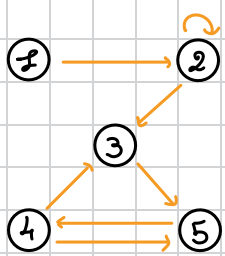
Fiche

Théorie des Graphes

• Matrice d'adjacence

→ On met 1 dans la case (i,j) si "i visite j".
→ Symétrique si graphe non-orienté



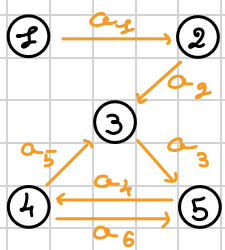
	1	2	3	4	5	d^{0+}
1	0	1	0	0	0	1
2	0	1	1	0	0	2
3	0	0	0	0	1	1
4	0	0	1	0	1	2
5	0	0	0	1	0	1
d^{0-}	0	2	2	1	2	

Degrés : $d^0 = d^{0-} + d^{0+}$

⚠ Pour un graphe non-orienté, s'il y a boucle (1 sur la diagonale de la matrice), alors on ajoute 2 au degré.

• Matrice d'incidence

→ Lignes = sommets ; Colonnes = arêtes
→ Pour graphe orienté : +1 si arc sortant ; -1 si arc entrant
→ Pour graphe non-orienté : 1 si arête simple ; 2 si boucle

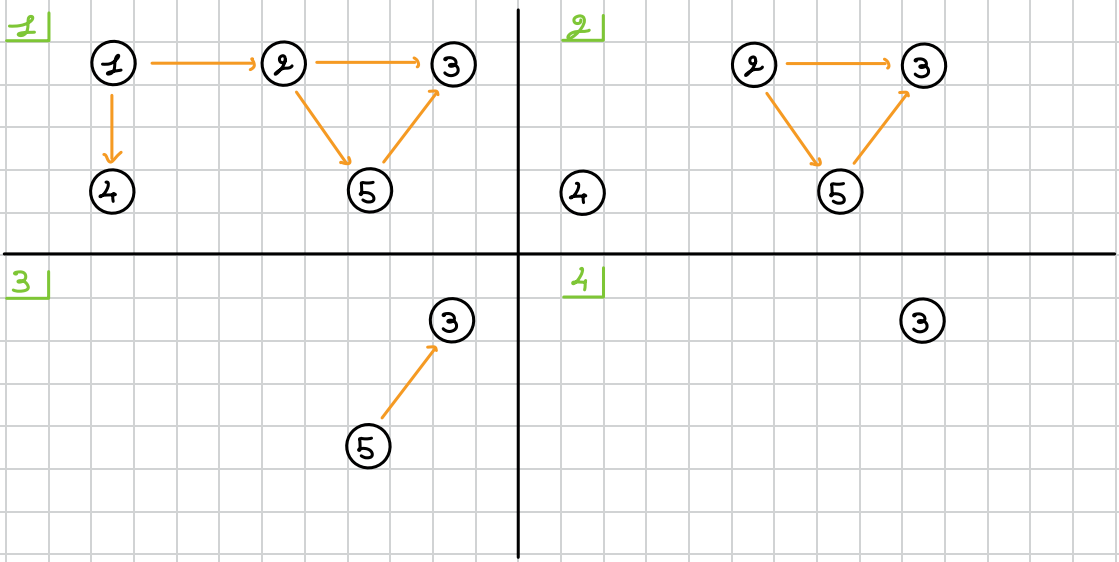


	a_1	a_2	a_3	a_4	a_5	a_6
1	+1	0	0	0	0	0
2	-1	+1	0	0	0	0
3	0	-1	+1	0	-1	0
4	0	0	0	-1	+1	+1
5	0	0	-1	+1	0	-1

Rq : Chiant pour les boucles !

• Détection de circuit

→ Supprimer tous les arcs sans prédécesseur tant que c'est possible
→ Si plus de sommet à la fin, alors pas de circuit.

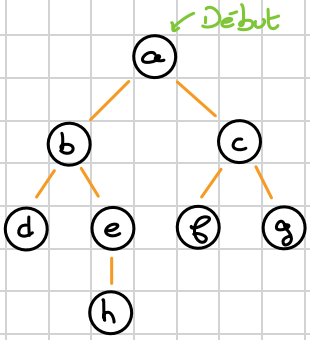


• Méthode de Roy-Warshall

→ Trouver la matrice d'adjacence de la fermeture transitive
→ On initialise la fermeture transitive par la matrice d'adjacence
→ Pour chaque sommet, on note ses prédécesseurs y et ses successeurs z. On met à jour la fermeture transitive en affectant 1 à tous les coefficients (y, z) possibles.
→ On répète cela jusqu'au dernier sommet.

Consultez cette page pour un exemple complet.

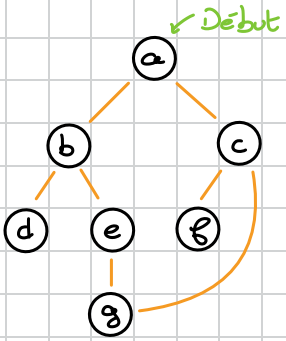
• Parcours en largeur et en profondeur



• Largeur : "Lire chaque niveau horizontal"

Visités	À étudier
a	b, c ← voisins de b
a, b	c, d, e ← voisins de c
a, b, c	d, e, f, g
a, b, c, d	e, f, g
...	...

À la fin :
a, b, c, d, e, f, g, h

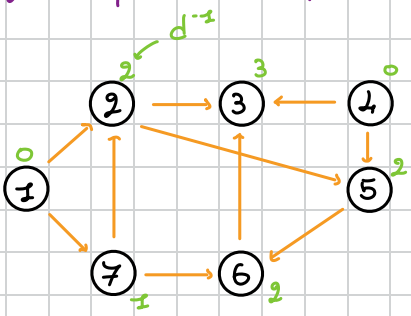


• **Profondeur** : "Lire jusqu'en bas chaque chemin"
On pose que la gauche est "prioritaire".

a → b → d → plus de successeur
On remonte à b..
b → e → g → c → f → plus de successeur
On a tout exploré donc...
Parcours : a, b, d, e, g, c, f

• Attribution des rangs

→ On calcule les demi-degrés intérieurs de chaque sommet. On attribue le rang $k = 0$ aux sommets dont $d^{-1} = 0$.
→ Pour chaque itération, on diminue de 1 le d^{-1} de chaque successeur des sommets ayant $d^{-1} = 0$.
→ On supprime un sommet dès qu'il a $d^{-1} = 0$. Puis on recommence en considérant les nouveaux sommets avec $d^{-1} = 0$ tant qu'il en reste.



$k = 0$ · $S_0 = \{1; 4\}$

$r(1) = 0$
$\Gamma(1) = \{2; 7\}$
$d^{-1}(2) = 1$
$d^{-1}(7) = 0$

$r(4) = 0$
$\Gamma(4) = \{3; 5\}$
$d^{-1}(3) = 2$
$d^{-1}(5) = 1$

$k = 1$ · $S_1 = \{7\}$

$r(7) = 1$
$\Gamma(7) = \{2, 6\}$
$d^{-1}(2) = 0$
$d^{-1}(6) = 1$

$k = 2$: $S_2 = \{2\}$

$r(2) = 2$
$\Gamma(2) = \{3, 5\}$
$d^{-1}(3) = 1$
$d^{-1}(5) = 0$

$k = 3$: $S_3 = \{5\}$

$r(5) = 3$
$\Gamma(5) = \{6\}$
$d^{-1}(6) = 0$

$k = 4$: $S_4 = \{6\}$

$\Gamma(6) = 4$

$\Gamma(6) = \{3\}$

$d^{-1}(3) = 0$

$k = 5$: $S_5 = \{3\}$

$\Gamma(3) = 5$

$\Gamma(3) = \varnothing$

FIN

• Algorithme de Dijkstra

⚠ Valable si les coûts des arcs sont tous positifs ou nuls

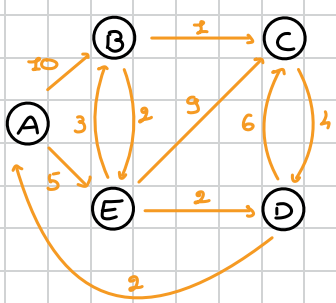
→ On commence par mettre le sommet d'origine dans la première ligne CC.

→ Dans cette même ligne, on écrit la distance pour les villes accessibles depuis la source.

→ On choisit la distance la plus petite sur cette ligne et on rajoute le sommet associé aux CC de la ligne suivante

→ Sur cette deuxième ligne, on met à jour les distances des villes accessibles depuis ce nouveau sommet.

⚠ En mettant à jour une distance, on additionne le coût de l'arête entre source et sommet avec la distance déjà attribuée à la source (ligne au-dessus).



CC	A	B	C	D	E
A	0	10_A	∞	∞	5_A
AE	•	8_E	14_E	7_E	•
ADE	•	8_E	13_D	•	•
ABDE	•	•	9_B	•	•
ABCDE	•	•	•	•	•

It. 2 : $B = \min(10_A ; 5 + 3_E) = 8_E$

$C = \min(\infty ; 5 + 9_E) = 14_E$

$D = \min(\infty ; 5 + 2_E) = 7_E$

It. 3 : $C = \min(14_E ; 7 + 6_D)$

$= 13_D$

It. 4 : $C = \min(13_D ; 8 + 1_B)$

$= 9_B$

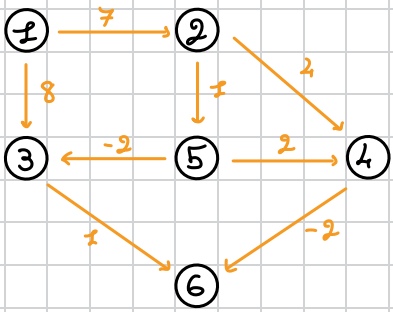
• Algorithme de Bellman ✓ OK pour arcs négatifs !

⚠ Valable pour les graphes sans circuit absorbant

→ On note 1 le sommet source et $\pi^k(i)$ la longueur courante du plus court chemin entre 1 et i après k itérations.

→ À chaque itération, on regarde les successeurs des sommets précédemment étudiés (sommet 1 à k = 1) et on met à jour leur $\pi^k(i)$ en additionnant le poids de l'arête avec le $\pi^{k-1}(p)$ du prédécesseur p.

→ Si on a deux itérations identiques, alors on s'arrête. Si après n itérations on observe un / des changement(s), alors il y a un circuit absorbant → pas de court chemin valide.



Init : $\pi^1(1) = 0$

k = 1 : $\Gamma(1) = \{2;3\}$

$\pi^1(2) = 7_1$

$\pi^1(3) = 8_1$

k = 2 : $\Gamma(2,3) = \{4;5;6\}$

$\pi^2(4) = 7+4_2 = 11_2$

$\pi^2(5) = 7+1_2 = 8_2$

$\pi^2(6) = 8+1_3 = 9_3$

	1	2	3	4	5	6
k = 1	0_1	7_1	8_1	∞	∞	∞
k = 2	0_1	7_1	8_1	11_2	8_2	9_3
k = 3	0_1	7_1	6_5	10_5	8_2	9_{3ou4}
k = 4	0_1	7_1	6_5	10_5	8_2	7_3
k = 5	0_1	7_1	6_5	10_5	8_2	7_3

k = 3 : $\Gamma(4;5;6) = \{3;4;6\}$

$\pi^3(3) = \min(8_1; 8-2_5) = 6_5$

$\pi^3(4) = \min(11_2; 8+2_5) = 10_5$

$\pi^3(6) = \min(9_3; 11-2_4) = 9_{3ou4}$

k = 4 : $\Gamma(3;4;6) = \{6\}$

$\pi^4(6) = \min(9_{3ou4}; 6+1_3; 10-2_4) = 7_3$

k = 5 : $\Gamma(6) = \varnothing$

FIN

• Graphes eulériens et semi-eulériens

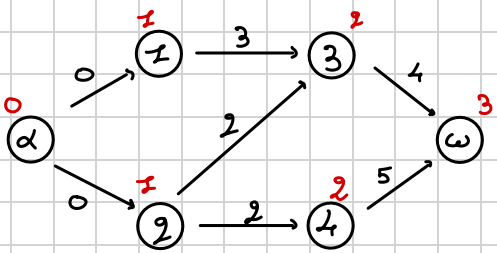
→ Utilisés dans les problèmes de la forme "ce parcours existe-t-il ?"

① Graphe eulérien : Tous les sommets sont incidents à un nombre pair d'arêtes
→ On peut parcourir le graphe depuis n'importe quel sommet et emprunter exactement 1 fois chaque arête pour revenir au sommet de départ

② Graphe semi-eulérien : Il existe au plus 2 sommets incidents à un nombre impair d'arêtes
→ On prend comme points de départ et arrivée du parcours ceux incidents à un nombre impair d'arêtes. On parcourt une unique fois toutes les arêtes.

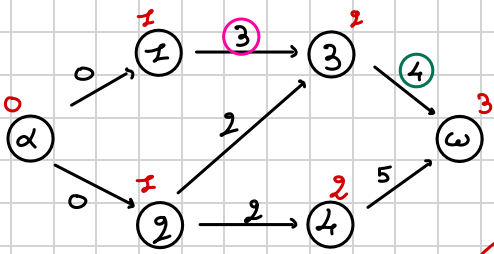
• Algorithme d'ordonnancement

→ Ajouter une tâche de début α, ses successeurs sont les tâches sans contrainte. Ajouter une tâche de fin ω, ses prédécesseurs sont les tâches n'apparaissant pas dans les contraintes.
→ Calculer le rang de chaque sommet et faire le tri topologique.



Rang	0	1	1	2	2	3
Tâche	α	1	2	3	4	ω

→ En parcourant dans l'ordre croissant, calculer les dates au plus tôt par prédécesseur pour chaque sommet. On met 0 pour α, puis pour les suivants on réalise le calcul illustré ici :



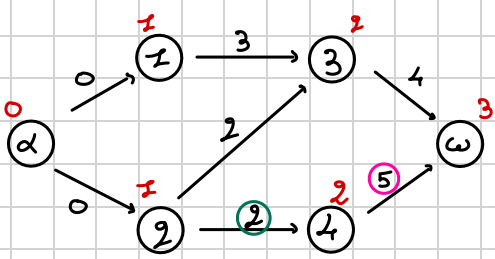
On garde le max !

durée = 3

Rang	0	1	1	2	2	3
Tâche	α	1	2	3	4	ω
Pred.	-	α	α	1,2	2	3,4
Dates pred.	0	0_α	0_α	$3_1, 2_2$	2_2	$7_3, 7_4$
Dates + tôt	0	0_α	0_α	3_1	2_2	$7_{3 \text{ ou } 4}$

Pour le préd. 3 : 3 + 4 = 7

→ En parcourant dans l'ordre décroissant, on calcule les dates au plus tard par tâche. On met ω à sa date au plus tôt (= durée totale du projet). Puis on réalise le calcul illustré ici :



On garde le min !

durée = 5

Rang	0	1	1	2	2	3
Tâche	α	1	2	3	4	ω
Pred.	-	α	α	1,2	2	3,4
Dates pred.	0	0_α	0_α	$3_1, 2_2$	2_2	$7_3, 7_4$
Dates + tôt	0	0_α	0_α	3_1	2_2	$7_{3 \text{ ou } 4}$
Succ.	1,2	3	3,4	ω	ω	-
Dates succ.	$0_2, 0_2$	0_3	$1_3, 0_4$	3_ω	2_ω	7
Dates + tard	$0_{2 \text{ ou } 2}$	0_3	0_4	3_ω	2_ω	7

2 - 2 = 0 ; 7 - 5 = 2

→ On calcule les marges totales : (date + tard) - (date + tôt)

Rang	0	1	1	2	2	3
Tâche	α	1	2	3	4	ω
Marge totale	0	0	0	0	0	0

→ On détermine le(s) chemin(s) critique(s). Un sommet i est critique si sa marge totale est de 0 et que :

$$(i) \xrightarrow{a} (j) \qquad (\text{date} + \text{tôt})(j) = (\text{date} + \text{tôt})(i) + a$$

Rang	0	1	1	2	2	3
Tâche	α	1	2	3	4	ω
Pred.	-	α	α	1,2	2	3,4
Dates pred.	0	0_α	0_α	$3_1, 2_2$	2_2	$7_3, 7_4$
Dates + tôt	0	0_α	0_α	3_1 ✗	2_2	$7_{3 \text{ ou } 4}$
Succ.	1,2	3	3,4	ω	ω	-
Dates succ.	$0_1, 0_2$	0_3	$1_3, 0_4$	3_ω	2_ω	7
Dates + tard	$0_{1 \text{ ou } 2}$	0_3	0_4	3_ω	2_ω	7
Marge totale	0	0	0	0	0	0

Chemins critiques :

α → 1 → 3 → ω ✓

α → 2 → 3 → ω ✗

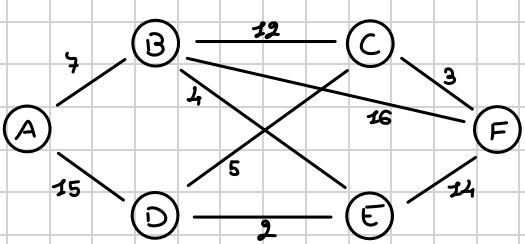
α → 2 → 4 → ω ✓

Le 2e chemin est faux car $0 + 2 \neq 3$

(0 + 2 : date + tôt de ② ; 3 : date + tôt de ③ ; 2 : durée de ②)

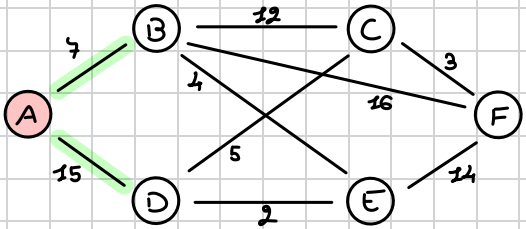
• Algorithme de Prim

→ On choisit un sommet dans le graphe G, on le place dans l'ensemble CC, et on regarde ses successeurs.
→ Parmi les arêtes, on prend celle de poids minimal et on la place dans T (notre arbre) et on met l'autre sommet associé dans CC.
→ On regarde les arêtes adjacentes aux sommets de CC et on sélectionne celle de poids le plus faible à condition qu'elle ne relit pas 2 sommets déjà dans CC.
→ On continue tant que tous les sommets ne sont pas dans CC.

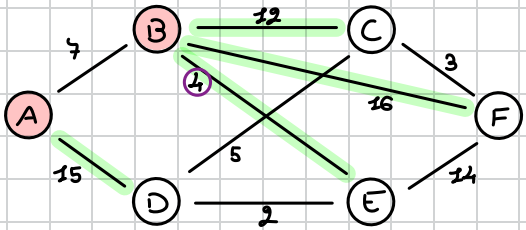


1. CC = {A}
T = ∅

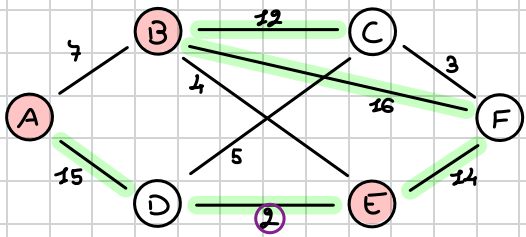
Arêtes possibles en vert



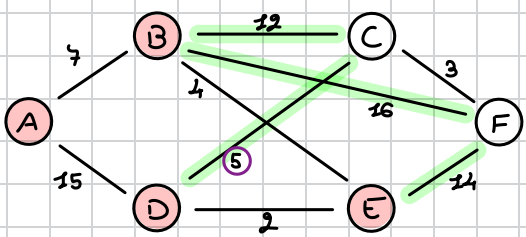
2. $CC = \{A, B\}$
$|(A,B)| = 7 \leq |(A,D)| = 15$
$T = \{(A,B)\}$



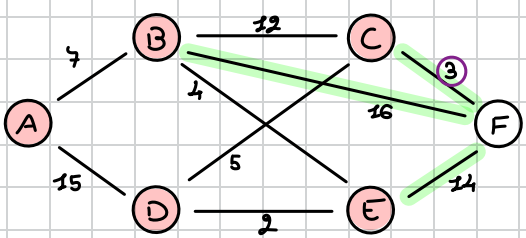
3. $CC = \{A, B, E\}$
$T = \{(A,B), (B,E)\}$



4. $CC = \{A, B, E, D\}$
$T = \{(A,B), (B,E), (E,D)\}$



5. $CC = \{A, B, E, D, C\}$
$T = \{(A,B), (B,E), (E,D), (D,C)\}$



6. $CC = \{A, B, E, D, C, F\}$
$T = \{(A,B), (B,E), (E,D), (D,C), (C,F)\}$

- **Coloration**

Consultez cette page pour une fiche complète.