POLYTECH LILLE
Université Lille1 Sciences et Technologies

Département IMA / 4A (S8)
Informatique Fondamentale 2012/2013

Laure Gonnord
http://laure.gonnord.org/pro/

Exercices de TD IF - Feuille 5 Graphes

1 Modélisation

On modélisera les problèmes suivants sous forme de graphe et d'une question de graphe.

Exercice 1 *Une chèvre, un chou et un loup se trouvent sur la rive d'un fleuve ; un passeur souhaite les transporter sur l'autre rive mais, sa barque étant trop petite, il ne peut transporter qu'un seul d'entre eux à la fois. Comment doit-il procéder afin de ne jamais laisser ensemble et sans surveillance le loup et la chèvre, ainsi que la chèvre et le chou ?*

Exercice 2 *On souhaite prélever 4 litres de liquide dans un tonneau. Pour cela, nous avons à notre disposition deux récipients (non gradués !), l'un de 5 litres, l'autre de 3 litres... Comment doit-on faire ?*

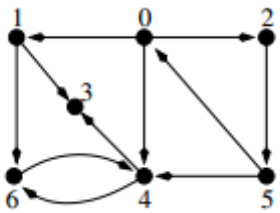
Ces deux exercices viennent de http://mathematiques.ac-bordeaux.fr/pedalyc/seqdocped/graphes/cahier/cahier.htm

Exercice 3 *(source M. Hurand, X) Des étudiants A, B, C, D, E et F doivent passer des examens dans différentes disciplines, chaque examen occupant une demi-journée :*
- *Algorithmique : étudiants A et B.*
- *Compilation : étudiants C et D.*
- *Bases de données : étudiants C, E, F et G.*
- *Java : étudiants A, E, F et H.*
- *Architecture : étudiants B, F, G et H.*

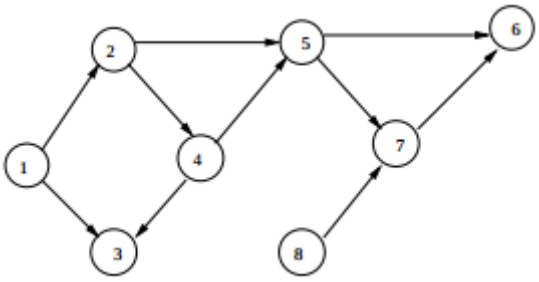
On cherche à organiser la session d'examen la plus courte possible.

2 Algorithmes classiques

Exercice 4 *Appliquer l'algorithme DFS au graphe suivant (en suivant les étapes du cours) :*



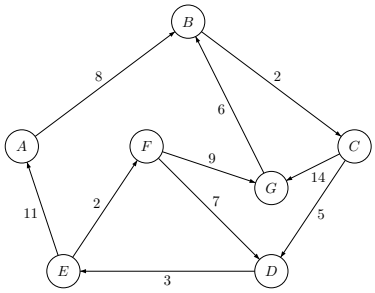
Exercice 5 *(source http://www-lipn.univ-paris13.fr/~sadki/TD3.pdf) Appliquer l'algorithme BFS au graphe suivant (en suivant les étapes du cours) :*



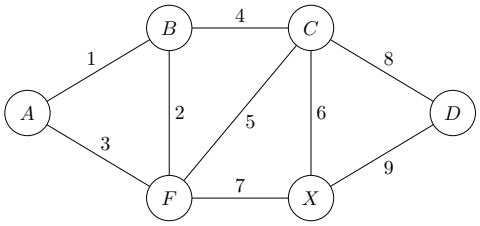
EXERCICE 6 *Calculer la fermeture transitive du graphe du cours en utilisant l'algorithme en* $O(n^4)$ *sur la matrice d'adjacence.*

EXERCICE 7 *Effectuer les étapes de l'algorithme de Floyd Warshall pour calculer les plus courts chemin entre toutes les paires de sommes de l'exemple du cours (transparent 32)*

EXERCICE 8 *En utilisant l'algorithme de Djikstra, calculer les plus courts chemins entre A et D de sommets du graphe suivant :*



EXERCICE 9 *Appliquer les algorithmes de PRIM et de KRUSKAL (cf feuille annexe) au graphe suivant pour trouver un arbre couvrant de poids minimum (pour PRIM, on commencera par le sommet X).*



7 Arbres couvrant de poids minimum

Un **sous-graphe** de $G = (S, A)$ est un graphe $G' = (S', A')$ tel que $S' \subset S$ et $A' \subset A$. Le sous-graphe G' **recouvre** G si $S = S'$. On parle aussi de sous-graphe couvrant.

Le but des deux algorithmes qui suivent est de construire, pour un graphe connexe valué aux arêtes, un arbre recouvrant de **coût minimum**, c'est-à-dire dont la somme des valuations des arêtes est minimum. Cet arbre contiendra tous les sommets du graphe, sera connexe avec un nombre minimum d'arêtes et sera minimum en coût.

Soit $G = (S, A)$ un graphe connexe d'ordre n et de taille p, valué aux arêtes.

Algorithme de Kruskal

```
Ordonner les arêtes par coût croissant sous la forme (a_1, ..., a_p).
Faire G' ← (S, ∅).
Faire i ← 0.
Tant que G' non connexe faire :
    i ← i + 1.
    Si a_i connecte deux composantes connexes de G' ajouter a_i à G' ; fin si.
fin tant que.
Retourner G'.
```

Remarque. On peut remplacer la condition `Tant que` G' `non connexe` par `Tant que` G' `a moins de` $n-1$ `arêtes`.

Proposition 8. *L'algorithme de* Kruskal *renvoie un arbre recouvrant de coût minimum.*

Algorithme de Prim

```
Choisir un sommet s dans S.
Faire G' ← ({s}, ∅).
Tant que G' ne recouvre pas G faire
    parmi les arêtes xx' de A avec x ∈ G \ G' et x' ∈ G',
    en choisir une de coût minimum ;
    ajouter x aux sommets de G' et xx' aux arêtes de G'.
Fin tant que.
Retourner G'.
```

Proposition 9. *L'algorithme de* Prim *renvoie un arbre recouvrant de coût minimum.*

Remarque. Des arbres recouvrant obtenus par les algorithmes de Kruskal et de Prim peuvent être distincts mais ont le même coût (à savoir le minimum possible).

Remarque. On peut montrer que pour un graphe G d'ordre n et de taille p, l'algorithme de Kruskal a une complexité de l'ordre de p^2, tandis que l'algorithme de Prim a une complexité de l'ordre de n^3.