INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Exercices et solutions 2022

Années HarmoS 7/8

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

Éditeurs :

Susanne Datzko, Nora A. Escherle,
Elsa Pellet, Jean-Philippe Pellet

SSIE
www.svia-ssie-ssii.ch
schweizerischer verein für informatik in der ausbildung // société suisse pour l'information dans l'enseignement // società svizzera per l'informatica nell'insegnamento

HASLERSTIFTUNG ETH zürich | ABZ AUSBILDUNGS- UND BERATUNGSZENTRUM FÜR INFORMATIKUNTERRICHT hep/ Scuola universitaria professionale della Svizzera italiana SUPSI

Ont collaboré au Castor Informatique 2022

Masiar Babazadeh, Susanne Datzko, Jean-Philippe Pellet, Giovanni Serafini, Bernadette Spieler

Cheffe de projet : Nora A. Escherle

Nous adressons nos remerciements pour le travail de développement des exercices du concours à :
Juraj Hromkovič, Christian Datzko, Jens Gallenbacher, Regula Lacher : ETH Zurich, Ausbildungges- und Beratungszentrum für Informatikunterricht
Tobias Berner : Pädagogische Hoschschule Zürich
Waël Almoman : Collège Voltaire

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, Dave Oostendorp, Alieke Stijf, Kyra Willekes, Jo-Ann Bolten : 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, Lena Frölich : Senarclens Leu + Partner AG

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

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

INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Le Castor Informatique 2022 a été réalisé par la Société Suisse pour l'Informatique dans l'Enseignement (SSIE) et soutenu de manière déterminante par la Fondation Hasler. Les sponsors du concours sont l'Office de l'économie et du travail du canton de Zurich et l'UBS.

Cette brochure a été produite le 22 novembre 2023 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 2022.

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.

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 2022 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

Chaque tranche d'âge avait des exercices classés en trois niveaux de difficulté : facile, moyen et difficile. 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 (en étant classés différemment dans les niveaux de difficulté).

Certains exercices sont indiqués comme « bonus » pour certaines catégories d'âge : ils ne comptent pas dans le total des points, mais servent à départager plusieurs scores identiques en cas de qualification pour les éventuels tours suivants.

Pour de plus amples informations :

SVIA-SSIE-SSII Société Suisse pour l'Informatique dans l'Enseignement
Castor Informatique
Jean-Philippe Pellet

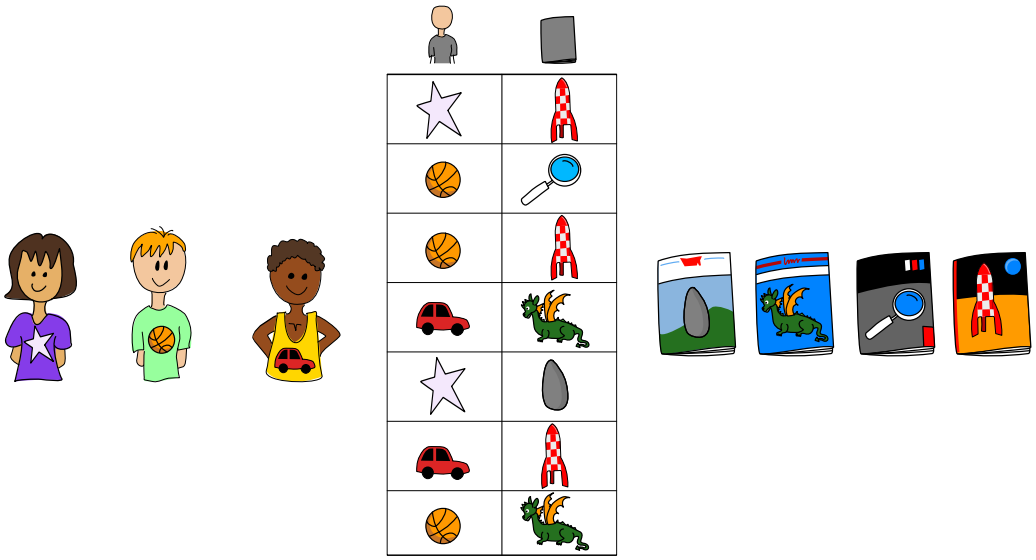
`https://www.castor-informatique.ch/fr/kontaktieren/`
`https://www.castor-informatique.ch/`

Table des matières

1. Bibliothèque

Les enfants empruntent des livres à la bibliothèque. La bibliothécaire note dans une table quel enfant a emprunté quel livre.

Quel livre les enfants ont-ils emprunté le plus souvent ?

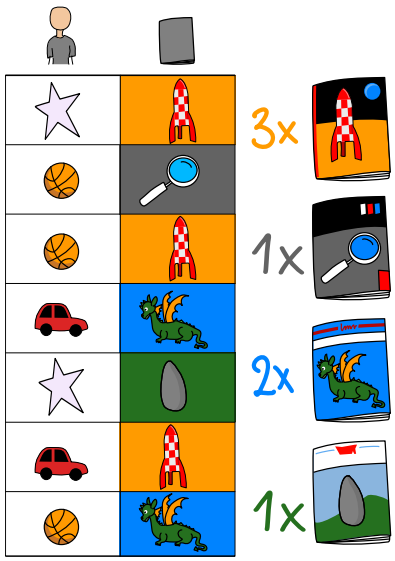


Solution

La bonne réponse est C)  .

La liste indique que :

- Trois enfants ont emprunté le livre avec la fusée.
- Un enfant a emprunté le livre avec la loupe.
- Deux enfants ont emprunté le livre avec le dragon.
- Un enfant a emprunté le livre avec le menhir.

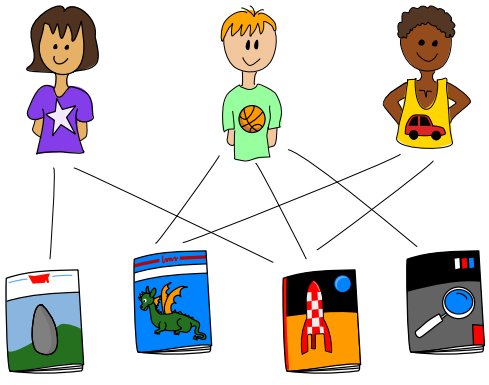


C'est donc le livre avec la fusée qui a été emprunté le plus souvent.

C'est de l'informatique !

C'est super que les participants au concours du Castor Informatique aiment lire !

Mais avons-nous vraiment besoin d'une table avec les enfants et les livres pour représenter les goûts des enfants ? Est-ce qu'on ne pourrait pas simplement dessiner des lignes ?



Ce serait ainsi plus faciles pour des êtres humains, mais pas des ordinateurs. Les ordinateurs n'arrivent pas bien à lire des lignes, mais peuvent très bien travailler avec des tables. Si l'on veut que les ordinateurs nous aident dans notre travail, par exemple pour savoir quel enfant a emprunté quel livre, ou à quelle personne appartient quel compte en banque, c'est en général une bonne idée d'utiliser des tables.

Les tables étaient déjé utilisées à Babylone il y a 4000 ans pour enregistrer des informations concernant des *relations*. Les tables sont un concept central des *bases de données relationnelles* grâce à leur capacité d'enregistrer des relations.

Les tables représentent les relations entre des objets, et ces relations déterminent comment les informations sont représentées dans la table. Par exemple, si chaque enfant n'avait le droit d'emprunter qu'un seul livre, la table n'aurait qu'une ligne par chaque enfant. Dans notre exemple avec la bibliothèque, chaque enfant peut emprunter plusieurs livres ; ils peuvent même emprunter les mêmes livres que d'autres enfants. Dans ce cas-là, nous avons besoin de cette table particulière qui relie les enfants et les livres – et qui peut contenir plusieurs fois chaque enfant et chaque livre.

La table d'emprunts est pratique. Si un livre manque, la bibliothécaire peut par exemple regarder s'il a été emprunté. La table a deux colonnes et beaucoup de lignes. Dans la première colonne, l'enfant qui emprunte un livre est enregistré, et le livre dans la deuxième colonne. On peut alors répondre à la question du livre le plus souvent emprunté facilement en comptant le nombre de fois que le livre apparaît dans la deuxième colonne.

Cette tâche pourrait aussi être accomplie par un ordinateur. Ce n'est pas possible autrement quand il s'agit d'une grande bibliothèque avec plusieurs milliers de livres ! Dans une telle bibliothèque, il n'y a pas seulement une table des emprunts, mais aussi un fichier client (une table client) dans laquelle les informations sur les clients comme leur nom, leur adreese et leur numéro de téléphone sont répertoriées, et un registre des livres (table des livres) avec des informations sur les livres comme l'auteur et le titre. Comme ça, le table d'emprunts reste légère et n'indique que la relation entre les clients et les livres (c'est-à-dire qui a emprunté quel livre).

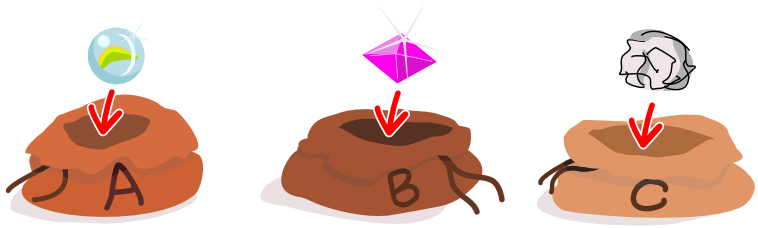
En informatique, de telles tables sont appelées bases de données relationnelles.

Mots clés et sites web

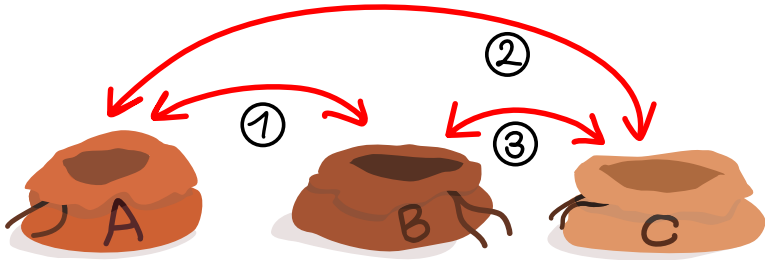
- Base de données relationnelle :
  `https://fr.wikipedia.org/wiki/Base_de_données_relationnelle`

2. Permutations

Lila met une bille dans le sac A, une pierre précieuse dans le sac B et un bout de papier dans le sac C.



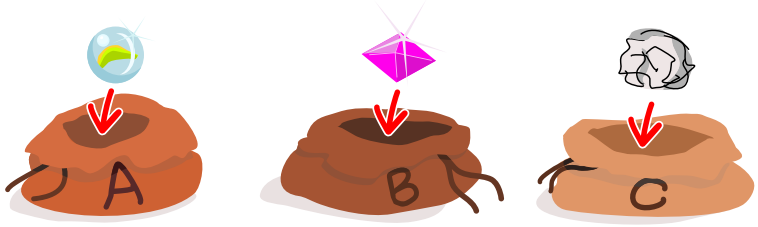
Elle échange ensuite le contenu du sac A et du sac B, puis du sac A et du sac C et enfin du sac B et du sac C.



Où se trouvent les trois objets ?

Solution

Au départ, les objets se trouvent dans ces sacs :



Lila échange les objets trois fois. Après le premier échange (A-B), les objets sont répartis dans les sacs comme ceci :

Après le deuxième échange (A-C), ils sont répartis comme ceci :

Après le troisième échange (B-C), ils sont répartis comme ceci :

À la fin, le bout de papier est dans le sac A, la pierre précieuse dans le sac B et la bille dans le sac C. On aurait pu arriver à cette solution plus facilement avec un seul échange des contenus des sac A et C.

C'est de l'informatique !

Cet exercice se concentre sur les séquences d'objets. Une telle séquence est aussi appelée arrangement. Une séquence différente représente un autre arrangement. Une permutation change la séquence et génère ainsi un nouvel arrangement. Dans cet exercice, nous commençons avec l'arrangement bille-pierre précieuse-papier et terminons après trois permutations par l'arrangement papier-pierre précieuse-bille.

Une question intéressante est de déterminer combien d'arrangements differents de trois objets existent. Pour simplifier les choses, nous pouvons commencer par chercher les arrangements commençant par

un objet précis ; il ne reste que deux arrangements possibles pour les deux objets restants. Si la bille est en première place, les deux arrangements sont :

Bille-pierre précieuse-papier
Bille-papier-pierre précicieuse

Il existe donc aussi deux arrangements différents avec chacun des deux autres objets en première place, donc quatre autres arrangements des trois objets :

Pierre précicieuse-bille-papier
Pierre précieuse-papier-bille
Papier-bille-pierre précieuse
Papier-pierre précicieuse-bille

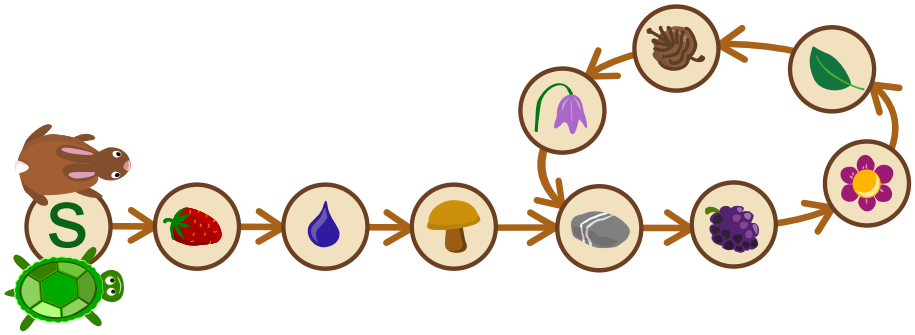
C'est aussi intéressant de savoir que l'on peut obtenir n'importe quel arrangment par permutation. Il faut pour cela au maximum $n - 1$ permutations pour n objets.

Mots clés et sites web

- Permutations : `https://fr.wikipedia.org/wiki/Permutation`

3. Le lièvre et la tortue

Une tortue et un lièvre font la course. Ils utilisent la piste ci-dessous :



Ils partent en même temps de la case départ. Ils avancent de case en case en suivant les flèches.

- La tortue avance d'une case par minute.
- Le lièvre avance de deux cases par minute.

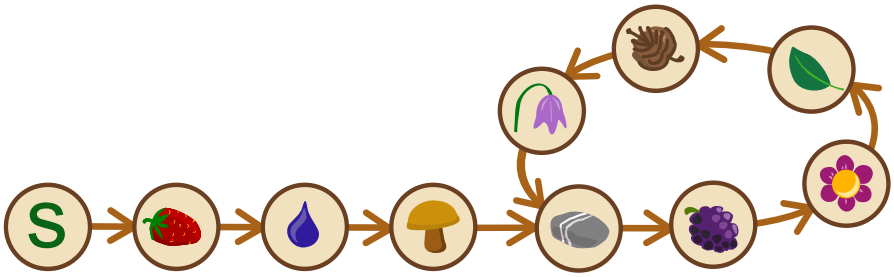
Sur quelle case le lièvre et la tortue se rencontrent-ils pour la première fois après le départ ?

Solution

Le lièvre et la tortue se rencontrent pour la première fois sur la case . On peut le voir facilement en utilisant deux doigts.

La table suivante indique les cases sur lesquelles le lièvre et la tortue se trouvent minute par minute :

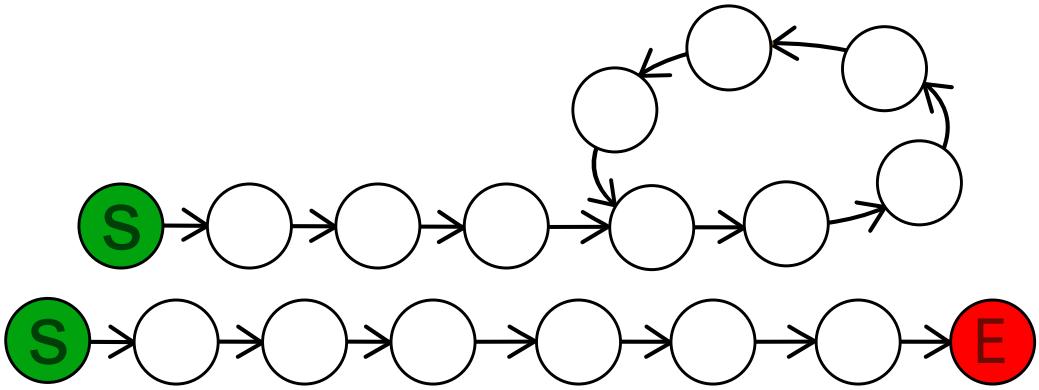
Min. après le départ	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**...**
	S														...
	S														...



C'est de l'informatique !

La course de cet exercice a lieu sur une piste spéciale. Elle est constituée de cases et de flèches qui montrent la case suivante. Sa particularité est qu'elle se termine par un cercle sur lequel les coureurs peuvent courir indéfiniment. Dans cet exercice, le lièvre et la tortue ne peuvent se rencontrer que parce que six cases forment un cercle, ou un *cycle*.

En informatique, une piste comme celle décrite dans cet exercice serait appelée une *liste*. Un cercle de cases qui renvoient les unes aux autres serait appelé un *cycle*. Dans la liste, chaque nœud renvoie à un seul autre nœud au maximum. Il existe des listes avec un cycle, comme dans cet exercice, et des listes sans cycle.



Si une liste ne contient pas de cycle, elle est constituée d'une chaîne linéaire de nœuds. Il doit alors y avoir une case d'arrivée de laquelle ne part plus de flèche. Le célèbre informaticien Robert W. Floyd

(1936−2001) a développé un algorithme qui peut déterminer de manière simple si une liste contient un cycle ou est constituée d'une chaîne linéaire. Comme dans notre exercice, il fait partir le lièvre et la tortue de la case départ ; s'ils se rencontrent sur la même case, il y a un cycle dans la liste. Au moment où le lièvre atteint la case d'arrivée ou celle d'avant, on sait qu'il n'y a pas de cycle et l'algorithme se termine.

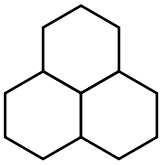
Mots clés et sites web

- Liste : `https://fr.wikipedia.org/wiki/Liste_chaînée`
- Cycle : `https://fr.wikipedia.org/wiki/Cycle_(théorie_des_graphes)`
- Nœud : `https://fr.wikipedia.org/wiki/Sommet_(théorie_des_graphes)`
- Robert W. Floyd : `https://fr.wikipedia.org/wiki/Robert_Floyd`
- Algorithme : `https://fr.wikipedia.org/wiki/Algorithme`

4. Pyramide colorée

Sami assemble des hexagones blancs, puis il les colorie de trois couleurs différentes.

Trois hexagones assemblés côte à côte comme montré ci-contre (deux en bas et un au milieu en dessus) doivent toujours avoir :

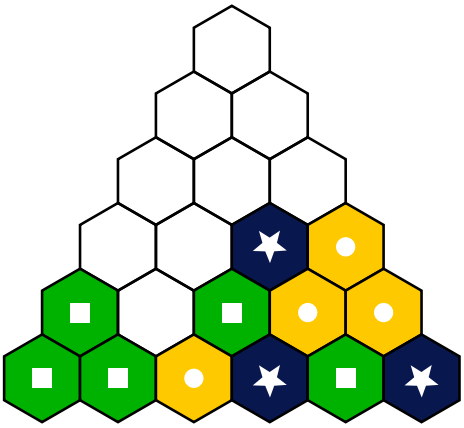


- soit tous la même couleur,
- soit trois couleurs différentes.

Sami trouve cela joli comme ça.

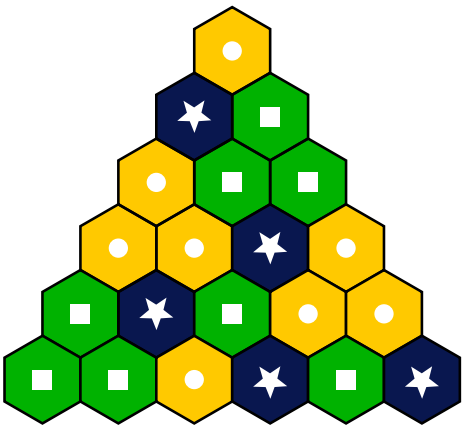
Sami a assemblé beaucoup d'hexagones et en a déjà colorié quelques-uns.

Colorie le reste des hexagones comme Sami aime.



Solution

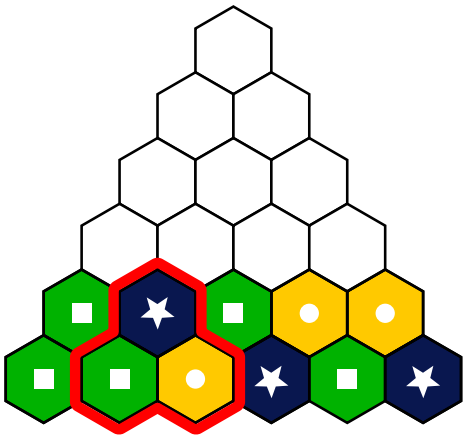
Voici la bonne solution :

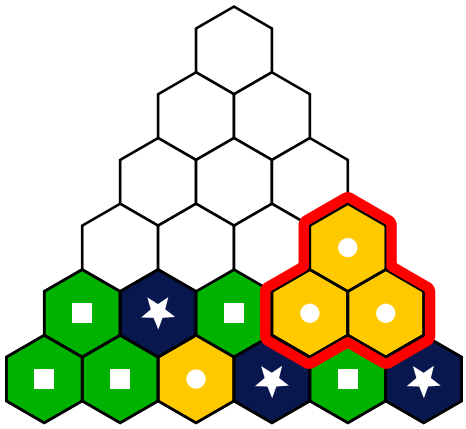


Dès que deux hexagones qui sont côte à côte dans la pyramide sont coloriés, la couleur du troisième est déterminée :

S'ils sont de couleurs différentes, le troisième hexagone est colorié de la troisième couleur. L'hexagone blanc du bas est par exemple colorié en bleu.

S'ils sont de la même couleur, le troisième est aussi colorié de la même couleur. Par exemple, l'hexagone en dessus des deux jaunes est aussi colorié en jaune.





On peut colorier ainsi les hexagones restants ligne par ligne en commençant en bas de manière à ce que Sami trouve ça joli.

C'est de l'informatique !

Comment résout-on cet exercice du Castor ? En coloriant un hexagone, on exécute une action. Pour choisir la bonne action (la bonne couleur), il faut considérer les hexagones en dessous et vérifier quelles *conditions* ils remplissent : ont-ils la même couleur ou des couleurs différentes ? Cette vérification et l'action qui s'ensuit sont *répétées* pour chaque hexagone blanc situé en dessus de deux hexagones déjà coloriés.

Actions, conditions, répétitions : il s'agit là des bases de tout *algorithme*. Un algorithme est une méthode décrite précisément qui peut être implémentée comme programme informatique. En résolvant cet exercice, tu as donc inventé un algorithme. C'est là une des tâches les plus importantes des informaticiennes et informaticiens : inventer des algorithmes ou utiliser des algorithmes existants et en faire des programmes informatiques afin de résoudre des exercices et des problèmes en traitant les informations automatiquement.

Mots clés et sites web

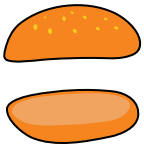
- Algorithme : `https://fr.wikipedia.org/wiki/Algorithme`
- Instruction conditionnelle : `https://fr.wikipedia.org/wiki/Instruction_conditionnelle_(programmation)`
- Boucle : `https://fr.wikipedia.org/wiki/Structure_de_contrôle#Boucles`

5. Recette de hamburger

Le castor Jess prépare des hamburgers. Il suit pour cela trois règles :

1. La sauce est mise directement sur la viande.
2. La viande et le fromage sont mis sous les cornichons, la salade et les oignons.
3. Les oignons ne touchent pas le pain.

Ingrédients des hamburgers :

pain	**viande**	**sauce**	**cornichons**	**salade**	**oignons**	**fromage**
						

Lequel des hamburgers a été préparé en suivant les trois règles ?

A) 

B) 

C) 

D) 

Solution

La bonne réponse est D. 

Pour trouver la solution, nous devons regarder si chaque hamburger a été préparé en suivant les trois règles.

A) Ce hamburger correspond aux règles 1 et 2, mais les oignons touchent le pain du haut. Il ne respecte donc pas la règle 3.

B) Ce hamburger correspond à la règle 1, mais la salade est sous la viande et le fromage. Il ne respecte donc pas la règle 2.

C) Ce hamburger correspond à la règle 2, car la viande et le fromage sont sous les cornichons, la salade et les oignons. Il correspond aussi à la règle 3, car les oignons ne touchent pas le pain. Par contre, la sauce n'est pas directement sur la viande ; il ne respecte donc pas la règle 1.

D) Ce hamburger respecte toutes les règles. Le hamburger D est donc un vrai hamburger de castor.

C'est de l'informatique !

Les hamburgers de cet exercice sont préparés en suivant trois règles. Castor Jess doit suivre chacune des trois règles pour chaque hamburger qu'il prépare. Si l'une des règles n'est pas respectée, il ne s'agit pas d'un vrai hamburger de castor. Chacune des règles est une condition qui doit être remplie pour que le hamburger soit un hamburger de castor.

En informatique, il faut souvent vérifier des conditions ou *contraintes* (*Constraint Checking*) pour savoir si une solution respecte toutes les règles données. Lors de cette vérification, toutes les règles sont reliées par *et*, ce qui veut dire que toutes les règles (les contraintes) doivent être respectées en même temps.

La vérification qu'une solution satisfait l'ensemble des contraintes est fondamentalement différente de la recherche de solution. On appelle cela un *problème de satisfaction de contraintes*. Il est souvent beaucoup plus difficile de trouver une solution qui satisfait toutes les contraintes que de vérifier si une solution les satisfait, même pour un ordinateur.

Mots clés et sites web

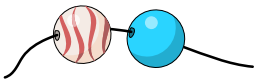
- Programmation par contraintes :
  `https://fr.wikipedia.org/wiki/Programmation_par_contraintes`
- Problème de satisfaction de contraintes :
  `https://fr.wikipedia.org/wiki/Problème_de_satisfaction_de_contraintes`

- ET (fonction logique) : `https://fr.wikipedia.org/wiki/Fonction_ET`
- NP (complexité) : `https://fr.wikipedia.org/wiki/NP_(complexité)`

6. Collier de marin

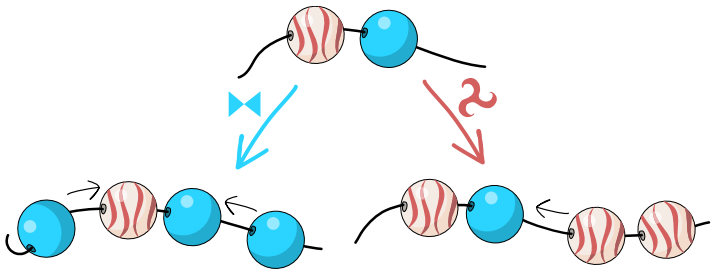
Voici les instructions de Monica pour faire son collier de marin avec des perles blanches à vagues rouges et des perles unies bleues.

Tu commences toujours par une perle à vagues puis une perle bleue, dans cet ordre :



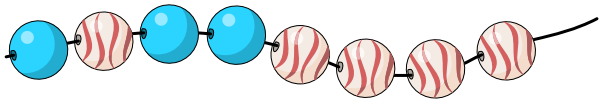
Tu peux ensuite allonger le collier :

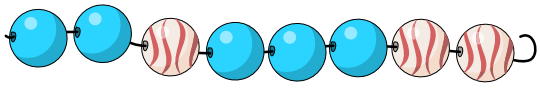
- en ajoutant une perle bleue de chaque côté du fil (⋈) ;
- en ajoutant deux perles à vagues du côté droit du fil (ᔓ).

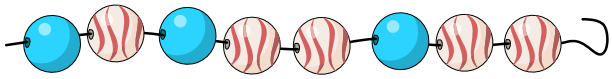


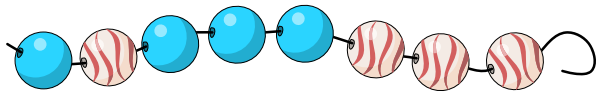
Tu peux répéter ces actions plusieurs fois pour obtenir un collier de plus en plus long.

*Lequel des colliers suivants n'est **pas** un collier de marin de Monica ?*

A) 

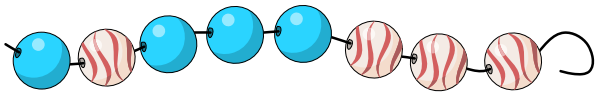
B) 

C) 

D) 

Solution

La bonne réponse est D).



Tu peux résoudre cet exercice de différentes manières.

Par exemple, tu peux commencer par chercher les deux perles de départ dans chaque collier, puis effectuer une suite d'action ⋈ et ↷.

- Pour le collier A, tu peux commencer avec la deuxième et troisième perle et effectuer ensuite les actions ⋈ - ↷ - ↷.
- Pour le collier B, tu peux commencer avec la troisième et quatrième perle et effectuer ensuite les actions ⋈ - ⋈ - ↷.
- Pour le collier C, tu peux commencer avec la deuxième et troisième perle et effectuer ensuite les actions ↷ - ⋈ - ↷.
- Pour le collier D, les deuxième et troisième perles devraient être les perles de départ. Tu pourrais ensuite effectuer l'action ⋈ une fois, mais il n'y a ensuite plus d'autres actions permettant d'obtenir le reste du collier.

Cette méthode ne fonctionne pas bien lorsque le collier est très long et a beaucoup de perles de départ possibles. Dans ce cas-là, une méthode déconstructive mène plus facilement à la solution. Pour cela, tu enlèves des perles petit à petit en effectuant les actions ⋈ ou ↷ à l'envers jusqu'à ce qu'il ne reste que deux perles.

Une autre stratégie utilise la *parité*. D'après les instructions pour fabriquer les colliers de marin, ils ont toujours un nombre pair de perles unies bleues et un nombre impair de perles blanches à vagues rouges (« parité impaire »). Tu vois pourquoi c'est le cas ?

Le collier D a un nombre pair des deux sortes de billes et ne peut donc pas âtre un collier de marin.

C'est de l'informatique !

Dans cet exercice, tu ne peux ajouter des perles qu'aux bouts du collier. Tu ne peux pas insérer une perle au milieu, et tu ne peux pas non plus enlever une perle du milieu sans avoir d'abord enlevé les perles du bout du collier.

Cette forme de structure de stockage, à laquelle il est facile d'ajouter et d'enlever des éléments aux bouts mais pas au milieu, s'appelle une *file d'attente à double extrémité* ou *deque* (de l'anglais « double-ended queue »).

Les deques peuvent être utilisées pour enregistrer l'activité d'un browser, pour planifier des ordres d'impression ou encore pour vérifier la validité d'expressions mathématiques. Dans ce cas-là, on peut

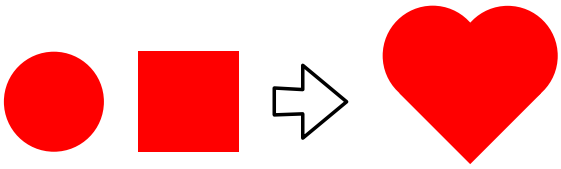
vérfifier qu'une parenthèse fermante correspond toujours à une parenthèse ouvrante de manière très similaire à celle utilisée pour vérifier si un collier est un collier de marin à Monica.

Mots clés et sites web

- deque : `https://fr.wikipedia.org/wiki/File_d'attente_à_double_extrémité`

7. Cœur composé

Tina a deux formes : un rond et un carré. Elle les transforme en cœur.



Elle utilise pour cela ces trois transformations :

- *tourner* : tourner une forme autant que désiré
- *déplacer* : déplacer une forme autant que désiré
- *dupliquer* : dupliquer une forme de manière à ce que les deux formes restent au même endroit.

Quelles transformations a-t-elle effectuées et dans quel ordre ?

A) *dupliquer* le rond, *tourner* le carré, *dépacer* le rond, *déplacer* le rond

B) *dupliquer* le carré, *tourner* le carré, *déplacer* le carré, *déplacer* le rond

C) *dupliquer* le rond, *tourner* le rond, *déplacer* le rond, *déplacer* le carré

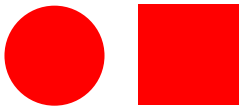
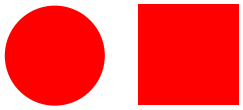
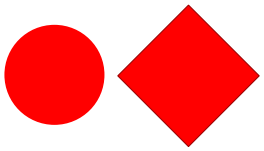
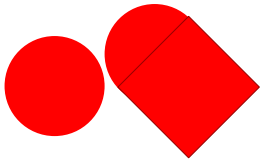
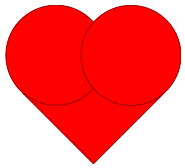
D) *déplacer* le rond, *déplacer* le rond, *dupliquer* le rond, *déplacer* le carré

Solution

Si l'on observe attentivement le cœur, on constate qu'il est formé de deux ronds et d'un carré tourné d'1/8 de tour. Les transformations doivent donc inclure *dupliquer* le rond pour obtenir deux ronds et *tourner* le carré pour que le carré soit tourné d'1/8 de tour. Les réponses B), C) et D) sont donc exclues, car :

- Dans la réponse B), un carré est dupliqué et non un rond
- Dans la réponse C), un rond est tourné, mais pas le carré
- Dans la réponse D), aucune forme n'est tournée, donc le carré n'est pas tourné.

Mais la réponse A) est-elle juste ? Les formes doivent encore être déplacées ! Les transformations suivantes sont indiquées :

- Ceci : 
- est transformé par la duplication du rond en 
- est transformé par la rotation du carré en 
- est transformé par le déplacement d'un rond en 
- est transformé par le déplacement de l'autre rond en 

La réponse A) Dupliquer le rond, tourner le carré, délpacer le rond, déplacer le rond est donc correcte.

C'est de l'informatique !

Les logiciels de traitement d'images permettent de réaliser beaucoup de transformations sur une image. Dans cet exercice, il s'agit de transformations comme tourner, déplacer ou dupliquer. Cependant, cela n'est pas suffisant : il faut encore indiquer à l'ordinateur comment une forme doit être tournée ou vers quel endroit elle doit être déplacée.

Tu pourrais bien sûr aussi décrire les étapes pour transformer un rond et un carré en cœur par un texte plus long, mais en informatique, c'est souvent mieux d'utiliser aussi peu de transformations de base que possible, et de les répéter ou de les effectuer de manières différentes. Le développement

de solutions générales à partir d'exemples précis s'appelle la généralisation. De telles commandes pourraient par exemple avoir la forme suivante :

- Tourner une forme : tourne la forme, angle
- Déplacer une forme : déplace la forme, destination
- Dupliquer une forme : duplique la forme

Le logiciel de traitement d'images de Tina peut paraître inhabituel : au lieu que l'image soit enregistrée sous forme de *pixels* comme une photo, une description de la forme (par exemple « rond, rayon 2 cm, couleur rouge ») est enregistrée. C'est ainsi possible que deux formes soient l'une sur l'autre, comme les ronds, et que l'une d'elles soit ensuite déplacée sans que l'autre n'ait été effacée. De tels images sont appelées *images vectorielles*. Elles sont souvent utilisées pour dessiner des formes abstraites en haute qualité. Les autres images, appelées *images matricielles*, sont souvent des photos ou des dessins réalistes.

Mots clés et sites web

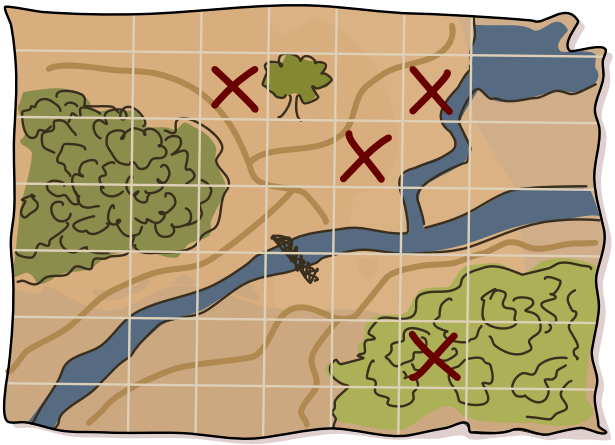
- Pixel : `https://fr.wikipedia.org/wiki/Pixel`
- Image matricielle : `https://fr.wikipedia.org/wiki/Image_matricielle`
- Image vectorielle : `https://fr.wikipedia.org/wiki/Image_vectorielle`

8. Carte au trésor

Bilbo le castor a deux bonnes cachettes pour ses réserves de nourriture. Sur une carte, il indique d'un ✕ les deux cases dans lequelles se trouvent ses cachette. Mais que faire si d'autres castors trouvent sa carte ?

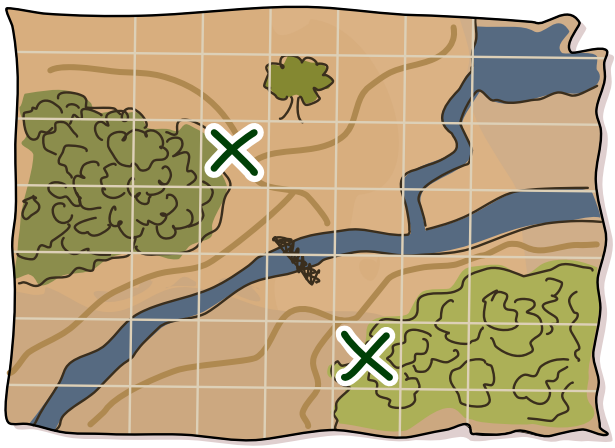
Pour brouiller les pistes, Bilbo marque d'autre cases d'un ✕. Il le fait de telle manière que chaque ligne et chaque colonne de la carte contienne un nombre pair de cases avec un ✕. Il efface ensuite les deux ✕ indiquant ses cachettes. Tu vois le résulat ci-dessous.

Dans quelles cases se trouvent les cachettes de Bilbo ?

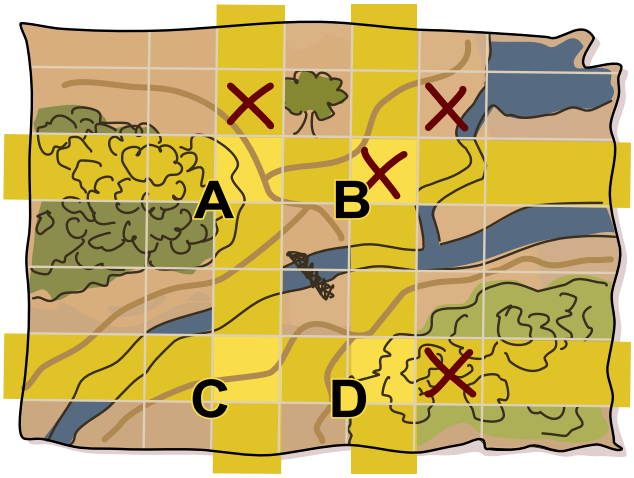


Solution

Voici les deux cachettes :



Pour les trouver, nous observons la carte de départ et constatons qu'il s'y trouve deux lignes et deux colonnes ne contenant pas un nombre pair de ✕ : les lignes 3 et 6 et les colonnes 3 et 5.



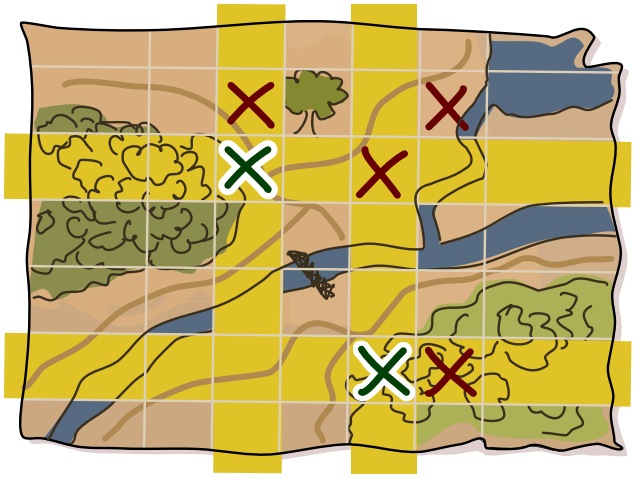
Les ✕ qui indiquent les cachettes ont été effacés. Nous savons qu'il doit y avoir un nombre pair de ✕ dans chaque ligne et chaque colonne une fois que les deux ✕ effacés y sont de nouveau dessinés.

Les lignes et colonnes concernées se croisent et ont quatre cases en commun (A, B, C et D). Ces « cases communes » sont spécialement intéressantes pour nous. Si nous mettions un ✕ dans un case autre qu'une case commune, nous pourrions obtenir un nombre pair de ✕ dans une colonne, mais un nombre impair dans une ligne et inversement. Les deux ✕ indiquant les cachettes doivent donc se trouver dans des cases communes.

La case commune B contient déjà un ✕ et ne peut donc pas être une cachette, car nous savons que Bilbo a effacé les ✕ des cases où se trouvent ses cachettes.

Pour obtenir un nombre pair de ✕ dans la ligne 2, il doit donc y avoir un ✕ dans la case commune A. Une cachette s'y trouve. L'autre cachette ne peut pas se trouver dans la case commune C, car il y aurait alors trois ✕ dans la colonne correspondante. L'autre cachette se trouve donc dans la case

commune D. Voici la carte avant que Bilbo n'efface les ✕ de ses cachettes, qui contient un nombre pair de ✕ dans chaque ligne et chaque colonne :



C'est de l'informatique !

Biblo utilise ici une astuce souvent utilisée en informatique : les *bits de parité*. Ils font partie d'une série de techniques connues sous le nom de *codes correcteurs*. Le principe est d'ajouter des bits à des données enregistrées ou transmises sous forme de séries de bits. Les bits ajoutés nous aident à determiner s'il y a eu des erreurs lors de la transmission ou de l'enregistrement de la série – typiquement, si un bit a été inversé, c'est à dire si un 1 a été enregistré ou transmis au lieu d'un 0 ou inversement.

Un exemple simple de code correcteur serait d'ajouter un bit de parité afin que le nombre de 1 soit toujours pair. On ajouterait alors un 0 à la série 0110101, qui deviendrait 01101010 (le nombre de 1 resterait donc pair). Si le deuxième bit était inversé et la série 00101010 était transmise, le message reçu ne remplirait plus la condition de parité (trois bits vaudraient 1). Cette méthode ne peut cependant pas détecter les erreurs lorsque plusieurs bits sont inversée.

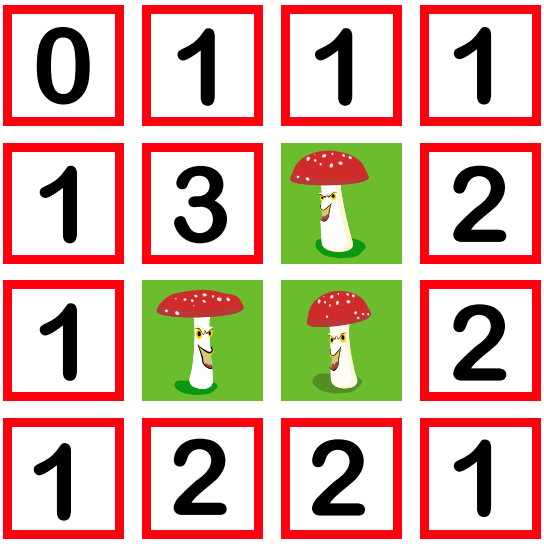
Mots clés et sites web

- Bit : `https://fr.wikipedia.org/wiki/Bit`
- Bit de parité : `https://fr.wikipedia.org/wiki/Somme_de_contrôle#Exemple_:_bit_de_parité`
- Somme de contrôle : `https://fr.wikipedia.org/wiki/Somme_de_contrôle`
- Code correcteur : `https://fr.wikipedia.org/wiki/Code_correcteur`

9. Déchampignonneur

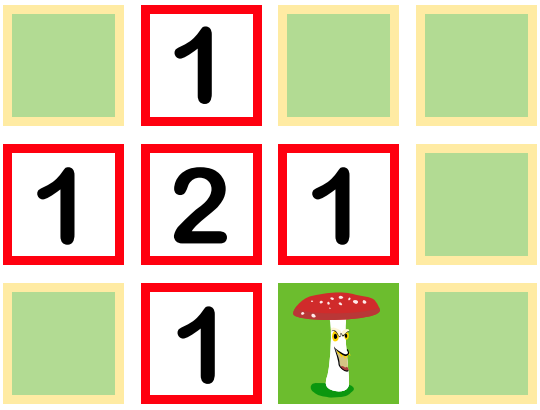
Au début de jeu du « déchampignonneur », un seul champignon et quelques cases contenant des chiffres sont visibles. Toutes les autres cases du jeu sont cachées. Lorsque tu découvres une case, un champignon ou le nombre de champignons présents dans les cases voisines apparaît. Tu gagnes le jeu si tu arrives à découvrir uniquement toutes les cases sans champignon.

Voici un exemple de jeu complètement découvert :



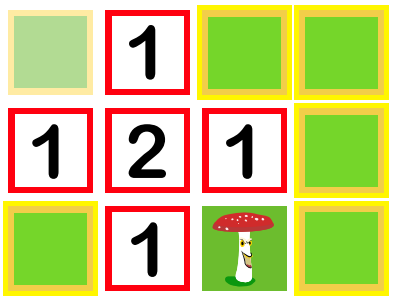
Tu commences un nouveau jeu – regarde en dessous.

Sur quelles cases ne peut-il pas y avoir de champignon ?

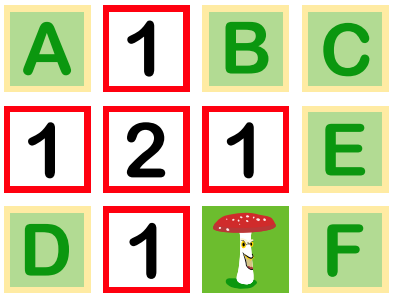


Solution

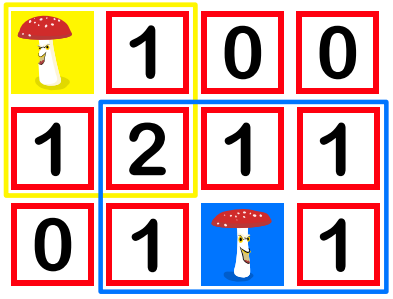
Voici la bonne solution :



Afin d'expliquer la bonne réponse, nous assignons des lettres aux cases couvertes. Nous disons qu'un chiffre N sur une case est épuisé lorsque que nous savons quelles N cases voisines contiennent un champignon ; il ne peut alors plus y avoir de champignon sur les autres cases voisines.



- Il n'y a pas de champignon sur la case D, parce que le chiffre 1 à côté est épuisé.
- Il n'y a pas de champignon sur les cases B, C, E et F, parce que le chiffre 1 sur leur case voisine commune est épuisé.
- Il y a un champignon sur la case A, car les chiffres 1, 2 et 1 sur les cases voisines n'indiqueraient sinon pas le bon nombre de champignon sur les cases voisines.



Il y a donc un champignon caché sur la case A. Les cases B, C, D, E et F peuvent être découvertes.

C'est de l'informatique !

Comment avons-nous résolu cet exercice ? Parfois, il faut commencer avec une supposition et continuer logiquement. Si l'on se trouve face à une contradiciton, on revient en arrière et continue avec la supposition suivante. Il s'agit alors d'une recherche « ciblée » et non pas aléatoire.

Comment un ordinateur résoudrait-il cet exercice ? Si au moins une case avec un champignon est découverte, de simples règles peuvent être énoncées. Par exemple, s'il y a déjà une case avec un champignon découvert à côté d'une case avec le chiffre 1, il ne peut plus y avoir de champignon sur les autres cases voisines. Si l'on formule cette règle précisément pour tous les chiffres, un ordinateur peut l'utiliser comme *instruction* à exécuter pas à pas. Nous obtenons ainsi un *algorithme* que nous pouvons « simplement » exécuter pour gagner le jeu (si au moins un champignon est déjà découvert).

Mots clés et sites web

- Démineur : `https://fr.wikipedia.org/wiki/Démineur_(genre_de_jeu_vidéo)`
- Instruction : `https://fr.wikipedia.org/wiki/Instruction_informatique`
- Algorithme : `https://fr.wikipedia.org/wiki/Algorithme`

10. Boulons et écrous

Ben assemble des pièces sur une ligne de montage : des écrous  et des boulons  .

Ben applique strictement la méthode suivante :

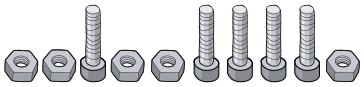
- Ben prend la pièce suivante sur la ligne de montage.
- Si c'est un écrou, il le met dans le seau.
- Si c'est un boulon, il prend un écrou dans le seau, le visse sur le boulon, et met la pièce terminée dans la boîte.

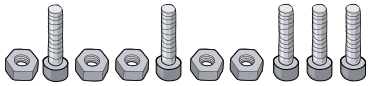
Deux erreurs peuvent se produire avec cette méthode :

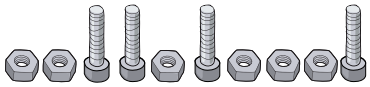
1. Ben prend un boulon sur la ligne de montage, mais il n'y a pas d'écrou à visser dessus dans le seau.
2. Ben a pris toutes les pièces sur la ligne de montage, mais il reste des écrous dans le seau.

Le seau pour les écrous est assez grand et est vide au départ. Laquelle des séquences suivantes Ben peut-il assembler de gauche à droite sans aucune erreur ?

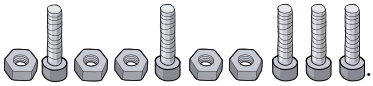
A) 

B) 

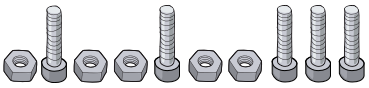
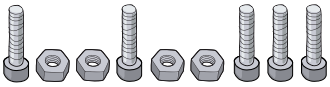
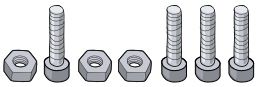
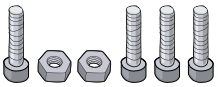
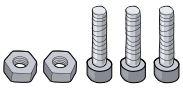
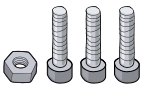
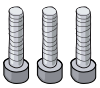
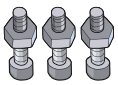
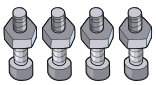
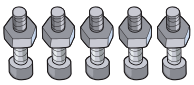
C) 

D) 

Solution

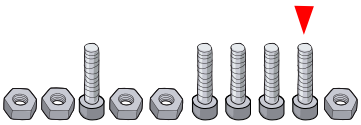
La bonne réponse est C) :  .

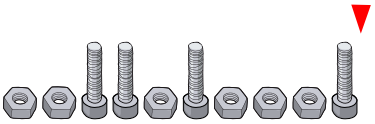
La table ci-dessous montre le contenu de la boîte pour les pièces terminées, du seau pour les écrous et de la ligne de montage.

Boîte	**Seau**	**Ligne de montage**
vide	*vide*	
vide		
	*vide*	
		
		
		
		
		
		
		
	*vide*	*vide*

Pourquoi les autres réponses sont-elles fausses ?

A) La séquence  cause une erreur à la position indiquée. Ben a pris un boulon, mais il n'y a plus d'écrou dans le seau.

B) La séquence  cause une erreur à la position indiquée. Ben a vissé quatre écrous sur quatre boulons, le seau est donc vide lorsqu'il prend le dernier boulon pour laquelle il n'a plus d'écrou.

D) La séquence  cause une erreur une fois la ligne de montage vide. En effet, quatre écrous ont été vissés sur quatre boulons et il reste encore deux écrous.

C'est de l'informatique !

Ben travaille avec des pièces qui sont livrées les unes après les autres sur la ligne de montage. Pour cela, il utilise un grand seau pour stocker les écrous. En *informatique théorique*, une séquence similaire est utilisée comme modèle pour les *algorithmes* qui peuvent résoudre un certain type de problèmes : les *automates à pile*.

Un automate à pile traite des données (des chiffres ou symboles) qu'il reçoit en entrée les unes après les autres. Il possède un seul espace de stockage infini, une pile. Au contraire du seau de l'exercice, les éléments de la pile ont un ordre précis, et seul le dernier élément ajouté à la pile peut en être ressorti (« last in, first out », LIFO). Un automate à pile peut être utilisé pour reconnaître un *langage non contextuel*.

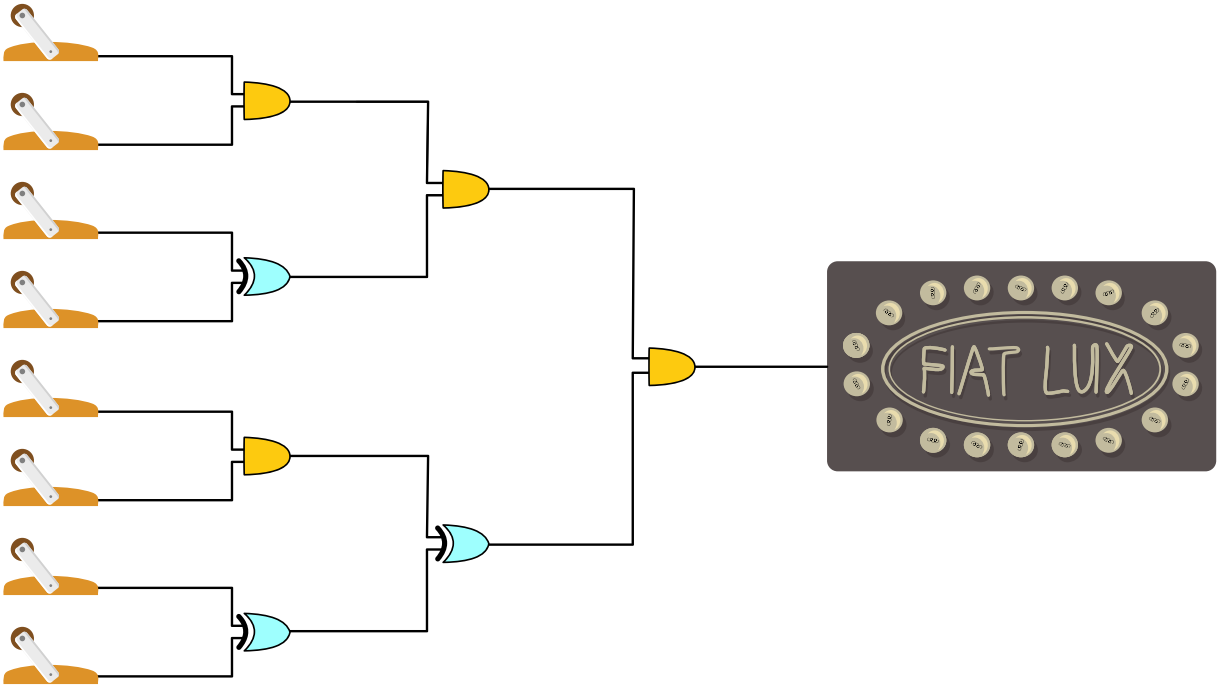
En informatique, un langage est constitué d'un ensemble de chaînes de symboles qui ont été assemblé d'après certaines règles. Les langages non contextuels appartiennent à une classe de langages simples. Par exemple, toutes les expressions entre parenthèses bien formulées sont un langage non contextuel. Une expression entre parenthèses bien formulée commence toujours par une parenthèse ouverte qui est ensuite refermée. Par exemple, les expressions ((())) et (()()) sont bien formulées. Par contre, les expressions (((() et ())(ne le sont pas. On peut se représenter les boulons et les écrous dans l'exercice comme des parenthèses ouvrantes et fermantes ; Ben traite alors une séquence de pièces sans erreurs seulement lorsqu'elle représente une expression entre parenthèses bien formulée. La vérification des parenthèses est une tâche importante pour un compilateur qui traduit des textes de programmes en programmes exécutables. En effet, dans la plupart des langages de programmation, les textes de programmes contiennent des appels de fonctions emboîtés et des expressions arithmétiques entre parenthèses.

Mots clés et sites web

- Informatique théorique : `https://fr.wikipedia.org/wiki/Informatique_théorique`
- Automate à pile : `https://fr.wikipedia.org/wiki/Automate_à_pile`
- Langage non contextuel : `https://fr.wikipedia.org/wiki/Langage_algébrique`

11. Que la lumière soit !

Le jeu « Que la lumière soit ! » est composé de 8 interrupteurs pouvant être actifs ou inactifs. Des fils relient ces interrupteurs à un panneau publicitaire lumineux en passant par différents composants.

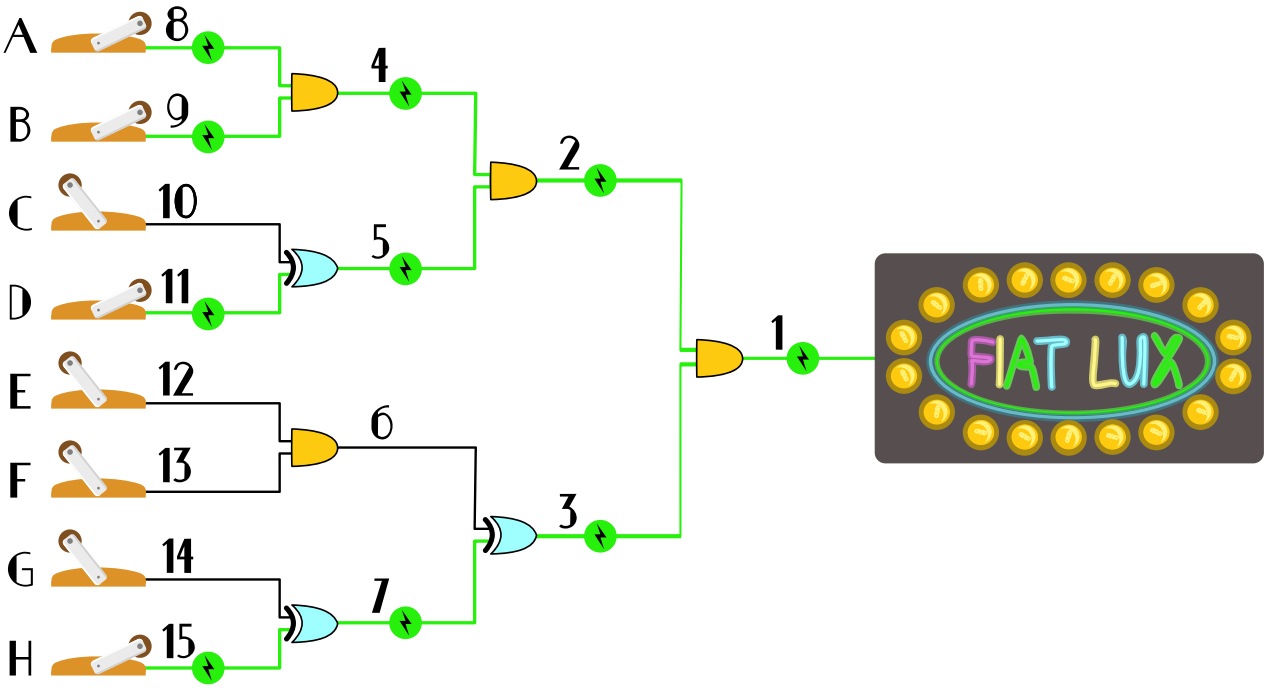


Les fils sortant des interrupteurs sont actifs lorsque l'interrupteur correspondant est allumé. La sortie du composant  est active seulement lorsque les deux fils entrants sont actifs. La sortie du composant  est active seulement lorsqu'un seul des deux fils entrants est actif.

Quels interrupteurs doivent être allumés  *afin d'allumer le panneau publicitaire ?*

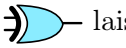
Solution

Une des solutions possibles est la suivante :



On arrive facilement à la solution en commençant à résoudre le problème par la fin. Le fil 1 est relié à un composant  . Pour que la sortie de ce composant soit *active*, les deux fils entrants, 2 et 3, doivent être *actifs*.

- Le fil 2 est relié à un composant  . Pour que sa sortie soit *active*, les deux fils entrants, 3 et 4, doivent être *actifs*.
- Le fil 3 est relié à un composant  . Pour que sa sortie soit *active*, seul l'un des deux fils entrants peut être *actif*, par exemple le fil 7. Le fil 6 doit alors être *inactif*.
- Le fil 4 est relié à un composant  . Pour que sa sortie soit *active*, les deux fils entrants, 8 et 9, doivent être *actifs* ; les deux interrupteurs A et B doivent donc aussi être *allumés* :  .
- Le fil 5 est relié à un composant  . Pour que sa sortie soit *active*, seul l'un des deux fils entrants peut être *actif*, par exemple le fil 11. Le fil 10 doit alors être *inactif*. L'interrupteur C doit donc être *éteint*  et l'interrupteur D *allumé*  .
- Le fil 6 est relié à un composant  . Pour que sa sortie soit *inactive*, au moins l'un des deux fils entrants, 12 et 13, doit être *inactif* ; les deux interrupteurs E et F peuvent donc être les deux *éteints* :  .
- Le fil 7 est relié à un composant  . Pour que sa sortie soit *active*, seul l'un des deux fils entrants peut être *actif*, par exemple le fil 15. Le fil 14 doit alors être *inactif*. L'interrupteur G doit donc être *éteint*  et l'interrupteur H *allumé*  .

Les composants  laissent des alternatives, car ils permettent de décider lequel des deux fils entrants est *actif*. De plus, pour le composant  avec fil 6 sortant, on peut décider si aucun des deux fils entrants ou seul l'un des deux est *inactif* afin que la sortie reste *inactive*. Si la sortie du composant  est *active*, les deux fils entrants doivent également être *actifs*, les deux entrées du composant  avec le fil 7 sortant doivent être soit les deux *actifs*, soit les deux *inactifs*. Cela donne au total 16 différentes combinaisons possibles :

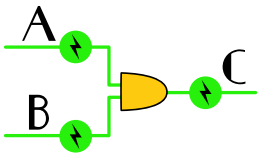
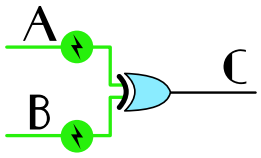
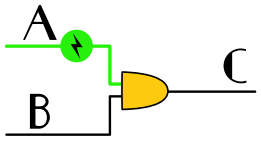
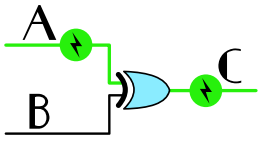
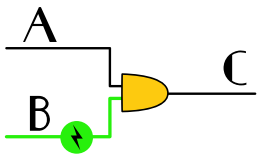
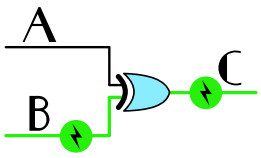
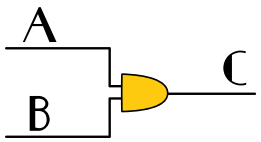
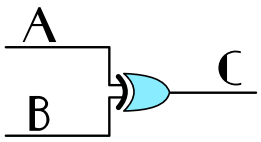
Interrupteur								**Fil**	
A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**6**	**7**
Toujours *allumés*		**Un seul *allumé***		**Les deux *allumés* si fil 6 *actif*, sinon au plus un *allumé***		**Un seul *allumé* si fil 7 *actif*, sinon les deux *allumés* ou *éteints***		**Un seul *actif***	
allumé	allumé	allumé	éteint	allumé	allumé	allumé	allumé	actif	inactif
allumé	allumé	éteint	allumé	allumé	allumé	allumé	allumé	actif	inactif
allumé	allumé	allumé	éteint	allumé	allumé	éteint	éteint	actif	inactif
allumé	allumé	éteint	allumé	allumé	allumé	éteint	éteint	actif	inactif
allumé	allumé	allumé	éteint	allumé	éteint	allumé	éteint	inactif	actif
allumé	allumé	éteint	allumé	allumé	éteint	allumé	éteint	inactif	actif
allumé	allumé	allumé	éteint	allumé	éteint	éteint	allumé	inactif	actif
allumé	allumé	éteint	allumé	allumé	éteint	éteint	allumé	inactif	actif
allumé	allumé	allumé	éteint	éteint	allumé	allumé	éteint	inactif	actif
allumé	allumé	éteint	allumé	éteint	allumé	allumé	éteint	inactif	actif
allumé	allumé	allumé	éteint	éteint	allumé	éteint	allumé	inactif	actif
allumé	allumé	éteint	allumé	éteint	allumé	éteint	allumé	inactif	actif
allumé	allumé	allumé	éteint	éteint	éteint	allumé	éteint	inactif	actif
allumé	allumé	éteint	allumé	éteint	éteint	allumé	éteint	inactif	actif
allumé	allumé	allumé	éteint	éteint	éteint	éteint	allumé	inactif	actif
allumé	allumé	éteint	allumé	éteint	éteint	éteint	allumé	inactif	actif

C'est de l'informatique !

Le courant peut passer ou non à travers les fils de cet exercice, les interrupteurs sont donc soit éteints, soit allumés. En informatique, de tels états représentent les valeurs de *variables booléennes*, qui sont souvent nommés *VRAI* et *FAUX* ou *1* et *0*.

Les ordinateurs actuels fonctionnent en règle générale uniquement avec ces variables, comme le jeu de cet exercice. Cela vient entre autre du fait que des milliards de *transistors* dont l'état est aussi *actif* ou *inactif* sont présents au cœur de l'ordinateur.

On peut construire des portes logiques avec plusieurs transistors. Deux de ces portes sont présentes dans cet exercice : le composant  est une *porte ET* dont la sortie est VRAI lorsque les deux entrées sont VRAI. Le composant  est une *porte OU exclusif* dont la sortie est VRAI lorsqu'exactement une des deux entrées est VRAI. On peut aussi les représenter dans une *table de vérité* :

Entrées		Porte ET		Porte OU exclusif	
Entrée A	**Entrée B**	**Image**	**Sortie C**	**Image**	**Sortie C**
VRAI	VRAI		VRAI		FAUX
VRAI	FAUX		FAUX		VRAI
FAUX	VRAI		FAUX		VRAI
FAUX	FAUX		FAUX		FAUX

D'autres portes répandues sont la *porte OU*, dont la sortie est VRAI lorsqu'au moins l'un des deux entrées est VRAI, et la *porte NON*, dont la sortie est VRAI lorsque l'entrée est FAUX. Souvent, on utilise des combinaisons de portes ET et de porte NON dans la construction de circuits, car cela demande peu de transistors. Les tables de vérité sont :

Entrée A	Entrée B	Sortie porte OU	Sortie porte NON-ET
VRAI	VRAI	VRAI	FAUX
VRAI	FAUX	VRAI	VRAI
FAUX	VRAI	VRAI	VRAI
FAUX	FAUX	FAUX	VRAI

Entrée	Sortie porte NON
VRAI	FAUX
FAUX	VRAI

Un ordinateur peut effectuer des calculs compliqués très rapidement en utilisant des combinaisons de *portes logiques* adaptées.

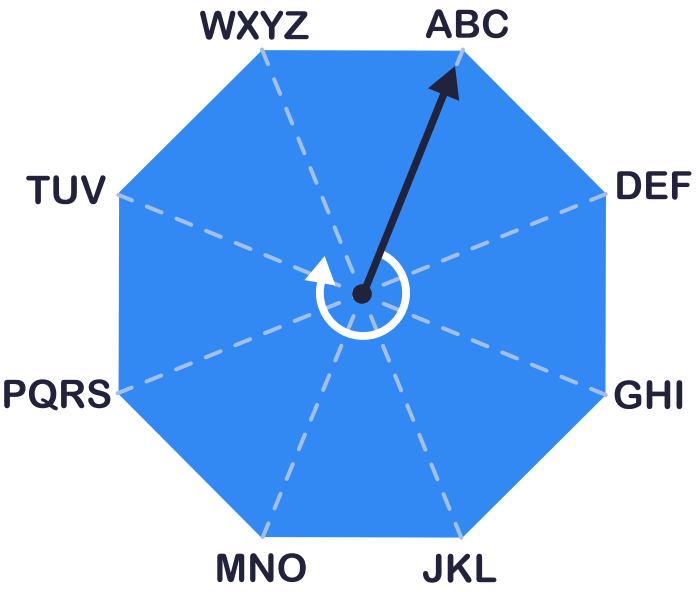
À un plus haut niveau, les portes logiques sont aussi utilisées en programmation : lorsque l'exécution d'une partie de programme dépend de plusieurs conditions, ces conditions peuvent être décrites par des combinaisons d'*opérateurs logiques* qui fonctionnent de la même manière que les portes logiques. On les trouve dans les programmes informatiques ; parfois, un ordinateur doit décider quelle action exécuter sur la base de ce qui a eu lieu précédemment.

Mots clés et sites web

- Variable booléenne : `https://fr.wikipedia.org/wiki/Booléen`
- Transistor : `https://fr.wikipedia.org/wiki/Transistor`
- Électronique numérique : `https://fr.wikipedia.org/wiki/Électronique_numérique`
- Porte ET : `https://fr.wikipedia.org/wiki/Fonction_ET`
- Porte OU exclusif : `https://fr.wikipedia.org/wiki/Fonction_NON-ET`
- Table de vérité : `https://fr.wikipedia.org/wiki/Table_de_vérité`
- Porte OU : `https://fr.wikipedia.org/wiki/Fonction_OU`
- Porte NON : `https://fr.wikipedia.org/wiki/Fonction_NON`
- Porte logique : `https://fr.wikipedia.org/wiki/Fonction_logique`

12. Code 8

Des textes en clair peuvent être chiffrés grâce au disque suivant :



Au départ, l'aiguille pointe sur « ABC ».

Chaque lettre est chiffrée individuellement. Pour cela, deux chiffres sont déterminés :

- Le premier chiffre indique de combien de positions l'aiguille doit être tournée dans le sens des aiguilles d'une montre pour qu'elle pointe le bloc contenant la lettre à chiffrer.
- Le deuxième chiffre indique la position de la lettre à chiffrer dans le bloc pointé.

Le cryptogramme du mot « CHAT », par exemple, est $03 - 22 - 61 - 61$.

Que signifie le cryptogramme 21-72-32-14 ?

A) GARS

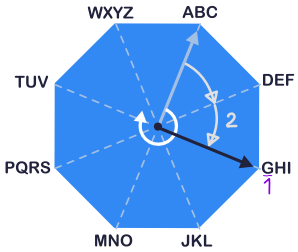
B) GENS

C) GEMIR

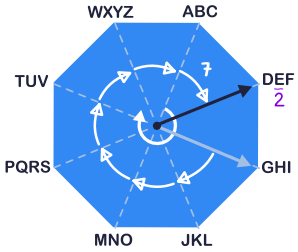
D) GELS

E) GENE

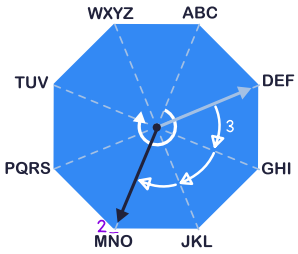
Solution



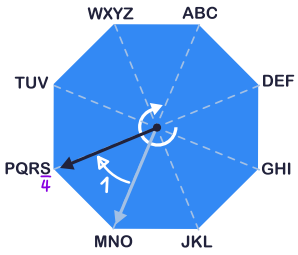
21 signifie que l'aiguille se déplace du bloc « ABC » au bloc « GHI » et que la première lettre, « G », est sélectionnée (le deuxième chiffre est 1).



72 signifie que l'aiguille se déplace du bloc « GHI » au bloc « DEF » et que la deuxième lettre, « E », est sélectionnée (le deuxième chiffre est 2).



32 signifie que l'aiguille se déplace du bloc « DEF » au bloc « MNO » et que la deuxième lettre, « N », est sélectionnée (le deuxième chiffre est 2).



14 signifie que l'aiguille se déplace du bloc « MNO » au bloc « PQRS » et que la quatrième lettre, « S », est sélectionnée (le deuxième chiffre est 4).

La réponse B) GENS est donc correcte.

Il existe aussi un moyen plus rapide de trouver la bonne réponse : la réponse C) GEMIR n'entre pas en question, car elle est composée de cinq lettres et le cryptogramme n'en contient que quatre. Le deuxième chiffre pour la quatrième lettre étant un 4, celle-ci ne peut être qu'un « S » ou un « Z ». Seules les réponses A), B) et D) remplissent cette condition. La lettre précédente doit venir du bloc situé à une position dans le sens inverse des aiguilles d'une montre du bloc « PQRS », donc du bloc « MNO ». Cela ne peut donc être que la réponse B) GENS.

C'est de l'informatique !

Depuis des milliers d'années, l'être humain cherche à cacher des informations afin que seul le destinataire ne puisse les déchiffrer. Ce qui commença avec des bouts de papier enroulés autour d'un bâton (scytale) se développa en la *cryptographie à clé publique* moderne (comme « GnuPG », qui utilise entre autres le chiffrement RSA) en passant par le chiffrement par transposition comme le « chiffre de César » et les *chiffrements polyalphabétiques* (comme le « chiffre de Vigenère »).

Le chiffrement de cet exercice est un chiffrement polyalphabétique, car une lettre n'est pas forcément toujours chiffrée la même chose : la lettre « A », par exemple, est chiffrée par 01 au début du texte en clair, mais par 31 si elle suit un « S ». Ces chiffrements peuvent tous être décryptés rapidement de nos jours à l'aide d'ordinateurs.

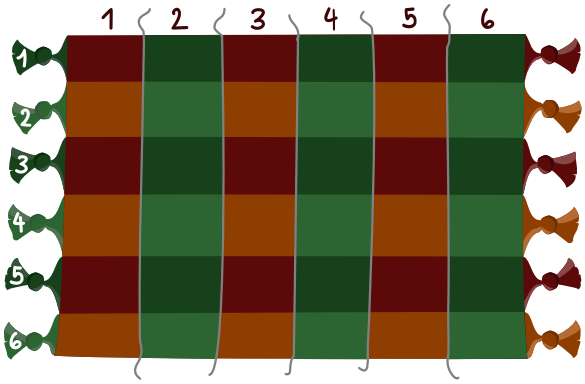
Dans ce cas, le déchiffrement est spécialement simple : il n'existe qu'une seule clé pour chiffrer un texte. Même si la position de départ de l'aiguille n'était pas toujours le bloc « ABC » mais un bloc au hasard, il n'y aurait que huit clés possibles… Même le chiffre de César, qui a plus de 2000 ans, est plus « sûr » que celui-ci ! On pourrait encore argumenter que le secret n'est pas la clé elle-même, mais le chiffrement. Mais le *principe de Kerckhoffs*, formulé par Auguste Kerckhoffs (1835 à 1903) en 1883 et encore valable aujourd'hui, montre que la sécurité d'un *cryptosystème* ne devrait pas se fonder sur la confidentialité d'une méthode de chiffrement, car celle-ci pourrait trop facilement être découvert par d'autres personnes.

Mots clés et sites web

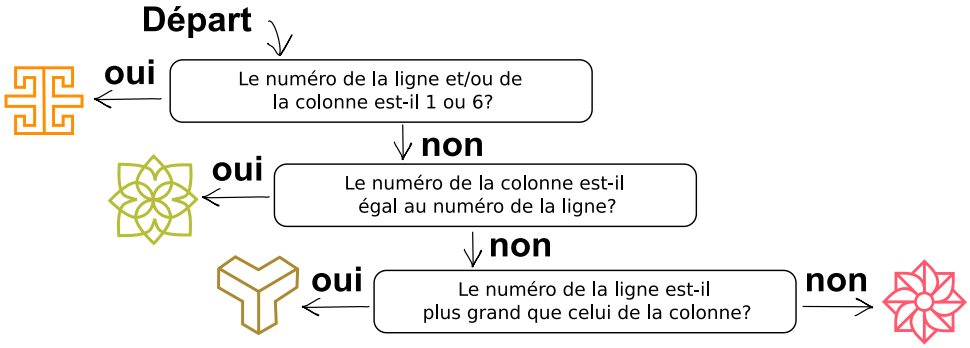
- Chiffre de César : `https://fr.wikipedia.org/wiki/Chiffrement_par_décalage`
- Substitution polyalphabétique/chiffre de Vigenère : `https://fr.wikipedia.org/wiki/Chiffre_de_Vigenère`
- Système de chiffrement : `https://fr.wikipedia.org/wiki/Chiffrement`
- Cryptographie à clé publique : `https://fr.wikipedia.org/wiki/Cryptographie_asymétrique`
- GnuPG : `https://fr.wikipedia.org/wiki/GNU_Privacy_Guard`
- Chiffrement RSA : `https://fr.wikipedia.org/wiki/Chiffrement_RSA`
- Principe de Kerckhoffs : `https://fr.wikipedia.org/wiki/Principe_de_Kerckhoffs`
- Auguste Kerckhoffs : `https://fr.wikipedia.org/wiki/Auguste_Kerckhoffs`
- Cryptosystème : `https://fr.wikipedia.org/wiki/Cryptosystème`
- Cryptographie : `https://fr.wikipedia.org/wiki/Cryptographie`

13. Tissage

Hale est une artiste turque. Elle définit le motif d'un tapis grâce à une grille de six colonnes et six lignes.

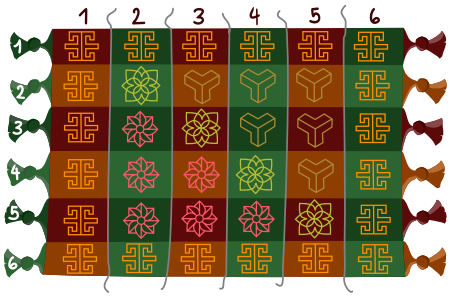


Hale numérote les lignes et les colonnes. Chaque case de la grille a donc un numéro de ligne et un numéro de colonne. Les employés de Hale doivent maintenant mettre un symbole dans chaque case. Pour cela, Hale leur a donné les instructions suivantes :

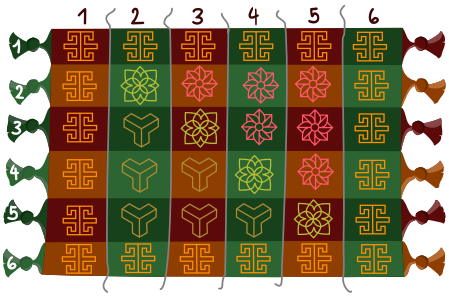


Comment sera le tapis une fois terminé ?

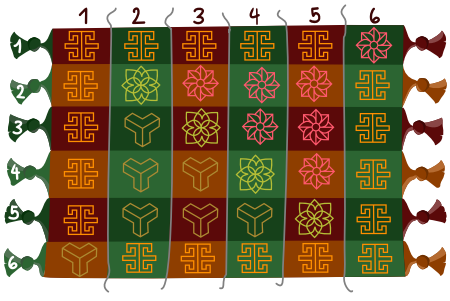
A)



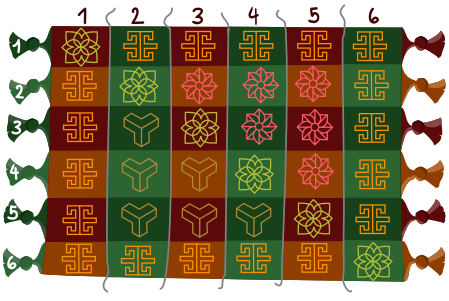
B)



C)



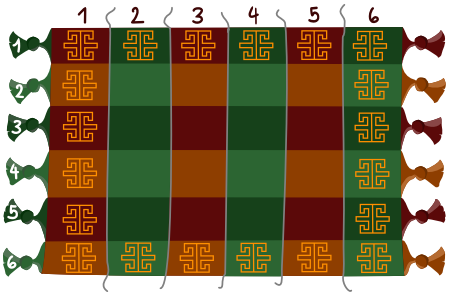
D)



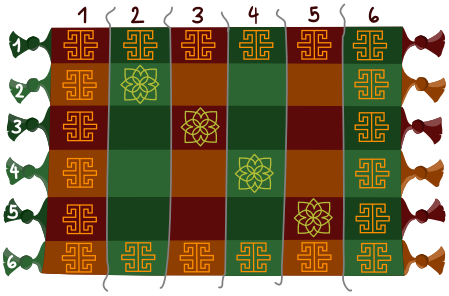
Solution

La bonne réponse est B).

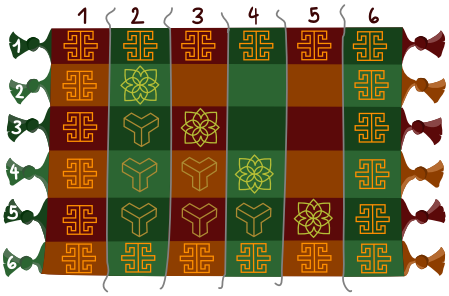
On répond à la première question de l'image d'instruction par « oui » pour toutes les cases au bord de la grille, car chaque case du bord se trouve dans la première ou sixième ligne et/ou colonne. Ces cases sont ornées du symbole  et le motif suivant en résulte :



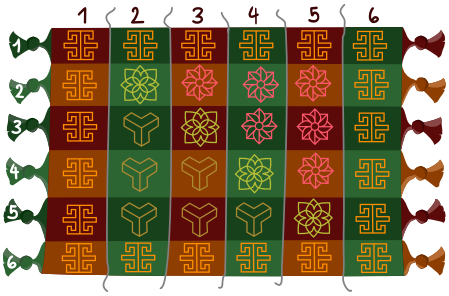
On répond à la deuxième question par « oui » pour toutes les cases de la diagonale, car les numéros de ligne et de colonne sont les mêmes sur la diagonale. Ces cases sont ornées du symbole  et le motif est le suivant :



La troisième question indique que toutes les cases dont le numéro de ligne est plus grand que le numéro de colonne doivent être ornées du symbole  .

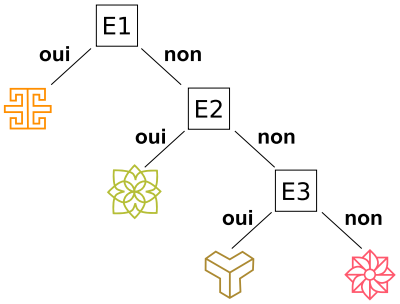


On répond « non » à la troisième question pour les cases restantes, car leur numéro de ligne n'est pas plus grand que leur numéro de colonne. Elles sont donc ornées du symbole  . On obtient ainsi le motif de la réponse B.



C'est de l'informatique !

L'image que l'artiste Hale a développé comme instruction s'appelle en informatique un *arbre de décision*. Comme un vrai arbre, il est composé de branchements. À chaque branchement (E1-E3) se trouve une question à laquelle on peut répondre par « oui » ou « non ». Si l'on parcourt l'arbre de haut en bas, répond aux questions et suit les lignes correspondante, une décision s'ensuit.



Dans l'exercice, l'arbre de décision est au cœur des instructions pour le tissage d'un tapis. Chaque personne qui suit ces instructions pour tisser fabrique exactement le même tapis. En principe, une machine pouvant lire et comprendre les instructions pourrait produire le même tapis.

En informatique, on appelle de telles instructions univoques un *algorithme*. Lorsqu'un algorithme est écrit dans un *langage de programmation* et peut être exécuté par un ordinateur, on parle d'un *programme informatique*.

On quotidien, on a souvent à faire à des programmes informatiques qui prennent des décisions : le contrôleur du feu de circulation décide quand le feu piéton devient vert ; le système d'exploitation du natel décide quand passer en mode d'économie d'énergie ; le contrôle automatique des passeports à l'aéroport décide si un passeport est valable.

Des arbres de décision se trouvent dans tous ces programmes informatiques.

Mots clés et sites web

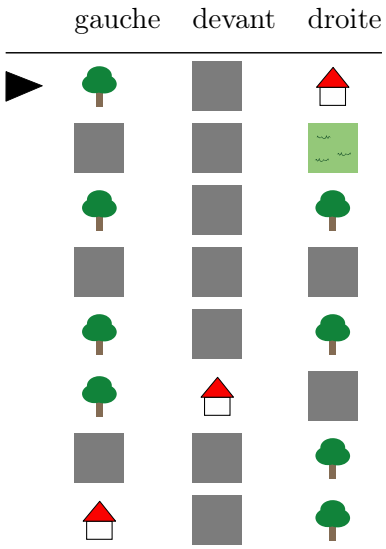
- Arbre de décision : `https://fr.wikipedia.org/wiki/Arbre_de_décision`
- Algorithme : `https://fr.wikipedia.org/wiki/Algorithme`
- Langage de programmation : `https://fr.wikipedia.org/wiki/Langage_de_programmation`
- Programme informatique : `https://fr.wikipedia.org/wiki/Programme_informatique`

14. Poste robotisée

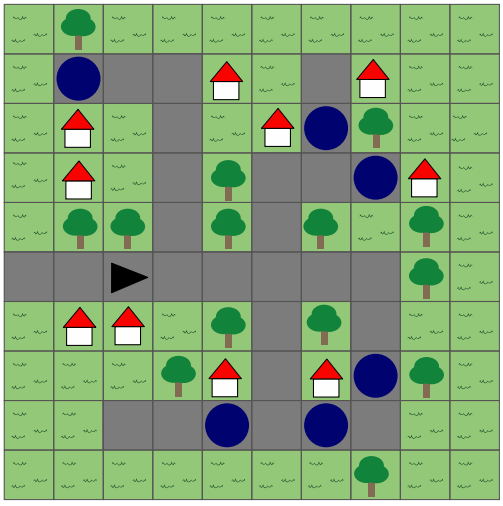
Tina le robot livre le courrier. Pour cela, elle utilise une carte du quartier, qui est divisée en cases. Tina se déplace le long de la rue de case en case en allant vers la droite, la gauche ou l'avant (pas en diagonale).

Tina a trois capteurs pour naviguer. Dès qu'elle arrive sur une case (et avant qu'elle ne puisse se tourner), les capteurs reconnaissent ce qui se trouve sur les cases à la droite, à la gauche et devant Tina.

Le contenu des cases reconnu par les capteurs de Tina sur son chemin est enregistré dans la table ci-dessous. Tina a commencé sur la case dans le sens de la flèche.

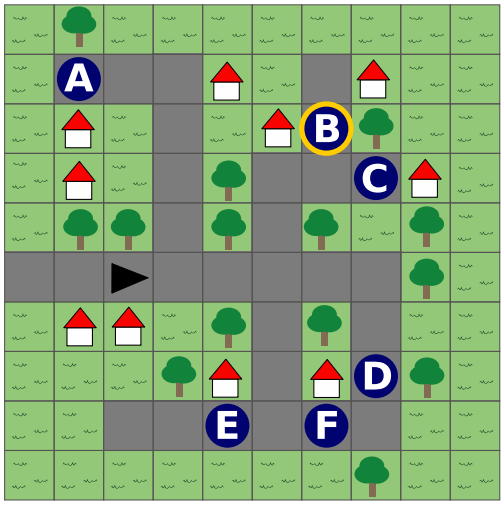


Sur quel point bleu Tina se trouve-t-elle à la fin de son chemin ?



Solution

La bonne réponse est le point B.

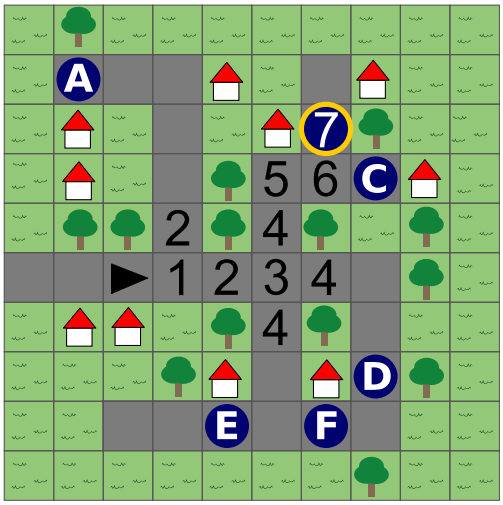


Étape	gauche	devant	droite
▶	arbre	mur	maison
1	mur	mur	herbe
2	arbre	mur	arbre
3	mur	mur	mur
4	arbre	mur	arbre
5	arbre	maison	mur
6	mur	mur	arbre
7	mur	arbre	arbre

Pour cet exercice, une mtéhode efficace consiste à se concentrer sur les six points d'arrivée et à vérifier si les données des capteurs pour l'étape 7 « maison mur arbre » pourraient y correspondre. On peut ainsi exclure les réponses C, E et F. Les données des capteurs pour l'étape 6 « mur mur arbre » permettent d'exclure les réponses A et D.

On pourrait également essayer de suivre le chemin enregistré dans la table. Les données ne correpondent qu'au chemin menant au point B.

On ne peut pas toujours décider tout de suite quel chemin Tina a suivi en suivant les informations des capteurs. À l'étape 4, Tina verrait des arbres à gauche et à droite quel que soit la direction dans laquelle elle est allée. Dans ce genre de situation, il faut prendre en compte les donnée des capteurs de l'étape suivante pour pouvoir déterminer le chemin exact de Tina.



C'est de l'informatique !

Dans cet exercice, nous rencontrons Tina le *robot*. Les robots sont des ordinateurs spécialement équipés qui utilisent des *capteurs* pour obtenir des informations sur leur environnement, traitent ces informations de manière automatisée (c'est-à-dire à l'aide d'un programme) et se basent sur leur environnement pour agir de manière autonome à l'aide d'*actionneurs*. Les capteurs de Tina receuillent le contenu des cases à sa gauche, à sa droite et devant elle. De manière concrète, on peut s'imaginer que les capteurs prennent des photos et que des données géométriques que l'ordinateur peut attribuer à une arbre, une maison ou une route sont extraites lors d'une analyse automatique. Les roues de Tina, ses actionneurs, peuvent donc être dirigés de manière à éviter les cases contenant des arbres ou une maison.

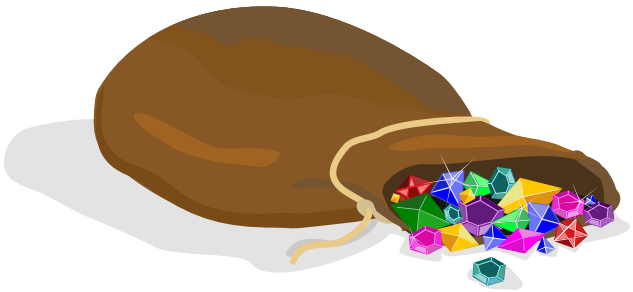
Les véhicules autonomes sont un exemple de tels robots. Ils sont équipés de nombreux capteurs qui ne mesurent pas seulement la vitesse et la position, mais aussi la distance au bord de la route, détectent les objets sur ou au bord de la route et encore beaucoup, beaucoup d'autres choses. Ces informations sont traitées à l'aide de programmes très complexes, qui peuvent par exemple reconnaître des enfants qui pourraient traverser la route et les distinguer de panneaux de circulation. Dans beaucoup de tels scénarios, l'*apprentissage automatique* est une technologie cruciale. Dans le cas des véhicules autonomes, les ordinateurs apprennent à l'aide de nombreux exemples donnés comment distinguer des enfants de panneaux de circulation. Les actionneurs sont par exemple les freins qui sont activés sans intervention humaine.

Mots clés et sites web

- Robot : `https://fr.wikipedia.org/wiki/Robot`
- Capteur : `https://fr.wikipedia.org/wiki/Capteur`
- Actionneur : `https://fr.wikipedia.org/wiki/Actionneur`
- Apprentissage automatique : `https://fr.wikipedia.org/wiki/Apprentissage_automatique`

15. Pierres précieuses

Peter a plusieurs pierres précieuses. Elles ont toutes une valeur différente.



Sarah connaît les pierres précieuses de Peter, mais pas leur valeur. Elle aimerait savoir quelle pierre a le plus de valeur.

Pour cela, elle fait la chose suivante trois fois de suite :

- Elle choisit quatre pierres parmi la collection de Peter et lui demande laquelle des quatre a le plus de valeur.

Elle choisit à chaque étape quatre pierres comme elle veut, et Peter lui répond toujours honnêtement.

À la fin, Sarah sait quelle pierre précieuse a le plus de valeur.

Combien de pierres précieuses Peter peut-il avoir au maximum ?

A) 8 pierres précieuses

B) 10 pierres précieuses

C) 11 pierres précieuses

D) 12 pierres précieuses

Solution

La réponse B) : 10 pierres précieuses est correcte.

Si Peter a 10 pierres précieuses, Sarah peut choisir en tout huit pierres différentes lors de ses deux premières questions. Les deux pierres « gagnantes » de chaque question (c'est à dire le deux qui ont la plus grande valeur parmi les deux groupes de quatre) peuvent chacune être la « grande gagnante », c'est à dire la pierre ayant la plus grande valeur de toutes. Les six autres pierres comparées sont éliminées. Pour sa dernière question, Sarah compare les deux gagnantes et les deux pierres qu'elle n'a pas encore choisi jusque-là. La gagnante de la dernière question est aussi la grande gagnante.

Sarah peut donc (entre autres) procéder de cette façon pour trouver la pierre ayant la plus grande valeur parmi 10 pierres. Si Peter a 11 pierres, cela ne fonctionne pas :

Si Sarah procède comme décrit plus haut et compare huit pierres différentes lors de ses deux premières questions, il lui reste ensuite les deux gagnantes et trois autres pierres à comparer pour trouver la grande gagnante, donc trop de pierres pour sa troisième question. Si Sarah décide de comparer la gagnate de sa première question avec trois nouvelles pierres lors de sa deuxième question, cela lui apprend quelle pierre a le plus de valeur parmi les sept pierres comparées. Elle doit ensuite encore comparer cette pierre avec les quatre autres pierres qu'elle n'a pas encore choisies, ce qui fait une pierre de trop pour sa troisième question.

Si Sarah choisissait six pierres ou moins parmi 11 pour ses deux premières questions, ou si Peter avait plus de 12 pierres, elle aurait besoin d'encore plus de questions pour déterminer laquelle a le plus de valeur.

C'est de l'informatique !

Cet exercice présente un *algorithme* qui est limité par des conditions. Ici, ces conditions sont que Sarah ne peut poser que trois questions et que chaque question ne peut contenir que quatre éléments.

Malgré cette limite, l'algorithme fonctionne bien pour des collections de moins de 11 éléments, mais échoue pour de plus grandes collections.

Il existe plusieurs raisons d'imposer des contraintes à un algorithme. On pourrait par exemple imposer qu'une opération soit effectuée en un temps limite, ce qui est nécessaire pour les systèmes d'exploitation temps réel. De plus, certains procédés peuvent engendrer des coûts externes ou endommager des composants.

Ce n'est pas un problème que l'algorithme échoue à partir d'un certain seuil du moment que l'on contrôle que ce seuil ne soit jamais dépassé. La stratégie limitée de cet exercice ne doit par exemple jamais être utilisée pour des collections de plus de 10 éléments.

Mots clés et sites web

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

- Complexité en temps : `https://fr.wikipedia.org/wiki/Complexité_en_temps`

A. Auteur·e·s des exercices

Gulgun Afacan
Esraa Almajhad
Waël Almoman
Leo Barichello
Liam Baumann
Wilfried Baumann
Linda Björk Bergsveinsdóttir
Tobias Berner
Sarah Chan
Byeonggyu Cho
Christian Datzko
Susanne Datzko
Justina Dauksaite
Nora A. Escherle
Gerald Futschek
Mark Edward M. Gonzales
Adam Grodeck
Yasemin Gülbahar
Benjamin Hirsch
Alisher Ikramov
Dauksaite Justina
Dong Yoon Kim
Hakin Kim
Jihye Kim
Seulki Kim
Vaidotas Kinčius
Lidija Kralj
Regula Lacher
Taina Lehtimäki
Karolína Miková
Jelena Milojkovic
Ágnes Erdősné Németh
Elsa Pellet
Jean-Philippe Pellet
Margot Phillipps
Zsuzsa Pluhár
Wolfgang Pohl
John-Paul Pretti
Le Quang Quan
Susannah Quidilla
Chris Roffey
Kirsten Schlüter
Giovanni Serafini
Yeh Yi Shan
Bernadette Spieler
Alieke Stijf
Goran Sukovic
Monika Tomcsányiová
Ahto Truu
Troy Vasiga
Michael Weigend
Kyra Willekes

B. Sponsoring : Concours 2022

HASLERSTIFTUNG

http://www.haslerstiftung.ch/

Kanton Zürich
Volkswirtschaftsdirektion
Amt für Wirtschaft und Arbeit

Standortförderung beim Amt für Wirtschaft und Arbeit Kanton Zürich

UBS

http://www.ubs.com/

verkehrshaus.ch

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

i-factory (Musée des transports, Lucerne)

senarclens
leu+partner
strategische kommunikation

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

ABZ
AUSBILDUNGS- UND BERATUNGSZENTRUM
FÜR INFORMATIKUNTERRICHT

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

hep/ haute école pédagogique vaud

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

Scuola universitaria professionale
della Svizzera italiana
SUPSI

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

C. Offres supplémentaires

IT tout feu tout flamme : `https://it-feuer.ch/fr/`
En Suisse, un nombre considérable d'organisations s'engagent à promouvoir la prochaine génération d'informaticiennes et d'informaticiens. L'initiative « IT tout feu tout flamme » souhaite unir ces forces et contribuer ensemble à mieux faire connaître le sujet au public dans toute la Suisse. IT tout feu tout flamme présente une variété d'offres destinées au corps enseignant et aux élèves.

Coding club des filles :
`https://www.epfl.ch/education/education-and-science-outreach/fr/jeunepublic/coding-club/`
Programmer une application ? Inventer un jeu vidéo ? Créer une animation ? Si une de ces activités t'intéresse, cet espace est fait pour toi ! Viens échanger et partager tes idées, apprendre à coder et découvrir les métiers liés à l'informatique. Les filles de 11 à 15 ans intéressées par la programmation et l'informatique peuvent participer aux ateliers du Coding club des filles.

Roteco : `https://www.roteco.ch/fr/`
Le projet Roteco existe autour d'une communauté d'enseignantes et enseignants qui souhaitent préparer leurs élèves à évoluer dans une société numérique. Au sein de cette communauté, ils cherchent, testent, développent et partagent des activités de robotique éducative et de science informatique adaptées pour leurs classes. Ils sont informés des derniers événements ou ateliers concernant la robotique et plus largement des activités de science informatique à proximité de leur établissement.

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.