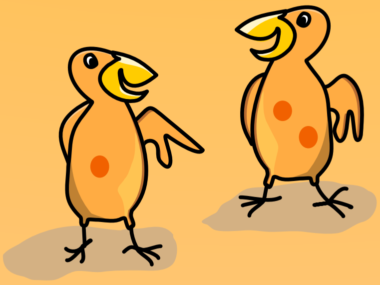
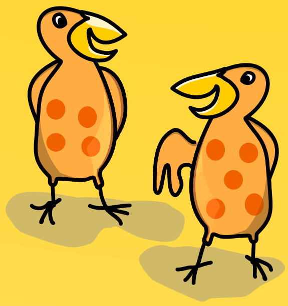
INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Exercices et solutions 2021

Années HarmoS 7/8

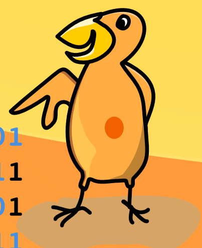


https://www.castor-informatique.ch/

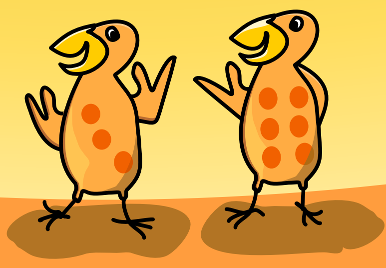


Éditeurs :

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



www.svia-ssie-ssii.ch
schweizerischervereinfürinformatikind erausbildung//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.

© Castor Informatique 2021, SSIE

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.

Les exercices sont protégés par une licence Creative Commons Paternité – Pas d'Utilisation Commerciale – Partage dans les Mêmes Conditions 4.0 International. Les auteur·e·s sont cité·e·s en p. 42.

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.

© Castor Informatique 2021, SSIE

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/`

© Castor Informatique 2021, SSIE

Table des matières

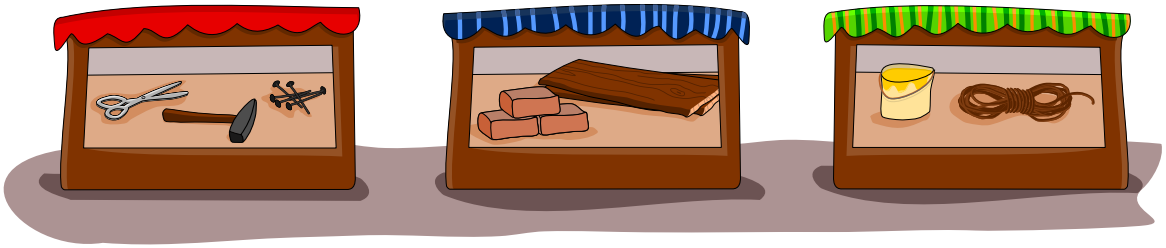
© Castor Informatique 2021, SSIE

1. Construction de pont

Bella aimerait construire un pont par-dessus un ruisseau. Elle a besoin d'un marteau, de clous, de planches et d'une corde. Elle trouve un marteau et une corde à la cave.

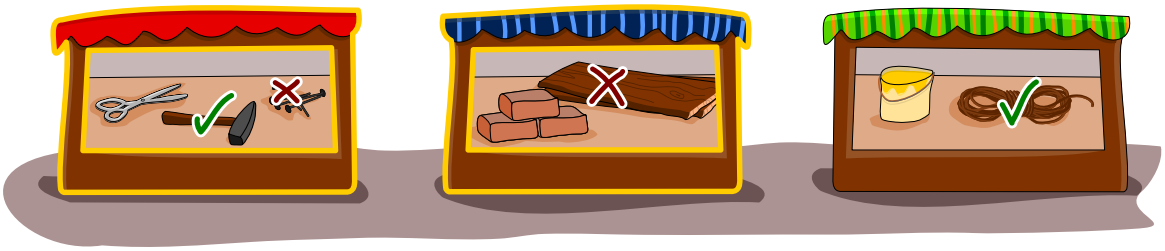
Elle doit acheter les autres objets. En bas, tu vois trois magasins et ce qu'ils vendent.

Dans quels magasins Bella peut-elle acheter les autres objets ?



Solution

La bonne réponse est :



C'est de l'informatique !

Dans cet exercice, les magasins vendent en tout sept ojects différents : des ciseaux, un marteau, des clous, des briques, des planches, une corde et un seau. C'est un grand ensemble d'objets ! Les objets que Bella doit acheter sont une *partie de cet ensemble*. On peut le représenter de la façon suivante : on montre tous les objets de l'ensemble complet et note pour chaque objet s'il appartient au sous-ensemble que Bella doit acheter ou pas :

De la même manière, on peut représenter quels objets sont vendus dans les magasins, par exemple dans le magasin de gauche :

On voit ainsi au premier coup d'œil ce que Bella peut acheter dans le magasin de gauche : Les clous sont marqué d'une coche verte dans le sous-ensemble de Bella et dans le sous-ensemble du magasin.

Les programmes informatiques doivent aussi souvent comparer des ensembles. Pour chaque objet pouvant être présent, on a besoin d'un *bit*. Un ordinateur peut enregistrer une de deux valeurs dans un bit, comme par exemple « oui » ou « non ». Dans notre cas, on enregistre si un objet appartient à un ensemble (« oui ») ou pas (« non »). Un programme peut ensuite comparer deux ensembles de la façon suivante : il vérifie si le bit correspondant à un objet vaut « oui » dans un ensemble et vaut aussi « oui » dans l'autre ensemble. Une telle comparaison de deux bits est très rapide pour un ordinateur. C'est pour cela qu'en informatique, les ensembles sont souvent décrits en utilisant des bits.

Mots clés et sites web

- Ensemble : `https://fr.wikipedia.org/wiki/Ensemble_(informatique)`
- Bits : `https://fr.wikipedia.org/wiki/Bit`
- Tableau de bits : `https://fr.wikipedia.org/wiki/Tableau_de_bits`

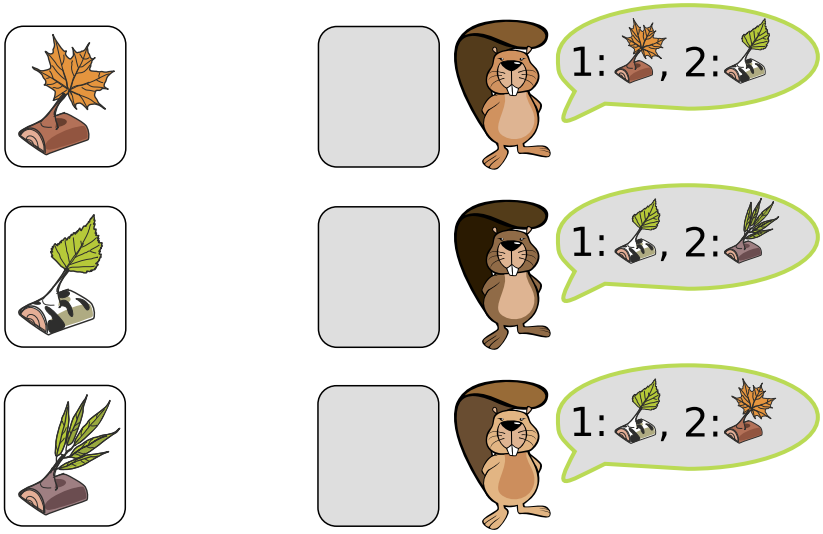
 © Castor Informatique 2021, SSIE

2. Cadeau favori

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

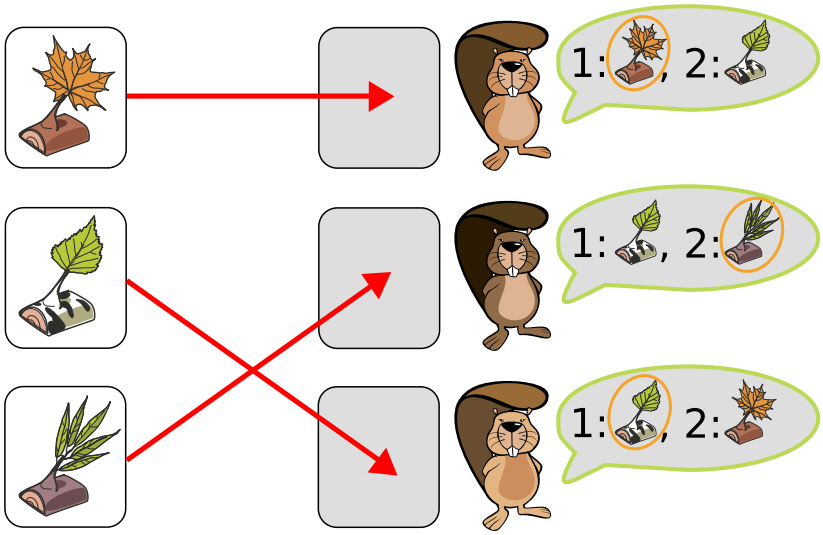
1. Le plus d'enfants possible doivent recevoir leur cadeau favori.
2. Les autres enfants doivent recevoir leur second choix.

Donne les bons cadeaux aux enfants.



Solution

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



Seul le deuxième castor désire le troisième cadeau, c'est donc lui qui doit le recevoir. Sinon, un autre castor recevrait un cadeau qui n'est ni son cadeau favori, ni son deuxième choix. La distribution des deux autres cadeaux est claire : chaque castor reçoit 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). Intéressant ? L'informatique est une branche très variée !

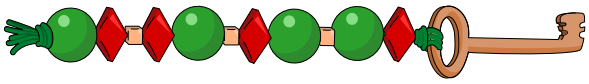
Mots clés et sites web

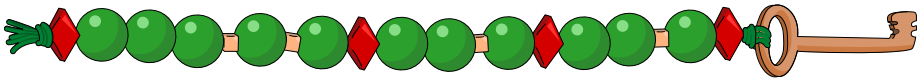
- Problème d'affectation : `https://fr.wikipedia.org/wiki/Problème_d'affectation`
- Couplage : `https://fr.wikipedia.org/wiki/Couplage_(théorie_des_graphes)`

 © Castor Informatique 2021, SSIE

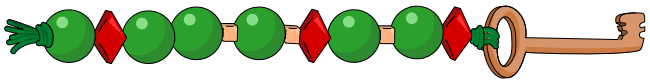
3. Porte-clés

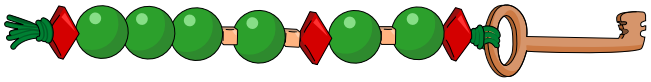
ANNA, BELLA et LENA font des porte-clés avec leur nom. Elles utilisent deux sortes de perles pour les lettres :  et  . Différentes lettres sont séparées par la perle  .

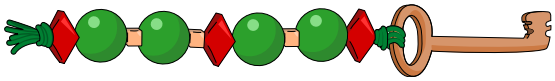
ANNA 

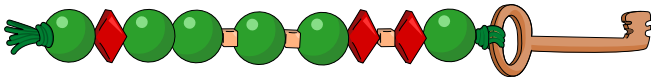
BELLA 

Quel porte-clés LENA a-t-elle fait ?

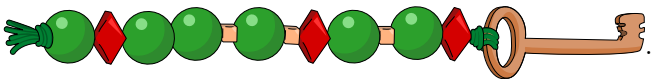
A) 

B) 

C) 

D) 

Solution

La bonne réponse est A)  .

Le mot LENA commence par un L. La lettre L est la troisième et quatrième lettre du mot BELLA, on peut donc reconnaître la suite de perles utilisée pour la lettre L dans BELLA :  . Seules les réponses A et D) commencent par cette lettre et peuvent être la solution. La deuxième lettre de Lena, le E, est aussi la deuxième lettre de BELLA, on peut trouve donc la perle  dans les deux porte-clés. Les réponses A) et D) ont les deux la perle  comme deuxième lettre, les deux peuvent donc toujours être la solution. On veut ensuite trouver quelles perles sont utilisées pour la lettre N. On trouve la suite de perles  dans le porte-clé d'ANNA. On ne trouve ces perles pour la lettre suivante que dans la réponse A).

Une autre possibilité pour trouver quelles perles sont utilisées pour le porte-clés de LENA est de faire un tableau montrant les perles correspondant aux lettres connues. Le porte-clés d'ANNA nous donne la suite de perles  pour la lettre A et la suite de perles  pour la lettre N. Le porte-clés de BELLA nous donne la suite de perles pour B :  , pour E :  et pour L :  .

Lettres	Perles
A	
N	
B	
E	
L	

On peut donc faire le porte-clés de LENA avec les suites de perles  ,  ,  et  si on sépare les différentes lettres avec la perle  . On obtient le porte-clés  , ce qui correspond à la réponse A). Si l'on décode les autres porte-clés à l'aide du tableau, on trouve BENA pour B), NENA pour C) et LEAN pour D).

C'est de l'informatique !

L'information est *encodée* pour pouvoir transférer des messages sous certaines conditions ou pour transmettre des informations de manière secrète (*chiffrée*). Le codage de cet exercice est basé sur l'alphabet Morse. le point • de l'alphabet Morse est représenté par la perle ronde  et le trait – par  . La lettre A s'écrit • – en alphabet Morse, donc  en perles. Pour encoder des textes quelconques, il faut un code pour chaque lettre de l'alphabet.

L'alphabet Morse a été créé au XIX^e siècle. Samuel Morse inventa un télégraphe électromagnétique simple en 1837. Le code utilisé à l'époque ne comprenait que les dix chiffres de 0 à 9 ; les chiffres transmis devaient être traduits en lettres puis en mots à l'aide d'une tableau. Albert Lewis Vail, un collègue de Morse, developpa dès 1838 le premier code comprenant aussi des lettres. Le code fut développé pour pouvoir transmettre des textes par voie acoustique, optique ou électromagnétique à

 © Castor Informatique 2021, SSIE

l'aide d'un télégraphe. Un point représente une courte durée de transmission et un trait une durée trois fois plus longue. Un pause sépare les lettres, et une pause plus longue sépare les mots. L'alphabet Morse est encore utilisé aujourd'hui pour le signal SOS. Un SOS en Morse • • • — — — • • • (3x court, 3x long, 3x court) peut être transmis facilement en criant, tapant ou avec une lampe de poche.

Dans le traitement de données informatiques, les caractères sont encodés par des valeurs numériques pour être transmis ou enregistrés.

Mots clés et sites web

- Codage des caractères : `https://fr.wikipedia.org/wiki/Codage_des_caractères`
- Alphabet Morse : `https://fr.wikipedia.org/wiki/Code_Morse_international`

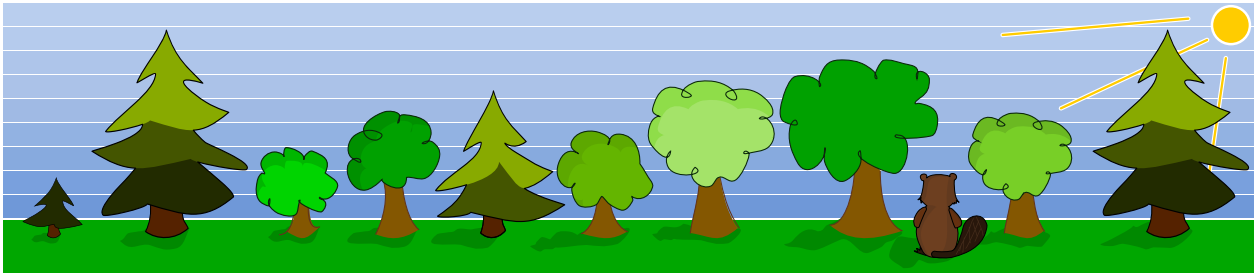
© Castor Informatique 2021, SSIE

4. Timber !

Un castor aimerait construire un barrage. Afin de toujours abattre les bons arbres, il s'est fixé deux règles. Il n'abat un arbre que si :

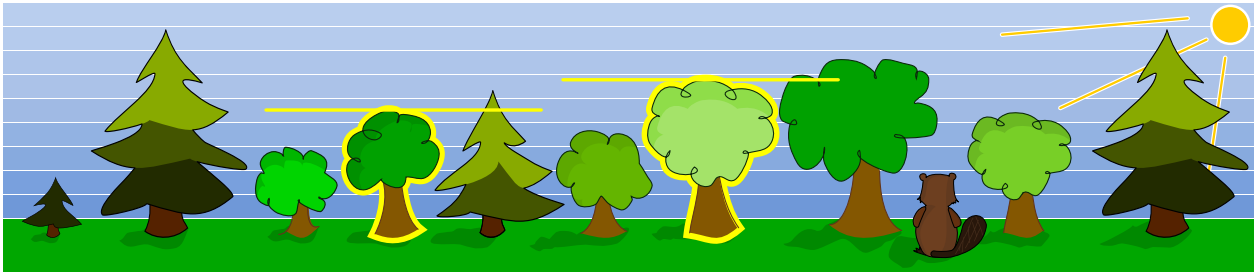
- un arbre plus petit pousse directement à sa gauche et
- un arbre plus grand pousse directement à sa droite.

Quels arbres le castor va-t-il abattre ?



Solution

Seuls les arbres à la quatrième et septième positions remplissent les conditions données : il y a un arbre plus petit directement à gauche ET un arbre plus grand directement à droite.



C'est de l'informatique !

En informatique, il faut souvent résoudre des problèmes qui sont spécifiés par une série de *contraintes* logiques. La tâche consiste à trouver une solution qui respecte toutes les contraintes. Des problème plus complexes que celui-ci peuvent être résolus en utilisant des *opérateurs logiques* pour combiner les contraintes. La conjonction ($\wedge$ ou encore opérateur ET) donne par exemple le résultat « vrai » pour l'expression A $\wedge$ B lorsque les deux contraintes A et B sont vraies. Dans cet exercice, ce serait donc : « l'arbre de gauche est plus petit » $\wedge$ « l'arbre de droite est plus grand ». On retrouve ce principe fondamental dans presque tous les domaines de l'informatique, par exemple dans beaucoup d'algorithmes de tri comme le *tri à bulles* lors duquel la satisfaction aux contraintes de plusieurs objets d'une liste est évaluée avant de les déplacer, si nécessaire, à une autre position. Ce procédé est répété jusqu'à ce que la liste soit complètement triée.

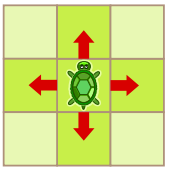
Mots clés et sites web

- Pensée algorithmique (*algorithmic thinking*)
- Opérateur logique : `https://fr.wikipedia.org/wiki/Connecteur_logique`
- Tri à bulles : `https://fr.wikipedia.org/wiki/Tri_à_bulles`
- Tri : `https://sorting.at/`
- Problème de satisfaction de contraintes : `https://fr.wikipedia.org/wiki/Problème_de_satisfaction_de_contraintes`

 © Castor Informatique 2021, SSIE

5. 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.

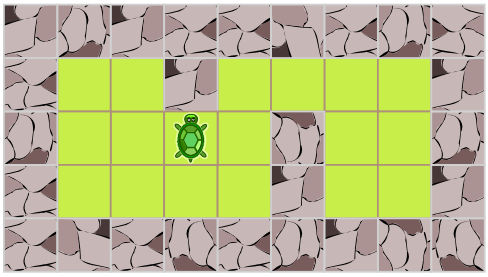


La tortue doit complètement brouter les jardins. elle commence sur la case sur laquelle elle est sur l'image. À la fin, elle doit avoir passé exactement une fois sur chaque case d'herbe.

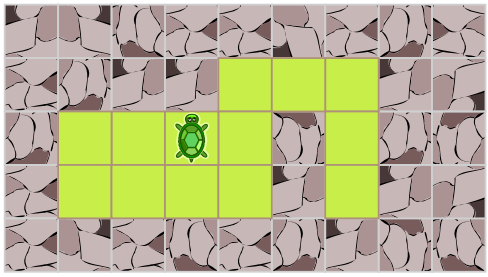
Il y a malheureusement un jardin qu'elle ne peut pas brouter complètement de cette façon.

De quel jardin s'agit-il ?

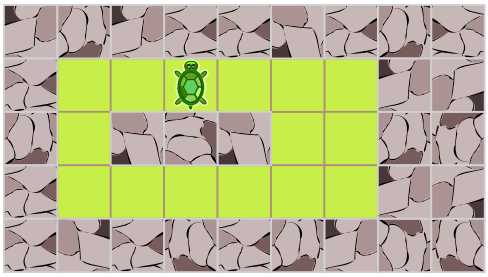
A)



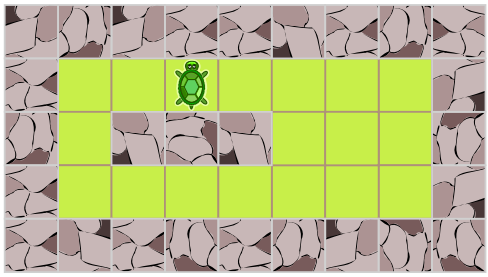
B)



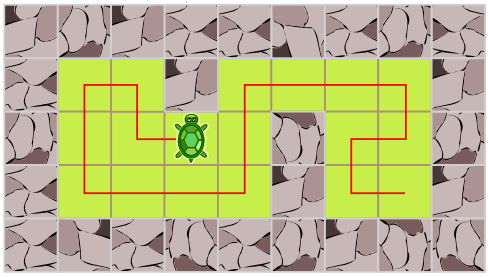
C)

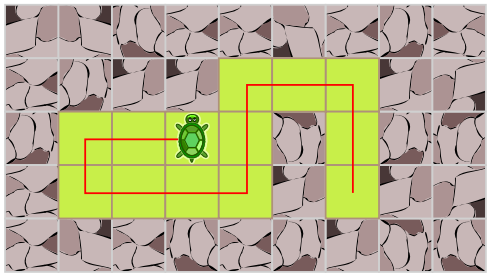


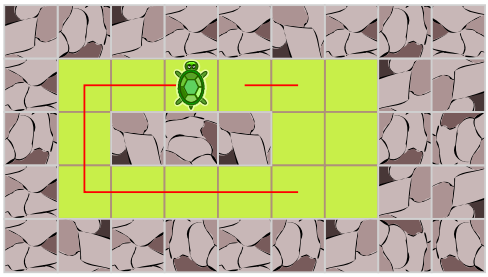
D)

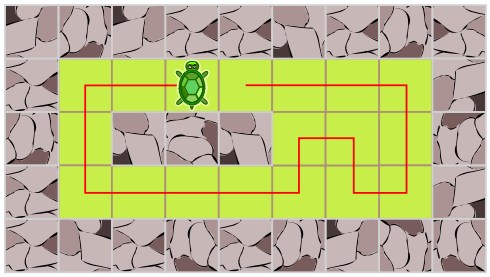


Solution

A) 

B) 

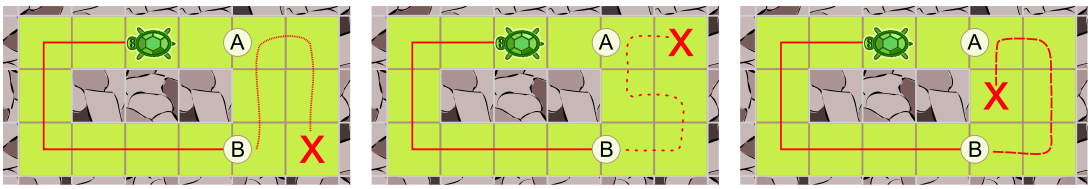
C) 

D) 

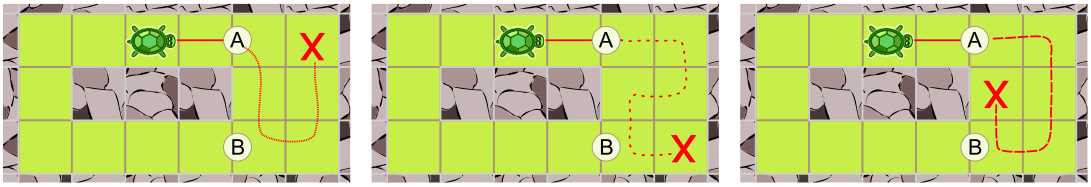
La tortue peut brouter les jardins A, B et D complètement.

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.

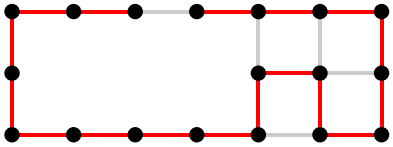


C'est de l'informatique !

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*

 © Castor Informatique 2021, SSIE

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`

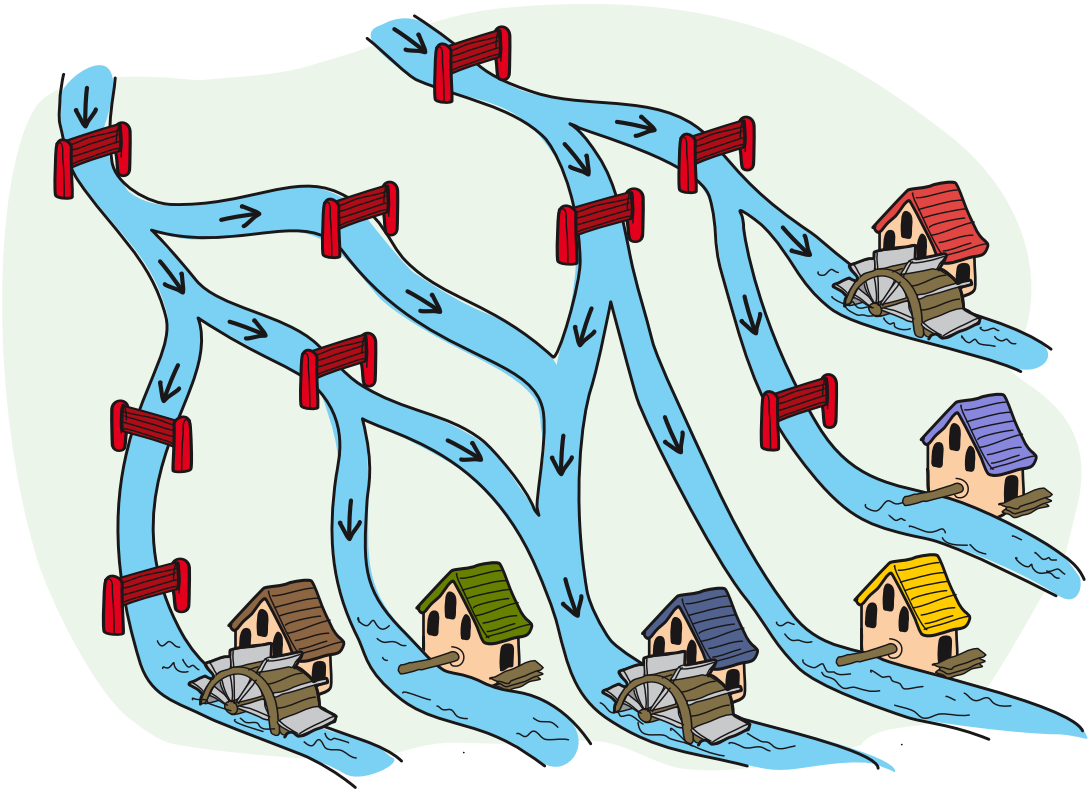
© Castor Informatique 2021, SSIE

6. 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.

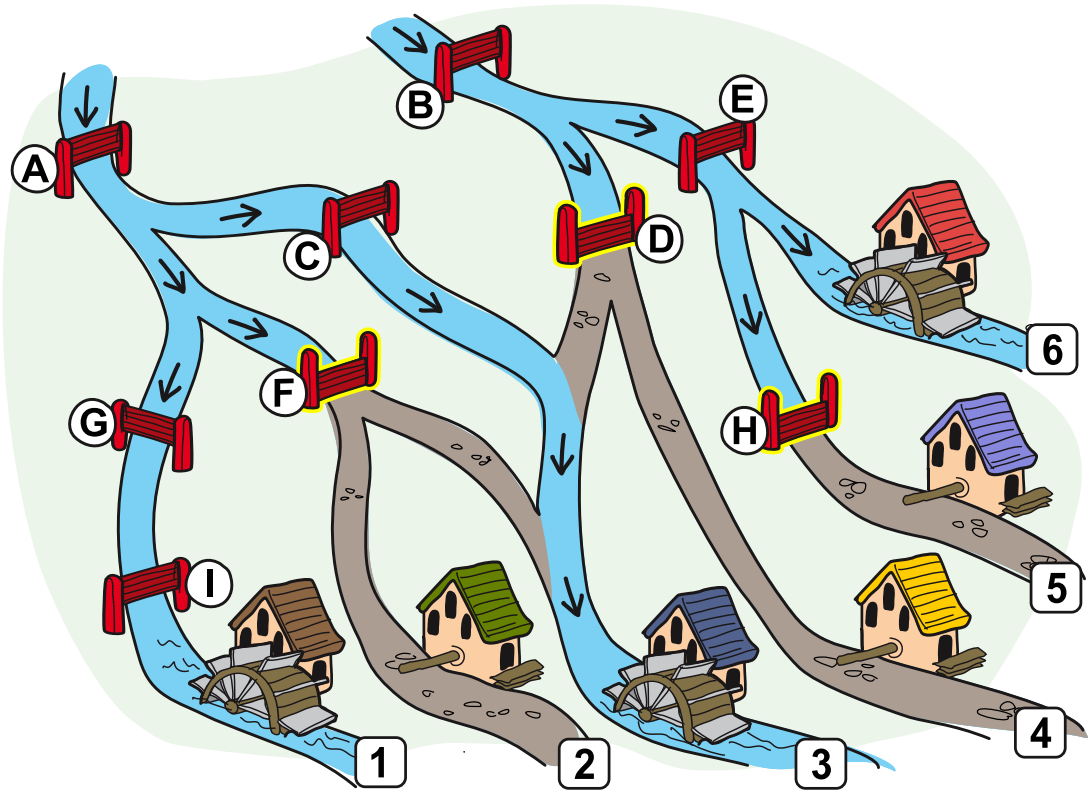
L'eau ne peut couler que vers le bas. Un clapet fermé empêche l'eau de couler.

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

 © Castor Informatique 2021, SSIE

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.

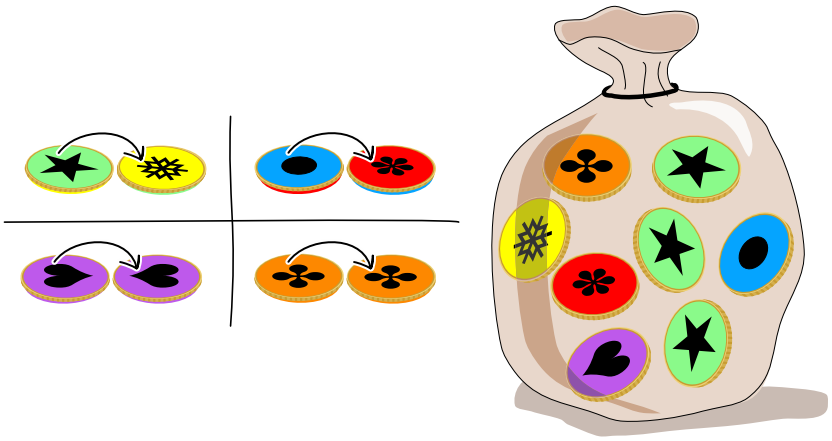
Mots clés et sites web

- 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)`

© Castor Informatique 2021, SSIE

7. Sac de pièces

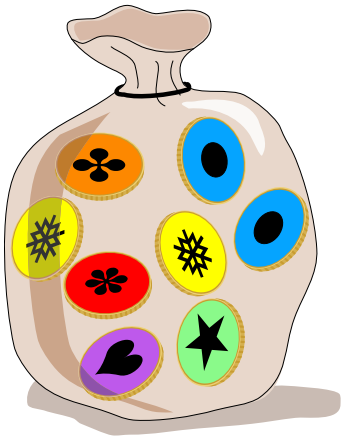
Il existe quatre sortes de pièces de monnaie différentes dans le pays d'Émile. Tu peux voir ici les deux côtés de ces pièces ainsi que le sac d'Émile avec ses pièces.



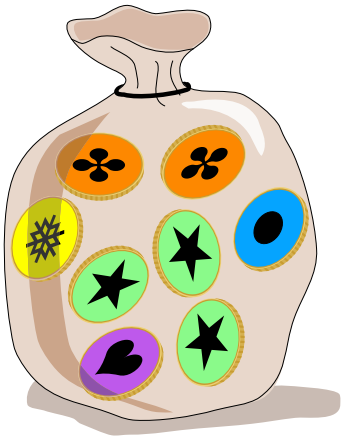
Émile secoue son sac de pièces.

Quel sac est celui d'Émile ?

A)



B)



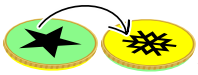
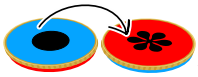
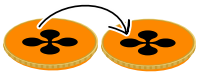
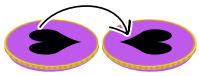
C)

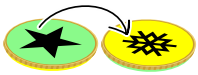
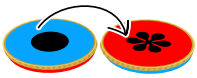
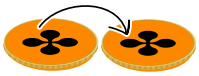
D)

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

 © Castor Informatique 2021, SSIE

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

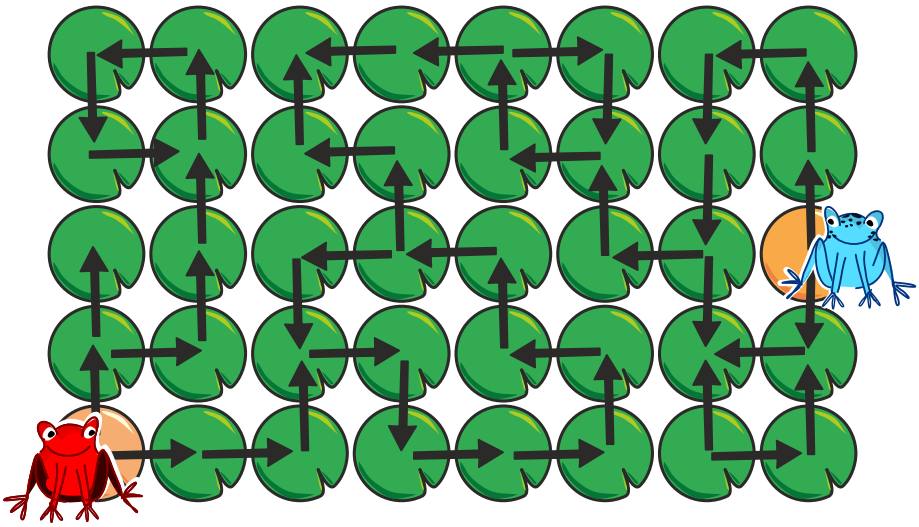
- Multiensemble : `https://fr.wikipedia.org/wiki/Multiensemble`
- Informations non structurées :
  `https://fr.wikipedia.org/wiki/Informations_non_structurées`

© Castor Informatique 2021, SSIE

8. 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.

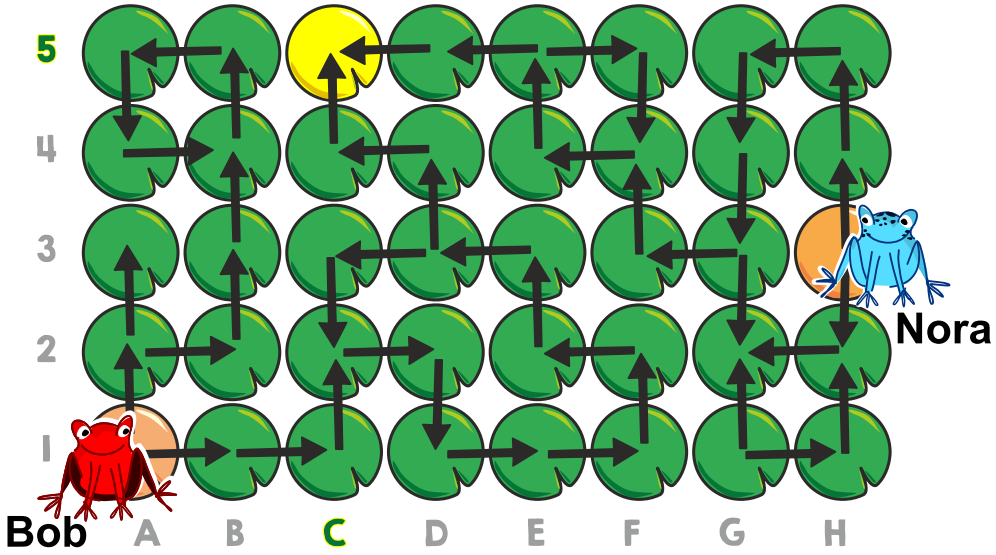
Sur quelle feuille les deux grenouilles peuvent-elles se rencontrer ?



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

Solution

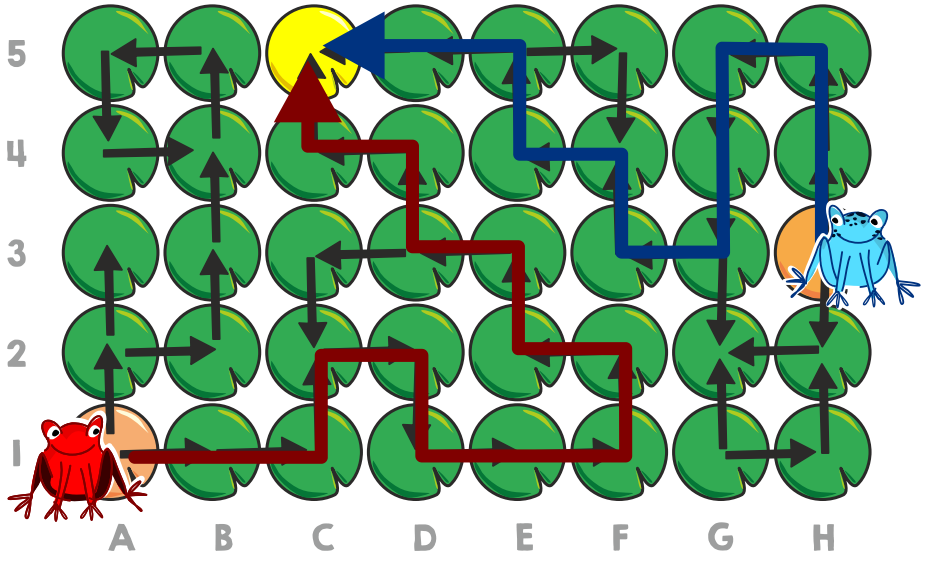
Les grenouilles ne peuvent se rencontrer que sur la feuille C5.



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.

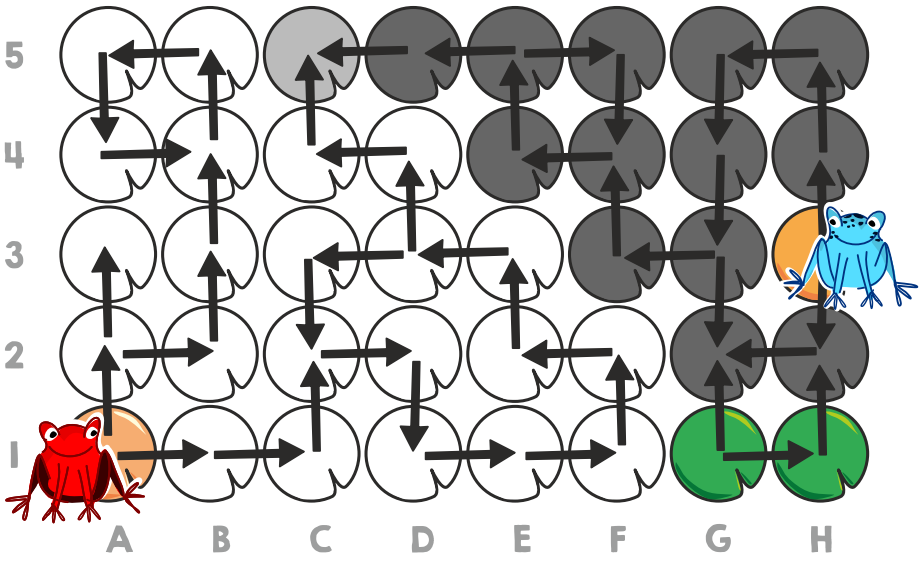
Nora, la grenouille bleue, a aussi le choix entre deux directions depuis sa feuille de départ. Si elle va vers le « bas », elle arrive dans le cul de sac sur G2. Si elle va vers le « haut », elle peut sauter jusqu'à G3. Depuis là, elle peut soit arriver dans le cul de sac sur G2 ou partir à « gauche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



 © Castor Informatique 2021, SSIE

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`

© Castor Informatique 2021, SSIE

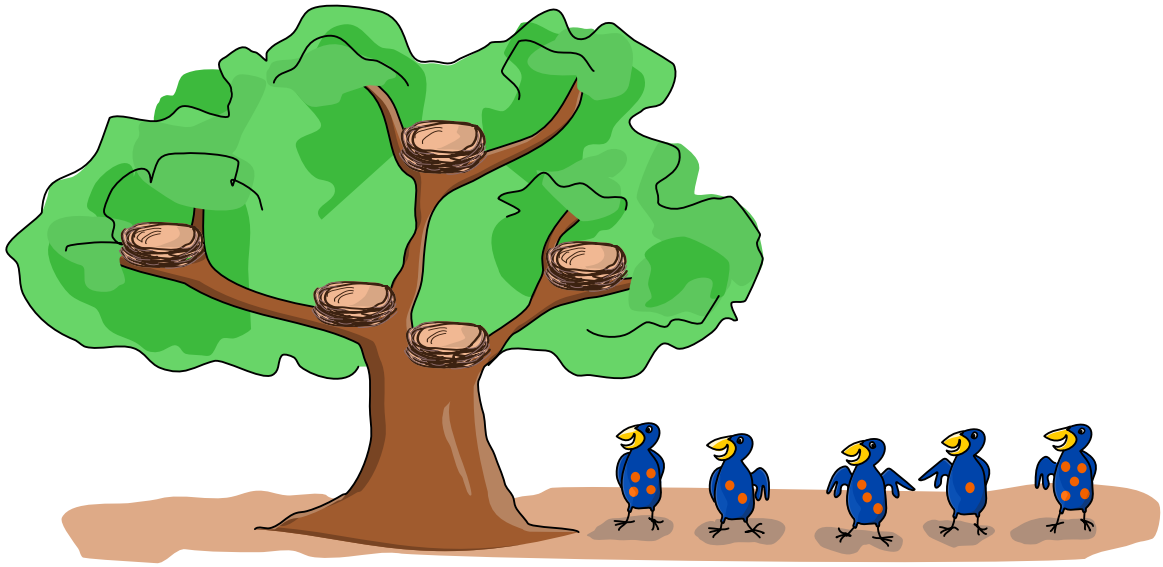
9. Emménagement

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.

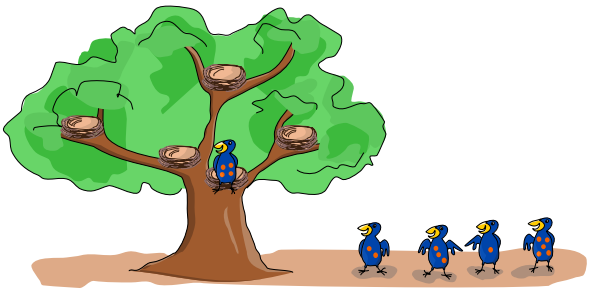
Où se trouvent les tachetés à la fin ? Place chaque tacheté dans le bon nid.



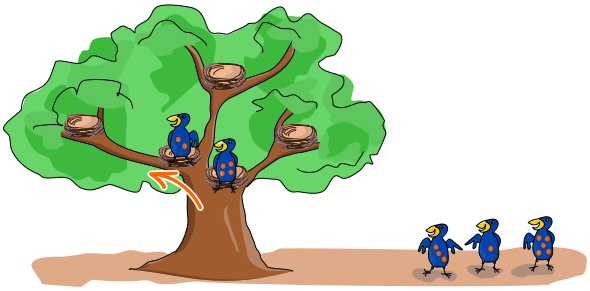
Solution

On obtient la bonne solution de la manière suivante :

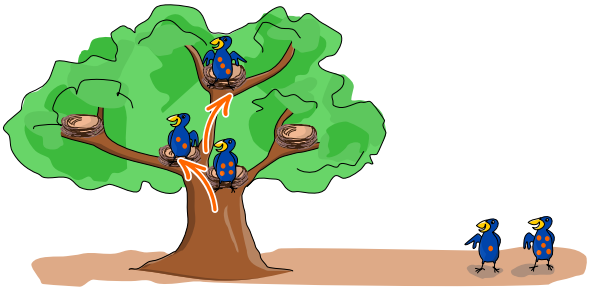
Le premier tacheté, celui à 4 points, arrive dans le nid du bas et y emménage.



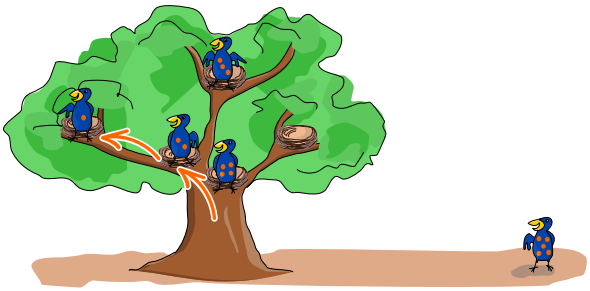
Le deuxième tacheté a 2 points. Le nid du bas est maintenant occupé par le premier tacheté à 4 points. Comme 4 est plus grand que 2, le deuxième tacheté continue à grimper vers la gauche et emménage dans le premier nid vide.



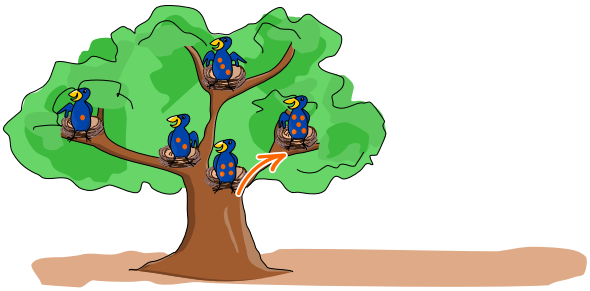
Le troisième tacheté a 3 points. Il grimpe vers la gauche après le nid du bas dans lequel est le tacheté à 4 points, car 4 est plus grand que 3. Le tacheté à 2 points est dans le nid suivant. Comme 3 est plus grand que 2, le troisième tacheté continue de grimper vers la droite. Il emménage dans le prochain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

 © Castor Informatique 2021, SSIE

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

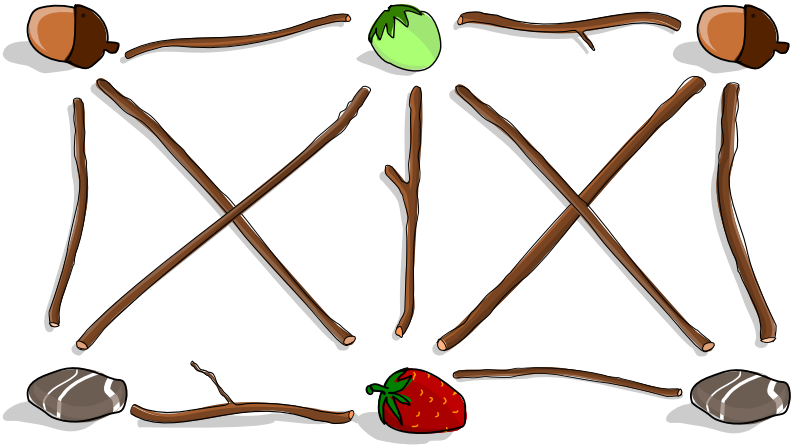
- Arbre binaire de recherche :
  `https://fr.wikipedia.org/wiki/Arbre_binaire_de_recherche`
- Structure de données : `https://fr.wikipedia.org/wiki/Structure_de_données`

© Castor Informatique 2021, SSIE

10. 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.

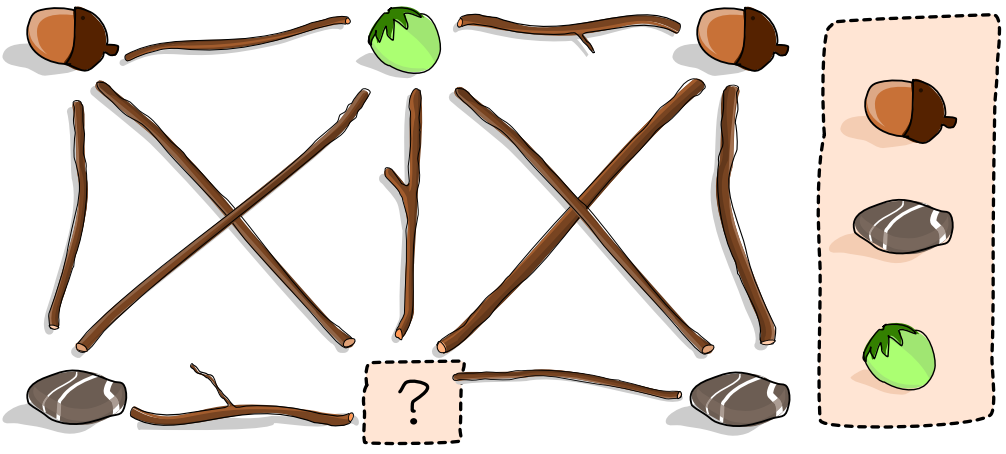
Ensuite, Anja dispose des branches entre ces objets. Elle suit pour cela la règle suivante : une branche ne doit pas se trouver entre deux objets pareils, par exemple entre deux glands. Voici l'œuvre d'art terminée :



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

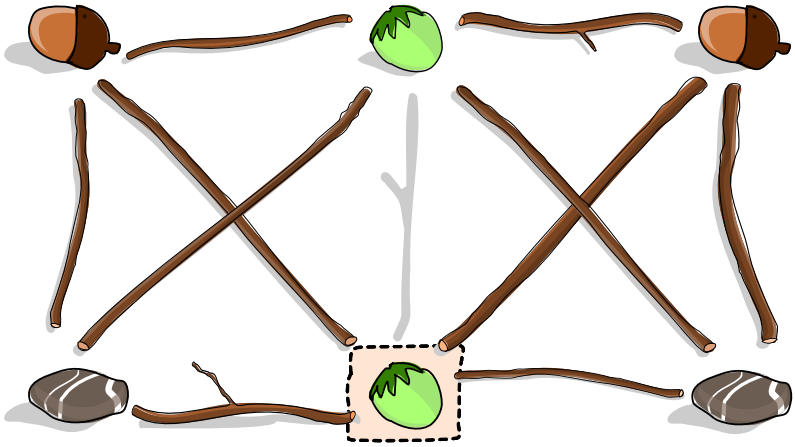
Peux-tu aider le frère d'Anja à dissimuler son méfait ?

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.



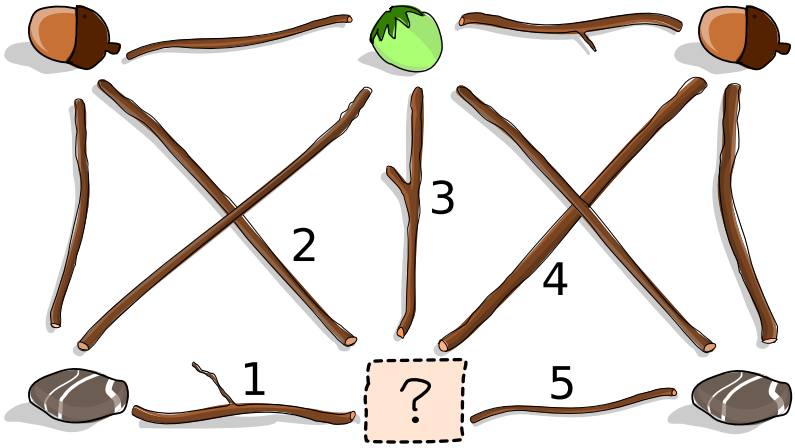
Solution

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

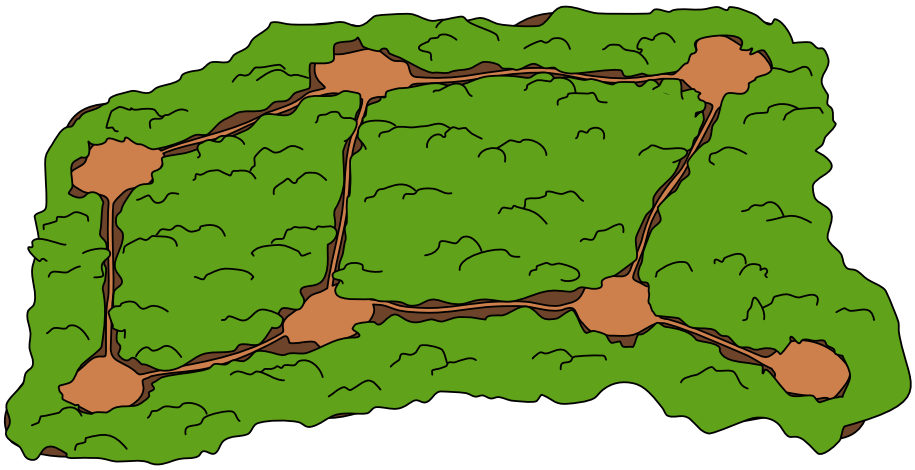
- Coloration de graphe : `https://fr.wikipedia.org/wiki/Coloration_de_graphe`
- Coloration des arêtes d'un graphe : `https://fr.wikipedia.org/wiki/Coloration_des_arêtes_d'un_graphe`
- Clique : `https://fr.wikipedia.org/wiki/Clique_(théorie_des_graphes)`

© Castor Informatique 2021, SSIE

11. Gardes forestiers

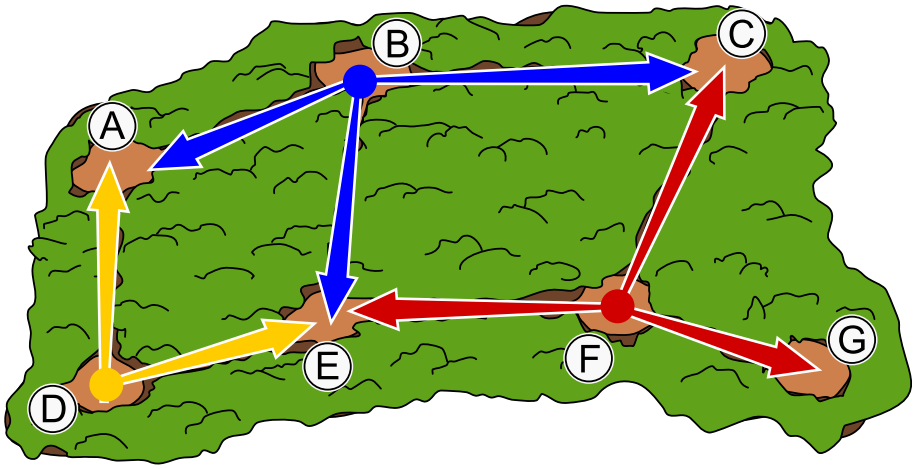
Les gardes forestiers veulent observer les animaux sur les sentiers de la forêt. Depuis chaque clairière, ils peuvent voir tous les sentiers allant de cette clairière à une clairière suivante. Il doit y avoir aussi peu de gardes forestiers que possible qui surveillent les sentiers.

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.

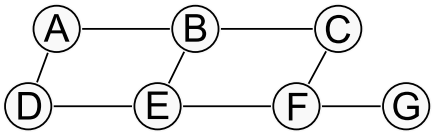
Il faut donc au minimum trois gardes forestiers pour assurer la surveillance de tous les sentiers. La solution présentée ici a donc le plus petit nombre de gardes forestiers possible. Il n'y a pas d'autre solution avec exactement trois gardes forestiers.

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 :



 © Castor Informatique 2021, SSIE

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.

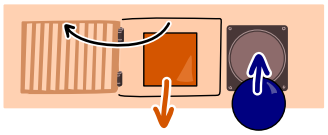
Mots clés et sites web

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

© Castor Informatique 2021, SSIE

12. Casse-tête d'anniversaire

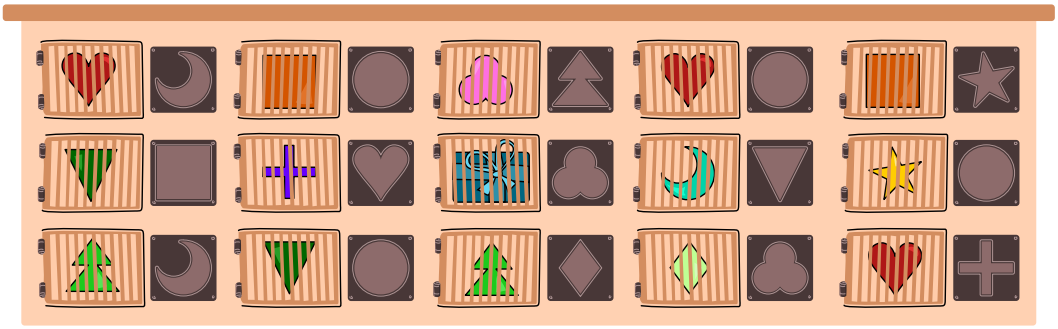
Pour son anniversaire, Bastien reçoit une boîte avec 15 portes. Il y a un autre cadeau derrière la porte du milieu et des plots de différentes formes derrière les autres portes. Chaque porte est associée à un trou qui se trouve à sa droite. Bastien peut ouvrir une porte en mettant un plot de la même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



Au début, Bastien a ce plot rond : 

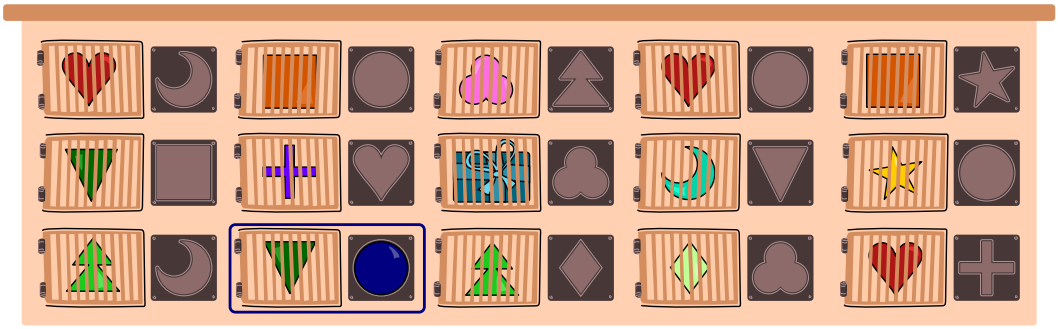
Il veut ouvrir au maximum cinq portes pour obtenir le cadeau.

Quelle porte Bastien doit-il ouvrir en premier ?

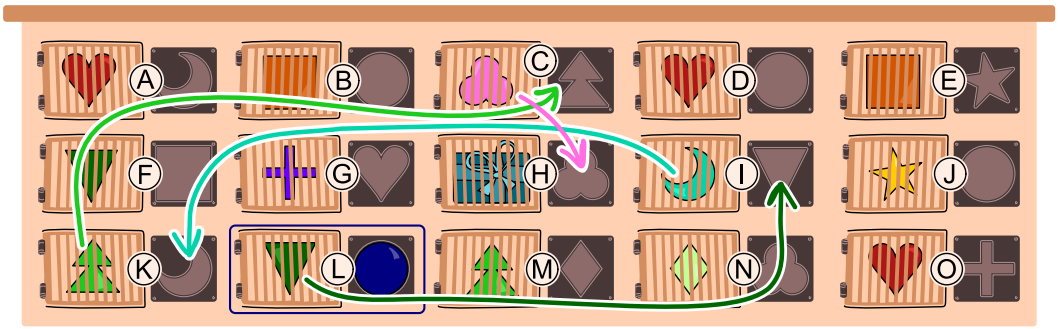


Solution

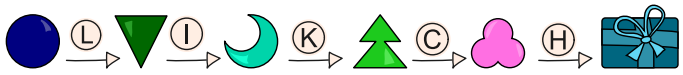
Bastien doit commencer par ouvrir la porte encadrée en bleu :



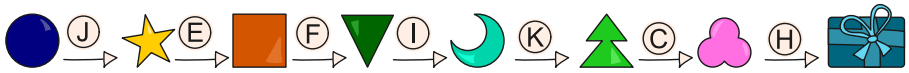
Dans l'illustration ci-dessous, chaque porte correspond a une lettre et les flèches montrent comment Bastien obtient le cadeau en n'ouvrant que cinq portes au maximum.



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 :



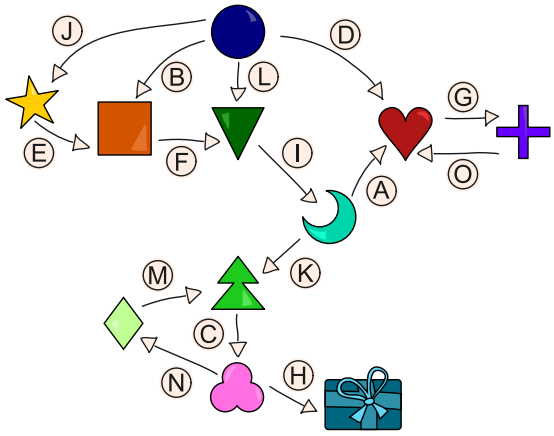
Mais ces chemins sont tous trop longs : il faut ouvrir plus de cinq portes. C'est un gros travail que d'essayer toutes les possibilités.

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* :

 © Castor Informatique 2021, SSIE



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`

A. Auteur·e·s des exercices

Sarah Atkins
Michael Barot
Liam Baumann
Linda Björk Bergsveinsdóttir
Sarah Chan
Valentina Dagienė
Christian Datzko
Susanne Datzko
Nora A. Escherle
Margarita Flores-Sicich
Fabian Frei
Gerald Futschek
Yasemin Gülbahar
Ezgi Arzu Güneş
Benjamin Hirsch
Juraj Hromkovič
Andrea Hrušecká
Tiberiu Iorgulescu
Ungyeol Jung
Filiz Kalelioğlu
Martin Kandlhofer
Vaidotas Kinčius
Víctor Koleszar
Regula Lacher
Taina Lehtimäki
Marielle Léonard
Tom Naughton
Graciela Oyhenard
Elsa Pellet
Jean-Philippe Pellet
Zsuzsa Pluhár
Wolfgang Pohl
Peter Rossmanith
Eljakim Schrijvers
Rosario Schunk
Troy Vasiga
Florentina Voboril
Michael Weigend

© Castor Informatique 2021, SSIE

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

© Castor Informatique 2021, SSIE

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.