18.10.21 — AL5 TD5

TDs | Rappels : Nom | Applicat° — BFS : distances — DFS : accessibilité ; graphes orientés : composantes fortement connexes (CFC)

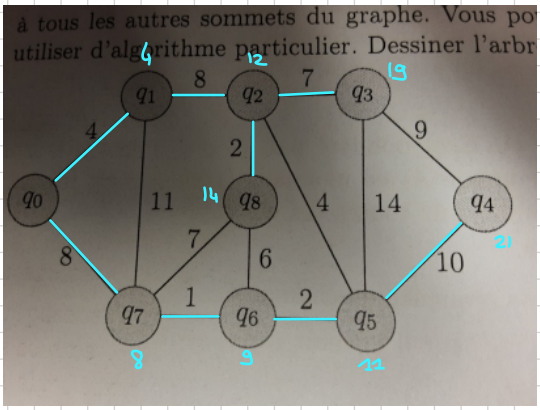
Dijkstra (G, s) : (s = source)

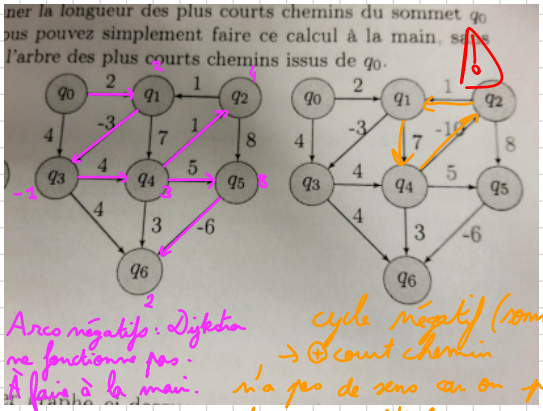
Init : file F = vide ; ∀x $d(x) = \infty$; $prev(x) = \emptyset$; (distance à la source) (la pile = les arêtes bleues du ex 1)

$d(s) = 0$; F.ajouter(s) ;

Tant que F non vide :
- extraire x, le sommet avec la plus petite distance
- ∀ (x, y) ∈ E
  si $d(y) > d(x) + l(xy)$ (le poids de l'arête (x,y))
  alors $d(y) = d(x) + l(xy)$ et $prev(y) = x$ (et MAJ de F pour que trouver le min reste simple)
  F.ajouter y

Exercice 1





Arcs négatifs : Dijkstra ne fonctionne pas. À faire à la main.

cycle négatif (somme des poids du circuit < 0) → ∃ court chemin n'a pas de sens car on peut tourner à l'infini.

Exercice 2

q_2 a des arcs négatifs donc Dijkstra ne fonctionne pas (la preuve de correct° de Dijkstra s'appuie sur le fait que les poids ≥ 0) (cf. ampli 5) (bug à q_5 et q_6)

q_1

q	0	1	2	3	4	5	6	7	8	
d	0	∞	∞	∞	∞	∞	∞	∞	∞	F = 0
↑	×									

q	0	1	2	3	4	5	6	7	8	
d	0	4	∞	∞	∞	∞	∞	8	∞	F = 1, 7
↑	×	0						0		s = 0

q	0	1	2	3	4	5	6	7	8	
d	0	4	12	∞	∞	∞	∞	8	∞	F = 7, 2
↑	×	0	1					0		s = 1

q	0	1	2	3	4	5	6	7	8	
d	0	4	12	∞	∞	∞	9	8	15	F = 6 2 8
↑	×	0	1				7	0	7	s = 7

q	0	1	2	3	4	5	6	7	8	
d	0	4	12	∞	∞	11	9	8	15	F = 5 2 8
↑	×	0	1			6	7	0	7	s = 6

q	0	1	2	3	4	5	6	7	8	
d	0	4	12	25	21	11	9	8	15	F = 3 2 8 4
↑	×	0	1	5	5	6	7	0	7	s = 5

$d(3) > d(2) + val(2,3)$; $d(8) > d(2) + val(2,8)$

q	0	1	2	3	4	5	6	7	8	
d	0	4	12	19	21	11	9	8	14	F = 3 8 4 (3 et 8 sont déjà ds la file)
↑	×	0	1	2	5	6	7	0	2	s = 2

q	0	1	2	3	4	5	6	7	8	
d	0	4	12	19	21	11	9	8	15	F = 3 4
↑	×	0	1	2	5	6	7	0	7	s = 8 (pas de chgt ds le tableau)

q	0	1	2	3	4	5	6	7	8	
d	0	4	12	19	21	11	9	8	15	F = 4
↑	×	0	1	2	5	6	7	0	7	s = 3 (pas de chgt ds le tableau)

q	0	1	2	3	4	5	6	7	8	
d	0	4	12	19	21	11	9	8	15	F = ∅ FIN
↑	×	0	1	2	5	6	7	0	7	s = 4 (pas de chgt ds le tableau)

✓

25.11.21

Rappels :

« Dijkstra » = ⊕ courts chemins 1 to all (depuis une source)

→ à chaque étape
- ens de sommets avec dist
- ens de sommets avec appro de dist

→ prend le ⊕ proche des appr

→ met à jour les approximatθ des autres

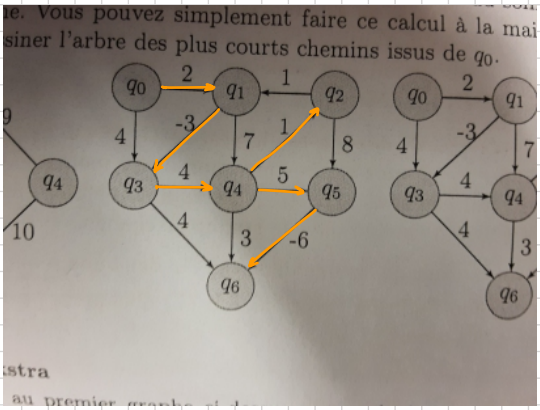
⚠ PAS de poids NÉGATIFS

Exercice 3

BF ⊖ efficace que D mais peut avoir des poids négatifs
↳ O(n m)

```
init (dist à ∞ pour all)
pour i de 1 à n-1 :
    ∀ arrête (u,v) : MAJ
```

↑ si d(v) < d(u) + poids(u,v) alors on MAJ dist(v)



sommet	init	i=1 dist	i=1 père	i=2 dist	i=2 père
0	0	0	X	0	X
1	∞	2	0	2	0
2	∞	4	4	4	4
3	∞	~~4~~ -1	~~4~~ 1	-1	1
4	∞	~~7~~ 3	~~1~~ 4	3	4
5	∞	8	4	8	4
6	∞	~~8~~ 2	~~3~~ 5	2	5

(i jusqu'à 6)

(normalement il faut encore finir la boucle, mais on voit bien qu'il ne va pas y avoir d'autres chgts).

On prend les arêtes de l'ordre alphabétique

(0,1) (0,3) (1,3) (1,4) (2,1) (2,5) (3,4)

(3,6) (4,2) (4,5) (4,6) (5,6)

(couleurs = pour se repérer de l'exécutθ)

Si on fait BF sur le 1er graphe, on trouvera les mêmes distances (mais peut-être en ⊕ d'opérations)

Exercice 5

1) Par exemple sur l'exo 3)

Un ⊕ simple si on regarde les arêtes de gauche à droite.

2) Le même graphe, mais en partant de la droite : il faut n-1 itérations.

Formellement : un chemin dirigé de taille n. On regarde les arêtes de l'ordre inverse du chemin. À l'étape i, seuls les i premiers sommets ont une valeur. Il faut n-1 itérat° pour que chaque sommet ait sa distance.

3) Mettre un flag : au moment de la MAJ, flag à vrai s'il y a un chgt

À la fin de l'itération, si le flag est faux, ça veut dire qu'on n'a rien changé ⇒ on peut s'arrêter.

Exercice 6

1) Le taux de change est $S \times c(A_1, A_2) \times c(A_2, A_3) \times \dots \times c(A_{k-1}, A_k)$

2) Si on a un cycle de change, et toutes les $c(A_i, A_j) > 1$.

3) Si $c(S_2) + c(C,B) > c(S_1)$
Alors on garde $S_2 \to (C,B)$

Exercice 4

$d(s) = \text{poids}(s)$ dans l'initialisation.
Tant que F non vide :
- extraire x, le sommet avec la plus petite distance
- $\forall (x,y) \in E$ si $d(y) > d(x) + \text{poids}(y)$
  alors $d(y) = d(x) + \text{poids}(y)$ et $\text{prev}(y) = x$
  F. ajouter y

C'est Dijkstra mais au lieu d'ajouter le poids des arêtes, on ajoute le poids du sommet concerné.

Sinon on peut se ramener à un graphe normal :

$$p(u,v) = w(u) + w(v)$$

On obtient des chemins qui minimisent $p(st) = 2 \times dist(st) - \underbrace{w(s) - w(t)}_{\text{la source et l'arrivée ne sont comptées qu'une seule fois.}}$

Pour retrouver la distance qu'on veut calculer : $dist(st) = \dfrac{p(st) + w(s) + w(t)}{2}$

OK car $w(s)$ et $w(t)$ sont des constantes.

Pour un graphe orienté :

$p(u \to v) = w(v)$

- pour chaque arrête, on met le poids du sommet au lequel l'arrête arrive
- pour avoir la vraie distance, ajouter $p(s)$.