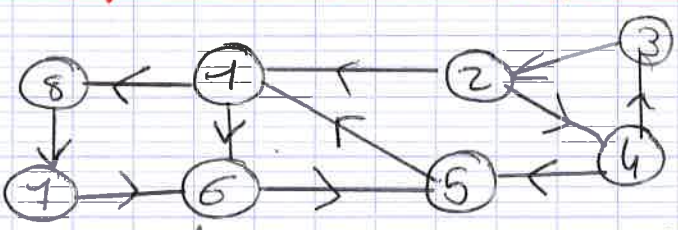
les sommets (5) ; (6) et (7)

Caractéristiques d'un algorithme de parcours en profondeur (p. 22)

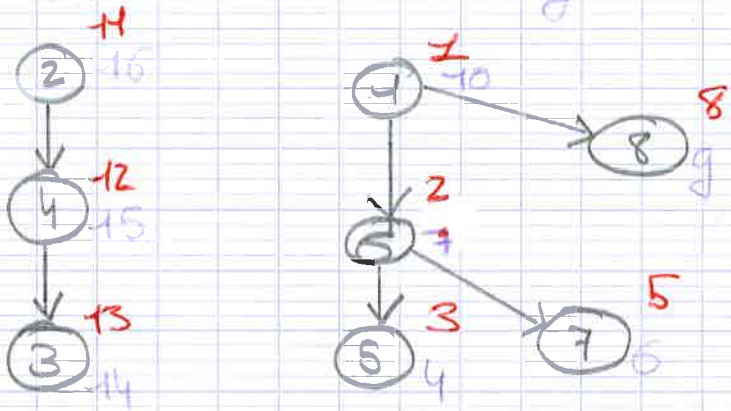
Exemple :

Données :



Consigne : Appliquer un algorithme de DFS (Depth First Search) DFS(6) sur l'arbre suivant.

Solution : Notons le sommet (i) d / f, on déroule l'algorithme ainsi



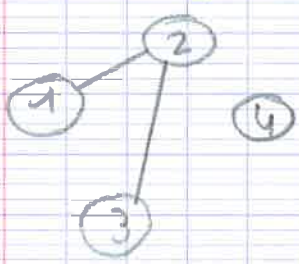
Rq : le sommet (7) est déjà connu

16/11/2022

II. Parcours de graphes et problèmes de chemin

Connexité (diapo. 24)

Matrice d'adjacence : rempli 1 s'il existe une arrête entre les sommets



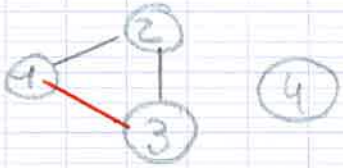
$$M = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

matrice carrée (et symétrique pour un graphe non orienté)

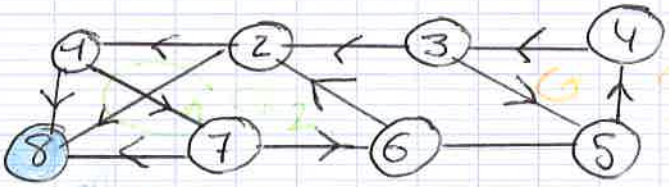
On appelle **Fermeture Transitive** le parcours de la matrice tel que

```
Pour i = 1 à N
    Pour j = 1 à N
        Pour k = 1 à N
            si M[i,k] = M[k,j] = 1 alors
                M[i,j] ← 1
```

Ici, $M = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$

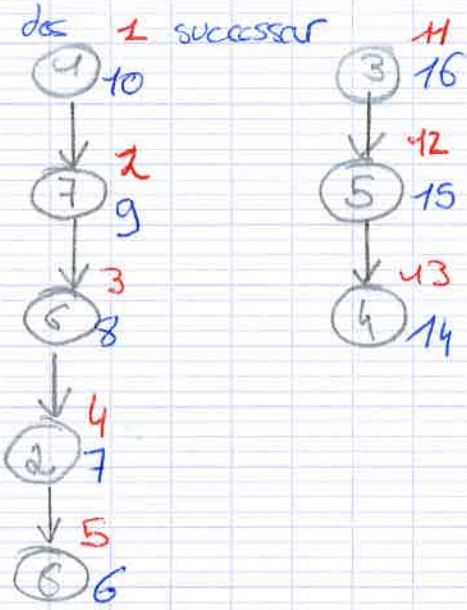


Exemple : Soit le graphe suivant



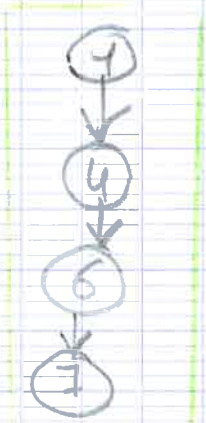
Etape

1. DFS (G, 1) parcours en profondeur depuis le sommet 1 avec des 1 successeur



2. DFS (G, 3) partant du sommet ayant le plus grand nombre post fixe : On considère la relation de voisinage prédécesseurs

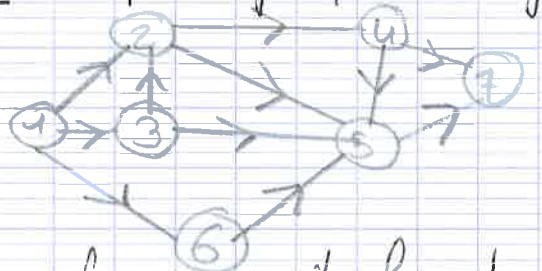
3. DFS (G; 7) vue prédécesseurs



4. DFS (G; 8) vue prédécesseurs

Tri topologique :

Exemple : Graphe acyclique (sans cycle)



Pour un graphe sans circuit, le nombre de cfc est égal au nombre de sommets.

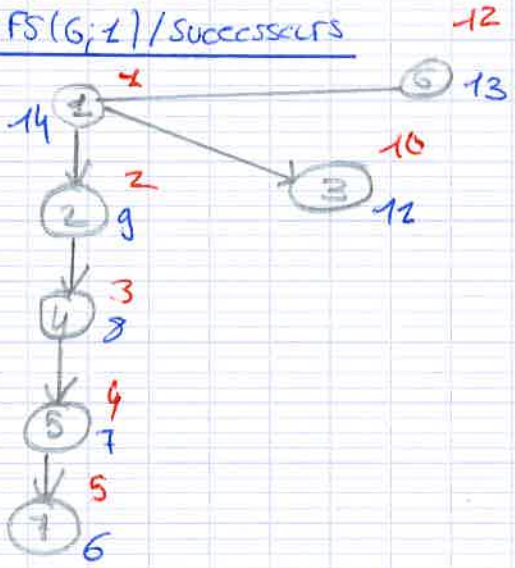
Tri topologique : Trouver un chemin selon des contraintes d'ordre de

de parcours.

Algorithme

Etapes :

1. DFS(G; 1)/Successeurs

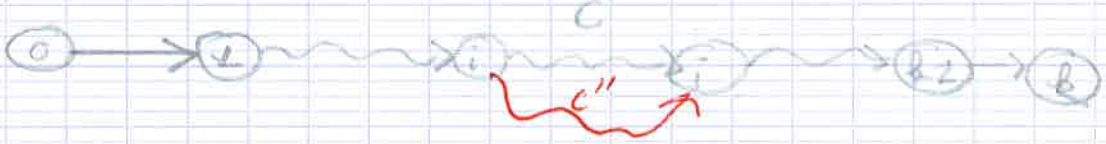


2. Tri

On a un parcours de gauche à droite sans retour => tri valide

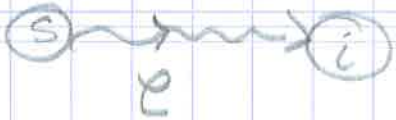
III Plus courts chemins

Généralités (p. 28)



$l_{c''} \geq l_{c'}$ est la longueur des deux chemins.

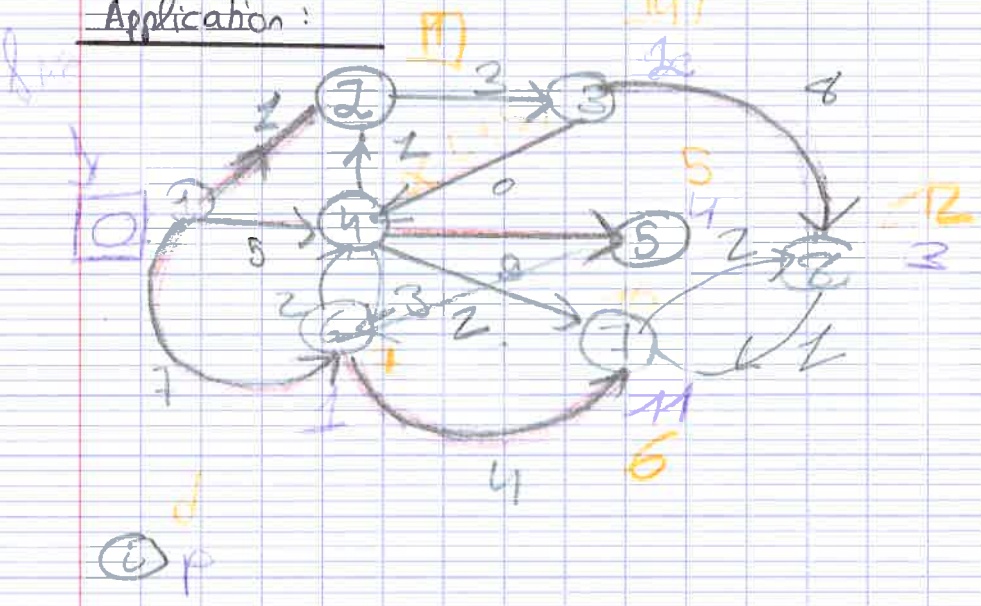
Algorithme de Dijkstra :



étiquette : $L_e \geq l_{eg^*}$

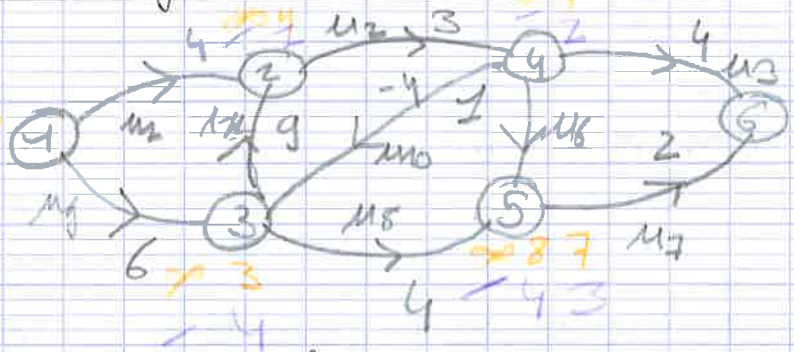
avec e^* = chemin court

Application :



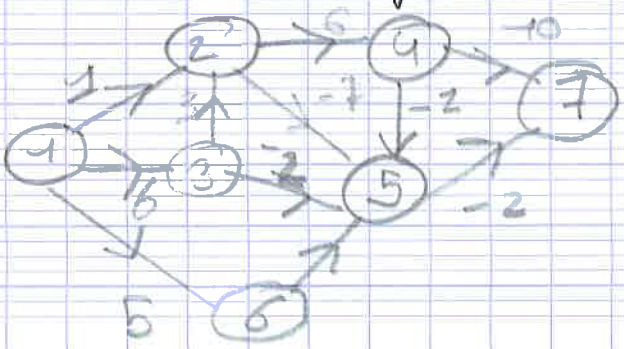
On a une arborescence des plus courts chemins selon Dijkstra

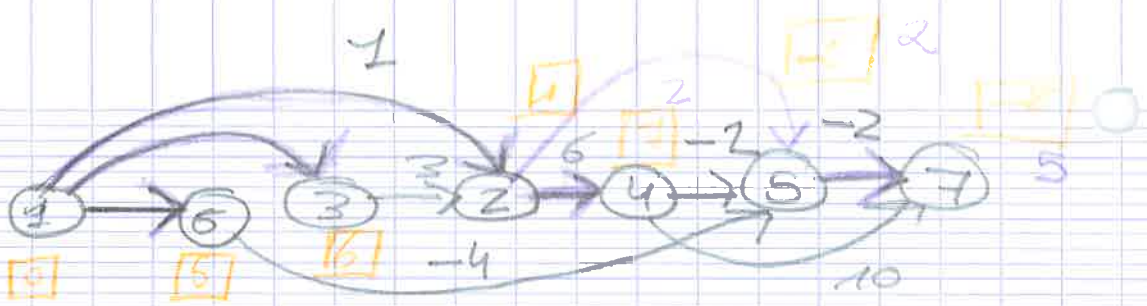
Exemple : Algorithme de Bellman :



N.B : Une fois le graphe stable, on peut s'arrêter. Il n'est pas nécessaire d'aller jusqu'à $|V|-1$. C'est à dire qu'il n'y a plus aucune actualisation possible

Exemple dans le cas des graphes sans circuits :



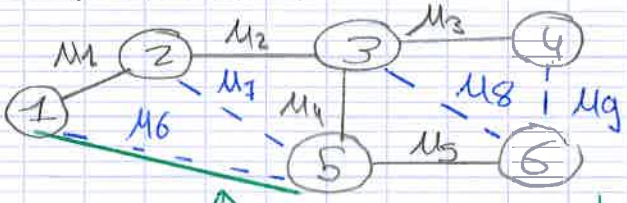


18/11/2022

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