

# TP n° 1: Manipulation des tableaux, utilisation des instructions conditionnelles et répétitives

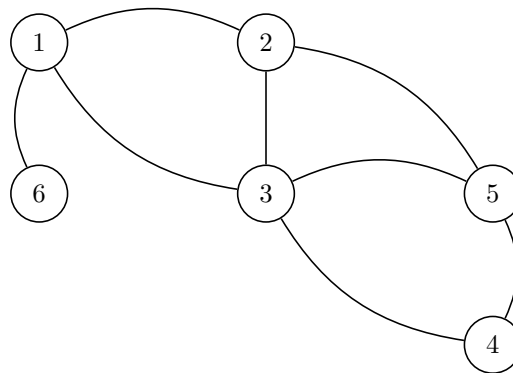
## 1 Exercice 1 : Modélisation des graphes

Un graphe  $G$  est un couple  $G = (S, A)$  où :

- $S$  est un ensemble fini de sommets
- $A$  est un ensemble fini d'arêtes

Dans l'exemple ci-dessous, le graphe  $G$  est composé de l'ensemble des sommet  $S = \{1, 2, 3, 4, 5, 6\}$  et de l'ensemble d'arêtes  $A = \{(1, 2), (1, 3), (1, 6), (2, 3), (2, 5), (3, 4), (3, 5), (4, 5)\}$

On appelle  $G$  un graphe **non orienté** (les arêtes n'ont aucune orientation) et **non valué** (aucune valeur n'est attribuée aux arêtes)



Créer une classe Ex1, dans le projet TP1. Dans la méthode `main`, écrire un programme en Java qui :

1. modélise un graphe non orienté et non valué en utilisant une structure de tableau à 2 dimensions. Le nombre des sommets et les arrêtes sont saisis par l'utilisateur en respectant le modèle suivant :

Donner le nombre des sommets et les arêtes :

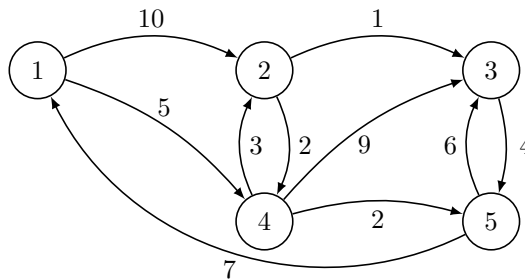
6 1-2 1-3 1-6 2-3 2-5 3-4 3-5 4-5

2. affiche les sommets voisins d'un sommet donné.

Utiliser `Integer.parseInt` pour convertir un objet de `String` à un entier

## 2 Exercice 2 : Graphes valués et orientés

Modifier votre programme (dans une nouvelle classe Ex2) pour modéliser un graphe orienté et valué. Prendre le graphe suivant comme exemple :



Écrire une méthode `succ` qui retourne l'ensemble de successeurs, et `pred` qui retourne l'ensemble de prédécesseurs, d'un sommet donné.

### 3 Exercice 3 : Application sur les graphes

Le calcul du plus court chemin est l'une des applications les plus courantes sur les graphes valués. L'algorithme de **Dijkstra** permet de résoudre le problème du plus court chemin en calculant la liste de chemins les plus courts d'un sommet donné vers tous les autres sommets. Écrire en Java la méthode **dijkstra** qui implémente l'algorithme suivant :

---

**Algorithm 3.1** Algorithme de Dijkstra

---

**Entrée :** S : sommet de départ, G : matrice du graphe orienté et valué

**Sortie :** chemin : tableau des chemins les plus courts

distance[S]  $\leftarrow$  0

chemin[S]  $\leftarrow$  S

**pour tout** sommet T  $\neq$  S **alors**

distance[T]  $\leftarrow \infty$

chemin[T]  $\leftarrow$  null

**fin pour**

**tant que** il reste des sommets non marqué **alors**

Choisir un sommet T non marqué tel que distance[T] soit minimale

marquer T

**pour tout** successeur U de T **alors**

**si** distance[U] > distance[T] + G[T][U] **alors**

distance[U]  $\leftarrow$  distance[T] + G[T][U]

chemin[U]  $\leftarrow$  T

**fin si**

**fin pour**

**fin tant que**

---