Chemins les plus courts depuis une source

On suppose un graphe simple sans boucle, avec une fonction de poids définie sur les arcs/arêtes.

La longueur d'un chemin (cas orienté) ou d'une chaîne (cas non orienté) est la somme des poids de ses arcs/arêtes

$$C = s_0, a_1, s_1, a_2, s_2, \ldots, a_n, s_n$$

$$L(C) = \sum_{i=1}^{n} p(a_i).$$

On choisit un sommet r, et pour $s \in S$ on souhaite trouver ~~le~~ chemin/la chaîne de longueur minimum de r à s ; cette longueur sera la distance de r à s. S'il n'existe pas de tel chemin, cette distance est posée $= +\infty$.

Exemples : 1) sur le réseau routier, chaque tronçon a un poids qui peut être :

- le prix du péage
- le nombre de kilomètres
- le temps de parcours à la vitesse maximum autorisée.

Trouver le chemin le | moins cher / plus court / plus rapide | entre deux villes

2) Dans un réseau informatique, une

connexion entre deux noeuds a un poids qui peut être :

— le coût de transmission par MB

— le temps de transmission par MB

Il faut trouver le chemin le moins coûteux ou le plus rapide entre deux sites.

Dans le cas où tous les poids d'arcs /arêtes valent 1, le problème a été résolu par le parcours en largeur.

On va généraliser celui-ci. On définit pour tout sommet s :

d(s) : estimation de la longueur du plus court chemin de r à s, = cette longueur à la fin de l'algo.

père(s) : prédécesseur de s dans ce chemin.

Ce chemin le plus court de r à s peut-il avoir un circuit ?

Soit $q_0, a_1, q_1, a_2, \ldots, a_n, q_n$ un circuit (resp. cycle) dans un chemin (resp. une chaîne) de r vers s.

— Si ce circuit est de longueur > 0 : $\sum_{i=1}^{n} p(a_i) > 0$, alors en l'enlevant, on raccourcit le chemin : donc le chemin le plus court n'a pas de circuit de longueur > 0

– Si sa longueur $=0$: $\sum_{i=1}^{n} p(a_i)=0$, alors en l'enlevant la longueur du chemin ne change pas.

On pose par convention qu'on prend le chemin le plus court sans circuit de longueur 0, qui est redondant.

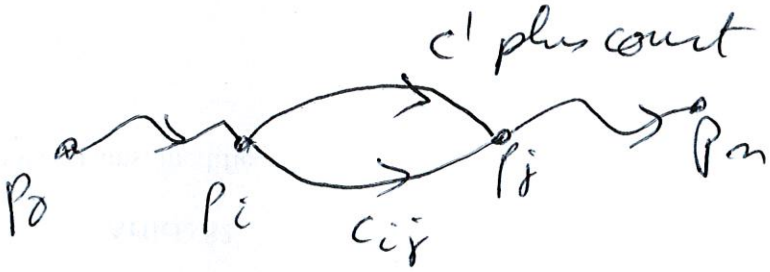
– Si sa longueur est <0 : $\sum_{i=1}^{n} p(a_i)<0$

Alors en répétant ce circuit on diminue chaque fois la longueur du chemin. Il n'y a alors pas de chemin le plus court.

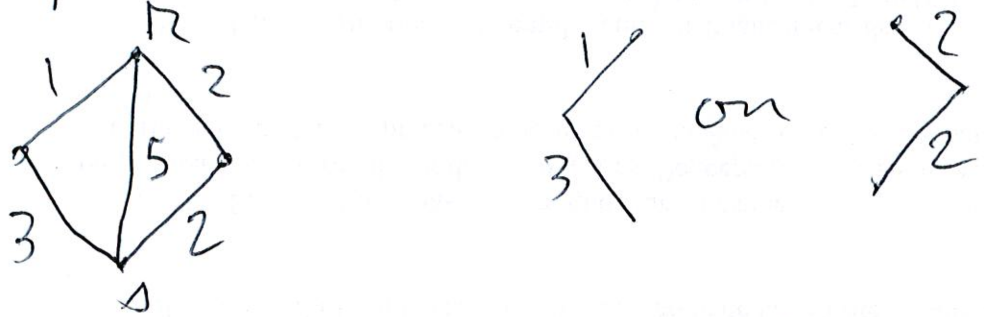
Donc on peut supposer que les chemins les plus courts, s'il existent, n'ont pas de circuit.

<u>Propriété</u>. Si $C=p_0, a_1, p_1, \dots, a_n, p_n$ est un chemin le plus court entre p_0 et p_n, alors pour $0 \leq i \leq j \leq n$, $p_i, \dots, a_j, p_j = C_{ij}$ est un chemin le plus court entre p_i et p_j.

Car s'il y en avait un autre C' plus court de p_i à p_j, en remplaçant C_{ij} par C' dans C, on aurait un chemin plus court entre p_0 et p_n

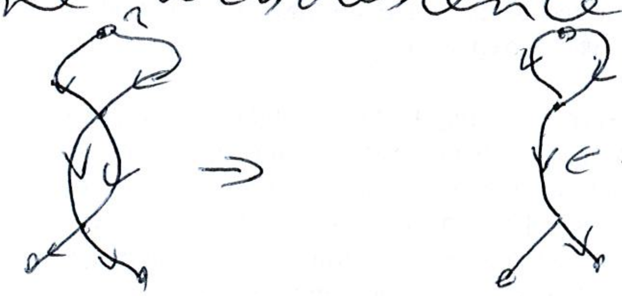


On peut avoir plusieurs chemins les plus courts entre 2 sommets



On en choisit un des deux.

Les chemins les plus courts de r aux sommets $s \in S$ joignables forment une arborescence.



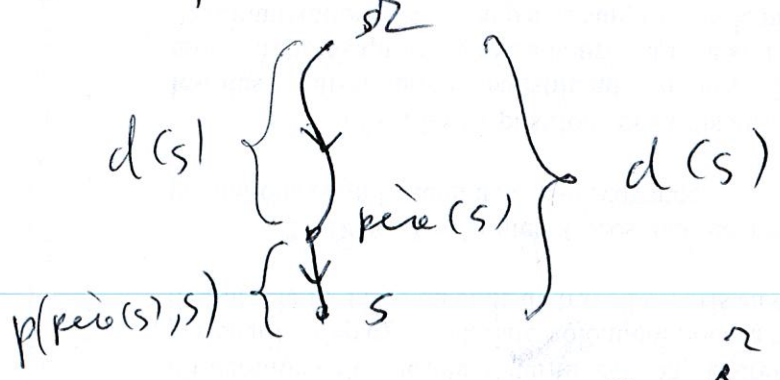
← le plus court des 2, si de même longueur, choisir un des 2

impossible : dans l'un $d(p_i) < d(p_j)$
dans l'autre $d(p_i) > d(p_j)$

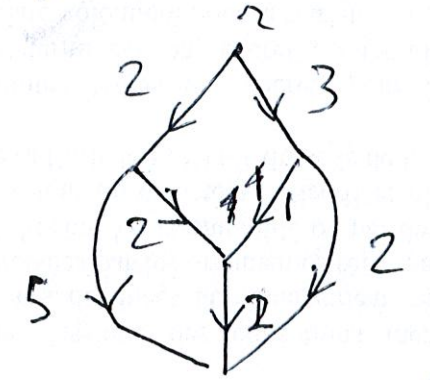
$d(x)$ = distance de r à x

Dans cette arborescence, tout sommet s a un père noté père(s).

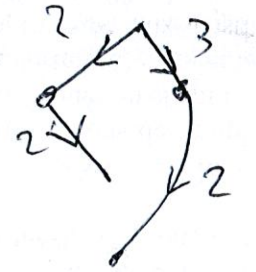
La distance $d(x)$ de r à x doit satisfaire $d(s) = d(\text{père}(s)) + p(\text{père}(s), s)$



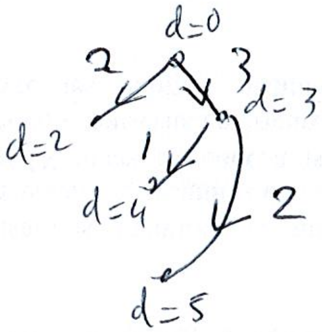
Exemple



donne



ou



On va au départ surestimer $d(s)$, puis on va successivement réduire sa valeur par la "relaxation"

Initialisation(r) $d(r) := 0$;

$\forall s \in S \setminus \{r\}$, $d(s) := \infty$;

$\forall s \in S$, $\text{père}(s) := NIL$;

Relaxation(u, v) $(u, v \in S)$

Si $d(v) > d(u) + p(u,v)$

Alors | $d(v) := d(u) + p(u,v)$
| $père(v) := u$

Explication : on avait un chemin de longueur $d(v)$ de r à v et un de longueur $d(u)$ de r à u. Si $d(v) > d(u) + p(u,v)$, on voit qu'en prenant le chemin de r à u puis l'arc (u,v), on obtient un chemin plus court de r à v : on choisit donc celui-ci à la place du précédent.

Les algorithmes pour calculer l'arborescence des plus courts chemins depuis r utilisent l'initialisation, puis répètent la relaxation dans un certain ordre, jusqu'à aboutir au plus courts chemins.

Si le graphe a un ~~cycle~~ circuit de longueur < 0 ~~partie~~, l'algorithme ne peut pas aboutir.

Soit ~~cycle~~ circuit $s_0, a_1, s_1, \ldots, a_n, s_n = s_0$

avec $\sum_{i=1}^{n} p(a_i) < 0$

Comme $\sum_{i=1}^{n} d(s_i) - d(s_{i-1}) = \sum_{i=1}^{n} d(s_i) - \sum_{j=0}^{n-1} d(s_j)$

$= d(s_n) - d(s_0) = 0$, il y a un $i \in \{1, \dots, n\}$

avec $d(s_i) - d(s_{i-1}) > p(a_i)$

c.-à-d. $d(s_i) > d(s_{i-1}) + p(s_{i-1}, s_i)$

donc il faudrait encore appliquer la relaxation, les $d(s_i)$ continueront de diminuer.

Algorithme de Bellman – Ford

Initialisation(r)
Pour i allant de 1 à $|S|-1$ faire
Pour j allant de 1 à $|A|$ faire
relaxation (a_j). (on relaxe toutes les arêtes)
s'il existe une ~~arête~~ un arc (u,v) avec
$d(v) > d(u) + p(u,v)$
Alors FAUX → il y a un cycle de longueur <0, on n'a abouti pas
Sinon VRAI → pas de cycle de longueur <0, on a abouti.

Explication. S'il y a un cycle de longueur <0, on ne peut jamais aboutir, voir plus haut.

Supposons qu'il n'y en a pas.

Soit $r = s_0, \dots, s_m = s$ le chemin le plus court de r à s. Alors pour $1 \le i \le m$, $r = s_0, \dots, s_i$ est le chemin le plus court de r à s_i.

A l'étape $i=1$, la relaxation sur (s_0, s_1) donne la bonne valeur de $d(s_1)$. A l'étape $i=2$, la relaxation sur (s_1, s_2) donne la bonne valeur de $d(s_2)$. Etc. A l'étape m la relaxation sur (s_{m-1}, s) donne la bonne valeur de $d(s)$. On a $m \le |S|-1$ car $s_0, \dots, s_m$ tous distincts.

Cas des graphes dirigés sans circuit

Appliquer un tri topologique.

Donc on ordonne les sommets, $S = \{s_1, \dots, s_n\}$, où tout arc est de la forme (s_i, s_j) pour $j > i$.

Ensuite :

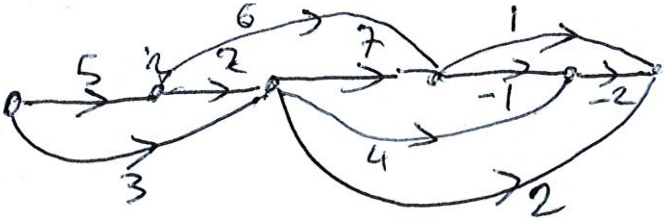
Initialisation(r)

Pour i allant de 1 à $n-1$ faire

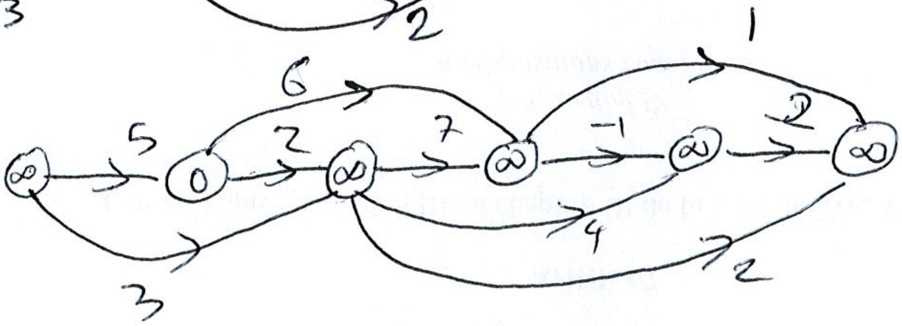
Pour tout $j > i$ avec $(s_i, s_j) \in A$ faire

relaxation(s_i, s_j).

En fait, on peut limiter la boucle sur i : ~~pour~~ si $r = s_t$, on fait : pour i allant de t à n.



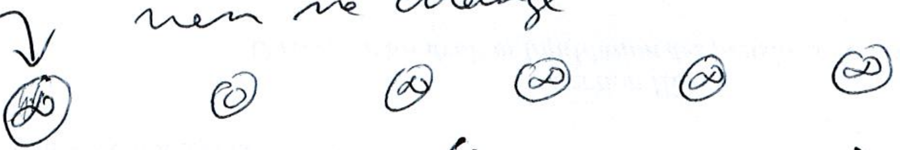
Init



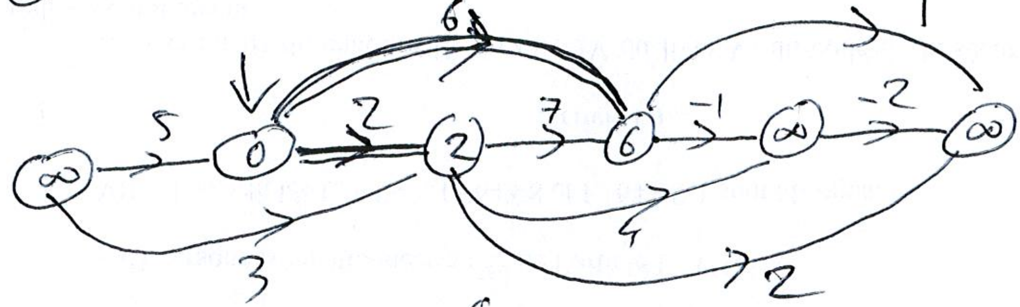
then no change

~~Relax~~

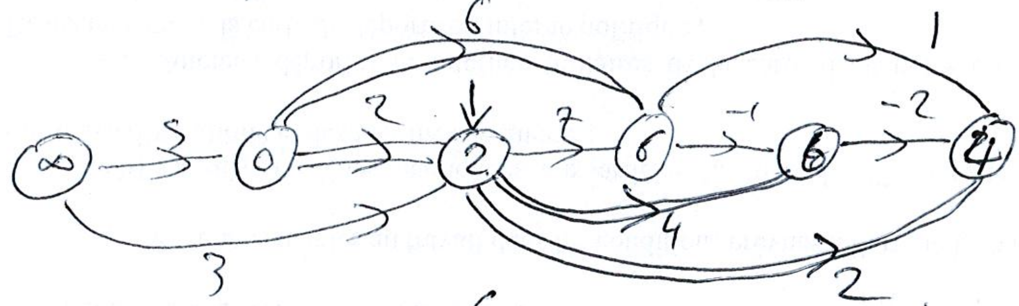
i = 1



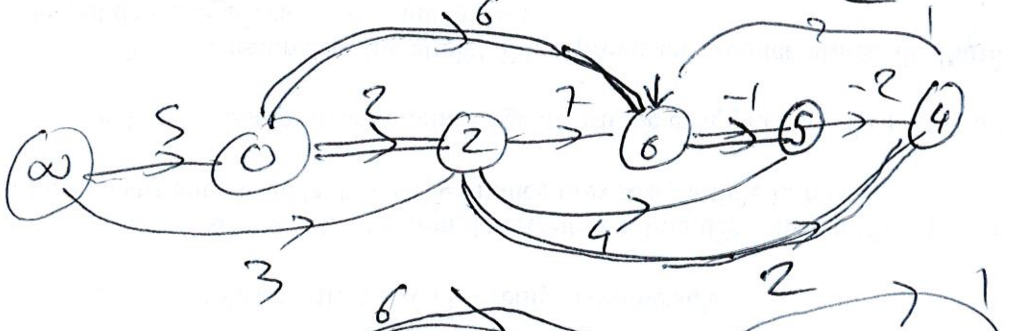
i = 2



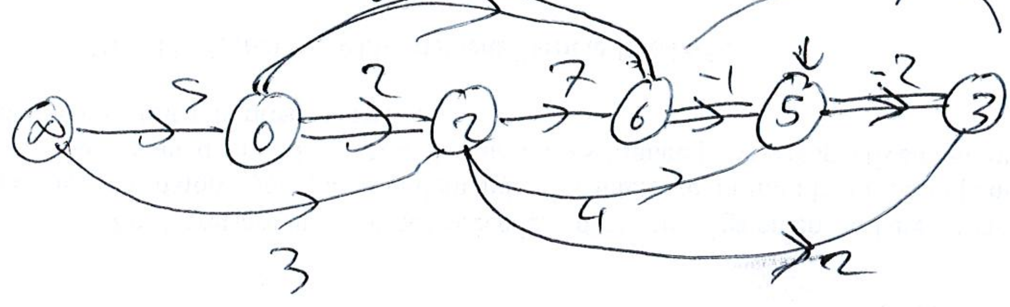
i = 3



i = 4



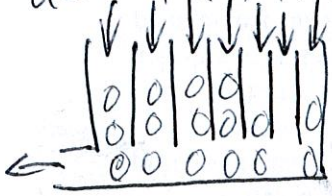
i = 5



Algorithme de Dijkstra

On suppose tous les poids d'arête ≥ 0. L'algorithme utilise une file à priorité indexée sur la valeur de d : le sommet de la file ayant la plus petite valeur de d sort le premier

$d = 0\ 1\ 2\ 3\ 4\ 5\ 6$



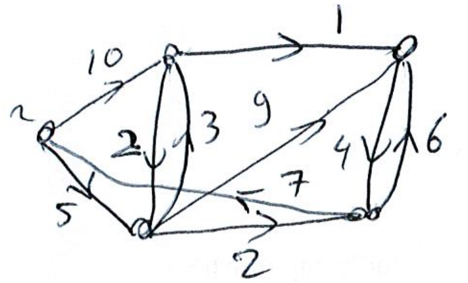
Initialisation(r)

Mettre tous les sommets dans la file

Tant que la file est non-vide, faire

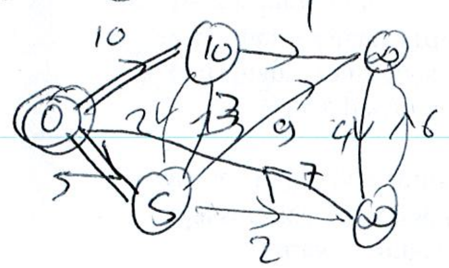
- extraire le sommet u de la file dont la valeur $d(u)$ est la plus petite possible
- pour chaque $v \in \Gamma^+(u)$, faire relaxation(u, v)

NB en faisant relaxation(u, v), la valeur de $d(v)$ peut diminuer, dans ce cas la priorité de v dans la file augmente.

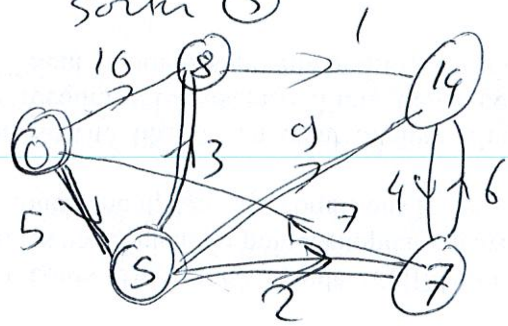


init

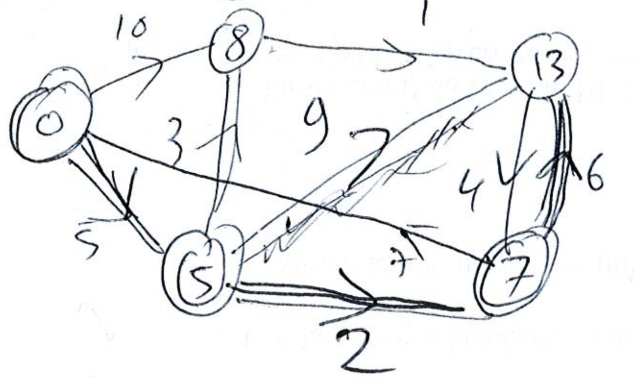
sortir 0



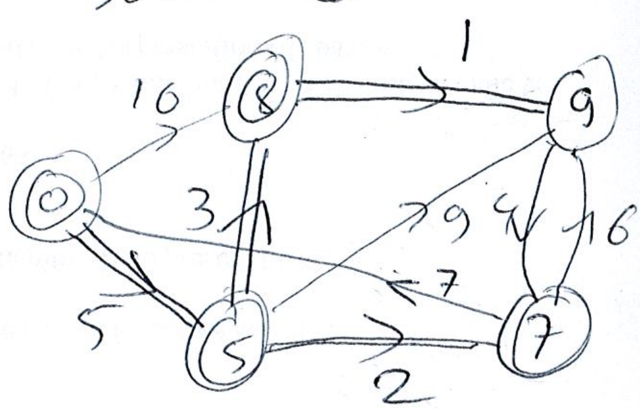
sortir 5



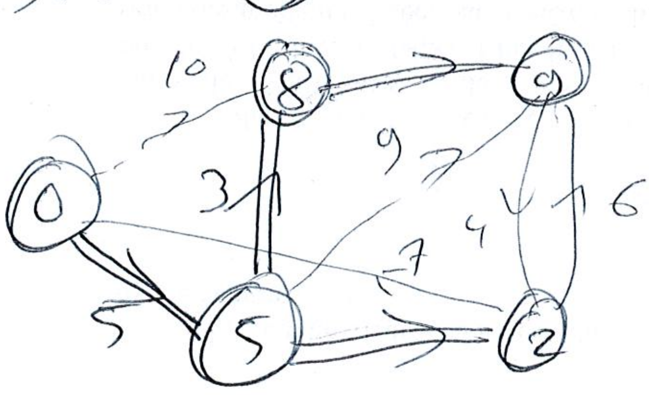
sortir 7



sortir 8



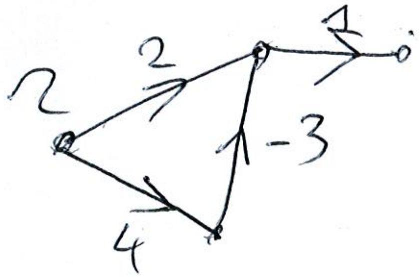
sortir 9



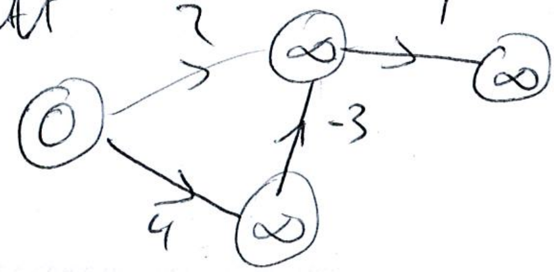
plus rien ne change

L'algorithme ne fonctionne pas correctement si on a des poids < 0.

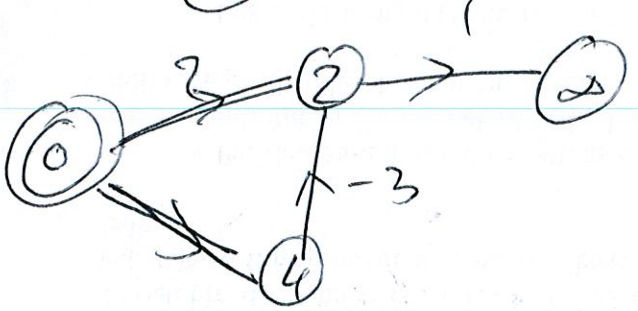
~~21/10~~



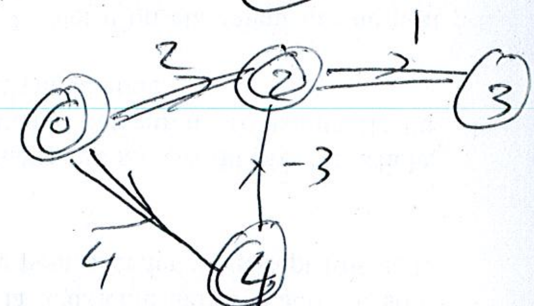
Init



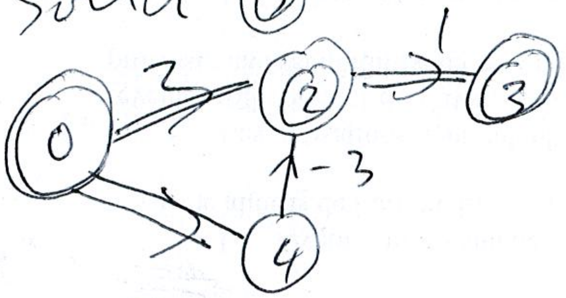
sorti ⓪



sorti ②

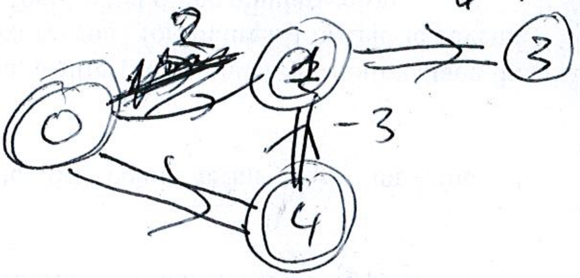


sorti ③



sorti ④

ne change plus



pour que ça marche, il faudrait remettre le sommet ② → ① dans la file !

Explication du fonctionnement pour des poids ≥ 0.

Quand on sort de la file les sommets u avec $d(u) = t$, leurs voisins sortants voient leur valeur de d diminuer, passant de

$d(v) > d(u) + p(u,v)$ à $d(v) = d(u) + p(u,v)$ $\geq d(u)$, car $p(u,v) \geq 0$. Donc les valeurs de d dans la file restent toutes $\geq t$. Par conséquent, les sommets sortent de la file pour des valeurs croissantes de d.

Soit $r = s_0, \dots, s_m = s$ le chemin le plus court de r à s. A tout moment, la valeur de $d(s_i)$ $(i = 0, \dots, m)$ est $\geq d(r, s_i)$.
A la sortie de ~~A ce temps~~ 0, $d(r, s_0) = d(s_0) = 0$.
Supposons qu'on sort de la file le niveau $d(s_i)$ et que $d(s_i) = d(r, s_i)$. Alors s_{i+1} sera mis à jour $d(s_{i+1}) = d(s_i) + p(s_i, s_{i+1}) = d(r, s_i) + p(s_i, s_{i+1}) = d(r, s_{i+1})$. Donc quand on sortira le niveau $d(s_{i+1})$ de la file, s_{i+1} sortira avec $d(s_{i+1}) = d(r, s_{i+1})$.
Par induction, vrai pour n, $s_n = s$ sort au niveau $d(s_n)$ avec $d(s_n) = d(r, s_n)$.
~~= ...~~