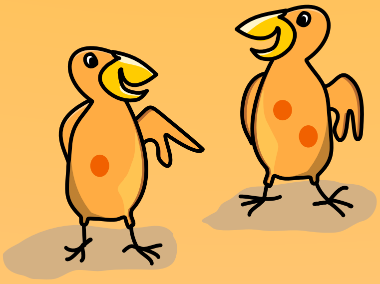
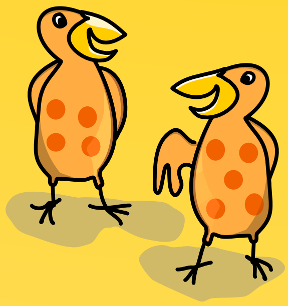
INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Exercices et solutions 2021

Années HarmoS 11/12



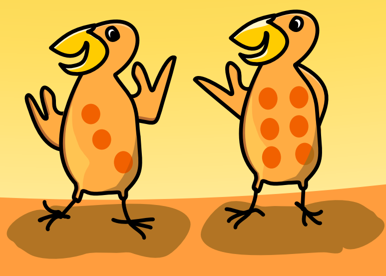
https://www.castor-informatique.ch/



Éditeurs :

Susanne Datzko, Elsa Pellet, Jean-Philippe Pellet,
Fabian Frei, Gabriel Parriaux

SSIE
www.svia-ssie-ssii.ch
schweizerischervereinfürinformatikinderausbildung//sociétésuissepourl'informatiquedansl'enseignement//societàsvizzeraperl'informaticanell'insegnamento



Ont collaboré au Castor Informatique 2021

Masiar Babazadeh, Susanne Datzko, Fabian Frei, Martin Guggisberg, 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
Bernadette Spieler : Pädagogische Hochschule Zürich

Le choix des exercices a été fait en collaboration avec les organisateur de Bebras en Allemagne, Autriche, Hongrie, Slovaquie et Lithuanie. Nous remercions en particulier :
Valentina Dagienė, Tomas Šiaulys, Vaidotas Kinčius : Bebras.org
Wolfgang Pohl, Hannes Endreß, Ulrich Kiesmüller, Kirsten Schlüter, Michael Weigend : Bundesweite Informatikwettbewerbe (BWINF), Allemagne
Wilfried Baumann, Liam Baumann, Anoki Eischer, Thomas Galler, Benjamin Hirsch, Martin Kandlhofer, Katharina Resch-Schobel : 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 : UK Bebras Administrator, Royaume-Uni

Pour le support pendant les semaines du concours, nous remercions en plus :
Hanspeter Erni : Direction, école secondaire de Rickenbach

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

Ces brochures sont dédiées à la mémoire de Martin Guggisberg.

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 2021 a été réalisé par la Société Suisse pour l'Informatique dans l'Enseignement (SSIE) et soutenu par la Fondation Hasler.

HASLERSTIFTUNG

Cette brochure a été produite le 24 août 2022 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 2021.

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 2021 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 pour l'Informatique dans l'Enseignement
Castor Informatique
Gabriel Parriaux

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

Table des matières

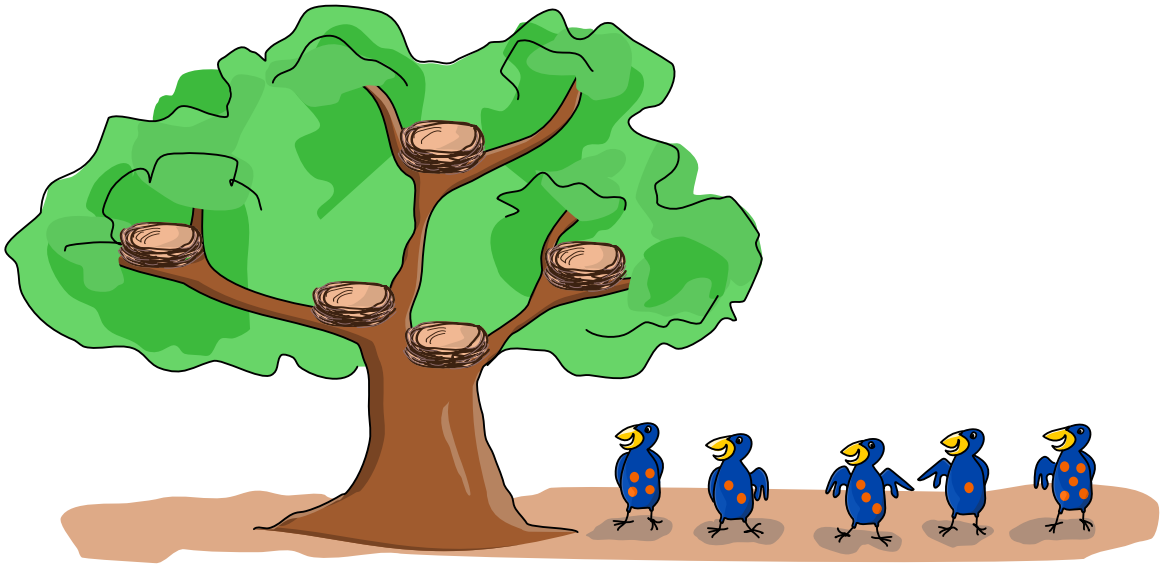
1. Emménagement

Les tachetés sont des oiseaux qui ont des points sur leurs plumes. Cinq tachetés sont à côté d'un arbre. Ils grimpent dans l'arbre l'un après l'autre — de gauche à droite — et emménagent dans les nids vides. Le tacheté avec quatre points commence. Chaque tacheté procède comme suit :

Il commence en bas de l'arbre. Il effectue les étapes suivantes jusqu'à ce qu'il ait trouvé un nid vide :

1. Il grimpe jusqu'à ce qu'il trouve un nid.
2. Si le nid est vide, il y emménage et reste là.
3. Sinon, il continue à grimper :
   - vers la gauche si le tacheté dans le nid a plus de points que lui ;
   - vers la droite si le tacheté dans le nid a le même nombre ou moins de points que lui.

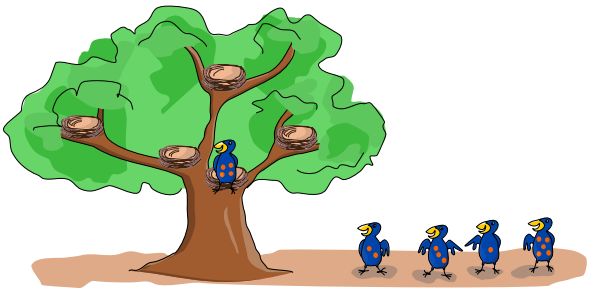
Où se trouvent les tachetés à la fin ? Place chaque tacheté dans le bon nid.



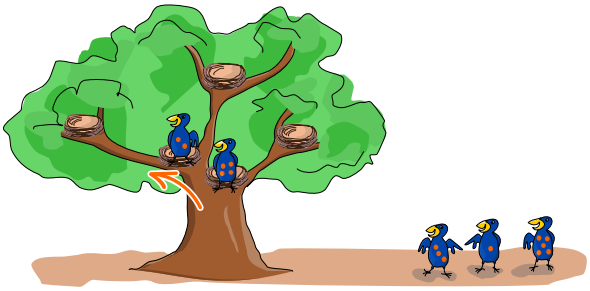
Solution

On obtient la bonne solution de la manière suivante :

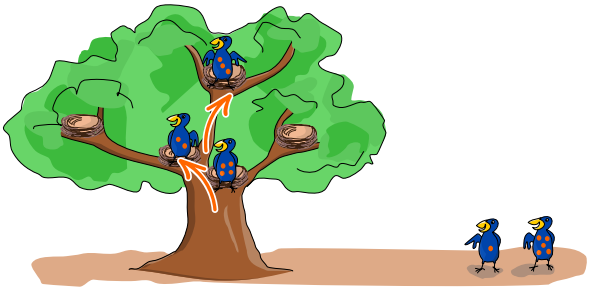
Le premier tacheté, celui à 4 points, arrive dans le nid du bas et y emménage.



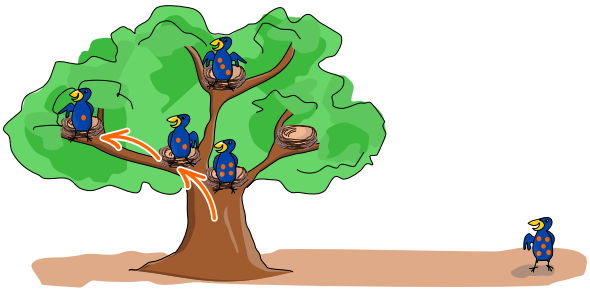
Le deuxième tacheté a 2 points. Le nid du bas est maintenant occupé par le premier tacheté à 4 points. Comme 4 est plus grand que 2, le deuxième tacheté continue à grimper vers la gauche et emménage dans le premier nid vide.



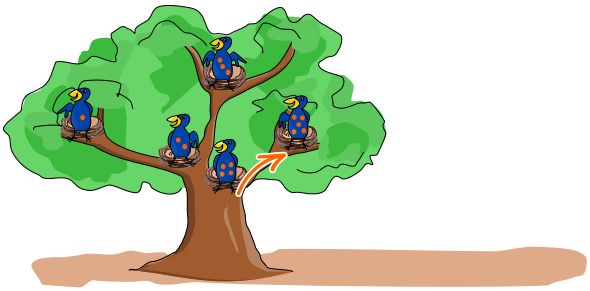
Le troisième tacheté a 3 points. Il grimpe vers la gauche après le nid du bas dans lequel est le tacheté à 4 points, car 4 est plus grand que 3. Le tacheté à 2 points est dans le nid suivant. Comme 3 est plus grand que 2, le troisième tacheté continue de grimper vers la droite. Il emménage dans le prochain nid vide. C'est le nid le plus haut.



Le quatrième tacheté a 1 point. Comme tous les autres tachetés ont plus de points que lui, il grimpe vers la gauche après chaque nid occupé. Il arrive ainsi dans le nid tout à gauche et y emménage.



Le dernier tacheté a 5 points. Comme aucun tacheté n'a plus de points que lui, il grimpe vers la droite après chaque nid occupé. Il fait cela une fois après le nid du bas et emménage ensuite dans le nid tout à droite.



C'est de l'informatique !

La manière dont les tachetés choisissent leur nid a un avantage intéressant : on peut vite trouver un tacheté particulier. Si le tacheté que tu cherches a moins de points que celui que tu es en train

de regarder, tu dois continuer à chercher à sa gauche. Sinon, tu continue à chercher à sa droite. A chaque évaluation d'un oiseau, tu peux ainsi réduire le domaine de recherche à une de deux moitiés. C'est pour cela que tu vas rapidement trouver ton tacheté.

Il y a beaucoup de manières d'organiser des données ; on parle de différentes *structures de données*. La structure de donnée utilisée dans cet exercice est un *arbre binaire de recherche*. Le mot « binaire » vient du mot latin *bis* qui veut dire « deux fois ». En effet, il y a au maximum deux branches plus petites qui partent d'une branche (là où se trouve un nid dans cet exercice). Les arbres binaires de recherche sont utilisés en programmation lorsque beaucoup de données doivent être trouvées rapidement. Ils sont en général beaucoup plus grands que le petit arbre de l'exercice. Il y a encore une différence supplémentaire : l'arbre de cet erercice acceuille un nombre fixe de cinq tachetés, alors que l'on peut en général toujours ajouter des données à un arbre binaire de recherche. On ajoute pour cela simplement une nouvelle branche au bout d'une branche existante, ce qui agrandit l'arbre. Les stuctures de données pouvant être modifiées de cette façon s'appellent *structures de données dynamiques*.

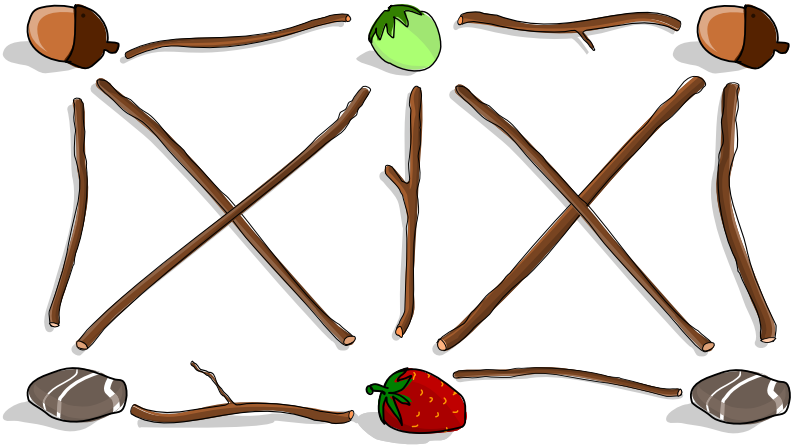
Mots clés et sites web

- Arbre binaire de recherche :
  `https://fr.wikipedia.org/wiki/Arbre_binaire_de_recherche`
- Structure de données : `https://fr.wikipedia.org/wiki/Structure_de_données`

2. Voleur de fraise

Anja veut créer une œuvre d'art dans le jardin et a ramassé pour cela différents objets : plusieurs glands, noisettes et cailloux ainsi qu'une fraise. Elle met quelques objets dans l'herbe.

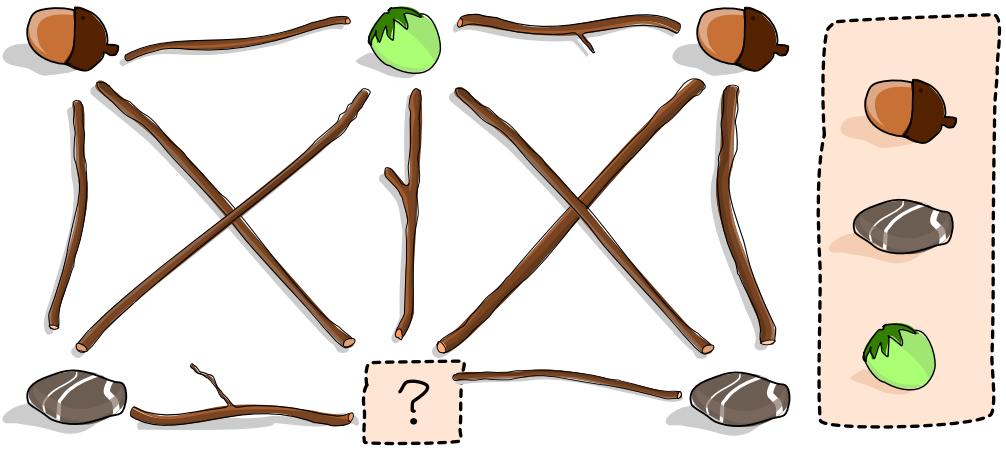
Ensuite, Anja dispose des branches entre ces objets. Elle suit pour cela la règle suivante : une branche ne doit pas se trouver entre deux objets pareils, par exemple entre deux glands. Voici l'œuvre d'art terminée :



Le frère d'Anja vient et mange la fraise pendant qu'elle n'est pas là.

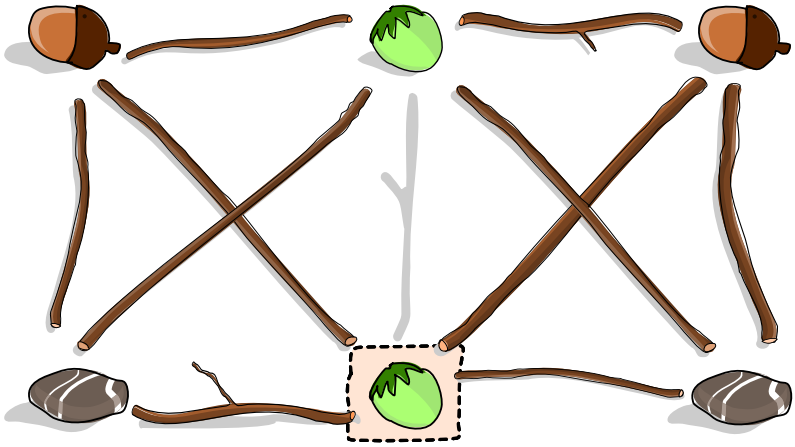
Peux-tu aider le frère d'Anja à dissimuler son méfait ?

Place un autre objet à la place de la fraise et enlève exactement une branche. L'œuvre d'art modifiée doit respecter la règle d'Anja.



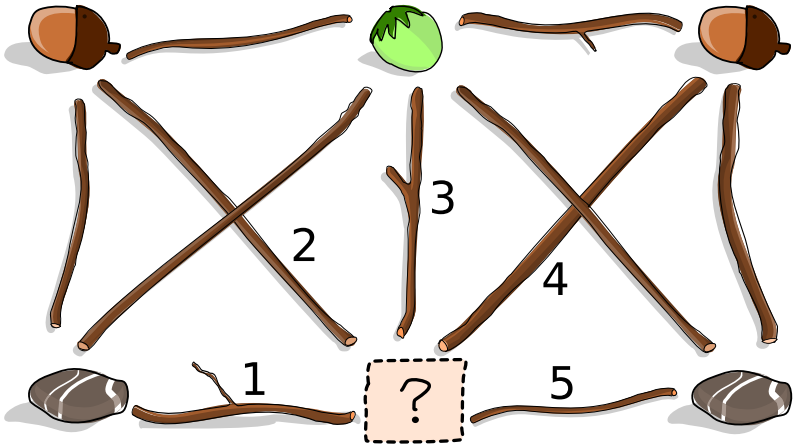
Solution

Lorsque l'on remplace la fraise par une noisette, la branche 3 ne respecte plus la règle d'Anja : elle se trouve entre deux objets pareils, à savoir deux noisettes. Cette branche doit donc être enlevée.



Les deux autres remplacements possibles nécessitent d'enlever plus d'une branche :

- Si la fraise est remplacée par un gland, il faut enlever les branches 2 et 4.
- Si la fraise est remplacée par un caillou, il faut enlever les branches 1 et 5.



C'est de l'informatique !

L'œuvre d'Anja peut être représentée par un *graphe*. Un graphe est composé de *nœuds* (les emplacements des objets) et d'*arêtes* (les branches) qui relient deux objets chacune. Les graphes sont des outils polyvalents et sont souvent utilisés lors de la modélisation de problèmes en informatique.

Lorsque deux nœuds sont directement reliés par une arête, ils sont *voisins* l'un de l'autre. Un groupe de nœuds dans lequel chaque nœud est voisin de chaque autre nœuds est appelé une *clique*. Nous avons deux cliques de quatre nœuds dans notre graphe : la moitié gauche et la moitié droite du graphe (la noisette en haut et le point d'interrogation en bas appartiennent aux deux cliques).

La règle d'Anja implique que tous les nœuds d'une clique doivent être occupés par des objets différents. Pour respecter la règle, nous avons donc besoin d'autant d'objets différents qu'il y a de nœuds dans une clique. Nous n'avons cependant plus que trois objets différents une fois que la fraise a été enlevée. Il ne peut donc rester que des cliques comportant au maximum 3 nœuds si la règle doit être respectée. Il faut donc enlever une arête (une branche) afin de détruire les deux cliques à quatre nœuds.

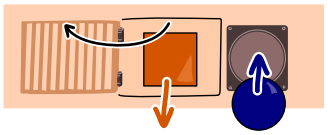
La règle d'Anja correspond à une règle du problème de *coloration de graphe* : on assigne une couleur à chaque nœud d'un graphe, et les nœuds voisins doivent être de couleurs différentes (les couleurs correspondent ici aux différents types d'objets). Le but est souvent d'utiliser le moins de couleurs possibles. Le problème consistant à colorer un graphe avec le moins de couleurs possibles a beaucoup d'applications. Quelques exemple sont la planification d'une compétition, l'élaboration d'un plan de table et même la résolution d'un sudoku.

Mots clés et sites web

- Coloration de graphe : `https://fr.wikipedia.org/wiki/Coloration_de_graphe`
- Coloration des arêtes d'un graphe : `https://fr.wikipedia.org/wiki/Coloration_des_arêtes_d'un_graphe`
- Clique : `https://fr.wikipedia.org/wiki/Clique_(théorie_des_graphes)`

3. Casse-tête d'anniversaire

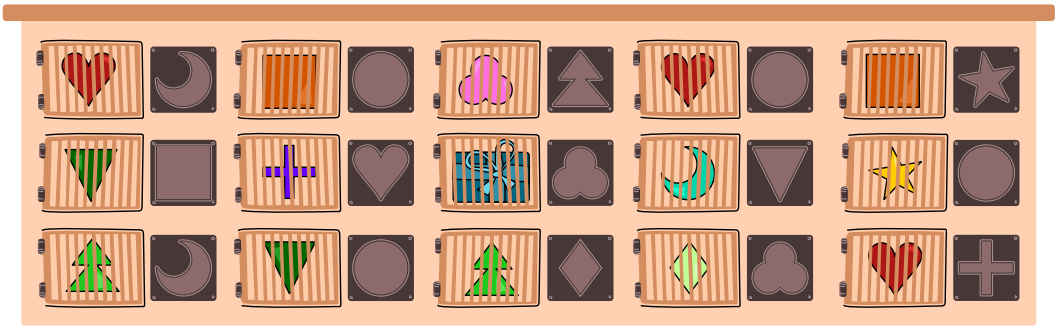
Pour son anniversaire, Bastien reçoit une boîte avec 15 portes. Il y a un autre cadeau derrière la porte du milieu et des plots de différentes formes derrière les autres portes. Chaque porte est associée à un trou qui se trouve à sa droite. Bastien peut ouvrir une porte en mettant un plot de la même forme dans le trou — comme une clé dans une serrure.



Au début, Bastien a ce plot rond : 

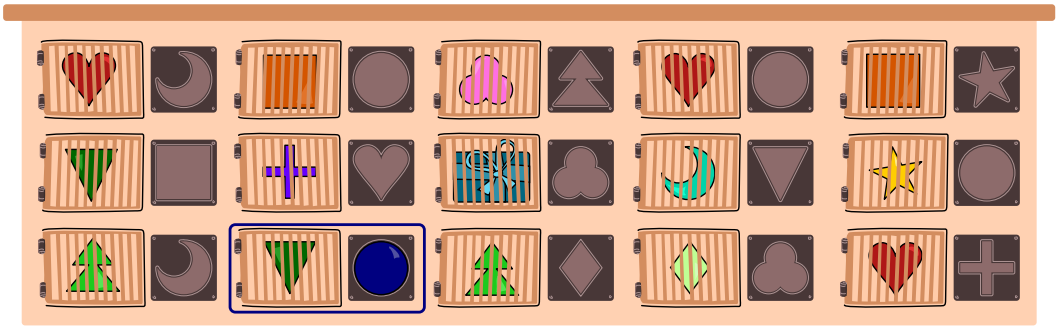
Il veut ouvrir au maximum cinq portes pour obtenir le cadeau.

Quelle porte Bastien doit-il ouvrir en premier ?

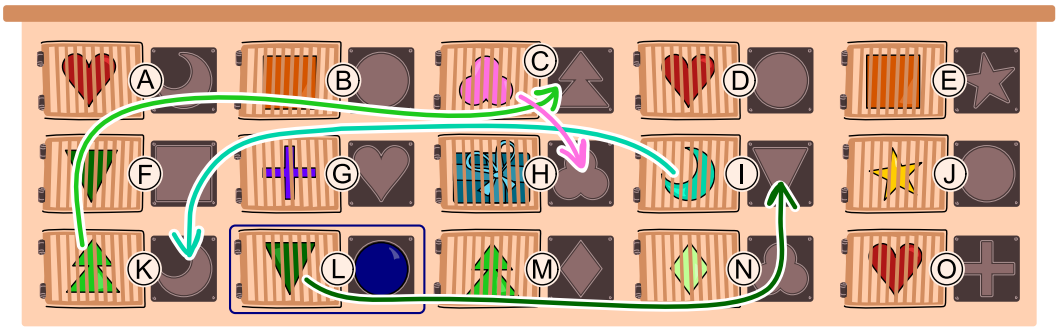


Solution

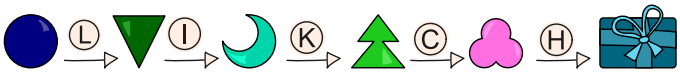
Bastien doit commencer par ouvrir la porte encadrée en bleu :



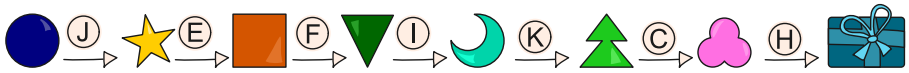
Dans l'illustration ci-dessous, chaque porte correspond a une lettre et les flèches montrent comment Bastien obtient le cadeau en n'ouvrant que cinq portes au maximum.



On peut également représenter l'ordre dans lequel Bastien ouvre les cinq portes de la façon suivante :



Il y a aussi d'autres chemins menant au cadeau, par exemple celui-ci :

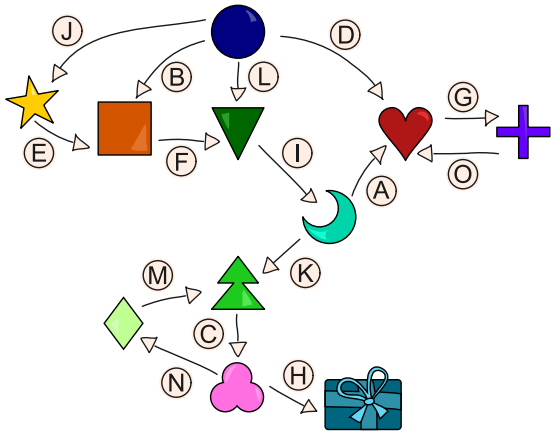


Mais ces chemins sont tous trop longs : il faut ouvrir plus de cinq portes. C'est un gros travail que d'essayer toutes les possibilités.

Dans notre cas, on trouve le chemin le plus court et donc la bonne solution en faisant ce que l'on appelle une *recherche inversée* : on commence par la porte derrière laquelle se trouve le cadeau et on regarde ensuite de quel plot on a besoin.

C'est de l'informatique !

Avec un peu plus de temps et de travail, on peut également représenter la situation de cet exercice sous forme de *graphe* :



Un graphe est composé de *nœuds* (cercles) et *arêtes* (lignes) entre les nœuds. Ici, nous avons un nœud pour chaque forme et pour le cadeau. Les arêtes sont dans notre cas des flèches (aussi appelées arêtes *orientées*) qui correspondent aux portes. Chaque arête mène de la forme qui ouvre une porte à la forme se trouvant derrière la porte.

En informatique, on travaille volontiers avec des graphes. D'un côté, ils permettent souvent de représenter des relations abstraites de manière claire. D'un autre côté, il existe des algorithmes pour répondre à des questions liées aux graphes de manière efficace. Le travail nécessaire à l'élaboration du graphe en vaut vite la peine pour des problèmes compliqués.

Dans cet exercice, nous cherchons un chemin de longueur 5 au maximum entre le plot reçu  et le cadeau  . Un bon algorithme pour cela est le *parcours en largeur*. Il fonctionne aussi bien pour les graphes avec des arêtes orientées comme dans l'exercice que pour les graphes non orientés.

Mots clés et sites web

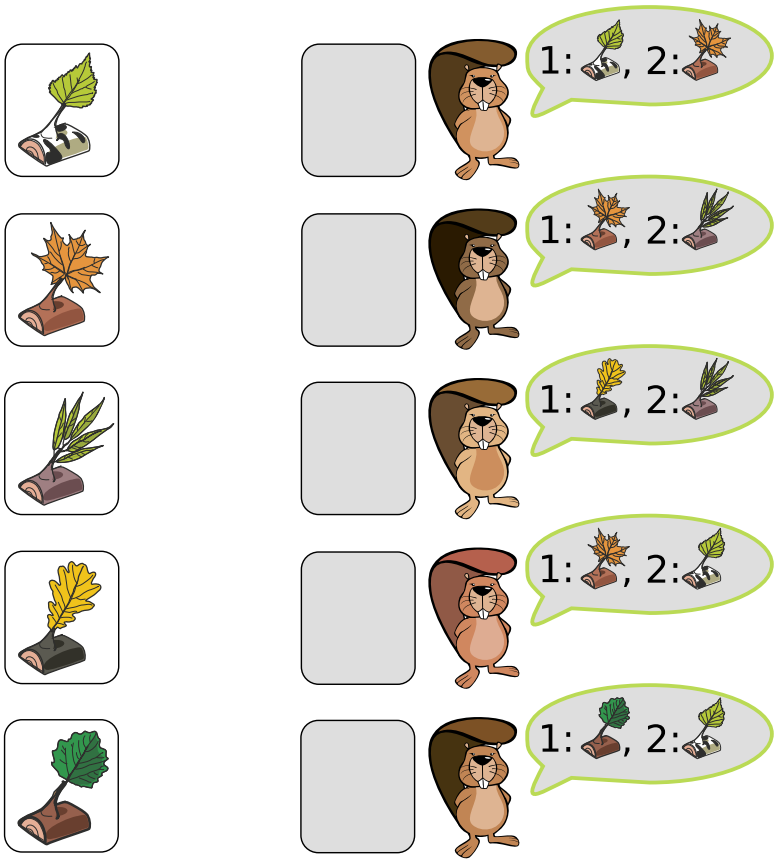
- Graphe orienté : `https://fr.wikipedia.org/wiki/Graphe_orienté`
- Algorithme de parcours en largeur : `https://fr.wikipedia.org/wiki/Algorithme_de_parcours_en_largeur`

4. Lieblingsgeschenk

La famille castor a cinq cadeaux pour ses cinq enfants. Chaque enfant indique d'abord son cadeau favori, puis son second choix. Les cadeaux doivent être bien distribués :

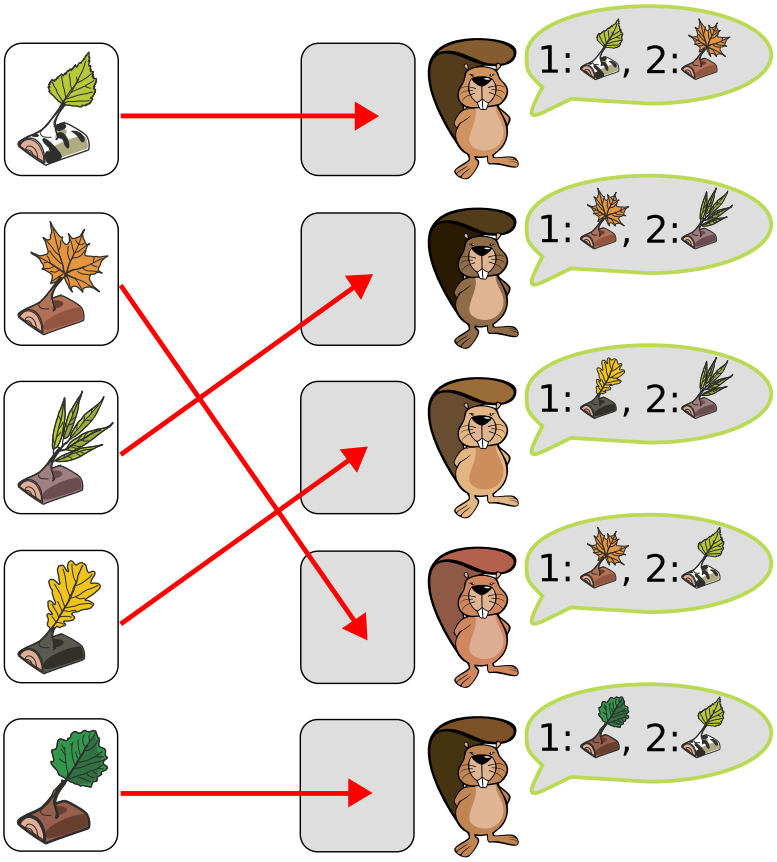
1. Le plus d'enfants possible doivent recevoir leur cadeau favori.
2. Les autres enfants doivent recevoir leur second choix.

Donne les bons cadeaux aux enfants.



Solution

Voici la seule manière de distribuer les cadeaux en respectant les deux conditions.



La distribution du graphique ci-dessus donne leur cadeau favori à quatre castors et son deuxième choix à un castor. Tous les castors ne peuvent pas recevoir leur cadeau favori, car deux castors ont le même. Il n'y a donc pas d'autre distribution donnant leur cadeau favori à plus de castors. Tu peux remarquer que si tu commences la distribution par le haut et donnes son cadeau favori au deuxième castor, le quatrième castor ne pourra avoir aucun de ses deux cadeaux préférés. Il ne suffit donc pas de donner à chaque castor le meilleur cadeau disponible dans cet exercice.

Une méthode de résolution consiste à distribuer d'abord tous les cadeaux qui ne sont le favori que d'un seul castor. Il ne reste ensuite que deux castors avec le même cadeau favori. On regarde ensuite lequel des deux deuxièmes choix est disponible, et on donne à l'autre castor son cadeau favori.

C'est de l'informatique !

Dans cet exercice, nous avons affaire à un *problème d'affectation* univoque : nous voulons affecter les cadeaux de manière à ce que tous les enfants recoivent un cadeau. Les enfants n'ont ici pas qu'un seul souhait, mais une liste de préférence. De tels problèmes d'affectation avec listes de préférence peuvent devenir très compliqués. L'informatique nous aide à résoudre de tels problèmes rapidement.

Une possibilité est de donner une valeur aux affectations : le cadeau favori a la valeur 1 et le deuxième choix la valeur 2. Un *couplage* (*matching* en anglais) est optimal s'il n'existe pas d'autre couplage avec plus de premiers choix distribués. Un tel couplage est appelé *couplage parfait de poids minimum*. Il existe beaucoup de problèmes d'affectation. L'un deux est appelé *problème des marriages stables* (*Stable Marriage Problem* en anglais). Cela t'intéresse ? Alors tu devrais étudier l'informatique !

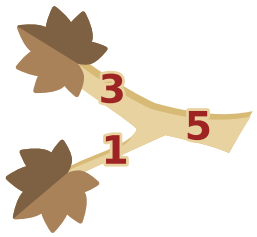
Mots clés et sites web

- Problème d'affectation : `https://fr.wikipedia.org/wiki/Problème_d'affectation`
- Couplage : `https://fr.wikipedia.org/wiki/Couplage_(théorie_des_graphes)`

5. Sauvetage d'arbre

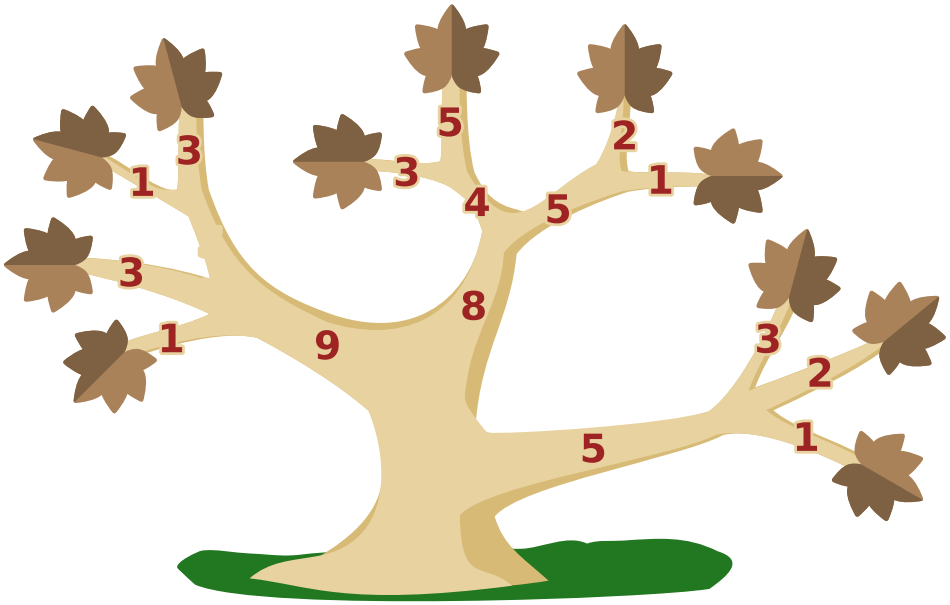
Un des arbres du jardin de Bruno est malade : toutes ses feuilles sont sèches. Bruno veut le sauver. Pour cela, il doit scier certaines branches de façon à enlever toutes les feuilles. De nouvelles branches avec de nouvelles feuilles peuvent ensuite pousser.

Bruno aimerait terminer le plus vite possible. L'image montre un exemple :



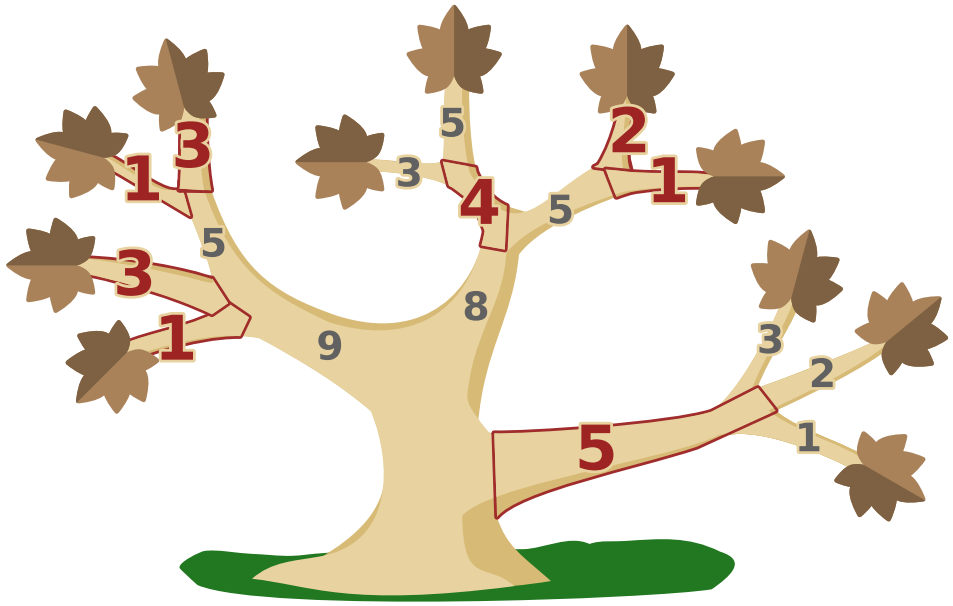
Pour enlever les deux feuilles, Bruno peut soit scier les deux branches portant les feuilles, soit scier la branche de laquelle partent les deux autres branches. Les chiffres sur chaque branche indiquent combien de temps est nécessaire pour la scier. Bruno va donc scier les deux branches avec des feuilles, étant donnée que $3 + 1 < 5$. Tu peux voir l'arbre complet ci-dessous.

Quelles branches Bruno va-t-il scier pour terminer le plus vite possible ?



Solution

La bonne solution est la suivante : Bruno scie les branches encadrées en rouge pour terminer le plus vite possible.



Mais pourquoi est-ce la bonne solution ? On peut commencer par calculer de combien de temps Bruno a besoin pour scier uniquement les branches portant des feuilles — il aurait ainsi terminé :

$$1 + 3 + 1 + 3 + 3 + 5 + 2 + 1 + 3 + 2 + 1 = 25$$

Maintenant, on va en direction du tronc et vérifie à chaque étape si ce serait plus rapide de scier la branche à laquelle les branches précédentes sont directement ou indirectement reliées. On dérive le calcul suivant de la première de ces étapes (la fonction « min » calcule le minimum de ses arguments) :

$$\begin{aligned} & 1 + 3 + \min(5, 1 + 3) + \min(4, 3 + 5) + \min(5, 2 + 1) + \min(5, 3 + 2 + 1) \\ = \; & 1 + 3 + (1 + 3) + 4 + (2 + 1) + 5 \\ = \; & 20 \end{aligned}$$

On ne calcule pas tout de suite le temps total afin de mieux voir quelles branches doivent être sciées. L'étape suivant nous mène déjà au tronc :

$$\begin{aligned} & \min(9, 1 + 3 + 1 + 3) + \min(8, 4 + 2 + 1) + 5 \\ = \; & (1 + 3 + 1 + 3) + (4 + 2 + 1) + 5 \\ = \; & 20 \end{aligned}$$

Bruno ne peut pas terminer plus rapidement.

C'est de l'informatique !

Imaginons que les branches sciées ne tombent pas tout de suite de l'arbre — comme lorsque l'on résout cet exercice à l'écran. On pourrait alors dire que l'arbre est séparé en deux parties lorsque l'on scie : une partie comporte tous les morceaux sciés, et surtout toutes les feuilles, et l'autre partie comporte le tronc et les branches qui n'ont pas été sciées. Cette séparation ou *coupe* de l'arbre est minimale par rapport au temps nécessaire à Bruno pour enlever toutes les feuilles.

L'informatique traite aussi d'arbres, qui y sont utilisés pour réprésenter des objets reliés entre eux de manière spécifique. Les objets sont appelés *nœuds* et les liens entre eux *arêtes*. Il n'existe toujours qu'un seul chemin entre deux nœuds le long des arêtes — comme il n'existe qu'un seul chemin entre une feuille ou un branchement jusqu'au tronc le long des branches. Si l'on renonce à cette contrainte, on parle plus généralement d'un *graphe*.

Pour les graphes généraux, ce n'est pas si facile de calculer la *coupe minimale*, c'est à dire la séparation en deux ou plusieurs parties à un coût minimal, que pour un arbre comme nous l'avons fait ici, mais pas trop compliqué non plus. Heureusement, car il existe des applications intéressantes. Les coupes minimales peuvent être utilisées pour diviser des images en parties similaires. Dans les *réseaux de flot*, un type de graphe spécial avec lequel on peut, entre autres, modéliser le flot des données dans des réseaux, le coût d'une coupe minimale correspond au flot maximal dans le réseau complet.

Mots clés et sites web

- Arbre : `https://fr.wikipedia.org/wiki/Arbre_(théorie_des_graphes)`
- Coupe : `https://fr.wikipedia.org/wiki/Coupe_(théorie_des_graphes)`
- Réseau de flot : `https://fr.wikipedia.org/wiki/Réseau_de_flot`
- Théorème flot-max/coupe-min : `https://fr.wikipedia.org/wiki/Théorème_flot-max/coupe-min`

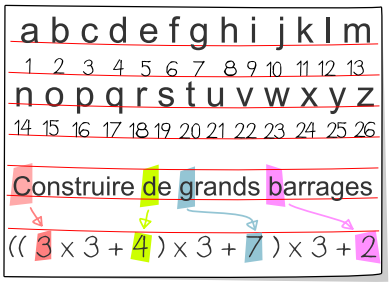
6. Bibliothèque

Susi est à la bibliothèque des castors avec Tim. Ils veulent emprunter un livre appelé « Construire de grands barrages ».

Tim va vers l'étagère 1, regarde dans la rangée 4 et sort le livre du casier 0. Susi est impressionnée. Tim explique à Susi comment on détermine l'emplacement d'un livre :

On prend la première lettre de chaque mot du titre du livre et détermine sa position dans l'alphabet. On additionne ensuite ces positions après avoir multiplié par 3 la valeur obtenue à l'addition précédente.

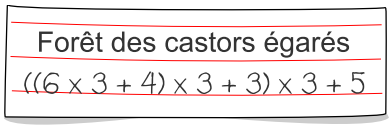
Le livre désiré donne 140. L'emplacement du livre est ainsi tout de suite clair.



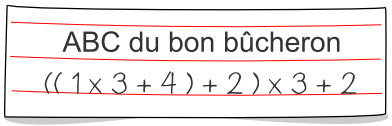
Susi écrit maintenant les calculs correspondant à ses livres préférés, mais elle a fait une erreur pour l'un d'entre eux.

Lequel ?

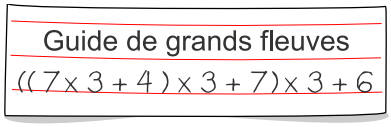
A)



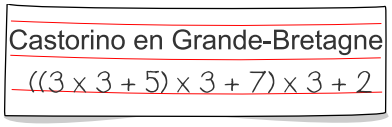
B)



C)

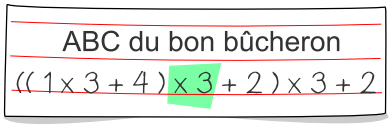


D)



Solution

Susi a presque tout fait juste : elle a toujours additionné les bonnes positions et a multiplié les résultats intermédiaires par 3 — avec une exception : elle a oublié une multiplication dans la réponse B.



C'est de l'informatique !

Le système d'expressions correspondant à l'emplacement des livres permet aux visiteurs de la bibliothèque de déterminer l'endroit exact où les livres sont rangés. Comme ça, personne ne doit chercher longtemps. La bibliothèque et les visiteurs doivent cependant faire attention à une chose : différents livres peuvent avoir la même expression et donc le même résultat. Par exemple, les livres « Guide des grands fleuves » et « Guide des grandes familles » sont dans le même casier. Les casiers doivent donc être assez grands ou pouvoir être agrandis selon les besoins.

C'est aussi une bonne idée que l'endroit auxquel les données sont enregistrées dans un ordinateur puisse être calculé directement à partir des données elles-même. Pour cela, des *fonctions de hachage* ont été développées en en informatique : des fonctions mathématiques qui calculent une valeur à partir du contenu des données ou d'une partie des données, valeur qui indique directement l'emplacement mémoire — comme dans cet exercice du castor. De bonnes fonctions de hachage minimisent le nombre de fois où la même valeur est calculée. Si un tel conflit a lieu, l'informatique dispose de bonnes méthodes pour le gérer.

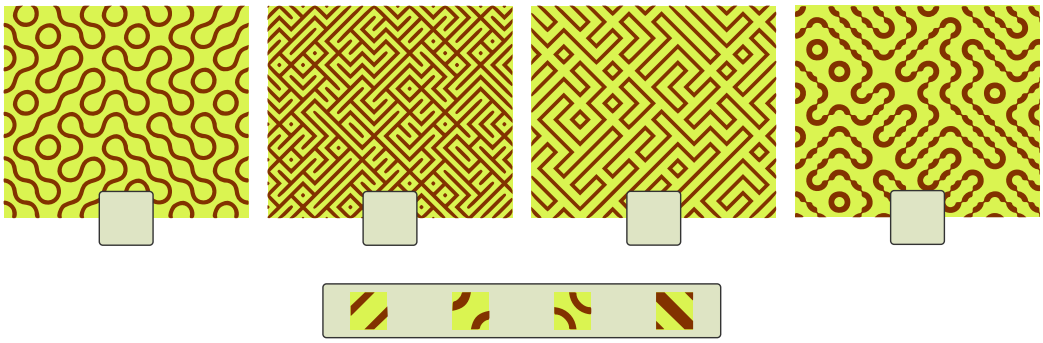
Mots clés et sites web

- Fonction de hachage : `https://fr.wikipedia.org/wiki/Fonction_de_hachage`
- Table de hachage : `https://fr.wikipedia.org/wiki/Table_de_hachage`

7. Pavage de Truchet

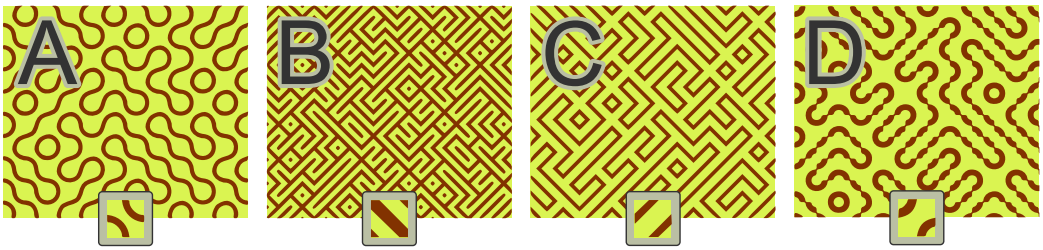
Les motifs suivants ont été créés en n'utilisant qu'un seul type de pavé. Les images des pavés individuels sont agrandies.

Assigne chaque pavé au motif correspondant.

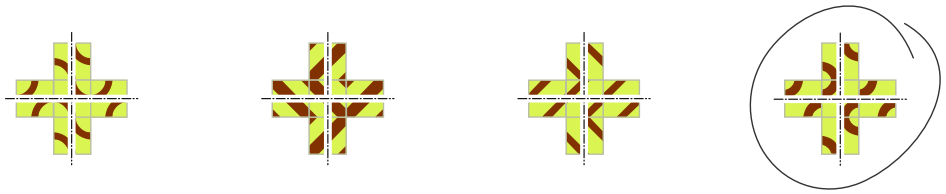


Solution

Voici la bonne attribution :



En mettant cinq mêmes pavés côte à côte et en comparant les différents pavés, on voit de claires différences :



Le pavé  est le seul pavé dont les quatre côtés ne vont pas exatement ensemble. C'est le seul moyen pour obtenir des lignes de largueurs différentes comme dans le motif D. Le pavé  est le seul avec lequel on peut créer des points carrés comme sur le motif B, en mettant bout à bout quatre coins avec un triangle. De plus, il a la plus grande proportion de brun par rapport au jaune, comme le motif B. Il ne reste donc que le pavé  pour former le motif A aux formes arrondies, alors que les lignes droites du motif C ne peuvent être créées qu'avec le pavé  .

C'est de l'informatique !

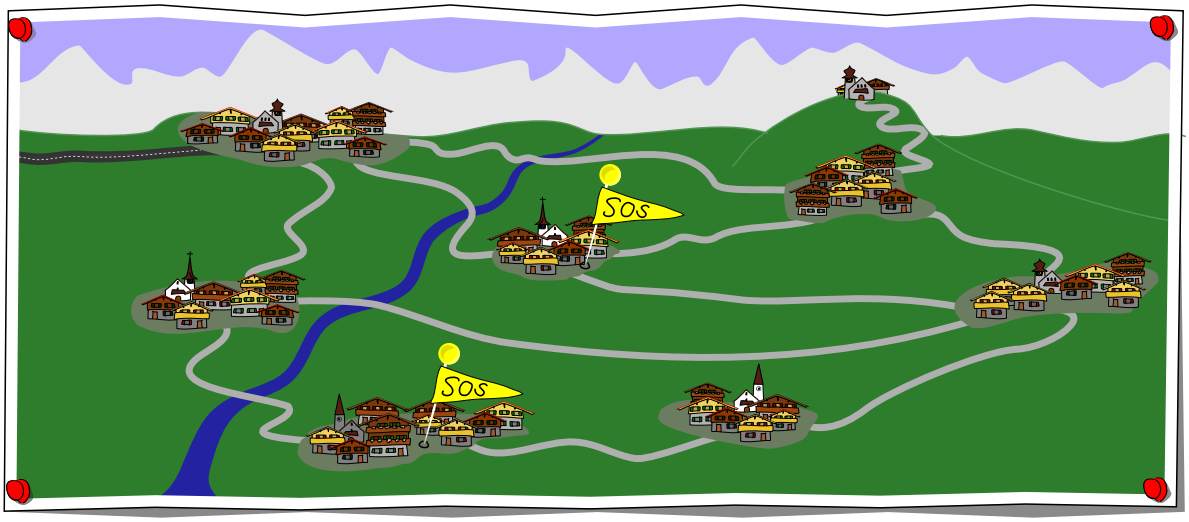
Ces pavés sont nommé d'après Sébastien Truchet (* 1657 ; † 1729), qui en a développé différentes variantes. Les pavés ayant quatre côtés pareils forment un sous-ensemble des pavés de Truchet (mais les pavés de Truchet ne doivent pas forcément avoir quatre côtés pareils, comme dans trois des motifs de cet exercice). Le fait que des motifs complets peuvent être générés à partir d'éléments très simple est une propriété intéressante que l'on rencontre souvent en informatique. Les pavés de Truchet sont étudiés en mathématiques et en informatique, et sont utilisé dans les jeux vidéos pour créér des labyrinthes et des décors.

Mots clés et sites web

- Pavés de Truchet : `https://en.wikipedia.org/wiki/Truchet_tiles`
- Sébastien Truchet : `https://fr.wikipedia.org/wiki/Sébastien_Truchet`

8. Villages isolés

Plusieurs villages de montagne sont approvisionnés par la grande ville grâce au réseau de routes suivant :

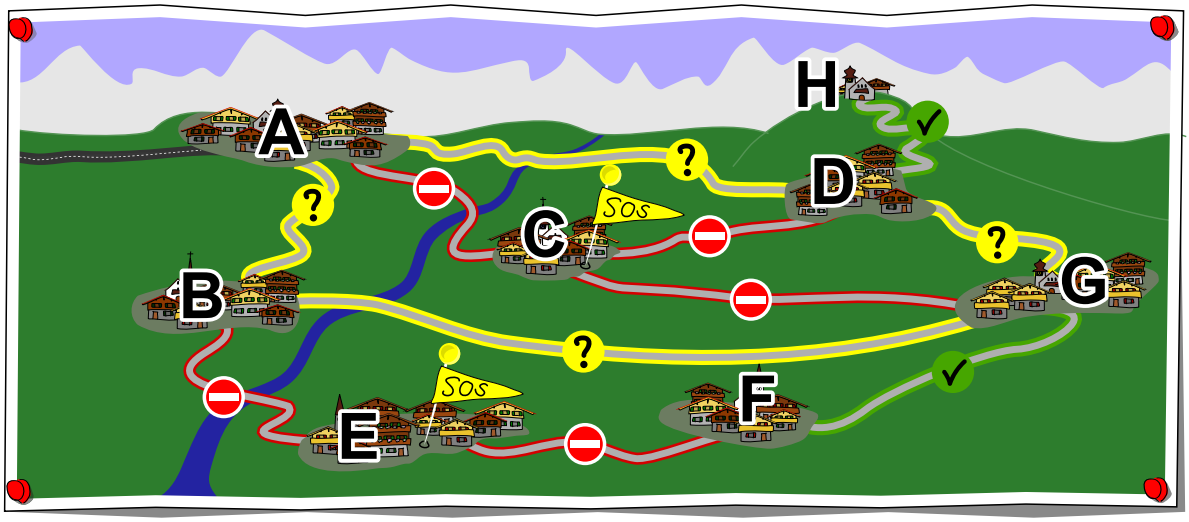


Après une tempête, plusieurs de ces villages ne sont plus du tout accessibles et le signalent avec un drapeau « SOS ». On peut en déduire que certaines des routes sont bloquées.

Indique pour chaque route du réseau entre les villages si elle est (1) bloquée ⛔, (2) ouverte ✅ ou (3) si on ne peut pas le savoir sans informations supplémentaires ❓.

Solution

La carte suivante montre ce que l'on sait sur les connexions dans le réseau routier :



Nous commençons par déterminer quelles routes sont bloquées. Les deux routes menant au village E sont bloquées, car le village E serait accessible si ce n'était pas le cas. De même, les trois routes menant au village C sont bloquées, car il serait accessible sinon.

Ensuite, nous cherchons les routes qui doivent être ouvertes. La route entre les villages G et F doit être ouverte, car le village F ne serait pas accessible autrement, la route entre les villages F et E étant bloquée. La route entre l'église H et le village D doit aussi être ouverte, vu que H est accessible et qu'on ne peut y accéder qu'en passant par D.

Il ne reste que les routes qui sont peut-être accessibles. Comme les village B, G et D sont reliés plusieurs fois au village A, on ne peut pas déterminer quelles routes parmis celles qui restent sont ouvertes. On pourrait par exemple accéder au village B par le village A, mais aussi par le village G. C'est la même chose pour le village D. On peut accéder on village G par le village B ou par le village D. N'importe laquelle des routes dans le circuit A - B - G - D - A pourrait donc être fermée et les quatre villages resteraient accessibles.

C'est de l'informatique !

Comme dans les réseaux routiers, il peut aussi il y avoir des connexions fautives, surchargées ou défectueuses dans les réseaux informatiques. Pour éviter les pannes, on planifie souvent des mesures de sécurité, comme par exemple plusieurs connexion menant au même endroit. On appelle cela la *redondance*.

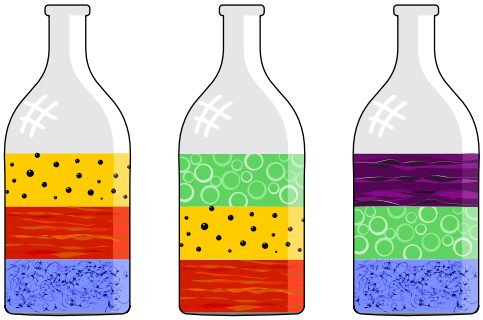
La correction de dysfonctionnements d'un système est une tâche que les informaticiens doivent souvent effectuer ; pas seulement dans les réseaux informatiques, mais aussi lors de développement de programmes. Pour corriger un problème, il faut identifier sa source exacte ; c'est un processus qui se fait en général étape par étape. Certains programmeurs disent que l'on ne peut jamais trouver toutes les erreurs et tous les bugs d'un programme.

Mots clés et sites web

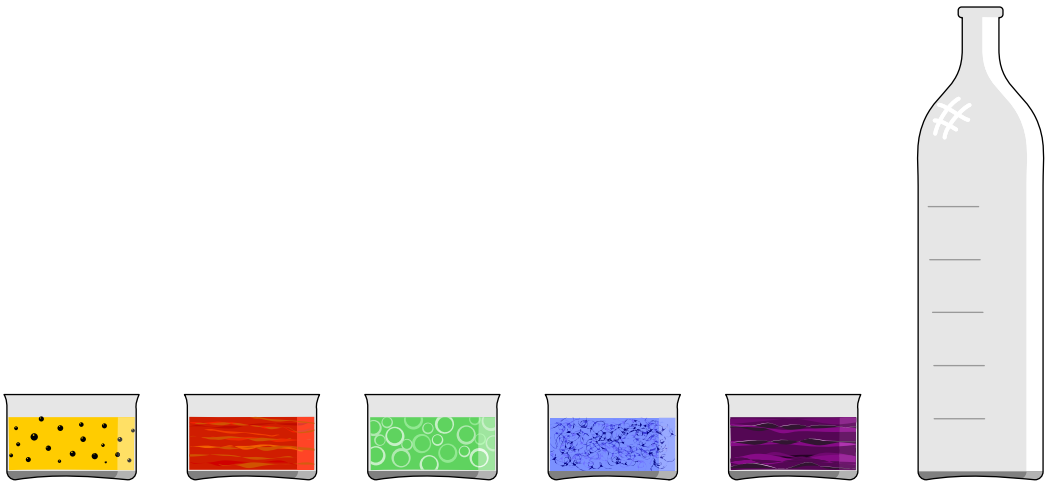
- Redondance : `https://fr.wikipedia.org/wiki/Redondance_(ingénierie)`
- Bug : `https://fr.wikipedia.org/wiki/Bug_(informatique)`

9. Couches de liquides

Marc a des des bouteilles qui contiennent chacune trois liquides formant des couches superposées. Il sait que les liquides à densité plus faible se mettent toujours au dessus des liquides à densité plus forte. Il aimerait maintenant voir à quoi une grande bouteille dans laquelle on met tous les liquides colorés ressemble.

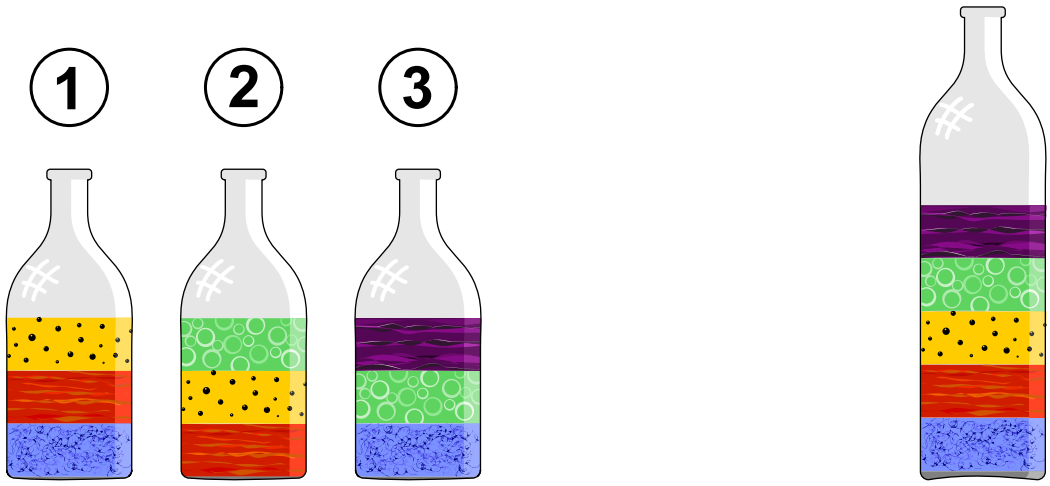


Arrange les cinq couches de liquides colorés dans la bouteille dans leur ordre final.



Solution

L'image montre le bon arrangement des couches de liquide dans la grande bouteille.



Tu trouves dans quel ordre se trouvent les couches de liquide de la façon suivante : étape par étape, tu enlèves dans ta tête les liquides qui ne sont pas au dessus d'autres liquides dans aucune des trois bouteilles données, et les verses dans la grande bouteille.

Au départ, le liquide bleu est tout au fond des bouteilles 1 et 3 et jamais au dessus d'une autre couche de liquide. Le liquide rouge est tout au fond de la bouteille 2, mais au-dessus du liquide bleu dans la bouteille 1 et doit donc avoir une densité plus faible que le liquide bleu. On enlève donc le liquide bleu des bouteilles et le verse dans la grande bouteille.

La liquide rouge est maintenant le seul qui n'est pas au dessus d'un autre liquide. On l'enlève des bouteilles 1 et 2 et le met dans la grande bouteille. Ensuite viennent le liquide jaune, puis le vert et finalement le violet, qui a la plus faible densité et au dessus duquel ne se trouve aucun autre liquide.

C'est de l'informatique !

Dans cet exercice, tu as évalué l'arrangement des liquides dans les trois bouteilles et trié les liquides par densité.

Une substance a beaucoup de propriétés mesurables, par exemple la température d'ébulition, la température de fusion, la conductivité et la densité. Dans le cas présent, la densité a été utilisée comme critère pour trier des substances.

Le tri de données joue un rôle important dans beaucoup de programmes informatiques. La méthode utilisée dans cet exercice pour déterminer l'ordre des couches de liquide s'appelle *tri topologique*. Elle est utilisée pour trier des objets lorsque l'on connait la *relation d'ordre* entre certains des objets (si l'on sait déjà que certains objets en précèdent ou suivent d'autres).

Mots clés et sites web

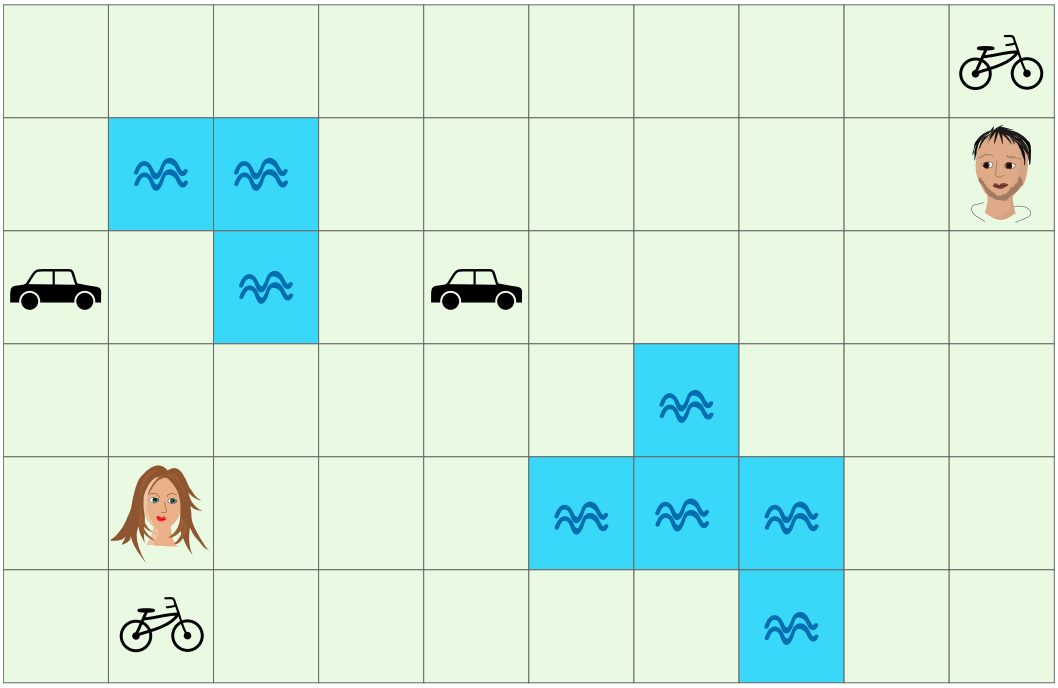
- Relation d'ordre : `https://fr.wikipedia.org/wiki/Relation_d'ordre`
- Tri topologique : `https://fr.wikipedia.org/wiki/Tri_topologique`

10. Un, deux, trois, partez, feu !

Deux amis veulent se voir le plus vite possible. Ils peuvent passer d'une case à la case voisine de droite, de gauche, du haut ou du bas.

La traversée d'une case à une autre prend 1 minute à pied. S'ils arrivent sur une case avec un véhicule, il peuvent l'utiliser. Ils peuvent alors avancer de 2 cases en 1 minute à vélo et de 5 cases en 1 minute en voiture.

Les changements de direction sont toujours possible. Ils ne peuvent pas traverser les étendues d'eau.

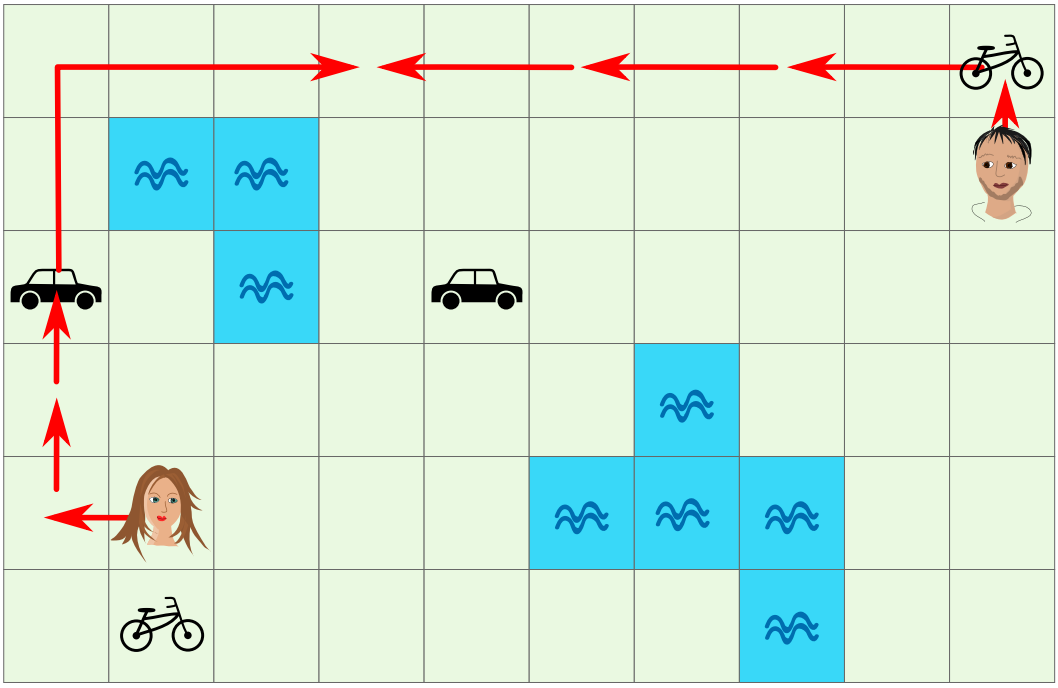


De combien de minutes au minimum les deux amis ont-ils besoin pour se retrouver sur la même case ?

A) 1 minute
B) 2 minutes
C) 3 minutes
D) 4 minutes
E) 5 minutes
F) 6 minutes

Solution

La bonne réponse est 4. L'image montre un chemin permettant aux deux amis de se retrouver sur la même case en 4 minutes.



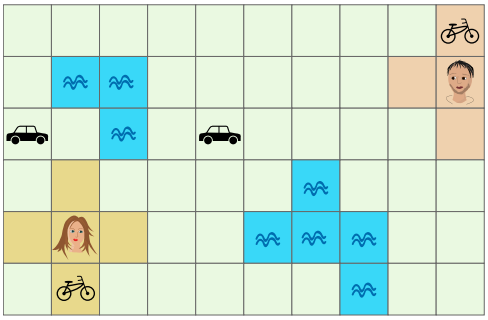
Maintenant, il faut encore prouver qu'ils ne peuvent pas se retrouver en 3 minutes. Les deux amis sont à 11 cases de distance l'un de l'autre. En trois minutes, s'ils se déplacent à pied, il ne peuvent se rapprocher que de 6 cases en tout. Si l'un des deux a atteint un vélo et que l'autre va à pied, ils peuvent se rapprocher de 9 cases, ce qui ne suffit pas non plus. Même s'ils se déplacent les deux à vélo, ils n'y arrivent pas : ils pourraient se rapprocher de 12 cases en 3 minutes, mais les deux vélos sont à 13 cases l'un de l'autre.

Il ne leur reste donc que la possibilité d'utiliser une voiture. En 3 minutes, la fille peut atteindre une voiture, mais n'aurait plus le temps de l'utiliser. Le garçon ne peut pas atteindre de voiture en 3 minutes.

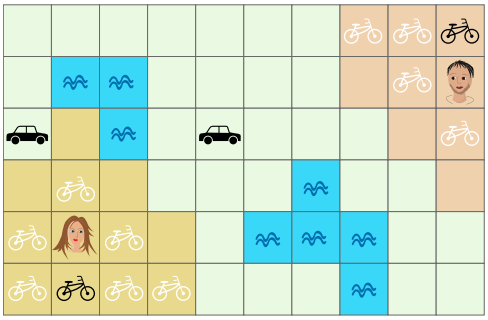
C'est de l'informatique !

Comment as-tu résolu l'exercice ? As-tu trouvé un chemin rapide par hasard et espéré qu'il n'en existe pas de plus rapide ? Ou as-tu essayé beaucoup de chemins différents en te rappelant du plus rapide ?

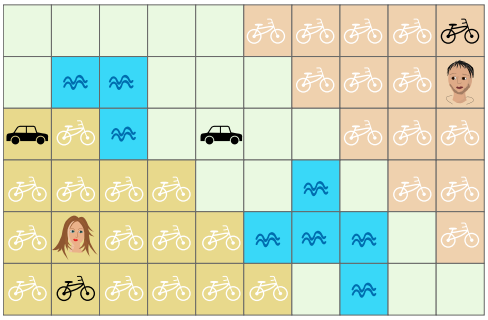
Les programmes informatiques qui ont été développés pour résoudre ce genre de problème travaillent souvent avec une méthode appelée *parcours en largeur*. Dans cet exercice, le parcours en largeur se passe comme cela :



1. Sélectionne toutes les cases que chacun des deux amis peut atteindre en 1 minute.

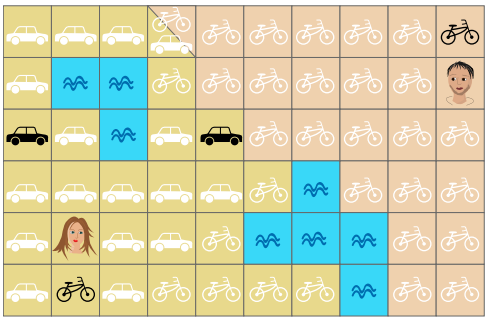


2. Sélectionne toutes les cases qui peuvent être atteinte en (maximum) 1 minute depuis les cases sélectionnées à l'étape 1. Note quel moyen de transport a été utilisé.



3. Sélectionne toutes les cases qui peuvent être atteinte en (maximum) 1 minute depuis les cases sélectionnées à l'étape 2.

Comme les deux régions que tu as sélectionnées ne se chevauchent pas encore, les deux amis ne peuvent pas encore se voir après 3 minutes.



4. Sélectionne toutes les cases qui peuvent être atteinte en (maximum) 1 minute depuis les cases sélectionnées à l'étape 3.

Maintenant, les deux régions se chevauchent sur une case. Elle peut être atteinte en 4 minutes par la fille en voiture et par le garçon en vélo.

Les systèmes de navigation trouvent le chemin le plus rapide entre deux points. Pour cela, ils font attention à ce que le chemin passe par des routes adaptées, et non pas à travers champs ou par des rivières. Cet exercice ressemble à un problème de navigation, mais ici, ce n'est pas une personne qui doit arriver à un but, mais deux personnes qui doivent arriver à un but commun inconnu au départ.

Comme un ordinateur procède de manière systématique lors d'un parcours en largeur, il peut aussi trouver des solutions qui ne sont pas évidentes. Parfois, un détour par une route avec moins de feux

peut être plus rapide que le chemin le plus court entre le départ et l'arrivée. Un voyage en train avec changement peut être plus rapide qu'un voyage direct en bus.

Il existe en informatique plusieurs méthodes pour résoudre des problèmes de ce type. En plus de la méthode de parcours en largeur discutée ici, il existe une méthode appelée *Branch and Bound* (*séparation et évaluation* en français). Le parcours en largeur tient compte de chaque solution partielle obtenue après un certain nombre d'étapes. Avec *Branch and Bound*, les solutions partielles ne menant pas à la solution optimale ne sont plus considérées dans les étapes suivantes.

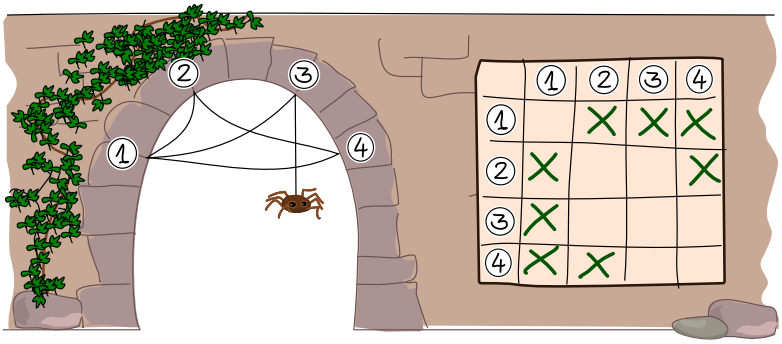
Lorsqu'un problème devient trop complexe, cela peut durer trop longtemps d'essayer toutes les possibilités pour trouver la meilleure solution, même avec l'ordinateur le plus rapide du monde. En pratique, il suffit à un système de navigation de trouver un très bon chemin, même si ce n'est pas le meilleur chemin possible — si tu peux atteindre ton but en 78 minutes, ça ne te fait probablement rien qu'on puisse théoriquement aussi l'atteindre en 77 minutes.

Mots clés et sites web

- Parcours en largeur :
  `https://fr.wikipedia.org/wiki/Algorithme_de_parcours_en_largeur`
- Séparation et évaluation : `https://fr.wikipedia.org/wiki/Séparation_et_évaluation`

11. Toiles d'araignée

Thekla l'araignée aimerait construire autant de toiles différentes que possible. Pour cela, elle a mis au point une méthode lui permettant de décrire la structure exacte de ses toiles.

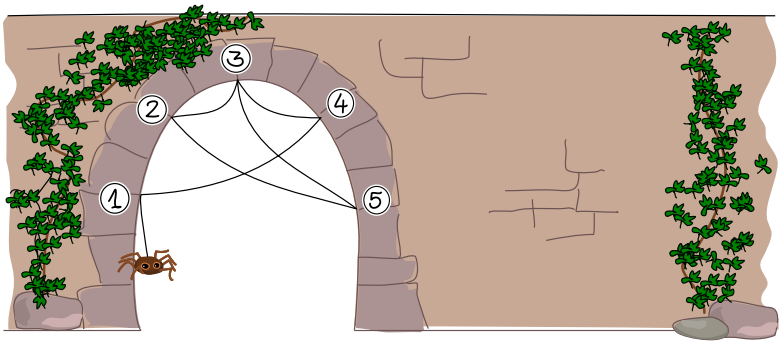


Elle procède de la façon suivante : elle numérote les points d'ancrage de la toile de 1 à N et utilise les case d'une grille comme cela :

- S'il y a un fil qui relie le point d'ancrage x au point d'ancrage y, elle met un « X » dans la case située dans la colonne x et dans la ligne y.

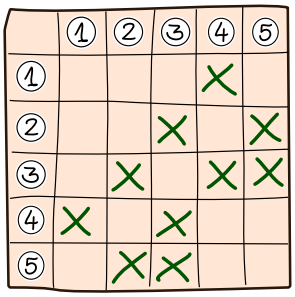
Un fil qui relie le point d'ancrage x au point d'ancrage y relie également le point d'ancrage y au point d'ancrage x.

Thekla construit à présent cette toile :

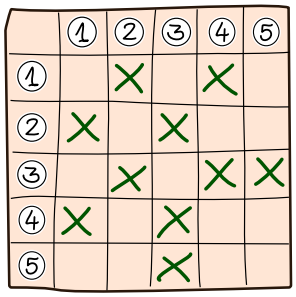


Comment Thekla décrit-elle la structure de cette toile ?

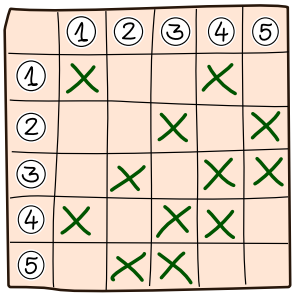
A)



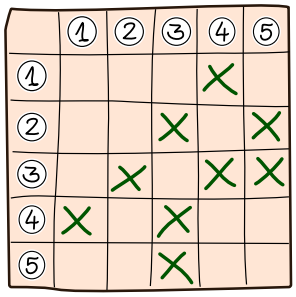
B)



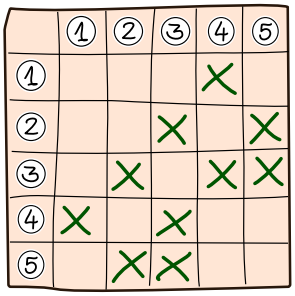
C)



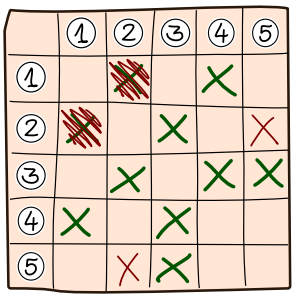
D)



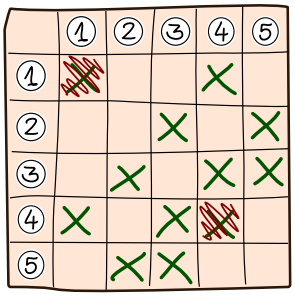
Solution



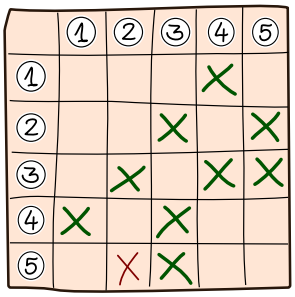
La réponse A est correcte, car toutes les cases sont remplies correctement d'après la règle.



La réponse B comporte une connection supplémentaire erroné (entre le point d'ancrage 1 et le point d'ancrage 2 dans les deux directions) et une connection a été oubliée (entre le point d'ancrage 2 et le point d'ancrage 5 dans les deux directions).



Concernant la réponse C : d'après la règle, aucun « X » ne peut être présent dans la diagonale allant du haut à gauche au bas à droite. ils représenteraient en effet des connections entre d'un point d'ancrage et lui-même. Ceux-ci peuvent exister dans certains réseaux, mais pas dans notre toile d'araignée. La réponse C comporte cependant deux telles connections (au points d'ancrage 1 et 4).



Le réponse D n'est pas symétrique par rapport à la diagonale allant du haut à gauche au bas à droite, alors que toutes les descriptions de toiles devraient l'être. Dans cette réponse, le point d'ancrage 2 et le point d'ancrage 5 sont connectés, mais la connection correspondante dans l'autre direction manque.

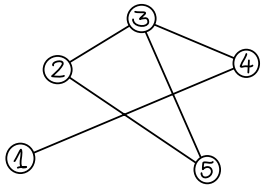
C'est de l'informatique !

La toile d'araignée peut être considérée comme un *graphe* ; un concept qui apparait fréquemment en informatique.

Un graphe est composé de *nœuds* (les points d'ancrage de la toile) et d'*arêtes* (les fils entre les points d'ancrage). Les graphes peuvent aussi être utilisés pour représenter des objets et les relations entre ces objets. Par exemple, un graphe pourrait représenter la manière dont des personnes sont connectées sur un réseau social ou des vols d'avion entre différents pays.

Cet exercice montre comment la structure d'une toile d'araignée peut être enregistrée dans une grille. Certaines propriétés de la toiles sont perdues dans cette représentation, comme par exemple l'apparence exacte de la toile. Souvent, on ne s'intéresse pas aux propriétés géométriques exactes

d'une toile, mais seulement à sa structure. Les informations essentielles sont conservées : combien y a-t-il de nœuds ? Quelles paires de nœuds sont-elles reliées par une arête ?



	1	2	3	4	5
1	0	0	0	1	0
2	0	0	1	0	1
3	0	1	0	1	1
4	1	0	1	0	0
5	0	1	1	0	0

La représentation utilisée ici n'est qu'une possibilité parmi beaucoup de représenter la structure d'un réseau. Cette méthode n'est pas très économe, car toutes les connections sont enregistrées dans les deux sens, ce qui ne serait pas nécessaire ; les cases de la diagonale ne seraient pas nécessaire non plus. Cette méthode a par contre l'avantage de mettre en évidence les erreurs de représentation. Les réponses C et D, par exemple, peuvent être éliminées sans même considérer la toile d'araignée.

La méthode de représentation utilisée dans cet exercice s'appelle une *matrice d'adjacence*.

Mots clés et sites web

- Matrice d'adjacence : `https://fr.wikipedia.org/wiki/Matrice_d'adjacence`

12. Pile de fruits

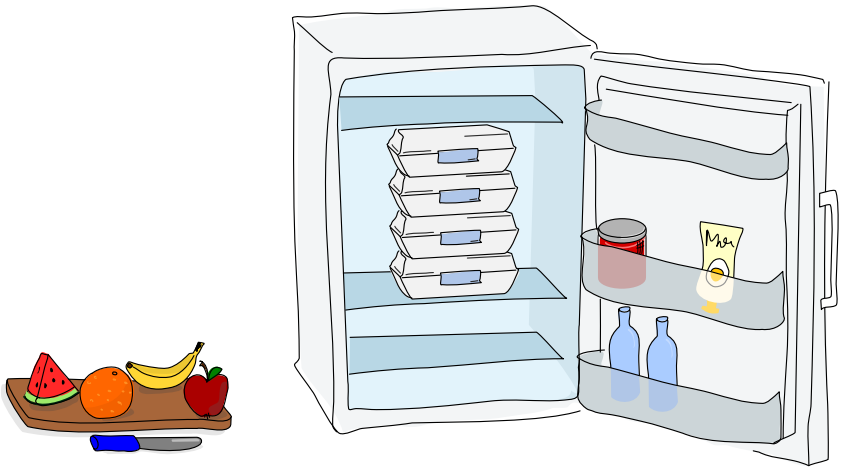
Papa, Maman, Dorie et Ron Castor préparent quatre boîtes avec un fruit différent dans chacune : pomme, banane, orange et pastèque. Les boîtes sont empilées dans le réfrigérateur. Le matin, les castors sont encore très fatigués et prennent simplement la boîte du haut de la pile en quittant le gîte sans la regarder plus en détail.

On ne sait pas exactement dans quel ordre les castors quittent le gîte, mais Maman part dans tous les cas avant Dorie et Papa sort toujours en dernier.

Les membres de la famille aiment des fruits différents. Le tableau suivant indique ce que chaque membre de la famille aime :

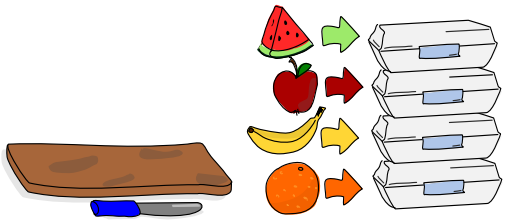
				
Papa	—	—	✓	—
Maman	✓	—	✓	✓
Dorie	✓	✓	✓	—
Ron	✓	✓	—	✓

Mets les fruits dans les boîtes de manière à ce que chaque castor prenne une boîte contenant un fruit qu'il aime.



Solution

Il n'y a qu'une possibilité de répartir les fruits de manière à garantir que chacun ait quelque chose qu'il aime :



Papa n'aime que les oranges et part en dernier. L'orange va donc dans la boîte la plus basse. Ron part en premier, deuxième ou troisième. Comme Maman part avant Dorie, on connait l'ordre de départ exact des castor si l'on sait quand Ron quitte le gîte. Les trois ordres de départ suivants sont possibles :

1.	Maman	Maman	Ron
2.	Dorie	Ron	Maman
3.	Ron	Dorie	Dorie
4.	Papa	Papa	Papa

Maman, Dorie et Ron peuvent chacun partir en deuxième. Il faut donc mettre un fruit que les trois aiment dans la deuxième boîte, et ce n'est le cas que de la pomme. Il ne reste ainsi que la banane et la pastèque pour la boîte du haut. Comme maman n'aime pas les bananes, il faut y mettre la pastèque. La banane va ainsi dans la troisième boîte.

C'est de l'informatique !

Le bon ordre d'une séquence est important dans beaucoup de domaines de l'informatique : beaucoup de calculs doivent utiliser des résultats intermédiaires pour arriver au résultat final. Si différentes étapes de calcul sont effectuées sur différents ordinateurs sans planning consciencieux, des *interblocages* (ou *àtreintes fatales*, *deadlock* en anglais) peuvent avoir lieu. Ce sont des situations dans lesquelles plusieurs ordinateurs s'attendent mutuellement et qui empêchent le programme de se terminer. Un mauvais ordre ne mène cependant souvent qu'à des erreurs (comme il peut amener la mauvaise humeurs chez les castors qui découvrent leur fruit). Par exemple, si l'on doit calculer à l'aide de la formule $Z = (A + B) \times (A - B)$, on peut partager cela en plusieurs étapes d'un programme comme cela :

Entrée A
Entrée B
Calcule $X = A + B$
Calcule $Y = A - B$
Calcule $Z = X \times Y$

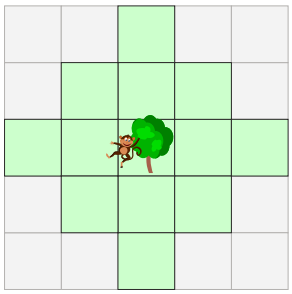
Si l'on essaie maintenant d'effectuer par exemple l'étape $Z = X \times Y$ avant d'avoir calculé X, cela cause une erreur et l'interruption du programme. Alternativement, une valeur par défaut va être

utilisée pour X, ce qui mène à un faux résultat dans la plupart des cas. L'ordre dans lequel les commandes sont effectuées est donc très important en informatique.

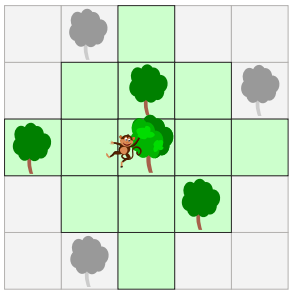
Mots clés et sites web

- Interblocage : `https://fr.wikipedia.org/wiki/Interblocage`

13. Petit singe



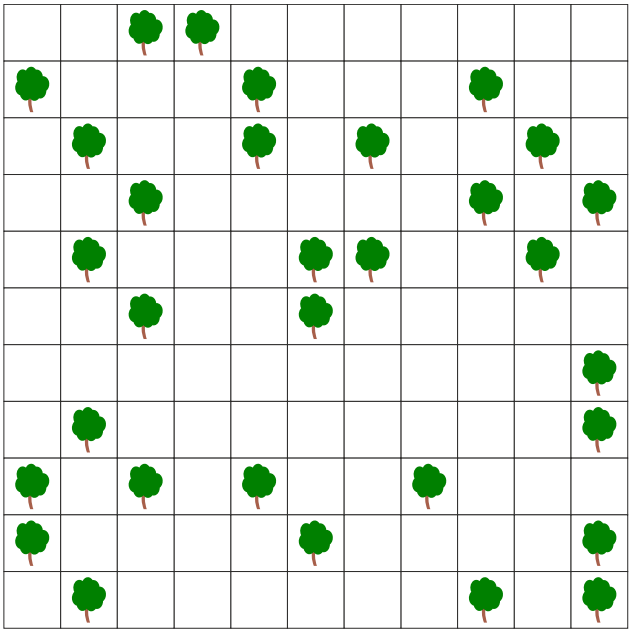
Coco le petit singe peut sauter depuis un arbre aussi loin que le montre le domaine coloré en vert.



Dans l'exemple suivant, Coco peut atteindre les arbres en couleur d'un seul saut. En deux sauts, il peut aussi atteindre les deux arbres gris du haut, mais pas l'arbre gris du bas.

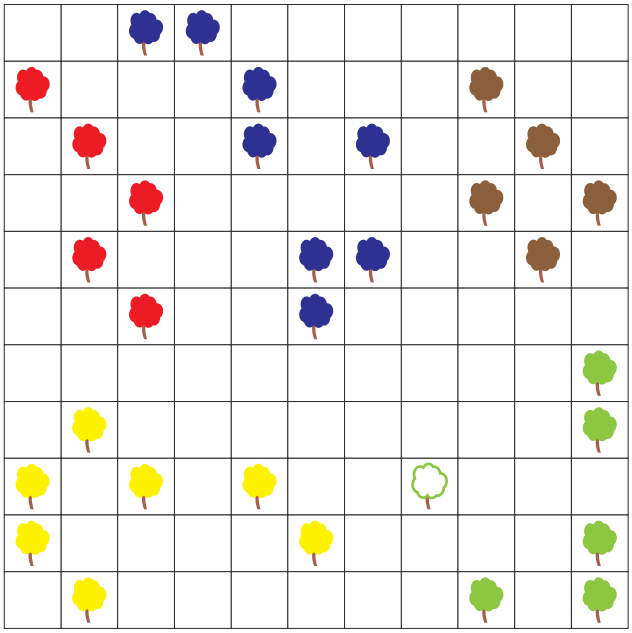
Il existe des groupes d'arbres dans lequels Coco peut se déplacer sans toucher le sol une seule fois.

Sélectionne tous les arbres du plus grand de ces groupes.



Solution

Sur l'image ci-dessous, deux arbres ont la même couleur si Coco peut passer de l'un à l'autre sans toucher le sol.

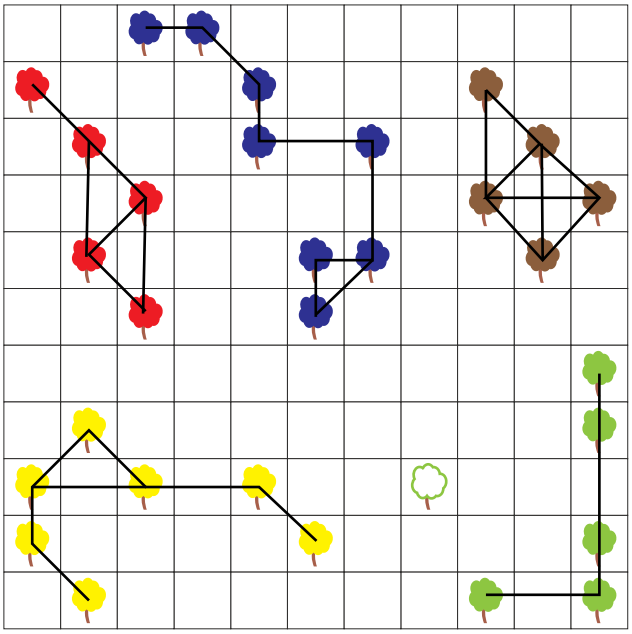


On voit que le groupe d'arbres bleus comptant huit arbres est le plus grand des groupes.

C'est de l'informatique !

Si Coco peut sauter directement d'un arbre à l'autre, ils sont pour ainsi dire connectés l'un à l'autre. On peut représenter cela à l'aide d'un ligne entre les arbres comme montré plus bas. On obtient donc un graphe dont les arbres sont les nœuds avec des arêtes entre les arbres connectés. Coco peut passer d'un arbre à l'autre en sautant seulemnt s'il existe un chemin le long des arêtes menant d'un arbre à l'autre.

On appelle un groupe de nœuds *connexe* si tous les nœuds sont connectés ensemble par des arêtes. Lorsqu'un tel groupe ne peut plus être agrandi sans perdre la connection entre les nœuds, on parle d'une *composante connexe*. Un graphe peut être divisé de manière unique en composantes connexes ; elles sont colorées sur l'image ci-dessous.



On peut faciliement identifier une composante connexe en partant d'un nœud quelconque puis en cherchant tous les nœuds accessibles par des arêtes.

Mots clés et sites web

- Graphe connexe : `https://fr.wikipedia.org/wiki/Graphe_connexe`

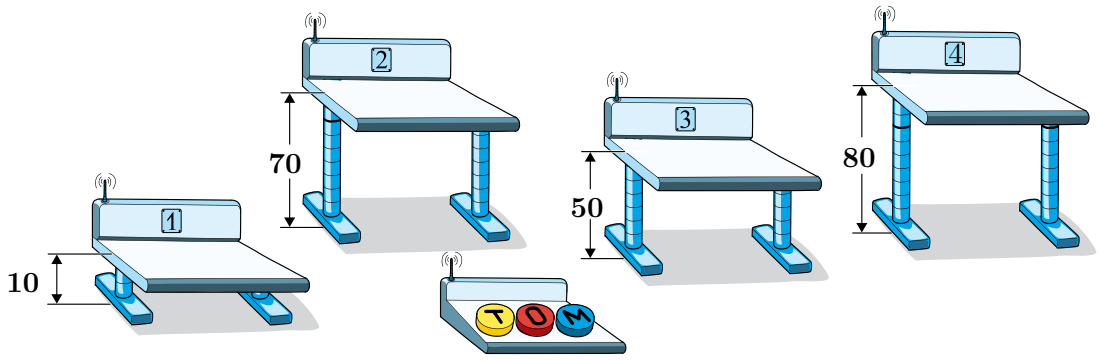
14. Sacrés pupitres

Dans la salle de classe, il y a des pupitres dont la hauteur est réglable électriquement. Tous les pupitres doivent être réglés à 60 cm pour les cours. La hauteur peut être changée à l'aide des touches T, O et M d'une télécommande. Quelqu'un a joué avec la télécommande et l'a reprogrammée. Maintenant, les trois touches fonctionnent comme cela :

- T monte les pupitres 1, 2 et 3 de 10 cm chacun.
- O baisse les pupitres 2, 3 et 4 de 10 cm chacun.
- M monte les pupitres 1, 3 et 4 de 10 cm chacun.

Ces actions sont exécutées chaque fois que l'on appuie sur le bouton.

En ce moment, les hauteurs des pupitres 1, 2, 3 et 4 sont de à 10 cm, 70 cm, 50 cm et 80 cm :



Comment peut-on régler la hauteur des quatre pupitres à 60 cm ?

A) Appuie 4 × sur T, 5 × sur O et 1 × sur M.
B) Appuie 5 × sur T, 1 × sur O et 0 × sur M.
C) Appuie 3 × sur T, 4 × sur O et 2 × sur M.
D) Appuie 2 × sur T, 4 × sur O et 6 × sur M.

Solution

La bonne réponse est C) Appuie 3 × sur T , 4 × sur O et 2 × sur M.

Tu peux constater que les trois touches de la télécommande changent la hauteur de 10 cm, donc toujours de la même distance. Deux des touches font monter les pupitres (T et M) et seulement un touche les fait descendre (O). De plus, chacune des trois touches modifie la hauteur de trois pupitres ; il y a donc toujours un pupitre dont la hauteur ne change pas. La touche O n'a aucun effet sur le pupitre 1, on ne peut donc pas le faire descendre.

Le pupitre 1 est 50 cm trop bas. On peut en déduire que l'on doit appuyer exactement 5 fois sur les touches T ou M (la somme des nombres de fois où les touches T et M sont utilisées doit être égale à 5). On peut exprimer cela par l'équation $T + M = 5$. On peut donc éliminer la solution D), car $T + M = 8$ pour cette solution. D'après la suite de touches de la solution D), la pupitre 1 serait haut de $10 + 20 + 60 = 90$ cm, c'est à dire la hauteur de départ 10 cm plus $2 * 10$ cm pour T plus $6 * 10$ cm pour M.

le pupitre 2 est 10 cm trop haut. M n'a pas d'effet sur le pupitre 2. La bonne solution doit donc satisfaire l'équation $T - O = -1$. On peut donc éliminer la solution B), car en l'utilisant, le pupitre 2 aurait à la fin une hauteur de $70 + 50 - 10 = 110$ cm.

Le pupitre 3 est 10 cm trop bas, ce qui donne l'équation $T - O + M = 1$. Les solutions A) et B) peuvent donc être éliminées. Avec la solution A), le pupitre aurait la même hauteur à la fin qu'au début : $50 + 40 - 50 + 10 = 50$ cm ; avec la solution B, la hauteur du puptire serait $50 + 50 - 10 = 90$ cm. Toutes les solutions excepté la solution C) ont maintenant été éliminées.

Il faut encore vérifier que la solution C) met aussi le pupitre 4 à la bonne hauteur. Le pupitre 4 est 20 cm trop haut et la touche T n'a aucun effet sur sa hauteur. Il faut donc appuyer deux fois sur O et une fois de plus pour chaque fois que la touche M est utilisée. La suite de touches de la solution C) donne la hauteur $80 - 40 + 20 = 60$ cm.

Comme on a déjà constaté plus haut que la solution C) permettait de mettre les pupitres 1, 2 et 3 à la bonne hauteur, on est maintenant sûr que cette solution fonctionne.

Un autre moyen de trouver la solution est de résoudre quatre équations linéaires. Pour chaque pupitre, on écrit une équation décrivant quelles touches changent sa hauteur et quelle est le changement de hauteur désiré. Par exemple, la hauteur du pupitre 1 ne change qu'avec les touches T et M et le changement désiré est de 50 cm, ce que l'on peut obtenir en appuyant sur 5 touches (étant donné que pression sur une touche change la hauteur de 10 cm).

Comme il y a quatre pupitres et trois touches, on obtient quatre équations linéaires avec trois inconnues :

$$\begin{aligned} T + M &= 5 \\ T - O &= -1 \\ T - O + M &= 1 \\ -O + M &= -2 \end{aligned}$$

En soustrayant la troisième équation de la première, on obtient $O = 4$. en substituant dans la deuxièeme équation, on obtient $T = 3$. Toutes les équations sont résolues juste en prenant $M = 2$. C'est donc la seule bonne solution.

C'est de l'informatique !

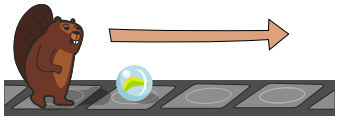
Cet exercice est un problème typique du domaine de l'*optimisation linéaire en nombres entiers*, aussi appelée *programmation linéaire en nombres entiers*. Le problème est donné par un certain nombre de contraintes. Dans ce cas particulier, on peut toutes les formuler sous la forme d'équations linéaires. L'optimisation est un but typique en informatique : on cherche une suite d'action permettant d'arriver à un but prédéfini. On pourrait même décrire tout l'exercice par la recherche d'un chemin dans un espace quadridimensionnel avec trois déplacments possibles pour passer du point $(10, 70, 50, 80)$ au point $(60, 60, 60, 60)$. Cet exercice n'a qu'une solution, mais de tels problèmes ont souvent de multiples solutions, rendant l'optimisation possible. On cherche alors le minimum de la fonction linéaire $T + M + O$.

Mots clés et sites web

- Optimisation linéaire en nombres entiers :
  `https://fr.wikipedia.org/wiki/Optimisation_linéaire_en_nombres_entiers`

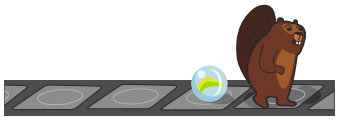
15. Ruban de billes

Sur un ruban, un castor se déplace d'une case à la suivante de gauche à droite. Sur chaque case du ruban peut se trouver une bille.



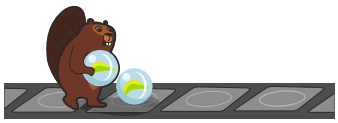
Si le castor arrive sur une case avec une bille et a les mains libres, il la soulève et la prend avec en la portant dans ses bras.

Il dépose la bille sur la prochaine case libre.



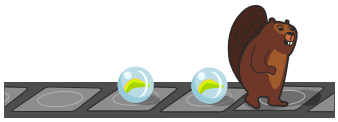
Le castor ne peut porter qu'une seul bille à la fois, et il n'y a la place que pour une bille sur chaque case.

Si le castor porte déjà une bille en arrivant sur une case avec une autre bille...



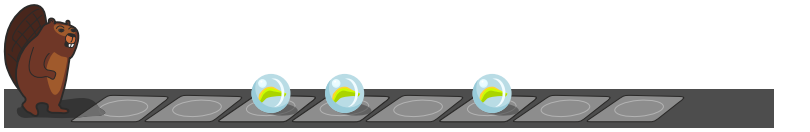
... il dépasse celle-ci...

... et pose sa bille sur la prochaine case vide.



Il peut ensuite à nouveau soulever la bille suivante.

Le castor se trouve devant une partie du ruban sur laquelle sont posées trois billes. Sur quelles cases les billes se trouvent-elles une fois que le castor a traversé cette partie du ruban ?

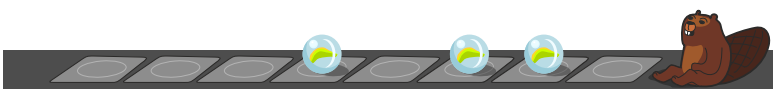


A) 

B) 

C) 

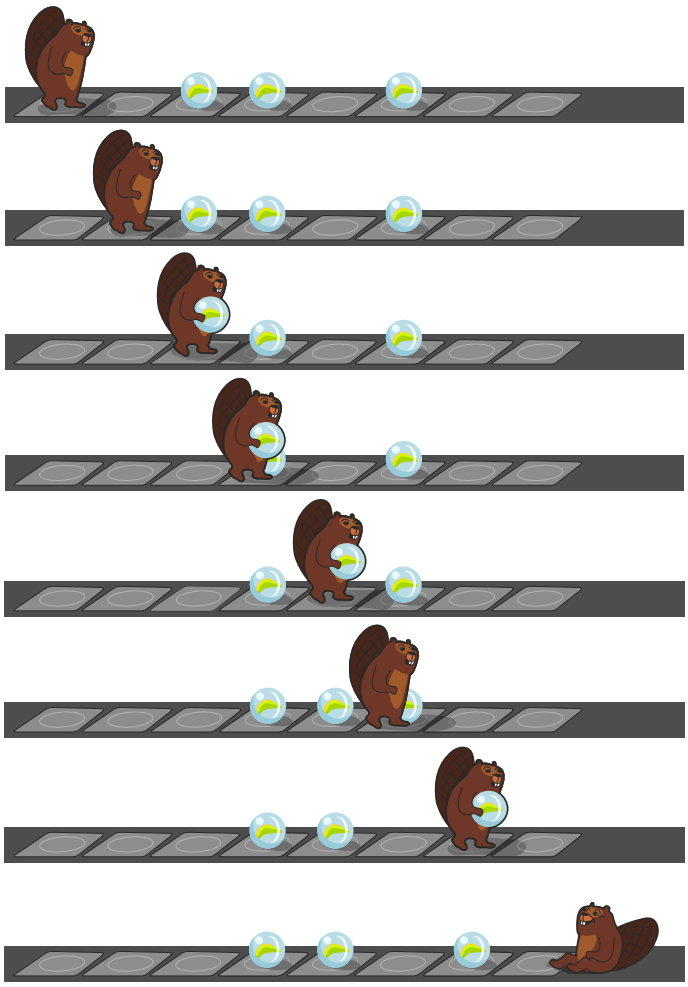
D) 

E) 

Solution

La bonne réponse est C :

L'image suivante montre le déroulement :



C'est de l'informatique !

En informatique, des opérations relativement simples donnent souvent des résultats intéressants. Cet exercice en est un bon exemple. La démarche du castor est un *algorithme*. Il se base sur le fait que le castor peut adopter deux états différents (portant une bille ou n'en portant pas) et qu'il peut trouver deux sortes de cases sur son chemin (occupées ou libres).

Aussi simple que soit l'exemple dans cet exercice, il contient plusieurs des éléments essentiels d'une *machine de Turing*. Une machine de Turing (qui doit son nom au mathématicien Alan Turing) est un ordinateur particulier qui a une structure très simple. En principe, une machine de Turing peut exécuter tous les algorithmes qu'un ordinateur traditionnel peut exécuter. En pratique, les machines

de Turing ne sont pas utilisées comme ordinateur, car nous pouvons construire des ordinateurs qui sont bien plus efficaces, même s'ils sont plus compliqués. Les machines de Turing sont surtout utiles pour la théorie. Leur structure simple permet de prouver des affirmations simples les concernant ; et si l'on peut prouver qu'un exercice ne peut pas être résolu par une machine de Turing, cela veut dire qu'aucun de nos ordinateurs ne peut le résoudre non plus.

Une machine de Turing est composée :

- D'un *ruban* de longueur infinie divisé en *cases*. Chaque case peut contenir un *symbole*. Dans notre cas, il s'agit des cases sur lesquelles le castor se déplace.
- D'une quantité finie de *symboles*. Souvent, on n'utilise que `0` et `1` comme symboles. Dans notre exemple, une bille représente le `1` et une position libre le `0`.
- D'une tête de lecture/écriture qui peut se déplacer sur le ruban dans les deux directions tout en lisant les symboles écrits et en écrivant de nouveaux symboles sur le ruban. Dans notre exemple, le castor joue le rôle de la tête de lecture/écriture.
- D'un registre d'*états* de taille finie. Le comportement de la tête de lecture/écriture peut changer en fonction de l'état. Dans notre cas, il n'y a que deux états : « portant une bille » ou « ne portant pas de bille ».
- D'un ensemble de règles, ou *table d'actions* : que ce passe-t-il, en fonction de l'état, lorsqu'un symbole précis est lu sur le ruban ? Les actions possibles sont : un changement d'état, l'écriture d'un nouveau symbole et le déplacement de la tête de lecture d'une case vers la gauche ou vers la droite.

Mots clés et sites web

- Machine de Turing : `https://fr.wikipedia.org/wiki/Machine_de_Turing`

A. Auteur·e·s des exercices

Michael Barot

Wilfried Baumann

Linda Björk Bergsveinsdóttir

Javier Bilbao

Lucia Budinská

Sarah Chan

Kris Coolsaet

Christian Datzko

Susanne Datzko

Janez Demšar

Fabian Frei

Jens Gallenbacher

Thomas Galler

Mathias Hiron

Juraj Hromkovič

Alisher Ikramov

Vaidotas Kinčius

Regula Lacher

Taina Lehtimäki

Angélica Herrera Loyo

Mochammad Irfan Noviana

Gabriela Lourdes Rodríguez Parada

Elsa Pellet

Jean-Philippe Pellet

Zsuzsa Pluhár

Wolfgang Pohl

Rodrigo Santamaría

Eljakim Schrijvers

Tomas Šiaulys

Timur Sitdikov

Bernadette Spieler

Ahto Truu

Florentina Voboril

Eslam Wageed

Michael Weigend

Kyra Willekes

Mija Zaļūksne

B. Sponsoring : Concours 2021

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.phzh.ch/`
Pädagogische Hochschule Zürich

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.