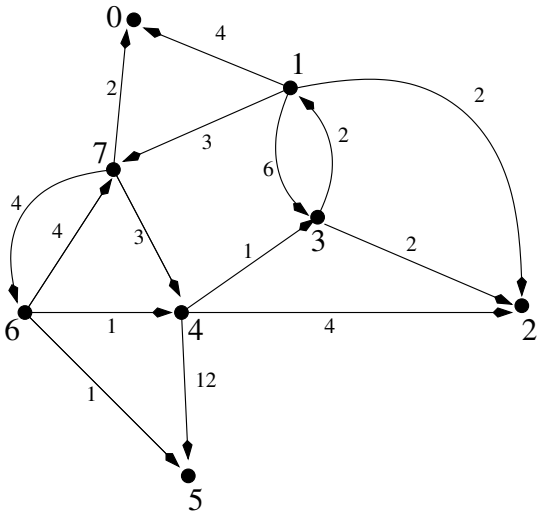
info404 : algorithmes sur les graphes

TD 5 : chemin optimaux

Pierre Hyvernat
Laboratoire de mathématiques de l'université de Savoie
bâtiment Chablais, bureau 22
téléphone : 04 79 75 94 22
email : Pierre.Hyvernat@univ-savoie.fr
www : http://www.lama.univ-savoie.fr/~hyvernat/

Exercice 1 : algorithme de Dijkstra

Question 1 : appliquez l'algorithme de Dijkstra sur le graphe suivant pour calculer les chemins optimaux à partir du sommet 4.



Question 2 : si on sait que tous les poids sont identiques, comment peut-on calculer les chemins optimaux à partir d'un sommet ?

Question 3 : cherchez un graphe qui comporte des poids négatifs pour lequel l'algorithme de Dijkstra produit un résultat faux.

Exercice 2 : algorithme par tri topologique

La question 3 de l'exercice précédent montre que Dijkstra ne fonctionne pas avec des poids négatifs. On va maintenant chercher un algorithme qui fonctionne avec des poids négatifs, mais pour éviter des problème, on interdit les cycles dans le graphe. (Un cycle de poids négatif empêche l'existence de chemin de poids minimal...)

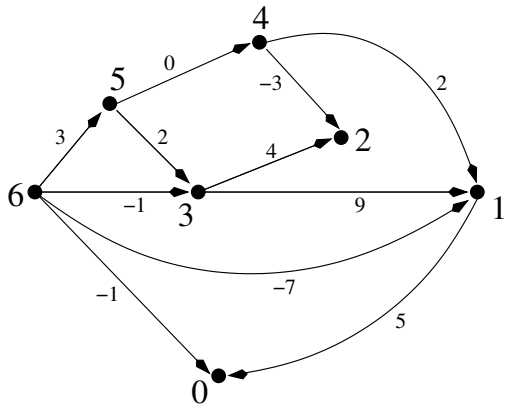
Question 1 : donnez une manière de calculer la distance minimale de x vers y en fonction des distances minimales de x vers les prédécesseurs de y.

Question 2 : d'après la question précédente, il suffit de calculer les distances minimales en vérifiant la condition suivante :

> si z est un prédécesseur de y, on calcule la distance de x vers z avant la distance de x vers y.

Expliquez comment utiliser le tri topologique (TD 3) pour calculer les distances minimales à partir d'un sommet x.

Question 3 : donnez quelques détails d'implantation pour votre algorithme, et testez le sur le graphe suivant :



Question 4 : essayez de trouver une variante de votre algorithme qui utilise elle aussi le tri topologique.

Exercice 3 : algorithme de Bellman

Un dernier algorithme est possible : on recherche un chemin de poids minimal dans un graphe qui contient (peut-être) des cycle et (peut-être) des poids négatifs. Comme dans certain cas il n'existe pas de chemin minimal, il faut pouvoir le repérer et en avertir l'utilisateur.

Question 1 : l'idée est relativement simple. On commence par initialiser les distances à $+\infty$ (sauf pour le sommet `x`). On fait ensuite un certain nombre de passages pour mettre à jour les distances de la manière suivante :

> pour chaque arc de `y` vers `z`, si `dist[y]+poids(yz) < dist[z]`, alors on met à jours la distance `dist[z]`. (Il est dans ce cas préférable de passer par `y`.)

Combien de fois faut-il faire ceci ?

Question 2 : que se passe-t'il si le graphe contient des cycles de poids négatif ?

Question 3 : comment repérer ces cycles de poids négatif, et que proposez-vous de faire dans ce cas ?

Question 4 : testez votre algorithme sur un graphe de votre choix (5 sommets et 8 arcs).

Question 5 : quelle est la complexité de cet algorithme ? Quel algorithme (Disjkstra, tri topologique ou Bellman) préféreriez en pratique ?