SAE S2.02 – Rapport pour la ressource Graphes ===

Desmee Nathan, Ouhdda Anas, Belguebli Rayane Groupe A1

Version 1

Étude d'un premier exemple

Comme appariement compatible il y a :

- Adonia, A – Xolag, X
- Adonia, A – Zander, Z
- Bellatrix, B – Xolag, X
- Callista, C – Yak, Y

Ce sont les appariements compatibles car les autres appariements c'est soit ils n'ont pas de loisirs en commun et donc ils ne sont pas compatibles car dans l'énnoncé il est dit que les adolescents français ne peuvent qu'aller chez un hôte ayant au moins un loisir en commun. Et la deuxième raison que les autres appariements ne soient pas compatibles est l'allergie aux animaux que possède seulement Bellatrix, B qui n'est donc pas compatible avec Yak, Y qui à un animal même si ils ont un loisir en commun

Donc l'appariement optimal est Bellatrix–Xolag, Adonia–Zander, et Callista–Yak car Callista et Bellatrix ne peuvent aller que chez Yak et Xolag respectivement donc il ne reste plus que Zander pour Adonia parce qu'on s'occupe d'abord de ceux ayant le moins de possibilité.

Modélisation de l'exemple

Figure 1: Modélisation du Graphe

	Adonia, A	Bellatrix, B	Callista, C
Xolag, X	8	8	100
Yak, Y	110	108	6
Zander, Z	8	110	110

Figure 2: Matrice d'Adjacence

Dans l'exemple ci-dessus :

- les poids des arrêtes qui sont de 8 veulent dire que les adolescents on 1 loisir en commun.
- le poids de 6 veut dire que les adolescents ont 2 loisirs en commun.
- les poids qui ont une valeur de 110 veulent dire qu'il y a une imcompatibilité entre les adolescents
- et le poids de 108 veut dire qu'il y a une incompatibilité malgrès qu'ils aient un loisir en commun.

Modélisation pour la Version 1

Donc nous avons décidé de mettre un poids de base entre chaque adolescent de 10. On augmente de 100 si il y a imcompatibilité dû à la raison que les adolescents français ne peuvent qu'aller chez un hôte ayant au moins un loisir en commun et à l'allergie aux animaux. On enlève 2 pour chaques loisirs en commun.

Implémentation de la Version 1

(voir 'AffectationUtil.java')

(voir 'TestAffectationVersion1.java')

Exemple de vérification de l'incompatibilité

La particularité de cette exemple est que des adolescents n'ont pas mis de loisir, que d'autres en ont mis beaucoup et que l'adolescent A A est allergique aux animaux et il possède un animal.

Dans notre méthode weight dans la classe `*AffectationUtil.java*` nous avons passé de 100 à 1000 l'incompatibilité entre adolescents car si ils sont imcopatibles mais qu'ils ont beaucoup d'hobbies en commun le poids peut très vite descendre comme on enlève 2 pour chaque loisir en commun.

(voir 'TestAffectationVersion1.java méthode *testCompatibilityVsHobbies()*')

Voici les graphes en rapport avec les affectations :

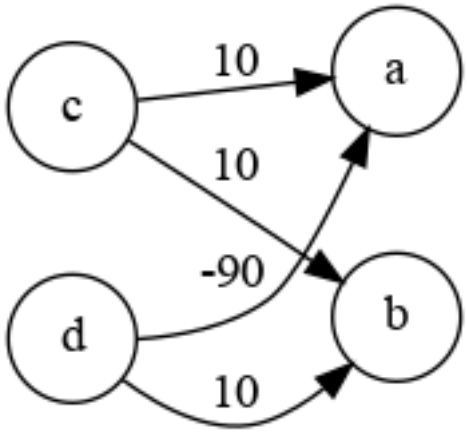

Figure 3: Les allemands qui vont chez les italiens

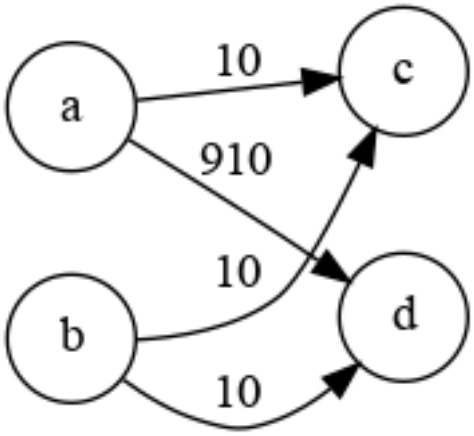

Figure 4: Les italiens qui vont chez les allemands

Version 2

Sera évaluée à partir du tag git `Graphes-v2`

Exemple minimal pour la gestion de l'historique

Voici le première exemple pris pour gérer l'historique qu'avec des contraintes liées à l'historique.

NAME	COUNTRY	BIRTH_DATE	GUEST_ANIMAL_ALLERGY	HOST_HAS_ANIMAL	GUEST_FOOD_CONSTRAINT	HOST_FOOD	HOBBIES	PAIR_GENDER	HISTORY
Jack	GERMANY		no	no					same
Foden	GERMANY		no	no					same
Walker	GERMANY		no	no					
Kane	GERMANY		no	no					other
Pedri	SPAIN		no	no					other
Nico	SPAIN		no	no					same
David	SPAIN		no	no					
Xavi	SPAIN		no	no					other

Figure 5: Exemple 1

Jack	Xavi
Harry	David
Kyle	Pedri
Phil	Nico

Figure 6: Ancien Correspondants

liste des appariements compatibles :

- Jack – Pedri
- Jack – Nico
- Jack – David
- Phil – Pedri
- Phil – Nico
- Phil – David
- Phil – Xavi
- Harry – Pedri
- Harry – Nico
- Harry – David
- Harry – Xavi
- Kyle – Nico
- Kyle – David
- Kyle – Xavi

Deuxième exemple pour la gestion d'historique

Voici le deuxième exemple pris pour gérer l'historique avec des contraintes liées à l'historique, la nourriture, les alergies aux animaux, les hobbies.

FOREAME	NAME	COUNTRY	BIRTH_DATE	GUEST_ANIMAL_ALLERGY	HOST_HAS_ANIMAL	GUEST_FOOD_CONSTRAINT	HOST_FOOD	HOBBIES	PAIR_GENDER	HISTORY
Adonia	A	FRANCE		no	yes			sports,technology		same
Bellatrix	B	FRANCE		yes	no	vegetarian		culture,science		
Callista	C	FRANCE		no	no	nonuts	meal	science,reading		same
Xolag	X	SPAIN		no	no			culture,technology		
Yak	Y	SPAIN		no	yes		nonuts	science,reading		
Zander	Z	SPAIN		yes	no	vegatarian	meal	technology		same

Figure 7: Exemple

A	X
B	Y
C	Z

Figure 8: Ancien Correspondants

liste des appariements compatibles :

- A – X
- B – X
- B – Y
- C – Y

Modélisation pour les exemples

Modélisation pour l'historique de la Version 2

Pour modéliser l'historique on a décidé que si ils ont été correspondant que ce soit l'hôte ou le visiteur si l'un dès de 2 mets "other" c'est considéré comme une contrainte donc on ajoute au poids 1000 même si l'autre mets "same", si l'un des 2 et l'autre ne mets rien ou les 2 mettent "same" on enlève 10 au poids.

Implémentation de l'historique de la Version 2

Ducoup pour implémenter l'historique nous avons ajouter une classe `*Affectations.java*` qui nous permet d'instancier une HashMap avec Teenager en clé et en valeur, ajouter une combinaison de Teenager ayant déjà été correspondant et elle a une méthode `history` qui retourne un poids -10 si il y a un des 2 qui a répondu "same" et aucun des 2 ayant répondu "other", 1000 si l'un des 2 a répondu "other" et 0 si ils nous jamais été correspondant. Nous avons aussi modifié la méthode `weight` dans la class `*AffectationUtil.java*`, nous avons ajouté en paramètre "Affectations history" qui est une HashMap de la class `*Affectations.java*` pour nous permettre de faire appel à la métode `history` situé dans *Affectations.java*.

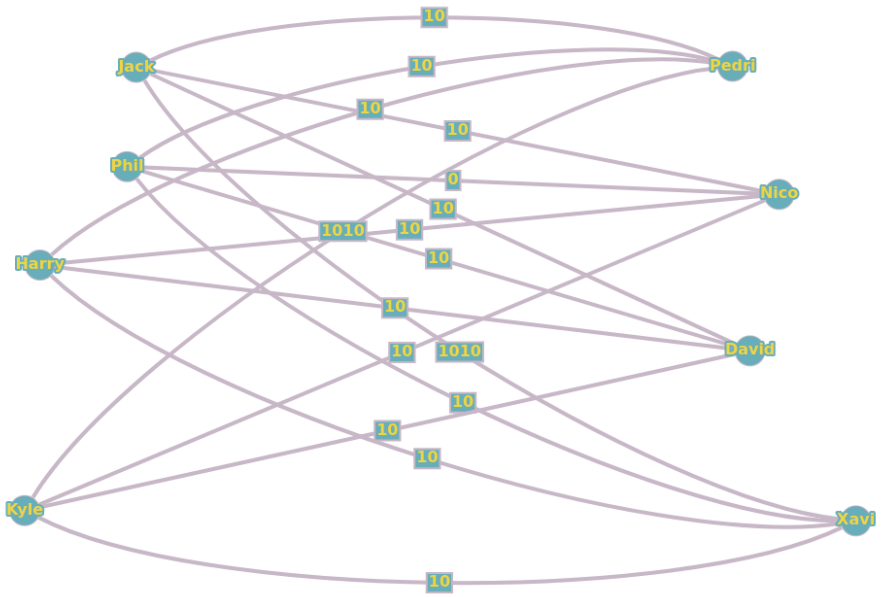

Figure 9: Modélisation du Graphe Exemple 1

	Pedri	Nico	David	Xavi
Jack	10	10	10	1010
Phil	10	0	10	10
Harry	10	10	10	10
Kyle	1010	10	10	10

Figure 10: Matrice d'Adjacence Exemple 1

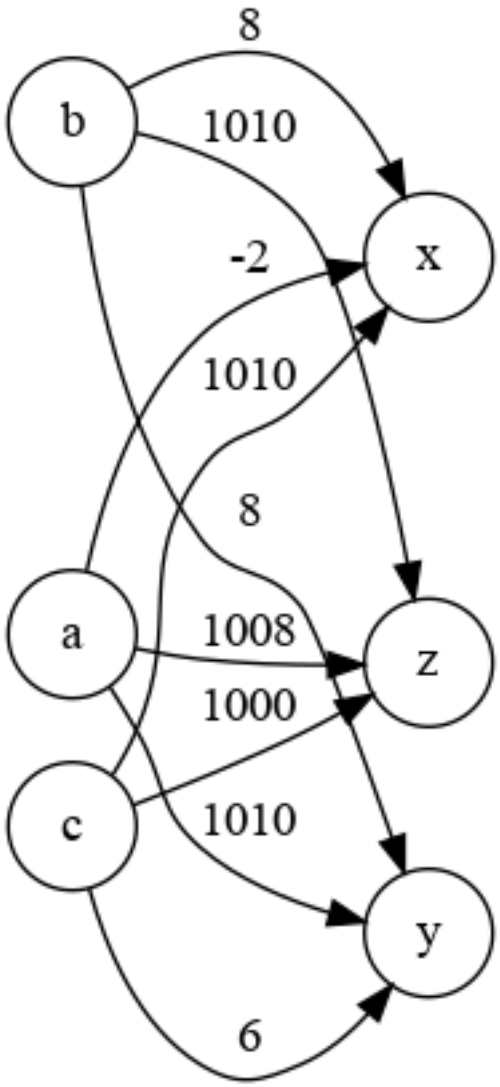

Figure 11: Modélisation du Graphe Exemple 2

	A	B	C
X	-2	8	1010
Y	1010	8	6
Z	1008	1010	1000

Figure 12: Matrice d'Adjacence Exemple 2

(voir 'AffectationUtil.java méthode *weight*')

(voir 'Affecatations.java')

Test pour l'historique de la Version 2

(voir 'TestAffectationVersion2.java méthode *testExemple1* & *testExemple2*')

Prendre en compte les autres préférences

Dans la fonction weight nous chargeons un fichier de séjour précédent pour faire appel à la méthode `history` et nous avons ajouté une vérification si 2 teenagers on plus de 5 hobbies en commun on arrête de diminuer le poids par 2. Et comme dernière préférence celle du genre si elle est respecté on enlève 4 qui est le double du poids de la préférence.