INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Exercices et solutions 2019
Années HarmoS 13/14/15

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

Éditeurs :
Gabriel Parriaux, Jean-Philippe Pellet, Elsa Pellet, Christian Datzko, Susanne Datzko, Juraj Hromkovič, Regula Lacher

SSIE
www.svia-ssie-ssii.ch
schweizerischervereinfürinformatikind
erausbildung//sociétésuissepourl'infor
matiquedansl'enseignement//societàsviz
zeraperl'informaticanell'insegnamento

Ont collaboré au Castor Informatique 2019

Christian Datzko, Susanne Datzko, Olivier Ens, Hanspeter Erni, Nora A. Escherle, Martin Guggisberg, Saskia Howald, Lucio Negrini, Gabriel Parriaux, Elsa Pellet, Jean-Philippe Pellet, Beat Trachsler.

Nous adressons nos remerciements à :
Juraj Hromkovič, Michelle Barnett, Michael Barot, Anna Laura John, Dennis Komm, Regula Lacher, Jacqueline Staub, Nicole Trachsler : ETHZ
Gabriel Thullen : Collège des Colombières
Valentina Dagienė : Bebras.org
Wolfgang Pohl, Hannes Endreß, Ulrich Kiesmüller, Kirsten Schlüter, Michael Weigend : Bundesweite Informatikwettbewerbe (BWINF), Allemagne
Chris Roffey : University of Oxford, Royaume-Uni
Carlo Bellettini, Violetta Lonati, Mattia Monga, Anna Morpurgo : ALaDDIn, Università degli Studi di Milano, Italie
Gerald Futschek, Wilfried Baumann, Florentina Voboril : Oesterreichische Computer Gesellschaft, Austria
Zsuzsa Pluhár : ELTE Informatikai Kar, Hongrie
Eljakim Schrijvers, Justina Dauksaite, Arne Heijenga, Dave Oostendorp, Andrea Schrijvers, Kyra Willekes, Saskia Zweerts : Cuttle.org, Pays-Bas
Christoph Frei : Chragokyberneticks (Logo Castor Informatique Suisse)
Andrea Leu, Maggie Winter, Brigitte Manz-Brunner : Senarclens Leu + Partner

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

INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Le Castor Informatique 2019 a été réalisé par la Société Suisse de l'Informatique dans l'Enseignement SSIE et soutenu par la Fondation Hasler.

HASLERSTIFTUNG

Tous les liens ont été vérifiés le 1er novembre 2019. Ce cahier d'exercice a été produit le 2 janvier 2020 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 2019 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 :

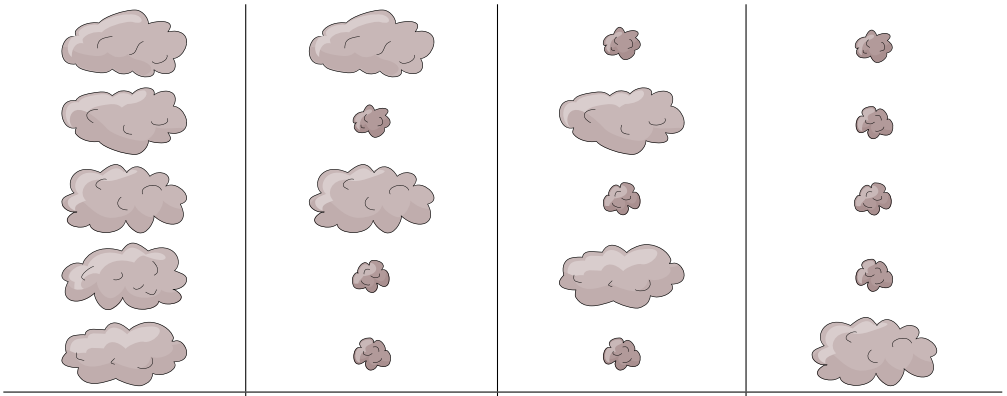
SVIA-SSIE-SSII Société Suisse de l'Informatique dans l'Enseignement
Castor Informatique
Gabriel Parriaux

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

Table des matières

1. Signaux de fumée

Un castor est toujours en haut de la montagne et observe la météo. Il transmet les prévisions météo aux castors dans la vallée. Pour cela, il utilise des signaux de fumée qui sont composés de cinq nuages de fumée. Un nuage de fumée peut être soit petit, soit grand. Les castors se sont mis d'accord sur les signaux de fumée suivants :



Ce sera orageux.	Ce sera pluvieux.	Ce sera nuageux.	Ce sera ensoleillé.

Un jour où il y a beaucoup de vent, les castors dans la vallée n'arrivent pas bien à reconnaître les nuages de fumée. Ils interprètent le signal de fumée comme cela :

Comme ce n'est aucun des signaux de fumée convenus, ils supposent qu'ils ont mal interprété l'un des nuages de fumée : l'un des petits nuages de fumée devrait en fait être grand ou l'un des grands nuages de fumée devrait en fait être petit.
Que voudrait dire le signal de fumée si exactement un nuage de fumée avait été mal interprété ?

A) Ce sera orageux.

B) Ce sera pluvieux.

C) Ce sera nuageux.

D) Ce sera ensoleillé.

Solution

Cinq signaux de fumée différents sont possibles si exactement un nuage de fumée a été mal interprété. Si l'on interprète le premier, deuxième, quatrième ou cinquième nuage de fumée différemment, on n'obtient aucun des quatre signaux de fumée convenus. Par contre, si l'on interprète le troisième nuage de fumée comme un petit nuage, cela donne la bonne réponse C) « Ce sera nuageux ».
On peut aussi comparer le signal de fumée interprété aux quatre signaux de fumée convenus et regarder combien de nuages de fumée sont différents. Pour le signal « Ce sera orageux », deux nuages de fumée sont différents (le plus haut et le plus bas), pour le signal « Ce sera pluvieux », trois nuages de fumée sont différents (les deux plus hauts et le plus bas), pour le signal « Ce sera nuageux », un nuage de fumée est différent (celui du milieu, ce qui en fait la bonne réponse comme décrit ci-dessus) et pour le signal « Ce sera ensoleillé », quatre nuages de fumée sont différents (tous sauf le plus haut).

C'est de l'informatique !

Lorsque l'on doit transmettre un message, on aimerait que ce message arrive correctement à son destinataire. Les messages de cet exercice sont transmis à l'aide de petits et de grands nuages de fumée. Dans le cas général, on parle de *symboles*. C'est donc raisonnable de choisir une suite de symboles qui permette de comprendre le message même s'il est endommagé en cours de route. On peut faire cela en transmettant plus d'informations qu'il n'est strictement nécessaire. On appelle ces informations supplémentaires *redondantes*.
Lorsque l'on peut reconstruire un message avec au maximum n erreurs, on parle de code correcteur avec une capacité de correction n. La représentation de messages par des suites de symboles de manière à ce que l'on puisse les reconstruire même lorsque cette représentation a été endommagée lors de la transmission est une tâche typique pour les informaticiens. Ils nous permettent ainsi par exemple de lire de la musique à partir de CD ou des vidéos à partir de DVD même lorsque quelques erreurs ont eu lieu lors de la transmission.
Dans cet exercice, deux nuages de fumée auraient suffi pour transmettre les quatre messages différents :

Ce sera orageux.	Ce sera pluvieux.	Ce sera nuageux.	Ce sera ensoleillé.

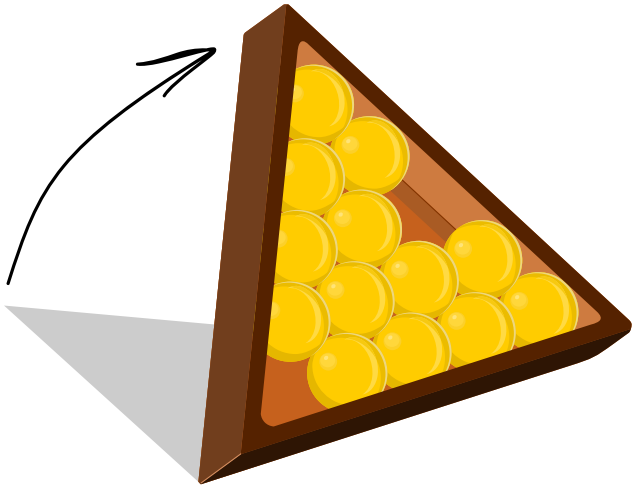
Les castors utilisent cependant cinq nuages de fumée. Cela leur permet de comprendre le message même dans les cas où deux voire parfois trois des nuages de fumée sont « illisibles ». Les castors ont de plus choisi les messages de manière à ce qu'il y ait au moins trois positions différentes entre chaque paire de messages.

Mots clés et sites web

Code correcteur
— `https://fr.wikipedia.org/wiki/Code_correcteur`

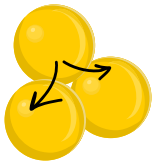
2. Boules instables

Une boîte triangulaire peut contenir quinze boules de la même taille. Deux boules sont retirées de la boîte comme dans le dessin ci-dessous. La boîte est ensuite inclinée sur le côté.



Lorsque l'on incline la boîte, certaines boules peuvent devenir «instables». Un boule est instable lorsque...

— ... la boule à gauche ou à droite en dessous d'elle a été retirée, ...
— ... ou la boule à gauche ou à droite en dessous d'elle est instable.

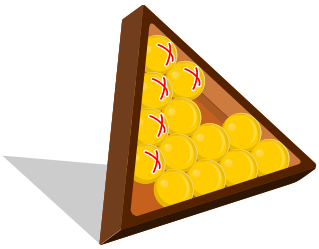


Les boules de la rangée du bas sont stables.
Combien des treize boules sont instables ?

A) Aucune boule
B) 1 boule
C) 2 boules
D) 3 boules
E) 4 boules
F) 5 boules
G) 6 boules
H) 7 boules
I) 8 boules
J) 9 boules
K) 10 boules
L) 11 boules
M) 12 boules
N) Toutes les boules

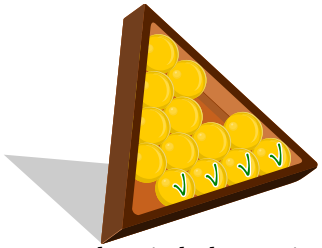
Solution

Cinq boules sont instables. Elles sont marquées d'une croix dans le dessin suivant :

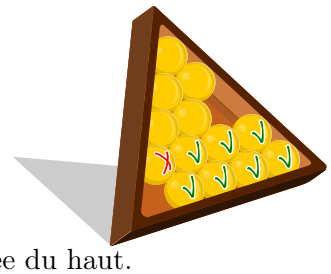


Le plus simple est d'y réfléchir en allant du bas vers le haut :

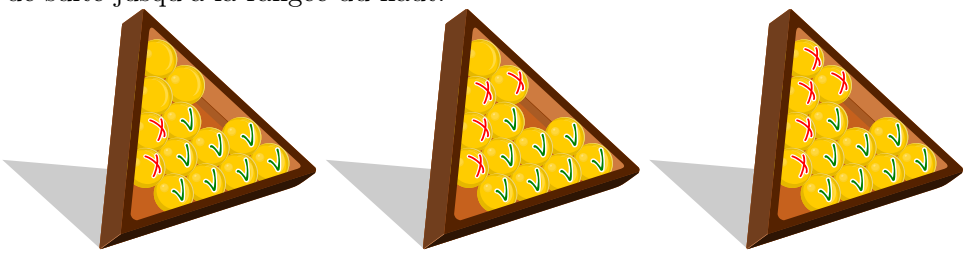
— Toutes les boules de la rangée du bas sont stables.



— Toutes les boules de la deuxième rangée depuis le bas qui ont deux boules stables en dessous d'elles sont également stables. Toutes les autres sont instables.



— Et ainsi de suite jusqu'à la rangée du haut.



C'est de l'informatique !

Il existe deux conditions pour qu'une boule soit classifiée comme instable. La première condition peut être testée directement. Afin de pouvoir tester la deuxième condition, on doit d'abord savoir si l'une des boules de la rangée en dessous est instable. C'est facile pour la rangée du bas, car comme il n'y a aucune boule en dessous, toutes les boules y sont stables. Comme expliqué dans la solution, on peut ensuite tester la rangée en dessus et déterminer quelles boules sont instables. De cette manière, on peut tester systématiquement chaque rangée du bas vers le haut et déterminer si chaque boule est instable.

Le principe d'après lequel une condition dépend du résultat d'une condition similaire s'appelle *récursivité*. Les conditions récursives sont construites de manière à ce que le résultat soit évident (*cas de base*, dans notre cas, toutes les boules de la dernière rangée sont stables) soit dépendant du résultat d'autres conditions récursives (*cas général*, dans notre cas, il s'agit de toutes les boules qui ne se

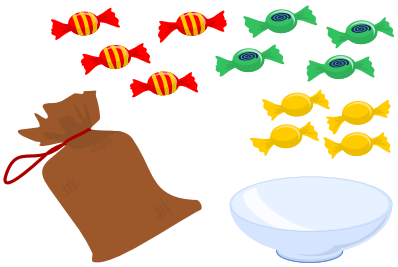
trouvent pas dans la dernière rangée et pour lesquelles il faut donc d'abord tester les boules du dessous).
Le principe de la récursivité est souvent utilisé en informatique. Il permet de résoudre de manière simple et élégante beaucoup de problèmes complexes. C'est également possible de transformer une solution récursive en une solution pas à pas (*itérative*). Un exemple classique pour lequel il existe une solution récursive simple est le jeu des tours de Hanoï.

Mots clés et sites web

Récursivité, tours de Hanoï

- `https://fr.wikipedia.org/wiki/R%C3%A9cursivit%C3%A9`
- `https://fr.wikipedia.org/wiki/Tours_de_Hano%C3%AF`

3. Un sac de bonbons



Petra a quatre bonbons rouges, quatre bonbons verts et quatre bonbons jaunes dans un sac opaque. Elle a aussi une coupe vide. Petra et Moritz jouent à un jeu. Pendant trois tours, Moritz peut tirer un bonbon du sac. Les règles suivantes valent pour chaque bonbon :

- Tant que le bonbon tiré est vert, il le met dans la coupe et peut tirer un autre bonbon pendant le même tour.
- Si le bonbon tiré est rouge, Moritz le met dans la coupe et termine le tour.
- Si le bonbon tiré est jaune, Moritz le mange directement sans le mettre dans la coupe et termine le tour.

Combien de bonbons au maximum Moritz peut-il avoir mis dans la coupe à la fin du jeu ?

A) 0
B) 1
C) 2
D) 3
E) 4
F) 5
G) 6
H) 7
I) 8
J) 9
K) 10
L) 11
M) 12

Solution

La bonne réponse est H) 7.
Dans le meilleur des cas, les quatre bonbons verts sont tirés. Cela signifie d'une part que les quatre bonbons verts sont dans la coupe et d'autre part que Moritz a pu tirer quatre fois un bonbon de plus au cours des trois tours, donc sept bonbons en tout.
Pour les trois bonbons restants, Moritz tire dans le meilleur des cas un bonbon rouge à chaque fois, qui sont ensuite aussi dans la coupe. Cela fait en tout quatre bonbons verts et trois bonbons rouges, donc sept bonbons dans la coupe en tout.
Il ne peut pas y avoir plus de sept bonbons dans la coupe. À chaque tirage, un bonbon au maximum peut être mis dans la coupe, et comme il n'y a que quatre bonbons avec lesquels on peut tirer un bonbon supplémentaire, cela fait au maximum sept bonbons.
L'ordre dans lequel les bonbons sont tirés dans le meilleur des cas est relativement égal tant que le dernier bonbon tiré est un bonbon rouge, car on peut dans ce cas toujours en tirer un de plus grâce aux bonbons verts.

C'est de l'informatique !

Deux des trois règles de l'exercice sont formulées comme des *instructions conditionnelles* : *si* une certaine condition est remplie, *alors* une certaine action est exécutée. De telles instructions conditionnelles sont très souvent utilisées en informatique. Souvent, les mots clés de langue anglaise *if* (« si » en anglais) et *then* (« alors » en anglais) sont utilisés. L'une des règles est formulée de manière à ce que quelque chose soit répété tant qu'une condition est remplie. On appelle cela une *boucle* pour laquelle on utilise souvent le mot de langue anglaise *while* (« tant que » en anglais). De tels boucles peuvent aussi être formulées comme des *boucles itératives* qui déterminent un nombre de répétitions fixe.
On pourrait donc formuler le jeu de Petra de la manière suivante :

fixe Tours égal à 3
*tant qu'*il reste un tour :
  diminue Tours de 1
  tire un bonbon
  tant que le bonbon est vert, alors mets-le dans la coupe et tire un bonbon
  si le bonbon est rouge, *alors* mets-le dans la coupe
  si le bonbon est jaune, *alors* mange-le

Pour résoudre l'exercice, on doit *analyser* le programme. Dans un cas simple comme celui de ce programme, on pourrait simplement essayer tous les ordres possibles de bonbons. Cela pourrait même être exécuté par un ordinateur de manière automatisée. L'explication de la solution, par contre, se base sur la compréhension des relations qui permet de prouver qu'une certaine solution est vraie sans exécuter le programme. De telles analyses ne peuvent pas toujours être réalisées par un ordinateur, comme l'a montré la *théorie de la calculabilité*. Donald Knuth, un des grands informaticiens du XX^e siècle, l'a exprimé ainsi : « Faites attention aux erreurs dans ce code ; je n'ai fait que démontrer qu'il était correct, je ne l'ai pas essayé. »

Mots clés et sites web

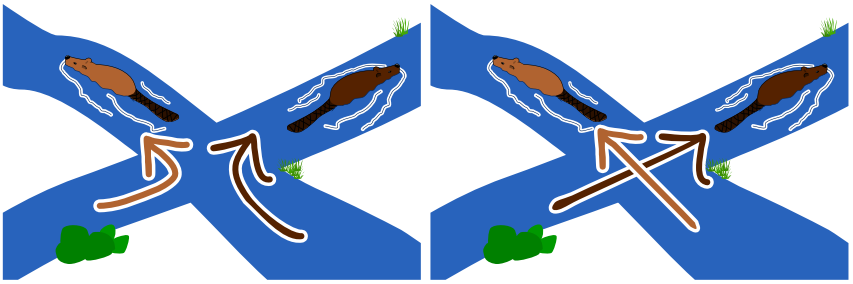
Instruction conditionnelle, boucle, théorie de la calculabilité

- `https://fr.wikipedia.org/wiki/Instruction_conditionnelle_(programmation)`
- `https://fr.wikipedia.org/wiki/Structure_de_contr%C3%B4le#Boucles`

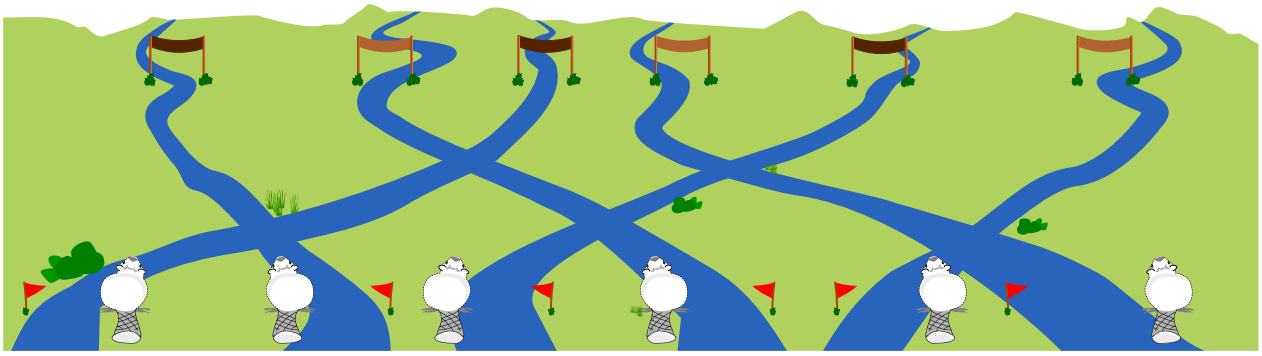
— `https://fr.wikipedia.org/wiki/Calculabilit%C3%A9`
— `https://en.wikiquote.org/wiki/Donald_Knuth`

4. Réseau de castors

Trois castors brun clair et trois castors brun foncé nagent dans un système de canaux du bas vers le haut. Deux castors se rencontrent à chaque croisement de deux canaux. Si les deux castors qui se rencontrent sont de couleurs différentes, le castor brun clair continue vers la gauche et le castor brun foncé vers la droite. Sinon, ils continuent simplement chacun dans la même direction.

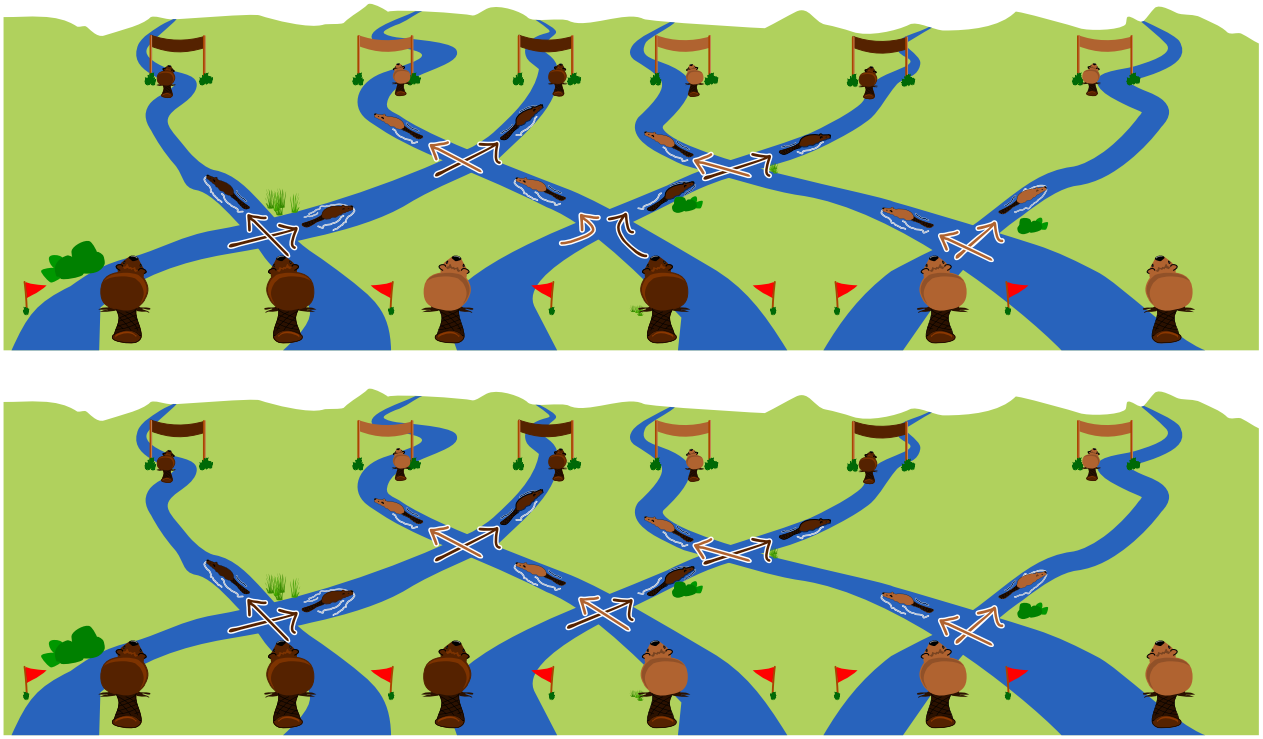


À la fin, les castors doivent être ordonnés de gauche à droite de la manière suivante : brun foncé, brun clair, brun foncé, brun clair, brun foncé, brun clair.
À quelles positions les castors brun clair et brun foncé doivent-ils commencer afin d'arriver dans le bon ordre ?



Solution

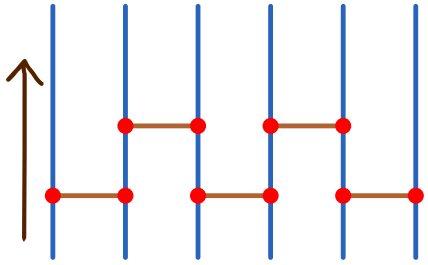
Il y a deux bonnes réponses :



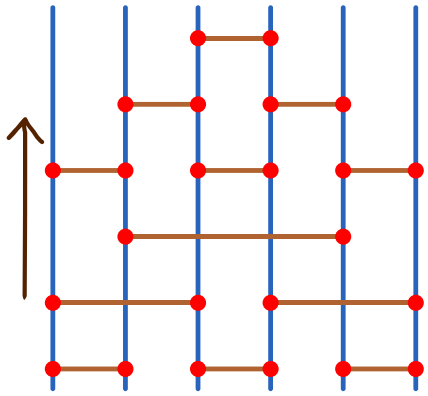
Ce sont également les deux seules réponses justes. En effet, pour que le castor arrivant tout à gauche soit brun foncé, il ne doit pas y avoir de castor brun clair qui passe par le premier croisement à gauche, car celui-ci devrait prendre le canal de gauche. Les deux positions de départ de gauche doivent donc être occupées par des castors brun foncé.
Le même raisonnement vaut pour la position d'arrivée du castor brun clair tout à droite : pour que le castor arrivant tout à droite soit brun clair, deux castors brun clair doivent se rencontrer au premier croisement depuis la droite. Les deux positions de départ de droite doivent donc être occupées par des castors brun clair.
Pour les castors du milieu, cela ne fait pas de différence si le troisième castor brun clair part à gauche et le troisième castor brun foncé part à droite ou l'inverse, car après le premier croisement, le castor brun clair continue de toute façon à gauche et le castor brun foncé à droite.

C'est de l'informatique !

Le système de canaux des castors et la règle déterminant qui nage à gauche et qui nage à droite forment une partie d'un *réseau de tri*. Dans un réseau de tri, des données se déplacent le long d'un fil (les canaux dans cet exercice), et à chaque comparateur (les croisements dans cet exercice), on teste pour décider si les données doivent être échangées ou non. On peut par exemple représenter le castor brun foncé par le chiffre 0 et le castor brun clair par le chiffre 1. Le réseau de tri ressemble alors à cela :



Dans ce diagramme, les lignes bleues représentent les canaux des castors et les points orange reliés par un trait représentent les croisements où les castors changent potentiellement de direction. Ce réseau de tri n'est pas *complet* : il ne va pas forcément disposer tous les castors brun clair à gauche et tous les castors brun foncé à droite. En ce sens, il n'effectue qu'un tri *partiel* des castors.
Le réseau de tri suivant est un réseau de tri optimal (complet et minimal) pour cet exercice : en suivant les mêmes règles, ce réseau serait toujours capable de trier tous les castors, quel que soit leur disposition initiale. On peut voir que le réseau de tri ci-dessus est intégré dans le réseau optimal (aux deuxième et troisième lignes depuis le haut) :



Les réseaux de tri sont particulièrement efficaces lorsque les comparaisons peuvent être effectuées parallèlement l'une à l'autre. De tels réseaux de tri optimaux sont difficiles à trouver pour de grands ensembles de données.
Si l'on généralise, on peut aussi se représenter le système de canaux des castors comme le système de câbles d'un réseau informatique comme Internet. Les canaux représentent alors les connexions par câbles directes entre deux routeurs, les intersections. Généralement, des tables de routage fixes permettant de diriger les données vers leur but sont programmées dans de tels routeurs.

Mots clés et sites web

Réseau de tri, réseau informatique, routeur, table de routage

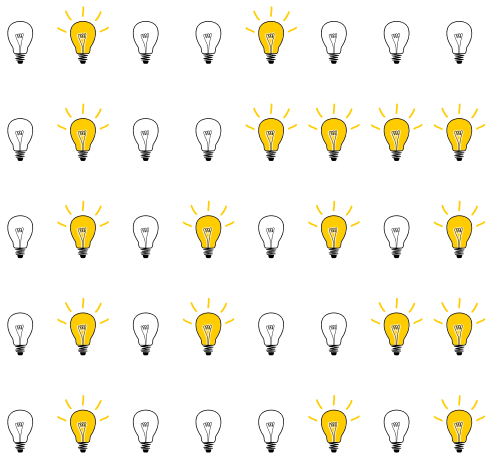
- `https://fr.wikipedia.org/wiki/R%C3%A9seau_de_tri`
- `http://www.inf.fh-flensburg.de/lang/algorithmen/sortieren/networks/optimal/optimal-sorting-networks.htm`
- `https://www.computernetworkingnotes.com/ccna-study-guide/basic-routing-concepts-and-protocols-explained.html`
- `https://fr.wikipedia.org/wiki/Routage`
- `https://fr.wikipedia.org/wiki/Table_de_routage`

5. Signaux lumineux

Sophie a huit lampes reliées par des interrupteurs et des câbles. Elle peut les utiliser pour envoyer des messages. Pour cela, elle utilise la table de codage suivante, dans laquelle `0` signifie que la lampe correspondante est éteinte ( ) et `1` que la lampe correspondante est allumée ( ) :

`A`: 01000001	`J`: 01001010	`S`: 01010011
`B`: 01000010	`K`: 01001011	`T`: 01010100
`C`: 01000011	`L`: 01001100	`U`: 01010101
`D`: 01000100	`M`: 01001101	`V`: 01010110
`E`: 01000101	`N`: 01001110	`W`: 01010111
`F`: 01000110	`O`: 01001111	`X`: 01011000
`G`: 01000111	`P`: 01010000	`Y`: 01011001
`H`: 01001000	`Q`: 01010001	`Z`: 01011010
`I`: 01001001	`R`: 01010010	

Sophie envoie à présent les signaux lumineux suivants :



Que signifient les signaux lumineux de Sophie ?

A) `HOUSE`

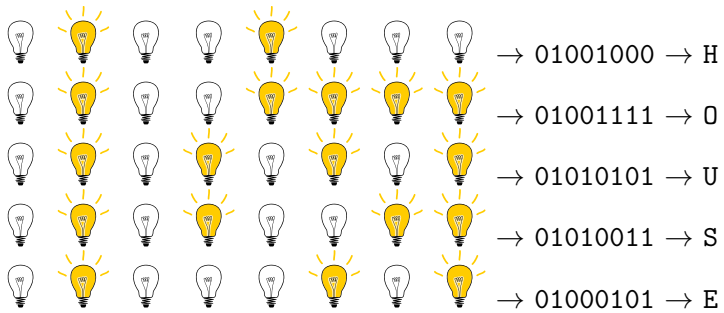
B) `HAPPY`

C) `HORSE`

D) `HONEY`

Solution

Les signaux lumineux signifient :



La bonne réponse est donc le mot A) `HOUSE`.
On peut par ailleurs trouver cette réponse très facilement : la lettre du milieu est différente dans chaque mot : A) `U`, B) `P`, C) `R` et D) `N`. Comme le troisième signal lumineux signifie `U`, seule la réponse A) peut être juste.

C'est de l'informatique !

Le codage de Sophie n'a pas été choisi au hasard. Elle utilise une partie du code appelé ASCII qui a été développé pour l'échange de messages il y a plus de 50 ans. Il est basé sur le principe du code binaire que Gottfried Leibnitz (1646–1716) a déjà décrit en 1679 et 1703 sur la base de systèmes précurseurs indiens et chinois pour la représentation de nombres et le calcul avec ces nombres. Claude Shannon (1916–2001) l'appliqua ensuite lors du développement de l'ordinateur.
Aujourd'hui, les ordinateurs utilisent une version mise à jour de l'ASCII. Comme l'ASCII ne contenait que 95 caractères imprimables (les lettres latines majuscules et minuscules, les chiffres de 0 à 9 ainsi que quelques signes de ponctuation) et que les 33 autres étaient des caractères de contrôle (pour les imprimantes par exemple), on a bientôt eu besoin d'additions pour les accents et les autres alphabets. Cela eut lieu d'abord sous la forme d'un premier code utilisant 8 bits par caractère (ANSI) et ensuite sous la forme du standard Unicode utilisé presque universellement. Les lettres de Sophie sont encore codées exactement de la même manière en UTF-8, la version de l'Unicode la plus répandue.
Le premier groupe de lettres (identique en ASCII, 8 bits et Unicode) est par ailleurs (les caractères de contrôle sont laissés vides, ␣ représente l'espace) :

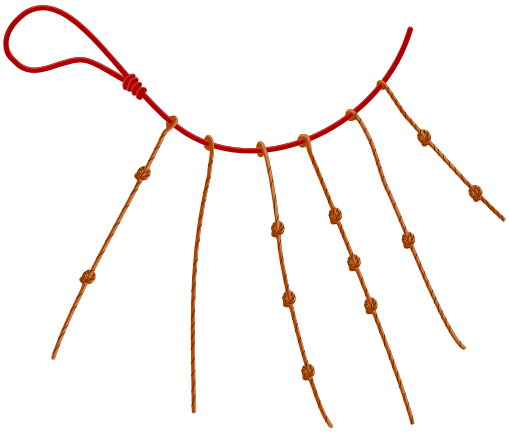
	000 ...	001 ...	010 ...	011 ...	100 ...	101 ...	110 ...	111 ...
...0000			␣	0	@	P	‘	p
...0001			!	1	A	Q	a	q
...0010			"	2	B	R	b	r
...0011			#	3	C	S	c	s
...0100			$	4	D	T	d	t
...0101			%	5	E	U	e	u
...0110			&	6	F	V	f	v
...0111			’	7	G	W	g	w
...1000			(	8	H	X	h	x
...1001			)	9	I	Y	i	y
...1010			*	:	J	Z	j	z
...1011			+	;	K	[	k	{
...1100			,	<	L	\	l	|
...1101			-	=	M	]	m	}
...1110			.	>	N	^	n	~
...1111			/	?	O	_	o	

Mots clés et sites web

ASCII, Unicode, Codage

- https://fr.wikipedia.org/wiki/American_Standard_Code_for_Information_Interchange
- https://fr.wikipedia.org/wiki/Code_binaire
- https://fr.wikipedia.org/wiki/Gottfried_Wilhelm_Leibniz
- https://fr.wikipedia.org/wiki/Claude_Shannon
- https://fr.wikipedia.org/wiki/Unicode
- https://fr.wikipedia.org/wiki/UTF-8
- https://www.unicode.org/charts/PDF/U0000.pdf

6. Quipu



Les Incas utilisaient à l'époque des nœuds pour la transmission de messages. Plusieurs cordelettes sur lesquelles des nœuds étaient noués étaient attachées à une corde. Ces assemblages de cordelettes appelés « quipus » étaient grands et difficiles à fabriquer.
Imagine qu'il faut développer une version simplifiée des quipus. Les conditions sont :

- Il y a toujours le même nombre de cordelettes attachées à la corde.
- Les cordelettes ne diffèrent que par le nombre de nœuds.
- Une cordelette a 0, 1, 2 ou 3 nœuds.
- L'ordre des cordelettes est déterminé à l'aide d'un nœud sur la corde.
- Il doit pouvoir y avoir 30 quipus discernables pour différents messages.

Quel est le nombre minimum de cordelettes de la version simplifiée des quipus dans ces conditions ?

A) 2

B) 3

C) 4

D) 5

E) 8

F) 10

Solution

La réponse B) 3 est correcte.
Chaque cordelette peut enregistrer une des 4 valeurs différentes (0, 1, 2 ou 3). Avec deux cordelettes, on aurait $4 \cdot 4 = 16$ combinaisons possibles, avec trois cordelettes $4 \cdot 4 \cdot 4 = 64$ combinaisons possibles, et ainsi de suite. Trois cordelettes sont donc suffisantes, plus de cordelettes seraient en contradiction avec la condition stipulant qu'il faut attacher le moins de cordelettes possible à la corde. Comme l'ordre des valeurs est déterminé par le nœud sur la corde, on ne doit pas s'inquiéter du fait que l'on pourrait lire la corde dans un sens ou dans l'autre.

C'est de l'informatique !

Les *quipus* étaient en effet utilisés par les Incas en Amérique du Sud. Des quipus gris étaient utilisés pour la comptabilité et la perception d'impôts. On suppose que jusqu'à 95 syllabes différentes pouvaient être codées à l'aide de cordelettes colorées, permettant une correspondance écrite. Au contraire de la version simplifiée de cet exercice, différentes sortes de nœuds et dans certains cas des sous-cordelettes attachées aux cordelettes étaient utilisées
L'exemple de cet exercice est une version simplifiée. Comme l'ordre est fixé par le nœud sur la corde, les valeurs individuelles (0, 1, 2 ou 3) créent une *notation positionnelle*, dans ce cas un système en base 4. Les notations positionnelles sont très répandues : en règle générale, le système décimal (en base 10) est utilisé, les ordinateurs utilisent le *système binaire* (en base 2). Au temps des premiers ordinateurs, il y a eu des essais de construction d'ordinateurs basés sur le *système ternaire* (en base 3, représentés dans ce cas par -1, 0 et $+1$). En utilisant un système en base b avec n positions, on peut représenter exactement b^n valeurs différentes. Un octet (8 bits qui peuvent valoir chacun 0 ou 1) peut de cette manière enregistrer $2^8 = 256$ valeurs différentes (de 0 à 255), le quipu de cet exercice $4^3 = 64$ valeurs différentes
Les Incas n'auraient eu besoin que d'une seule cordelette pour enregistrer les valeurs de 1 à 30. Ils utilisaient également un système décimal comme nous pour l'écriture des nombres, simplement en utilisant différents nœuds sur une cordelette. La position des unités aurait été codée entre autres avec des demi-nœuds et la position des dizaines avec un demi-nœud auquel un nombre de tours correspondants était ajouté. Cependant, ils auraient eu besoin d'au moins 4 nœuds et de plusieurs sortes de nœuds.

Mots clés et sites web

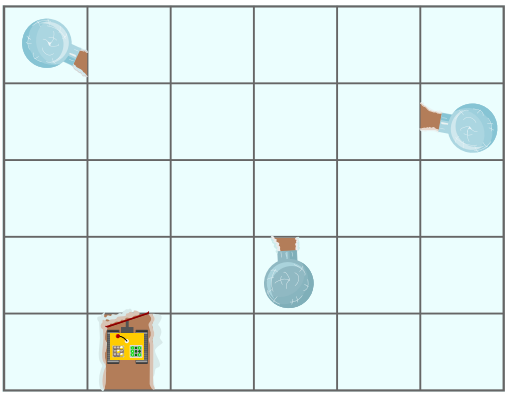
Quipu, notation positionnelle

- `https://fr.wikipedia.org/wiki/Quipu`
- `https://fr.wikipedia.org/wiki/N%C5%93ud_double`
- `https://en.wikipedia.org/wiki/Stopper_knot`
- `https://fr.wikipedia.org/wiki/Notation_positionnelle`
- `https://fr.wikipedia.org/wiki/Ordinateur_ternaire`

7. Tempête de neige

Il y a des congères partout après une forte tempête de neige et les habitants des trois iglous sont isolés. Les habitants peuvent dégager des chemins à l'aide de leur chasse-neige télécommandé. Cela fonctionne ainsi :

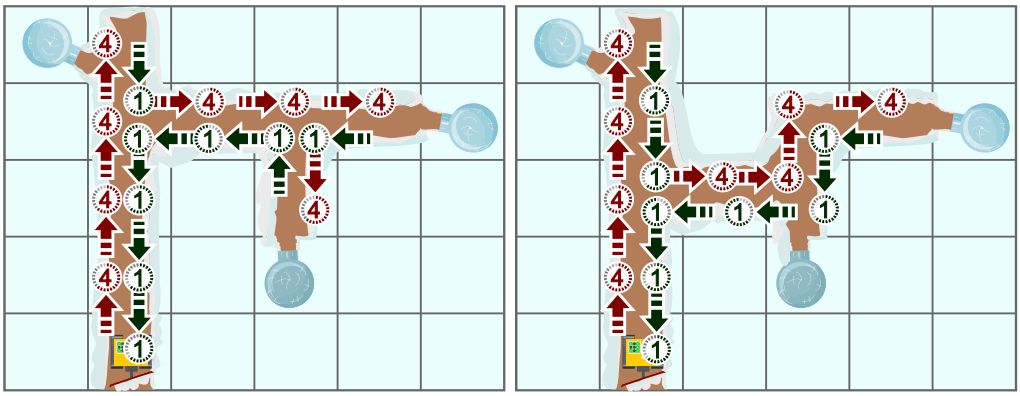
- Le chasse-neige a besoin de 4 minutes pour passer d'une case à une case voisine en la dégageant.
- Le chasse-neige a besoin d'une minute pour passer d'une case déjà déneigée à une case voisine.
- Les case voisines ne sont que les cases qui sont situées directement en dessus, en dessous, à gauche ou à droite d'une case sur la carte. La chasse-neige ne peut donc pas rouler en diagonale.
- Dès que la case devant l'entrée d'un iglou est dégagée, les habitants de l'iglou peuvent en dégager l'entrée avec une pelle et ne sont plus isolés.



Dans le cas idéal, de combien de minutes le chasse-neige a-t-il besoin pour libérer tous les iglous de leur isolement et retourner à sa case de départ ?

Solution

La bonne réponse est 40. Les illustrations suivantes montrent les deux chemins optimaux pour le chasse-neige :



Pourquoi est-ce que ça ne va pas plus rapidement ? Pour atteindre l'iglou en haut à gauche, il faut dégager quatre cases. Cela prend 16 minutes. Pour atteindre l'iglou à droite, trois cases de plus doivent être dégagées. Cela prend 12 minutes de plus. Pour atteindre l'iglou du bas, une case doit encore être dégagée, car il faut soit dégager un petit chemin jusqu'au chemin transversal, soit ajouter un virage au chemin transversal. Cela prend encore 4 minutes. Pour que le chasse-neige retourne à la case départ, il doit encore parcourir quatre cases vers le bas et trois cases vers la gauche. Cela dure encore 7 minutes. Le détour par le petit chemin ou le virage dans le chemin transversal dure encore une minute supplémentaire. Il a donc besoin de 40 minutes en tout.
Si le chasse-neige dégageait le chemin plus vite, ce serait éventuellement plus efficace s'il rentrait de l'iglou du bas en passant par la case enneigée à gauche en la dégageant. Mais il a besoin de 4 minutes pour la dégager et d'une minute pour passer sur la case déjà déneigée, donc de 5 minutes, alors que le détour par les cases déjà déneigées ne lui prend que 4 minutes.

C'est de l'informatique !

Il s'agit dans cet exercice de trouver un réseau de chemins qui relie tous les endroits (les iglous et la case de départ) à un coût minimal (le temps dont le chasse-neige a besoin). De tels réseaux ne sont pas forcément composés des chemins les plus courts entre tous les endroits, mais les coûts nécessaires à la réalisation du réseau doivent être aussi bas que possible. On appelle de tels réseaux des *arbres de Steiner*. Ils sont utilisés, par exemple, pour la fabrication des cartes mères des ordinateurs ou la construction de réseaux ferroviaires peu exploités pour le transport de marchandises. La recherche d'arbres de Steiner est un problème d'optimisation difficile et qui prend beaucoup de temps en informatique, ce qui fait que l'on utilise souvent des algorithmes qui trouvent des solutions suffisamment bonnes, mais pas forcément la meilleure solution.
Dans cet exercice, les coûts sont calculés de manière spéciale, car il n'y a pas seulement un coût fixe pour l'élaboration d'un chemin (les 4 minutes pour dégager une case), mais également un coût pour le déplacement de la machine sur les cases déneigées. Cet exercice est donc une généralisation du problème de l'arbre de Steiner.

Mots clés et sites web

Problème de l'arbre de Steiner
— `https://fr.wikipedia.org/wiki/Probl%C3%A8me_de_l%27arbre_de_Steiner`

8. Quel bonheur que les arbres !

Sergio a écrit une chanson qui décrit comment plusieurs choses différentes peuvent se former sur un arbre. Voici un couplet :

> Quel bonheur que les arbres !
> Sur un arbre poussent des feuilles,
> Sur un arbre poussent des fleurs,
> Les fleurs donnent des fruits,
> Avec des feuilles et des fleurs, je peux tresser des couronnes.

C'était important pour Sergio de n'utiliser après le premier vers que des objets déjà mentionnés auparavant.
Lequel des couplets suivants est faux d'après Sergio ?

A) Quel bonheur que les arbres !
Sur un arbre poussent des fleurs,
Sur un arbre poussent des feuilles,
Avec des feuilles et des fleurs, je peux tresser des couronnes,
Les fleurs donnent des fruits.

B) Quel bonheur que les arbres !
Sur un arbre poussent des fleurs,
Sur un arbre poussent des feuilles,
Les fleurs donnent des fruits,
Avec des feuilles et des fleurs, je peux tresser des couronnes.

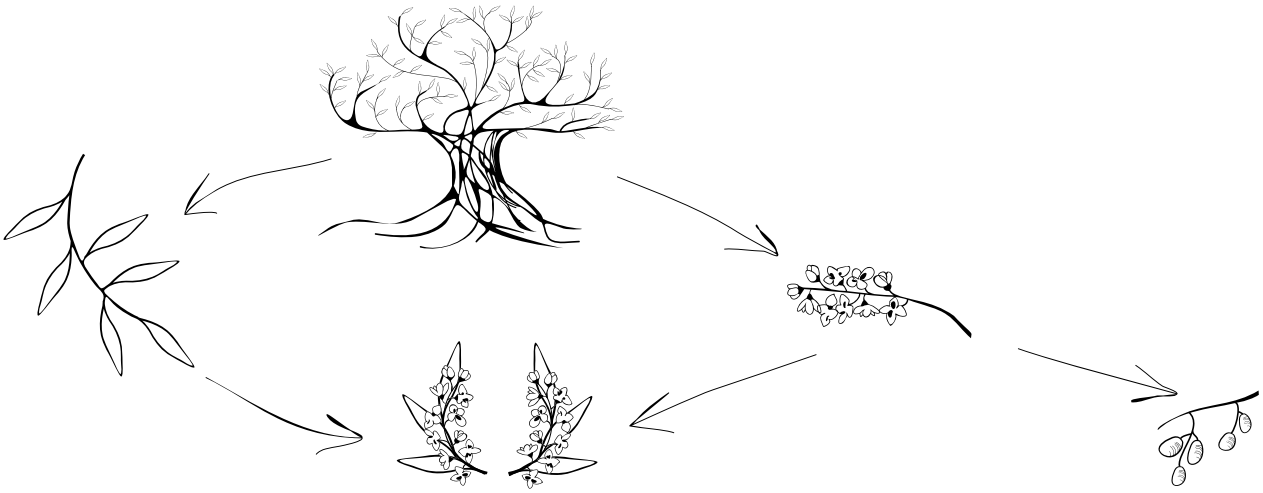
C) Quel bonheur que les arbres !
Sur un arbre poussent des feuilles,
Les fleurs donnent des fruits,
Sur un arbre poussent des fleurs,
Avec des feuilles et des fleurs, je peux tresser des couronnes.

D) Quel bonheur que les arbres !
Sur un arbre poussent des fleurs,
Les fleurs donnent des fruits,
Sur un arbre poussent des feuilles,
Avec des feuilles et des fleurs, je peux tresser des couronnes.

E) Quel bonheur que les arbres !
Sur un arbre poussent des feuilles,
Sur un arbre poussent des fleurs,
Avec des feuilles et des fleurs, je peux tresser des couronnes,
Les fleurs donnent des fruits.

Solution

On peut représenter les dépendances des objets « arbre », « feuilles », « fleurs », « couronnes » et « fruits » à l'aide d'un graphe sur lequel une flèche signifie qu'un objet est nécessaire à la formation de l'autre :



D'après ce graphe, les fleurs doivent être mentionnées avant les fruits, et les feuilles et les fleurs avant les couronnes.
La séquence des objets dans les différentes réponses est la suivante :

A) Arbre, fleurs, feuilles, couronnes, fruits

B) Arbre, fleurs, feuilles, fruits, couronnes

C) Arbre, feuilles, *fruits*, *fleurs*, couronnes

D) Arbre, fleurs, fruits, feuilles, couronnes

E) Arbre, feuilles, fleurs, couronnes, fruits

Dans la réponse C), les fruits sont mentionnés avant les fleurs (en italique plus haut), ce qui est une contradiction car il faut des fleurs pour faire des fruits. Tous les autres vers respectent les conditions.

C'est de l'informatique !

En 1974, le musicien italien Sergio Endrigo (1933–2005) a écrit la chanson pour enfants « Ci vuole un fiore » d'après un texte de Gianni Rodari (1920–1980). Dans cette chanson, il chante que l'on a d'abord besoin de bois pour avoir une table, d'un arbre pour avoir du bois, d'une graine pour avoir un arbre, d'un fruit pour avoir une graine et d'une fleur pour avoir un fruit. Il décrit aussi que pour avoir une fleur, on a également besoin d'une fleur en passant par une branche, un arbre, une forêt, une montagne et la terre. Il finit par dire qu'on a en fin de compte besoin d'une fleur pour tout.
La précédence d'un objet sur un autre peut être décrite à l'aide d'un *graphe orienté*. Un tel graphe est dessiné dans l'explication de la réponse. C'est un *graphe orienté acyclique* qui décrit un *ensemble de séquences admissibles*. Si l'on veut un nœud (l'un des objets), on doit déjà avoir tous les objets pointant vers lui. Le même chose est vraie pour ces objets-là, ce qui fait que l'on doit remonter de manière *récursive* jusqu'à ce que l'on arrive à un objet sur lequel rien ne pointe. On peut utiliser cet objet comme objet de départ
On ne peut d'ailleurs pas décrire la chanson de Sergio Endrigo à l'aide d'un graphe orienté acyclique. Dans la deuxième partie de la chanson décrite plus haut, il chante que l'on a en fin de compte besoin d'une fleur pour avoir une fleur. C'est en contradiction avec le fait que le graphe doit être acyclique

et ne peut donc pas contenir de circuit. Avec cette rupture de logique, il rend sa déclaration encore plus claire : « Ci vuole un fiore » — « Nous avons besoin d'une fleur » !

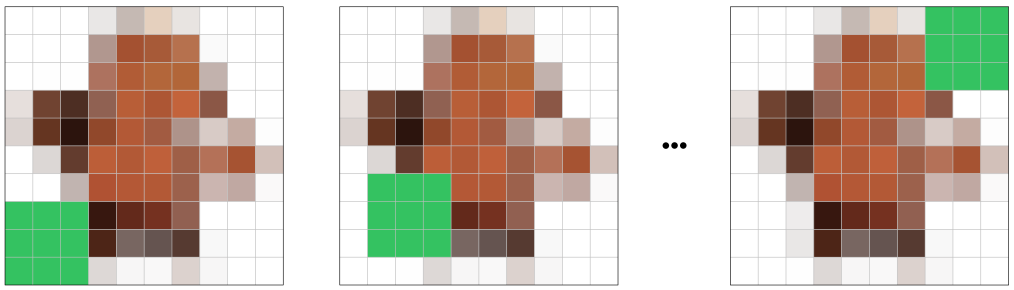
Mots clés et sites web

Graphe orienté acyclique, tri topologique

- `https://www.filastrocche.it/contenuti/ci-vuole-un-fiore/`
- `https://www.youtube.com/watch?v=9ht4tIot8XY`
- `https://en.wikipedia.org/wiki/Precedence_graph`
- `https://fr.wikipedia.org/wiki/Tri_topologique`

9. Compression vidéo

Les vidéos occupent beaucoup d'espace de stockage. Pourtant, deux images fixes consécutives se ressemblent souvent beaucoup.
La vidéo suivante a une taille de 10×10 points. Le carré vert dans le coin en bas à gauche fait 3×3 points. Il se déplace d'image fixe à image fixe d'un point vers la droite et d'un point vers le haut à chaque image jusqu'à ce qu'il arrive dans le coin en haut à droite.



Pour économiser de l'espace de stockage, à partir de la deuxième image, seuls les points qui ont changé sont enregistrés.
Combien de points doivent être enregistrés pour toute la vidéo ?

A) 100
B) 135
C) 140
D) 170
E) 180
F) 700
G) 800
H) 1000

Solution

Les images fixes individuelles ressemblent à cela lorsque l'on marque les points ayant changé :

On remarque tout d'abord que la première image contient $10 \cdot 10 = 100$ points. Pour chacune des images suivantes, seuls les points qui ont changé doivent être enregistrés. Il s'agit des cinq points en bas à gauche du carré qui sont remplacés par les points de l'arrière-plan et des cinq points en haut à droite sur le carré qui représentent la nouvelle position du carré. 10 points changent donc par image fixe. Le carré a besoin de 7 images fixes supplémentaires pour se déplacer d'en bas à gauche vers en haut à droite, il faut donc ajouter $10 \cdot 7 = 70$ points pour les points qui changent aux 100 points de la première image. La bonne réponse est donc D) 170 points.

C'est de l'informatique !

Comme décrit dans l'exercice, la compression vidéo joue un rôle important de nos jours. La méthode décrite ici n'est qu'une des approches permettant de compresser des vidéos. Une autre approche consiste à enlever certaines informations qu'une personne ne perçoit habituellement pas. Le format d'image JPEG utilise de telles relations. Sur des images compressées spécialement fortement, on peut le reconnaître à la formation de blocs, car la différence entre les couleurs du bloc a été interprétée à tort comme imperceptible. Une autre possibilité est de réduire la dimension de l'espace de couleur. C'est sur ces idées qu'est basé le standard MPEG. Comme dans cet exercice, il fait la différence entre différents types d'images fixes. Un type d'images fixes (appelées images «intra») représente une image fixe complète (similairement à notre première image fixe). Un autre type d'images fixes

est basé sur la différence avec l'image fixe précédente (images « P », comme nos autres images fixes) ou même sur la différence avec l'image fixe suivante en plus (images « D », ne sont pas utilisées dans cet exercice). Pour minimiser l'utilisation de mémoire tampon et pour pouvoir « retrouver le fil » en cas d'erreurs de transmission, des images intra sont insérées à intervalles réguliers. Sur des vidéos fortement compressées, on peut reconnaître les images P et D lorsque qu'un arrière-plan sombre « saute » tout à coup alors que la scène n'a changé que lentement pendant un certain temps.
Le besoin en espace de stockage n'est pas vraiment aussi petit que suggéré dans l'exercice : en plus des valeurs des couleurs, la position du pixel qui a changé doit aussi être enregistrée. Cela multiplie peut-être par deux l'espace de stockage nécessaire à un pixel changé. Même ainsi, 240 unités de stockage comparées à 800 unités de stockage représentent une économie de place impressionnante, d'autant plus que la méthode décrite dans l'exercice est, contrairement au MPEG, une méthode sans perte !

Mots clés et sites web

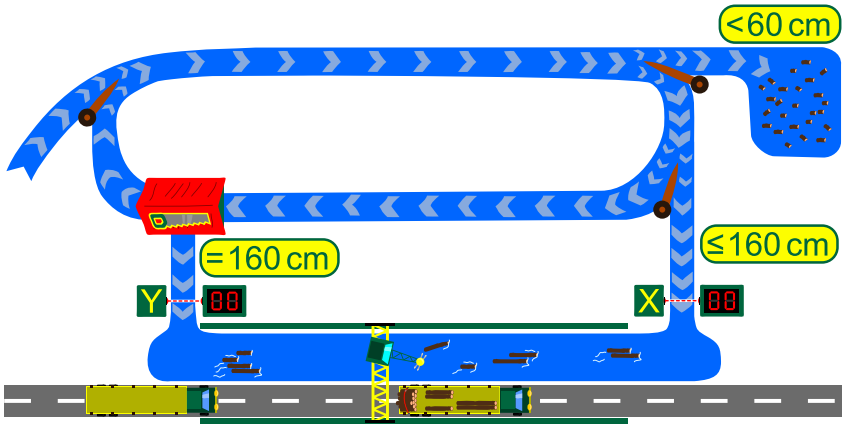
Compression vidéo

- `https://fr.wikipedia.org/wiki/Compression_vid%C3%A9o`
- `https://fr.wikipedia.org/wiki/JPEG`
- `https://fr.wikipedia.org/wiki/MPEG-1`

10. Scierie

Dans une scierie, des troncs d'arbres sont raccourcis pour avoir une longueur entre 60 cm et 160 cm avant d'être chargés sur des camions. À l'intérieur de la scierie, le transport des troncs d'arbres se fait par des canaux. Il y existe les stations de travail suivantes :

- En haut à gauche, les troncs d'arbres sont livrés.
- En haut à droite, les troncs plus courts que 60 cm sont mis à part (<60 cm).
- Au milieu à droite, tous les troncs de 160 cm ou moins sont chargés sur des camions (≤160 cm). Ceux-ci sont comptés à hauteur du détecteur X.
- Au milieu à gauche, un tronçon de 160 cm est coupé à l'extrémité de chaque tronc. Ces troncs sont chargés sur des camions (=160 cm) et comptés à hauteur du détecteur Y. Le reste du tronc est remis en circulation.



Trois troncs d'arbres de 60 cm, 140 cm et 360 cm sont livrés et traités dans la scierie.
Combien de troncs d'arbres sont comptés à hauteur du détecteur X et combien à hauteur du détecteur Y ?

A) Détecteur X : aucun tronc, détecteur Y : 4 troncs

B) Détecteur X : 1 tronc, détecteur Y : 3 troncs

C) Détecteur X : 2 troncs, détecteur Y : 2 troncs

D) Détecteur X : 3 troncs, détecteur Y : 1 tronc

Solution

Le tronc de 60 cm n'est pas mis à part en haut à droite, car il n'est pas plus court que 60 cm. Il est par contre chargé sur un camion au milieu à droite, car il mesure 160 cm ou moins. Un tronc a donc déjà été compté à hauteur du détecteur.
Le tronc de 140 cm n'est pas mis à part en haut à droite non plus, car il n'est pas plus court que 60 cm. Par contre, comme il mesure 160 cm ou moins, il est chargé sur un camion au milieu à droite. Un deuxième tronc a ainsi été compté à hauteur du détecteur X.
Le tronc de 360 cm n'est pas mis à part en haut à droite non plus, car il n'est pas plus court que 60 cm. Par contre, vu qu'il fait plus de 160 cm, il est dirigé vers la scie lorsqu'il passe au milieu à droite. La scie en coupe un tronçon de 160 cm qui est chargé sur un camion. Ainsi, un tronc a été compté à hauteur du détecteur Y. Le reste du tronc mesurant 200 cm est remis en circulation. Le tronc mesurant à présent 200 cm n'est pas mis à part en haut à droite non plus, car il n'est pas plus court que 60 cm. Vu qu'il fait plus de 160 cm, il est également dirigé vers la scie lorsqu'il passe au milieu à droite. La scie en coupe un deuxième tronçon de 160 cm qui est chargé sur un camion. Ainsi, un deuxième tronc est compté à hauteur du détecteur Y. Le reste du tronc mesurant 40 cm est remis en circulation. Le tronc mesurant à présent 40 cm est mis à part en haut à droite.
La réponse correcte est donc C) Détecteur X : 2 troncs, détecteur Y : 2 troncs.

C'est de l'informatique !

La seule caractéristique importante des troncs en circulation est leur longueur. On peut donc considérer la scierie comme un programme dans lequel des nombres entiers sont entrés et qui fait certaines mesures. Dans ce cas, on peut voir la scierie comme un *programme réactif* : pendant que le nombre est traité par le programme, les mesures changent avec le temps. La programmation réactive joue un rôle majeur dans les tableurs. Les valeurs calculées à l'aide de formules réagissent aux changements de valeurs dans d'autres cases.
Concrètement, plusieurs opérations réactives sont utilisées dans cet exercice : en haut à gauche, deux trains de données convergent (*merge* en anglais), en haut à droite, les données sont filtrées (*filter* en anglais), en bas à droite aussi, et en bas à gauche les données sont transformées (*transform* en anglais). Les deux détecteurs font des mesures (*scan* en anglais).
L'analyse de processus dynamiques comme dans cet exercice est centrale à l'informatique. Bien avant que le terme *pensée computationnelle* (*computational thinking* en anglais) n'apparaisse et ne soit rendu mondialement connu par Jeanette Wing en 2006, des termes tels que *procedural thinking* ou *algorithmic thinking* (*pensée procédurale* ou *pensée algorithmique*) étaient utilisés comme paradigmes propres à l'informatique.

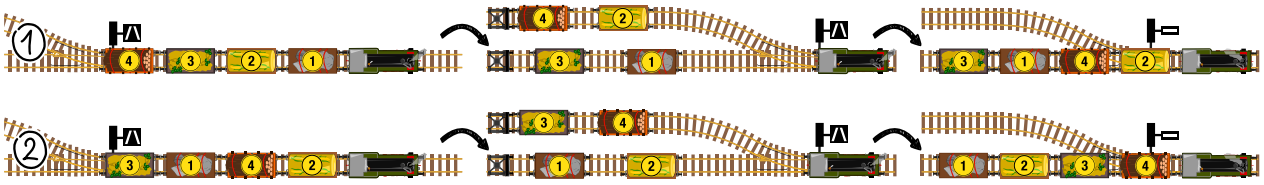
Mots clés et sites web

Reaktive Programmierung

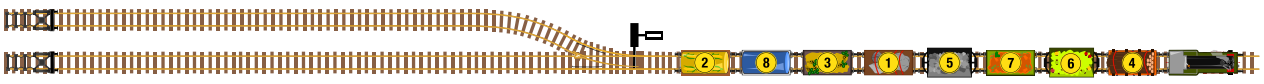
- `https://fr.wikipedia.org/wiki/Programmation_r%C3%A9active`
- `https://fr.wikipedia.org/wiki/Pens%C3%A9e_computationnelle`
- `https://www.cs.cmu.edu/~15110-s13/Wing06-ct.pdf`

11. Gare de triage

Un train de marchandises doit amener des wagons de marchandises individuels sur des voies de raccordement le long de la ligne principale. Pour économiser du temps et éviter de manœuvrer sur la ligne principale, les wagons de marchandises doivent être classés d'après leur numéro de façon à ce que le wagon portant le numéro 1  soit tout à gauche.
Dans la gare de triage, il y a une butte sur laquelle les wagons de marchandises sont refoulés vers le bas de gauche à droite. Sur la butte, on décide pour chaque wagon sur laquelle des deux voies de garage il est orienté. Ensuite, la locomotive tracte les wagons hors des voies de garage : d'abord tous ceux se trouvant sur une voie, ensuite tous ceux se trouvant sur l'autre voie de garage. On considère ces actions comme une étape de tri.
Par exemple, lorsque quatre wagons de marchandises doivent être triés, deux étapes de tri suffisent (étape ① et étape ②) :



Ce n'est pas possible de trier les wagons en une seule étape de tri.
Si les wagons de marchandises sont dans l'ordre 2 – 8 – 3 – 1 – 5 – 7 – 6 – 4*, de combien d'étapes de tri a-t-on besoin au minimum pour trier le train de marchandises ?*



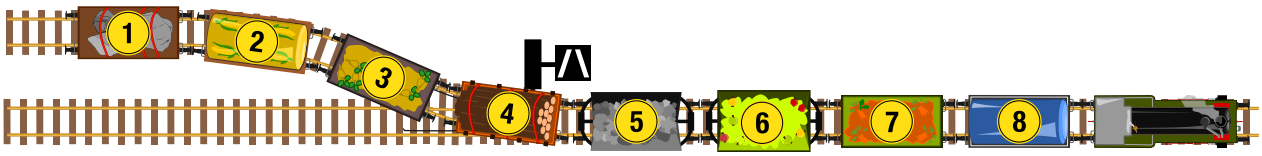
A) 3
B) 4
C) 5
D) 6
E) 7
F) 8

Solution

La bonne réponse est que l'on a besoin de A) 3 étapes de tri.
Bien sûr que l'on peut utiliser plusieurs méthodes pour trier le train de marchandises, mais l'une des meilleures méthodes est de commencer par refouler les wagons de marchandises 1, 3, 5 et 7 sur la voie du haut et les wagons 2, 4, 6, et 8 sur la voie du bas, puis de tracter d'abord les wagons de la voie du bas puis ceux de la voie du haut :

Ainsi, pour chaque paire (1 et 2, 3 et 4, 5 et 6, 7 et 8) de wagons de marchandises, le wagon portant le plus petit numéro se trouve toujours à gauche de celui portant le plus grand numéro.
Ensuite, c'est raisonnable de refouler les wagons de marchandises 1, 2, 5 et 6 sur la voie du haut et les wagons 3, 4, 7 et 8 sur la voie du bas, puis de tracter d'abord les wagons de la voie du bas puis ceux de celle du haut :

On n'a ainsi pas changé l'ordre des paires obtenues auparavant, car chaque paire a été refoulée sur la même voie. On a en plus mis les wagons 1 à 4 et 5 à 8 dans le bon ordre relatif les uns aux autres, mais les deux groupes ne sont pas encore mélangés.
Finalement, il ne faut plus que refouler les wagons de marchandises 1 à 4 sur la voie du haut et les wagons 5 à 8 sur la voie du bas, puis de tracter d'abord les wagons de marchandise de la voie du bas, puis de la voie du haut :



L'ordre dans chaque groupe n'a pas été modifié, car tous les wagons de marchandises du groupe 1 à 4 ont été refoulés sur une voie et tous les wagons du groupe 5 à 8 sur l'autre voie. Maintenant, les deux groupes sont composés de wagons de marchandises ordonnés et tous les wagons de l'un des deux groupes ont des numéros plus petits que tous les wagons de l'autre groupe.
On ne peut pas trier les huit wagons plus rapidement. Un preuve complète de cela serait trop longue pour cet exercice, mais l'idée de base est la suivante : lors d'une étape de tri, on peut changer l'ordre d'un sous-ensemble relativement aux autres sous-ensembles, mais pas l'ordre dans les sous-ensembles mêmes. On ne peut donc trier que deux wagons de marchandises dans un ordre défavorable lors de la première étape de tri. Chaque étape de tri suivante double le nombre de wagons ordonnés défavorablement que l'on peut trier. Les huit wagons de marchandises de cet exercices sont choisis de manière à être ordonnés défavorablement, donc deux étapes de tri ne suffisent pas.

C'est de l'informatique !

Les cheminots du monde entier doivent résoudre de tels problèmes quotidiennement, car le tri de wagons de marchandises est une tâche qui demande beaucoup de temps et de travail : chaque fois,

les wagons doivent être connectés et déconnectés, ce qui est encore un travail manuel. Cela prend du temps et bloque la ligne principale, surtout lorsque plusieurs wagons de marchandises doivent être sécurisés et déconnectés. C'est pour cela que des cheminots ont très tôt développé de grandes gares de triage avec beaucoup de voies de garage. En Suisse, il y a des gares de triage à Muttenz près de Bâle, à Buchs (Saint-Gall), entre Spreitenbach et Dietikon près de Zurich, à Denges près de Lausanne et à Chiasso. Dans cet exercice, la gare de triage n'a que deux voies de garage, un défi pour les grands trains de marchandises mais une situation typique pour les lignes ferroviaires secondaires, en particulier pour les lignes de chemin de fer à voie étroite qui n'ont pas de connexion directe aux grandes entreprises ferroviaires.
L'informatique peut être très utile pour trier efficacement les trains de marchandise. Dans ce cas, le principe consistant à résoudre encore et encore le même problème simplifie fortement l'exercice : une méthode connue en informatique sous le nom de « diviser pour régner » (*divide & conquer* en anglais). Dans ce cas, on trie d'abord les paires de wagons de marchandises, puis des groupes de quatre wagons, puis le groupe de huit wagons.
Les voies de garage pour les wagons de marchandises fonctionnent comme le type de données abstrait *pile*, qui est beaucoup utilisé en informatique. Les seules opérations autorisées sont : *dépiler* (enlever l'élément du dessus, *pop* en anglais) et *empiler* (ajouter un élément sur la pile, *push* en anglais). Parfois, on peut aussi *voir l'élément de tête* (*top* en anglais) et *vérifier si la pile est vide* (*empty* en anglais).

Mots clés et sites web

Diviser pour régner (Divide & Conquer), pile
- `https://fr.wikipedia.org/wiki/Gare_de_triage`
- `https://fr.wikipedia.org/wiki/Diviser_pour_r%C3%A9gner_(informatique)`
- `https://fr.wikipedia.org/wiki/Pile_(informatique)`

12. Jeu de billes

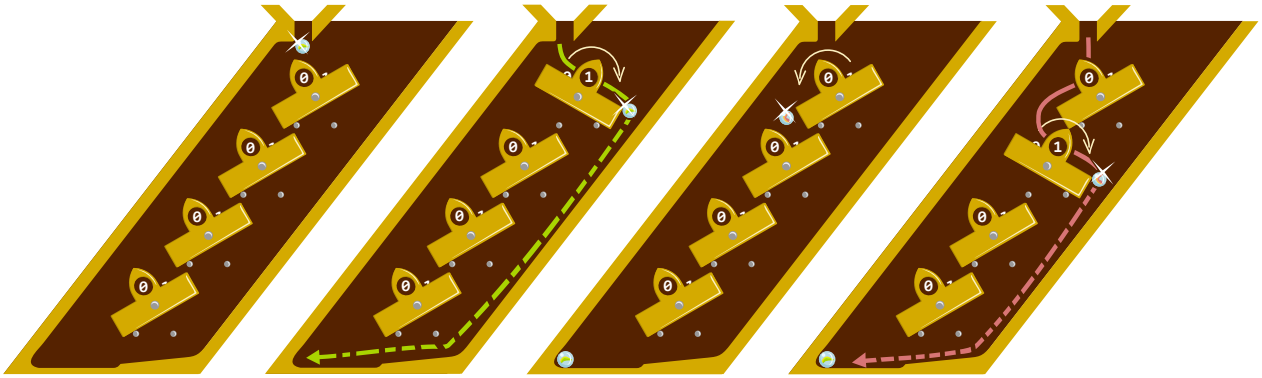
Un jeu de billes contient quatre balançoires qui peuvent s'incliner de deux manières :

— Si la balançoire est penchée vers la gauche, son inclinaison est 0.

— Si la balançoire est penchée vers la droite, son inclinaison est 1.

Lorsqu'une bille atterrit sur une balançoire, celle-ci change son inclinaison et la balle roule vers le bas.

Lorsque deux billes sont mises en jeu, les balançoires se penchent de façon à ce que l'inclinaison de la balançoire la plus haute soit 1 après la première bille, et qu'après la deuxième bille, l'inclinaison de la balançoire du haut soit à nouveau 0 et que celle de la deuxième balançoire soit 1 :



À la fin, les inclinaisons des balançoires (d'en bas à gauche vers en haut à droite) sont 0, 0, 1 et 0. *Toutes les balançoires sont remises à l'inclinaison* 0. *Quelle sera l'inclinaison finale des quatre balançoires si dix billes sont mises en jeu ?*

A) 0, 0, 0 et 0
B) 1, 0, 0 et 0
C) 0, 1, 0 et 0
D) 0, 0, 1 et 0
E) 0, 0, 0 et 1
F) 1, 1, 0 et 0
G) 1, 0, 1 et 0
H) 1, 0, 0 et 1
I) 0, 1, 1 et 0
J) 0, 1, 0 et 1
K) 0, 0, 1 et 1
L) 1, 1, 1 et 0
M) 1, 1, 0 et 1
N) 1, 0, 1 et 1
O) 0, 1, 1 et 1
P) 1, 1, 1 et 1

Solution

La balançoire la plus haute change d'inclinaison à chaque bille. Elle se retrouve donc en position 0 quand, comme dans ce cas, un nombre pair de billes (10) sont mises en jeu. Le dernier chiffre est donc 0.
La deuxième balançoire depuis le haut ne change d'inclinaison que si la balançoire la plus haute se trouve en position 1 au départ de la bille. Ce n'est le cas qu'une fois sur deux. Elle change donc cinq fois d'inclinaison et termine avec l'inclinaison 1. L'avant-dernier chiffre est donc 1.
La troisième balançoire depuis le haut ne change d'inclinaison que si la deuxième balançoire a l'inclinaison 1 au départ de la bille et qu'une bille y atterrit, c'est-à-dire si la balançoire la plus haute a également l'inclinaison 1 au départ. La troisième balançoire ne change donc d'inclinaison qu'une fois sur quatre. C'est la cas lors du passage de la quatrième et huitième billes, donc deux fois en tout. Elle termine donc avec l'inclinaison 0 et le deuxième chiffre est 0.
La balançoire du bas ne change d'inclinaison que si les trois balançoires précédentes ont l'inclinaison 1 au départ de la bille et qu'une bille peut donc y atterrir. Ce n'est le cas que lors du passage de la huitième bille. La dernière balançoire ne change donc qu'une fois d'inclinaison et le premier chiffre est 1.
L'inclinaison des balançoires est donc 1, 0, 1 et 0.

C'est de l'informatique !

Les balançoires du jeu de billes représentent un interrupteur électronique capable d'alterner entre deux états. De tels interrupteurs sont des éléments de bases d'appareils électroniques, celui de l'exercice est une variante de *bascule*.
Les balançoires fonctionnent ensemble comme un *compteur* (binaire). Les bascules y sont reliées de manière à former des *demi-additionneurs*. Un tel demi-additionneur prend comme valeur d'entrée l'état enregistré de la bascule ainsi qu'une impulsion. Sa sortie est un nouvel état à enregistrer ainsi qu'un reste.

État enregistré	Impulsion	État à enregistrer	Reste
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Le circuit doit assurer que le changement des états enregistrés se passe dans le bon ordre et que les valeurs des états ne changent pas de manière automatique après un changement.
Les états des balançoires représentent donc finalement un nombre binaire qui augmente de 1 au passage de chaque bille. Dix billes donnent donc, si on inclut l'état de départ, les 11 premiers nombres binaires : 0000 → 0001 → 0010 → 0011 → 0100 → 0101 → 0110 → 0111 → 1000 → 1001 → 1010.

Mots clés et sites web

Système binaire, compteur, demi-additionneur, bascule

- `https://fr.wikipedia.org/wiki/Compteur`
- `https://www.youtube.com/watch?v=zELAfmp3fXY`
- `https://fr.wikipedia.org/wiki/Bascule_(circuit_logique)`
- `https://fr.wikipedia.org/wiki/Additionneur#Demi-additionneur`
- `https://fr.wikipedia.org/wiki/Syst%C3%A8me_binaire`

13. Quatre poissons

En informatique, le mode de fonctionnement d'opérateurs tels que + ou * dépend en partie de quels types de données sont impliqués. La table suivante montre différentes combinaisons typiques utilisées dans des expressions :

Général	Exemple
Nombre + Nombre → Nombre (additionner)	`2+3` → `5`
Nombre + Texte → Erreur	`2+"3"` → erreur
Texte + Nombre → Erreur	`"2"+3` → erreur
Texte + Texte → Texte (mettre bout à bout)	`"2"+"3"` → `"23"`
Nombre * Nombre → Nombre (multiplier)	`2*3` → `6`
Nombre * Texte → Text (mettre le texte nombre fois bout à bout)	`2*"3"` → `"33"`
Texte * Nombre → Text (mettre le texte nombre fois bout à bout)	`"2"*3` → `"222"`
Texte * Texte → Erreur	`"2"*"3"` → erreur

Lorsque le résultat est «erreur», cela signifie qu'aucun mode de fonctionnement n'est défini pour cette combinaison. S'il y a une erreur dans une expression, le résultat est également «erreur».
Les opérateurs sont combinés suivant l'ordre habituel des opérations, les multiplications et divisions avant les additions et soustractions. L'opération * est donc effectuée avant l'opération +. Cet ordre peut être modifié à l'aide de parenthèses. Les opérations entre parenthèses sont effectuées de l'intérieur vers l'extérieur.
Quelle expression, parmi les expressions suivantes, génère cette ligne de texte ?

```
"...><(((°>.....><(((°>.....><(((°>.....><(((°>..."
```

A) `(3*"."+"><"+3*"("+"°>"+3*".")*"2"*2`

B) `(3*"."+"><"+3*"("+"°>")*2*2+3*"."`

C) `(3*"."+"><"+"3"*"("+"°>"+3*".")*2*2`

D) `(3*"."+"><"+3*"("+"°>"+3*".")*2*2`

Solution

La bonne réponse est D) `(3*"."+"><"+3*"("+"°>"+3*".")*2*2`.
Les cinq termes dans la parenthèse génèrent chacun le texte suivant :

- `3*"."` → `"..."`
- `"><"` → `"><"`
- `3*"("` → `"((("`
- `"°>"` → `"°>"`
- `3*"."` → `"..."`

Mis bout à bout, cela donne `"...><(((°>..."`.
Les deux multiplications suivantes `*2*2` causent chacune la mise bout à bout du texte, ce qui donne d'abord `"...><(((°>.....><(((°>..."` et finalement `"...><(((°>.....><(((°>.....><(((°>.....><(((°>..."`.
La réponse A) génère une erreur, car l'expression entre parenthèses est un texte (`"...><(((°>..."`) qui doit être multiplié par un autre texte (`"2"`).
La réponse B) génère le texte `"...><(((°>...><(((°>...><(((°>...><(((°>..."`. C'est très proche du résultat souhaité, mais le nombre de points entre les poissons ne correspond pas au résultat souhaité.
La réponse C) génère une erreur, car l'expression `"3"*"("` a déjà « erreur » comme résultat.

C'est de l'informatique !

On parle de *surcharge* lorsque plusieurs modes de fonctionnement sont assignés à un opérateur (ou à un sous-programme) en fonction des opérandes (ou des paramètres). Cela arrive le plus souvent dans le cas d'opérateurs qui sont également utilisés fréquemment en dehors du langage de programmation.
La forme de surcharge des opérateurs `+` et `*` décrite plus haut est présente dans beaucoup de langages de programmation. Aussi pratique la surcharge d'opérateur soit-elle, elle comporte le risque de rendre le code du programme moins lisible.
Le court programme suivant génère la même chaîne de caractères que celle de l'exercice (les poissons), mais le type de chaque variable et le mode de fonctionnement de chaque opérateur en fonction du type de variable est difficile à déterminer et vérifier :

```
a ← 3
b ← "."
c ← "><"
d ← "("
e ← "°>"
f ← 2
Sortie (a*b+c+a*d+e+a*b)*f*f
```

La surcharge de termes avec différentes significations n'est d'ailleurs pas un phénomène propre à l'informatique. Dans le contexte de la langue, cela s'appelle *polysémie*. Le mot « figure », par exemple, a un sens différent si l'on parle de géométrie, de ballet, de littérature ou d'un visage.
Le poisson généré ici est un exemple classique d'art ASCII. C'est une variante du « hareng rouge », qui est parfois envoyé aux trolls sur Internet.

Mots clés et sites web

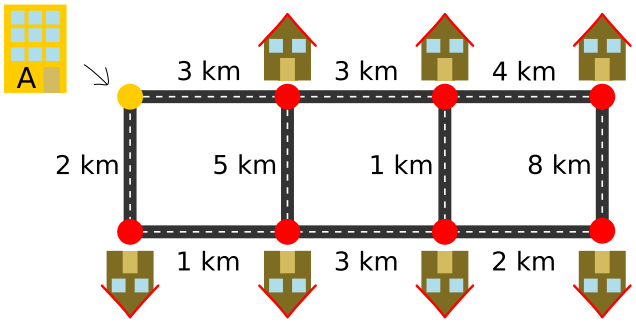
Surcharger, art ASCII

- `https://fr.wikipedia.org/wiki/D%C3%A9finition_d%27op%C3%A9rateur`

— `https://fr.wikipedia.org/wiki/Polys%C3%A9mie`
— `https://fr.wikipedia.org/wiki/Art_ASCII`

14. Job de vacances

Ton job de vacances consiste à livrer des paquets à vélo. Tu commences à l'endroit A et livres un paquet à chacun des sept autres endroits. Ta tournée se termine au dernier endroit et ton employeur vient te chercher ainsi que ton vélo.
Pour rester en forme, tu aimerais faire le plus long trajet possible avec les paquets. La longueur de chaque chemin est inscrite sur la carte ci-dessous. Ton employeur te laisse décider quels chemins prendre, mais tu ne veux pas passer deux fois au même endroit.



Quelle est la longueur du plus long chemin qui ne passe pas deux fois au même endroit ?

A) 22 km

B) 23 km

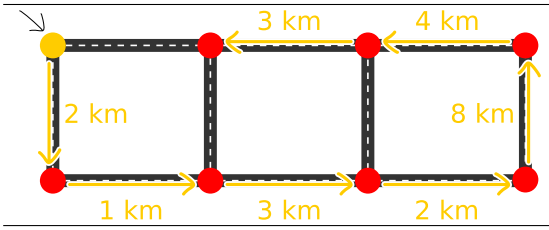
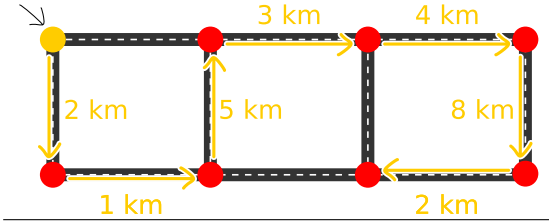
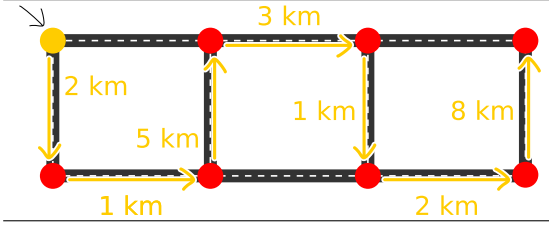
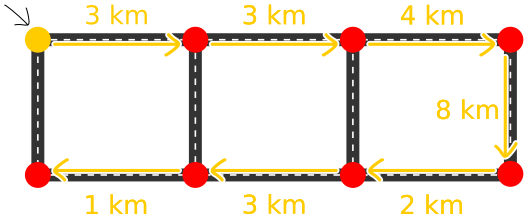
C) 24 km

D) 25 km

E) 26 km

Solution

Il n'y a en tout que quatre chemins possibles qui ne passent pas deux fois au même endroit. Leurs longueurs sont :

Chemin	Longueur
	2 km + 1 km + 3 km + 2 km + 8 km + 4 km + 3 km = 23 km
	2 km + 1 km + 5 km + 3 km + 4 km + 8 km + 2 km = 25 km
	2 km + 1 km + 5 km + 3 km + 1 km + 2 km + 8 km = 22 km
	3 km + 3 km + 4 km + 8 km + 2 km + 3 km + 1 km = 24 km

Il ne peut pas y avoir d'autre chemin : dans tous les autres cas, il faut passer deux fois par l'un des endroits afin d'arriver à un autre.
C'est donc clair que D) 25 km est la bonne réponse.

C'est de l'informatique !

Dans cet exercice, il s'agit de trouver un chemin passant exactement une fois par chaque endroit sur une carte. On appelle un tel chemin un *chemin hamiltonien*. Comme on peut interpréter une carte comme un *graphe*, les endroits comme des *nœuds* et les chemins comme des *arêtes*, la recherche d'un chemin hamiltonien relève de la théorie des graphes. La recherche d'un chemin hamiltonien dans un graphe quelconque, ou le fait de déterminer si un tel chemin existe, est un problème NP-complet qui ne peut donc pas être résolu de manière efficace par ordinateur.
Dans ce cas, on ne recherche pas n'importe quel chemin hamiltonien, mais celui qui a comme somme des arêtes la valeur la plus haute parmi tous les chemins hamiltoniens possibles. Cela rend le problème encore plus difficile à résoudre.
Les approches trouvant une solution acceptable mais suboptimale dans d'autres cas ne peuvent pas résoudre ce problème. Une approche classique serait un *algorithme glouton*, qui sélectionne d'abord les valeurs les plus élevées, avec lequel on commencerait donc par le chemin de 3 km. Cependant, cette approche ne trouverait que le chemin de 24 km. Et si l'on continue de manière si « gloutonne »

en prenant ensuite le chemin de 5 km, on élimine complètement la possibilité de ne pas devoir passer deux fois au même endroit.

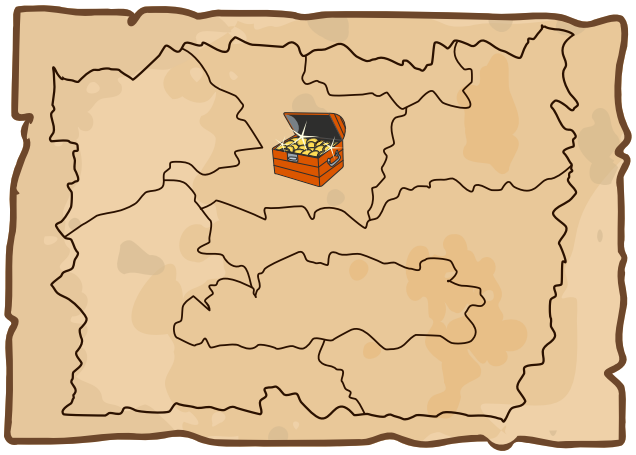
Mots clés et sites web

Chemin hamiltonien, NP-complet

- `https://fr.wikipedia.org/wiki/Graphe_hamiltonien`
- `https://fr.wikipedia.org/wiki/Probl%C3%A8me_NP-complet`
- `https://fr.wikipedia.org/wiki/Algorithme_glouton`

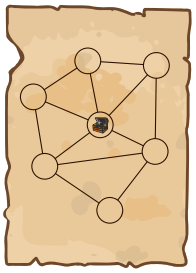
15. Carte au trésor

Le roi des castors règne sur sept provinces dont les frontières sont représentées sur la carte ci-dessous. Il a caché son trésor dans l'une des provinces :

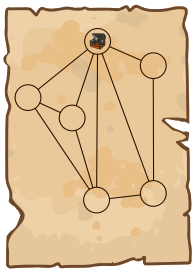


Le roi a fait dessiner une carte au trésor sur laquelle les provinces sont représentées par des cercles. Il a mis la province où se trouve le trésor en évidence. Deux cercles sont reliés si les provinces correspondantes ont une frontière en commun. Pour empêcher des voleurs de trouver le trésor, le roi a fait dessiner quatre fausses cartes au trésor supplémentaires.
Quelle carte au trésor est la bonne ?

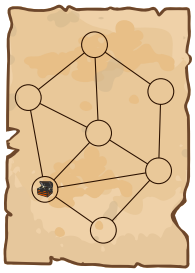
A)



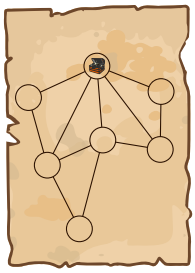
B)



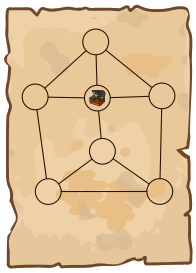
C)



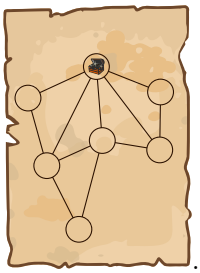
D)



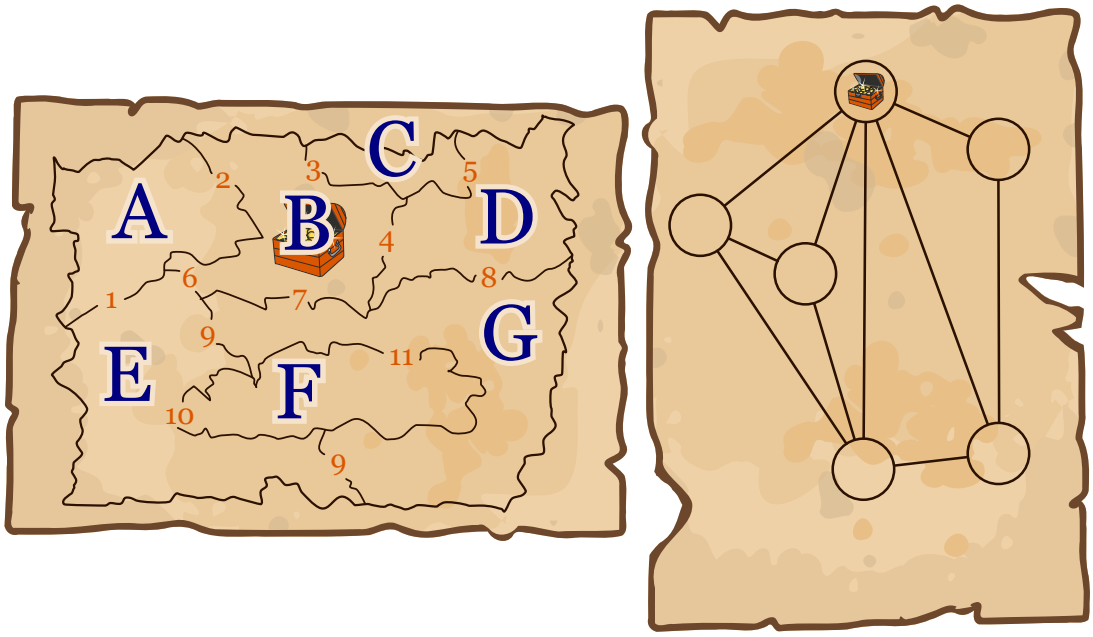
E)



Solution

La bonne réponse est D)  .

Les provinces sont annotées des lettres A, B, C, D, E, F et G dans la représentation suivante. Les frontières entre les provinces sont annotées des nombres de 1 à 11. Comme il y a deux frontières communes entre les provinces E et G mais que la condition ne demande qu'une seule frontière commune, les deux frontières sont annotées du chiffre 9. On est rapidement convaincu que la carte D) est la bonne carte à l'aide de cette représentation.

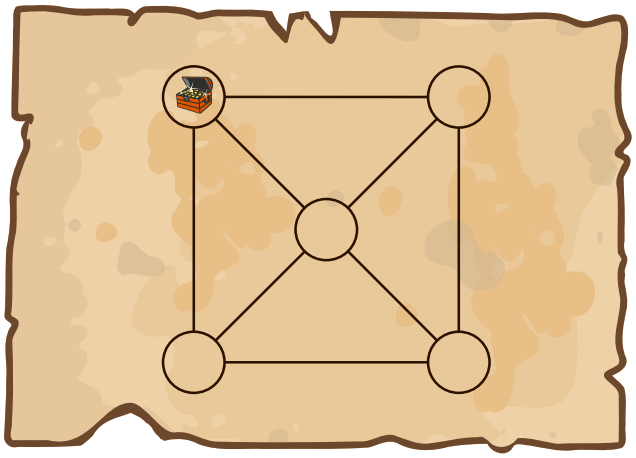


La réponse A) est fausse : les trois provinces A, C et F n'ont de frontières communes qu'avec deux autres provinces. Il devrait donc y avoir trois cercles desquels deux lignes partent sur la carte. Il n'y a pourtant qu'un seul cercle dont dont deux lignes partent.
La réponse B) ne peut pas être juste, car elle ne contient que six cercles alors qu'il y a sept provinces.
La réponse C) ne peut pas être juste non plus : la province où est le trésor est voisine des cinq provinces A, C, D, E et G, il devrait donc y avoir cinq lignes partant du cercle mis en évidence. Il n'y en a pourtant que quatre.
La réponse E) ressemble le plus à la carte des provinces. Les frontières n'y sont pourtant pas représentées correctement, il manque une frontière à la province B où est le trésor, par exemple.

C'est de l'informatique !

Dans cet exercice, il s'agit de représenter une carte à l'aide d'un *graphe*. Un tel graphe est une *abstraction* de la carte (comme la carte qui est elle-même déjà une abstraction de la réalité). Comme dans toutes les abstractions, les informations sans importance sont perdues. Dans cet exercice, il s'agit des positions géographiques des provinces les unes par rapport aux autres : même si les provinces

B et D se trouvent à la même hauteur sur la carte, elles se trouvent à des hauteurs différentes sur le graphe. Par contre, les informations essentielles sont conservées dans l'abstraction, dans le cas de cet exercice, il s'agit de savoir quelles provinces sont voisines. Cependant, même une abstraction correcte comporte le risque de perdre une information apparemment sans importance : le graphe suivant d'un autre état est symétrique et ne permet donc pas de déterminer où le trésor est caché exactement :



Un graphe est composé de *nœuds* (les cercles dans cet exercice) et d'*arêtes* (les lignes dans cet exercice). Dans cet exercice, les arêtes du graphe ne peuvent que relier deux nœuds. On ne peut pas non plus relier deux nœuds avec plusieurs arêtes, ce qui aurait été envisageable entre les provinces E et G.
Les graphes sont souvent utilisés en informatique pour abstraire des informations. Dans de nombreux cas, cette étape d'abstraction est déjà presque la solution du problème : comme les graphes sont beaucoup étudiés et très bien définis en informatique, il suffit souvent de ramener un problème à un problème de graphe typique et d'y appliquer la solution que l'on connaît déjà pour ce problème.

Mots clés et sites web

Graphe, théorie des graphes

- `https://fr.wikipedia.org/wiki/Graphe_(math%C3%A9matiques_discr%C3%A8tes)`
- `https://fr.wikipedia.org/wiki/Abstraction_(philosophie)`

A. Auteurs des exercices

Tony René Andersen
Michelle Barnett
Michael Barot
Wilfried Baumann
Jan Berki
Linda Bergsveinsdóttir
Laura Braun
Špela Cerar
Mony Chanroath
Sébastien Combéfis
Kris Coolsaet
Christian Datzko
Maria Suyana Datzko
Susanne Datzko
Guillaume de Moffarts
Gerald Futschek
Arnheiður Guðmundsdóttir
Martin Guggisberg
Juraj Hromkovič
Alisher Ikramov
Takeharu Ishizuka
M. Faiz Ahmad Ismail
Anna Laura John
Mile Jovanov
Ungyeol Jung
Ilya Kaysin
Jihye Kim
Vaidotas Kinčius
Mária Kiss
Bohdan Kudrenko
Regula Lacher
Anh Vinh Lê
Greg Lee
Judith Lin
Lynn Liu
Violetta Lonati
Vũ Văn Luân
Mattia Monga
Samart Moodleah
Madhavan Mukund
Tom Naughton
Tomohiro Nishida
Assylkan Omashev
Margot Phillipps
Zsuzsa Pluhár
Sergei Pozdniakov
Nol Premasathian
J.P. Pretti
Milan Rajković
Chris Roffey
Eljakim Schrijvers
Humberto Sermeno
Daigo Shirai
Jacqueline Staub
Nikolaos Stratis
Bundit Thanasopon
Peter Tomcsányi
Nicole Trachsler
Jiří Vaníček
Troy Vasiga
Michael Weigend

B. Sponsoring : Concours 2019

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.oxocard.ch/
OXOcard
OXON

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

https://educatec.ch/
educaTEC

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

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

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

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

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

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

https://www.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.