Attention au loup !

Année 2024 – 2025

BALOGE Thomas P10, CHAPOULAUD Enson T08, DESIAUME Manon P10, ECHASSERIAU Dorian P10, ESSERS Maïa 2nd10, FLEURIET Vianney P06, MOUROUX--PINTO DOS SANTOS Léna T09 et POISSON Elliott 2nd10.

Établissement(s) : Lycée Marguerite de Navarre, Bourges.

Enseignant·e(s) : M.Créchet, Mme Herminier, M. Pelletier.

Chercheur·Chercheuse(s) : L.Besnier, X.Bultel, A.Lefebvre, B.Nguyen du LIFO de l'INSA Centre Val de Loire.

Sommaire

I. Présentation du sujet

II. Stratégie "je te coincerai"

A. Les configurations en ligne

B. Les configurations en boucle

C. Les configurations en croix

D. Le raisonnement par récurrence

III. Stratégie probabiliste

A. Formule de probabilités

B. Utilisation des matrices

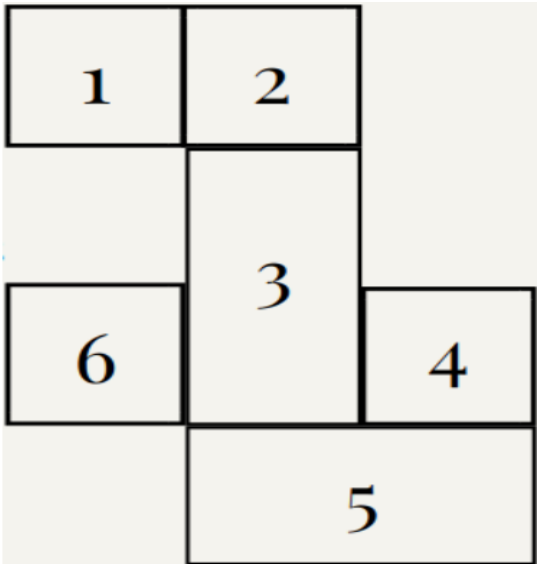
C. Modéliser et résoudre le problème en Python

IV. Comparaison des deux méthodes

I. Présentation du sujet

Problème :

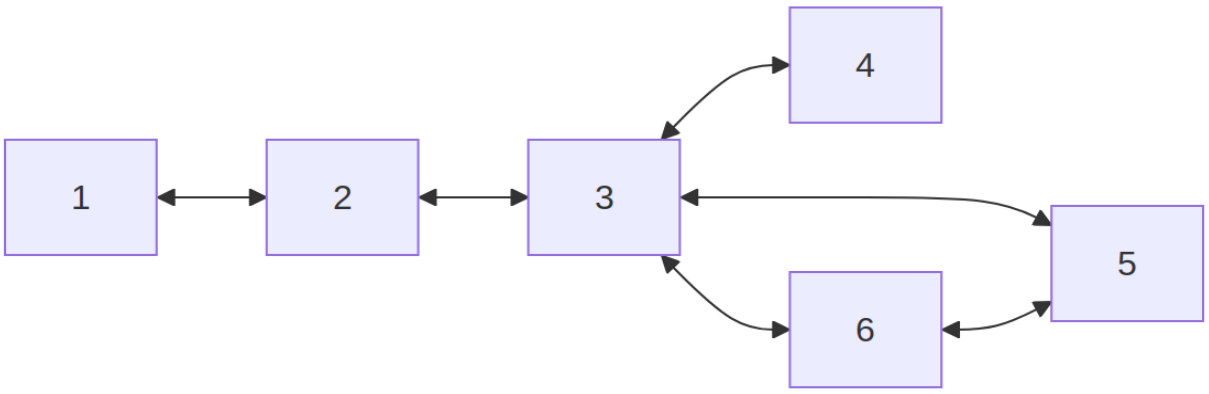
Un loup rôde dans les bois avoisinants et sème le chaos parmi les troupeaux de moutons. Un berger décide alors de le capturer afin de mettre fin au massacre des moutons. Les bois sont composées de plusieurs zones adjacentes. Chaque zone est ici numérotée de 1 à 6. Le premier jour, le loup choisit aléatoirement une zone et s'y cache. Chaque nuit, le loup se déplace dans une zone voisine. Chaque jour, le berger fouille une zone de son choix pour voir si le loup s'y cache et ainsi le capturer, puis rentre dans sa maison la nuit. Selon la configuration des bois, l'objectif est de discuter d'une stratégie efficace pour aider le berger à trouver le loup seul si possible ou avec un minimum d'aide (d'autres bergers) sinon.



Nous avons décidé de modéliser les bois en graphes pour simplifier la visualisation :

- Chaque nœud est une zone du bois ;
- deux zones voisines sont reliées par une arête (le loup peut "passer" d'une zone à une zone voisine durant la nuit).

Modélisation des bois du modèle de départ :

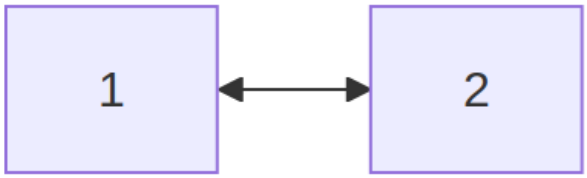


Nous avons pu diviser les bois en 3 types de configurations différentes : les configurations en ligne, celles en boucle et enfin celles en croix.

II. Stratégie "je te coincerai"

A. Les configurations en ligne

Nous avons commencé par une configuration avec 2 zones adjacentes.



Les déplacements du loup commencent lors de la 1ère nuit, le berger fouille quant à lui une 1ère zone le jour suivant, qu'on appellera le jour 1.

Nous avons constaté qu'en fouillant dans la zone 1 le 1er jour, 2 cas sont possibles :

- **le loup est sur cette zone, on l'a donc trouvé (1 chance sur 2)**
- **le loup n'est pas là.**

Durant la 2e nuit, le loup se retrouve obligé de se déplacer, par conséquent il nous suffit de revenir sur la zone 1 le jour 2 afin de trouver le loup à coup sûr.

Nous savons donc maintenant, que lorsqu'on utilise un seul berger pour une configuration de 2 zones, nous trouvons le loup soit en un jour, soit en 2 jours maximum.

En passant a une configuration de 3 zones alignées :

Nous décidons de commencer par fouiller la zone 2 :

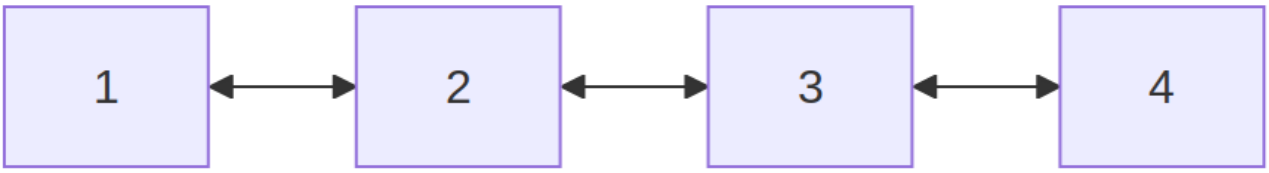
- **le loup est sur cette zone, on l'a donc trouvé (1 chance sur 3)**
- **le loup n'est pas là.**

La deuxième nuit le loup va forcément changer de zone et aller dans la zone 2. On utilise la même stratégie, on ne change pas de zone.

- **Le loup est là**

Nous savons donc maintenant, que lorsqu'on utilise un seul berger pour une configuration de 3 zones, nous trouvons le loup soit en un jour, soit en 2 jours maximum.

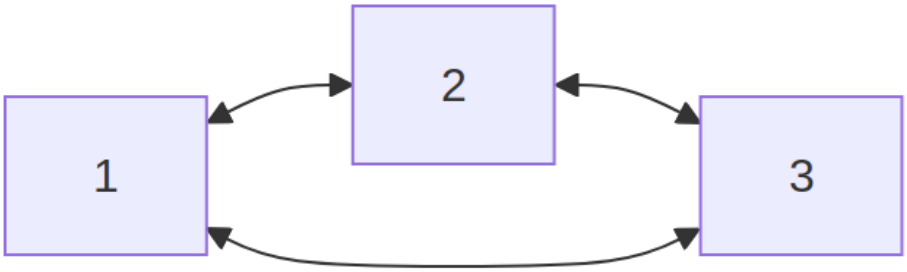
Passons maintenant à une configuration avec 4 zones en ligne :



Ici on ne peut pas appliquer la même stratégie que pour 3 zones puisqu'il n'y a pas de zone centrale, il faut donc trouver une autre stratégie. On commence par chercher dans la zone 2, on y reste une fois, puis on va

dans la zone 3 et on y reste pendant 2 jours, puis on retourne dans la zone 2 et on le trouvera forcément. De cette façon, [1]

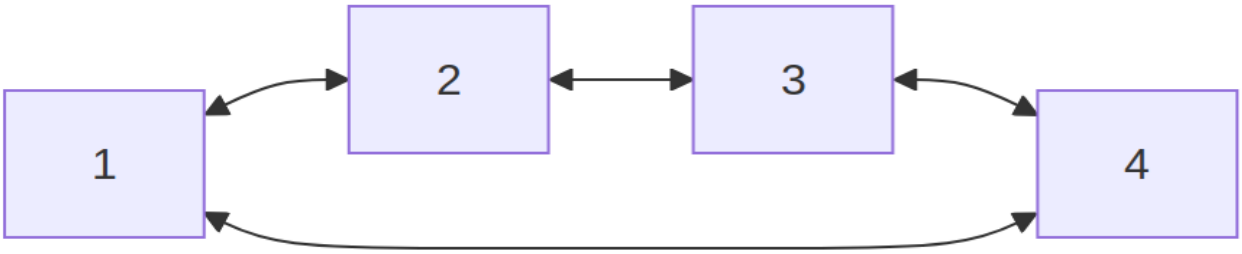
B. Les configurations en boucle



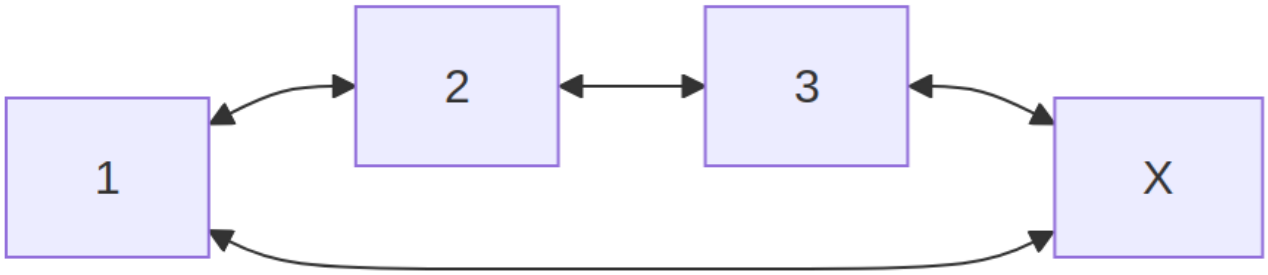
Il n'existe aucune bonne stratégie pour le bloquer car il y a toujours une chance sur trois de le trouver, le loup peut aller où il veut.

La solution serait de mettre 2 bergers et de rester les 2 fois de suite au même endroit. En effet il y a deux chances sur trois de le trouver la 1ère nuit et on le trouve forcément la 2ème nuit.

Si on rajoute une zone dans cette configuration :

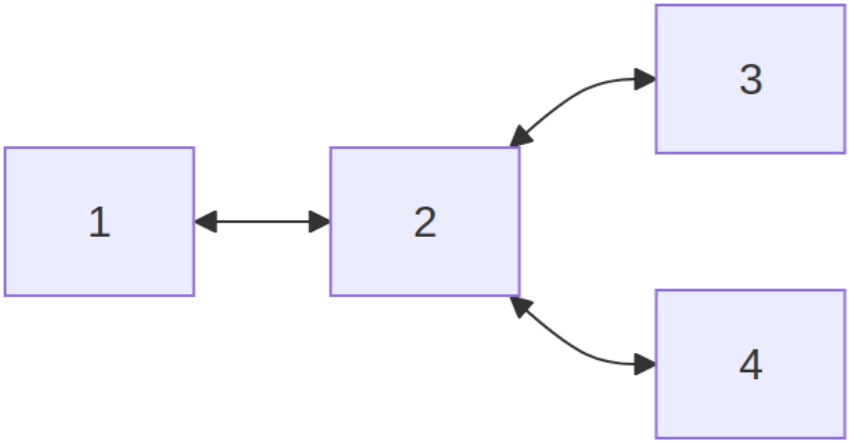


Pour n'importe quel cas en boucle avec plus de 3 zones, on applique la stratégie suivante, toujours avec 2 bergers : Le 1er berger se place sur une zone et y reste pendant tous les tours suivants. Ainsi, cela se rapporte à une configuration en ligne : le deuxième berger applique donc la stratégie pour les cas en ligne.



Ici, la croix représente le berger qui ne bouge pas, on retrouve donc bien une configuration en ligne avec 3 zones.

C. Les configurations en croix



On reste 2 fois en zone 2, pour le trouver forcément.

Nous n'avons pas eu le temps de réfléchir à une méthode pour des configurations en croix avec des branches de plus d'une zone chacune.

D. Le raisonnement par récurrence

Idée de stratégie :

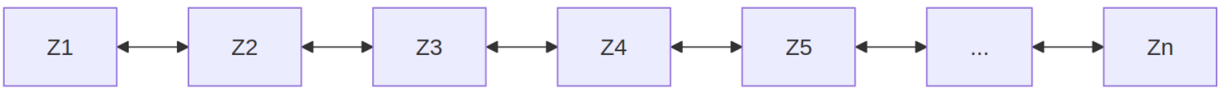
Pour une ligne quelconque de n zones :

- **on se place dans un premier temps sur la zone 2, puis sur la 3, ..., puis la zone en position n – 1 ;**
- **On se place à nouveau sur la zone en position n – 1, puis n – 2, ..., puis la zone 2.**
- **On reste à nouveau sur la zone 2.**

Il semblerait que cette stratégie semble fonctionner dans tous les cas. Nous allons le démontrer dans la suite.

Notre stratégie repose sur les éléments suivants:

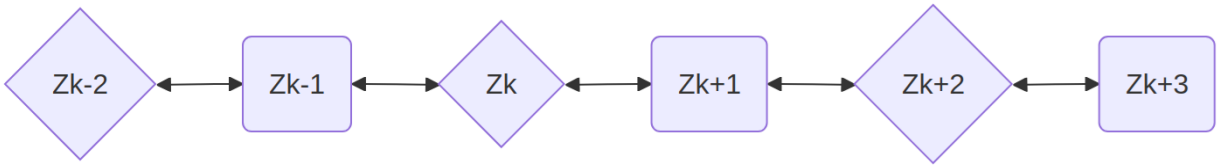
1. On numérote les zones de gauche à droite.
2. Pour commencer nos recherches, on se positionne dans la zone numéro 2 au jour 1. Le jour d'après le berger va fouiller la zone suivante, la zone numéro 3.



3. Si on est en Zk, sachant que k doit être compris entre 2 et n – 2, alors on avance en Zk+1
4. **On émet l'hypothèse suivante :**

a. **Si le berger fouille la zone Zk**

b. **La nuit passe et le loup se déplace, il peut se trouver en Zk, Zk – 2, Zk+2, …**



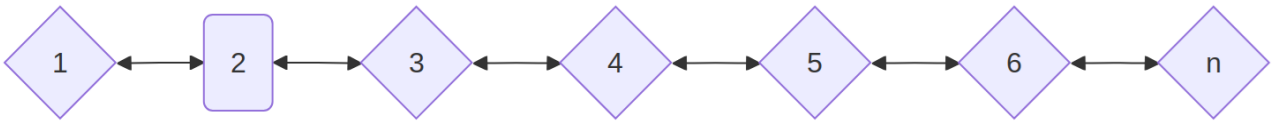
Nous définissons les losanges comme une zone dans laquelle le loup peut être présent et les rectangles une zone où nous sommes sûrs que le loup n'est pas présent.

C'est donc cette hypothèse que nous allons démontrer.

Démonstration pour n zones

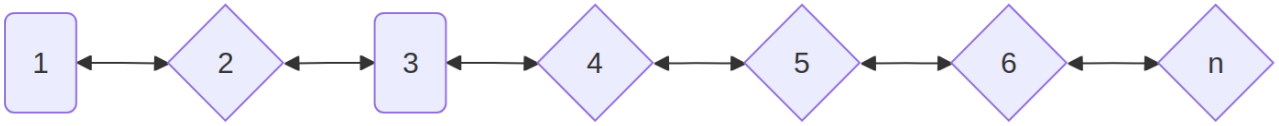
jour 1 :

Nous foullions la zone 2, le loup ne peut plus être dans cette zone.



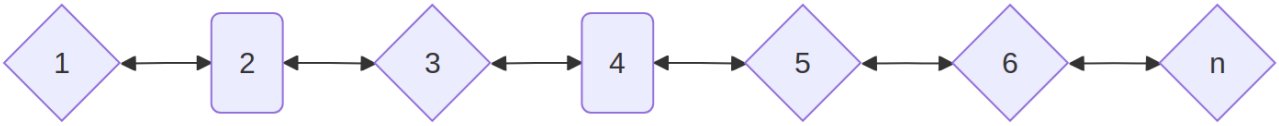
jour 2 :

Le loup s'est déplacé pendant la nuit et nous fouillons cette fois-ci la zone 3.



jour 3 :

Le loup s'est à nouveau déplacé et nous fouillons la zone 4.



Nous observons un modèle se créer, en effet derrière la dernière zone que le berger a fouillé, nous savons que le loup peut être derrière nous, derrière dans la zone 2 nous sommes sûr que le loup ne peut pas être là et dans la zone 1 nous savons que le loup peut aussi être là. Mais nous n'avons aucune information dans les zones de 5 à n, nous estimons donc que le loup peut également se trouver dans ces zones là.

Hypothèses

Hypothèse 1 :

Si le jour K, le berger fouille la zone K+1, alors au début du jour K+1 nous savons que :

- le loup peut être dans la zone K

- Il n'y a pas de loup dans la zone K – 1

- le loup peut être dans la zone K – 2

- Il n'y a pas de loup dans la zone K – 3

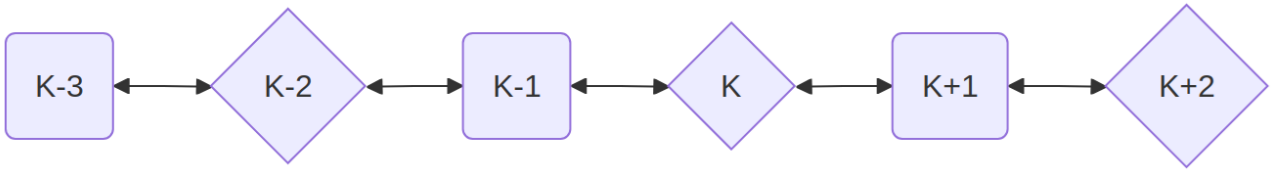
Et ainsi de suite...

- le loup peut être dans les zones avec un numéro supérieur à K

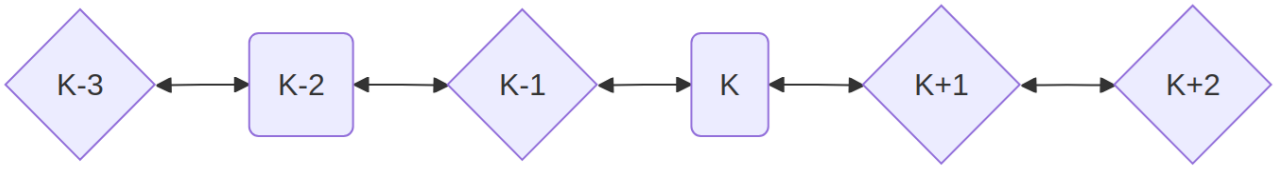
Initialisation :
On vérifie facilement que notre hypothèse est vraie lorsque k = 1 :

- le berger fouille la zone 2 ;
- Il y a potentiellement un loup dans la zone 1.

Hérédité :
Nous supposons que l'on est au jour K avec notre hypothèse nous allons trouver :



Au jour K+1 nous avons :



Notre hypothèse est donc vérifiée au rang k + 1. Ainsi notre hypothèse est vraie quelle que soit la valeur de k ($2 \leq k \leq n - 1$). [2]

Hypothèse 2 :
Une fois les déplacements précédents réalisés, le berger va se déplacer de la zone n−1 à la zone 2. Notre hypothèse est alors la suivante : une fois le berger sur la zone k (avec $2 \leq k \leq n - 1$), il n'y aura aucune chance que le loup se trouve sur l'une des cases à droite du berger (en positions k+1, k+2, ..., n).

Cette hypothèse se démontre également à l'aide d'un raisonnement par récurrence comme nous l'avons fait pour l'hypothèse 1.

Enfin, en restant à nouveau sur la zone 2, nous garantissons la capture du loup (s'il était en zone 1 le jour précédent).

III. La stratégie probabiliste.

A. Formule de probabilités

Au jour 1, on cherche la probabilité P_{loup}^{i} de trouver le loup sur la case i. On pose N le nombre total de cases.

On a alors que $P_{loup}^{i} = \frac{1}{N}$ lors du premier jour.

Attention ! Lorsqu'une zone i est fouillée, la probabilité que le loup se trouve dans cette zone tombe à 0.

Au jour 2, on cherche ensuite la probabilité $P_{i,j}$ que le loup se déplace d'une case i à une case j voisine, qui peut se calculer avec P_{loup}^{i} et N_i qui est le nombre de voisins de la case i. En effet, le déplacement du loup étant considéré comme aléatoire, on a une situation d'équiprobabilité. On obtient donc :

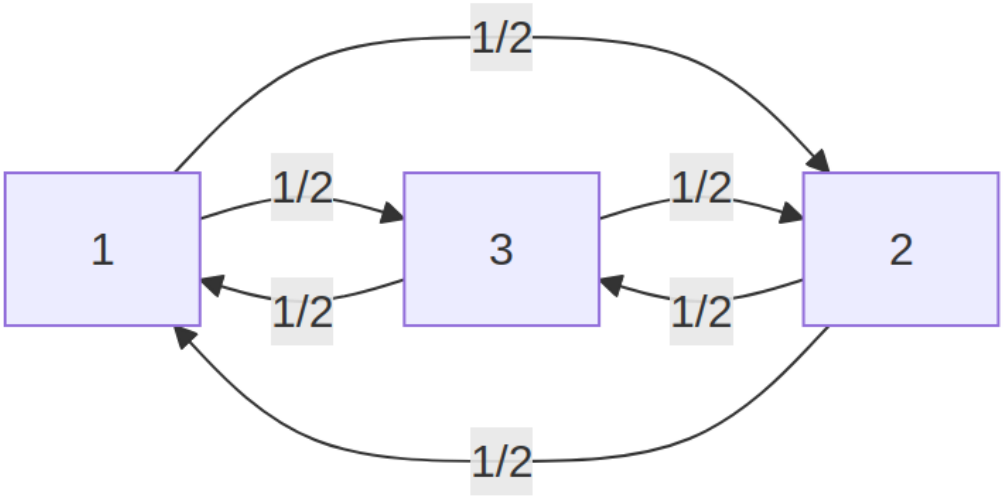
$$P_{i,j} = \frac{P_{loup}^{i}}{N_i}$$

On peut ensuite déterminer la nouvelle probabilité de trouver le loup au jour 2 sur une case i en réalisant la somme des Pj,i où les j sont les différents voisins de i.

$$P_{loup}^{i} = \sum P_{j,i}$$

[3]

Prenons maintenant un exemple plus concret pour montrer comment on peut utiliser ces formules. On va donc étudier une configuration en boucle fermée à trois cases avec 1 seul berger. Nous avons donc un graphe en cycle où chaque case est connectée à deux cases voisines.



Au jour 1, on calcule (par exemple) la probabilité de trouver le loup sur la case 1 :

$$P_{loup}^{1} = \frac{1}{3}$$

[4]

Au jour 2 :

$$P^1_{loup} = P_{2,1} + P_{3,1} = 2 \times \frac{\frac{1}{3}}{2} = \frac{1}{3}$$

Pour les deux autres zones, il faut considérer que $P^1_{loup} = 0$ (si on ne trouve pas le loup en case 1).

$$P^2_{loup} = P_{1,2} + P_{3,2} = \frac{1}{6}$$

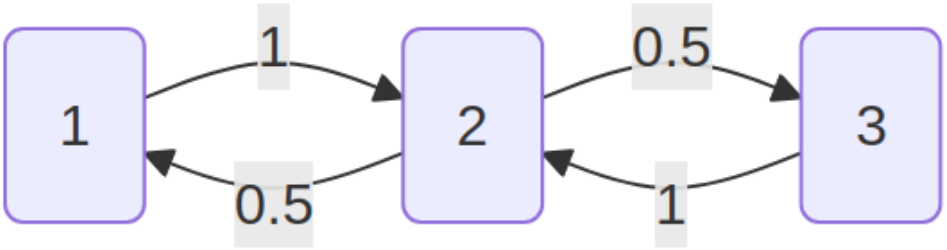
Même résultat pour P^3_{loup}
Intuitivement, il semble logique de rester sur la zone de plus grande probabilité (la case 1 donc).
Au jour 3, les calculs donnent :

$$P^1_{loup} = \frac{1}{6} \text{ et } P^2_{loup} = P^3_{loup} = \frac{1}{12}$$

Etc. [5]

B. Utilisation des matrices

- **Notion de graphe probabiliste et pondéré**
  Pour modéliser notre forêt, nous allons utiliser un graphe pondéré comme précédemment. Voici un exemple :



Une flèche correspond au passage d'une zone à une autre et le nombre correspond à la probabilité de ce déplacement. Ici, si le loup est sur la case 1 il ne peut aller que sur la case 2 donc la probabilité est de 1. S'il est sur la case 2 il peut aller avec les mêmes probabilités sur les cases 1 et 3, donc la probabilité est de 0.5. Pour la case 3 c'est la même chose que pour la case 1 donc la probabilité est de 1.
On a en fait représenté la configuration d'une forêt avec 3 zones alignées.

- **Matrice d'adjacence d'un graphe**

Pour obtenir des données plus claires à propos de nos probabilités nous utilisons des matrices d'adjacence qui répertorient avec des lignes et des colonnes numérotées dans l'ordre croissant nos probabilités.

Dans notre cas on a :

$$T = \begin{pmatrix} 0 & 1 & 0 \\ 0{,}5 & 0 & 0{,}5 \\ 0 & 1 & 0 \end{pmatrix}$$

Ici le coefficient $T_{1,2}$ (coefficient de la ligne 1 et de la colonne 2) correspond à la probabilité de passer de la case 1 à la case 2, et ainsi de suite pour tous les coefficients. Cette matrice est invariable pour cette forêt et nous permet de réaliser des calculs de probabilités pour déterminer où placer le chasseur. On peut utiliser cette matrice pour obtenir toutes les probabilités futures.

Après, nous devons créer la matrice des probabilités P_0 au jour 0. A ce moment précis, le loup a les mêmes probabilités d'être sur chaque case.

Soit N le nombre de cases : $P_0 = \left(\frac{1}{n}\frac{1}{n}\dots\right)$ et P_0^i (valeur en position i) la probabilité qu'il soit sur une case "i" au jour 0.
Par exemple pour notre forêt précédente, on a :

$$P_0 = \left(\frac{1}{3}\ \frac{1}{3}\ \frac{1}{3}\right)$$

En multipliant les matrices : $P_{n+1}=P_n'\times T$

En effectuant le calcul, un trouve P_1 :

$$P_1 = \left(\frac{1}{6}\ \frac{2}{3}\ \frac{1}{6}\right)$$

On trouve la probabilité que le loup soit sur la case i après la nuit n. On peut répéter cette opération plusieurs jours et observer l'évolution des probabilités et des cases les plus choisies par le loup dans le cas d'un déplacement aléatoire. Avec ces résultats nous pouvons maximiser nos chances de trouver le loup en plaçant chaque jour le berger sur la case ayant la plus grande probabilité. Comme ces calculs sont très longs, nous avons réalisé un programme qui réalise ces calculs et ces placements.

Attention cependant, il est important de "mettre à jour" la probabilité de la case i fouillée un jour n.
Ainsi, dans notre exemple précédent, après avoir obtenu P_1, nous devons réaliser le calcul suivant avec $P_1' = \left(\frac{1}{6}\ 0\ \frac{1}{6}\right)$ et donc on aura $P_2=P_1'\times T$.

C. Modéliser et résoudre le problème en python

Afin de résoudre ce problème,on peut utiliser les techniques développées précédemment pour les coder et résoudre n'importe quelle version de notre sujet.

Tout d'abord, il faut importer le module python 'math' pour avoir à disposition un certain nombre d'outils mathématiques.

Ensuite, on va représenter la configuration de notre problème sous la forme d'une matrice d'adjacence, comme utilisée lors de la présentation de la stratégie probabiliste. Elles sont implémentées sous la forme de listes de listes et chaque valeurs est inférieur ou égale à 1.

```
from math import *

T1bis = [[0,0.5,0.5,0],
         [1/3,0,1/3,1/3],
         [0.5,0.5,0,0],
         [0,1,0,0]]

T2bis = [[0,1,0,0,0,0,0],
         [0.5,0,0.5,0,0,0,0],
         [0,0.5,0,0.5,0,0,0],
         [0,0,0.5,0,0.5,0,0],
         [0,0,0,0.5,0,0.5,0],
         [0,0,0,0,0.5,0,0.5],
         [0,0,0,0,0,1,0]]

T3bis = [[0,0.5,0,0,0,0.5],
         [0.25,0,0.25,0.25,0.25,0],
         [0,1,0,0,0,0],
         [0,1/3,0,0,1/3,1/3],
         [0,1/3,0,1/3,0,1/3],
         [1/3,0,0,1/3,1/3,0]]

T4bis = [[0,1,0,0,0],
         [1/3,0,1/3,1/3,0],
         [0,1,0,0,0],
         [0,0.5,0,0.5,0],
         [0,0,0,1,0]]

T5bis = [[0,1,0,0,0,0],
         [0.5,0,0,0.5,0,0],
         [0,0,0,1,0,0],
         [0,0.25,0.25,0,0.25,0.25],
         [0,0,0,0.5,0,0.5],
         [0,0,0,0.5,0.5,0]]
```

Après avoir défini les exemples à tester, on commence à écrire la fonction de résolution res prenant en paramètre une matrice représentant un exemple du problème.

Au sein de la fonction, on commence par définir trois variables :

- perc, un nombre décimal "float" compris entre 0 et 1 représentant la probabilité de trouver le loup ;
- jours, un entier "int" qui représente le nombre de jours passés ;
- valList, une liste "list" qui représente la matrice des probabilités, commençant avec les valeur $\frac{1}{N}$.

```
def res(matrice):
    perc = 1
    jours = 0
    valList = len(matrice) * [1/len(matrice)]
```

Le code commence par réaliser une première itération de la fonction, soit le jour 0 et nuit 0 :

- on retourne la meilleure probabilité au jour 0 (soit $\frac{1}{N}$) et la zone correspondante (l'indice de la plus grande valeur de la matrice)
- on retourne la matrice des probabilités elle-même pour montrer le résultat à la nuit 0
- on redéfinit les trois variables :
    - la valeur de valList qui a été choisie comme meilleure probabilité est mise à 0
    - à la variable perc, on affecte la somme des valeurs de la matrice (on enlève la meilleure probabilité)
    - on augmente de 1 jour
- enfin on retourne la probabilité de réussite au jour 0

```
print(("Meilleur probabilité : "+str(max(valList)),"Meilleur choix : "+str((valList.index(max(valList)))+1)))
    print(valList)
    valList[valList.index(max(valList))] = 0
    perc = sum(valList)
    jours += 1
    print(str(round((1-perc)*100)) + " % au jour " + str(jours))
```

Enfin, on répète ce procédé jusqu'à un seuil donné (dans ce cas, jusqu'à avoir 99,9 % de chance de trouver le loup) :

- on crée une liste temporaire ayant pour valeur le produit de la matrice d'adjacence et celle des probabilités, elle nous servira de nouvelle base pour appliquer la méthode précédente.

```
while perc > 0.01 :
    valList2 = len(matrice) * [0]
    for l in range(len(matrice)):
        valTemp = 0
        for e in range(len(matrice)):
            valTemp += matrice[e][l]*valList[e]
        valList2[l] = round(valTemp, 5)
    valList = valList2
    print(("Meilleur probabilité : "+str(max(valList)),"Meilleur choix : "+str((valList.index(max(valList)))+1)))
    print(valList)
    valList[valList.index(max(valList))] = 0
    perc = sum(valList)
    jours += 1
    print(str(round((1-perc)*100)) + " % au jour " + str(jours))
```

Après avoir calculé jusqu'au seuil, on obtient une liste d'itérations de chaque jour avec toutes les informations nécessaires. À la fin, on peut estimer un nombre de jours pour trouver le loup avec une chance de 99,9% grâce à la méthode probabiliste.

```
print("Le loup aura " + str(round((1-sum(valList))*100)) + " % de chance d'être trouvé après "+str(jours)+" "+"jours.")
```

IV. Comparaison des deux méthodes

Avec la stratégie probabiliste :

Dans le cas de la configuration originale, on trouve le loup en 7 jours avec plus de 99% de chance.

Dans la configuration de base, il faudrait rester sur cet enchaînement de cases : 3-3-3-3-2-2-3

Avantages :

- On peut l'appliquer dans toutes les configurations ;
- On maximise la probabilité de trouver le loup ;
- La probabilité de trouver le loup augmente rapidement.

Inconvénients :

- Aucune garantie de le trouver à coup sûr, surtout lors de la présence d'une boucle dans la configuration.

Avec la stratégie "je te coincerai" :

Dans le cas de la configuration originale, on trouve le loup à coup sûr en 4 jours avec 2 bergers.

Le premier reste sur les cases 3-2-2-3

Le second reste sur la case 5

Avantages :

- Garantie de trouver le loup à coup sûr ;
- Utile dans des configurations connues et définies.

Inconvénients :

- On ne peut pas l'utiliser dans toutes les configurations ;
- Elle peut nécessiter plusieurs bergers.

NOTES DE L'ÉDITION

[1] La phrase est incomplète. On peut toutefois facilement en imaginer le contenu. Il faudrait peut-être observer que des "échanges de zone" sont possibles. Par exemple, si un jour le berger se trouve dans la zone 2 et le loup dans la zone 3, puisque pendant la nuit le berger rentre chez lui, il est possible que le lendemain le loup soit dans la zone 2 et le berger dans la zone 3.

[2] Le losange le plus à droite dans la figure ci-dessus est un rectangle. La vérification de l'hypothèse pour K+1 est assez intuitive. On aurait pu observer, pour la rendre plus claire, que chaque jour les suites de losanges et de rectangles alternés sont transformées, en raison du passage possible du loup, en suites du même type, mais avec des rectangles à la place des losanges et vice versa.

[3] Dans les expressions de probabilité, il aurait été opportun d'indiquer explicitement aussi le jour. Il aurait donc fallu écrire $P^{i}_{\text{loup}}(k)$pour désigner la probabilité que le loup se trouve dans la zone $i$le jour k. De cette manière, la formule $P^{i}_{\text{loup}} = \sum P_{j,i}$ devient $P^{i}_{\text{loup}}(k+1) = \sum P_{j,i}(k)$.

[4] On pouvait observer ici que nous avions aussi $P^{2}_{loup} = P^{3}_{loup} = \frac{1}{3}$. Et, par la suite, comment peut-il arriver que nous obtenions $P^{1}_{loup} = 0$?

[5] Cette partie sur la probabilité mériterait un traitement plus détaillé.