Renard y es-tu ?

François Dessenne, Jean Fromentin, Denis Vekemans

Renard y es-tu ?

François Dessenne[1] Jean Fromentin[2] Denis Vekemans[2]

Une variation autour du problème du renard et des terriers

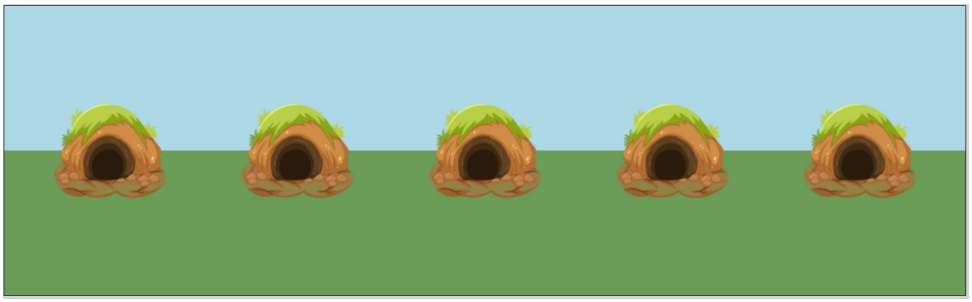
Résumé

Nous nous intéressons à l'énigme relativement connue "Fox and Holes". Nous envisageons certaines généralisations de cette énigme, que nous résolvons. Nous avons tenu à ce que cet article puisse être apprécié et compris par un public ayant reçu un enseignement secondaire.

Introduction

Voici un petit problème de logique tel que l'a envoyé *Rabah Bouhallel*, un ami de longue date.

Problème "Fox and Holes"



*Un renard, Fox, est dans l'un des **5 terriers**.*
*Chaque nuit, **Fox passe dans un terrier voisin de celui qu'il occupe**.*
Chaque jour, nous pouvons regarder dans l'un des terriers pour voir si Fox y est.
Donner une stratégie pour trouver Fox.

Appropriation, modélisation

Remarque à propos des déplacements du renard : chaque nuit, Fox passe *obligatoirement* dans un terrier *strictement* voisin de celui occupé la nuit précédente, et les premier et dernier terriers ne sont pas voisins.

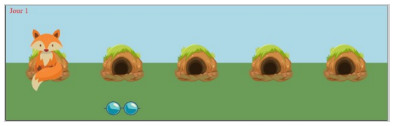
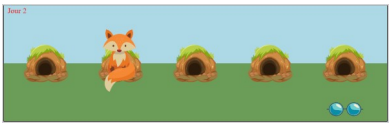
Levons d'emblée un point qui a fait débat : pour *Rabah*, "trouver Fox" désigne le fait de pouvoir identifier le terrier dans lequel il se trouve à l'issue d'une observation, même s'il ne se trouve pas dans le terrier observé. Nous, qui rédigeons cet article, avons choisi d'exiger que "trouver Fox" signifie trouver une stratégie qui nous assure de débusquer Fox dans son terrier.

1 Université de Lille ; Cité Scientifique ; Institut d'électronique de microélectronique et de nanotechnologie ; Avenue Henri Poincaré, CS 60069 ; 59 652 Villeneuve d'Ascq cedex ; France
2 Université du Littoral Côte d'Opale ; Centre Universitaire de la Mi-Voix ; Laboratoire de mathématiques pures et appliquées Joseph Liouville ; 50, rue Ferdinand Buisson, BP 699 ; 62 228 Calais cedex ; France

Une petite APPLICATION avec son mode d'emploi nous accompagnera tout au long de l'article et nous aidera dans notre réflexion ...

Pour modéliser le problème, nous numérotons les terriers de 1 à 5. Ensuite, pour résumer une stratégie (pas nécessairement qui permette de trouver Fox) nous écrivons une suite comme "2, 5, 4" qui se lit "deux, cinq, quatre" et qui résume "d'abord, nous regardons dans le terrier 2, puis dans le 5 et enfin dans le 4". Plus généralement, nous donnons la suite des terriers observés.

Comme le montre l'exemple ci-dessous, la stratégie "2, 5, 4" ne permet pas de trouver **à coup sûr** Fox :

		
Jour 1 : Si Fox est dans le terrier 1, on regarde dans le terrier 2 et Fox ne s'y trouve pas.	Jour 2 : Si Fox passe ensuite dans le terrier 2, on regarde dans le terrier 5 et Fox ne s'y trouve pas non plus.	Jour 3 : Si Fox passe enfin dans le terrier 3, on regarde dans le terrier 4 et Fox ne s'y trouve toujours pas.

Le fait de trouver un seul exemple de positions successives du renard dans lequel ce dernier se soustrait à notre regard suffit à prouver qu'une stratégie n'est pas gagnante, elle ne garantit pas de trouver Fox.

Invitation à chercher une solution

Ce problème peut sembler décourageant car

- nous n'avons aucune idée de la longueur de la stratégie cherchée ;
- pour tester une stratégie, nous avons de nombreuses éventualités à évaluer.

Ci-dessous, une aide un peu plus conséquente à la résolution du problème. Ne poursuivez pas votre lecture si vous préférez chercher seul une solution.

Pour amoindrir les effets des points précédemment cités pouvant mener à un certain découragement, voici quelques indices pour trouver une solution au problème.

- Sachez qu'il nous est possible de trouver une stratégie pour trouver Fox en six observations.
- Pour tester une stratégie, au lieu de regarder tous les cas, un à un, on peut aussi regrouper ces cas (par exemple, selon la parité du terrier dans lequel Fox se trouve) et / ou limiter ces cas (par exemple, en utilisant de la symétrie).

Ci-après, une solution du problème est donnée. Ne poursuivez pas votre lecture si vous préférez chercher seul une solution.

Stratégie pour trouver Fox

Premier temps : si Fox est à l'origine dans un terrier dont le numéro est pair

Fox est donc soit en 2, soit en 4.
Quand nous regardons en 2, si Fox s'y trouve, c'est terminé, sinon, c'est qu'il était en 4.
Nous allons alors l'empêcher de revenir vers 2 en regardant ensuite dans le terrier 3, si Fox s'y trouve, c'est terminé, sinon, c'est qu'il est passé du 4 dans le 5.
Cependant, du terrier 5, la seule possibilité qu'il a est d'aller dans le terrier 4 et en regardant enfin dans le terrier 4, on le trouve.

Conclusion Dans ce premier temps, la suite "2, 3, 4" est une stratégie qui permet de trouver Fox.

Remarque De façon analogue, par symétrie, la suite "4, 3, 2" aussi.

Second temps : si Fox est à l'origine dans un terrier dont le numéro est impair

Fox est donc soit en 1, soit en 3, soit en 5.

On applique la stratégie "2, 3, 4" ou "4, 3, 2". À l'issue, Fox s'est déplacé d'un nombre impair de terriers (par trois fois d'un terrier) ce qui nous permet de conclure que, maintenant, Fox se trouve bel et bien dans un terrier dont le numéro est pair. Nous sommes donc ramenés dans le premier temps où nous trouvons Fox par "2, 3, 4" ou par "4, 3, 2".

Conclusion Dans ce deuxième temps, faire suivre "2, 3, 4" ou "4, 3, 2" de "2, 3, 4" ou "4, 3, 2" permet de trouver Fox.

Conclusion globale

Nous avons quatre stratégies pour trouver Fox, à savoir "2, 3, 4, 2, 3, 4", "2, 3, 4, 4, 3, 2", "4, 3, 2, 2, 3, 4" et "4, 3, 2, 4, 3, 2".

Des stratégies plus longues sont correctes aussi : on peut insérer n'importe quelle suite avant ou après l'une de ces quatre suites ; on peut même insérer n'importe quelle suite d'un nombre pair de termes au "milieu" de n'importe laquelle de ces quatre suites ; on peut remplacer "2, 3, 4" par "2, 3, 3, 3, 4" ; etc.

Cependant, trouver une stratégie plus courte est impossible. Malheureusement, nous n'avons pas de "jolie démonstration" de ceci et n'en avons qu'une preuve dans laquelle nous regardons toutes les stratégies en cinq observations (il s'agit d'un algorithme "glouton" ; il existe 5^5 stratégies possibles *a priori* : nous faisons 5 observations parmi 5 terriers possibles) et nous concluons que chacune laisse à Fox une possibilité de se soustraire à notre regard.

Preuve par Maple 5.0 → Annexe A

Notons que nous avons validé quatre stratégies pour trouver Fox en six observations, mais nous n'avons pas prouvé qu'il n'en existe pas d'autre. Là encore, point de "jolie démonstration" et nous n'avons qu'une preuve dans laquelle nous regardons toutes les stratégies en six observations (il s'agit là encore d'un algorithme "glouton" ; il existe 5^6 stratégies possibles *a priori* : nous faisons 6 observations parmi 5 terriers possibles) et nous concluons que seules les quatre stratégies données sont valables pour trouver Fox.

Preuve par Maple 5.0 → Annexe A

On peut trouver une solution vidéo sur [LY,MA].

Remarque Pour ce problème "Fox and Holes", nous nous sommes efforcés de justifier qu'il n'existe pas de stratégie plus courte que celles proposées et aussi de déterminer toutes les stratégies les plus courtes.

Quelques remarques sur la solution et les algorithmes gloutons précédemment évoqués

L'algorithme glouton, par conception, est loin d'être économique. Par exemple, quand pour la première observation on regarde en 2, il envisage aussi que pour la seconde observation, on regarde en 1, ce qui est complètement idiot : en effet, ayant observé en 2 lors de la première observation,

- s'il y est, c'est terminé et il n'est pas nécessaire de poursuivre,
- s'il n'y est pas, il ne sera assurément pas en 1 lors de la deuxième observation (en effet, il ne peut se rendre en 1 que s'il était en 2 précédemment) et la deuxième observation est donc stérile.

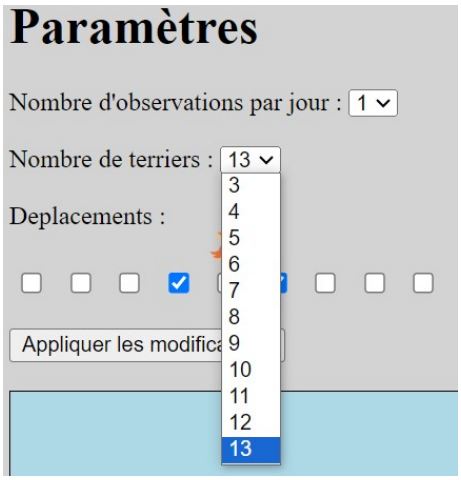
Observons maintenant de plus près la solution "2, 3, 4, 2, 3, 4" et ce que cette stratégie opère sur les possibles positions du renard :

Le renard est initialement en l'un des terriers 1, 2, 3, 4 ou 5.
Jour 1 Quand on observe en 2, si on n'a pas trouvé le renard, il est en l'un des terriers 1, 3, 4 ou 5.
Nuit 1 Le renard se déplace pour aller dans l'un des terriers 2, 3, 4 ou 5.
Jour 2 Quand on observe en 3, si on n'a pas trouvé le renard, il est en l'un des terriers 2, 4 ou 5.
Nuit 2 Le renard se déplace pour aller dans l'un des terriers 1, 3, 4 ou 5.
Jour 3 Quand on observe en 4, si on n'a pas trouvé le renard, il est en l'un des terriers 1, 3 ou 5.
Nuit 3 Le renard se déplace pour aller dans l'un des terriers 2 ou 4.
Jour 4 Quand on observe en 2, si on n'a pas trouvé le renard, il est en le terrier 4.
Nuit 4 Le renard se déplace pour aller dans l'un des terriers 3 ou 5.
Jour 5 Quand on observe en 3, si on n'a pas trouvé le renard, il est en le terrier 5.
Nuit 5 Le renard se déplace pour aller en le terrier 4.
Jour 6 Quand on observe en 4, on le trouve forcément !

De cette observation, on pourrait remplacer l'algorithme glouton par un algorithme qui ne considère que les positions possibles du renard !

Cependant, si on décidait de ne considérer que des observations telles que le cardinal des positions possibles du renard décroisse de nuit en nuit, cela tiendrait d'une heuristique car il n'a jamais été démontré que cette suite devait forcément être décroissante (même si elle décroît effectivement dans l'exemple considéré, mais pas strictement).

Une première extension de l'énigme, avec *n* terriers



n terriers alignés ($n \geq 2$).

APPLICATION

mode d'emploi

*Un renard, Fox, est dans l'un des **n terriers**.*
Chaque nuit, Fox passe dans un terrier voisin de celui qu'il occupe.
Chaque jour, nous pouvons regarder dans l'un des terriers pour voir si Fox y est.
Donner une stratégie pour trouver Fox.

Ci-dessous, une solution du problème est donnée. Ne poursuivez pas votre lecture si vous préférez chercher seul une solution.

Une stratégie

Notre stratégie se démontre de la même façon que pour le problème "Fox and Holes" (à 5 terriers), en deux temps.

Dans un premier temps, nous traitons l'éventualité où Fox débute d'un terrier pair en commençant par le terrier 2 et nous empêchons Fox de se rapprocher des terriers déjà fouillés tout en progressant sur la file des terriers.

Dans un second temps si Fox avait commencé dans un terrier impair, nous nous ramenons au premier temps. Pour se ramener du deuxième temps au premier, il faut que le nombre de terriers visités auparavant soit impair, on complète donc par autant d'observations farfelues que nécessaire.

Si n est impair, on peut poser $n = 2k + 1$ avec k entier naturel non nul. La stratégie "2, 3, 4, 5, ..., 2*k* - 1, 2k, 2, 3, 4, 5, ..., 2*k* - 1, 2*k*" permet de trouver Fox.

Si n est pair, on peut poser n=2k avec k entier naturel supérieur ou égal à 1. La stratégie "2, 3, 4, 5, ..., 2*k* - 2, 2*k* - 1, *x*, 2, 3, 4, 5, ..., 2*k* - 2, 2*k* - 1" où *x* est un nombre quelconque permet de trouver Fox. L'ajout du "*x*" permet de se ramener au premier temps.

Remarque Pour le problème "Fox and Holes" à *n* terriers, nous nous sommes contentés de fournir une seule stratégie valide, mais nous n'avons aucune preuve que cette solution soit la plus courte possible et encore moins pu donner l'ensemble des stratégies de longueur minimale.

Discussion sur notre stratégie

Indépendamment de la parité de *n*, on peut trouver une autre solution dans le même contexte sur [MSE] ou sur [GTJ] : "2, 3, 4, 5, ..., *n* - 2, *n* - 1, *n* - 1, *n* - 2, ..., 5, 4, 3, 2". Cette solution est

différente de celle proposée dans cette section, et est même plus courte dans le cas où *n* est pair. La validité de cette solution ne se justifie pas de la même façon : le premier temps des deux stratégies est identique et se justifie d'égale façon, mais si dans notre deuxième temps, nous avions eu pour objectif de nous ramener au premier en chassant à nouveau le renard des terriers pairs, le deuxième temps sur [MSE] ou sur [GTJ] chasse le renard des terriers impairs dans le cas où *n* est pair (en commençant par "*n* - 1, ...") et chasse autrement le renard des terriers pairs dans le cas où *n* est impair (en commençant par "*n* - 1, *n* - 2, ..." et non en reprenant de "2, 3, ...").

On peut aussi trouver une solution dans le cas où *n*=17 dans un autre contexte sur [PSE]. La démarche est analogue à celle sur [MSE] ou sur [GTJ].

Pourquoi conserver notre solution qui est moins performante que celle trouvée sur sur [MSE] ou sur [GTJ] ? La raison principale est que nous ne nous intéressons qu'à l'existence d'une solution sans forcément nous confronter à l'exigence d'optimalité, c'est-à-dire sans s'efforcer à produire une solution la plus courte possible. Notre solution se généralise, tout comme celle sur sur [MSE] ou sur [GTJ], mais nous préférons l'idée de se ramener dans chaque temps au premier à celle de traiter dans chaque temps des éventualités différentes qui deviendront finalement exhaustives.

<u>Remarque</u> Avant cette remarque, les problèmes présentés ont déjà été proposés par d'autres auteurs, mais il ne semble pas que cela soit le cas pour les suivants.

Une seconde extension de l'énigme, avec de nouveaux déplacements du renard

APPLICATION

mode d'emploi

> *Un renard, Fox, est dans l'un des* ***6 terriers****.*
> *Chaque nuit,* ***Fox passe du terrier occupé à l'un des deux terriers parmi celui juste à gauche et celui juste à droite de celui juste à droite****.*
> *Chaque jour, nous pouvons regarder dans l'un des terriers pour voir si Fox y est.*
> *Donner une stratégie pour trouver Fox.*

Il vous est encore possible de chercher une stratégie en arrêtant ici votre lecture. Ci-dessous, une aide légère à la résolution du problème.

Modélisation

Pour modéliser le problème, nous pouvons toujours numéroter les terriers, mais cette fois de 1 à 6. Et, pour tenir compte de la distinction entre le déplacement vers la gauche et celui vers la droite, nous allons convenir que le terrier 1 est "tout à gauche", que le terrier 6 est "tout à droite", et ainsi que "celui juste à gauche" signifie que Fox a diminué de 1 le numéro du terrier dans lequel il se trouve et que "celui juste à droite de celui juste à droite" signifie que Fox a augmenté de 2 le numéro du terrier dans lequel il se trouve.

Ci-dessous, une solution du problème est donnée. Ne poursuivez pas votre lecture si vous préférez chercher seul une solution.

Une stratégie

Premier temps Si Fox est à l'origine dans un terrier dont le numéro est un multiple de 3

Ainsi, Fox est soit en 3, soit en 6.

Quand nous regardons en 3, si Fox s'y trouve, c'est terminé, sinon, c'est qu'il était en 6.

Cependant, du terrier 6, la seule possibilité qu'il a est d'aller dans le terrier 5 et en regardant enfin dans le terrier 5, on le trouve.

Conclusion Dans ce premier temps, la suite "3, 5" est une stratégie qui permet de trouver Fox.

Deuxième temps Si Fox est à l'origine dans un terrier dont le numéro est le prédécesseur d'un multiple de 3

Ainsi, Fox est, à l'origine, soit en 2, soit en 5.

Quand nous avons déjà procédé par "3, 5", Fox s'est déplacé d'abord sur le successeur d'un multiple de 3, puis sur un multiple de 3. Nous sommes donc ramenés dans le premier cas où nous trouvons Fox par "3, 5".

Conclusion Dans ce deuxième temps, faire suivre "3, 5" de "3, 5" permet de trouver Fox.

Troisième temps Si Fox est à l'origine dans un terrier dont le numéro est le successeur d'un multiple de 3

Ainsi, Fox est, à l'origine, soit en 1, soit en 4.

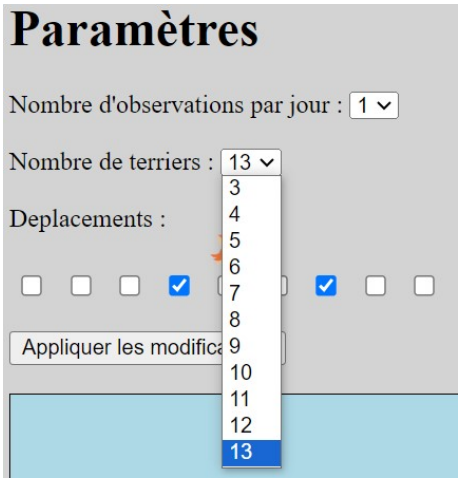
Quand nous avons déjà procédé par "3, 5, 3, 5", Fox s'est déplacé d'abord sur un multiple de 3, puis sur le prédécesseur d'un multiple de 3, puis sur le successeur d'un multiple de 3, puis sur un multiple de 3. Nous sommes donc ramenés dans le premier cas où nous trouvons Fox par "3, 5".

Conclusion Dans ce troisième temps, faire suivre "3, 5, 3, 5" de "3, 5" permet de trouver Fox.

Conclusion globale Nous avons une stratégie pour trouver Fox, à savoir "3, 5, 3, 5, 3, 5".

Remarque Pour le problème 3, nous nous sommes efforcés de justifier qu'il n'existe pas de stratégie plus courte que celle proposée et aussi de déterminer toutes les stratégies les plus courtes. Les stratégies qui permettent de trouver assurément Fox sont : "2, 4, 2, 4, 2, 4", "2, 4, 4, 3, 3, 5", "3, 3, 4, 4, 4, 3", "3, 5, 2, 4, 4, 3", "3, 5, 3, 5, 3, 5", "4, 3, 3, 3, 4, 4", "4, 3, 3, 5, 2, 4" et "4, 3, 4, 3, 4, 3" (résultat obtenu de façon informatique).

Encore une extension de l'énigme, avec de nouveaux déplacements du renard et *n* terriers



n terriers alignés ($n \geq 3$).

APPLICATION

mode d'emploi

*Un renard, Fox, est dans l'un des **n terriers**.*
Chaque nuit, Fox passe du terrier occupé à l'un des deux terriers parmi celui juste à gauche et celui juste à droite de celui juste à droite.
Chaque jour, nous pouvons regarder dans l'un des terriers pour voir si Fox y est.
Donner une stratégie pour trouver Fox.

Ci-dessous, une solution du problème est donnée. Ne poursuivez pas votre lecture si vous préférez chercher seul une solution.

Une stratégie

Notre stratégie se démontre de la même façon que pour le problème "Fox and Holes" avec de nouveaux déplacements du renard (avec 6 terriers), en trois temps.

Dans un premier temps, nous traitons l'éventualité où Fox débute d'un terrier multiple de 3 en commençant par le terrier 3 et nous empêchons Fox de se rapprocher des terriers déjà fouillés tout en progressant sur la file des terriers.

Dans un second temps, si Fox avait commencé dans un terrier précédant un terrier multiple de 3, nous nous ramenons au premier temps.

Et dans un troisième temps, si Fox avait commencé dans un terrier succédant un terrier multiple de 3, nous nous ramenons encore au premier temps. Pour se ramener du deuxième temps au premier, il faut que le nombre de terriers visités auparavant soit un prédécesseur d'un multiple de 3 et si nécessaire, on complète donc par autant d'observations farfelues que nécessaire. Pour se ramener du troisième temps au premier, il faut que le nombre de terriers visités auparavant soit un successeur d'un multiple de 3 et si nécessaire, on complète donc par autant d'observations farfelues que nécessaire.

Si n est impair sans être un multiple de 3, on peut poser $n = 2k + 1$ avec k entier naturel. La stratégie "3, 5, 7, ..., $2k$ - 1, 3, 5, 7, ..., $2k$ - 1, 3, 5, 7, ..., $2k$ - 1" permet de trouver Fox.

Si n est impair et multiple de 3, on peut poser $n = 2k + 1$ avec k entier naturel. La stratégie "3, 5, 7, ..., $2k$ - 1, x, 3, 5, 7, ..., $2k$ - 1, y, 3, 5, 7, ..., $2k$ - 1" où x et y sont des nombres quelconques permet de trouver Fox.

Si n est pair sans être prédécesseur d'un multiple de 3, on peut poser $n = 2k$ avec k entier naturel. La stratégie "3, 5, 7, ..., 2k - 1, 3, 5, 7, ..., 2k - 1, 3, 5, 7, ..., 2k - 1" permet de trouver Fox.

Si n est pair et prédécesseur d'un multiple de 3, on peut poser $n = 2k$ avec k entier naturel. La stratégie "3, 5, 7, ..., 2k - 1, x, 3, 5, 7, ..., 2k - 1, y, 3, 5, 7, ..., 2k - 1" où x et y sont des nombres quelconques permet de trouver Fox.

Remarque Pour ce problème, nous nous sommes contentés de fournir une seule stratégie valide, mais nous n'avons aucune preuve que cette solution soit la plus courte possible et encore moins pu donner l'ensemble des stratégies de longueur minimale.

Toujours plus fort, avec *n* terriers et encore de nouveaux déplacements du renard

Attention, ici il y a un peu de maths !

Rappels sur les ensembles

Notion de groupe avec structure additive

Légers rappels sur Z/2Z muni de la loi additive : (Z/2Z,+)

Rappels analogues sur Z/3Z muni de la loi additive : (Z/3Z,+)

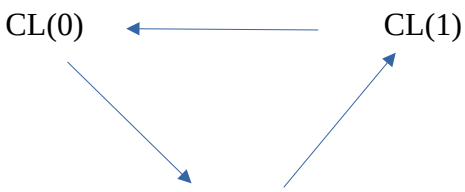
Retour au problème : "Qu'avons-nous appris des quelques exemples traités ?"

Ce qui est crucial dans les démonstrations des problèmes qui précèdent, c'est de regrouper correctement les cas.

Dans les problèmes où les déplacements sont de 1 vers la gauche ou de 1 vers la droite, on s'intéresse aux classes de nombres CL(0) des nombres pairs et CL(1) des nombres impairs. Sur ces classes de nombres, nous avons le diagramme suivant, que la flèche désigne "se déplacer dans le terrier voisin d'un côté" ou "de l'autre" **(on dit que les opérations "retirer 1" ou "ajouter 1" sont confondues sur les classes modulo 2)**.

CL(0) ⇄ CL(1)

Dans les problèmes où les déplacements sont de 1 vers la gauche ou de 2 vers la droite, on s'intéresse aux classes de nombres CL(0) des nombres multiples de 3, CL(1) des nombres succédant aux multiples de 3 et CL(2) des nombres précédant les multiples de 3. Sur ces classes de nombres, nous avons le diagramme suivant, que la flèche désigne "se déplacer dans le terrier juste à gauche" ou "juste à droite de celui juste à droite" **(on dit que les opérations "retirer 1" ou "ajouter 2" sont confondues sur les classes modulo 3)**.



CL(2)

Problème "Fox and Holes", avec *n* terriers et encore de nouveaux déplacements du renard

Un dessin est fourni présentant n terriers alignés de la gauche vers la droite, puis le texte suivant :

> *Un renard, Fox, est dans l'un des* ***n terriers****.*
> *Chaque nuit,* ***Fox passe du terrier occupé à l'un des deux terriers parmi celui situé à g terriers vers la gauche et celui situé à d terriers vers la droite****. On suppose* $n \geq g + d$.
> *Chaque jour, nous pouvons regarder dans l'un des terriers pour voir si Fox y est.*
> *Donner une stratégie pour trouver Fox.*

Remarque L'hypothèse $n \geq g + d$ garantit que de n'importe quel terrier, le renard a toujours au moins un déplacement possible.

Rappels analogues aux précédents sur Z/(g + d)Z muni de la loi additive : (Z/(g + d)Z,+)

Une modélisation adaptée

Il s'agit du cas le plus général quand le renard a deux possibilités de déplacement : les déplacements sont imposés de sens contraires, l'un vers la gauche et l'autre vers la droite, car s'ils allaient tous deux dans le même sens, au bout d'un moment, le renard n'aurait plus de déplacement à disposition.

Nous pouvons d'abord imposer, sans perte de généralité, que $g \leq d$, quitte à changer l'ordre dans lequel on numérote les terriers (au lieu de numéroter de la gauche vers la droite, on numérote de la droite vers la gauche) et on inverse ainsi gauche et droite.

Nous allons ensuite supposer que g et d sont premiers entre eux. En effet, s'ils ne sont pas premiers entre eux, ils ont un PGCD égal à δ qui est différent de 1 et à supposer que la position initiale du renard soit égale à ρ modulo δ, il ne va se déplacer que sur des terriers dont la position est égale à ρ modulo δ. Ainsi, on peut renuméroter les terriers sur lesquels il se déplace de la gauche vers la droite et considérer qu'avec cette nouvelle numérotation, le renard se déplace de g/δ vers la gauche ou de d/δ vers la droite. Selon la position initiale de Fox (c'est-à-dire de la valeur de ρ), c'est comme si nous avions δ ensembles disjoints de terriers sur lesquels nous recherchons Fox successivement (d'abord pour ρ = 0, puis pour ρ = 1, ... jusqu'à ρ = δ - 1).

La caractérisation des générateurs de Z/(*g* + *d*)Z est essentielle pour comprendre la stratégie générale déployée dans la section à suivre.

Définition d'une stratégie

Ce problème admet toujours une stratégie pour trouver Fox. Nous pouvons même décrire complètement une stratégie dans le cas général. **Cependant, nous choisissons de ne pas justifier cette stratégie car notre justification actuelle mobilise des propriétés des groupes additifs cycliques (idée qui est connectée aux diagrammes qui précèdent dans cette section) Z/(g + d)Z et ses éléments générateurs.**

Grosso modo, la stratégie que nous proposons ...

Premier temps

Soit k le plus grand entier naturel tel que $g + k\,d < n$ est effective.
On applique la stratégie "$g + d$, $g + 2d$, $g + 3d$, ..., $g + k\,d$".
Ensuite, on calcule le reste r de la division euclidienne de $k + 1$ par $g + d$ et on va agir différemment selon si CL(r) est ou n'est pas générateur de Z/($g + d$)Z ...

Premier cas Si CL(r) est générateur de Z/($g + d$)Z, alors du deuxième au ($g + d$)ème temps : on applique à nouveau la stratégie "$g + d$, $g + 2d$, $g + 3d$, ..., $g + k\,d$".

Second cas Si CL(r) n'est pas générateur de Z/($g + d$)Z, alors on cherche le plus petit δ tel que CL($r + \delta$) (ce δ existe toujours car CL($g + d - 1$) est toujours générateur) et du deuxième au ($g + d$)ème temps, on applique δ observations farfelues "x_1, x_2, ..., x_δ" avant de reprendre la stratégie "$g + d$, $g + 2d$, $g + 3d$, ..., $g + k\,d$".

Explication de la stratégie précédente avec g = 1, d = 2 et n = 7

Recherche de k 3 (pour $k = 0$) et $5 = 3 + 2$ (pour $k = 1$) sont les seules possibilités pour que $g + k\,d < n$ soit vraie, on obtient $k = 1$ comme la plus grande des éventualités.

Dans le premier temps, on applique la stratégie "3, 5".

Ensuite, on calcule $r = 2$ comme reste dans la division euclidienne de $2 = 1 + 1$ par 3 et comme CL(2) est générateur de Z/3Z (voir dans les rappels sur Z/3Z), on entre dans le premier cas : dans les deuxième et troisième temps, on applique encore la stratégie "3, 5".

Conclusion La stratégie "3, 5, 3, 5, 3, 5" fonctionne, comme cela a déjà été mentionné précédemment, dans l'article (1).

Explication de la stratégie précédente avec g = 1, d = 2 et n = 8

Recherche de k 3 (pour $k = 0$), $5 = 3 + 2$ (pour $k = 1$) et $7 = 3 + 2 \times 2$ sont les seules possibilités pour que $g + k\,d < n$ soit vraie, on obtient $k = 2$ comme la plus grande des éventualités.

Dans le premier temps, on applique la stratégie "3, 5, 7".

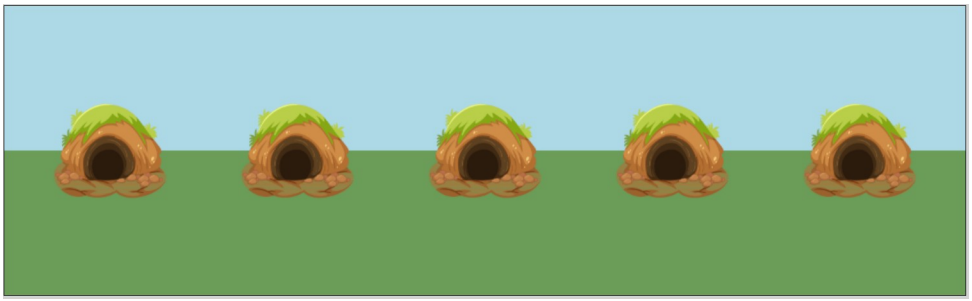
Ensuite, on calcule $r = 0$ comme reste dans la division euclidienne de $3 = 2 + 1$ par 3 et comme CL(0) n'est pas générateur de Z/3Z (voir dans les rappels sur Z/3Z), on entre dans le deuxième cas : on obtient $\delta = 1$ car CL(1) est générateur de Z/3Z (voir dans les rappels sur Z/3Z) dans les deuxième et troisième temps, on applique la stratégie "x, 3, 5, 7" avec x une observation farfelue.

Conclusion La stratégie "3, 5, 7, x, 3, 5, 7, x, 3, 5, 7" fonctionne comme cela a déjà été mentionné précédemment, dans l'article (1).

Avec augmentation des possibilités du renard

Cette section est purement récréative, sans démonstration.

Problème "Fox and Holes" avec augmentation des possibilités du renard



Un renard, Fox, est dans l'un des 5 terriers.
Chaque nuit, ***Fox reste dans son terrier ou passe dans un terrier voisin de celui qu'il occupe****.*
Chaque jour, nous pouvons regarder dans l'un des terriers pour voir si Fox y est.
Pouvons-nous trouver une stratégie pour trouver Fox ?

Appropriation

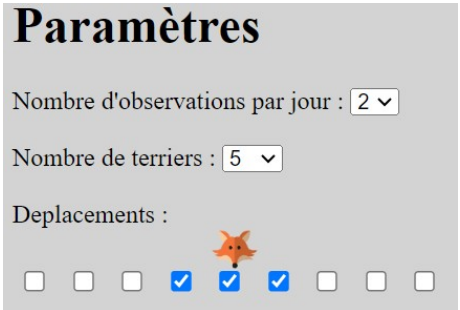
Une stratégie valide doit l'être pour n'importe quel trajet du renard et, en particulier, quand le renard connaît notre stratégie et tente de s'y soustraire. Le problème précédent n'a pas de solution !

En effet, admettons que le renard soit sur une case i pour i allant de 1 à 5, de cette case, il a toujours au moins deux (il en a trois pour $i = 2$, $i = 3$ et $i = 4$, mais seulement deux pour $i = 1$ et $i = 5$) possibilités de déplacement (ce minimum n'était que de 1 pour le problème "Fox and Holes" classique et cela nous a permis de trouver une stratégie solution pour ce problème). S'il sait que nous allons observer dans l'un de ces deux terriers, il lui suffit d'opter pour l'autre afin de se soustraire à notre observation. Cet argument suffit à montrer qu'**il ne peut y avoir de stratégie capable de déceler Fox à coup sûr**.

Du coup, si on autorise trois déplacements au renard, on doit sans doute autoriser deux observations par jour pour pouvoir trouver Fox.

En réalité, en mobilisant la même démonstration que la précédente, on pourrait démontrer la propriété plus générale suivante : "si de n'importe quelle position, le renard a au moins k déplacements disponibles, avec strictement moins de k observations, il sera impossible de trouver une stratégie capable de trouver Fox de façon certaine".

Problème "Fox and Holes" avec augmentation des possibilités du renard et du nombre d'observations



APPLICATION

mode d'emploi

Un renard, Fox, est dans l'un des 5 terriers.
Chaque nuit, Fox reste dans son terrier ou passe dans un terrier voisin de celui qu'il occupe.

*Chaque jour, **nous pouvons regarder dans deux des terriers** pour voir si Fox y est.*
Donner une stratégie pour trouver Fox.

Ce problème est très facile et est laissé en exercice !

Une solution. → Annexe B

Problème "Fox and Holes" avec augmentation des possibilités du renard et du nombre d'observations, un peu plus complexe

APPLICATION

mode d'emploi

*Un renard, Fox, est dans l'un des **6 terriers**.*
Chaque nuit, Fox reste dans son terrier ou passe dans un terrier voisin de celui qu'il occupe.
Chaque jour, nous pouvons regarder dans deux des terriers pour voir si Fox y est.
Donner une stratégie pour trouver Fox.

Ce problème est également très facile et est laissé en exercice !

Une solution. → Annexe B

Problème "Fox and Holes" avec augmentation des possibilités du renard et du nombre d'observations, encore un peu plus complexe

APPLICATION

mode d'emploi

Un renard, Fox, est dans l'un des 6 terriers.
*Chaque nuit, **Fox reste dans son terrier ou passe du terrier occupé à l'un des deux terriers parmi celui situé à 2 terriers vers la gauche et celui situé à 3 terriers vers la droite**.*
Chaque jour, nous pouvons regarder dans deux des terriers pour voir si Fox y est.
Donner une stratégie pour trouver Fox.

Ce problème est un peu plus résistant que les deux précédant et est laissé en exercice !

Les 117 solutions en 5 jours.→ Annexe B

Problème "Fox and Holes" avec augmentation des possibilités du renard et du nombre d'observations, complexe

APPLICATION

mode d'emploi

*Un renard, Fox, est dans l'un des **7 terriers**.*
Chaque nuit, Fox reste dans son terrier ou passe du terrier occupé à l'un des deux terriers parmi celui situé à 2 terriers vers la gauche et celui situé à 3 terriers vers la droite.
Chaque jour, nous pouvons regarder dans deux des terriers pour voir si Fox y est.
Donner une stratégie pour trouver Fox.

Ce problème est encore un peu plus résistant que les trois précédant et est laissé en exercice !

Les 102 solutions en 6 jours → Annexe B

Programmation en C++

Nous proposons également un programme en C++ qui génère dans un cadre limité à 32 terriers l'ensemble des solutions en un minimum de jours. → Annexe C

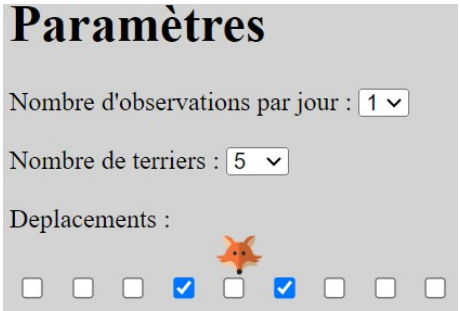
Les 102 solutions en 6 jours du dernier problème sont à nouveau données. → Annexe C

Annexe : mode d'emploi

Fox : mode d'emploi de l'application

Lien vers l'application : APPLICATION

Choix des paramètres



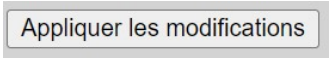
Pour choisir le nombre d'observations par jour, sélectionner ce nombre de 1 à 3. Par défaut, ce nombre est 1.

Pour définir le nombre de terriers souhaités, sélectionner ce nombre de 3 à 13. Par défaut, ce nombre est 5.

Pour déterminer les déplacements nocturnes du renard, cocher les possibilités offertes depuis sa position précédente désignée par son image, au centre des choix, il peut ainsi se mouvoir jusqu'à 4

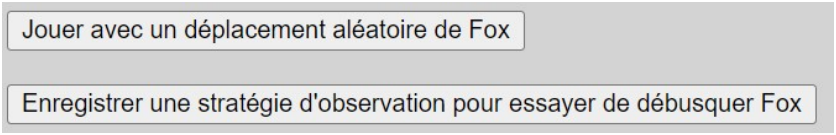
terriers vers la gauche ou vers la droite. Par défaut, il peut se déplacer vers le terrier juste à gauche ou juste à droite.

En cas de modification des paramètres ne pas omettre de cliquer le "Appliquer les modifications" :



Jouer ou enregistrer une stratégie

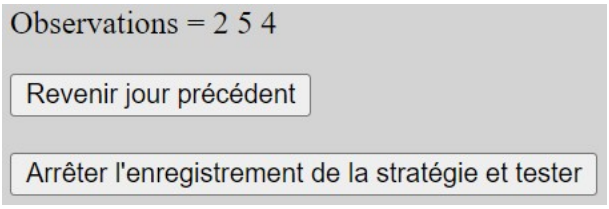
Après génération de l'image prenant en compte les paramètres, l' APPLICATION vous propose alors deux options (à cliquer pour valider) : soit de simuler une recherche de Fox avec un déplacement aléatoire du renard, soit d'enregistrer un stratégie garantissant de trouver Fox.



Pour répondre au problème, il faut **trouver une stratégie permettant de trouver Fox *quel que soit le déplacement du renard*** et non pas pour un déplacement particulier de Fox comme celui généré aléatoirement par la simulation.

Le *bouton* "Jouer avec un déplacement aléatoire de Fox" permet de jouer en cliquant sur les différents terriers dans lesquels observer, chronologiquement.

Enregistrement de la stratégie "2, 5, 4".

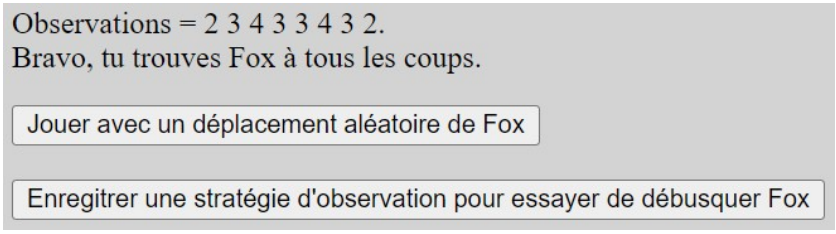


En cliquant sur le *bouton* "Enregistrer une stratégie d'observation pour essayer de débusquer Fox", vous avez la possibilité d'enregistrer une stratégie en cliquant successivement sur les différents terriers dans lesquels observer, chronologiquement, puis sur le *bouton* "Arrêter l'enregistrement de la stratégie et tester".

Remarque : le *bouton* "Revenir au jour précédent" permet d'annuler l'enregistrement de la dernière observation et de poursuivre.

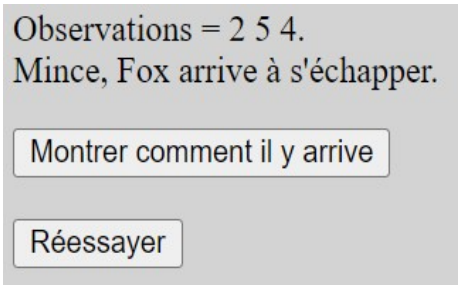
Une fois la stratégie enregistrée, deux éventualités sont traitées :

- soit la stratégie est efficace (et elle peut l'être même si elle n'est pas optimale au niveau du nombre d'observations) :

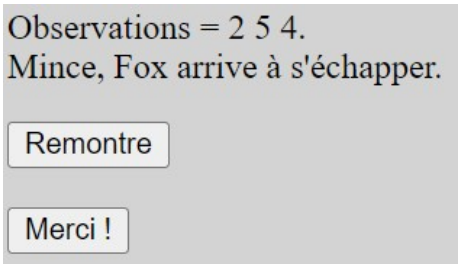


  dans ce cas, il n'est pas forcément utile de réessayer (sauf pour chercher en moins d'observations) ;

- soit la stratégie laisse échapper Fox :



dans ce cas, le *bouton* "Réessayer" permet de proposer un nouvelle stratégie sans consulter la stratégie d'évitement du renard et le *bouton* "Montrer comment il y arrive" permet de visualiser une stratégie dans laquelle le renard se soustrait aux observations



et le *bouton* "Remontre", comme son nom l'indique permet de voir à nouveau la stratégie d'évitement du renard, le *bouton* "Merci !", quant à lui, permet de revenir au menu.

Annexe A

Programme en "Maple 5.0" pour le problème initial

En préambule, on définit deux fonctions : *g* qui permet de passer d'un entier à son prédécesseur ; et *d* qui permet de passer d'un entier à son successeur.

```
> g:=proc(x);
> RETURN(x-1)
> end:
> d:=proc(x);
> RETURN(x+1)
> end:
```

On cherche d'abord les stratégies valables en 5 observations

Dans le programme suivant, tous les trajets de Fox sont numérotés par l'indice *n* et stockés dans PF[n]. Il apparaît que *n* varie de 1 à 42 (nous avons 42 trajets possibles).

```
> n:=1:
> for Fox1 from 1 to 5 do
>       for f1 in {g,d} do
>       if f1(Fox1)>0 and f1(Fox1)<6 then Fox2:=f1(Fox1):
>             for f2 in {g,d} do
>             if f2(Fox2)>0 and f2(Fox2)<6 then Fox3:=f2(Fox2):
>                   for f3 in {g,d} do
>                   if f3(Fox3)>0 and f3(Fox3)<6 then Fox4:=f3(Fox3):
```

```
>                                 for f4 in {g,d} do
>                                 if f4(Fox4)>0 and f4(Fox4)<6 then Fox5:=f4(Fox4):
>                                 PosFox:=Fox1,Fox2,Fox3,Fox4,Fox5: PF[n]:=[PosFox]: n:=n+1
>                                 fi: od:
>                         fi: od:
>                 fi: od:
>         fi: od:
> od:
```

Nous récupérons les 42 déplacements possibles du renard au bout de 5 déplacements stockés comme une liste des 5 positions successivement occupées ...

Par exemple, PF(1) est [1, 2, 3, 4, 5] et se lit : "Fox est au terrier 1 au départ, puis il passe en 2, puis en 3, puis en 4, et enfin en 5".

D'un autre côté, nous stockons aussi les 3 125 stratégies ...

```
> n:=1:
> for i1 from 1 to 5 do
>         for i2 from 1 to 5 do
>                 for i3 from 1 to 5 do
>                         for i4 from 1 to 5 do
>                                 for i5 from 1 to 5 do
>                                 Strat[n]:=[i1,i2,i3,i4,i5]:n:=n+1
>                                 od:
>                         od:
>                 od:
>         od:
> od:
```

Une stratégie où l'on regarde 2 fois en 2 puis 3 fois en 4 est notée [2, 2, 4, 4, 4]. Ces stratégies sont numérotées de 1 à 3 125.

Une stratégie parmi les 3 125 (*n* allant de 1 à 3 125) est valable si pour chacun des 42 trajets de Fox possibles (*k* allant de 1 à 42), on peut trouver un indice i (*i* allant de 1 à 5) commun entre la stratégie et le trajet du renard pour lequel les positions en cet indice soient égales.

```
> for n from 1 to 3125 do
> valid[n]:=1:
>         for k from 1 to 42 do
>         val[n,k]:=0:
>                 for i from 1 to 5 do
>                 if PF[k][i]=Strat[n][i] then val[n,k]:=1 fi:
```

```
>                 od:
>         valid[n]:=valid[n]*val[n,k]:
>         od:
> if valid[n]=1 then print(Strat[n]) fi:
> od:
```

Le programme s'est arrêté sans avoir révélé de stratégie valable, ce qui indique qu'il n'existe aucune stratégie permettant de trouver automatiquement Fox en 5 observations.

On recommence pour chercher les stratégies valables en 6 observations

```
> n:=1:
> for Fox1 from 1 to 5 do
>         for f1 in {g,d} do
>         if f1(Fox1)>0 and f1(Fox1)<6 then Fox2:=f1(Fox1):
>                 for f2 in {g,d} do
>                 if f2(Fox2)>0 and f2(Fox2)<6 then Fox3:=f2(Fox2):
>                         for f3 in {g,d} do
>                         if f3(Fox3)>0 and f3(Fox3)<6 then Fox4:=f3(Fox3):
>                                 for f4 in {g,d} do
>                                 if f4(Fox4)>0 and f4(Fox4)<6 then Fox5:=f4(Fox4):
>                                         for f5 in {g,d} do
>                                         if f5(Fox5)>0 and f5(Fox5)<6 then Fox6:=f5(Fox5):
>                                         PosFox:=Fox1,Fox2,Fox3,Fox4,Fox5,Fox6:
>                                         PF[n]:=[PosFox]: n:=n+1
>                                         fi: od:
>                                 fi: od:
>                         fi: od:
>                 fi: od:
>         fi: od:
> od:
```

Nous récupérons les 72 déplacements possibles du renard au bout de 6 déplacements ...

D'un autre côté, nous stockons aussi les 15 625 stratégies ...

```
> n:=1:
> for i1 from 1 to 5 do
>         for i2 from 1 to 5 do
```

```
>                 for i3 from 1 to 5 do
>                         for i4 from 1 to 5 do
>                                 for i5 from 1 to 5 do
>                                         for i6 from 1 to 5 do
>                                         Strat[n]:=[i1,i2,i3,i4,i5,i6]:n:=n+1
>                                         od:
>                                 od:
>                         od:
>                 od:
>         od:
> od:
```

Une stratégie parmi les 15 625 (*n* allant de 1 à 15 625) est valable si pour chacun des 72 trajets de Fox possibles (*k* allant de 1 à 72), on peut trouver un indice i (*i* allant de 1 à 6) commun entre la stratégie et le trajet du renard pour lequel les positions en cet indice soient égales.

```
> for n from 1 to 15625 do
> valid[n]:=1:
>         for k from 1 to 72 do
>         val[n,k]:=0:
>                 for i from 1 to 6 do
>                 if PF[k][i]=Strat[n][i] then val[n,k]:=1 fi:
>                 od:
>         valid[n]:=valid[n]*val[n,k]:
>         od:
> if valid[n]=1 then print(Strat[n]) fi:
> od:
```

```
[2, 3, 4, 2, 3, 4]
[2, 3, 4, 4, 3, 2]
[4, 3, 2, 2, 3, 4]
[4, 3, 2, 4, 3, 2]
```

Nous obtenons ci-dessus les 4 seules stratégies permettant de trouver assurément Fox en 6 observations.

Annexe B

Solutions des Problèmes "Fox and Holes" avec augmentation des possibilités du renard et du nombre d'observations

Cette fois, pour résumer une stratégie (pas nécessairement qui permette de trouver Fox) nous pouvons écrire une suite comme "(2 ; 5), (4 ; 5)" qui résume "d'abord, nous regardons dans les terriers 2 et 5, puis dans les 4 et 5". Plus généralement, nous donnons la suite des couples de terriers observés.

Solution du Problème "Fox and Holes" avec augmentation des possibilités du renard et du nombre d'observations

Initialement, le renard est en 1, 2, 3, 4 ou 5.

En observant en 1 et en 2, soit on le trouve, soit il était en 3, 4 ou 5 et peut donc atteindre 2, 3, 4 ou 5.

En observant ensuite en 2 et en 3, soit on le trouve, soit il était juste précédemment en 4 ou 5 et peut donc atteindre 3, 4 ou 5.

En observant ensuite en 3 et en 4, soit on le trouve, soit il était juste précédemment en 5 et peut donc atteindre 4 ou 5.

Enfin, en observant ensuite en 4 et en 5, on le trouve.

Cette stratégie se note "(1 ; 2), (2 ; 3), (3 ; 4), (4 ; 5)".

Solution du Problème "Fox and Holes" avec augmentation des possibilités du renard et du nombre d'observations, un peu plus complexe

Initialement, le renard est en 1, 2, 3, 4, 5 ou 6.

En observant en 1 et en 2, soit on le trouve, soit il était en 3, 4, 5 ou 6 et peut donc atteindre 2, 3, 4, 5 ou 6.

En observant ensuite en 2 et en 3, soit on le trouve, soit il était juste précédemment en 4, 5 ou 6 et peut donc atteindre 3, 4, 5 ou 6.

En observant ensuite en 3 et en 4, soit on le trouve, soit il était juste précédemment en 5 ou 6 et peut donc atteindre 4, 5 ou 6.

En observant ensuite en 4 et en 5, soit on le trouve, soit il était juste précédemment en 6 et peut donc atteindre 5 ou 6.

Enfin, en observant ensuite en 5 et en 6, on le trouve.

Cette stratégie se note "(1 ; 2), (2 ; 3), (3 ; 4), (4 ; 5), (5 ; 6)".

Solution du Problème "Fox and Holes" avec augmentation des possibilités du renard et du nombre d'observations, encore un peu plus complexe

Initialement, le renard est en 1, 2, 3, 4, 5 ou 6.

En observant en 2 et en 5, soit on le trouve, soit il était en 1, 3, 4 ou 6 et peut donc atteindre 1, 2, 3, 4 ou 6.

En observant ensuite en 1 et en 3, soit on le trouve, soit il était juste précédemment en 2, 4 ou 6 et peut donc atteindre 2, 4, 5 ou 6.

En observant ensuite en 2 et en 5, soit on le trouve, soit il était juste précédemment en 4 ou 6 et peut donc atteindre 2, 4 ou 6.

En observant ensuite en 2 et en 4, soit on le trouve, soit il était juste précédemment en 6 et peut donc atteindre 4 ou 6.

Enfin, en observant ensuite en 4 et en 6, on le trouve.

Cette stratégie se note "(2 ; 5), (1 ; 3), (2 ; 5), (2 ; 4), (4 ; 6)".

Les 117 stratégies pour trouver Fox en 5 jours sont :

"(1 ; 3), (3 ; 5), (2 ; 5), (2 ; 4), (4 ; 6)" ;
"(1 ; 3), (3 ; 5), (2 ; 5), (2 ; 6), (2 ; 4)" ;
"(1 ; 3), (3 ; 5), (2 ; 5), (4 ; 6), (2 ; 5)" ;
"(1 ; 3), (3 ; 5), (4 ; 6), (2 ; 3), (3 ; 5)" ;
"(1 ; 3), (3 ; 5), (4 ; 6), (3 ; 5), (2 ; 5)" ;
"(1 ; 3), (3 ; 5), (5 ; 6), (2 ; 4), (3 ; 5)" ;
"(1 ; 3), (3 ; 5), (5 ; 6), (2 ; 5), (2 ; 4)" ;
"(1 ; 3), (3 ; 5), (5 ; 6), (4 ; 5), (2 ; 5)" ;
"(1 ; 3), (3 ; 6), (3 ; 4), (2 ; 3), (3 ; 5)" ;
"(1 ; 3), (3 ; 6), (3 ; 4), (3 ; 5), (2 ; 5)" ;
"(1 ; 3), (3 ; 6), (3 ; 5), (2 ; 4), (3 ; 5)" ;
"(1 ; 3), (3 ; 6), (3 ; 5), (2 ; 5), (2 ; 4)" ;
"(1 ; 3), (3 ; 6), (3 ; 5), (4 ; 5), (2 ; 5)" ;
"(2 ; 4), (3 ; 5), (1 ; 2), (2 ; 4), (4 ; 6)" ;
"(2 ; 4), (3 ; 5), (1 ; 2), (2 ; 6), (2 ; 4)" ;
"(2 ; 4), (3 ; 5), (1 ; 2), (4 ; 6), (2 ; 5)" ;
"(2 ; 4), (3 ; 5), (1 ; 6), (2 ; 4), (3 ; 5)" ;
"(2 ; 4), (3 ; 5), (1 ; 6), (2 ; 5), (2 ; 4)" ;
"(2 ; 5), (2 ; 4), (1 ; 3), (2 ; 4), (4 ; 6)" ;
"(2 ; 5), (2 ; 4), (1 ; 3), (2 ; 6), (2 ; 4)" ;
"(2 ; 5), (2 ; 4), (1 ; 3), (4 ; 6), (2 ; 5)" ;
"(2 ; 5), (2 ; 4), (3 ; 4), (1 ; 4), (4 ; 6)" ;
"(2 ; 5), (2 ; 4), (3 ; 4), (1 ; 6), (2 ; 4)" ;
"(2 ; 5), (2 ; 4), (3 ; 4), (4 ; 6), (1 ; 4)" ;
"(2 ; 5), (2 ; 4), (3 ; 6), (1 ; 2), (2 ; 4)" ;
"(2 ; 5), (2 ; 4), (3 ; 6), (1 ; 4), (2 ; 5)" ;
"(2 ; 5), (2 ; 4), (3 ; 6), (2 ; 4), (1 ; 4)" ;
"(2 ; 5), (3 ; 6), (1 ; 4), (2 ; 3), (3 ; 5)" ;
"(2 ; 5), (3 ; 6), (1 ; 4), (3 ; 5), (2 ; 5)" ;
"(2 ; 5), (3 ; 6), (1 ; 5), (2 ; 4), (3 ; 5)" ;
"(2 ; 5), (3 ; 6), (1 ; 5), (2 ; 5), (2 ; 4)" ;
"(2 ; 5), (3 ; 6), (1 ; 5), (4 ; 5), (2 ; 5)" ;
"(3 ; 5), (1 ; 5), (2 ; 5), (2 ; 6), (2 ; 4)" ;
"(3 ; 5), (1 ; 5), (2 ; 5), (4 ; 6), (2 ; 5)" ;
"(3 ; 5), (1 ; 5), (4 ; 6), (2 ; 3), (3 ; 5)" ;
"(3 ; 5), (1 ; 5), (4 ; 6), (3 ; 5), (2 ; 5)" ;

"(2 ; 4), (3 ; 5), (1 ; 6), (4 ; 5), (2 ; 5)" ;

"(2 ; 4), (3 ; 5), (2 ; 4), (1 ; 4), (4 ; 6)" ;

"(2 ; 4), (3 ; 5), (2 ; 4), (1 ; 6), (2 ; 4)" ;

"(2 ; 4), (3 ; 5), (2 ; 4), (4 ; 6), (1 ; 4)" ;

"(2 ; 4), (3 ; 5), (2 ; 6), (1 ; 2), (2 ; 4)" ;

"(2 ; 4), (3 ; 5), (2 ; 6), (1 ; 4), (2 ; 5)" ;

"(2 ; 4), (3 ; 5), (2 ; 6), (2 ; 4), (1 ; 4)" ;

"(2 ; 4), (4 ; 5), (1 ; 3), (2 ; 4), (4 ; 6)" ;

"(2 ; 4), (4 ; 5), (1 ; 3), (2 ; 6), (2 ; 4)" ;

"(2 ; 4), (4 ; 5), (1 ; 3), (4 ; 6), (2 ; 5)" ;

"(2 ; 4), (4 ; 5), (3 ; 4), (1 ; 4), (4 ; 6)" ;

"(2 ; 4), (4 ; 5), (3 ; 4), (1 ; 6), (2 ; 4)" ;

"(2 ; 4), (4 ; 5), (3 ; 4), (4 ; 6), (1 ; 4)" ;

"(2 ; 4), (4 ; 5), (3 ; 6), (1 ; 2), (2 ; 4)" ;

"(2 ; 4), (4 ; 5), (3 ; 6), (1 ; 4), (2 ; 5)" ;

"(2 ; 4), (4 ; 5), (3 ; 6), (2 ; 4), (1 ; 4)" ;

"**(2 ; 5), (1 ; 3), (2 ; 5), (2 ; 4), (4 ; 6)**" (celle trouvée plus haut) ;

"(2 ; 5), (1 ; 3), (2 ; 5), (2 ; 6), (2 ; 4)" ;

"(2 ; 5), (1 ; 3), (2 ; 5), (4 ; 6), (2 ; 5)" ;

"(2 ; 5), (1 ; 3), (4 ; 6), (2 ; 3), (3 ; 5)" ;

"(2 ; 5), (1 ; 3), (4 ; 6), (3 ; 5), (2 ; 5)" ;

"(2 ; 5), (1 ; 3), (5 ; 6), (2 ; 4), (3 ; 5)" ;

"(2 ; 5), (1 ; 3), (5 ; 6), (2 ; 5), (2 ; 4)" ;

"(2 ; 5), (1 ; 3), (5 ; 6), (4 ; 5), (2 ; 5)" ;

"(2 ; 5), (2 ; 3), (1 ; 2), (2 ; 4), (4 ; 6)" ;

"(2 ; 5), (2 ; 3), (1 ; 2), (2 ; 6), (2 ; 4)" ;

"(2 ; 5), (2 ; 3), (1 ; 2), (4 ; 6), (2 ; 5)" ;

"(2 ; 5), (2 ; 3), (1 ; 6), (2 ; 4), (3 ; 5)" ;

"(2 ; 5), (2 ; 3), (1 ; 6), (2 ; 5), (2 ; 4)" ;

"(2 ; 5), (2 ; 3), (1 ; 6), (4 ; 5), (2 ; 5)" ;

"(3 ; 5), (1 ; 5), (5 ; 6), (2 ; 4), (3 ; 5)" ;

"(3 ; 5), (1 ; 5), (5 ; 6), (2 ; 5), (2 ; 4)" ;

"(3 ; 5), (1 ; 5), (5 ; 6), (4 ; 5), (2 ; 5)" ;

"(3 ; 5), (1 ; 6), (3 ; 4), (2 ; 3), (3 ; 5)" ;

"(3 ; 5), (1 ; 6), (3 ; 4), (3 ; 5), (2 ; 5)" ;

"(3 ; 5), (1 ; 6), (3 ; 5), (2 ; 4), (3 ; 5)" ;

"(3 ; 5), (1 ; 6), (3 ; 5), (2 ; 5), (2 ; 4)" ;

"(3 ; 5), (1 ; 6), (3 ; 5), (4 ; 5), (2 ; 5)" ;

"(3 ; 5), (2 ; 5), (1 ; 2), (2 ; 4), (4 ; 6)" ;

"(3 ; 5), (2 ; 5), (1 ; 2), (2 ; 6), (2 ; 4)" ;

"(3 ; 5), (2 ; 5), (1 ; 2), (4 ; 6), (2 ; 5)" ;

"(3 ; 5), (2 ; 5), (1 ; 6), (2 ; 4), (3 ; 5)" ;

"(3 ; 5), (2 ; 5), (1 ; 6), (2 ; 5), (2 ; 4)" ;

"(3 ; 5), (2 ; 5), (1 ; 6), (4 ; 5), (2 ; 5)" ;

"(3 ; 5), (2 ; 5), (2 ; 4), (1 ; 4), (4 ; 6)" ;

"(3 ; 5), (2 ; 5), (2 ; 4), (1 ; 6), (2 ; 4)" ;

"(3 ; 5), (2 ; 5), (2 ; 4), (4 ; 6), (1 ; 4)" ;

"(3 ; 5), (2 ; 5), (2 ; 6), (1 ; 2), (2 ; 4)" ;

"(3 ; 5), (2 ; 5), (2 ; 6), (1 ; 4), (2 ; 5)" ;

"(3 ; 5), (2 ; 5), (2 ; 6), (2 ; 4), (1 ; 4)" ;

"(3 ; 5), (5 ; 6), (1 ; 4), (2 ; 3), (3 ; 5)" ;

"(3 ; 5), (5 ; 6), (1 ; 4), (3 ; 5), (2 ; 5)" ;

"(3 ; 5), (5 ; 6), (1 ; 5), (2 ; 4), (3 ; 5)" ;

"(3 ; 5), (5 ; 6), (1 ; 5), (2 ; 5), (2 ; 4)" ;

"(3 ; 5), (5 ; 6), (1 ; 5), (4 ; 5), (2 ; 5)" ;

"(3 ; 5), (5 ; 6), (2 ; 5), (1 ; 2), (2 ; 4)" ;

"(3 ; 5), (5 ; 6), (2 ; 5), (1 ; 4), (2 ; 5)" ;

"(3 ; 5), (5 ; 6), (2 ; 5), (2 ; 4), (1 ; 4)" ;

"(3 ; 6), (1 ; 3), (3 ; 4), (2 ; 3), (3 ; 5)" ;

"(3 ; 6), (1 ; 3), (3 ; 4), (3 ; 5), (2 ; 5)" ;

"(3 ; 6), (1 ; 3), (3 ; 5), (2 ; 4), (3 ; 5)" ;

"(2 ; 5), (2 ; 3), (2 ; 4), (1 ; 4), (4 ; 6)" ; "(3 ; 6), (1 ; 3), (3 ; 5), (2 ; 5), (2 ; 4)" ;

"(2 ; 5), (2 ; 3), (2 ; 4), (1 ; 6), (2 ; 4)" ; "(3 ; 6), (1 ; 3), (3 ; 5), (4 ; 5), (2 ; 5)" ;

"(2 ; 5), (2 ; 3), (2 ; 4), (4 ; 6), (1 ; 4)" ; "(3 ; 6), (3 ; 5), (1 ; 4), (2 ; 3), (3 ; 5)" ;

"(2 ; 5), (2 ; 3), (2 ; 6), (1 ; 2), (2 ; 4)" ; "(3 ; 6), (3 ; 5), (1 ; 4), (3 ; 5), (2 ; 5)" ;

"(2 ; 5), (3 ; 6), (2 ; 5), (1 ; 2), (2 ; 4)" ; "(3 ; 6), (3 ; 5), (1 ; 5), (2 ; 4), (3 ; 5)" ;

"(2 ; 5), (3 ; 6), (2 ; 5), (1 ; 4), (2 ; 5)" ; "(3 ; 6), (3 ; 5), (1 ; 5), (2 ; 5), (2 ; 4)" ;

"(2 ; 5), (3 ; 6), (2 ; 5), (2 ; 4), (1 ; 4)" ; "(3 ; 6), (3 ; 5), (1 ; 5), (4 ; 5), (2 ; 5)" ;

"(3 ; 5), (1 ; 5), (2 ; 5), (2 ; 4), (4 ; 6)" ; "(3 ; 6), (3 ; 5), (2 ; 5), (1 ; 2), (2 ; 4)" ;

"(2 ; 5), (2 ; 3), (2 ; 6), (1 ; 4), (2 ; 5)" ; "(3 ; 6), (3 ; 5), (2 ; 5), (1 ; 4), (2 ; 5)" ;

"(2 ; 5), (2 ; 3), (2 ; 6), (2 ; 4), (1 ; 4)" ; "(3 ; 6), (3 ; 5), (2 ; 5), (2 ; 4), (1 ; 4)".

Solution du Problème "Fox and Holes" avec augmentation des possibilités du renard et du nombre d'observations, complexe

Initialement, le renard est en 1, 2, 3, 4, 5, 6 ou 7.

En observant en 1 et en 3, soit on le trouve, soit il était en 2, 4, 5, 6 ou 7 et peut donc atteindre 2, 3, 4, 5, 6 ou 7.

En observant ensuite en 3 et en 6, soit on le trouve, soit il était juste précédemment en 2, 4, 5 ou 7 et peut donc atteindre 2, 3, 4, 5 ou 7.

En observant ensuite en 3 et en 4, soit on le trouve, soit il était juste précédemment en 2, 5 ou 7 et peut donc atteindre 2, 3, 5 ou 7.

En observant ensuite en 2 et en 3, soit on le trouve, soit il était juste précédemment en 5 ou 7 et peut donc atteindre 3, 5 ou 7.

En observant ensuite en 3 et en 5, soit on le trouve, soit il était juste précédemment en 5 et peut donc atteindre 5 ou 7.

Enfin, en observant ensuite en 5 et en 7, on le trouve.

Cette stratégie se note "(1 ; 3), (3 ; 6), (3 ; 4), (2, 3), (3, 5), (5, 7)"

Les 102 stratégies pour trouver Fox en 6 jours sont :

"(1 ; 3), (3 ; 5), (4 ; 6), (2 ; 3), (3 ; 5), (5 ; 7)" ; "(3 ; 5), (1 ; 6), (3 ; 5), (2 ; 4), (3 ; 5), (5 ; 7)" ;

"(1 ; 3), (3 ; 5), (4 ; 6), (2 ; 3), (3 ; 7), (3 ; 5)" ; "(3 ; 5), (1 ; 6), (3 ; 5), (2 ; 4), (3 ; 7), (3 ; 5)" ;

"(1 ; 3), (3 ; 5), (4 ; 6), (3 ; 5), (2 ; 5), (5 ; 7)" ; "(3 ; 5), (1 ; 6), (3 ; 5), (4 ; 5), (2 ; 5), (5 ; 7)" ;

"(1 ; 3), (3 ; 5), (4 ; 6), (3 ; 5), (2 ; 7), (3 ; 5)" ; "(3 ; 5), (1 ; 6), (3 ; 5), (4 ; 5), (2 ; 7), (3 ; 5)" ;

"(1 ; 3), (3 ; 5), (4 ; 6), (3 ; 5), (5 ; 7), (2 ; 5)" ; "(3 ; 5), (1 ; 6), (3 ; 5), (4 ; 5), (5 ; 7), (2 ; 5)" ;

"(1 ; 3), (3 ; 5), (4 ; 6), (3 ; 7), (2 ; 3), (3 ; 5)" ; "(3 ; 5), (1 ; 6), (3 ; 5), (4 ; 7), (2 ; 3), (3 ; 5)" ;

"(1 ; 3), (3 ; 5), (4 ; 6), (3 ; 7), (3 ; 5), (2 ; 5)" ;

"(1 ; 3), (3 ; 5), (5 ; 6), (2 ; 4), (3 ; 5), (5 ; 7)" ;

"(1 ; 3), (3 ; 5), (5 ; 6), (2 ; 4), (3 ; 7), (3 ; 5)" ;

"(1 ; 3), (3 ; 5), (5 ; 6), (4 ; 5), (2 ; 5), (5 ; 7)" ;

"(1 ; 3), (3 ; 5), (5 ; 6), (4 ; 5), (2 ; 7), (3 ; 5)" ;

"(1 ; 3), (3 ; 5), (5 ; 6), (4 ; 5), (5 ; 7), (2 ; 5)" ;

"(1 ; 3), (3 ; 5), (5 ; 6), (4 ; 7), (2 ; 3), (3 ; 5)" (celle trouvée plus haut) ;

"(1 ; 3), (3 ; 5), (5 ; 6), (4 ; 7), (3 ; 5), (2 ; 5)" ;

"(1 ; 3), (3 ; 6), (3 ; 4), (2 ; 3), (3 ; 5), (5 ; 7)" ;

"(1 ; 3), (3 ; 6), (3 ; 4), (2 ; 3), (3 ; 7), (3 ; 5)" ;

"(1 ; 3), (3 ; 6), (3 ; 4), (3 ; 5), (2 ; 5), (5 ; 7)" ;

"(1 ; 3), (3 ; 6), (3 ; 4), (3 ; 5), (2 ; 7), (3 ; 5)" ;

"(1 ; 3), (3 ; 6), (3 ; 4), (3 ; 5), (5 ; 7), (2 ; 5)" ;

"(1 ; 3), (3 ; 6), (3 ; 4), (3 ; 7), (2 ; 3), (3 ; 5)" ;

"(1 ; 3), (3 ; 6), (3 ; 4), (3 ; 7), (3 ; 5), (2 ; 5)" ;

"(1 ; 3), (3 ; 6), (3 ; 5), (2 ; 4), (3 ; 5), (5 ; 7)" ;

"(1 ; 3), (3 ; 6), (3 ; 5), (2 ; 4), (3 ; 7), (3 ; 5)" ;

"(1 ; 3), (3 ; 6), (3 ; 5), (4 ; 5), (2 ; 5), (5 ; 7)" ;

"(1 ; 3), (3 ; 6), (3 ; 5), (4 ; 5), (2 ; 7), (3 ; 5)" ;

"(1 ; 3), (3 ; 6), (3 ; 5), (4 ; 5), (5 ; 7), (2 ; 5)" ;

"(1 ; 3), (3 ; 6), (3 ; 5), (4 ; 7), (2 ; 3), (3 ; 5)" ;

"(1 ; 3), (3 ; 6), (3 ; 5), (4 ; 7), (3 ; 5), (2 ; 5)" ;

"(2 ; 4), (4 ; 7), (4 ; 5), (3 ; 4), (1 ; 4), (4 ; 6)" ;

"(2 ; 4), (4 ; 7), (4 ; 5), (3 ; 4), (4 ; 6), (1 ; 4)" ;

"(3 ; 5), (1 ; 5), (4 ; 6), (2 ; 3), (3 ; 5), (5 ; 7)" ;

"(3 ; 5), (1 ; 5), (4 ; 6), (2 ; 3), (3 ; 7), (3 ; 5)" ;

"(3 ; 5), (1 ; 5), (4 ; 6), (3 ; 5), (2 ; 5), (5 ; 7)" ;

"(3 ; 5), (1 ; 5), (4 ; 6), (3 ; 5), (2 ; 7), (3 ; 5)" ;

"(3 ; 5), (1 ; 5), (4 ; 6), (3 ; 5), (5 ; 7), (2 ; 5)" ;

"(3 ; 5), (1 ; 5), (4 ; 6), (3 ; 7), (2 ; 3), (3 ; 5)" ;

"(3 ; 5), (1 ; 6), (3 ; 5), (4 ; 7), (3 ; 5), (2 ; 5)" ;

"(3 ; 5), (5 ; 6), (1 ; 4), (2 ; 3), (3 ; 5), (5 ; 7)" ;

"(3 ; 5), (5 ; 6), (1 ; 4), (2 ; 3), (3 ; 7), (3 ; 5)" ;

"(3 ; 5), (5 ; 6), (1 ; 4), (3 ; 5), (2 ; 5), (5 ; 7)" ;

"(3 ; 5), (5 ; 6), (1 ; 4), (3 ; 5), (2 ; 7), (3 ; 5)" ;

"(3 ; 5), (5 ; 6), (1 ; 5), (4 ; 5), (5 ; 7), (2 ; 5)" ;

"(3 ; 5), (5 ; 6), (1 ; 5), (4 ; 7), (2 ; 3), (3 ; 5)" ;

"(3 ; 5), (5 ; 6), (1 ; 5), (4 ; 7), (3 ; 5), (2 ; 5)" ;

"(3 ; 6), (1 ; 3), (3 ; 4), (2 ; 3), (3 ; 5), (5 ; 7)" ;

"(3 ; 5), (5 ; 6), (1 ; 4), (3 ; 5), (5 ; 7), (2 ; 5)" ;

"(3 ; 5), (5 ; 6), (1 ; 4), (3 ; 7), (2 ; 3), (3 ; 5)" ;

"(3 ; 5), (5 ; 6), (1 ; 4), (3 ; 7), (3 ; 5), (2 ; 5)" ;

"(3 ; 5), (5 ; 6), (1 ; 5), (2 ; 4), (3 ; 5), (5 ; 7)" ;

"(3 ; 5), (5 ; 6), (1 ; 5), (2 ; 4), (3 ; 7), (3 ; 5)" ;

"(3 ; 5), (5 ; 6), (1 ; 5), (4 ; 5), (2 ; 5), (5 ; 7)" ;

"(3 ; 5), (5 ; 6), (1 ; 5), (4 ; 5), (2 ; 7), (3 ; 5)" ;

"(3 ; 6), (1 ; 3), (3 ; 4), (2 ; 3), (3 ; 7), (3 ; 5)" ;

"(3 ; 6), (1 ; 3), (3 ; 4), (3 ; 5), (2 ; 5), (5 ; 7)" ;

"(3 ; 6), (1 ; 3), (3 ; 4), (3 ; 5), (2 ; 7), (3 ; 5)" ;

"(3 ; 6), (1 ; 3), (3 ; 4), (3 ; 5), (5 ; 7), (2 ; 5)" ;

"(3 ; 6), (1 ; 3), (3 ; 4), (3 ; 7), (2 ; 3), (3 ; 5)" ;

"(3 ; 6), (1 ; 3), (3 ; 4), (3 ; 7), (3 ; 5), (2 ; 5)" ;

"(3 ; 6), (1 ; 3), (3 ; 5), (2 ; 4), (3 ; 5), (5 ; 7)" ;

"(3 ; 6), (1 ; 3), (3 ; 5), (2 ; 4), (3 ; 7), (3 ; 5)" ;

"(3 ; 6), (1 ; 3), (3 ; 5), (4 ; 5), (2 ; 5), (5 ; 7)" ;

"(3 ; 6), (1 ; 3), (3 ; 5), (4 ; 5), (2 ; 7), (3 ; 5)" ;

"(3 ; 6), (1 ; 3), (3 ; 5), (4 ; 5), (5 ; 7), (2 ; 5)" ;

"(3 ; 6), (1 ; 3), (3 ; 5), (4 ; 7), (2 ; 3), (3 ; 5)" ;

"(3 ; 6), (1 ; 3), (3 ; 5), (4 ; 7), (3 ; 5), (2 ; 5)" ;

"(3 ; 6), (3 ; 5), (1 ; 4), (2 ; 3), (3 ; 5), (5 ; 7)" ;

"(3 ; 6), (3 ; 5), (1 ; 4), (2 ; 3), (3 ; 7), (3 ; 5)" ;

"(3 ; 5), (1 ; 5), (4 ; 6), (3 ; 7), (3 ; 5), (2 ; 5)" ;
"(3 ; 5), (1 ; 5), (5 ; 6), (2 ; 4), (3 ; 5), (5 ; 7)" ;
"(3 ; 5), (1 ; 5), (5 ; 6), (2 ; 4), (3 ; 7), (3 ; 5)" ;
"(3 ; 5), (1 ; 5), (5 ; 6), (4 ; 5), (2 ; 5), (5 ; 7)" ;
"(3 ; 5), (1 ; 5), (5 ; 6), (4 ; 5), (2 ; 7), (3 ; 5)" ;
"(3 ; 5), (1 ; 5), (5 ; 6), (4 ; 5), (5 ; 7), (2 ; 5)" ;
"(3 ; 5), (1 ; 5), (5 ; 6), (4 ; 7), (2 ; 3), (3 ; 5)" ;
"(3 ; 5), (1 ; 5), (5 ; 6), (4 ; 7), (3 ; 5), (2 ; 5)" ;
"(3 ; 5), (1 ; 6), (3 ; 4), (2 ; 3), (3 ; 5), (5 ; 7)" ;
"(3 ; 5), (1 ; 6), (3 ; 4), (2 ; 3), (3 ; 7), (3 ; 5)" ;
"(3 ; 5), (1 ; 6), (3 ; 4), (3 ; 5), (2 ; 5), (5 ; 7)" ;
"(3 ; 5), (1 ; 6), (3 ; 4), (3 ; 5), (2 ; 7), (3 ; 5)" ;
"(3 ; 5), (1 ; 6), (3 ; 4), (3 ; 5), (5 ; 7), (2 ; 5)" ;
"(3 ; 5), (1 ; 6), (3 ; 4), (3 ; 7), (2 ; 3), (3 ; 5)" ;
"(3 ; 5), (1 ; 6), (3 ; 4), (3 ; 7), (3 ; 5), (2 ; 5)" ;
"(3 ; 6), (3 ; 5), (1 ; 4), (3 ; 5), (2 ; 5), (5 ; 7)" ;
"(3 ; 6), (3 ; 5), (1 ; 4), (3 ; 5), (2 ; 7), (3 ; 5)" ;
"(3 ; 6), (3 ; 5), (1 ; 4), (3 ; 5), (5 ; 7), (2 ; 5)" ;
"(3 ; 6), (3 ; 5), (1 ; 4), (3 ; 7), (2 ; 3), (3 ; 5)" ;
"(3 ; 6), (3 ; 5), (1 ; 4), (3 ; 7), (3 ; 5), (2 ; 5)" ;
"(3 ; 6), (3 ; 5), (1 ; 5), (2 ; 4), (3 ; 5), (5 ; 7)" ;
"(3 ; 6), (3 ; 5), (1 ; 5), (2 ; 4), (3 ; 7), (3 ; 5)" ;
"(3 ; 6), (3 ; 5), (1 ; 5), (4 ; 5), (2 ; 5), (5 ; 7)" ;
"(3 ; 6), (3 ; 5), (1 ; 5), (4 ; 5), (2 ; 7), (3 ; 5)" ;
"(3 ; 6), (3 ; 5), (1 ; 5), (4 ; 5), (5 ; 7), (2 ; 5)" ;
"(3 ; 6), (3 ; 5), (1 ; 5), (4 ; 7), (2 ; 3), (3 ; 5)" ;
"(3 ; 6), (3 ; 5), (1 ; 5), (4 ; 7), (3 ; 5), (2 ; 5)" ;
"(4 ; 7), (2 ; 4), (4 ; 5), (3 ; 4), (1 ; 4), (4 ; 6)" ;
"(4 ; 7), (2 ; 4), (4 ; 5), (3 ; 4), (4 ; 6), (1 ; 4)".

Annexe C

Programmation en C++

Programme de résolution

Vous trouverez ci-dessous un code source en C++ qui, une fois compilé, est capable de calculer les solutions pour N terriers (N < 32), X observations (X < N) et M déplacements du renard (M < N).

```
#include <iostream>
#include <vector>
using namespace std ;
typedef unsigned int uint ;
struct terriers {
uint PresenceRenard ;          /// champ de bits : 0 absence certaine du renard
static uint size ;    /// nbre de terriers pour le renard et l'observateur (<32)
vector<uint> histoire ;              /// histoire des observations successives
static vector<int> MvtRenard ;   /// déplacements rel. du renard ds les terriers
};
```

```
uint terriers::size = 0 ;
vector<int> terriers::MvtRenard ;
/** Tableau à deux dimensions (jours x tableau de terriers) avec les présences
de renard à cardinalité minimale et unique pour chaque jour                **/
vector<vector<terriers>> CardMin(1,vector<terriers>()) ;
/** retourne le nbre de terriers occupables par le renard (~ popcnt)        **/
uint PossiblementOccupes(const terriers&) ;
/** retourne une structure terriers après l'application des déplacements possi-
bles du renard juste après les observations (paramètre passé en argument)    **/
terriers DeplaceRenard(const terriers&) ;
/** fonct. recursive appliquant toutes les observations dans les terriers **/
void AjouterObservation(const terriers&, uint, uint,const uint,const uint) ;
/// *** Fonction principale ***************************************************
int main(void)
{
/** -- La saisie des données d'entrée (aucun contrôle de validité) --------- **/
cout << "Des terriers et un renard" << endl ;
cout << "Donnez le nombre de terriers (<32) : " ;
cin >> terriers::size ;             /// saisie du nombre de terriers accessibles
cout << "Donnez le nombre d'observations (<" << terriers::size << ") : " ;
uint NbreObservations ;             /// nombre d'observations possibles par jour.
cin >> NbreObservations ;
uint NbrDeplacements ;
cout << "Donnez le nombre de deplacements possibles du renard (<"
    << terriers::size << ") : " ;
cin >> NbrDeplacements ;
cout << "Donnez le(s) " << NbrDeplacements << " deplacements un a un : " ;
while (terriers::MvtRenard.size() < NbrDeplacements) {
  int d ;
  cin >> d ;
  terriers::MvtRenard.push_back(d) ;
  }
cout << "Voulez-vous afficher les solutions (o/n) : " ;
string rep ;
cin >> rep ;
```

```
bool AfficheSol = (rep == "o") ;      /// (des-)active l'affichage des solutions
/** --- Résumé des données d'entrée avant de débuter le calcul ---------- **/
cout << "> " << terriers::size << " terriers." << endl
<< "> " << NbreObservations << " observations par jour." << endl
<< "> " << terriers::MvtRenard.size() << " deplacements autorises du renard :" ;
for (int d : terriers::MvtRenard) cout << " " << d  ;
cout << endl ;
/** --- Début des calculs ------------------------------------------------ **/
uint day(0U) ;                  /// permet de compter les jours d'observations.
CardMin.at(day).push_back(terriers{0xFFFFFFFFU}) ; /// tous les terriers occupés
/** tant que l'on pas atteint des terriers tous non occupés par le renard    **/
while (CardMin.at(day).begin()->PresenceRenard != 0x00000000U) {
 /** allocation du vecteur de terriers pour le jour suivant **/
 CardMin.push_back(vector<terriers>()) ;    /// allocation pour le jour suivant
 /** parcours de toutes les configurations de terriers à analyser           **/
 for (terriers t : CardMin.at(day)) {
  /** pour la config de terriers en cours, on fait les observations puis les
  déplacements de renards en commançant par le terrier 0 et l'observation 1**/
  AjouterObservation(t, 0, 1,NbreObservations, day) ;
  }
 day++ ;                                /// incrément du jours en cours
 }
/** --- Affichage des resultats ----------------------------------------- **/
cout << "Trouve en " << day << " jours." << endl ;
cout << "Nombre de solutions : " << CardMin.at(day).size() << endl ;
if (AfficheSol)
   for (terriers soluce: CardMin.at(day)) {
   cout << "<" ;
   for (uint i = 0 ; i < soluce.histoire.size() ; ++i) {
     cout << soluce.histoire.at(i) ;
     if (i <soluce.histoire.size() -1) cout << "," ;
     }
   cout << "> " ;
   }
cout << endl ;
```

```
return 0 ;
}

/// Permet d'effectuer une observation de terrier (le bit associé est mis à 0)
/// holes : configuration des terriers avant les observations
/// l_holes : configuration (locale) des terriers avec les observations
/// new_holes : configuration des terriers avec les observations et les déplace-
///     ments du renard destinée à être stocker si sa cardinalité est minimale.
/// TerriersOccMin : nombre minimal de terriers occupés par le renard (cardinal
///                  des bits à 1). Initialisé au nombre de terriers.
void AjouterObservation(const terriers &holes, uint h, uint obs,
                        const uint nbr_obs, const uint d)
{
static uint TerriersOccMin = terriers::size ;
terriers l_holes ;
/// avec une observation on scrute les différents terriers
for (uint num_ter = h ; num_ter < terriers::size ; ++num_ter) {
  /// Mise à 0 du bit correspondant au terrier observé
  l_holes.PresenceRenard = holes.PresenceRenard & ~(1<< num_ter) ;
  l_holes.histoire = holes.histoire ;
  /// comme on vient d'appliquer une observation on la rend historique
  l_holes.histoire.push_back(num_ter+1) ;
  /// s'il ne s'agit pas de la dernière observation on recommence à partir des
  /// terriers qui restent (num_ter+1) et l'observation suivante (obs +1)
  if (obs < nbr_obs) {
    AjouterObservation(l_holes, num_ter+1, obs+1, nbr_obs, d) ;
    }
  else {   /// ici toutes les observations ont généré une nouvelle configuration
    terriers new_holes = DeplaceRenard(l_holes) ; /// alors on déplace le renard
    /// on compte le nombre de terriers occupables
    uint NbreTerriersOccupes = PossiblementOccupes(new_holes) ;
    if (NbreTerriersOccupes < TerriersOccMin) {   /// si cardinalité plus faible
      CardMin.at(d+1).clear() ;          /// on efface le stockage déjà effectué
      TerriersOccMin = NbreTerriersOccupes ;   /// on update la cardinalité min.
      }
```

```
    /// si la cardinalité des terriers occupables est = celles stockées
    if (NbreTerriersOccupes == TerriersOccMin)
      CardMin.at(d+1).push_back(new_holes);   /// on rajoute la config. en cours
    }
  }
return ;
}
/// Retourne le nombre de terriers possiblement occupés (Cardinalité des 1)
uint PossiblementOccupes(const terriers &t)
{
uint occupes = 0 ;
for (uint nb_ter = 0; nb_ter < terriers::size ; ++nb_ter)
  if ((t.PresenceRenard & (1U<<nb_ter)) != 0)
    occupes++ ;
return occupes ;
}
/// Déplace le renard selon les mvts autorisés et à partir des terriers occupés
/// holes : configuration des terriers après toutes les observations et avant
///         application de tous les déplacements autorisés du renard.
/// new_holes : configuration des terriers après application de tous les dépla-
///             cements autorisés du renard. Retourné à la fonction appelante.
terriers DeplaceRenard(const terriers &holes)
{
terriers new_holes { 0x00000000U } ; /// aucun terrier n'est possiblement occupé
/// passage en revue de tous les terriers après observation(s)
for (uint num_ter = 0 ; num_ter < terriers::size ; ++num_ter) {
  /// présence possible d'un renard (bit associé au terrier différent de 0)
  if ((holes.PresenceRenard & (1U<<num_ter)) != 0) {
    /// passage en revue de tous les déplacements du renard
    for (int d : terriers::MvtRenard) {
      /// dans le respect des limites (terrier cible entre 0 et size-1)
      int cible = int(num_ter) + d ;
      if ((cible >= 0) and (uint(cible) < terriers::size)) {
        /// mise à 1 du bit correspondant au terrier cible
        new_holes.PresenceRenard = new_holes.PresenceRenard | (1U << cible) ;
```

```
    }    }    }    }
new_holes.histoire = holes.histoire ;   /// copie de l'histoire des observations
return new_holes ;
}
```

Exemple de résolution

Des terriers et un renard

Donnez le nombre de terriers (<32) : 7

Donnez le nombre d'observations (<7) : 2

Donnez le nombre de deplacements possibles du renard (<7) : 3

Donnez le(s) 3 deplacements un a un : -2 0 3

Voulez-vous afficher les solutions (o/n) : o

> 7 terriers.

> 2 observations par jour.

> 3 deplacements autorises du renard : -2 0 3

Trouve en 6 jours.

Nombre de solutions : 102

<1,3,3,5,4,6,2,3,3,5,5,7>	<1,3,3,5,4,6,2,3,3,7,3,5>	<1,3,3,5,4,6,3,5,2,5,5,7>
<1,3,3,5,4,6,3,5,2,7,3,5>	<1,3,3,5,4,6,3,5,5,7,2,5>	<1,3,3,5,4,6,3,7,2,3,3,5>
<1,3,3,5,4,6,3,7,3,5,2,5>	<1,3,3,5,5,6,2,4,3,5,5,7>	<1,3,3,5,5,6,2,4,3,7,3,5>
<1,3,3,5,5,6,4,5,2,5,5,7>	<1,3,3,5,5,6,4,5,2,7,3,5>	<1,3,3,5,5,6,4,5,5,7,2,5>
<1,3,3,5,5,6,4,7,2,3,3,5>	<1,3,3,5,5,6,4,7,3,5,2,5>	<1,3,3,6,3,4,2,3,3,5,5,7>
<1,3,3,6,3,4,2,3,3,7,3,5>	<1,3,3,6,3,4,3,5,2,5,5,7>	<1,3,3,6,3,4,3,5,2,7,3,5>
<1,3,3,6,3,4,3,5,5,7,2,5>	<1,3,3,6,3,4,3,7,2,3,3,5>	<1,3,3,6,3,4,3,7,3,5,2,5>
<1,3,3,6,3,5,2,4,3,5,5,7>	<1,3,3,6,3,5,2,4,3,7,3,5>	<1,3,3,6,3,5,4,5,2,5,5,7>
<1,3,3,6,3,5,4,5,2,7,3,5>	<1,3,3,6,3,5,4,5,5,7,2,5>	<1,3,3,6,3,5,4,7,2,3,3,5>
<1,3,3,6,3,5,4,7,3,5,2,5>	<2,4,4,7,4,5,3,4,1,4,4,6>	<2,4,4,7,4,5,3,4,4,6,1,4>
<3,5,1,5,4,6,2,3,3,5,5,7>	<3,5,1,5,4,6,2,3,3,7,3,5>	<3,5,1,5,4,6,3,5,2,5,5,7>
<3,5,1,5,4,6,3,5,2,7,3,5>	<3,5,1,5,4,6,3,5,5,7,2,5>	<3,5,1,5,4,6,3,7,2,3,3,5>
<3,5,1,5,4,6,3,7,3,5,2,5>	<3,5,1,5,5,6,2,4,3,5,5,7>	<3,5,1,5,5,6,2,4,3,7,3,5>
<3,5,1,5,5,6,4,5,2,5,5,7>	<3,5,1,5,5,6,4,5,2,7,3,5>	<3,5,1,5,5,6,4,5,5,7,2,5>
<3,5,1,5,5,6,4,7,2,3,3,5>	<3,5,1,5,5,6,4,7,3,5,2,5>	<3,5,1,6,3,4,2,3,3,5,5,7>
<3,5,1,6,3,4,2,3,3,7,3,5>	<3,5,1,6,3,4,3,5,2,5,5,7>	<3,5,1,6,3,4,3,5,2,7,3,5>

<3,5,1,6,3,4,3,5,5,7,2,5>	<3,5,1,6,3,4,3,7,2,3,3,5>	<3,5,1,6,3,4,3,7,3,5,2,5>
<3,5,1,6,3,5,2,4,3,5,5,7>	<3,5,1,6,3,5,2,4,3,7,3,5>	<3,5,1,6,3,5,4,5,2,5,5,7>
<3,5,1,6,3,5,4,5,2,7,3,5>	<3,5,1,6,3,5,4,5,5,7,2,5>	<3,5,1,6,3,5,4,7,2,3,3,5>
<3,5,1,6,3,5,4,7,3,5,2,5>	<3,5,5,6,1,4,2,3,3,5,5,7>	<3,5,5,6,1,4,2,3,3,7,3,5>
<3,5,5,6,1,4,3,5,2,5,5,7>	<3,5,5,6,1,4,3,5,2,7,3,5>	<3,5,5,6,1,4,3,5,5,7,2,5>
<3,5,5,6,1,4,3,7,2,3,3,5>	<3,5,5,6,1,4,3,7,3,5,2,5>	<3,5,5,6,1,5,2,4,3,5,5,7>
<3,5,5,6,1,5,2,4,3,7,3,5>	<3,5,5,6,1,5,4,5,2,5,5,7>	<3,5,5,6,1,5,4,5,2,7,3,5>
<3,5,5,6,1,5,4,5,5,7,2,5>	<3,5,5,6,1,5,4,7,2,3,3,5>	<3,5,5,6,1,5,4,7,3,5,2,5>
<3,6,1,3,3,4,2,3,3,5,5,7>	<3,6,1,3,3,4,2,3,3,7,3,5>	<3,6,1,3,3,4,3,5,2,5,5,7>
<3,6,1,3,3,4,3,5,2,7,3,5>	<3,6,1,3,3,4,3,5,5,7,2,5>	<3,6,1,3,3,4,3,7,2,3,3,5>
<3,6,1,3,3,4,3,7,3,5,2,5>	<3,6,1,3,3,5,2,4,3,5,5,7>	<3,6,1,3,3,5,2,4,3,7,3,5>
<3,6,1,3,3,5,4,5,2,5,5,7>	<3,6,1,3,3,5,4,5,2,7,3,5>	<3,6,1,3,3,5,4,5,5,7,2,5>
<3,6,1,3,3,5,4,7,2,3,3,5>	<3,6,1,3,3,5,4,7,3,5,2,5>	<3,6,3,5,1,4,2,3,3,5,5,7>
<3,6,3,5,1,4,2,3,3,7,3,5>	<3,6,3,5,1,4,3,5,2,5,5,7>	<3,6,3,5,1,4,3,5,2,7,3,5>
<3,6,3,5,1,4,3,5,5,7,2,5>	<3,6,3,5,1,4,3,7,2,3,3,5>	<3,6,3,5,1,4,3,7,3,5,2,5>
<3,6,3,5,1,5,2,4,3,5,5,7>	<3,6,3,5,1,5,2,4,3,7,3,5>	<3,6,3,5,1,5,4,5,2,5,5,7>
<3,6,3,5,1,5,4,5,2,7,3,5>	<3,6,3,5,1,5,4,5,5,7,2,5>	<3,6,3,5,1,5,4,7,2,3,3,5>
<3,6,3,5,1,5,4,7,3,5,2,5>	<4,7,2,4,4,5,3,4,1,4,4,6>	<4,7,2,4,4,5,3,4,4,6,1,4>

La première solution <1,3,3,5,4,6,2,3,3,5,5,7> indique que le premier jour l'observateur a visité les terriers 1 et 3, puis le deuxième jour les terriers 3 et 5 et ainsi de suite jusqu'au sixième jour avec les terriers 5 et 7 pour débusquer le renard, quels que soient son terrier d'origine et ses déplacements successifs dans l'ensemble {-2,0,3}.

Remerciements

Pour la relecture largement annotée, pour les commentaires tant sur la forme que sur le fond et même pour le contenu, nous tenons à remercier cordialement Philippe Colliard.

Bibliographie

[MSE] Mathematics Stack Exchange ; Devinette du renard et des trous avec *n* trous (consulté le 17 octobre 2023) ; url : https://math.stackexchange.com/questions/2961942/riddle-of-the-fox-and-the-holes-with-n-holes

[GTJ] Gateways to joy ; casse-tête ; Renard dans un trou (consulté le 17 octobre 2023) ; url : https://gurmeet.net/puzzles/fox-in-a-hole/

[PSE] Puzzling Stack Exchange ; Pourquoi cette solution garantit-elle que le prince frappe à la bonne porte pour retrouver la princesse ? (consulté le 17 octobre 2023) ; url : https://puzzling.stackexchange.com/questions/455/why-does-this-solution-guarantee-that-the-prince-knocks-on-the-right-door-to-fin

[LY,MA] Logically yours, Mohammed Ammar ; Seemingly impossible fox puzzle ; fox in a hole (consulté le 17 octobre 2023) ; url : https://www.youtube.com/watch?v=0Prp9n7XfP8