SAé 2.02 : Exploration d'un problème

Développement orienté objet

Objectifs

L'objectif est de développé des classes permettant de manipuler des graphes que vous utiliserez pour implémenter vos algorithmes. Vous devez donc implémenter les classes suivantes :

- `Graphe` : classe abstraite représentant un graphe ;
- `GrapheOriente` : classe représentant un graphe orienté qui implémente la classe abstraite `Graphe`.

Indication : votre modélisation et votre code doivent permettre de définir d'autres types de graphes : graphe non orienté, graphe valué, graphe orienté valué, graphe dirigé acyclique ... Vous devez donc être attentif à la conception de votre code pour qu'il soit le plus générique possible et vous permettre de définir d'autres types de graphes selon les besoins du sujet de la SAé.

Consignes

- le graphe doit être représenté par une liste d'adjacence ;
- vous pouvez définir d'autres classes ;
- vous définirez les méthodes selon les besoins que vous aurez pour implémenter les algorithmes du sujet ;
- vous serez attentif dans quelle classe une méthode doit être définie ;
- vous éviterez au maximum de dupliquer du code ;
- vous respecterez le principe d'encapsulation ;
- le langage de programmation est TypeScript.

Construction d'un graphe

Votre classe `Graphe` devra accepter deux type de construction :

- créer un graphe vide
- créer un graphe à partir d'un fichier texte contenant la description du graphe

On souhaitera aussi avoir la possibilité de sauvegarder un graphe dans un fichier texte.

Format des fichiers textes

On va être amené à lire et écrire des graphes dans un fichier texte. Le format du fichier est le suivant :

```
10 12
3 5 4
3 0 8
```

```
5 9 7
5 8 5
9 1 5
9 4 8
9 2 9
1 8 3
2 7 1
2 0 3
7 6 4
0 8 1
```

- la première ligne contient le nombre de sommets et le nombre d'arcs du graphe ;
- les lignes suivantes contiennent les arcs du graphe. Chaque ligne contient trois entiers : le premier est le sommet de départ, le deuxième est le sommet d'arrivée et le troisième est le poids de l'arc.

Des exemples de petits graphes sont fournis dans ARCHE.

Remarque : il peut y avoir plusieurs arcs reliant deux sommets. Dans ce cas, vous ne garderez que le premier arc lu.

Manipulation de fichiers texte en TypeScript

Module `fs` de Node

Pour lire un fichier, le plus simple est d'utiliser le module `fs` de Node. Ce module est installé par défaut. Les fonctions dont vous aurez besoin sont importées par l'instruction :

```
import { readFileSync, createWriteStream } from "fs";
```

Lecture d'un fichier

L'instruction suivante lit le contenu d'un fichier et stocke chaque ligne comme une chaîne de caractères dans un tableau de chaînes de caractères :

```
let lignes = readFileSync("~/Instances/Random4-n/Random4-n.21.0.gr",
"utf8").split("\n");
```

Le premier argument de la fonction `readFileSync` est une chaîne de caractères contenant le chemin jusqu'au fichier que l'on souhaite lire. Le second argument est l'encodage du fichier.

La fonction `readFileSync` retourne le contenu du fichier sous la forme d'une chaîne de caractères. La méthode `split` des chaînes de caractères permet de découper une chaîne de caractères en plusieurs chaînes en se basant sur un caractère comme délimiteur, ici le retour à la ligne.

Indication : si vous voulez découper une chaîne par rapport aux espaces par exemple, vous pouvez le faire avec la méthode `split`. Après les instructions :

```
let chaîne = "a 1 3";
let info = chaîne.split(" ");
```

la variable `info` sera un tableau de chaînes de caractères égal à `[ 'a', '1', '3' ]`.

Ecriture dans un fichier

Pour écrire dans un fichier, on commence par ouvrir un flux vers un fichier avec la fonction `createWriteStream` dont l'argument est une chaîne de caractères contenant le chemin vers le fichier :

```
let output = createWriteStream("IOFiles/output.txt");
```

Attention : cela efface le contenu du fichier s'il existe déjà.

Ensuite, la méthode `write` des flux d'écriture permet d'écrire dedans :

```
output.write("c random graph n=2097152 m=8388608 lo=0 hi=2097152 seed=1292225576\n");
output.write("p sp 2097152 8388608\n");
output.write("a 1182391 1259944 1442304\n");
output.write("a 1259944 1647105 225203\n");
```

Lorsque l'écriture dans le fichier est terminée, il faut fermer le flux :

```
output.end();
```

Rendu

- le diagramme de classes final ;
- le code final avec les algorithmes demandés dans le sujet ;
- il doit y avoir une cohérence entre votre diagramme de classes et votre code.