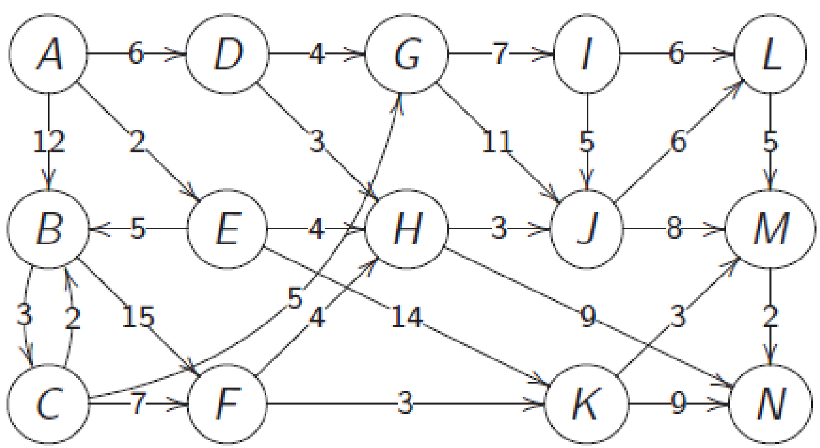
1 Les algorithmes des plus courts chemins

Service de livraison **Service de livraison des produits**

- Point de départ : usine de fabriquation
- **Objectif: optimiser le temps et/ou le coût de transport.**
- Réseau de transport avec les distances entre les sommets/points de livraison.
- Il faut un algorithme pour calculer le PCC en fonction d'une destination.

Modélisation du réseau de transport :
-des points de livraison: un ensemble X de sommets
-des liaisons : un ensemble d'arcs U
-des longueurs : d(u) pour tout $u \in U$
Contextes d'utilisation:
-réseaux géographiques

-états d'un système Exemple



Chemin Un chemin dans un graphe G = (X, U) orienté reliant deux sommets x et y de longueur q est:

- une suite d'arcs consécutifs $(u_1, u_2, ..., u_q)$
- u_1 est incident à x par son extrémité initiale $I(u_1) = x$.

- u_q est incident à y par son extrémité terminale $T(u_q) = y$.
- les arcs intérmidaires u_i pour $i = 2, ..., q$ sont incidents à u_{i-1} $(T(u_{i-1}) = I(u_i))$.

La longueur d'un chemin c dans un graphe G = (X, U) est une application $d : U - - - \ R$:

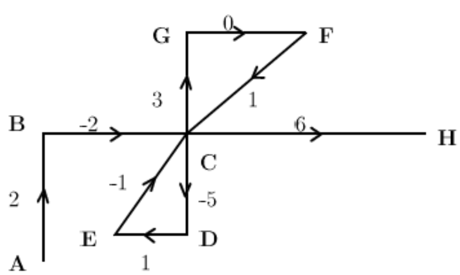
$$d(c) = \sum_{u \in C} d(u)$$

Chaque arc u a une longueur $d(u)$ (distance/poids).
Problème: Trouver un chemin le plus court entre deux sommets sur un réseau $R = (X, U, d)$. Définition du plus court chemin (PCC)

Plus court chemin entre deux sommets i et j est un chemin dont la longueur est le minimum:

$$\delta(i,j) = \begin{cases} min\{d(c)/c \ est \ un \ chemin \ de \ i \ à \ j\} & s'il \ existe \ un \ chemin \\ \infty, & sinon \end{cases}$$

Condition d'existence du PCC entre i et j dans un réseau est : descendant de i et les ascendant de j ont des sommets en commun.
Pas de circuit absorbant sur les points en commun.



PCC Circuit négatif Un circuit C tel que $\sum_{u \in C} d(u) < 0$ est dit circuit absorbant. Définition Etant donné un réseau $R = (X, U, d)$ sans circuits absorbants et admettant que le sommet s comme racine, à chaque sommet x on associe la longueur du PCC de s à x dans R tel que : $\delta(x) = d(s, x)$. Théoréme Etant donné un réseau $R = (X, U, d)$ où chaque arc $u = (i, j) \in U$ a une longueur d(u), tel que ,

$$\delta(T(u)) - \delta(I(u)) \leq d(u)$$

si et seulement si R n'a pas de circuit de longueur négatif.
Le sommet s a une distance $\delta(s) = 0$. Arborescence de plus courts chemins

Sur un graphe G=(X, U) orienté pondéré qui ne contient aucun ciruit absorbant/négatif depuis la racine s. Arborescence Une arborescence de plus cours chemins de racine s est un sous graphe orienté G'=(X', U'), tel que :

- X' est l'ensemble de sommets accessible à partir de s dans G.
- G' forme une arborescence de racine s.
- pour tout $x \in X'$, le chemin unique de s à x dans G' est un plus court chemin de s vers x dans G.

$\boxed{\text{Algorithmes de recherche du plus court chemin}}$ Deux algorithmes :

- Algorithme de **Dijkstra**.

  Si les longueurs d'arcs $d(u) \geq 0$ pour tout $u \in U$.
- **Objectif**:
  Recherche des plus courts chemins d'un sommet à tous ses descendants.
- **Résultat:** Arborescence unique des plus courts chemins de la racine aux autres sommets de R.

- Algorithme de **Bellman**
  -S'il n'y a pas de circuits dans R
  -Le seul sommet sans prédésseurs est la racine s.
- **Objectif**: De proche en proche, on calcul le plus court chemin de s à x si on calculé le plus court chemin à ses prédésseurs.
  Il calcul le plus court chemin de s à tous les sommets.
- **Résultat:** Plusieurs Arborescences des plus courts chemins de la racine aux autres sommets de R.

$\boxed{\text{Initialisation des algorithmes}}$

- Pour chaque sommet $x \in X - s$, $d(x) = \infty$
- l'ensemble de prédécesseurs de chaque sommet est vide.
- la racine d(s)=0.

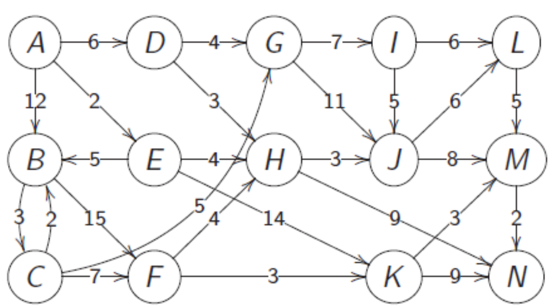
Application des algorithmes: Routage à vecteur de distance et à état de liens dans les réseaux informatiques.

Algorithme Dijkstra **Entrée** :
un graphe $G = (X, U)$ avec des longueurs d'arcs $d(i, j) \geq 0$; un sommet particulier s.
Résultat : longueur δ_i des plus courts chemins de s à i.
Début
$Z = \{s\}$, $\delta_s = 0$
pour tout $v \in X - \{s\}$ faire $\delta_v = d(s, v)$
s'il n'y a pas d'arc entre s et v on pose $d(s, v) = \infty$
Tant que $Z \neq X$ faire
début
calculer $\delta_x = min\{\delta_v | v \notin Z\}$

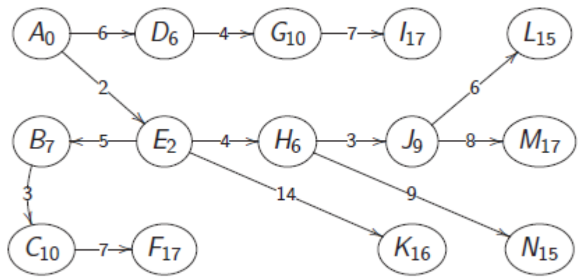
$$Z = Z \cup \{x\}$$

pour tout $v \in X - Z$ faire $\delta_v = min\{\delta_v, \delta_x + d(x, v)\}$
fin
Fin Tant que

Algorithme Dijkstra Algorithme gère un sous ensemble de sommets dont les longueurs finales de plus court chemin à partir de la racine (s=A dans l'exemple) ont déjà été calculées.



Exemple d'arborescence de racine A Les plus courts chemins sont marqués sur les sommets.



$\boxed{\text{Algorithme Bellman}}$ Sur un réseau $R = (X, U, d)$ sans circuits et un seul sommet s dont les prédésseurs sont vides.
on connaît seulement le point de départ s
on marque X si on découvre un chemin de s à X
la marque de X ne peut que décroître
si la marque est définitive, le chemin doit être minimal !

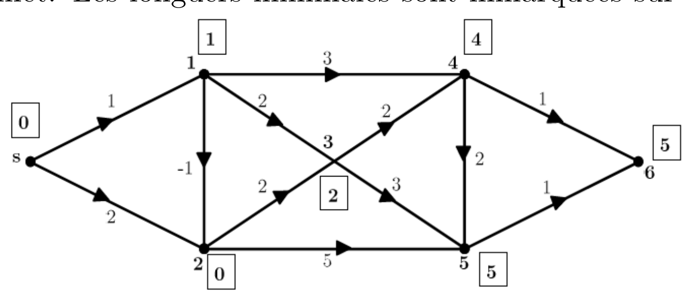
$\boxed{\text{Algorithme Bellman}}$ **Entrée** : un graphe $G = (X, U)$; un sommet particulier s.
Résultat : longueur δ des plus courts chemins, A.
Début
$Z = \{s\}$, $\delta_s = 0$, $A(s) = \emptyset$,
$\delta_i = \infty$ pour tout $i \neq s$
Tant que $j \notin Z$ et $P(j) \subset Z$ et $P(j) \neq \emptyset$ faire

calculer

$$\delta_j = min\{\delta(I(u)) + d(u)\}$$
$$(u \in U/T(u) = j)$$

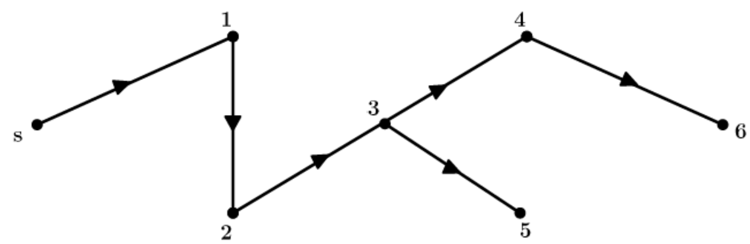
$Z = Z \cup \{j\}$ $A(j) = \{u\}$
Fin Tant que

$\boxed{\text{Exemple}}$ La figure suivante présente un graphe avec les distances minimales de s à chaque sommet. Les longuers minimales sont mmarquées sur chaque



sommet depuis s.

$\boxed{\text{Arborescence}}$ Itération 1: $\delta_1 = 1, A(1) = \{(s,1)\}, Z = \{s,1\}$



Iteration 6: $\delta_6 = 5, A(6) = \{(4,6)\}, Z = \{s,1,2,3,4,5,6\}$