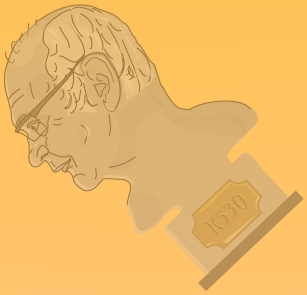
INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Exercices et solutions 2018
Tous les âges

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

Éditeurs :
Gabriel Parriaux, Jean-Philippe Pellet, Elsa Pellet, Julien Ragot, Christian Datzko, Susanne Datzko, Hanspeter Erni



SSIE
www.svia-ssie-ssii.ch
schweizerischervereinfürinformatikinderausbildung//sociétésuissepourl'informatiquedansl'enseignement//societàsvizzeraperl'informaticanell'insegnamento

Ont collaboré au Castor Informatique 2018

Andrea Adamoli, Christian Datzko, Susanne Datzko, Olivier Ens, Hanspeter Erni, Martin Guggisberg, Carla Monaco, Gabriel Parriaux, Elsa Pellet, Jean-Philippe Pellet, Julien Ragot, Beat Trachsler.

Nous adressons nos remerciements à :
Juraj Hromkovič, Urs Hauser, Regula Lacher, Jacqueline Staub : ETHZ
Andrea Maria Schmid, Doris Reck : PH Luzern
Gabriel Thullen : Collège des Colombières
Valentina Dagienė : Bebras.org
Hans-Werner Hein, Ulrich Kiesmüller, Wolfgang Pohl, Kirsten Schlüter, Michael Weigend : Bundesweite Informatikwettbewerbe (BWINF), Allemagne
Chris Roffey : University of Oxford, Royaume-Uni
Anna Morpurgo, Violetta Lonati, Mattia Monga : ALaDDIn, Università degli Studi di Milano, Italie
Gerald Futschek, Wilfried Baumann : Oesterreichische Computer Gesellschaft, Austria
Zsuzsa Pluhár : ELTE Informatikai Kar, Hongrie
Eljakim Schrijvers, Daphne Blokhuis, Arne Heijenga, Dave Oostendorp, Andrea Schrijvers : Eljakim Information Technology bv, Pays-Bas
Roman Hartmann : hartmannGestaltung (Flyer Castor Informatique Suisse)
Christoph Frei : Chragokyberneticks (Logo Castor Informatique Suisse)
Andrea Adamoli (page web)
Andrea Leu, Maggie Winter, Brigitte Maurer : Senarclens Leu + Partner

La version allemande des exercices a également été utilisée en Allemagne et en Autriche.
L'adaptation française a été réalisée par Nicole Müller et Elsa Pellet et la version italienne par Andrea Adamoli.

INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Le Castor Informatique 2018 a été réalisé par la Société Suisse de l'Informatique dans l'Enseignement SSIE. Le Castor Informatique est un projet de la SSIE, aimablement soutenu par la Fondation Hasler.

HASLERSTIFTUNG

Tous les liens ont été vérifiés le 1er novembre 2018. Ce cahier d'exercice a été produit le 9 octobre 2019 avec avec le logiciel de mise en page LaTeX.

Préambule

Très bien établi dans différents pays européens depuis plusieurs années, le concours « Castor Informatique » a pour but d'éveiller l'intérêt des enfants et des jeunes pour l'informatique. En Suisse, le concours est organisé en allemand, en français et en italien par la SSIE, la Société Suisse pour l'Informatique dans l'Enseignement, et soutenu par la Fondation Hasler dans le cadre du programme d'encouragement « FIT in IT ».
Le Castor Informatique est le partenaire suisse du concours « Bebras International Contest on Informatics and Computer Fluency » (`https://www.bebras.org/`), initié en Lituanie.
Le concours a été organisé pour la première fois en Suisse en 2010. Le Petit Castor (années HarmoS 5 et 6) a été organisé pour la première fois en 2012.
Le Castor Informatique vise à motiver les élèves à apprendre l'informatique. Il souhaite lever les réticences et susciter l'intérêt quant à l'enseignement de l'informatique à l'école. Le concours ne suppose aucun prérequis quant à l'utilisation des ordinateurs, sauf de savoir naviguer sur Internet, car le concours s'effectue en ligne. Pour répondre, il faut structurer sa pensée, faire preuve de logique mais aussi de fantaisie. Les exercices sont expressément conçus pour développer un intérêt durable pour l'informatique, au-delà de la durée du concours.
Le concours Castor Informatique 2018 a été fait pour cinq tranches d'âge, basées sur les années scolaires :

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

Les élèves des années HarmoS 5 et 6 avaient 9 exercices à résoudre : 3 faciles, 3 moyens, 3 difficiles. Les élèves des années HarmoS 7 et 8 avaient, quant à eux, 12 exercices à résoudre (4 de chaque niveau de difficulté). Finalement, chaque autre tranche d'âge devait résoudre 15 exercices (5 de chaque niveau de difficulté).
Chaque réponse correcte donnait des points, chaque réponse fausse réduisait le total des points. Ne pas répondre à une question n'avait aucune incidence sur le nombre de points. Le nombre de points de chaque exercice était fixé en fonction du degré de difficulté :

	Facile	Moyen	Difficile
Réponse correcte	6 points	9 points	12 points
Réponse fausse	−2 points	−3 points	−4 points

Utilisé au niveau international, ce système de distribution des points est conçu pour limiter le succès en cas de réponses données au hasard.
Chaque participant·e obtenait initialement 45 points (ou 27 pour la tranche d'âge « Petit Castor », et 36 pour les années HarmoS 7 et 8).
Le nombre de points maximal était ainsi de 180 (ou 108 pour la tranche d'âge « Petit Castor », et 144 pour les années HarmoS 7 et 8). Le nombre de points minimal était zéro.
Les réponses de nombreux exercices étaient affichées dans un ordre établi au hasard. Certains exercices ont été traités par plusieurs tranches d'âge.

Pour de plus amples informations :

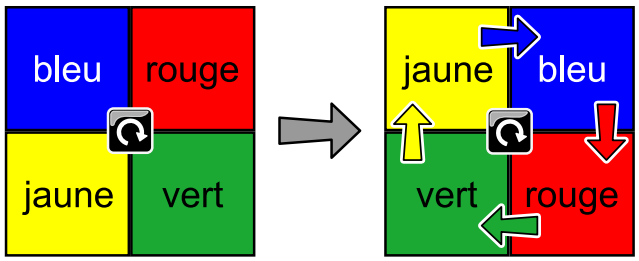
SVIA-SSIE-SSII Société Suisse de l'Informatique dans l'Enseignement
Castor Informatique

Gabriel Parriaux
https://www.castor-informatique.ch/fr/kontaktieren/
https://www.castor-informatique.ch/
 https://www.facebook.com/informatikbiberch

Table des matières

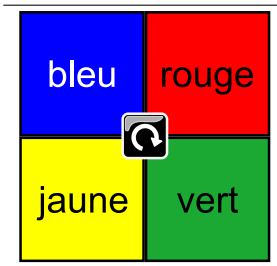
1. Jacques a dit

Chaque fois que Jacques appuie sur le bouton central, les carrés se déplacent comme montré dans l'exemple :

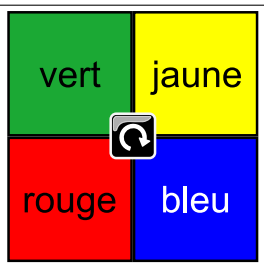


Jacques appuie deux fois de plus sur le bouton central après l'exemple précédent. Où se trouvent alors les carrés ?

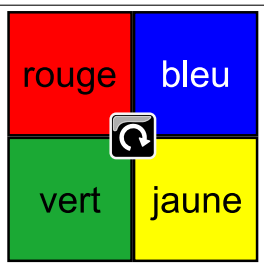
A)



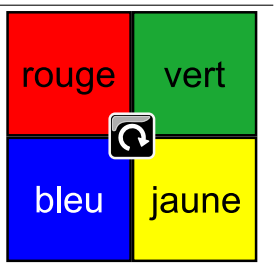
B)



C)

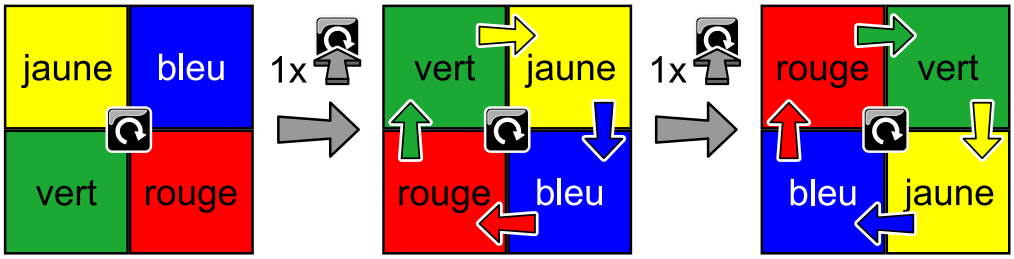


D)



Solution

La bonne réponse est D) :



C'est de l'informatique !

Cet exercice décrit une machine à quatre positions, chacune possédant un statut défini : rouge, vert, bleu ou jaune. Le statut de chaque position change dans l'ordre rouge → bleu → jaune → vert → rouge lorsque le bouton est actionné. En informatique, une telle machine est appelée *automate fini* ou *automate avec nombre fini d'états*. Les êtres humains considèrent plutôt la position dans laquelle chaque carré coloré se retrouve, c'est-à-dire le fait qu'ils se déplacent dans le sens des aiguilles d'une montre.
Il existe beaucoup de jeux durant lesquels il faut suivre des règles données, « Jacques a dit » en est un exemple. Le meneur ou la meneuse de jeu donne des tâches aux joueurs qui ne doivent être accomplies que si les consignes sont précédées de « Jacques a dit ». Par exemple, si le meneur ou la meneuse de jeu dit « tirez la langue », personne n'obéit, alors que s'il ou elle dit « Jacques a dit : sautez sur un pied », tout le monde saute sur un pied. Celui ou celle qui fait une erreur est éliminé.

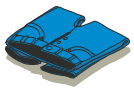
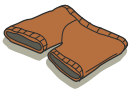
Mots clés et sites web

Automate fini

- `https://fr.wikipedia.org/wiki/Jacques_a_dit`
- `https://fr.wikipedia.org/wiki/Automate_fini`

2. La pile d'habits

La maman castor empile les habits de son fils Bruno sur la table.

Chemise	Maillot	Pantalon	Caleçon	Bretelles	Chaussettes	Chaussures
						

Bruno enfile ses habits dans l'ordre dans lequel ils sont posés sur la table. Il commence toujours avec l'habit en haut de la pile. Bruno ne veut pas porter ses bretelles sous sa chemise.
Quelle pile d'habits peut être utilisée par Bruno ?

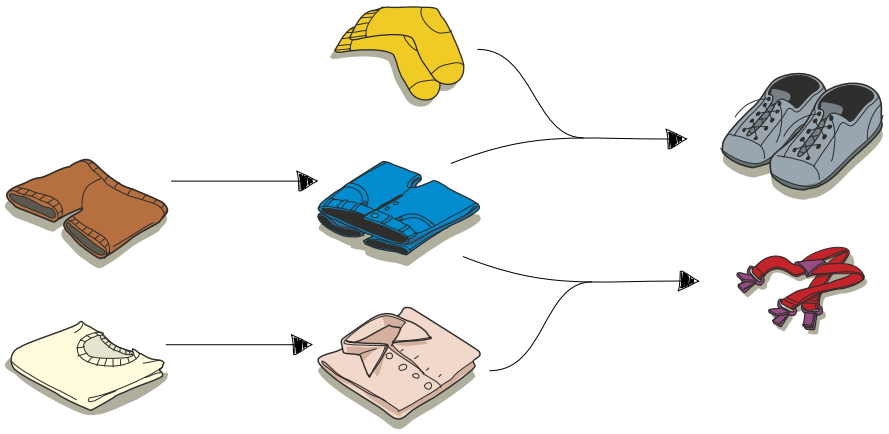
A)	B)	C)	D)
			

Solution

B) est la bonne pile d'habits.
Pour trouver la solution, nous devons commencer par l'habit du haut de la pile et vérifier que l'ordre dans lequel les habits sont empilés correspond aux restrictions (la chemise doit être posée plus haut que les bretelles).
Les réponses A), C) et D) sont fausses, entre autres car les bretelles seront enfilées avant la chemise.

C'est de l'informatique !

« Tu dois ouvrir la porte avant de pouvoir entrer dans une pièce » est un exemple de restriction claire : la porte doit être ouverte avant que tu ne puisses entrer dans une pièce pour y prendre ce que tu veux.
Cet exercice peut être résolu en vérifiant quelle liste remplit la condition « l'habit X doit être enfilé avant l'habit Y ». Si une liste d'habits remplit toutes les conditions, elle est correcte. Si n'importe laquelle des conditions n'est pas remplie, elle n'est pas correcte.
Les piles d'habits de cet exercice ont plus de restrictions que la place des bretelles ; on ne peut par exemple pas enfiler les chaussures avant les chaussettes.

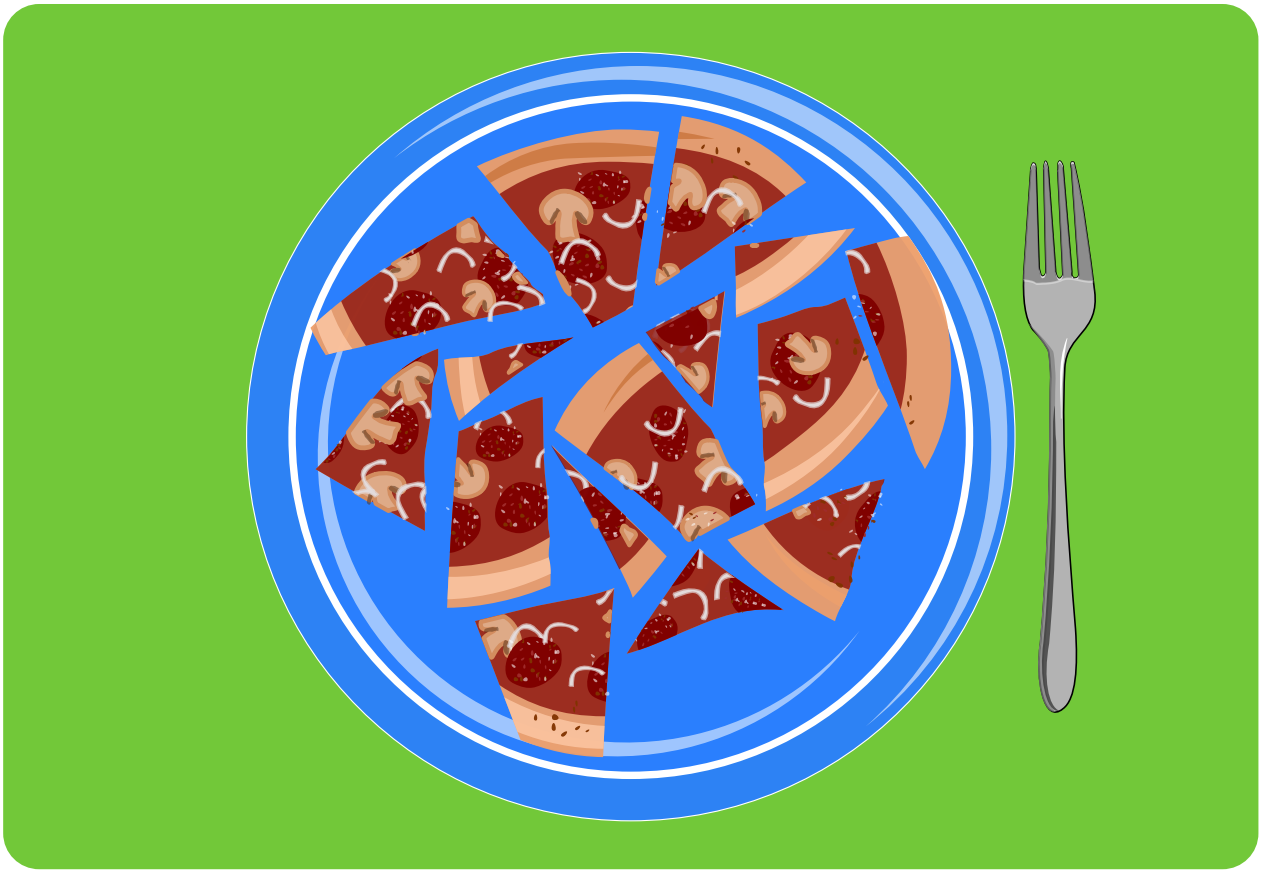


Mots clés et sites web

Ordonnancement
— `https://fr.wikipedia.org/wiki/Ordonnancement_de_travaux_informatiques`

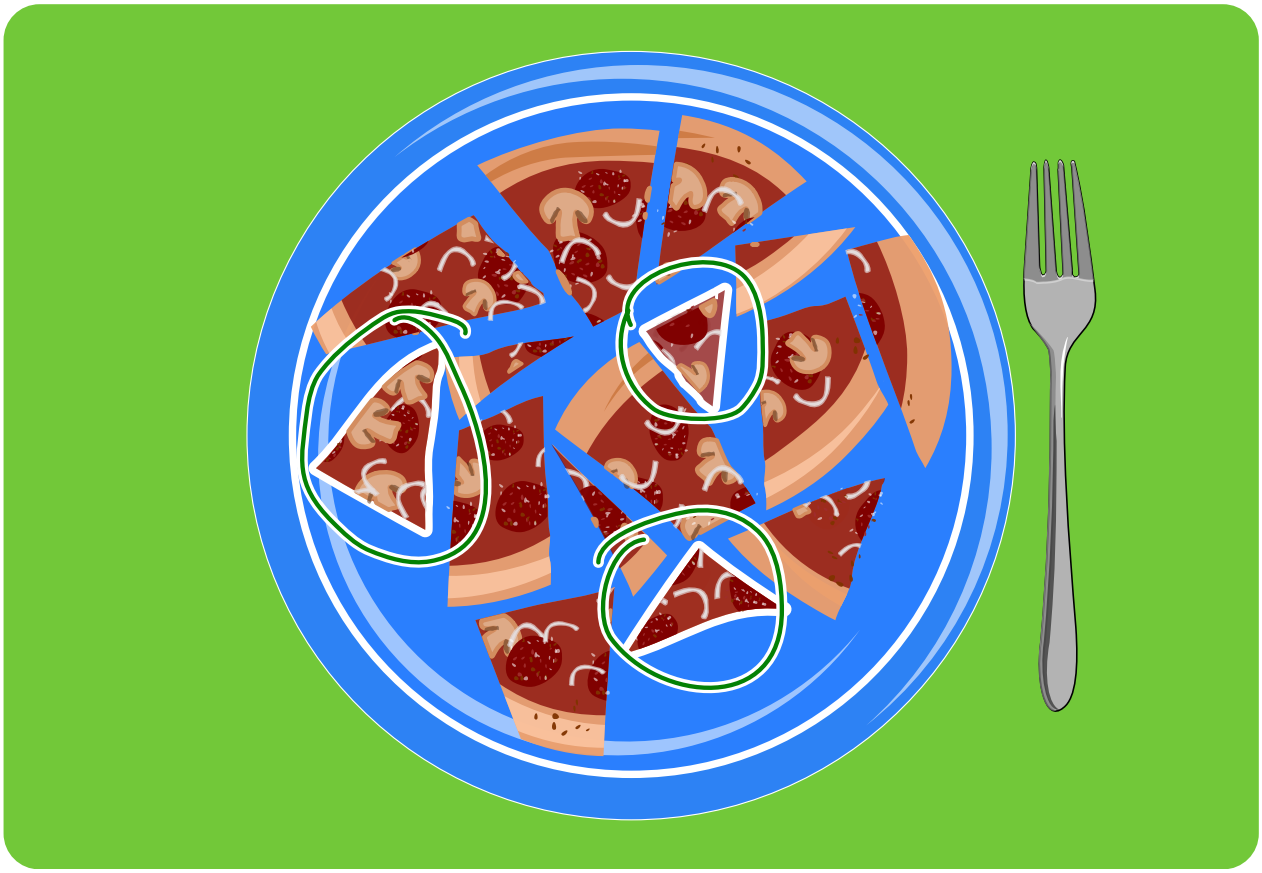
3. Pizza

La maman de Lucilla a coupé la pizza en parts. Lucilla aimerait tout manger avec les doigts, mais sa maman lui demande de manger les parts sans croûte avec une fourchette. De quelles parts s'agit-il ?



Solution

La bonne réponse est :



Les trois parts entourées n'ont pas de croûte, toutes les autres en ont une.

C'est de l'informatique !

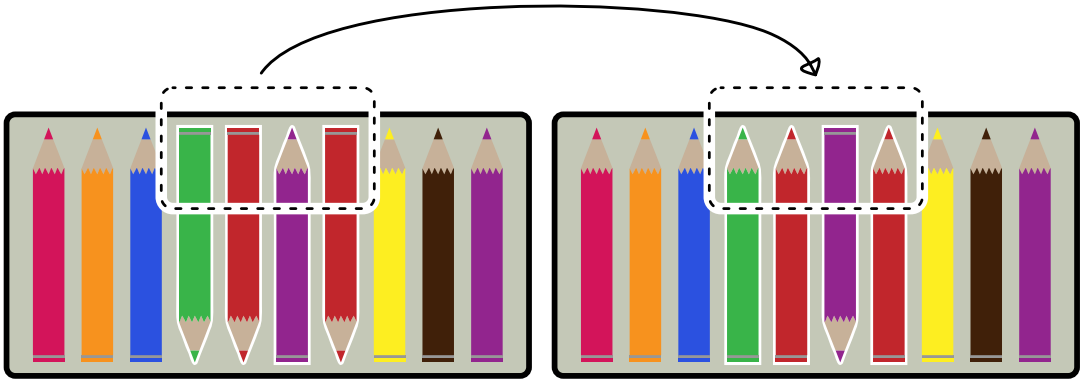
Pour chaque part de pizza, Lucilla doit se demander si elle a une croûte ou pas. Un ordinateur doit souvent prendre ce genre de décision sur beaucoup de choses. On appelle cela une *instruction conditionnelle*. Dans un programme informatique, ce serait par exemple écrit de cette façon :
SI la part a une croûte
    ALORS mange-la avec les doigts
    SINON mange-la avec une fourchette

Mots clés et sites web

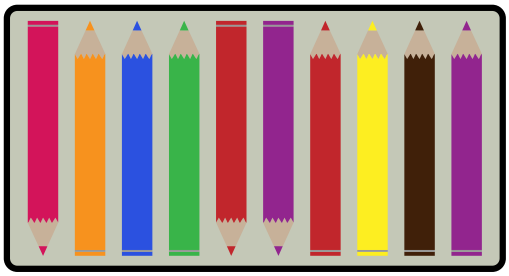
Instruction conditionnelle
- `https://fr.wikipedia.org/wiki/Instruction_conditionnelle_(programmation)`
- `https://fr.wikipedia.org/wiki/Branchement`

4. Les crayons de couleur d'Ada

Ada a une boîte de crayons de couleur avec 10 crayons. Certains sont rangés avec la pointe vers le haut, d'autres avec la pointe vers le bas. Ada aimerait que tous les crayons aient la pointe vers le haut.
Pour s'amuser, elle ne tourne que plusieurs crayons situés côte à côte d'un coup, mais jamais un seul. Dans l'exemple qui suit, elle tourne les quatrième, cinquième, sixième et septième crayons d'un coup.

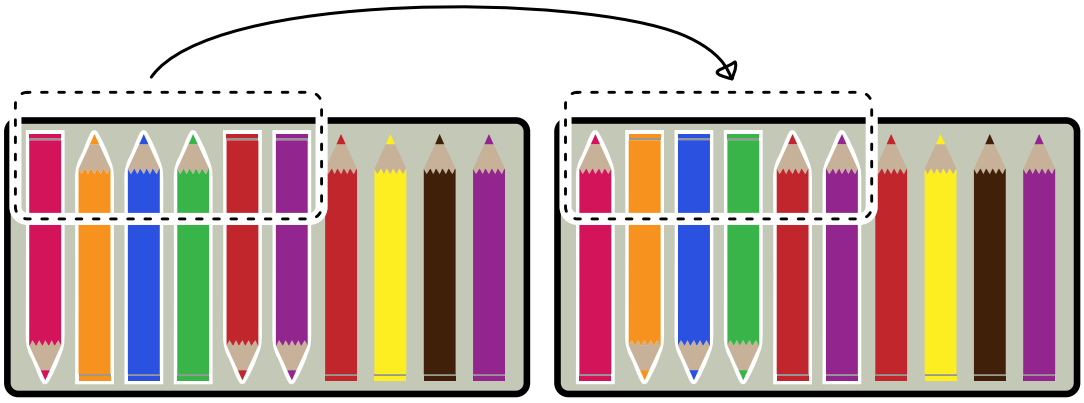


Ada aimerait que tous les crayons dans la boîte pointent vers le haut. Elle veut le faire en aussi peu d'étapes que possible. En combien d'étapes y arrive-t-elle ?

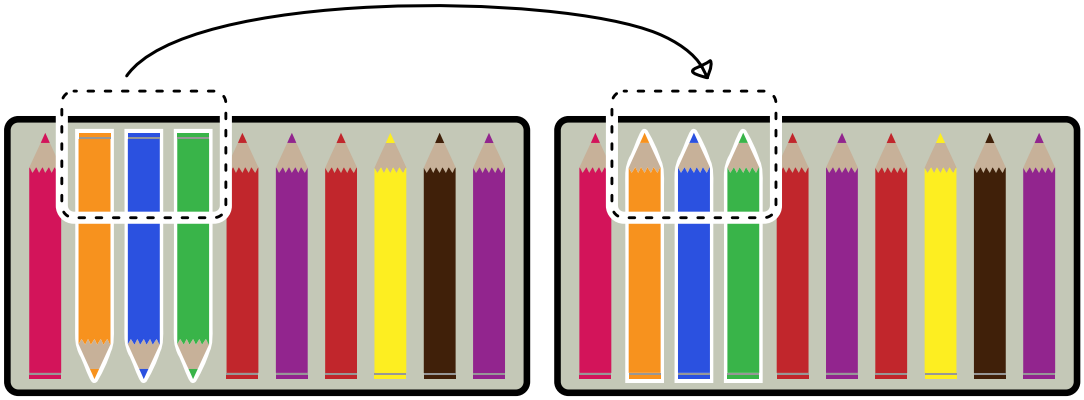


Solution

Il suffit de tourner deux groupes de crayons en suivant les règles du jeu. Ce n'est pas possible d'y arriver en une fois, car les crayons dans le mauvais sens ne sont pas tous côte à côte.
Mais si Ada tourne les crayons 1 à 6...



...puis les crayons 2 à 4...



...tous les crayons se retrouvent dans le sens souhaité.

C'est de l'informatique !

Ada a probablement le temps de ranger les crayons en plus de deux étapes, et si elle n'a plus envie de suivre les règles, elle peut bien sûr aussi tourner les crayons l'un après l'autre.
Ce n'est pas si facile pour les ordinateurs. Une fois qu'un ordinateur a été programmé, il se tient aux règles du jeu. De plus, les ordinateurs n'ont pas affaire à quelques crayons, mais doivent souvent appliquer les mêmes règles à beaucoup de données.
Ce principe est par exemple important pour la mémoire non volatile des ordinateurs. De nos jours, un SSD (Solid State Disk) est en général utilisé. Une unité de stockage de l'information ne peut pas être effacée seule, mais uniquement sous forme de bloc. Lors de l'écriture, l'ordinateur doit donc vérifier que le bloc entier est inutilisé, ou alors il doit d'abord lire les parties inutilisées, puis effacer le bloc entier avant d'y écrire les nouvelles et les anciennes données. C'est beaucoup plus long que si l'ordinateur sait déjà quels blocs sont inutilisés et les efface pendant qu'il n'a rien d'autre à faire. S'il a beaucoup à écrire, ça vaut la peine pour lui d'enregistrer les données en blocs entiers, car une seule unité ou un bloc complet de stockage de l'information lui prennent le même temps d'écriture.

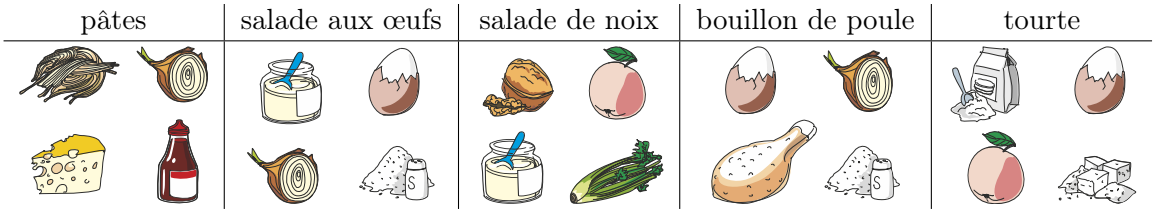
Mots clés et sites web

Efficience

- `https://www.codechef.com/problems/ADACRA`
- `https://fr.wikipedia.org/wiki/Comparaison_asymptotique`
- `https://it.wikipedia.org/wiki/O-grande`
- `https://fr.wikipedia.org/wiki/Trim_(informatique)`

5. Mets semblables

Un cuisinier aimerait préparer deux mets. Ces deux mets ne doivent pas être semblables. Pour le cuisinier, deux mets sont semblables s'ils ont au moins deux ingrédients en commun.



Quels mets sont semblables ?

A) Le bouillon de poule et les pâtes

B) Le bouillon de poule et la salade de noix

C) Le bouillon de poule et la salade aux œufs

D) La salade de noix et la tourte

Solution

La bonne réponse est C) Le bouillon de poule et la salade aux œufs.
Il y a de l'œuf, de l'oignon et du sel dans le bouillon de poule et dans la salade aux œufs.
Les autres combinaisons de mets ont au maximum un ingrédient en commun : le bouillon de poule et les pâtes contiennent les deux de l'oignon. Le bouillon de poule et la salade de noix n'ont pas d'ingrédient en commun, la salade de noix et la tourte non plus.

C'est de l'informatique !

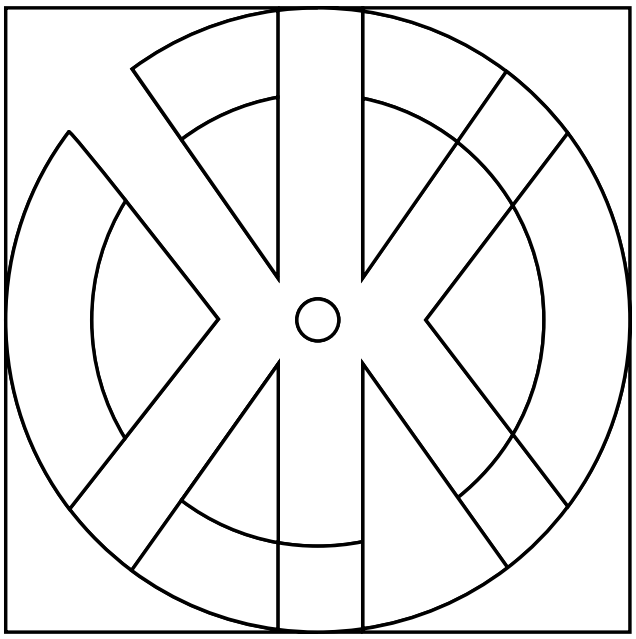
Il y a beaucoup de situations dans lesquelles il faut comparer des choses et déterminer ce qui est semblable et ce qui est différent. Par exemple, les biologistes comparent le génome des bactéries, les chimistes les propriétés de substances, les astronomes comparent des galaxies, des étoiles et des planètes, et ainsi de suite.
Afin de comparer des choses, il faut définir quelles propriétés l'on compare. On peut ensuite déterminer à partir de quand deux choses sont semblables ou non. De cette manière, on peut par exemple dire qu'une chaise et une table sont semblables parce que les deux sont en bois. On peut tout aussi bien dire qu'une table n'est pas faite pour s'asseoir, et qu'une chaise n'est pas faite pour y écrire une lettre (même si ces deux choses sont bien sûr possibles). On peut aussi dire que deux chaises en bois ne sont semblables que si elles sont faites du même bois.
Pour cet exercice, il faut comparer cinq mets composés de quatre ingrédients chacun. Les biologistes, chimistes, astronomes et beaucoup d'autres scientifiques ne comparent pas un si petit nombre de choses, mais des milliers, millions ou milliards de choses qui peuvent posséder beaucoup de propriétés à considérer pour évaluer leur similarité. C'est ici que l'informatique entre en jeu en permettant la comparaison systématique d'un grand nombre de données d'après des critères de similarité prédéfinis.

Mots clés et sites web

Objet, propriété, critère de similarité, Big Data

- `https://fr.wikipedia.org/wiki/Similarité_(informatique)`
- `https://fr.wikipedia.org/wiki/Big_data`

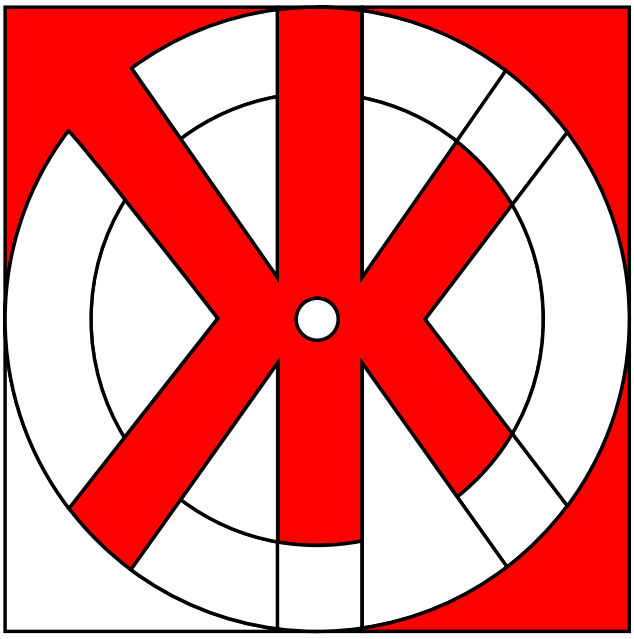
6. Colorier un motif



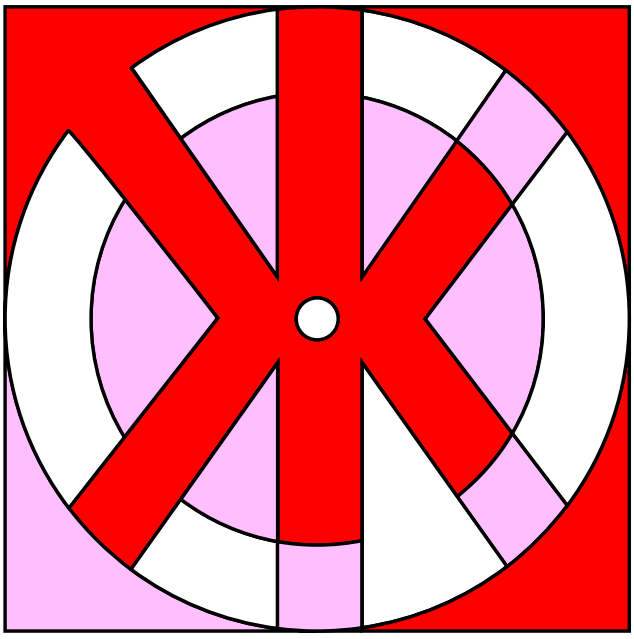
Les castors aimeraient bien colorier ce motif. Pour les aider, colorie les différentes surfaces du motif de sorte que chaque surface avoisinante présente une couleur différente. De plus, pour ce faire, choisis le moins de couleurs possibles.

Solution

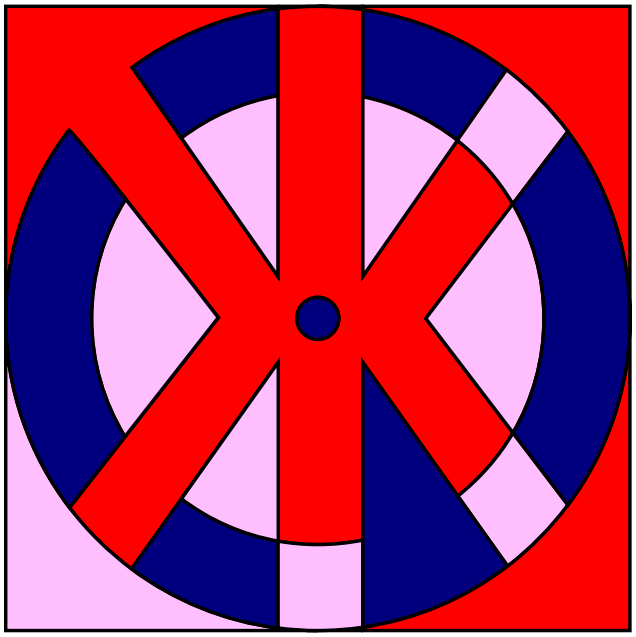
Il est possible de le faire à l'aide de trois couleurs différentes.
Il existe différentes solutions selon la première couleur que l'on choisit. Si l'on commence, par exemple, avec la couleur rouge dans l'angle supérieur gauche et que l'on colorie ensuite de la même couleur toutes les surfaces qui ne sont pas en contact direct avec elle, on obtient le résultat suivant :



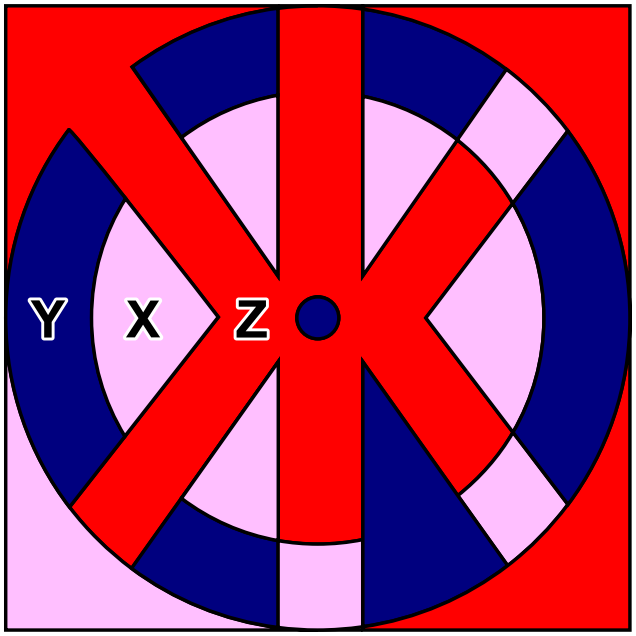
Si l'on continue avec une deuxième couleur, par exemple le rose, dans l'angle inférieur gauche et que l'on colorie chaque surface possible, on obtient le résultat suivant :



En principe, si l'on admet que le blanc est une couleur, on a ainsi déjà terminé, car avec les trois couleurs, rouge, rose et blanc, on a colorié le motif en respectant les deux conditions définies. Bien sûr, on est libre de colorier les surfaces blanches avec une troisième couleur (par exemple le bleu) :



Par contre, il ne serait pas possible de colorier le motif en question avec moins de trois couleurs. La surface X jouxte la surface Y, elles doivent donc être coloriées en différentes couleurs. Toutes deux délimitent la troisième surface Z et donc la couleur de celle-ci doit impérativement différer des deux autres.



C'est de l'informatique !

Quel est le nombre maximum de couleurs dont on a besoin pour colorier un nombre arbitraire de surfaces de sorte qu'aucune surface adjacente ne présente la même couleur ? La réponse correcte est : quatre couleurs suffisent tant que l'on n'autorise pas les « enclaves ». Une enclave est une zone partielle autonome qui appartient à une autre zone, mais qui n'y est pas reliée, comme par exemple la commune fribourgeoise d'Estavayer qui est, avec 11 autres communes, à l'intérieur du canton de Vaud, ou Büsingen sur le Rhin supérieur ou encore Campione d'Italia. À l'étranger, on pourrait aussi mentionner Baarle aux Pays-Bas et en Belgique.

Cette tâche présente le théorème des quatre couleurs qui est très difficile à prouver. Il y a 200 ans, on était arrivé à la conclusion que le nombre maximum était de cinq couleurs. Ce n'est qu'en 1976 que les mathématiciens Kenneth Appel et Wolfgang Haken ont prouvé qu'il suffisait de quatre couleurs. Ils ont utilisé des ordinateurs pour vérifier diverses exceptions et contre-exemples. Mais comme ce n'était plus possible de vérifier manuellement les calculs effectués par l'ordinateur, beaucoup de mathématiciens ont remis en cause l'utilisation de l'ordinateur pour ces types de preuves. Aujourd'hui encore, certains mathématiciens se demandent s'il est permis d'utiliser un ordinateur pour établir la véracité d'un énoncé mathématique ou pour prouver un théorème.
Dans la vie quotidienne, le théorème des quatre couleurs est très utile dans différents domaines, par exemple dans celui de l'aéronautique pour créer des plans de vol lorsque les avions sont assignés à des couloirs afin qu'ils aient toujours une distance suffisante, ou encore dans celui de la téléphonie mobile lorsque l'on assigne des plages de fréquences aux antennes de téléphonie mobile afin qu'elles n'interfèrent pas entre elles et que la réception ne se détériore pas malgré les nombreuses antennes.

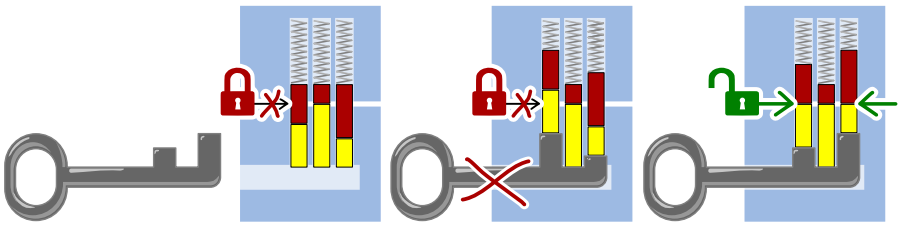
Mots clés et sites web

le théorème des quatre couleurs

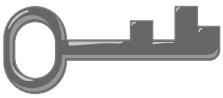
- `https://fr.wikipedia.org/wiki/Théorème_des_quatre_couleurs`
- `http://www.mathepedia.de/Vier-Farben-Satz.html`
- `https://en.wikipedia.org/wiki/Enclave_and_exclave`
- `https://en.wikipedia.org/wiki/Baarle`

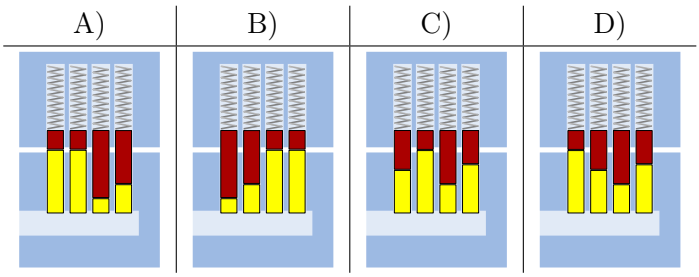
7. Serrure

Henry travaille chez un serrurier. Les serrures fonctionnent de la manière suivante :



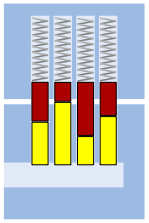
Quelle serrure peut être ouverte avec la clé suivante ?



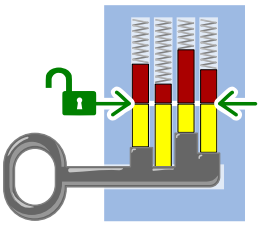


Solution

La bonne réponse est C) :



Lorsque la clé est introduite, les crans de la clé poussent toutes les goupilles de la serrure à la même hauteur de façon à ce que le cylindre puisse être tourné :



C'est de l'informatique !

Pour qu'une clé puisse ouvrir et fermer une serrure, il faut que chacun de ses membres corresponde aux éléments de la serrure. Pour cela, les longs crans de la clé doivent se trouver à la hauteur des courtes goupilles de la serrure, et les crans courts à la hauteur des longues goupilles. Les deux motifs doivent correspondre l'un à l'autre. Dans cet exercice, c'est le cas lorsqu'ils sont exactement opposés. La recherche d'un motif ou d'une forme spécifique est une tâche fondamentale de l'informatique, par exemple la recherche d'un mot dans un texte ou d'images similaires.

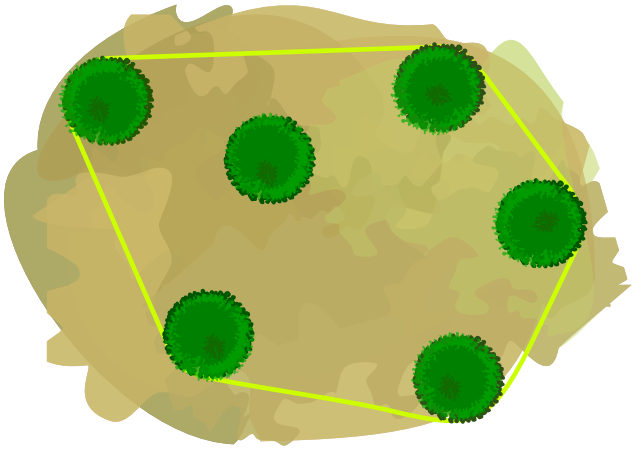
Mots clés et sites web

Reconnaissance de formes, serrure

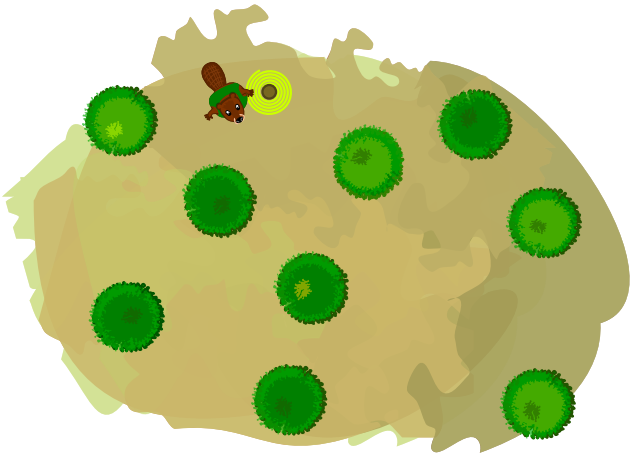
- `https://fr.wikipedia.org/wiki/Serrure_à_goupilles`
- `https://fr.wikipedia.org/wiki/Reconnaissance_de_formes`

8. Ensemble buissonnier

Les castors entourent d'une corde le groupe de buissons qu'ils veulent abattre.
Hier, ils voulaient abattre six buissons. La corde ne touchait que cinq des buissons. Voilà à quoi cela ressemblait vu du ciel :



Aujourd'hui, les castors veulent abattre ces neuf buissons :

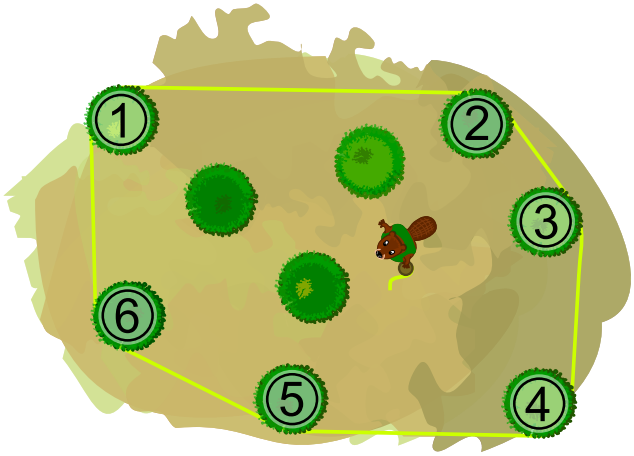


Combien de buissons la corde touche-t-elle cette fois-ci ?

A) 3 buissons
B) 4 buissons
C) 5 buissons
D) 6 buissons
E) 9 buissons

Solution

La réponse D) est correcte. Les castors tendent la corde autour des buissons de la manière suivante :



La corde touche les six buissons numérotés.

C'est de l'informatique !

La corde entourant les buissons délimite la plus petite surface sur laquelle se trouvent tous les buissons qui doivent être abattus. La seule restriction est que les limites de la surface doivent être des lignes droites. Si les buissons sur l'image étaient des points, la corde aurait la forme d'un hexagone.
Le plus petit polygone contenant tous les points d'un ensemble donné est appelé l'*enveloppe convexe* de cet ensemble de points. Ici, *convexe* est utilisé pour quelque chose qui s'étend vers l'extérieur comme la lentille convexe d'une loupe. Une *enveloppe* est quelque chose qui entoure autre chose sans être plus grand que nécessaire, comme la peau entoure le corps. La corde des castors entoure donc les buissons comme une enveloppe convexe.
En informatique, on détermine souvent quelle est l'enveloppe convexe d'un ensemble de points :

- Reconnaissance de forme : l'image contient-elle un visage ?
- Reconnaissance de l'écriture manuelle : est-ce qu'un signe écrit est la lettre B ?
- Systèmes d'information géographique : quelle est la taille d'une plaine d'inondation ou d'un réseau de drainage ?
- Emballage : Quelle est la plus petite quantité de matériel nécessaire à l'emballage d'un objet ?

Il existe des méthodes informatiques qui déterminent l'enveloppe convexe d'un ensemble de points de manière efficace. Ces méthodes fonctionnent également pour un très grand nombre de points.

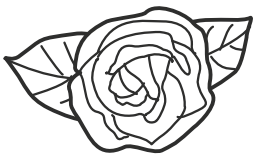
Mots clés et sites web

Graphe, enveloppe convexe

- `https://fr.wikipedia.org/wiki/Enveloppe_convexe`
- `https://fr.wikipedia.org/wiki/Calcul_de_l'enveloppe_convexe`
- `https://brilliant.org/wiki/convex-hull`

9. Les fleurs de Clara

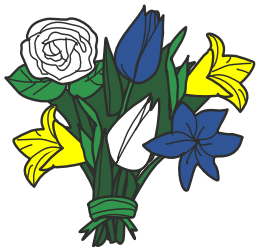
Clara va chez le fleuriste, car elle aime les bouquets de fleurs colorés. Elle y trouve les sortes de fleurs suivantes :

glaïeuls	lys	tulipes	roses
			

Chaque sorte de fleur est disponible en trois couleurs : blanc, bleu et jaune.
Clara aimerait un bouquet de six fleurs qui remplit les conditions suivantes :

1. Il doit y avoir deux fleurs de chaque couleur (blanc, bleu, jaune),
2. Les fleurs de la même sorte ne doivent jamais être de la même couleur,
3. Il ne doit pas y avoir plus de deux fleurs de la même sorte.

Quel est le bouquet qui remplit les trois conditions ?

A)	B)	C)	D)
			

Solution

La bonne réponse est D). Le bouquet A) contient trois fleurs blanches (contrevient à la règle 1), le bouquet B) trois roses (contrevient à la règle 3), et le bouquet C) deux glaïeuls jaunes (contrevient à la règle 2).

C'est de l'informatique !

Les problèmes informatiques courants sont décrits par un ensemble de restrictions, la tâche étant de trouver une solution qui respecte toutes ou le plus possible de ces restrictions.
On peut considérer des tâches plus complexes lors desquelles des opérateurs logiques comme le connecteur ET (A ET B signifie que les deux conditions A et B doivent être remplies, comme les trois règles dans notre exercice) ou le connecteur OU (A OU B signifie que seulement une des deux conditions doit être remplie).

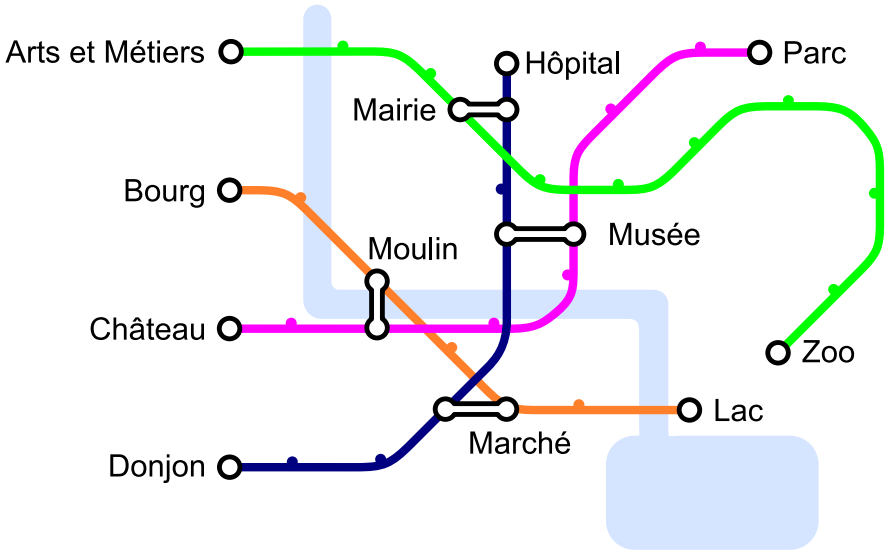
Mots clés et sites web

Conditions, opérateurs logiques

- `https://bookofbadarguments.com/`
- `https://fr.wikipedia.org/wiki/Connecteur_logique`
- `https://www.iep.utm.edu/prop-log/`

10. Réseau de lignes

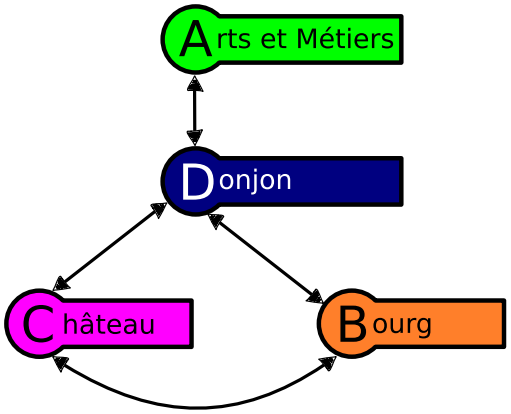
Dans la ville des castors, il y a quatre lignes avec pour point de départ quatre stations différentes : les stations « Arts et Métiers », « Bourg », « Château » et « Donjon ». Chaque ligne comprend au moins une station de transit qui permet de changer de ligne : la station « Musée », la station « Marché », la station « Moulin » et la station « Mairie ».



Aujourd'hui, Jean se rend au zoo. Il sait qu'il ne va changer de ligne qu'une seule fois. De quelle station de départ Jean est-il parti ?

Solution

La réponse correcte est « Donjon ». Quand on rebrousse chemin depuis la station « Zoo », on voit qu'il n'y a qu'une seule station de transit sur la ligne verte : « Mairie ». On en déduit que Jean est venu par la ligne bleue et que sa station de départ est « Donjon », étant donné qu'il n'a changé qu'une fois.
En informatique, il est possible de représenter le réseau de lignes à l'aide d'un graphe. Dans celui-ci, on représente les lignes par des nœuds et on les relie lorsqu'une station de transit permet de passer directement d'une ligne à l'autre. Par exemple, la ligne verte de « Arts et Métiers » à « Zoo », représentée par le A vert, est connectée à la ligne bleue de « Donjon » à « Hôpital », représentée par le D bleu, via la station de transit « Mairie » et donc le A est connecté au D dans ce schéma.



De cette ligne…	…on atteint ces lignes en un seul changement		
Arts et Métiers ↔ Zoo	Donjon ↔ Hôpital		
Bourg ↔ Lac	Château ↔ Place du parc	Donjon ↔ Hôpital	
Château ↔ Place du parc	Bourg ↔ Lac	Donjon ↔ Hôpital	
Donjon ↔ Hôpital	Bourg ↔ Lac	Château ↔ Place du parc	Arts et Métiers ↔ Zoo

Si on veut prendre la ligne « Arts et Métiers » ↔ « Zoo » pour arriver au zoo en ne changeant qu'une seule fois, on ne peut le faire qu'à partir de la station de départ « Donjon ». Ceci se lit dans le graphe en constatant que D est connecté uniquement à A, ou dans le tableau en voyant que A) n'apparaît qu'en regard de D), à la dernière ligne.

C'est de l'informatique !

Si tout cela te semble familier, c'est parce que beaucoup de réseaux de lignes de bus, de tramways ou de métros ressemblent à ce diagramme schématique qui représente les lignes et les stations de ces transports publics. Il s'agit là d'une véritable invention : en 1931, Henry Beck a élaboré un diagramme schématique pour le système de métro de Londres.
En informatique, un tel modèle abstrait est appelé un graphe. Un graphe se constitue par des nœuds (les stations) et des arêtes (le trajet entre deux stations). Dans notre tâche, il faut distinguer les nœuds qui présentent une ou deux arêtes (les stations de départ de fin ainsi que les stations intermédiaires) de ceux qui présentent un plus grand nombre d'arêtes (les stations de transit).

Dans la vie quotidienne, les graphes ont de nombreuses applications : les algorithmes élaborés d'un graphe peuvent par exemple résoudre des problèmes dans le domaine des réseaux sociaux, des guides routiers ou encore en matière de recherches sur les suggestions d'achats sur Internet. C'est la raison pour laquelle la maîtrise des graphes est une des compétences informatiques essentielles.

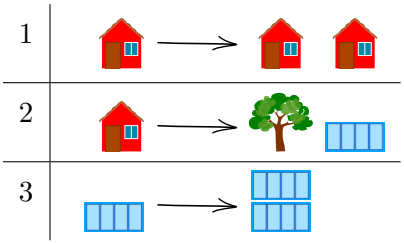
Mots clés et sites web

réseau de lignes, graphe

- `https://fr.wikipedia.org/wiki/Plan_du_métro_de_Londres`
- `https://fr.wikipedia.org/wiki/Graphe_(mathématiques_discrètes)`

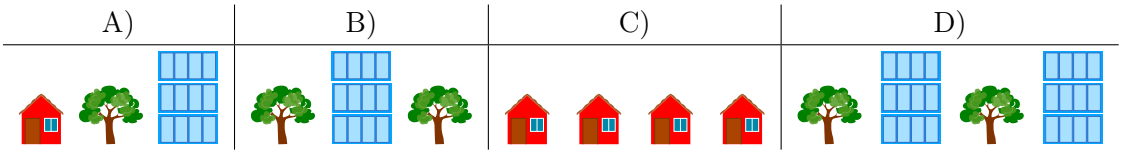
11. Planète Z

Les habitants de la planète Z construisent toutes leurs villes de la même manière. Ils commencent chaque ville avec une maison, puis remplacent les bâtiments construits les uns après les autres en suivant les règles suivantes :



Par exemple, en appliquant d'abord la règle 1, puis la règle 2, puis deux fois la règle 3, on obtient la ville à droite de l'image ci-dessous :

L'ordre dans lequel sont arrangés les bâtiments et les arbres ne peut pas être modifié.
Laquelle des villes suivantes ne *peut-elle pas se trouver sur la planète Z ?*



Solution

La bonne réponse est B). Les arbres ne sont plantés dans la ville qu'en suivant la règle 2, et la règle 2 spécifie qu'il doit y avoir un immeuble à droite de chaque arbre. Dans la ville B), il n'y a pas d'immeuble à droite du deuxième arbre. Comme il n'existe pas de règle permettant d'enlever un immeuble, la ville B) ne peut pas être sur la planète Z.
La ville A) peut être construite en appliquant la suite de règles 1, 2, 3, et 3.
La ville C) peut être construite en appliquant trois fois la règle 1.
La ville D) peut être construite en appliquant d'abord la règle 1, puis la règle deux sur chacune des deux maisons. Finalement, on applique deux fois la règle 3 sur chaque immeuble.

C'est de l'informatique !

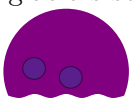
Les règles de cet exercices sont appelées règles de réécriture : un symbole ou un objet est remplacé par une série d'autres symboles ou objets. Si chaque règle ne remplace qu'un seul objet à la fois, le set de règles est appelé algébrique (ou non contextuel). Un symbole ou un objet est remplacé sans que le contexte (c'est-à-dire ce qui se trouve à sa gauche et à sa droite) ne joue de rôle.
En informatique, les règles de réécriture sont par exemple utilisées pour définir la syntaxe d'un langage de programmation. Les symboles ou objets sont des termes clés et les règles décrivent de quelle manière on peut les assembler en un programme (syntaxiquement) correct. Dans cet exercice, les symboles ou objets sont les maisons, les arbres et les immeubles. Les symboles ou objets et les règles de réécriture forment ensemble la grammaire décrivant un langage.
Lorsqu'un ordinateur traduit (compile) un programme en langage machine ou l'exécute directement (via un interpréteur pour les programmes écrits dans un langage de script), il commence par vérifier si le texte du programme suit bien les règles du langage de programmation. Il essaie donc, à l'aide d'un arbre syntaxique, de reconstruire les règles de réécriture qui transforment le symbole de départ (une maison dans cet exercice) en un texte de programme (les quatre réponses possibles dans cet exercice). Dans notre cas, c'est facilement réalisable, car les *mots* (des suites de symboles ou objets, dans cet exercice les villes) deviennent toujours plus grands et ne peuvent pas rétrécir.

Mots clés et sites web

Règle de réécriture, grammaire, langage algébrique

- `https://en.wikipedia.org/wiki/Production_(computer_science)`
- `https://fr.wikipedia.org/wiki/Langage_algébrique`
- `https://fr.wikipedia.org/wiki/Arbre_syntaxique`
- `https://fr.wikipedia.org/wiki/Grammaire_non_contextuelle`
- `https://fr.wikipedia.org/wiki/Règles_de_réécriture`

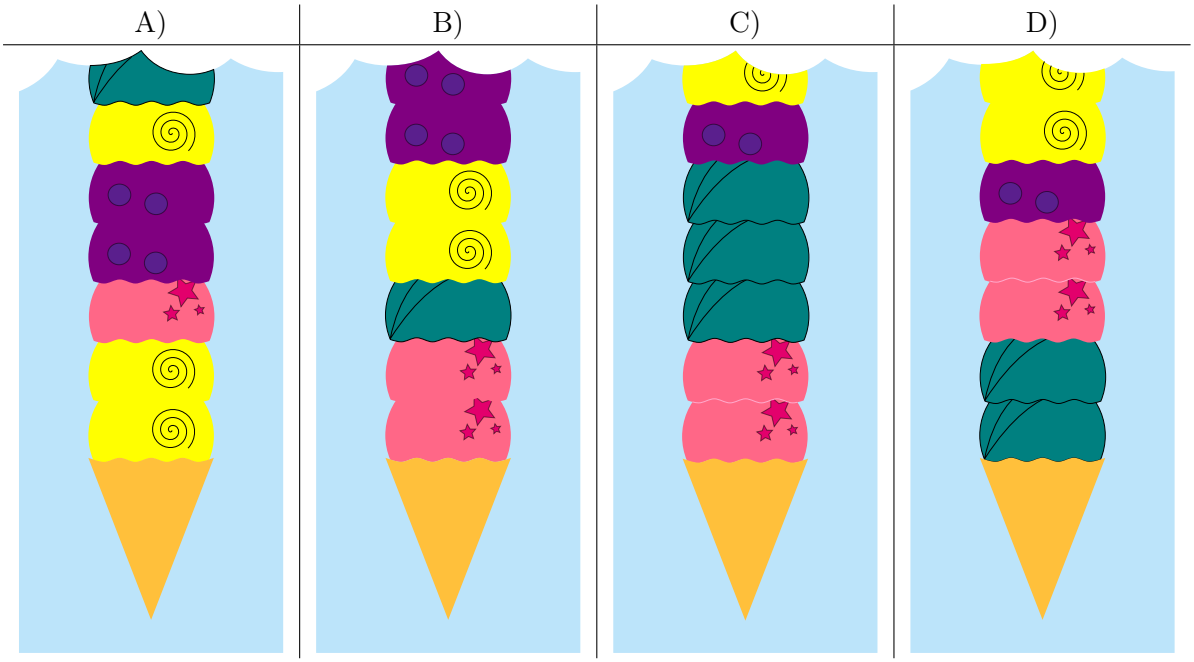
12. Glacier

Il y a deux glaciers sur la place du village. Ils offrent les quatre mêmes sortes de glace :  ,  ,  et  .

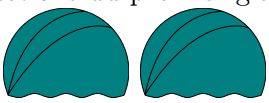
Le premier glacier suit les instructions suivantes pour préparer un cornet de glace :

1. Prends un cornet vide.
2. Choisis une sorte de glace au hasard et mets-en deux boules dans le cornet.
3. Ajoute une boule d'une des trois autres sortes de glace.
4. Si le nombre de boules souhaité est atteint, arrête. Sinon, recommence à l'étape 2.

Le deuxième glacier ne suit aucune règle.
L'image suivante représente les premières boules de quelques cornets. Lequel provient à coup sûr du deuxième glacier ?



Solution

La bonne réponse est D). C'est le seul cornet qui ne peut pas avoir été préparé en suivant les instructions du premier glacier. Il commence en accord avec les règles avec deux boules de la même sorte  suivies d'une boule d'une autre sorte  , mais ensuite viennent deux boules de deux sortes différentes  , alors que les instructions spécifient que deux boules de la même sorte devraient être ajoutées.
Les réponses A), B) et C) ne sont pas correctes. Tous ces cornets pourraient avoir été préparés d'après les instructions du premier glacier.

C'est de l'informatique !

Une série d'instructions permet de créer des motifs dans des cornets de glace, des textes ou des images. Les informaticiennes et informaticiens développent des programmes informatiques permettant de reconnaître des motifs et des variations dans ces motifs. Parfois, des motifs sont générés par la répétition d'instructions. Par exemple, ce simple motif  est généré par la répétition de  suivi de  . De tels motifs sont faciles à reconnaître. Le problème est plus compliqué dans cet exercice, car les instructions du premier glacier contiennent aussi des décisions laissées au hasard.
Il n'est de manière générale pas possible d'être sûr qu'une séquence a été générée par hasard ou en suivant une série d'instructions. Dans cet exercice, nous avons pu déterminer que l'un des cornets ne correspondait pas aux instructions et devait donc venir du deuxième glacier. On ne peut par contre jamais être sûr qu'une glace vient du premier glacier et pas du deuxième, car la composition correspondant aux instructions pourrait avoir été générée par hasard.

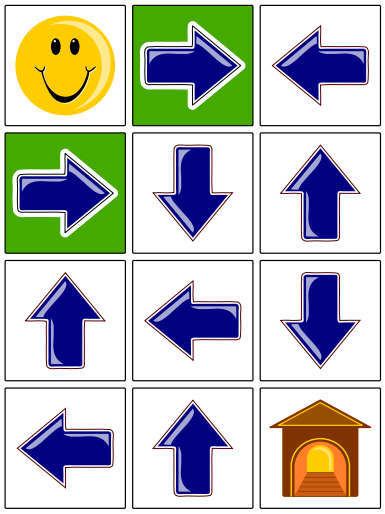
Mots clés et sites web

Reconnaissance de motifs

— `https://fr.wikipedia.org/wiki/Reconnaissance_de_formes`

13. Labyrinthe fléché

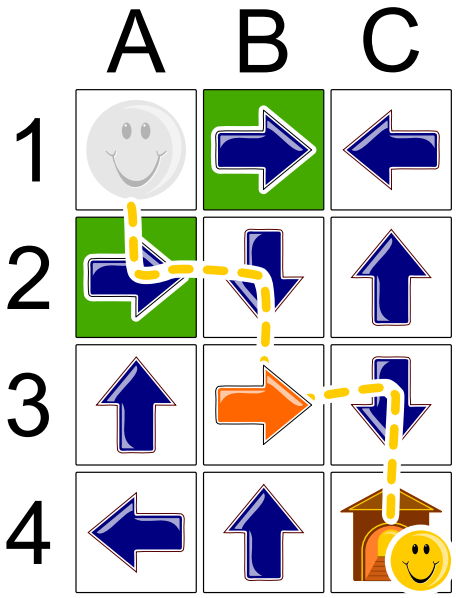
Le smiley aimerait rentrer à la maison, mais pour y arriver, il doit d'abord traverser un labyrinthe fléché. Il peut utiliser l'une des deux entrées (cases vertes). Lorsqu'il se trouve sur une case contenant une flèche, il doit quitter cette case dans le sens de la flèche. La position actuelle des flèches ne lui permet en aucun cas de rentrer à la maison.



Quelle flèche doit changer de direction pour permettre au smiley de rentrer à la maison ?

Solution

La flèche dont la direction doit changer est colorée en rouge. Le chemin du smiley jusqu'à la maison est indiqué par une ligne : A1 – A2 – B2 – B3 – C3 – C4.



Il est même possible de prouver que c'est la seule solution possible. Si l'on commence par la case cible C4 et se déplace à l'envers, on voit qu'il y a deux possibilités d'arriver à C4 : depuis C3 ou depuis B4. La flèche B4 ne pointe pas dans la bonne direction et doit donc être changée. Seulement, comme aucune des flèches voisines de B4 ne pointe en direction de B4, il faudrait changer la direction d'une deuxième flèche, ce qui n'est pas permis.
On ne peut donc arriver à C4 qu'en passant par la case C3. La flèche C3 pointe déjà en direction de la maison et ne doit pas être modifiée, mais les cases voisines de C3 ne contiennent pas de flèche pointant vers C3. Il faut donc changer la direction de la flèche B3 ou C2, les deux voisines de C3. Comme il n'y a pas de flèche pointant vers C2, C3 ne peut pas être atteinte en passant par C2 et en ne modifiant qu'une seule flèche. Il faut donc passer par la case B3. On peut atteindre B3 depuis l'une des deux entrées sans modifier la direction de flèches supplémentaires (A1 – A2 – B2). C'est donc la seule solution possible.

C'est de l'informatique !

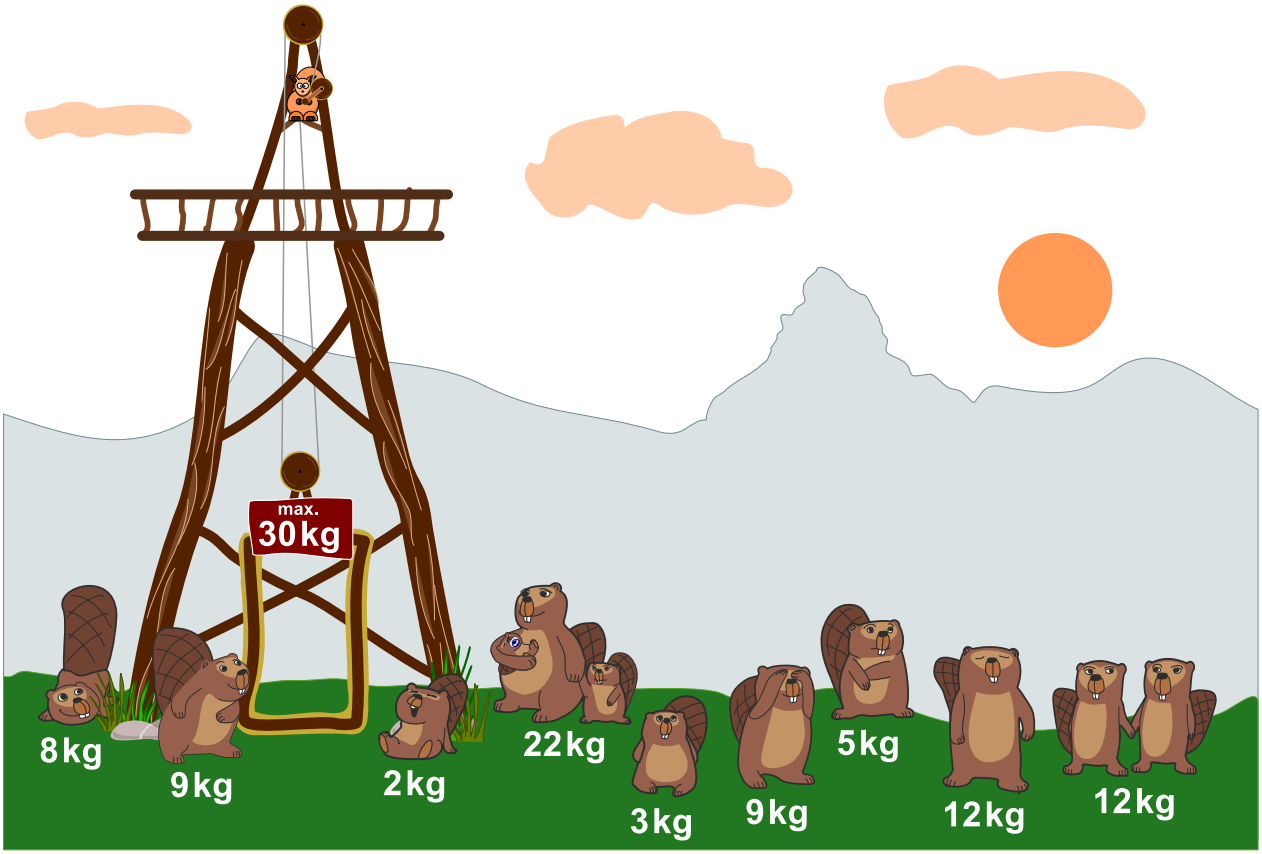
Lors de cet exercice, il a fallu trouver un chemin par le labyrinthe fléché en ne modifiant qu'une seule flèche. Comment trouve-t-on la solution d'un tel problème ? Comment un ordinateur pourrait-il procéder ? Un procédé appelé *retour sur trace* a été utilisé pour la démonstration du paragraphe précédent. Cela se fait grosso modo de la manière suivante : On suit une trace (depuis le départ ou depuis la fin) jusqu'à ce que l'on soit bloqué. On revient ensuite à l'étape précédente et on cherche une autre trace depuis là. De cette manière, on peut exclure les chemins impossibles dès le départ et être sûr de trouver la solution, étant donné que l'on essaie chaque possibilité.

Mots clés et sites web

Labyrinthe, retour sur trace
— `https://fr.wikipedia.org/wiki/Retour_sur_trace`

14. Excursion avec vue

Une famille de castors fait une excursion jusqu'à une tour panoramique. Ils sont en retard. L'ascenseur ne monte plus que deux fois avant la fermeture, et ne peut pas transporter plus de 30 kg d'un coup. Les jumeaux ne veulent monter qu'ensemble sur la tour. La maman castor porte le bébé dans ses bras et tient la main d'un petit castor. Et pourtant, on aimerait faire monter le plus grand nombre possible de castors au sommet de la tour.



Il faut se décider rapidement et seules les cinq options suivantes sont possibles. Qui doit rester en bas pour que le plus de castors possible puissent atteindre le sommet de la tour panoramique ?

A) Tout le monde peut monter.

B) La maman castor avec le bébé et le petit castor.

C) Les jumeaux et le castor de 5 kg.

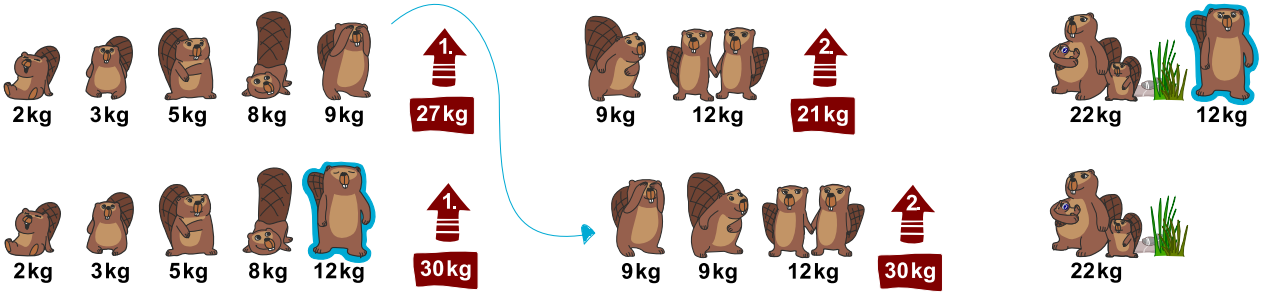
D) Les jumeaux et la maman castor avec le bébé et le petit castor.

E) La maman castor avec le bébé et le petit castor et le castor de 12 kg.

Solution

B) est l'une des bonnes solutions :

Une solution pourrait être de charger la cabine avec les castors les plus légers en premier : 2 kg + 3 kg + 5 kg + 8 kg + 9 kg = 27 kg lors de la première montée, et 9 kg + 12 kg = 21 kg lors de la seconde montée (cela fait 8 castors). Mais il y a encore de la place pour un castor supplémentaire dans l'ascenseur :



Cette stratégie permet d'utiliser les deux trajets en ascenseur de manière optimale : le castor de 9 kg est remplacé par celui de 12 kg lors du premier trajet. Le poids maximal de 30 kg est ainsi atteint. Le castor de 9 kg peut monter dans la cabine pour le second trajet, qui transporte également le poids maximal de 30 kg.
Les autres solutions proposées ne sont pas possibles : soit elles dépassent la charge maximale (le poids de la famille de castors entière dépasse 60 kg et même sans les jumeaux et le castor de 5 kg, elle atteint encore 65 kg), soit elles sont moins bonnes (si la maman avec les petits castors monte, ni les jumeaux, ni le castor de 9 kg, ni celui de 12 kg ne peuvent l'accompagner).

C'est de l'informatique !

Un des problèmes classiques en informatique est de trouver la combinaison optimale comme solution d'un problème. C'est souvent impossible de trouver un telle solution assez rapidement, ou dans un temps réaliste, parce qu'il y a trop de solutions possibles qui doivent être examinées. En informatique, on appelle cela un problème insoluble en pratique.
Ce problème est du même type que le *problème du sac à dos* lors duquel il faut ranger le plus d'objets possible dans un sac sans dépasser un poids maximal. Ce problème et les problèmes similaires sont appelés *NP-complets*. Ils ne peuvent être résolus que de manière approximative, ce qui veut dire que l'on peut trouver une bonne solution, mais pas nécessairement la solution optimale. On y parvient en réfléchissant à une stratégie judicieuse (« heuristique ») pour résoudre le problème.

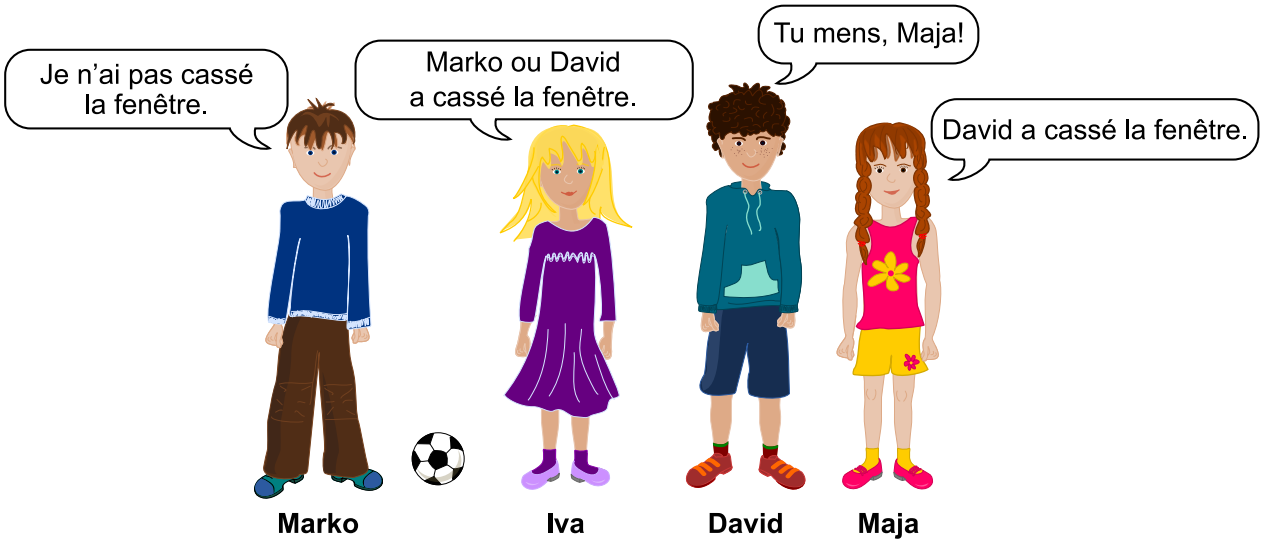
Mots clés et sites web

Optimisation, problème du sac à dos

- `https://fr.wikipedia.org/wiki/Optimisation_combinatoire`
- `https://fr.wikipedia.org/wiki/Problème_du_sac_à_dos`

15. Les mensonges ne mènent pas loin

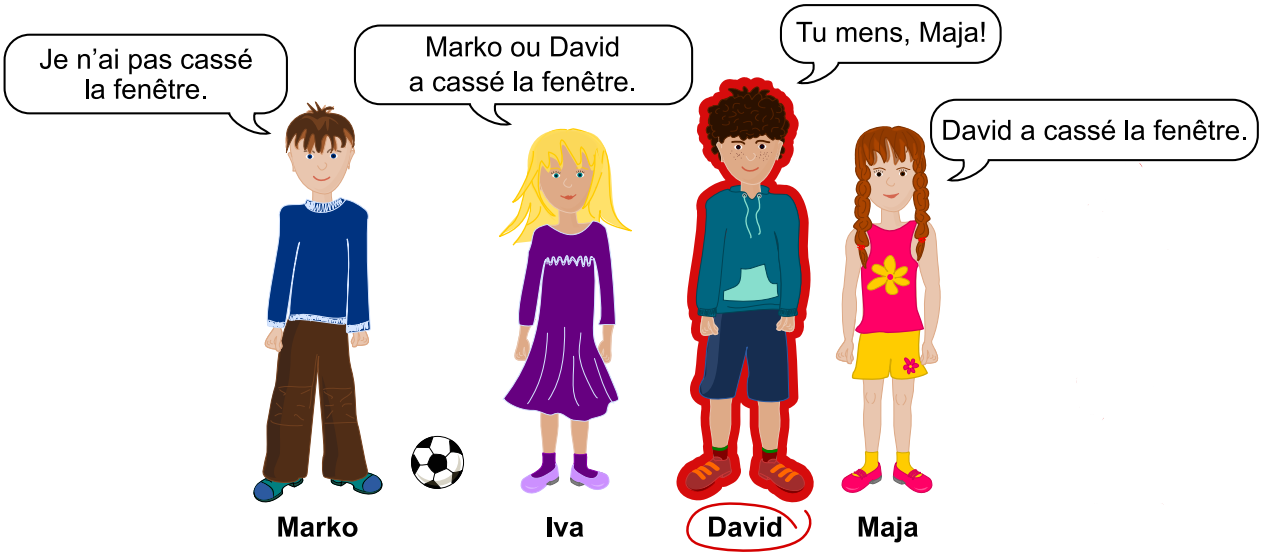
Par un jour de beau temps, Maja, David, Iva et Marko jouent au football près de la maison d'Anna. Tout à coup, une des fenêtres se casse et Anna cherche à savoir qui est responsable. Elle connaît les quatre enfants et sait que trois d'entre eux disent toujours la vérité ; elle ne sait pas ce qu'il en est du quatrième.
Les quatre enfants disent :



Clique sur l'enfant qui a cassé la fenêtre.

Solution

David a cassé la fenêtre.



Les déclarations de Maja et David ne peuvent pas être vraies toutes les deux ; l'un d'entre eux doit donc mentir. Si Maja disait la vérité, David mentirait et les déclarations d'Iva et de Marko seraient vraies. Par contre, si David disait la vérité, cela voudrait dire que Maja ment, mais aussi que soit Marko, soit Iva ment également, ce qui n'est pas possible étant donné que trois des enfants disent toujours la vérité.

C'est de l'informatique !

Tu dois raisonner de manière logique afin de résoudre ce problème. Le raisonnement se base sur la logique formulée en 1854 par George Boole (1815 – 1864), qui a décrit de manière formelle les bases d'énoncés logiques.
D'après lui, une déclaration (ou *assertion*) est soit *vraie* soit *fausse* (principe du tiers exclu). Plusieurs assertions peuvent être combinées à l'aide d'*opérateurs*. Des opérateurs logiques simples comme *ET* et *OU* lient deux assertions pour en former une nouvelle. Il existe aussi des opérateurs comme *NON* qui ne modifient qu'une seule assertion. La véracité de telles assertions combinées peut être déterminée à l'aide de *tables de vérité*.
L'un des opérateur qui représente la déduction « SI » → « ALORS » est l'implication. On parle alors de « tirer des conclusions logiques », c'est ce qui est nécessaire à la résolution de cet exercice.
Les ordinateurs fonctionnent également sur la base d'assertions booléennes et d'opérateurs logiques simples, car de tels ordinateurs peuvent être produits facilement en grandes quantités. Il existe quelques ordinateurs basés sur d'autres systèmes (par exemple les ordinateurs ternaires de la fin des années 50 en Union soviétique), mais soit ils sont restés au stade expérimental, soit ils n'ont jamais atteint une production de masse.

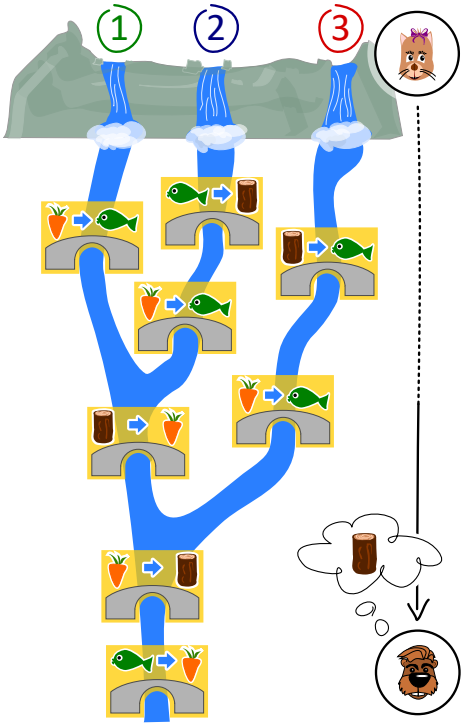
Mots clés et sites web

Logisches Schliessen

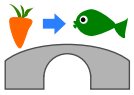
- `https://fr.wikipedia.org/wiki/Principe_du_tiers_exclu`
- `https://fr.wikipedia.org/wiki/George_Boole`
- `https://fr.wikipedia.org/wiki/Algèbre_de_Boole_(logique)`

— `https://fr.wikipedia.org/wiki/Implication_(logique)`
— `https://fr.wikipedia.org/wiki/Ordinateur_ternaire`
— `https://it.wikipedia.org/wiki/Calcolatore_ternario`

16. Chutes d'eau



Katja est au sommet d'une montagne. Cette montagne a trois chutes d'eau qui se rejoignent dans une rivière en bas de la vallée.
Katja peut lancer un poisson ou une carotte dans l'une des trois chutes d'eau. Les cours d'eau sont enjambés par plusieurs ponts sous lesquels vivent des trolls. Les trolls remplacent les objets passant sous les ponts par d'autres objets.



Par exemple, si une carotte passe sous un pont comme celui ci-dessus, les trolls la remplacent par un poisson.
Justus est au bord de la rivière en bas de la vallée.
Justus a besoin de bois. Quel objet Katja doit-elle lancer dans quelle chute d'eau afin que Justus reçoive du bois ?

A) Elle lance un poisson dans la chute d'eau numéro 1.

B) Elle lance un poisson dans la chute d'eau numéro 2.

C) Elle lance une carotte dans la chute d'eau numéro 2.

D) Elle lance une carotte dans la chute d'eau numéro 3.

Solution

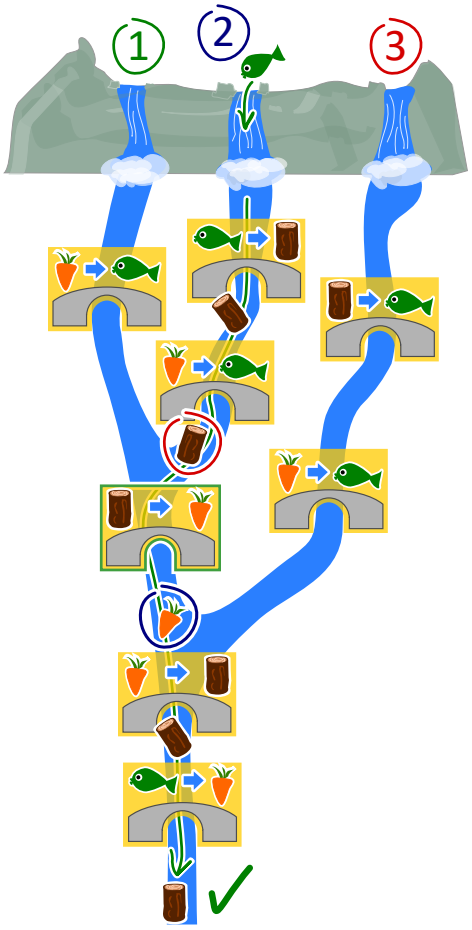
La bonne réponse est B). Elle lance un poisson dans la chute d'eau numéro 2.
Voilà ce qu'il se passe lors des différentes solutions proposées :

A) Un poisson lancé dans la chute d'eau numéro 1 ne sera remplacé que sous le dernier pont. Justus reçoit donc une carotte.

B) Un poisson lancé dans la chute d'eau numéro 2 est remplacé par du bois, qui est remplacé par une carotte, elle-même ensuite remplacée par du bois. Justus reçoit donc du bois. C'est la bonne solution.

C) Une carotte lancée dans la chute d'eau numéro 2 est remplacée par un poisson, qui est ensuite remplacé par une carotte. Justus reçoit donc une carotte.

D) Une carotte lancée dans la chute d'eau numéro 3 est remplacée par un poisson, qui est ensuite remplacé par une carotte. Justus reçoit donc une carotte.



Une autre manière de résoudre cet exercice consiste à commencer par la fin : Pour obtenir du bois en bas de la montagne, l'objet flottant doit être une carotte quand il passe sous l'avant-dernier pont. La seule possibilité d'avoir une carotte à cette étape est que du bois passe sous le pont commun des chutes numéro 1 et 2 (et pas 3) . La seule possibilité d'avoir du bois à cet endroit est de lancer un poisson dans la chute d'eau numéro 2.

C'est de l'informatique !

On peut se représenter un ordinateur comme une machine qui lit des données, les traite, puis écrit des données sortantes. Mais comment l'ordinateur « sait »-il comment traiter les données ? Il a reçu des ordres concernant les tâches à accomplir. Ceci se fait en écrivant des programmes.
Il existe beaucoup de langages de programmation qui fonctionnent selon différents paradigmes. La programmation fonctionnelle est un tel paradigme. Ce style de programmation est lui-même comme un petit ordinateur : il est composé de beaucoup de fonctions (ou routines) qui traitent des données et retournent des valeurs sortantes. Les ponts de cet exercice sont comme de petites fonctions, et le système complet comme un programme écrit dans un langage de programmation fonctionnel.

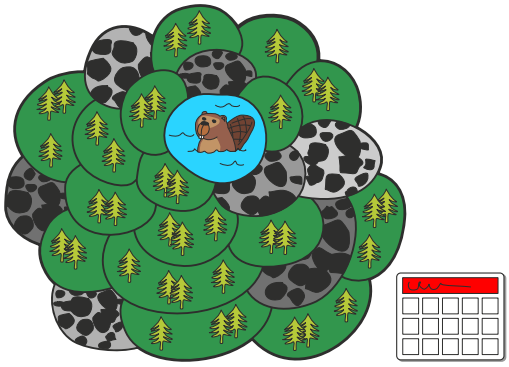
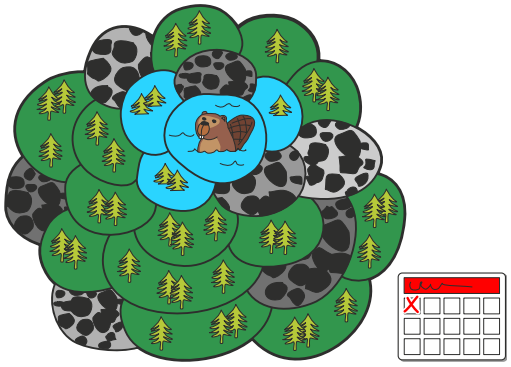
Mots clés et sites web

Paradigme de programmation, programmation fonctionnelle, fonctions et paramètres

- `https://fr.wikipedia.org/wiki/Test_(informatique)`
- `https://fr.wikipedia.org/wiki/Boîte_blanche`
- `https://fr.wikipedia.org/wiki/Test_de_la_boîte_noire`
- `https://fr.wikipedia.org/wiki/Paradigme_(programmation)`
- `https://fr.wikipedia.org/wiki/Programmation_fonctionnelle`
- `https://fr.wikipedia.org/wiki/Routine_(informatique)`

17. L'étang des castors

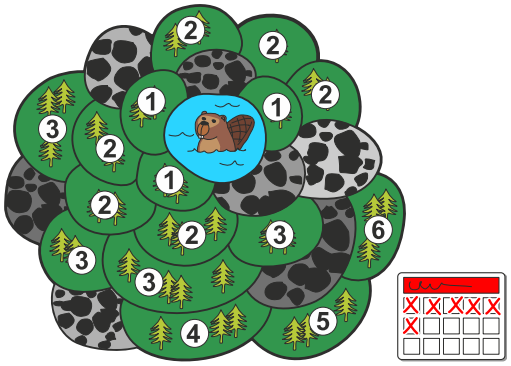
Il y a un petit étang dans une vallée. Il est entouré de parcelles de terrain forestier ou rocailleux. Plusieurs castors vivent dans l'étang.
Il vient un jour où les castors trouvent l'étang trop petit et décident d'inonder des parcelles de forêt. Chaque jour, ils inondent toutes les parcelles de forêt partageant une bordure avec une parcelle déjà inondée. Trois parcelles de forêt sont inondées le premier jour.

L'étang dans la vallée	Après un jour
	

Après combien de jours en tout (y compris le premier jour représenté plus haut) les parcelles forestières sont-elles toutes inondées ?

Solution

Toutes les parcelles de forêt sont inondées en six jours.
L'image ci-dessous montre au combientième jour chaque parcelle a été inondée. Les parcelles voisines de l'étang sont inondées après le premier jour et donc marquées du chiffre 1. Les parcelles voisines de ces dernières sont marquées du chiffre 2 ; elles sont inondées après le deuxième jour, et ainsi de suite. La dernière parcelle est marquée du chiffre six et est inondée le sixième jour – toutes les parcelles forestières sont donc inondées à ce moment-là.



C'est de l'informatique !

Dans cet exercice, les castors inondent un espace forestier connexe qui est composé, en plus de l'étang, de parcelles séparées. L'espace est connexe car l'on peut atteindre chaque parcelle de forêt en passant par d'autres parcelles sans sortir de l'espace forestier.
Il existe également en dehors de la vallée de l'étang des castors des espaces connexes devant être inondés. Une zone de couleur unie sur une image n'est finalement rien d'autre qu'un espace connexe de pixels de la même couleur. Un groupe d'adolescents, dans lequel chacun est relié à chacun par des liens d'amitiés directs ou par l'ami d'un ami d'un ami, est également un « espace connexe », si l'on considère un lien d'amitié entre deux personnes comme du voisinage.
En informatique, il y a des méthodes permettant de découvrir et d'investiguer les espaces connexes, comme les algorithmes de parcours en largeur ou en profondeur. Ces méthodes permettent par exemple de changer la couleur d'une zone sur une image ou de découvrir des groupements sur les réseaux sociaux.

Mots clés et sites web

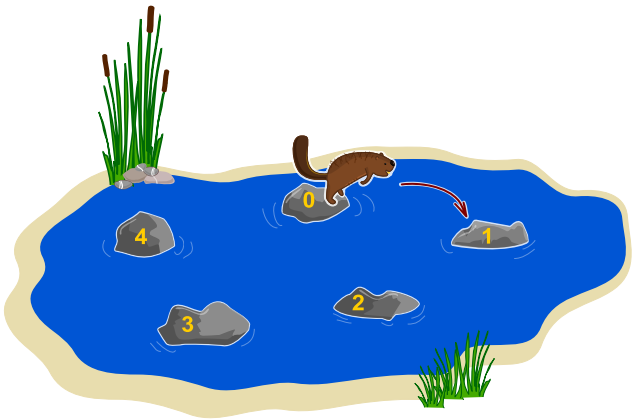
Algorithme à front d'onde, algorithme de parcours en largeur

- `https://fr.wikipedia.org/wiki/Graphe_connexe`
- `https://fr.wikipedia.org/wiki/Algorithme_de_parcours_en_largeur`

18. Compétition des castors

Plusieurs castors suivent un entraînement intensif en préparation à la compétition annuelle des castors. L'entraînement du jour consiste en un parcours de saut de pierre en pierre, dans le sens des aiguilles d'une montre, comme indiqué par la flèche. Si le castor saute 8 fois, il termine son parcours sur la pierre numéro 3 :

$$0 \to 1 \to 2 \to 3 \to 4 \to 0 \to 1 \to 2 \to 3.$$



Le castor le plus fort a sauté 129 *fois aujourd'hui. Sur quelle pierre se trouvait-il lorsqu'il s'est arrêté ?*

Solution

5 sauts amènent le castor sur la pierre de laquelle il est parti. Ces 5 sauts constituent un tour. Pour déterminer où le castor arrive après 129 sauts, nous devons déterminer combien de tours il fait, ainsi que combien de sauts il fait après le dernier tour complet. Ce sont dans ce cas-là $129 = 25 \cdot 5 + 4$ sauts (25 tours et 4 sauts), ce qui veut dire que le castor termine son parcours sur la même pierre que s'il n'avait sauté que 4 fois, donc sur la pierre numéro 4.

C'est de l'informatique !

Tu as peut-être déjà rencontré une opération semblable en cours de mathématique ; il s'agit d'une *division avec reste*, appelée aussi division euclidienne ou division posée.
Dans cet exercice, il s'agit d'effectuer la division euclidienne de $129 : 5 = 25$ reste 4. En informatique, le calcul du reste est fréquemment utilisé et a donc un nom spécifique : le modulo. Les signes « % » ou « mod » sont habituellement utilisés comme opérateurs, on peut donc écrire $129\%5 = 4$.
Cet opérateur est par exemple utilisé dans des boucles (comme le castor qui fait des tours en sautant) lorsque que des variables débordent et dans un procédé de cryptographie très répandu appelé « chiffrement RSA ».

Mots clés et sites web

Modulo-Operation

- `https://fr.wikipedia.org/wiki/Modulo_(opération)`
- `https://fr.wikipedia.org/wiki/Division_posée`
- `https://fr.wikipedia.org/wiki/Division_euclidienne`
- `https://fr.wikipedia.org/wiki/Chiffrement_RSA`

19. Maison numéro 29

Milo fait un stage dans un lotissement de maisons de vacances. Aujourd'hui, il doit fixer des plaques numérotées aux maisons de vacances. Certaines maisons sont déjà numérotées. Il commence par la maison numéro 50. Depuis là, il doit :

— aller à gauche si le nouveau numéro est plus petit que celui de la maison devant laquelle il se trouve,
— aller à droite si le nouveau numéro est plus grand que celui de la maison devant laquelle il se trouve,
— fixer la plaque numérotée à la maison devant laquelle il se trouve si celle-ci n'est pas encore numérotée.

A quelle maison de vacances Milo doit-il fixer le numéro 29 *?*

Solution

La maison de vacances correcte est la troisième depuis la gauche :

Le nouveau numéro 29 est plus petit que le numéro de la maison 50, donc Milo doit commencer par aller à gauche. Il arrive à la maison 24, dont le numéro est plus petit que 29, et doit donc aller à droite. Le numéro 29 est plus petit que celui de la maison 34 devant laquelle il arrive, donc il va à gauche. La maison suivante n'a pas encore de numéro, donc il y fixe la plaque numéro 29.

C'est de l'informatique !

La numérotation des maisons de vacances correspond à un *arbre binaire de recherche*, une structure de données souvent utilisée en informatique. Un arbre binaire de recherche permet de retrouver rapidement des données enregistrées.
Un arbre binaire de recherche est construit de telle façon qu'un « *élément* » est enregistré à chaque intersection (« *nœuds* »). Après chaque intersection, il y a au maximum deux chemins (« *arêtes* ») menant à d'autres intersections. Lors de l'enregistrement de nouvelles données, le chemin de gauche (par exemple) est toujours choisi lorsque le nouvel élément a une valeur plus petite que l'élément situé à l'intersection, et sinon le chemin de droite est suivi. Le nouvel élément est enregistré à la première intersection de libre.
Lorsque l'on cherche un élément précis, on peut ainsi facilement savoir quel chemin suivre à chaque intersection. Si l'arbre binaire de recherche est « *équilibré* » (appelé alors un *arbre AVL*), chaque étape de la recherche réduit de moitié le nombre d'intersections devant encore être au maximum parcourues avant de trouver l'élément recherché. Cela veut dire qu'un arbre de 1000 éléments peut être exploré en seulement 10 étapes, un arbre de 1'000'000 éléments en 20 étapes et un arbre de 1'000'000'000 éléments en 30 étapes (donc, pour n éléments, $\log_2(n)$ étapes).

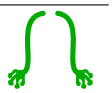
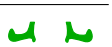
Mots clés et sites web

Arbre binaire de recherche, arbre AVL

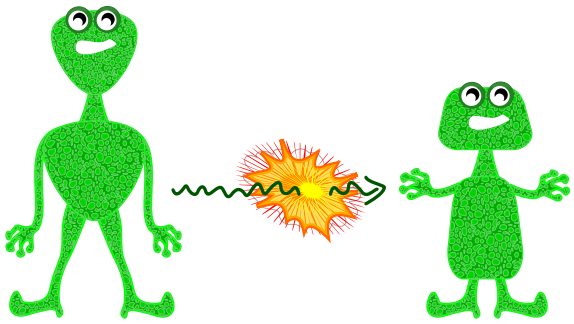
- `https://fr.wikipedia.org/wiki/Arbre_binaire_de_recherche`
- `https://fr.wikipedia.org/wiki/Arbre_AVL`

20. Un extraterrestre !

Un extraterrestre possède une tête, un tronc, deux bras et deux jambes. Cet extraterrestre peut être modifié par les instructions suivantes ; chaque partie du corps peut être modifiée plusieurs fois.

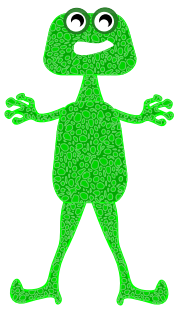
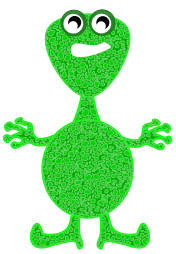
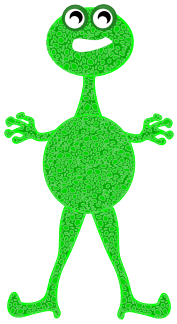
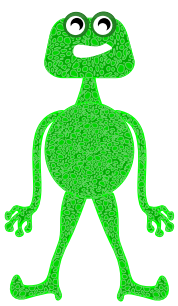
$Te(e)$	La tête devient elliptique.	
$Te(4)$	La tête devient rectangulaire.	
$Te(3)$	La tête devient triangulaire.	
$Tr(e)$	Le tronc devient elliptique.	
$Tr(4)$	Le tronc est rectangulaire.	
$Tr(3)$	Le tronc devient triangulaire.	
$B(+)$	Les bras deviennent longs.	
$B(-)$	Les bras deviennent courts.	
$J(+)$	Les jambes deviennent longues.	
$J(-)$	Les jambes deviennent courtes.	

Les instructions sont effectuées les unes après les autres de gauche à droite. Par exemple, les instructions $Te(e)$, $Tr(4)$, $Te(4)$, $B(-)$, $J(-)$ résultent en l'extraterrestre suivant :

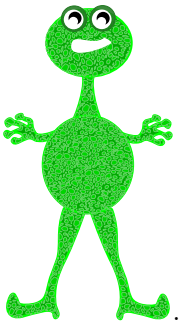


Quelle est l'apparence de l'extraterrestre après les instructions suivantes ?

$$Te(3),\ J(+),\ Tr(3),\ B(+),\ Te(e),\ B(-),\ Tr(e)$$

A)	B)	C)	D)
			

Solution

La bonne réponse est C)  .
La forme de chaque partie du corps est déterminée par la dernière instruction la concernant. Les instructions précédentes n'influencent pas l'apparence finale de l'extraterrestre, étant donné que l'instruction la plus récente remplace les instructions précédentes.
Le résultat des instructions ci-dessus est donc un extraterrestre avec un tronc elliptique ($Tr(e)$), des bras courts ($B(-)$), une tête elliptique ($Te(e)$) et de longues jambes ($J(+)$). Les autres extraterrestres proposés ont au moins deux caractéristiques différentes, et ne correspondent donc manifestement pas à la description.

C'est de l'informatique !

En informatique, lors de l'exécution d'un programme, les instructions sont traitées les unes après les autres. Les instructions actuelles peuvent donc remplacer ou modifier (« écraser ») les instructions les précédant.
Cela arrive souvent lorsque la valeur de variables doit être modifiée pendant l'exécution d'un programme. On peut imaginer que la forme de chaque partie du corps est enregistrée dans une variable. L'instruction « $Te(e)$ » enregistre alors la valeur « e » dans la variable nommée « Te ».
La notation « $Te(e)$ » est fonctionnelle. On appelle la fonction « $Te()$ » tout en lui donnant l'argument « e ». Cette notation est souvent utilisée, car la fonction « $Te()$ » peut aussi vérifier si l'argument, en l'occurence « e », est valable. Si cela n'est pas nécessaire, où alors si la variable « Te » n'est utilisée que localement, on peut aussi directement écraser sa valeur en utilisant l'opérateur d'affectation « $=$ ». On écrirait dans ce cas « $Te = e$ ».

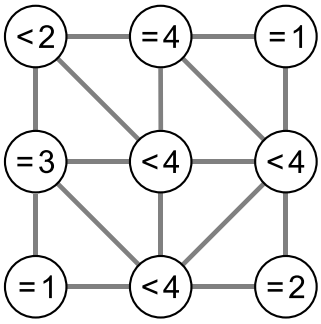
Mots clés et sites web

Variable, séquence

- `https://fr.wikipedia.org/wiki/Variable_(informatique)`
- `https://fr.wikipedia.org/wiki/Programmation_structurée`

21. Voisins

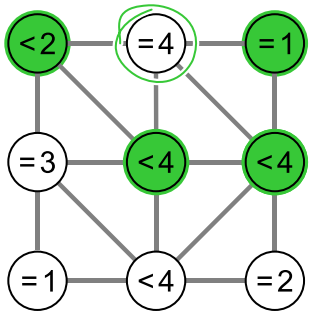
L'image ci-dessous montre neuf cercles partiellement connectés les uns aux autres. Une connexion entre deux cercles en fait des voisins. Les cercles peuvent être sélectionnés par un clic ; ils sont alors colorés en vert, alors que les cercles non-sélectionnés sont blancs.
Dans chaque cercle, une expression indique combien de cercles doivent être sélectionnés parmi les cercles voisins. Par exemple, le cercle portant l'expression « $= 3$ » doit avoir trois de ses quatre voisins sélectionnés, et les cercles portant l'expression « < 4 » peuvent en avoir au maximum trois qui sont sélectionnés.



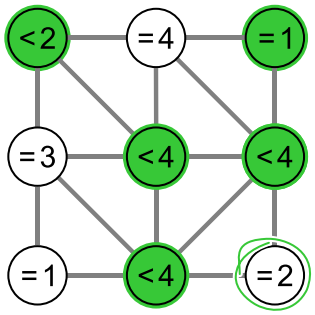
Sélectionne les cercles de manière à ce que toutes les conditions soient remplies.

Solution

Le cercle central de la rangée du haut porte l'expression « = 4 » et a quatre voisins, ce qui fait que chacun de ses quatre voisins doit être sélectionné :



Similairement, le cercle en bas à droite porte l'expression « = 2 » et possède deux voisins, qui doivent donc être les deux sélectionnés (le cercle du centre à droite est déjà sélectionné) :



Toutes les conditions sont ainsi remplies.
Si l'on sélectionnait un cercle supplémentaire, certaines conditions seraient violées :

- Si le cercle « = 4 » au centre de la rangée du haut était sélectionné, la condition « = 1 » du cercle en haut à droite ne serait plus remplie.
- Si le cercle « = 3 » à gauche de la rangée centrale était sélectionné, la condition « < 2 » du cercle en haut à gauche ne serait plus remplie.
- Si le cercle « = 1 » en bas à gauche était sélectionné, la condition « = 3 » du cercle à gauche de la rangée centrale ne serait plus remplie.
- Si le cercle « = 2 » en bas à droite était sélectionné, la condition « < 4 » du cercle à droite de la rangée centrale ne serait plus remplie.

C'est de l'informatique !

De combien d'essais a-t-on besoin pour résoudre le problème ? Si l'on essaie simplement chaque solution possible, on a deux réglages différents pour chacun des neuf cercles, ce qui fait $2^9 = 512$ possibilités différentes. On appelle cette méthode *recherche exhaustive* ou *recherche par force brute*. Pour chacune de ces 512 possibilités, il faudrait vérifier si toutes les conditions sont remplies.
Dans ce cas-là, il est plus judicieux de procéder de manière logique et cohérente. On cherche d'abord les cercles ayant des conditions ne pouvant être remplies que de manière unique. Il s'agit par exemple des cercles ayant exactement n connexions, donc n voisins, et la condition « = n ». On peut continuer à partir de là de manière logique en regardant s'il reste des conditions que l'on ne peut remplir que d'une seule façon. On peut ainsi trouver la bonne solution à moindre effort. De manière générale, on peut trouver de cette manière qu'il n'existe pas de solution, ou trouver au moins une solution parmi

plusieurs. On parle de démarche analytique lorsque seule une solution prometteuse parmi toutes les possibilités est considérée « *heuristique* ».

Mots clés et sites web

Voisinage dans la théorie de graphes, approche logique

- `https://fr.wikipedia.org/wiki/Voisinage_(théorie_des_graphes)`
- `https://fr.wikipedia.org/wiki/Heuristique_(mathématiques)`
- `https://fr.wikipedia.org/wiki/Recherche_exhaustive`

22. Jeu vidéo

Andrea a programmé un jeu vidéo à l'école. Les règles sont simples :
Le jeu se joue en plusieurs tours. Une feuille tombe lors de chaque tour. Le castor essaie d'attraper la feuille avant qu'elle ne touche le sol. Le castor gagne s'il attrape 15 feuilles avant que 4 feuilles ne touchent le sol.
La durée du jeu est égale au nombre de tours (et donc au nombre de feuilles tombées en tout).
Dans l'exemple suivant, le castor perd après 6 tours, car il a atteint le maximum de 4 feuilles touchant le sol. La durée du jeu dans cet exemple est de 6 tours.

Tour	Résultat	Score – nombre total de feuilles	
		Attrapées	Pas attrapées
1	Attrapée	1	0
2	Pas attrapée	1	1
3	Attrapée	2	1
4	Pas attrapée	2	2
5	Pas attrapée	2	3
6	Pas attrapée	2	4

Quelle est la durée maximale d'un jeu ?

A) 4 tours

B) 15 tours

C) 18 tours

D) 19 tours

E) 20 tours

F) La durée du jeu est illimitée.

Solution

Afin de déterminer la durée maximale d'un jeu, nous devons combiner toutes les situations lors desquelles le jeu continue. Pour ceci, nous combinons le nombre maximal de feuilles attrapées avant la fin du jeu (14 tours) avec le nombre maximal de feuilles touchant le sol avant la fin du jeu (3 tours). Au tour suivant, on peut soit attraper une 15^e feuille, soit en laisser tomber une 4^e. La durée maximale est donc $15 + 3 = 14 + 4 = 18$ tours et la bonne réponse est C).
La réponse A) 4 tours est la durée minimale du jeu si aucune feuille n'est attrapée.
La réponse B) 15 tours est la durée minimale du jeu si toutes les feuilles sont attrapées.
Les réponses D), E) et F) sont fausses, car le nombre maximal de feuilles attrapées ou pas attrapées est atteint avant.

C'est de l'informatique !

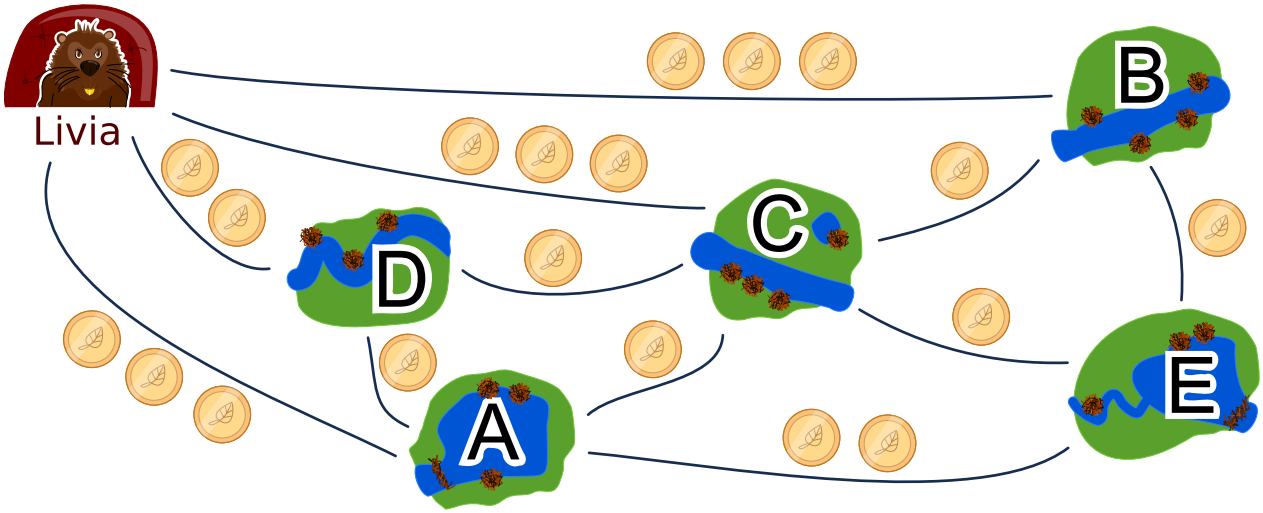
Lors de la programmation d'un jeu, les règles doivent être clairement définies. Les conséquences des règles doivent être bien comprises afin que le jeu permette de gagner et de perdre (le nombre de feuilles doit être suffisant) et que le jeu ne dure ni trop longtemps, ni pas assez.
Un jeu consistant en plusieurs tours est un processus, c'est-à-dire une suite d'opérations ordonnées. Les informaticiens sont des spécialistes de la modélisation et description de processus. Une des tâches principales est de déterminer tout ce qui peut se passer lors du déroulement d'un processus et combien de temps celui-ci peut durer.

Mots clés et sites web

Analyse, vérification et validation de logiciel
- `https://en.wikipedia.org/wiki/Software_verification`
- `https://en.wikipedia.org/wiki/Verification_and_validation`

23. Tournée des castors

Livia aimerait rendre visite à chacun de ses amis dans les villages A, B, C, D et E en transports publics. Elle fait la tournée de tous ses amis lors d'un seul voyage, sans passer deux fois par le même village. Elle rentre chez elle à la fin de sa tournée de visites. Le prix de transport de chaque ligne est affiché ci-dessous.



Une des routes possible pour voir ses amis est :

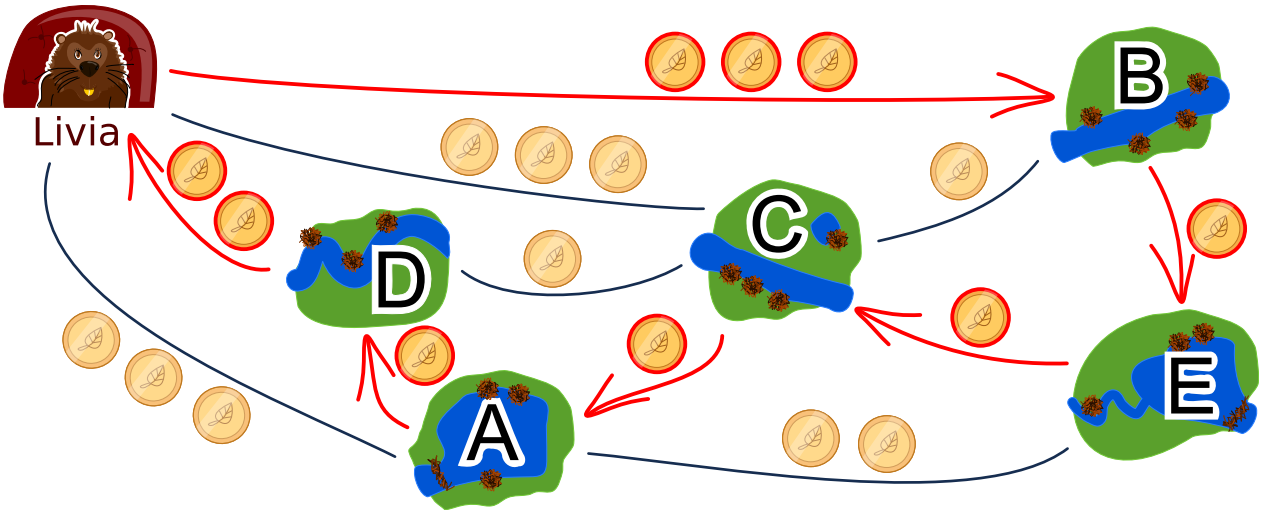
$$\text{départ} \to \text{B} \to \text{E} \to \text{A} \to \text{D} \to \text{C} \to \text{départ}.$$

Cette route coûte $3 + 1 + 2 + 1 + 1 + 3 = 11$ francs castor.
Dans quelle ordre Livia doit-elle rendre visite à ses amis ?

Solution

Il existe deux solutions optimales :

— départ → B → E → C → A → D → départ
— départ → D → A → C → E → B → départ



Les deux solutions sont semblables, mis à part le sens de trajet, et coûtent 9 francs castor. Il n'y a pas de meilleure solution. Depuis chez elle, Livia peut prendre une fois le chemin à 2 francs castor puis un chemin à 3 francs castor dans l'autre sens. Les quatre autres nœuds à visiter correspondent à quatre autres chemins coûtant au moins un franc castor chacun, ce qui fait déjà 9 francs castor au total.

Toutes les autres solutions sont plus chères :

— Coût de 10 francs castor : départ → A → D → C → E → B → départ
— Coût de 10 francs castor : départ → A → E → B → C → D → départ
— Coût de 10 francs castor : départ → B → C → E → A → D → départ
— Coût de 10 francs castor : départ → B → E → A → C → D → départ
— Coût de 10 francs castor : départ → B → E → C → D → A → départ
— Coût de 10 francs castor : départ → C → B → E → A → D → départ
— Coût de 10 francs castor : départ → D → A → E → B → C → départ
— Coût de 10 francs castor : départ → D → A → E → C → B → départ
— Coût de 10 francs castor : départ → D → C → A → E → B → départ
— Coût de 10 francs castor : départ → D → C → B → E → A → départ
— Coût de 11 francs castor : départ → B → E → A → D → C → départ
— Coût de 11 francs castor : départ → C → D → A → E → B → départ

Une des méthodes pour trouver le parcours le moins cher consiste à emprunter le chemin le moins cher, puis à chercher un solution à partir de là.

C'est de l'informatique !

La recherche de bonnes solutions, voire de solutions optimales, est l'un des problèmes fondamentaux de l'informatique. La description de notre problème d'optimisation peut être visualisée dans un diagramme ayant les amis comme nœuds et les chemins comme arêtes. La tâche consiste à passer par chacun des nœuds tout en minimisant la somme des poids des arêtes (le coût en francs castor). C'est un exercice semblable au célèbre problème du voyageur de commerce (Travelling Salesman Problem, TSP).

Ce type de problème est habituellement très difficile à résoudre de manière computationnelle. Afin d'éviter de devoir essayer chaque solution possible, on peut utiliser une bonne heuristique (un exemple d'heuristique est de commencer par le chemin le plus court) puis éliminer toutes les solutions qui sont moins bonnes. Dans ce cas, nous ne permettons qu'un passage par nœud. Si plusieurs passages par nœud étaient permis, le problème deviendrait plus complexe car il faudrait prendre beaucoup plus de possibilités en considération.

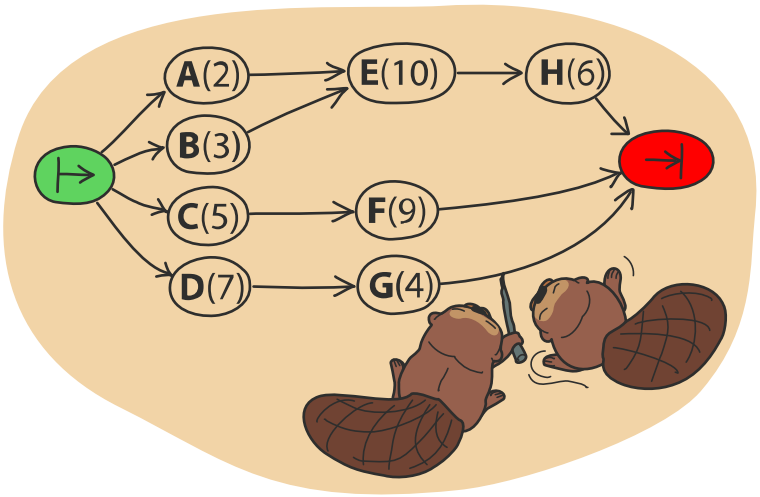
Mots clés et sites web

Optimisation, problème du voyageur de commerce

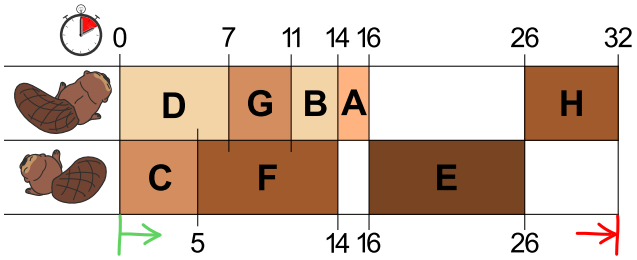
- `https://fr.wikipedia.org/wiki/Problème_du_voyageur_de_commerce`
- `https://en.wikipedia.org/wiki/Optimization_problem`
- `https://fr.wikipedia.org/wiki/Optimisation_(mathématiques)`

24. Deux castors au travail

Deux castors construisent un barrage et doivent pour cela réaliser huit tâches : abattre des arbres, enlever les branches des troncs, amener les troncs dans l'eau, et ainsi de suite. Chaque tâche est définie par une lettre (son nom) et un chiffre entre parenthèses qui donne le nombre d'heures de travail nécessaire à la réalisation de la tâche.
Certaines tâches ne peuvent être commencées que lorsque certaines autres sont terminées. Ce déroulement est représenté par des flèches dans le schéma ci-dessous. Les deux castors peuvent travailler en même temps à différentes tâches, mais ils ne peuvent pas travailler ensemble à la même tâche.



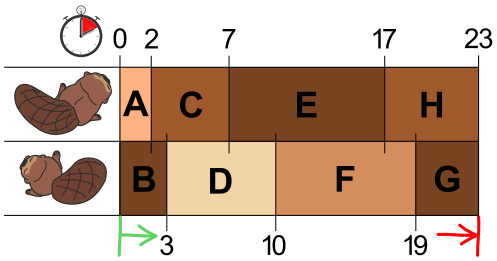
L'image ci-dessous montre un plan de travail possible pour les deux castors qui prévoit 32 heures de travail en tout, mais c'est possible de réaliser le barrage plus rapidement !



De combien de temps les castors ont-ils au minimum besoin pour construire le barrage ?

Solution

Au moins 23 heures sont nécessaires à la construction.
L'image dans la donnée montre un plan de travail possible. Dans celui-ci, le premier castor a une longue pause de 10 heures et le deuxième deux pauses de huit heures en tout. S'ils travaillaient sans pause, ils finiraient plus rapidement.



Si l'on veille à ce que les deux longues tâches E(10) et F(9) ne soient pas réalisées par le même castor, on trouve facilement un plan de travail qui prévoit 23 heures en tout. Ce n'est pas possible de construire le barrage plus rapidement, car les deux castors travaillent déjà sans interruption.

C'est de l'informatique !

Une possibilité pour trouver un des plans de travail les plus courts serait de suivre la règle suivante : « Choisis parmi les tâches à réaliser celle qui nécessite le plus d'heures de travail ». En informatique, on parle alors de stratégie *gloutonne*. On commence par réaliser les tâches qui nous font progresser le plus vers la solution finale du problème.
Les stratégies gloutonnes fonctionnent bien dans beaucoup de cas, mais parfois – comme dans cet exercice – elles ne sont pas adaptées. Cet exercice a été développé de manière à ce que la stratégie gloutonne ne fonctionne pas. Le fait de trouver de tels problèmes peu pratiques à résoudre est une tâche importante : en informatique théorique, par exemple, on analyse de manière ciblée la pire des situations (« worst case ») pour un programme afin de pouvoir mieux estimer le temps d'exécution d'un algorithme.
Il n'existe qu'une manière sûre de trouver la meilleure solution à ce problème, c'est d'essayer tous les plans de travail possibles qui respectent les règles données. Mais lors de grands projets, le nombre de possibilités peut être tellement grand que l'élaboration d'un plan de travail durerait trop longtemps. C'est dans ces cas-là qu'une stratégie gloutonne entre en jeu, car elle permet de trouver relativement rapidement une solution suffisamment bonne, même si ce n'est pas la solution optimale.

Mots clés et sites web

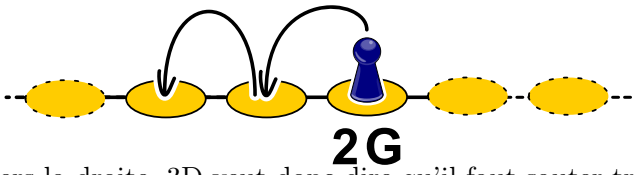
Ordonnancement, algorithme glouton

- `https://fr.wikipedia.org/wiki/Ordonnancement_de_travaux_informatiques`
- `https://fr.wikipedia.org/wiki/Tri_topologique`
- `https://fr.wikipedia.org/wiki/Algorithme_glouton`

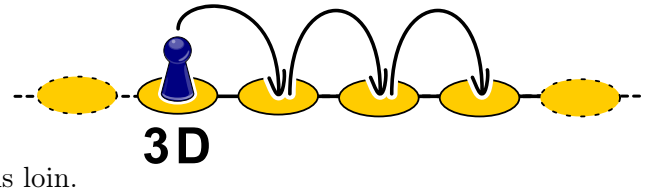
25. Marelle

Comme pour tout jeu de marelle, il s'agit ici de sauter de case en case en suivant certaines règles. Dans ce jeu-ci, une règle est associée à chaque case. Il y a trois sortes de règles :

— nG : sauter n cases vers la gauche, 2G veut donc dire qu'il faut sauter deux fois vers la gauche.

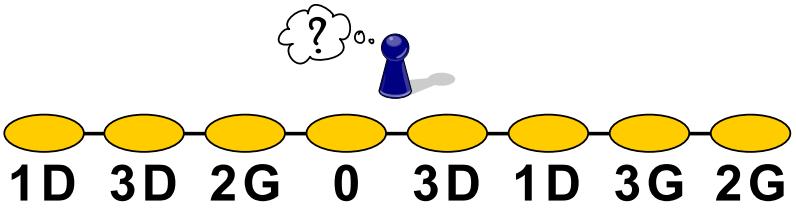


— nD : sauter n cases vers la droite, 3D veut donc dire qu'il faut sauter trois fois vers la droite.



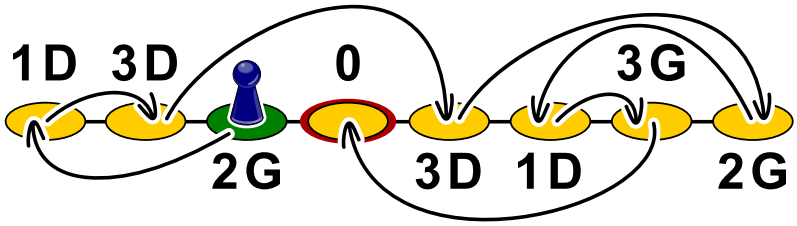
— 0 : ne pas sauter plus loin.

De quelle case doit-on partir afin d'être passé une fois par chaque case durant le jeu ?



Solution

En partant de la troisième case depuis la gauche (« 2G »), on passe par chaque case avant d'avoir terminé le jeu.
On trouve la solution en cherchant la case depuis laquelle on peut atteindre la case « 0 ». Dans notre cas, il s'agit de la deuxième case depuis la droite (« 3G »). Celle-ci peut être atteinte depuis la troisième case depuis la droite (« 1D »), qui peut être atteinte depuis la case tout à droite (« 2G »). Celle-ci peut être atteinte depuis la quatrième case depuis la droite (« 3D »), après être passé par la deuxième case depuis la gauche (« 3D »), qui peut pour sa part être atteinte depuis la case tout à gauche (« 1D »), en partant de la dernière case par laquelle l'on n'est pas encore passé, soit la troisième depuis la gauche (« 2G »).



Le traçage du chemin par des flèches fait des cases un graphe orienté qu'il suffit de parcourir à l'envers depuis la case « 0 » pour arriver sur la case de départ, la troisième depuis la gauche.

C'est de l'informatique !

En informatique, la structure de données appelée « *liste chaînée* » fonctionne similairement aux cases de la marelle : dans la mémoire vive, les objets sont enregistrés dans des cellules contenant l'adresse de la cellule contenant l'objet suivant. Le système de gestion de la mémoire peut ainsi enregistrer un objet à n'importe quel endroit de la mémoire vive et n'a pas besoin de temps pour établir un espace de stockage connexe pour tous les objets. De son côté, le programmeur ne doit s'occuper de rien à part de réserver un espace de stockage de taille suffisante.
Mais que se passe-t-il lorsque des objets enregistrés ne sont plus utiles ? Alors qu'auparavant le programmeur devait s'occuper de libérer de l'espace de stockage (ce qui, malheureusement, créait souvent des problèmes, les programmes gaspillant de l'espace de stockage avant de se bloquer à cause d'un manque de mémoire), les langages de programmation modernes ont pour cela une sorte de service de ramassage des ordures, le « *ramasse-miettes* », qui vérifie régulièrement si les objets sont encore référencés (donc si d'autres objets possèdent leur adresse). Parfois, de grosses structures ne sont plus référencées, et il faut donc suivre les références jusqu'à l'adresse de départ comme dans l'exemple.

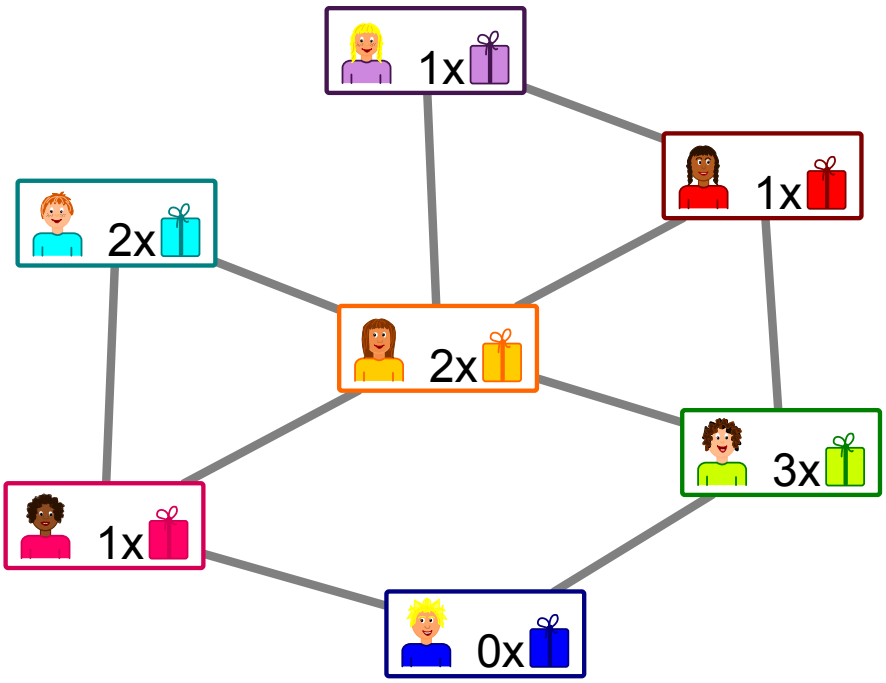
Mots clés et sites web

File, gestion de la mémoire, GOTO

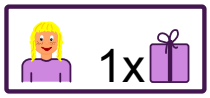
- `https://fr.wikipedia.org/wiki/Ramasse-miettes_(informatique)`
- `https://en.wikipedia.org/wiki/St-connectivity`

26. Cadeaux

L'image suivante montre les liens d'amitiés entre les enfants habitant le même immeuble. Un trait reliant deux enfants signifie qu'ils sont amis.



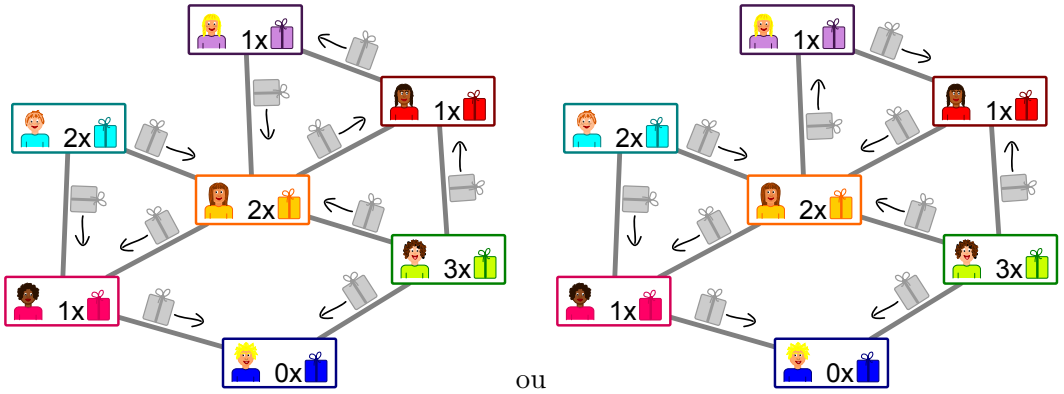
Les habitants de l'immeuble organisent une fête avec des cadeaux pour les enfants. L'un des enfants de chaque paire d'amis doit offrir un cadeau à l'autre.

L'image montre le nombre de cadeaux que chaque enfant peut offrir : 

veut dire que la fille en violet peut offrir un cadeau.
Tu n'as pas encore décidé qui offre le cadeau à qui pour chaque paire d'amis.

Solution

Il y a deux possibilités d'organiser la distribution de cadeaux sans qu'un enfant ne doive en offrir plus qu'il ne peut :



On commence par l'enfant tout en bas : il ne peut offrir aucun cadeau et va donc recevoir un cadeau de chacune de ses amies à gauche et à droite. Son amie à gauche ne pouvant offrir qu'un seul cadeau, elle va en recevoir de ses deux autres amis au-dessus et au centre. La direction des autres flèches est ainsi claire.
Le seul endroit où il reste un choix est la partie en haut à droite : les trois enfants peuvent s'offrir des cadeaux dans le sens des aiguilles d'une montre ou en sens inverse.

C'est de l'informatique !

Les liens d'amitié entre les enfants forment un réseau constitué de nœuds (les enfants) et d'arcs (les liens d'amitié). Les « réseaux sociaux » sont formés de manière similaire avec des millions d'utilisateurs. Ces systèmes possèdent un aspect pouvant toutefois les différencier fondamentalement : certains possèdent des « amitiés » réciproques pour lesquels les liens n'ont pas de direction, comme dans cet exercice. D'autres fonctionnent avec des abonnés (« follower » en anglais), ce qui fait que les liens ont une direction : par exemple, si tu suis une utilisatrice célèbre, cela ne veut pas dire qu'elle te suit également.
Dans cet exercice, il faut assigner une direction aux liens d'amitié pour une distribution de cadeaux. C'est un nouvel aspect du système, car chaque enfant a une capacité limitée d'offrir, ce qui met indirectement des limites au choix de la direction. Le but est qu'un cadeau soit offert dans chaque amitié sans dépasser la capacité d'offrir des enfants. Il existe des problèmes similaires en informatique : dans un réseau (comme par exemple les câbles constituant Internet), la capacité des connections est limitée. Cette capacité doit être utilisée de manière optimale tout en respectant les limites.
Le problème du débit maximum dans un réseau peut être résolu de manière efficace. Étant donné que la structure de notre exercice et celle du problème du débit maximum sont semblables, on peut appliquer la même méthode pour le résoudre. Ceci arrive souvent en informatique : un problème peut être transformé en un autre problème ayant la même structure qui a déjà été facilement résolu.

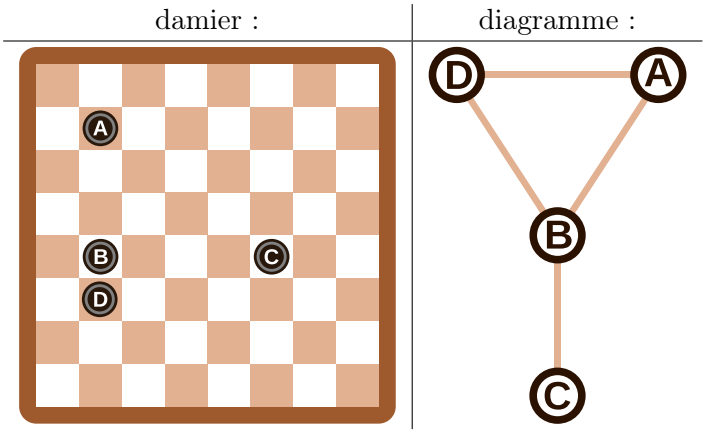
Mots clés et sites web

Débit dans un réseau, réduction de problème

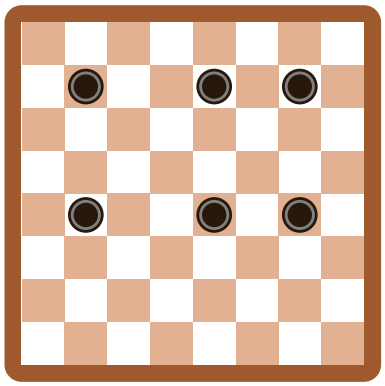
- `https://fr.wikipedia.org/wiki/Problème_de_flot_maximum`
- `https://fr.wikipedia.org/wiki/Réduction_(complexité)`

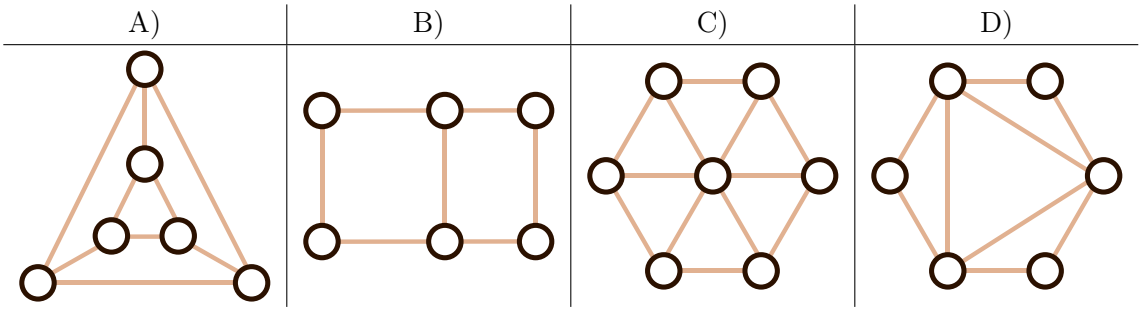
27. Rangées et colonnes

Le diagramme des palets de jeu montré à droite du damier a été construit de la manière suivante :
- Chaque palet est représenté par un cercle,
- deux palets sont reliés par une ligne s'ils se trouvent sur la même rangée ou colonne du damier.



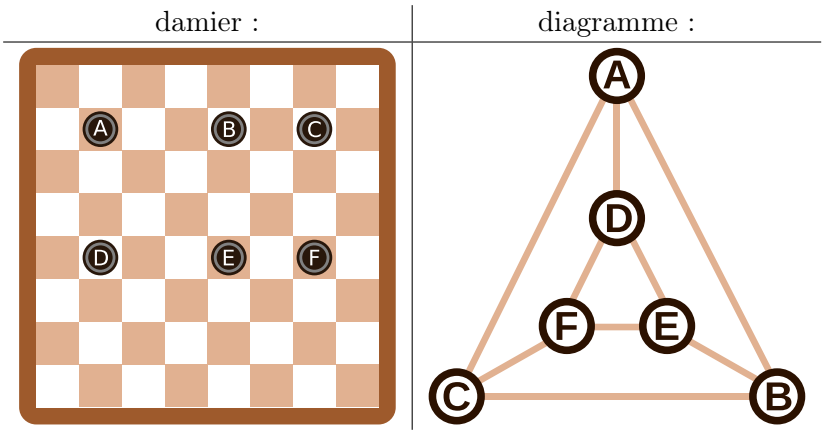
Pour cet exemple, les palets sur le damier et les cercles du diagramme sont annotés d'une lettre afin de mettre leur relation en évidence.
Quel diagramme correspond au damier à six palets suivant ?



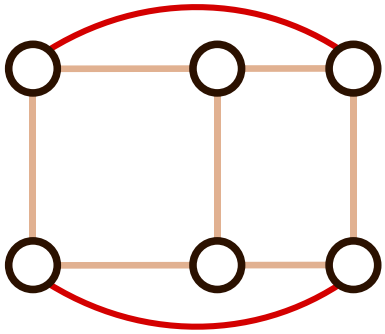


Solution

Le diagramme A) correspond au damier. On peut le vérifier à l'aide du graphique suivant sur lequel les palets et les cercles sont annotés :



Les diagrammes B), C) et D) peuvent être exclus de la manière suivante : chaque palet se trouve dans la même rangée que deux autres palets et dans la même colonne qu'un autre palet. Cela veut dire que chaque cercle dans le diagramme doit être relié à $2 + 1 = 3$ autres cercles, ce qui n'est le cas que sur le diagramme A). De plus, le diagramme C) comporte sept cercles, donc un de trop.
Le diagramme B) est faux, même s'il ressemble au damier. Les quatre cercles extérieurs ne sont reliés qu'à deux autres cercles. Il faudrait ajouter deux lignes au diagramme pour le corriger :



C'est de l'informatique !

En informatique, de tels diagrammes sont souvent utilisés pour représenter les informations essentielles d'un problème. On appelle ces diagrammes des graphes. Les cercles sont appelés « nœuds » et les lignes « arêtes ».
L'important dans un graphe est de savoir quels nœuds sont reliés par des arêtes. L'arrangement des nœuds ou la forme des arêtes ne jouent aucun rôle. Le même graphe peut donc être représenté de différentes manières, comme nous l'avons vu plus haut : le graphe de la réponse A) et le dernier graphe de l'explication sont les deux des solutions correctes, deux représentations du même graphe.
Les graphes sont une forme d'abstraction. Ils représentent l'essentiel d'un problème. Dans notre cas, on peut par exemple utiliser les graphes pour répondre à la question « Quel est le plus petit nombre de palets à enlever afin qu'il n'y ait jamais plus d'un palet par rangée et par colonne ? ». Une partie essentielle du travail d'informaticien consiste à trouver une bonne représentation du problème aidant à sa résolution.

Mots clés et sites web

Graphe

- `https://fr.wikipedia.org/wiki/Graphe_(mathématiques_discrètes)`
- `https://fr.wikipedia.org/wiki/Théorie_des_graphes`
- `https://fr.wikipedia.org/wiki/Graphe_(type_abstrait)`

28. Classement de livres

Trois castors sont assis chacun à une table avec deux livres. Ils veulent classer les livres voisins en échangeant leurs places. Chaque livre peut être déplacé au maximum une fois par tour. Les castors travaillent ensemble à chaque tour.
Il existe deux sortes de tours qui sont toujours effectués l'un après l'autre :

A. Chaque castor peut (mais ne doit pas) inverser les deux livres sur sa table (exemple A).
B. Chaque livre peut (mais ne doit pas) être échangé avec le livre le plus proche sur une table voisine (exemple B).

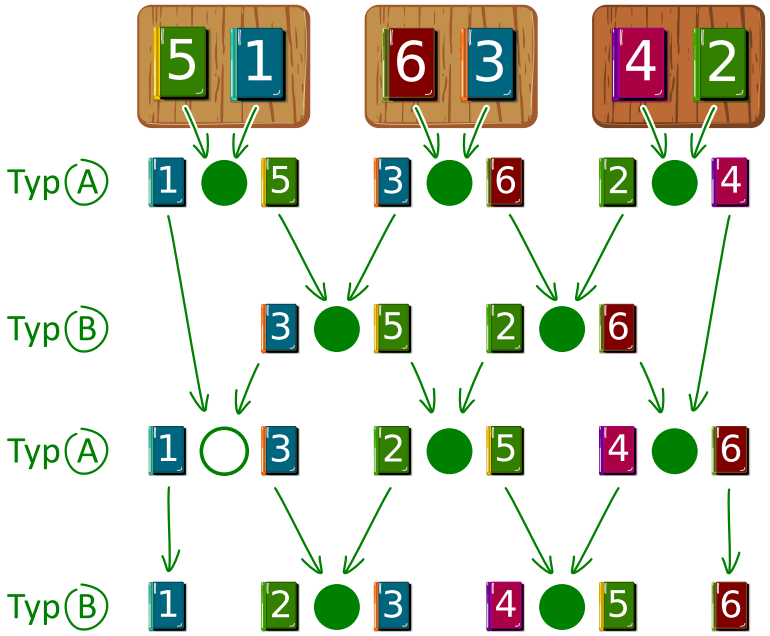
Au départ, les livres sont placés comme suit :

Lors du premier tour, chaque castor inverse les deux livres sur sa table.
Quel est le nombre de tours minimal nécessaire au classement des livres par ordre croissant, c'est-à-dire dans l'ordre 1, 2, 3, 4, 5, 6 *?*

A) trois tours
B) quatre tours
C) cinq tours
D) six tours

Solution

La bonne réponse est B).
L'illustration montre de quelle manière les livres peuvent être classés en échangeant leur place. Les castors utilisent une stratégie « gloutonne ». Cela veut dire qu'ils essaient de se rapprocher de la solution à chaque étape : ils comparent les livres voisins pouvant être échangés durant le tour en cours, et ne les échangent que s'ils ne sont pas encore dans le bon ordre (donc si le livre de gauche porte un numéro plus grand que celui de droite) ; s'ils sont déjà classés, ils ne font rien.
Lors du premier tour (sorte A), les deux livres sur chaque table sont échangés. Lors du deuxième tour (sorte B), les livres voisins situés sur les tables adjacentes sont échangés, lors du troisième tour (sorte A), seuls les livres sur les deux tables les plus à droites sont échangés. Finalement, lors du quatrième tour (sorte B), les quatre livres voisins situés sur des tables adjacentes sont inversés. Les livres sont maintenant classés. Ce n'est pas possible d'arriver à la solution en moins de tours, car le livre 5, par exemple, doit être déplacé de quatre positions en quatre tours pour arriver à la cinquième place.



C'est de l'informatique !

Le classement, ou tri, effectué dans cet exercice est un exemple d'algorithme parallèle. Plusieurs acteurs travaillent en même temps à la résolution d'un problème. Un procédé de tri parallèle peut être représenté par un réseau de tri comme dans l'illustration plus haut. Un réseau de tri est composé d'arêtes directionnelles appelées « fils » qui sont représentés par des flèches, et de nœuds appelés « comparateurs » qui sont représentés par des cercles.
Lors de chaque tour, les deux livres reliés à un comparateur sont comparés et, si nécessaire, inversés. Plusieurs paires de livres, reliées à des comparateurs adjacents, peuvent être comparées en même temps. On peut suivre le chemin d'un livre d'un échange à l'autre l'amenant à la position souhaitée en suivant le fil qui lui correspond de haut en bas.

Mots clés et sites web

Tri parallèle, réseau de tri

— `https://fr.wikipedia.org/wiki/Réseau_de_tri`
— `https://fr.wikipedia.org/wiki/Algorithme_glouton`

29. Soundex

Donald aimerait encoder des mots d'après leur prononciation. Il procède de la façon suivante :

- Garde la première lettre.
- Supprime A, E, I, O, U, H, W et Y parmi toutes les lettres suivant la première.
- Remplace les lettres suivantes comme suit :
  - B, F, P ou V → 1
  - C, G, J, K, Q, S, X ou Z → 2
  - D ou T → 3
  - L → 4
  - M ou N → 5
  - R → 6
- Si deux lettres ou plus encodées par le même nombre sont adjacentes dans le mot d'origine, ne retiens que la première des deux lettres. Cela vaut également pour la première lettre du mot.
- Ne garde que les quatre premiers signes (y compris la première lettre) en complétant si nécessaire par des zéros.

Les mots suivants sont encodés comme suit :

Euler → E460
Gauss → G200
Heilbronn → H416
Kant → K530
Lloyd → L300
Lissajous → L222

Quel est le code pour le mot « Hilbert » ?

A) H410

B) B540

C) H041

D) H416

Solution

La première lettre étant un H, le premier signe du code est également H.
Tous les A, E, I, O, U, H et W sont ensuite enlevés, il ne reste donc que « lbrt » à traduire.
En remplaçant les lettres par le chiffre correspondant, on obtient H4163.
Il ne faut rien éliminer car il n'y a pas de lettres doubles.
En ne gardant que les quatre premiers signes, on obtient H416.

C'est de l'informatique !

Le procédé Soundex, plus exactement le Soundex américain, a été développé et patenté il y a 100 ans par Robert C. Russel et Margaret King Odell. Il a été utilisé pour trouver des mots, en particulier des noms propres, à consonance similaire dans la langue anglaise. Cela fonctionne parce que les groupes de lettres encodées par un même chiffre ont une phonétique similaire : en anglais, B, F, P et V sont des consonnes bilabiales, C, G, J, K, Q, S, X et Z des labio-dentales, D et T des dentales, L une alvéolaire, M et N des vélaires et R une laryngale.

Ce procédé étant très simple et donnant de relativement bons résultats également dans d'autres langues que l'anglais (en faisant éventuellement quelques modifications au code pour refléter la phonétique de la langue), il est souvent utilisé pour la recherche phonétique, c'est-à-dire la recherche de mots prononcés similairement. Il est inclus comme standard dans beaucoup de bases de données.
Les exemples cités plus haut viennent de Donald Knuth, un des grands informaticiens du 20^e siècle, qui travaille encore aujourd'hui à son livre « The Art of Computer Programming ». Le procédé décrit ici se trouve dans le troisième volume, « Sorting and Searching ».

Mots clés et sites web

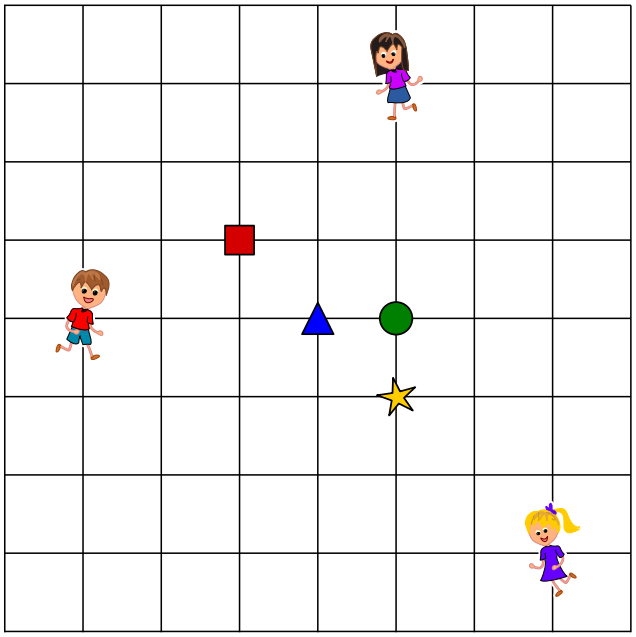
Recherche phonétique, Soundex

- `https://www.functions-online.com/soundex.html`
- `https://fr.wikipedia.org/wiki/Soundex`
- `https://www-cs-faculty.stanford.edu/~knuth/taocp.html`
- `http://www.highprogrammer.com/alan/numbers/soundex.html`

30. Trois amis

Alice  , Bob  et Céline  habitent à la Chaux-de-Fonds. Ils ont marqué leurs domiciles sur le plan. Ils aimeraient trouver un lieu de rendez-vous pour lequel la somme de leurs distances de trajet est la plus petite possible. La distance de trajet est calculée en additionnant le nombre de segments entre deux intersections.

 ,  ,  et  sont des lieux de rendez-vous possibles. Par exemple, la distance de trajet la plus courte pour Alice  jusqu'au lieu de rendez-vous  est 4, car il y a quatre segments entre l'intersection où Alice se trouve et celle où le triangle se trouve.



Quel est le lieu de rendez-vous pour lequel la somme des distances de trajet des trois amis est la plus courte ?

A)	B)	C)	D)
			

Solution

La bonne réponse est C)  . La somme des distances de trajet des amis jusqu'au rond vert est $3 + 4 + 5 = 12$.

Mauvaise solution :  . La somme des distances de trajet des amis jusqu'au carré rouge est $4 + 3 + 8 = 15$.

Mauvaise solution :  . La somme des distances de trajet des amis jusqu'au triangle bleu est $4 + 3 + 6 = 13$.

Mauvaise solution :  . La somme des distances de trajet des amis jusqu'à l'étoile jaune est $4 + 5 + 4 = 13$.

C'est de l'informatique !

Afin de déterminer lequel des quatre lieux de rendez-vous est le meilleur, il faut calculer la somme des distances de trajet des trois amis pour chacun d'entre eux. Le lieu de rendez-vous ayant la plus petite somme est le meilleur. C'est facile et ça ne prend pas beaucoup de temps – le calcul peut même se faire mentalement. Mais le même problème pour un grand nombre d'amis et un grand nombre de lieux de rendez-vous possibles devient un problème d'optimisation qu'on ne peut en général pas résoudre en un temps raisonnable.
Il existe en informatique des procédés permettant de trouver une solution proche de la solution optimale en un temps raisonnable pour ce type de problème. La solution ne diffère alors de la solution optimale que de moins de 1%, par exemple. S'il s'agit, au lieu d'amis se retrouvant, de livreurs de journaux allant chercher les journaux à distribuer à un endroit commun, 6 secondes de plus ou de moins ne font pas grande différence sur un trajet de 10 minutes (6 secondes représentent 1% de 10 minutes).
La recherche locale est un exemple de procédé pour trouver une solution presque optimale : afin de trouver un lieu de rendez-vous pour un grand nombre d'amis, on commence lors d'une recherche locale par n'importe quel lieu de rendez-vous qui peut être choisi aléatoirement. On compare ensuite la somme des distances de trajet pour ce lieu et les lieux de rendez-vous voisins et détermine le meilleur d'entre eux. On peut de cette manière se rapprocher de l'optimum.
Mais pourquoi les trois amis habitent-ils à la Chaux-de-Fonds ? La Chaux-de-Fonds et Le Locle font partie du patrimoine mondiale de l'humanité de l'UNESCO depuis 2009, pas uniquement en raison de l'histoire horlogère, mais également parce que les deux villes ont été reconstruites au 19^{e} siècle suite à des incendies selon un plan en damier, similaire à celui de la donnée de l'exercice.

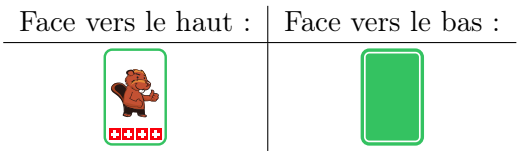
Mots clés et sites web

Problème d'optimisation, recherche locale

- `https://fr.wikipedia.org/wiki/Recherche_locale_(optimisation)`
- `https://whc.unesco.org/fr/list/1302/`

31. Tour de cartes

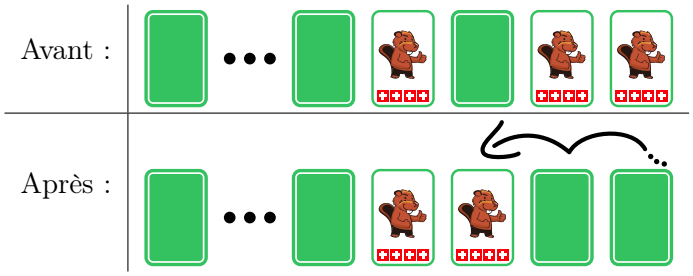
Tu reçois un paquet de cartes à jouer toutes pareilles. Elles sont comme cela :



Tu peux utiliser ces cartes pour jouer à « retourner ». Pour cela, tu alignes des cartes en une rangée devant toi.
Lors d'un tour de jeu, tu passes d'une carte à l'autre de droite à gauche comme suit :

— Si la carte est face vers le haut, tu la retournes à l'envers et passes à la suivante.
— Si la carte est face vers le bas, tu la retournes à l'endroit et finis ton tour de jeu sans toucher les autres cartes.

Un tour de jeu pourrait par exemple se passer comme ça :



Tu retournes les deux cartes de droite à l'envers. La suivante est face vers le haut. Tu la retournes à l'envers et finis ton tour.
Cette fois, le jeu commence avec 16 cartes face vers le bas.

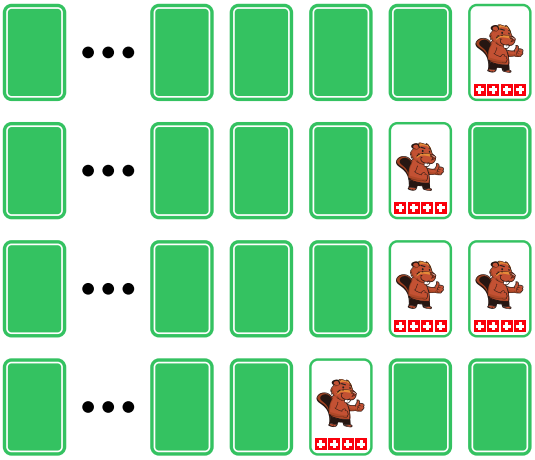
Combien de cartes sont face vers le haut après 16 tours ?

Solution

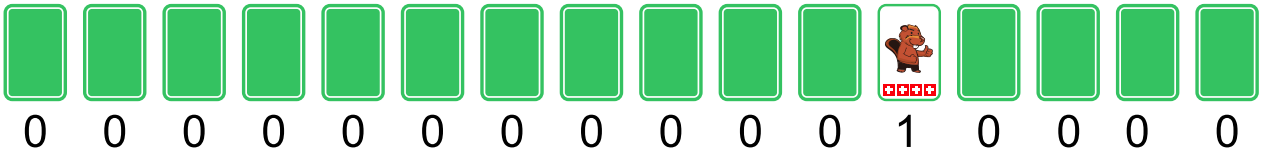
Il y a exactement une carte face vers le haut.
On peut se représenter une rangée de cartes pouvant être face vers le bas ou vers le haut comme un nombre binaire. Les nombres binaires ne sont composés que des chiffres 0 et 1. Par exemple, une carte face vers le bas peut représenter le chiffre 0 et une carte face vers le haut le chiffre 1.
De manière analogue au système décimal, chaque position d'un nombre binaire nous informe s'il faut inclure la valeur de la puissance de deux correspondante au nombre ou pas. Par exemple, si la troisième position (depuis la droite) d'un nombre binaire est occupée par le chiffre 1, il faut additionner la troisième puissance de deux au nombre final – donc 2^2, vu que $1 = 2^0$ est la première puissance de deux.
Les nombres binaires sont incrémentés de 1 de cette manière. On commence par le chiffre tout à droite :

— Si le chiffre actuel est 0, remplace-le par un 1. Tu as ainsi incrémenté le nombre de 1.
— Si le chiffre actuel est 1, remplace-le par un 0 et passe au nombre suivant pour le report.

Cela correspond exactement à un tour dans le jeu « retourner ». Un tour incrémente donc de 1 la valeur du nombre binaire représenté par la rangée de cartes. La rangée de cartes face vers le bas de départ représente un nombre binaire composé uniquement de zéros, qui a donc la valeur 0.
L'image suivante montre les résultats des quatre premiers tours qui correspondent aux nombres 1 à 4. On peut observer que pour les nombres 1, 2 et 4 (donc les puissances de deux 2^0, 2^1 et 2^2), il y a exactement une carte découverte, ce qui veut dire qu'un nombre binaire ayant une puissance de deux comme valeur n'a de 1 qu'à la position correspondant à cette puissance.



Après 16 tours, nous obtenons donc la représentation du nombre binaire ayant la valeur 16. Comme $16 = 2^4$, ce nombre n'a le chiffre 1 qu'à la cinquième position depuis la droite : 0000000000010000. C'est donc uniquement la cinquième carte depuis la droite qui est face vers le haut dans la représentation par la rangée de cartes :



C'est de l'informatique !

La plus petite unité de mémoire des ordinateurs actuels ne peut différencier que deux valeurs : allumé ou éteint, VRAI ou FAUX, 0 ou 1. Nous devons nous représenter toutes les données enregistrées ou

traitées par un ordinateur comme une série de chiffres binaires, donc des nombres binaires. C'est pourquoi les opérations avec des nombres binaires ont beaucoup d'importance en informatique.
Depuis que les ordinateurs existent, le calcul de telles opérations est implémenté aussi efficacement que possible. Certaines opérations combinent deux nombres binaires, similairement aux opérations arithmétiques d'addition ou de multiplication. D'autres opérations ne modifient qu'un seul nombre binaire, comme le décalage de tous les chiffres d'une position vers la gauche ou, justement, l'incrémentation, qui est l'addition du chiffre 1 comme dans cet exercice.
Les bons processeurs se démarquent par leur capacité à effectuer de telles opérations rapidement et en consommant peu d'énergie, et cela des millions de fois par seconde. C'est ensuite la tâche du programmeur et de ses outils de réduire la complexité des processus, y compris de ces simples opérations, afin que l'utilisateur puisse utiliser n'importe quel programme.

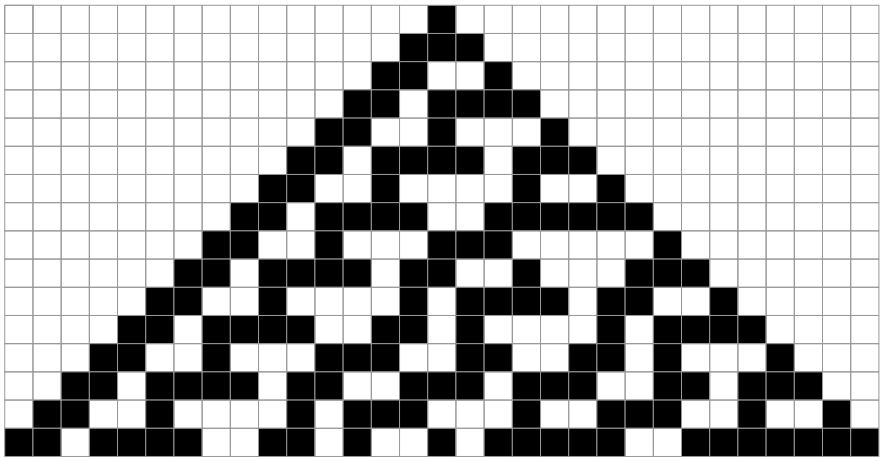
Mots clés et sites web

Nombres binaires
- `https://fr.wikipedia.org/wiki/Système_binaire`
- `https://fr.wikipedia.org/wiki/Compteur#Électronique`

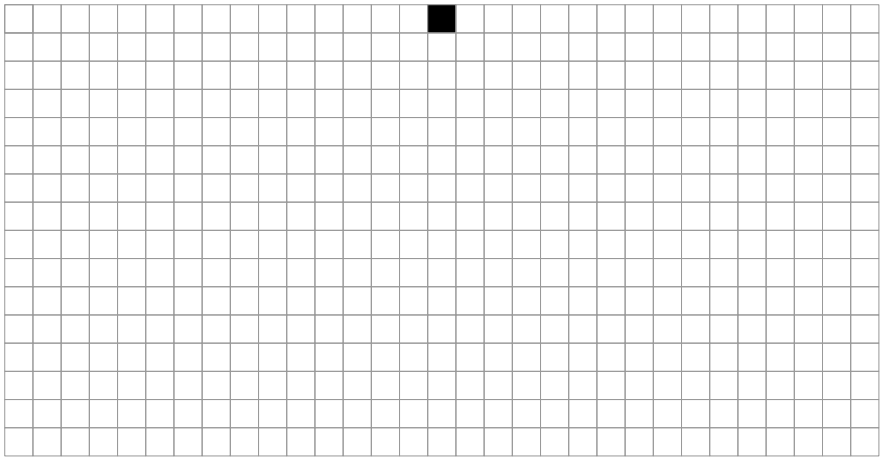
32. Catelles

Tina doit poser des catelles sur une surface d'une largeur de 31 catelles et d'une hauteur de 16 catelles. Elle aimerait arranger les catelles en suivant un set de règles simples.
Une règle simple est construite de manière à ce que trois catelles adjacentes déterminent l'apparence de la catelle devant être posée au centre en dessous de ces trois catelles. Un set de règles simples est constitué de huit règles simples : chaque combinaison possible de trois catelles adjacentes correspond à une règle simple (les bords sont considérés comme des catelles blanches) :

Tina commence au centre de la rangée du haut par une catelle noire ; toutes les autres catelles de la rangée du haut sont blanches. En appliquant ses règles, la surface est comme ceci :



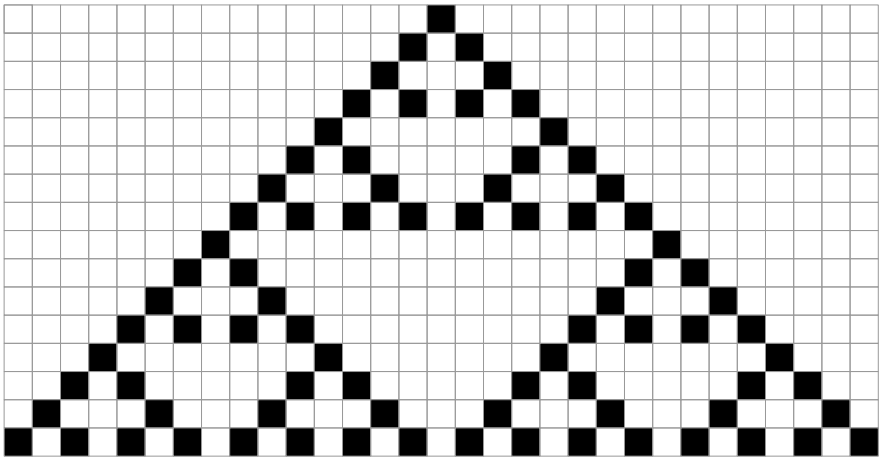
Etablis ton propre set de règles de façon à ce que la rangée du bas de la surface soit constituée tour à tour d'une catelle noire et d'une catelle blanche.

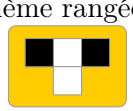


Solution

Cet exercice a beaucoup de solutions. Voici une des solutions possibles :

Elle génère le motif suivant :



Si l'on observe attentivement le motif, on peut voir qu'il s'agit en fait de neuf « triangles » : la troisième rangée générée est composée de catelles noires et blanches en alternance, ce qui fait que la règle  est toujours appliquée, excepté aux bords droit et gauche du motif où de nouveaux triangles peuvent être formés.
Si l'on dénote la catelle résultant d'une règle par B pour une catelle blanche et N pour une catelle noire, on peut écrire chacun des 256 sets de règles possibles (B ou N pour chacune de 8 règles, $2^8 = 256$) en tant que code de 8 lettres. L'exemple de Tina dans l'exercice aurait le code BBBNNNNB.
Les sept codes suivants génèrent un motif alterné noir et blanc dans la 16^e rangée :

SWSSWWSS
SSSSSWSW
WSSSSWSW
SWSSSWSW
WWSSSWSW
SSWSSWSW
WSWSSWSW
SWWSSWSW
WWWSSWSW
SSSSWWSW
WSSSWWSW
SWSSWWSW
WWSSWWSW
SSWSWWSW
WSWSWWSW
SWWSWWSW
WWWSWWSW (la solution donnée en exemple)

C'est de l'informatique !

Les règles dans cet exercice ressemblent au jeu de la vie de Conway (*Conway's Game of Life* en anglais), qui a été publié par le mathématicien anglais John Horton Conway en 1970. Ce jeu se base sur un automate cellulaire en deux dimensions. Un tel automate ressemble à une surface carrelée : chaque catelle est une « cellule » dont l'état dépend des huit cellules voisines. Le nouvel état de chaque cellule est calculé d'après l'état (précédent) de ses huit voisines. De cette manière, on peut par exemple simuler la prolifération et la disparition d'êtres vivants dans une certaine région. Dans notre cas, on utilise un set de règles réduit, où chaque cellule ne dépend que des trois cellules au dessus d'elle.

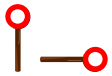
Mots clés et sites web

Motif, automate cellulaire

- `http://web.stanford.edu/~cdebs/GameOfLife/`
- `https://en.wikipedia.org/wiki/Rule_90`
- `https://fr.wikipedia.org/wiki/Automate_cellulaire`
- `https://fr.wikipedia.org/wiki/Jeu_de_la_vie`

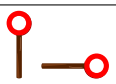
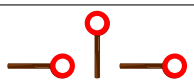
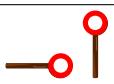
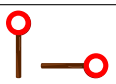
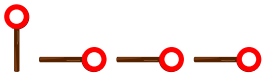
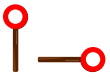
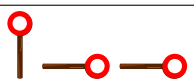
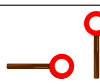
33. Où est le planeur ?

Jana et Robin jouent avec leur planeur. L'un d'eux le fait voler depuis une petite colline et l'autre va le rechercher après chaque atterrissage. Malheureusement, l'herbe de la prairie n'a pas été tondue depuis assez longtemps, ce qui fait qu'après l'atterrissage, on ne voit l'avion que depuis la colline et plus depuis la prairie. Jana et Robin doivent donc bien pouvoir communiquer. Pour ce faire, ils se sont mis d'accord sur un code de signaux.

gauche	droite	direction colline	direction vallée
			

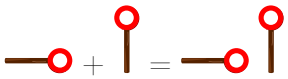
Il y a malheureusement un problème avec ce code. Si l'on envoie par exemple l'instruction suivante…

…elle peut être interprétée comme « gauche – en direction de la colline – gauche », mais aussi comme « gauche – droite– gauche – gauche ».
Jana et Robin se sont mis d'accord sur quatre nouveaux signaux pour leur code. Quel groupe de signaux peut-il être utilisé sans ambiguïté ?

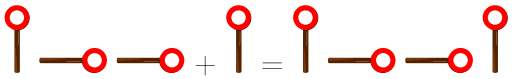
	gauche	droite	direction colline	direction vallée
A)				
B)				
C)				
D)				

Solution

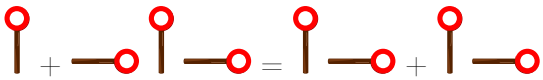
La bonne réponse est C). Cela peut être déterminé, par exemple, en observant que chaque instruction commence par le signal  et est suivie de 0, 1, 2 ou 3 fois  , sans que  ne réapparaisse. De cette manière, on sait qu'à chaque  , une nouvelle instruction commence et on ne doit que compter combien de fois  suit.
La réponse B) n'est pas un bon code de signaux, car « gauche » suivi de « droite » peut être interprété comme « direction colline » :



La réponse D) n'est pas un bon code non plus, car les instructions « gauche » suivies de « direction vallée » utilisent les mêmes signaux que l'instruction « direction colline » :

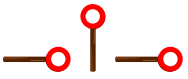
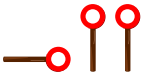


C'est un peu plus compliqué pour la réponse A). Si l'on signale « direction vallée » puis « gauche », c'est la même chose que si l'on signale deux fois « droite » :



C'est de l'informatique !

Lorsqu'un ordinateur transmet des données par câble ou par radiocommunication, il le fait en utilisant des suites rapides de signaux dans lesquels chaque signal de base peut avoir deux valeurs. On peut se représenter cela comme de l'électricité étant soit éteinte, soit allumée (naturellement, les choses sont souvent plus complexes). C'est exactement ce que font Jana et Robin : eux aussi utilisent deux signaux de bases pour coder leurs messages.
Cette manière de transformer des instructions ou des messages en signaux est appelée code (binaire). Dans ce cas-là, il s'agit d'un code à longueur variable, étant donné que le nombre de signaux utilisés pour un message ou une instruction ne doit pas toujours être le même.
C'est important que le destinataire d'un message codé puisse traduire les signaux du code au message original sans faire d'erreur. Autrement dit, tu dois faire attention lorsque tu développes un tel code. Les codes que nous qualifions de « bons » sont des codes qui peuvent être décodés de manière univoque.
Le code préfixe est une sorte spécifique de code univoque. Dans ce code, aucun des mots valables ne commence avec la suite complète des signaux formant un autre mot, par exemple :

gauche	droite	direction colline	direction vallée
			

Les codes préfixes ont des propriétés pratiques : ils restent relativement courts et sont faciles à décoder. Il ne sont pas uniquement utilisés dans la communication, mais aussi dans plusieurs algorithmes de compression.

Mots clés et sites web

Code, code préfixe, disque de signalisation

— `https://fr.wikipedia.org/wiki/Code_préfixe`

— `https://fr.wikipedia.org/wiki/Sémaphore_(communication)`

34. Horaire de répétition

Cinq danseurs répètent pour un spectacle : Alex, Bojan, Coco, Denis et Émile.
Lors de la représentation, les danseurs forment l'un après l'autre les binômes suivants :

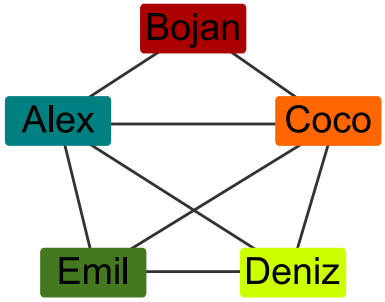
— Alex – Bojan
— Coco – Alex
— Émile – Deniz
— Alex – Émile
— Coco – Deniz
— Bojan – Coco
— Deniz – Alex
— Coco – Émile

Demain, les binômes vont répéter les uns après les autres. L'horaire doit être fixé de manière à ce qu'un membre du binôme répétant appartienne également au binôme suivant et puisse directement continuer la répétition. Afin de ne pas trop se fatiguer, aucun danseur ne doit répéter trois fois de suite. Par exemple, le binôme répétant après Alex – Bojan doit comprendre soit Alex, soit Bojan ; ce sera donc soit Coco – Alex, soit Alex – Émile, soit Bojan – Coco, soit Denis – Alex.
L'un des danseurs constate qu'il peut dans tous les cas venir plus tard à la répétition, car il ne fera pas partie du premier binôme sur l'horaire.
De quel danseur s'agit-il ?

A) Alex
B) Bojan
C) Coco
D) Deniz
E) Émile

Solution

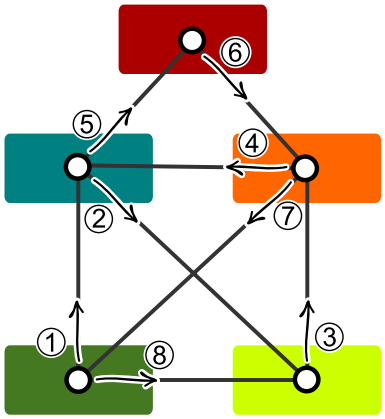
B) Bojan est la bonne réponse.
Chacun des danseurs est représenté par un rectangle encadrant son nom dans le diagramme suivant. Les rectangles représentant des danseurs qui font partie du même binôme sont reliés par une ligne. Un horaire valable peut alors être représenté par un chemin traversant le diagramme ; ce chemin commence par un rectangle, puis suit chaque ligne une seule fois. Un retour direct à un danseur après une étape n'est pas possible, sinon ce danseur devrait répéter trois fois de suite.



Le chemin doit quitter chaque rectangle par lequel il passe, excepté le dernier. Chaque rectangle par lequel le chemin passe de cette manière doit donc avoir un nombre pair de ligne (ce qui veut dire que le danseur fait partie d'un nombre pair de binômes). Les rectangles desquels le chemin commence et finit ont un nombre impair de lignes (les danseurs font donc partie d'un nombre impair de binômes). Les rectangles représentant Denis et Émile sont reliés à un nombre impair de lignes ; c'est donc les seuls devant faire partie du premier ou dernier binôme. Bojan est le seul danseur ne formant pas de binôme avec Denis ou Émile, et ne peut donc pas faire partie du premier binôme devant répéter selon l'horaire.

C'est de l'informatique !

L'image ci-dessus montre comment les binômes de danse peuvent être représentés sous forme de *graphe*. Un graphe est constitué de *nœuds* (ici, les danseurs) et d'*arêtes* (ici, la formation de binômes, donc les lignes). Un tel graphe a une structure polyvalente qui est souvent utilisée en informatique pour la modélisation des données d'un problème, par exemple pour des réseaux de transport ou de communication. Dans cet exercice, les danseurs forment un réseau de binômes.
Il est nécessaire dans beaucoup de réseaux de trouver un chemin partant d'un nœud et arrivant à un autre (qui peut varier) en parcourant toutes les connexions. La question se pose alors, pour des raisons d'efficacité par exemple, s'il est possible de trouver un tel chemin ne passant qu'une seule fois par chaque connexion. Un chemin ne parcourant qu'une seule fois chaque arête est appelé chemin eulérien en l'honneur de Leonhard Euler, qui a écrit le premier théorème des graphes. Euler a prouvé qu'il existe un chemin eulérien dans un graphe connexe si exactement deux de ses nœuds possèdent un nombre impair d'arêtes (les autres nœuds en possédant donc un nombre pair). Seuls ces deux nœuds peuvent être les points de départ ou d'arrivée du chemin eulérien.



Tu as peut-être déjà vu le réseau de binômes de cet exercice : s'il l'on réduit la taille des nœuds et que l'on modifie légèrement leurs positions, on reconnaît une enveloppe ou une maison (parfois appelée *maison de Saint-Nicolas*) pouvant être dessinée sans lever le crayon ni repasser sur un trait. En résolvant le problème de l'enveloppe, on dessine donc un chemin eulérien le long des lignes de l'enveloppe.

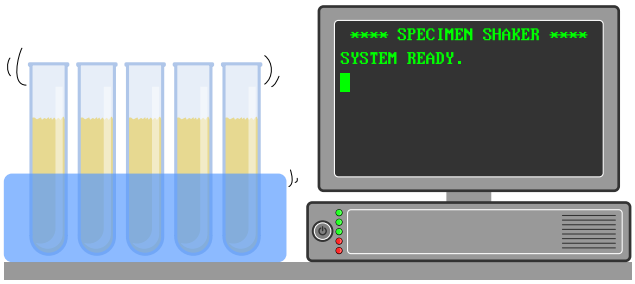
Mots clés et sites web

Graphe, chemin eulérien

— `https://fr.wikipedia.org/wiki/Graphe_eulérien`
— `https://fr.wikipedia.org/wiki/Problème_du_dessin_de_l'enveloppe`

35. Laboratoire

Dans un laboratoire d'analyse médicale, l'échantillon d'un patient doit être secoué régulièrement. Le laboratoire utilise pour cela une machine qui exécute un programme. Le programme est exécuté ligne par ligne.
La machine est programmée avec les instructions suivantes :




```
1 ENREGISTRE 0 SOUS A
2 INCRÉMENTE A DE 1
3 VA À LIGNE 6
4 SI A ÉGALE 60, VA À LIGNE 8
5 ENREGISTRE 0 SOUS A
6 INCRÉMENTE A DE 1
7 VA À LIGNE 2
8 RÉPÈTE A FOIS SECOUE
9 FIN
```

Les instructions possibles sont :

- `ENREGISTRE` *`Chiffre`* `SOUS` *`Nom`* : le chiffre *`Chiffre`* est enregistré sous le nom *`Nom`*.
- `INCRÉMENTE` *`Nom`* `DE 1` : lit le chiffre enregistré sous *`Nom`*, additionne 1 et enregistre le chiffre incrémenté sous *`Nom`*.
- `VA À LIGNE` *`Ligne`* : continue à exécuter le programme à partir de la ligne numéro *`Ligne`*.
- `SI` *`Nom`* `ÉGALE` *`Chiffre`*`, ALORS` *`Instruction`* : compare le chiffre enregistré sous *`Nom`* avec le chiffre *`Chiffre`*. S'ils sont égaux, exécute l'instruction *`Instruction`*, sinon pas.
- `RÉPÈTE` *`Nom`* `FOIS` *`Instruction`* : Exécute l'instruction *`Instruction`* aussi souvent que la valeur du chiffre enregistré sous *`Nom`*.
- `SECOUE` : secoue l'échantillon une fois.
- `FIN` : arrête l'exécution du programme.

Combien de fois la machine va-t-elle secouer l'échantillon ?

A) L'échantillon n'est pas secoué.

B) L'échantillon est secoué une fois.

C) L'échantillon est secoué 60 fois.

D) La machine ne va pas arrêter de secouer l'échantillon.

Solution

La bonne réponse est A).
Le programme passe sans cesse de la ligne 3 à la ligne 6 et de la ligne 7 à la ligne 2. Mis à part la ligne 1 au départ, seules les lignes 2, 3, 6 et 7 sont donc exécutées. L'échantillon serait secoué à la ligne 8, mais elle n'est jamais exécutée, ce qui veut dire que le programme ne va jamais faire secouer l'échantillon. Comme la ligne 9 n'est jamais exécutée non plus, le programme ne s'arrête jamais.

C'est de l'informatique !

Le programme utilise l'instruction « `VA À LIGNE` » (« `GO TO` » en anglais) comme structure de contrôle pour passer à d'autres parties du programme. Cette structure de contrôle, étroitement liée au matériel informatique (« hardware »), était fréquemment utilisée dans les premiers langages de programmation (jusqu'aux années 1980) pour permettre au programme de réagir à des entrées de l'utilisateur ou à d'autre conditions. Ce n'est cependant pas toujours simple pour un être humain de lire et de comprendre un code tel que celui compris par les ordinateurs, ce qui génère fréquemment des erreurs. Les langages de programmation modernes, tels qu'ils sont développés depuis les années 1950 et qui s'imposent de plus en plus, n'utilisent plus cette structure de contrôle. On utilise à la place des boucles (comme `RÉPÈTE` dans l'exemple), des branchements à d'autres parties du programme ou des sous-routines.

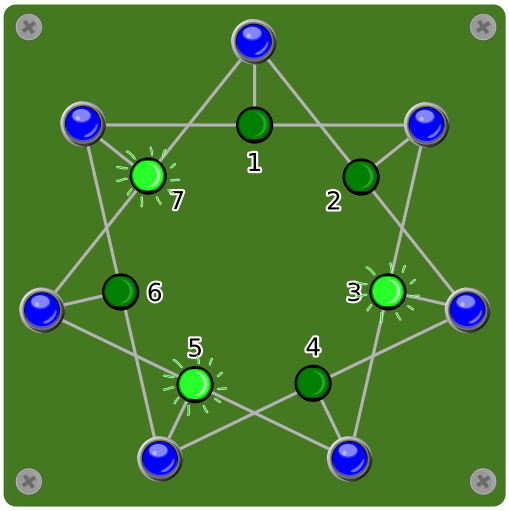
Mots clés et sites web

Structure de contrôle, analyse statique de programme, `GOTO`

- `https://homepages.cwi.nl/~storm/teaching/reader/Dijkstra68.pdf`
- `https://fr.wikipedia.org/wiki/Goto_(informatique)`
- `https://fr.wikipedia.org/wiki/Analyse_statique_de_programmes`
- `https://fr.wikipedia.org/wiki/Structure_de_contrôle`
- `https://fr.wikipedia.org/wiki/Routine_(informatique)`

36. Lumière !

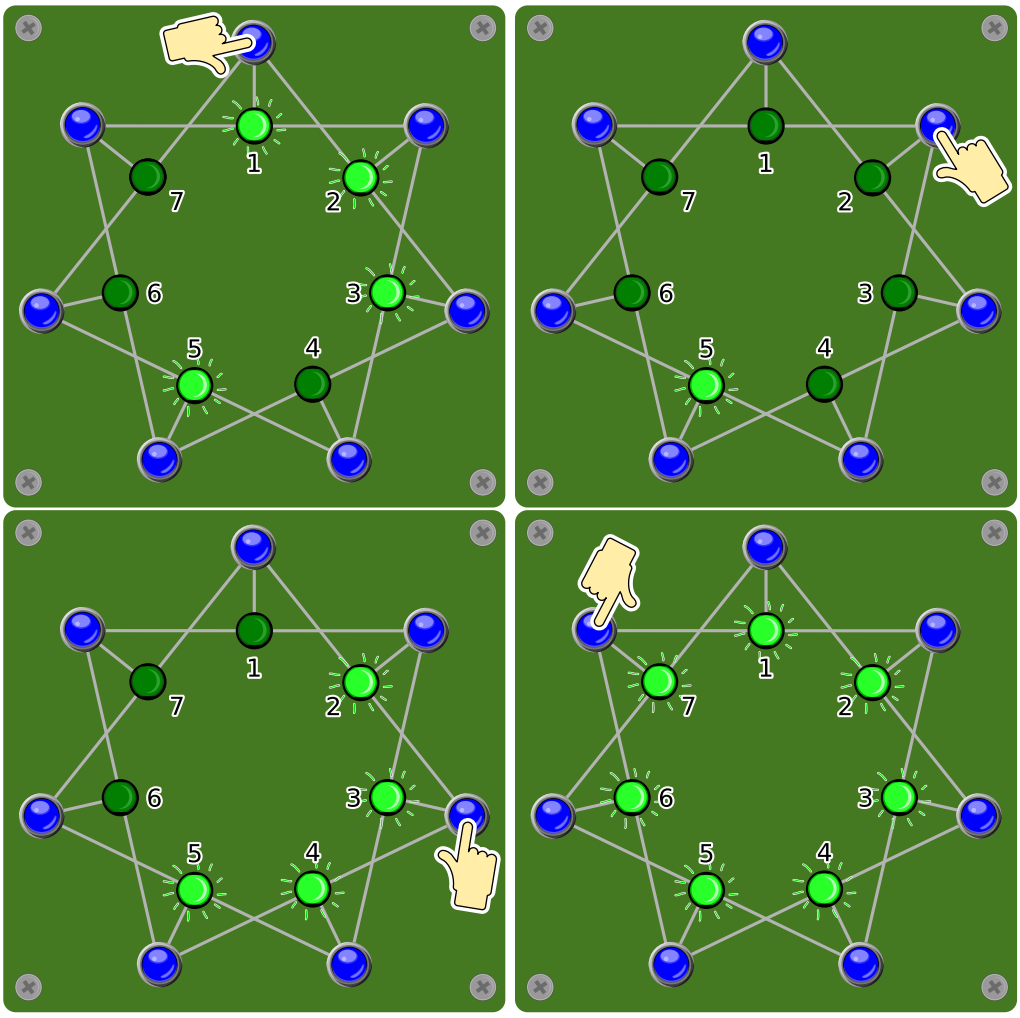
Sept interrupteurs sont reliés à sept lampes, et cela de manière à ce que chaque interrupteur contrôle trois lampes. Lorsque l'on appuie sur un interrupteur, les lampes y étant reliées s'éteignent si elles étaient allumées et s'allument si elles étaient éteintes.



Appuie sur les interrupteurs afin que toutes les lampes soient allumées en même temps !

Solution

Lorsque l'on appuie sur l'interrupteur près de la lampe 1 (ou de la lampe 2), les lampes 1 et 2 s'allument et la lampe voisine (7 ou 3) s'éteint. Trois lampes voisines sont ainsi allumées (1, 2, 3 ou 7, 1, 2). En appuyant sur l'interrupteur au centre de ces trois lampes voisines (2 ou 1, suivant quel interrupteur a été utilisé auparavant), ces trois lampes s'éteignent. Toutes les lampes excepté la lampe 5 sont maintenant éteintes. Comme il s'agit de six lampes, elles peuvent être allumées en appuyant juste sur les deux interrupteurs situés au milieu des deux groupes de trois (interrupteurs 3 et 7). Toutes les lampes sont à présent allumées.



On arrive au bon résultat quelque soit l'ordre dans lequel on actionne ces quatre interrupteurs (1, 2, 3, 7). En appuyant deux fois sur un interrupteur, on annule l'action suivant la première utilisation de l'interrupteur. La question est donc uniquement sur quels interrupteurs il faut appuyer. Avec sept interrupteurs, cela donne tout de même $2^7 - 1 = 127$ possibilités différentes d'activer les interrupteurs (c'est manifestement faux de n'appuyer sur aucun interrupteur).

C'est de l'informatique !

Le but de cet exercice est de faire passer un système d'un état connu (les lampes 3, 5 et 7 sont allumées) à un autre état également connu (les lampes 1, 2, 3, 4, 5, 6, et 7 sont allumées) tout en respectant certaines règles. La tâche principale est de trouver le chemin amenant d'un état à l'autre.

On rencontre souvent de telles questions en informatique. On pourrait bien sûr naïvement commencer en essayant toutes les combinaisons (ce qui nous amènerait aux 127 possibilités mentionnées plus haut). Cependant, si l'on veut arriver à la solution plus rapidement, cela vaut la peine de considérer le problème de manière bidirectionnelle : d'un côté, en partant de l'état de départ pour atteindre l'état final et, parallèlement, en partant de l'état final pour atteindre l'état de départ. Plus le système est grand, plus l'économie de temps pouvant être réalisée grâce à cette méthode est grande.

Mots clés et sites web

Recherche bidirectionnelle, automate fini

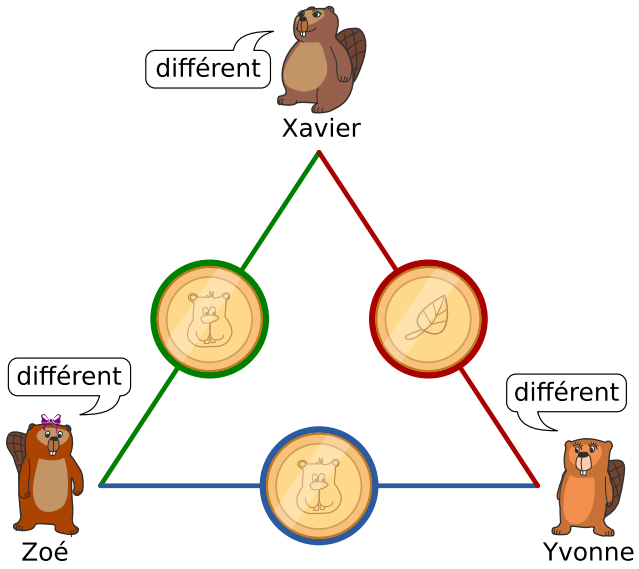
— `https://en.wikipedia.org/wiki/Bidirectional_search`
— `https://fr.wikipedia.org/wiki/Parcours_de_graphe`

37. Top secret

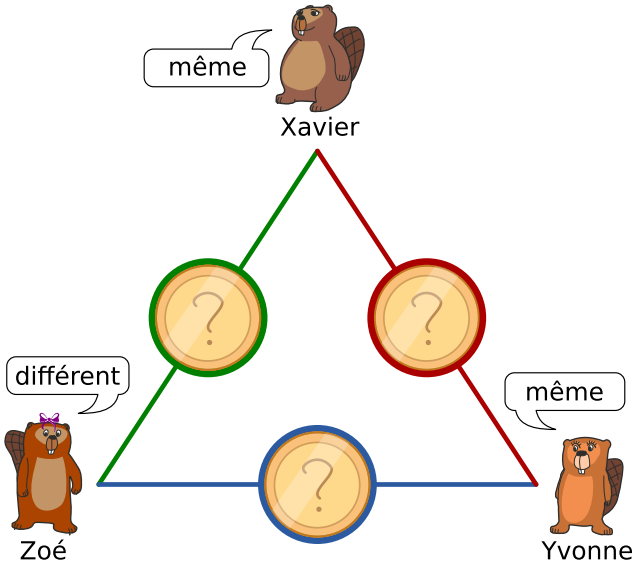
Xavier, Yvonne et Zoé sont amis et jouent de temps en temps à la tombola. Ils viennent de découvrir qu'une personne habitant leur ville a gagné le gros lot. Ils aimeraient savoir si, par hasard, l'un d'entre eux a gagné ; d'un autre côté, l'identité du gagnant doit rester secrète. Ils procèdent de la manière suivante :

1. Xavier et Yvonne tirent au sort en lançant une pièce ; eux seuls connaissent le résultat.
2. Xavier et Zoé tirent au sort en lançant une pièce ; eux seuls connaissent le résultat.
3. Yvonne et Zoé tirent au sort en lançant une pièce ; eux seuls connaissent le résultat.
4. Ensuite, chacun doit annoncer si ses deux tirages ont eu le « même » résultat ou deux résultats « différents ».
   — Quelqu'un n'ayant pas gagné le gros lot doit dire la vérité (c'est à dire « même » si ses deux jets ont eu le même résultat et « différent » sinon).
   — Quelqu'un ayant gagné le gros lot doit mentir (c'est-à-dire dire « même » si ses deux jets ont eu des résultats différents et vice-versa).

Ci-dessous un exemple de jet partant du principe que Zoé a gagné le gros lot.



Considère la situation suivante ; tu ne sais pas si l'un des trois amis a gagné le gros lot.



Laquelle des affirmations suivantes est vraie ?

A) Aucun des trois amis n'a gagné le gros lot.

B) L'un des trois amis a gagné le gros lot, mais nous ne savons pas lequel.

C) L'un des trois amis a gagné le gros lot, et nous savons exactement lequel.

D) Nous ne savons pas si l'un des trois amis a gagné le gros lot.

Solution

La bonne réponse est B). L'un des trois amis a gagné le gros lot, mais nous ne savons pas lequel.
Il y a plusieurs possibilité de résoudre ce problème. L'une d'entre elle est de procéder comme suit.
Si l'on effectue trois jets, on peut obtenir l'un des deux résultats suivants :

— Les trois pièces tombent du même côté.
— L'une des trois pièces tombe d'un côté différent des deux autres.

En partant du principe que personne n'a gagné le gros lot, il existe donc les possibilités suivantes :

— Si toutes les pièces sont tombées du même côté, les trois amis disent « même ».
— Si l'une des pièce est tombée d'un côté différent des deux autres, deux amis disent « différent » et le dernier dit « même ».

Mais comme deux des amis disent « même » et un dit « différent », l'un d'entre ment obligatoirement et a donc gagné le gros lot.
On ne peut cependant pas savoir lequel a gagné : en effet, si l'un des deux ayant dit « même » a menti, nous ne savons pas lequel des deux ; il se peut également que le troisième ait menti en disant « différent » et que les trois jets aient eu le même résultat.

C'est de l'informatique !

La manière de procéder choisie par Xavier, Yvonne et Zoé a été décrite pour la première fois sous le nom de *Dining Cryptographers Problem*.
Cette méthode est spécialement intéressante pour les informaticiens, car elle permet à l'auteur et au destinataire d'un message de rester anonyme. Si la méthode est utilisée correctement, elle permet à tout le monde de savoir avec certitude si l'un d'entre eux a gagné sans que personne ne sache de qui vient l'information, à part le gagnant lui-même. Le destinataire reste également anonyme, étant donné que tous les participants ont reçu l'information.

Mots clés et sites web

Anonymat, Dining Cryptographers Problem

— `https://en.wikipedia.org/wiki/Dining_cryptographers_problem`

A. Auteurs des exercices

Andrea Adamoli
Jared Asuncion
Wilfried Baumann
Carlo Bellettini
Javier Bilbao
Daphne Blokhuis
Laura Briviba
Lucia Budinská
Špela Cerar
William Chan
Kessarapan Charoensueksa
Anton Chukhnov
Kris Coolsaet
Valentina Dagienė
Darija Dasović Rakijašić
Christian Datzko
Susanne Datzko
Dilek Doğan
Marissa Engels
Hanspeter Erni
Veerle Fack
Georgios Fessakis
Gerald Futschek
Ionuț Gorgos
Shuchi Grover
Yasemin Gülbahar
Martin Guggisberg
Bent Halden
Urs Hauser
Wei-fu Hou
Juraj Hromkovič
Takeharu Ishizuka
Svetlana Jakšić
Zhang Jinbao
Emil Kelevedjiev
Dong Yoon Kim
Vaidotas Kinčius
Iryna Kirynovich
Jia-Ling Koh
Regula Lacher
Anh Vinh Le
Dan Lessner
Judith Lin
Violetta Lonati
Nils Mak
Dimitris Mavrovouniotis
Karolína Mayerová
Mattia Monga
Samart Moodleah
Anna Morpurgo
Tom Naughton
Henry Ong
Sanja Pavlovic Šijanović
Péter Piltmann
Zsuzsa Pluhár
Wolfgang Pohl
Ilya Posov
Stavroula Prantsoudi
Nol Premasathian
J.P. Pretti
Doris Reck
Alei Reyes
Chris Roffey
Kirsten Schlüter
Andrea Maria Schmid
Victor Schmidt
Andrea Schrijvers
Eljakim Schrijvers
Vipul Shah
Mohamed El-Sherif
Jacqueline Staub
Allira Storey
Gabrielė Stupurienė
Faisal Al-Sudani
Márta Szabó
Aliaksei Tolstsikau
Peter Tomcsányi
Ahto Truu
Willem van der Vegt
Jiří Vaníček
Troy Vasiga
Rechilda Villame
Eslam Wageed
Pieter Waker
Michael Weigend
Khairul A. Mohamad Zaki
Magdalena Zarach

B. Sponsoring : Concours 2018

http://www.haslerstiftung.ch/

http://www.roborobo.ch/

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

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

Standortförderung beim Amt für Wirtschaft und Arbeit Kanton Zürich

i-factory (Musée des transports, Lucerne)

http://www.ubs.com/

http://www.bbv.ch/

http://www.presentex.ch/

http://www.zubler.ch/
Zubler & Partner AG Informatik

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

http://www.diartis.ch/
Diartis AG

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

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

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

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

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

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

C. Offres ultérieures

Devenez vous aussi membre de la SSIE
`http://svia-ssie-ssii.ch/la-societe/devenir-membre/`
et soutenez le Castor Informatique par votre adhésion
Peuvent devenir membre ordinaire de la SSIE toutes les personnes qui enseignent dans une école primaire, secondaire, professionnelle, un lycée, une haute école ou donnent des cours de formation ou de formation continue.
Les écoles, les associations et autres organisations peuvent être admises en tant que membre collectif.