Rapport: SAE Graphes

LATH Victor
LIM HOUN TCHEN Aimé

Représentation graphique

Question 11:
On a ajouté comme tests :

- Calcul du plus court chemin par point fixe

Question 13:

```
fonction pointFixe(G InOut:Graphe, depart:Noeud)
  debut
    pour chaque sommet v de G faire
      v.distance <- infini
      v.précédent <- indéfini
    fpour
    depart.distance <- 0

    cpt <- 0
    vtmp <- null
    tant que non egal(v, vtmp) et cpt<nbSommets faire
      vtmp <- v.toString()
      pour chaque sommet v de G faire
        pour chaque arc (u, v) de G faire
          tmp <- u.distance + poids(u, v)
          si tmp < v.distance alors
            v.distance <- tmp
            v.précédent <- u
          fsi
        fpour
        cpt <- cpt+1
      fpour
    ftant
  fin
```

LEXIQUE:
G : Graphe, graphe orienté avec poids positive des arcs
depart : Noeud, un sommet de G
cpt : entier, nombre d'itérations
nbSommets : entier, nombre de sommets du graphe G

Question 15:
A -> V:0.0 p:null
B -> V:12.0 p:A
C -> V:76.0 p:D
D -> V:66.0 p:E
E -> V:23.0 p:B

[A, B, E, D, C]

Validation et experimentation

Question 21:

08 p:null

Dijkstra:

Iteration 0 :

A -> V:0.0 p:null

B -> V:1.7976931348623157E308 p:null

C -> V:1.7976931348623157E308 p:null

D -> V:1.7976931348623157E308 p:null

E -> V:1.7976931348623157E308 p:null

F -> V:1.7976931348623157E308 p:null

G -> V:1.7976931348623157E308 p:null

Iteration 1 :

A -> V:0.0 p:null

B -> V:20.0 p:A

C -> V:1.7976931348623157E308 p:null

D -> V:3.0 p:A

E -> V:1.7976931348623157E308 p:null

F -> V:1.7976931348623157E308 p:null

G -> V:1.7976931348623157E3

Iteration 2 :

A -> V:0.0 p:null

B -> V:20.0 p:A

C -> V:7.0 p:D

D -> V:3.0 p:A

E -> V:1.7976931348623157E308 p:null

F -> V:1.7976931348623157E308 p:null

G -> V:1.7976931348623157E308 p:null

Iteration 3 :

A -> V:0.0 p:null

B -> V:9.0 p:C

C -> V:7.0 p:D

D -> V:3.0 p:A

E -> V:1.7976931348623157E308 p:null

F -> V:1.7976931348623157E308 p:null

G -> V:1.7976931348623157E308 p:null

Iteration 4 :

A -> V:0.0 p:null

B -> V:9.0 p:C

C -> V:7.0 p:D

D -> V:3.0 p:A

E -> V:1.7976931348623157E308 p:null

F -> V:1.7976931348623157E308 p:null
G -> V:19.0 p:B

Iteration 5 :
A -> V:0.0 p:null
B -> V:9.0 p:C
C -> V:7.0 p:D
D -> V:3.0 p:A
E -> V:1.7976931348623157E308 p:null
F -> V:24.0 p:G
G -> V:19.0 p:B

Iteration 6 :
A -> V:0.0 p:null
B -> V:9.0 p:C
C -> V:7.0 p:D
D -> V:3.0 p:A
E -> V:27.0 p:F
F -> V:24.0 p:G
G -> V:19.0 p:B

Bellman Ford:
Iteration 0 :
A -> V:0.0 p:null
B -> V:1.7976931348623157E308 p:null
C -> V:1.7976931348623157E308 p:null
D -> V:1.7976931348623157E308 p:null
E -> V:1.7976931348623157E308 p:null
F -> V:1.7976931348623157E308 p:null
G -> V:1.7976931348623157E308 p:null

Iteration 1 :
A -> V:0.0 p:null
B -> V:9.0 p:C
C -> V:7.0 p:D
D -> V:3.0 p:A
E -> V:38.0 p:F
F -> V:35.0 p:G
G -> V:30.0 p:B

Iteration 2 :
A -> V:0.0 p:null
B -> V:9.0 p:C
C -> V:7.0 p:D
D -> V:3.0 p:A
E -> V:27.0 p:F
F -> V:24.0 p:G
G -> V:19.0 p:B

Iteration 3 :
A -> V:0.0 p:null
B -> V:9.0 p:C
C -> V:7.0 p:D
D -> V:3.0 p:A
E -> V:27.0 p:F
F -> V:24.0 p:G
G -> V:19.0 p:B

Point fixe:
Dijkstra : 6 itérations
Bellman Ford : 3 itérations

La méthode Dijkstra cherche le chemin minimum pour chaque sommet alors que la méthode Bellman-Ford regarde le coût minimum en additionnant les antécédents des sommets.

Question 22:

Dans le cas de ce graphe,l'algorithme de Bellman-Ford est plus rapide car il ne prend que 3 itérations.

Question 23:

L'algorithme de Bellman-Ford semble plus rapide pour des petits graphes

Question 25:

```
digraph {
n1 -> n2 [label = 48]
n2 -> n3 [label = 40]
n2 -> n2 [label = 18]
n3 -> n4 [label = 61]
n3 -> n6 [label = 75]
n4 -> n5 [label = 19]
n6 -> n7 [label = 9]
n5 -> n6 [label = 5]
n7 -> n8 [label = 19]
n8 -> n9 [label = 61]
n9 -> n10 [label = 7]
}

digraph {
n1 -> n2 [label = 50]
n1 -> n1 [label = 39]
n2 -> n3 [label = 72]
n3 -> n4 [label = 84]
n3 -> n6 [label = 76]
```

```
n4 -> n5 [label = 46]
n6 -> n7 [label = 86]
n5 -> n6 [label = 33]
n7 -> n8 [label = 87]
n8 -> n9 [label = 11]
n9 -> n10 [label = 44]
}

digraph {
n1 -> n2 [label = 88]
n1 -> n8 [label = 84]
n2 -> n3 [label = 4]
n2 -> n4 [label = 79]
n8 -> n9 [label = 48]
n8 -> n4 [label = 9]
n3 -> n4 [label = 87]
n3 -> n6 [label = 6]
n4 -> n5 [label = 35]
n4 -> n1 [label = 84]
n6 -> n7 [label = 72]
n6 -> n9 [label = 79]
n5 -> n6 [label = 95]
n5 -> n2 [label = 13]
n7 -> n8 [label = 14]
n7 -> n3 [label = 5]
n9 -> n10 [label = 55]
}
```

Question 26:

Nb itérations BellMan-Ford	Nb itérations Dijkstra	Noeuds	Arcs	Temps BellMan-Ford (nanosecondes)	Temps Dijkstra (nanosecondes)
2	10	10	16	16899700	509100
2	10	10	13	17237100	501900
2	10	10	14	16832500	447400
4	100	100	147	36512900	3999500
4	100	100	146	39001200	6161700
6	100	100	157	39724600	3414200
8	1000	1000	1493	477588800	120856900

9	1000	1000	1497	509431300	114067200
7	1000	1000	1527	488236200	118297000

Question 27:
L’algorithme de Dijkstra est 33 fois plus rapide que l’algorithme de BellMan-Ford avec 10 noeuds, 10 fois plus rapide avec 100 noeuds et 4 fois plus rapide avec 1000 noeuds

Question 28:
On peut en conclure que ...

Extension : Intelligence Artificielle et labyrinthe

Question 30:
Départ : (0, 0), Arrivée : (4, 0)
Chemin le plus court :

Question 31:

Bilan:
Cette SAÉ nous à permis de mettre en pratique les algorithmes vue en cours donc cela nous a permis de mieux appréhender la notion de recherche de chemin le plus court.
Nous avons appris que l'algorithme de Dijkstra est bien plus rapide que celui du point fixe.

Tests:
-