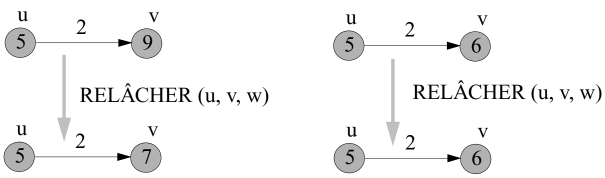
Les plus courts chemins.

représentation des plus courts chemins (graphe des prédécesseurs)

relâchement

```
RELÂCHER (u, v, w)
si d(v) > d(u) + w(u,v) alors
        d(v) ← d(u) + w(u,v)
        PARENT (v) ← u
```




```
INITIALISER (G, s)
pour chaque sommet s de G
        faire   d(s) ← ∞
                PARENT (u) ← NIL
d(s) ← 0
```

Propriétés des plus courts chemins et du relâchement


```
PLUS-COURT-SANS-POIDS (G, s)
INITIALISER (G, s)
file F ← vide
ENFILER (F, s)
tant que F ≠ vide
        u ← DEFILER (F)
        pour chaque sommet v adjacent à u faire
                RELÂCHER (u, v, 1)
                ENFILER (F, v)
```

```
PLUS-COURT-SANS-POIDS (G, s)
INITIALISER (G, s)
file F ← vide
ENFILER (F, s)
tant que F ≠ vide
        u ← DEFILER (F)
        pour chaque sommet v adjacent à u faire
                si d(v) = ∞ alors
                        d(v) ← d(v) + 1
                        PARENT (v) ← u
                        ENFILER (F, v)
```


Plus courts chemins dans les graphes admettant les arcs à poids négatifs.

cycles négatifs

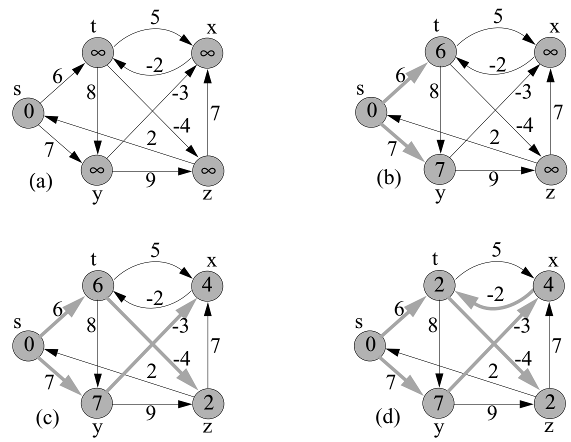
```
BELLMAN-FORD(G, w, s)
INITIALISER (G, s)
pour i ← 1 à |S[G]|-1faire
        pour chaque arc (u, v) ∈ A[G] faire
                RELÂCHER (u, v, w)
pour chaque arc (u, v) ∈ A[G] faire
        si d(v) > d(u) + w(u,v) alors
                retourner FAUX
retourner VRAI
```

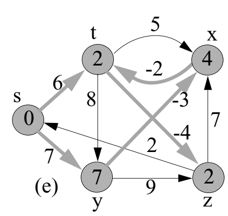
validité de l'algorithme de BELLMAN-FORD

compléxité

Plus courts chemins dans les graphes orientés sans circuit

```
PLUS-COURT-CHEMINS-GSS (G, s)
trier topolgiquement les sommets de G
INITIALISER (G, s)
pour chaque sommet u pris dans l'ordre topolgique faire
        pour chaque sommet v adjacent à u faire
                RELÂCHER (u, v, w)
```



L'ordre de relâchement dans chaque phase:
(t,x), (t,y), (t,z), (x,t), (y,x), (y,z), (z,x), (z,s), (s,t), (s,y)

Algorithme de Dijkstra

```
DIJKSTRA (G, W, s)
1   INITIALISER (G, s)
2   E ← vide
3   F ← S[G]
4   tant que F ≠ vide faire
5         u ← EXTRAIRE-MIN(F)
6         E ← E ∪ {u}
7         pour chaque sommet v adjacent à u faire
8               RELÂCHER (u, v, w)
```

file de priorité F

- 3 F ← S[G] - INSÉRER
- 8 RELÂCHER (u, v, w) implique DIMINIUER-CLÉ

validité

compléxité

Plus courts chemins pour tout couple de sommets

matrices de liaison


```
EXTENSION-PLUS-COURTS-CHEMINS(D, W)
n ← lignes[D]
soit D' = (d'_ij) une matrice n par n
pour i ← 1 à n faire
   pour j ← 1 à n faire
      d'_ij ← ∞
      pour k← 1 à n faire
         d'_ij ← min (d'_ij, d'_ik + w'_kj)
retourner D'
```

matrices de liaison de longueur supérieure, matrices d'accès

```
PLUS-COURT-CHEMIN-TOUS-COUPLES-RELENTI (W)
n ← lignes[D]
D^(1) ← W
pour m ← 2 à n-1 faire
D^(m) ← EXTENSION-PLUS-COURTS-CHEMINS (D^(m-1) , W)
retourner D^(n-1)
```

```
PLUS-COURT-CHEMIN-TOUS-COUPLES-ACCÉLÉRÉ (W)
n ← lignes[D]
D^(1) ← W
m ← 1
tant que m < n-1 faire
   D^(2m) ←
      EXTENSION-PLUS-COURTS-CHEMINS (D^(m) , D^(m))
m ← 2m
   retourner D^(m)
```