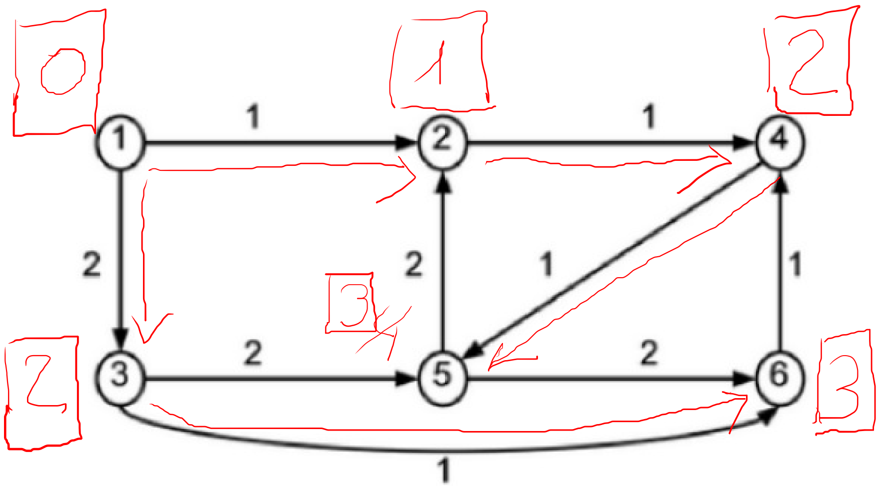
Exercice 12 – Parcours en largeur et algorithme de Dijkstra

Q 12.1 Dérouler l'algorithme de Dijkstra pour déterminer l'arborescence des plus courts chemins depuis le sommet 1 dans le graphe $G = (S, A, c)$ orienté de la figure 1 (on représentera l'arborescence partielle des plus courts chemins à chaque étape de l'algorithme). En cas d'égalité, on examinera en priorité le sommet de plus petit indice.

Forêt couvrante en rouge

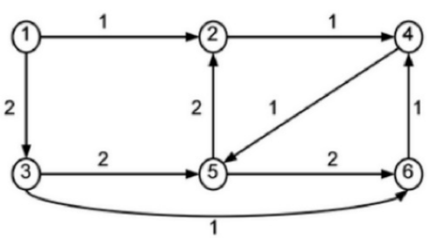


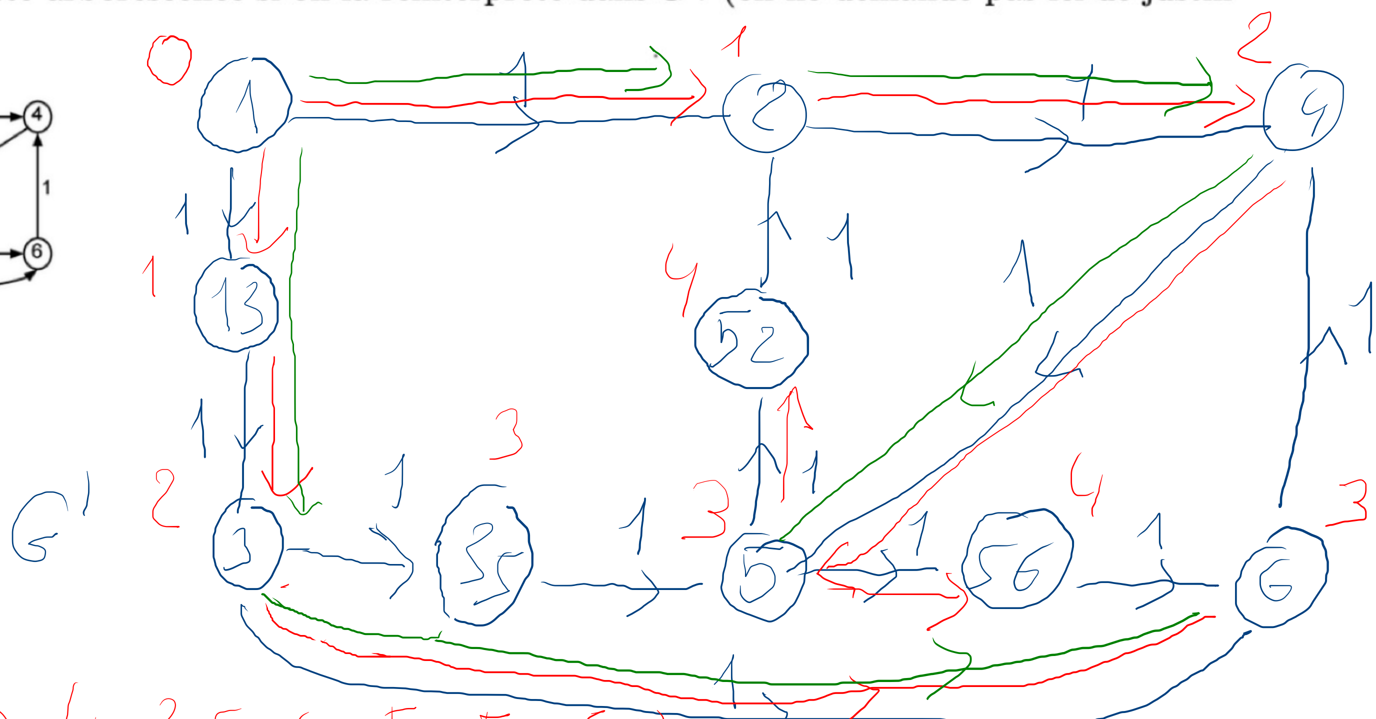
"Parcours" Dijkstra (1, 2, 3, 4, 5, 6)

On définit un nouveau graphe $G' = (S \cup S', A'_1 \cup A'_2, c')$ à partir de G, où :

- S' comporte un sommet x_a pour chaque arc $a = (x, y)$ de A pour lequel $c(a) = 2$,
- $A'_1 = \{a \in A : c(a) = 1\}$,
- A'_2 comporte deux arcs (x, x_a) et (x_a, y) pour chaque arc $a = (x, y)$ de A pour lequel $c(a) = 2$,
- c' est une fonction de coût unitaire (qui associe un coût de 1 à chaque arc de $A'_1 \cup A'_2$).

Q 12.2 Représenter G' ainsi que l'arborescence couvrante associée à un parcours en largeur de G' depuis le sommet 1.
A quoi correspond cette arborescence si on la réinterprète dans G ? (on ne demande pas ici de justification)



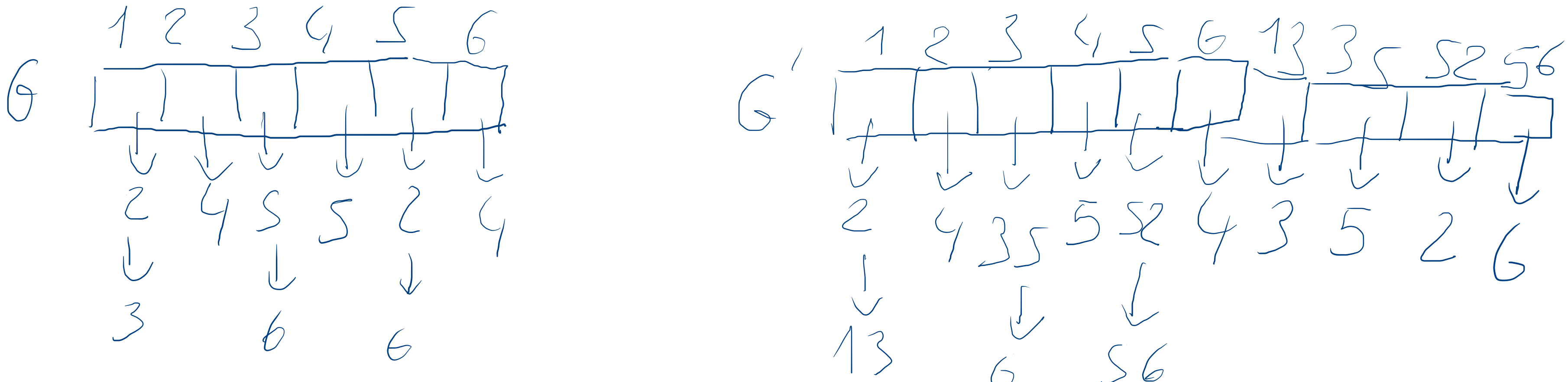


(1, 13, 2, 3, 4, 35, 6, 5, 52, 56)

L'arborescence "verte" est l'arborescence des + courts chemins,

Q 12.3 On suppose ici que le graphe est représenté sous forme de tableau de listes de successeurs. Pour un graphe G avec des coûts 1 et 2, quelle est la complexité de l'algorithme consistant à construire un nouveau graphe G' à partir de G puis à réaliser un parcours en largeur de G' ? Rappeler la complexité de l'algorithme de Dijkstra. Conclusion ?

<u>Complexité de construction de G'</u> : $O(n+m)$



Pour chaque (x,y) de G, si le coût de (x,y) est unitaire, alors la m̂ cellule est créé dans le tableau de listes de succ de G', sinon un nouveau sommet z est créé dans G', une cellule pour (x,z) et une autre (z,y).

Parcours de G' : $O(|S| + |S'| + |A'_1| + |A'_2|)$

$$O\big(\underbrace{|S|}_{= n} + \underbrace{|S'|}_{\leq m} + \underbrace{|A'_1|}_{\leq m} + \underbrace{|A'_2|}_{\leq 2m}\big)$$

(au pire tous les arcs sont de coût 2)

$\rightarrow O(n + 4m)$

Or, $n + 4m \leq 4(n+m)$

$\rightarrow \boxed{O(n+m)}$

$d(s_1) = 0$; ouvrir(s_1); $O(1)$

Pour tout k de 2 à n faire

$d(s_k) = +\infty$; $O(n)$

FinPour

Pour k de 1 à n faire

Soit x un sommet ouvert tel que d(x) est minimum ; $O(n)$

Examiner(x); $O\left(\sum_x (1+d^+(x))\log n\right) = O((n+m)\log n)$

FinPour.

Examiner(x) dans Dijkstra : $O((1+d^+(x))\log n)$

Pour tout successeur y de x faire

Si d(y) > d(x)+c(x,y) $O(d^+(x))$

alors d(y)= d(x)+c(x,y); (complexité cumulée)

père(y)=x; $O(d^+(x))$

Si y n'est pas ouvert ouvrir(y); $O(\log n)$ à chaque itération $\rightarrow O(d^+(x)\log n)$

FinSi

FinPour

fermer(x); $O(\log n)$

$\boxed{O((n+m)\log n)}$

$O(d^+(x)\log n + \log n)$

$= O((1+d^+(x))\log n)$

Si les coûts des arcs sont bornés par une constante, on voit qu'il est plus intéressant de procéder par duplication des arcs puis parcours en largeur, du point de vue de la complexité temporelle.

NÉANMOINS si les coûts des arcs ne sont pas bornés par une constante, alors l'algorithme de Dijkstra est de meilleure complexité.

EN PRATIQUE, toujours utiliser l'algorithme de Dijkstra.