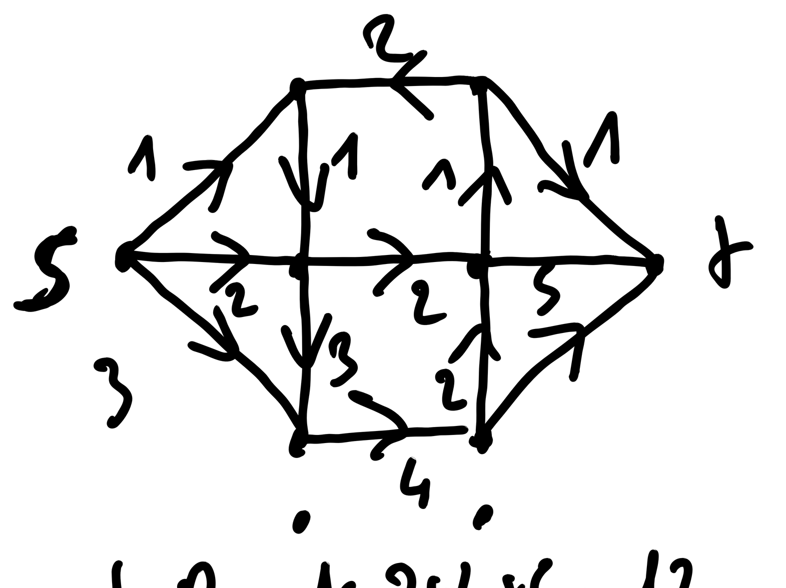
Plus court chemin

Déf: Étant donné un graphe orienté $G=(V,E)$, avec pondération $\ell \in \mathbb{R}^{|E|}$ (à chaque arc e on associe un nombre réel ℓ_e), alors la longueur d'un chemin P est définie comme $\sum_{e \in E(P)} \ell_e$.

Exemple



La longueur de P est $3+4+5=12$

la longueur de $Q = 2+2+5 = 9$

" " $R = 1+1+2+1+1 = 6$

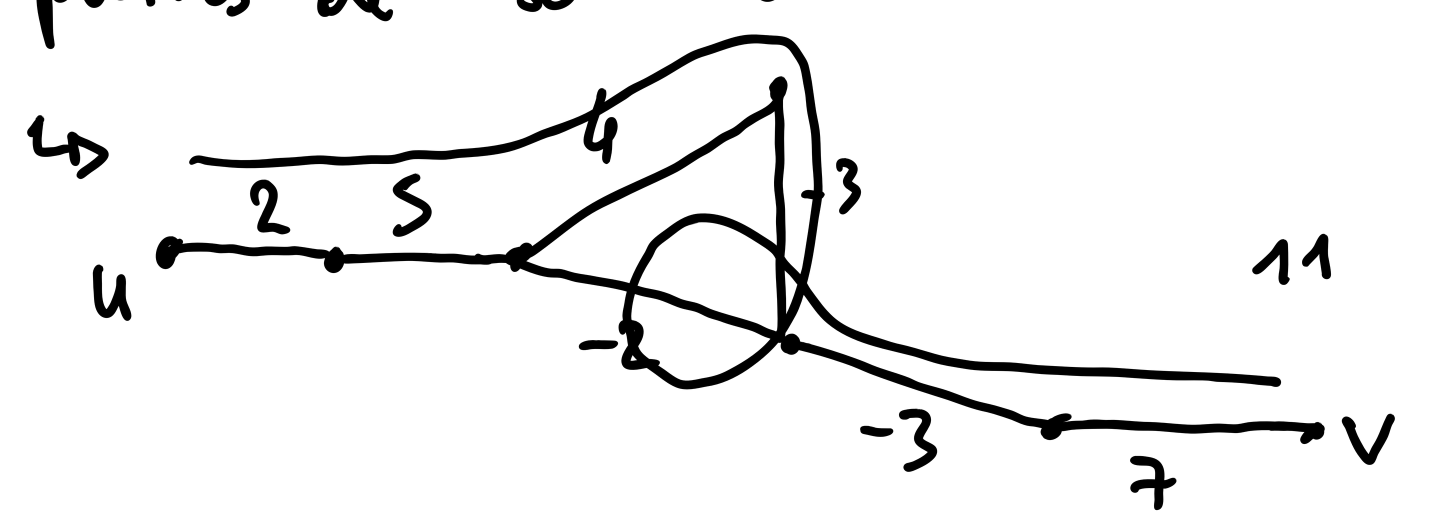
$dist(s,t) \overset{?}{=} 6$

Déf : La distance de u à v (où $u,v \in V$) est la longueur d'un plus court chemin de u à v.

Càd, $dist(u,v) = \min \{ \sum_{e \in E(P)} \ell_e : \text{chemin de } u \text{ à } v \}$

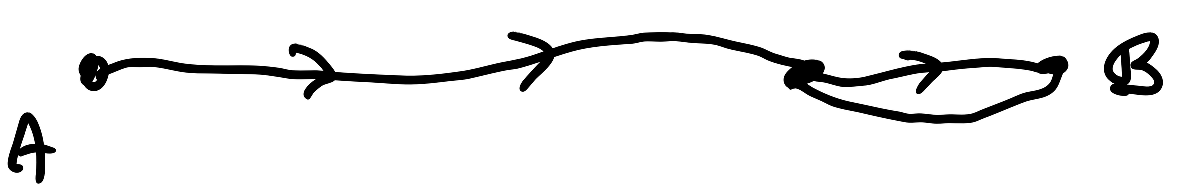
Remarque : Il y a des cas où $dist(u,v)$ n'est pas défini.

1. Soit il n'y a aucun chemin de u à v. Dans ce cas on écrit $dist(u,v) = +\infty$
2. Si le graphe contient un cycle négatif, alors la distance pour certaines paires de sommets est $-\infty$



$2+5+4-3-3+7=12$

Principe de sous-optimalité



Si P est un plus court chemin de u à v, et $w \in V(P)$, alors le sous chemin de w à v est un plus court chemin de w à v.

Preuve : Supposons par l'absurde qu'il existe un chemin P' de w à v de longueur inférieure à celle du sous chemin P_{wv}. Alors le chemin $P_{uw} + P'$ est un chemin de u à v de longueur inférieure à celle de P, contradiction !

Algorithme de Dijkstra

Entrées : Un graphe orienté $G(V,E)$ avec pondération $l \in \mathbb{R}^{|E|}$ tq $l_e \geq 0$ $\forall e$ un sommet $s \in V$ (la source)

Sortie : Distance de s à tous les sommets

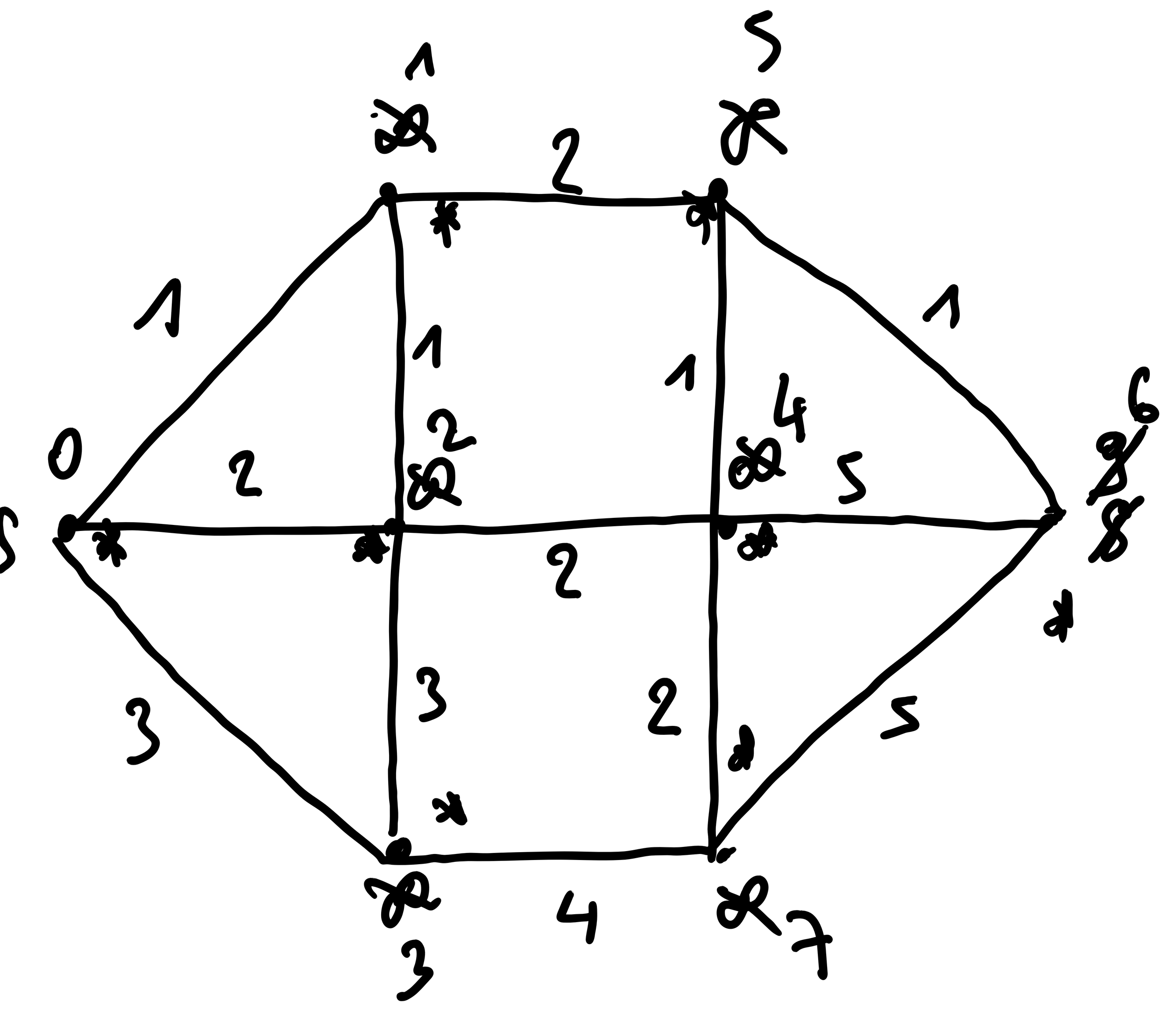
$S \leftarrow \emptyset$

$D[s] \leftarrow 0$

pour tous les $u \in V \setminus \{s\}$:
    $D[u] \leftarrow \infty$

tant que $S \neq V$:
    Trouver $u \in V \setminus S$ t.q. $D[u]$ est minimum
    $S \leftarrow S \cup \{u\}$
    pour tous les $v \in V \setminus S$ tq $(u,v) \in E$:
        $D[v] \leftarrow \min\{D[v], D[u] + l(u,v)\}$

retourner D

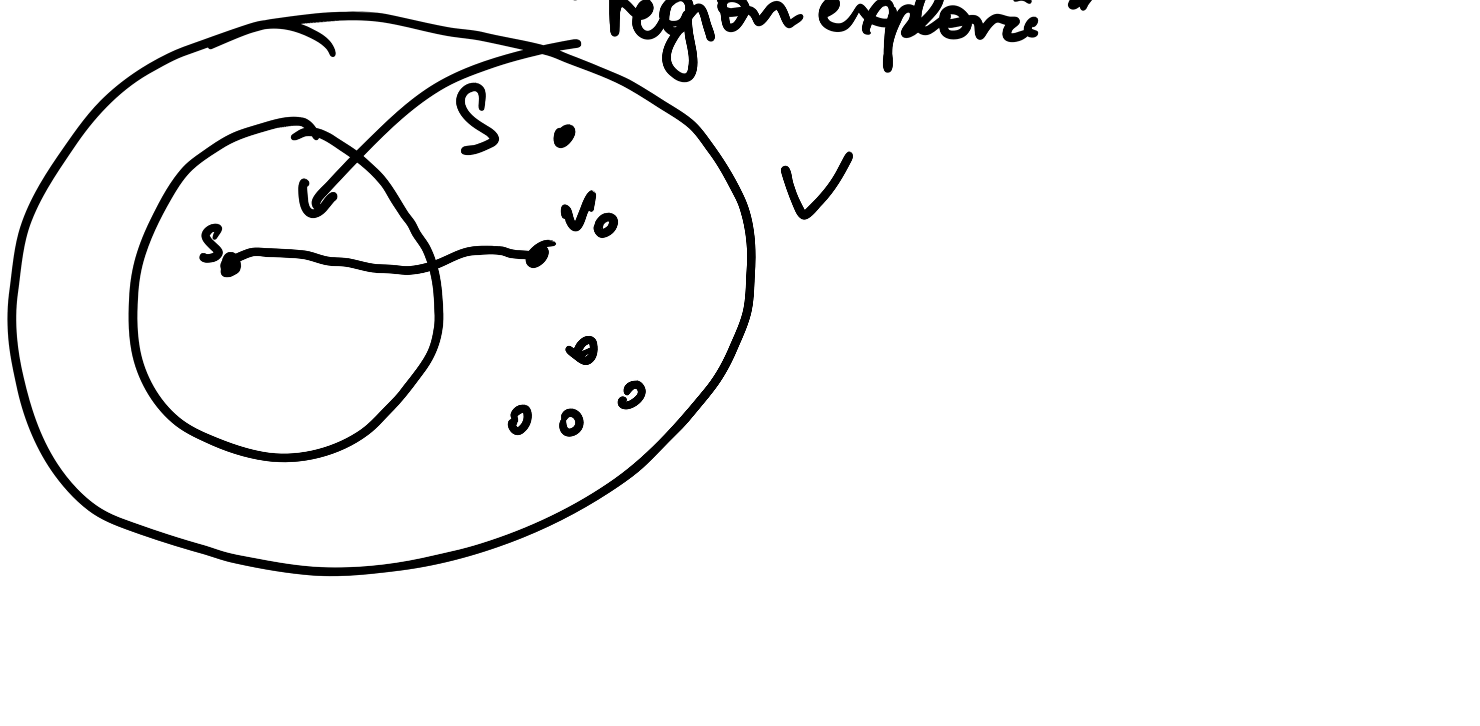


// Mettre ∞ symbole sur les sommets non découverts

* sommet visité

Déf : Soit $G=(V,E)$ un graphe orienté avec pondération $l \in \mathbb{R}^{|E|}$ tq $l_e \geq 0$ $\forall e$, et soit $s \in S \subseteq V$.

Pour tout $u \in V$, on note $d(u) = dist(s,u)$. Pour tout sommet $v \in V \setminus S$, on définit $D(v) = \min\{d(u) + l(u,v) : u \in S \text{ et } (u,v) \in E\}$



Lemme :

Soit $v_0 \in V \setminus S$ tq $D(v_0)$ est minimum, parmi tous les sommets dans $V \setminus S$.
Alors, $D(v_0) = d(v_0) = dist(s, v_0)$

Preuve : Soit P un plus court chemin de s à v_0. La longueur de P est $d(v_0)$.
Comme $D(v_0)$ est la longueur d'un chemin de s à v_0, on a forcément $D(v_0) \geq d(v_0)$
On veut prouver que $D(v_0) = d(v_0)$.
Supposons par l'absurde que $D(v_0) > d(v_0)$

Soit x le premier sommet de P qui $\notin$ à S.
Soit y le prédécesseur de x dans P.
Par le choix de x, on a $y \in S$.
Soit P_y le sous chemin de P de s à y. Par le principe de sous-optimalité, P_y est un plus court chemin de s à y.
Soit P_x le sous chemin de P de s à x.
La longueur de P_x est égale à la longueur de P_y plus $\ell(y,x)$.
C-à-d, $\ell(P_x) = d(y) + \ell(y,x)$.
Ceci entraine :

$$\underset{\substack{\uparrow \\ \text{définition de } D}}{D(x) \leq} d(y) + \ell(y,x) \underset{\substack{\uparrow \\ \text{pas d'arcs négatifs}}}{\leq} d(v_0) \underset{\substack{\uparrow \\ \text{hypothèse}}}{<} D(v_0)$$

Le fait que $D(x) < D(v_0)$ contredit le choix initiale de v_0.
Donc, on a bien $d(v_0) = D(v_0)$.

Les files de priorité

Une file de priorité est un type sur laquelle on peut effectuer les opérations suivantes
Insert : insérer un élément
Decrease-key : diminuer la valeur de la clé d'un élément particulier
Delete-min : retourne l'élément ayant la plus petite clé et supprimer-le de la file
Make-queue : créer une file de priorité à partir des éléments donnés.

Complexité des opérations dans les files de priorité :

implémentation	delete-min	insert	decrease-key
liste	n	1	1
tas binaire	$\log n$	$\log n$	$\log n$
tas fibonacci	$\log n$	1	1

Dijkstra (avec files de priorité)

```
pour tous les u ∈ V :
    D[u] ← +∞
    prev[u] ← ∅
D[s] = 0
H ← make-queue
tant que H ≠ ∅
    u ← delete-min(H)
    pour tous les (u,v) ∈ E :
        si D[v] > D[u] + l(u,v)
            D[v] = D[u] + l(u,v)
            prev[v] ← u
            decrease-key(H,v)
```

Complexité de l'algo de Dijkstra

Bien que Dijkstra est très proche au BFS les files de priorité sont plus exigeantes que les files de BFS.

- make-queue prend au plus n opérations d'insertion
- il y a n delete-min
- il a $n+m$ decrease-key.

Implémentation de file de priorité	complexité de Dijkstra
liste	n^2
tas binaire	$(n+m)\log n$
tas de Fibonacci	$m + n\log n$

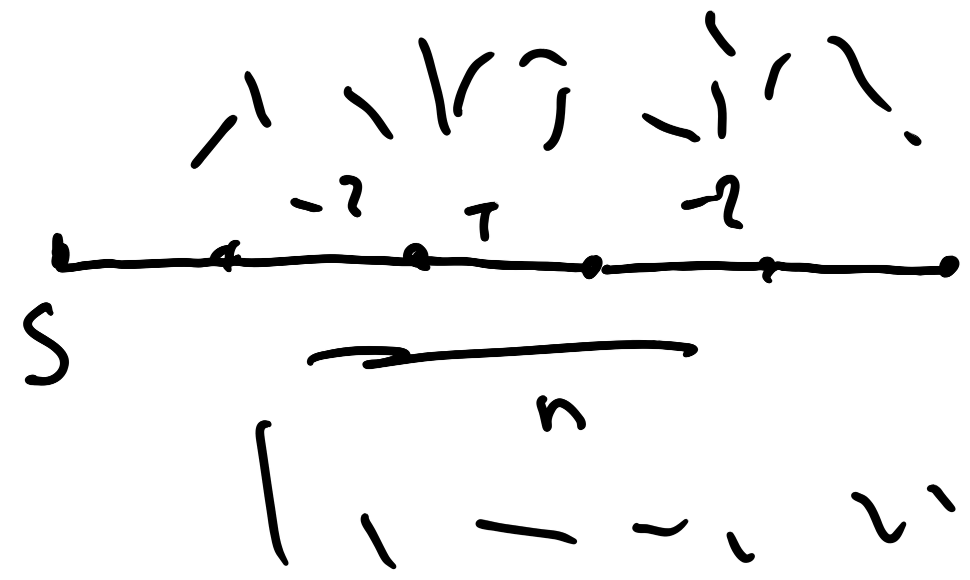
On peut considérer Dijkstra comme une séquence de mises à jour :

procédure maj(u,v).

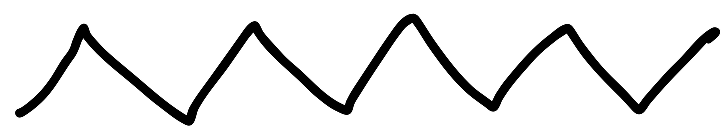
$$D[v] \leftarrow \min\{D[v], D[u] + l(u,v)\}$$

Cette procédure a les propriétés suivantes :

- elle donne la vraie distance de s à v lorsque u est l'avant-dernier sommet d'un plus court chemin de s à v
- elle ne rendra jamais $D[v]$ trop petit



n sommets dans G car pas de cycle négatif



il y a au plus $n-1$ arrêtes dans un chemin de s à t.