INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Exercices et solutions 2020

Années HarmoS 13/14/15

https://www.castor-informatique.ch/

Éditeurs :

Jean-Philippe Pellet, Elsa Pellet, Gabriel Parriaux,
Susanne Datzko, Fabian Frei, Juraj Hromkovič, Regula Lacher

www.svia-ssie-ssii.ch
schweizerischervereinfürinformatikind erausbildung//sociétésuissepourl'infor matiquedansl'enseignement//societàsviz zeraperl'informaticanell'insegnamento

Ont collaboré au Castor Informatique 2020

Susanne Datzko, Fabian Frei, Martin Guggisberg, Lucio Negrini, Gabriel Parriaux, Jean-Philippe Pellet

Cheffe de projet : Nora A. Escherle

Nous adressons nos remerciements pour le travail de développement des exercices du concours à :
Juraj Hromkovič, Michael Barot, Christian Datzko, Jens Gallenbacher, Dennis Komm, Regula Lacher, Peter Rossmanith : ETH Zurich, Ausbildunges- und Beratungszentrum für Informatikunterricht

Le choix des exercices a été fait en collaboration avec les organisateur de Bebras en Allemagne, Autriche, Hongrie, Slovaquie et Lithuanie. Nosu remercions en particulier :
Valentina Dagienė : Bebras.org
Wolfgang Pohl, Hannes Endreß, Ulrich Kiesmüller, Kirsten Schlüter, Michael Weigend : Bundesweite Informatikwettbewerbe (BWINF), Allemagne
Wilfried Baumann, Anoki Eischer : Österreichische Computer Gesellschaft
Gerald Futschek, Florentina Voboril : Technische Universität Wien
Zsuzsa Pluhár : ELTE Informatikai Kar, Hongrie
Michal Winzcer : Université Comenius de Bratislava, Slovaquie

La version en ligne du concours a été réalisée sur l'infrastructure cuttle.org. Nous remercions pour la bonne collaboration :
Eljakim Schrijvers, Justina Dauksaite, Arne Heijenga, Dave Oostendorp, Andrea Schrijvers, Alieke Stijf, Kyra Willekes : cuttle.org, Pays-Bas
Chris Roffey : Université d'Oxford, Royaume-Uni

Pour le support pendant les semaines du concours, nous remercions en plus :
Hanspeter Erni : Direction, école secondaire de Rickenbach
Gabriel Thullen : Collège des Colombières
Beat Trachsler : Kantonsschule Kreuzlingen

Christoph Frei : Chragokyberneticks (Logo Castor Informatique Suisse)
Dr. Andrea Leu, Maggie Winter, Brigitte Manz-Brunner : Senarclens Leu + Partner AG

La version allemande des exercices a également été utilisée en Allemagne et en Autriche.

L'adaptation française a été réalisée par Elsa Pellet et l'adaptation italienne par Christian Giang.

INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Le Castor Informatique 2020 a été réalisé par la Société Suisse de l'Informatique dans l'Enseignement SSIE et soutenu par la Fondation Hasler.

HASLERSTIFTUNG

Cette brochure a été produite le 9 septembre 2021 avec le système de composition de documents LaTeX. Nous remercions Christian Datzko pour le développement et maintien de la structure de génération des 36 versions de cette brochure (selon les langues et les degrés). La structure actuelle a été mise en place de manière similaire à la structure précédente, qui a été développée conjointement avec Ivo Blöchliger dès 2014. Nous remercions aussi Jean-Philippe Pellet pour le développement de la série d'outils `bebras`, qui est utilisée depuis 2020 pour la conversion des documents source depuis les formats Markdown et YAML.

Tous les liens dans les tâches ci-après ont été vérifiés le 1er décembre 2020.

Préambule

Très bien établi dans différents pays européens et plus largement à l'échelle mondiale depuis plusieurs années, le concours « Castor Informatique » a pour but d'éveiller l'intérêt des enfants et des jeunes pour l'informatique. En Suisse, le concours est organisé en allemand, en français et en italien par la SSIE, la Société Suisse pour l'Informatique dans l'Enseignement, et soutenu par la Fondation Hasler dans le cadre du programme d'encouragement « FIT in IT ».

Le Castor Informatique est le partenaire suisse du concours « Bebras International Contest on Informatics and Computer Fluency » (`https://www.bebras.org/`), initié en Lituanie.

Le concours a été organisé pour la première fois en Suisse en 2010. Le Petit Castor (années HarmoS 5 et 6) a été organisé pour la première fois en 2012.

Le Castor Informatique vise à motiver les élèves à apprendre l'informatique. Il souhaite lever les réticences et susciter l'intérêt quant à l'enseignement de l'informatique à l'école. Le concours ne suppose aucun prérequis quant à l'utilisation des ordinateurs, sauf de savoir naviguer sur Internet, car le concours s'effectue en ligne. Pour répondre, il faut structurer sa pensée, faire preuve de logique mais aussi de fantaisie. Les exercices sont expressément conçus pour développer un intérêt durable pour l'informatique, au-delà de la durée du concours.

Le concours Castor Informatique 2020 a été fait pour cinq tranches d'âge, basées sur les années scolaires :

- Années HarmoS 5 et 6 (Petit Castor)
- Années HarmoS 7 et 8
- Années HarmoS 9 et 10
- Années HarmoS 11 et 12
- Années HarmoS 13 à 15

Les élèves des années HarmoS 5 et 6 avaient 9 exercices à résoudre : 3 faciles, 3 moyens, 3 difficiles. Les élèves des années HarmoS 7 et 8 avaient, quant à eux, 12 exercices à résoudre (4 de chaque niveau de difficulté). Finalement, chaque autre tranche d'âge devait résoudre 15 exercices (5 de chaque niveau de difficulté).

Chaque réponse correcte donnait des points, chaque réponse fausse réduisait le total des points. Ne pas répondre à une question n'avait aucune incidence sur le nombre de points. Le nombre de points de chaque exercice était fixé en fonction du degré de difficulté :

	Facile	**Moyen**	**Difficile**
Réponse correcte	6 points	9 points	12 points
Réponse fausse	−2 points	−3 points	−4 points

Utilisé au niveau international, ce système de distribution des points est conçu pour limiter le succès en cas de réponses données au hasard.

Chaque participant·e obtenait initialement 45 points (ou 27 pour la tranche d'âge « Petit Castor », et 36 pour les années HarmoS 7 et 8).

Le nombre de points maximal était ainsi de 180 (ou 108 pour la tranche d'âge « Petit Castor », et 144 pour les années HarmoS 7 et 8). Le nombre de points minimal était zéro.

Les réponses de nombreux exercices étaient affichées dans un ordre établi au hasard. Certains exercices ont été traités par plusieurs tranches d'âge.

Pour de plus amples informations :

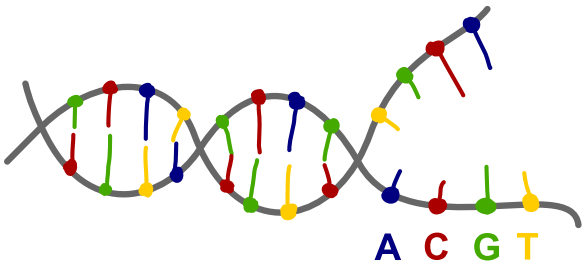
SVIA-SSIE-SSII Société Suisse de l'Informatique dans l'Enseignement
Castor Informatique
Gabriel Parriaux

`https://www.castor-informatique.ch/fr/kontaktieren/`
`https://www.castor-informatique.ch/`

Table des matières

1. Séquence ADN

Notre patrimoine génétique est enregistré sous forme de séquences d'ADN. Une séquence d'ADN est essentiellement une suite de bases dont quatre formes existent : A, C, D et T.



Nous considérons les trois sortes de mutations suivantes :

Mutation	**Description**	**Exemple**
Substitution	Une base est remplacée par une autre.	AT**G**GT → AT**A**GT
Délétion	Une base est perdue sans être remplacée.	ATG**G**T → ATGT
Insertion	Une base est ajoutée dans une séquence.	ATGGT → A**C**TGGT

*Il y a exactement une des séquences suivantes qui ne peut **pas** être générée par trois mutations de la séquence GTATCG. Laquelle est-ce ?*

A) GCAATG
B) ATTATCCG
C) GAATGC
D) GGTAAAC

Solution

La bonne réponse est D) GGTAAAC.

La meilleure méthode pour trouver la réponse est de procéder par élimination, étant donné que les trois autres séquences peuvent résulter de trois mutations.

Réponse A : GTATCG ⇒ G**C**ATCG ⇒ GCA**A**CG ⇒ GCAATG
Réponse B : GTATCG ⇒ **A**TATCG ⇒ AT**T**ATCG ⇒ ATTAT**C**CG
Réponse C : GTATCG ⇒ G**A**ATCG ⇒ GAAT**G**G ⇒ GAATG**C**

En comparaison, il faut quatre mutations pour obtenir la séquence de la réponse D), par exemple celles-ci :

GTATCG ⇒ G**G**TATCG ⇒ GGT**A**ATCG ⇒ GGTAA**A**CG ⇒ GGTAAAC

Ce n'est pas facile de prouver que moins de quatre mutations ne sont pas suffisantes.

C'est de l'informatique !

La représentation d'information à l'aide de *chaînes de caractères* (des séquences de lettres) et leur utilisation est une tâche centrale de l'informatique.

Une question importante est de déterminer quel est le degré de différence entre deux chaînes de caractères. Il existe plusieurs méthodes pour mesurer le degré de différence entre deux chaînes de caractères. Une méthode fréquemment utilisée est la *distance de Levenshtein*, qui est définie à base des trois sortes de mutations décrites plus haut : la distance de Levenshtein entre deux chaînes de caractères est le nombre minimal de mutations permettant de transformer une chaîne en l'autre.

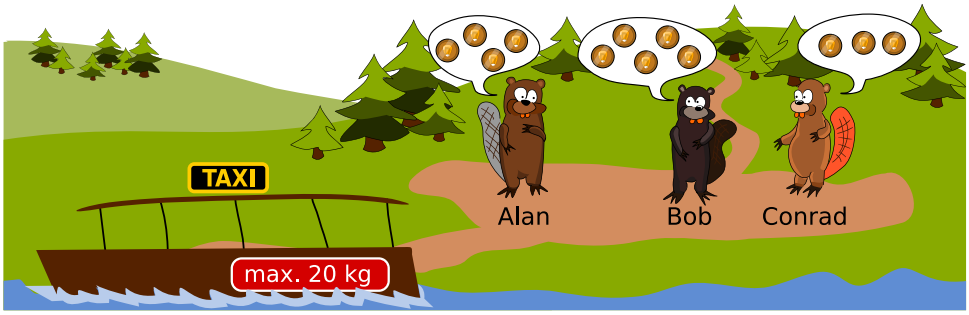
L'algorithme courant utilisé pour calculer la distance de Levenshtein se base sur la *programmation dynamique* : la distance de Levenshtein entre des préfixes toujours plus longs des deux chaînes de caractères sont inscrites dans un tableau jusqu'à ce que les préfixes correspondent aux mots entiers et que l'on puisse lire les résultats dans la table.

Lorsque l'exactitude de l'algorithme est prouvée, on peut calculer que la distance entre la séquence d'origine et celle de la réponse D) est exactement 4. On a ainsi prouvé que moins de quatre mutations ne suffisent pas.

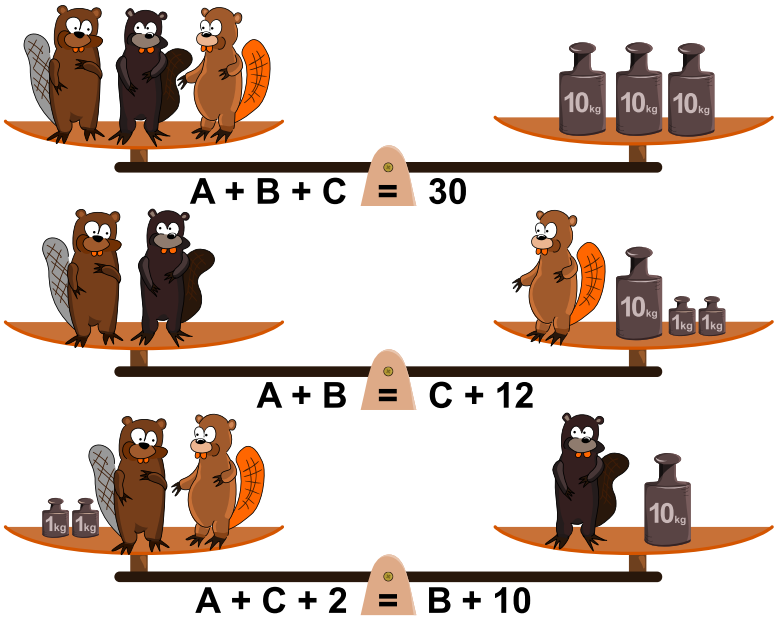
Mots clés et sites web

- Distance de Levenshtein : `https://fr.wikipedia.org/wiki/Distance_de_Levenshtein`
- `https://fr.wikipedia.org/wiki/Chaîne_de_caractères`

2. Bateau-taxi



Les trois castors Alan, Bob et Conrad veulent prendre un bateau-taxi. Il n'y a qu'un bateau-taxi. Alan est prêt à payer 4 francs castor (4×), Bob 5 francs castor (5×) et Conrad seulement 3 francs castor (3×). Le taxi peut transporter au maximum 20 kg. Le chauffeur de taxi fait donc les pesées suivantes :



Quel(s) castor(s) le chauffeur prend-il avec s'il veut gagner le plus d'argent possible ?

A) Seulement Bob
B) Alan et Bob
C) Bob et Conrad
D) Alan et Conrad
E) Tous les trois : Alan, Bob et Conrad

Solution

La bonne réponse est C) Bob et Conrad.

Pour pouvoir faire une liste de toutes les solutions possibles et les évaluer, nous devons d'abord savoir combien pèse chaque castor.

Nous savons que les trois castors ensemble pèsent 30 kg et que le chauffeur ne peut donc pas tous les prendre avec. Si nous ajoutons un copie de C(onrad) du côté gauche et du côté droit de la deuxième balance, cela donne à gauche $A + B + C = 30$ kg et à droite $C + C + 12$ kg. Donc, nous avons $2C = 18$ kg et $C = 9$ kg.

Si nous ajoutons un copie de B(ob) du côté gauche et du côté droit de la troisième balance, nous obtenons à gauche $A + B + C + 2$ kg $= 32$ kg et à droite $2B + 10$ kg. Cela donne $2B = 22$ kg et donc $B = 11$ kg.

Comme $A + B + C = 30$ kg, $A = 10$ kg.

Le chauffeur de taxi peut donc :

- Prendre Alan et Conrad avec et gagner $4 + 3 = 7$ francs castor.
- Prendre Bob et Conrad avec et gagner $5 + 3 = 8$ francs castor.
- Prendre Alan et Bob avec et gagner le plus avec 9 francs castors, mais comme les deux castors pèsent ensemble plus de 21 kg, le bateau-taxi est surchargé.

La bonne réponse est donc C).

Ce n'est cependant pas la seule possibilité de déterminer le poids des castors. On aurait aussi pu remplacer $A + B$ par $C + 12$ à gauche de la première balance. Ceci donne ensuite $2C + 12$ kg $= 30$ kg, et on peut en déduire que $C = 9$ kg.

De manière plus formelle, les trois pesées peuvent être écrite comme un système d'équations :

I. $A + B + C = 30$ kg
II. $A + B - C = 12$ kg
III. $A - B + C = 8$ kg

Ces équations peuvent ensuite être soustraites les une aux autres. La différence I. – II. donne l'équation :

$$2C = 18 \text{ kg} \rightarrow C = 9 \text{ kg}$$

La différence I. – III. donne :

$$2B = 22 \text{ kg} \rightarrow B = 11 \text{ kg}$$

On déduit ensuite de I. que $A = 10$ kg.

C'est de l'informatique !

Tous les problèmes d'optimisation discrète de la classe NP peuvent être représentés par des équations et des inéquations (on parle aussi de d'*optimisation linéaire*). Les équations et inéquations sont appelées *contraintes* et doivent être satisfaites par les valeurs des variables. On optimise ensuite la valeur d'une fonction des variables tout en respectant les contraintes. Dans cet exercice, on a trois variables booléennes, x_A, x_B et x_C. Si $x_A = 1$, le castor A prend le bateau, sinon $x_A = 0$. On optimise la fonction linéaire $4x_A + 5x_B + 3x_C$, pour laquelle on cherche la valeur maximale. La seule contrainte est :

$$Poids(A) \cdot x_A + Poids(B) \cdot x_B + Poids(C) \cdot x_C \leq 20.$$

On ne peut formuler l'exercice complètement que si l'on connaît le poids des castors. Cette instance de problème est un cas particulier du *problème du sac à dos*. On doit mettre la plus grande valeur possible dans le sac à dos sans dépasser la valeur maximale.

Il y a 80 ans, ce genre de questions était encore du ressort des mathématiciens, mais comme des ordinateurs de plus en plus performants ont été à disposition, des méthodes de résolution (par exemple la méthode de *séparation et évaluation* ou des *plans sécants*) avec lesquelles de tels problèmes peuvent être résolus ont été développées. Aujourd'hui, ces méthodes de résolution sont utilisées par exemple dans l'optimisation de la production, la logistique ou les réseaux de transport public.

Malgré tout, la résolution de problèmes d'optimisation est encore un exercice difficile en pratique qui demande une modélisation adroite et des algorithmes spécialement développés pour la structure et la taille du problème. Souvent, plusieurs méthodes de résolution sont combinées.

Mots clés et sites web

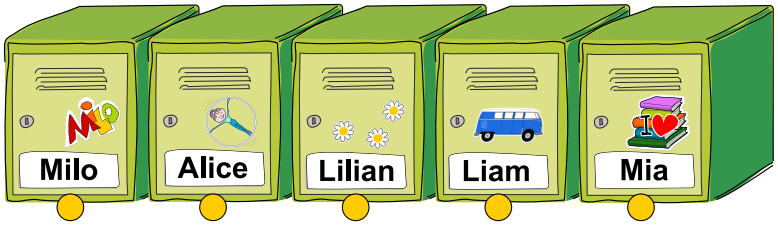
- Optimisation linéaire en nombre entiers : `https://fr.wikipedia.org/wiki/Optimisation_linéaire_en_nombres_entiers`
- Contrainte : `https://fr.wikipedia.org/wiki/Contrainte_(mathématiques)`
- Séparation et évaluation : `https://fr.wikipedia.org/wiki/Séparation_et_évaluation`
- Méthode des plans sécants : `https://fr.wikipedia.org/wiki/Méthode_des_plans_sécants`

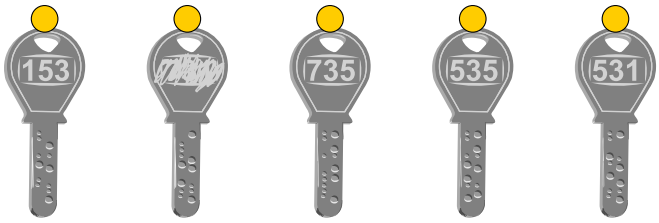
3. Casiers

Cinq enfants ont chacun un casier étiqueté à l'école. Un nombre à trois chiffres est gravé sur chacune des clés correspondantes. Malheureusement, le nombre sur l'une des clés est rayé.

Chaque nombre à trois chiffres représente les trois premières lettres d'un nom. Un chiffre représente toujours la même lettre, par exemple 8 pour « C » ou « c ».

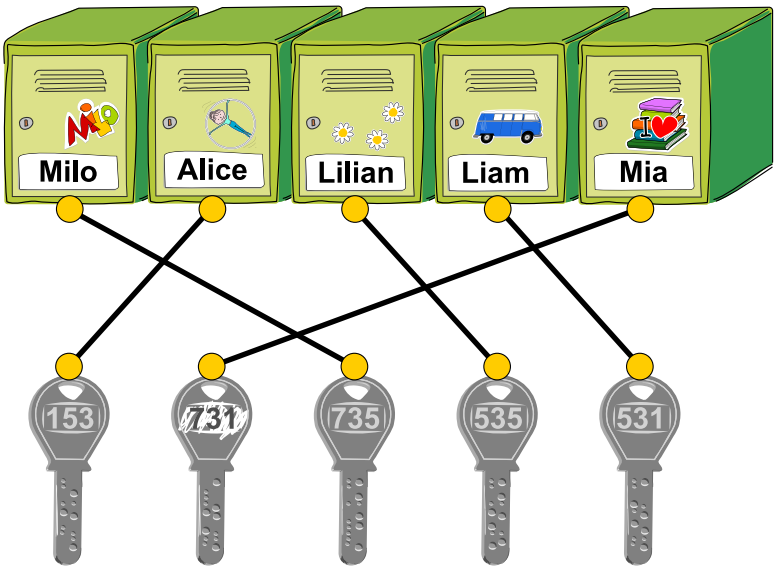
Relie les clés aux bons casiers. Pour cela, trace des lignes entre les points jaunes.





Solution

La bonne solution est illustrée ci-dessous :



Les quatre nombres connus sont 153, 735, 535 et 735. Les trois premières lettres des cinq noms sont MIL, ALI, LIL, LIA et MIA.

Il n'y a que LIL qui commence et se termine par la même lettre. Il doit donc y avoir un nombre à trois chiffres correspondant qui commence et se termine par le même chiffre, et il ne peut y avoir qu'un tel nombre. Le nombre 535 correspond à ce motif ; la clé 535 correspond donc à LIL. Cela veut dire que 5 représente L et 3 représente I. On peut maintenant voir que 531 doit correspondre à LIA, car il n'y a pas d'autre nom commençant par L. 1 représente donc la lettre A. De plus, 153 doit correspondre à ALI, car il n'y a pas d'autre nom avec un L en deuxième position. Il ne reste plus que le chiffre 7 et la lettre A qui ne sont pas attribués, ils doivent donc correspondre l'un à l'autre. On a ainsi l'attribution univoque 1 = A, 3 = I, 5 = L et 7 = M. 735 représente donc MIL et 531 LIA. On voit également que la clé avec le nombre rayé appartient à Mia et que ce nombre doit être 731.

Un méthode alternative pour trouver la bonne attribution est de compter la fréquence des chiffres et des lettres. Les lettres A et M apparaissent deux fois chacune dans MIL, ALI, LIL, LIA et MIA, et les lettres I et L cinq fois chacune. Malheureusement, cela ne suffit pas encore pour attribuer une lettre à chaque chiffre de manière univoque. On doit donc faire des observations supplémentaires telles que celles décrites plus haut.

C'est de l'informatique !

En informatique, les noms et les textes sont très souvent chiffrés à l'aide de nombres.

Dans la donnée de l'exercice, il est spécifié que l'on peut déduire les nombres sur les clés de manière univoque à partir des noms. Cela fonctionne car chaque lettre est chiffrée par exactement un chiffre et qu'il n'y a que peu de lettres. On parle d'un *chiffrement* (ou d'une *substitution*) *monoalphabétique*, car chaque lettre est toujours remplacée par le même symbole. Par contre, la donnée ne spécifie pas

quel chiffre correspond concrètement à quelle lettre. La solution montre cependant que l'on peut trouver la bonne attribution à l'aide de peu d'informations structurelles.

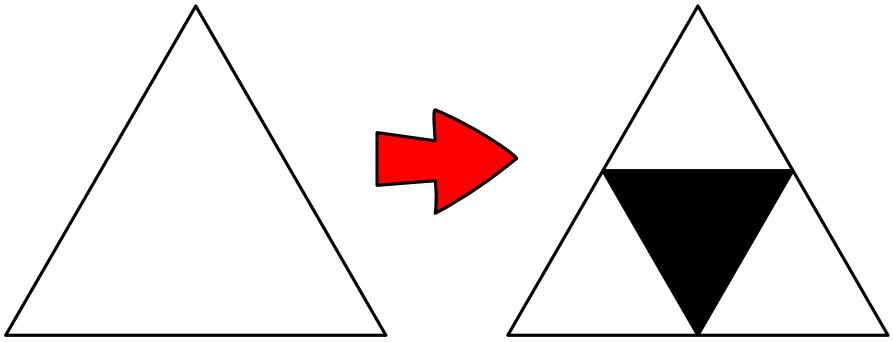
Si l'on n'utilise pas seulement dix chiffres pour le chiffrement, mais un symbole pour chaque lettre, on peut utiliser une telle substitution monoalphabétique comme un code secret simple. Malheureusement, la méthode de chiffrement par substitution monoalphabétique n'est pas très sûre, parce que l'on peut souvent déterminer l'attribution en utilisant quelques astuces. L'exercice en est un exemple. La *cryptographie* est un domaine important de l'informatique dans lequel des *chiffres* sont développés et analysés.

Mots clés et sites web

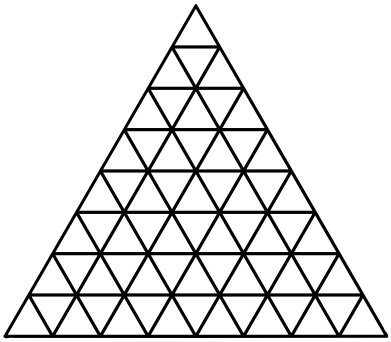
- Code, substitution monoalphabétique : `https://fr.wikipedia.org/wiki/Chiffrement_par_substitution#Substitution_monoalphabétique`
- Cryptographie : `https://fr.wikipedia.org/wiki/Cryptographie`

4. Triangle de Sierpiński

Pour obtenir un triangle de Sierpiński, on dessine d'abord un triangle équilatéral blanc, puis on procède étape par étape. À chaque étape, chaque triangle blanc existant est divisé en quatre triangles plus petits et celui du centre est coloré en noir, comme montré ci-dessous :

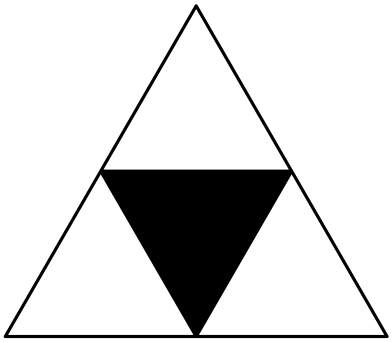


Dessine la figure obtenue après trois étapes. Pour cela, colorie les bons petits triangles en noir.

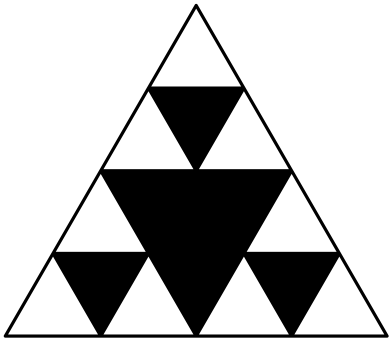


Solution

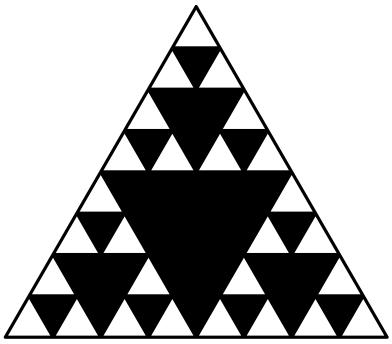
Après la première étape, le triangle central est noir et il y a trois triangles blancs :



Lors de la deuxième étape, les trois triangles blancs sont également divisés en quatre triangles plus petits, et chaque triangle central est coloré en noir. Il y a maintenant $3 \cdot 3 = 9$ petits triangles blancs :



Lors de la troisième et dernière étape, ces neufs triangles blancs sont à nouveau divisés en quatre triangles chacun, et chaque triangle central est coloré en noir. Il en résulte la figure suivante avec $3 \cdot 9 = 27$ triangles blancs :



C'est de l'informatique !

Le triangle de Sierpiński est une *fractale* qui a été décrite pour la première fois par le mathématicien polonais Waclaw Franciszek Sierpiński (1882–1969) en 1915. Les fractales sont des figures dans lesquelles apparaissent des éléments toujours plus petits, éléments qui sont semblables à la figure complète. C'est un travail très fastidieux de dessiner des images de fractales. Lorsque des ordinateurs

capable de faire les calculs nécessaires sont apparus au 20^e siècle, les fractales sont devenues très populaires. Le *flocon de Koch* et l'*ensemble de Mandelbrot* sont des fractales connues.

La construction du triangle de Sierpiński est récursive (du latin *re-currere* : se reproduire). Cela signifie que les règles de construction contiennent une instruction stipulant qu'il faut répéter l'application des règles. Dans cet exercice, cette instruction dit : « Divise le triangle blanc en quatre triangles plus petits, colore le triangle central en noir, puis répète cette instruction pour les triangles blancs résultants. » Une application de l'instruction s'appelle une *étape récursive*, et l'instruction demandant de réappliquer les règles s'appelle un *appel récursif* (dans l'exemple, il y a trois appels récursifs par étape récursive). Comme chaque appel récursif contient d'autres appels récursifs, on doit encore et toujours répéter l'étape récursive, ce qui peut durer indéfiniment. On peut éviter cela avec une condition de terminaison. Dans l'exemple, les appels récursifs s'arrêtent lorsque les triangles deviennent trop petits.

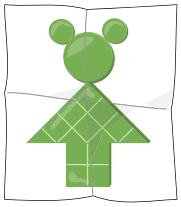
Le concept de la récursivité a un large domaine d'application en informatique, car beaucoup d'objects complexes – par exemple les fractales – peuvent être décrits de manière compacte grâce à la récursivité, et beaucoup de tâches complexes – par exemple les tours de Hanoï – peuvent être résolues à l'aide d'algorithmes récursifs très simples.

Mots clés et sites web

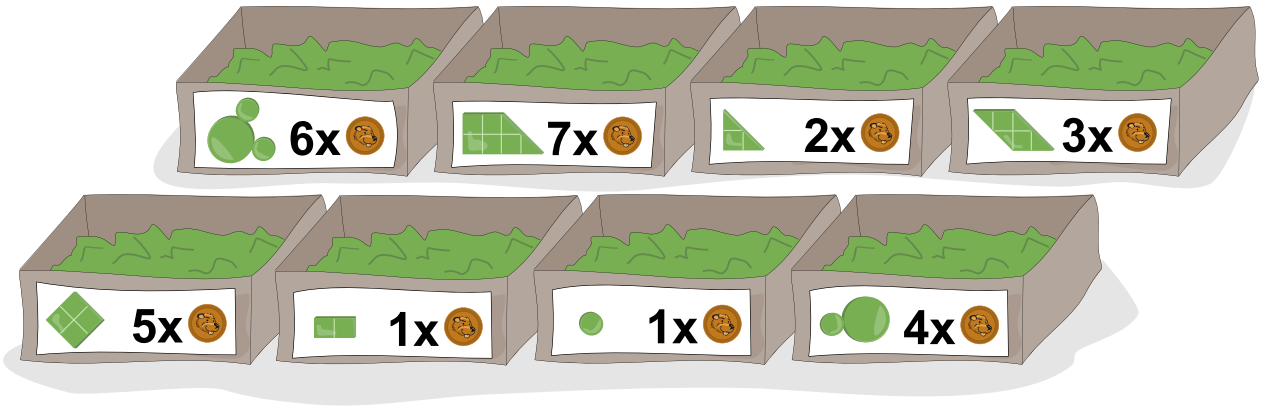
- Triangle de Sierpiński : `https://fr.wikipedia.org/wiki/Triangle_de_Sierpiński`
- Récursivité : `https://fr.wikipedia.org/wiki/Récursivité`
- Fractale : `https://fr.wikipedia.org/wiki/Fractale`
- `https://fr.wikipedia.org/wiki/Wacław_Sierpiński`
- `https://fr.wikipedia.org/wiki/Tours_de_Hanoï#Solution_récursive`
- `https://fr.wikipedia.org/wiki/Flocon_de_Koch`
- `https://fr.wikipedia.org/wiki/Ensemble_de_Mandelbrot`

5. Mosaïque

Giulia veut acheter des tesselles pour réaliser ce personnage en mosaïque :



Le magasin de jouets propose différentes tesselles en quantité au choix. Le prix par tesselle varie entre 1 et 7 pièces de monnaie.



Les tesselles peuvent être tournées comme désiré lors de l'assemblage, mais elles ne peuvent pas se chevaucher.

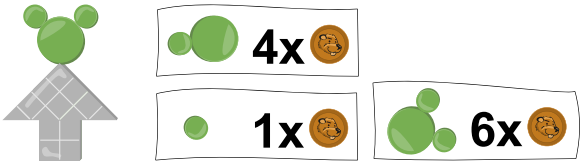
Combien de pièces de monnaie Giulia doit-elle dépenser si elle choisit l'option la moins chère ?

A) 13 pièces de monnaie
B) 14 pièces de monnaie
C) 16 pièces de monnaie
D) 20 pièces de monnaie

Solution

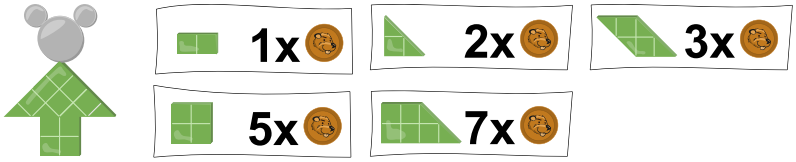
La bonne réponse est 13 pièces de monnaie.

L'une des méthodes pour résoudre cet exercice consiste à considérer séparément les différentes parties du personnage. Le plus simple est de commencer par la tête, qui ne peut être assemblée qu'avec des tesselles rondes :

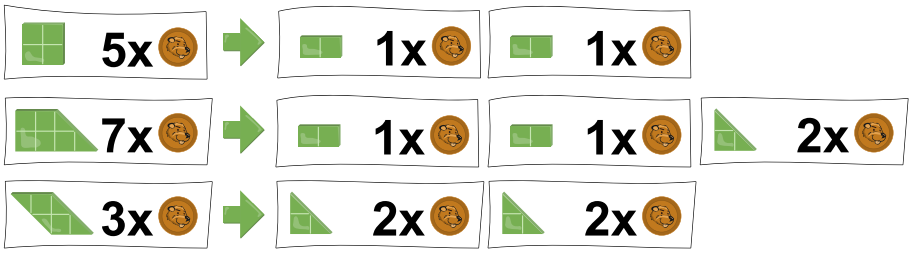


Il n'y a que deux possibilités pour assembler la tête : soit on utilise directement la tesselle adaptée pour 6 pièces de monnaie, soit on l'assemble à partir des deux autres tesselles ronde qui coûtent ensemble $4 + 1 = 5$ pièces de monnaie. Comme la deuxième option est moins chère, c'est celle que l'on utilise.

Le reste du personnage ne peut être assemblé qu'avec des tesselles anguleuses.



On pourrait maintenant essayer toutes les variantes possibles pour assembler le personnage et calculer le prix de chacune, mais ceci prend beaucoup de temps. Les observations suivantes concernant les tesselles anguleuse nous mènent rapidement à la solution :



- Une tesselle carrée coûtant 5 pièces de monnaie peut toujours être remplacée par deux rectangles à $1 + 1 = 2$ pièces de monnaie, ce qui est toujours moins cher.
- On pourrait également remplacer une tesselle carrée par deux tesselles triangulaires, ce qui serait un peu plus cher avec $2 + 2 = 4$ pièces de monnaies. C'est donc une moins bonne option.

Giulia n'achète donc jamais de carré même s'il irait bien à un certain endroit, mais toujours deux rectangles.

- Un trapèze coûtant 7 pièces de monnaie peut être assemblé avec un carré et un triangle. En remplaçant le carré par deux rectangles, on arrive à $1 + 1 + 2 = 4$ pièces de monnaie pour un trapèze.

Giulia n'achète donc jamais de trapèze même s'il irait bien à un endroit, mais l'assemble toujours à partir de deux rectangles et un triangle.

- On peut remplacer le parallélogramme à 3 pièces de monnaie par deux triangles à $2 + 2 = 4$ pièces de monnaie. Ce n'est pas une bonne option, car c'est plus cher que le parallélogramme. Un parallélogramme pourrait être utile à Giulia, mais cela n'est déterminé que par une analyse plus précise.

Version A		**Version B**	
	• Tête à 5 pièces de monnaie • Corps fait de 4 rectangles et 2 triangles : $1 + 1 + 1 + 1 + 2 + 2 = 8$ pièces		• Tête à 5 pièces de monnaie • Corps fait d'1 parallélogramme, 2 rectangles et 2 triangles : $3 + 1 + 1 + 2 + 2 = 9$ pièces

Si Giulia n'utilise pas de parallélogramme, elle a besoin de deux triangles pour assembler les pointes triangulaires à gauche et à droite du personnage. Elle peut assembler le reste avec des rectangles, comme dans la version A, qui coûte $5 + 8 = 13$ pièces de monnaie.

Le parallélogramme ne va qu'à un endroit du personnage comme montré dans la version B (ou reflété horizontalement). Si un parallélogramme est placé ainsi et que le reste de la figure est assemblé avec des rectangles et des triangles, le personnage coûte $5 + 9 = 14$ pièces de monnaie. Tous les autres placements du parallélogramme laisseraient des trous ne pouvant pas être remplis.

C'est donc clair que la version la moins chère coûte 13 pièces de monnaie.

C'est de l'informatique !

L'exercice consistant à assembler une figure précise avec des tesselles données peut vite devenir très compliqué même avec peu de pièces. Le jeu de puzzle Tangram en est un exemple.

Le problème ci-dessus est encore plus compliqué car il faut aussi optimiser le prix total des tesselles. En informatique, on appelle un tel problème un *problème d'optimisation*.

Le problème a été résolu grâce à un principe important en informatique : il s'agit de diviser un problème en problèmes plus petits que l'on peut résoudre indépendamment les uns des autres et dont les solutions peuvent se combiner pour obtenir la solution complète. Concrètement, le problème a été divisé en deux plus petits problèmes, un pour les tesselles rondes et un pour les tesselles anguleuse. Pour les tesselles anguleuses, on peut ensuite réutiliser partout la combinaison de tesselles la moins chère pour assembler un carré sans devoir y réfléchir à chaque fois. C'est la même chose pour le parallélogramme.

La division d'un problème en sous-problèmes indépendants est un concept très important en programmation. La réutilisation de solutions pour des problèmes apparaissant plusieurs fois permet de gagner beaucoup de temps. On parle ici du principe de *modularité*. La division en sous-problèmes est

aussi à la base des programmes suivant le principe *« diviser pour régner »* (« *divide et impera* » en latin, « *divide and conquer* » en anglais).

Mots clés et sites web

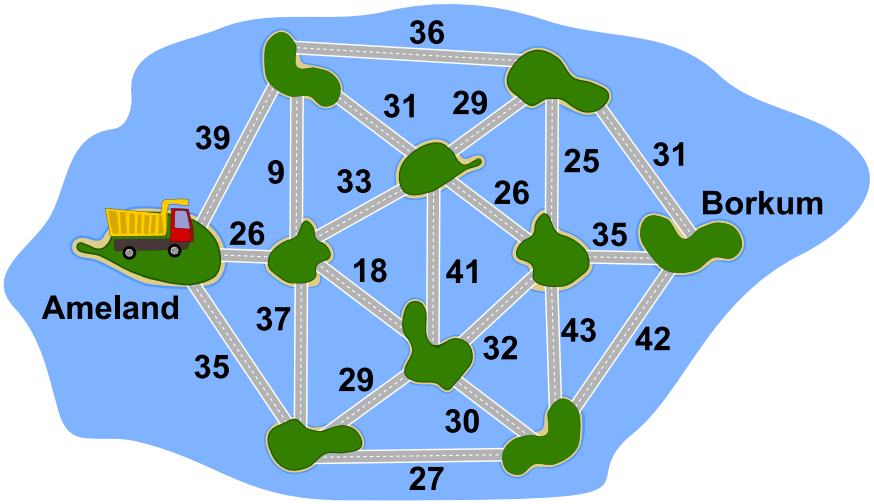
- Problème d'optimisation :
  `https://fr.wikipedia.org/wiki/Optimisation_(mathématiques)`
- Diviser pour régner :
  `https://fr.wikipedia.org/wiki/Diviser_pour_régner_(informatique)`
- Modularité : `https://fr.wikipedia.org/wiki/Notion_de_module`,
  `https://fr.wikipedia.org/wiki/Programmation_modulaire`
- Tangram : `https://fr.wikipedia.org/wiki/Tangram`

6. L'archipel des castors

Dans l'archipel des castors, il y a dix îles qui sont reliées par des ponts, comme sur la carte ci-dessous. Le nombre près de chaque pont indique le poids maximal en tonnes d'un camion pour qu'il puisse le traverser.

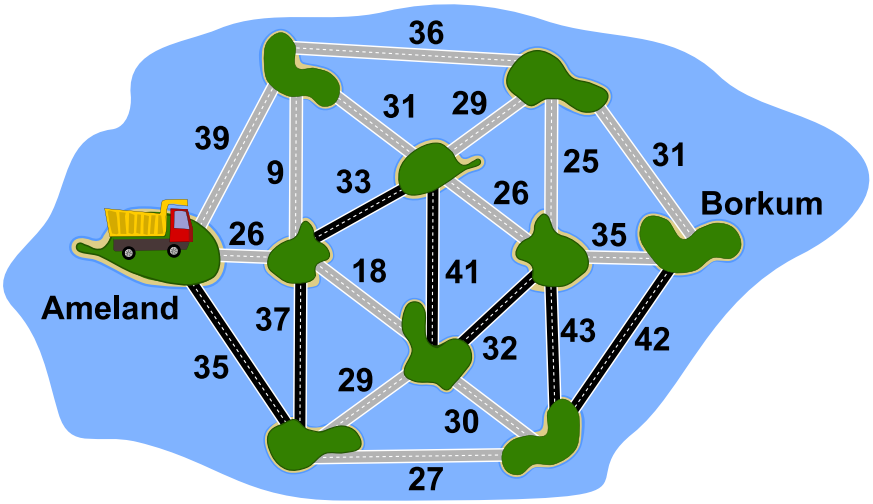
Le castor Knuth aimerait amener du sable sur une plage de l'île de Borkum. Il veut donc transporter autant de sable que possible de l'île d'Ameland à l'île de Borkum en un seul voyage. La longueur de la route à parcourir lui est égale, mais il ne veut prendre aucun pont plus d'une fois.

Quelle route devrait-il emprunter avec son camion pour atteindre Borkum ? Dessine-la sur la carte.

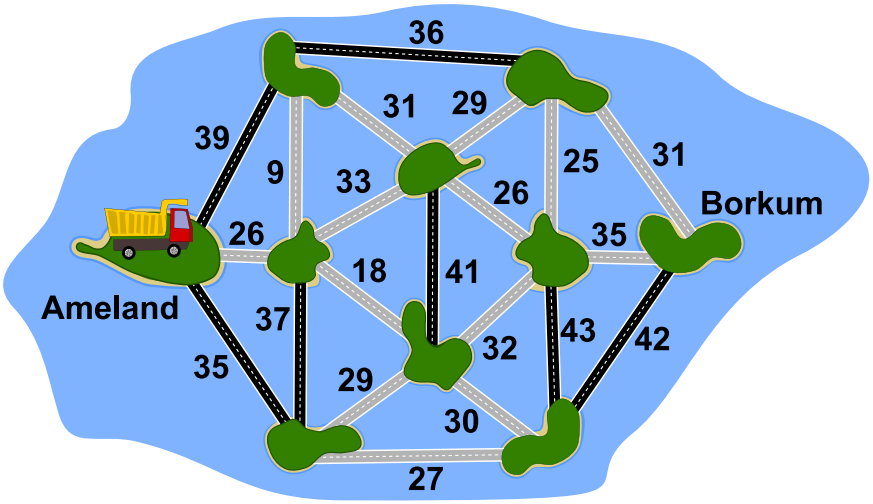


Solution

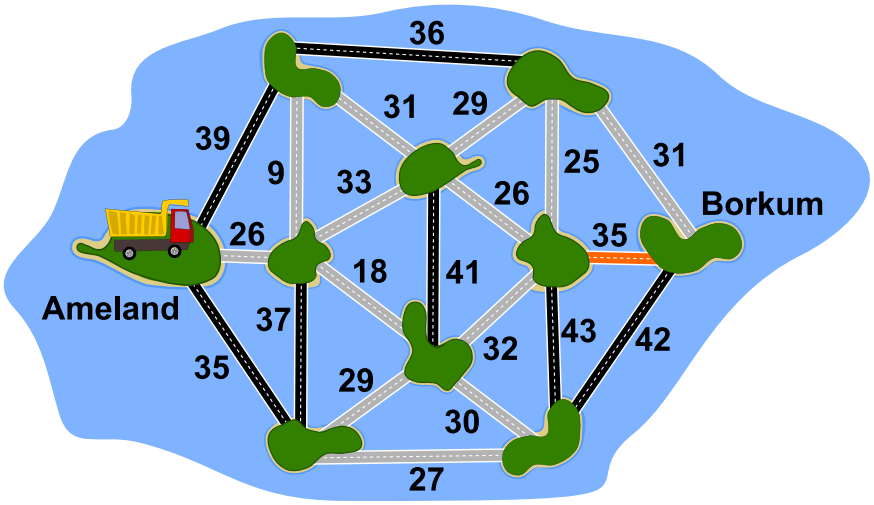
Le poids maximal d'un camion pour ce voyage est de 32 tonnes. Il suit par exemple la route suivante :



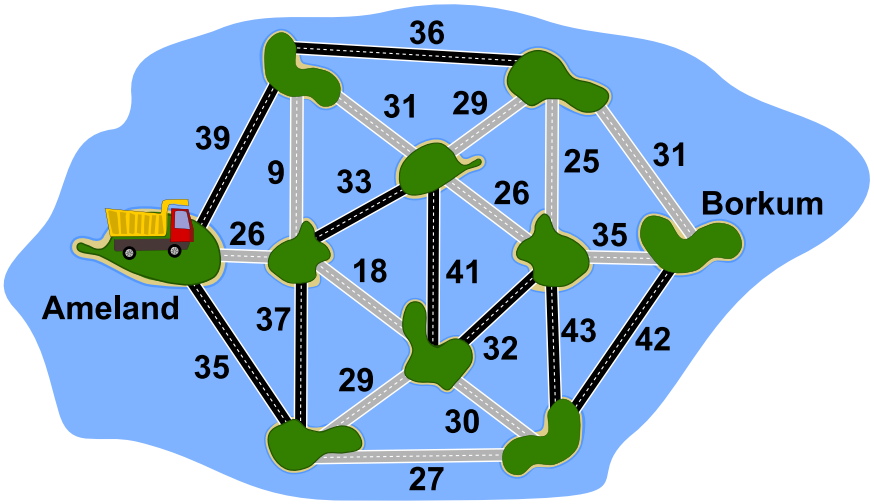
Afin de trouver ce chemin, nous pouvons par exemple commencer par virtuellement enlever tous les ponts de la carte. Nous les trions par capacité de charge. Nous commençons par ajouter à la carte les ponts ayant la plus grande capacité, puis ceux ayant une capacité de charge moindre, et ainsi de suite. Sur la carte suivante, les ponts avec une capacité de charge de 43, 42, 41, 39, 37, 36 et 35 tonnes sont indiqués en noir.



Si l'ajout d'un pont crée un cycle, donc un chemin qui tourne en rond, nous le laissons de côté, car toutes les îles de ce cycle peuvent déjà être atteintes en passant par des ponts avec une plus grande capacité. Dans le diagramme ci-dessous, le pont suivant avec une capacité de charge de 35 tonnes a été ajouté, mais il ne ferait que raccourcir une route déjà présente.



Nous répétons ce procédé jusqu'à ce que toutes les îles soient reliées. Il n'y a maintenant qu'une seule route possible entre chaque paire d'île et le pont avec la plus petite capacité de charge nous indique le poids maximal.



C'est de l'informatique !

Une application réelle de la solution de l'exercice de l'archipel des castors est d'identifier le « goulot d'étranglement » dans un réseau informatique, c'est-à-dire la vitesse de transfert maximale entre deux ordinateurs dans le réseau. Dans cet exercice, le goulot d'étranglement est le poids maximal d'un camion en route entre deux îles. Celui-ci est déterminé par la capacité de charge du pont le plus faible. Dans un réseau informatique, ce serait la connexion avec la plus petite bande passante.

Pour trouver une solution, on peut comme plus haut commencer par modéliser le réseau, donc le simplifier. Dans notre cas, l'*algorithme de Kruskal* est utilisé pour générer un *arbre couvrant* de poids maximal dans lequel le goulot d'étranglement est apparent.

Mots clés et sites web

- Graphe : `https://fr.wikipedia.org/wiki/Graphe_(mathématiques_discrètes)`
- Arbre couvrant de poids minimal : `https://fr.wikipedia.org/wiki/Arbre_couvrant`
- Algorithme de Kruskal : `https://fr.wikipedia.org/wiki/Algorithme_de_Kruskal`

7. Table incomplète

Les castors utilisent un code secret dans lequel chaque lettre est remplacée par un tout nouveau symbole. La table ci-dessous décrit comment les nouveaux symboles sont assemblés. Malheureusement, la table est incomplète car certaines parties ont été effacées.

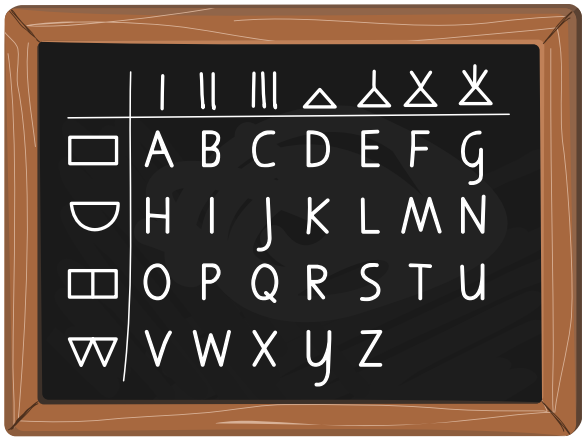
Reconstruis le texte original à partir du cryptogramme suivant (déchiffre le cryptogramme). Laquelle des quatre solutions proposées est-elle juste ?

A) INFORMATIQUE MALINE
B) ELECTRONIQUE MALINE
C) INFORMATION SECRETE
D) INFORMEZ EXACTEMENT

Solution

La bonne réponse est A), le texte clair est : INFORMATIQUE MALINE.

Voici la table de chiffrage complète :



C'est facile de compléter la table. Les lettres de l'alphabet latin sont écrites dans l'ordre, horizontalement et de gauche à droite. On remarque que la partie inférieure des nouveaux symboles correspond à l'intitulé des rangées et la partie supérieure à l'intitulé des colonnes de la table. La seule partie inférieure présente dans le cryptogramme qui manque dans la table est le ▭. C'est donc ce symbole qui est l'intitulé de la première rangée. On peut tout aussi rapidement déterminer les trois symboles manquants dans les colonnes.

Ce n'est cependant pas nécessaire de compléter la table. On peut placer les lettres que l'on peut directement lire dans la table incomplète. On obtient alors le texte à trous suivant :

I N _ O _ _ _ _ I _ _ _ _ _ L I N _

Ce texte à trous permet d'éliminer toutes les solutions sauf A) : B) ne commence pas par « IN », C) et D) ne finissent pas par « LIN_ ».

Une autre solution possible est de remarquer que le cryptogramme possède les deux mêmes symboles à son début et en avant-dernière position. La seule solution avec cette même répétition est la solution B).

C'est de l'informatique !

Garder des informations secrètes ou protéger des données est une tâche vielle de 4000 ans. D'innombrables écritures secrètes ont été développées et utilisées dans ce but. Aujourd'hui, la sécurité des données est l'un des thèmes majeurs de l'informatique. Une des méthodes pour empêcher la lecture non autorisée de données est de les *chiffrer*. Le chiffrement transforme un *texte clair* en *cryptogramme*. La reconstruction du texte clair à partir du cryptogramme s'appelle *déchiffrement*. L'étude des cryptogrammes s'appelle *cryptologie*.

Les cultures antiques utilisaient le plus souvent des écritures secrètes remplaçant des lettres par d'autres lettres ou de tout nouveaux symboles. L'écriture secrète utilisée ici a été développée spécialement pour le Castor Informatique, mais se base sur un concept venant de la Palestine antique. À l'époque, la règle de sécurité était que seules des écriture secrètes faciles à apprendre par cœur pouvaient être utilisées. C'était considéré comme un trop grand risque de garder une description écrite de l'écriture secrète. Une table comme celle utilisée ici est facile à apprendre par cœur. Le célèbre chiffre des francs-maçon se base sur ce principe.

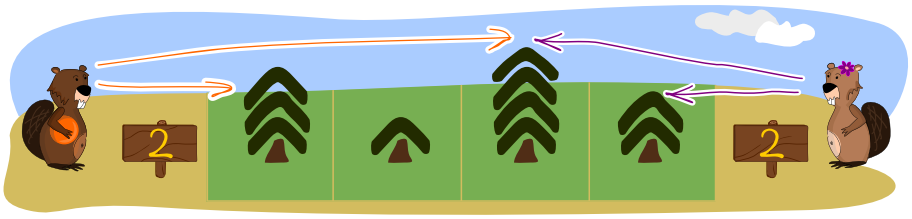
Mots clés et sites web

- Cryptologie : `https://fr.wikipedia.org/wiki/Cryptologie`
- Cryptogramme
- Chiffrer
- Déchiffrer

8. Sudoku boisé 4×4

Les castors plantent 16 arbres (quatre arbres de hauteur 4, quatre arbres de hauteur 3, quatre arbres de hauteur 2 et quatre arbres de hauteur 1) dans un champ de taille 4×4. Pour cela, ils suivent les règles suivantes :

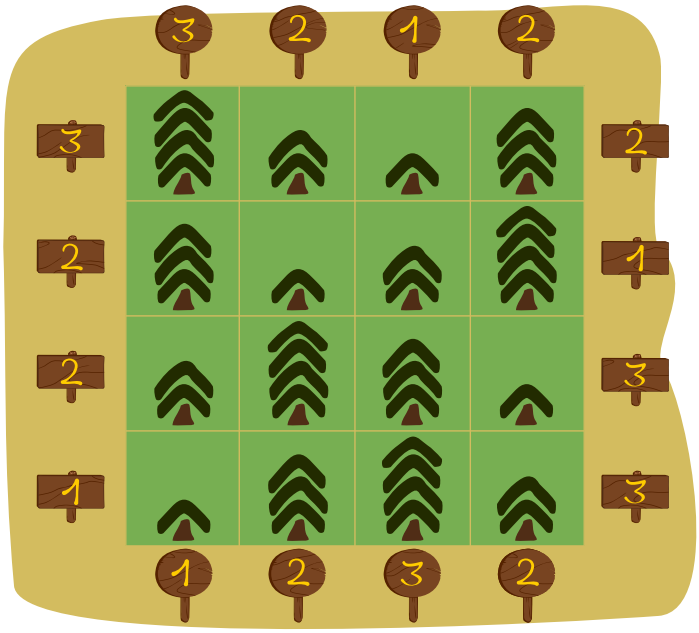
- dans chaque ligne, il y a exactement un arbre de chaque hauteur ;
- dans chaque colonne, il y a exactement un arbre de chaque hauteur.



Lorsque les castors observent une rangée d'arbres depuis l'une de ses extrémités, il ne peuvent **pas** voir les plus petits arbres qui sont cachés derrière de plus grands arbres. C'est écrit sur un panneau au bout de chaque rangée combien de sapins l'on peut voir depuis cet endroit-là. Les panneaux indiquant le nombre de sapins visibles sont plantés tout autour du champ.

Kubko a essayé de représenter le champ d'après sa description sur une feuille de papier. Il a reporté les chiffres sur les panneaux correctement, mais il a fait des erreurs en dessinant quatre des arbres.

Entoure les quatre positions auxquelles les arbres dessinés sont faux et note à côté la hauteur de l'arbre qui devrait s'y trouver.



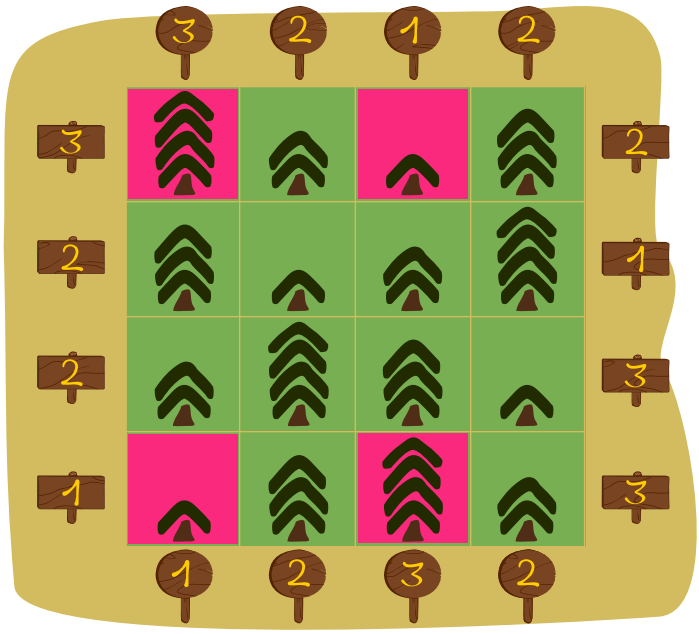
Solution

On remarque d'abord que les deux règles du « sudoku » ont été suivies : il y a exactement un arbre de chaque hauteur dans chaque rangée.

On peut ensuite vérifier pour quelles rangées les chiffres sur les panneaux sont justes et pour lesquels ils ne sont pas. On détermine ainsi que les chiffres pour les lignes 2 et 3 et les colonnes 2 et 4 sont justes. Les chiffres pour les autres rangées ne sont pas justes ; on appelle ces rangées *problématiques*.

Cela ne suffit pas encore. On aimerait savoir quelles positions causent les chiffres erronés. Pour cela, on remarque qu'il y a exactement quatre positions qui se trouvent en même temps dans une ligne et dans une colonne problématique. C'est les positions auxquelles les lignes problématiques (1 et 4) et les colonnes problématiques (1 et 3) se croisent.

On obtient la bonne solution en échangeant les deux arbres d'un ligne ou colonne se trouvant au croisement problématique de cette ligne et de cette colonne (marqués en rouge).



On peut vérifier que ceci est en effet la seule solution possible de la manière suivante : d'après l'énoncé de l'exercice, il y a exactement quatre arbres qui ne sont pas indiqués correctement. Lorsque l'on change un arbre à une position, il faut en changer au minimum deux autres pour que la règle du sudoku soit respectée : un arbre dans la colonne concernée et un dans la ligne concernée. On a donc déjà changé trois arbres. Les deux derniers changements demandent à leur tour un changement chacun dans chaque nouvelles ligne et colonne concernées. Comme l'on ne peut faire que quatre changements en tout, c'est possible uniquement si les deux derniers changements tombent sur la même position et ne font qu'un, ce qui n'arrive que si les quatre positions à changer sont disposées de manière rectangulaire. Comme il fait faire au moins un changement dans chaque rangée problématique, la solution ci-dessus est la seule possible.

C'est de l'informatique !

Cet exercice est centré sur trois compétences fondamentales pour les informaticiennes et informaticiens.

Premièrement, il s'agit de trouver une solution respectant certaines contrainte, ou si nécessaire de corriger une solution proposée.

Deuxièmement, il s'agit de la capacité de reconstruire des objets en se basant sur leur représentation à partir d'informations partielles. Ceci est lié à la génération d'objets (*représentation d'objets*) à partir d'informations disponibles limitées lorsque leur conformité aux lois est connue. On peut aussi utiliser de tels procédés dans la *compression de données*.

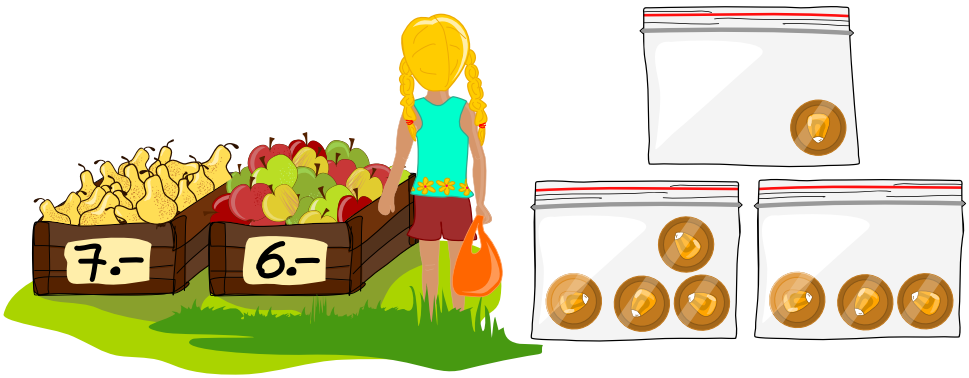
Troisièmement, on peut utiliser de tels champs d'arbres avec des panneaux pour créer des *codes correcteurs*. Des erreurs arrivant lors de l'entrée des données ou du transfert d'information peuvent ainsi être automatiquement reconnues ou même corrigées.

Mots clés et sites web

- Sudoku : `https://fr.wikipedia.org/wiki/Sudoku`
- Représentation d'objets
- Compression de données : `https://fr.wikipedia.org/wiki/Compression_de_données`
- Détection et correction d'erreurs : `https://fr.wikipedia.org/wiki/Code_correcteur`

9. Transport d'argent

Bina aime bien nager. Pour aller dans l'eau, elle met sa monnaie dans des sachets étanches pour que le métal ne commence pas à rouiller. Hier, Bina avait pris trois sachets avec 1, 3 et 4 pièces de monnaie. Comme cela, elle a pu payer une poire exactement (sans qu'on ne lui rende de monnaie) sans devoir ouvrir de sachet, mais pas de pomme.



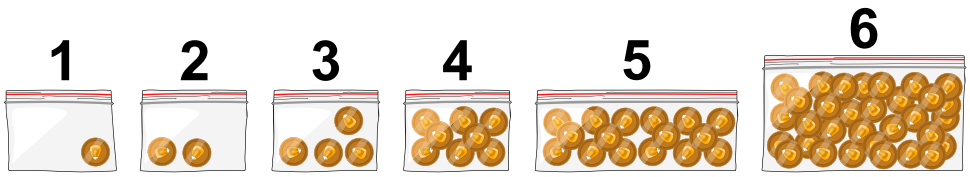
Aujourd'hui, Bina a pris 63 pièces pareilles. Elle aimeraient les répartir dans différents sachets de manière à pouvoir payer tous les montants entre 1 et 63 pièces exactement et sans devoir ouvrir de sachet.

Quel est le plus petit nombre de sachets dont Bina a besoin ?

A) 4 sachets
B) 5 sachets
C) 6 sachets
D) 7 sachets
E) 8 sachets
F) 15 sachets
G) 16 sachets
H) 31 sachets
I) 32 sachets ou plus

Solution

La bonne réponse est C) 6 sachets :



Bina peut répartir les sachets de la manière suivante :

- Sachet 1 : 1 pièce
- Sachet 2 : 2 pièces
- Sachet 3 : 4 pièces
- Sachet 4 : 8 pièces
- Sachet 5 : 16 pièces
- Sachet 6 : 32 pièces

Bina a donc ainsi $1 + 2 + 4 + 8 + 16 + 32 = 63$ pièces dans les sachets et peut payer chaque montant entre 1 et 63 pièces exactement sans qu'on ne lui rende de monnaie et sans devoir ouvrir de sachet.

Pour payer 13 pièces, elle peut par exemple utiliser les sachets 1, 3 et 4.

La table ci-dessous montre comment chaque montant peut être payé exactement en sélectionnant les bons sachets parmi les 6. Une cellule contient un 1 si Bina utilise le sachet correspondant pour payer et un 0 sinon.

Montant	32	16	8	4	2	1	Montant	32	16	8	4	2	1
0	0	0	0	0	0	0	**32**	1	0	0	0	0	0
1	0	0	0	0	0	1	**33**	1	0	0	0	0	1
2	0	0	0	0	1	0	**34**	1	0	0	0	1	0
3	0	0	0	0	1	1	**35**	1	0	0	0	1	1
4	0	0	0	1	0	0	**36**	1	0	0	1	0	0
5	0	0	0	1	0	1	**37**	1	0	0	1	0	1
6	0	0	0	1	1	0	**38**	1	0	0	1	1	0
7	0	0	0	1	1	1	**39**	1	0	0	1	1	1
8	0	0	1	0	0	0	**40**	1	0	1	0	0	0
9	0	0	1	0	0	1	**41**	1	0	1	0	0	1
10	0	0	1	0	1	0	**42**	1	0	1	0	1	0
11	0	0	1	0	1	1	**43**	1	0	1	0	1	1
12	0	0	1	1	0	0	**44**	1	0	1	1	0	0
13	0	0	1	1	0	1	**45**	1	0	1	1	0	1
14	0	0	1	1	1	0	**46**	1	0	1	1	1	0
15	0	0	1	1	1	1	**47**	1	0	1	1	1	1
16	0	1	0	0	0	0	**48**	1	1	0	0	0	0
17	0	1	0	0	0	1	**49**	1	1	0	0	0	1
18	0	1	0	0	1	0	**50**	1	1	0	0	1	0
19	0	1	0	0	1	1	**51**	1	1	0	0	1	1
20	0	1	0	1	0	0	**52**	1	1	0	1	0	0
21	0	1	0	1	0	1	**53**	1	1	0	1	0	1
22	0	1	0	1	1	0	**54**	1	1	0	1	1	0
23	0	1	0	1	1	1	**55**	1	1	0	1	1	1
24	0	1	1	0	0	0	**56**	1	1	1	0	0	0
25	0	1	1	0	0	1	**57**	1	1	1	0	0	1
26	0	1	1	0	1	0	**58**	1	1	1	0	1	0
27	0	1	1	0	1	1	**59**	1	1	1	0	1	1
28	0	1	1	1	0	0	**60**	1	1	1	1	0	0
29	0	1	1	1	0	1	**61**	1	1	1	1	0	1
30	0	1	1	1	1	0	**62**	1	1	1	1	1	0
31	0	1	1	1	1	1	**63**	1	1	1	1	1	1

Bina ne peut pas atteindre son but avec moins de 6 sachets. Elle peut utiliser ou non chaque sachet pour payer, il y a donc exactement deux possibilités par sachet. Avec 5 sachets ou moins, elle n'aurait au maximum que $2^5 = 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 32$ possibilités de les combiner. Elle pourrait donc payer exactement au maximum 32 montant différents, ce qui n'est pas suffisant pour tous les montants de 1 à 63 pièces.

C'est de l'informatique !

Cet exercice traite des *nombres binaires*. Les nombres binaires sont étudiés de manière différente en mathématiques et en informatique. En mathématiques, on se concentre surtout sur leurs propriétés, alors qu'en informatique, on s'intéresse à leurs applications. Les ordinateurs utilisent les nombres binaires pour représenter toutes sortes d'informations différentes : des documents, des images, des voix, des vidéos et des nombres, même les programmes et les apps que nous utilisons tous sont codées en nombres binaires. L'unité utilisée est le *bit* (de l'anglais « *b*inary dig*it* », chiffre binaire) qui peut valoir soit 0 soit 1. Un bit ne peut donc représenter que deux possibilités. Avec deux bits, on peut par contre déjà représenter 4 possibilités : 00, 01, 10 et 11. Dans l'exercice ci-dessus, Bina utilise 6 bits (sachets) afin de représenter $2^6 = 64$ montants.

Les ordinateurs rassemblent habituellement les bits en groupes de 8 ; un tel groupe de 8 s'appelle un octet. Un octet peut représenter $2^8 = 256$ nombres différents, de 0 à 255.

Mots clés et sites web

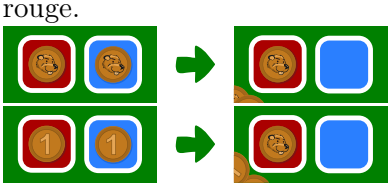
- Nombre binaire : `https://fr.wikipedia.org/wiki/Code_binaire`
- Représentation de données
- Logique
- `https://fr.wikipedia.org/wiki/Bit`, `https://fr.wikipedia.org/wiki/Octet`

10. Las Bebras

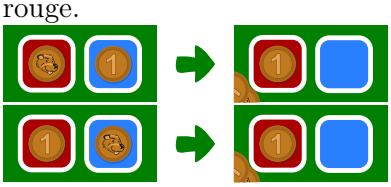
Au casino « Las Bebras », Gloria peut jouer avec des pièces de monnaie à la table de John. Gloria a 4 pièces de monnaie avec, d'un côté, une face, et de l'autre côté, et un chiffre 1. Gloria jette les deux premières pièces et en pose une sur la case rouge et l'autre sur la case bleue.

John échange les deux pièces contre une seule pièce qu'il pose sur la case rouge.

- Si les deux pièces sont pareilles, John met la nouvelle pièce face vers le haut sur la case rouge.



- Si les deux pièces sont différentes, John met la nouvelle pièce chiffre vers le haut sur la case rouge.



Gloria jette maintenant une nouvelle pièce et la met sur la case bleue. John échange à nouveau les pièces en suivant les même règles, et ainsi de suite jusqu'à ce que Gloria ait joué ses 4 pièces. Le jeu est terminé lorsque John pose la dernière pièce sur la case rouge. Si cette pièce est posée chiffre 1 vers le haut, Gloria a gagné !

Gloria jette les quatre pièces dans l'ordre indiqué de droite à gauche. Quelle suite permet à Gloria de gagner ?

A) 

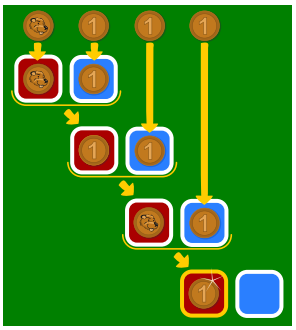
B) 

C) 

D) 

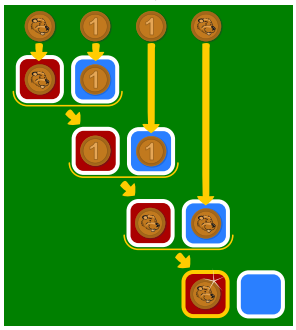
Solution

La bonne réponse est C). C'est la seule réponse pour laquelle la dernière pièce est posée chiffre vers le haut sur la case rouge.

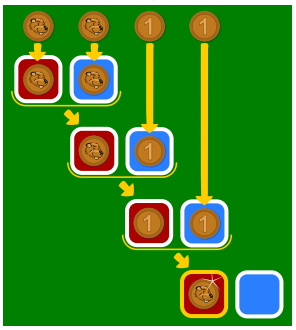


Pour toutes les autres suites, la dernière pièce est posée face vers le haut sur la case rouge.

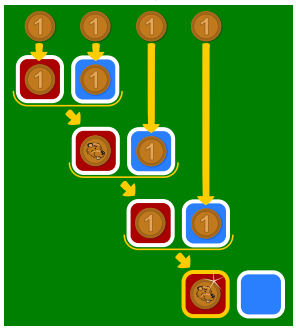
A)



B)



D)



Pour chacune des quatre pièces jouée par Gloria, il y a deux possibilités de les poser (ou), on peut donc jouer $2^4 = 16$ suites différentes avec 4 pièces. Si un nombre pair de pièces est posé face (ou chiffre) vers le haut dans la suite, la dernière pièce du jeu est posée face vers le haut sur la case rouge. Si un nombre impair de pièces est posé face (ou chiffre) vers le haut dans la suite, la dernière pièce du jeu est posée chiffre vers le haut sur la case rouge. Les suites de pièces avec un nombre impair de pièces posées face (ou chiffre) vers le haut sont donc les « suites gagnantes ». Il existe exactement 8 suites différentes ayant un nombre impair et 8 suites différentes ayant un nombre pair de pièces face (ou chiffre) vers le haut.

C'est de l'informatique !

Comme les ordinateurs sont des machines électroniques, l'électricité y est utilisée pour représenter les informations. Deux états peuvent être simplement représentés par la présence ou l'absence d'un courant électrique. Les informaticiens et informaticiennes représentent habituellement ces deux états par les chiffres 0 et 1. Nous appelons cela une *représentation binaire*. Une unité d'information est appelée un *bit*.

Nous pouvons effectuer des opérations sur de tels bits et les combiner, comme l'orientation de deux pièces de monnaie mène à une nouvelle orientation de pièce dans cet exercice.

L'une de ces opérations, appelée *OU exclusif* ou *XOR* (« *eXclusive OR* » en anglais), est présentée dans cet exercice et fonctionne de la manière suivante :

0 XOR 0 = 0
0 XOR 1 = 1
1 XOR 0 = 1
1 XOR 1 = 0

Nous rencontrons aussi de telles opérations dans notre vie quotidienne, par exemple lorsque deux interrupteurs qui allument et éteignent la même lampe se trouvent aux deux bouts d'un escalier. Si les deux interrupteurs sont enclenchés ou éteints, la lampe est allumée. Si l'un des interrupteurs est allumé et l'autre éteint, la lampe est éteinte.

Une porte logique XOR est une application électronique de l'opération XOR dans des ordinateurs. Une porte XOR a 1 comme valeur de sortie si exactement une des valeurs d'entrée est 1. Si les deux valeurs d'entrée sont égales, la valeur de sortie est 0.

L'opération XOR a plusieurs applications en informatique, par exemple :

- Elle nous dit si deux bits sont égaux ou inégaux.
- Elle nous dit si le nombre de bits valant 1 dans une suite de bits est pair ou impair (le résultat du XOR d'une suite de bits est « vrai » quand un nombre impair de bits sont « vrais »).
- En cryptographie, l'opération XOR est utilisée lors du chiffrement symétrique à l'aide de masques jetables.

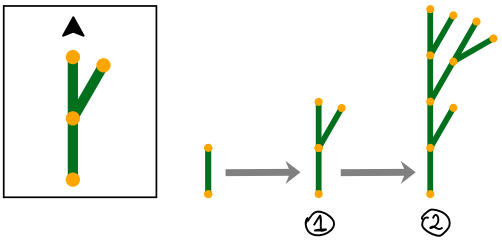
Mots clés et sites web

- Opération binaire : `https://fr.wikipedia.org/wiki/Opération_binaire`
- XOR : `https://fr.wikipedia.org/wiki/Fonction_OU_exclusif`
- Porte logique : `https://fr.wikipedia.org/wiki/Fonction_logique`

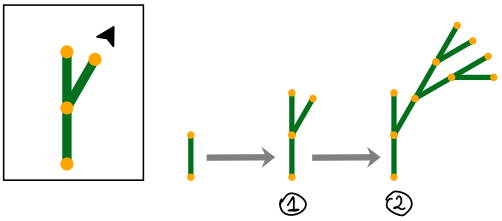
11. Arbres digitaux

Un arbre digital est fait de tronçons d'arbre comme celui-ci :  .

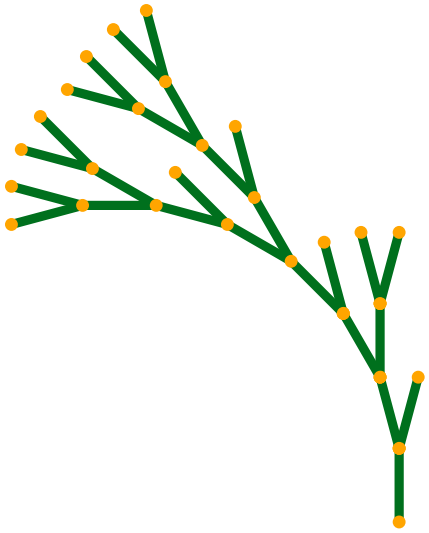
Il pousse étape par étape d'après une règle de croissance définie.
La règle de croissance indique de quelle manière un tronçon est remplacé par une structure composée de nouveaux tronçons. Lors de chaque étape, chaque tronçon est remplacé de cette manière. Une pointe de flèche indique où et dans quelle direction les tronçons sont assemblés.



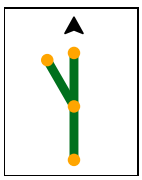
Les exemples à droite montrent deux règles de croissance et les deux premières étapes de croissance correspondantes.

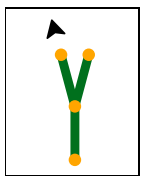


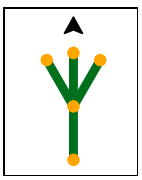
L'arbre suivant a poussé en trois étapes :

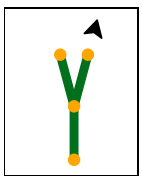


D'après quelle règle de croissance l'arbre digital a-t-il poussé ?

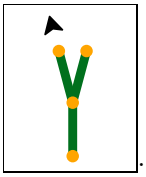
A) 

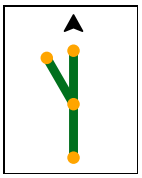
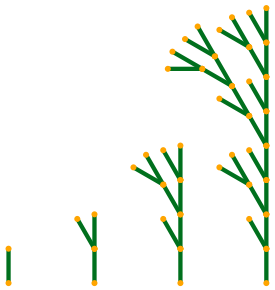
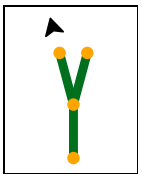
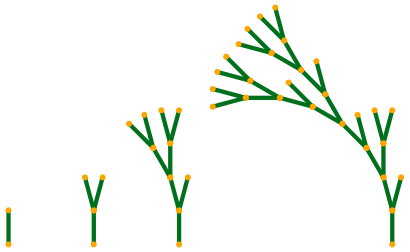
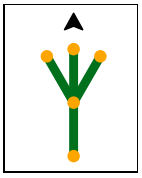
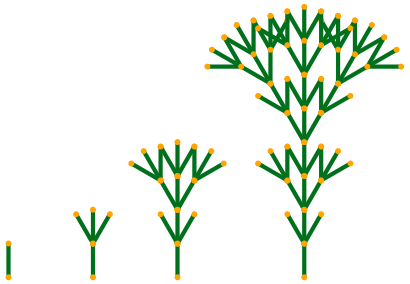
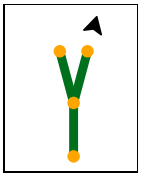
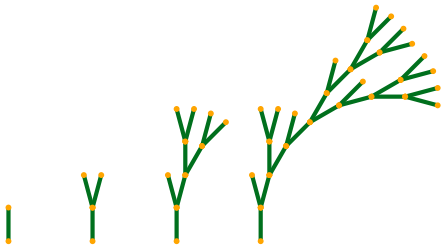
B) 

C) 

D) 

Solution

La bonne réponse est B)  .

Règle de croissance	Trois étapes de croissance	Description
		Le reste de l'arbre est toujours ajouté à la branche dirigée tout droit vers le haut. Il se forme ainsi un tronc droit avec des branches toutes orientées à gauche.
		Le reste de l'arbre est toujours ajouté à la branche dirigée vers la gauche et le haut. L'arbre est donc penché vers la gauche.
		Le reste de l'arbre est toujours ajouté à la branche du milieu. Les deux branchements à gauche et à droite génèrent une structure régulière et symétrique.
		Le reste de l'arbre est toujours ajouté à la branche dirigée vers la droite et le haut. L'arbre est donc penché vers la droite.

C'est de l'informatique !

Dans cet exercice, on voit comment l'application répétée de règles simples peuvent générer des structures compliquées. De telles figures formées de parties semblables à la figure complète sont aussi appelées des *fractales*. En informatique, on recourt très souvent aux fractales, par exemple pour créer des paysages ou des effets spéciaux pour des films.

En biologie, on utilise ce qu'on appelle des *systèmes de Lindenmayer* (ou *L-systèmes*) pour simuler la croissance des plantes. Ce système génère également des fractales. Dans cet exercice, nous avons vu des exemples très simples de L-systèmes.

Les arbres dans cet exercice sont générés par l'application d'une règle sur chaque tronçon d'arbre, puis à nouveau sur chaque tronçon ainsi généré, et ainsi de suite. De tels procédés sont appelés *récursifs*. La concept de la récursivité est important en informatique. Grâce à la récursivité, il est possible de décrire de manière très simple beaucoup de choses compliquées.

Mots clés et sites web

- Fractale : `https://fr.wikipedia.org/wiki/Fractale`
- L-système : `https://fr.wikipedia.org/wiki/L-Système`, `http://paulbourke.net/fractals/lsys/`
- Récursivité : `https://fr.wikipedia.org/wiki/Récursivité`

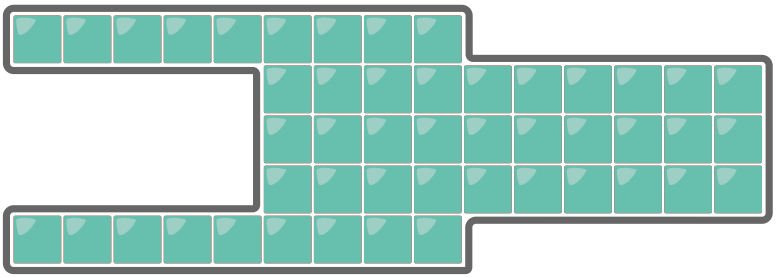
12. Chauffage au sol

Luis n'aime pas se changer dans la salle de bain froide le matin, c'est pourquoi il aimerait installer un chauffage au sol dans la nouvelle maison. Le chauffagiste lui conseille l'innovant « chauffage au sol à hotspots » : un hotspot est installé directement sous une catelle. Lorsque l'on allume le hotspot, cette catelle devient tout de suite chaude.

En une minute, la chaleur se propage à toutes les catelles voisines, c'est-à-dire à toutes les catelles qui touchent le bord ou un angle de la catelle déjà chauffée. Le nombre sur chaque catelle indique au bout de combien de minutes elle devient chaude.

Luis veut installer quatre hotspots dans sa salle de bain de manière à ce que toutes les catelles deviennent chaudes le plus vite possible.

Sous quelles quatre catelles le chauffagiste doit-il installer les quatre hotspots ?

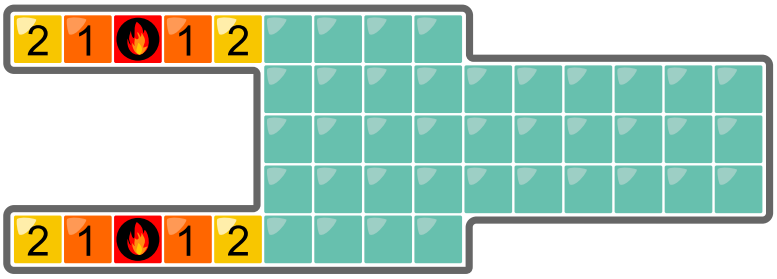


Solution

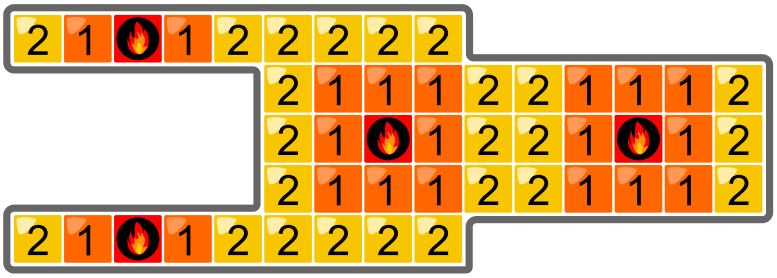
Lorsque les quatre hotspots sont installés comme dans l'image ci-dessous, toutes les catelles de la salle de bain sont chaudes deux minutes après avoir allumé le chauffage.

C'est optimal, car c'est impossible de chauffer toutes les catelles en une minute avec quatre hotspots. On peut le voir de la manière suivante. Chaque hotspot peut chauffer au maximum neuf catelles pendant la première minute, celle sous laquelle il se trouve et jusqu'à 8 catelles autour. Quatre hotspots peuvent donc chauffer au maximum $4 \cdot 9 = 36$ catelles pendant la première minute. La salle de bain a 48 catelles en tout, donc une minute ne peut pas suffire. Cela pourrait par contre marcher en deux minutes, pendant lesquelles $4 \cdot 25 = 100$ catelles pourraient théoriquement être chauffées.

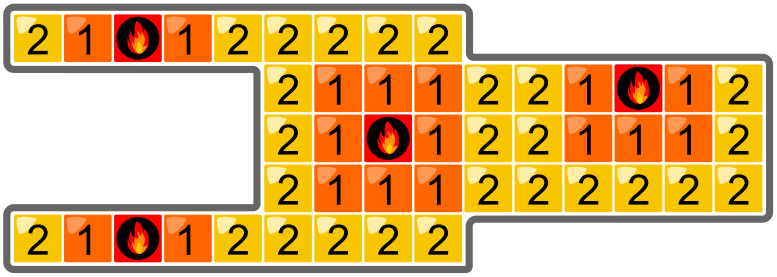
Il serait maintenant logique de commencer à répartir les hotspots dans les deux couloirs à gauche. En mettant un hotspot au milieu de chaque couloir, toutes les catelles des couloirs sont chauffées au bout de deux minutes :

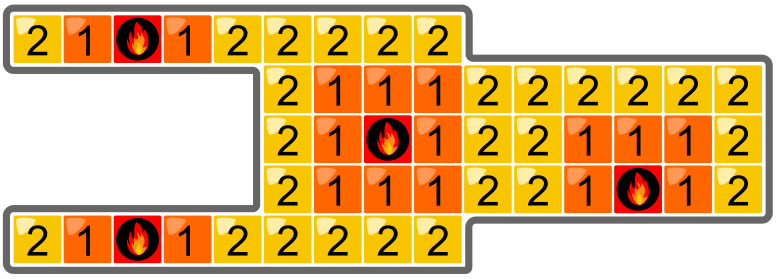


On peut placer les deux autres hotspots comme cela :



Les deux placements suivants sont également possibles :





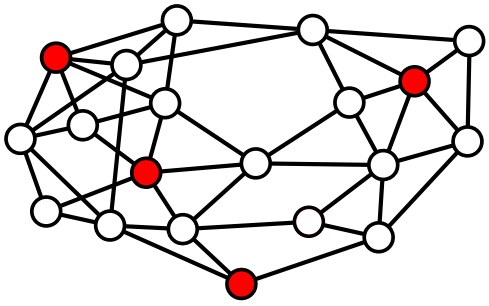
Si la salle de bain avait une forme différente, deux hotspots pourraient suffire à chauffer la même surface en deux minutes.

C'est de l'informatique !

Le problème résolu dans cet exercice est proche d'un problème d'optimisation très connu : on y cherche un petit nombre de *nœuds* dans un *graphe*, nœuds que l'on appelle *ensemble dominant*.

Un ensemble dominant est défini ainsi : chaque nœud du graphe doit soit faire partie de l'ensemble dominant, soit avoir un voisin qui en fait partie. Les catelles de la salle de bain peuvent être représentées avec des nœuds. Les nœuds sont reliés par des arêtes lorsque la catelle suivante est chauffée au bout d'une minute. Un ensemble dominant du graphe résultant indique alors les endroits auxquels des hotspots peuvent être installés pour chauffer la salle de bain en deux minutes.

Dans le cas général, c'est très difficile de trouver un ensemble dominant minimal, mais il existe des algorithmes efficients pour des graphes spéciaux. Le dessin suivant montre un exemple. Comme l'on peut voir, chaque nœud blanc a au moins un nœud rouge comme voisin. Les nœuds rouge sont donc un ensemble dominant.



Une application typique est le placement de bornes Wi-Fi dans un grand bâtiment. Les nœuds du graphe sont les pièces individuelles. Deux d'entre elles sont voisines dans le graphe si les deux pièces sont couvertes par une même borne. Les pièces formant un ensemble dominant minimal sont les endroits appropriés pour placer les bornes Wi-Fi.

Mots clés et sites web

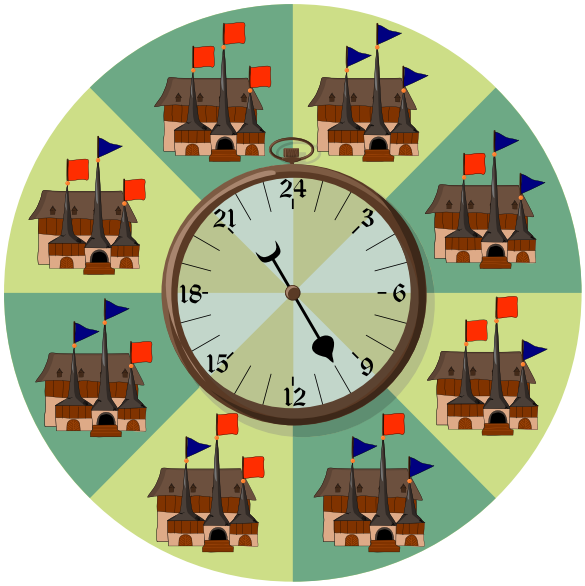
- Ensemble dominant : `https://fr.wikipedia.org/wiki/Ensemble_dominant`

13. Journée tranquille

Les castors vivant dans un petit village tranquille sont très détendus. Ils divisent leurs journées en seulement 8 tranches horaires de 3 heures chacune. La tranche horaire en cours est indiquée par trois drapeaux sur l'hôtel de ville comme représenté sur l'image ci-dessous. Les castors utilisent deux sortes de drapeaux, un carré rouge et un triangle bleu.

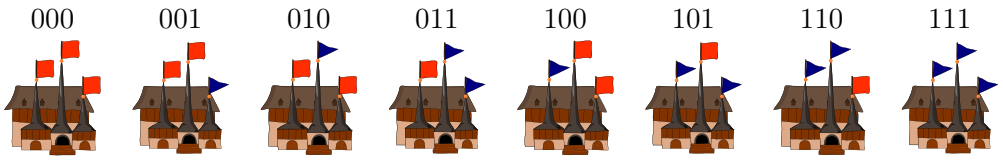
L'arrangement des drapeaux ci-dessus ne demande le changement que d'un seul drapeau à presque chaque transition. Il n'y a qu'à minuit où trois drapeaux doivent être changés d'un coup. Les castors aimeraient trouver un arrangement plus commode qui permette de ne changer qu'un seul drapeau à chaque transition.

Trouve un tel arrangement commode pour les castors et dessine les trois drapeaux de chaque tranche horaire.



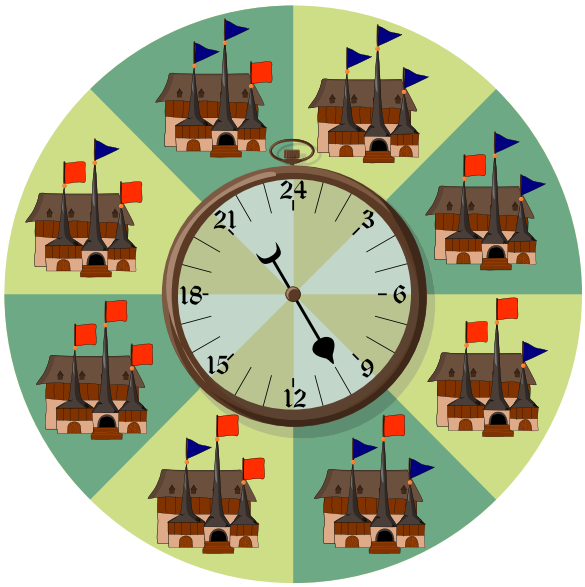
Solution

On peut utiliser des nombres binaires à trois chiffres pour représenter les 8 motifs de drapeaux : 0 représente un carré rouge et 1 un triangle bleu.



Les 8 motifs sont donc 000, 001, 010, 011, 100, 101, 110, 111. Nous devons à présent mettre ces nombres dans un ordre dans lequel les nombres voisins ainsi que le premier et dernier nombre ne diffèrent qu'à une seule position.

On peut y arriver par tâtonnement. Une solution possible est 111, 011, 001, 101, 100, 000, 010, 110. Voici l'horloge correspondante :



On peut trouver une solution de manière systématique avec la méthode suivante :

Nous ne considérons d'abord que les nombres qui commencent avec deux zéros, donc 000 et 001. Ici, il y a deux ordres possibles, et les deux remplissent la condition décrite plus haut. Nous choisissons 000, 001.

Maintenant, nous écrivons les deux mêmes nombres dans l'ordre inverse après les deux premiers (donc 001, 000), mais en changeant la deuxième position de 0 à 1 (donc 011, 010). Nous obtenons ainsi la suite de nombres 000, 001, 011, 010. Elle remplit également la condition.

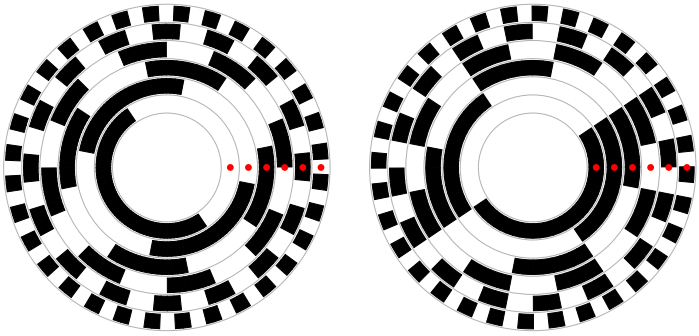
Nous écrivons à nouveau cette nouvelle suite de nombre à l'envers à la suite de la précédente en changeant cette fois la première position de 0 à 1. Nous obtenons ainsi 000, 001, 011, 010, 110, 111, 101, 100, ce qui remplit à nouveau notre condition. Nous avons trouvé la solution recherchée.

Cette méthode (la symétrie de la suite de nombre existante et le changement de la position supérieure de 0 à 1) peut être répétée autant de fois que nécessaire pour obtenir de tels arrangements pour n'importe quel nombre de drapeaux au lieu de trois. On peut se demander pourquoi cette méthode fonctionne toujours et si tous les motifs possibles sont toujours utilisés.

C'est de l'informatique !

Un tel arrangement de nombres binaires s'appelle le *code de Gray* et a beaucoup d'applications. Le fait qu'un seul bit diffère entre deux nombres voisins peut par exemple servir à économiser de l'énergie. Le changement de plusieurs bits demande plus d'énergie, et il y a souvent plusieurs bits qui changent en même temps lors de l'énumération ascendante normale dans le système binaire.

Une application connue du code de gray en ingénierie est la mesure des angles d'une plaque tournante (appelée roue codeuse). On dessine le code de gray sur la plaque comme montré en dessous, en blanc pour 0 et en noir pour 1. Les points rouges sont des détecteurs installés en ligne droite pouvant différencier le blanc du noir. Les détecteurs peuvent ainsi lire un nombre binaire qui code la valeur de l'angle de la roue.



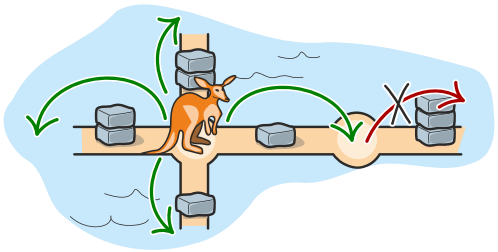
Sur l'image de gauche, on voit que le quatrième détecteur se trouve exactement à la limite entre le blanc et le noir. Le détecteur va donc lire soit 001010 soit 001110. Les deux options sont acceptables, étant donné que la valeur de l'angle se situe exactement au milieu des deux codes. Si l'on n'utilise pas de code de Gray, la situation est plus difficile. Considérons le code binaire normal sur l'image de droite. Ici, les codes 111010 et 111001 se suivent. Si les détecteurs se trouvent exactement entre ces deux codes, les deux derniers détecteurs ne peuvent pas différencier entre le blanc et le noir. Les détecteurs pourraient donc lire le code 111011 qui se trouve plus loin sur la roue. Dans le pire des cas, les détecteurs se trouvent à la limite entre le code blanc 000000 et le code noir 111111, et chaque détecteur peut arbitrairement lire soit 0, soit 1, ce qui rend la mesure de l'angle complètement inutilisable.

Mots clés et sites web

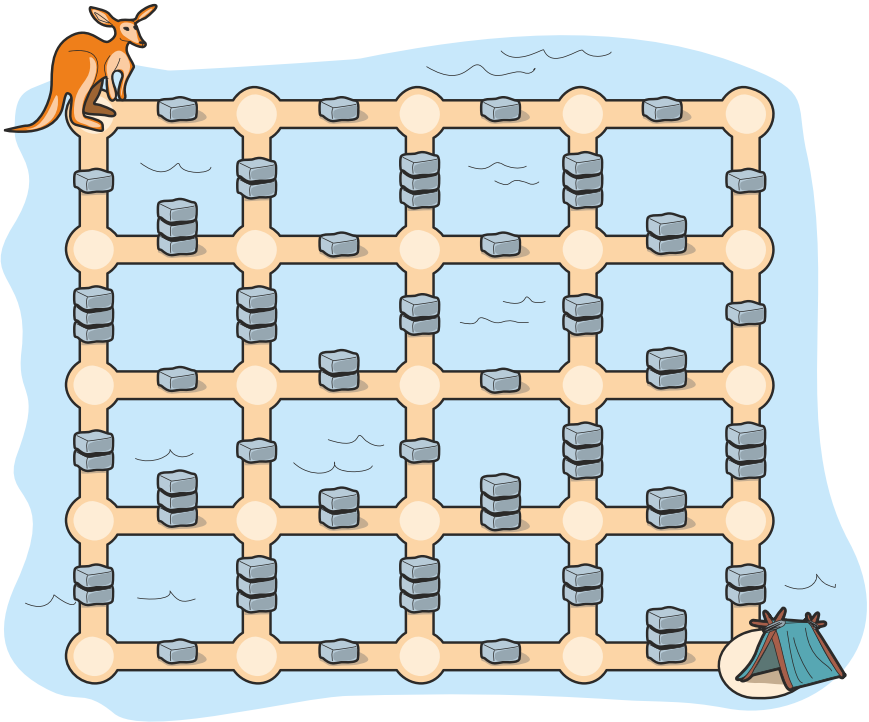
- Code de Gray : `https://fr.wikipedia.org/wiki/Code_de_Gray`
- Roue codeuse : `https://fr.wikipedia.org/wiki/Roue_codeuse`

14. Kangourou bondissant

Un kangourou saute jusqu'à la maison. Il ne peut sauter que sur le chemin et atteint le croisement suivant d'un grand saut. À un croisement, il peut sauter soit à gauche, soit à droite, soit vers le haut, soit vers le bas. Il n'arrive pas à sautant au dessus d'un tas de 3 cailloux.



Le kangourou aimerait rentrer à la maison par le chemin le plus court.

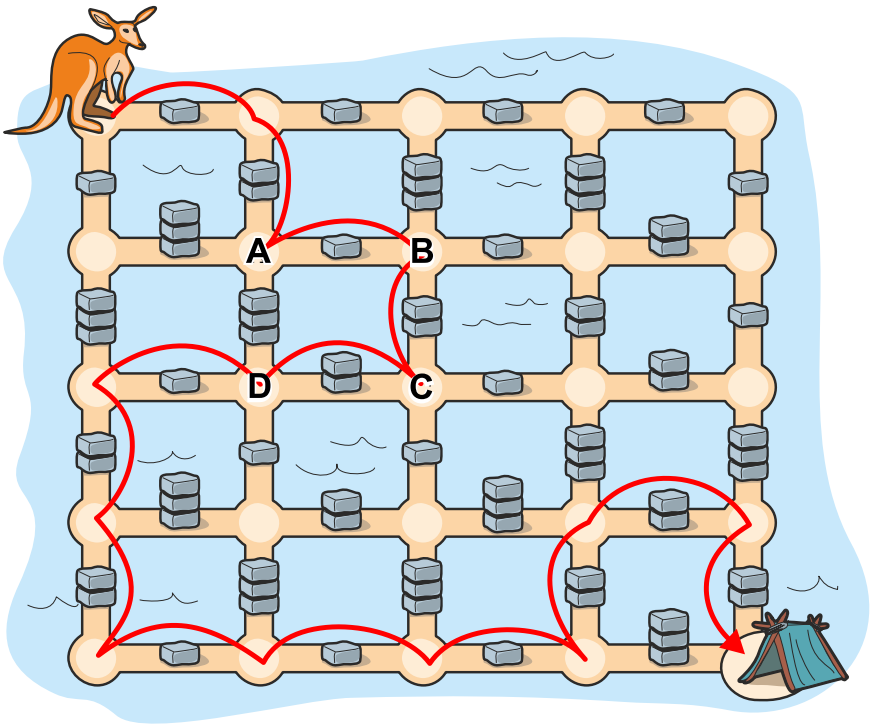


Combien de sauts le kangourou doit-il faire s'il rentre à la maison par le chemin le plus court ?

A) 10 sauts
B) 11 sauts
C) 12 sauts
D) 13 sauts
E) 14 sauts
F) 15 sauts
G) 16 sauts
H) 17 sauts
I) 18 sauts
J) 19 sauts
K) 20 sauts

Solution

La bonne réponse est E) 14 sauts :



Le plus simple est de commencer la recherche par la fin. On voit rapidement qu'il n'y a qu'un chemin possible sur une grande distance depuis l'arrivée, à savoir 9 sauts jusqu'au point D. Maintenant, on ne doit plus que trouver le chemin le plus court depuis le départ jusqu'au point D. En deux sauts, le kangourou arrive au point A. Il ne peut pas sauter directement du point A au point D, car il y a un tas de trois cailloux entre deux. Le détour le plus court pour aller de A à D passe par B et C, le kangourou doit pour cela faire 3 sauts. Le kangourou doit donc faire $2 + 3 + 9 = 14$ sauts en tout ; tous les autres chemins sont plus longs.

C'est de l'informatique !

On peut procéder de la manière suivante pour trouver n'importe quel chemin : on suit un chemin au choix pas à pas. Dès que l'on arrive à une impasse où toutes les directions sont bloquées ou que l'on arrive à un endroit déjà visité du chemin, on revient en arrière jusqu'à trouver une autre direction possible, et on essaie ensuite dans cette direction.

En informatique, cette méthode de résolution s'appelle *retour sur trace* ou *retour arrière* (« *backtrack* » en anglais). Elle est utilisée de manière variée dans différents algorithmes. Elle peut être utilisée pour trouver la solution de puzzles, de sudokus et d'autres problèmes d'optimisation combinatoire.

Cet exercice montre qu'il est parfois plus efficace de commencer par la fin pour trouver une solution. On parle alors d'une *recherche en arrière*. Dans notre cas, on a besoin de faire moins de retour sur

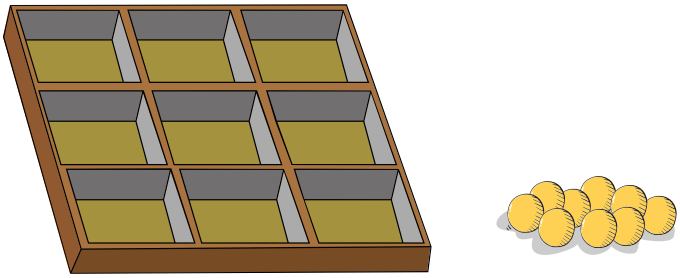
trace, car on a plus d'options du tout à la fin du chemin. On ne peut pas dire de manière générale qu'une recherche en arrière ou une recherche en avant est mieux, cela dépend du problème concret.

Mots clés et sites web

- Retour sur trace : `https://fr.wikipedia.org/wiki/Retour_sur_trace`

15. Des cases et des billes

Hira a une boite qui est divisée en 9 cases, et un nombre de billes illimité :



Hira met des billes dans les cases de la boîte. Elle suit les règles suivantes :

- elle met au maximum une bille dans chaque case ;
- le nombre de billes total dans chaque ligne et chaque colonne est pair quand elle a fini.

Combien de motifs différents Hira peut-elle créer ?

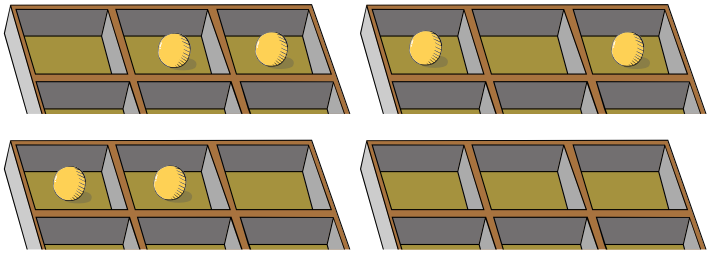
(La boîte ne peut pas être tournée. Le motif avec une bille en haut à gauche est par exemple différent du motif avec un bille en haut à droite.)

A) 12
B) 16
C) 64
D) 512

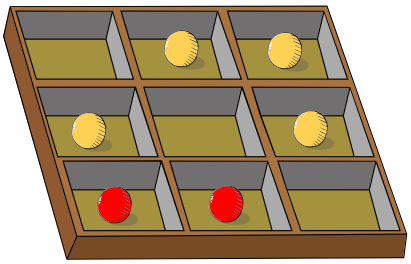
Solution

La bonne réponse est B) 16.

De combien de manières différente Hira peut-elle remplir la première ligne ? Il doit y avoir un nombre pair de billes dans la première ligne, donc 0 ou 2. Il y a donc 4 possibilités de remplir la première ligne :



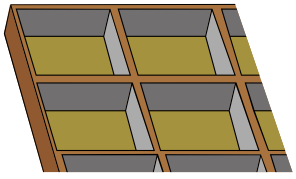
De la même manière, Hira a 4 possibilités de remplir la deuxième ligne. Pour la troisième ligne, elle ne peut plus choisir, car il doit aussi y avoir un nombre pair de bille dans chacune des trois colonnes. S'il y a un nombre impair de billes dans les deux cases du haut d'une colonne (donc exactement une bille), Hira doit mettre une bille dans la troisième case de cette colonne, comme illustré dans les deux premières ligne de l'exemple suivant (billes rouges) :



S'il y a un nombre pair de billes dans les deux premières cases d'une colonne (donc 0 ou 2 billes), elle ne peut pas mettre de bille dans la troisième case de cette colonne, comme c'est le cas dans la troisième colonne de l'exemple en dessus.

Comme le choix pour la première ligne est complètement indépendant du choix pour la deuxième ligne, Hira a 4 possibilités pour la première ligne, et a ensuite à nouveau 4 possibilités pour la deuxième ligne pour chacune de ces quatre possibilités. Elle a donc en tout $4 \cdot 4 = 16$ possibilités.

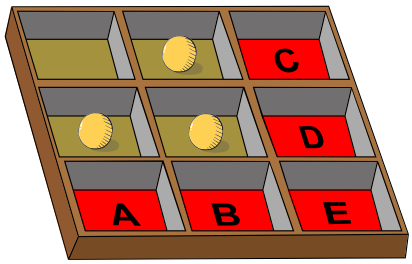
Un autre option pour compter les possibilités est la suivante : on commence par considérer une partie de la boîte faisant 2×2 cases.



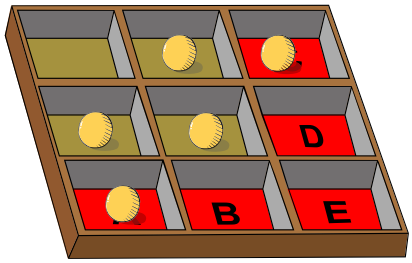
Dans cette partie de la boîte, il y a 4 cases qui peuvent chacune contenir une bille ou pas. Il y a donc $2^4 = 16$ possibilités de remplir cette partie de boîte avec des billes.

Une observation importante est la suivante : une fois que les billes ont été placées dans cette partie de la boîte, Hira n'a plus aucun choix concernant le remplissage des cases restantes. Pour chaque case restante au bord à droite ou dans la ligne du bas, Hira doit obligatoirement soit mettre un bille dans la case, soit la laisser vide, afin que le nombre total de bille soit pair.

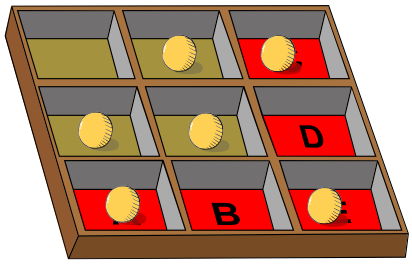
Hira pourrait par exemple remplir la partie de boîte de 2×2 que l'on considère comme cela :



Comme la première colonne ne contient qu'un bille, Hira doit mettre une bille dans la case A pour que le nombre de billes dans cette colonne soit pair. Dans la deuxième colonne, il y a déjà un nombre pair de billes, donc Hira ne peut pas mettre de billes dans la case B. Avec le même raisonnement, on voit que la case D doit rester vide et que Hira doit mettre une bille dans la case C.



Le nombre de billes dans $A + B$ est pair seulement lorsque le nombre de billes dans la partie de boîte de 2×2 cases est pair. Le même chose est vraie pour la somme de $C + D$. Si ces deux sommes sont paires, la case E doit rester vide ; si ces deux sommes sont impaires, Hira doit mettre une bille dans la case E.



Ceci montre que Hira peut mettre des billes de 16 façons différentes dans les cases de la boîte.

C'est de l'informatique !

Une tâche importante de l'informatique est la transmission de données de manière sûre. Une manière d'assurer la transmission de données contre les erreurs de transmission est d'instaurer une *convention de parité*.

Un *bit de parité* est calculé sur la base des données à transmettre et est ajouté à la fin des données. Le bit de parité peut à nouveau être calculé lors de la réception des données. Si les données ne correspondent pas au bit de parité, on sait qu'il y a eu une erreur lors de la transmission.

Dans cet exercice, les cases de la dernière ligne et colonne servent de bits de parité. Si le nombre de billes dans les cases est transmis en tant que données, le destinataire peut calculer la somme des lignes et des colonnes. Si celles-ci ne sont pas paires, le destinataire peut informer Hira qu'il y a eu une erreur lors de la transmission.

Un autre compétence informatique est la capacité à compter toutes les solutions ayant certaines propriétés et ainsi de déterminer leur nombre.

Mots clés et sites web

- Bit de parité :
  `https://fr.wikipedia.org/wiki/Somme_de_contrôle#Exemple_:_bit_de_parité`

A. Auteur·e·s des exercices

Tony René Andersen
Michael Barot
Wilfried Baumann
Maksim Bolonkin
Andrey Brodnik
Sarah Chan
Marios O. Choudary
Valentina Dagienė
Tolmantas Dagys
Christian Datzko
Susanne Datzko
Amirmohammad Djazbi
Nora A. Escherle
Lidia Feklistova
Fabian Frei
Gerald Futschek
Jens Gallenbacher
Tom Grubb
Mathias Hiron
Juraj Hromkovič
Alisher Ikramov
Thomas Ioannou
Ungyeol Jung
Vaidotas Kinčius
Ritambhra Korpal
Regula Lacher
Vu Van Luan
Pedro Marcelino
Hamed Mohebbi
Kwangsik Moon
Xavier Muñoz
Vania Natali
Rana R. Natawigena
Ágnes Erdősné Németh
Andrei Nicolicioiu
Elsa Pellet
Jean-Philippe Pellet
Wolfgang Pohl
Raymond Chandra Putra
Peter Rossmanith
Vipul Shah
Fei Shang
Wenpan Sheng
Timur Sitdikov
Maciej M. Sysło
Congyu Tian
Jiří Vaníček
Troy Vasiga
Fan Wang
Yang Xing
Binru Zhi

B. Sponsoring : Concours 2020

http://www.haslerstiftung.ch/

http://www.baerli-biber.ch/

http://www.verkehrshaus.ch/
Musée des transports, Lucerne

Standortförderung beim Amt für Wirtschaft und Arbeit Kanton Zürich

i-factory (Musée des transports, Lucerne)

http://www.ubs.com/

http://www.oxocard.ch/
OXOcard
OXON

https://educatec.ch/
educaTEC

http://senarclens.com/
Senarclens Leu & Partner

http://www.abz.inf.ethz.ch/
Ausbildungs- und Beratungszentrum für Informatikunterricht der ETH Zürich.

`http://www.hepl.ch/`
Haute école pédagogique du canton de Vaud

`http://www.phlu.ch/`
Pädagogische Hochschule Luzern

`https://www.fhnw.ch/de/die-fhnw/hochschulen/ph`
Pädagogische Hochschule FHNW

`http://www.supsi.ch/home/supsi.html`
La Scuola universitaria professionale della Svizzera italiana (SUPSI)

`https://www.zhdk.ch/`
Zürcher Hochschule der Künste

C. Offres ultérieures

Devenez vous aussi membre de la SSIE
`http://svia-ssie-ssii.ch/la-societe/devenir-membre/`
et soutenez le Castor Informatique par votre adhésion
Peuvent devenir membre ordinaire de la SSIE toutes les personnes qui enseignent dans une école primaire, secondaire, professionnelle, un lycée, une haute école ou donnent des cours de formation ou de formation continue.
Les écoles, les associations et autres organisations peuvent être admises en tant que membre collectif.