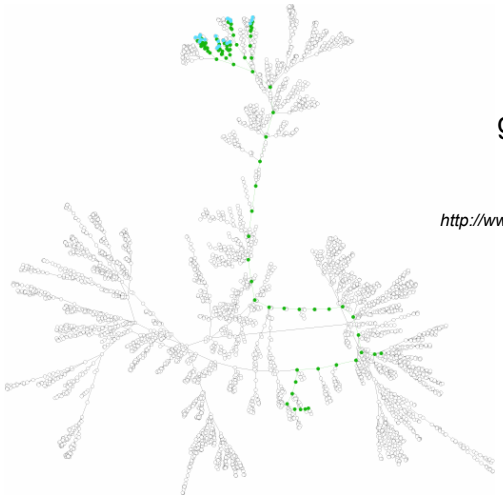
Plus courts chemins



graphe de solution
d'un puzzle
rush hour
http://www.puzzles.com/products/rushhour.htm

Plus courts chemins

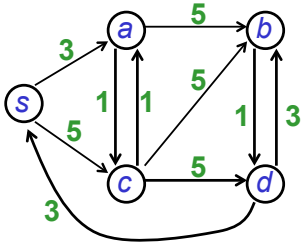
Chemins de même origine (ou même destination)
(selon les hypothèses)
algorithme de Dijkstra O(card S^2)
ou O(card S + card A.log card S)
algorithme de Bellman-Ford O(card A.card S)

Toutes paires d'états
algorithme de Floyd-Warshall O(card S^3)

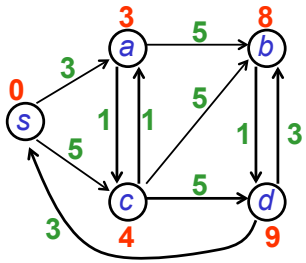
Problème

Graphe valué : $G = (S, A, v)$ avec valuation $v : A \rightarrow \mathbf{R}$
Source du graphe : $s \in S$
Problème : pour tout $t \in S$ calculer
$\delta(s, t) = \min \{ v(c) ; c \text{ chemin de } s \text{ à } t \} \cup \{+\infty\}$

Exemple

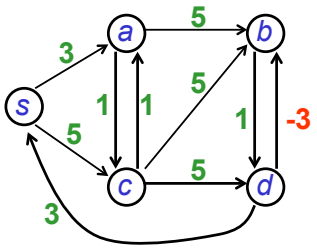


Coûts δ



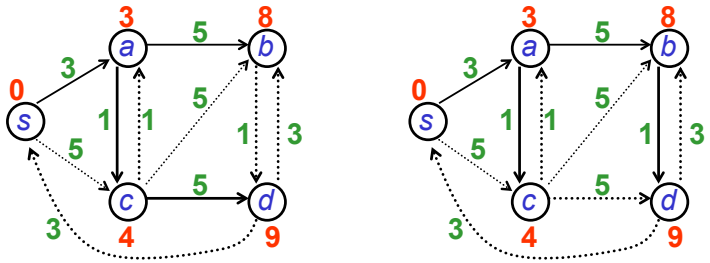
Existence

Proposition
pour tout $t \in S$ $\delta(s, t) > -\infty$ **ssi**
le graphe n'a pas de circuit de coût < 0
accessible depuis s



Arbres des plus courts chemins

Arbres de racine *s*
dont les branches sont des chemins de coût minimal



Propriétés de base

Propriété 1 : $G = (S, A, v)$
soit c un plus court chemin de p à r
dont l'avant-dernier sommet est q
Alors $\delta(p, r) = \delta(p, q) + v(q, r)$

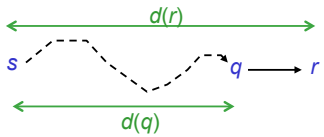
Propriété 2 : $G = (S, A, v)$
soit c un chemin de p à r
dont l'avant-dernier sommet est q
Alors $\delta(p, r) \leq \delta(p, q) + v(q, r)$

Relaxation

Calcul des $\delta(s,t)$ par approximations successives
$t \in S$ $d(t)$ = estimation de $\delta(s, t)$
$\pi(t)$ = prédécesseur de t : avant-dernier sommet
d'un chemin de s à t ayant pour coût $d(t)$

Initialisation de d et π

```
INIT
    pour chaque  t ∈ S faire
    { d(t) ← ∞ ;  π(t) ← nil ; }
    d(s) ← 0 ;
```



Relaxation de l 'arc (q, r)

```
RELAX(q, r)
    si  d(q) + v(q, r) < d(r)
    alors { d(r) ← d(q) + v(q, r) ;  π(r) ← q ; }
```


Relaxation (suite)

Proposition :
la propriété « pour tout $t \in S$ $d(t) \geq \delta(s, t)$ »
est un invariant de **`relax`**

Preuve par induction sur le nombre
d 'exécution de **`relax`**

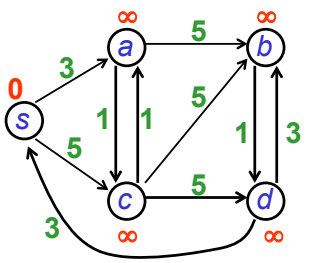
Algorithme de Dijkstra

Condition : $v(p, q) \geq 0$ pour tout arc (p, q)

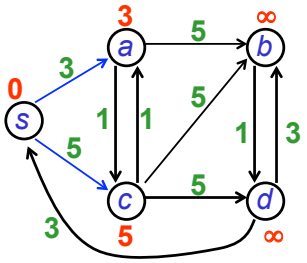
```
début
     INIT ;
     Q ← S ;
     tant que Q ≠ ∅ faire {
          q ← MIN_d(Q) ; Q ← Q - {q} ;
          pour chaque r successeur de q faire
               RELAX(q, r) ;
     }
fin
```

Algorithme glouton (séquentiel)

Exemple

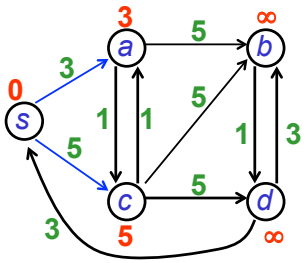


$Q = \{ s, a, b, c, d \}$
$\pi(s) = nil$
$\pi(a) = nil$
$\pi(b) = nil$
$\pi(c) = nil$
$\pi(d) = nil$

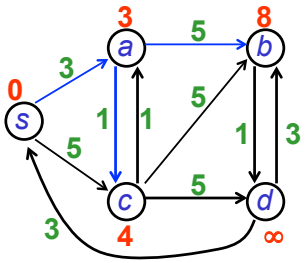


$Q = \{ a, b, c, d \}$
$\pi(s) = nil$
$\pi(a) = s$
$\pi(b) = nil$
$\pi(c) = s$
$\pi(d) = nil$

Exemple (suite)

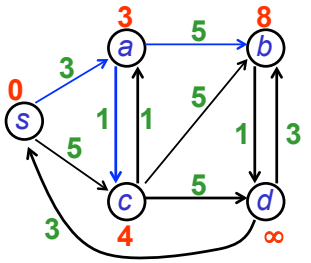


$Q = \{ a, b, c, d \}$
$\pi(s) = nil$
$\pi(a) = s$
$\pi(b) = nil$
$\pi(c) = s$
$\pi(d) = nil$

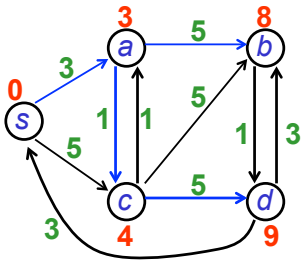


$Q = \{ b, c, d \}$
$\pi(s) = nil$
$\pi(a) = s$
$\pi(b) = a$
$\pi(c) = a$
$\pi(d) = nil$

Exemple (suite)

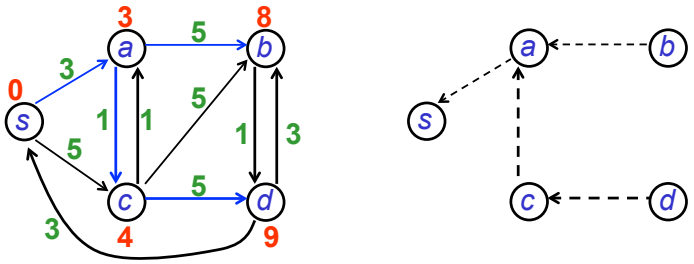


$Q = \{ b, c, d \}$
$\pi(s) = nil$
$\pi(a) = s$
$\pi(b) = a$
$\pi(c) = a$
$\pi(d) = nil$



$Q = \{ b, d \}$
$\pi(s) = nil$
$\pi(a) = s$
$\pi(b) = a$
$\pi(c) = a$
$\pi(d) = c$

Exemple (suite)



Q = { b, d }, Q = { d } puis $Q = \varnothing$

$\pi(s)$ = nil

$\pi(a) = s$

$\pi(b) = a$

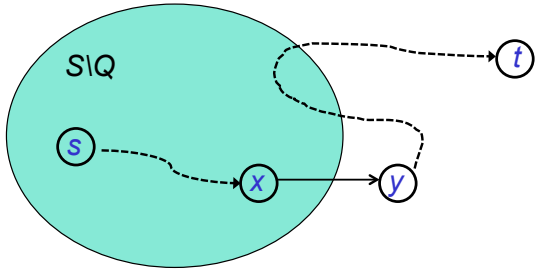
$\pi(c) = a$

$\pi(d) = c$

Validité de l'algorithme de Dijkstra

Proposition : Après l'exécution de l'algorithme de Dijkstra sur un graphe $G = (S, A, v)$, $d(t) = \delta(s, t)$ pour tout $t \in S$.

Démonstration par contradiction: soit $d(t) \neq \delta(s, t)$



Implémentation

Par matrice d'adjacence
temps global O(card S^2)

Par listes de successeurs
Q : file de priorité (tas)
card S opérations $\mathbf{MIN}_d$: O(card S.log card S)
card A opérations **RELAX** : O(card A.log card S)
temps global O((card S+card A).log card S)

Algorithme de Bellman-Ford

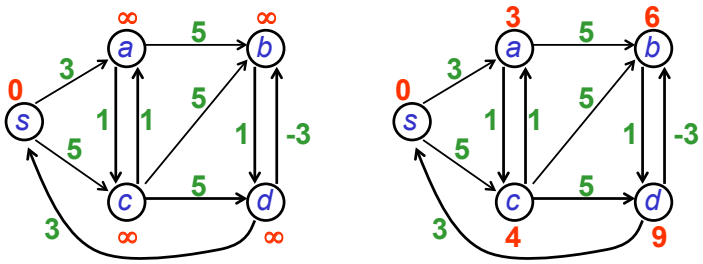
Aucune condition : pour tout arc (p, q), $v(p, q) \in \mathbf{R}$

```
début
      INIT;
      Q ← S ;
      répéter card S-1 fois
            pour chaque (q, r) ∈ A faire
                  RELAX(q, r) ;
      pour chaque (q, r) ∈ A faire
            si d(q) + v(q, r) < d(r) alors
                  retour « cycle de coût négatif »
            sinon
                  retour « coûts calculés »
fin
```

Temps : O(card S.card A)

Exemple 1

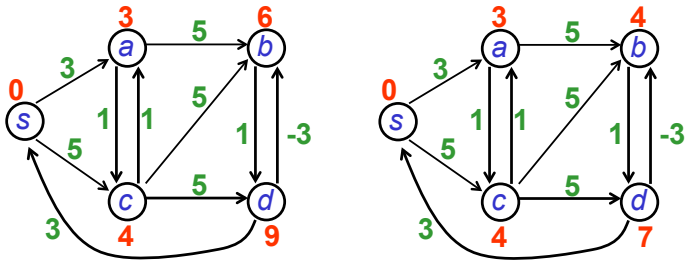
UMLV ©



Étape 1 relaxation de tous les arcs dans l'ordre :
(*s*,*a*) (*s*,*c*) (*a*,*b*) (*a*,*c*) (*b*,*d*) (*c*,*a*) (*c*,*b*) (*c*,*d*) (*d*,*b*) (*d*,*s*)

Exemple 1 (suite)

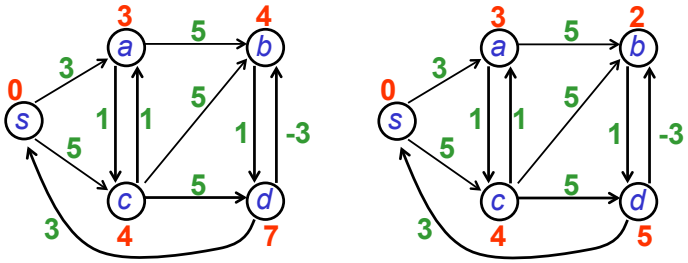
UMLV ©



Étape 2 relaxation de tous les arcs dans l'ordre :
(*s*,*a*) (*s*,*c*) (*a*,*b*) (*a*,*c*) (*b*,*d*) (*c*,*a*) (*c*,*b*) (*c*,*d*) (*d*,*b*) (*d*,*s*)

Exemple 1 (suite)

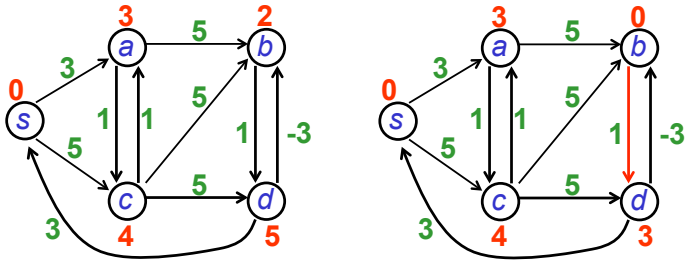
UMLV ©



Étape 3 relaxation de tous les arcs dans l'ordre :
(*s*,*a*) (*s*,*c*) (*a*,*b*) (*a*,*c*) (*b*,*d*) (*c*,*a*) (*c*,*b*) (*c*,*d*) (*d*,*b*) (*d*,*s*)

Exemple 1 (suite)

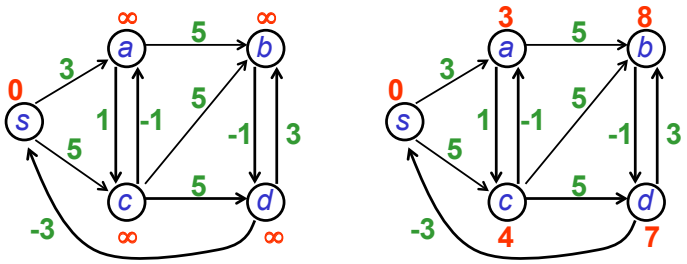
UMLV ©



Étape 4 relaxation de tous les arcs dans l'ordre :
(*s*,*a*) (*s*,*c*) (*a*,*b*) (*a*,*c*) (*b*,*d*) (*c*,*a*) (*c*,*b*) (*c*,*d*) (*d*,*b*) (*d*,*s*)

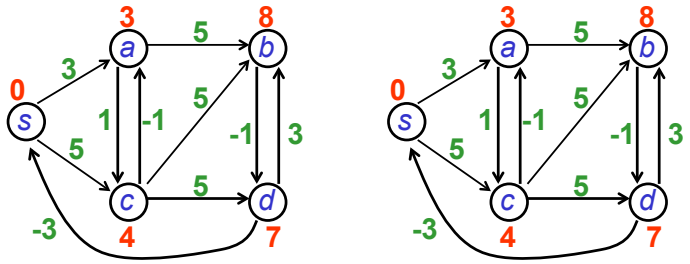
relaxation possible : cycle de coût négatif

Exemple 2



Étape 1 relaxation de tous les arcs dans l'ordre :
(*s*,*a*) (*s*,*c*) (*a*,*b*) (*a*,*c*) (*b*,*d*) (*c*,*a*) (*c*,*b*) (*c*,*d*) (*d*,*b*) (*d*,*s*)

Exemple 2 (suite)



Étape 2 relaxation de tous les arcs dans l'ordre :
(*s*,*a*) (*s*,*c*) (*a*,*b*) (*a*,*c*) (*b*,*d*) (*c*,*a*) (*c*,*b*) (*c*,*d*) (*d*,*b*) (*d*,*s*)

pas de réduction possible : coûts corrects

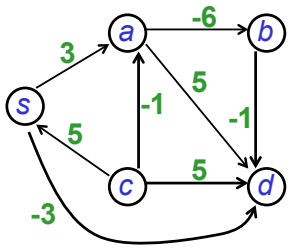
Graphes acycliques

Aucune condition : pour tout arc (*p*, *q*), $v(p, q) \in \mathbf{R}$
Calcul après ordre topologique

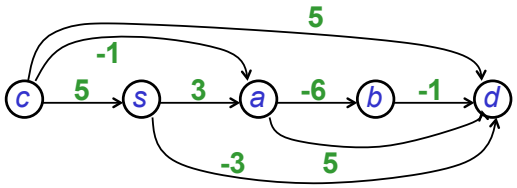
```
début
      INIT;
      pour chaque q ∈ S en ordre topologique faire
            pour chaque r successeur de q faire
                  RELAX(q, r) ;
fin
```

Temps : O(card *S* + card *A*)
chaque sommet et chaque arc est examiné une fois

Exemple

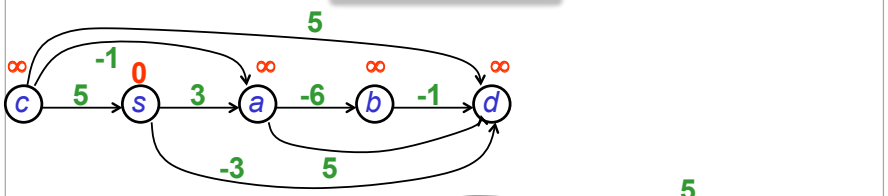


Ordre topologique
c, *s*, *a*, *b*, *d*

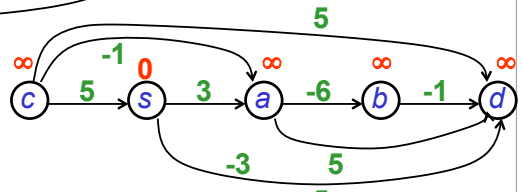


Calcul des coûts

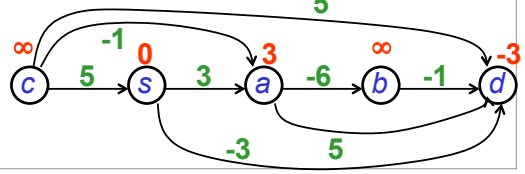
UMLV ©



Examen de *c*



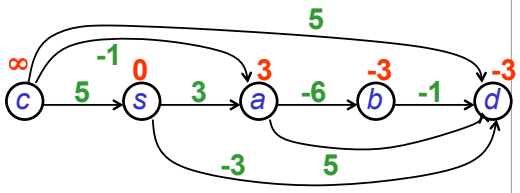
Examen de *s*



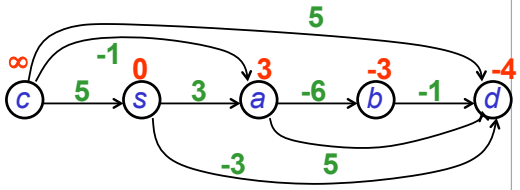
Calcul des coûts (suite)

UMLV ©

Examen de *a*



Examen de *b*



Examen de *d* inutile