Projet TP1forALL.

Vous pouvez utiliser librement cette implémentation, à condition de garder les noms des classes (identifiées par un « 4A » à la fin, pour « for All »).

La programmation est faite de la manière la plus simple, pour faciliter la compréhension.

Pour ce qui est de l'implémentation des versions orienté/non-orienté pondéré/non-pondéré matrice/liste d'adjacence, il faut essayer de factoriser ce qui est factorisable. Il est à noter qu'un graphe non-orienté est toujours représenté – en tant que matrice ou listes d'adjacence - de la même manière qu'un graphe orienté (avec une information redondante pour le graphe non-orienté). De plus, beaucoup d'algorithmes (surtout ceux de parcours) fonctionnent pareil pour les deux types de graphes. Et, enfin, le format d'entrée permet de récupérer l'information ligne par ligne de manière identique pour les deux types.

Par conséquent, l'implémentation fait le choix de ne pas différencier les deux types de graphes, mais de différencier au niveau de la représentation matrice/liste. Une fois que cela est fait, dans la représentation matrice on gère facilement (avec la même matrice) les graphes pondérés ou non. Dans la représentation liste, il y a deux types de nœuds (selon que le graphe soit pondéré ou non) qui sont définis séparément (sans héritage, qui aurait permis d'éviter quelques duplications au niveau des algorithmes, mais aurait été plus difficile à comprendre pour ceux qui ne connaissent que très peu Java). Cela implique deux attributs de type « liste d'adjacence » dont seulement un est utilisé, selon la pondération ou non du graphe.

Des méthodes sont fournies à la fois pour des graphes orientés et non-orientés. Lors de l'utilisation de ces méthodes, il faudra faire attention pour les utiliser seulement sur le bon type de graphe.

Compilation et exécution.

En allant dans le fichier /src dans lequel le fichier d'entrée <<nomdufichier>> est présent :
>javac -source 1.7 -target 1.7 *.java
>java Main4A <<nomdufichier>>