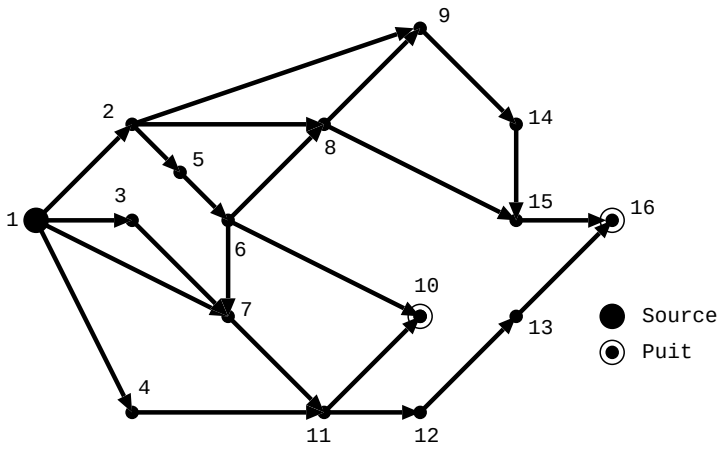
TP4 : Collections, *cast* & graphes orientés

Exercice

Nous proposons d'analyser la topologie et de fournir une librairie concernant les graphes orientés non cycliques.
Notre graphe a les propriétés suivantes :

- Il possède une ou plusieurs sources. Cela représente les points d'entrées du graphe. En d'autre terme, aucun nœud pointe sur les nœuds sources.
- Il possède un ou plusieurs puits. Cela représente les points de destination du graphe. En d'autre terme, ces nœuds pointent sur aucun autre nœud.
- Le graphe est composé de différents nœuds inter-connectés par des arcs.
- Chaque arc est orienté.
- Nous simplifions notre graphe, en interdisant un cycle de nœuds.
- Chaque nœud possède une valeur représentée par un objet de type `Object`.

Voici un exemple de graphe orienté non cyclique :



Questions :

1. Programmez une classe NODE permettant de modéliser un nœud de notre graphe.
2. Programmez une classe GRAPH permettant de modéliser un graphe.
3. Programmez une méthode capable de donner le nombre minimum de nœuds qu'il est nécessaire de parcourir pour atteindre un puit partant d'une source.
4. Programmez une méthode capable de donner le nombre maximum de nœuds qu'il est nécessaire de parcourir pour atteindre un puit partant d'une source.
5. Concevez un premier itérateur à l'aide d'une classe emboitée permettant de parcourir chaque nœud en profondeur d'abord. Dans l'exemple, ce type de parcours donne : 1, 2, 9, 14, 15, 16, 8, 5, 6, 10, 7, 11, 12, 13, 3, 4.
6. Puis un autre type itérateur capable de parcourir chaque nœud en largeur d'abord. Dans l'exemple, ce type de parcours donne : (1), (2, 3, 7, 4), (9, 8, 5, 11), (14, 15, 6, 12), (16, 10, 13).

Remarque : *Ce type de graphe peut représenter un certain type de labyrinthe, un parcours de marchandise à optimiser, un arbre d'héritage multiple, ...*