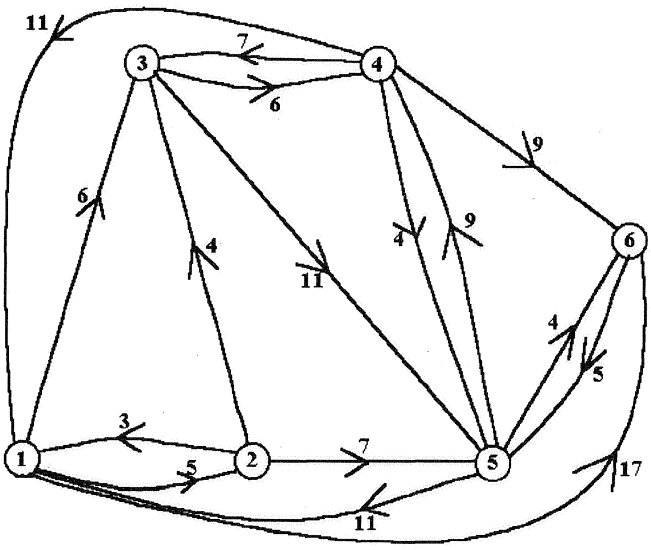
Calculatrices et ordinateurs

Jean M. Turgeon

TI-82, TI-83 et les plus courts chemins

Vous cherchez le plus court chemin pour vous rendre au travail ? Pour le trouver, je vous recommande l'algorithme de Dijkstra. Et si le nombre de chemins possibles pour vous rendre au travail est très grand, une calculatrice TI-82 ou TI-83 peut faciliter beaucoup votre recherche.

Supposons que vous habitez au sommet **1** du graphe ci-dessous, et que vous travaillez au sommet **6**.



Dans un graphe relativement simple, comme celui-ci, il suffit d'examiner le graphe et de calculer un peu pour trouver le chemin le plus court. Mais comment être sûr de son résultat ?

La méthode usuelle de résoudre ce problème consiste, curieusement, à déterminer le chemin le plus court non seulement du sommet **1** au sommet **6**, mais aussi du sommet **1** à chacun des autres sommets du graphe.

On procède par étapes. Soit U l'ensemble des sommets dont on connaît déjà le chemin le plus court depuis le sommet **1**. Au départ, U ne contient que le sommet **1** lui-même. À la fin des calculs, U contient tous les sommets du graphe. Soit V l'ensemble des sommets qui ne sont pas encore dans U. Soit W l'ensemble des sommets de V qui sont liés par une arête à un sommet de U.

i	U	Arêtes	Toutes distances	Sommet choisi	Distance minimale
1	1	1,2 1,3 1,6	0 + 5 = 5 0 + 6 = 6 0 + 17 = 17	2	5
2	1, 2	1,3 1,6 2,3 2,5	0 + 6 = 6 0 + 17 = 17 5 + 4 = 9 5 + 7 = 12	3	6
3	1, 2, 3	1,6 2,5 3,4 3,5	0 + 17 = 17 5 + 7 = 12 6 + 6 = 12 6 + 11 = 17	5	12
4	1, 2, 3, 4	1,6 3,4 5,4 5,6	0 + 17 = 17 6 + 6 = 12 12 + 9 = 21 12 + 4 = 16	4	12
5	1, 2, 3, 4, 5,	1,6 4,6 5,6	0 + 17 = 17 12 + 9 = 21 12 + 4 = 16	6	16

Le tableau indique l'ordre des calculs. Une étape consiste à choisir un sommet de W à transférer dans U. Il est commode de numéroter les étapes dans une première colonne. À l'étape i, l'ensemble U contient i sommets. La deuxième colonne contient la liste des sommets déjà transférés dans U. La troisième colonne contient la liste des arêtes qui relient un sommet de U à un sommet de W. Dans la quatrième colonne, on calcule les longueurs de tous les chemins qui partent de A, vont d'un sommet de U à un autre et se terminent par une arête qui joint un sommet de U à un sommet de W. Ce calcul est simplifié par le fait que nous connaissons déjà les distances minimales de tous les sommets de U au sommet **1**. On choisit ensuite la somme minimale et on l'inscrit dans la sixième colonne, puis on inscrit dans la cinquième colonne le nom du sommet qui réalise cette distance minimale.

Dans notre exemple, le chemin le plus court est de longueur 16. Par quels sommets passe-t-il ? D'après l'étape 5, l'avant-dernier sommet à traverser est **5**. D'après l'étape 3, on doit arriver au sommet **5** du sommet **2**. De proche en proche, on trouve ainsi que le chemin le plus court est

1 – 2 – 5 – 6.

C'est ce qu'on appelle l'algorithme de Dijkstra.

Mon but est de vous présenter un programme qui permet à une calculatrice TI-82 ou TI-83 d'appliquer l'algorithme de Dijkstra à un graphe orienté quelconque. Comment introduire le graphe dans la calculatrice ? C'est simple, parce que les seules données utilisées dans les calculs sont le nombre des sommets et, pour chacune des arêtes, son sommet de départ, son sommet d'arrivée et sa longueur. Dans notre exemple, le graphe comporte six sommets et sa *matrice des arêtes* est la suivante.

Départ	Arrivée	Longueur
1	2	5
1	3	6
1	6	17
2	1	3
2	3	4
2	5	7
3	4	6
3	5	11
4	5	4
4	6	9
4	1	11
4	3	7
5	4	9
5	6	4
5	1	11
6	5	5

On sauvegarde cette matrice dans la mémoire [B] de la calculatrice. Si un graphe n'est pas orienté, on inscrit chaque arête deux fois dans la matrice des arêtes, une fois pour chaque orientation.

Une première version du programme trouve le chemin le plus court d'un sommet donné à un autre sommet donné.

```
PROGRAM:DIJK01
ClrHome
Disp "SOIT [B] LA MA-"
Disp "TRICE DES ARETES"
Disp " "
Disp "NOMBRE DES"
Input "SOMMETS ? ", M
{M,2}→dim([A])
dim([B])→L1:L1(1)→A
Input "SOMMET-DEPART ? ",S
Disp "SOMMET-"
Input "DESTINATION ? ",T
ClrHome
Disp "UN MOMENT SVP"
Fill (9999,[A])
S→[A](S,1)
0→[A](S,2)
1→F
While F=1
0→F
For (J,1,A)
[A](J,1)+[B](J,3)→D
[B](J,2)→E
If D<[A](E,2)
Then
1→F
D→[A](E,2)
[B](J,1)→[A](E,1)
End
End
End
1→L
1→dim(L1)
T→L1(1):T→H
Repeat H=S
L+1→L
L→dim(L1)
[A](H,1)→H
H→L1(L)
End
L→dim(L2)
For (I,1,L)
L1(I)→L2(L-I+1)
End
ClrHome
Disp "CHEMIN ", L2
Disp "DISTANCE ",[A](T,2)
```

Avec **1** comme sommet de départ et **6** comme sommet d'arrivée, le programme, après quelques secondes, donne le chemin {**1**, **2**, **5**, **6**} et le coût 16.

La seconde version donne tous les chemins les plus courts. Après chaque résultat, il suffit d'appuyer sur ENTER pour avoir le résultat suivant.

```
PROGRAM:DIJK02
ClrHome
Disp "SOIT [B] LA MA-"
Disp "TRICE DES ARETES"
Disp " "
Disp "NOMBRE DES"
Input "SOMMETS ? ", M
{M,2}→dim([A])
dim([B])→L1:L1(1)→A
For (K,1,M)
ClrHome
Disp "UN MOMENT SVP"
Fill (9999,[A])
K→[A](K,1)
0→[A](K,2)
1→F
While F=1
0→F
For (J,1,A)
[A]([B](J,1),2)+[B](J,3)→D
[B](J,2)→E
If D<[A](E,2)
Then
1→F
D→[A](E,2)
[B](J,1)→[A](E,1)
End
End
End
For (J,1,M)
1→L
1→dim(L1)
J→L1(1):J→H
Repeat H=K
L+1→L
L→dim(L1)
[A](H,1)→H
H→L1(L)
End
L→dim(L2)
For (I,1,L)
L1(I)→L2(L-I+1)
End
ClrHome
Disp "CHEMIN ", L2
Disp "DISTANCE ",[A](J,2)
Pause
End
End
```

Dans notre exemple, le programme DIJK02 produit des résultats qui sont résumés dans le tableau ci-dessous.

	1	2	3	4	5	6
1	{1,1} 0	{1,2} 5	{1,3} 6	{1,3,4} 12	{1,2,5} 12	{1,2,5,6} 16
2	{2,1} 3	{2,2} 0	{2,3} 4	{2,3,4} 10	{2,5} 7	{2,5,6} 11
3	{3,4,1} 17	{3,4,1,2} 22	{3,3} 0	{3,4} 6	{3,4,5} 10	{3,4,5,6} 14
4	{4,1} 11	{4,1,2} 16	{4,3} 7	{4,4} 0	{4,5} 4	{4,5,6} 8
5	{5,1} 11	{5,1,2} 16	{5,4,3} 16	{5,4} 9	{5,5} 0	{5,6} 4
6	{6,5,1} 16	{6,5,1,2} 21	{6,5,4,3} 21	{6,5,4} 14	{6,5} 5	{6,6} 0

La grandeur des graphes traitables dépend seulement de la capacité de mémoire de votre calculatrice. Si la vôtre est une TI-83 Plus, dont la mémoire est très grande, vous ne rencontrerez des difficultés que si votre appareil est vraiment surchargé ! ■