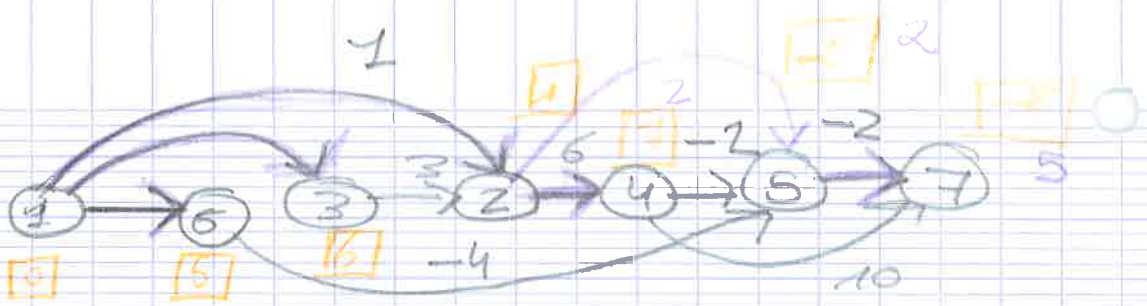
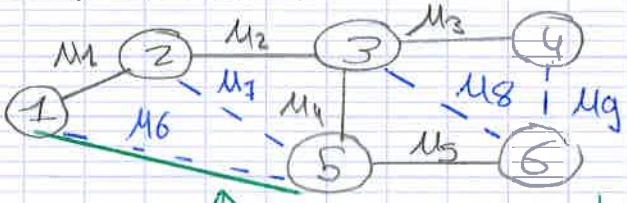


18/11/2022

II.2. Arbres :

Arbre couvrant de coût minimal (diapo 42)

Soit l'arbre couvrant :



on associe un poids w_i à chaque arrête u_i

↑ si on ajoute une arrête ⇒ on perd la notion d'arbre car cycle

On note $U_S = \{u_1; \dots; u_5\}$ le spanning tree

⇒ c'est un arbre couvrant car U_S définit un graphe connexe et comportant N-1 arrêtes

Propriété 1 : Dans un arbre couvrant, ajouter une arrête provoque la création d'un cycle.

cycle créé par ajout de u_6

Dans notre cas, ajouter u_6 ⇒ créer un cycle $= \{u_1; u_2; u_4; u_6\} = \mu_6$

Propriété 2 : Si on supprime dans ce même cycle 1 arrête, on obtient de nouveau un arbre.

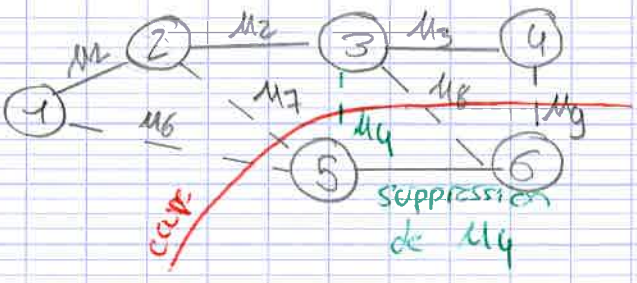
$U_{\bar{S}} = \{u_6; u_7; u_8; u_9\}$: l'ensemble des arrêtes non créées (en pointillés bleus). Ainsi, $S + \bar{S} = U$

Condition suffisante d'optimalité :

Si dans l'arbre, on ajoute une arrête, alors cette dernière a forcément le poids le plus fort du cycle qu'elle engendre (sinon on a tout intérêt à la supprimer).

Propriété duale : Supprimer une arrête u_5 de l'arbre, on crée deux composantes connexes, ce qui engendre un co-cycle.

Exemple : Enlever u_4 ⇒ cocycle $\mu_4 = \{u_6; u_7; u_4; u_8; u_9\}$ dans notre cas



u_4 coupe créant un co-cycle représentant l'ensemble des arrêtes que l'on peut choisir pour recréer un arbre

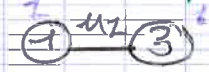
Si on ajoute une arrête de ce cocycle dans U_S on crée de nouveau un arbre couvrant.
Si u_6 avait un poids plus faible que u_4, on aurait eu tout intérêt à remplacer u_4.

Exemple d'algorithme de Kruskal (diapo 43)

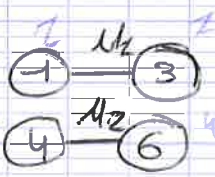
Etapes :

1) On indice les arêtes par coût ↑

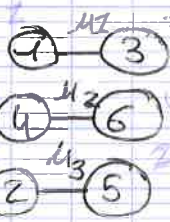
2) $k=1$:

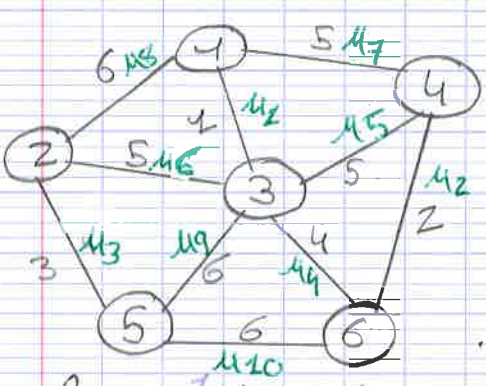


$k=2$:

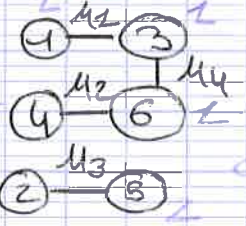


$k=3$

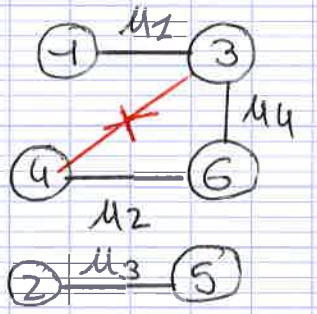




$k=4$:

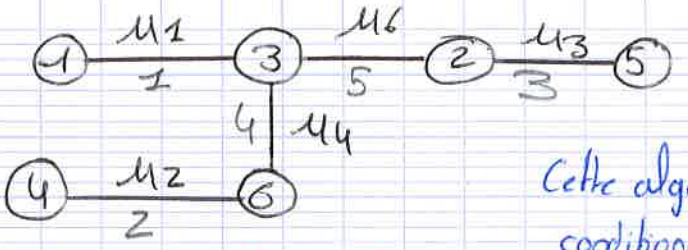


$k=5$



L'ajout de u_5 crée un cycle $\{u_2, u_4, u_5\}$
=> on rejette u_5

$k=6$:



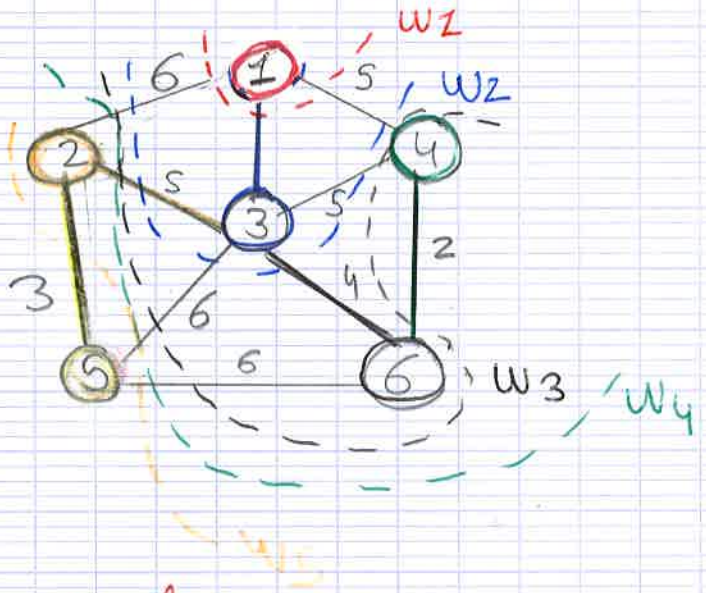
Cette algo exploite la condition suffisante d'optimalité

On obtient le coût optimal $W^*(U_S) = 15$

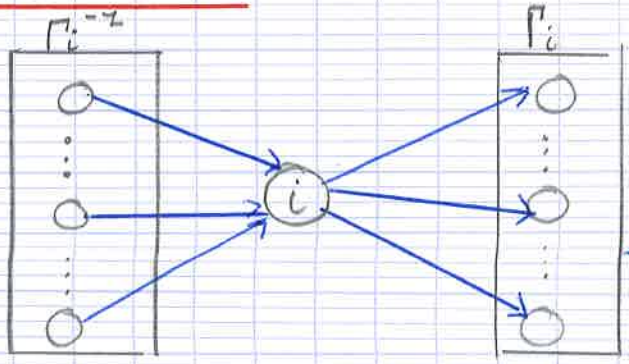
Remarque :

En pratique, on repère la création d'un cycle de manière algorithmique en associant à chaque sommet un identifiant correspondant à l'indice de plus petit sommet de la composante noté cc.
Si cet identifiant est différent alors on peut connecter deux CC.
=> algorithme moins complexe qu'une recherche en profondeur pour la recherche mathématique de cycle.

Exemple d'algorithme de Prim (diapo 44)



IV. 1. Généralités :



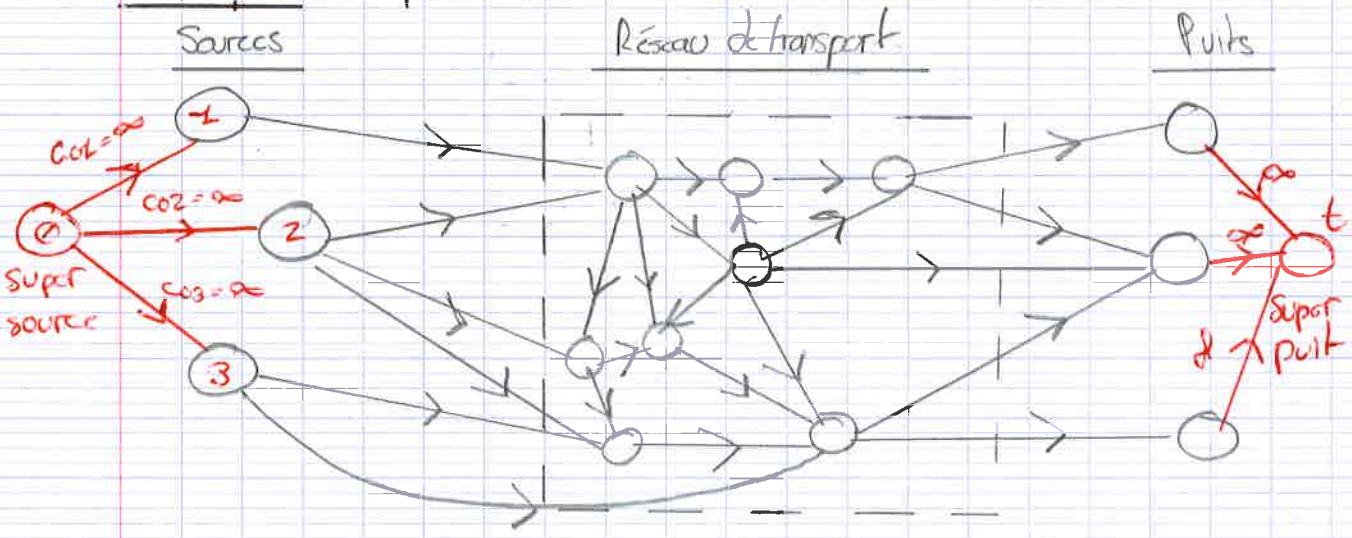
Flot f_{pt} =

$\sum_{j \in \Gamma_i^{-1}} f_{ji} = \sum_{j \in \Gamma_i} f_{ij}$

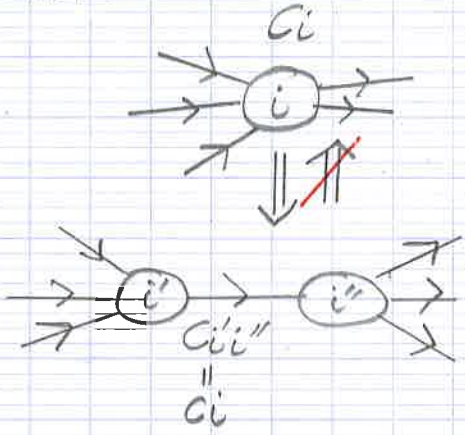
et

$f_{ij} \leq c_{ij} \quad \forall (i,j) \in U$

Remarque : (diapo. 47)

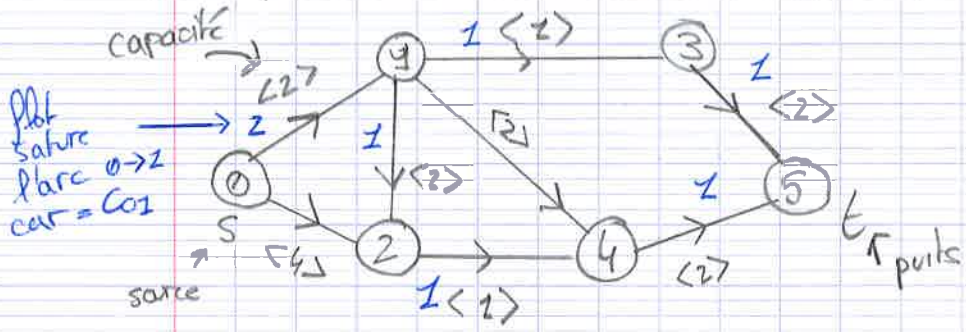


Transformation :



IV 2. Algorithme de Ford et Fulkerson

Exemple de chaîne améliorantes (diapo 50)



Prenons un flot initial $\varphi^{(0)}$ (en bleu) de valeur 2, il se répartit ensuite dans le graphe. Ce flot est saturé, il est impossible d'augmenter sa valeur même en passant par l'arc 0->2 et (2) n'a qu'un arc

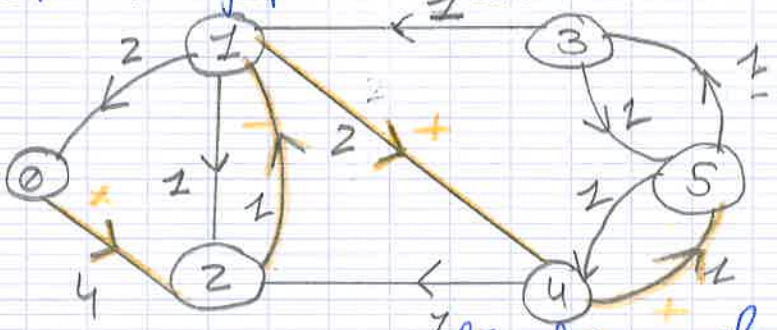
sortant déjà saturé.

Solution de Ford-Fulkerson :

↳ prouver que ce flot est optimal

ou ↳ trouver un autre flot de valeur supérieure

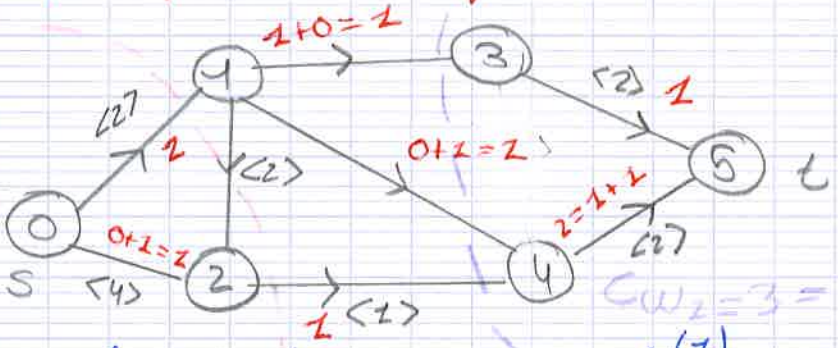
$G_e(\varphi^{(0)})$ graphe en considérant les saturations engendrées par $\varphi^{(0)}$



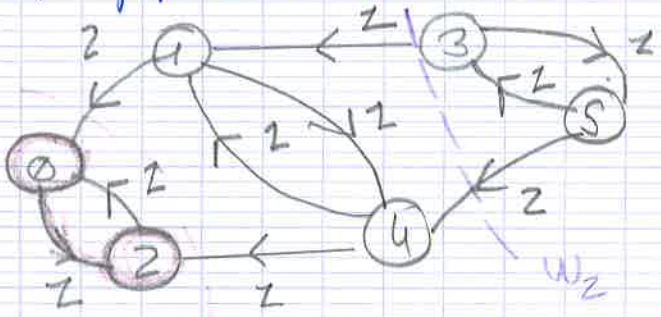
Il existe un chemin entre la source ⓪ et ⑤ appelé chaîne améliorante.

On note | + un arc avant (sens d'origine)
| − un arc arrière (sens inverse à l'original)

On peut définir un nouveau flot $\varphi^{(1)}$ de valeur 3



Ainsi, le graphe écart associé à $\varphi^{(1)}$



NB : Il peut y avoir plusieurs coupes minimales de valeur 3