COURS 6 : LE PLUS COURT CHEMIN DANS LE GRAPHE 3

I. RAPPEL :

A. ALGORITHME NOUYAU

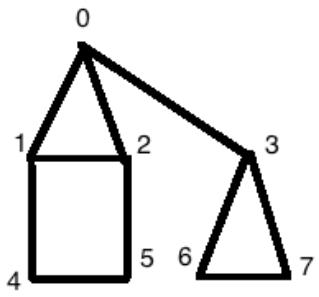
```
while Q:
        z = Q.delete()
        for x in G[z]:
                if (z,x) <- tense :
                        relax(z,x)
```

B. NON PONDERE (cf pcc_nn_pondere.py) :

```
P = []
for i in G:
        P.append(-1)

Q = []
P[s] = s
Q.append(s)
while Q:
        z = Q.pop()
        for x in G[z]:
                if x == t:
                        P[t] = z
                        print "Le pcc entre ", s ,"et", t est :"
                        return path(P,s,t)

                elif P[x] == -1 :
                        P[x] = z
                        Q.append(x)

print "il n'existe pas de pcc entre le point", s ,"et le point",
t
return -1
```

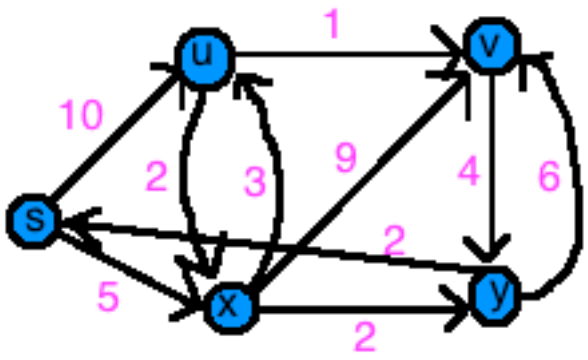
METHODE 2 :

```
if P[x] == -1 :
                P[x]=z
                Q.append(x)


return path(P,s,t)
```



La méthode 1 est plus rapide et effectue moins de calcul
La méthode 2 est «plus joli» mais effectue plus de calcul : elle permet de calculer le chemin de chaque sommet avec le sommet s.

C. PONDERE AVEC POIDS POSITIFS :Algorithme de Dijkstra :



1. AVEC FILE (cf dijkstra_file.py):

```
Q.append(s)
while Q :
        z = deleteMin(Q,d)
        for x in G[z] :
                tense = d[z] + w(z,x)
                if d[x] > tense :
                        d[x] = tense
                        if x∉Q:
                                Q.append(x)
```

COMPLEXITE

o(V)

o(E)

= o(V.E)

2. AVEC TAS (cf dijkstra_tas.py):

```
Q= makeheap()
Q.heappush(s)
while Q :
        z = heappop()
        for x in G[z] :
                tense = d[z] + w(z,x)
                if d[x] > tense :
                        d[x] = tense
                        if x∉Q:
                                Q.heappush(x)
                        else Q.modifie(x,d[x])
```

COMPLEXITE

o(log(V))

o(log(V)

o(E)

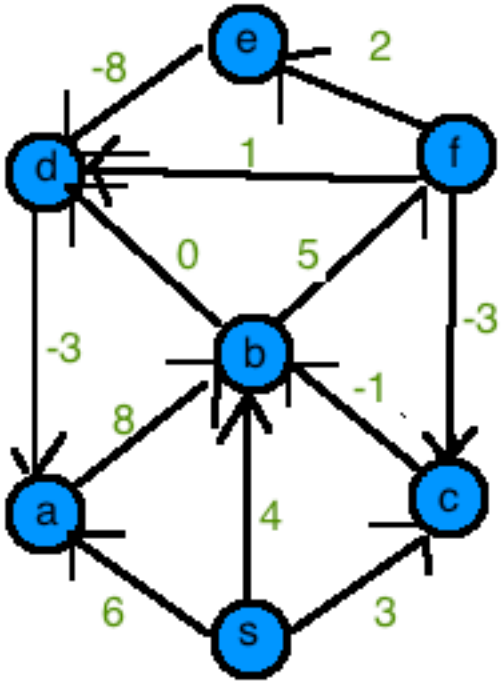
= o(V*log(V) + E*log(V))
= o(V+E)log(V)

II. BELLMAN FORD : PONDERE avec poids positif ou negatif

```
M = [ 0 for u in G]
Dist = [ infini for u in G]
M[s]  = 1
Dist[s] = 0
Q.enqueue(s)
P[s] = s

while Q :
        z = Q.dequeue(s)
        M[z] = 0   #permet de savoir si est dans Q ou pas si z y est alors M[z] = 1 sinon M[z] = 0
        for x in Adj[z] :
                if Dist [x] ==  Dist[z] + w(z, x) :
                        Dist [x] =  Dist[z] + w(z, x)
                        P[x] = z
                        if M[x] = 0 :
                                M[x] = 1
                                Q.enqueue(x)
```

Exemple :

Dist : [s, a, b, c, d, e, f]



Q : ~~s~~ Dist : [0, 'inf','inf', 'inf', 'inf', 'inf', 'inf']	Q : ~~e~~, d, f Dist : [0, 1, 2, 3, 2, 11, 7]
Q : ~~a~~, b, c Dist : [0, 6, 4, 3, 'inf', 'inf', 'inf']	Q : ~~d~~, f Dist : [0, 1, 2, 3, 2, 11, 7]
Q : ~~b~~,c Dist : [0, 6, 4, 3, 'inf', 'inf', 'inf']	Q : ~~f~~, a Dist : [0, -1, 2, 3, 2, 11, 7]
Q : ~~c~~, d, f Dist : [0, 6, 4, 3, 4, 'inf', 9]	Q : [~~a~~, e] Dist : [0, -1, 2, 3, 2, 9, 7]
Q : ~~d~~, f, b Dist : [0, 6, 2, 3, 4, 'inf', 9]	Q : [e] Dist : [0, -1, 2, 3, 2, 9, 7]
Q : ~~f~~, b, a Dist : [0, 1, 2, 3, 4, 'inf', 9]	Q : [~~d~~] Dist : [0, -1, 2, 3, 1, 9, 7]
Q : ~~b~~, a, e Dist : [0, 1, 2, 3, 4, 11, 9]	Q : [a] Dist : [0, -2, 2, 3, 1, 9, 7]
Q : ~~a~~, e, d, f Dist : [0, 1, 2, 3, 2, 11, 7]	->EN RESULTAT <- Dist : [0, -2, 2, 3, 1, 9, 7]

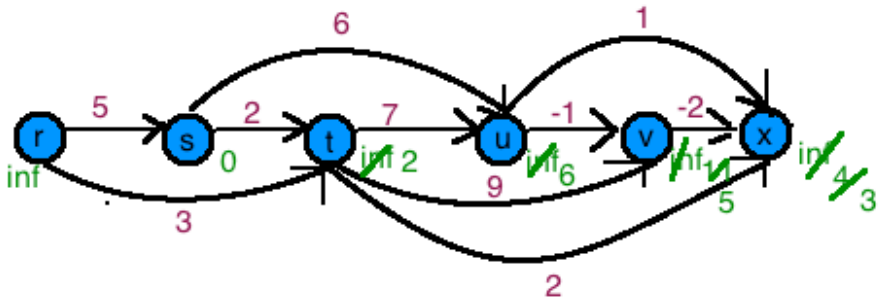
III. SHORTEST PATH DAGS(G,W,S)

ne peut s'appliquer que dans un graphe sans cycle

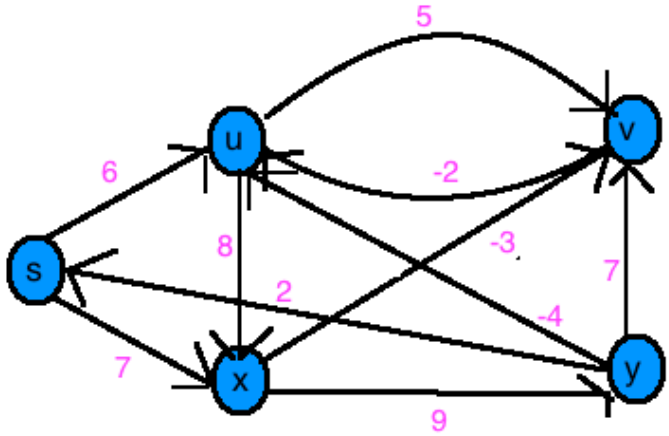
```
Q = TriTopologique(V)                    -> o(v+E)
While Q :
        z = Q.delete()
        for x in adj[z] :
                if (z,x) <- tense :          ->o(E)      ->o(V+E)
                    relax(z,x, α)
```

= o(V+E)

~~s~~
~~t~~u
~~u~~vx
~~v~~x
~~x~~



IV. Exercice : Ecrire le parcours Bellman-ford du graphe suivant :



	s	y	u	x	v	Q
Dist	0	∞	∞	∞	∞	s
	0	∞	6	7	∞	~~u~~x
	0	2	6	7	11	~~x~~vy
	0	2	2	7	4	~~v~~yu
	0	2	2	7	4	~~y~~u
	0	-2	2	7	4	~~u~~y
	0	-2	2	7	4	~~y~~