**ALGORITHMIQUE AVANCEE**
**TP N°2**

**Objectif du TP :** Implémenter les algos les plus connus sur les graphes. Les TP doivent être finis avant le début de la séance suivante. Faites donc valider ce que vous avez fait au cours du TP précédent.

**ENONCE :**

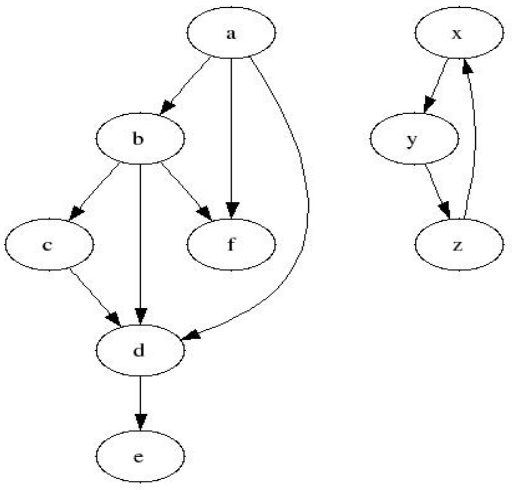
*Vous trouverez sur le site un fichier tp2.tgz contenant les sources de code java qui vous seront nécessaires pour ce TP. Enregistrez ce fichier sur votre compte et décompressez le avec la commande* **tar xvfz tp2.tgz**

Parmi les fichiers décompressés, vous trouverez :

- **Sommet.java** et **SommetLabelise.java** : Implémentent les Sommets. Regardez le code source... c'est simple. Posez des questions si vous ne comprenez pas quelque chose !
- **ArcOriente.java** et **ArcOrienteValue.java** : Implémentent les ArcsOrientes... Idem
- **GrapheOriente.java** : Implémente les GrapheOriente. Ne regardez pas le code source tout de suite...
- **TestGraphOriente.java** : Une classe qui ne contient qu'un main. Regardez le code, il est simple
- **lancer_graphviz** : un fichier en bash qui va nous aider a visualiser les graphes que l'on utilise... (voir plus loin)

**Exercice 1 : Compréhension des classes présentes : (3 points)**

1. Compilez l'ensemble des fichiers java. : **javac *.java**
2. Executez l'application TestGraphOriente : **java TestGraphOriente**
3. Vous constaterez que cette application a crée un nouveau fichier nommé "**Graphe1.dot**". Ce fichier décrit le graphe dans le language de Graphviz. Nous allons transformer ce fichier en une image à l'aide du programme lancer_graphviz : **./lancer_graphviz** Ceci a crée un fichier nommé Graphe1.dot.jpg. Regardez le !
4. Vérifiez que le parcours en largeur d'abord est cohérent avec l'image proposée.
5. Modifiez le code du fichier TestGraphOriente.java pour obtenir un graphe correspondant à celui de l'image ci dessous




**Vérifiez également le parcours en profondeur de cet arbre pour un départ en "a"**

**Exercice 2 : Un peu de profondeur d'abord. (5 points)**
Observez maintenant le contenu du fichier **GrapheOriente.java**
Posez des questions si vous ne comprenez pas quelque chose !

Tapez le code qui permettra de faire fonctionner la fonction de parcours en profondeur d'abord.
Vérifiez qu'elle fonctionne !

**Exercice 3 : Recherche du plus court chemin. (7 points)**
Vous allez devoir écrire ici deux fonctions.

- La première permettra de calculer le plus court chemin entre un sommet de départ et un sommet d'arrivée. La fonction renverra une liste chainée contenant les sommets rencontrés lors de ce chemin. Si aucun chemin n'existe, la fonction renverra une liste vide.
- La seconde permettre d'afficher les labels des sommets rencontrés lors du plus court chemin entre un sommet de départ et un sommet d'arrivée. *Vous pouvez utiliser la méthode* ***afficheLabelsCollectionSommet*** *cela devrait vous simplifer la vie.*

**Exercice 4 : Cycles (5 points)**
Ici, l'objectif sera d'implémenter une fonction **containsCycle** qui permettra de savoir si le graphe contient un cycle. A vous de trouver l'algorithme permettant de faire cela. Idéalement, votre algorithme doit être assez efficace (éviter de faire des opérations inutiles...).