

## PLUS COURT CHEMIN

Def. Soit  $G(V, E)$  un graphe orienté.  
avec pondération  $l \in \mathbb{R}^{|E|}$ , alors  
la longueur d'un chemin  $P$  est

$$\sum_{e \in E(P)} l_e$$

Def: La distance de  $u$  à  $v$   
est la longueur du plus court chemin  
de  $u$  jusqu'à  $v$ .

$$\text{dist}(u, v) = \min \left\{ \sum_{e \in E(P)} l_e : P \text{ le chemin de } u \text{ à } v \right\}$$

Remarque: Si il n'existe pas de chemin de  $u$  à  $v$  alors on note  $\text{dist}(u, v) = +\infty$

## Principe de sous-optimalité:

" Si de Paris à Marseille on passe  
par Lyon  
alors le plus court chemin de Paris  
à Marseille est un plus court chemin  
de Paris à Lyon, puis Lyon à Marseille. "

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  $v$ .  
 (il peut y en avoir plusieurs)

## Algorithm de Dijkstra:

- Entrée:
- $G$  un graphe orienté avec une pondération  $\ell \in \mathbb{R}^{|E|}$  tq  $\forall e \in E, \ell_e \geq 0$
  - Un sommet  $s$  (la source)

Sortie: Distance de  $s$  à tous les autres sommets.

= Initialization  
 $S \leftarrow \emptyset$   
 $D[s] \leftarrow 0$   
 Pour  $v \in V(G) \setminus \{s\}$   
 |  $D[v] \leftarrow \infty$

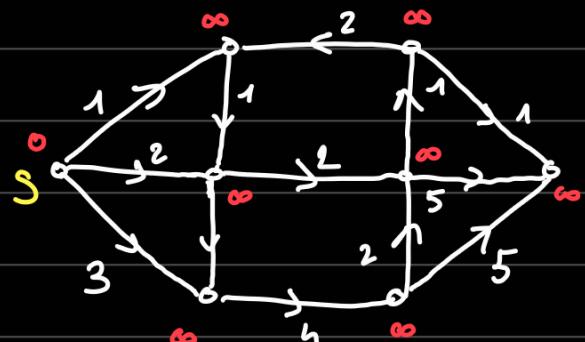
Tant que  $S \neq V$

$ $ Trouver $v \in V \setminus S$ tq $D[v]$ est minimum $S \leftarrow S \cup \{v\}$ (union) pour tous les $u \in V \setminus S$ tq $(u, v) \in E$ $D[v] \leftarrow \min \{D[v], D[u] + \ell(u, v)\}$
--

retourner D

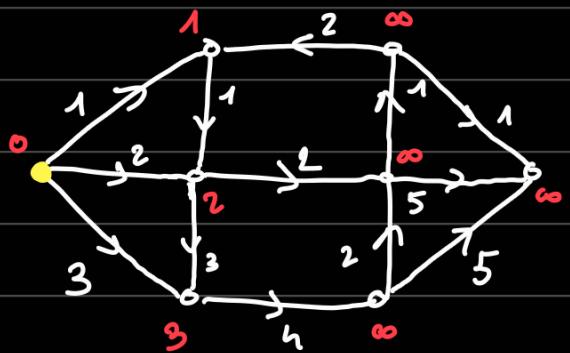
permet que si  
sommets de  $\tilde{v}$  à  $\tilde{w}$   
ne fait rien

Exemple :

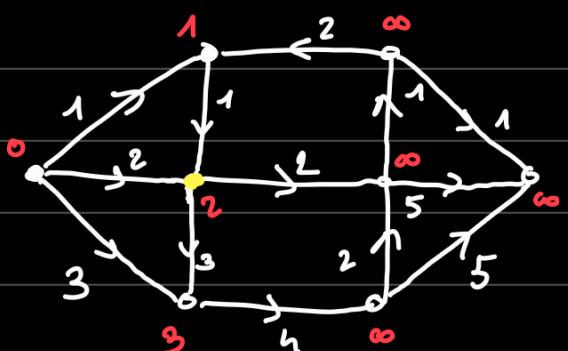
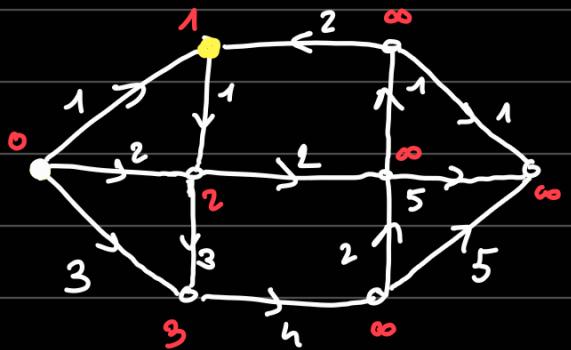


$D : m$

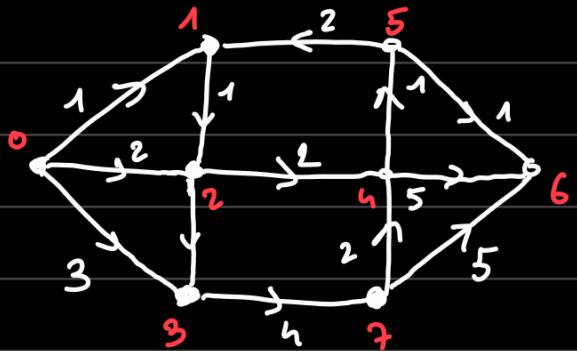
$S : m$



$m = \text{sommel } v. (\text{sommel courant})$



⋮



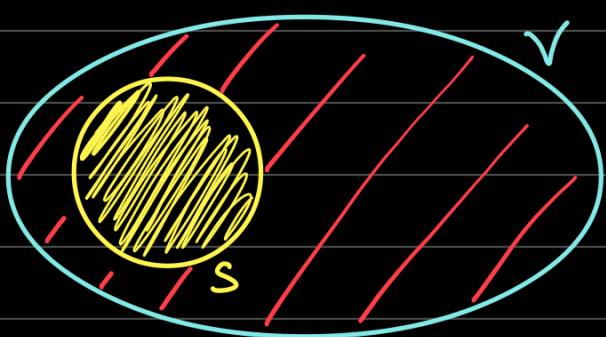
Le poids ne peut pas être négatif  
sinon on pourra reassigner des sommets  
déjà marqués.

Def: Soit  $G(V, E)$  un graphe orienté  
avec pondération  $\ell \in \mathbb{R}^{|E|}$  tq  $\forall e \in E \quad \ell_e \geq 0$  ( $\Rightarrow \ell_e \geq 0$ )  
et soit  $s \in S \subseteq V$

Pour tout  $v \in V$ , on note  $d(v) = \text{dist}(s, v)$

Pour tout  $v \in V \setminus S$  on définit

$$D(v) = \min \left\{ d(v) + \ell(u, v) : u \in S \text{ et } uv \in E \right\}$$



$S$  = région explorée

$V \setminus S$

Lemme: Soit  $v_0 \in V \setminus S$  tq  $D(v_0)$  est minimum,  
parmi tous les sommets dans  $V \setminus S$   
alors  $D(v_0) = d(v_0) = \text{dist}(s, v_0)$

Une file de priorité est un type sur lequel on peut utiliser les opérations:

- Insert (insérer un élément)
- Decrease-key (diminuer la valeur de la clé d'un élément particulier)
- Delete-min (retourne et supprime le plus petit élément)
- Make-queue (créer une file de priorité)

Complexité des opérations dans les files de priorité

Implementation	delete-min	insert	decrease-key
list	$O(n)$	$O(1)$	$O(1)$
binary-heap	$O(\log n)$	$O(\log n)$	$O(\log n)$
Tas de Fibon	$O(\log n)$	$O(1)$	$O(1)$

↳ Dijkstra (avec file de priorité)

nombre d'appel

$n \rightarrow H \leftarrow \text{make-queue}()$       Initialisation de la file de priorité

$n \rightarrow$       Tant que  $H \neq \emptyset$ :

$\left| \begin{array}{l} \text{pour tous les } v \in V \\ | \quad D[v] \leftarrow +\infty \\ | \quad \text{prev}[v] \leftarrow \emptyset \\ | \quad D[s] \leftarrow 0 \end{array} \right.$

$\left| \begin{array}{l} \text{while } \exists v \in H \text{ tel que } D[v] < \infty \\ | \quad u \leftarrow \text{delete-min}(H) \\ | \quad \text{pour tous les } (u,v) \in E : \\ | \quad | \quad \text{si } D[v] > D[u] + l(u,v) : \\ | \quad | \quad \text{alors } D[v] \leftarrow D[u] + l(u,v) \\ | \quad | \quad \text{et } \text{prev}[v] \leftarrow u \end{array} \right. \end{array} \right. \end{array}$

$$n+m \rightarrow \left| \begin{array}{c} D[v] = D[u] + l(u,v) \\ \text{prev}[v] \leftarrow u \\ \text{decrease-key}(H, v) \end{array} \right.$$

return ( $D$ ,  $\text{prev}$ )

### Complexité de l'algorithme de Dijkstra:

Bien que l'algo est très proche du parcours en largeur, les règles de priorité sont plus exigeantes que les FIFO basiques de BFS.

Implementation des FOP	Complexité de Dijkstra
Liste	$O(n^2)$
Tas binnaire	$O(\log(n)(n+m))$
Tas de Fibonacci	$O(n\log(n) + m)$

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

procédure  $\text{maj}(u, v)$ :

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

La procédure a les propriétés suivantes :

- Elle donne la vraie distance de  $s$  à  $v$  <sup>chemin de</sup> ~~six~~
- lorsque  $v$  est l'avant-dernier sommet d'un plus court.
- Elle ne rendra jamais  $\delta[v]$  trop petit.