ministère éducation nationale

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

éduscol

Ressources pour le lycée général et technologique

Ressources pour le cycle terminal général et technologique

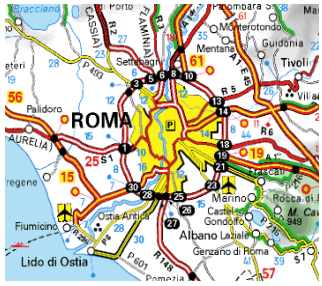
Informatique et Sciences du Numérique

À la recherche du plus court chemin

Juin 2012

Présentation / À la recherche du plus court chemin

1 / Thème abordé



1.1 Problématique, situation d'accroche

Les applications utilisant les signaux des satellites GPS sont en plein essor. Elles ne se limitent pas aux véhicules automobiles, puisqu'elles sont par exemple intégrées désormais dans certains appareils photographiques. Cependant, dans le cas des véhicules, s'y ajoute le calcul de l'itinéraire optimal entre deux lieux géographiques -optimal au regard de critères tels que distance, temps ou coût total.

Ce calcul fait appel à la théorie des graphes et utilise différents algorithmes dont celui de Dijkstra, qui est un algorithme du type parcours en largeur ou BFS (Breadth First Search). À la différence d'un algorithme DFS (Depth First Search) où l'on explore un sommet adjacent à celui de départ, puis un autre adjacent au précédent, et ainsi de suite jusqu'à se retrouver bloqué et revenir en arrière, on examine ici dès le départ tous les sommets adjacents au premier. L'algorithme de Dijkstra est actuellement enseigné en spécialité maths en terminale ES.

Le GPS est donc un point d'entrée très intéressant pour aborder un exemple d'algorithme de recherche de plus court chemin dans un graphe, que l'on peut présenter ainsi :

Comment un terminal de navigation GPS(1) calcule-t-il l'itinéraire entre le lieu de départ et le lieu d'arrivée ? Comment prend-il en compte des critères tels que distance, temps, coût (carburant et péages d'autoroutes) ?
Et comment fait-il pour proposer en quelques secondes un nouvel itinéraire alors que l'on vient d'oublier de tourner à droite comme il le suggérait ?

Même si elle n'est pas au cœur de la problématique évoquée ici, la géolocalisation pourra être abordée en utilisant les connaissances de physique de terminale S sur les ondes (relation entre distance, temps, et célérité).

1.2 Frontières de l'étude et prolongements possibles

L'étude proposée porte donc sur la recherche du plus court chemin entre le lieu de départ et le lieu d'arrivée. L'algorithme étudié ici est celui de Dijkstra, plus court chemin pouvant s'entendre en terme de distance, de temps, ou de coût. Il s'agit dans un premier temps que les élèves s'approprient cet algorithme à travers différentes activités, puis l'implémentent au moins en partie, en langage Python (ou autre).

Le principe et les applications de la géolocalisation sont un prolongement possible, de même que l'étude des principes physiques du GPS (durée de propagation des ondes en tenant compte de la correction relativiste).

2 / Objectifs pédagogiques

2.1 Disciplines impliquées

Mathématiques : théorie des graphes (qui n'est pas au programme de la série S)

Physique : propagation des ondes et effets relativistes

(1) Le terminal de navigation inclut un récepteur GPS, une base cartographique (SIG), un logiciel de recherche de plus courts chemins, et un logiciel de navigation.

2.2 Éléments du programme

Contenus

Algorithmique; algorithmes plus avancés : recherche d'un plus court chemin par un parcours en largeur (BFS).

Compétences et capacités

- Comprendre et expliquer (oralement ou par écrit) ce que fait un algorithme (variante : lire et comprendre un algorithme ou un programme conçu par d'autres).
- Concevoir et programmer un algorithme (au moins en partie).

3 / Modalités de mise en œuvre

3.1 Durée prévue pour la partie se déroulant en classe

Deux séances, sachant qu'un travail préparatoire peut être donné.

3.2 Type de l'animation

Activités en classe entière ; les travaux en mini-projets peuvent être présentés à la classe.

3.3 Prérequis

Aucun sauf si on se lance dans la programmation, qui nécessite boucles et tableaux.

3.4 Projet

On peut envisager un mini-projet portant sur l'implémentation d'une partie de l'algorithme ou de sa totalité, ou sur la compréhension d'un programme fourni aux élèves. Un langage tel que Python permet de traduire la représentation du graphe sous forme de matrice d'adjacence à l'aide d'une structure unique de type liste. L'algorithme fait quant à lui appel à une structure de boucle « tant que » dont l'implémentation peut être introduite à partir d'un exemple simple. Une implémentation menée dans sa totalité risque cependant d'être chronophage, mais elle peut être proposée.

3.5 Recherches documentaires

Recherche sur le GPS : historique, caractéristiques orbitales des satellites, systèmes concurrents…

3.6 Production des élèves

Exercices de mise en œuvre de l'algorithme de Dijkstra à remettre en fin de séance ou lors de la séance suivante. Implémentation complète ou partielle sur la base d'un exemple simple.

4 / Outils

Tableur, notamment pour détailler les étapes de l'algorithme.

Logiciel GRIN (GRaph INterface) : http://www.apprendre-en-ligne.net/graphes/logiciel/index.html

Ce logiciel permet notamment de tracer un graphe, d'en définir la pondération, et de trouver le plus court chemin entre 2 sommets de ce graphe. Il peut être utilisé par les élèves pour vérifier leurs résultats.

Langage Python par exemple si l'on veut faire implémenter l'algorithme : http://www.python.org/download/

Éventuellement, logiciel Graphviz pour dessiner des graphes (afin de préparer des activités pour les élèves) :

http://www.graphviz.org/Download.php

Une approche similaire est possible dans l'environnement Java's Cool :

http://javascool.gforge.inria.fr/index.php?page=proglets&action=show&id=gogleMaps

5 / Auteur

François Passebon, professeur de Sciences Physiques, académie de Nantes

À la recherche du plus court chemin

1 / Activités autour de l'algorithme de Dijkstra

Le vocabulaire de base lié aux graphes (sommets, arêtes, graphe pondéré, …) est introduit au fur et à mesure des activités proposées aux élèves. Hormis le premier exemple de mise en œuvre de l'algorithme, et encore (voir annexe), les élèves travaillent en autonomie.

Graphe orienté : les sommets sont reliés par des arcs et non des arêtes.
Graphe valué : un nombre réel est associé à chaque arc ou arête.
Graphe pondéré : ce nombre est positif; on l'appelle le poids (distance, durée, coût, ...).

Vocabulaire des graphes	*Vocabulaire routier*
Graphe (pondéré)	*Réseau routier*
Sommet ou nœud	*Ville, ou intersection de routes*
Arête	*Route*
Arc	*Route à sens unique*
Poids	*Distance, durée, ou coût*

La représentation d'un graphe sous forme d'une matrice d'adjacence ou d'une liste d'adjacence peut être abordée lors de ces activités : on peut donner la structure du graphe sous cette forme. Voir détails en annexe, notamment l'intérêt d'une représentation par rapport à l'autre.

Pour aborder l'algorithme de Dijkstra proprement dit, il peut être intéressant d'utiliser l'article de 1959 dans lequel il est présenté : « *A note on two problems in connexion with graphs* », et qui pose en fait comme son intitulé l'indique deux problèmes.

Problem 1. *Construct the tree of minimum total length between n nodes*[1].

Problem 2. *Find the path of minimum length between two given nodes P and Q*[2].

Le début de réponse que donne Dijkstra au problème 2 est un bon point d'entrée en matière :

"*We use the fact that, if R is a node on the minimal path from P to Q, knowledge of the latter implies the knowledge of the minimal path from P to R. In the solution presented, the minimal paths from P to the other nodes are constructed in order of increasing length until Q is reached.*"[3]

On peut préciser à ce sujet que la solution exposée dans l'article suggère de répartir les branches en trois ensembles à chaque étape de l'algorithme, et de faire de même pour les nœuds, ce qui peut être et sera simplifié.

Une démarche progressive, décrite ci-dessous, peut alors être adoptée (voir annexe pour plus de détails). L'ENT est bien sûr mis à profit pour distribuer les différentes activités et les récupérer une fois terminées.

1 Construire un arbre de longueur minimale passant par n sommets.
2 Trouver le plus court chemin entre deux sommets donnés P et Q.
3 Nous utilisons le fait que, si R est un sommet sur le chemin minimal allant de P à Q, la connaissance de celui-ci implique la connaissance du chemin minimal allant de P à R. Dans la solution proposée, les chemins minimaux de P vers les autres nœuds sont construits par longueurs croissantes jusqu'à ce que Q soit atteint.

Première étape - *Les élèves réfléchissent sur un premier exemple de mise en œuvre de l'algorithme. Même si l'on parle de villes et de routes, il ne s'agit pas encore d'un exemple réel afin de pouvoir faire mentalement les calculs de distances et de se concentrer sur l'essentiel. En utilisant un simple tableur, on facilite ce travail, notamment la rectification d'éventuelles erreurs.*
Deuxième étape - *A l'issue du premier exemple, destiné à s'approprier l'algorithme, il paraît tout indiqué de donner aux élèves un cas réel de détermination du plus court chemin entre deux villes. Plusieurs groupes travaillent sur le même graphe (recherche de l'itinéraire le moins long, le plus rapide, ou le moins cher), puis ils présentent leurs résultats et les confrontent. La rotondité de la Terre pouvant poser problème, on travaille à une échelle suffisamment petite pour s'affranchir de cela.*
Troisième étape - *Préalablement à une implémentation de l'algorithme, au moins en partie, les élèves réfléchissent à une représentation des données du graphe : on peut donner quelques exercices sur les listes en langage Python (voir annexe).*
Quatrième étape - *Il s'agit d'un travail de compréhension d'un programme implémentant l'algorithme de Dijkstra (programme fourni), ou de l'implémentation proprement dite, au moins partielle. Il convient de personnaliser le niveau de difficulté. On peut demander par exemple aux élèves de compléter certaines lignes de programme, de rectifier une erreur (il est important de savoir déboguer lorsqu'on développe).*

Quelles sont les limites de cet algorithme ? Il fonctionne également pour les graphes orientés (il est donc capable de prendre en compte les voies à sens unique), mais ne marche pas si les arêtes sont pondérées négativement (ce qui n'est jamais le cas pour ce qui nous concerne).

Un logiciel tel que GRIN peut éventuellement être utilisé pour que les élèves travaillent en autonomie et corrigent eux-mêmes leurs activités : saisie des sommets, des arêtes et de leur poids respectifs, recherche automatisée du plus court chemin.

2 / Liens avec les autres disciplines

SI / réseaux : on peut préciser que les algorithmes de recherche de plus court chemin sont mis en œuvre dans les réseaux de télécommunications, dans l'Internet en particulier (protocole OSPF -Open Shortest Path First par exemple).

Physique (prolongements) :

- Détermination de la position du récepteur GPS par triangulation, où l'on peut faire l'analogie avec la détermination de l'épicentre d'un séisme à partir des enregistrements de 3 stations synchronisées entre elles. En fait, 4 satellites au moins sont utilisés dans le cas du GPS afin d'éliminer les situations ambiguës ainsi que d'augmenter la précision.
- Corrections relativistes d'horloges GPS (l'une en rapport avec la théorie de la relativité restreinte, l'autre avec celle de la relativité générale); la première peut éventuellement être abordée dans le cadre du programme de physique de terminale S 2012. Sur une journée, la correction globale correspond à 38 µs (46 µs pour la correction liée à la relativité générale et -8 µs pour celle liée à la relativité restreinte). 38 µs-lumière représentent 12 km environ, ce qui fait qu'une absence de correction serait incompatible avec une précision de l'ordre du mètre, voire moins.
- On peut aussi interroger les élèves quant à la raison pour laquelle la vitesse d'un avion est déterminée à l'aide de sondes Pitot, et non à l'aide d'un GPS : c'est la vitesse de l'avion par rapport à l'air qui détermine la portance. Sa position par rapport au sol peut bien-sûr être déterminée par GPS (le GPS est utilisé pour la navigation horizontale).
- Éventuellement, recherches sur les facteurs perturbants qui modifient la célérité des ondes (ionosphère, pression, et humidité) et les parades associées (SBAS).

Références du programme de physique de terminale S 2012 : les ondes dans la matière (ondes sismiques); caractéristiques des ondes (relation entre retard, distance, et célérité); temps, cinématique, et dynamique newtoniennes (établir l'expression de la vitesse d'un satellite et de sa période dans le cas d'un mouvement circulaire uniforme); temps et relativité restreinte (postulats d'Einstein, caractère relatif du temps, temps propre, dilatation des durées).

3 / Références

Sitographie

"A note on two problems in connexion with graphs", de E.W. Dijkstra, sur le site de la Technische Universität de Munich : http://www-m3.ma.tum.de/foswiki/pub/MN0506/WebHome/dijkstra.pdf

L'article suivant est une bonne introduction au problème ; il pose la question du temps de calcul :

http://interstices.info/jcms/c_15578/le-plus-court-chemin

Document d'accompagnement du programme de terminale ES, sur les graphes :

http://media.eduscol.education.fr/file/Programmes/16/8/graphes_109168.pdf

Page proposée par SwissEduc sur les algorithmes de routage :

http://www.swisseduc.ch/informatik/diskrete_mathematik/routing_gruppenarbeit/

avec notamment un exécutable à télécharger qui permet de tester l'algorithme sur un réseau configurable de 2 à 48 nœuds; il peut servir d'aide à une résolution manuelle de l'algorithme.

Activité « goglemaps » de Java's Cool :

http://javascool.gforge.inria.fr/index.php?page=proglets&action=show&id=gogleMaps

Une présentation claire de l'algorithme, illustrée d'animations flash :

http://yallouz.arie.free.fr/terminale_cours/graphes/dijkstra.php

Un applet java permettant de créer son propre graphe et de trouver le plus court chemin entre deux nœuds :

http://www.dgp.toronto.edu/people/JamesStewart/270/9798s/Laffra/DijkstraApplet.html

Sur un site académique (Bordeaux) :

http://mathematiques.ac-bordeaux.fr/pedalyc/seqdocped/graphes/paf_02-03/graphes_02-03_intro.htm

Sujet de bac ES 2009 (exercice 2) : trajet comportant un minimum de feux tricolores :

http://www2.ac-rennes.fr/crdp/doc/docadmin/sujets/examens/_0310160440_005.pdf

Sujet de bac ES 2004 (exercice 2) : minimisation du temps de parcours entre deux bâtiments :

http://www2.ac-rennes.fr/crdp/doc/docadmin/sujets/examens/bac04es-MASPE.PDF

Informations officielles sur le GPS :

http://www.gps.gov/french.php

Un cours très clair et accessible sur le GPS :

http://lpc2e.cnrs-orleans.fr/~ddwit/gps/

http://lpc2e.cnrs-orleans.fr/~ddwit/gps/cours-GPS.pdf

Une animation illustrant le fait que 3 satellites suffisent pour la géolocalisation (à condition d'avoir une référence temporelle suffisamment précise pour mesurer ces distances, ce qui est rarement le cas) :

http://www.sciences.univ-nantes.fr/sites/genevieve_tulloue/Meca/Planetes/3spheres.html

Bibliographie

ROUX Ch. *Initiation à la théorie des graphes*. Éditions Ellipses, 2009.

MOATTI A., *Les indispensables mathématiques et physiques pour tous*. Éd. Odile Jacob, 2006, p. 165-168.

SEMAY Cl., SILVESTRE-BRAC B. *Relativité restreinte.* Éditions Dunod, 2010, p. 86-90 et p. 269.

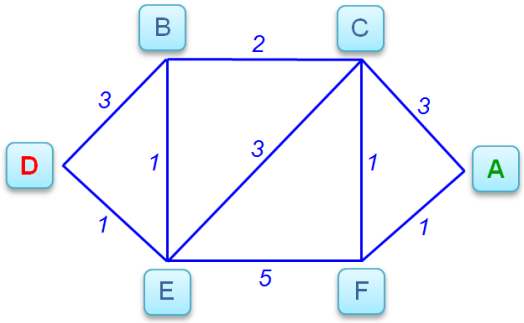
HOBSON M.P., EFSTATHIOU G.P., LASENBY A.N. *Relativité générale.* Éditions de Bœck, p. 151-153.

Annexe – exemples d'activités

1 / De la découverte de l'algorithme de Dijkstra à l'implémentation

1.1 Première étape : mise en œuvre de l'algorithme

Quel est le plus court chemin pour aller de D à A ?



Le coefficient en italique associé à une route représente la distance en km entre les deux villes adjacentes. La distance peut être remplacée par la durée ou le coût.

On sélectionne successivement les villes en commençant par celle de départ, D. C'est l'algorithme qui impose l'ordre de sélection. Dans le tableau que l'on va construire, 6(E) dans la colonne F par exemple signifie que provisoirement, la plus courte distance entre D et F est de 6 km au total en passant par E. Lorsque la ville F sera sélectionnée, le provisoire deviendra définitif (case sur fond gris qui n'est plus modifiée par la suite).

Le travail peut être fait sur tableur : une nouvelle ligne du tableau est créée par copier-coller à chaque étape.

Partant de D, B est à une distance totale de 3 km : on note 3(D) dans la colonne B.

E est à une distance totale d' 1 km : on note 1(D) dans la colonne E.

Pour les autres villes, pas encore visitées, on note « inf » pour infini.

La prochaine ville sélectionnée est E (plus courte distance : 1 km).

D	B	E	C	F	A	prochaine ville sélectionnée
	3 (D)	1 (D)	inf	inf	inf	**E**

Partant de D, **via E,** B est à une distance totale de 2 km (1+1) : on note 2 (E) dans la colonne B

car 2 est plus petit que 3 déjà présent. Si le résultat est plus grand, on ne change rien.

Idem en l'absence de route directe entre E et B.

C est à une distance totale de 4 km (1+3) : on note 4 (E) dans cette colonne.

F est à une distance totale de 6 km (1+5) : on note 6 (E) dans la colonne F.

A n'est pas directement reliée à E : on ne change rien dans la colonne A.

La prochaine ville sélectionnée est B (plus courte distance : 2 km).

D	B	E	C	F	A	prochaine ville sélectionnée
	2(E)	1 (D)	4 (E)	6 (E)	inf	**B**

Etc. Les élèves continuent le travail commencé.

L'algorithme se poursuit tant que la ville d'arrivée n'est pas sélectionnée. On obtient finalement :

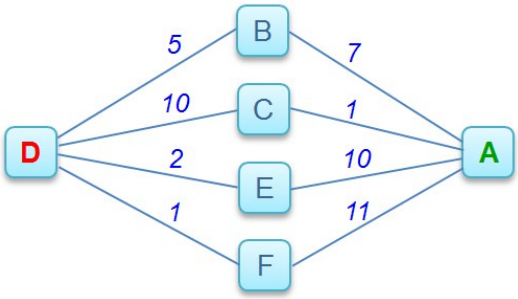
D	B	E	C	F	A	prochaine ville sélectionnée
	2(E)	1 (D)	4 (E)	5 (C)	6 (F)	**A**

Puisqu'on a mémorisé à chaque fois la ville précédente, on remonte à la ville de départ par le plus court chemin. Dans la colonne A, on voit qu'on vient de F. On regarde dans la colonne F : on arrive en F par C. Etc. On obtient AFCED. Le plus court chemin est donc DECFA avec une distance totale de 6 km (résultat direct, voir colonne A, puisqu'on mémorise à chaque fois la distance totale).

L'algorithme de Djjkstra est bien un algorithme de parcours en largeur du graphe : les villes sélectionnées sont d'abord D, puis E et B, puis C et F, puis A.

Une synthèse de l'algorithme sera utile pour la suite (quatrième étape); elle peut se faire à l'issue de ce qui suit.

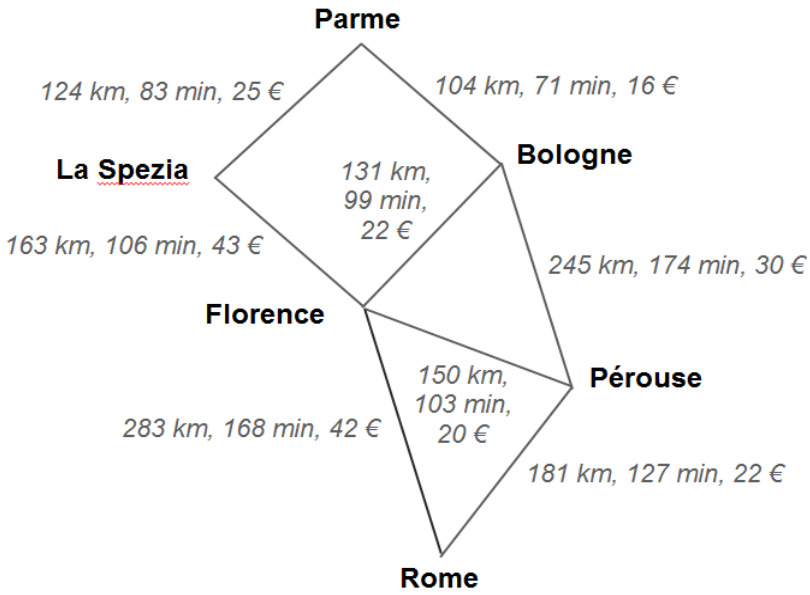
Pour les élèves ayant du mal à comprendre comment fonctionne l'algorithme, un exemple de ce type peut être proposé avant de passer à la suite (plus court chemin entre D et A) :



1.2 Deuxième étape : un cas réel de recherche

En général, on recherche le chemin le moins long, le plus rapide, ou le moins cher …

Tous les chemins mènent à Rome, mais pour un parmesan (un habitant de Parme !), quel est le moins long ? Le plus rapide ? Le moins cher ? Est-ce le même chemin dans les trois cas ?



Source (distances, durées, et coûts) : ViaMichelin.

Exploitation pédagogique : on propose aux élèves de réinvestir ce qu'ils ont appris. On peut les répartir en 3 groupes : l'un recherche la plus courte distance entre Parme et Rome, le second la plus courte durée, et le troisième le moindre coût. Ils confrontent ensuite leurs résultats. On peut aussi proposer une version orientée du graphe (certaines voies à sens unique).

1.3 Troisième étape : représentation des données du graphe en vue d'une implémentation de l'algorithme

Les élèves représentent la matrice d'adjacence du graphe précédent (au choix : distances, durées, coûts); c'est la structure qui sera retenue pour l'implémentation (plus simple).

Dans cette matrice, a_{ij} représente le coefficient (distance, durée, ou coût) de la route reliant la ville i à la ville j. Pour deux villes non reliées entre elles par une route directe, on a choisi $a_{ij}=0$.

$$\begin{pmatrix} 0 & 3 & 1 & 0 & 0 & 0 \\ 3 & 0 & 1 & 2 & 0 & 0 \\ 1 & 1 & 0 & 3 & 5 & 0 \\ 0 & 2 & 3 & 0 & 1 & 3 \\ 0 & 0 & 5 & 1 & 0 & 1 \\ 0 & 0 & 0 & 3 & 1 & 0 \end{pmatrix}$$

Dans le premier exemple, en considérant que les villes D, B, E, C, F, A correspondent respectivement aux lignes et colonnes 0, 1, 2, 3, 4, 5 de la matrice, on obtient la matrice ci-contre.

Dans la liste d'adjacence, à la suite de chaque ville, on indique celles qui lui sont adjacentes (villes reliées par une route directe) :

D	→	B, E
B	→	D, E, C
E	→	D, B, C, F
C	→	B, E, F, A
F	→	E, C, A
A	→	C, F

Remarque : avec la matrice d'adjacence, la recherche de l'existence d'une route directe entre deux villes est immédiate, alors que lorsqu'on veut trouver si une ville a une voisine, c'est avec la liste d'adjacence que la recherche est la plus rapide.

En vue d'une implémentation (en langage Python par exemple), les élèves peuvent utiliser la fenêtre « Python shell » pour s'approprier le fonctionnement des listes; il faut prévoir quelques exercices simples.

Création de la liste correspondant à la matrice :

```
>>> m_adjac = list()
```

Création des 6 listes correspondant aux lignes de la matrice :

```
>>> m_adjac.append([0,3,1,0,0,0])
>>> m_adjac.append([3,0,1,2,0,0])
>>> m_adjac.append([1,1,0,3,5,0])
>>> m_adjac.append([0,2,3,0,1,3])
>>> m_adjac.append([0,0,5,1,0,1])
>>> m_adjac.append([0,0,0,3,1,0])
```

```
>>> print m_adjac
[[0, 3, 1, 0, 0, 0], [3, 0, 1, 2, 0, 0], [1, 1, 0, 3, 5, 0],
[0, 2, 3, 0, 1, 3], [0, 0, 5, 1, 0, 1], [0, 0, 0, 3, 1, 0]]
```

Seconde ligne de la matrice (la première porte l'indice 0) :

```
>>> print m_adjac[1]
[3, 0, 1, 2, 0, 0]
```

Accès à un coefficient de la matrice :

```
>>> print m_adjac[4][2]
5
```

1.4 Quatrième étape : implémentation de l'algorithme (ou d'une partie), ou compréhension d'un programme fourni

Le programme ci-dessous est un exemple assez concis d'implémentation en Python, dont le principe est directement inspiré du travail fait précédemment à l'aide des tableaux (première et deuxième étapes). Il suffit d'adapter la partie en rouge au graphe concerné.

```
Nvilles = 6
L0 = [0,3,1,0,0,0]                    # la matrice d'adjacence est mise en place de façon très simple ici
L1 = [3,0,1,2,0,0]                    # la méthode append (cf. étape 3) est utilisée dans la suite
L2 = [1,1,0,3,5,0]
L3 = [0,2,3,0,1,3]
L4 = [0,0,5,1,0,1]
L5 = [0,0,0,3,1,0]
m_adjac = [L0,L1,L2,L3,L4,L5]         # m_adjac[4][2] contient la valeur 5, etc. (cf. étape 3)
DIJ=list()                            # la liste DIJ mémorise les données du tableau (cf. étape 1)
for i in range (Nvilles):
    DIJ.append([1000000,"X","N"])     # voir commentaire page suivante
ville_select=0                        # numéro de la ville sélectionnée; 0 = ville de départ
dist_interm=0                         # distance pour arriver à la ville sélectionnée; 0 au départ
while ville_select != Nvilles-1:
    minimum=1000000
    for n in range(1,Nvilles):
        if DIJ[n][2]=="N":
            dist=m_adjac[ville_select][n]
            dist_totale=dist_interm+dist
            if dist != 0 and dist_totale < DIJ[n][0]:
                DIJ[n][0]=dist_totale
                DIJ[n][1]=ville_select
            if DIJ[n][0]<minimum:
                minimum=DIJ[n][0]
                pville_select=n
    ville_select=pville_select        # pville_select = numéro de la prochaine ville sélectionnée
    DIJ[ville_select][2]="O"
    dist_interm=DIJ[ville_select][0]
    for i in range(1,Nvilles):
        print DIJ[i]
    print "\n"
chemin=list()                         # reconstitution du plus court chemin, en partant de la ville
d'arrivée
ville=Nvilles-1
chemin.append(ville)
while ville != 0:
    ville=DIJ[ville][1]
    chemin.append(ville)
print "plus court chemin, à lire à l'envers : ",chemin
print "distance totale : ",DIJ[Nvilles-1][0]
```

Interprétation des résultats donnés par le programme proposé

À chaque étape, on affiche pour chaque ville, sauf celle de départ, la distance totale pour la rejoindre, la ville précédente, et 'O' si ladite ville est sélectionnée ou l'a déjà été, 'N' sinon.

Pour chaque ville, la distance totale est préalablement initialisée à une grande valeur, et la ville précédente à 'X'.

Rappelons la numérotation des villes (voir page 8) : D, B, E, C, F, A ↔ 0, 1, 2, 3, 4, 5. Le dernier affichage correspond exactement à ce qu'on obtient en déroulant l'algorithme manuellement (voir page 6).

La ligne 1 : [2, 2, 'O'] signifie que pour la ville 1, c'est à dire B, la distance totale est de 2 km en venant de la ville 2, c'est à dire E, et que cette ville a déjà été sélectionnée : 'O'.

Etc. La ligne 5 : [6, 4, 'O'] signifie que pour la ville 5, c'est à dire A, la distance totale est de 6 km en venant de la ville 4, c'est à dire F, et que cette ville est maintenant sélectionnée ; elle ne l'était pas à l'étape précédente. Or c'est la ville d'arrivée, donc le programme s'arrête : le plus court chemin est trouvé.

C'est exactement ce que donne le tableau en bas de page 6, dans le cas où l'on déroule l'algorithme à la main.

```
>>>
[3, 0, 'N']
[1, 0, 'O']
[1000000, 'X', 'N']
[1000000, 'X', 'N']
[1000000, 'X', 'N']

[2, 2, 'O']
[1, 0, 'O']
[4, 2, 'N']
[6, 2, 'N']
[1000000, 'X', 'N']

[2, 2, 'O']
[1, 0, 'O']
[4, 2, 'O']
[6, 2, 'N']
[1000000, 'X', 'N']

[2, 2, 'O']
[1, 0, 'O']
[4, 2, 'O']
[5, 3, 'O']
[7, 3, 'N']

[2, 2, 'O']
[1, 0, 'O']
[4, 2, 'O']
[5, 3, 'O']
[6, 4, 'O']

plus court chemin, à lire à l'envers :  [5, 4, 3, 2, 0]
distance totale :  6
>>>
```

2 / Quelques propositions d'activités

Facile : tester les activités de la seconde étape à l'aide de ce programme; comprendre en détail le rôle de la liste DIJ; faire tourner à la main le programme en parcourant une fois la boucle while, écrire tous les résultats et les comparer à ce qu'on obtient manuellement.

Plus difficile : la structure des données utilisées est fournie dans un petit cahier des charges (rôle de chaque liste ou variable); il s'agit alors d'implémenter la partie en caractères bleus, par exemple, les autres lignes étant accompagnées de commentaires.

Difficile : les éléments fournis sont les mêmes, mais il s'agit d'implémenter une grande partie voire tout l'algorithme, qui peut être établi comme travail de synthèse après l'étape 2 ou 3.