INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Exercices et solutions 2020

Années HarmoS 11/12

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
schweizerischervereinfürinformatikinderausbildung//sociétésuissepourl'informatiquedansl'enseignement//societàsvizzeraperl'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

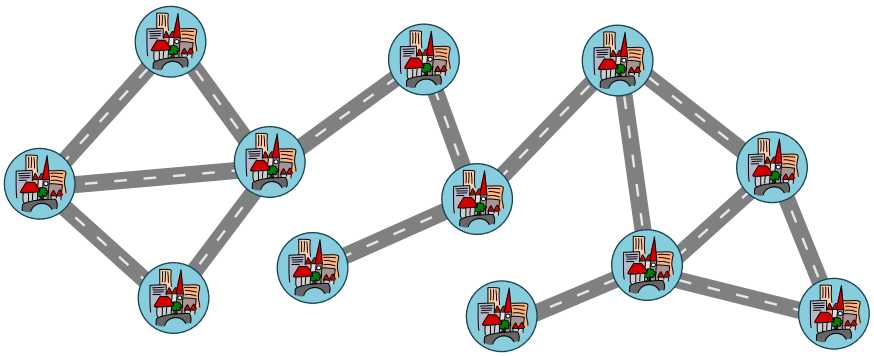
1. Épidémiologie

Castorland comporte 12 villes qui sont reliées par des routes. Les villes qui sont reliées de manière directe ou indirecte forment une communauté commerciale. La carte dans sa forme actuelle montre donc une seule communauté commerciale de 12 villes.

Pour endiguer une épidémie, la circulation doit être réduite. Le parlement des castors décide de fermer exactement deux routes pour diviser les villes en trois communautés commerciales.

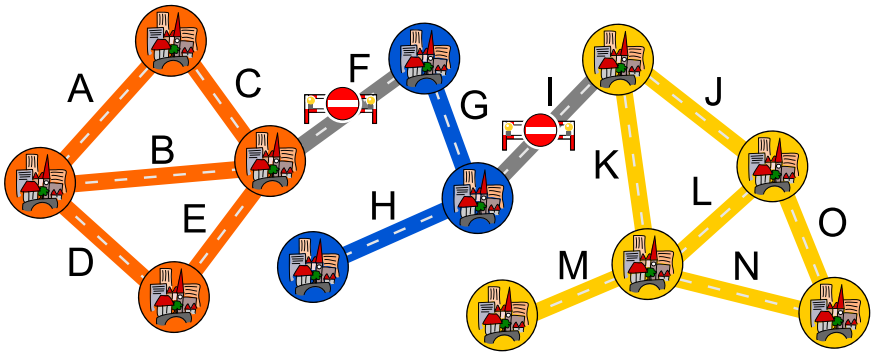
Pour n'isoler personne plus que nécessaire, la plus petite communauté commerciale devrait compter autant de villes que possible

Quelles sont les deux routes qui doivent être fermées ? Biffe-les.



Solution

La bonne réponse est : les routes F et I sur l'image ci-dessous doivent être fermées. De cette manière, trois communautés commerciales de 3, 4 et 5 villes, respectivement, sont formées.



C'est évident que nous ne devons considérer que les routes dont la fermeture engendre une division de la communauté commerciale car elles représentent une connexion unique. Nous avons en effet besoin de deux vraies divisions pour créer trois unités. Ça n'apporte donc rien de fermer la route B, par exemple, car on peut encore atteindre toutes les villes en passant par les routes A ou C. Les seules candidates pour la fermeture sont donc les routes F, G, H, I et M.

On arrive à la réponse du dessus en essayant toutes les dix possibilités de fermer deux de ces cinq routes. En tant qu'être humain, on remarque de plus tout de suite que la fermeture des routes H ou M isolerait une seule ville et n'entre donc pas en question. Cela limite encore le nombre de possibilités à considérer.

C'est de l'informatique !

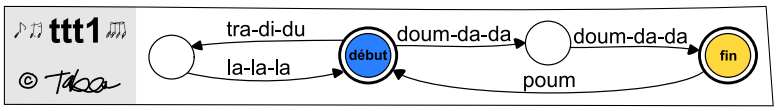
En informatique, on cherche souvent à diviser un certain réseau en *composantes connexes*. Toutes les parts d'une composante connexe sont reliées de manière directe ou indirecte, alors qu'il n'y a aucun lien entre différentes composantes connexes. L'utilisation dans les réseaux informatiques dans lesquels il est important de déterminer quels ordinateurs peuvent être atteints depuis quels autres est évidente. C'est aussi important de déterminer quels points sont reliés dans la reconnaissance optique de caractères (OCR).

Mots clés et sites web

- Composante connexe : `https://fr.wikipedia.org/wiki/Graphe_connexe`
- Parcours d'arbre : `https://fr.wikipedia.org/wiki/Parcours_d'arbre`

2. Les textes tendres de Tabea

Tabea a beaucoup de succès avec ses textes de chanson de la marque ttt : les textes tendres de Tabea. Ceux-ci peuvent être produits avec le diagramme ttt1 suivant :



Pour produire une chanson, Tabea commence par le « début » (début) et suit l'une des flèches partant de ce point. Lorsqu'il y a plusieurs possibilités, elle choisit. Elle chante les syllabes correspondantes le long de chemin dans l'ordre donné. Lorsqu'elle atteint la « fin » (fin), sa chanson peut se terminer ou continuer.

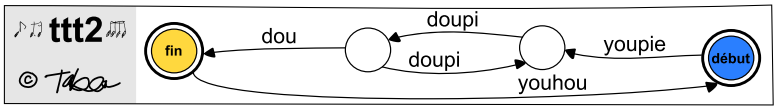
Voici des exemples de chansons possibles :

« Tra-di-du La-La-La Tra-di-du La-La-La
Doum-da-da Doum-da-da Poum Doum-da-da Doum-da-da »

Ou

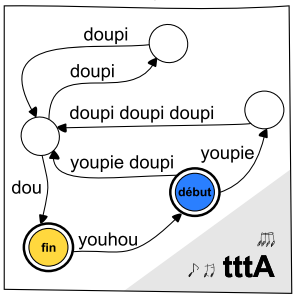
« Doum-da-da Doum-da-da Poum Tra-di-du La-La-La
Doum-da-da Doum-da-da Poum Tra-di-du La-La-La
Doum-da-da Doum-da-da Poum Doum-da-da Doum-da-da »

En novembre 2020, Tabea commence la production avec les nouveaux textes ttt2 :

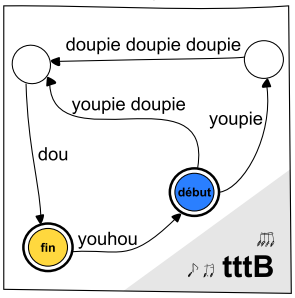


Lequel des diagrammes suivants permet de produire exactement les mêmes textes que ttt2 ?

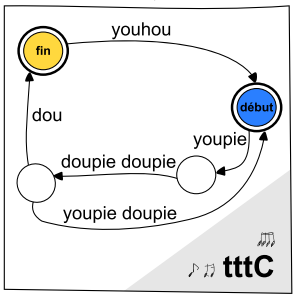
A)



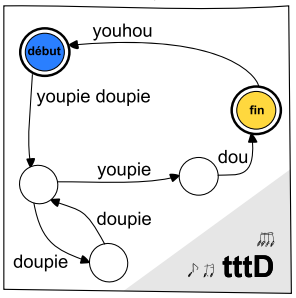
B)



C)

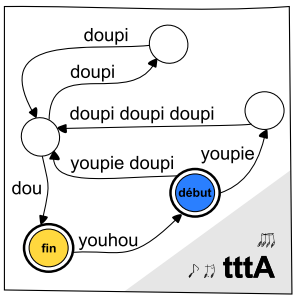


D)

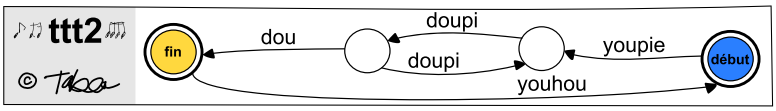


Solution

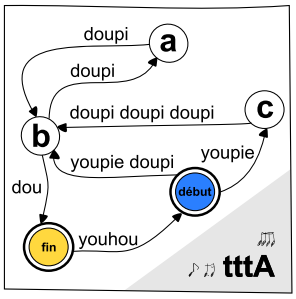
La bonne réponse est A), le diagramme tttA :



Lorsque l'on produit une chanson avec ttt2, elle commence dans tous les cas avec « youpie » qui est suivi d'au moins un « doupie ». On peut ensuite continuer soit avec « dou », soit avec un nombre pair de « doupie » suivi de « dou ». La chanson peut ensuite se terminer ou continuer avec un « youhou » avant de reprendre depuis le début.



Le diagramme tttA permet de produire exactement la même chose : depuis le « début », la chanson peut aller directement à **b** et ainsi commencer avec « youpie doupie » ou y aller en passant par **c** avec « youpie doupie doupie doupie ». Ensuite viennent, avec un détour par **a**, un nombre pair de « doupie » avant d'arriver à la fin de la chanson avec « dou ». Comme avec ttt2, on peut alors continuer la chanson avec un « youhou ».



Aussi bien ttt2 que tttA peuvent générer un nombre impair de « doupie » l'un après l'autre, après le « youpie » du début. Par contre, tttB ne peut générer qu'un ou trois « doupie » qui se suivent, et tttC seulement un ou deux. Quant à tttD, il peut certes générer un nombre impair de « doupie » l'un après l'autre, mais va toujours insérer, avant le « dou » final, un « youpie » de plus, ce que ttt2 ne fait pas.

La bonne réponse est donc tttA.

C'est de l'informatique !

Une tâche importante de l'informatique est de trouver des structures dans des données, par exemple dans le langage comme celui du texte d'une chanson. Les diagrammes représentent ce que l'on

appelle des automates finis, qui utilisent des règles très strictes pour générer et reconnaître certains langages. C'est primordial dans le développement des langages de programmation. Dans notre exemple, l'automate fini décrit l'ensemble des chansons que celui-ci peut générer.

La reconnaissance de motifs est également importante dans beaucoup d'autres domaines, comme par exemple le traitement du langage naturel.

Mots clés et sites web

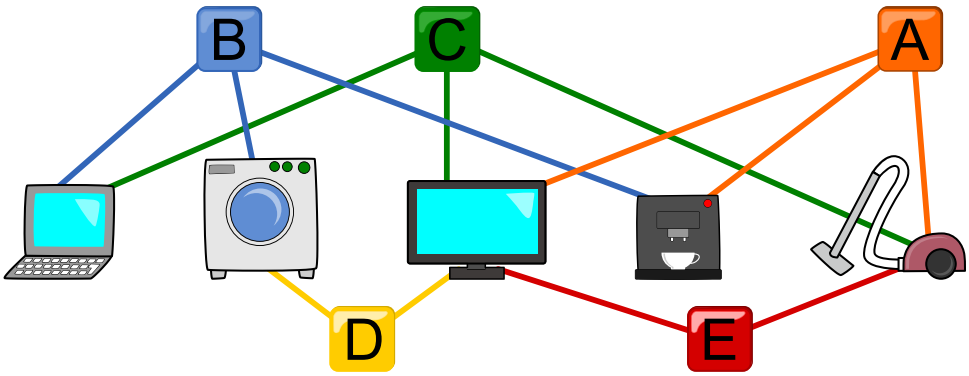
- Automate fini : `https://fr.wikipedia.org/wiki/Automate_fini_déterministe`
- Langage formel : `https://fr.wikipedia.org/wiki/Langage_formel`
- `https://sites.google.com/isabc.ca/computationalthinking/pattern-recognition`

3. Appareils ménagers

Dans la maison de Bruno le castor, il y a cinq appareils électriques (un ordinateur, un lave-linge, une télévision, une machine à café et un aspirateur) et cinq interrupteurs (A, B, C, D et E) pour allumer et éteindre des appareils. Le raccordement électrique est très inhabituel. Chaque interrupteur est connecté à plusieurs appareils, comme montré sur l'image en dessous. Chaque fois que l'on appuie sur un interrupteur, il change l'état de tous les appareils connectés : les appareils éteints s'allument et les appareils allumés s'éteignent.

Au départ, tous les appareils sont éteints. Si l'on appuie par exemple sur les interrupteurs A, C et E, l'aspirateur est allumé, car le premier bouton l'allume, le deuxième l'éteint et le troisième le rallume.

Sur quels interrupteurs Bruno doit-il appuyer pour que seules la télévision et la machine à café soit allumées ?



Solution

Lorsque l'on appuie, dans n'importe quel ordre, sur les interrupteurs B, C, D et E, seules la télévision et la machine à café sont allumées.

Nous pouvons aussi déterminer de manière systématique comment allumer et éteindre chaque appareil indépendamment des autres. Commençons avec deux combinaisons simples :

- A + E (appuyer sur A et B) contrôle la machine à café seulement.
- C + E (appuyer sur C et E) contrôle l'ordinateur seulement.

On observe ensuite que le lave-linge peut être contrôlé indépendamment des autres en appuyant d'abord sur B avant de remettre l'ordinateur et la machine à café dans le même état qu'avant, soit en appuyant sur A + E et sur C + E. Le lave-linge peut donc être contrôlé indépendamment du reste en appuyant sur B + A + E + C + E. L'interrupteur E est ici utilisé deux fois. Appuyer deux fois sur un interrupteur revient à ne pas l'utiliser du tout, on peut donc aussi contrôler le lave-linge indépendamment avec B + A + C. En utilisant cette méthode, on obtient la liste de combinaisons suivante pour contrôler les appareils séparément :

- Ordinateur : C + E
- Machine à café : A + E
- Lave-linge : A + B + C
- Télévision : A + B + C + D
- Aspirateur : A + B + C + D + E

Pour allumer seulement la télévision et la machine à café, nous devons donc appuyer sur A + B + C + D + A + E ce que l'on peut simplifier en B + C + D + E étant donné que les deux A s'annulent.

C'est de l'informatique !

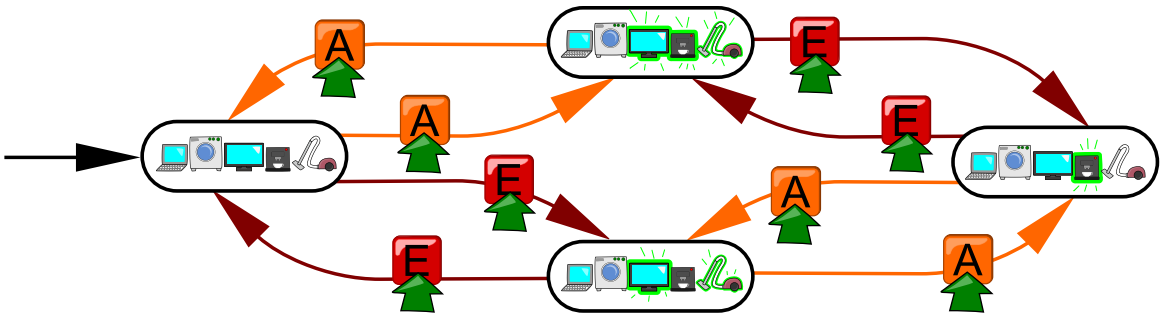
Le sytème composé des appareils et des interrupteurs peut être modélisé à l'aide d'un *automate fini* de la façon suivante.

Le système des cinq appareils a beaucoup d'*états* différents. Un de ces états est par exemple le cas où seule la télévision est allumée. Un autre état est le cas où tous les appareils sont éteints (comme tous les appareils sont éteints au départ, on appelle cet état l'*état initial*). Un autre état est le cas où seules la télévision et la machine à café sont allumées (c'est l'*état final* dans notre exercice, car c'est ce que nous voulons obtenir).

En appuyant sur un interrupteur, on fait passer le système d'un état à un autre. Par exemple, lorsque le système est à l'état initial et que l'on appuie sur l'interrupteur E, il passe à l'état auquel seuls la télévision et l'aspirateur sont allumés. Un tel passage d'un état à un autre s'appelle une *transition*.

Si l'on représente les états du système avec des cercles et les transitions avec des flèches, on obtient une image comme celle ci-dessous (pour des raisons de place, seuls quatre états et les transitions

entre ceux-ci sont représentés). L'état initial est marqué par une flèche spéciale. En informatique, ceci s'appelle un automate fini (un automate fini est simplement un graphe spécifique dans lequel les nœuds sont les états et les arêtes sont les transitions). On peut facilement voir sur l'image dans quel état l'on arrive lorsque l'on appuie sur différents interrupteurs.



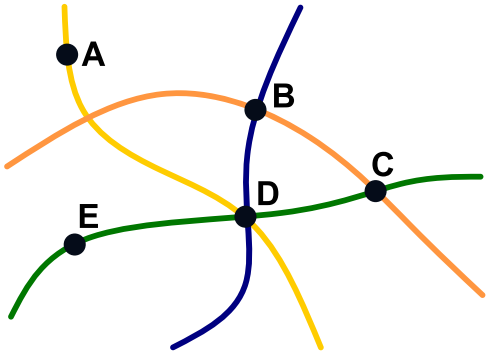
Dans cet exercice, il s'agit de déterminer comment passer de l'état initial (tous les appareils ont éteints) à l'état final (seules la télévision et la machine à café sont allumées). On veut donc trouver un chemin allant de l'état initial à l'état final. Le recherche de chemin dans des graphes est une tâche fondamentale de l'informatique.

Mots clés et sites web

- Automate fini : `https://fr.wikipedia.org/wiki/Automate_fini`

4. Réseau ferroviaire

Voici une carte de cinq villes et quatre lignes de train. Les points noirs représentent les villes, les lignes colorées les lignes de train.

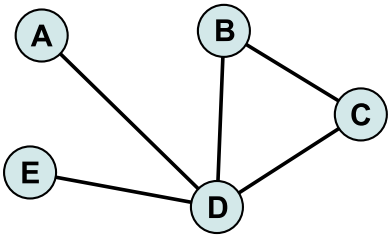


Un diagramme doit représenter cette carte de manière à ce que :

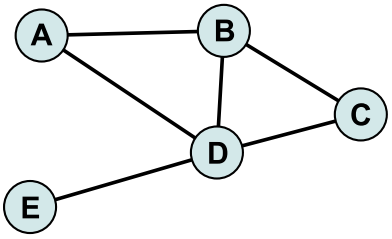
- les villes soient représentées par des cercles ;
- deux villes soient reliées d'un trait si elles sont situées sur la même ligne de train.

Quel diagramme représente la carte correctement ?

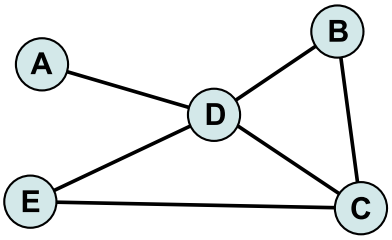
A)



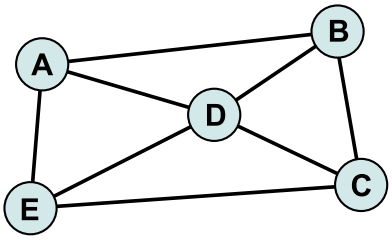
B)



C)

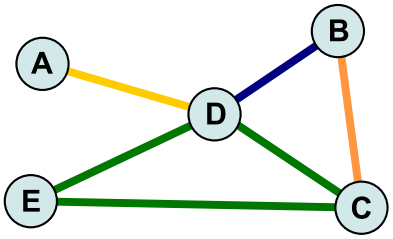


D)



Solution

La bonne réponse est C).



En observant bien la carte, on voit que :

- les villes A et D sont les deux sur la ligne de train jaune ;
- les villes B et C sont les deux sur la ligne de train orange ;
- les villes B et D sont les deux sur la ligne de train bleue ; et
- les villes C, D et E sont les trois sur la ligne de train verte.

Toutes les autres réponses sont fausses :

- Dans la réponse A, il manque le trait entre les villes C et E, qui doit être présent en raison de la ligne de train verte.
- Dans la réponse B, il y a le même problème que dans la réponse A, et il y a en plus un trait qui relie les villes A et B alors qu'elles ne sont pas sur la même ligne.
- Dans la réponse D, il y a deux traits entre les villes A et B ainsi qu'A et E, alors que la ville A n'est ni sur la même ligne que la ville B, ni que la ville E.

Il faut porter attention aux deux points suivants :

- Même s'il l'on peut arriver de la ville A à la ville B en prenant plusieurs lignes de train, les deux villes ne sont pas sur la même ligne.
- Même si une autre ville se trouve entre la ville C et la ville E sur la ligne verte, les deux villes sont sur la même ligne de train.

C'est de l'informatique !

Il y a plusieurs manières possibles de représenter la réalité. La carte plus haut avec les lignes colorées, par exemple, est déjà une représentation relativement abstraite de la situation réelle. Une forme de représentation très importante est le *graphe* – un diagramme qui comporte de nœuds (les cercles) et des arêtes (les traits entre les cercles). Cette forme de représentation est utilisée dans la solution.

Beaucoup de choses sont simplifiées par l'utilisation d'une forme de représentation adaptée. C'est pour cela qu'il est important de connaître beaucoup de formes de représentation pour programmer. En général, on ne peut pas dire qu'une forme de représentation soit mieux qu'une autre. Suivant l'usage prévu, une forme de représentation sera plus adaptée qu'une autre. Le graphe de la solution, par exemple, est pratique car on peut directement y voir que l'on peut aller de C à E sans changer

de train. On perd par contre l'information présente sur la carte que l'on passe par la ville D en allant de C à E avec cette ligne de train.

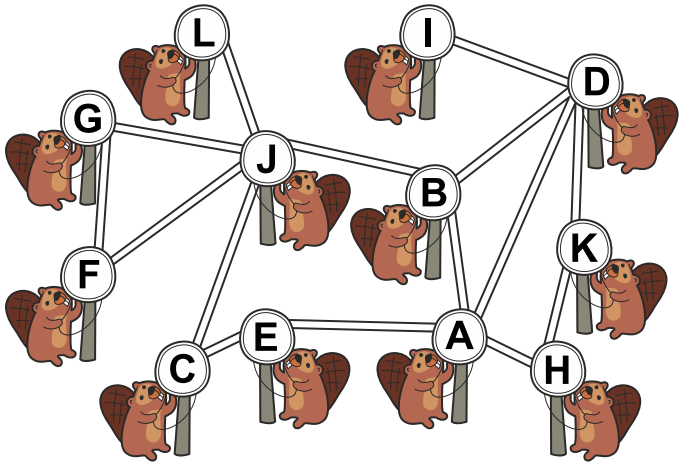
Mots clés et sites web

- Graphe : `https://fr.wikipedia.org/wiki/Graphe_(mathématiques_discrètes)`
- Théorie des graphes : `https://fr.wikipedia.org/wiki/Théorie_des_graphes`

5. Réseau de communication

Les castors aiment bien diffuser des informations entre eux. Pour cela, ils utilisent le réseau de communication ci-dessous. Lorsqu'un castor reçoit une nouvelle information, il l'envoie à tous les castors avec qui il partage un canal de communication direct (une ligne blanche). L'envoi d'information se passe en étapes. Une étape se passe entre l'envoi et la réception d'une information.

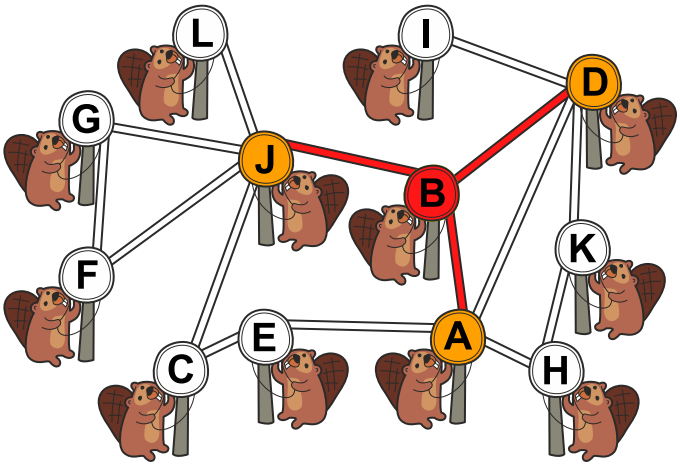
À partir de quel castor une nouvelle atteint-elle tous les autres castors le plus vite possible, c'est-à-dire en le moins d'étapes possible ?



Solution

La bonne réponse est le castor B. Il peut diffuser une information à tous les autres castors en deux tours.

Lors du premier tour, le castor B envoie l'information à ses voisins, donc aux castors A, D et J, avec qui il partage un canal de communication direct. L'image ci-dessous montre qui possède l'information après un tour.

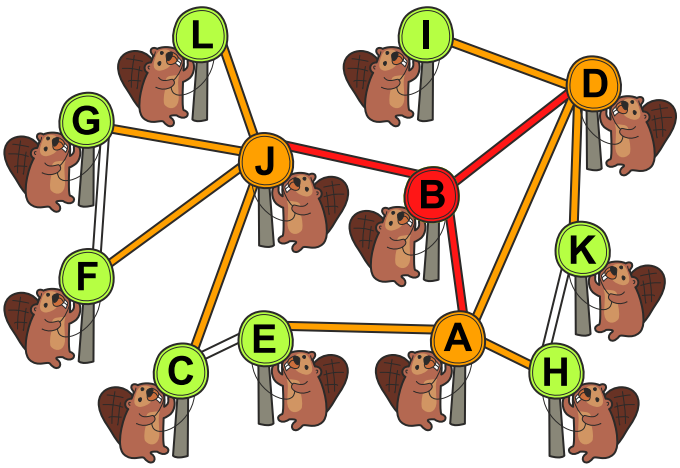


Lors du deuxième tour, les castors A, D et J envoient l'information à leurs voisins respectifs :

- le castor A envoie l'information aux castors E et H ;
- le castor D envoie l'information aux castors I et K ;
- le castor J envoie l'information aux castors C, F, G et L.

De plus, le castor B reçoit l'information trois fois en retour, car c'est aussi un voisin des castors A, D et J, mais comme ce n'est pas une nouvelle information, le castor B ne ne va pas la diffuser lors des tours suivants. Les castors A et D vont également s'envoyer l'information mutuellement par leur canal de communication direct, mais ne vont pas continuer à la diffuser non plus, étant donné qu'elle n'est pas nouvelle.

L'image ci-dessous montre la situation à la fin du deuxième tour.



L'information a donc atteint tous les castors en seulement deux tours.

Une diffusion plus rapide n'est pas possible, car pour cela, l'un des castors devrait être relié à tous les autres castors par une ligne blanche afin d'envoyer l'information à tout le monde en un tour.

Le castor B est le seul castor pouvant diffuser une information à tous les castors en deux tours : les castors C, E, F, G, H et J ne pourraient pas atteindre le castor I en deux tours, et les castors A, D, E, H, I et K ne pourraient pas atteindre le castor L en deux tours.

C'est de l'informatique !

On peut utiliser un *graphe* pour décrire le réseau de communication des castors. Chaque castor ce trouve à ce que l'on appelle un *nœud*, qui est dans ce cas désigné par une lettre. Les lignes blanches s'appellent des *arêtes* et relient deux nœuds. Les informations dont diffusées dans le réseau de communications en tours *synchronisés*, tous les castors envoient donc l'information en même temps. Lors d'un tour, chaque castor envoie les nouvelles informations à chacun de ses ses voisins. Ce que les castors font ici est appelé un *broadcast* sur un *réseau informatique* par les informaticiens et informaticiennes. Dans l'exercice ci-dessus, nous avons analysé à quelle vitesse un tel broadcast peut avoir lieu, c'est-à-dire à quelle vitesse une information peut atteindre tous les participants.

Un exercice encore plus exigeant consisterait à former un réseau de manière à ce qu'un broadcast rapide soit possible à partir de tous les nœuds sans que le nombre de connections ne soit trop grand.

Le nœud du castor B s'appelle le centre du graphe. D'une manière abstraite, le centre est un nœud qui minimise la distance entre lui-même et nœud le plus éloigné de lui. Il n'y a donc pas d'autre nœud ayant une plus petite distance à tous les autres nœuds. Dans l'exercice précédent, il n'y a qu'un centre. Dans certains graphes, il peut aussi y avoir plusieurs nœuds qui minimisent chacun la distance au nœud le plus éloigné ; un graphe peut donc avoir plusieurs centres.

Ce n'est pas toujours aussi simple de trouver le centre d'un graphe que dans cet exercice. C'est possible par exemple que la transmission entre certains nœuds reliés directement dure plusieurs tours. Les graphes peuvent aussi être beaucoup plus grands et complexes. Pour de tels graphes, on peut par exemple utiliser l'*algorithme de Floyd-Warshall* pour trouver un centre de manière efficiente.

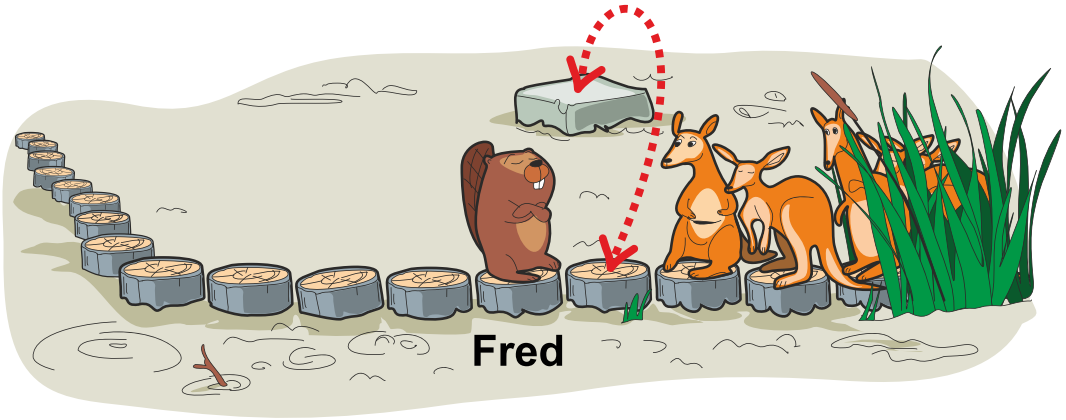
Mots clés et sites web

- Graphe : `https://fr.wikipedia.org/wiki/Graphe_(mathématiques_discrètes)`, `https://fr.wikipedia.org/wiki/Chaîne_(théorie_des_graphes)`
- Centre d'un graphe : `https://fr.wikipedia.org/wiki/Centralité#Centralité_de_proximité`
- Algorithme de Floyd-Warshall : `https://fr.wikipedia.org/wiki/Algorithme_de_Floyd-Warshall`

6. Fred le têtu

Des kangourous se déplacent en direction du castor Fred sur un chemin de rondins. Le chemin est assez étroit, ce qui fait que Fred et les kangourous ne peuvent pas s'y croiser. Il y a un certain rondin depuis lequel les kangourous peuvent sauter sur une pierre pour libérer le chemin avant de retourner le même rondin comme montré sur l'image. Un seul animal peut se tenir sur chaque rondin et sur la pierre.

Fred aimerait avancer. Il est assez têtu et n'est prêt à reculer d'un rondin que 10 fois. Par contre, il avance d'un rondin aussi souvent que nécessaire.



Quel est le nombre maximal de kangourous que Fred peut laisser passer ?

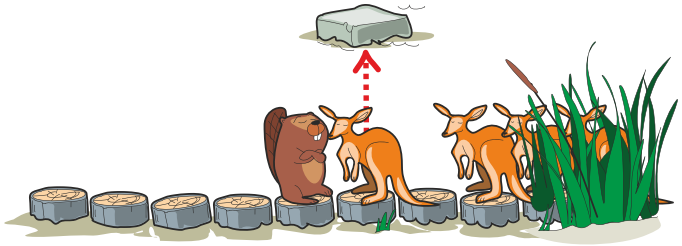
A) Plus de 10 kangourous
B) Exactement 10 kangourous
C) Exactement 6 kangourous
D) Exactement 4 kangourous
E) Moins de 4 kangourous
F) On ne peut pas savoir exactement

Solution

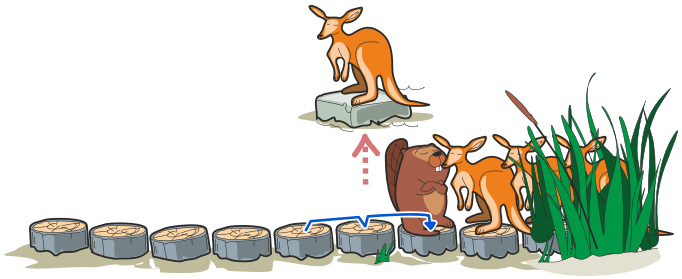
Fred peut laisser passer au maximum 6 kangourous.

Un kangourou dépasse Fred de la manière suivante :

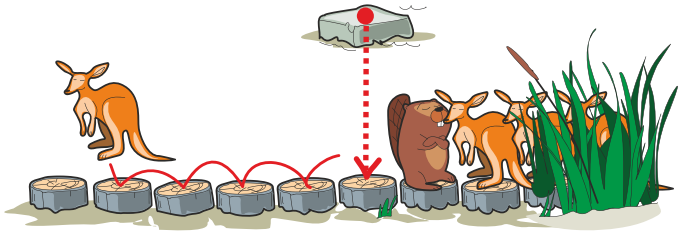
Le kangourou saute sur la pierre.



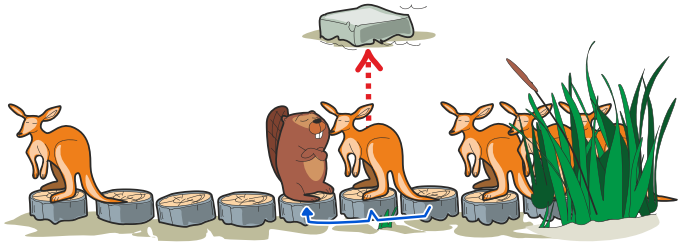
Fred avance de deux rondins.



Le kangourou revient sur le chemin en sautant et continue.



Si Fred recule à présent de deux rondins, il est à nouveau à sa position de départ et peut répéter le procédé pour laisser passer un autre kangourou.



Comme il ne recule que de dix rondins au maximum, il peut faire cela cinq fois, et donc laisser passer six kangourous en comptant le premier.

C'est de l'informatique !

En informatique, les tâches sont résolues entre autres en utilisant des algorithmes : des suites d'*instructions* qui sont effectuées pas à pas – exactement comme « Fred avance d'un rondin » ou « un kangourou saute sur la pierre ».

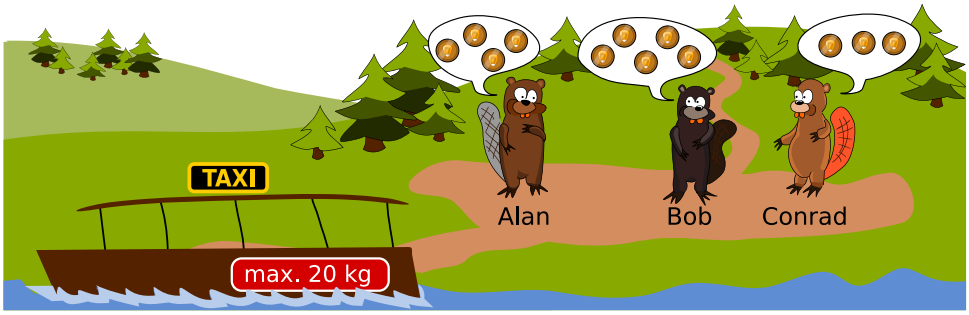
Dans ce qu'on appelle une *boucle*, une suite d'instructions peut être répétée. De cette manière, des tâches répétitives peuvent êtres exécutées plusieurs fois de manière fiable. Pour cela, c'est souvent un avantage de commencer chaque passage de la boucle par la même situation – ce qu'on appelle un

invariant de boucle. Dans notre cas, Fred doit toujours retourner à sa position de départ afin que le même procédé fonctionne à nouveau pour le kangourou suivant.

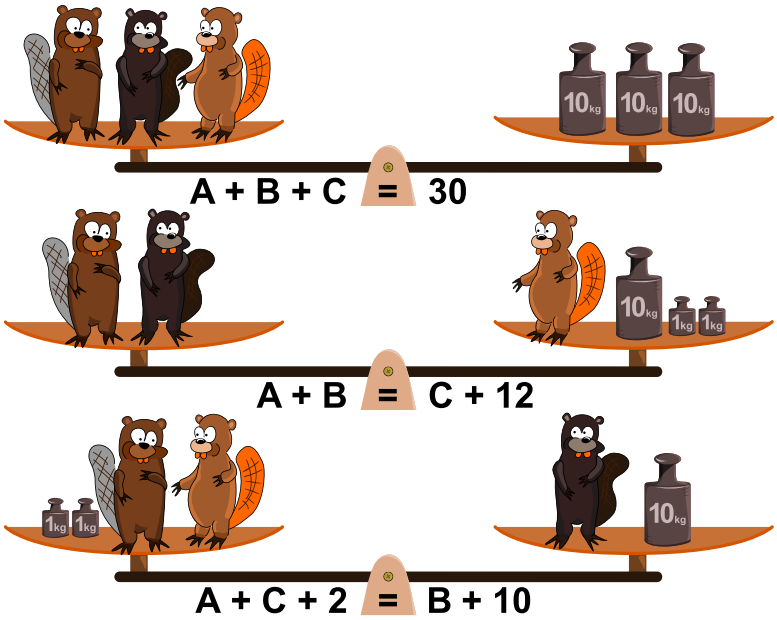
Mots clés et sites web

- Algorithme : `https://fr.wikipedia.org/wiki/Algorithme`
- `https://fr.wikipedia.org/wiki/Programmation_structurée`
- Boucle : `https://fr.wikipedia.org/wiki/Structure_de_contrôle#Boucles`
- Invariant : `https://fr.wikipedia.org/wiki/Invariant_de_boucle`

7. 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

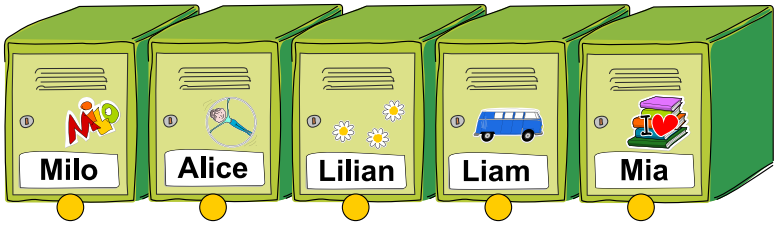
- Optimisation linéaire en nombre entiers : `https://fr.wikipedia.org/wiki/Optimisation_linéaire_en_nombres_entiers`
- Contrainte : `https://fr.wikipedia.org/wiki/Contrainte_(mathématiques)`
- Séparation et évaluation : `https://fr.wikipedia.org/wiki/Séparation_et_évaluation`
- Méthode des plans sécants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

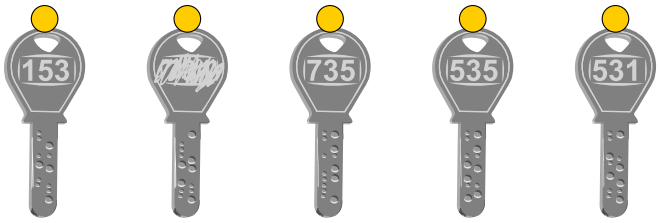
8. Casiers

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 ».

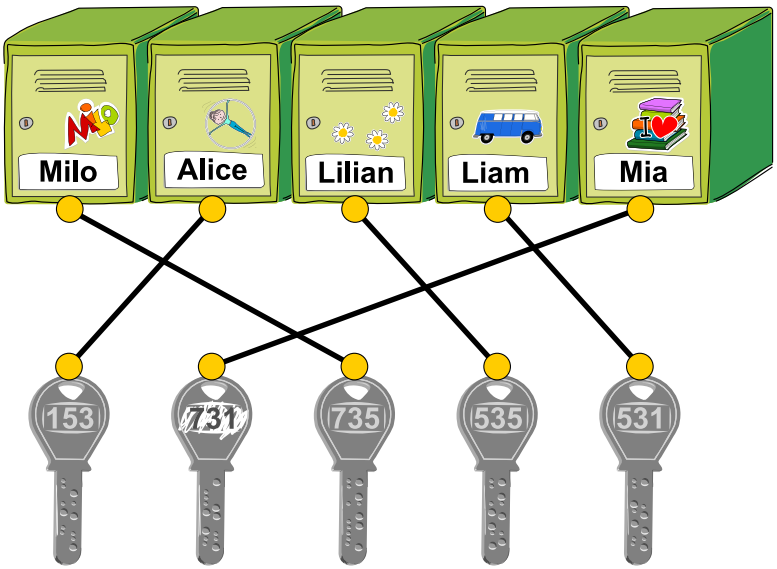
Relie les clés aux bons casiers. Pour cela, trace des lignes entre les points jaunes.





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.

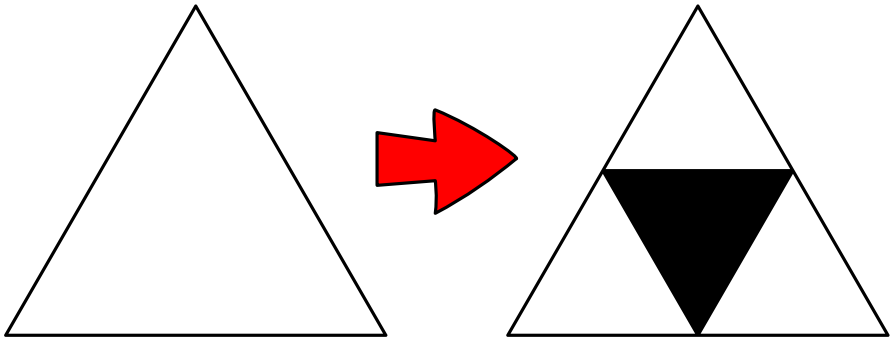
Si l'on n'utilise pas seulement dix chiffres pour le chiffrement, mais un symbole pour chaque lettre, on peut utiliser une telle substitution monoalphabétique comme un code secret simple. Malheureusement, la méthode de chiffrement par substitution monoalphabétique n'est pas très sûre, parce que l'on 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

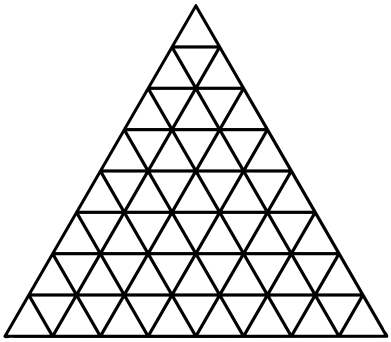
- Code, substitution monoalphabétique : `https://fr.wikipedia.org/wiki/Chiffrement_par_substitution#Substitution_monoalphabétique`
- Cryptographie : `https://fr.wikipedia.org/wiki/Cryptographie`

9. 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 :

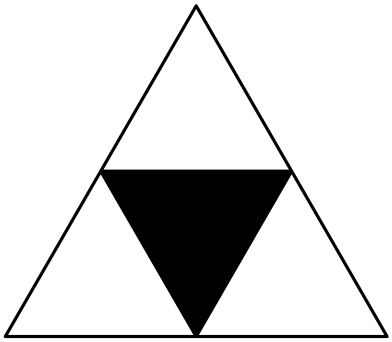


Dessine la figure obtenue après trois étapes. Pour cela, colorie les bons petits triangles en noir.

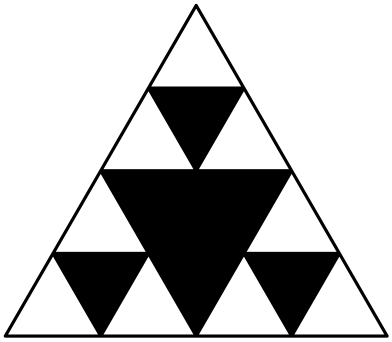


Solution

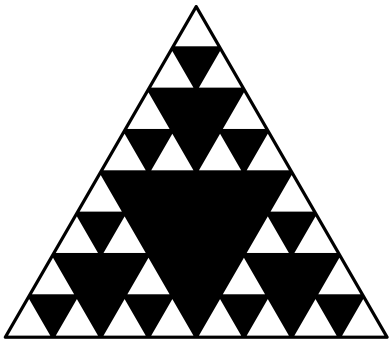
Après la première étape, le triangle central est noir et il y a trois triangles blancs :



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

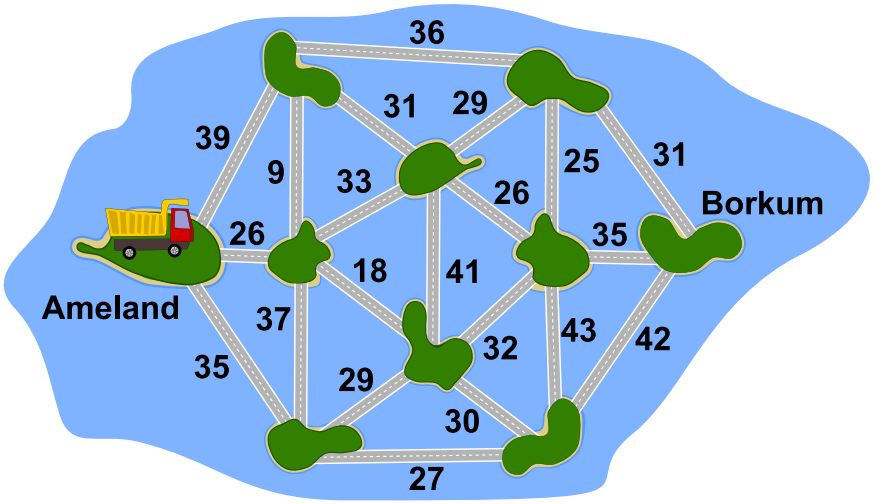
- Triangle de Sierpiński : `https://fr.wikipedia.org/wiki/Triangle_de_Sierpiński`
- Récursivité : `https://fr.wikipedia.org/wiki/Récursivité`
- Fractale : `https://fr.wikipedia.org/wiki/Fractale`
- `https://fr.wikipedia.org/wiki/Wacław_Sierpiński`
- `https://fr.wikipedia.org/wiki/Tours_de_Hanoï#Solution_récursive`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

10. 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.

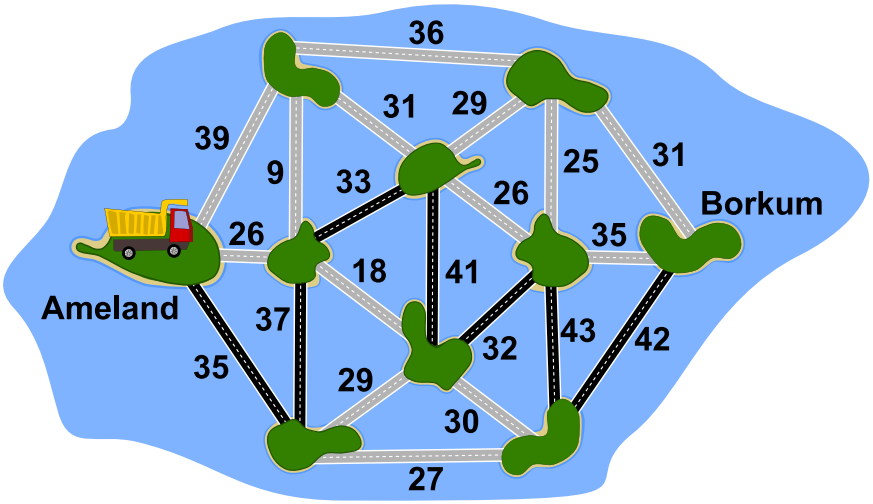
Le castor Knuth aimerait amener du sable sur une plage de l'île de Borkum. Il veut donc transporter autant de sable que possible de l'île d'Ameland à l'île de Borkum en un seul voyage. La longueur de la route à parcourir lui est égale, mais il ne veut prendre aucun pont plus d'une fois.

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

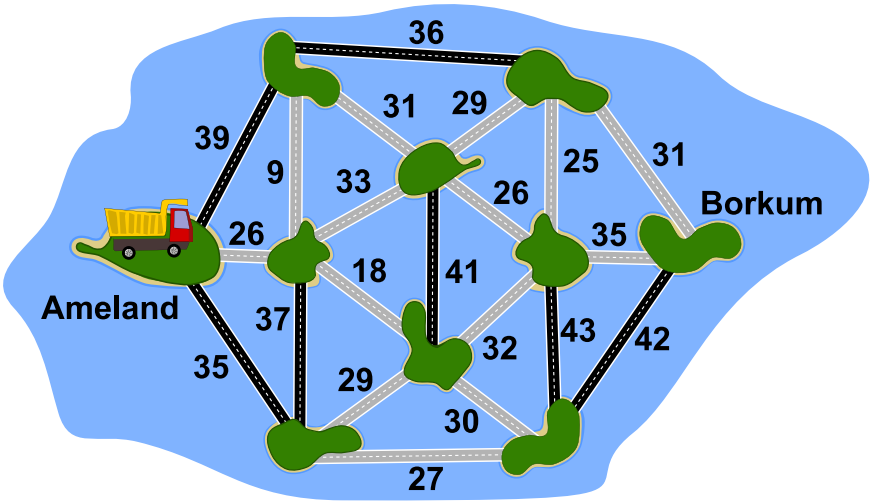


Solution

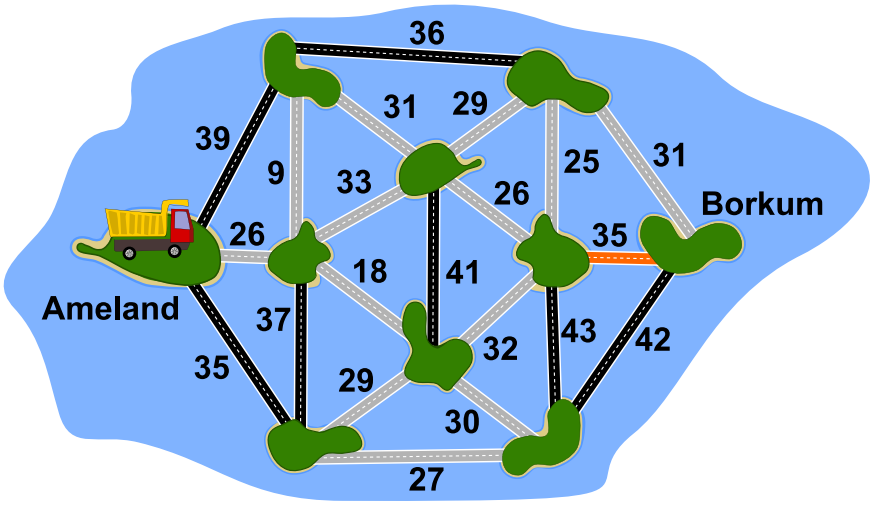
Le poids maximal d'un camion pour ce voyage est de 32 tonnes. Il suit par exemple la route suivante :



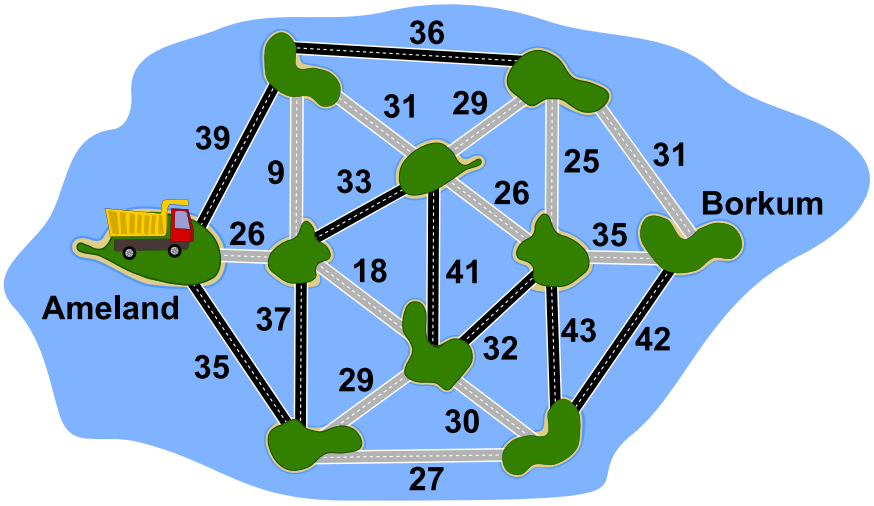
Afin de trouver ce chemin, nous pouvons par exemple commencer par virtuellement enlever tous les ponts de la carte. Nous les trions par capacité de charge. Nous commençons par ajouter à la carte les ponts ayant la plus grande capacité, puis ceux ayant une capacité de charge moindre, et ainsi de suite. 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`

11. 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.

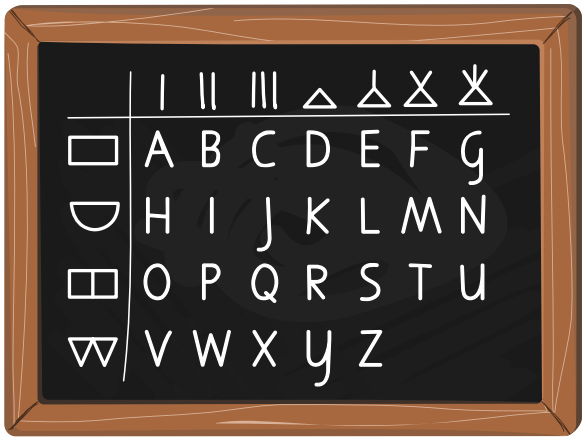
Reconstruis le texte original à partir du cryptogramme suivant (déchiffre le cryptogramme). Laquelle des quatre solutions proposées est-elle juste ?

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.

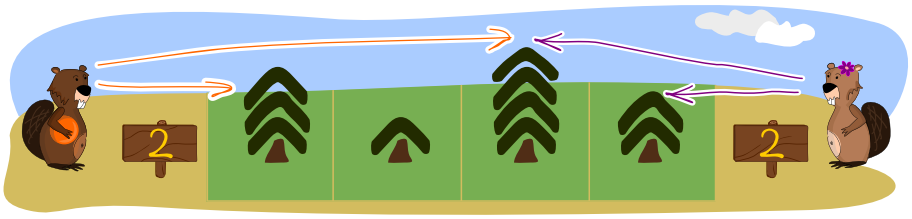
Mots clés et sites web

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

12. 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 :

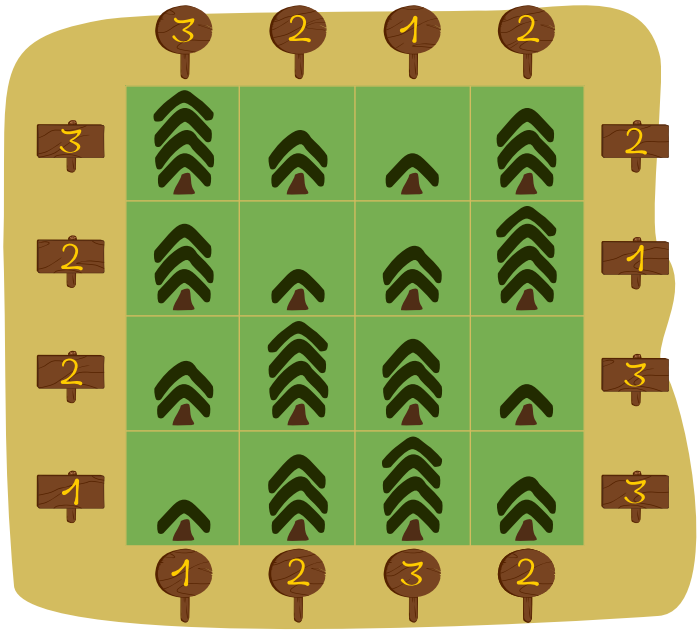
- dans chaque ligne, il y a exactement un arbre de chaque hauteur ;
- dans chaque colonne, il y a exactement un arbre de chaque hauteur.



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.



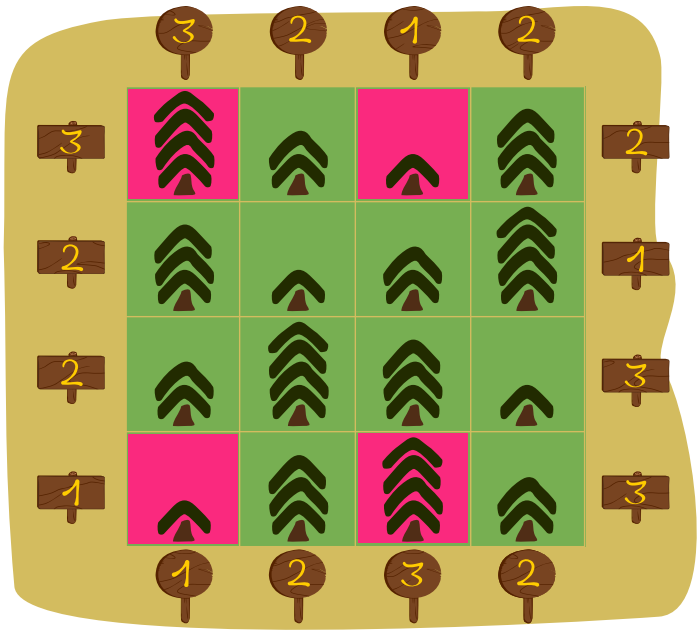
Solution

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*.

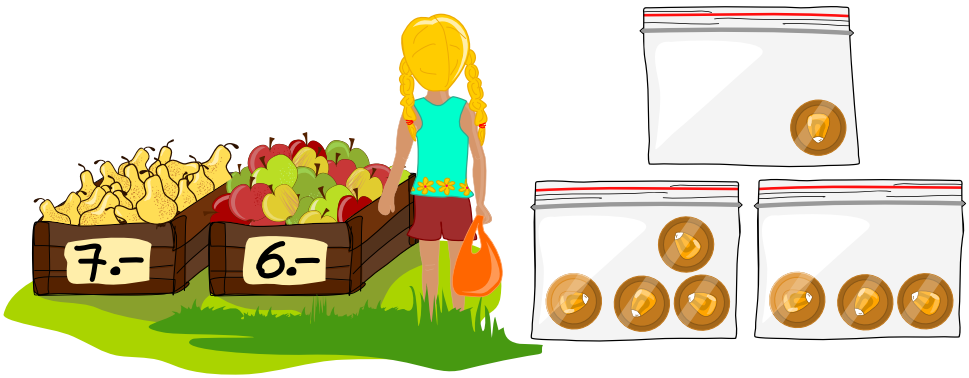
Troisièmement, on peut utiliser de tels champs d'arbres avec des panneaux pour créer des *codes correcteurs*. Des erreurs arrivant lors de l'entrée des données ou du transfert d'information peuvent ainsi être automatiquement reconnues ou même corrigées.

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`

13. 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.



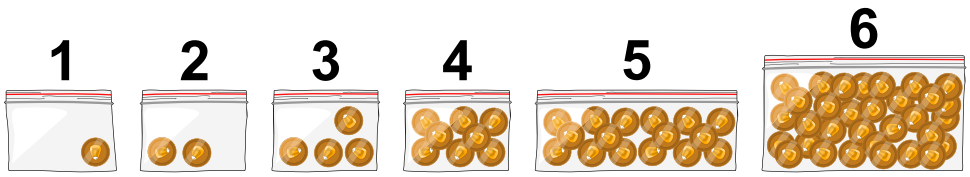
Aujourd'hui, Bina a pris 63 pièces pareilles. Elle aimeraient les répartir dans différents sachets de manière à pouvoir payer tous les montants entre 1 et 63 pièces exactement et sans devoir ouvrir de sachet.

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`

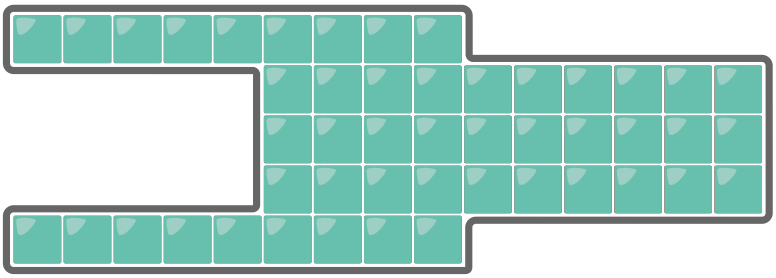
14. Chauffage au sol

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 ?

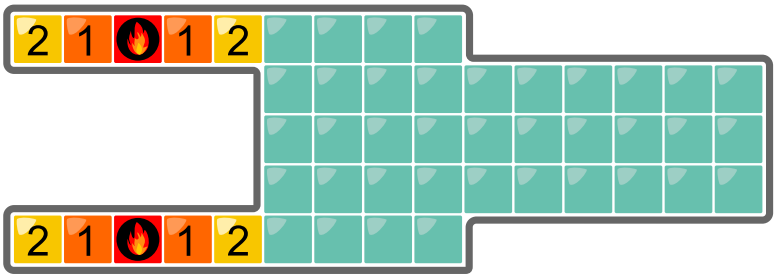


Solution

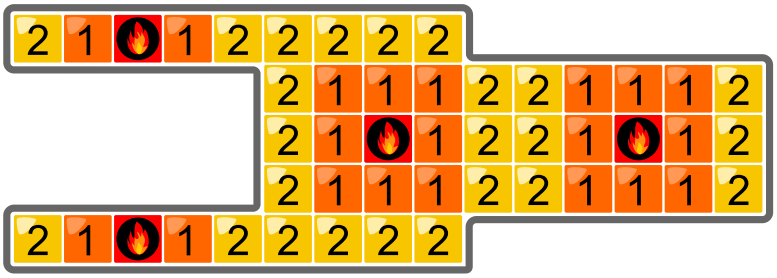
Lorsque les quatre hotspots sont installés comme dans l'image ci-dessous, toutes les catelles de la salle de bain sont chaudes deux minutes après avoir allumé le chauffage.

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.

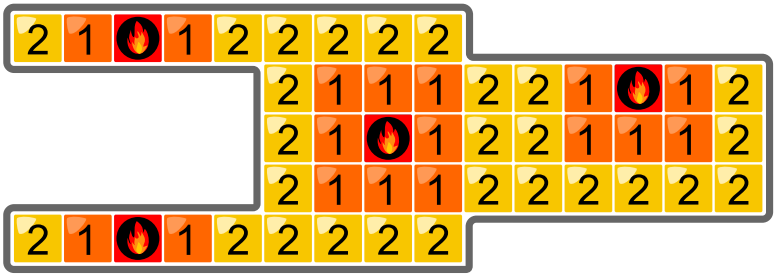
Il serait maintenant logique de commencer à répartir les hotspots dans les deux couloirs à gauche. En mettant un hotspot au milieu de chaque couloir, toutes les catelles des couloirs sont chauffées au bout de deux minutes :

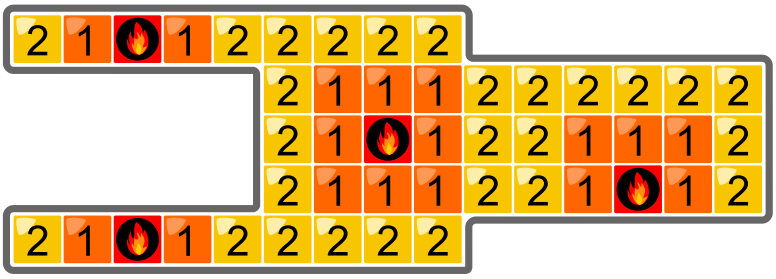


On peut placer les deux autres hotspots comme cela :



Les deux placements suivants sont également possibles :





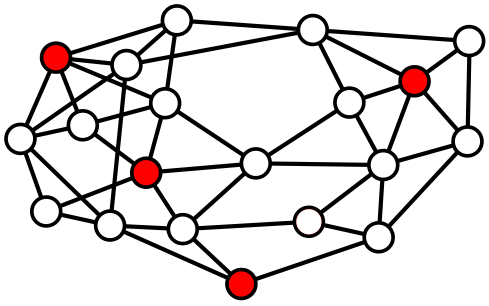
Si la salle de bain avait une forme différente, deux hotspots pourraient suffire à chauffer la même surface en deux minutes.

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

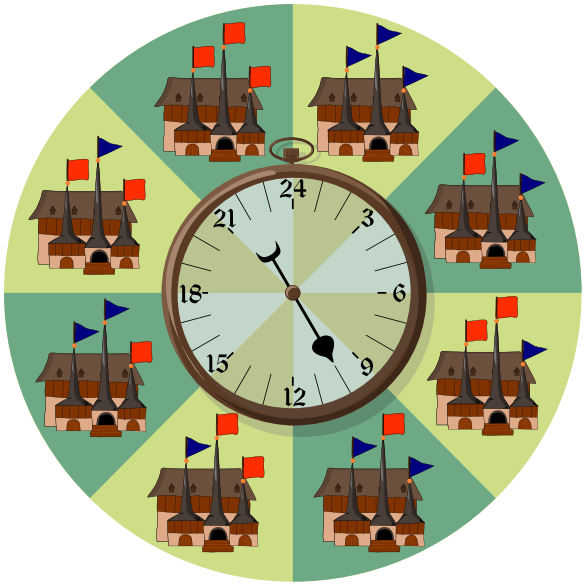
- Ensemble dominant : `https://fr.wikipedia.org/wiki/Ensemble_dominant`

15. 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.

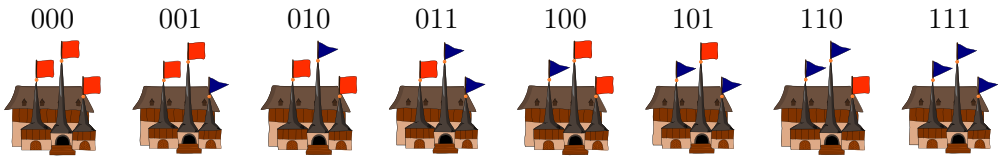
L'arrangement des drapeaux ci-dessus ne demande le changement que d'un seul drapeau à presque chaque transition. Il n'y a qu'à minuit où trois drapeaux doivent être changés d'un coup. Les castors aimeraient trouver un arrangement plus commode qui permette de ne changer qu'un seul drapeau à chaque transition.

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



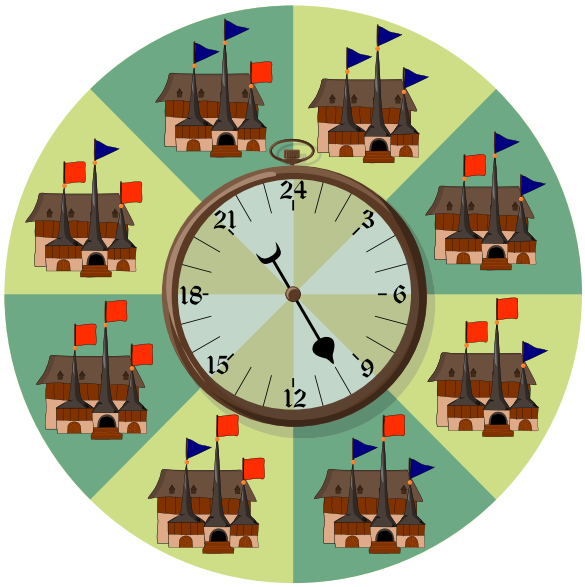
Solution

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.

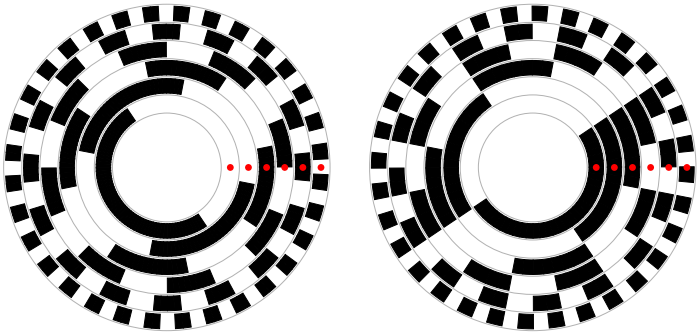
Nous écrivons à nouveau cette nouvelle suite de nombre à l'envers à la suite de la précédente en changeant cette fois la première position de 0 à 1. Nous obtenons ainsi 000, 001, 011, 010, 110, 111, 101, 100, ce qui remplit à nouveau notre condition. Nous avons trouvé la solution recherchée.

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`

A. Auteur·e·s des exercices

Michael Barot
Maksim Bolonkin
Andrey Brodnik
Lucia Budinská
Marios O. Choudary
Kris Coolsaet
Valentina Dagienė
Christian Datzko
Susanne Datzko
Amirmohammad Djazbi
Lidia Feklistova
Fabian Frei
Jens Gallenbacher
Tom Grubb
Yasemin Gulbahar
Mathias Hiron
Juraj Hromkovič
Alisher Ikramov
Thomas Ioannou
Mile Jovanov
Ungyeol Jung
Vaidotas Kinčius
Sophie Koh
Dennis Komm
Ritambhra Korpal
Chia-Yi Ku
Regula Lacher
Marielle Léonard
Judith Lin
Lynn Liu
Matija Lokar
Vu Van Luan
Pedro Marcelino
Hamed Mohebbi
Kwangsik Moon
Anna Morpurgo
Xavier Muñoz
Ágnes Erdősné Németh
Andrei Nicolicioiu
Elsa Pellet
Jean-Philippe Pellet
Peter Rossmanith
Eljakim Schrijvers
Vipul Shah
Maiko Shimabuku
Timur Sitdikov
Emil Stankov
Maciej M. Sysło
Monika Tomcsányiová
Meng-ting Tsai
Jiří Vaníček
Khairul Anwar Mohamad Zaki

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.