



INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Exercices et solutions 2020

Années HarmoS 7/8

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

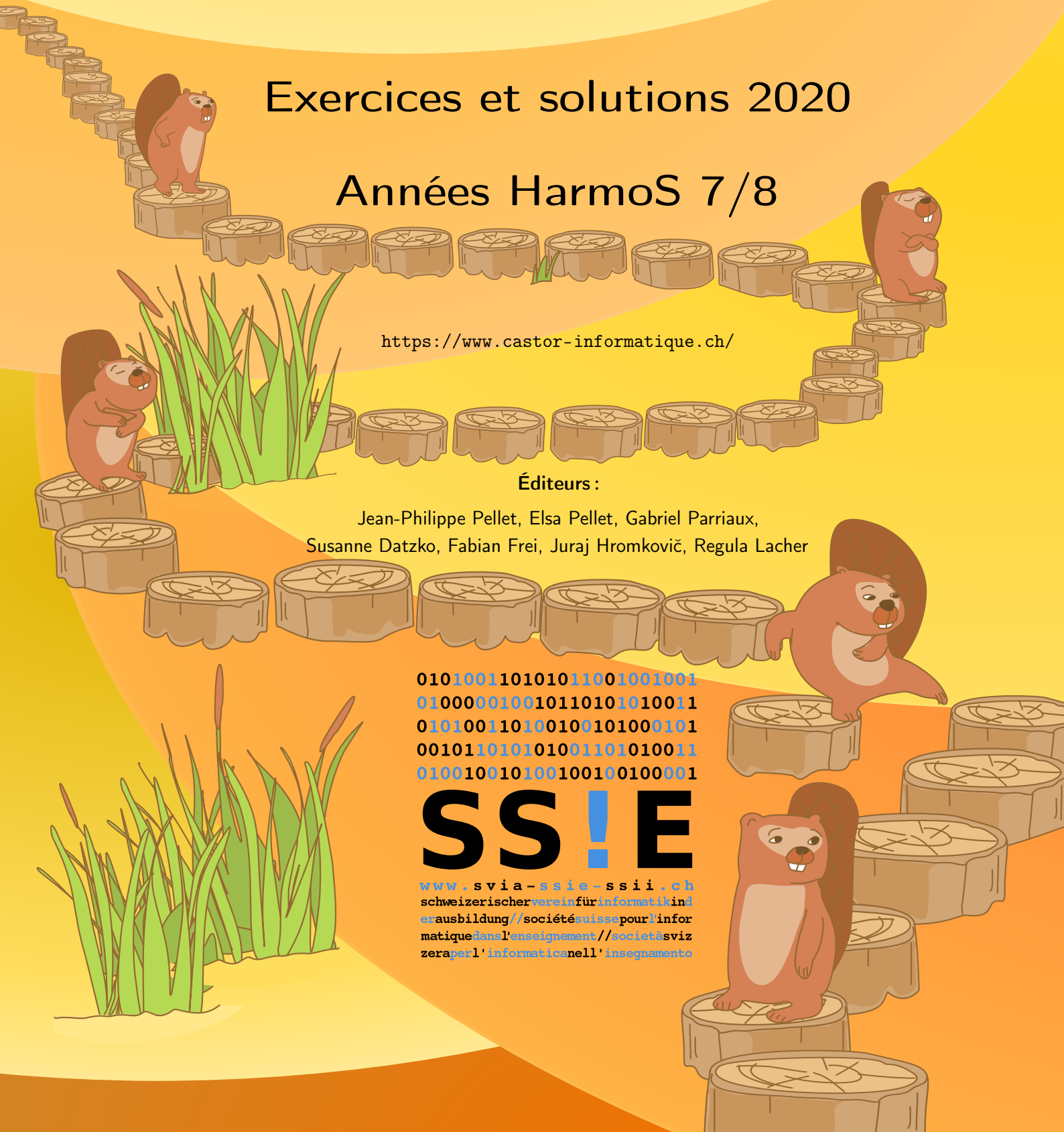
Éditeurs :

Jean-Philippe Pellet, Elsa Pellet, Gabriel Parriaux,
 Susanne Datzko, Fabian Frei, Juraj Hromkovič, Regula Lacher

010100110101011001001001
 010000010010110101010011
 010100110100100101000101
 001011010101001101010011
 010010010100100100100001

SS!E

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





Ont collaboré au Castor Informatique 2020

Susanne Datzko, Fabian Frei, Martin Guggisberg, Lucio Negrini, Gabriel Parriaux, Jean-Philippe Pellet

Cheffe de projet : Nora A. Escherle

Nous adressons nos remerciements pour le travail de développement des exercices du concours à :
Juraj Hromkovič, Michael Barot, Christian Datzko, Jens Gallenbacher, Dennis Komm, Regula Lacher,
Peter Rossmann : ETH Zurich, Ausbildungs- und Beratungszentrum für Informatikunterricht

Le choix des exercices a été fait en collaboration avec les organisateurs de Bebras en Allemagne, Autriche, Hongrie, Slovaquie et Lituanie. Nos remerciements en particulier :

Valentina Dagienė : Bebras.org

Wolfgang Pohl, Hannes Endreß, Ulrich Kiesmüller, Kirsten Schlüter, Michael Weigend : Bundesweite Informatikwettbewerbe (BWINF), Allemagne

Wilfried Baumann, Anoki Eischer : Österreichische Computer Gesellschaft

Gerald Futschek, Florentina Voboril : Technische Universität Wien

Zsuzsa Pluhár : ELTE Informatikai Kar, Hongrie

Michal Winzcer : Université Comenius de Bratislava, Slovaquie

La version en ligne du concours a été réalisée sur l'infrastructure cuttle.org. Nous remercions pour la bonne collaboration :

Eljakim Schrijvers, Justina Dauksaite, Arne Heijenga, Dave Oostendorp, Andrea Schrijvers, Alieke Stijf, Kyra Willekes : cuttle.org, Pays-Bas

Chris Roffey : Université d'Oxford, Royaume-Uni

Pour le support pendant les semaines du concours, nous remercions en plus :

Hanspeter Erni : Direction, école secondaire de Rickenbach

Gabriel Thullen : Collège des Colombières

Beat Trachsler : Kantonsschule Kreuzlingen

Christoph Frei : Chragokyberneticks (Logo Castor Informatique Suisse)

Dr. Andrea Leu, Maggie Winter, Brigitte Manz-Brunner : 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 2020 a été réalisé par la Société Suisse de l'Informatique dans l'Enseignement SSIE et soutenu par la Fondation Hasler.

HASLERSTIFTUNG

Cette brochure a été produite le 20 janvier 2026 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 2020.



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. 40.



Préambule

Très bien établi dans différents pays européens et plus largement à l'échelle mondiale depuis plusieurs années, le concours « Castor Informatique » a pour but d'éveiller l'intérêt des enfants et des jeunes pour l'informatique. En Suisse, le concours est organisé en allemand, en français et en italien par la SSIE, la Société Suisse pour l'Informatique dans l'Enseignement, et soutenu par la Fondation Hasler dans le cadre du programme d'encouragement « FIT in IT ».

Le Castor Informatique est le partenaire suisse du concours « Bebras International Contest on Informatics and Computer Fluency » (<https://www.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 2020 a été fait pour cinq tranches d'âge, basées sur les années scolaires :

- Années HarmoS 5 et 6 (Petit Castor)
- Années HarmoS 7 et 8
- Années HarmoS 9 et 10
- Années HarmoS 11 et 12
- Années HarmoS 13 à 15

Les élèves des années HarmoS 5 et 6 avaient 9 exercices à résoudre : 3 faciles, 3 moyens, 3 difficiles. Les élèves des années HarmoS 7 et 8 avaient, quant à eux, 12 exercices à résoudre (4 de chaque niveau de difficulté). Finalement, chaque autre tranche d'âge devait résoudre 15 exercices (5 de chaque niveau de difficulté).

Chaque réponse correcte donnait des points, chaque réponse 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.

Pour de plus amples informations :

SVIA-SSIE-SSII Société Suisse de l'Informatique dans l'Enseignement

Castor Informatique

Gabriel Parriaux

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

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



















Table des matières

Ont collaboré au Castor Informatique 2020	i
Préambule	iii
Table des matières	v
1. La pièce de théâtre	1
2. Chiffres secrets	5
3. Sudoku boisé 3×3	7
4. Troc au château	11
5. Prochain arrêt, gare!	15
6. Piles de troncs d'arbres	17
7. Quartier coloré	21
8. Épidémiologie	25
9. Les textes tendres de Tabea	27
10. Bols	31
11. Abeille assidue	33
12. Lourdes comparaisons	37
A. Auteur-e-s des exercices	40
B. Sponsoring : Concours 2020	42
C. Offres ultérieures	44



1. La pièce de théâtre

Une pièce de théâtre raconte l'histoire d'une belle princesse , d'un noble chevalier , d'un roi sage  et d'un méchant dragon . Au début de la pièce, la scène est vide. Pendant la représentation, les quatre personnages entrent en scène et quittent la scène dans l'ordre suivant :

Premier acte			Deuxième acte	
Le roi entre en scène		E N T R A C T E	Le dragon entre en scène	
La princesse entre en scène			Le chevalier entre en scène	
Le roi quitte la scène			Le dragon quitte la scène	
Le dragon entre en scène			La princesse entre en scène	
La princesse quitte la scène			Le chevalier quitte la scène	
Le dragon quitte la scène			La princesse quitte la scène	
Entracte			Fin	

Quelle situation n'aura pas lieu ?

















- A) La princesse et le chevalier sont ensemble sur scène.
- B) Le roi et le dragon sont ensemble sur scène.
- C) Le chevalier n'entre en scène qu'après l'entracte.
- D) Le chevalier et le dragon sont ensemble sur scène.



Solution

La bonne réponse est B) Le roi et le dragon sont ensemble sur scène, car cette affirmation n'est jamais vraie au cours de la pièce de théâtre.

On peut y réfléchir pas à pas :

Intrigue	 Roi sur scène ?	 Princesse sur scène ?	 Dragon sur scène ?	 Chevalier sur scène ?	Réponses correspondantes
Premier acte					
	Oui	Non	Non	Non	
	Oui	Oui	Non	Non	
	Non	Oui	Non	Non	
	Non	Oui	Oui	Non	
	Non	Non	Oui	Non	
	Non	Non	Non	Non	
Entracte					
Deuxième acte					
	Non	Non	Oui	Non	
	Non	Non	Oui	Oui	C), D)
	Non	Non	Non	Oui	
	Non	Oui	Non	Oui	A)
	Non	Oui	Non	Non	
	Non	Non	Non	Non	

Fin

On peut vérifier pour chaque réponse si l'affirmation qui y est faite est vraie ou pas en parcourant la table ligne par ligne.



Pour la réponse A), on cherche une ligne à laquelle la princesse et le chevalier sont présents sur scène. C'est la cas à la deuxième ligne du deuxième acte, car la princesse entre en scène alors que le chevalier y est déjà depuis la deuxième ligne et y reste jusqu'à la cinquième ligne. L'affirmation de la réponse A) est donc vraie à au moins un moment de la pièce.

Pour la réponse D), on cherche une ligne à laquelle le chevalier et le dragon sont présents sur scène. C'est la cas à la deuxième ligne du deuxième acte, car le chevalier monte sur scène alors que le dragon y est déjà depuis la première ligne et y reste jusqu'à la troisième ligne. L'affirmation de la réponse D) est donc vraie à au moins un moment de la pièce.

L'affirmation de la réponse C) est d'un genre différent. Si cette affirmation est vraie, le chevalier ne doit pas avoir été sur scène de tout le premier acte. On doit donc regarder la colonne du chevalier pendant le premier acte. Dans celle-ci, c'est toujours écrit « non », donc le chevalier n'a en effet pas été sur scène pendant le premier acte. Par contre, il entre en scène à la deuxième ligne du deuxième acte, donc l'affirmation de la réponse C) est également vraie.

Si l'affirmation de la réponse B) était vraie, le roi et le dragon devraient être les deux sur scène à l'une des lignes. Cependant, il n'y a aucune ligne dans laquelle c'est écrit « oui » dans les deux colonnes. Le roi quitte déjà la scène à la troisième ligne du premier acte et n'y entre plus jusqu'à la fin. Le dragon, lui, entre en scène seulement à la quatrième ligne du premier acte. Peut-être qu'ils se rencontrent dans les coulisses, mais ils ne sont jamais ensemble sur scène. L'affirmation de la réponse B) n'est donc jamais vraie, et B) est la bonne réponse.

C'est de l'informatique !

On peut s'imaginer toute une histoire pendant le déroulement de la pièce de théâtre, mais seule une des propriétés de chaque personnage est importante pour cet exercice : se trouve-t-il sur scène à un moment précis ou pas ? Cette restriction de la perspective à certaines propriétés s'appelle l'*abstraction*.

De telles abstractions peuvent facilement être formulées en informatique. Pour chaque personnage, on définit ce que l'on appelle une *variable*, qui répond à la question de la présence du personnage sur scène à ce moment-là. Les quatre variables sont : « Roi sur scène ? », « Princesse sur scène ? », « Dragon sur scène ? » et « Chevalier sur scène ? ». La réponse à chacune de ces questions change plusieurs fois pendant la pièce de théâtre ; la réponse à chaque question est parfois « oui » et parfois « non ». En informatique, on appelle la réponse actuelle à une question la valeur actuelle de la variable correspondante. La valeur d'une variable peut donc changer autant de fois que nécessaire en informatique (c'est différent en mathématiques où les variables ne changent pas de valeur avec le temps). La table dans l'explication de la réponse montre les quatre variables et les valeurs correspondantes à chaque moment.

Il y a d'autres manières de considérer la pièce de théâtre. On peut regarder quels personnages sont sur scène en ce moment (on observe alors la valeur momentanée des quatre variables). On appelle chaque combinaison de personnages un état de la scène. Lorsqu'un personnage entre en scène ou la quitte, l'état de la scène change. On appelle aussi cela une transition de la scène d'un état à un



autre. Si l'on dessine un rond séparé pour chaque état (combinaison de personnages) sur une feuille de papier, on peut voir l'ensemble comme une abstraction de la scène.

De plus, on peut représenter les transitions possibles par des flèches reliant un état à un autre. En faisant cela, on obtient ce que s'appelle en informatique un *diagramme états-transitions* de la scène.

Au début de la pièce de théâtre, la scène est vide. On appelle l'état correspondant l'*état initial*. On peut dessiner le déroulement de la pièce de théâtre comme un chemin dans le diagramme états-transitions. Le chemin commence à l'état initial et suit ensuite les flèches qui correspondent à l'intrigue de la pièce.

Les diagrammes états-transitions sont très importants en informatique. On doit réfléchir au diagramme états-transitions de chaque système complexe à un moment donné. Pour les êtres humains, c'est souvent laborieux de travailler avec de tels états et transitions abstraits, alors que les ordinateurs peuvent très bien le faire. Cela vaut donc la peine pour les êtres humains de représenter leurs problèmes dans des diagrammes états-transitions afin que les ordinateurs puissent les résoudre.

Mots clés et sites web

- Variable : [https://fr.wikipedia.org/wiki/Variable_\(informatique\)](https://fr.wikipedia.org/wiki/Variable_(informatique))
- État, transition, diagramme états-transitions :
https://fr.wikipedia.org/wiki/Diagramme_états-transitions



2. Chiffres secrets

L'année de construction de chaque hutte de castor est écrite sur un panneau en dessus de l'entrée. Les castors utilisent leurs propres symboles pour représenter les chiffres. La table à droite montre comment les symboles des castors sont assemblés à partir des chiffres :

	-	=	≡	▷	▷
□	0	1	2	3	4
◁	5	6	7	8	9

Par exemple, les castors représentent le chiffre « 5 » par le nouveau symbole ◁▷, qui est assemblé comme ça :

	-	=	≡	▷	▷
□	0	1	2	3	4
◁▷	5	6	7	8	9

Voici la hutte de Cleverias :



En quelle année la hutte de Cleverias a-t-elle été construite ?

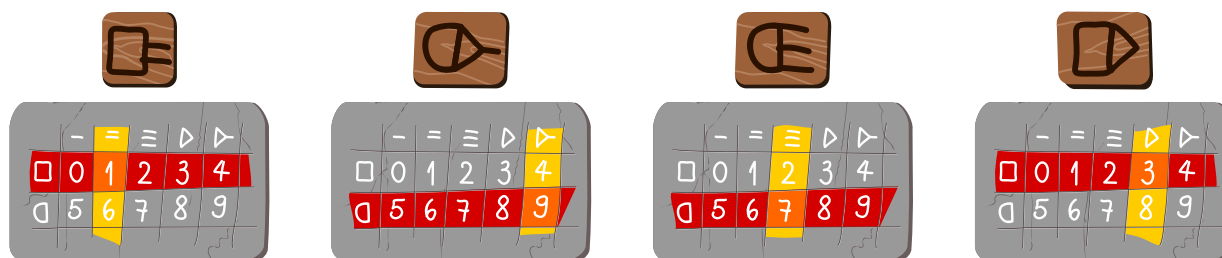
- A) 0978
- B) 1574
- C) 1923
- D) 1973
- E) 1993
- F) 2973
- G) 6378



Solution

Tu peux trouver l'année de construction de la hutte en déterminant la ligne et la colonne correspondant à chaque symbole. Le chiffre recherché se trouve à l'intersection de la ligne et de la colonne.

Comme il y a quatre symboles, tu fais cela quatre fois.



Les quatre chiffres dans le bon ordre donnent le nombre 1973.

C'est de l'informatique !

Garder des informations secrètes ou protéger des données est une tâche vieille de 4000 ans. D'innombrables écritures secrètes ont été développées et utilisées dans ce but. Aujourd'hui, la sécurité des données est l'un des thèmes majeurs de l'informatique. Une des méthodes pour empêcher la lecture non autorisée de données est de les *chiffrer*. Le chiffrement transforme un *texte clair* en *cryptogramme*. La reconstruction du texte clair à partir du cryptogramme s'appelle *déchiffrement*. L'étude des cryptogrammes s'appelle *cryptologie*.

Les cultures antiques utilisaient le plus souvent des écritures secrètes remplaçant des lettres par d'autres lettres ou de tout nouveaux symboles. L'écriture secrète utilisée ici a été développée spécialement pour le Castor Informatique, mais se base sur un concept venant de la Palestine antique. À l'époque, la règle de sécurité était que seules des écriture secrètes faciles à apprendre par cœur pouvaient être utilisées. C'était considéré comme un trop grand risque de garder une description écrite de l'écriture secrète. Une table comme celle utilisée ici est facile à apprendre par cœur. Le célèbre chiffre des francs-maçon se base sur ce principe.

Au lieu d'assembler de nouveaux symboles seulement pour les chiffres, on peut également inventer une écriture secrète pour les textes. Pour cela, on écrit toutes les lettres dans une table, puis on invente de nouveaux symboles pour les lignes et les colonnes. Ainsi, on obtient un nouveau symbole pour chaque lettre.

Mots clés et sites web

- Cryptographie : <https://fr.wikipedia.org/wiki/Cryptographie>
- Cryptogramme



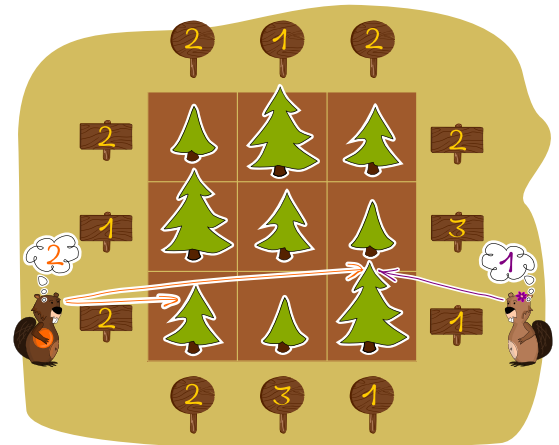
3. Sudoku boisé 3×3

Les castors plantent des rangées de sapins. Les sapins ont trois hauteurs différentes (1 🌲, 2 🌲 et 3 🌲) et il y a exactement un sapin de chaque hauteur sur chaque rangée.

Lorsque les castors observent une rangée de sapin depuis l'une de ses extrémités, il ne peuvent **pas** voir les plus petits sapins qui sont cachés derrière de plus grands sapins.

C'est écrit sur un panneau au bout de chaque rangée combien de sapins l'on peut voir depuis cet endroit-là.

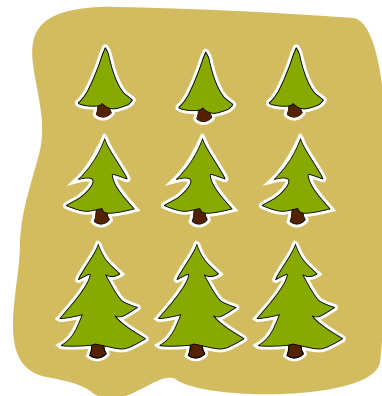
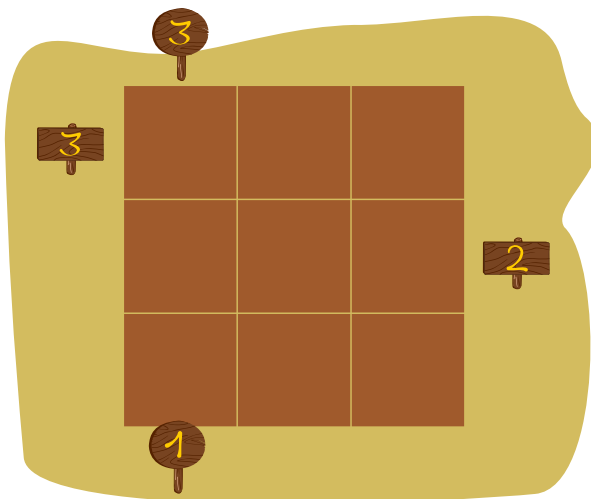
Les castors plantent à présent neuf sapins sur un champ de 3×3 cases, comme dans l'exemple à droite.



Pour cela, les règles suivantes s'appliquent :

- dans chaque ligne, il y a exactement un sapin de chaque hauteur ;
- dans chaque colonne, il y a exactement un sapin de chaque hauteur ;
- les panneaux indiquant le nombre de sapins visibles sont plantés tout autour du champ de 3×3 cases.

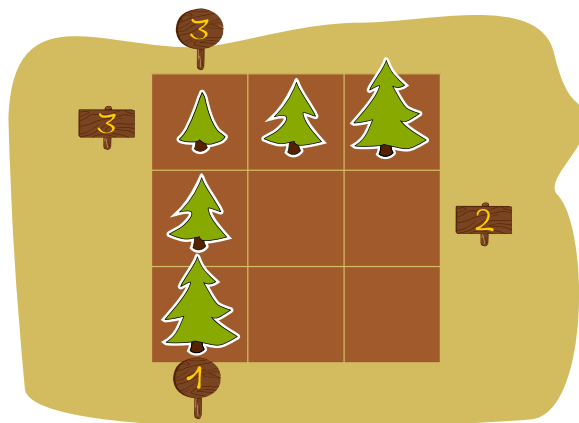
Écris dans chaque case la hauteur du sapin qui s'y trouve.





Solution

Il y a dans le champ deux panneaux indiquant que l'on peut voir trois sapins depuis leurs positions. On ne peut voir trois sapins dans une rangée que lorsque les sapins sont dans un ordre croissant, donc depuis cette position. La colonne de gauche et la ligne du haut sont ainsi déterminées :

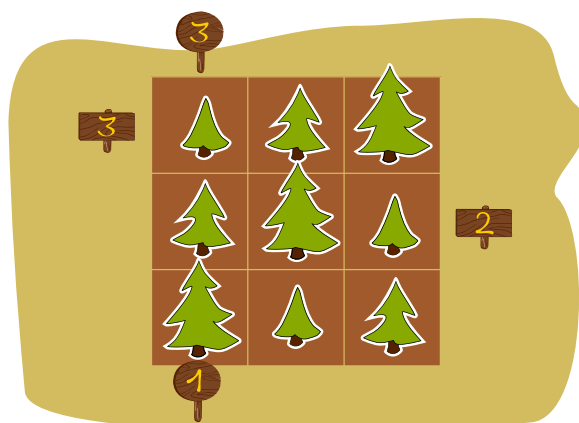


Le panneau avec le 2 à droite indique que l'on peut voir deux sapins depuis là, il doit donc y avoir un sapin de hauteur 3 au milieu et la ligne centrale est ainsi 2 () , 3 () , 1 () .

Les cases suivantes sont remplies d'après la règle du «sudoku» qui oblige chaque rangée à avoir exactement un sapin de chaque hauteur.

Il doit y avoir un sapin de hauteur 1 () au milieu de la ligne du bas, car les deux autres hauteurs de sapin sont déjà présentes dans la colonne du milieu. Il manque un sapin de hauteur 2 () tout en bas à droite pour compléter la rangée.

Voici la solution complète :



C'est de l'informatique !

Cet exercice est centré sur trois compétences fondamentales pour les informatiennes et informaticiens.

Premièrement, il s'agit de trouver une solution respectant certaines contraintes, ou si nécessaire de corriger une solution proposée.



Deuxièmement, il s'agit de la capacité de reconstruire des objets en se basant sur leur représentation à partir d'informations partielles. Ceci est lié à la génération d'objets (représentation d'objets) à partir d'informations disponibles limitées lorsque leur conformité aux lois est connue. On peut aussi utiliser de tels procédés dans la compression de données.

Troisièmement, on peut utiliser de tels champs d'arbres avec des panneaux pour créer des codes correcteurs. Des erreurs arrivant lors de l'entrée des données ou du transfert d'information peuvent ainsi être automatiquement reconnues ou même corrigées.

Mots clés et sites web

- Sudoku : <https://fr.wikipedia.org/wiki/Sudoku>
- Détection et correction d'erreurs : https://fr.wikipedia.org/wiki/Code_correcteur
- Reconstruction d'objets à partir d'informations partielles
- Vérification de l'exactitude de la représentation de données






































4. Troc au château

Un castor malin a besoin d'un sapin 🌲 pour construire un barrage sur la rivière. Malheureusement, il n'a qu'une carotte 🥕. C'est un jour de marché au château aujourd'hui, et le castor veut y troquer sa carotte 🥕 contre un sapin 🌲.

Dans chaque salle du château, deux types de troc sont proposés. Le table suivante liste ces propositions :

Salle A :		→		ou		→	
Salle B :		→		ou		→	
Salle C :		→		ou		→	
Salle D :		→		ou		→	
Salle E :		→		ou		→	
Salle F :		→		ou		→	
Salle G :		→		ou		→	
Salle H :		→		ou		→	

Dans la salle B, le castor peut par exemple troquer une bague  contre une glace , mais pas l'inverse.

Dans quel ordre le castor doit-il passer dans les salles du château pour finalement avoir le sapin 🌲 désiré ?

- A) DGE : D'abord la salle D, puis la salle G et finalement la salle E.
- B) GCE : D'abord la salle G, puis la salle C et finalement la salle E.
- C) AGE : D'abord la salle A, puis la salle G et finalement la salle E.
- D) DBC : D'abord la salle D, puis la salle B et finalement la salle C.

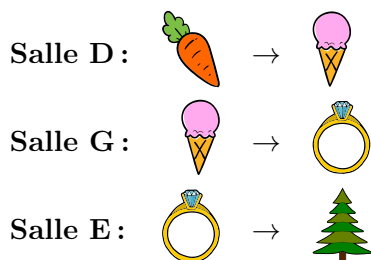


Solution

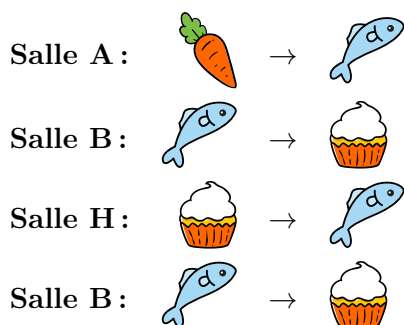
La bonne réponse est A) DGE. D'abord la salle D, puis la salle G et finalement la salle E.

Dans la salle D, le castor troque sa carotte contre une glace . Ensuite il va dans la salle G, où il troque la glace contre une bague . Finalement, le castor va dans la salle E pour troquer la bague contre un sapin .

Voici le déroulement complet :



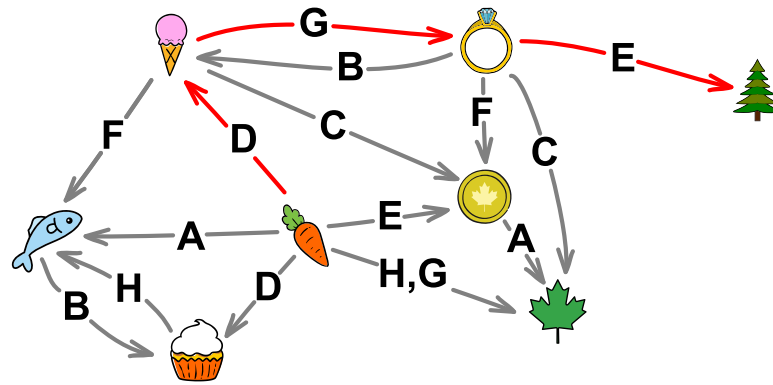
Il y a deux stratégies appropriées pour trouver le bon ordre dans lequel visiter les salles. La première essaie de prendre toutes les possibilités de troc en considération. Elle commence avec le premier troc, lors duquel on peut échanger la carotte dans cinq salles (A, D, E, G et H) contre six objets différents. Ensuite, toutes les possibilités de troc pour chacun de ces six objets sont considérées. C'est complexe et peut même tourner en rond, comme dans l'exemple suivant dans lequel le castor peut passer dans les salles B et H indéfiniment :



Cette première stratégie est donc très complexe et n'arrive rapidement à une solution qu'avec beaucoup de chance.

La deuxième stratégie atteint rapidement le but dans cet exemple concret. Le principe est de commencer la recherche par le résultat souhaité, donc le sapin . Le castor ne peut obtenir un sapin que dans la salle E. On ne reçoit un sapin qu'en échange d'une bague . Le prochain objet désiré est donc une bague! La bague ne peut aussi être obtenue que dans une salle, la salle G, en échange d'une glace . On peut obtenir une glace soit dans la salle B contre une bague , soit dans la salle D contre une carotte . Le castor malin doit donc commencer sa série de trocs dans la salle D.

On peut représenter les trocs possibles par un graphe avec des arêtes orientées (flèches) pour avoir une meilleure vue d'ensemble. Chaque nœud du graphe représente un des objets du troc et chaque arête qui en part une possibilité de troc. Sur l'arête est aussi noté le nom de la salle dans laquelle le troc est possible.



Cette représentation visuelle des objets, des possibilités de troc et des salles permet de trouver facilement comment aller de la carotte au sapin, soit en suivant un chemin sur le graphe orienté : d’abord la salle D, puis la salle G et finalement la salle E.

C’est de l’informatique !

De manière générale, on peut considérer les *processus de calcul* comme des *suites de transformations* (ici, ce sont des trocs) ou comme des *suites d’états* d’un système. L’état de départ du système est dans notre cas la carotte, et la transformation (la transition) de la carotte à la glace change cet état en glace.

Une *transition* mène donc d’un état à un autre. On appelle aussi une suite de transitions un *calcul*.

Cet exercice traite donc aussi de calcul à un niveau très général. Le système de l’exemple est *non déterministe* : cela veut dire qu’il y a parfois plusieurs étapes de calcul possibles, comme il y a plusieurs possibilité de trocs dans l’exercice. Le non-déterminisme est un concept important dans la modélisation en informatique. Les étapes de calcul possibles sont décrites par des règles de transformation (la table montrant les possibilités de troc). Il s’agit du célèbre *problème d’accessibilité* lorsque l’on veut déterminer si le castor peut troquer une carotte contre un sapin, donc si un certain *état final* peut être atteint depuis un certain *état initial* — un problème qui a de nombreuses applications.

L’exercice ci-dessus montre que c’est parfois une bonne idée de chercher l’état initial en partant de l’état final plutôt que le contraire. Cette stratégie s’appelle aussi *recherche en arrière*.

En comparant les différentes stratégies pour résoudre le problème, on voit que le graphe orienté offre une possibilité claire de représenter l’*espace d’états* d’un système avec toutes les transitions possibles entre les états. Dans ce modèle de base, on pourrait appliquer les algorithmes de parcours de graphes fondamentaux, comme le *parcours en largeur* et le *parcours en profondeur*.

Mots clés et sites web



- Théorie des graphes : https://fr.wikipedia.org/wiki/Théorie_des_graphes
- Problème d’accessibilité : https://fr.wikipedia.org/wiki/Problème_d’accessibilité



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



5. Prochain arrêt, gare !

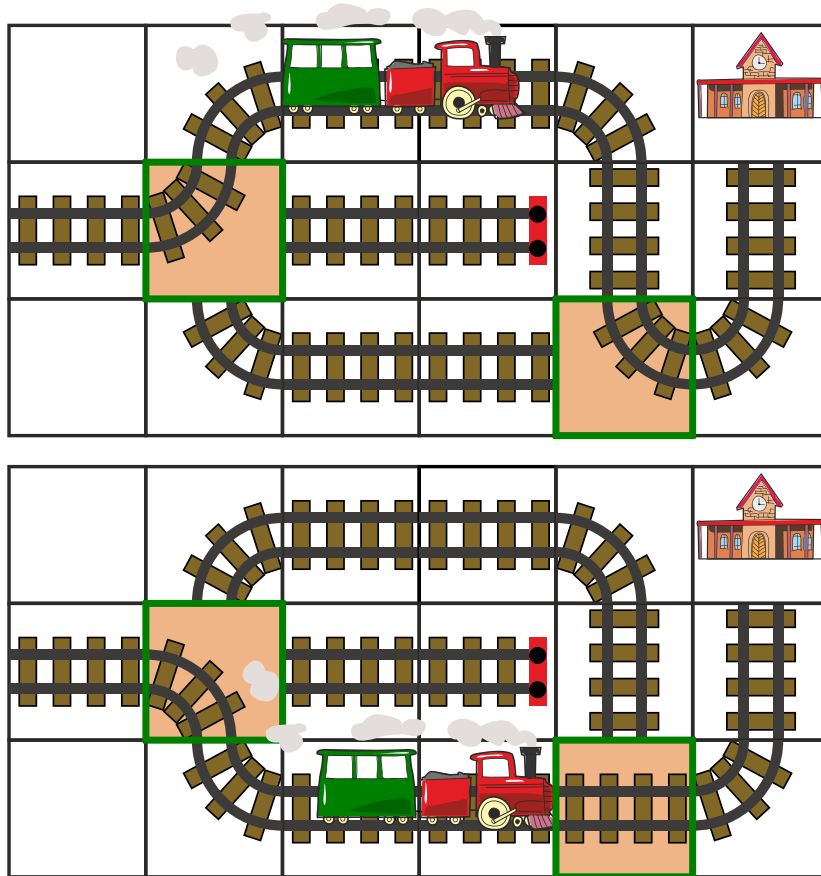
Choisis les bons rails pour les deux cases avec les points verts de façon à ce que le train  puisse aller à la gare .

The puzzle consists of a 3x5 grid. The top row contains a station icon in the rightmost cell. The middle row contains a train icon on the left, followed by a horizontal rail piece, a green circle, a horizontal rail piece with a red signal light, a vertical rail piece, and another vertical rail piece. The bottom row contains a curved rail piece, a horizontal rail piece, a green circle, a curved rail piece, and a vertical rail piece. Below the grid is a selection bar with six options: two curved rail pieces (one left, one right), two horizontal rail pieces (one with signal light, one without), and two vertical rail pieces (one with signal light, one without).



Solution

Ce problème a les deux solutions suivantes :



Avec toutes les autres combinaisons, le train déraile ou fonce dans le buttoir.

C'est de l'informatique !

Comme un train qui suit simplement les rails en roulant, un ordinateur suit simplement les instructions d'un programme. Il ne peut pas savoir si le programme contient une erreur et peut alors « planter », comme un train peut dérailler si les rails ne sont pas assemblés correctement. On doit donc être beaucoup plus attentif en écrivant un programme que lorsque l'on indique la direction de la gare à quelqu'un, par exemple.

Dans cet exercice, il s'agit d'ajouter les instructions manquantes aux bons endroits d'un programme pour pouvoir atteindre l'objectif.

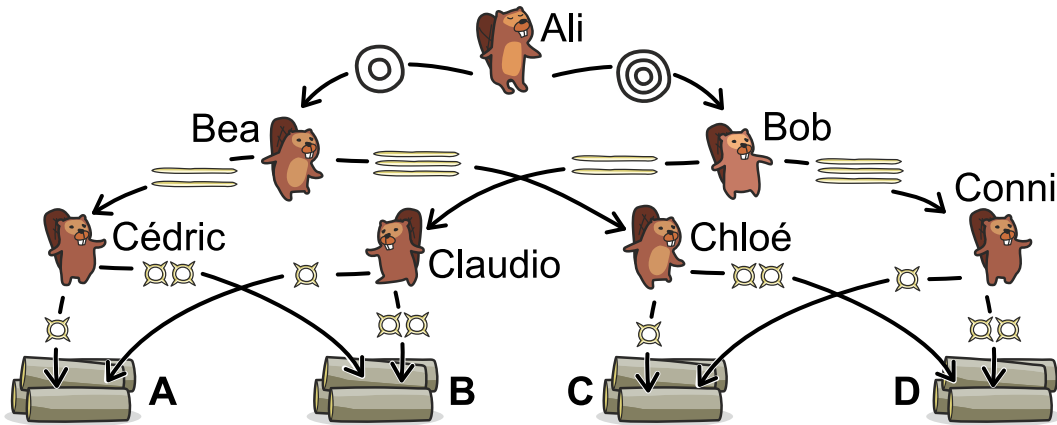
Mots clés et sites web

- Programme
- Instruction : https://fr.wikipedia.org/wiki/Instruction_informatique
- <https://fr.wikipedia.org/wiki/Algorithme>



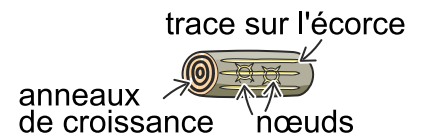
6. Piles de troncs d'arbres

Dans le village des castors, les troncs d'arbres sont répartis dans quatre groupes (A, B, C, D) d'après trois propriétés (le nombre d'anneaux de croissance, le nombre de traces sur l'écorce et le nombre de nœuds dans le bois). Le diagramme de décision montre comment cela se passe.



Par exemple, ce tronc-ci est posé sur la pile D en raison des décisions suivantes :

- Ali voit trois anneaux de croissance et donne le tronc à Bob ;
- Bob voit trois traces sur l'écorce et donne le tronc à Conni ;
- Conni voit deux nœuds dans le bois et pose le tronc sur la pile D.



Sur quelle pile ce tronc va-t-il être posé ?



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



Solution

La bonne réponse est la pile C, car Ali voit deux anneaux de croissance et donne le tronc à Bea. Bea voit trois traces sur l'écorce et transmet le tronc à Chloé. Chloé voit un nœud dans le bois et pose le tronc sur la pile C.

Si l'on veut, on peut déterminer quels troncs correspondent à chaque pile. Il y a deux types de troncs sur chaque pile.

Sur la pile **A** :

- Les troncs avec 2 anneaux de croissance, 2 traces sur l'écorce et 1 nœud dans le bois.
- Les troncs avec 3 anneaux de croissance, 2 traces sur l'écorce et 1 nœud dans le bois.

Sur la pile **B** :

- Les troncs avec 2 anneaux de croissance, 2 traces sur l'écorce et 2 nœuds dans le bois.
- Les troncs avec 3 anneaux de croissance, 2 traces sur l'écorce et 2 nœuds dans le bois.

Sur la pile **C** :

- Les troncs avec 2 anneaux de croissance, 3 traces sur l'écorce et 1 nœud dans le bois.
- Les troncs avec 3 anneaux de croissance, 3 traces sur l'écorce et 1 nœud dans le bois.

Sur la pile **D** :

- Les troncs avec 2 anneaux de croissance, 3 traces sur l'écorce et 2 nœuds dans le bois.
- Les troncs avec 3 anneaux de croissance, 3 traces sur l'écorce et 2 nœuds dans le bois.

C'est de l'informatique !

Cet exercice touche à plusieurs concepts informatiques.

En premier lieu, le concept des *diagrammes de décision* est traité, diagrammes qui ont des applications très variées en informatique. Ici, on les utilise pour la *classification* d'objets dans certaines catégories (très souvent, on utilise des arbres de décision, une forme spéciale de diagrammes de décision. Le diagramme de décision de l'exercice n'est pas un arbre de décision, car deux groupes sont posés sur la même pile au dernier niveau du diagramme).

On peut aussi voir le diagramme de décision de cet exercice comme la représentation abstraite des valeurs d'une fonction à plusieurs variables. La terminologie exacte en informatique est *diagramme de décision binaire*.

De plus, on aborde ici le concept des *attributs* (caractéristiques ou propriétés) d'objets. Dans cet exemple, les objets ont trois attributs (anneaux de croissance, trace sur l'écorce, nœuds dans le bois), et chaque attribut a deux valeurs possible (deux ou trois anneaux ou traces et un ou deux nœud[s]).

Il y a beaucoup d'applications possibles pour de tel diagramme de décision. L'une d'entre elles est la classification de paquets envoyés sur un réseau (par des routeurs ou des commutateurs réseau).



Mots clés et sites web

- Arbre de décision : https://fr.wikipedia.org/wiki/Arbre_de_d%C3%A9cision
- Classification : https://fr.wikipedia.org/wiki/Classement_automatique





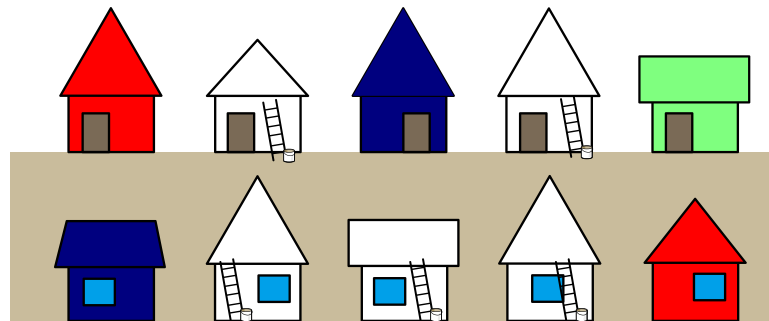
7. Quartier coloré

Les habitants d'une rue veulent peindre leurs maisons blanches en couleur. Chaque maison doit être peinte en l'une de ces trois couleurs : vert clair, rouge ou bleu foncé. Pour que ça n'ait pas l'air ennuyeux, ils suivent les règles suivantes :

- Deux maisons directement l'une à côté de l'autre ne doivent pas être de la même couleur.
- Deux maisons directement face à face dans la rue ne doivent pas avoir la même couleur.

Quelques habitants ont déjà peint leur maison en couleur. Les autres habitants doivent maintenant peindre leur maison de manière à respecter les règles.

Trouve les couleurs appropriées pour les habitants.

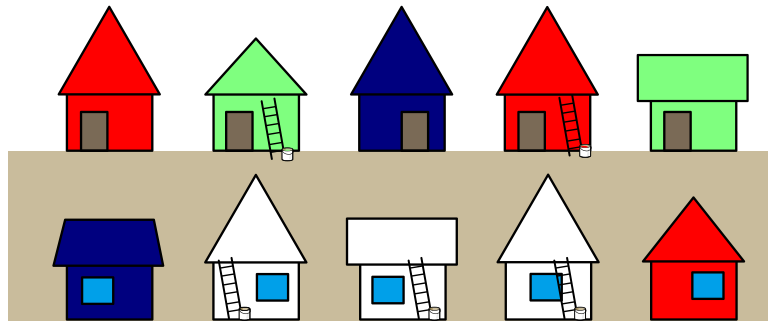




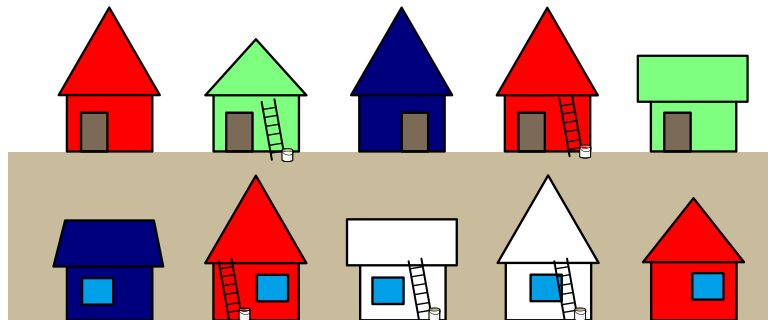
Solution

Le plus facile est de trouver la couleur de chaque maison l'une après l'autre.

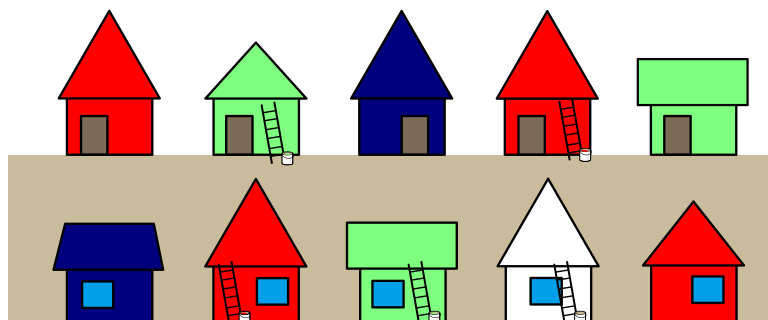
Les deux maisons blanches du côté supérieur de la route sont chacune entourées de deux maisons de couleurs différentes à gauche et à droite. On ne peut donc les peindre qu'en une seule couleur si l'on suit les règles : La maison blanche en haut à gauche en vert clair et la maison blanche en haut à droite en rouge.



Ensuite, on peut voir que la maison blanche en bas à gauche doit être peinte en rouge, car la maison directement à sa gauche est bleu foncé et la maison directement en face est vert clair :

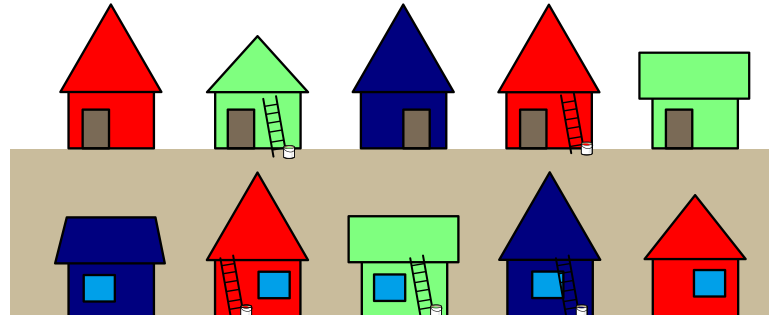


On peut faire presque le même raisonnement pour la maison au milieu du côté inférieur de la rue : Elle doit être peinte en vert clair, car directement à sa droite est la maison que l'on vient de peindre en rouge et directement en face se trouve une maison bleu foncé.





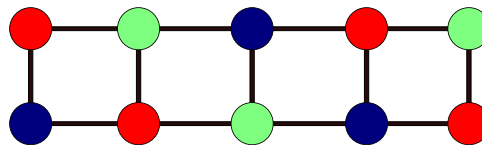
Finalement, on peut aussi trouver la couleur appropriée pour la maison blanche en bas à droite : la maison directement à sa droite et celle directement en face sont les deux rouges, mais comme la maison directement à sa gauche est maintenant vert clair, il ne reste plus que la possibilité de peindre la maison en bleu foncé :



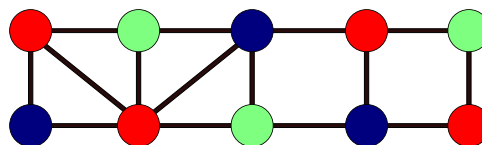
C'est de l'informatique !

Vu de manière abstraite, il s'agit dans cet exercice de trouver une solution qui satisfasse certaines contraintes (règles). C'est un problème rencontré très souvent en informatique.

Les maisons et leur voisinage direct (aussi bien à gauche qu'à droite et que de l'autre côté de la rue en face) peuvent très bien être représentées à l'aide d'un *graphe*, une structure de données très répandue en informatique. Dans ce graphe, chaque maison est un *nœud* et chaque lien de voisinage direct est une *arête* :



Sur l'image, les nœuds sont déjà colorés comme les maisons correspondantes. Les maisons devaient suivre la règle de ne pas avoir la même couleur que leurs voisines. C'est pour cela que les nœuds reliés directement par une arête sur l'image ne sont jamais de la même couleur. Le fait qu'il existe une *coloration valable* du graphe avec trois couleurs ne va pas de soi. Si l'on ajoute deux arêtes au graphe comme sur l'image suivante, il n'y a plus de coloration valable : qu'importe comment on répartit les couleurs dans ce graphe, il y a toujours deux nœuds directement reliés qui sont de la même couleur.



C'est à nouveau possible en utilisant quatre couleurs. Peut-être que c'est toujours possible avec quatre couleurs ? La réponse est à nouveau non. Mais il existe au moins une certaine sorte de graphe que l'on peut toujours colorer de manière valable avec quatre couleurs : on les appelle les graphes planaires. Ce sont des graphes que l'on peut dessiner sans que leurs arêtes ne se croisent (le graphe sur la dernière image n'est pas planaire à cause des liens entre les quatre nœuds à gauche). Le fait



que les graphes planaires aient toujours une coloration à quatre couleurs valable est décrit par le *théorème des quatre couleurs*.

Le théorème des quatre couleurs est spécialement intéressant pour la réalisation de cartes géographiques. Si l'on se représente chaque pays comme un nœud et que l'on relie les pays voisins par une arête, on obtient toujours un graphe planaire (pour être exact, nous devons pour cela exclure l'existence d'enclaves et d'exclaves, c'est-à-dire de parties de pays se trouvant au milieu d'un autre pays). On peut donc colorer ces graphes de manière valable avec quatre couleurs, et on peut donc aussi colorier ces pays sur la carte de manière à ce que les pays voisins ne soient jamais de la même couleur.

La preuve que quatre couleurs suffisent n'est pas facile à faire. On savait déjà il y a 200 ans que cinq couleurs suffisent. La preuve que quatre couleurs suffisent a été faite en 1976 par les mathématiciens Kenneth Appel and Wolfgang Haken. Ils ont pour cela utilisé un ordinateur pour tester un grand nombre d'exceptions et de contre-exemples. L'ordinateur a eu besoin de plus de mille heures pour faire cela. Cela aurait été totalement impossible à faire à la main. Beaucoup de mathématiciens se sont demandé si une telle preuve était valable, car il faut pour cela faire confiance à l'ordinateur.

Mots clés et sites web

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



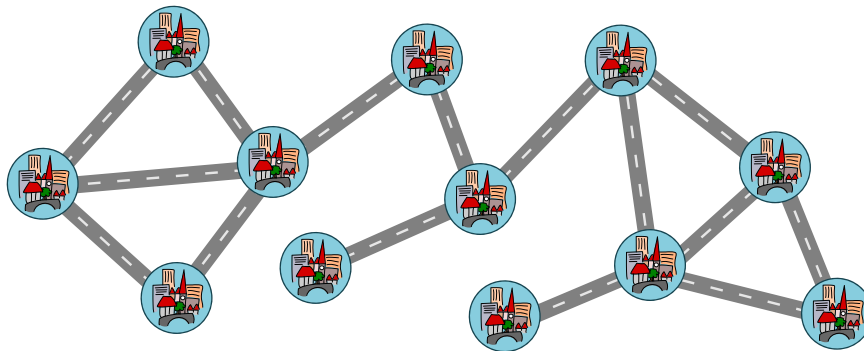
8. Épidémiologie

Castorland comporte 12 villes qui sont reliées par des routes. Les villes qui sont reliées de manière directe ou indirecte forment une communauté commerciale. La carte dans sa forme actuelle montre donc une seule communauté commerciale de 12 villes.

Pour endiguer une épidémie, la circulation doit être réduite. Le parlement des castors décide de fermer exactement deux routes pour diviser les villes en trois communautés commerciales.

Pour n'isoler personne plus que nécessaire, la plus petite communauté commerciale devrait compter autant de villes que possible

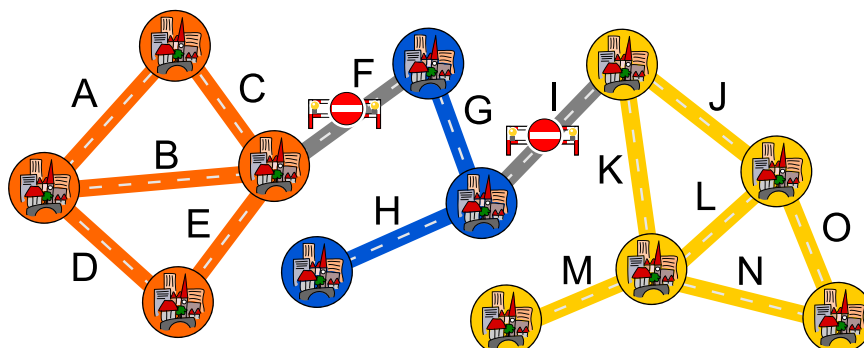
Quelles sont les deux routes qui doivent être fermées ? Biffe-les.





Solution

La bonne réponse est : les routes F et I sur l'image ci-dessous doivent être fermées. De cette manière, trois communautés commerciales de 3, 4 et 5 villes, respectivement, sont formées.



C'est évident que nous ne devons considérer que les routes dont la fermeture engendre une division de la communauté commerciale car elles représentent une connexion unique. Nous avons en effet besoin de deux vraies divisions pour créer trois unités. Ça n'apporte donc rien de fermer la route B, par exemple, car on peut encore atteindre toutes les villes en passant par les routes A ou C. Les seules candidates pour la fermeture sont donc les routes F, G, H, I et M.

On arrive à la réponse du dessus en essayant toutes les dix possibilités de fermer deux de ces cinq routes. En tant qu'être humain, on remarque de plus tout de suite que la fermeture des routes H ou M isolerait une seule ville et n'entre donc pas en question. Cela limite encore le nombre de possibilités à considérer.

C'est de l'informatique !

En informatique, on cherche souvent à diviser un certain réseau en *composantes connexes*. Toutes les parts d'une composante connexe sont reliées de manière directe ou indirecte, alors qu'il n'y a aucun lien entre différentes composantes connexes. L'utilisation dans les réseaux informatiques dans lesquels il est important de déterminer quels ordinateurs peuvent être atteints depuis quels autres est évidente. C'est aussi important de déterminer quels points sont reliés dans la reconnaissance optique de caractères (OCR).

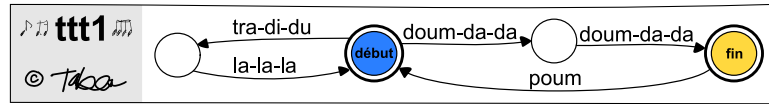
Mots clés et sites web

- Composante connexe : https://fr.wikipedia.org/wiki/Gr%C3%A0phe_connexe
- Parcours d'arbre : https://fr.wikipedia.org/wiki/Parcours_d'arbre



9. Les textes tendres de Tabea

Tabea a beaucoup de succès avec ses textes de chanson de la marque ttt : les textes tendres de Tabea. Ceux-ci peuvent être produits avec le diagramme ttt1 suivant :



Pour produire une chanson, Tabea commence par le « début » (début) et suit l'une des flèches partant de ce point. Lorsqu'il y a plusieurs possibilités, elle choisit. Elle chante les syllabes correspondantes le long de chemin dans l'ordre donné. Lorsqu'elle atteint la « fin » (fin), sa chanson peut se terminer ou continuer.

Voici des exemples de chansons possibles :

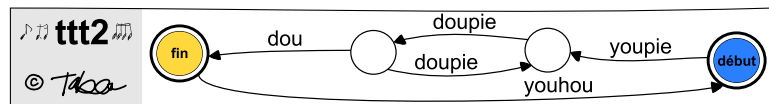
« Tra-di-du La-La-La Tra-di-du La-La-La
Doux-da-da Doux-da-da Poum Doux-da-da Doux-da-da »



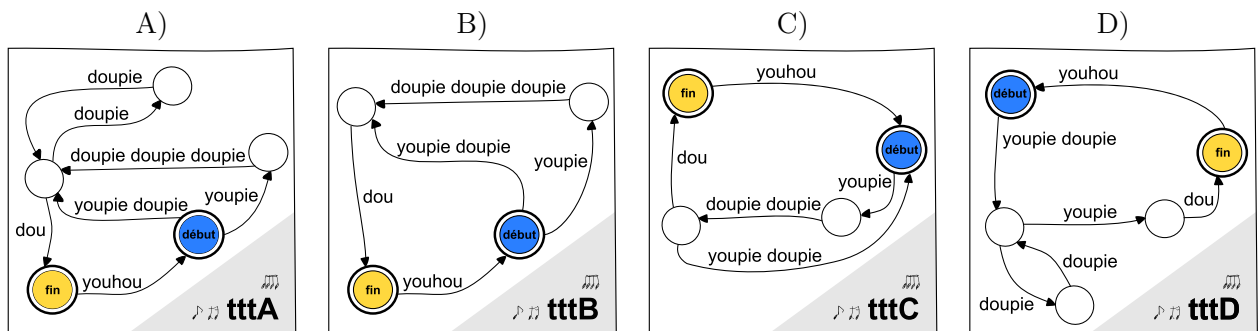
Ou

« Doux-da-da Doux-da-da Poum Tra-di-du La-La-La
Doux-da-da Doux-da-da Poum Tra-di-du La-La-La
Doux-da-da Doux-da-da Poum Doux-da-da Doux-da-da »

En novembre 2020, Tabea commence la production avec les nouveaux textes ttt2 :



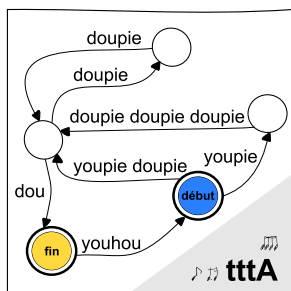
Lequel des diagrammes suivants permet de produire exactement les mêmes textes que ttt2 ?



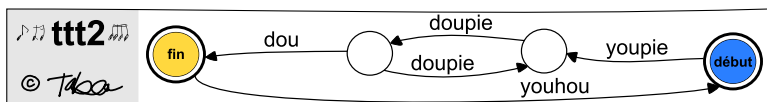


Solution

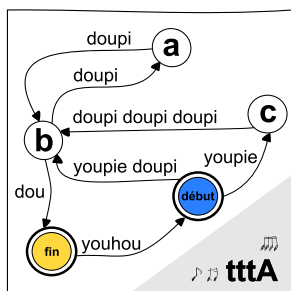
La bonne réponse est A), le diagramme tttA :



Lorsque l'on produit une chanson avec ttt2, elle commence dans tous les cas avec « youpie » qui est suivi d'au moins un « doupie ». On peut ensuite continuer soit avec « dou », soit avec un nombre pair de « doupie » suivi de « dou ». La chanson peut ensuite se terminer ou continuer avec un « youhou » avant de reprendre depuis le début.



Le diagramme tttA permet de produire exactement la même chose : depuis le « début », la chanson peut aller directement à **b** et ainsi commencer avec « youpie douple » ou y aller en passant par **c** avec « youpie douple douple douple ». Ensuite viennent, avec un détour par **a**, un nombre pair de « doupie » avant d'arriver à la fin de la chanson avec « dou ». Comme avec ttt2, on peut alors continuer la chanson avec un « youhou ».



Aussi bien ttt2 que tttA peuvent générer un nombre impair de « doupie » l'un après l'autre, après le « youpie » du début. Par contre, tttB ne peut générer qu'un ou trois « doupie » qui se suivent, et tttC seulement un ou deux. Quant à tttD, il peut certes générer un nombre impair de « doupie » l'un après l'autre, mais va toujours insérer, avant le « dou » final, un « youpie » de plus, ce que ttt2 ne fait pas.

La bonne réponse est donc tttA.

C'est de l'informatique !

Une tâche importante de l'informatique est de trouver des structures dans des données, par exemple dans le langage comme celui du texte d'une chanson. Les diagrammes représentent ce que l'on



appelle des automates finis, qui utilisent des règles très strictes pour générer et reconnaître certains langages. C'est primordial dans le développement des langages de programmation. Dans notre exemple, l'automate fini décrit l'ensemble des chansons que celui-ci peut générer.

La reconnaissance de motifs est également importante dans beaucoup d'autres domaines, comme par exemple le traitement du langage naturel.

Mots clés et sites web

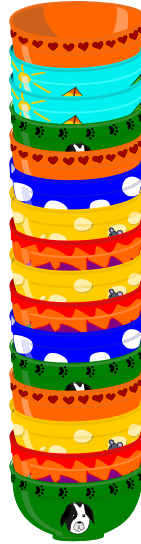
- Automate fini : https://fr.wikipedia.org/wiki/Automate_fini_déterministe
- Langage formel : https://fr.wikipedia.org/wiki/Langage_formel
- <https://sites.google.com/isabc.ca/computationalthinking/pattern-recognition>





10. Bols

Trois frères et sœurs veulent manger leur petit-déjeuner dans trois bols pareils. Ils ont une grande pile de bols. Par précaution, ils n'enlèvent toujours qu'un bol à la fois du haut de la pile.



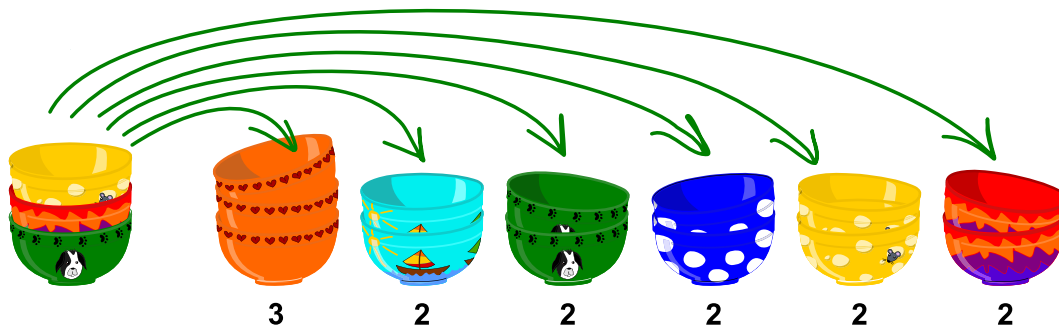
Quel est le plus petit nombre de bols qu'ils doivent enlever de la pile dessinée pour en avoir trois pareils ?

- A) 3 bols
- B) 4 bols
- C) 5 bols
- D) 6 bols
- E) 7 bols
- F) 8 bols
- G) 9 bols
- H) 10 bols
- I) 11 bols
- J) 12 bols
- K) 13 bols
- L) 14 bols
- M) 15 bols
- N) 16 bols



Solution

Réponse K) : Au moins 13 bols doivent être enlevés de la pile pour avoir trois bols pareils.



C'est de l'informatique !



Une *pile*, aussi appelée *stack* en informatique, est une façon très répandue d'enregistrer des choses. Une pile est une structure très simple, mais très puissante, que l'on utilise souvent en programmation. Il y a des règles qui décrivent comment on peut ajouter ou enlever des choses de la pile, le plus souvent seulement depuis le haut. Dans cet exercice, nous n'avons travaillé qu'avec des choses à enlever de la pile. La règle indique que seul l'objet le plus haut de la pile peut être enlevé. Si l'on veut le dixième bol de la pile, on doit donc enlever dix bols les uns après les autres. Pour cela, c'est important d'avoir un autre endroit à disposition où poser les neuf autres bols ; c'est pareil en programmation. Si l'on a une deuxième pile et que les piles peuvent être aussi hautes que l'on veut, on peut en théorie déjà tout calculer ce qui est calculable avec un ordinateur (en informatique, on dit d'une telle chose qu'elle est *complète au sens de Turing*) ! De simples piles comme ça sont vraiment puissantes !

Mots clés et sites web

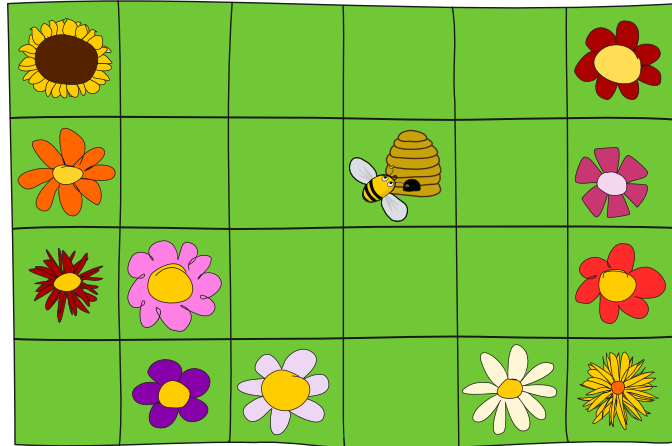
- Pile: [https://fr.wikipedia.org/wiki/Pile_\(informatique\)](https://fr.wikipedia.org/wiki/Pile_(informatique))
- Machine de Turing: https://fr.wikipedia.org/wiki/Machine_de_Turing



11. Abeille assidue

Une abeille  met 10 minutes pour voler d'une case vers le haut, le bas, la gauche ou la droite. Après être partie de la ruche , elle vole pendant 30 minutes au maximum avant de revenir en arrière.

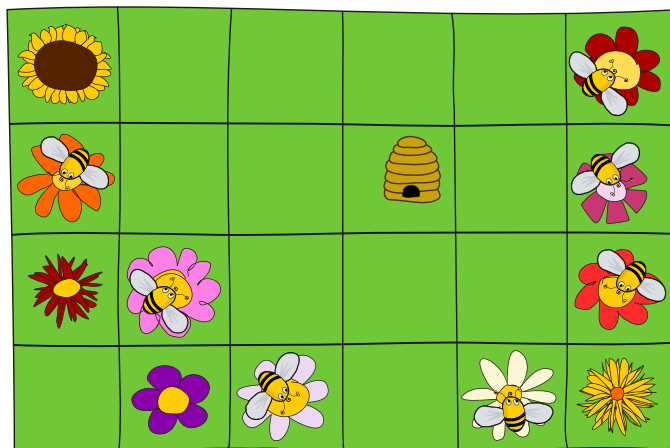
Entoure les fleurs qui peuvent être atteintes en 30 minutes au maximum depuis la ruche.





Solution

Les fleurs sur lesquelles une abeille est posée peuvent être atteinte en 30 minutes au maximum depuis la ruche :



L'image ci-dessous montre pour chaque fleur le nombre de minutes dont l'abeille a besoin pour y arriver en partant de la ruche. En une demi-heure, l'abeille peut donc atteindre toutes les cases dans lesquelles 10, 20 ou 30 est écrit.



Pour remplir les cases avec les nombres, on procède ainsi : dans les cases à côté de la ruche, on écrit 10, car l'abeille met 10 minutes à y arriver depuis la ruche. Ensuite, on écrit 20 dans toutes les cases vides à côté des cases « 10 », car l'abeille met 10 minutes pour passer d'une case à une autre. On continue à faire comme cela. On écrit donc 30 dans toutes les cases vides à côté des cases « 20 », puis 40 dans toutes les cases vides à côté des cases « 30 », et pour finir 50 dans toutes les cases vides à côté des cases « 40 ».

C'est de l'informatique !

Pour résoudre l'exercice, nous avons calculé pour chaque case le temps dont l'abeille a besoin pour y arriver depuis la ruche. D'abord, on détermine quelles cases sont atteignables en 10 minutes. On



les utilise ensuite pour déterminer quelles cases sont atteignables en 20 minutes. À l'aide des cases éloignées de 20 minutes, on trouve les cases atteignables en 30 minutes, et ainsi de suite.

Nous utilisons donc des résultats déjà calculés et enregistrés (les nombres dans les cases remplies) pour calculer les résultats suivants (les nombres dans les cases voisines encore vides). Ce principe très général s'appelle *programmation dynamique*. L'ordre dans lequel les résultats sont calculés est pour cela en général très important. Il faut aussi y faire attention pour le vol de l'abeille.

Dans l'exercice, l'abeille ne vole que vers le haut, le bas, la gauche ou la droite en 10 minutes. C'est un peu irréaliste, car en réalité, une abeille vole sûrement aussi en diagonale par dessus les cases. Avec cette hypothèse plus réaliste, les cases atteignables en 30 minutes seraient délimitées par un cercle et non par un losange comme dans l'exercice.

La mesure de distance habituelle qui génère un cercle s'appelle *distance euclidienne*. La mesure de distance utilisée dans l'exercice avec laquelle on ne peut se déplacer que verticalement ou horizontalement sur des carrés s'appelle la *distance de Manhattan* (le nom vient de l'arrangement quadrillé des villes modernes comme Manhattan).






Mots clés et sites web



- Programmation dynamique : https://fr.wikipedia.org/wiki/Programmation_dynamique
- Distance euclidienne : [https://fr.wikipedia.org/wiki/Distance_\(mathématiques\)](https://fr.wikipedia.org/wiki/Distance_(mathématiques))
- Distance de Manhattan : https://fr.wikipedia.org/wiki/Distance_de_Manhattan
- https://fr.wikipedia.org/wiki/Espace_euclidien

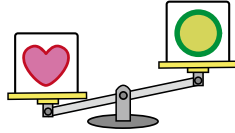




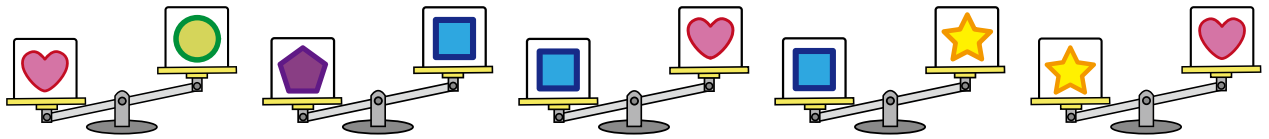
12. Lourdes comparaisons

Cinq boîtes sont marquées de cinq symboles différents : , , ,  et .

Une balance est utilisée pour comparer deux boîtes. La comparaison suivante montre par exemple que  est plus lourde que .



En tout, cinq comparaisons ont lieu :



Quelle est la boîte la plus lourde ?

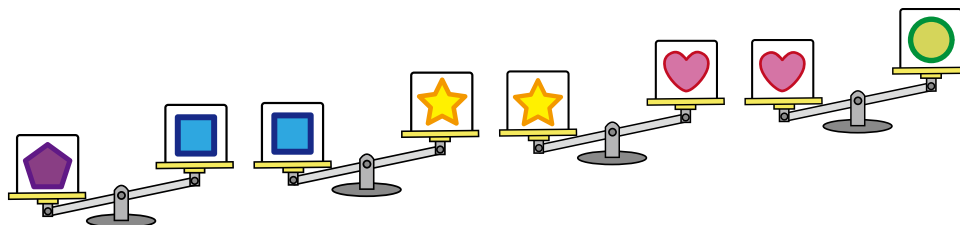
- A)  B)  C)  D)  E) 



Solution

La boîte C) avec le pentagone est la plus lourde.

L'image suivante montre quatre des cinq comparaisons faites et toutes les boîtes :



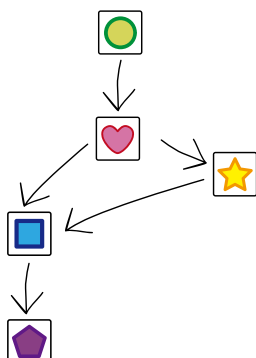
Comme cela, on voit tout de suite que : la boîte avec le pentagone est plus lourde que la boîte avec le carré . La boîte avec le carré est plus lourde que la boîte avec l'étoile . La boîte avec l'étoile est plus lourde que la boîte avec le cœur . Et finalement, la boîte avec le cœur est plus lourde que la boîte avec le cercle .

On peut déduire de cela que la boîte avec le pentagone est plus lourde que chacune des autres. Cela vient d'une propriété spéciale de la comparaison de poids : Si A est plus lourd que B et que B est plus lourd que C, alors A est aussi plus lourd que C. Cette propriété évidente s'appelle la *transitivité*.

Il existe une méthode maline pour raccourcir cet exercice. Comme on cherche *la* boîte la plus lourde, il suffit de chercher la boîte qui n'est jamais plus légère qu'une autre, et cela n'est vrai que de la boîte avec le pentagone .

C'est de l'informatique !

En fin de compte, il s'agit dans cet exercice de trier des objets quelconques. En informatique, on utilise souvent des *graphes* spéciaux pour trier, qui sont composé de *nœuds* (les objets à trier) et d'*arêtes* (les comparaisons entre les objets). Dans cet exercice, les objets sont les boîtes et les comparaisons sont les pesées. En dessinant les arêtes comme des flèches montrant l'objet qui est plus lourd, on obtient le graphe suivant pour cet exercice :



Les objets doivent à présent être arrangés sur une ligne de manière à ce que les flèches aillent toujours de gauche à droite. Un tel arrangement s'appelle un *tri topologique*. On obtient un tri topologique



facilement en enlevant étape par étape un objet sur lequel n'arrive aucune flèche, puis en mettant les objets ainsi enlevés les uns après les autres dans le même ordre.

Mais attention : ce ne sont pas tous les graphes qui ont un tri topologique. Il n'en existe par exemple pas s'il y a un endroit dans le graphe où trois arêtes fléchées sont dirigées de manière à former un cercle lorsqu'on les suit.

Mots clés et sites web

- Transitivité : https://fr.wikipedia.org/wiki/Relation_transitive
- Graphe : https://fr.wikipedia.org/wiki/Théorie_des_graphes
- Tri topologique : https://fr.wikipedia.org/wiki/Tri_topologique



A. Auteur·e·s des exercices

 Serge Adam	 Taina Lehtimäki
 Faisal Al-Sudani	 Marielle Léonard
 Carlo Bellettini	 Judith Lin
 Linda Björk Bergsveinsdóttir	 Lynn Liu
 Daniela Bezáková	 Vu Van Luan
 Sarah Chan	 Hiroki Manabe
 Marios O. Choudary	 Hamed Mohebbi
 Kris Coolsaet	 Kwangsik Moon
 Valentina Dagienė	 Anna Morpurgo
 Christian Datzko	 Xavier Muñoz
 Susanne Datzko	 Hiroyuki Nagataki
 Hanspeter Erni	 Tom Naughton
 Lidia Feklistova	 Ágnes Erdősne Németh
 Fabian Frei	 Gabriel Parriaux
 Gerald Futschek	 Elsa Pellet
 Jens Gallenbacher	 Jean-Philippe Pellet
 Juraj Hromkovič	 Margot Phillipps
 Alisher Ikramov	 Wolfgang Pohl
 Tiberiu Iorgulescu	 Pedro Ribeiro
 Takeharu Ishizuka	 Chris Roffey
 Ungyeol Jung	 Peter Rossmannith
 Vaidotas Kinčius	 Vipul Shah
 Sophie Koh	 Maiko Shimabuku
 Dennis Komm	 Peter Tomcsányi
 Chia-Yi Ku	 Monika Tomcsányiová
 Regula Lacher	 Meng-ting Tsai



Michael Weigend



Jonas Winckler



B. Sponsoring : Concours 2020

HASLERSTIFTUNG <http://www.haslerstiftung.ch/>



<http://www.baerli-biber.ch/>



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



Kanton Zürich
Volkswirtschaftsdirektion
Amt für Wirtschaft und Arbeit

Standortförderung beim Amt für Wirtschaft und Arbeit Kanton Zürich



i-factory (Musée des transports, Lucerne)



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



<http://www.oxocard.ch/>
OXOcard
OXON



<https://educatec.ch/>
educaTEC



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



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



hep/ haute
école
pédagogique
vaud

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

PH LUZERN
PÄDAGOGISCHE
HOCHSCHULE

<http://www.phlu.ch/>
Pädagogische Hochschule Luzern

n|w Fachhochschule
Nordwestschweiz

<https://www.fhnw.ch/de/die-fhnw/hochschulen/ph>
Pädagogische Hochschule FHNW

Scuola universitaria professionale
della Svizzera italiana

SUPSI

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

z — hdk
—
Zürcher Hochschule der Künste
Game Design

<https://www.zhdk.ch/>
Zürcher Hochschule der Künste



C. Offres ultérieures

010100110101011001001001
010000010010110101010011
010100110100100101000101
001011010101001101010011
010010010100100100100001

SS!E

www.svia-ssie-ssii.ch
schweizerischervereinfürinformatikind
erausbildung//sociétésuissepourl'infor
matique dans l'enseignement//societasviz
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.