



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

Exercices et solutions 2023

Années HarmoS 5/6

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

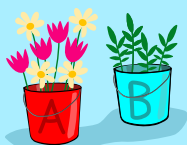
Éditeurs :

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

010100110101011001001001
01000010010110101010011
010100110100100101000101
001011010101001101010011
01001001010010010010001

SS!E

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





Ont collaboré au Castor Informatique 2023

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

Cheffe de projet : Nora A. Escherle

Nous adressons nos remerciements pour le travail de développement des exercices du concours à :
Juraj Hromkovič, Angélica Herrera Loyo, Regula Lacher, Manuel Wettstein : ETH Zurich,
Ausbildunges- und Beratungszentrum für Informatikunterricht

Tobias Berner : Pädagogische Hochschule Zürich

Christian Datzko : Wirtschaftsgymnasium und Wirtschaftsmittelschule, Basel

Fabian Frei : CISPA - Helmholtz-Zentrum für Informationssicherheit

Sebastian Knüsli : Gymnasium Kirschgarten, Basel

Le choix des exercices a été fait en collaboration avec les organisateur de Bebras en Allemagne, Autriche, Hongrie, Slovaquie et Lituanie. Nous remercions en particulier :

Valentina Dagienė, Vaidotas Kinčius : Bebras.org, Lituanie

Wolfgang Pohl, Jakob Schilke : Bundesweite Informatikwettbewerbe (BWINF), Allemagne

Hannes Endreß : Materna Information & Communications SE, Allemagne

Ulrich Kiesmüller : Simon-Marius-Gymnasium Gunzenhausen, Allemagne

Kirsten Schlüter : Bayerisches Staatsministerium für Unterricht und Kultus, Allemagne

Margareta Schlüter : Universität Tübingen, Allemagne

Jacqueline Staub : Universität Trier, Allemagne

Michael Weigend : WWU Münster, Allemagne

Wilfried Baumann, Liam Baumann, Josefine Hiebler : Österreichische Computer Gesellschaft, Autriche

Gerald Futschek : Technische Universität Wien, Autriche

Zsuzsa Pluhár : ELTE Informatikai Kar, Hongrie

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

Eljakim Schrijvers, Justina Dauksaite, Arjan Huijsers, Dave Oostendorp, Alieke Stijf, Kyra Willekes : cuttle.org, Pays-Bas

Chris Roffey : UK Bebras Administrator, Royaume-Uni

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

Hanspeter Erni : Direction, école secondaire de Rickenbach

Gabriel Thullen : Collège des Colombières, Versoix

Nous remercions les personnes suivantes pour l'organisation et la réalisation de la finale suisse :

Dennis Komm, Hans-Joachim Bückenhauer, Jan Lichensteiger, Moritz Stocker : ETH Zurich,
Ausbildunges- und Beratungszentrum für Informatikunterricht

Pour la correction des épreuves :

Fiona Binder, Joel Birrer, Marlene Bötschi, Danny Camenisch, Gianluca Danieletto, Alexander Frey,
Sven Grübel, Laure Guerrini, Charlotte Knierim, Richard Královič, Yanik Künzi, Kenli Lao, Sandro



Marchon, Zoé Meier, Dario Näpfer, Kai Zürcher

Pour la traduction française des épreuves :

Jan Schönbächler : Lycée-Collège de l'Abbaye de St-Maurice

Christoph Frei : Chragokyberneticks (Logo Castor Informatique Suisse)

Andrea Leu, Maggie Winter, Lena Frölich : Senarclens Leu + Partner AG

Des remerciements particuliers sont dûs pour leur grand soutien à Juraj Hromkovič, Dennis Komm, Gabriel Parriaux et la Fondation Hasler. Sans eux, ce concours n'existerait pas.

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

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



INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Le Castor Informatique 2023 a été réalisé par la Société Suisse pour l'Informatique dans l'Enseignement (SSIE) et soutenu de manière déterminante par la Fondation Hasler. Les sponsors du concours sont l'Office de l'économie et du travail du canton de Zurich et l'UBS.

Cette brochure a été produite le 10 janvier 2024 avec le système de composition de documents \LaTeX . Nous remercions Christian Datzko pour le développement et maintien de la structure de génération des 36 versions de cette brochure (selon les langues et les degrés). La structure actuelle a été mise en place de manière similaire à la structure précédente, qui a été développée conjointement avec Ivo Blöchliger dès 2014. Nous remercions aussi Jean-Philippe Pellet pour le développement de la série d'outils `bebras`, qui est utilisée depuis 2020 pour la conversion des documents source depuis les formats Markdown et YAML.

Tous les liens dans les tâches ci-après ont été vérifiés le 1^{er} décembre 2023.



Les exercices sont protégés par une licence Creative Commons Paternité – Pas d'Utilisation Commerciale – Partage dans les Mêmes Conditions 4.0 International. Les auteur·e·s sont cité·e·s en p. 50.



Préambule

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

Le Castor Informatique est le partenaire suisse du concours « Bebras International Contest on Informatics and Computer Fluency » (<https://www.bebas.org/>), initié en Lituanie.

Le concours a été organisé pour la première fois en Suisse en 2010. Le Petit Castor (années HarmoS 5 et 6) a été organisé pour la première fois en 2012.

Le Castor Informatique vise à motiver les élèves à apprendre l'informatique. Il souhaite lever les réticences et susciter l'intérêt quant à l'enseignement de l'informatique à l'école. Le concours ne suppose aucun prérequis quant à l'utilisation des ordinateurs, sauf de savoir naviguer sur Internet, car le concours s'effectue en ligne. Pour répondre, il faut structurer sa pensée, faire preuve de logique mais aussi d'imagination. Les exercices sont expressément conçus pour développer un intérêt durable pour l'informatique, au-delà de la durée du concours.

Le concours Castor Informatique 2023 a été fait pour cinq tranches d'âge, basées sur ces années scolaires :

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

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

Chaque réponse correcte donnait des points, chaque réponse fautive réduisait le total des points. Ne pas répondre à une question n'avait aucune incidence sur le nombre de points. Le nombre de points de chaque exercice était fixé en fonction du degré de difficulté :

	Facile	Moyen	Difficile
Réponse correcte	6 points	9 points	12 points
Réponse fautive	-2 points	-3 points	-4 points

Utilisé au niveau international, ce système de distribution des points est conçu pour limiter le succès en cas de réponses données au hasard.



Chaque participant·e obtenait initialement 45 points (ou 27 pour la tranche d'âge «Petit Castor», et 36 pour les années HarmoS 7 et 8).

Le nombre de points maximal était ainsi de 180 (ou 108 pour la tranche d'âge «Petit Castor», et 144 pour les années HarmoS 7 et 8). Le nombre de points minimal était zéro.

Les réponses de nombreux exercices étaient affichées dans un ordre établi au hasard. Certains exercices ont été traités par plusieurs tranches d'âge (en étant classés différemment dans les niveaux de difficulté).

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

Pour de plus amples informations :

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

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



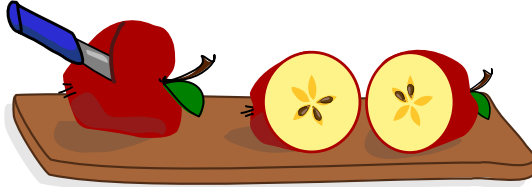
Table des matières

Ont collaboré au Castor Informatique 2023	i
Préambule	iii
Table des matières	v
1. Moitiés de pommes	1
2. Terre – eau	5
3. Tri par chapeau	9
4. Visite au zoo	13
5. Parapluie	17
6. Fleuriste	21
7. L'arbre magique	25
8. La maison de Karla	29
9. Graines de carottes	31
10. Riccas	35
11. Jardin potager	39
12. Fontaine	43
13. Ogham	47
A. Auteur-e-s des exercices	50
B. Partenaires académiques	51
C. Sponsoring	52
D. Offres supplémentaires	53



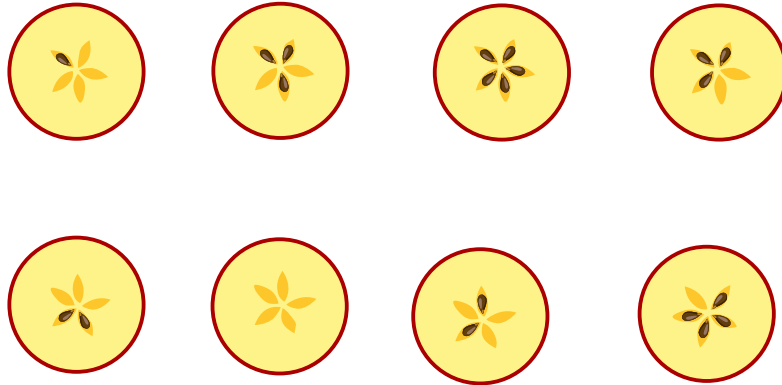
1. Moitiés de pommes

On peut partager des pommes en deux, ce qui donne la moitié du bas et la moitié du haut. Certains pépins de pomme restent dans la moitié du haut, les autres dans la moitié du bas. On peut voir que les moitiés vont ensemble en regardant les pépins et les trous :



On fait cela pour Noël en Tchéquie. Gala partage quatre pommes. Elle arrange les moitiés du haut et celles du bas en deux rangées.

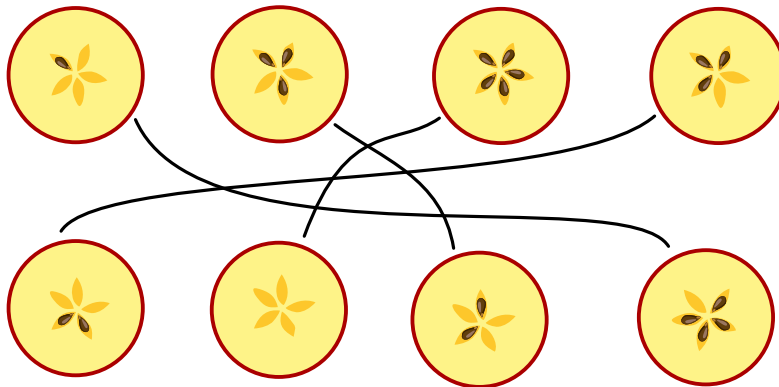
Quelles moitiés de pommes vont ensemble ? Relie les moitiés de pommes.



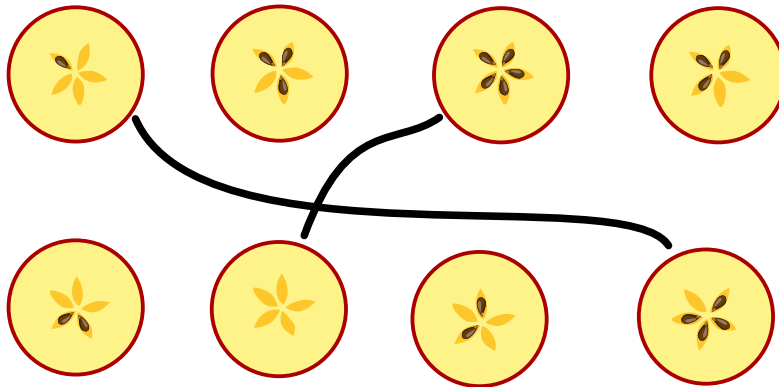


Solution

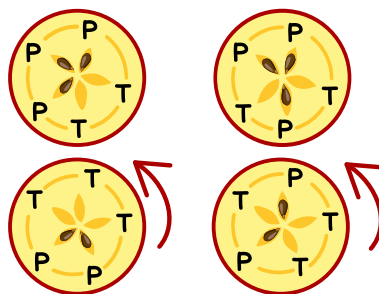
Voici la bonne réponse :



Chaque pomme a cinq pépins. Deux moitiés de pomme allant ensemble doivent donc avoir cinq pépins en tout. Ces moitiés peuvent donc facilement être reliées :



Les quatre moitiés restantes ne sont pas si faciles à relier. Les deux moitiés du haut ont trois pépins chacune, les deux moitiés du bas deux pépins chacune. Il faut donc regarder le motif formé par les pépins plus exactement, car si deux moitiés vont ensemble, leurs motifs vont aussi ensemble. Pour vérifier cela, il peut être nécessaire de tourner les moitiés de pomme. Les moitiés de pommes peuvent ensuite être reliées comme sur l'image ci-dessous : à gauche, la moitié du haut a trois pépins côte à côte, puis deux trous (P-P-P-T-T), la moitié du bas a trois trous côte à côte, puis deux pépins (T-T-T-P-P) : les moitiés vont ensemble. Les moitiés de droite vont également ensemble : la moitié du haut a le motif P-T-P-T-P (en commençant en haut au centre), et celle du bas T-P-T-P-T.





C'est de l'informatique !

On a vu dans l'explication de la réponse que si deux moitiés de pomme vont ensemble, il n'y a pas que le nombre de pépins qui correspond, mais aussi l'ordre des pépins et des trous. Il faut donc également considérer cet ordre lors de l'attribution des moitiés de pomme. Il ne suffit pas de savoir combien de pépins a chaque moitié.

Des questions semblables sont présentes dans des problèmes qui doivent être résolus à l'aide d'ordinateurs. Les informaticiennes et informaticiens doivent réfléchir à la manière dont les informations que le programme doit considérer sont décrites. On essaie en général de faire aussi simple que possible : les programmes simples ont moins de risque de contenir des erreurs. Pour le problème des moitiés de pomme, il semblait d'abord suffisant de décrire les moitiés uniquement à l'aide du nombre de pépins. Ensuite, on a vu que ce n'était pas suffisant dans toutes les situations. Il faut donc pouvoir décrire un ordre afin de pouvoir décrire les moitiés de pomme dans un programme informatique. C'est possible à l'aide de la structure de données appelée *liste*, par exemple, qui existe dans la plupart des langages de programmation.

Mots clés et sites web

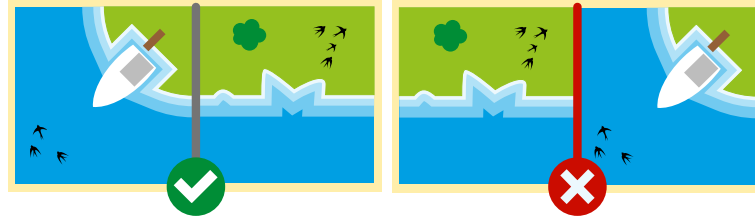
- Ordre
- Liste : [https://fr.wikipedia.org/wiki/Liste_\(informatique\)](https://fr.wikipedia.org/wiki/Liste_(informatique))





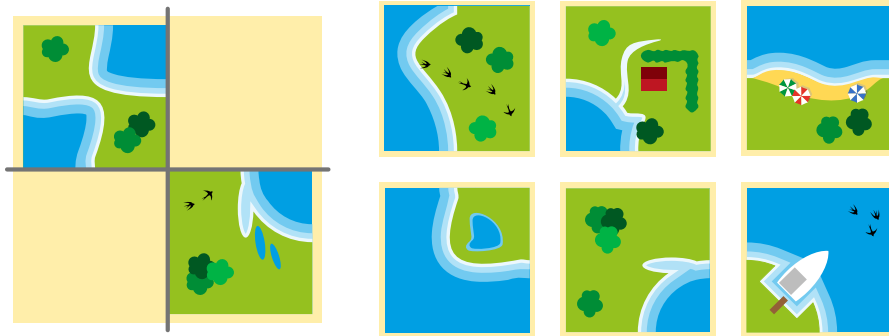
2. Terre – eau

Edu a un nouveau jeu. Il est fait de cartes avec des zones de terre et d'eau. Edu peut former des paysages avec ses cartes. Les cartes doivent *aller ensemble* : terre contre terre, eau contre eau.



Edu pose deux cartes et laisse deux trous.

Quelles cartes vont dans les trous ? Tu n'as pas le droit de tourner les cartes.





Solution

Voici la bonne réponse :



Les deux cartes vont dans les trous : il y a partout de la terre contre de la terre et de l'eau contre de l'eau. Seules ces deux cartes vont dans les trous parmi les six cartes.

Si l'on avait le droit de tourner les cartes, d'autres cartes iraient aussi dans les trous.

C'est de l'informatique !

Regardons les cartes d'Edu plus en détail. Chaque carte peut être divisée en quatre parties. Les bords extérieurs de ces parties montrent de la terre (🌿) ou de l'eau (🌊).














Il n'y a donc que deux sortes de parties, car les bords extérieurs montrent soit de l'eau (🌊), soit de la terre (🌿).



Deux cartes ne vont ensemble que si leurs parties voisines sont les mêmes. Pour trois parties de chaque trou, nous pouvons donc indiquer quelle sorte de partie est nécessaire. La quatrième partie peut être de l'eau ou de la terre, donc nous indiquons ★.



De cette manière, nous créons un motif pour chaque trou. Les cartes devant aller dans ces trous doivent correspondre à ces motifs : pour  et , la partie de la carte doit également être  et  ; pour , la partie de la carte peut être  ou .

Nous avons découvert une propriété des cartes. Nous avons utilisé cette propriété pour remplacer les cartes par un arrangement des symboles  et . Nous avons ainsi réduit largement la quantité d'information présente dans les images. Nous nous concentrons sur les informations qui sont nécessaires à la résolution de cet exercice. Les informaticiens et informaticiennes se référeraient à l'arrangement des symboles sur les images. Le fait de réduire les images aux types de partie  et  crée un modèle des cartes manquantes. L'*abstraction* est nécessaire à la *modélisation*, et l'abstraction réduit la quantité d'information. Les ordinateurs doivent travailler avec des modèles de la réalité. Il faut être attentif à ne pas perdre certaines propriétés importantes de la réalité lors de la création de tels modèles.

Mots clés et sites web

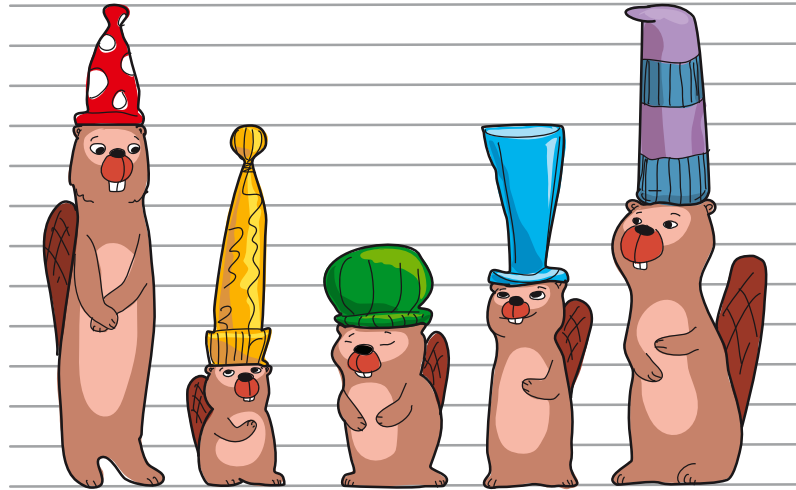
- Modélisation : <https://fr.wikipedia.org/wiki/Modélisation>
- Codage : https://fr.wikipedia.org/wiki/Codage_de_l'information
- Abstraction : [https://fr.wikipedia.org/wiki/Abstraction_\(informatique\)](https://fr.wikipedia.org/wiki/Abstraction_(informatique))





3. Tri par chapeau

Les castors ont de nouveaux chapeaux.

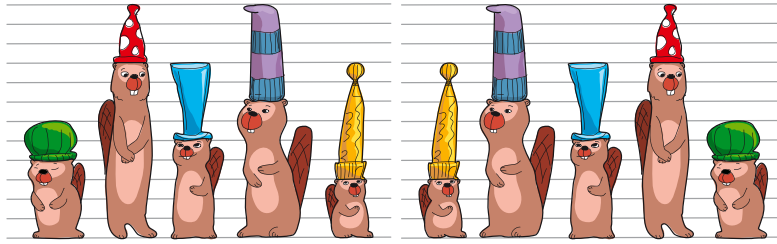


Trie les chapeaux selon leur taille.



Solution

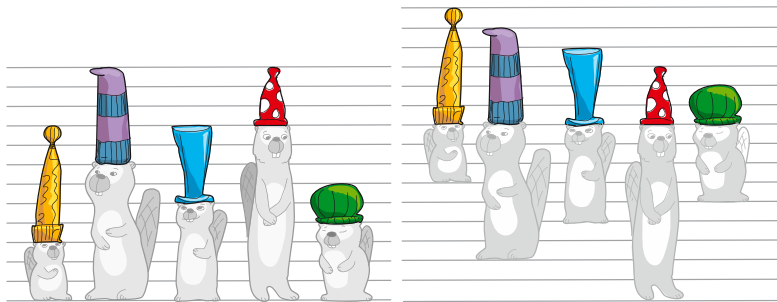
Voici les deux manières de trier les chapeaux :



Il y a deux bonnes réponses, les chapeaux peuvent devenir

- de plus en plus grands, ou
- de plus en plus petits en allant de gauche à droite.





En triant les castors, on ne fait attention qu'aux chapeaux. C'est alors beaucoup plus facile de les trier par taille.

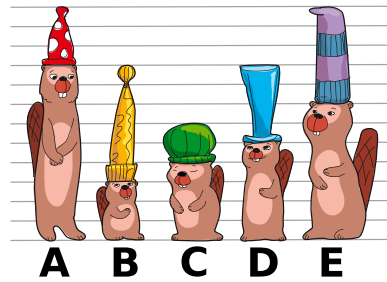


C'est de l'informatique !





Beaucoup d'objets autour de nous sont triés pour mieux pouvoir choisir parmi eux : si les outils sont triés par taille, c'est plus facile de trouver un outil précis. On peut trouver un mot facilement dans un dictionnaire parce que les mots y sont triés par ordre alphabétique.

Dans cet exercice, tu devais trier les castors d'après la taille de leur chapeau. La difficulté est que la *propriété* « taille du chapeau » n'est pas facile à reconnaître. Nous pouvons trier les castors d'après au moins trois tailles :

- Taille des castors ()
- Taille des chapeaux ()
- Taille totale ( + )



Le tri des castors est différent pour chacune de trois propriétés de taille.

Castor			 + 
A	3	9	12
B	6	3	9
C	2	4	6
D	4	5	9
E	5	7	12

Pour trier, il est donc important de commencer par bien définir la propriété d’après laquelle il faudra trier. Ensuite, les valeurs de cette propriété doivent être triables : on peut trier d’après des propriétés qui sont exprimées en nombre (comme la taille, la longueur, le poids . . .) : on peut dire quel nombre est le plus petit entre deux nombres. On peut trier des mots car l’ordre des lettres dans l’alphabet est défini et que c’est donc clair lequel de deux mots vient avant dans le dictionnaire. De manière générale, on peut dire que l’on peut trier d’après une propriété s’il existe une relation « plus petit que » (une *relation d’ordre*) entre ses valeurs individuelles.

Les ordinateurs gèrent de grands volumes de données. Pour pouvoir y trouver des données précises, les données doivent être triées. Il existe beaucoup de méthodes rapides de tri en informatique, et quelle méthode doit être utilisée dans quel cas est un sujet bien étudié.

Mots clés et sites web

- Algorithme de tri : https://fr.wikipedia.org/wiki/Algorithme_de_tri
- Relation d’ordre : https://fr.wikipedia.org/wiki/Relation_d'ordre
- Algorithme de recherche : https://fr.wikipedia.org/wiki/Algorithme_de_recherche

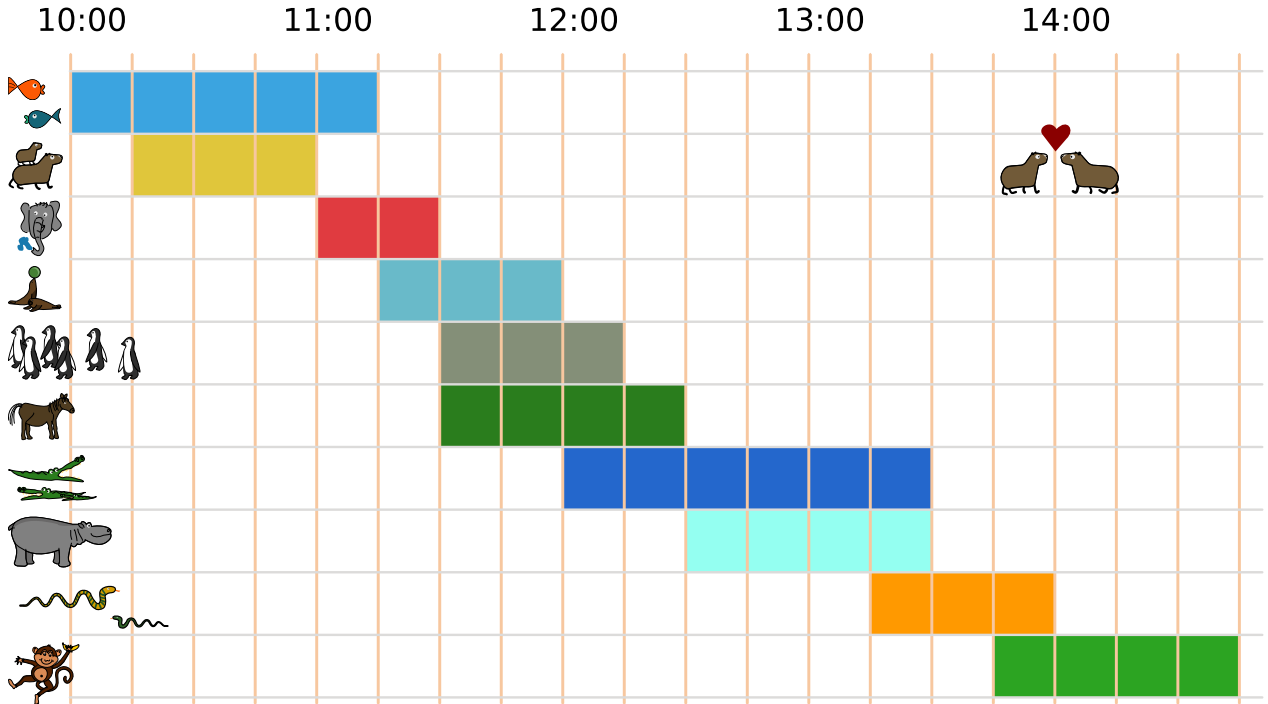




4. Visite au zoo

Aujourd'hui, Anja passe la journée au zoo. Elle veut voir le plus de présentations possible.

Voici un programme avec toutes les présentations. Tout en bas, tu vois par exemples que la présentation des singes commence à 13h45 et finit à 14h45.



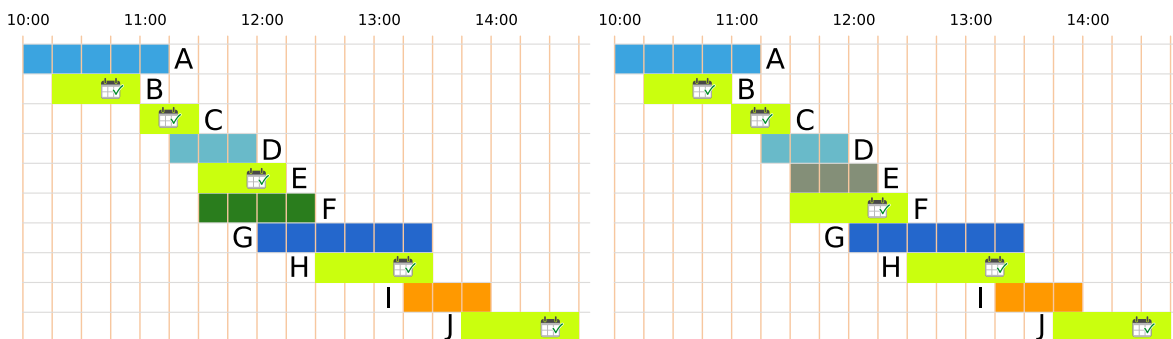
Anja assiste toujours à une présentation en entier, du début à la fin. Peux-tu l'aider ?

Choisis le plus de présentations possible qu'Anja peut voir les unes après les autres.



Solution

Anja peut voir au maximum cinq présentations les unes après les autres. Voici les deux réponses justes :



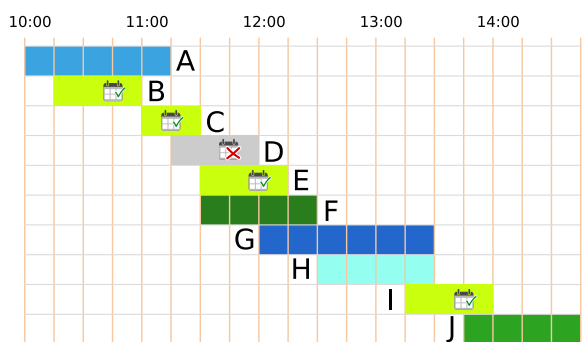
Il y a différentes façons d'arriver aux bonnes réponses.

Un plan de visite pour Anja est une sélection de présentations qu'elle peut voir les unes après les autres. Un moyen de trouver les bonnes réponses est de faire une liste de tous les plans de visite possibles. Les bonnes réponses sont les plans contenant le plus de présentations dans cette liste. Cela prend malheureusement beaucoup de temps pour trouver tous les plans possibles.

Ne pourrait-il pas y avoir de plan de visite avec six présentations ? Essayons d'en faire un. Pour commencer, nous observons la durée des présentations : sur le programme, la journée est divisée en 19 unités de temps d'un quart d'heure chacune. Les présentations durent 2, 3, 4, 5 ou 6 unités de temps.

Unités de temps	Présentation
2	C
3	B, D, E, I
4	F, H, J
5	A
6	G

Pour pouvoir mettre le plus de présentations possible dans un plan de visite, nous choisissons les présentations les plus courtes. Les six présentations les plus courtes durent en tout 18 unités de temps (2 + 3 + 3 + 3 + 3 + 4). Les présentations C, D et E font partie des 6 présentations les plus courtes ; mais comme les présentations C et E sont directement l'une après l'autre, Anja ne peut pas aller voir la présentation D entre deux.





Nous devons donc remplacer la présentation D par une autre présentation aussi courte que possible. Il ne reste que des présentations durant au moins quatre unités de temps. Sans la présentation D, nous avons donc besoin d'au moins 19 unités de temps pour voir six présentations : $2 + 3 + 3 + 3 + 4 + 4$. Mais quelles que soient les deux présentations à quatre unités de temps que nous choisissons, l'une d'entre elles a lieu en même temps qu'une présentation à 3 unités de temps. Nous devrions donc remplacer l'une d'elles par une présentation d'au moins quatre unités de temps et aurions besoin d'au moins 20 unités de temps en tout pour voir six présentations. Mais nous n'avons que 19 unités de temps à disposition et ne pouvons donc pas faire de plan de visite à plus de cinq présentations.

C'est de l'informatique !

Cet exercice du Castor contient un horaire des présentations du zoo. Ce n'est pas facile de créer de tels horaires ; en informatique, on parle de *séquençage de tâches*. Le zoo aimerait évidemment permettre à ses visiteurs de voir le plus de présentations possible, mais d'autres contraintes doivent aussi être prises en compte. Par exemple, une présentation ne peut être proposée que quand les gardiens animaliers sont disponibles, que les arènes sont libres et que les heures sont compatibles avec le rythme de vie des animaux.

Il existe beaucoup de problèmes de ce type dans la vie quotidienne auxquels les mêmes réflexions peuvent être appliquées, par exemple l'élaboration d'un horaire pour l'école ou la programmation des films dans les salles d'un cinéma. L'élaboration de ces horaires est si compliquée que même ces simples exemples (les horaires de ton école) ne peuvent pas être fait manuellement. Les *processeurs* de ton ordinateurs doivent eux aussi effectuer beaucoup de tâches les unes après les autres. Le programme déterminant quel processeur fait quoi à quel moment est créé très rapidement par le *système d'exploitation* sans que l'on ne le remarque. Le *séquençage de tâches* est un thème important en informatique et en recherche.

Mots clés et sites web

- Ordonnancement : https://fr.wikipedia.org/wiki/Ordonnancement_de_travaux_informatiques
- Système d'exploitation : https://fr.wikipedia.org/wiki/Système_d'exploitation
- Séquençage des tâches : https://fr.wikipedia.org/wiki/Séquençage_de_tâches





5. Parapluie



Voici le parapluie d'Anna :

Une des quatre images montre le parapluie d'Anna. Laquelle ?



A)



B)



C)

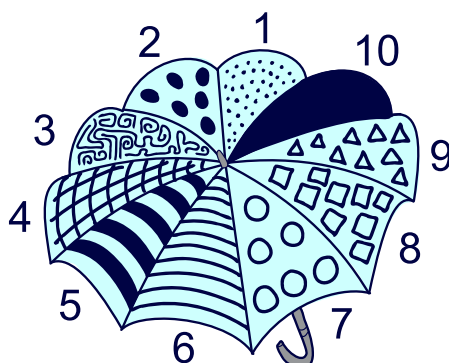


D)



Solution

Chaque motif n'apparaît qu'une seule fois sur le parapluie d'Anna.



Pour trouver la bonne image, nous comparons chacune des images l'une après l'autre avec le parapluie d'Anna :

- Nous cherchons la position du motif situé tout à gauche du parapluie de la réponse possible sur le parapluie d'Anna,
- Nous vérifions que les motifs voisins soient les mêmes sur le parapluie de la réponse possible que sur le parapluie d'Anna.

	A)	B)	C)	D)
Réponse possible				
Parapluie d'Anna				

Chacune des quatre images montre une suite de seulement cinq motifs et pas tous les dix. Nous ne pouvons pas savoir si la suite de cinq motifs d'une des quatre images correspond à la suite de dix motifs complète du parapluie d'Anna.

L'image C est la seule qui montre un parapluie avec cinq motifs correspondants à ceux présents sur le parapluie d'Anna. Toutes les autres images montrent des suites de motifs qui ne correspondent pas, ou seulement en partie, au parapluie d'Anna. Seule l'image C peut donc montrer le parapluie d'Anna.

C'est de l'informatique !

Les réponses possibles ne montrent qu'une partie de la suite de motifs. Même si elles ne contiennent que des *informations partielles*, nous pouvons déterminer laquelle des quatre images montre le parapluie d'Anna : une image ne montre le parapluie d'Anna que si sa suite de motifs correspond exactement à une partie de la suite de motifs du parapluie d'Anna.



Lors d'une recherche dans un document texte, le même principe est appliqué que pour la recherche de motifs sur les parapluies. L'ordinateur recherche des chaînes de caractères correspondant à une information partielle donnée (le mot recherché) dans le document. Une chaîne de caractères est une suite de caractères (par exemple des lettres, des chiffres, des caractères spéciaux). Lors d'une recherche :

- plus le mot recherché est long, moins il y a de correspondances possible dans le texte et plus la chance de trouver l'endroit recherché dans le texte est élevée,
- plus le mot recherché est court, plus il y a de correspondances possible dans le texte et moins la recherche est exacte.

Pour améliorer la recherche et le parcours des données, différentes méthodes de recherche (ou *algorithmes de recherche*) ont été développées. Leur but est d'effectuer une recherche exacte le plus rapidement possible et de générer un résultat adapté. Ces algorithmes de recherche sont sans cesse améliorés et peuvent parcourir d'immenses quantités de données en très peu de temps (les moteurs de recherche sur internet utilisent de tels algorithmes).

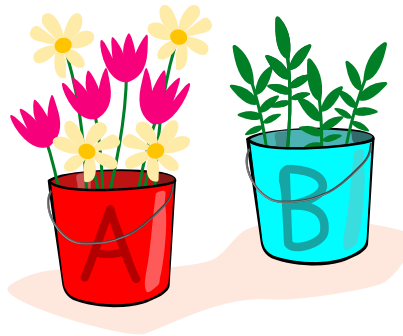
Mots clés et sites web

- Chaîne de caractères, string : https://fr.wikipedia.org/wiki/Chaîne_de_caractères
- Algorithme de recherche : https://fr.wikipedia.org/wiki/Algorithme_de_recherche








6. Fleuriste



Florian vend des bouquets de fleurs. Il assemble chaque bouquet d'après ces instructions :

1. Prendre une première fleur du seau A.
2. Si cette première fleur est une marguerite , prendre une deuxième marguerite .
3. Prendre maintenant une branche  du seau B jusqu'à ce que le bouquet ait quatre éléments.
Voilà !

Aide Florian : suis les instructions et choisis des fleurs et des branches pour un bouquet.



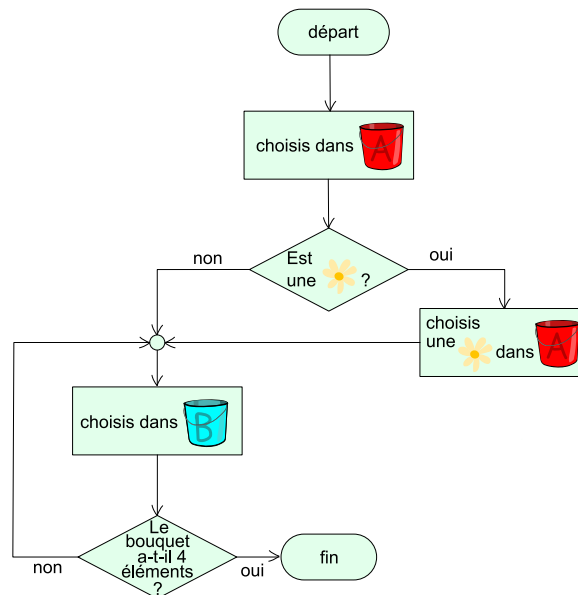


Solution

Il y a deux solutions possibles :



Pour assembler les bouquets de fleurs correctement, Florian doit suivre les instructions. On peut représenter ces instructions à l'aide d'un diagramme :



Après que Florian a choisi la première fleur du seau A, il doit prendre une décision qui dépend de la première fleur. Soit il prend une deuxième marguerite , soit il suit la flèche « non » et prend une branche .

Ensuite, il vérifie si son bouquet a déjà quatre éléments. Si non, il suit la flèche « non » et prend encore une branche avant de vérifier à nouveau le nombre d'éléments.

S'il commence par prendre une marguerite , il va donc prendre une deuxième marguerite puis deux fois une branche. Par contre, s'il commence par prendre une tulipe , il va ensuite directement prendre des branches du seau B jusqu'à avoir 4 éléments, donc 3 branches en tout.



C'est de l'informatique !

Les *instructions* pour l'assemblage de bouquets de fleurs sont claires et pourraient être effectuées par une machine. En informatique, cela s'appelle un *algorithme*. Certaines instructions utilisées ici sont aussi souvent utilisées dans les programmes informatiques :

- La première instruction est la sélection d'un objet au hasard parmi un ensemble d'objets ;
- La deuxième instruction s'appelle une *instruction conditionnelle*, car il faut choisir entre deux possibilités ou plus ;
- La troisième instruction a l'air relativement simple, mais doit être bien structurée dans un programme informatique. La partie intérieure de l'instruction (une instruction en elle-même : « prend une branche dans le seau B ») doit être répétée plusieurs fois jusqu'à ce que le bouquet de fleurs soit composé de quatre éléments. L'instruction intérieure est donc effectuée jusqu'à ce que la condition « le bouquet a quatre éléments » soit remplie. Une telle *instruction itérative* est aussi appelée *boucle*.

Il existe différentes manières de représenter un algorithme. Dans cet exercice, l'algorithme « bouquet » de Florian est formulé par des instructions en langage naturel. Dans l'explication de la solution, il est représenté sous la forme d'un organigramme de programmation.

Les fleuristes sont des artisans. Il existe des traditions et des règles gouvernant l'assemblage des bouquets et couronnes de fleurs. C'est un exemple de situation de la vie quotidienne dans laquelle les instructions et les algorithmes jouent un rôle.

Mots clés et sites web




- Instruction conditionnelle :
[https://fr.wikipedia.org/wiki/Instruction_conditionnelle_\(programmation\)](https://fr.wikipedia.org/wiki/Instruction_conditionnelle_(programmation))
- Boucle : https://fr.wikipedia.org/wiki/Structure_de_contrôle#Boucles
- Organigramme de programmation :
https://fr.wikipedia.org/wiki/Organigramme_de_programmation
- Fleuriste : <https://fr.wikipedia.org/wiki/Fleuriste>



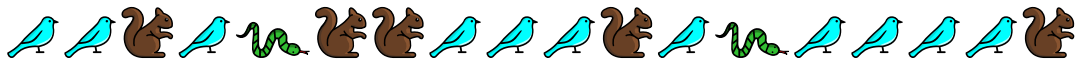


7. L'arbre magique

Ben a un arbre magique dans son jardin :

- Si un oiseau  se pose sur l'arbre, deux pommes y poussent tout de suite.
- Si un écureuil  grimpe sur l'arbre, une pomme en tombe. S'il n'y a pas de pomme sur l'arbre, il ne se passe rien.
- Si un serpent  vient sous l'arbre, toutes les pommes disparaissent tout de suite.

Ce matin, 25 pommes sont sur l'arbre. Plusieurs animaux rendent ensuite visite à l'arbre, un écureuil en dernier. Ben a noté leur ordre exact :



Combien de pommes y a-t-il sur l'arbre après la dernière visite ?

- A) 3 pommes
- B) 7 pommes
- C) 17 pommes
- D) 31 pommes



Solution

La bonne réponse est B. Après que le dernier écureuil est monté sur l'arbre, il y reste sept pommes.

On peut calculer combien de pommes se trouvent sur l'arbre à chaque visite d'un animal :

Animal :	Départ					
Instruction :	-	+2	+2	-1	+2	reset
Nombre de pommes :	25	27	29	28	30	0

Animal :	Report								
Instruction :	-	-	-	+2	+2	+2	-1	+2	reset
Nombre de pommes :	0	0	0	2	4	6	5	7	0

Animal :	Report					
Instruction :	-	+2	+2	+2	+2	-1
Nombre de pommes :	0	2	4	6	8	7

Comme toutes les pommes disparaissent lorsqu'un serpent vient sous l'arbre, nous pouvons ignorer tout ce qui se passe avant l'arrivée du deuxième (et dernier) serpent. Comme indiqué dans le tableau, quatre oiseaux se posent sur l'arbre après le passage du dernier serpent. Il y a ensuite $4 \times 2 = 8$ pommes sur l'arbre. Ensuite, un écureuil grimpe, faisant tomber une pomme ; il reste donc $8 - 1 = 7$ pommes sur l'arbre.

C'est de l'informatique !

La visite d'un animal modifie l'état de l'arbre magique – mais d'une manière bien définie : seul le nombre de pommes sur l'arbre change. La visite des animaux ne change pas les autres propriétés de l'arbre, comme son nombre de feuilles, la longueur de ses branches ou la forme de son tronc, par exemple. Pour cet exercice, il est donc suffisant de considérer le nombre de pommes.

Un programme informatique a lui aussi un état qui peut être modifié par les instructions du programme. On considère généralement les données stockées par le programme comme son état ; ces données sont stockées par le programme dans des *variables* définies lors de la programmation.

La suite des visites des animaux à l'arbre dans cet exercice est comme un programme informatique : chaque visite est une instruction qui change l'état de l'arbre. Cet état (ici, le nombre de pommes sur l'arbre) peut être stocké à l'aide d'une seule variable.

Tu as peut-être remarqué que tu n'avais pas besoin de considérer le « programme » en entier pour résoudre l'exercice, mais uniquement la partie suivant la dernière visite d'un serpent. En observant attentivement l'effet des différentes instructions sur l'état du programme, on peut découvrir certaines



propriétés de ce programme. Une telle analyse de programmes (informatiques) fait partie des tâches des informaticiens et informaticiennes.

Mots clés et sites web

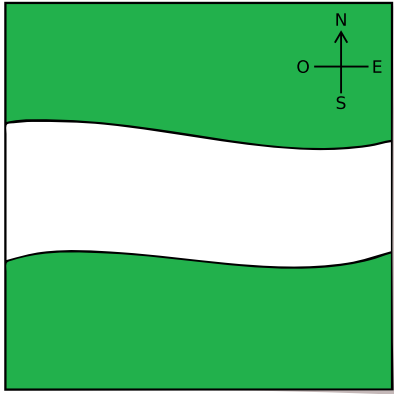
- Variable: [https://fr.wikipedia.org/wiki/Variable_\(informatique\)](https://fr.wikipedia.org/wiki/Variable_(informatique))
- État d'un programme: [https://fr.wikipedia.org/wiki/État_\(informatique\)#Processus](https://fr.wikipedia.org/wiki/État_(informatique)#Processus)



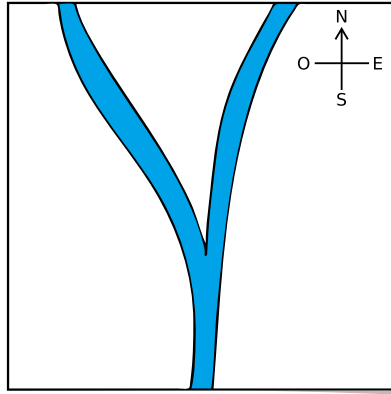


8. La maison de Karla

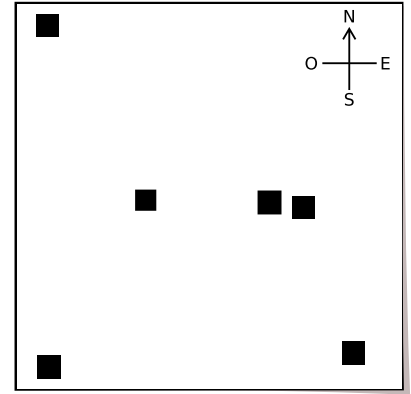
Karla a trois cartes qui montrent exactement la même région. Une carte montre les forêts; une autre, les rivières, et la troisième, les maisons dans cette région. La maison de rêve de Karla se trouve dans la forêt et près d'une rivière.



Carte des forêts



Carte des rivières



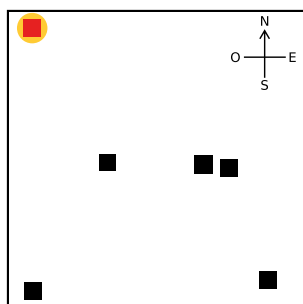
Carte des maisons

Quelle est la maison de rêve de Karla ?

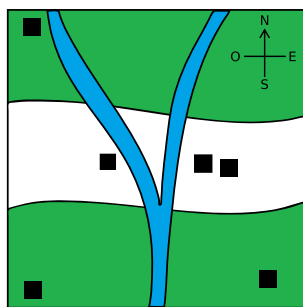


Solution

La maison en haut à gauche de la carte des maisons est la maison de rêve de Karla :



Pour trouver la maison de rêve de Karla, il faut analyser les informations des trois cartes. La maison de rêve doit se trouver dans une forêt et près d'une rivière. Ce n'est vrai que pour la maison en haut à gauche. C'est facile à voir en superposant les trois cartes :



C'est de l'informatique !

Lorsque les informations sur les forêts, les rivières et les maisons sont représentées sur une seule carte, c'est facile de trouver la maison recherchée.

Un *système d'information géographique* (SIG) assemble une multitude d'informations spatiales (par exemple les forêts, routes, frontières, stations service, maisons, etc.) et les représente sur une carte. Un SIG sert donc à la visualisation et à l'analyse de *données géographiques*. Un SIG permet par exemple à la protection civile de mettre en place des plans d'évacuation.

Les programmes graphiques utilisent aussi plusieurs *niveaux* avec des informations graphiques différentes (appelés *calques*). Une question importante est toujours quel niveau est le plus haut et est donc représenté au premier plan. Ici, ce sont par exemple les maisons qui doivent être au premier plan afin qu'elles ne soient pas cachées par les forêts.

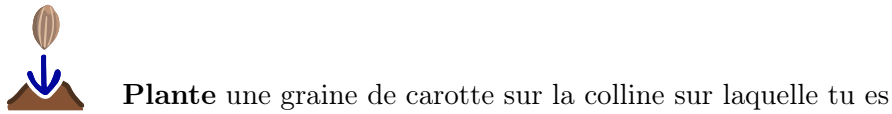
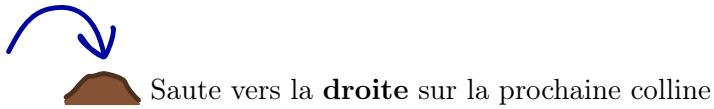
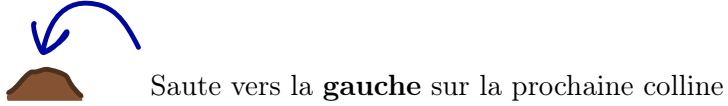
Mots clés et sites web

- SIG : https://fr.wikipedia.org/wiki/Système_d'information_géographique
- Calque : [https://fr.wikipedia.org/wiki/Calque_\(infographie\)](https://fr.wikipedia.org/wiki/Calque_(infographie))



9. Graines de carottes

Le robot lapin peut exécuter les instructions suivantes :



Le robot lapin a suivi la suite d'instructions suivante :



Il a passé sur quatre collines. Nous ne savons pas sur quelle colline il a commencé.

Sur quelles collines le robot a-t-il planté des graines de carottes ?

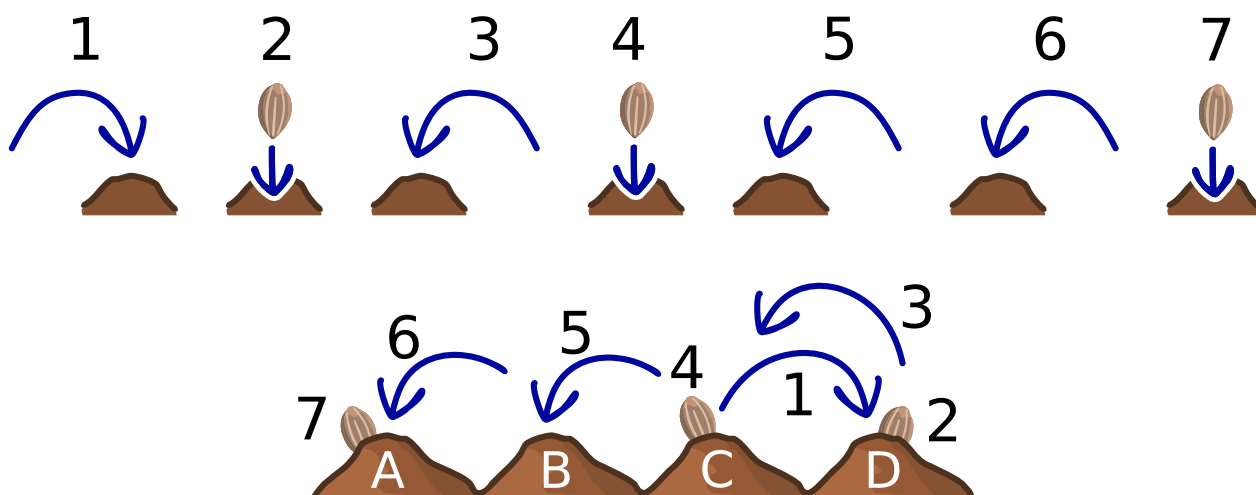




Solution



Pour mieux pouvoir expliquer la bonne réponse, nous donnons des lettres aux collines (voir ci-dessus) et des numéros aux instructions :



Nous commençons par déterminer le point de départ du robot : avant de sauter trois fois de suite vers la gauche (instructions 3, 5 et 6), il doit se trouver sur la colline D. Avant cela, il saute une fois vers la droite (instruction 1). Le robot a donc commencé sur la colline C. Les graines de carottes sont donc plantées sur les collines D, puis C et finalement A, d'après les instructions 2, 4 et 7.

C'est de l'informatique !

Les vrais robots ont des ordinateurs intégrés, et ils sont programmés de manière similaire au robot lapin. Un programme informatique est constitué de plusieurs *instructions* individuelles.

Dans notre cas, la suite d'instructions pour l'ordinateur du robot est donnée sous forme d'images. Le résultat (*sortie*, *output* en anglais) du programme ne dépend pas que de la position de départ (*entrée*, *input* en anglais), mais aussi des instructions et de l'ordre dans lequel elles sont données.

Cet exercice du Castor illustre l'utilisation de robot dans l'agriculture. Les robots ne peuvent pas seulement planter, mais aussi arroser, polliniser et répandre des produits de protection de manière ciblée.

Mots clés et sites web

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



- Smart Farming : <https://www.agroscope.admin.ch/agroscope/fr/home/themes/economie-technique/smart-farming.html>
- Les robots et l'agriculture : <https://cordis.europa.eu/article/id/441912-robots-help-farmers-say-goodbye-to-repetitive-tasks/fr>





10. Riccas

Évelyne a cinq images de riccas. Elle écrit des phrases qui les décrivent.



Son amie Lydia lui montre une sixième image de ricca :



Évelyne remarque alors qu'une de ses phrases sur les riccas est fausse.

Laquelle de ces phrases sur les riccas est fausse ?

- A) Tous les riccas ont des dents.
- B) Certains riccas ont des ailes.
- C) Les riccas ont soit des cornes, soit trois yeux, mais jamais des cornes *et* trois yeux.
- D) Si un ricca a exactement deux bras, alors il a aussi exactement deux jambes.



Solution

La réponse D) est la bonne réponse : *Si un ricca a exactement deux bras, alors il a aussi exactement deux jambes.*

La réponse A) est une affirmation qui doit être vraie pour tous les riccas. Si un seul ricca n'avait pas de dents, l'affirmation serait fausse. Comme tous les riccas qu'Évelyne connaît ont des dents, la phrase de la réponse A) n'est pas forcément fausse.

La réponse B) est une affirmation qui ne doit être vraie que pour certains riccas. Comme un des six riccas qu'Évelyne connaît a des ailes, cette phrase est juste pour les six riccas. Même si aucun des six riccas n'avait d'ailes, ce serait possible que d'autres riccas en aient, et la phrase pourrait quand même être juste. Cette phrase ne serait forcément fausse que si Évelyne connaissait tous les riccas et qu'aucun n'avait d'ailes.

La réponse C) relie deux affirmations avec « soit-soit ». Cette affirmation reliée est vraie lorsqu'exactlyement une des deux affirmations simples est vraie. C'est le cas pour les six images : quatre riccas ont des cornes mais n'ont pas trois yeux et les deux autres riccas n'ont pas de cornes, mais trois yeux. Pour que la phrase soit fausse, il faudrait qu'il y ait un ricca avec des cornes *et* trois yeux, ou un ricca sans cornes et avec un autre nombre d'yeux que trois. Ce n'est le cas d'aucun des six riccas qu'Évelyne ne connaît ; la phrase n'est donc pas forcément fausse.

Il reste la phrase de la réponse D). Elle est formulée à l'aide d'une condition « si-alors » : l'affirmation qui suit le « alors » doit être vraie chaque fois que l'affirmation qui suit le « si » est vraie. La condition « si » est vraie pour tous les six riccas qu'Évelyne connaît : ils ont tous exactement deux bras. Tous les riccas sur les cinq premières images d'Évelyne ont également deux jambes ; pour eux, la phrase d'Évelyne est donc juste. Par contre, le ricca sur l'image de Lydia a plus de deux jambes, cinq exactement. Cette phrase est donc forcément fausse.

C'est de l'informatique !

Le nombre d'ailes, de bras, de jambes et d'yeux, et la présence de dents ou de cornes sont des *propriétés* des riccas. Lorsque l'on décrit des riccas, on formule des *affirmations* sur ces propriétés. Cela mène à un *modèle* de ce que sont les riccas.

Les ordinateurs utilisent beaucoup de modèles. Certains sont formulés de manière explicite, comme un modèle des écolières et écoliers sous forme d'une banque de données contenant noms, dates de naissance et adresses. D'autres modèles sont construits par les ordinateurs lorsqu'on leur donne, par exemple, des images à comparer pour entraîner un réseau de neurones.

Les phrases d'Évelyne – donc son modèle des riccas dans cet exercice – sont formulées sous la forme d'*affirmations logiques*. Certaines ont des quantifications (« tous », « certains », « il existe »), d'autres utilisent des *opérateurs logiques* (« soit-soit », « si-alors »). Ces expressions logiques sont *formalisées* : cela veut dire que leur utilisation et signification sont bien définies.



Cette définition permet de :

- relier des affirmations (simples) à l'aide de quantifications et d'opérateurs pour en faire des affirmations complexes,
- dériver la signification des affirmations complexes à partir des affirmations simples.

Les affirmations logiques sont une méthode répandue pour décrire des modèles en informatique.

Mots clés et sites web

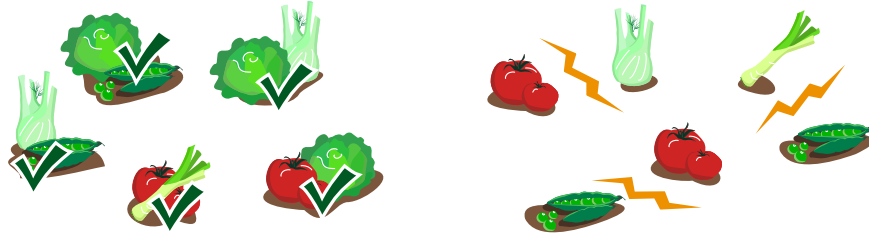
- Modèle : <https://fr.wikipedia.org/wiki/Modèle>
- Modélisation : <https://fr.wikipedia.org/wiki/Modélisation>
- Apprentissage automatique :
https://fr.wikipedia.org/wiki/Apprentissage_automatique





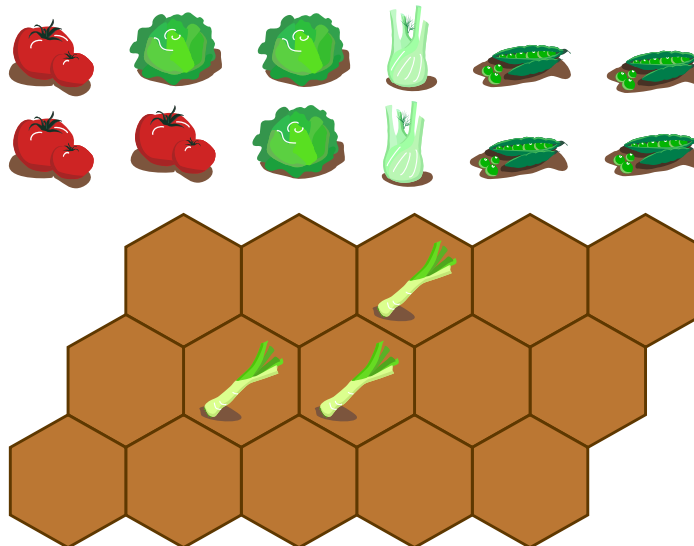
11. Jardin potager

Lisa prépare un jardin potager. Elle veut y cultiver cinq sortes de légumes différents. Certaines sortes de légumes se supportent bien et sont compatibles ✓, d'autres sont incompatibles ⚡ :



Lisa a divisé le jardin en domaines hexagonaux. Elle veut planter exactement une sorte de légumes dans chaque domaine.

Lisa a déjà planté des poireaux  dans trois domaines.



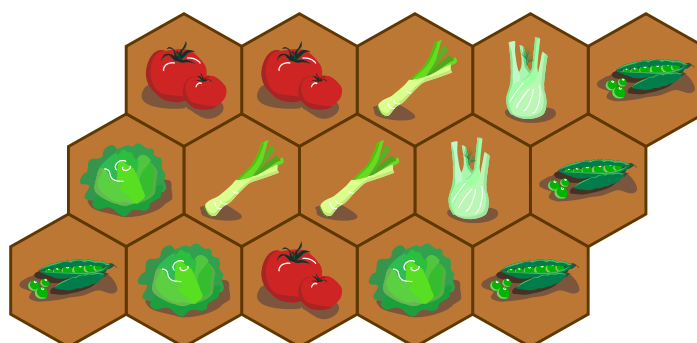
Lisa plante en suivant la règle suivante : les légumes incompatibles ne peuvent pas être plantés dans des domaines qui se touchent.

Plante une sorte de légumes dans chaque domaine encore libre en respectant la règle de Lisa.



Solution

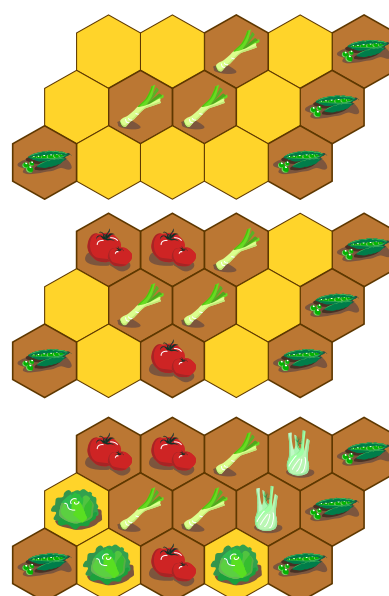
Voici la bonne réponse :



Comme les petits pois ne sont pas compatibles avec les poireaux, Lisa ne plante pas de petits pois dans les domaines jaunes. Il ne reste que les autres domaines pour le petits pois.

Comme les tomates ne sont pas compatibles avec les petits pois, Lisa ne plante pas de tomates dans les domaines jaunes. Elle peut planter des tomates dans les autres domaines, car les tomates et les poireaux sont compatibles.

Comme les fenouils ne sont pas compatibles avec les tomates, Lisa ne plante pas de fenouil dans les domaines jaunes. Elle peut planter du fenouil dans les deux domaines entre les poireaux et les petits pois. Lisa peut planter de la salade dans les domaines jaunes, car la salade est compatible avec tous les autres légumes.



C'est de l'informatique !

Pour planter des légumes afin d'avoir une récolte aussi grande que possible, il faut respecter beaucoup de *conditions* : chaque sorte a des besoins de place, de nutriments et de lumière différents, par exemple. Dans cet exercice du Castor, nous ne considérons qu'une sorte de condition : la compatibilité des différentes sortes de légumes entre elles.

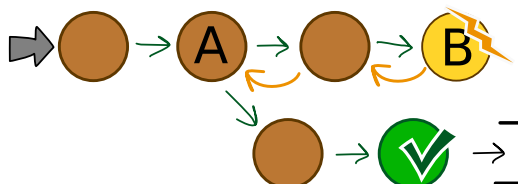
Pour déterminer quoi planter où dans le jardin de Lisa tout en respectant les conditions de compatibilité, on pourrait procéder de la manière suivante : on essaie toutes les combinaisons de légumes de manière systématique. Une fois que le jardin est rempli, on vérifie si les conditions sont remplies et si la combinaison est une solution au problème de Lisa. En informatique, on appelle un telle manière d'essayer toutes les combinaisons possibles une *recherche exhaustive*. Cette méthode peut prendre beaucoup de temps si elle est appliquée à des problèmes ayant beaucoup de combinaisons possibles et peu de solutions.



C'est souvent mieux de procéder étape par étape et de prendre en compte toutes les conditions à chaque étape. C'est ainsi que nous avons trouvé la solution au problème de Lisa, et aucune « fausse » combinaison ou arrangement du jardin n'était possible.

Heureusement, c'était possible de trouver la solution directement : il y avait toujours des domaines dans lesquels nous pouvions planter certains des légumes restants. Ce n'est pas toujours le cas.

Lorsque l'on essaie d'assembler la réponse étape par étape, il peut y avoir plusieurs possibilités de remplir toutes les conditions à une certaine étape A :



Suivant le choix fait en A, il peut ne plus y avoir de possibilités à une étape suivante B. On revient alors en arrière sur les dernières étapes jusqu'à arriver à nouveau à l'étape A offrant plusieurs possibilités. On choisit alors une autre possibilité et essaie de trouver un solution depuis là.

Ce retour en arrière est appelé *retour sur trace* en informatique (*backtracking* en anglais).

Mots clés et sites web

- Recherche exhaustive : https://fr.wikipedia.org/wiki/Recherche_exhaustive
- Retour sur trace : https://fr.wikipedia.org/wiki/Retour_sur_trace




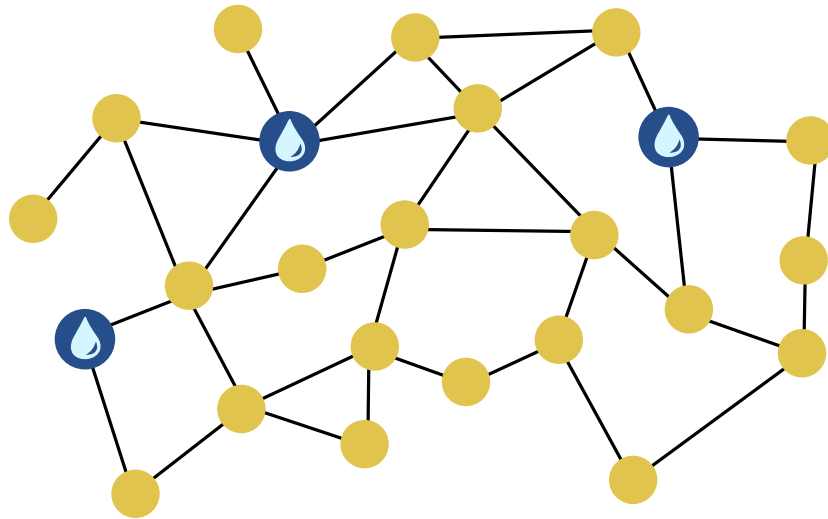


12. Fontaine

L'été est chaud dans la ville. La maire fait installer des fontaines d'eau potable.

Les fontaines doivent être installées de manière à ce qu'il ne faille jamais parcourir plus de deux tronçons de rue pour atteindre une fontaine depuis n'importe quel coin de rue. La maire sera alors satisfaite.

Voici un plan de la ville. Les lignes sont les tronçons de rue et les points les coins de rue. Il y a déjà des fontaines  à trois coins de rue.

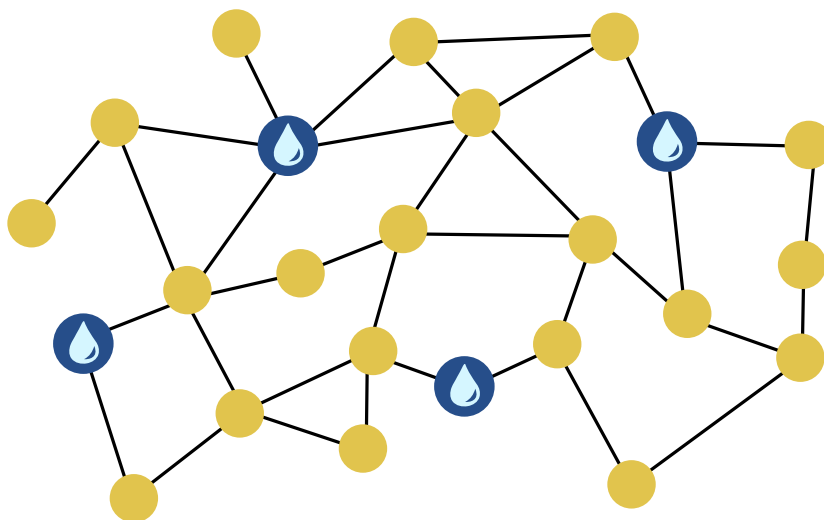


Installe une fontaine supplémentaire pour satisfaire la maire.



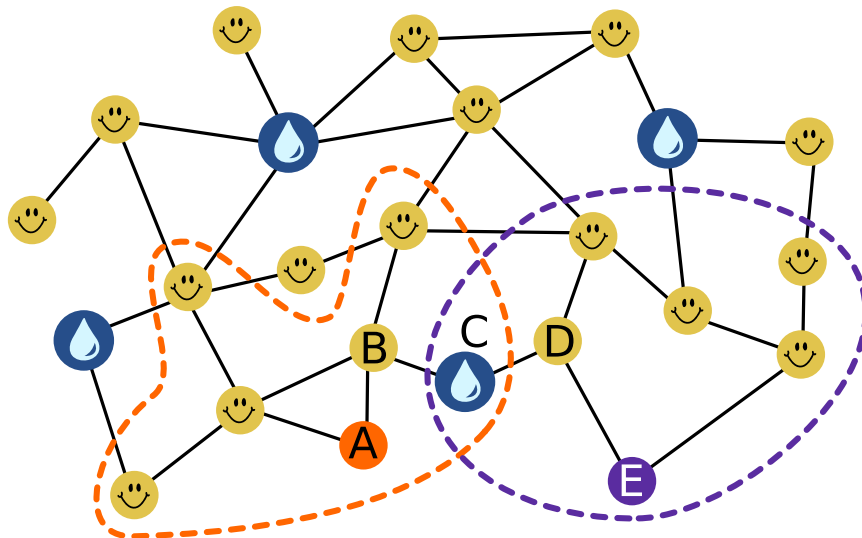
Solution

Voici la bonne réponse :



Si une fontaine supplémentaire est installée en bas au centre, il faut parcourir au maximum deux tronçons de rue pour atteindre une fontaine. La maire est alors satisfaite.

Comment pouvons-nous déterminer à quel coin de rue installer une fontaine supplémentaire ? Sur le plan, nous indiquons d'un 😊 tous les coins de rue se trouvant déjà à deux tronçons de rue d'une fontaine au maximum. La maire peut déjà être satisfaite de ces coins de rue.



Pour les cinq coins de rue A, B, C, D et E restants, nous ajoutons une fontaine au coin de rue C. Comme ça, ces coins-là sont aussi loin de deux tronçons au maximum d'une fontaine.

Le coin C est le seul endroit où installer une nouvelle fontaine permettant de satisfaire la maire : Si l'on considère les coins de rue se trouvant à deux tronçons des coins A et E (entourés d'une ligne sur l'image), seul le coin C remplit cette condition pour les coins A et E.



C'est de l'informatique !

Le plan de la ville peut être représenté par un *graphe*. C'est un outil important en informatique qui permet de modéliser les relations entre des objets et de répondre à des questions sur ces relations. Ici, on peut représenter les coins de rue comme des objets et donc des *nœuds* du graphe. Les relations entre deux objets sont modélisées par des *arêtes* qui relient deux nœuds. Ici, une arête entre deux coins de rue veut dire qu'ils sont reliés par un tronçon de rue. On peut appeler cette relation « voisinage ». Les arêtes peuvent aussi représenter d'autres relations, comme l'amitié.

Dans cet exercice du Castor, il faut trouver un sous-ensemble de nœuds (pour installer une fontaine) de manière à ce que chaque nœud à l'extérieur de ce sous-ensemble soit relié à un « nœud fontaine » par un chemin de deux arêtes de long au maximum. En langage technique informatique, on parlerait de la recherche d'un ensemble 2-dominant (*distance-2 dominating set* en anglais). En général (pour toutes les longueurs de chemin $d \geq 1$), cette recherche d'un sous-ensemble de taille minimale fait partie des problèmes les plus difficiles rencontrés en informatique.

Les « ensembles d -dominants » jouent un rôle croissant actuellement, en particulier dans le domaine du *social computing* : pour traiter des données venant de réseaux sociaux de manière automatique (par exemple pour détecter la diffusion de *fake news*), les relations entre les utilisateurs (fan, follower, ami) sont modélisées sous forme de graphes. Ces graphes peuvent être si grands que seule une sélection représentative (aussi petite que possible) peut être prise en considération – par exemple, un set 3-dominant. Comme la sélection la plus petite possible ne peut pas être calculée efficacement, on développe des méthodes informatiques qui permettent de rapidement déterminer de petites sélections (mais pas forcément *la* plus petite).

Mots clés et sites web

- Ensemble dominant : https://fr.wikipedia.org/wiki/Ensemble_dominant
- Théorie des graphes : https://fr.wikipedia.org/wiki/Théorie_des_graphes
- Analyse des réseaux sociaux :
https://fr.wikipedia.org/wiki/Analyse_des_réseaux_sociaux



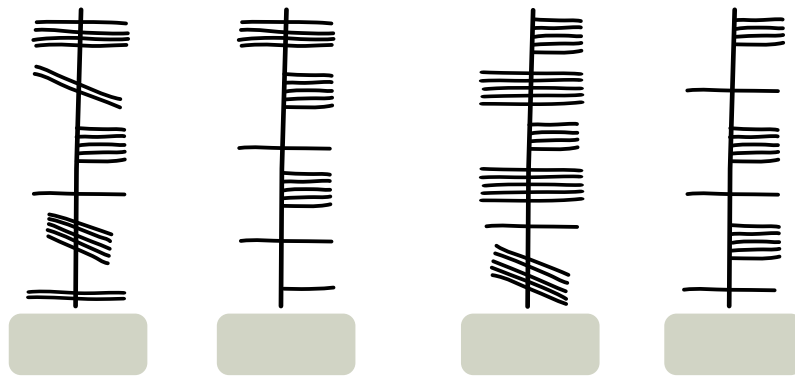


13. Ogham

Sue connaît le vieil alphabet irlandais utilisé en écriture oghamique. Chaque lettre est composée d'un ou plusieurs traits qui sont arrangés le long d'une longue ligne. Deux lettres qui se suivent sont séparées par un espace le long de la ligne.

Sue utilise l'écriture oghamique comme code secret. Elle écrit ainsi quatre mots – ses fruits préférés : ANANAS, BANANE, RAISIN et ORANGE.

Quel mot correspond à quel code en Ogham ?



ANANAS

BANANE

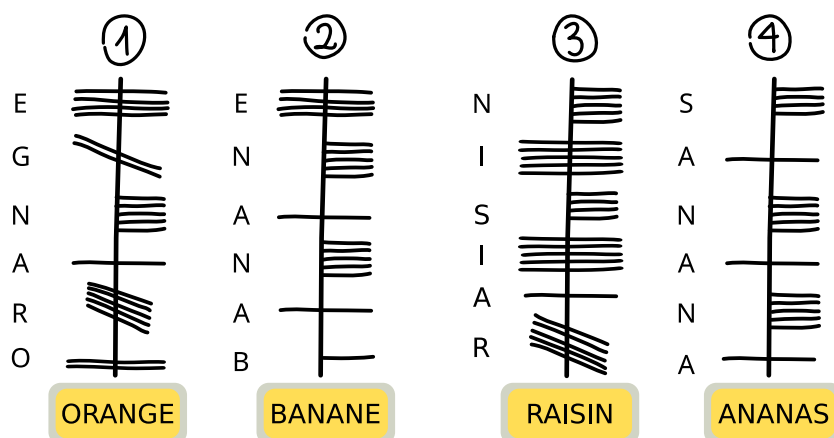
RAISIN

ORANGE



Solution

Voici la bonne réponse :



Il y a plusieurs possibilités de trouver la bonne assignation. Il faut dans tous les cas déterminer dans quel sens les lettres sont écrites le long de la ligne. Pour cela, le mot ANANAS est spécialement utile : il contient trois fois la lettre A, séparée par d'autres lettres.

Il n'y a que dans le code en Ogham 4 que la même lettre apparaît trois fois avec d'autres lettres entre deux. Le code 4 est donc le seul qui peut correspondre au mot ANANAS. On peut en déduire que les mots sont écrits de bas en haut en Ogham et que la lettre A s'écrit avec un trait horizontal traversant le ligne verticale.

La lettre A en Ogham n'est présente deux fois que dans le code 2. De plus, on connaît le symbole Ogham du N grâce au code d'ANANAS (cinq traits verticaux à droite de la ligne), et l'ordre des autres lettres indique que seul le mot BANANE correspond à ce code. ORANGE ne va qu'avec le code 1, entre autre parce qu'on n'y trouve la lettre A qu'une seule fois et en troisième position. Il ne reste que le code 3 pour le mot RAISIN, et on y retrouve en effet les lettres Ogham R, S et N connues des autres mots aux bonnes positions.

C'est de l'informatique !

Dans cet exercice du Castor, il faut déchiffrer un texte inconnu. Ici, ce n'est pas très difficile car le *texte clair* est connu. De plus, le texte inconnu est divisé en lettres et en mots comme le texte connu. Lorsque l'on déchiffre un texte secret ou dans un alphabet inconnu sans le connaître en texte clair, c'est souvent utile de réfléchir à la fréquence des mots et des lettres, et d'utiliser cela comme base pour trouver ces mots et lettres dans le texte. C'est de cette manière que plusieurs écritures et alphabets antiques ont été déchiffrés. Cela devient plus compliqué lorsque les symboles du texte inconnu ne sont pas faciles à assigner aux lettres et mots du texte connu comme il le sont en Ogham. Dans ce cas, il est souvent nécessaire de comparer le texte à des textes ou écritures connues, comme dans cet exercice. Les hiéroglyphes égyptiens, par exemple, n'ont pas pu être déchiffrés pendant des siècles, jusqu'à ce qu'une pierre avec des hiéroglyphes et deux textes connus soit trouvée par hasard, la pierre de Rosette. Sur la pierre se trouvait trois fois le même texte écrit dans des langues



différentes, mais contenant les mêmes noms. Ceci permet de déchiffrer des éléments essentiels des hiéroglyphes. Ce n'est cependant pas le cas de tous les alphabets : environ 650 symboles de la culture Maya ne sont toujours pas entièrement déchiffrés, ainsi que les écritures linéaires A et B de la région méditerranéenne.

En informatique aussi, il faut déchiffrer des textes et des symboles – après qu'ils ont été encryptés pour le transfert de données sécurisé. Pour cela, des méthodes très différentes de celles utilisées pour coder des mots dans d'autres écritures sont appliquées. De tels chiffres simples sont trop faciles à déchiffrer, surtout avec des ordinateurs, en général à l'aide des analyses de fréquence des mots et lettres mentionnées plus haut.


Mots clés et sites web

- Cryptographie : <https://fr.wikipedia.org/wiki/Cryptographie>
- Cryptoanalyse : <https://fr.wikipedia.org/wiki/Cryptanalyse>
- Ogham : <https://fr.wikipedia.org/wiki/Ogham>



A. Auteur·e·s des exercices

 Somayah Albaradei

 Esraa Almajhad

 Aldrich Ellis Catapang Asuncion

 Masiar Babazadeh

 Leonardo Barichello

 Liam Baumann


 Wilfried Baumann

 Javier Bilbao


 Špela Cerar


 Sarah Chan

 Marios Omar Choudary

 Gunnar Collier

 Eimear Colreavy

 Valentina Dagiené


 Darija Dasović

 Christian Datzko

 Nora A. Escherle

 Gerald Futschek


 Bence Gaál

 Emily Gates

 Ștefan Gura

 Josefine Hiebler

 Mathias Hiron

 Hyun-seok Jeon

 David Khachatryan

 Vaidotas Kinčius

 Mhairi King


 Víctor Koleszar

 Taina Lehtimäki

 Angélica Herrera Loyo

 Carlos Luna

 Yong Mao

 Yoshiaki Matsuzawa

 Natalia Natalia

 Marika Parviainen

 Elsa Pellet

 Jean-Philippe Pellet

 Zsuzsa Pluhár

 Wolfgang Pohl

 Estela Ramić

 Kirsten Schlüter

 Giovanni Serafini

 Alieke Stijf

 Gabrielé Stupuriené

 Marianne Thut

 Monika Tomcsányiová

 Svetlana Unković

 Jiří Vaníček

 Florentina Voboril

 Michael Weigend

 Kyra Willekes



B. Partenaires académiques

ABZ

AUSBILDUNGS- UND BERATUNGSZENTRUM
FÜR INFORMATIKUNTERRICHT

<http://www.abz.inf.ethz.ch/>

Ausbildungs- und Beratungszentrum für Informatikunterricht der
ETH Zürich.

hep/ haute
école
pédagogique
vaud

<http://www.hepl.ch/>

Haute école pédagogique du canton de Vaud

Scuola universitaria professionale
della Svizzera italiana

<http://www.supsi.ch/home/supsi.html>

La Scuola universitaria professionale della Svizzera italiana
(SUPSI)

SUPSI



C. Sponsoring

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



Kanton Zürich
Volkswirtschaftsdirektion
Amt für Wirtschaft und Arbeit

Standortförderung beim Amt für Wirtschaft und Arbeit Kanton Zürich



UBS

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



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



i-factory (Musée des transports, Lucerne)

senarclens
leu+partner
strategische kommunikation

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



D. Offres supplémentaires



IT tout feu tout flamme : <https://it-feuer.ch/fr/>

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



Coding club des filles :

<https://www.epfl.ch/education/education-and-science-outreach/fr/jeunepublic/coding-club/>

Programmer une application ? Inventer un jeu vidéo ? Créer une animation ? Si une de ces activités t'intéresse, cet espace est fait pour toi ! Viens échanger et partager tes idées, apprendre à coder et découvrir les métiers liés à l'informatique. Les filles de 11 à 15 ans intéressées par la programmation et l'informatique peuvent participer aux ateliers du Coding club des filles.



Roteco : <https://www.roteco.ch/fr/>

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



010100110101011001001001
010000010010110101010011
010100110100100101000101
001011010101001101010011
010010010100100100100001

SS!E

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

Devenez vous aussi membre de la SSIE

<http://svia-ssie-ssii.ch/la-societe/devenir-membre/>

et soutenez le Castor Informatique par votre adhésion

Peuvent devenir membre ordinaire de la SSIE toutes les personnes qui enseignent dans une école primaire, secondaire, professionnelle, un lycée, une haute école ou donnent des cours de formation ou de formation continue.

Les écoles, les associations et autres organisations peuvent être admises en tant que membre collectif.