de chemin reliant x et y.

$d(x,y)$: longueur d'arc $U=(x,y)$

$$D_A(F) = \min \begin{cases} \underbrace{D_A(D)}_{5} + \underbrace{d(D,F)}_{7} \\ \underbrace{D_A(E)}_{7} + \underbrace{d(E,F)}_{2} \end{cases} = 9$$

$$D_A(D) = \min \begin{cases} \underbrace{D_A(A)}_{0} + \underbrace{d(A,D)}_{6} \\ \underbrace{D_A(B)}_{3} + \underbrace{d(B,D)}_{2} \end{cases} = 5$$

$$D_A(E) = \min \begin{cases} \underbrace{D_A(C)}_{6} + \underbrace{d(C,E)}_{1} \\ \underbrace{D_A(B)}_{3} + \underbrace{d(B,E)}_{6} \end{cases} = 7$$

$$D_A(C) = \min \begin{cases} \underbrace{D_A(A)}_{0} + \underbrace{d(A,C)}_{8} \\ \underbrace{D_A(D)}_{5} + \underbrace{d(D,C)}_{1} \end{cases} = 6$$

$$D_A(B) = \min \begin{cases} D_A(A) + d(A,B) \\ d(A,B) = 3 \end{cases}$$

$$D_A(A) = 0$$

$F^{\prime}(y) = D_{x_0}(y)$; $P(y)$: Prédécesseur de y

Initialisation $F(x_0) = 0$; $\forall y \neq x_0$; $F(y) = +\infty$

pour k de 1 à $n-1$ faire
pour tout noeud y faire
$F(y) = \min \{ F^{\prime}(z) + d(z,y) \ ; z \in P(y) \}$
$F = F^{\prime}$

$k = 0$.

$F(A) = 0$, $F(B) = +\infty$, $F(C) = +\infty$, $F(D) = +\infty$, $F(E) = +\infty$, $F(F) = +\infty$

$k = 1$

$F(A) = 0$, $F(B) = 3$, $F(C) = 8$, $F(D) = 5$, $F(E) = 9$, $F(F) = 11$

$k = 2$

$F(A) = 0$, $F(B) = 3$, $F(C) = 6$, $F(D) = 5$

⋮

- Algo de Dijkstra :

L'algo de dijkstra permet de déterminer le chemin le plus court à partir d'un noeud donnée x_0 vers tout sort les autres du graphe à condition que les valeurs associée aux arcs soient positifs, l'algo est le suivant

Algo de Dijkstra

Entrée $G = (X, U)$, $\{x_0\}$.
sortie $\lambda(\ )$, $pred(\ )$
Début
$\forall x \in X$
$\{ \lambda(x) = +\infty$
$pred(x) = Null$
x non traité $\}$.
$\lambda(x_0) = 0$

sortie = faux.

tantque $\exists\, x \in X \mid x$ non traité et sortie == faux

selectionner x tq $\lambda(x) = \min(\lambda(y),\ y \in X,\ y$ non traité$)$

x traité

si $(\lambda(x) = +\infty)$ alors.

{sortie = vrai}

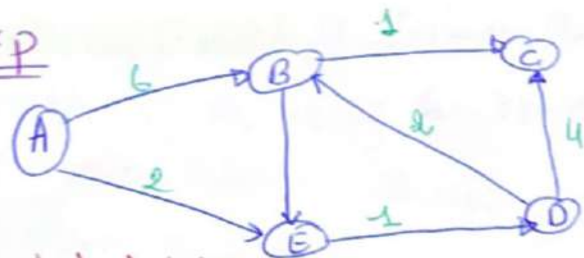
$\forall\, U = (x,y) \in U$ si $\lambda(y) > \lambda(x) + d(x,y)$

$\lambda(y) \leftarrow \lambda(x) + d(x,y)$, $pred(y) = x$

fintanque

Fin.

Exp



+Point de départ A

$\Rightarrow \lambda(A) = 0$, $pred(A) = Null$, $\lambda(B) = +\infty$, $pred(B) = Null$, $\lambda(C) = +\infty$
$pred(C) = Null$, $\lambda(D) = +\infty$, $pred(D) = Null$, $\lambda(E) = +\infty$, $pred(E) = Null$

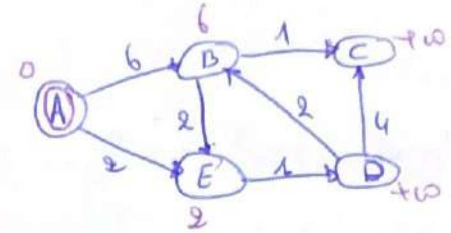
Nontraité $\{A, B, C, D, E\}$, traité $= \phi$, sortie = faux.

$\Rightarrow$ A est selectionner

Nontraité $\{B, C, D, E\}$ traité $\{A\}$, $\lambda(A) = 0$, sortie = faux,

$$\begin{cases} \lambda(B) = +\infty > \lambda(A) + d(A,B) = 0+6 \Rightarrow \lambda(B) = 6,\ pred(B) = A \\ \lambda(E) = +\infty > \lambda(A) + d(A,E) = 0+2 \Rightarrow \lambda(E) = 2,\ pred(E) = A \end{cases}$$

$\lambda(A) = 0$, $pred(A) = Null$, $\lambda(B) = 6$, $pred(B) = A$, $\lambda(C) = +\infty$, $pred(C) = Null$
$\lambda(D) = +\infty$, $pred(D) = Null$, $\lambda(E) = 2$, $pred(E) = A$



+ E sera sélectionnée

Non traité = {B, C, D}, traité = {A, E}

$\lambda(A) = 0$, $pred(A) = Null$, $\lambda(B) = 6$, $pred(B) = A$, $\lambda(C) = +\infty$, $pred(C) = Null$

$\lambda(D) = \underbrace{3}_{2+1 \leftarrow \lambda(E) + d(E,D)}$, $pred(D) = E$, $\lambda(E) = 2$, $pred(E) = A$

Sortie = faux

+ D sera sélectionnée :

Non traité = {B, C}, traité = {A, E, D}

$\lambda(A) = 0$, $pred(A) = Null$, $\lambda(B) = 5$, $pred(B) = D$, $\lambda(C) = 7$, $pred(C) = D$

$\lambda(D) = 3$, $pred(D) = E$, $\lambda(E) = 2$, $pred(E) = A$

sortie = faux.

(+ B sera sélectionnée :

Non traitée = {C}, traitée = {A, E, D, B}

$\lambda(A) = 0$, $pred(A) = Null$, $\lambda(B) = 5$, $pred(B) = D$, $\lambda(C) = 6$

$pred(C) = B$, $\lambda(D) = 3$, $pred(D) = E$, $\lambda(E) = 2$, $pred(E) = A$

Sortie = faux

* C sera sélectionnée

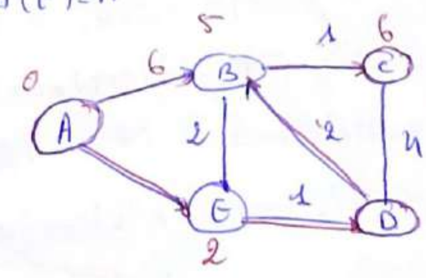
Non traitée = ϕ, traitée = X

$\lambda(A) = 0$, $pred(A) = Null$, $\lambda(B) = 5$, $pred(B) = D$, $\lambda(C) = 6$, $pred(C) = B$.

$\lambda(D) = 3$, $pred(D) = E$, $\lambda(E) = 2$, $pred(F) = A$

sortie = faux.

($\lambda = +\infty$, pas d'un chemin qui mène à ce pt à partir $X = 0$)



Exp 2

m̂ exple pt de départ D

* point de depart D

* $Nt = \{A, B, C, D, E\}$ $tR = \phi$

$\lambda(A) = +\infty$, $Pred(A) = Null$, $\lambda(B) = +\infty$, $Pred(B) = Null$, $\lambda(C) = +\infty$

$Pred(C) = Null$, $\lambda(D) = 0$, $Pred(D) = Null$, $\lambda(E) = +\infty$, $Pred(E) = Null$

sortie = faux

* D sera selectionnée

$Nt = \{A, B, C, D, E\}$ $tR = \{D\}$

$\lambda(A) = +\infty$, $Pred(A) = Null$, $\lambda(B) = 2$, $Pred(B) = D$, $\lambda(C) = 4$, $Pred(C) = D$

$\lambda(D) = 0$, $Pred(D) = Null$, $\lambda(E) = +\infty$, $Pred(E) = Null$,

* B sera selectionnée

$Nt = \{A, C, E\}$ $tR = \{B, D\}$

① $\begin{cases} \lambda(A) = +\infty,\ Pred(A) = Null,\ \lambda(B) = 2,\ Pred(B) = D,\ \lambda(C) = 3,\ Pred(C) = B \\ \lambda(D) = 0,\ Pred(D) = Null,\ \lambda(E) = 4,\ Pred(E) = D \\ \text{sortie} = \text{faux} \end{cases}$

* C sera ~~selectionnée~~ traitée

$Nt = \{A, E\}$, $tR = \{B, C, D\}$

①

sortie = faux

* E sera traité

$Nt = \{A\}$ $tR = \{B, C, D, E\}$

①

نفس sortie = faux

* A sera traitée

$Nt = \phi$, $tR = \{A, B, C, D, E\}$

①

sortie = vrai ← $\lambda(A) = +\infty$

RQ

Pour les graphes qui ont des valeurs (-) associée a ce arc, on utilise l'algo de Bellman Ford

- Algorithme de Bellman Ford :

L'algo de Bellman-Ford ne fonctionne qu'à condition circuit absorbant

on peut utiliser aussi pour détecter la puissance de circuit absorbant

Bellmanford $\begin{cases} \text{Entrée } G=(X,U),\ x_0 \\ \text{sortie } \lambda,\ pred \end{cases}$

$\forall x \in X$

$\lambda(x) = +\infty.$

$pred(x) = Null$

}

$\lambda(x_0) = 0$

Pour i de 1 à N-1 faire

$\forall u = (x,y) \in U$

si $(\lambda(y)) > \lambda(x) + d(x,y))$ alors

{ $\lambda(y) \leftarrow \lambda(x) + d(x,y)$

$pred(y) \leftarrow x$

}

Fin faire

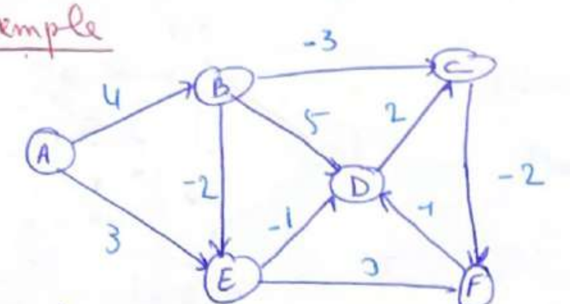
$\forall u = (x,y) \in U$

si $[\lambda(y) > \lambda(x) + d(x,y)]$ alors

Affiche " circuit absorbant "

Fin.

exemple



$n_c = 10.$

$x_0 = A$

$\lambda(y)$ = longueur du chemin qui mène dans vers y (le plus court)

Reponse

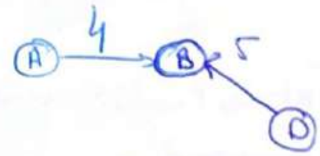
+ $i = 0$

$\lambda(A) = 0$, Pred(A) = Null, $\lambda(B) = +\infty$, Pred(B) = Null, $\lambda(C) = +\infty$, Pred(C) = Null

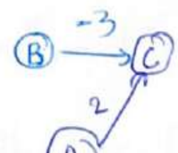
$\lambda(D) = +\infty$, Pred(D) = Null, $\lambda(E) = +\infty$, Pred(E) = Null, $\lambda(F) = +\infty$, Pred(F) = Null

+ $i = 1$

$\lambda(A) = 0$, Pred(A) = Null, $\lambda(B) = 4$, Pred(B) = A



$\lambda(C) = 1$, Pred(C) = B.



$\lambda(D) = +\infty$, Pred(D) = Null

$\lambda(E) = 2$, Pred(E) = B.

$\lambda(E) = -1$, Pred(F) = C.

$\underbrace{\lambda(B)}_{\lambda(A)} \neq +\infty > 0 + \underbrace{4}_{d(A,B)}$

$\lambda(C) = +\infty < \underbrace{4}_{\lambda(B)} + \underbrace{(-3)}_{d(B,C)}$

+ $i = 2$

$\lambda(A) = 0$, Pred(A) = Null, $\lambda(B) = 4$, pred(B) = A

$\lambda(C) = 1$, Pred(C) = B, $\lambda(D) = 0$, Pred(D) = F

$\lambda(E) = 2$, Pred(E) = B, $\lambda(F) = -1$, Pred(F) = C

+ $i = 3$

$\lambda(A) = 0$, Pred(A) = Null, $\lambda(B) = 4$, Pred(B) = A

$\lambda(C) = 1$, Pred(C) = B, $\lambda(D) = 0$, Pred(D) = F

$\lambda(E) = 2$, Pred(E) = B, $\lambda(F) = -1$, Pred(F) = C.

+ $i = 4$

$\lambda(A) = 0$, Pred(A) = Null, $\lambda(B) = 4$, Pred(B) = A

$\lambda(C) = 1$, Pred(C) = B, $\lambda(D) = 0$, Pred(D) = F

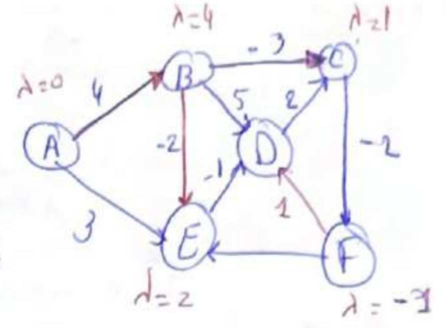
$\lambda(E) = 2$, Pred(E) = B, $\lambda(F) = -1$, Pred(F) = C

$i = 5$

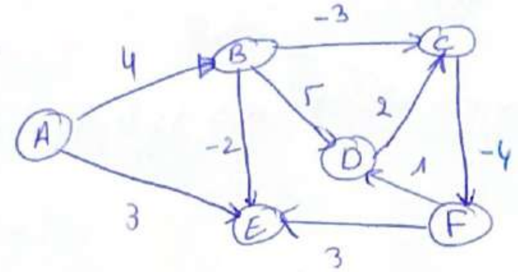
$\lambda(A)=0$, $Pred(A)=Null$, $\lambda(B)=4$, $Pred(B)=A$

$\lambda(C)=1$, $Pred(C)=B$, $\lambda(D)=0$, $Pred(D)=F$

$\lambda(E)=2$, $Pred(E)=B$, $\lambda(F)=-1$, $Pred(F)=C$



Exemple



$i = 0$

$\lambda(A)=0$, $Pred(A)=Null$, $\lambda(B)=+\infty$ $Pred(B)=Null$, $\lambda(C)=+\infty$, $Pred(C)=Null$

$\lambda(D)=+\infty$, $Pred(D)=Null$, $\lambda(E)=+\infty$, $Pred(E)=Null$, $\lambda(F)=+\infty$, $Pred(F)=Null$

$i = 1$

$\lambda(A)=0$, $Pred(A)=Null$, $\lambda(B)=4$, $Pred(B)=A$, $\lambda(C)=1$, $Pred(B)=B$

$\lambda(D)=+\infty$ $Pred(B)=Null$, $\lambda(E)=2$, $Pred(B)=3$, $\lambda(F)=-3$, $Pred(B)=C$

$i = 2$

$\lambda(A)=0$, $Pred(A)=Null$, $\lambda(B)=4$, $Pred(B)=A$, $\lambda(C)=1$, $Pred(C)=B$

$\lambda(D)=-2$, $Pred(D)=F$, $\lambda(E)=0$, $Pred(E)=F$, $\lambda(F)=-3$, $Pred(F)=C$

$i = 3$

$\lambda(A)=0$, $Pred(A)=Null$, $\lambda(B)=4$, $Pred(B)=A$, $\lambda(C)=1$, $Pred(C)=B$, $\lambda(D)=-2$

~~λB(4)~~

$Pred(D)=$ ~~E~~, $\lambda(E)=0$ $Pred(E)=F$, $\lambda(F)=-3$, $Pred(F)=C$

$i = 4$

$\lambda(A)=0$, $Pred(A)=Null$, $\lambda(B)=4$, $Pred(B)=A$, $\lambda(C)=1$, $Pred(C)=B$ ~~B~~

$\lambda(D)=-2$, $Pred(D)=F$, $\lambda(E)=0$, $Pred(E)=F$, $\lambda(F)=-3$ $Pred(F)=C$

$i = 5$

Dantzig:

L'algorithme de Dantzig permet de calculer le chemin le plus court entre chaque couple de noeud du graphe a condition d'absence de cycle absorbant. L'algorithme de Dantzig permet de retourner une matrice D telque $D_{(ij)}$ c'est la longueur le plus court entre le noeud x_i et x_j. Pour commencer on doit ranger les x de 1 à n.

L'algo est le suivant :

Entrée $G = (X, U)$
Sortie D (...)
Début

```
    Pour i de 1 à N faire
    Pour j de 1 à N faire
    | D(i,j) ← +∞
    |  si (i == j) alors
    |  | D(i,j) ← 0
    |  Fin si
    Fin
    Fin
```