Recherche Operationnelle : Introduction aux graphes

Vincent GRIPON

ENIB

2 décembre 2009

Algorithme Roy-Warshall (Floyd-Warshall):

Objectif

- Trouver l'ensemble des chemins les plus courts reliant des sommets deux à deux dans un graphe orienté valué ne contenant pas de cycle absorbant.

Cycles absorbants

- Cycle de poids strictement négatif.
- L'existence d'un tel cycle est en contradiction avec l'existence de chemins les plus courts...
- Donc Roy-Warshall fonctionne dès que le problème est bien posé !

Idée

- Démarche itérative en grossissant progressivement l'ensemble des sommets par lesquels l'algorithme s'autorise à passer.

Formalisation

Notations

- On notera $SP(i,j,k)$ la longueur du plus court chemin reliant i à j et passant uniquement par des sommets de numéro $\leq k$,
- Pour rappel, $\delta(i,j)$ est la valeur associée à l'arrête reliant le sommet i au sommet j ($\delta(i,j) = +\infty$ si pas d'arrête).

Identités

- $SP(i,j,0) = \delta(i,j)$,
- $SP(i,j,k+1) = \min\left(SP(i,j,k), SP(i,k+1,k) + SP(k+1,j,k)\right)$.

Algorithme

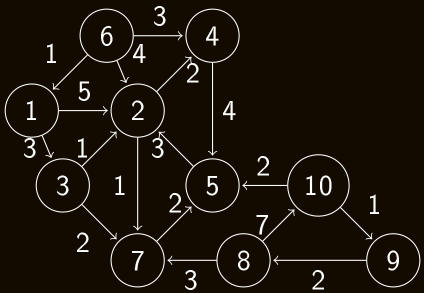
En langage

```
sp = matrice(n,n)

pour i de 1 à n
  pour j de 1 à n
   sp(i,j)<-delta(i,j)

pour k de 1 à n
 pour i de 1 à n
  pour j de 1 à n
   sp(i,j)<-
    min
      sp(i,j)
      sp(i,k)+sp(k,j)
```

Exemple



k=0	1	2	3	4	5	6	7	8	9	10
1		5	3							
2				2			1			
3		1					2			
4					4					
5		3								
6	1	4		3						
7					2					
8							3			7
9								2		
10					2				1	

Algorithme

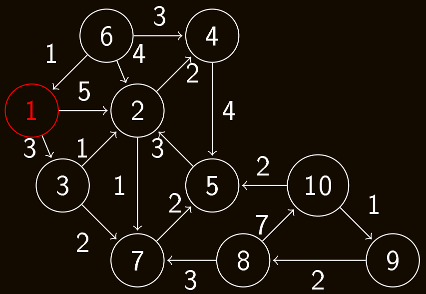
En langage

```
sp = matrice(n,n)

pour i de 1 à n
  pour j de 1 à n
   sp(i,j)<-delta(i,j)

pour k de 1 à n
 pour i de 1 à n
  pour j de 1 à n
   sp(i,j)<-
    min
      sp(i,j)
      sp(i,k)+sp(k,j)
```

Exemple



k=1	1	2	3	4	5	6	7	8	9	10
1		5	3							
2				2			1			
3		1					2			
4					4					
5		3								
6	1	4	4	3						
7					2					
8							3			7
9								2		
10					2				1	

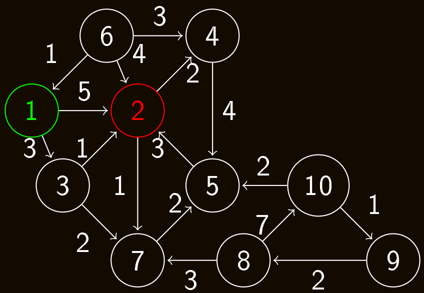
En langage

```
sp = matrice(n,n)

pour i de 1 à n
  pour j de 1 à n
   sp(i,j)<-delta(i,j)

pour k de 1 à n
 pour i de 1 à n
  pour j de 1 à n
   sp(i,j)<-
    min
     sp(i,j)
     sp(i,k)+sp(k,j)
```

Exemple



k=2	1	2	3	4	5	6	7	8	9	10
1		5	3	7			6			
2				2			1			
3		1		3			2			
4					4					
5		3		5			4			
6	1	4	4	3			5			
7					2					
8							3			7
9								2		
10					2				1	

Algorithme

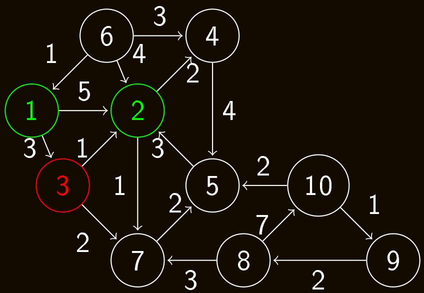
En langage

```
sp = matrice(n,n)

pour i de 1 à n
  pour j de 1 à n
   sp(i,j)<-delta(i,j)

pour k de 1 à n
 pour i de 1 à n
  pour j de 1 à n
   sp(i,j)<-
    min
      sp(i,j)
      sp(i,k)+sp(k,j)
```

Exemple



k=3	1	2	3	4	5	6	7	8	9	10
1		4(5)	3	5(7)			6			
2				2			1			
3		1		3			2			
4					4					
5		3		5			4			
6	1	4	4	3			5			
7					2					
8							3			7
9								2		
10					2				1	

Algorithme

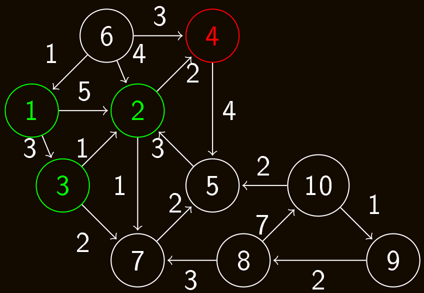
En langage

```
sp = matrice(n,n)

pour i de 1 à n
  pour j de 1 à n
   sp(i,j)<-delta(i,j)

pour k de 1 à n
 pour i de 1 à n
  pour j de 1 à n
   sp(i,j)<-
    min
      sp(i,j)
      sp(i,k)+sp(k,j)
```

Exemple



k=4	1	2	3	4	5	6	7	8	9	10
1		4	3	6	10		5			
2				2	6		1			
3		1		3	7		2			
4					4					
5		3		5	9		4			
6	1	4	4	3	7		5			
7					2					
8							3			7
9								2		
10					2				1	

Algorithme

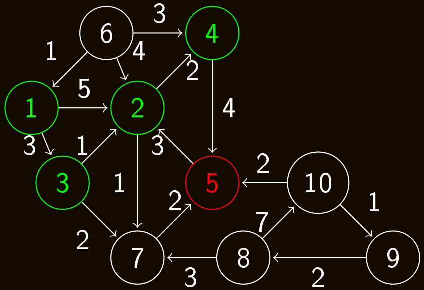
En langage

```
sp = matrice(n,n)

pour i de 1 à n
  pour j de 1 à n
   sp(i,j)<-delta(i,j)

pour k de 1 à n
 pour i de 1 à n
  pour j de 1 à n
   sp(i,j)<-
    min
      sp(i,j)
      sp(i,k)+sp(k,j)
```

Exemple



k=5	1	2	3	4	5	6	7	8	9	10
1		4	3	6	10		5			
2		9		2	6		1			
3		1		3	7		2			
4		7		9	4		8			
5		3		5	9		4			
6	1	4	4	3	7		5			
7		5		7	2		6			
8							3			7
9								2		
10		5		7	2		6		1	

Algorithme

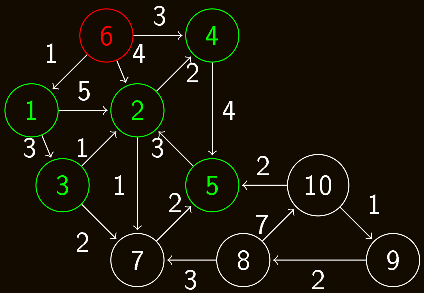
En langage

```
sp = matrice(n,n)

pour i de 1 à n
  pour j de 1 à n
   sp(i,j)<-delta(i,j)

pour k de 1 à n
 pour i de 1 à n
  pour j de 1 à n
   sp(i,j)<-
    min
     sp(i,j)
     sp(i,k)+sp(k,j)
```

Exemple



k=6	1	2	3	4	5	6	7	8	9	10
1		4	3	6	10		5			
2		9		2	6		1			
3		1		3	7		2			
4		7		9	4		8			
5		3		5	9		4			
6	1	4	4	3	7		5			
7		5		7	2		6			
8							3			7
9								2		
10		5		7	2		6		1	

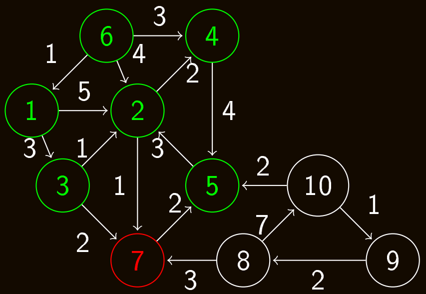
En langage

```
sp = matrice(n,n)

pour i de 1 à n
  pour j de 1 à n
   sp(i,j)<-delta(i,j)

pour k de 1 à n
 pour i de 1 à n
  pour j de 1 à n
   sp(i,j)<-
    min
     sp(i,j)
     sp(i,k)+sp(k,j)
```

Exemple



k=7	1	2	3	4	5	6	7	8	9	10
1		4	3	6	7(10)		5			
2		6(9)		2	3(6)		1			
3		1		3	4(7)		2			
4		7		9	4		8			
5		3		5	6(9)		4			
6	1	4	4	3	7		5			
7		5		7	2		6			
8		8		10	5		3			7
9								2		
10		5		7	2		6		1	

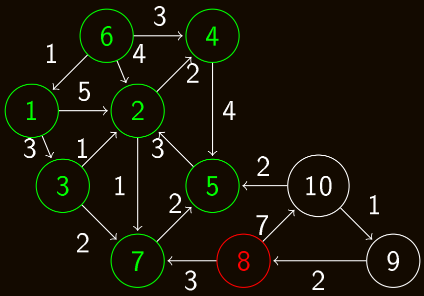
En langage

```
sp = matrice(n,n)

pour i de 1 à n
  pour j de 1 à n
   sp(i,j)<-delta(i,j)

pour k de 1 à n
 pour i de 1 à n
  pour j de 1 à n
   sp(i,j)<-
    min
     sp(i,j)
     sp(i,k)+sp(k,j)
```

Exemple



k=8	1	2	3	4	5	6	7	8	9	10
1		4	3	6	7		5			
2		6		2	3		1			
3		1		3	4		2			
4		7		9	4		8			
5		3		5	6		4			
6	1	4	4	3	7		5			
7		5		7	2		6			
8		8		10	5		3			7
9		10		12	7		5	2		9
10		5		7	2		6	3	1	

Algorithme

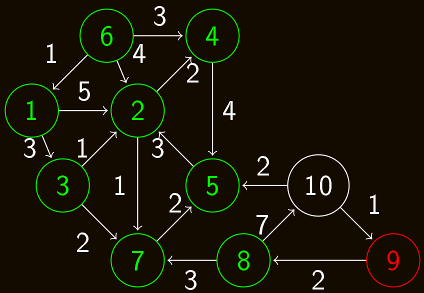
En langage

```
sp = matrice(n,n)

pour i de 1 à n
  pour j de 1 à n
   sp(i,j)<-delta(i,j)

pour k de 1 à n
 pour i de 1 à n
  pour j de 1 à n
   sp(i,j)<-
    min
     sp(i,j)
     sp(i,k)+sp(k,j)
```

Exemple



k=9	1	2	3	4	5	6	7	8	9	10
1		4	3	6	7		5			
2		6		2	3		1			
3		1		3	4		2			
4		7		9	4		8			
5		3		5	6		4			
6	1	4	4	3	7		5			
7		5		7	2		6			
8		8		10	5		3			7
9		10		12	7		5	2		9
10		5		7	2		6	3	1	10

Algorithme

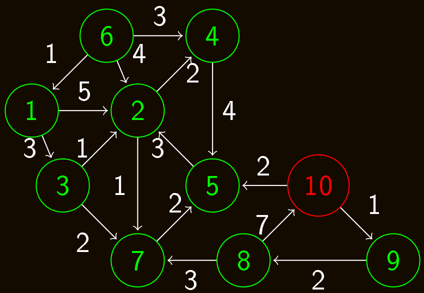
En langage

```
sp = matrice(n,n)

pour i de 1 à n
  pour j de 1 à n
   sp(i,j)<-delta(i,j)

pour k de 1 à n
 pour i de 1 à n
  pour j de 1 à n
   sp(i,j)<-
    min
      sp(i,j)
      sp(i,k)+sp(k,j)
```

Exemple



k=10	1	2	3	4	5	6	7	8	9	10
1		4	3	6	7		5			
2		6		2	3		1			
3		1		3	4		2			
4		7		9	4		8			
5		3		5	6		4			
6	1	4	4	3	7		5			
7		5		7	2		6			
8		8		10	5		3	10	8	7
9		10		12	7		5	2	10	9
10		5		7	2		6	3	1	10

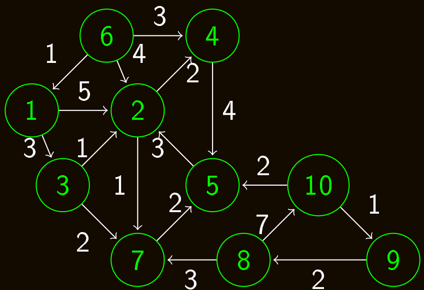
En langage

```
sp = matrice(n,n)

pour i de 1 à n
  pour j de 1 à n
   sp(i,j)<-delta(i,j)

pour k de 1 à n
 pour i de 1 à n
  pour j de 1 à n
   sp(i,j)<-
    min
     sp(i,j)
     sp(i,k)+sp(k,j)
```

Exemple



fini!	1	2	3	4	5	6	7	8	9	10
1		4	3	6	7		5			
2		6		2	3		1			
3		1		3	4		2			
4		7		9	4		8			
5		3		5	6		4			
6	1	4	4	3	7		5			
7		5		7	2		6			
8		8		10	5		3	10	8	7
9		10		12	7		5	2	10	9
10		5		7	2		6	3	1	10

Remarques

Complexité

- Trois boucles de taille n -> $O(n^3)$,
- Optimisations existantes pour réduire le coefficient 3 à des valeurs plus faibles (2 et quelques),
- Faisable sur un graphe de taille raisonable:

Taille du graphe	Temps requis
100	$\approx ms$
1 000	$\approx s$
1 000 000	$\approx$ 31 ans

Objectifs

- Trouver les chemins les plus courts entre un noeud et l'ensemble des noeuds accessibles à partir de celui-ci,
- Cette fois ci, le graphe ne doit comporter que des valuations positives.

Idée

- Construire l'ensemble des chemins des plus courts au plus longs. . .
- Donc sorte de parcours en largeur prenant en compte les valuations des arrêtes.
- Autre vision : innondation du graphe à partir d'un point source.

Idée plus en détail

Vision

- L'ensemble des sommets est divisé en deux:
  - Les sommets déjà considérés,
  - Ceux qui n'ont pas encore été visités.
- Heuristique sur le choix du prochain sommet à visiter:
  - Dijkstra considère le sommet le plus proche.

Pourquoi ça marche ?

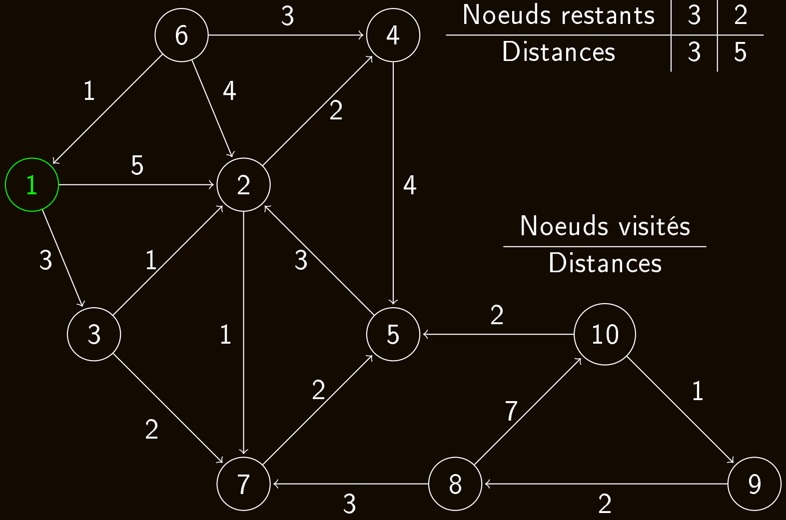
- L'ensemble des noeuds est visité par chemin de taille croissante,
- Tous les chemins sont considérés,
- Le plus court chemin est donc rencontré pour tout noeud accessible.

Algorithme

En langage

```
distances = matrice(n)
pour tout successeur s du noeud source
  distances(s)<-delta(noeud source,s)
pour tous les autres s
  distances(s)<-infini
tas t = tas(n)
tant que t non vide
  prendre n noeud de t de distance minimale
    t = t privé de n
    si distance(n)=infini alors stop
    pour tous les successeurs s de n
      distance(s)<-
        min
          distance(s)
          distance(n)+delta(n,s)
```

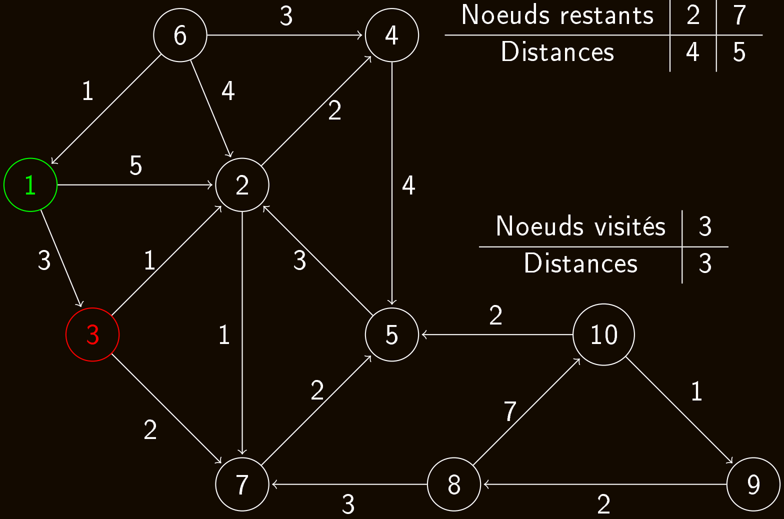
Sur le même graphe



Noeuds restants	3	2
Distances	3	5

Noeuds visités
Distances

Sur le même graphe

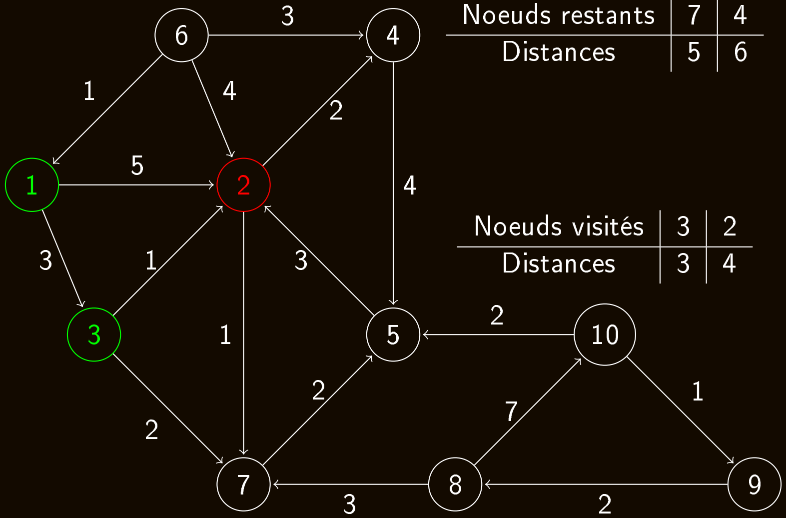


Noeuds restants	2	7
Distances	4	5

Noeuds visités	3
Distances	3

Exemple

Sur le même graphe

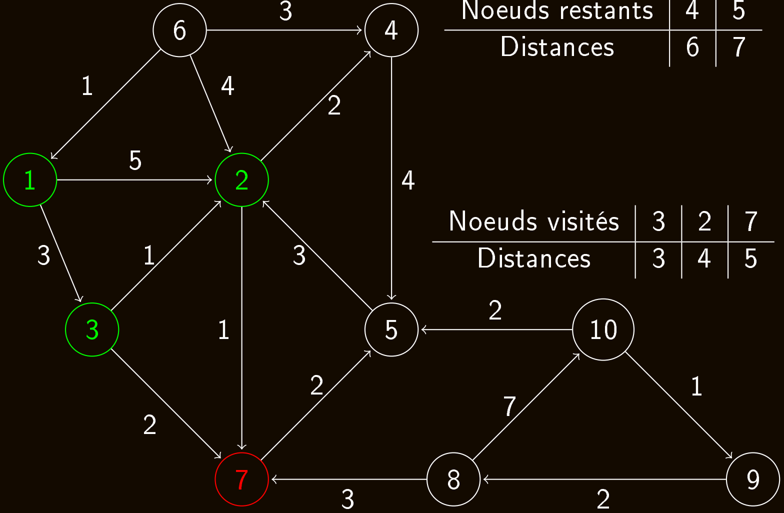


Noeuds restants	7	4
Distances	5	6

Noeuds visités	3	2
Distances	3	4

Exemple

Sur le même graphe

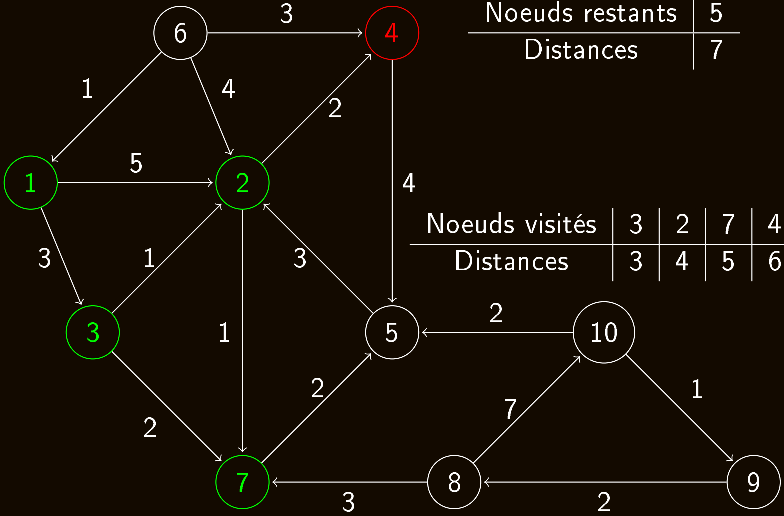


Noeuds restants	4	5
Distances	6	7

Noeuds visités	3	2	7
Distances	3	4	5

Exemple

Sur le même graphe



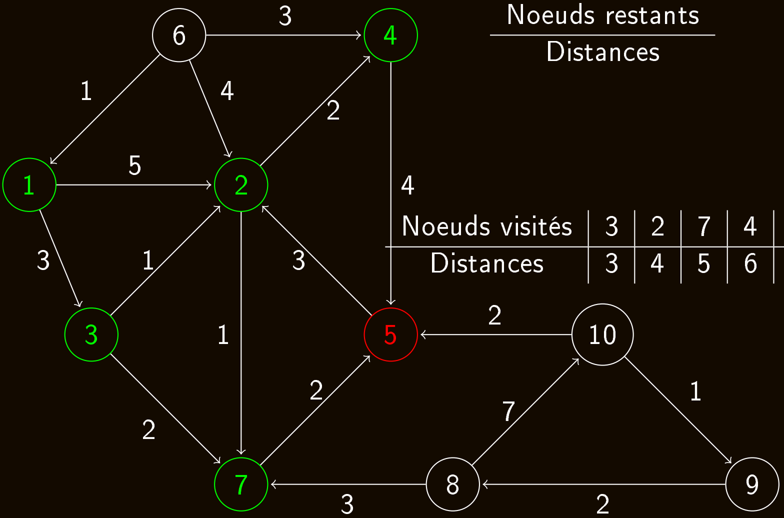
Noeuds restants	5
Distances	7

Noeuds visités	3	2	7	4
Distances	3	4	5	6

Exemple

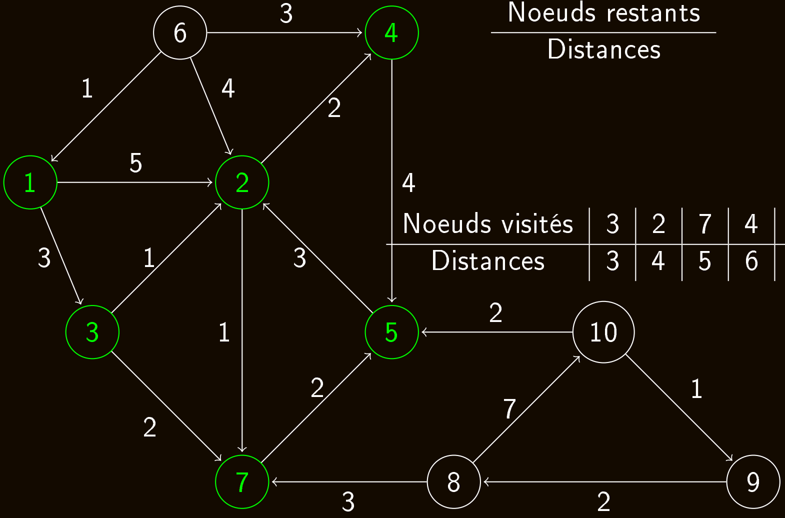
Sur le même graphe

Noeuds restants
Distances



Noeuds visités	3	2	7	4	5
Distances	3	4	5	6	7

Sur le même graphe



Noeuds restants
Distances

Noeuds visités	3	2	7	4	5
Distances	3	4	5	6	7

Remarques

Complexité

- Avec une structure de donnée bien choisie: $O(n^2)$,

Taille du graphe	Temps requis
100	$\approx \mu s$
1 000	$\approx ms$
1 000 000	moins de 20 minutes

- Si le graphe possède peu d'arrêtes (m), $O(m + nln(n))$,

Autres remarques

- Si l'on cherche seulement le chemin le plus court reliant deux noeuds, on peut stopper l'algorithme prématurément lorsqu'il a visité le noeud cible.
- Lors de la mise à jour du tableau des distances, on peut mettre à jour une table de routage permettant de retenir le chemin à prendre.