



**INFORMATIK-BIBER SCHWEIZ
 CASTOR INFORMATIQUE SUISSE
 CASTORO INFORMATICO SVIZZERA**

Exercices et solutions 2022

Années HarmoS 5/6

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

Éditeurs :

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

010100110101011001001001
 010000010010110101010011
 010100110100100101000101
 001011010101001101010011
 010010010100100100100001

SS!E

www.svia-ssie-ssii.ch
 schweizerischerverein für informatik in
 erausbildung // société suisse pour l'infor
 matique dans l'enseignement // società sviz
 zera per l'informatica nell'insegnamento



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, Ausbildungs- und Beratungszentrum für Informatikunterricht

Tobias Berner : Pädagogische Hochschule Zürich

Waël Almoman : Collège Voltaire

Le choix des exercices a été fait en collaboration avec les organisateurs de Bebras en Allemagne, Autriche, Hongrie, Slovaquie et Lituanie. 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 1^{er} décembre 2022.



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. 47.



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.bebas.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 fautive 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 fautive	-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/>

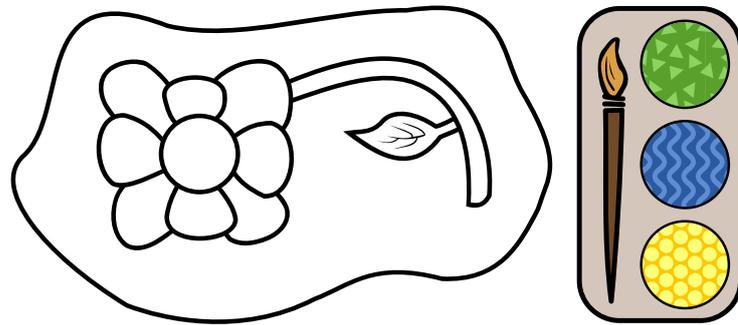


Table des matières

Ont collaboré au Castor Informatique 2022	i
Préambule	iii
Table des matières	v
1. Coloriage	1
2. Bonbons préférés	5
3. Bibliothèque	7
4. Ruche	11
5. Permutations	15
6. Le lièvre et la tortue	19
7. Pyramide colorée	23
8. Recette de hamburger	27
9. Collier de marin	31
10. Cœur composé	35
11. Boulons et écrous	39
12. Les voisins de Lili	43
A. Auteur-e-s des exercices	47
B. Sponsoring : Concours 2022	48
C. Offres supplémentaires	49



1. Coloriage

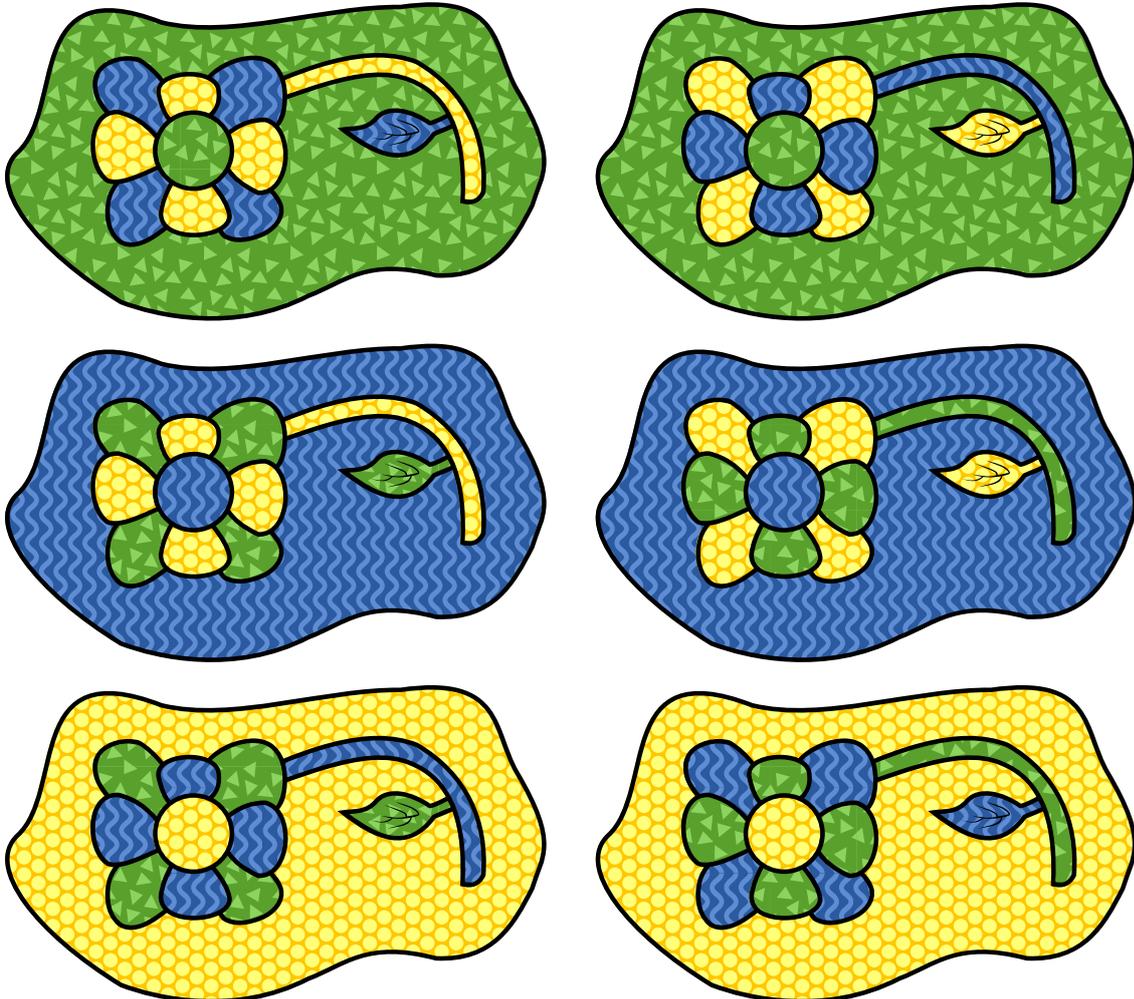


Colorie l'image en vert, jaune et bleu de manière à ce que deux surfaces de la même couleur ne se touchent jamais.



Solution

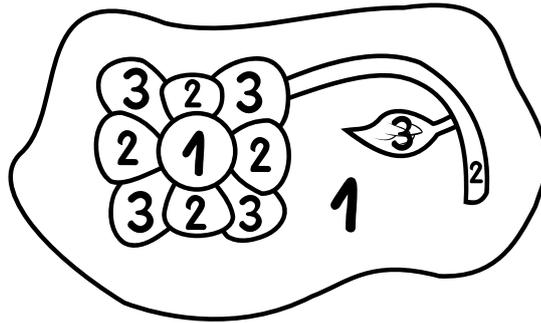
Voici toutes les possibilités de colorier l'image.



Comment peux-tu trouver une solution ?

Commence par choisir une couleur pour la surface extérieure, par exemple du jaune. Presque toutes les autres surfaces touchent la surface jaune. Celles-ci doivent donc être coloriées en vert ou en bleu. Commence par une de ces surfaces et alterne les couleurs. Le centre de la fleur peut être jaune comme la surface extérieure.

Cette image numérotée décrit la stratégie indépendamment des couleurs :



C'est de l'informatique !

Dans cet exercice, il s'agit d'attribuer des couleurs à des surfaces en respectant certaines contraintes. En informatique et en mathématiques, ce type de problème est connu sous le nom de *coloration de graphe*.

Les problèmes de coloration de graphe ont beaucoup d'applications pratiques et il est souvent important d'utiliser le moins de couleurs possibles. La répartition des équipes pour un tournoi sportif, de personnes dans des groupes ou encore l'attribution de fréquences à des chaînes de radio en sont des exemples. Les applications pratiques de coloration de graphes présentent souvent plus de surfaces que cet exercice, et il n'est souvent plus possible de trouver une solution avec peu de couleurs à la main. Les informaticien·nes utilisent des ordinateurs pour résoudre de tels problèmes.

La coloration de graphe peut aussi être utilisée pour des cartes géographiques. Le but est alors de trouver une coloration avec laquelle deux pays voisins n'ont jamais la même couleur. Une coloration à quatre couleurs existe pour n'importe quelle carte géographique. Ceci n'est pas facile à démontrer : c'est seulement en 1976 que les mathématiciens Kenneth Appel et Wolfgang Haken y sont arrivés. Pour cela, ils ont utilisé des ordinateurs pour tester un grand nombre d'exceptions et de contre-exemples. D'autres personnes ne peuvent pas le vérifier sans ordinateur. C'est pour cela que certain·es autres mathématicien·nes voient l'utilisation d'ordinateur pour cette démonstration ou pour des démonstrations en général d'un œil critique.

Mots clés et sites web

- Coloration de graohe : https://fr.wikipedia.org/wiki/Coloration_de_graphe
- Théorème des quatre couleurs :
https://fr.wikipedia.org/wiki/Théorème_des_quatre_couleurs





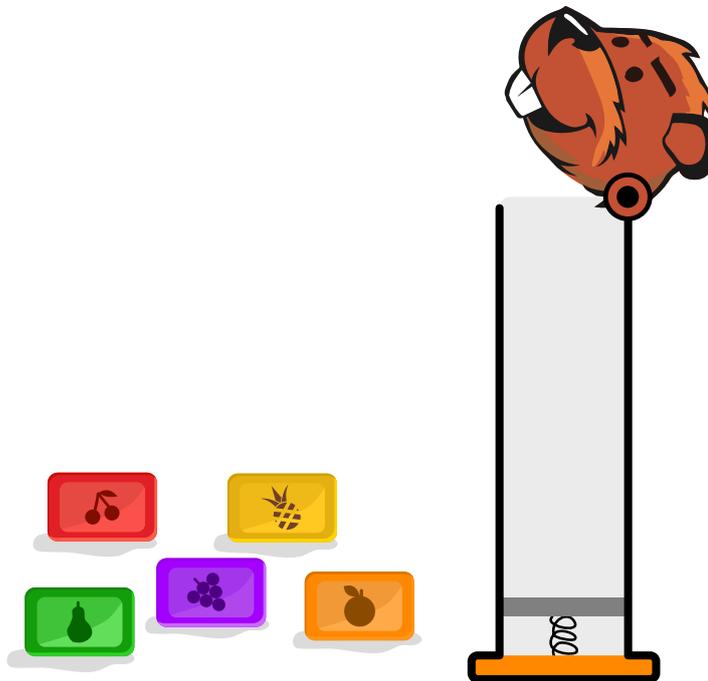
2. Bonbons préférés

Anna met cinq bonbons dans un distributeur. Elle peut ensuite les manger les uns après les autres dans l'ordre dans lequel ils sortent du haut du distributeur.

Elle aimerait les manger dans cet ordre :



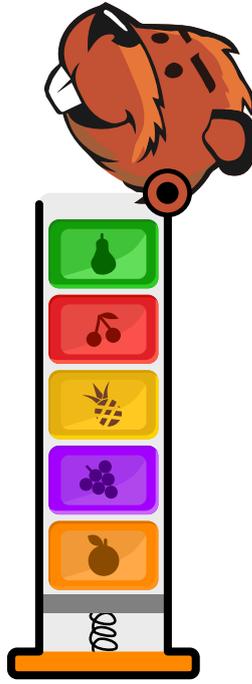
Dans quel ordre doit-elle les mettre dans le distributeur ?





Solution

Pour que les différents bonbons sortent du distributeur dans le bon ordre, c'est important de comprendre que le bonbon qui y a été mis en premier en sort en dernier. Cela veut dire que le distributeur doit être rempli de la manière suivante :



C'est de l'informatique !

Si Anna met les bonbons dans le distributeur dans le même ordre que celui dans lequel elle veut les manger, ils en ressortent exactement dans l'ordre inverse. C'est la raison pour laquelle elle décide d'abord dans quel ordre elle veut les manger, puis elle se représente comment elle doit remplir le distributeur pour obtenir les bonbons dans le bon ordre.

Pour les informaticiennes et informaticiens, c'est souvent important de réfléchir à l'ordre des choses, comme pour Anna. L'ordre utilisé dans cet exercice s'appelle « ordre de pile ». Les piles sont des structures de stockage dans lesquelles les objets peuvent être ajoutés et desquelles ils peuvent être sortis dans un ordre précis. Elles fonctionnent d'après le principe *dernier arrivé, premier sorti* (LIFO de l'anglais « Last in, first out »), ce qui signifie que l'objet qui a été ajouté à la pile en dernier doit en être sorti en premier.

Mots clés et sites web

- Pile : [https://fr.wikipedia.org/wiki/Pile_\(informatique\)](https://fr.wikipedia.org/wiki/Pile_(informatique))
- Dernier arrivé, premier sorti (LIFO) :
https://fr.wikipedia.org/wiki/Last_in,_first_out



3. Bibliothèque

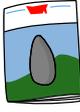
Les enfants empruntent des livres à la bibliothèque. La bibliothécaire note dans une table quel enfant a emprunté quel livre.

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











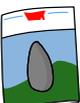

Solution



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.

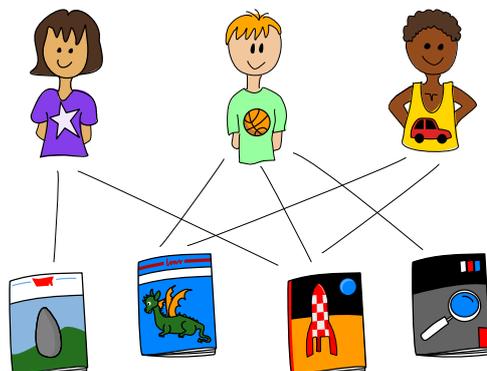
			
		3x	
		1x	
		2x	
		1x	
			
			
			

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 déjà 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 adresse 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

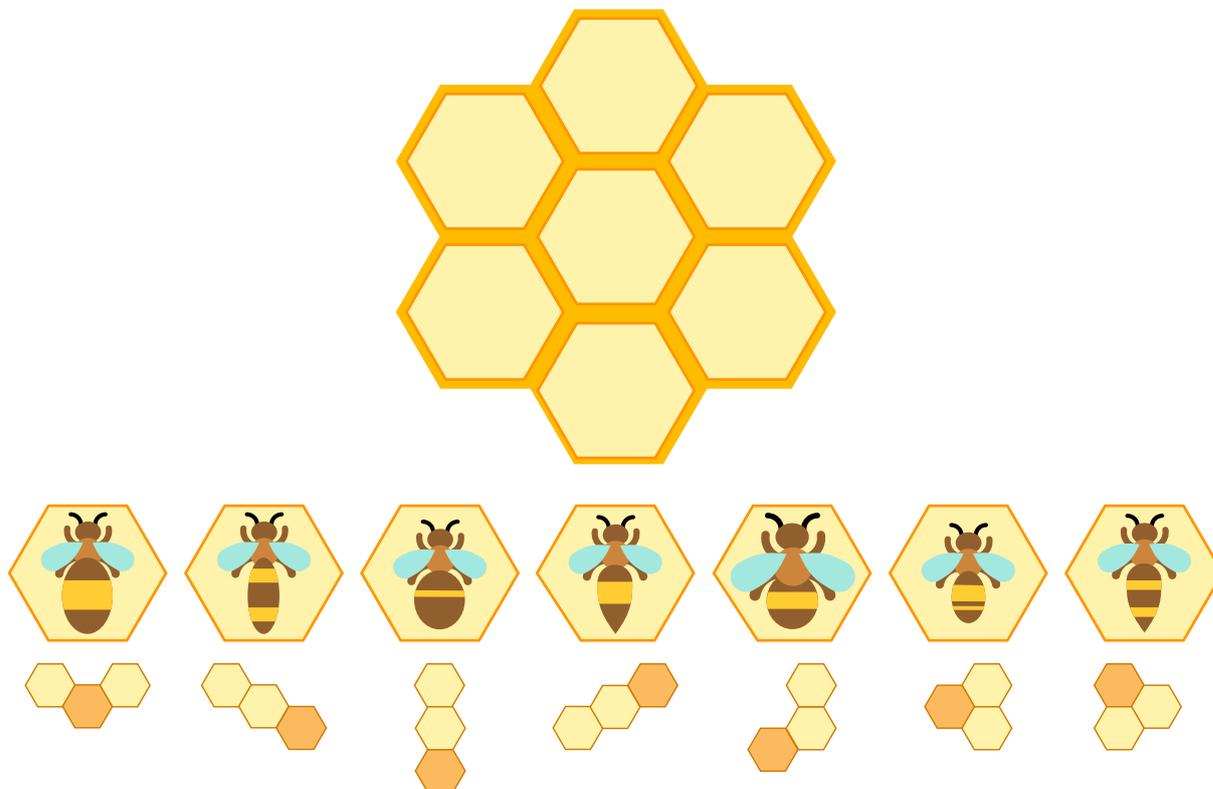
- Base de données relationnelle :
https://fr.wikipedia.org/wiki/Base_de_données_relationnelle





4. Ruche

Un castor a besoin d'aide pour loger toutes les abeilles dans sa ruche.



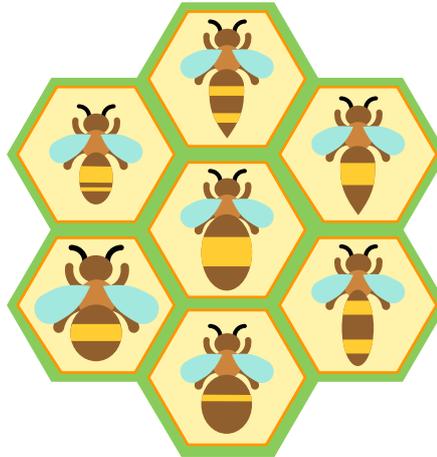
Sous chaque abeille, un dessin illustre la règle déterminant dans quel alvéole elle peut loger.

Loge les abeilles dans la ruche en respectant les règles illustrées sous les abeilles.

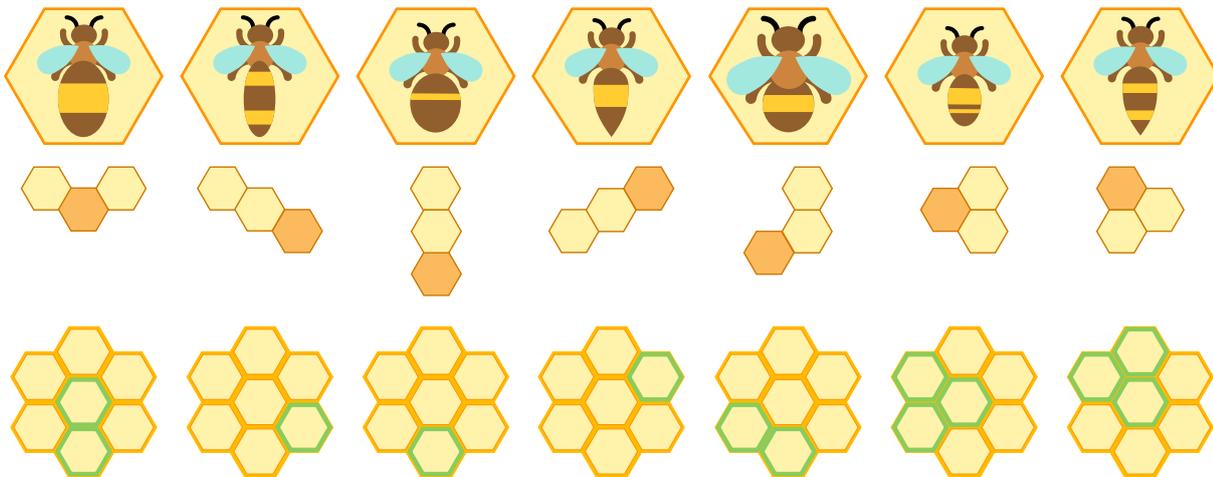


Solution

La seule manière de loger les abeilles est la suivante :



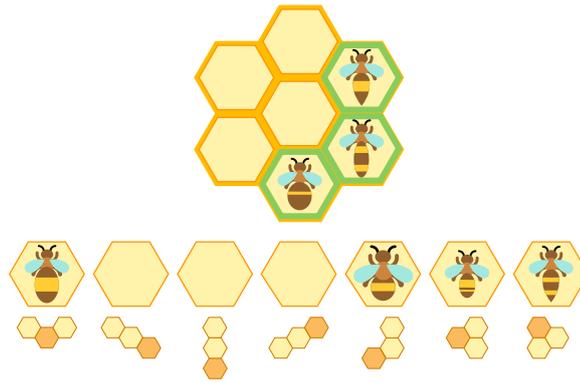
On peut résoudre l'exercice en essayant différentes solutions, mais cela peut prendre beaucoup de temps. Pour trouver une méthode plus rapide, observe attentivement les règles pour chaque abeille. Dans l'image ci-dessous, tu vois chaque abeille et la règle qui correspond. Les alvéoles dans lesquels les abeilles peuvent être logées d'après les règles sont entourés en vert.



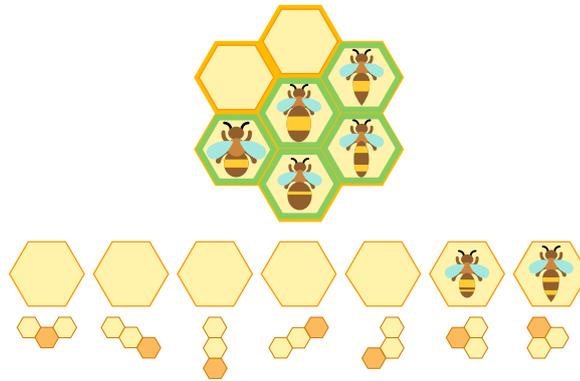
Tu vois que les règles permettent de loger certaines abeilles dans plusieurs alvéoles, et certaines autres pas. Trois abeilles ne peuvent être logées que dans un seul alvéole.

Pour résoudre l'exercice plus rapidement qu'en essayant plusieurs possibilités, on peut procéder comme suit :

Loge d'abord les abeilles qui ne peuvent loger que dans un seul alvéole.



Il ne reste alors plus qu'une possibilité pour les deux abeilles suivantes.



On loge les deux dernières abeilles l'une après l'autre de la même manière.

C'est de l'informatique !

Dans cet exercice, il faut loger sept abeilles dans sept alvéoles différents. Il y a beaucoup de possibilités de loger les abeilles. Le nombre de possibilités diminue déjà beaucoup si on prend les règles en compte, mais demanderait quand même encore beaucoup de travail. La clé pour arriver à résoudre l'exercice rapidement est de loger les abeilles dans le bon ordre. Dans notre cas, pour limiter le nombre de cas à prendre en compte, nous commençons avec les éléments les plus limités, c'est-à-dire les abeilles qui ne peuvent loger que dans un seul alvéole.

Une telle approche s'appelle une *heuristique* en informatique. La bonne solution a pu être trouvée en peu d'étapes en utilisant un ordre de résolution adapté. Pour certains problèmes, comme la planification d'un itinéraire entre différents endroits par un système de navigation, l'utilisation d'une heuristique implique une perte d'exactitude. En effet, il existe énormément de solutions. Pour s'assurer de trouver la meilleure solution, tous les itinéraires possibles sur l'ensemble du réseau routier doivent être calculés et comparés, ce qui demanderait énormément de calculs. En commençant par calculer les itinéraires qui correspondent probablement à de bonnes solutions, on peut fortement réduire la quantité de calculs nécessaire. On peut ainsi trouver un bon itinéraire en quelques secondes au lieu de trouver le meilleur en plusieurs années.



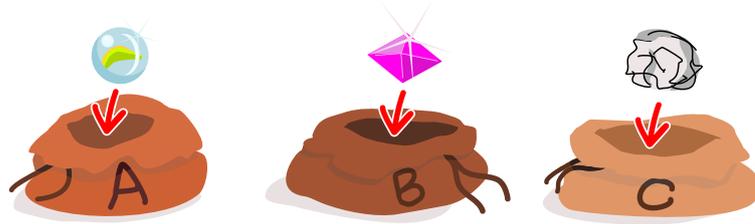
Mots clés et sites web

- Heuristique: [https://fr.wikipedia.org/wiki/Heuristique_\(mathématiques\)#Heuristique_au_sens_de_l'algorithmique](https://fr.wikipedia.org/wiki/Heuristique_(mathématiques)#Heuristique_au_sens_de_l'algorithmique)
- Problème du plus court chemin :
https://fr.wikipedia.org/wiki/Problème_de_plus_court_chemin
- Problème du voyageur de commerce :
https://fr.wikipedia.org/wiki/Problème_du_voyageur_de_commerce

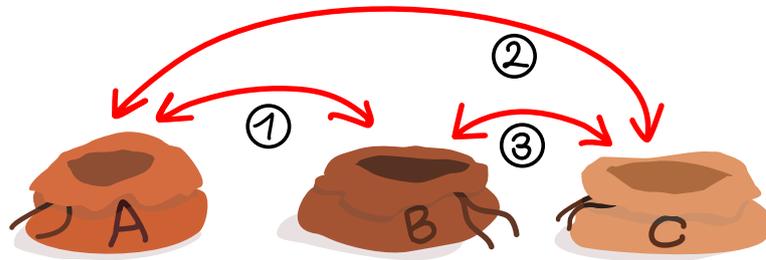


5. Permutations

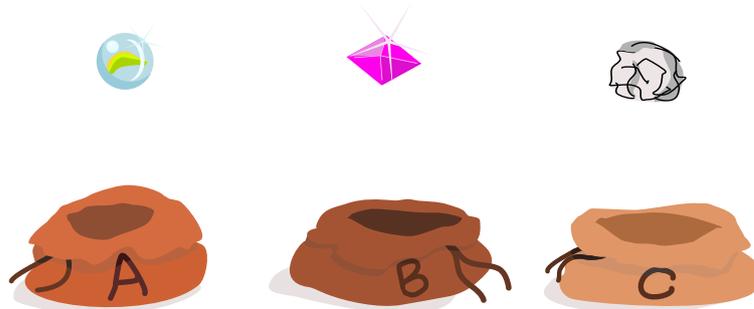
Lila met une bille dans le sac A, une pierre précieuse dans le sac B et un bout de papier dans le sac C.



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 :



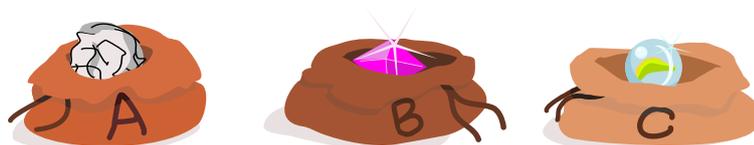
Lila échange les objets trois fois. Après le premier échange (A-B), les objets sont répartis dans les sacs comme ceci :



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 différents 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 précieuse

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

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

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

Mots clés et sites web

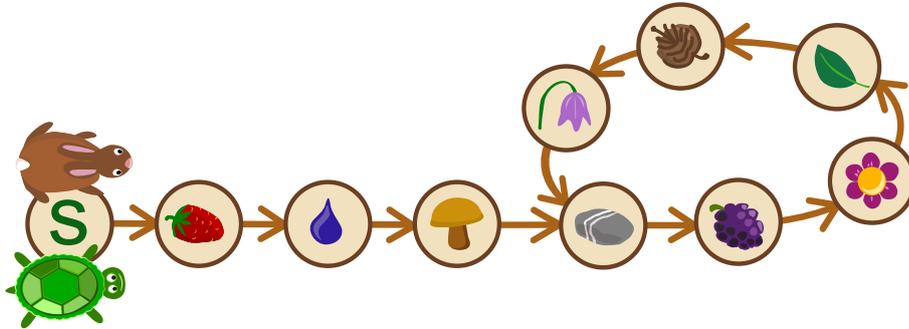
- Permutations : <https://fr.wikipedia.org/wiki/Permutation>





6. 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 ?

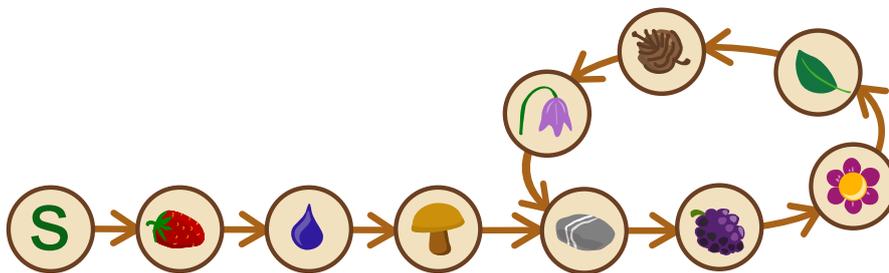


Solution

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 :

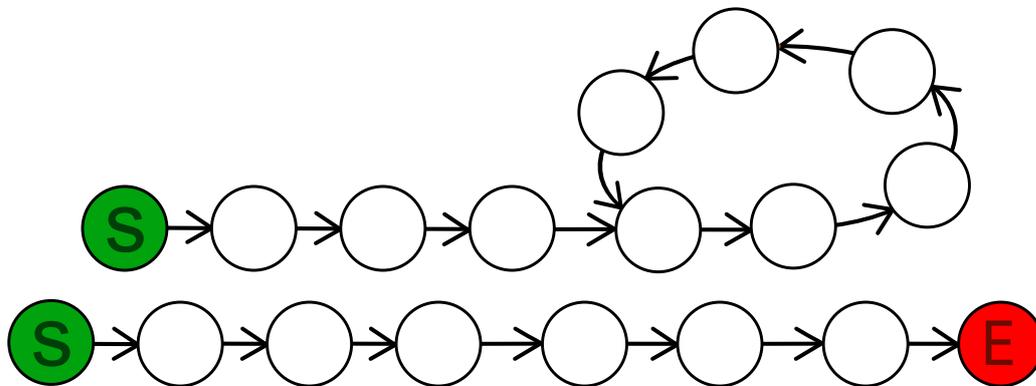
Min. après le départ	0	1	2	3	4	5	6	7	8	9	10	11	12	13	...
	S														...
	S														...



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_\(théorie_des_graphes\)](https://fr.wikipedia.org/wiki/Cycle_(théorie_des_graphes))
- Nœud: [https://fr.wikipedia.org/wiki/Sommet_\(théorie_des_graphes\)](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>

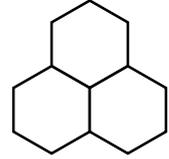




7. Pyramide colorée

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 :

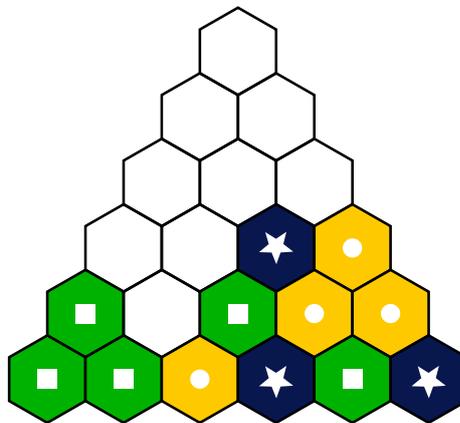


- soit tous la même couleur,
- soit trois couleurs différentes.

Sami trouve cela joli comme ça.

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

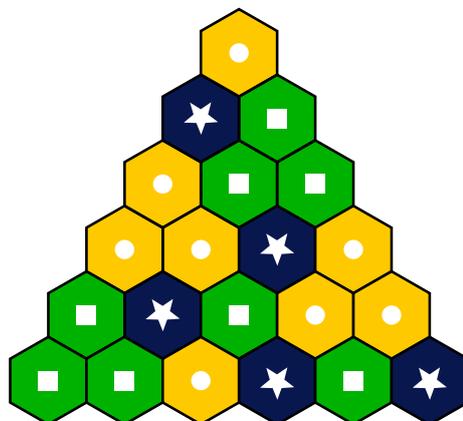
Colorie le reste des hexagones comme Sami aime.





Solution

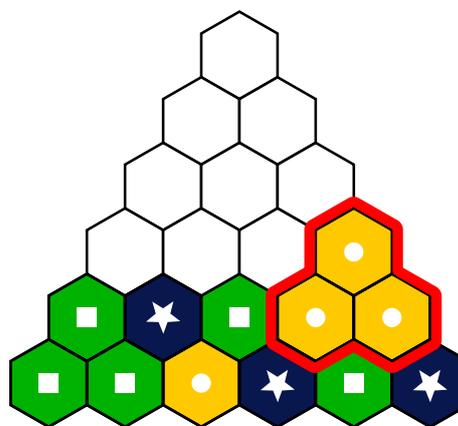
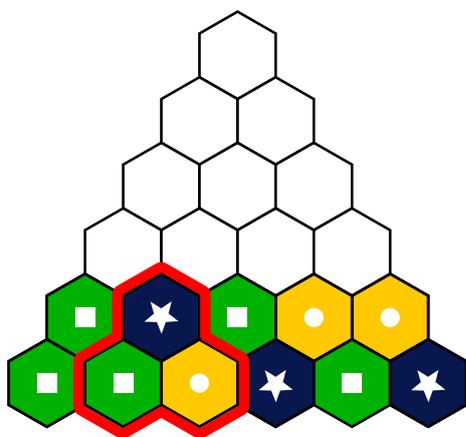
Voici la bonne solution :



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_\(programmation\)](https://fr.wikipedia.org/wiki/Instruction_conditionnelle_(programmation))
- Boucle : https://fr.wikipedia.org/wiki/Structure_de_contrôle#Boucles



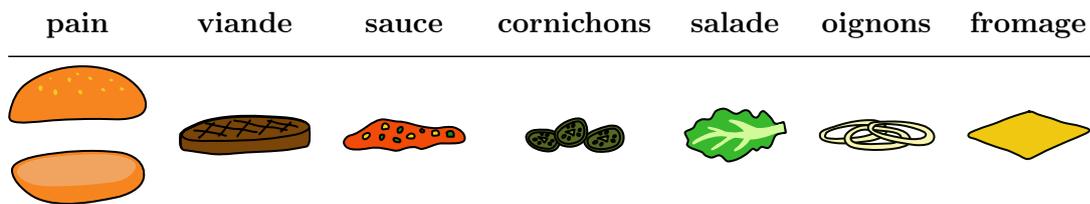


8. Recette de hamburger

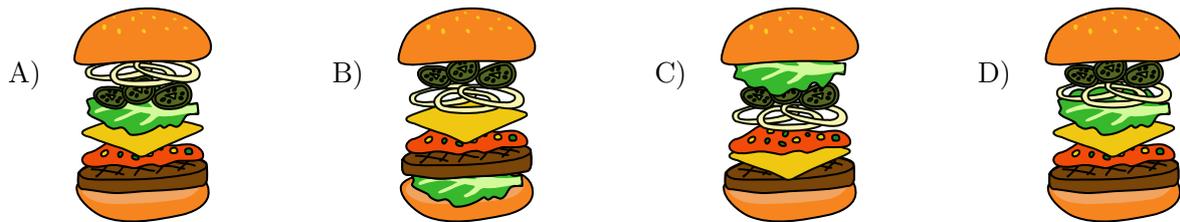
Le castor Jess prépare des hamburgers. Il suit pour cela trois règles :

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 :



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





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_\(complexité\)](https://fr.wikipedia.org/wiki/NP_(complexité))

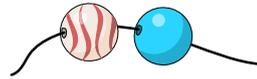




9. Collier de marin

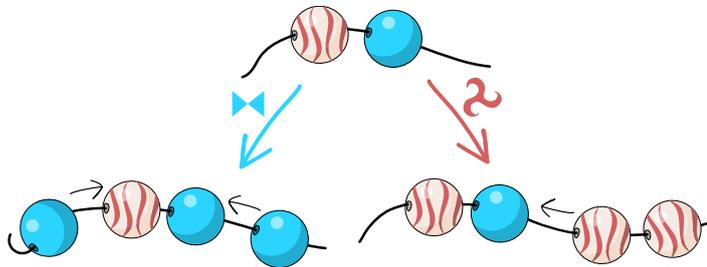
Voici les instructions de Monica pour faire son collier de marin avec des perles blanches à vagues rouges et des perles unies bleues.

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 (↷).



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

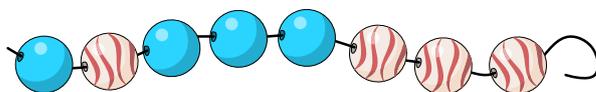
Lequel des colliers suivants n'est **pas** un collier de marin de Monica ?

- 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 être 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 deque 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



vérifier 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

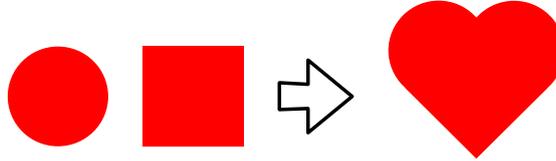
- deque : https://fr.wikipedia.org/wiki/File_d'attente_à_double_extrémité





10. 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, *déplacer* 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, déplacer 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





11. 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 :

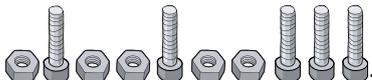
1. Ben prend un boulon sur la ligne de montage, mais il n'y a pas d'écrou à visser dessus dans le seau.
2. Ben a pris toutes les pièces sur la ligne de montage, mais il reste des écrous dans le seau.

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 ?

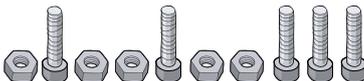
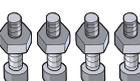
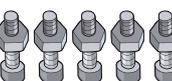
- A)
- B)
- C)
- D)



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.

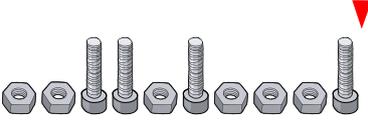
Boîte	Seau	Ligne de montage
<i>vide</i>	<i>vide</i>	
<i>vide</i>		
	<i>vide</i>	
		
		
		
		
		
		
		
	<i>vide</i>	<i>vide</i>

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



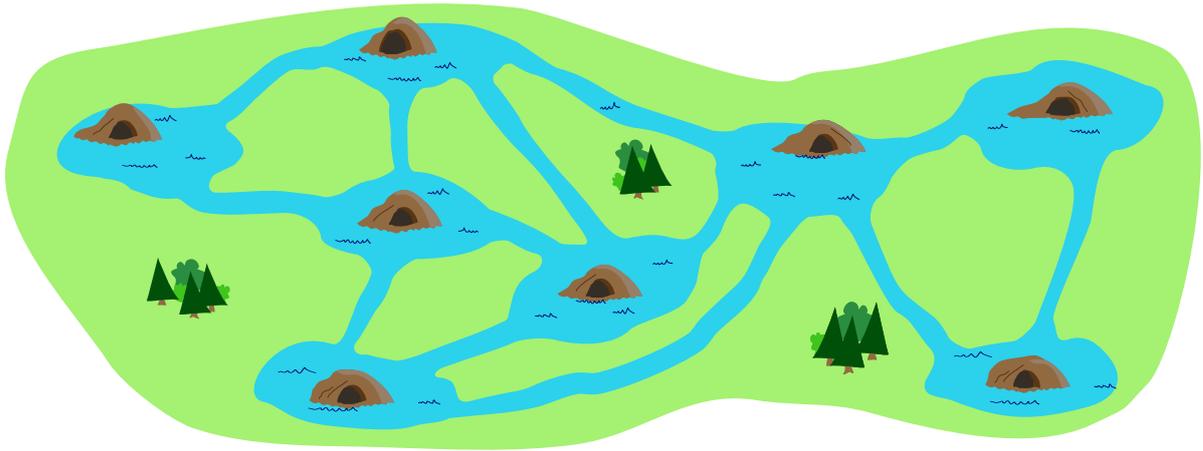


12. Les voisins de Lili

Tu vois sur la carte les huttes de huit castors. Deux castors sont voisins lorsqu'un canal relie leurs huttes.

- Lili, Simon, et Pierre ont quatre voisins chacun.
- Simon et Pierre sont les seuls voisins de Nina.

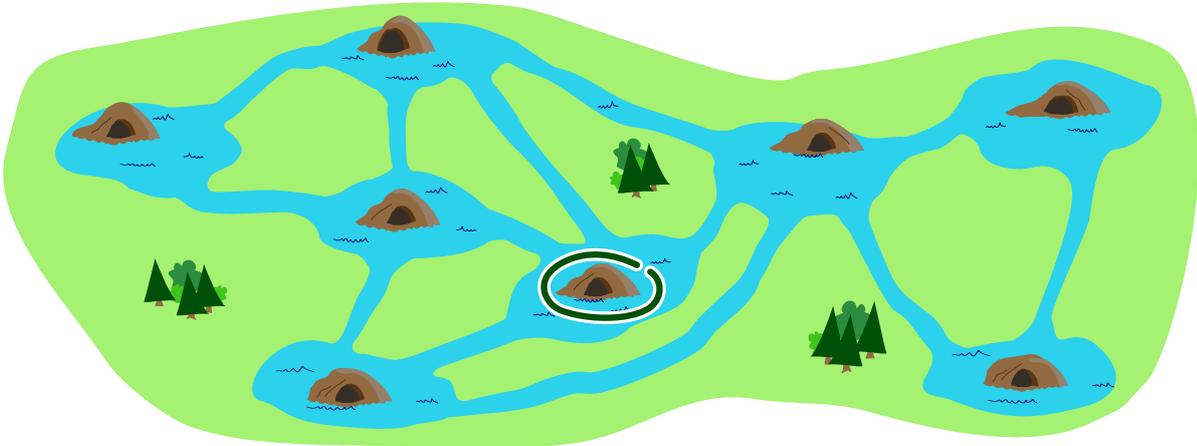
Dans quelle hutte Lili habite-t-elle ?



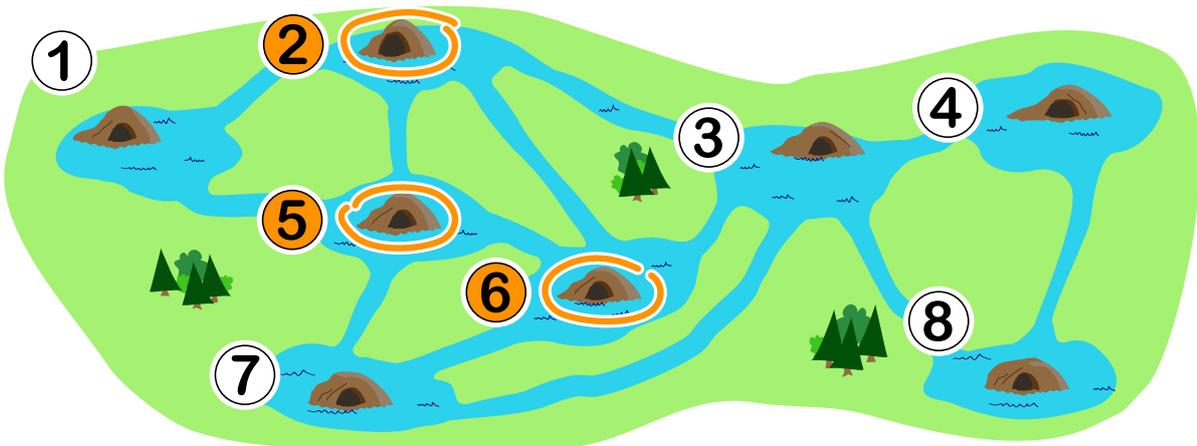


Solution

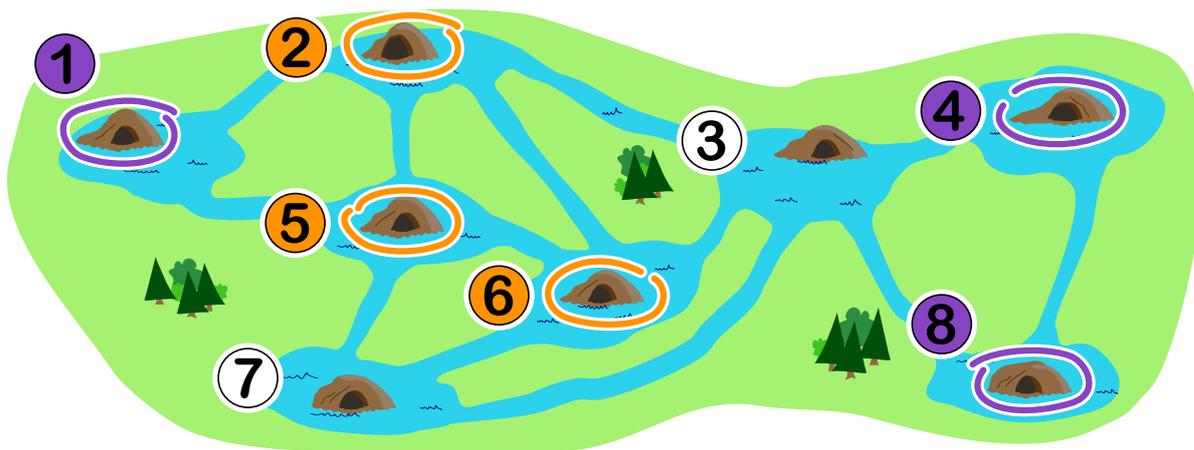
La bonne réponse est :



Pour résoudre le problème, il faut se concentrer sur les canaux entre les fortifications. Nous devons identifier les huttes dans lesquelles peuvent habiter Lili, Pierre et Simon. Comme ils ont chacun quatre voisins, exactement quatre canaux doivent rejoindre chacune de leurs huttes. Il y a trois huttes qui remplissent ces conditions : les huttes 2, 5 et 6.



Lili, Pierre et Simon habitent donc chacun dans une de ces trois huttes. Nous devons maintenant déterminer dans laquelle des trois vit Lili. Les deux autres informations concernent la hutte de Nina, et indiquent qu'exactly deux canaux rejoignent sa hutte. Nina vit donc dans une de ces trois huttes : 1, 4 et 8.



Comme nous savons que Simon et Pierre sont les deux voisins de Nina, nous pouvons en déduire que :

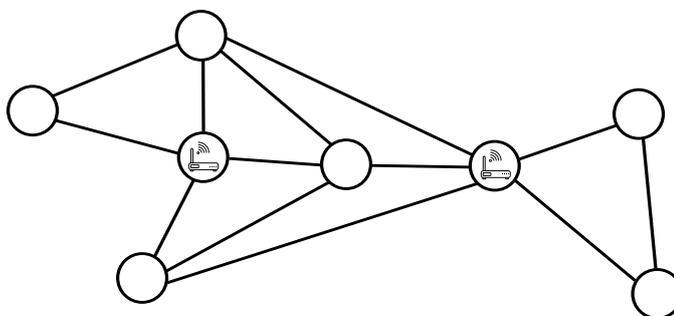
- Nina vit dans la hutte 1.
- Simon et Pierre vivent dans les huttes 5 et 7 (ou l'inverse).

Il ne reste donc qu'une seule hutte de laquelle partent quatre canaux et qui peut être celle de Lili. C'est la hutte 6.

C'est de l'informatique !

Dans cet exercice, deux huttes de castor sont reliées par un canal. L'ensemble des huttes et des canaux forme un réseau qui représente les *relations* entre toutes les huttes. Un tel réseau de relations entre des objets est appelé un *graphe* en informatique et en mathématiques. Un graphe peut être considéré comme un *ensemble* de *nœuds* qui sont reliés par des *arêtes*. Dans cet exercice, les nœuds sont les huttes et les arêtes sont les canaux.

La *théorie des graphes* peut être utilisée pour modéliser les relations entre des paires d'objets. Les graphes sont des modèles mathématiques de structures techniques ou naturelles, par exemple des structures sociales, des réseaux informatiques, des réseaux routiers, des circuits électriques, des réseaux d'approvisionnement ou des molécules. Les graphes peuvent être utiles pour décrire et résoudre des *problèmes de réseaux*, par exemple lorsqu'il s'agit de trouver un bon emplacement pour un routeur dans un bâtiment ou de s'assurer qu'un réseau Wi-Fi atteigne chaque pièce d'une maison.





Mots clés et sites web

- Graphe: [https://fr.wikipedia.org/wiki/Graphe_\(mathématiques_discrètes\)](https://fr.wikipedia.org/wiki/Graphe_(mathématiques_discrètes))
- Théorie des graphes: https://fr.wikipedia.org/wiki/Théorie_des_graphes
- Ensemble: <https://fr.wikipedia.org/wiki/Ensemble>
- Nœud: [https://fr.wikipedia.org/wiki/Sommet_\(théorie_des_graphes\)](https://fr.wikipedia.org/wiki/Sommet_(théorie_des_graphes))
- Arête: [https://fr.wikipedia.org/wiki/Arête_\(théorie_des_graphes\)](https://fr.wikipedia.org/wiki/Arête_(théorie_des_graphes))



A. Auteur·e·s des exercices

 Esraa Almajhad

 Leo Barichello

 Liam Baumann

 Wilfried Baumann

 Linda Björk Bergsveinsdóttir

 Tobias Berner

 Marta J. Burzanska

 Sarah Chan

 Byeonggyu Cho

 Kris Coolsaet

 Darija Dasović

 Christian Datzko

 Susanne Datzko

 Justina Dauksaite

 Nora A. Escherle

 Gerald Futschek

 Mark Edward M. Gonzales

 Yasemin Gülbahar

 Benjamin Hirsch

 Alisher Ikramov

 Thomas Ioannou

 Dauksaite Justina

 Hakin Kim

 Jihye Kim

 Seulki Kim

 Vaidotas Kinčius

 Regula Lacher

 Taina Lehtimäki

 Marielle Léonard

 Karolína Miková

 Jelena Milojkovic

 Madhavan Mukund

 Ágnes Erdősné Németh

 Elsa Pellet

 Zsuzsa Pluhár

 Wolfgang Pohl

 John-Paul Pretti

 Le Quang Quan

 Chris Roffey

 Kirsten Schlüter

 Bernadette Spieler

 Goran Sukovic

 Monika Tomcsányiová

 Ahto Truu

 Troy Vasiga

 Florentina Voboril

 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



<http://www.ubs.com/>



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



i-factory (Musée des transports, Lucerne)



<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

Scuola universitaria professionale della Svizzera italiana



<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.



010100110101011001001001
010000010010110101010011
010100110100100101000101
001011010101001101010011
010010010100100100100001

SS!E

www.svia-ssie-ssii.ch
schweizerischervereinfürinformatikind
erausbildung//sociétésuissepourl'infor
matique dans l'enseignement//societàsviz
zeraperl'informaticanell'insegnamento

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.