Hayar Malak
De Wancker Maddy
Margely Gwendal

BUT Informatique S2
IUT A de Lille

Rapport Graphes
SAE 2 1.2

Informations utiles :

Adresse de notre dépôt git :

https://gitlab.univ-lille.fr/sae2.01-2.02/2022/B-G3.git

numéro de de commit git :

a3e77a70a1bd327cc38ab5f0f13c430b8929698a

(message du commit : "**graphs : unused code removed**")

1. Un exemple d'illustration

Tutorés	Tuteurs
Bart Howl Année 1 Moyenne 10.1	Benoit Dupuis Année 2 Moyenne 14.1
Chris Hart Année 1, Moyenne 9	Caroline Baguelle Année 3 ,Moyenne 18.6
Donald MC Ronald Année 1,Moyenne 13	David Travo Année 3, Moyenne 14.1
Elon Musketeer Année 1,Moyenne 12.1	Emilie Paulin Année 2, Moyenne 16.9
Pierre Quiroule Année 1, Moyenne 6.7	Arthur Oui Année 2, Moyenne 15.2
Yann Ique Année 1, Moyenne 11	
Florine Deschamps Année 1, Moyenne 13.7	

2. La modélisation :

Introduction :

nous allons dans un premier temps faire une trace algorithmique des différentes affectations, et dans un deuxième temps, donner les matrices de chaque graphe.

petit point algorithmique :

Nous avons remarqué que l'algorithme calcule en effet le coût minimal de l'affectation mais donne des couples tuteurs/tutorés pas optimisés. Après (beaucoup) de tests, nous avons déduit que l'algorithme, après avoir calculé le coût minimal, retourne les couples tuteurs/tutorés en fonction de l'ordre dans lequel la première arrayList de la fonction ***CalculAffectation*** (ce qui représente la partie tuteurs de notre graphe ici).

La solution qu'on a trouvé pour remédier à ce problème est de **trier les arrayList des tuteurs par ordre décroissant** en fonction des critères qu'on voudrait implémenter, ici on les trié en fonction de leurs moyennes et leurs années d'études, **afin que les premiers dans l'ArrayList soit les étudiant les plus âgés et les plus forts, et donc ceux affectés par l'algorithme en premier. (**Nous avons aussi trié l'Arraylist de tutorés, bien que ça ne change cette fois-ci pas le résultat).

Le **coût minimal d'affectation** reste néanmoins le même, que les arrayLists soient triées ou pas, mais on a au final les couples qu'on veut, **c'est-à-dire les plus faibles avec les plus âgés et les plus forts.**

2.1 : Trace de l'algorithme

Précisions :

- Toutes les listes d'objets ci-dessous sont des **ArrayList<Etudiant>**

- Le méthode ***DefineAretes()*** : Elle définit le poids d'une arête entre deux sommets (un sommet = un étudiant) d'un graphes appartenant

à deux parties différentes, une première boucle parcourt *la liste des tuteurs* puis une deuxième parcourt *la liste des tutorés* afin que chaque tutoré (sommet) aie une arête avec chaque tutoré, le poids de l'arête est calculé en soustrayant la différence entre la moyenne es tutorés et des tuteurs, sachant que les deux parties ont été **sorted** auparavant en fonction de leur moyenne, en plus de l'année (pour les tuteurs).

- La méthode **copyOfRangeArrayList(ArrayList<Etudiant> liste, int from, int to) :** Fonctionne de la même façon que la méthode ***{copyOfRange}*** pour les tableaux.

- Notre classe **"Data"** fictive possède trois configuration d'affectation possibles (voir plus bas), mais on a intégré dans notre programme main la possibilité pour l'utilisateur de supprimer des tuteurs ou des tutorés manuellement.

Initialisation :

- On a une classe **« Etudiant »** qui sera l'objet type de notre graphe non orienté.

- On a deux parties **: « Tuteurs »** et **« tutores »,** les deux héritant de la classe **« Etudiant »** et pouvant donc être contenu dans le graphe.

- On implémente L'interface *Comparable* dans la classe **« Etudiant »** qui nous permet de trier les parties représentées dans le programme avec deux listes d'**Etudiant** : **Tuteurs** et **Tutorés,** en fonction de *leurs moyennes*.

- Les moyennes des **tutorés** sont triées par *ordre croissant*

- Les moyennes des **tuteurs** sont triées de la plus grande à la plus petite, une méthode **{sortByYear}** permet ensuite de classer en fonction

de l'année d'études, les 3ème années ayant la priorité et étant mit au début de la liste

- Ces deux parties sont encapsulées par une classe fictive **« Data »** qui définit Les deux listes avec le bon tri.

Début :

- On a trois configurations disponibles pour lorsqu'on instancie une affectation :

  § **Configuration [1] : MOINS de TUTEURS que de TUTORES**

  § **Configuration [2] : AUTANT de TUTEURS que de TUTORES**

  § **Configuration [3] : PLUS de TUTEURS que de TUTORES**

- Une fois la configuration choisie, un objet **Data** contenant les ArrayList des deux parties sera instancié.

- On entre dans une boucle dont on ne sortira que lorsque que tous les couples tuteurs-tutorés seront définis :

- On crée à l'intérieur une **liste d'Etudiant tutores temporaire**, pour ne pas modifier l'originale, sur laquelle on utilise la même méthode **{CopyOfRangeArrayList}** pour qu'elle ait le même nombre d'éléments que la liste des tuteurs.

- On est maintenant sûrs que **les deux parties**, tuteurs et tutorés ont la même taille, ce qui nous évitera une erreur avec le graphe.

- On y instancie un **graphe non orienté**, Contenant des **Etudiant**.

- On appelle la fonction **defineSommet()** pour définir tous les sommets du graphes, puis **defineAretes()** pour définir toutes les différentes arêtes du graphe.

- On effectue la première affectation avec en paramètre le graphe, la liste des tuteurs et la liste temporaire de tutorés.

Enchainement :

Les différentes configurations après la première affectation :

~ **Configuration [1] : MOINS de TUTEURS que de TUTORES :**

- Une fois la première affectation effectuée, il y'a encore des étudiants pas encore affectés.
- On set la liste des tutorés (avec la méthode **{CopyOfRangeArrayList}**) pour qu'elle ne contienne plus que ces derniers.
- Avec la même méthode on **set** la liste des **tuteurs** pour qu'elle soit de la même taille que celle des **tutorés** si plus grande, mais inchangée si plus petite.
- On reboucle encore pour une **nouvelle affectation**, on instancie un nouveau **graphe**, et on continue jusqu'à ce que tous les tutorés soient affectés.

~ **Configuration [2] : AUTANT de TUTEURS que de TUTORES**

- L'affectation se fera en un tour de boucle.

~ **Configuration [3] : PLUS de TUTEURS que de TUTORES**

- On a set au début de la boucle la liste des tuteurs pour qu'elle ait autant d'éléments que celle des **tutorés**, grâce à la méthode {**CopyOfRangeArrayList},**

- Elle prend les premiers donc en priorité les 3ème s années et ceux qui ont les meilleures moyennes.

- L'affectation se fera en un tour de boucle.

Fin :

- Une fois l'affectation faite, on se retrouve avec des couples de tuteurs-tutorés optimisés.

- Le programme affiche ensuite tous ces couples

2.2 Les graphes correspondants :

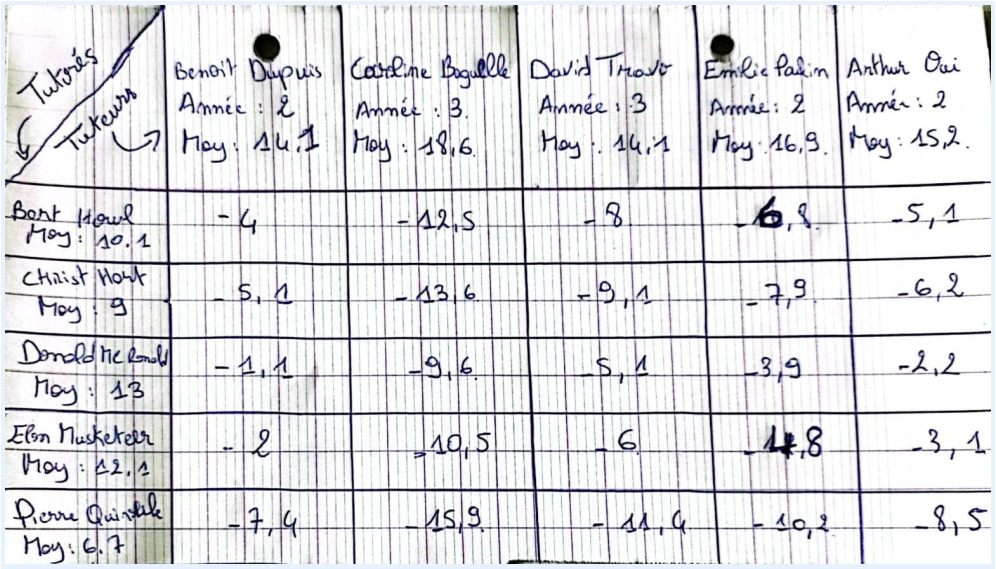
- Pour illustrer le fait qu'on donne la priorité aux 3ème années dans nos calculs, on soustrait un nombre x au poids de l'arête, ici on a soustrait 4 au total.
-
- § **Configuration [1] : MOINS de TUTEURS que de TUTORES**

+ Matrices des graphes : il y'en a deux car on a utilisé deux graphes pour cette configuration (il pourrait en avoir plus en fonction de combien il y'a de tuteurs)

  Note : on aurait pu utiliser un seul graphe pour modéliser cette configuration en ajoutant à l'ArrayList des tuteurs des poids null qui ne sont pas destinés à être affecté pour que le graphe soit parfaitement bipartie, mais on allait se retrouver avec des tutorés sans tueurs, faire un deuxième graphe comme

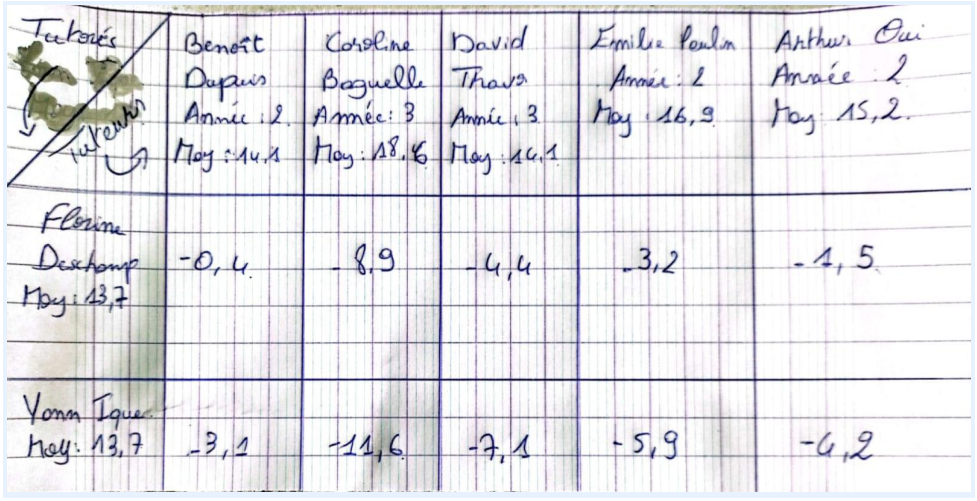
ci-dessous permettrait d'affecter ces tutorés qui aux meilleurs tuteurs selon le même calcul d'affectation.

- ***graphe 1 :***



Tutorés / Tuteurs	Benoit Dupuis Année : 2 Moy : 14,1	Caroline Boguelle Année : 3 Moy : 18,6	David Thavo Année : 3 Moy : 14,1	Emilie Paulin Année : 2 Moy : 16,9	Arthur Oui Année : 2 Moy : 15,2
Bart Houel Moy : 10,1	-4	-12,5	-8	-6,8	-5,1
Christ Hout Moy : 9	-5,1	-13,6	-9,1	-7,9	-6,2
Donald McDonald Moy : 13	-1,1	-9,6	-5,1	-3,9	-2,2
Elon Musketeer Moy : 12,1	-2	-10,5	-6	-4,8	-3,1
Pierre Quirelle Moy : 6,7	-7,4	-15,9	-11,4	-10,2	-8,5

- **graphe 2 :**



Tutorés / Tuteurs	Benoit Dupuis Année : 2 Moy : 14,1	Caroline Boguelle Année : 3 Moy : 18,6	David Thavo Année : 3 Moy : 14,1	Emilie Paulin Année : 2 Moy : 16,9	Arthur Oui Année : 2 Moy : 15,2
Florine Deschamp Moy : 13,7	-0,4	-8,9	-4,4	-3,2	-1,5
Yann Tique Moy : 13,7	-3,1	-11,6	-7,1	-5,9	-4,2

§ Configuration [2] : AUTANT de TUTEURS que de TUTORES

Tutorés / Tuteurs	Benoit Dupuis Année : 2 Moy : 14,1	Caroline Boguille Année : 3 Moy : 18,6	David Travo Année : 3 Moy : 14,1	Emilie Palin Année : 2 Moy : 16,9	Arthur Oui Année : 2 Moy : 15,2
Bart Houl Moy : 10,1	-4	-12,5	-8	-6,8	-5,1
Christ Hoyt Moy : 9	-5,1	-13,6	-9,1	-7,9	-6,2
Donald McDonald Moy : 13	-1,1	-9,6	-5,1	-3,9	-2,2
Elon Musketeer Moy : 12,1	-2	-10,5	-6	-4,8	-3,1
Pierre Quintale Moy : 6,7	-7,4	-15,9	-11,4	-10,2	-8,5

§ **Configuration [3] : PLUS de TUTEURS que de TUTORES**

Tutorés / Tuteurs	Benoit Dupuis Année : 2 Moy : 14,1	Caroline Boguille Année : 3 Moy : 18,6	David Travo Année : 3 Moy : 14,1
Bart Houl Moy : 10,1	-4	-12,5	-8
Christ Hoyt Moy : 9	-5,1	-13,6	-9,1
Donald McDonald Moy : 13	-1,1	-9,6	-5,1