INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Exercices et solutions 2021

Années HarmoS 9/10

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

Éditeurs :

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

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 :

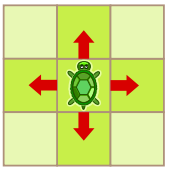
SVIA-SSIE-SSII Société Suisse pour l'Informatique dans l'Enseignement
Castor Informatique
Gabriel Parriaux

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

Table des matières

1. Chemins tortueux

Une tortue doit brouter plusieurs jardins. Chaque jardin est divisé en carrés qui sont recouverts soit de gazon, soit de pierres. La tortue ne peut pas traverser un carré avec des pierres, mais elle peut passer d'une case d'herbe à une autre case d'herbe qui se trouve directement à côté.

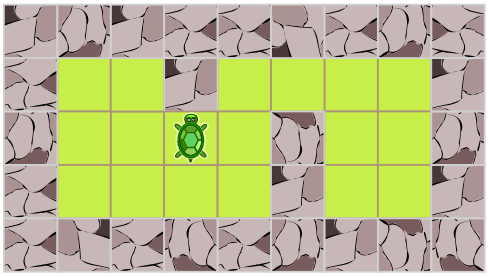


La tortue doit complètement brouter les jardins. elle commence sur la case sur laquelle elle est sur l'image. À la fin, elle doit avoir passé exactement une fois sur chaque case d'herbe.

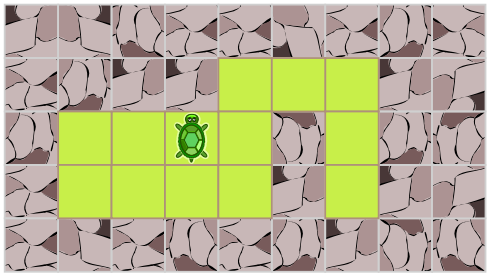
Il y a malheureusement un jardin qu'elle ne peut pas brouter complètement de cette façon.

De quel jardin s'agit-il ?

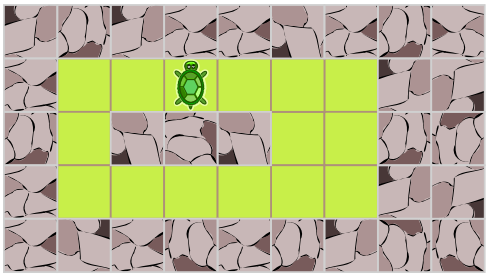
A)



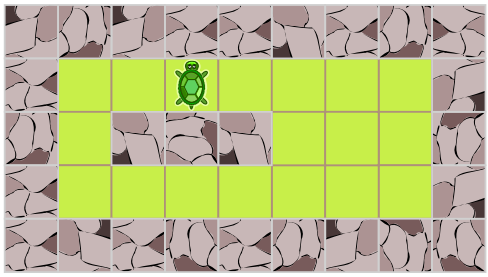
B)



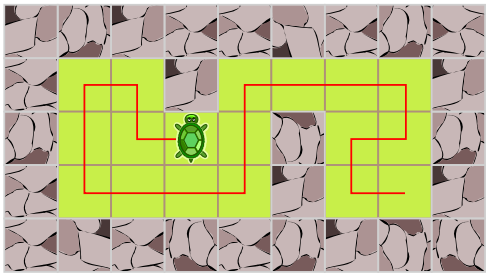
C)

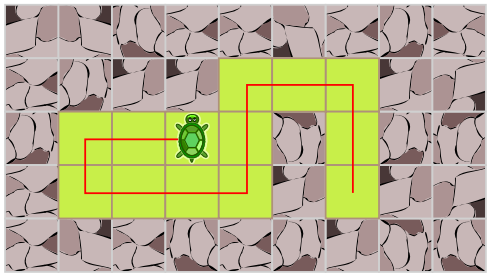


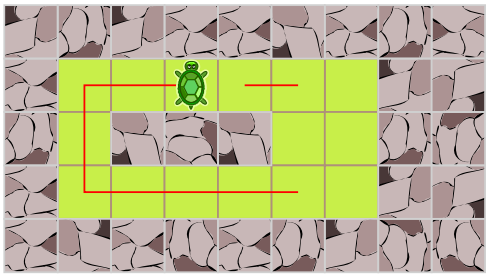
D)

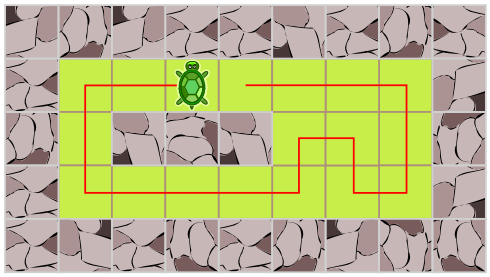


Solution

A) 

B) 

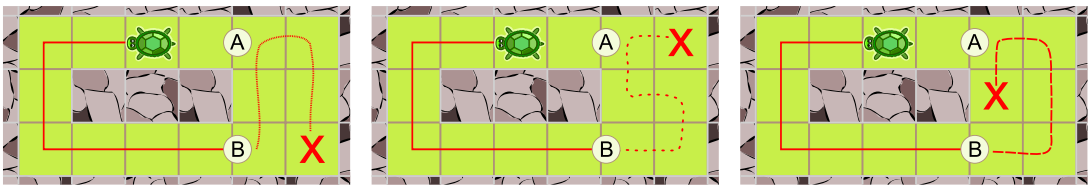
C) 

D) 

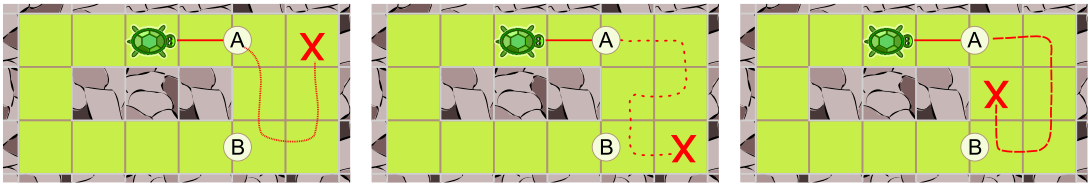
La tortue peut brouter les jardins A, B et D complètement.

La tortue ne peut pas brouter le jardin C de cette façon. La tortue a deux possibilités depuis son point de départ :

- Si elle part d'abord vers la gauche, elle arrive au point B. Depuis là, elle devrait brouter les 6 cases de droite de manière à terminer au point A, mais aucun des chemins possibles depuis le point B ne se termine au point A.



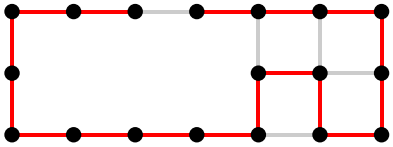
- Si la tortue part d'abord vers la droite, alle arrive au point B et devrait brouter les 6 cases de manière à atteindre le point A à la fin. On peut utiliser le même argument que si dessus en inversant le haut et le bas. Il n'y a donc pas non plus de chemin adapté.



C'est de l'informatique !

La tortue doit trouver un chemin à travers un jardin en passant exactement une fois par chaque case d'herbe. Le problème de cet exercice est un problème du type *chemin hamiltonien*

Les carrés d'herbes du jardin de la tortues peuvent être considérés comme des *nœuds*. On représente alors le jardin D comme ceci :



Au XIX^e siècle, Sir William Hamilton se demanda s'il existait pour de telle structures (appelées *graphes* en informatique et en mathématiques) un chemin les long des arêtes passant exactement une fois par chaque nœud. C'est pour cela que l'on appele un tel chemin un *chemin hamiltonien*. La question de savoir si un chemin hamiltonien existe dans un certain graphe est en règle générale très difficile à résoudre. Personne de connait d'*algorithme* permettant de déterminer si un chemin hamiltonien existe dans un graphe quelconque de manière efficiente (assez rapidement pour que ce soit utile). Cela est vrai de tous les problèmes dits *NP-complets*, dont le problème du chemin hamiltonien est le plus connu.

Mots clés et sites web

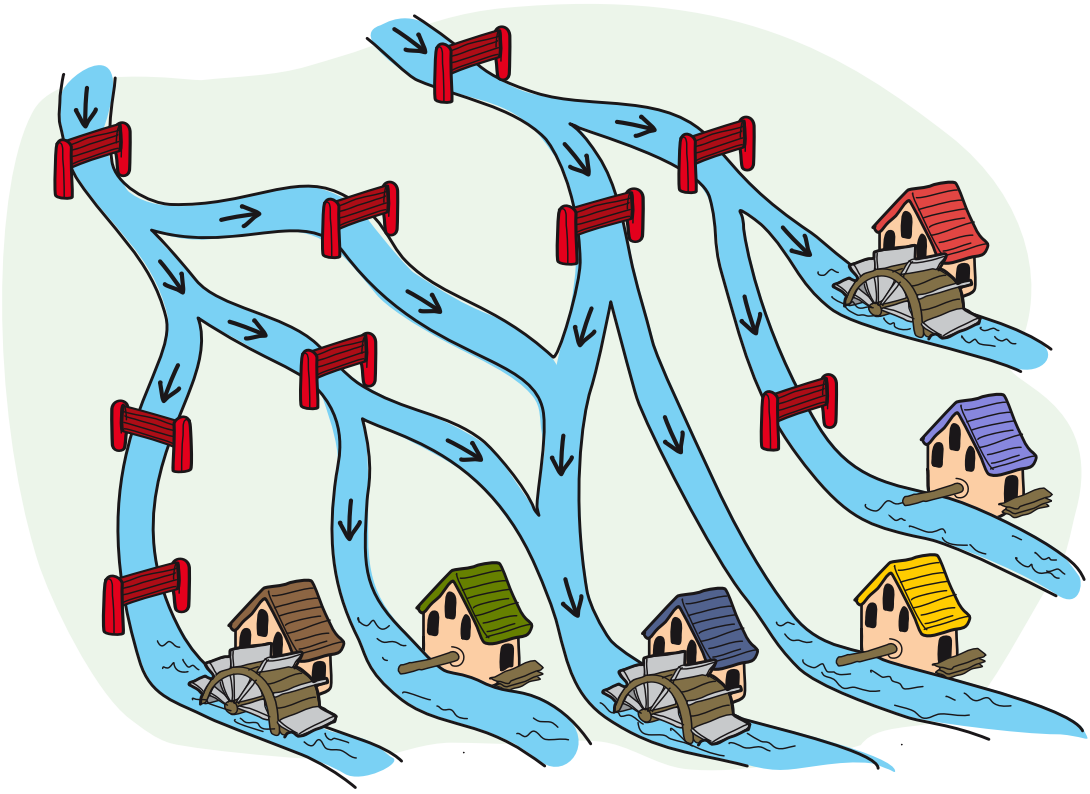
- Graphe hamiltonien, chemin hamiltonien :
  `https://fr.wikipedia.org/wiki/Graphe_hamiltonien`

2. Les moulins de castor Max

Le meunier Max a six moulins. Il doit encore fixer la roue de trois d'entre eux. Pour cela, il doit empêcher l'eau d'arriver à ces moulins. L'eau doit par contre continuer de couler jusqu'aux autres moulins.

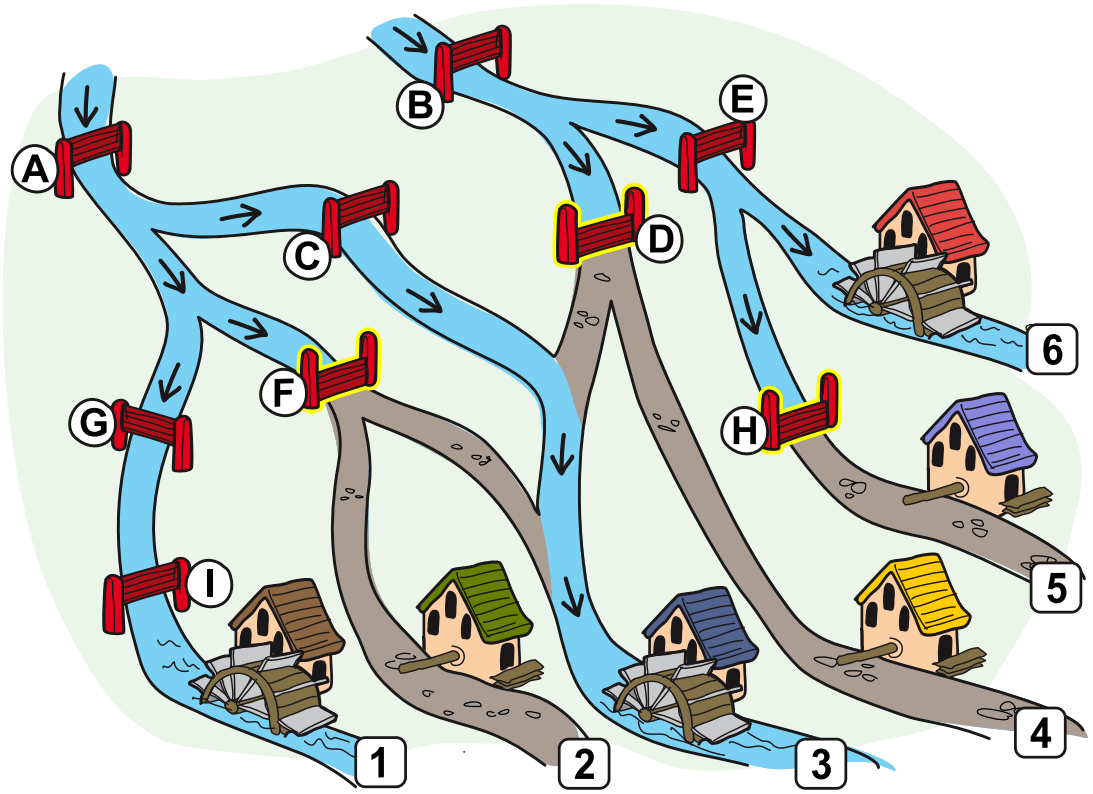
L'eau ne peut couler que vers le bas. Un clapet fermé empêche l'eau de couler.

Quels clapets faut-il fermer ?



Solution

La bonne réponse est qu'il faut fermer les trois clapets qui sont nommés D, F et H dans le dessin suivant.



C'est la seule possibilité permettant de couper l'eau aux moulins 2, 4 et 5 tout en continuer d'alimenter en eau les trois moulins 1, 3 et 6 :

- Les clapets A, G et I doivent rester ouverts, car sinon le moulin 1 n'est pas alimenté en eau.
- Les clapets B et E doivent également rester ouverts pour que le moulin 6 soit alimenté en eau.
- Comme les clapets B et E sont ouverts, le clapet H doit être fermé pour éviter que l'eau n'arrive au moulin 5.
- Comme le clapet A est ouvert, le clapet F doit être fermé pour éviter que l'eau n'arrive au moulin 2.
- Comme le clapet B est ouvert, le clapet D doit être fermé pour éviter que l'eau n'arrive au moulin 4.
- Comme les clapets D et F sont fermés, le clapet C doit être ouvert pour que le moulin 3 soit alimenté en eau.

C'est de l'informatique !

Dans cet exercice, l'écoulement de l'eau est régulé par des *conditions*. Par exemple, l'eau ne coule jusqu'au moulin 6 que si les deux clapets B et E sont ouverts. Voici un autre exemple un peu plus

complexe : l'eau ne coule jusqu'au moulin 3 que si au moins l'une des deux ou les deux conditions suivantes sont remplies :

- Le clapet A est ouvert et l'un des deux clapets C ou F est ouvert.
- Les deux clapets B et D sont ouverts.

De telles combinaisons de conditions sont obtenues à l'aide des *opérateurs logiques* ET (symbolisé par $\wedge$) ou OU (symbolisé par $\vee$). De tels opérateurs connectent des valeurs de vérité comme vrai et faux. Si A et B sont deux valeurs de vérité, on peut indiquer quelles valeurs de vérité les expressions « A ET B » et « A OU B » ont en fonction de A et B :

A	**B**	**A ET B**	**A OU B**
faux	faux	faux	faux
vrai	faux	faux	vrai
faux	vrai	faux	vrai
vrai	vrai	vrai	vrai

En informatique (et en mathématiques), l'expression « A OU B » est donc aussi considérée comme juste si A et B sont les deux justes. L'affirmation « le moulin 6 est alimenté » est équivalente à :

> « le clapet B est ouvert » ET « le clapet E est ouvert ».

Dans le deuxième exemple, l'affirmation « le moulin 3 est alimenté » est équivalente à :

> (« le clapet A et ouvert » ET (« le clapet C est ouvert » OU « le clapet F est ouvert ")) OU (« le clapet B est ouvert » ET « le clapet D est ouvert »).

En programmation, il est important de formuler les conditions de manière exacte. Chaque ET et chaque OU combine deux affirmations. Les parenthèses déterminent dans quel ordre les affirmations sont combinées. Les combinaisons à l'aide d'opérateurs logiques et de parenthèses sont utiles pour formuler des conditions complexes. Des conditions sont utilisées aussi bien pour des branchements avec `if` que pour des boucles `while` afin de guider le déroulement d'un programme.

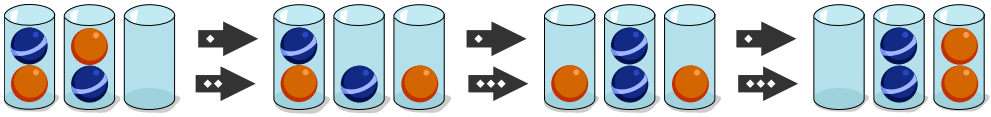
Mots clés et sites web

- Instruction conditionnelle : `https://fr.wikipedia.org/wiki/Instruction_conditionnelle_(programmation)`
- Variable booléenne : `https://fr.wikipedia.org/wiki/Booléen`
- Algèbre de Boole : `https://fr.wikipedia.org/wiki/Algèbre_de_Boole_(logique)`

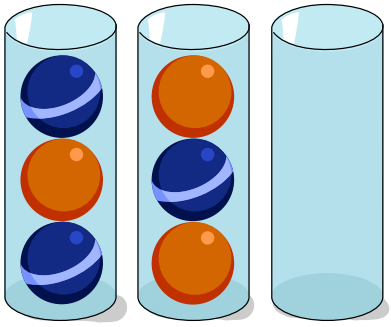
3. Jeu de balles

Les castors aimeraient trier des balles par couleur. À la fin, toutes les balles doivent se trouver dans deux verres. Toutes les balles qui se trouvent dans un verre doivent avoir la même couleur. Ils doivent suivre les règles suivantes :

- Règle 1 : Seule la balle la plus haute d'un verre peut être déplacée dans un autre verre.
- Règle 2 : Une balle peut toujours être mise dans un verre vide.
- Règle 3 : Une balle peut être mise dans un verre non vide uniquement s'il y reste de la place et que la balle en dessous a la même couleur que la balle déplacée.

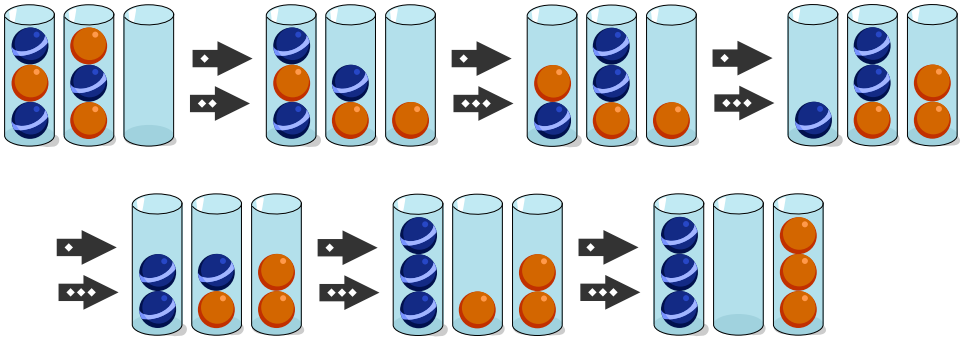


Trie les balles en les déplaçant d'après les trois règles.



Solution

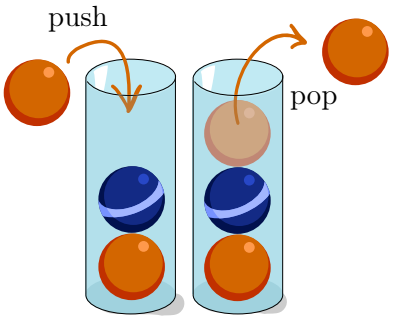
Les balles peuvent être déplacées dans l'ordre suivant :



Il faut au moins six étapes pour réarranger les balles. Il existe d'autres possibilités de réarranger les balles en seulement six étapes.

C'est de l'informatique !

Dans cet exercice, tu déplaces les balles comme un ordinateur gère les données enregistrées dans une *pile* : il ne peut qu'*empiler* (*push* en anglais) un élément en haut de la pile et *dépiler* (*pop* en anlgais) l'élément du haut de la pile.



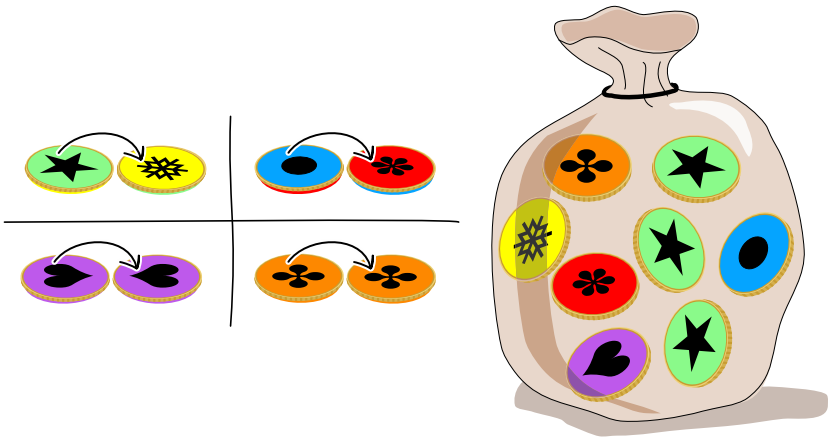
L'ordinateur ne peut accéder aux éléments du bas que si les balles du dessus ont d'abord été retirées. L'élément qui a été enregistré en dernier va être retiré en premier par l'ordinateur. Ce principe est appelé *dernier arrivé, premier sorti* en informatique.

Mots clés et sites web

- Pile : `https://fr.wikipedia.org/wiki/Pile_(informatique)`

4. Sac de pièces

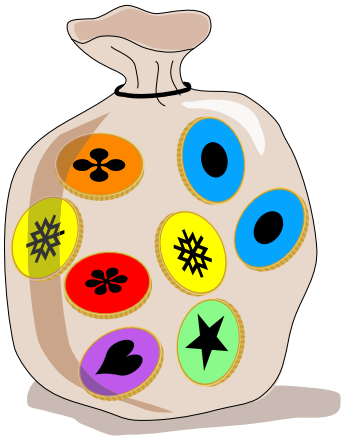
Il existe quatre sortes de pièces de monnaie différentes dans le pays d'Émile. Tu peux voir ici les deux côtés de ces pièces ainsi que le sac d'Émile avec ses pièces.



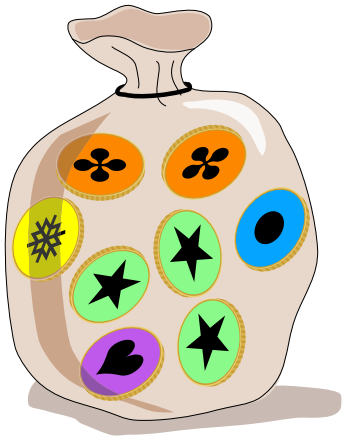
Émile secoue son sac de pièces.

Quel sac est celui d'Émile ?

A)



B)



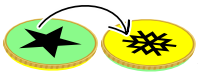
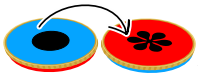
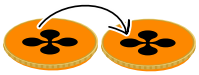
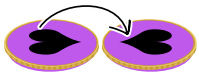
C)

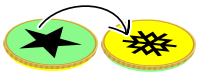
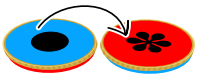
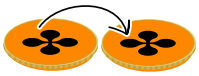
D)

Solution

La bonne réponse est C :

Dans le sac d'Émile, il y a :

- 4 pièces  ,
- 2 pièces  ,
- une pièce 
- et une pièce  .

	Sac d'Émile	Sac A	Sac B	**Sac C**	Sac D
	**4**	3	4	**4**	2
	**2**	3	1	**2**	4
	**1**	1	2	**1**	1
	**1**	1	1	**1**	1

Seul le sac C contient le même nombre de chaque sorte de pièces que le sac d'Émile. C'est donc la bonne solution.

C'est de l'informatique !

Dans cet exercice, il faut reconnaître les différentes sortes de pièces sans en voir les deux côtés. On n'a qu'une information partielle. Dans un système informatique, les objets du monde réel sont enregistrés avec leurs caractéristiques essentielles. Souvent, il suffit de connaître une partie de ces caractéristiques pour pouvoir reconnaître un objet. La caméra d'un véhicule autonome ne voit qu'une partie de la

réalité, mais doit quand même être en mesure de reconnaître des véhicules et usagers de la route et de réagir au traffic de manière adaptée. L'intelligence artificielle des systèmes informatiques apprend peu à peu et de mieux en mieux à reconnaître des objets à partir de fragments, comme les êtres humains le font.

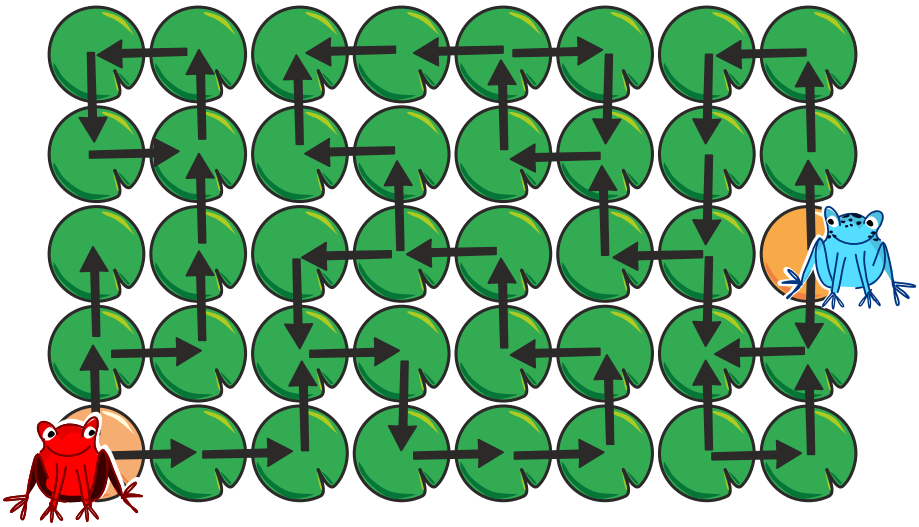
Mots clés et sites web

- Multiensemble : `https://fr.wikipedia.org/wiki/Multiensemble`
- Informations non structurées : `https://fr.wikipedia.org/wiki/Informations_non_structurées`

5. Rencontre

Sur un étang couvert de nénuphars, deux grenouilles peuvent se déplacer en sautant de feuille en feuille — mais seulement en suivant le sens des flèches.

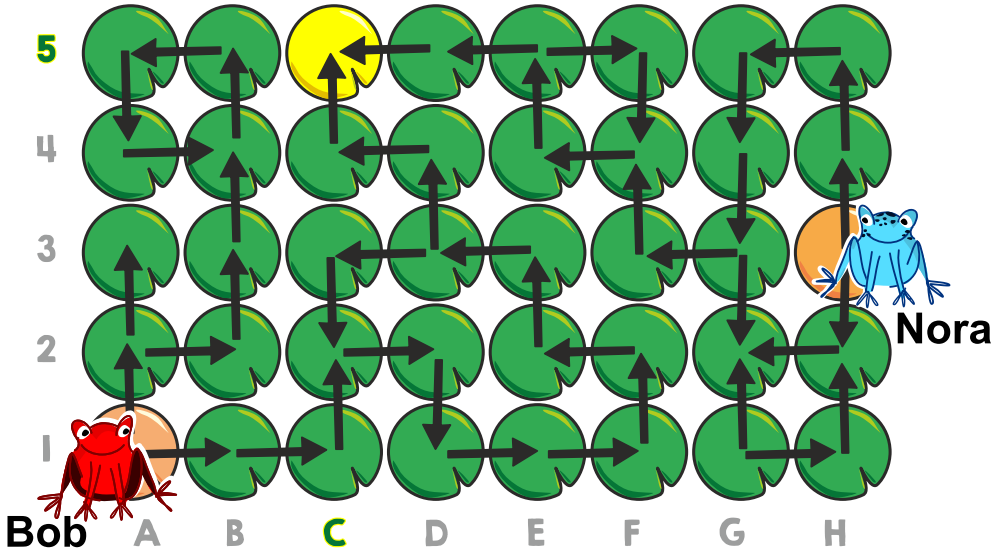
Sur quelle feuille les deux grenouilles peuvent-elles se rencontrer ?



Man kann auf die Blätter klicken. Klickt man auf ein Blatt, wird dieses ausgewählt und gleichzeitig ein bereits ausgewähltes Blatt wieder deaktiviert.

Solution

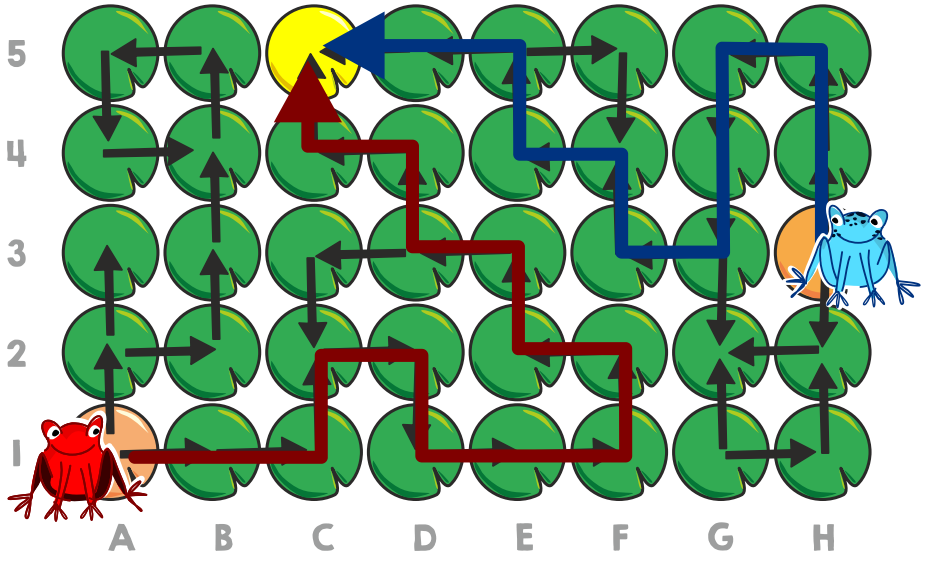
Les grenouilles ne peuvent se rencontrer que sur la feuille C5.



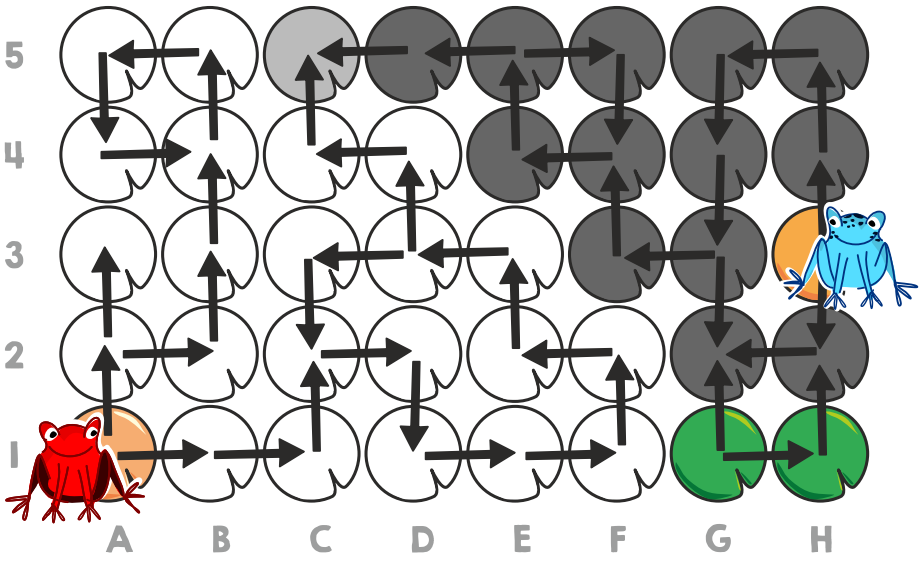
Bob, la grenouille rouge, a deux possibilités depuis sa feuille de départ : si elle va vers le « haut », elle arrive soit dans le cul de sac sur A3 ou elle reste bloquée dans le cercle qui commence sur B4. Si elle va vers la « droite » (en direction de B1), elle peut sauter jusqu'à D3. Depuis D3, elle peut soit tourner en rond en partant vers la « gauche », soit aller vers le « haut » jusqu'à un autre cul de sac sur C5.

Nora, la grenouille bleue, a aussi le choix entre deux directions depuis sa feuille de départ. Si elle va vers le « bas », elle arrive dans le cul de sac sur G2. Si elle va vers le « haut », elle peut sauter jusqu'à G3. Depuis là, elle peut soit arriver dans le cul de sac sur G2 ou partir à « gauche » et atteindre E5. Depuis E5, elle peut soit tourner en rond, soit partir à « gauche » et arriver dans le cul de sac sur C5.

Nous savons que Bob peut aussi atteindre C5, elles peuvent donc s'y rencontrer. Le dessin montre les chemins qu'elles suivent pour y arriver.



Cela ne garantit pas encore qu'elle ne puissent pas aussi se rencontrer ailleurs. Le dessin suivant montre toutes le feuilles que Bob (en blanc) en Nora (en gris foncé) peuvent atteindre en suivant les flèches de toutes les manières possibles. On voit que seule la feuille C5 peut être atteinte par les deux grenouilles.



C'est de l'informatique !

Comment peut-on générer la dernière image ? Les feuilles pouvant être atteintes par une grenouille peuvent être trouvées à l'aide d'un *parcours en largeur* ou d'un *parcours en profondeur*. Ce sont deux des méthodes les plus importantes en informatique. Grâce à elles, on peut déterminer quelles sont les feuilles blanches et les feuilles gris foncé. Il ne reste ensuite plus qu'à trouver les feuilles pouvant être atteintes par les deux grenouilles.

Mots clés et sites web

- Parcours en largeur :
  `https://fr.wikipedia.org/wiki/Algorithme_de_parcours_en_largeur`
- Parcours en profondeur :
  `https://fr.wikipedia.org/wiki/Algorithme_de_parcours_en_profondeur`

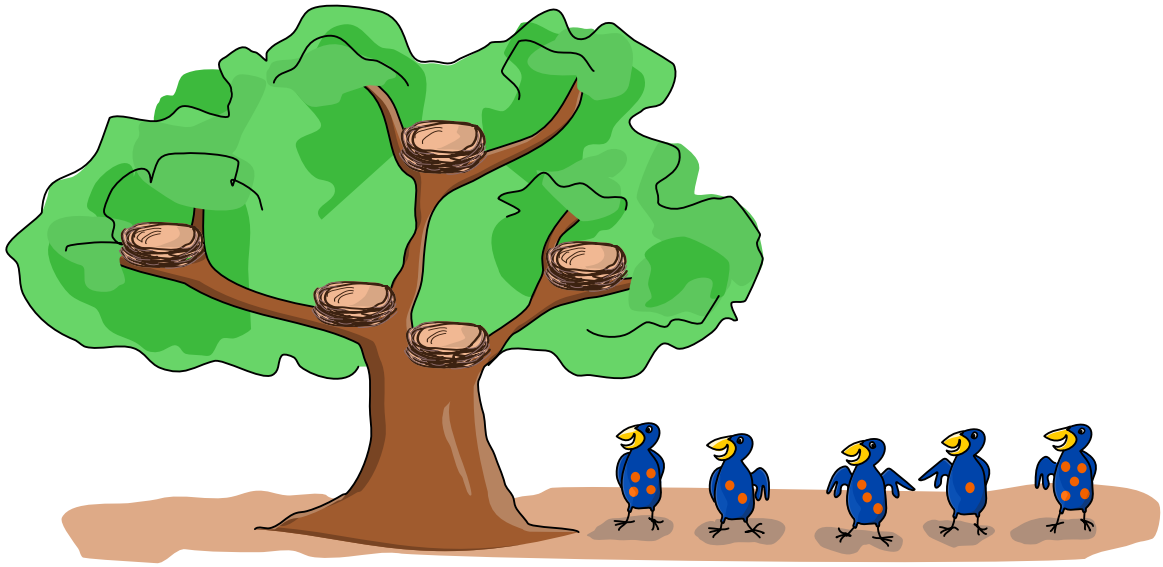
6. Emménagement

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.

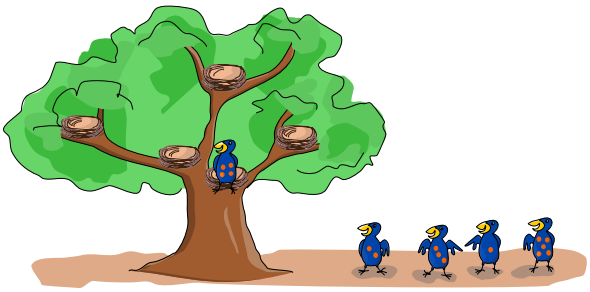
Où se trouvent les tachetés à la fin ? Place chaque tacheté dans le bon nid.



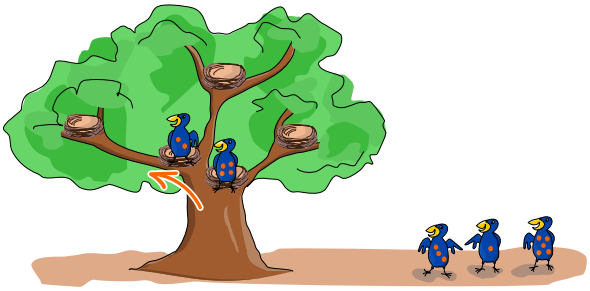
Solution

On obtient la bonne solution de la manière suivante :

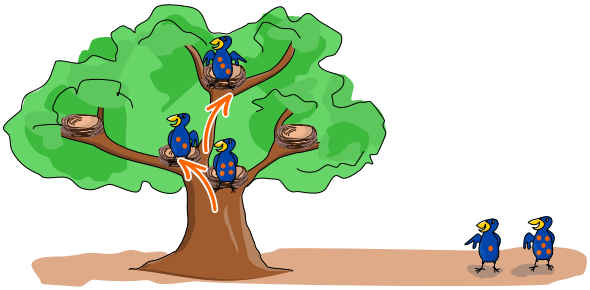
Le premier tacheté, celui à 4 points, arrive dans le nid du bas et y emménage.



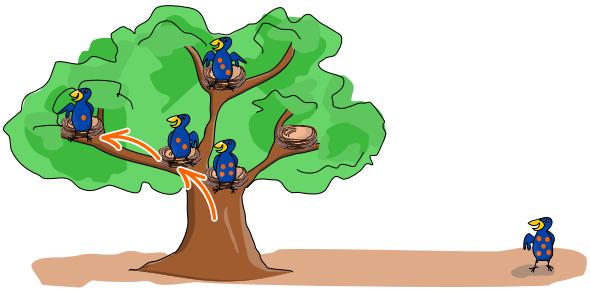
Le deuxième tacheté a 2 points. Le nid du bas est maintenant occupé par le premier tacheté à 4 points. Comme 4 est plus grand que 2, le deuxième tacheté continue à grimper vers la gauche et emménage dans le premier nid vide.



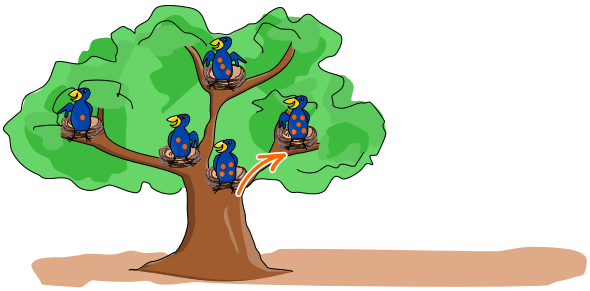
Le troisième tacheté a 3 points. Il grimpe vers la gauche après le nid du bas dans lequel est le tacheté à 4 points, car 4 est plus grand que 3. Le tacheté à 2 points est dans le nid suivant. Comme 3 est plus grand que 2, le troisième tacheté continue de grimper vers la droite. Il emménage dans le prochain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

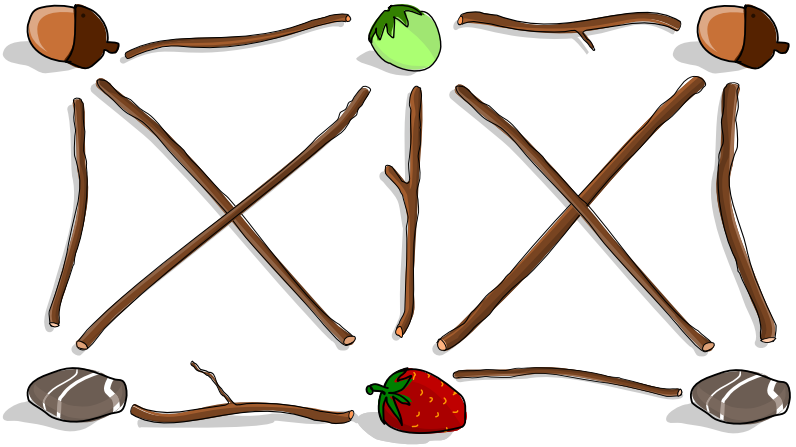
Mots clés et sites web

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

7. 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.

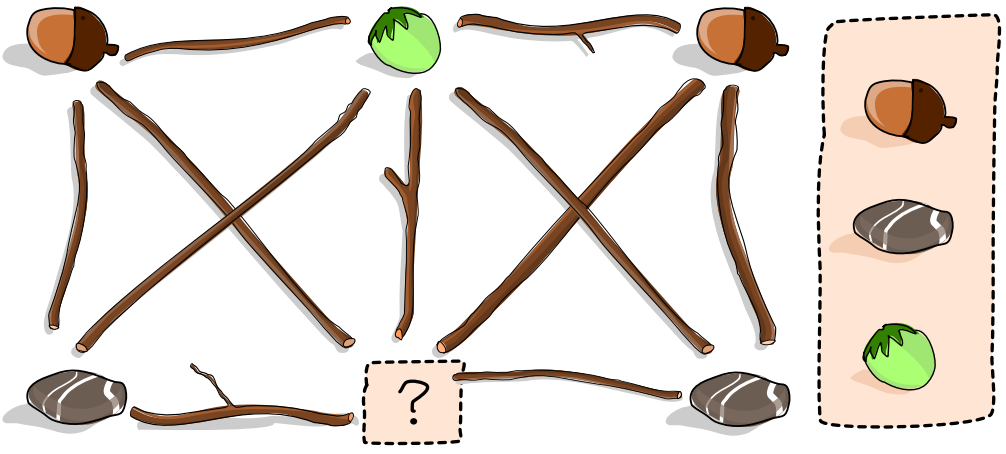
Ensuite, Anja dispose des branches entre ces objets. Elle suit pour cela la règle suivante : une branche ne doit pas se trouver entre deux objets pareils, par exemple entre deux glands. Voici l'œuvre d'art terminée :



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

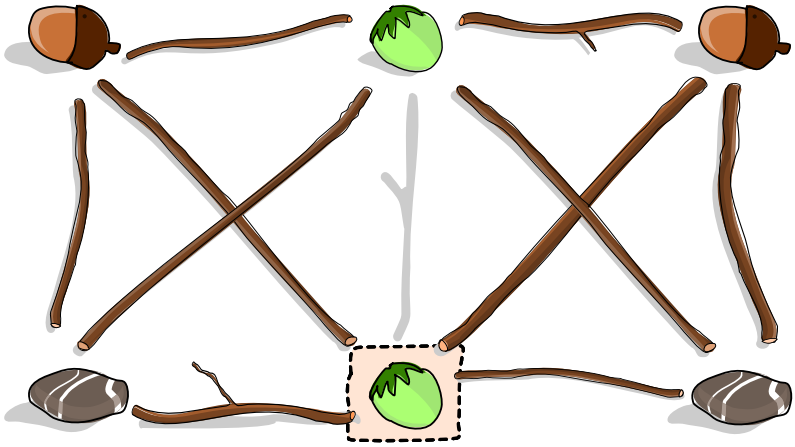
Peux-tu aider le frère d'Anja à dissimuler son méfait ?

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.



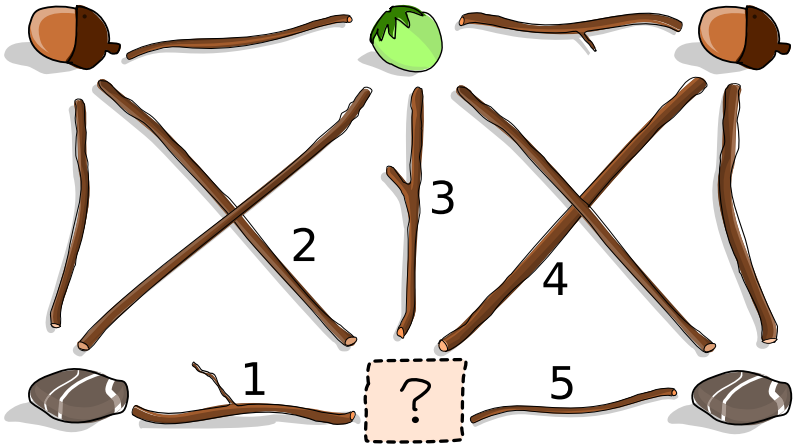
Solution

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.

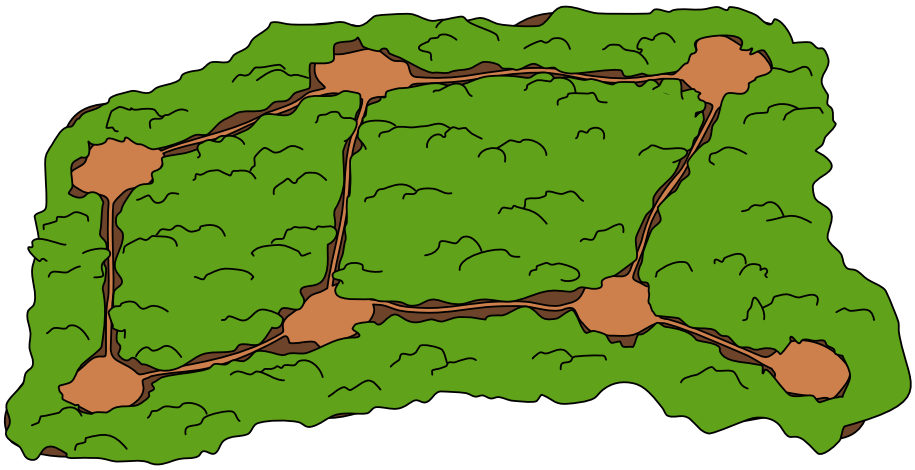
Mots clés et sites web

- 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)`

8. Gardes forestiers

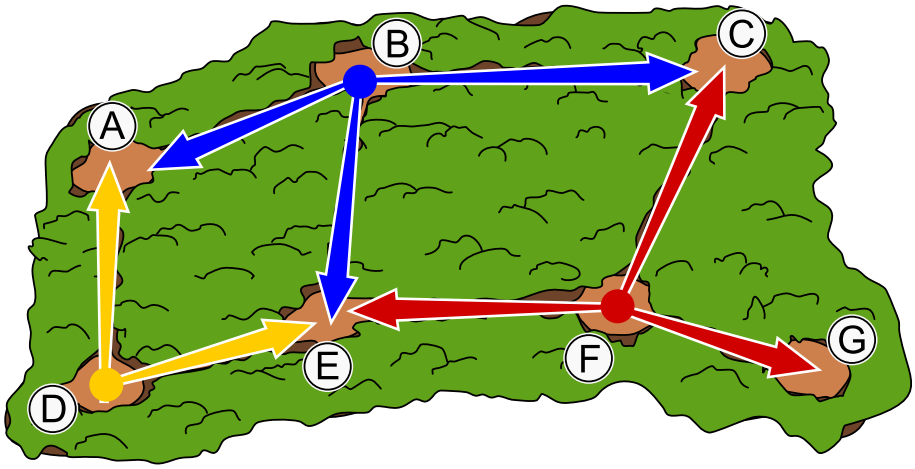
Les gardes forestiers veulent observer les animaux sur les sentiers de la forêt. Depuis chaque clairière, ils peuvent voir tous les sentiers allant de cette clairière à une clairière suivante. Il doit y avoir aussi peu de gardes forestiers que possible qui surveillent les sentiers.

Choisis des clairières afin que tous les sentiers puissent être surveillés par aussi peu de gardes forestiers que possible



Solution

L'image montre la solution minimale permettant aux gardes forestiers de surveiller tous les sentiers à partir de trois clairières.



Il y a huit sentiers à surveiller. Si seuls deux gardes forestiers suffisaient pour surveiller tous les sentiers, il devrait y avoir une clairière de laquelle partent au moins quatre sentiers, mais il n'y a pas de telle clairière dans cette forêt. Deux gardes forestiers ne suffisent donc pas.

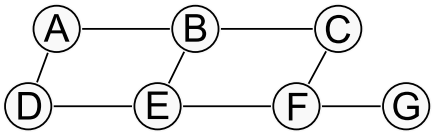
Il faut donc au minimum trois gardes forestiers pour assurer la surveillance de tous les sentiers. La solution présentée ici a donc le plus petit nombre de gardes forestiers possible. Il n'y a pas d'autre solution avec exactement trois gardes forestiers.

Nous pouvons déduire du nombre de sentiers et du fait qu'il n'y a aucune clairière de laquelle plus de trois sentiers partent que chaque garde forestier doit surveiller au moins deux sentiers qu'aucun autre garde forestier ne surveille.

Un garde forestier doit être placé sur la clairière F afin de surveiller le sentier entre les clairières F et G. Pour surveiller le sentier entre les clairières B et C, le deuxième garde forestier doit observer la forêt depuis la clairière B. Le dernier garde forestier doit être dans la clairière D pour que les deux derniers sentiers puissent être surveillés par un seul garde forestier. On obtient ainsi la solution donnée ici et il n'en existe pas d'autre.

C'est de l'informatique !

Les relations entre des choses (par exemple des sentiers entre des clairières) peuvent être représentées par ce que l'on appelle un *graphe*. Un graphe est constitué de *nœuds* (ici, les clairières) et d'*arêtes* (ici, les sentiers) représentées par des lignes reliant les nœuds. Le graphe de cet exercice ressemble à ceci :



Dans cet exercice du castor, il faut trouver le plus petit nombre de nœuds afin que chacune des arêtes commence ou finisse sur l'un des ces nœuds. Les informaticien·nes appelent un tel ensemble de nœuds une *couverture par sommets* ou une *transversale* (engl. *minimal vortex cover*). Dans la vie quotidienne, on recontre de tels problèmes de couverture par sommets lorsque l'on cherche les meilleurs emplacements pour des lampadaires ou pour des caméras de surveillance, par exemple.

Mots clés et sites web

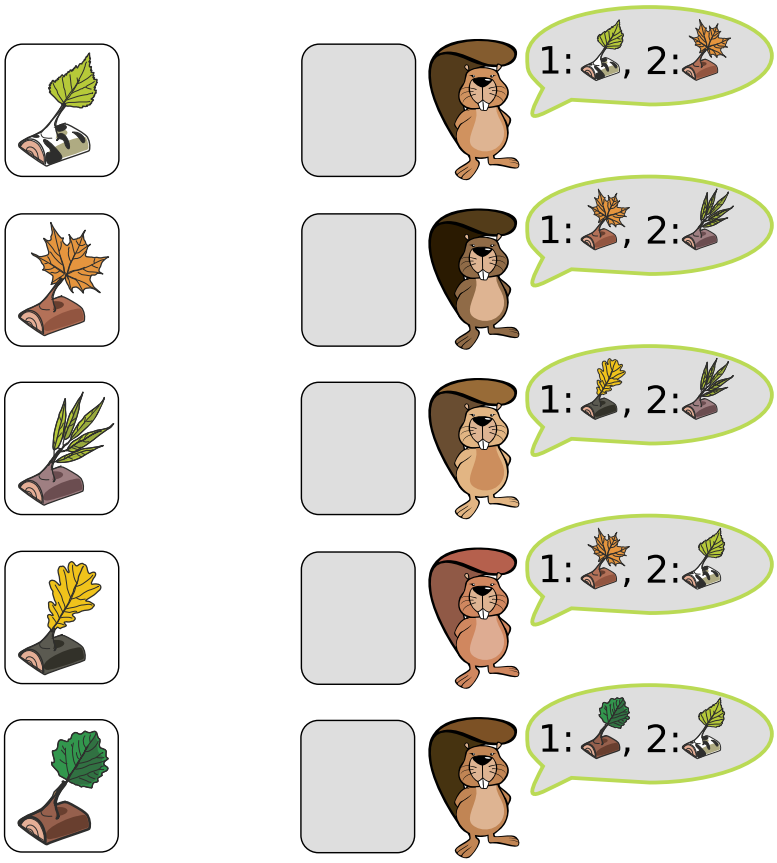
- Graphe : `https://fr.wikipedia.org/wiki/Graphe_(mathématiques_discrètes)`
- Couverture par sommets : `https://fr.wikipedia.org/wiki/Problème_de_couverture_par_sommets`

9. 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 :

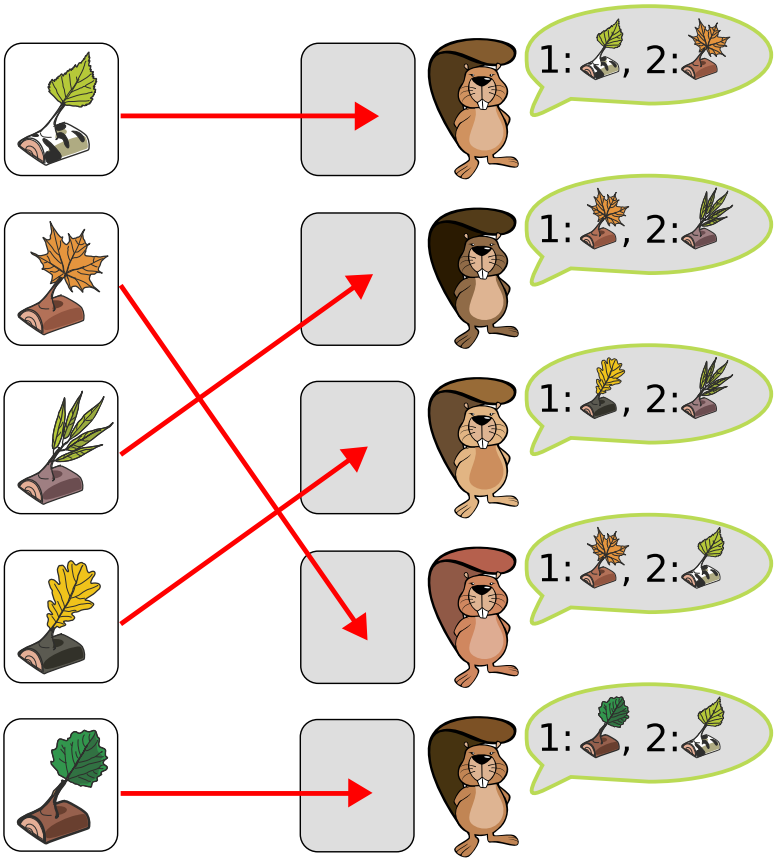
1. Le plus d'enfants possible doivent recevoir leur cadeau favori.
2. Les autres enfants doivent recevoir leur second choix.

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 !

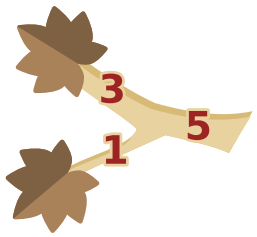
Mots clés et sites web

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

10. Sauvetage d'arbre

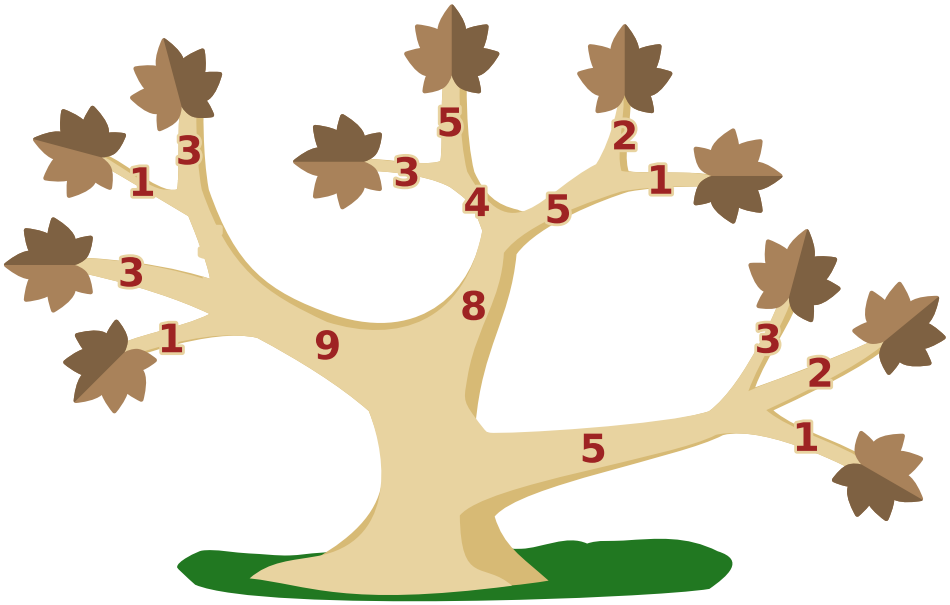
Un des arbres du jardin de Bruno est malade : toutes ses feuilles sont sèches. Bruno veut le sauver. Pour cela, il doit scier certaines branches de façon à enlever toutes les feuilles. De nouvelles branches avec de nouvelles feuilles peuvent ensuite pousser.

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



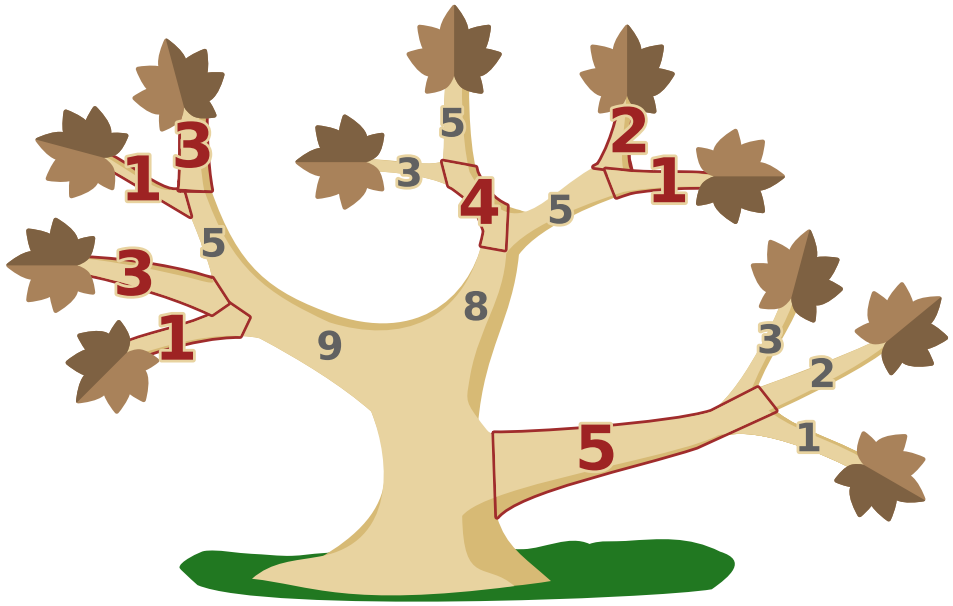
Pour enlever les deux feuilles, Bruno peut soit scier les deux branches portant les feuilles, soit scier la branche de laquelle partent les deux autres branches. Les chiffres sur chaque branche indiquent combien de temps est nécessaire pour la scier. Bruno va donc scier les deux branches avec des feuilles, 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`

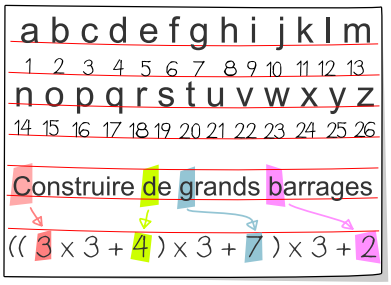
11. Bibliothèque

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.

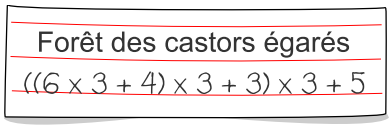
Le livre désiré donne 140. L'emplacement du livre est ainsi tout de suite clair.



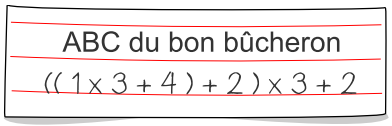
Susi écrit maintenant les calculs correspondant à ses livres préférés, mais elle a fait une erreur pour l'un d'entre eux.

Lequel ?

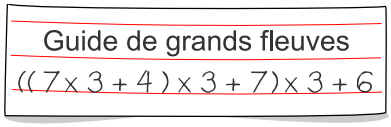
A)



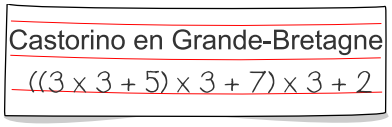
B)



C)

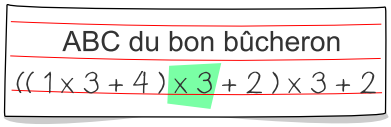


D)



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.

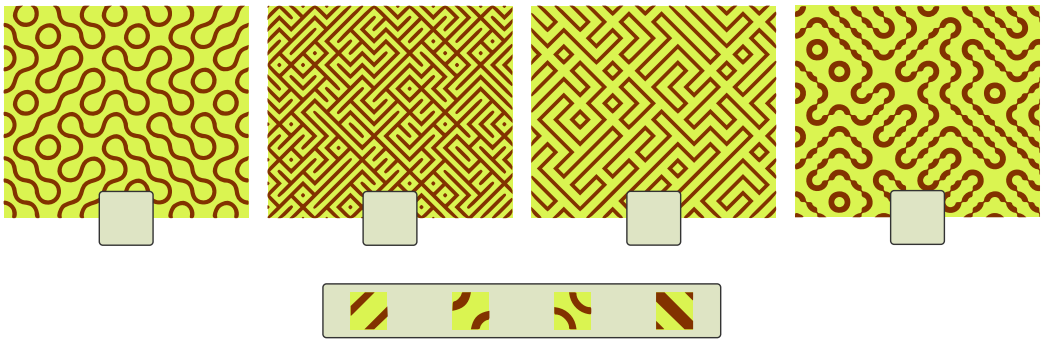
Mots clés et sites web

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

12. Pavage de Truchet

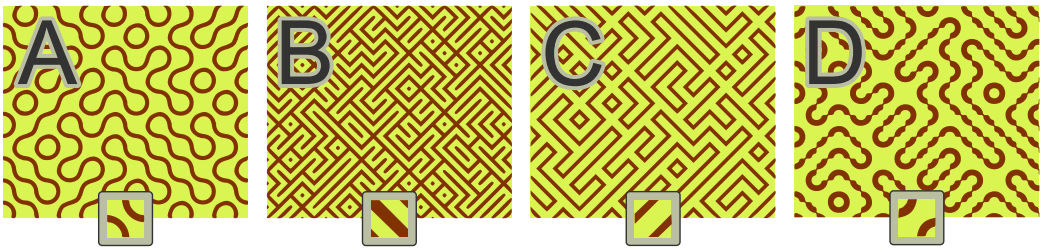
Les motifs suivants ont été créés en n'utilisant qu'un seul type de pavé. Les images des pavés individuels sont agrandies.

Assigne chaque pavé au motif correspondant.

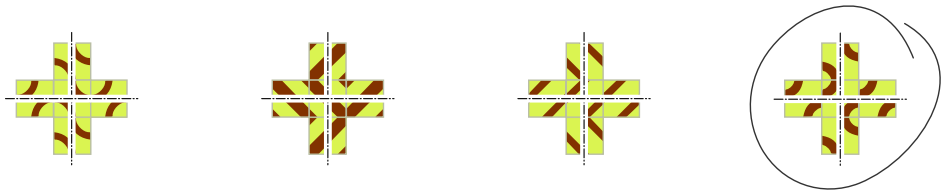


Solution

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 !

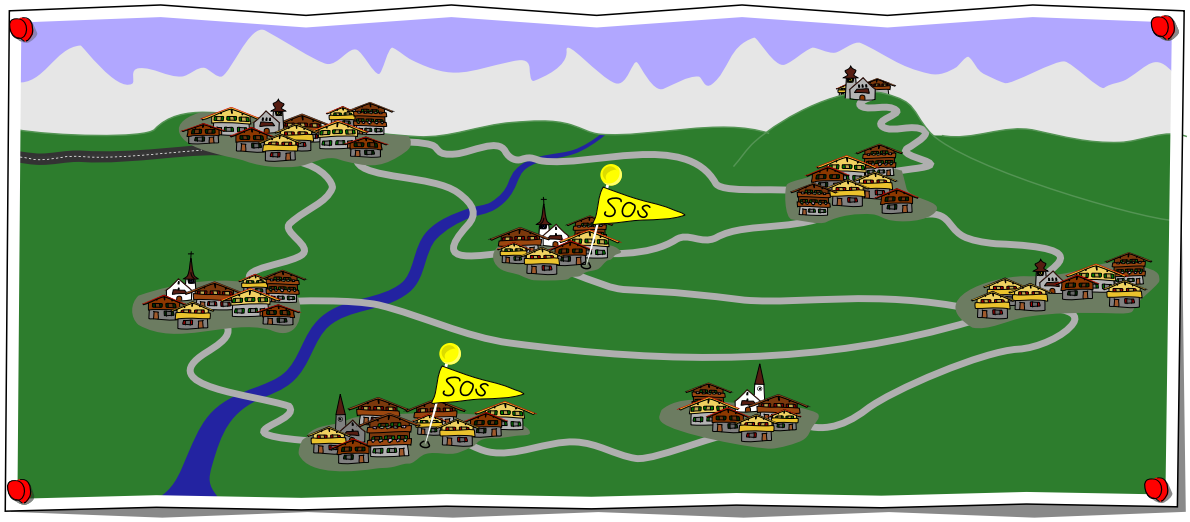
Ces pavés sont nommé d'après Sébastien Truchet (* 1657 ; † 1729), qui en a développé différentes variantes. Les pavés ayant quatre côtés pareils forment un sous-ensemble des pavés de Truchet (mais les pavés de Truchet ne doivent pas forcément avoir quatre côtés pareils, comme dans trois des motifs de 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`

13. Villages isolés

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

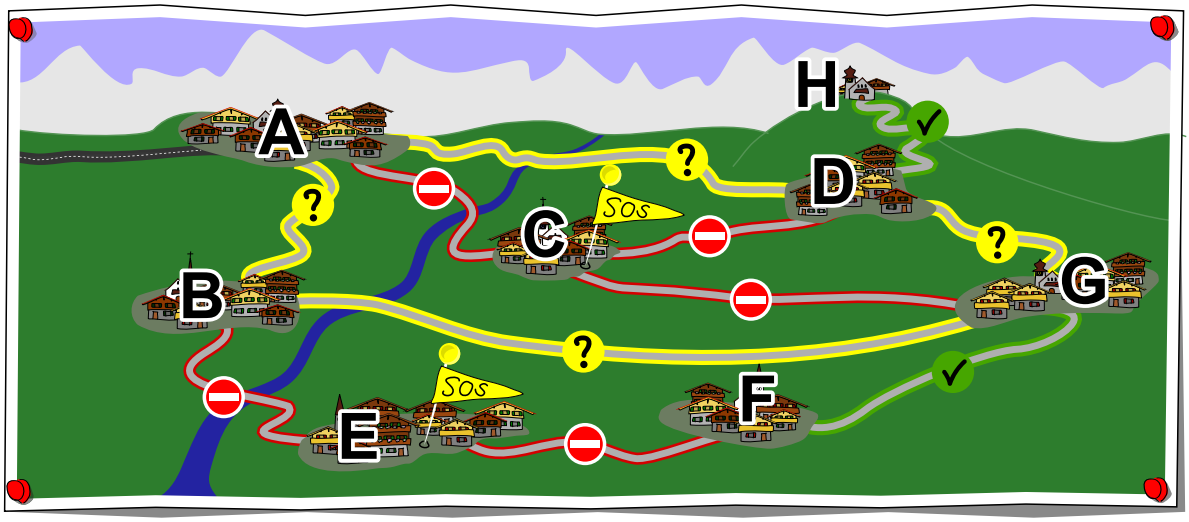


Après une tempête, plusieurs de ces villages ne sont plus du tout accessibles et le signalent avec un drapeau « SOS ». On peut en déduire que certaines des routes sont bloquées.

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*.

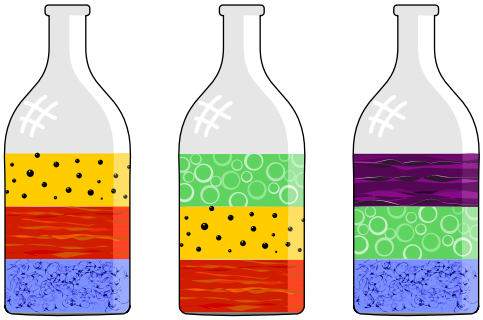
La correction de dysfonctionnements d'un système est une tâche que les informaticiens doivent souvent effectuer ; pas seulement dans les réseaux informatiques, mais aussi lors de développement de programmes. Pour corriger un problème, il faut identifier sa source exacte ; c'est un processus qui se fait 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

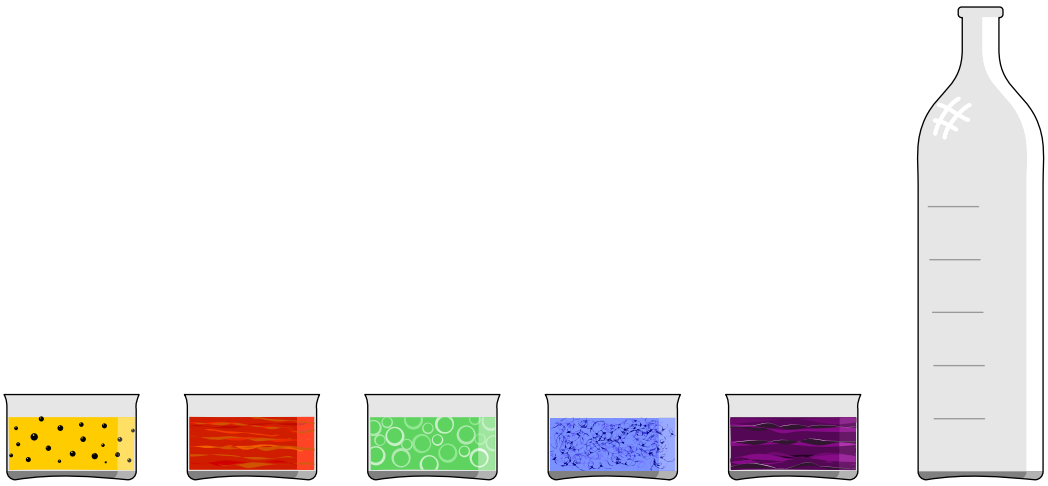
- Redondance : `https://fr.wikipedia.org/wiki/Redondance_(ingénierie)`
- Bug : `https://fr.wikipedia.org/wiki/Bug_(informatique)`

14. 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.

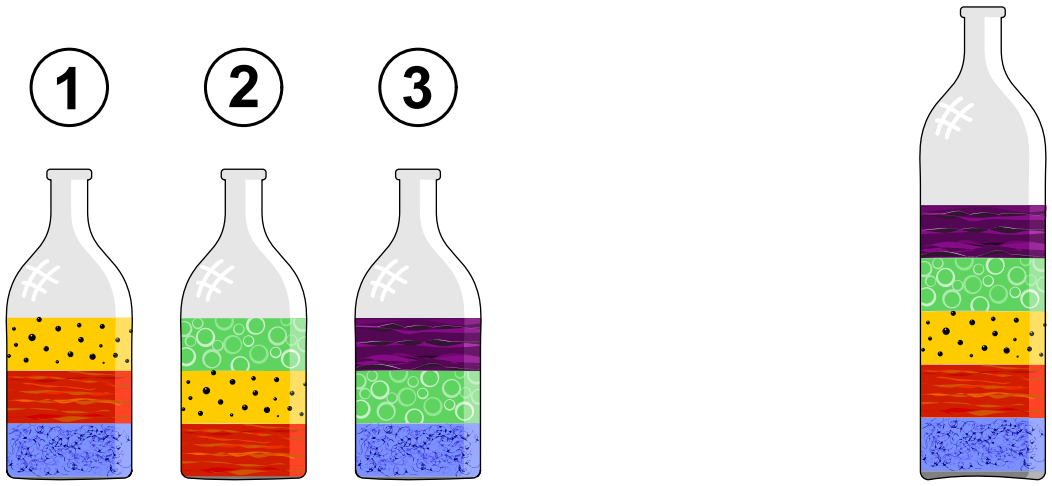


Arrange les cinq couches de liquides colorés dans la bouteille dans leur ordre final.



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

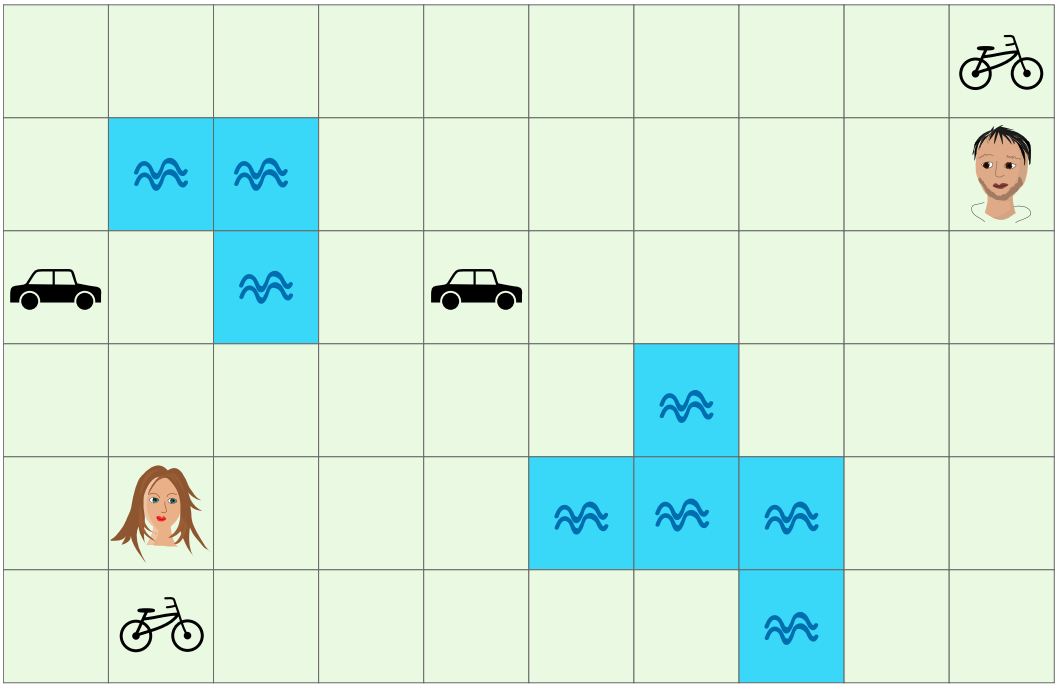
- Relation d'ordre : `https://fr.wikipedia.org/wiki/Relation_d'ordre`
- Tri topologique : `https://fr.wikipedia.org/wiki/Tri_topologique`

15. 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.

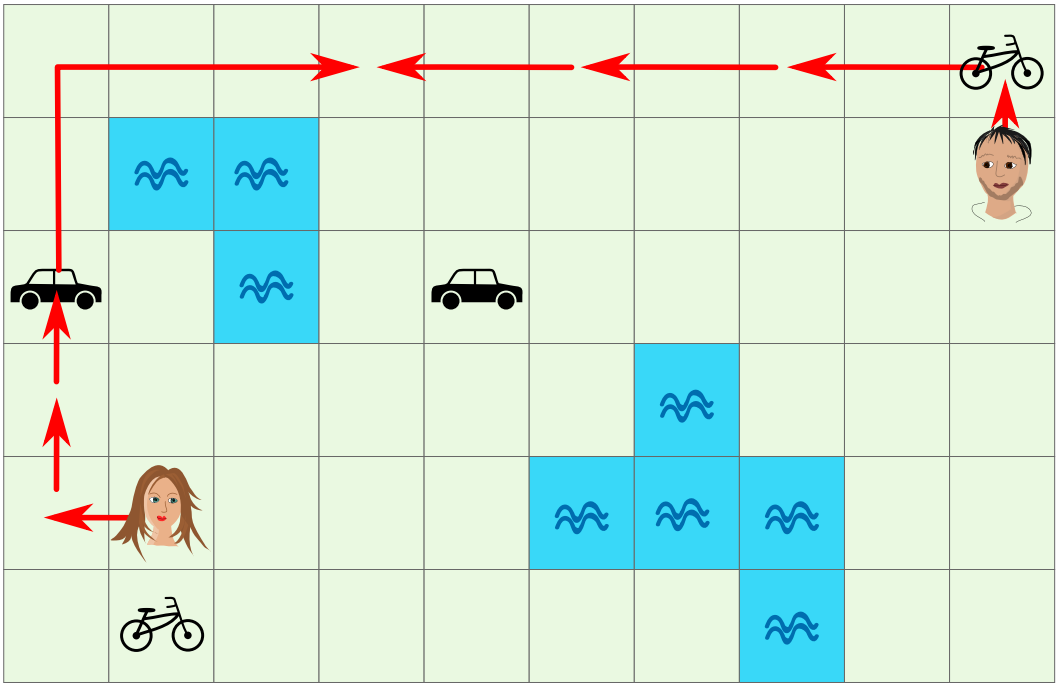


De combien de minutes au minimum les deux amis ont-ils besoin pour se retrouver sur la même case ?

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.



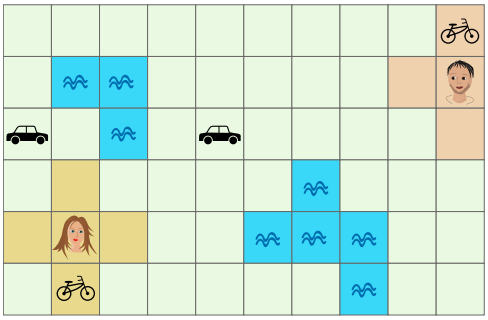
Maintenant, il faut encore prouver qu'ils ne peuvent pas se retrouver en 3 minutes. Les deux amis sont à 11 cases de distance l'un de l'autre. En trois minutes, s'ils se déplacent à pied, il ne peuvent se rapprocher que de 6 cases en tout. Si l'un des deux a atteint un vélo et que l'autre va à pied, 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.

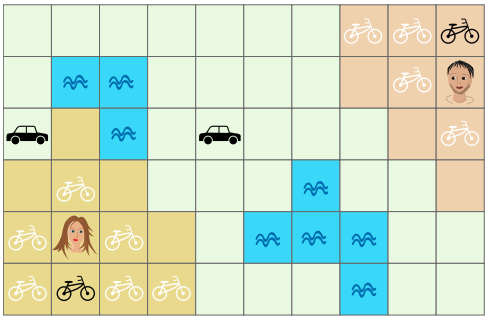
C'est de l'informatique !

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 ?

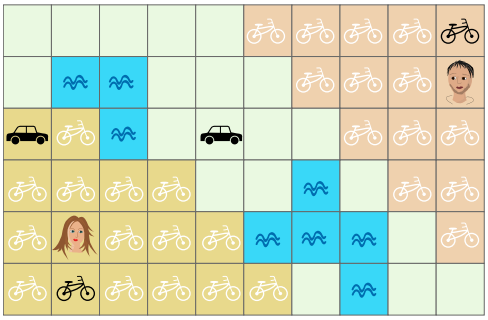
Les programmes informatiques qui ont été développés pour résoudre ce genre de problème travaillent souvent avec une méthode appelée *parcours en largeur*. Dans cet exercice, le parcours en largeur se passe comme cela :



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

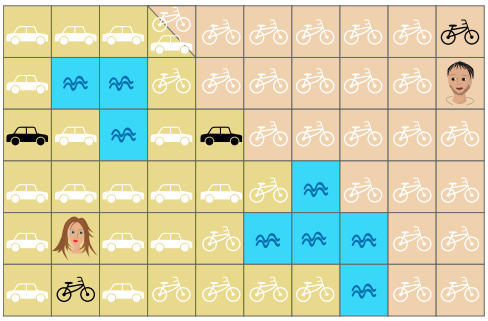


2. Sélectionne toutes les cases qui peuvent être atteinte en (maximum) 1 minute depuis les cases sélectionnées à l'étape 1. Note quel moyen de transport a été utilisé.



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`

A. Auteur·e·s des exercices

Sarah Atkins
Michael Barot
Liam Baumann
Wilfried Baumann
Javier Bilbao
Sarah Chan
Kris Coolsaet
Valentina Dagienė
Christian Datzko
Susanne Datzko
Margarita Flores-Sicich
Fabian Frei
Gerald Futschek
Thomas Galler
Yasemin Gülbahar
Ezgi Arzu Güneş
Juraj Hromkovič
Alisher Ikramov
YongJu Jeon
Soojin Jun
Filiz Kalelioğlu
Dong Yoon Kim
Jihye Kim
Vaidotas Kinčius
Regula Lacher
Taina Lehtimäki
Marielle Léonard
Tom Naughton
Mochammad Irfan Noviana
Elsa Pellet
Jean-Philippe Pellet
Zsuzsa Pluhár
Wolfgang Pohl
Peter Rossmanith
Eljakim Schrijvers
Tomas Šiaulys
Timur Sitdikov
Bernadette Spieler
Ahto Truu
Florentina Voboril
Michael Weigend
Kyra Willekes
Hongjin Yeh
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.