Projet N°01 - Python
Semestre 1 - 2019/2020
LE NOMBRE DE SCHUR S(n)

---

1. Préambule

Dans le cadre de votre apprentissage de l'algorithmique et du langage Python, nous vous proposons de réaliser un premier projet. Le but est de mettre en pratique les compétences acquises lors des cours magistraux, travaux dirigés et travaux pratiques afin de réaliser une application concrète. Le projet proposé dans ce document permettra de tester votre capacité à:

- Implémenter une solution algorithmique à partir d'une description d'un problème
- Utiliser / Choisir les boucles
- Utiliser les structures conditionnelles et définir les bonnes conditions
- Utiliser les tableaux

2. Présentation du projet

Il y a un siècle, en 1916, le grand mathématicien Issai Schur (1875-1941) donnait naissance à une suite de nombres entiers ***S(1)***,***S(2)***,***S(3)***,… très intéressants et si mystérieux que, même 100 ans après, on ne sait encore presque rien sur eux.

Le problème concerne globalement la coloration d'entiers en utilisant un certain nombres de couleurs et en respectant un certain nombres de contraintes.

Schur pense dans son théorème que (**n** étant le nombre de couleurs à considérer) :

> **Théorème (Schur, 1916).** Pour n donné, il existe un nombre $S(n)$ défini ainsi : c'est le plus grand entier N pour lequel il est possible de colorier en n couleurs les entiers de 1 à N de façon à exclure tout triplet monochromatique de la forme $(a, b, a+b)$ avec $a, b, a+b$ compris entre 1 et N.

Pour Schur, un triplet (a, b, c) avec c = a+b est dit monochromatique si les trois entiers ***a***, ***b*** et ***c*** ont eu la même couleur après la phase de coloration.

De plus, il n'est pas interdit d'avoir des triplets de type (a, a, c) (càd. a= b et c = 2a).

Le but de ce projet est de créer un programme qui permettra de calculer **N** pour tout **n** (***n* >= *2***) (nombre de couleurs) donné.

Afin d'y parvenir, nous vous décrivons dans la section qui suit la démarche à suivre afin de vous permettre de créer votre programme final de manière progressive.

3. Démarche à suivre

3.1. Comprendre le problème

En se donnant une palette de ***n*** couleurs (***n* >= 2**), la question à se poser est : jusqu'à quel entier maximal ***N*** peut-on colorier avec ces ***n*** couleurs (chaque couleur doit être présente au moins une fois) chaque entier de ***1*** à ***N*** de façon à éviter complètement l'émergence de triplets ***(a,b,a+b)*** monochromatiques ?

Une fois trouvé ce ***N*** maximal, on aura notre réponse : ***S(n)=N***.

Pour trouver cette valeur de ***N*** pour ***n*** couleurs, aucune formule hermétique n'est requise. Il suffit de:

- Etant donné **N>n**,
- Dresser la liste de tous les triplets de la forme ***(a,b,a+b)*** dont les éléments constitutifs ***a***,***b***,***a+b*** sont compris entre ***1*** et ***N***.
- Trouver au moins une coloration incluant l'ensemble des ***n*** couleurs et dans laquelle aucun triplet n'est monochromatique.
- S'assurer que cette condition n'est pas vraie pour tous les coloriages possibles avec ***n*** couleurs et ***(N+1)*** entiers.

Exemple:

Voici un exemple qui explique le calcul de ***S(2)***. Nous supposons que, pour le nombre de couleur ***n=2***, on utilisera les couleurs Rouge et Bleu pour colorier les nombres. On va créer des séries d'entiers de 1 à ***N*** avec N allant de **n+1** à **S(2)+1**. Pour chaque ***N,*** on va vérifier s'il y a au moins une coloration qui vérifie le prédicat suivant:

Tous les triplets **(a,b,c)**, *avec* ***c=a+b*** *(****a*** *peut être égal à* ***b****), créés avec les entiers de 1 à* ***N*** *ne sont pas monochromatiques. Il suffit d'avoir un seul triplet monochromatique pour que le prédicat soit faux.*

Le tableau ci-dessous comporte une ligne d'entête, et deux lignes pour expliquer le processus de calcul de ***S(2)***. Dans la première, on traitera en détail, toutes les séries d'entiers débutant avec l'entier "1" colorié en rouge. Dans la deuxième on ne gardera que les séries qui commencent par l'entier "1" colorié en bleu et dont l'évaluation du prédicat peut éventuellement être vraie. Toutes les colorations dont on a la certitude que le prédicat n'est pas vérifié ne seront pas traitées.

En fixant la couleur du premier entier au rouge "1", le deuxième peut être rouge ou bleu. De ce fait, avec ***N = 2***, on aura les deux colorations suivantes: "12" et "12". Les triplets respectifs à ces deux colorations sont (1, 1, 2) et (1, 1, 2).

N=1	N=2	N=3	N=4	N=5	Liste des triplets pour chaque coloration; Eval(Prédicat) : N
				1 2 3 4 5	(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (1, 4, 5); (2, 2 , 4); (2, 3, 5) **F:5**
			1 2 3 4		(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (2, 2 , 4) **F:4**
				1 2 3 4 5	(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (1, 4, 5) ; (2, 2 , 4); (2, 3, 5) **F:4**
		1 2 3			(1, 1 , 2) ; (1, 2 , 3) **F:3**
				1 2 3 4 5	(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (1, 4, 5); (2, 2 , 4); (2, 3, 5) **F:5**
			1 2 3 4		(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (2, 2 , 4) **F:4**
				1 2 3 4 5	(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (1, 4, 5); (2, 2 , 4); (2, 3, 5) **F:5**
	1 2				(1, 1 , 2) ; Premier triplet monochromatique **F:2**
				1 2 3 4 5	(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (1, 4, 5); (2, 2 , 4); (2, 3, 5) **F:5**
			1 2 3 4		(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (2, 2 , 4) **F:4**
				1 2 3 4 5	(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (1, 4, 5); (2, 2 , 4); (2, 3, 5) **F:5**
		1 2 3			(1, 1 , 2) ; (1, 2 , 3) **F:3**
				1 2 3 4 5	(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (1, 4, 5); (2, 2 , 4); (2, 3, 5) **F:5**
			1 2 3 4		(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (2, 2 , 4) **F:4**
				1 2 3 4 5	(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (1, 4, 5); (2, 2 , 4); (2, 3, 5) **F:5**
1					
				1 2 3 4 5	(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (1, 4, 5); (2, 2 , 4); (2, 3, 5) **F:5**
			1 2 3 4		(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (2, 2 , 4) **F:4**
				1 2 3 4 5	(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (1, 4, 5); (2, 2 , 4); (2, 3, 5) **F:5**
		1 2 3			(1, 1 , 2) ; (1, 2 , 3) **V:3**
				1 2 3 4 5	(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (1, 4, 5); (2, 2 , 4); (2, 3, 5) **F:5**
			1 2 3 4		(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (2, 2 , 4) **F:4**
				1 2 3 4 5	(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (1, 4, 5); (2, 2 , 4); (2, 3, 5) **F:5**
	1 2				(1, 1 , 2) ; **V:2**
				1 2 3 4 5	(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (1, 4, 5); (2, 2 , 4); (2, 3, 5) **F:5**
			1 2 3 4		(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (2, 2 , 4) **V:4**
				1 2 3 4 5	(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (1, 4, 5); (2, 2 , 4); (2, 3, 5) **F:5**
		1 2 3			(1, 1 , 2) ; (1, 2 , 3) **V:3**
				1 2 3 4 5	(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (1, 4, 5); (2, 2 , 4); (2, 3, 5) **F:5**
			1 2 3 4		(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (2, 2 , 4) **F:4**
				1 2 3 4 5	(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (1, 4, 5); (2, 2 , 4); (2, 3, 5) **F:5**
			1 2 3 4		(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (2, 2 , 4) **F:4**
		1 2 3			(1, 1 , 2) ; (1, 2 , 3) **V:3**
				1 2 3 4 5	(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (1, 4, 5); (2, 2 , 4); (2, 3, 5) **F:5**
	1 2				(1, 1 , 2) ; **V:2**
			1 2 3 4		(1, 1 , 2) ; (1, 2 , 3) ; (1, 3 , 4) ; (2, 2 , 4) **F:4**
		1 2 3			(1, 1 , 2) ; (1, 2 , 3); **V:3** (1, 1
			1 2 3 4		, 2) ; (1, 2, 3) ; (1, 3, 4) ; (2, 2, 4) **F:4**
1					
	1 2				(1, 1 , 2) **F:2**

On observe que le triplet qui correspond à la première coloration est monochromatique donc l'évaluation du prédicat pour cette coloration donnera "Faux". Cependant l'évaluation donnera "Vrai" pour le deuxième triplet. Donc pour N=2 on a déjà trouvé une coloration qui vérifie le prédicat avant même de traiter la deuxième ligne du tableau où le premier "1" est colorié en bleu. On doit vérifier les N > 2 jusqu'à ce qu'on trouve un N pour lequel aucune coloration ne vérifie le prédicat.

Pour la première ligne du tableau, toutes les colorations possibles, débutant avec un “1” rouge, de N=2 à N=5 sont calculées et détaillées. Dans la deuxième colonne, on énumère pour chaque coloration tous les triplets possibles, ensuite les triplets monochromatiques sont barrées. La liste des triplets est suivie de la lettre “F” ou “V” ainsi que la valeur N. On note “F” si le prédicat est faux, c’est à dire, qu’il y a au moins un triplet monochromatique et “V” sinon. Toutes les colorations qui sont évaluées à faux, sont barrées dans la première colonne. Nous avons vu dans le paragraphe précédent que la coloration “12” pour N=2 n’était pas valide et on remarque dans le tableau que toutes les séries de N>2 qui débutent par cette sous série “12” ne sont pas valides non plus. Ceci est compréhensible parce que la liste des triplets associés à la série “12” est incluse dans les listes des triplets de toutes les séries qui débutent par cette sous série “12”. Le triplet (1, 1, 2) monochromatique sera présent dans les autres listes ; toute liste le comportant sera évaluée à Faux qu’elle contienne ou non d’autres triplets monochromatiques. Partant de cette observation (“évidence”), nous avons appliqué la règle suivante pour la deuxième ligne du tableau (le cas où “1” est bleu pour N=1). Si à un niveau N une série ne vérifie pas le prédicat, toute autre série d’un niveau supérieur à N et qui débute par la série du niveau N ne sera pas traitée parce qu’on a la certitude qu’elle ne sera pas valide. Si on observe le couple Eval(Prédicat):N associé à chaque coloration, on constate qu’on a deux fois **V:2,** quatre fois **V:3**, deux fois **V:4** et **zéro** fois **V:5.** Donc **S(2) = 4.**

3.2. Quelques exercices d’échauffement

1) Ecrire un programme qui permet de colorer les ***N*** premiers nombres entiers en deux couleurs t.q., les nombres pairs sont affichés en rouge, et les nombres impairs sont affichés en bleu.

> **Exemple**: **1, 2, 3, 4, 5, 6, 7, 8, 9, 10, ...**

Pour afficher un texte en couleur:

```
74  import random
75
76  W    =  '\033[0m'   # white (normal)
77  R    =  '\033[31m'  # red
78  G    =  '\033[32m'  # green
79  O    =  '\033[33m'  # orange
80  B    =  '\033[34m'  # blue
81  P    =  '\033[35m'  # purple
82
83
84  my_color = [W, R, G, O, B, P]
85
86  N=10
87
88  print (random.choice(my_color), N)
89  print (G, N)
90
```

```
Shell
>>> %Run test.py
    10
    10
```

2) Ecrire un programme qui permet de colorer les ***N*** premiers nombres entiers en ***n*** couleurs choisies aléatoirement (les ***n*** couleurs doivent etre présentes).

Exemple:

Pour ***n=2***:

$$1, 2, 3, 4, 5, 6, 7, 8, 9, 10, \ldots$$

3.3. Préparer le terrain

1) Ecrire un programme qui, à partir de trois entiers et leurs trois couleurs respectives, affiche VRAI si le triplet est monochromatique, FAUX sinon.

Exemple:

Poun ***n = 3***

- Ce triplet (8, 5, 13) est monochromatique
- Ce triplet (1, 9, 10) n'est pas monochromatique

2) Ecrire un programme qui à partir d'un entier ***p ≤ 100***, génère et affiche la liste de tous les triplets ***(a,b,a+b)*** avec ***a***, ***b***, ***a+b ≤ p.***

Exemple:

Pour p = 4

La liste de tous les triplets de type ***(a, b, a+b)*** est:

(1, 1, 2), (1, 2, 3), (1, 3, 4), (2, 2, 4).

Cette liste est complète excepté (2, 1, 3) et (3, 1, 4) que l'on peut ignorer car l'ordre entre ***a*** et ***b*** ne joue aucun rôle.

3) Ecrire un programme qui utilise 2 couleurs pour colorier ***N*** nombres entiers aléatoirement. Ce programme doit afficher VRAI si tous les triplets ne sont pas monochromatiques, FAUX sinon.

Exemple:

Soit le coloriage suivant: 1, 2, 3, 4 pour n=2 et N=4.

Les quatres triplets concernés deviennent:(1, 1, 2), (1, 2, 3), (1, 3, 4), (2, 2, 4)

4) Ecrire un programme qui à partir d'un nombre de couleurs ***n*** et d'un nombre d'entiers ***N***, génère une coloration aléatoire. Il affiche VRAI si la coloration générée inclut les ***n*** couleurs utilisées. FAUX sinon.

Exemple:
Pour ***n=2***, ***N=4***, le coloriage 1, 2, 3, 4 est valide
Pour ***n=3***, ***N=4***, le coloriage 1, 2, 3, 4 n'est pas valide.

5) **QT**: Avec seulement 2 couleurs, quel est le nombre total de coloriages possibles pour colorier **N** nombres entiers ?

6) **QT**: Et si le nombre de couleurs est ***n***, quel est le nombre total de coloriages possibles pour colorier **N** nombres entiers ?

7) Ecrire un programme qui convertit un nombre décimal en un nombre binaire (11 → 1011).

8) Généraliser le programme précédent afin de convertir un nombre décimal en base ***b*** (>1).

9) Ecrire un programme qui stocke la liste de toutes les colorations possibles à partir de ***n*** couleurs pour ***N*** nombres entiers et où les ***n*** couleurs y figurent. Le programme doit afficher la liste obtenue à la fin.

3.4. Appliquer

1) En se basant sur les programmes écrits précédemment, ainsi que le premier exemple illustratif, écrire un programme qui calcule ***S(2)***.

2) **QT**: Quel est le résultat final pour ***S(2)*** ? Justifiez.

3) Généraliser le programme et calculer ***S(n) avec n > 2***.

4) **QT:** Combien de nombres de Schur avez-vous réussi à calculer ? Expliquez le problème rencontré.

5) En suivant du code suivant, calculer le temps nécessaire pour trouver ***S(n)***.

```
import time

# Debut du decompte du temps
start_time = time.time()

# Mettez votre code ici …

# Affichage du temps d execution
print("Temps d execution : %s secondes ---" % (time.time() -
start_time))
```

4. Fonctions avancées

Un résultat théorique démontre qu'à partir de ***n >= 6***, il est possible de borner la valeur de ***S(n)*** comme suit:

$$\frac{3^n - 1}{2} \leq S(n) \leq 3 \times n! - 1$$

1) Ecrire un programme qui permet d'afficher pour n'importe quel valeur de ***n*** (***n >= 6)***, l'intervalle dans lequel se trouverait son résultat ***N***.

2) Pour les plus motivés: Réécrire le programme ***S(n)*** en utilisant les fonctions.

5. Consignes générales

1) Le projet est à réaliser en binôme. Un seul groupe à trois étudiants pourrait être accepté dans le cas d'une imparité dans le groupe.

2) La remise de votre travail doit inclure:

- Un fichier .zip contenant l'ensemble des programmes demandés (il faut consacrer un fichier à chaque programme).
- Un rapport de 10 à 15 pages permettant de:
  - détailler vos solutions (en langage algorithmique) et vos choix de structures de données.

3) Tout dossier remis doit être renommé comme suit : **NOM1_NOM2** (avec NOM1 et NOM2 : noms des étudiants qui ont réalisé le projet).

4) Le projet est à rendre sur Moodle.

5) Les travaux sont à rendre au plus tard le **07/11/2019 à 23:59**

6. Planning

- Mercredi 9/10/2019: Lancement du projet 1
- Semaine du 14/10/2019: Suivi du projet 1
- Semaine du 04/11/2019: Soutenances du projet 1 pour les groupes A, B et INT1.
- Semaine du 11/11/2019: Soutenances du projet 1 pour les autres groupes.

7. Modalités d'évaluation

La note finale du projet N°1 sera composée de :

1) La note du code

2) La note du rapport

3) La note de soutenance.