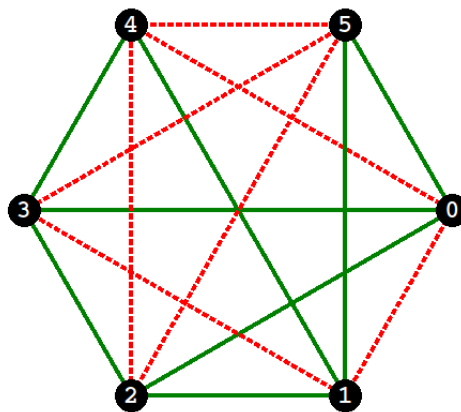


# Graphes et combinatoire de Ramsey

Tu vas voir qu'un problème tout simple, qui concerne les relations entre seulement six personnes, va demander énormément de calculs pour être résolu.



## Cours 1 (Le problème de Ramsey).

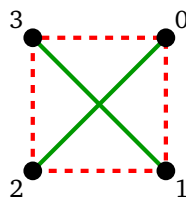
*Proposition. Dans un groupe de 6 personnes, il y a toujours 3 amis (les trois se connaissent deux à deux) ou 3 étrangers (les trois sont tous des inconnus les uns pour les autres).*

Le but de cette fiche est de faire démontrer à l'ordinateur cet énoncé. Pour cela nous allons modéliser le problème par des graphes, puis nous allons vérifier l'énoncé pour chacun des 32 768 graphes possibles.

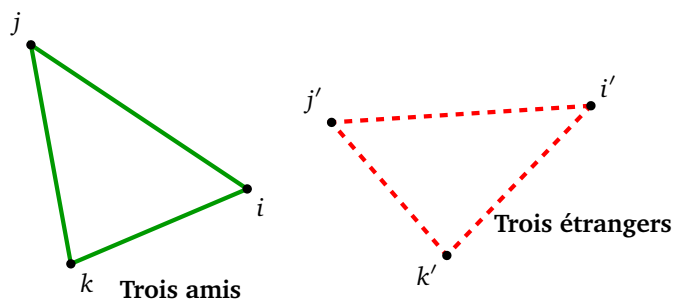
On considère  $n$  personnes. Pour deux personnes parmi elles, soit elles se connaissent (elles sont amies), soit elles ne se connaissent pas (elles sont étrangères l'une à l'autre). Nous schématisons cela par un graphe :

- une personne est représentée par un sommet (numéroté de 0 à  $n - 1$ ) ;
- si deux personnes sont amies, on relie les sommets correspondants par une arête verte ;
- sinon (elles sont étrangères), on relie les sommets correspondants par une arête pointillée rouge.

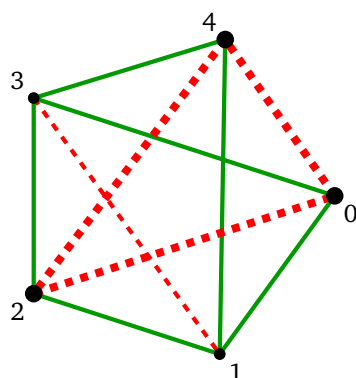
Le graphe ci-dessous signifie que 0 est ami avec 2 ; 1 est ami avec 3. Les autres paires ne se connaissent pas.



Un graphe vérifie le problème de Ramsey, s'il y a parmi ses sommets, ou bien 3 amis, ou bien s'il y a 3 étrangers.



Voici un exemple de graphe à 5 sommets qui vérifie l'énoncé : il possède bien 3 sommets étrangers (les sommets 0, 2 et 4), même s'il ne possède pas trois amis.



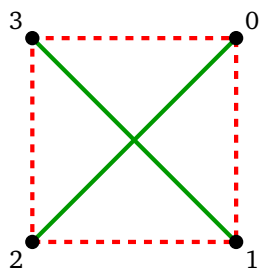
Un graphe avec  $n = 5$  qui vérifie l'énoncé de Ramsey

## Cours 2 (Modélisation).

Nous modélisons un graphe par un tableau à double entrée, contenant des 0 et des 1.

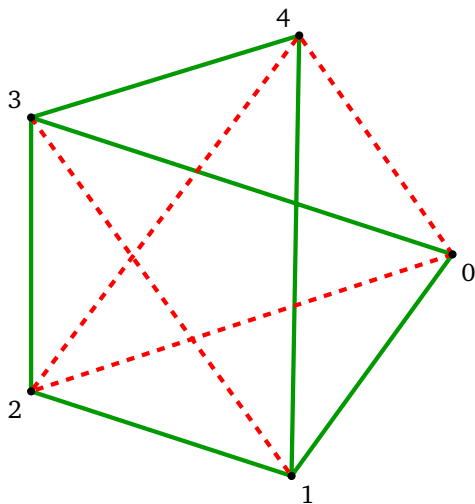
Soit un graphe ayant  $n$  sommets, numérotés de 0 à  $n - 1$ . Le **tableau du graphe** est un tableau de taille  $n \times n$  dans lequel on place un 1 en position  $(i, j)$  si les sommets  $i$  et  $j$  sont reliés par une arête, sinon on place un 0.

Premier exemple ci-dessous : les sommets 0 et 2 sont amis (car reliés par une arête verte) donc le tableau contient un 1 en position  $(0, 2)$  et aussi en  $(2, 0)$ . De même 1 et 3 sont amis, donc le tableau contient un 1 en position  $(1, 3)$  et  $(3, 1)$ . Le reste du tableau contient des 0.



		indice $j$				
		$j = 0$	$j = 1$	$j = 2$	$j = 3$	
indice $i$	$i = 0$	0	0	1	0	$n$
	$i = 1$	0	0	0	1	
	$i = 2$	1	0	0	0	
	$i = 3$	0	1	0	0	
		$n$				

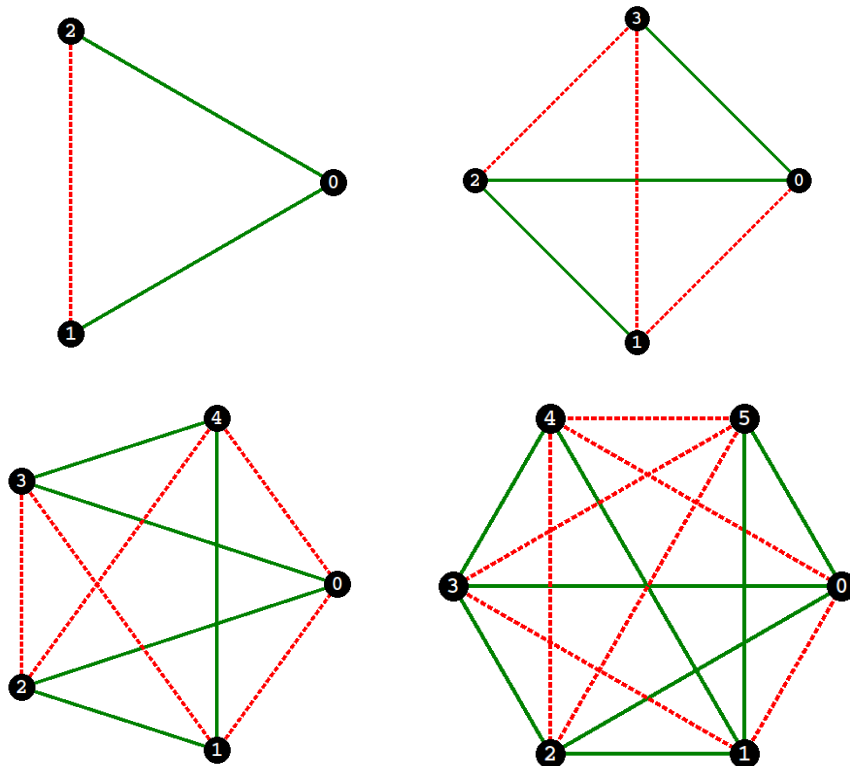
Voici un graphe plus compliqué et son tableau :



	$j = 0$	$j = 1$	$j = 2$	$j = 3$	$j = 4$
$i = 0$	0	1	0	1	0
$i = 1$	1	0	1	0	1
$i = 2$	0	1	0	1	0
$i = 3$	1	0	1	0	1
$i = 4$	0	1	0	1	0

### Activité 1 (Construire des graphes).

Objectifs : définir des graphes et tester si trois sommets donnés sont amis.



1. Définis le tableau des graphes des quatre exemples ci-dessus. Tu peux commencer par initialiser le tableau par

```
graphe = [[0 for j in range(n)] for i in range(n)]
```

Puis ajoute des commandes :

```
graphe[i][j] = 1 et graphe[j][i] = 1
```

N'oublie pas que si le sommet  $i$  est relié au sommet  $j$  par une arête, alors il faut mettre un 1 en position  $(i, j)$  mais aussi en position  $(j, i)$ .

2. Définis une fonction `voir_graphe(graphe)` qui permet d'afficher à l'écran le tableau d'un graphe. Ainsi le troisième exemple ci-dessus (avec  $n = 5$ ) doit s'afficher ainsi :

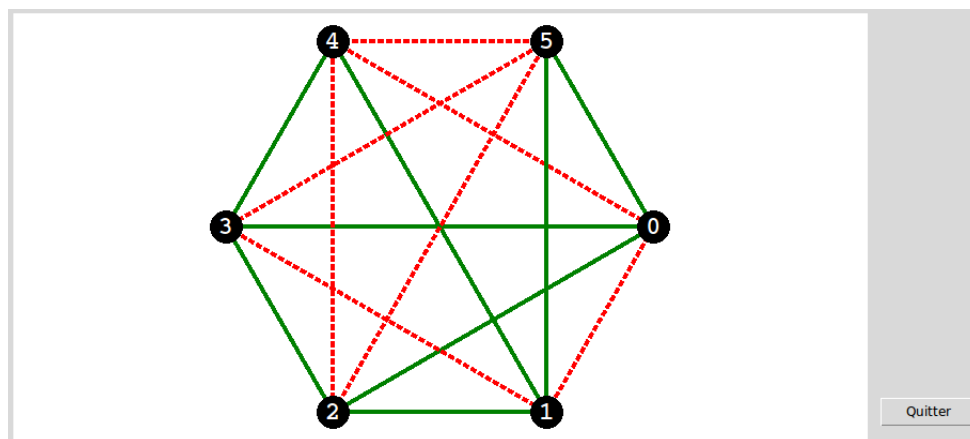
00110  
 00101  
 11000  
 10001  
 01010

3. On fixe trois sommets  $i, j, k$  d'un graphe. Écris une fonction `contient_3_amis_fixes(graphe,i,j,k)` qui teste si les sommets  $i, j, k$  sont trois amis (la fonction renvoie « vrai » ou « faux »). Fais le même travail avec une fonction `contient_3_etrangers_fixes(graphe,i,j,k)` pour savoir si ces sommets sont étrangers.

Trouve à la main sur le quatrième exemple, trois sommets amis ou étrangers et vérifie ta réponse à l'aide des fonctions que tu viens de définir.

## Activité 2 (Afficher des jolis graphes).

*Objectifs : dessiner un graphe ! Activité facultative.*



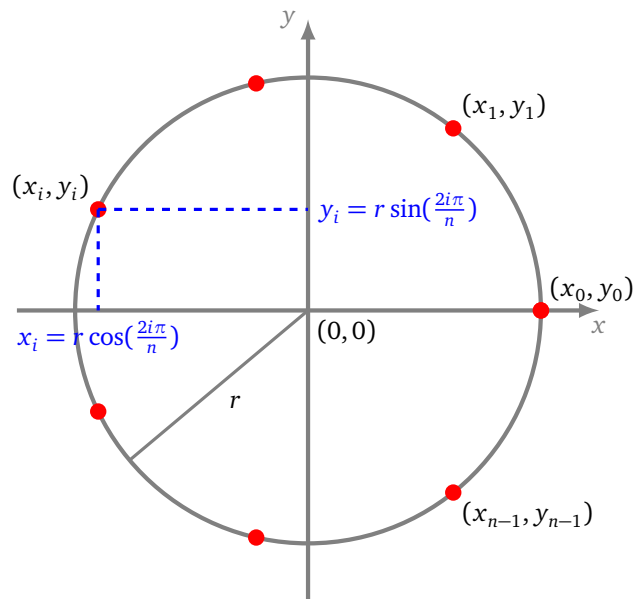
Programme l'affichage graphique d'un graphe par une fonction `afficher_graphe(graphe)`.

*Indications.* Cette activité n'est pas nécessaire pour la suite, elle aide juste à visualiser les graphes. Il faut utiliser le module `tkinter` et les fonctions `create_line()`, `create_oval()` et éventuellement `create_text()`.

Le point le plus délicat est d'obtenir les coordonnées des sommets. Tu auras besoin des fonctions sinus et cosinus (disponibles dans le module `math`). Les coordonnées  $(x_i, y_i)$  du sommet numéro  $i$  d'un graphe à  $n$  éléments peuvent être calculées par les formules :

$$x_i = r \cos\left(\frac{2i\pi}{n}\right) \quad \text{et} \quad y_i = r \sin\left(\frac{2i\pi}{n}\right).$$

Ces sommets sont situés sur le cercle de rayon  $r$ , centré en  $(0,0)$ . Tu devras choisir  $r$  assez grand (par exemple  $r = 200$ ) et décaler le cercle pour bien l'afficher à l'écran.



### Activité 3 (Écriture binaire avec des 0 non significatifs).

*Objectifs : convertir un entier en écriture binaire avec éventuellement des zéros non significatifs.*

Programme une fonction `decimal_vers_binaire(p,n)` qui affiche l'écriture binaire d'un entier  $p$  sur  $n$  bits. Le résultat est une liste de 0 et de 1.

Exemple.

- L'écriture binaire de  $p = 37$  est 1.0.0.1.0.1. Si on veut son écriture binaire sur  $n = 8$  bits alors il faut rajouter deux 0 non significatifs devant : 0.0.1.0.0.1.0.1.
- Ainsi le résultat de la commande `decimal_vers_binaire(37,8)` doit être `[0, 0, 1, 0, 0, 1, 0, 1]`.
- La commande `decimal_vers_binaire(37,10)` renvoie l'écriture de 37 en binaire sur 10 bits : `[0, 0, 0, 0, 1, 0, 0, 1, 0, 1]`.

*Indications.*

- Tu peux utiliser la commande `bin(p)` !
- La commande `list(ma_chaine)` renvoie la liste des caractères composant `ma_chaine`.
- Attention ! On veut une liste d'entiers 0 ou 1, pas des caractères '0' ou '1'. La commande `int('0')` renvoie 0 et `int('1')` renvoie 1.
- `ma_liste = ma_liste + [element]` ajoute un élément en fin de liste, alors que `ma_liste = [element] + ma_liste` ajoute l'élément en début de liste.

### Cours 3 (Sous-ensembles).

Soit  $E_n = \{0, 1, 2, \dots, n-1\}$  l'ensemble des entiers de 0 à  $n-1$ . L'ensemble  $E_n$  contient donc  $n$  éléments. Par exemple  $E_3 = \{0, 1, 2\}$ ,  $E_4 = \{0, 1, 2, 3\}$ ...

#### Sous-ensembles.

Quels sont les sous-ensembles de  $E_n$  ? Par exemple il y a 8 sous-ensembles de  $E_3$ , ce sont :

- le sous-ensemble  $\{0\}$  composé du seul élément 0 ;
- le sous-ensemble  $\{1\}$  composé du seul élément 1 ;

- le sous-ensemble  $\{2\}$  composé du seul élément 2 ;
- le sous-ensemble  $\{0, 1\}$  composé de l'élément 0 et de l'élément 1 ;
- le sous-ensemble  $\{0, 2\}$  ;
- le sous-ensemble  $\{1, 2\}$  ;
- le sous-ensemble  $\{0, 1, 2\}$  composé de tous les éléments ;
- l'ensemble vide  $\emptyset$  qui ne contient aucun élément !

**Proposition.** L'ensemble  $E_n$  contient  $2^n$  sous-ensembles.

Par exemple  $E_4 = \{0, 1, 2, 3\}$  possède  $2^4 = 16$  sous-ensembles possibles. Amuse-toi à les trouver tous !  
Pour  $E_6$  il y a  $2^6 = 64$  sous-ensembles possibles.

### Sous-ensembles de cardinal fixé.

On cherche seulement les sous-ensembles ayant un nombre  $k$  fixés d'éléments.

Exemples :

- Pour  $n = 3$  et  $k = 2$ , les sous-ensembles à deux éléments contenus dans  $E_3 = \{0, 1, 2\}$  sont les trois paires :  $\{0, 1\}$ ,  $\{0, 2\}$ ,  $\{1, 2\}$ .
- Pour  $n = 5$  et  $k = 3$ , les sous-ensembles à trois éléments contenu dans  $E_5 = \{0, 1, 2, 3, 4\}$  sont les 10 triplets :  $\{0, 1, 2\}$ ,  $\{0, 1, 3\}$ ,  $\{0, 2, 3\}$ ,  $\{1, 2, 3\}$ ,  $\{0, 1, 4\}$ ,  $\{0, 2, 4\}$ ,  $\{1, 2, 4\}$ ,  $\{0, 3, 4\}$ ,  $\{1, 3, 4\}$ ,  $\{2, 3, 4\}$ .

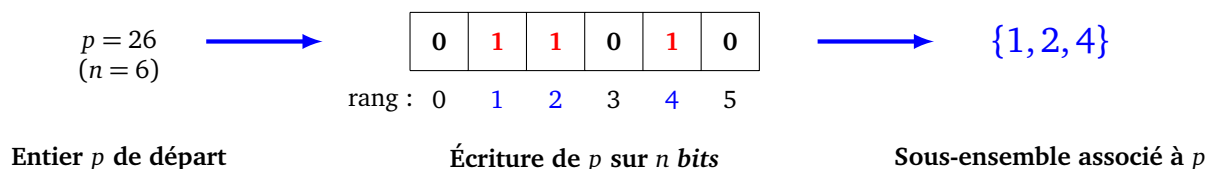
### Activité 4 (Sous-ensembles).

*Objectifs : générer tous les sous-ensembles afin de tester tous les triplets de sommets. Pour cela nous utiliserons l'écriture binaire.*

Voici comment nous associons à chaque entier  $p$  vérifiant  $0 \leq p < 2^n$  un sous-ensemble de  $E_n = \{0, 1, \dots, n-1\}$ .

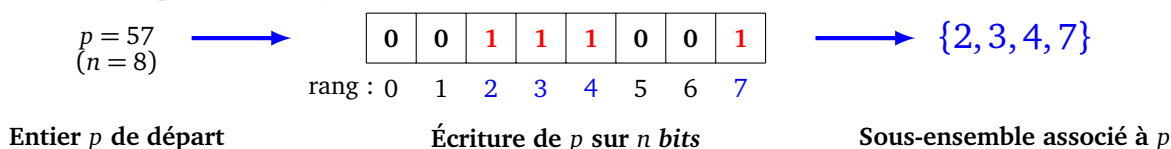
Commençons par un exemple, avec  $n = 6$  et  $p = 26$  :

- l'écriture binaire de  $p = 26$  sur  $n = 6$  bits est  $[0, 1, 1, 0, 1, 0]$  ;
- il y a des 1 au rang 1, 2 et 4 (en commençant au rang 0 à gauche) ;
- le sous-ensemble associé est alors  $\{1, 2, 4\}$ .



Autres exemples.

- Avec  $n = 8$  et  $p = 57$  dont l'écriture binaire sur 8 bits est  $[0, 0, 1, 1, 1, 0, 0, 1]$ , le sous-ensemble associé correspond aux rangs 2, 3, 4, 7, c'est donc  $\{2, 3, 4, 7\}$ .



- Avec  $p = 0$ , l'écriture binaire est formée uniquement de 0, le sous-ensemble associé est l'ensemble vide.
- Avec  $p = 2^n - 1$ , l'écriture binaire est formée uniquement de 1, le sous-ensemble associé est  $E_n = \{0, 1, \dots, n-1\}$  tout entier.

Nous modélisons un ensemble comme une liste d'éléments. Par exemple :

- L'ensemble  $E_4$  est pour nous la liste  $[0, 1, 2, 3]$ .
- Un sous-ensemble de  $E_4$  est par exemple la paire  $[1, 3]$ .
- L'ensemble vide est représenté par la liste vide  $[]$ .

1. Programme la fonction `sous_ensembles(n)` qui renvoie la liste de tous les sous-ensembles possibles de  $E_n = \{0, 1, 2, \dots, n-1\}$ .

Par exemple, pour  $n = 3$ , `sous_ensembles(n)` renvoie la liste (qui contient elle-même des listes) :

`[ [], [2], [1], [1, 2], [0], [0, 2], [0, 1], [0, 1, 2] ]`

C'est-à-dire les 8 sous-ensembles (en commençant par l'ensemble vide) :

$\emptyset \quad \{2\} \quad \{1\} \quad \{1, 2\} \quad \{0\} \quad \{0, 2\} \quad \{0, 1\} \quad \{0, 1, 2\}.$

*Indication.* Pour tester ton programme, vérifie que la liste renvoyée contient bien  $2^n$  sous-ensembles.

2. Dédus-en une fonction `sous_ensembles_fixe(n,k)` qui renvoie seulement les sous-ensembles de  $E_n$  ayant  $k$  éléments.

Par exemple, pour  $n = 3$  et  $k = 2$ , `sous_ensembles_fixe(n,k)` renvoie la liste des paires :

`[ [0, 1], [0, 2], [1, 2] ]`

Teste ton programme :

- Pour  $n = 4$  et  $k = 3$ , la liste renvoyée par `sous_ensembles_fixe(n,k)` contient 4 triplets.
- Pour  $n = 5$  et  $k = 3$ , il y a 10 triplets possibles.
- Pour  $n = 10$  et  $k = 4$ , il y a 210 sous-ensembles possibles !

Dans la suite nous utiliserons surtout les sous-ensembles à 3 éléments. En particulier, pour  $n = 6$ , les triplets inclus dans  $\{0, 1, 2, 3, 4, 5\}$  sont au nombre de 20 :

`[[3, 4, 5], [2, 4, 5], [2, 3, 5], [2, 3, 4], [1, 4, 5],  
[1, 3, 5], [1, 3, 4], [1, 2, 5], [1, 2, 4], [1, 2, 3],  
[0, 4, 5], [0, 3, 5], [0, 3, 4], [0, 2, 5], [0, 2, 4],  
[0, 2, 3], [0, 1, 5], [0, 1, 4], [0, 1, 3], [0, 1, 2]]`

#### Activité 5 (Théorème de Ramsey pour $n = 6$ ).

*Objectifs :* vérifier que tous les graphes ayant 6 sommets contiennent trois amis ou bien trois étrangers.

1. Programme une fonction `graphe_contient_3(graphe)` qui teste si un graphe contient 3 amis ou bien 3 étrangers. Il faut donc appeler les fonctions `contient_3_amis_fixes(graphe,i,j,k)` et `contient_3_etrangers_fixes(graphe,i,j,k)` de la première activité pour tous les triplets possibles de sommets  $(i, j, k)$ .

Pour les quatre exemples de la première activité, seul le quatrième (avec 6 sommets) vérifie le test.

2. Programme une fonction `voir_tous_graphes(n)` qui affiche tous les tableaux possibles de graphes à  $n$  sommets. Il y a  $N = \frac{(n-1)n}{2}$  tableaux possibles. Tu peux les générer par une méthode similaire à celle pour les sous-ensembles :

- pour chaque entier  $p$  qui vérifie  $0 \leq p < 2^N$ ,
- calcule l'écriture binaire de  $p$  sur  $N$  bits,
- remplis le tableau élément par élément, avec les 0 et les 1 de l'écriture binaire.

*Indications.* Pour remplir un tableau à partir d'une écriture binaire  $b$  donnée, tu peux utiliser une double boucle du type :

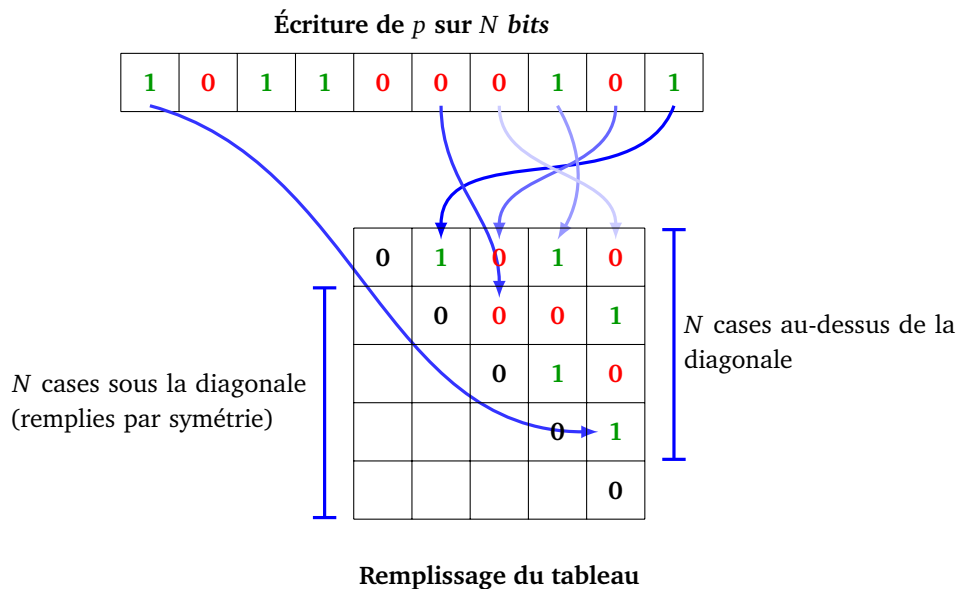
```
for j in range(0,n):
    for i in range(j+1,n):
```

```

b = liste_binaire.pop()
graphe[i][j] = b
graphe[j][i] = b

```

Voici le principe de cette boucle qui remplit la partie au-dessus de la diagonale (et aussi la partie en-dessous par symétrie). Cette boucle prend le dernier *bit* de la liste et le place sur la première case libre au-dessus de la diagonale ; puis l'avant-dernier *bit* est placé sur la seconde case libre... ; le premier *bit* de la liste remplit la dernière case libre.



3. Transforme la fonction précédente en une fonction `test_tous_graphes(n)` qui teste la conjecture « il y a trois amis ou trois étrangers » pour tous les graphes à  $n$  sommets. Tu dois trouver que :
  - pour  $n = 4$  et  $n = 5$  la conjecture est fausse. Donne un graphe à 4 sommets (puis à 5 sommets) qui n'a ni 3 amis, ni 3 étrangers ;
  - pour  $n = 6$  laisse l'ordinateur vérifier que, pour chacun des  $N = 2^{\frac{5 \times 6}{2}} = 32\,768$  graphes ayant 6 sommets, soit il possède 3 amis, soit il possède 3 étrangers.

### Activité 6 (Pour aller plus loin).

*Objectifs : améliorer ton programme et prouver d'autres conjectures. Activité facultative.*

1. Améliore ton programme afin qu'il vérifie la conjecture pour  $n = 6$  en moins d'une seconde.

*Idées.*

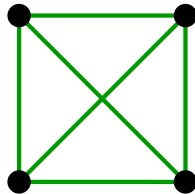
- Il faut générer la liste des triplets une fois pour toute au début du programme (et non à chaque nouveau graphe).
- Il ne faut pas générer une liste de tous les graphes possibles, puis les tester dans un second temps. Il faut en générer un puis le tester avant de passer au suivant.
- Dès que tu as trouvé 3 amis (ou 3 étrangers) c'est gagné ! Stoppe immédiatement la boucle quitte à utiliser l'instruction `break` et passe au graphe suivant.
- Tu peux seulement tester les graphes qui correspondent à  $p$  entre 0 et  $2^N/2$  (car pour les  $p$  suivants cela revient à échanger les segments verts en rouges et inversement).

Avec ces conseils voici les temps de calcul auxquels tu peux t'attendre :



Nombre de sommets	Nombre de graphes	Temps de calcul approximatif
$n = 6$	32 768	< 1 seconde
$n = 7$	2 097 152	< 1 minute
$n = 8$	268 435 456	< 1 heure
$n = 9$	68 719 476 736	< 10 jours

2. Il existe un énoncé plus difficile. Il s'agit de trouver à partir de quelle taille  $n$  un graphe contient toujours ou bien 4 amis ou bien 3 étrangers. Être 4 amis signifie que deux à deux ils sont reliés par un segment vert, comme ci-dessous :



- (a) Trouve des graphes à  $n = 6$  (puis  $n = 7$ ) sommets qui ne vérifient pas cet énoncé.
- (b) En cherchant un peu avec la machine trouve des graphes à 8 sommets qui ne vérifient pas cet énoncé.
- (c) Prouve que n'importe quel graphe ayant 9 sommets contient 4 amis ou bien 3 étrangers !
- Indications.* Il faut tester tous les graphes correspondants aux entiers  $p$  compris entre 0 et  $2^N = 2^{\frac{8 \times 9}{2}} = 68\,719\,476\,736$ . Le temps total de calcul est d'environ 20 jours ! Tu peux partager les calculs entre plusieurs ordinateurs : un ordinateur fait les calculs pour  $0 \leq p \leq 1\,000\,000$ , un deuxième ordinateur pour  $1\,000\,001 \leq p \leq 2\,000\,000$ ,...
3. • Il existe des raisonnements pour pouvoir démontrer à la main que pour  $n = 6$  il y a toujours 3 amis ou 3 étrangers. Cherche un tel raisonnement ! Avec un peu plus d'efforts, on prouve aussi que c'est  $n = 9$  qui répond au problème des 4 amis/3 étrangers.
- On sait prouver qu'il faut  $n = 18$  sommets pour avoir toujours 4 amis ou 4 étrangers.
- Par contre personne dans le monde ne sait quelle est la valeur du plus petit  $n$  pour le problème des 5 amis/5 étrangers !