Plus courts chemins : algorithme de Bellman-Ford

Aix-Marseille Université

2023-2024

Plus courts chemins dans un graphe

$G = (S, A, w)$ un graphe pondéré.

La longueur du chemin $(u_1, \ldots, u_k)$ est

$$\sum_{i=1}^{k-1} w(u_i, u_{i+1})$$

Notation :

$$\delta(u, v)$$

la longueur d'un plus court chemin de u à v.

Plus courts chemins dans un graphe

Arcs de poids positifs :

- algorithme de Dijkstra (source unique)

Circuits de longueur négative :

- pas de plus courts chemins,

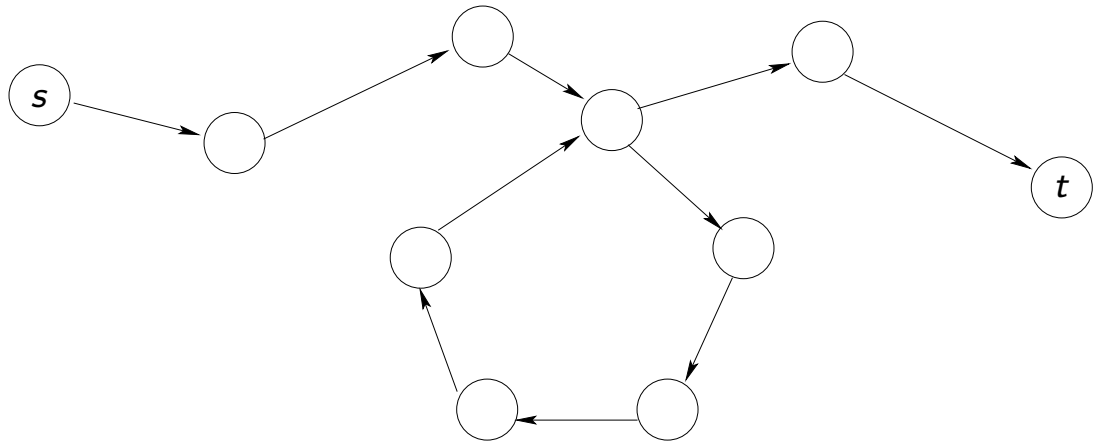
Arcs de poids quelconques :

- algorithme de Bellman-Ford (source unique)
- algorithme de Floyd-Warshall (tous les PCC)

utilisent le principe de la **programmation dynamique**

Plus courts chemins élémentaires

Si le graphe G ne contient pas de circuit de longueur négative, alors il existe un plus court chemin **élémentaire** entre s et t.

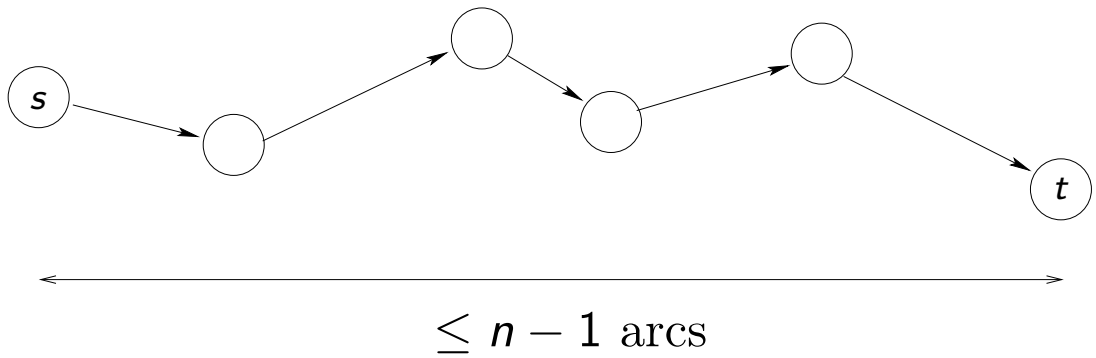


Chemin élémentaire :

- chaque sommet apparaît au plus une fois (pas de cycle dans le chemin)

Plus courts chemins élémentaires

Si le graphe G ne contient aucun cycle de longueur négative, alors il existe un plus court chemin **élémentaire** entre s et t.



Chemin élémentaire :

- au plus n sommets $\Longrightarrow$ au plus $n-1$ arcs

Algorithme de Bellman-Ford

Entrée : un graphe $G = (V, A)$, un sommet source s,

Sortie : les longueurs des plus courts chemins issus de s,

(*constitués de au plus* $n - 1$ *arcs*)

Algorithme de Bellman-Ford

Entrée : un graphe $G = (V, A)$, un sommet source s,

Sortie : les longueurs des plus courts chemins issus de s,

(*constitués de au plus* $n - 1$ *arcs*)

Sous-problèmes $OPT(i, v)$ **:** longueur minimale d'un chemin de s à v **constitué de au plus** i **arcs,**

Objectif final : $OPT(n - 1, v)$

(*plus courts chemins issus de s constitués de au plus* $n - 1$ *arcs :* $OPT(n - 1, v) = \delta(v)$)

Algorithme de Bellman-Ford

Entrée : un graphe $G = (V, A)$, un sommet source s,

Sortie : les longueurs des plus courts chemins issus de s,

(*constitués de au plus* $n - 1$ *arcs*)

Algorithme :

initialiser $OPT(0, v)$ pour tout v,

calculer $OPT(1, v)$ pour tout v,

...

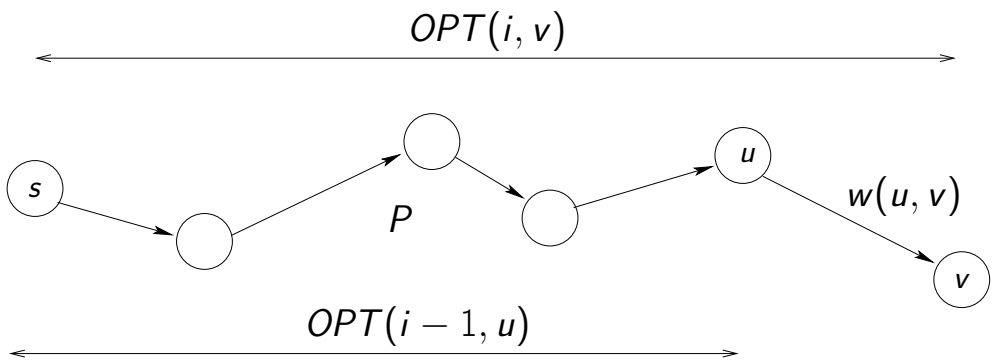
calculer $OPT(n - 1, v)$ pour tout v.

Algorithme de Bellman-Ford

Soit P un chemin optimal pour le sous-problème $OPT(i, v)$

- si P contient au plus $i-1$ arcs alors $OPT(i, v) = OPT(i-1, v)$,
- si P contient i arcs, alors il existe un arc $(u, v) \in A$ tel que

$$OPT(i, v) = OPT(i-1, u) + w(u, v)$$



Algorithme de Bellman-Ford

Définition récursive de $OPT(i, v)$:

$$OPT(0, s) = 0$$

$$OPT(0, v) = +\infty \text{ si } v \neq s,$$

$$OPT(i, v) = \text{MIN}\{\ OPT(i-1, v),\ \text{MIN}_{u \in Adj^{-1}(v)}(OPT(i-1, u) + w(u, v))\ \}$$

si $i \neq 0$

notation : $Adj^{-1}(v) = \{u \in S \text{ tel que } (u, v) \in A\}$

Algorithme de Bellman-Ford

algorithme BELLMAN-FORD(G, s)
in : $G = (S, A, w)$ un graphe pondéré, s un sommet de S,

1 **pour** $v \in S$ **faire** $d[0, v] := \infty$,
2 $d[0, s] := 0$,
3 **pour** $i := 1, \ldots, n-1$ **faire**
4 **pour** $v \in S$ **faire**
5 $d[i, v] := \min\big(d[i-1, v],$
6 $\min_{u \in Adj^{-1}(v)}(d[i-1, u] + w(u, v))\big)$,
7 **fin faire**,
8 **fin faire**,
9 **renvoyer** d,

fin fonction

A l'étape i, on considère chaque arc de G une fois :

Complexité : $O(m \times n)$ $\quad (n = |S|,\ m = |A|)$

Algorithme de Bellman-Ford

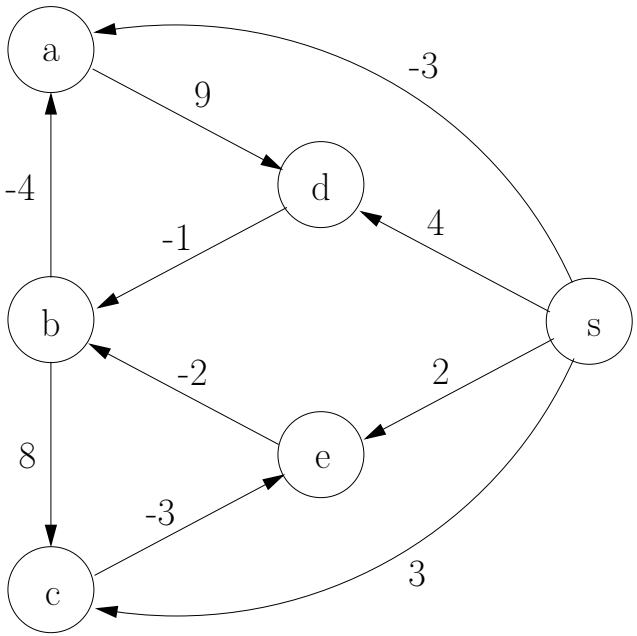
algorithme BELLMAN-FORD(G, s)
in : $G = (S, A, w)$ un graphe pondéré, s un sommet de S,

pour $v \in S$ **faire** $d[0, v] := \infty$,
$d[0, s] := 0$,
pour $i := 1, \ldots, n-1$ **faire**
  pour $v \in S$ **faire** $d[i, v] := d[i-1, v]$,
  pour $(u, v) \in A$ **faire**
    si $d[i, v] > d[i-1, u] + w(u, v)$ **alors**
      $d[i, v] := d[i-1, u] + w(u, v)$,
  fin faire,
fin faire,
renvoyer d,

fin fonction

Peu importe l'ordre dans lequel on considère les arcs.

Algorithme de Bellman-Ford : exemple

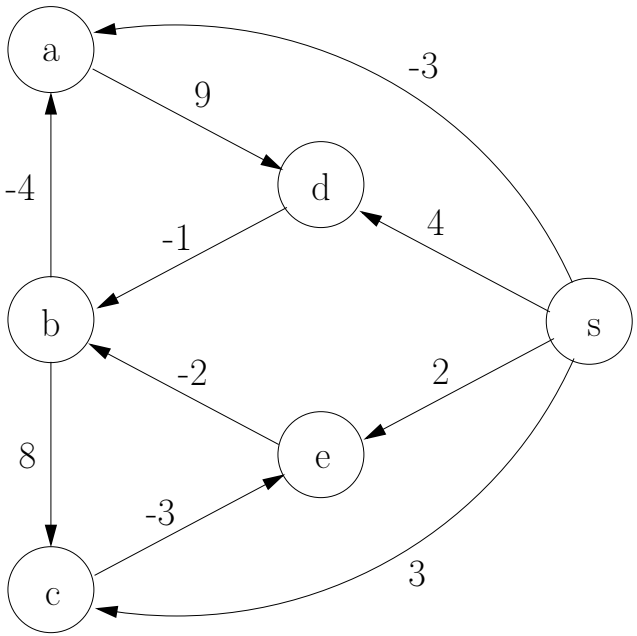


(a)

	0	1	2	3	4	5
s	0					
a	∞					
b	∞					
c	∞					
d	∞					
e	∞					

(b)

Algorithme de Bellman-Ford : exemple

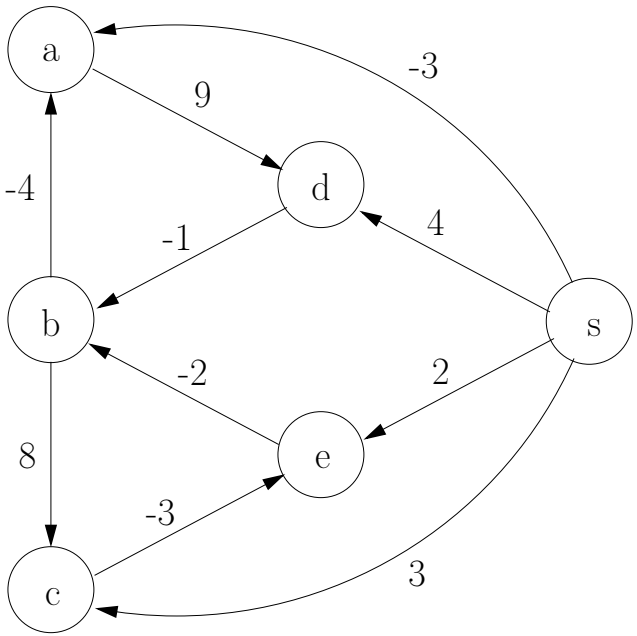


(a)

	0	1	2	3	4	5
s	0	0				
a	∞	-3				
b	∞	∞				
c	∞	3				
d	∞	4				
e	∞	2				

(b)

Algorithme de Bellman-Ford : exemple

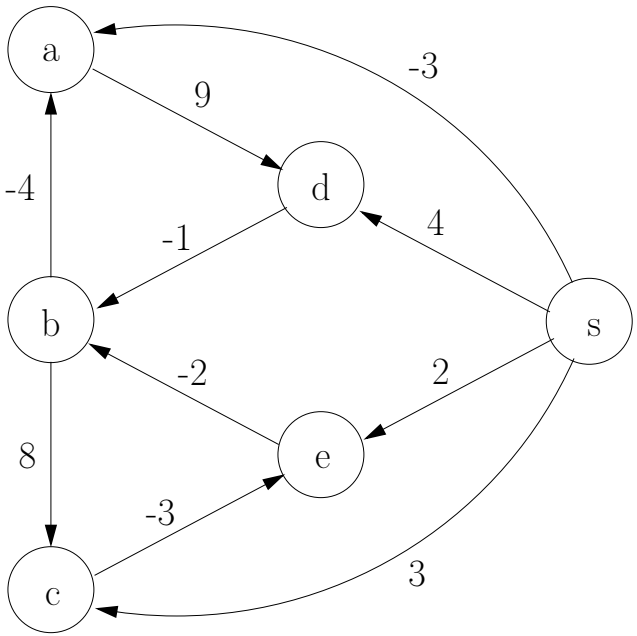


(a)

	0	1	2	3	4	5
s	0	0	0			
a	∞	-3	-3			
b	∞	∞	0			
c	∞	3	3			
d	∞	4	4			
e	∞	2	0			

(b)

Algorithme de Bellman-Ford : exemple

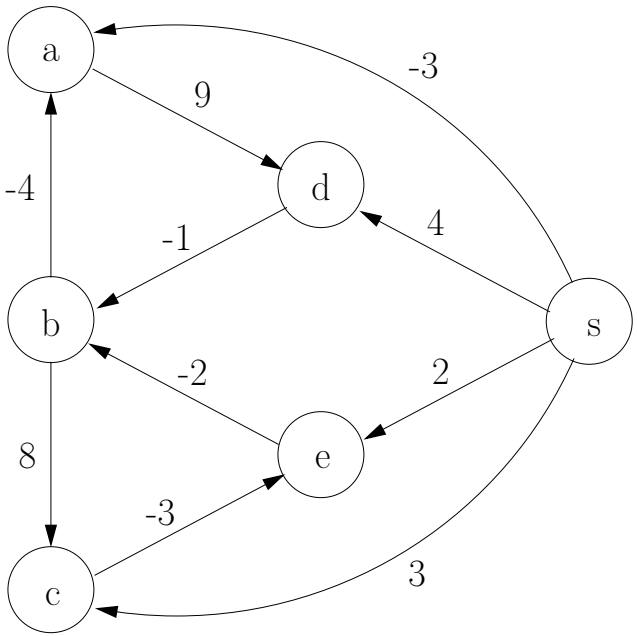


(a)

	0	1	2	3	4	5
s	0	0	0	0		
a	∞	-3	-3	-4		
b	∞	∞	0	-2		
c	∞	3	3	3		
d	∞	4	4	4		
e	∞	2	0	0		

(b)

Algorithme de Bellman-Ford : exemple

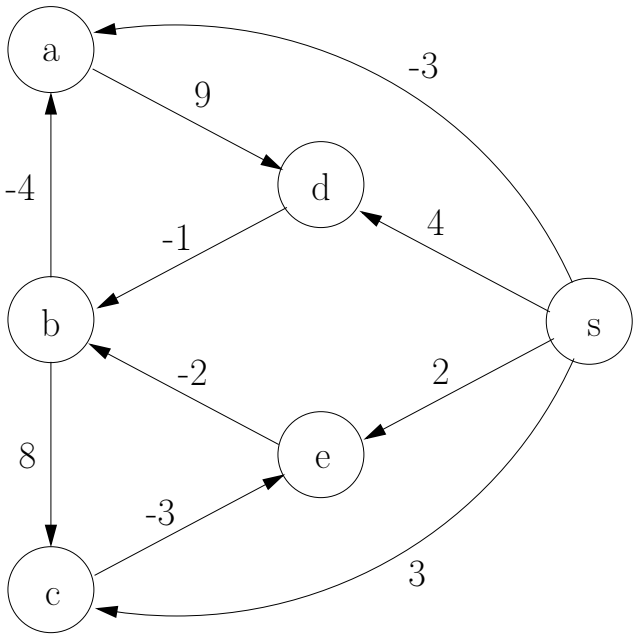


(a)

	0	1	2	3	4	5
s	0	0	0	0	0	
a	∞	-3	-3	-4	-6	
b	∞	∞	0	-2	-2	
c	∞	3	3	3	3	
d	∞	4	4	4	4	
e	∞	2	0	0	0	

(b)

Algorithme de Bellman-Ford : exemple



(a)

	0	1	2	3	4	5
s	0	0	0	0	0	0
a	∞	-3	-3	-4	-6	-6
b	∞	∞	0	-2	-2	-2
c	∞	3	3	3	3	3
d	∞	4	4	4	4	3
e	∞	2	0	0	0	0

(b)

Algorithme de Bellman-Ford : simplification

Tableau à une dimension $D[]$ pour représenter les distances

$D[v]$: longueur d'un chemin entre s et v

Algorithme de Bellman-Ford : simplification

Tableau à une dimension $D[]$ pour représenter les distances

$D[v]$: longueur d'un chemin entre s et v

initialement :

- $\forall v,\ v \neq s,\ D[v] = +\infty$,
- $D[s] = 0$.

Algorithme de Bellman-Ford : simplification

Tableau à une dimension $D[]$ pour représenter les distances

$D[v]$: longueur d'un chemin entre s et v

initialement :

- $\forall v,\ v \neq s,\ D[v] = +\infty$,
- $D[s] = 0$.

mise à jour :

- $D[v] = \text{MIN}\big(D[v], \text{MIN}_{u \in Adj^{-1}(v)}(D[u] + w(u,v))\big)$

Algorithme de Bellman-Ford : simplification

Tableau à une dimension $D[]$ pour représenter les distances

$D[v]$: longueur d'un chemin entre s et v

initialement :

- $\forall v,\ v \neq s,\ D[v] = +\infty$,
- $D[s] = 0$.

mise à jour :

- $D[v] = \text{MIN}\big(D[v], \text{MIN}_{u \in Adj^{-1}(v)}(D[u] + w(u, v))\big)$

Justification : après i mises à jour $D[v] \leq d[i, v]$.
$D[v]$ est **au plus** la longueur d'un plus court chemin entre s et v contenant au plus i arcs.

Algorithme de Bellman-Ford : simplification

algorithme BELLMAN-FORD(G, s)
in : $G = (S, A, w)$ un graphe pondéré, s un sommet de S,

pour $v \in S$ **faire** $D[v] := \infty$,
$D[s] := 0$,
pour $i := 1, \ldots, n-1$ **faire**
pour $v \in S$ **faire**
$D[v] := \text{MIN}\big(D[v], \text{MIN}_{u \in Adj^{-1}(v)}(D[u] + w(u, v))\big)$,
fin faire,
fin faire,
renvoyer D,

fin fonction

Espace : $O(n)$

Algo. de Bellman-Ford : reconstruction des PCC

si $D[v]$ est modifiée à l'étape i :

$$D[v] := D[u] + w(u, v)$$

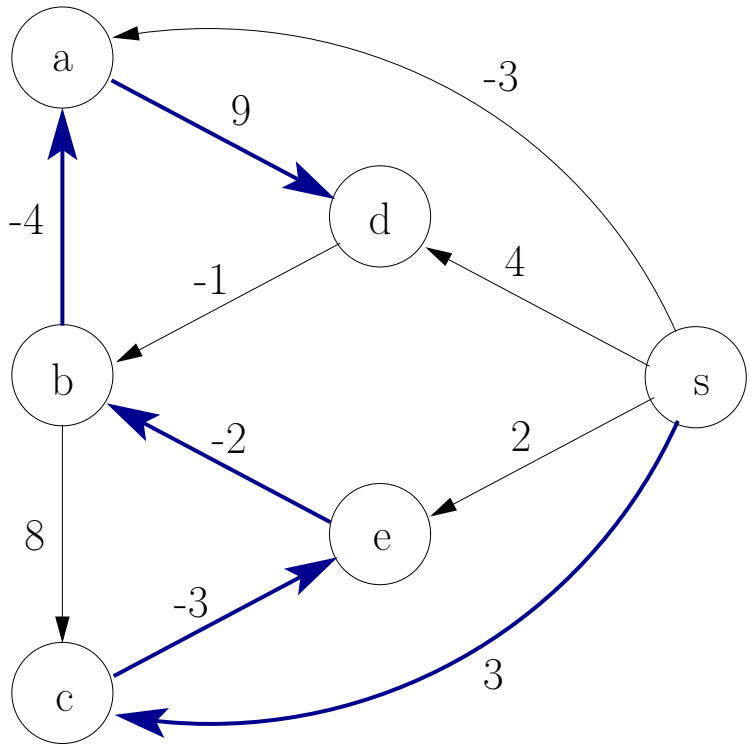
alors mémoriser u, dernier sommet intermédiaire

$$pred[v] := u$$

quand l'algorithme est terminé :

- le graphe constitué des arcs $(pred[v], v)$ est un arbre,
- dans ce graphe, le chemin entre s et v est un plus court chemin dans G.

Algo. de Bellman-Ford : reconstruction des PCC



	s	a	b	c	d	e
D	0	-6	-2	3	3	0
pred	—	b	e	s	a	c

Algorithme de Bellman-Ford

algorithme BELLMAN-FORD(G, s)
in : $G = (S, A, w)$ un graphe pondéré, s un sommet de S,

pour $v \in S$ **faire**
  $D[v] := \infty$,
  $pred[v] := NONE$,
fin faire,
$D[s] := 0$,
pour $i := 1, \ldots, n-1$ **faire**
  pour $(u, v) \in A$ **faire**
    si $D[v] > D[u] + w(u, v)$ **alors**
      $D[v] := D[u] + w(u, v)$,
      $pred[v] := u$,
    fin si,
  fin faire,
fin faire,
renvoyer $(D, pred)$,
fin algo

Routage dans les réseaux de communication

Une application importante des algorithmes de plus courts chemins :
Déterminer le meilleur chemin pour router les messages

Graphe	Réseau de Communications
Sommets	Routeurs
Arcs	Lignes de communications
Longueurs	Délais

Algorithme de Dijkstra : fonctionnement global

choix du sommet le plus proche parmis tous les sommets.

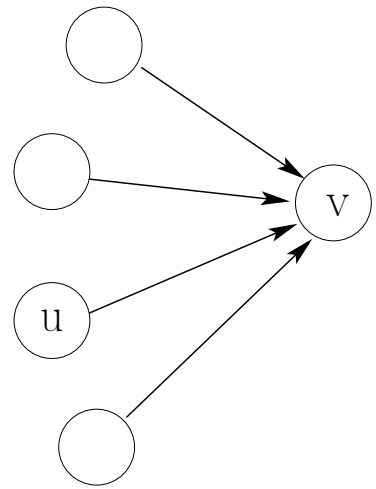
Algorithme de Bellman-Ford : fonctionnement local

chaque noeud a juste besoin de connaître les marques de ses voisins.

Bellman-Ford : amélioration pour le routage

Calcul de $D[v]$ à l'étape i :

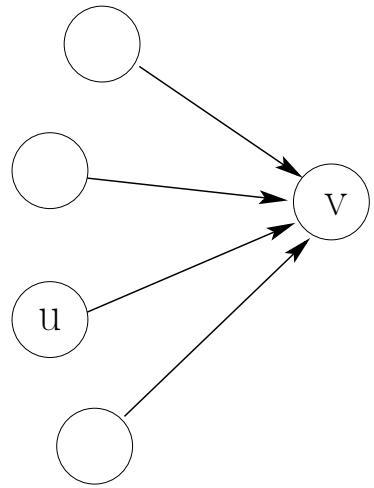
- calcul à partir des distances $D[u]$ des voisins. *Si aucune distance n'a changé le calcul est inutile.*



Bellman-Ford : amélioration pour le routage

Calcul de $D[v]$ à l'étape i :

- calcul à partir des distances $D[u]$ des voisins. *Si aucune distance n'a changé le calcul est inutile.*



- **nouvelle approche :** chaque sommet dont la distance change informe ses voisins.
- si aucune distance n'est modifiée alors **l'algorithme s'arrête.**

Bellman-Ford : amélioration pour le routage

algorithme BELLMAN-FORD-R1(G, s) $G = (S, A, w)$

  pour chaque $v \in S$ **faire** $D[v] := \infty$,
  $D[s] := 0$,
  pour $i := 1, \ldots, n-1$ **faire**
    pour chaque $u \in S$ tel que $D[u]$ a changé à l'étape $i-1$ **faire**
      pour chaque $(u, v) \in A$ faire
        $D[v] := \min\big(D[v], D[u] + w(u, v)\big)$,
    si D n'a pas été modifié à l'étape i **alors**
      break,
  fin faire,
  renvoyer $(D, pred)$,
fin algo

Bellman-Ford : amélioration pour le routage

Notifications :

- avertir les noeuds voisins d'un changement,
- les notifications ne sont pas synchronisées,
- version *asynchrone* de l'algorithme : noeuds actifs/inactifs.

Bellman-Ford : amélioration pour le routage

algorithme BELLMAN-FORD-Asynchrone(G, s) $\quad G = (S, A, w)$

pour chaque $v \in S$ **faire**
$D[v] := \infty$, $pred[v] := NONE$,
$D[s] := 0$,
$F := \{s\}$, *F : ensemble des noeuds actifs*

tant que $F \neq \emptyset$ **faire**
choisir et enlever un sommet u de F,
pour chaque arc (u, v) **faire**
si $D[v] > D[u] + w(u, v)$ **alors**
$D[v] := D[u] + w(u, v)$,
$pred[v] := u$,
$F := F \cup \{v\}$, *v devient actif*
fin si,
fin pour,
fin tant que,
renvoyer $(D, pred)$,

fin algo