



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

Exercices et solutions 2020

Années HarmoS 5/6

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

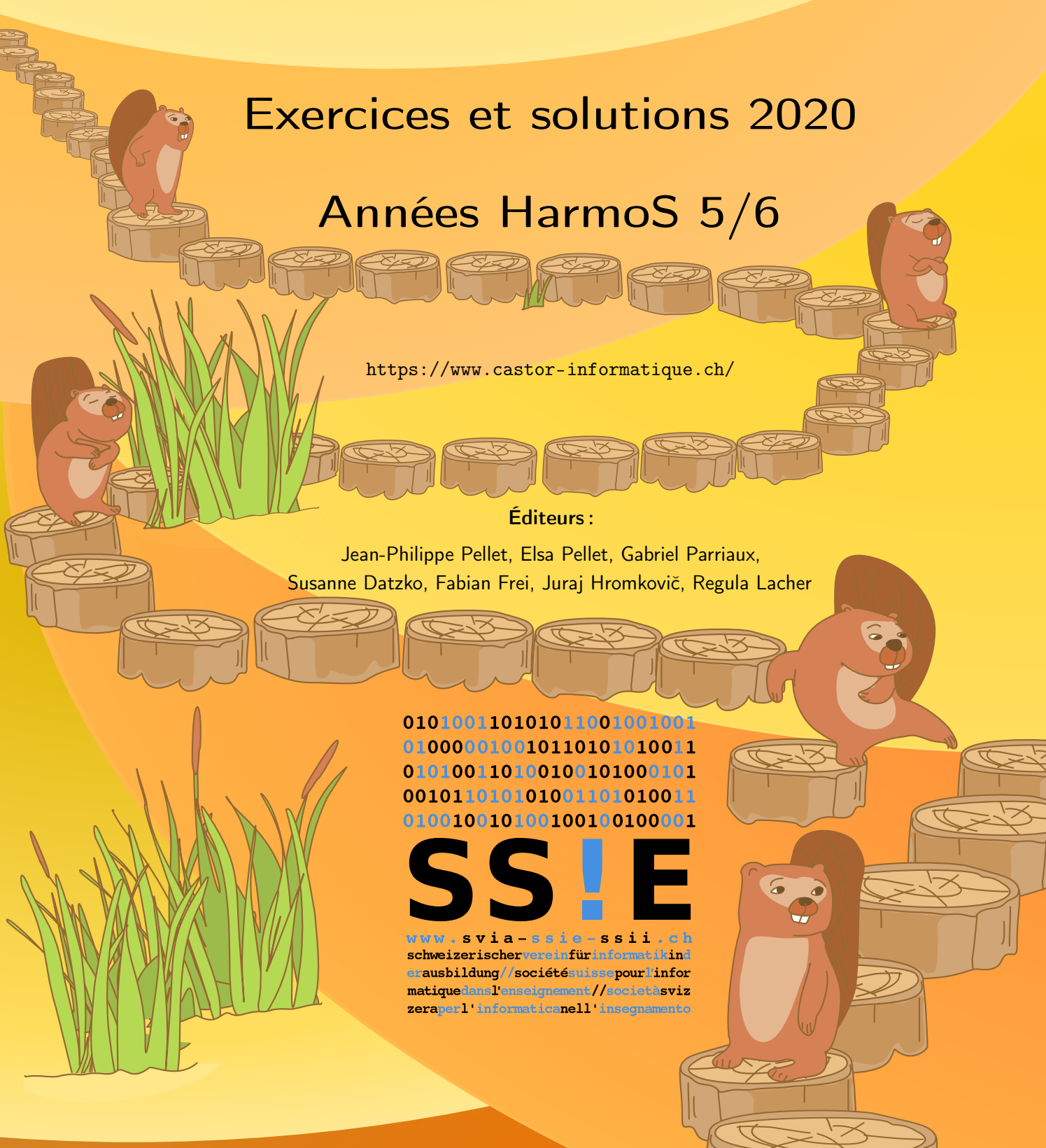
Éditeurs :

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

010100110101011001001001  
010000010010110101010011  
010100110100100101000101  
001011010101001101010011  
010010010100100100100001

**SS!E**

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







# Ont collaboré au Castor Informatique 2020

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

Cheffe de projet : Nora A. Escherle

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

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

Valentina Dagienė : Bebras.org

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

Wilfried Baumann, Anoki Eischer : Österreichische Computer Gesellschaft

Gerald Futschek, Florentina Voboril : Technische Universität Wien

Zsuzsa Pluhár : ELTE Informatikai Kar, Hongrie

Michal Winzcer : Université Comenius de Bratislava, Slovaquie

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

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

Chris Roffey : Université d'Oxford, Royaume-Uni

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

Hanspeter Erni : Direction, école secondaire de Rickenbach

Gabriel Thullen : Collège des Colombières

Beat Trachsler : Kantonsschule Kreuzlingen

Christoph Frei : Chragokyberneticks (Logo Castor Informatique Suisse)

Dr. Andrea Leu, Maggie Winter, Brigitte Manz-Brunner : SenarcLens Leu + Partner AG

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

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



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

Le Castor Informatique 2020 a été réalisé par la Société Suisse de l'Informatique dans l'Enseignement SSIE et soutenu par la Fondation Hasler.

## HASLERSTIFTUNG

Cette brochure a été produite le 20 janvier 2026 avec le système de composition de documents  $\text{\LaTeX}$ . 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<sup>er</sup> décembre 2020.



Les exercices sont protégés par une licence Creative Commons Paternité – Pas d'Utilisation Commerciale – Partage dans les Mêmes Conditions 4.0 International. Les auteur·e·s sont cité·e·s en p. 32.



# Préambule

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

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

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

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

Le concours Castor Informatique 2020 a été fait pour cinq tranches d'âge, basées sur les années scolaires :

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

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

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

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

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



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

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

Les réponses de nombreux exercices étaient affichées dans un ordre établi au hasard. Certains exercices ont été traités par plusieurs tranches d'âge.

**Pour de plus amples informations :**

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

Castor Informatique

Gabriel Parriaux

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

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



# Table des matières

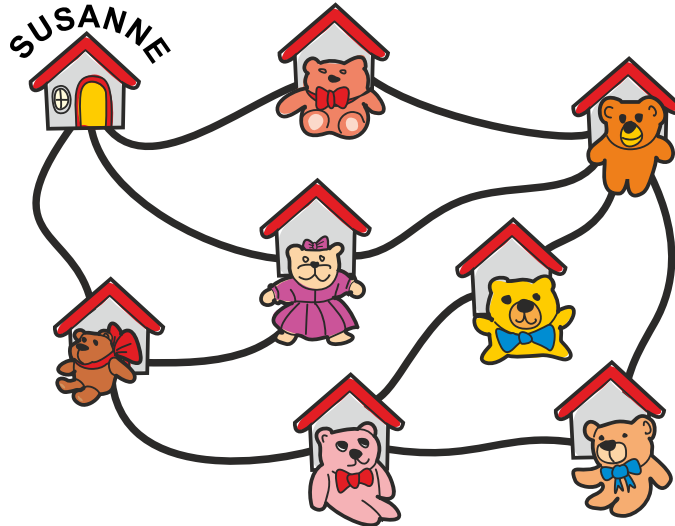
Ont collaboré au Castor Informatique 2020 . . . . .	i
Préambule . . . . .	iii
Table des matières . . . . .	v
1. Chasse à l'ours . . . . .	1
2. La pièce de théâtre . . . . .	5
3. Arrosage . . . . .	9
4. Chiffres secrets . . . . .	13
5. Sudoku boisé 3×3 . . . . .	15
6. Visite de musée . . . . .	19
7. Troc au château . . . . .	23
8. Prochain arrêt, gare! . . . . .	27
9. Piles de troncs d'arbres . . . . .	29
A. Auteur-e-s des exercices . . . . .	32
B. Sponsoring : Concours 2020 . . . . .	33
C. Offres ultérieures . . . . .	35



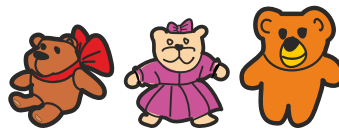


# 1. Chasse à l'ours

Dans le quartier de Susanne, on peut voir les nounours suivants devant les maisons :



Susanne a fait un tour depuis chez elle en passant devant exactement quatre maisons. Elle n'a pas passé deux fois devant la même maison. Elle n'a pas vu le nounours devant l'une des maisons. Les trois autres nounours étaient :



Quel est le nounours que Susanne n'a pas vu ?

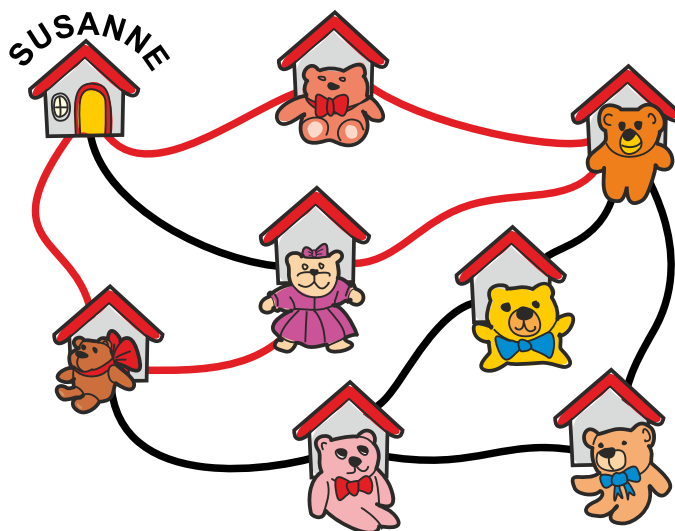
- A)  B)  C)  D) 



## Solution

La bonne réponse est C) 🧸.

Susanne doit être passée devant les maisons avec les trois nounours 🧸, 🧸 et 🧸 en faisant son tour. Ces trois nounours sont directement reliés par un chemin. Le premier nounours 🧸 est directement après sa maison. À la fin de ce chemin, elle se trouve près du troisième nounours 🧸. Depuis là, il n'y a qu'un chemin qui va chez elle en ne passant devant qu'une seule quatrième maison, et c'est le chemin qui passe devant le nounours 🧸. D'autres chemins possibles passent devant au moins deux autres nounours, et elle n'a passé que devant quatre maisons. La carte suivante montre le chemin :



Susanne peut faire son tour dans les deux directions, cela ne change rien.

## C'est de l'informatique !

Des textes à trous en cours de français, des exercices de mathématiques avec des champs vides ou une chasse à l'ours à laquelle il manque une image : ce sont tous des exercices dans lesquels on recherche une information manquante. Les exercices sont construits (ou *structurés*) de façon à ce que l'information manquante puisse être trouvée par *raisonnement logique* ou *déduction*.

Ce genre de choses arrivent fréquemment en informatique. Il peut y avoir des erreurs lors de la transmission ou de la sauvegarde de données. C'est pour cela que l'on travaille avec des méthodes qui *détection* ou même *corrige les erreurs*. Si l'erreur n'est pas trop importante, on peut le faire en enregistrant volontairement plus d'informations que nécessaire. Dans cet exercice, il s'agit de la carte et du fait que Susanne passe devant exactement quatre nounours. Ainsi, elle peut trouver l'information manquante, c'est-à-dire quel nounours elle n'a pas vu.

En 2020, de telles chasses aux ours lors desquelles des peluches étaient cachées aux fenêtres de différentes maisons ont été organisées dans plusieurs pays du monde. Cela permettait aux enfants de jouer ensemble à cacher et découvrir malgré les règles de distanciation durant la pandémie de coronavirus. L'idée de jouer à faire une chasse à l'ours vient à l'origine du livre d'image « *We're Going*



on a *Bear Hunt* » de Michael Rosen (1989). En français, on connaît le livre et le jeu correspondant sous le nom de « La chasse à l'ours ».





## Mots clés et sites web













- Détection et correction d'erreurs : [https://fr.wikipedia.org/wiki/Code\\_correcteur](https://fr.wikipedia.org/wiki/Code_correcteur)
- Déduction logique : [https://fr.wikipedia.org/wiki/Déduction\\_logique](https://fr.wikipedia.org/wiki/Déduction_logique)
- Chasse à l'ours : <https://www.insider.com/coronavirus-pandemic-sparked-worldwide-bear-hunt-to-entertain-kids-2020-4>,  
<https://www.actualitte.com/article/zone-51/pendant-toute-la-duree-du-confinement-la-chasse-a-l-ours-est-ouverte/100109>,  
<https://www.youtube.com/watch?v=OgyI6ykDwds>





## 2. La pièce de théâtre

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

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

Quelle situation n'aura pas lieu ?

















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



## Solution

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

On peut y réfléchir pas à pas :

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

### Fin

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



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

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

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

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

## C'est de l'informatique !

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

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

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



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

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

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

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

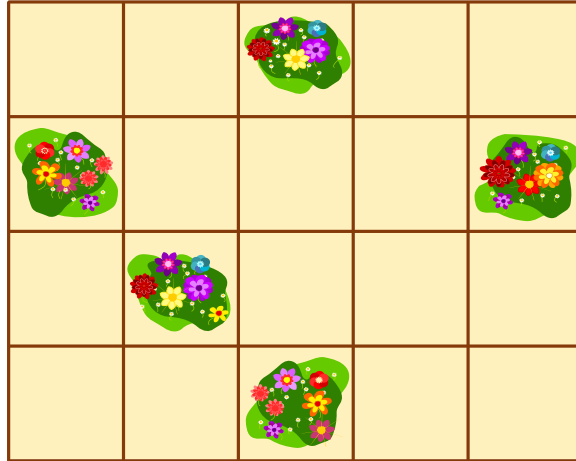
## Mots clés et sites web

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

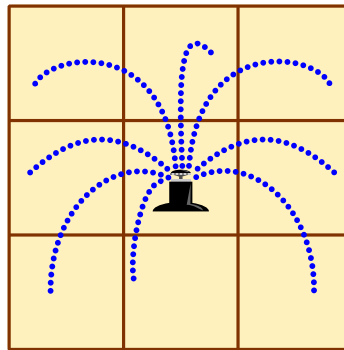


### 3. Arrosage

Le jardin de Daniel est composé de cases carrées. Il a planté des fleurs dans certaines de ces cases :



En été, il aimerait arroser ces fleurs avec des tourniquets arroseurs. Il ne peut pas mettre d'arroseur dans les cases avec des fleurs. Un arroseur arrose toutes les fleurs dans les 8 cases autour de lui :

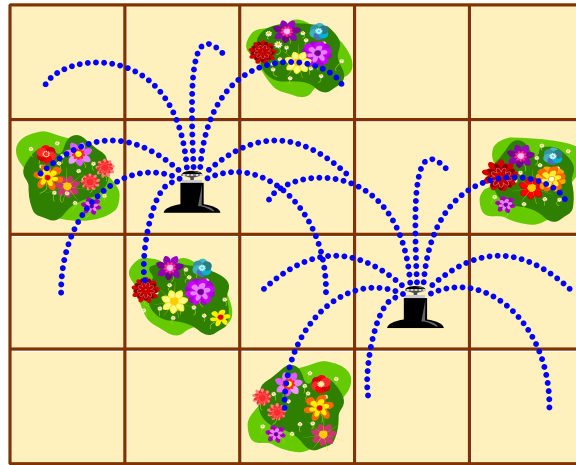


*Place aussi peu d'arroseurs que nécessaire pour arroser toutes les cases fleuries. Indique-le en cochant les cases correspondantes dans le jardin de Daniel.*



## Solution

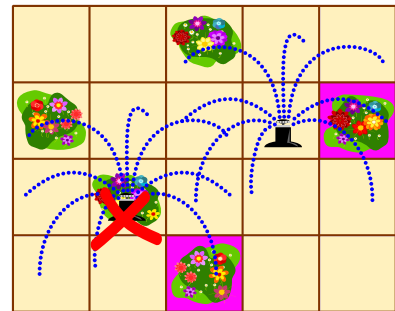
La solution suivante nécessite deux arroseurs pour arroser toutes les cases fleuries :



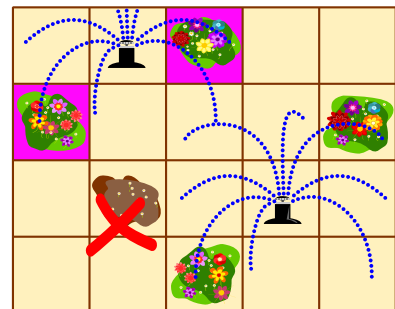
Entre la case fleurie tout à gauche et la cas fleurie tout à droite se trouvent trois cases. Un seul arroseur ne peut pas arroser deux cases si éloignées l'une de l'autre.

Il n'y a pas non plus d'autre solution que celle-ci en n'utilisant que deux arroseurs.

Pour arroser la case fleurie tout à droite et celle en bas au centre en même temps, il doit y avoir un arroseur là où il est placé dans la solution. S'il était placé plus haut pour arroser aussi la case fleurie en haut au centre, il ne pourrait plus arroser la case fleurie en bas au centre, et on ne pourrait pas arroser les trois cases fleuries restante avec un seul autre arroseur, car on ne peut pas en mettre dans une case fleurie.



Pour arroser la case fleurie tout à gauche et celle en haut au centre, un arroseur doit être mis soit comme dans la solution, soit une case plus haut. Si cet arroseur doit aussi arroser la case fleurie de la deuxième colonne depuis la gauche et troisième ligne depuis le haut, il ne peut pas être mis tout en haut.



## C'est de l'informatique !

Cet exercice est un problème d'*optimisation* typique : alors qu'il est clair que toutes les cases fleuries doivent être arrosées, le nombre d'arroseurs est variable et doit être le plus petit possible. Des problèmes d'optimisation similaires existent par exemple si l'on veut placer des stations de pompiers pour protéger une ville ou couvrir des maisons avec le réseau natel.



En informatique, on parle également de *problèmes de couverture*. Ces problèmes font partie d'une classe de problèmes très difficiles en informatique. Le placement d'un nombre minimal d'arroseurs dans cet exercice était encore très simple, mais la difficulté augmente tellement fortement avec le nombre de cases qu'on ne peut bientôt plus trouver de solution optimale en un temps raisonnable, même à l'aide d'ordinateurs.

Une possibilité dans de tels cas est de se satisfaire de solutions qui ne sont peut-être pas optimales mais qui sont quand même bonnes. Ça ne fait pas grande différence si l'on place 101 au lieu 100 stations de pompiers, ou 1000 antennes natel au lieu de seulement 990, mais cela rend souvent le problème beaucoup plus facile à résoudre.

## Mots clés et sites web

- Optimisation : [https://fr.wikipedia.org/wiki/Optimisation\\_\(mathématiques\)](https://fr.wikipedia.org/wiki/Optimisation_(mathématiques))
- Problème de couverture





## 4. Chiffres secrets

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

	-	=	≡	▷	▷
□	0	1	2	3	4
◁	5	6	7	8	9

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

	-	=	≡	▷	▷
□	0	1	2	3	4
◁	5	6	7	8	9

Voici la hutte de Cleverias :



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

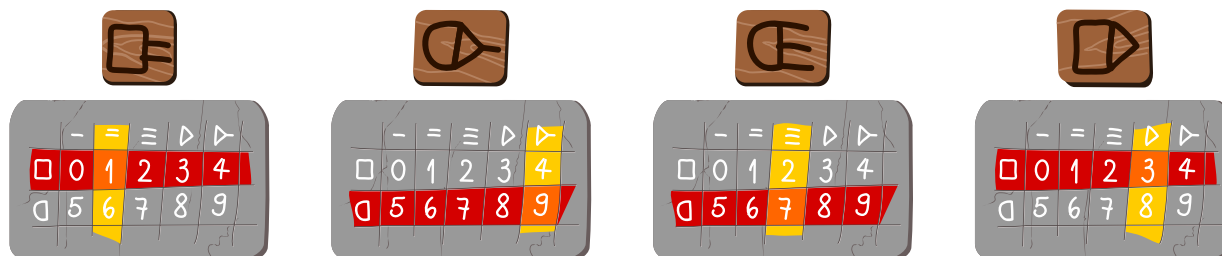
- A) 0978
- B) 1574
- C) 1923
- D) 1973
- E) 1993
- F) 2973
- G) 6378



## Solution

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

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



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

## C'est de l'informatique !

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

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

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

## Mots clés et sites web

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



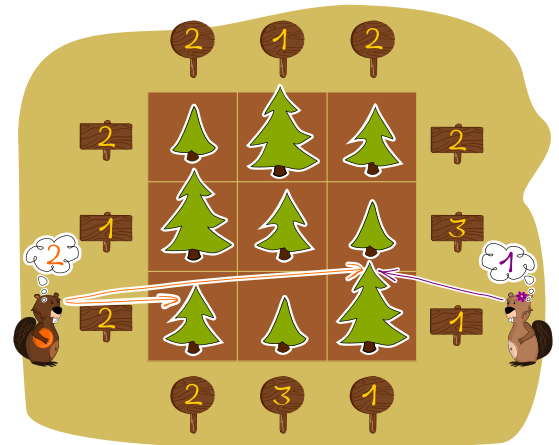
## 5. Sudoku boisé 3×3

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

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

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

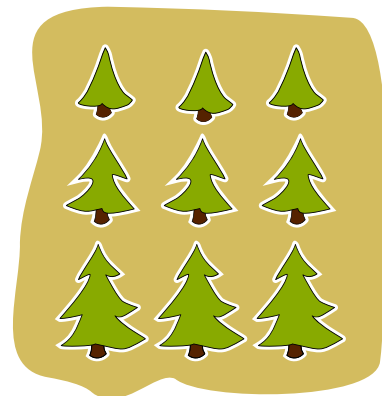
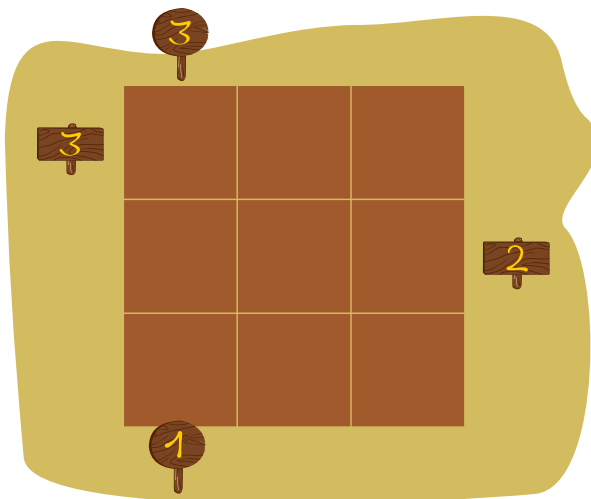
Les castors plantent à présent neuf sapins sur un champ de 3×3 cases, comme dans l'exemple à droite.



Pour cela, les règles suivantes s'appliquent :

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

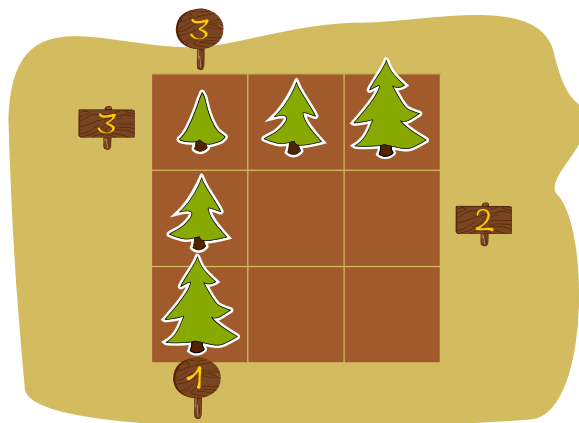
Écris dans chaque case la hauteur du sapin qui s'y trouve.





## Solution

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

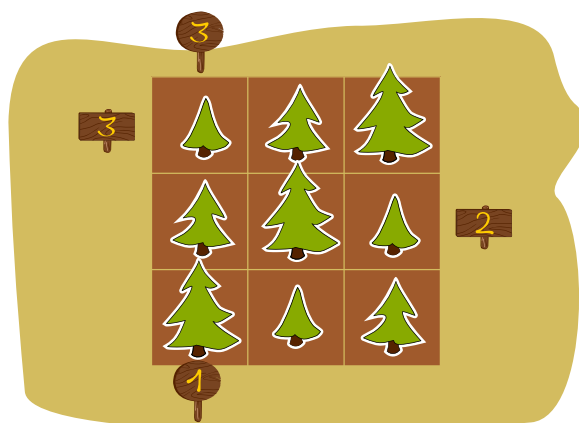


Le panneau avec le 2 à droite indique que l'on peut voir deux sapins depuis là, il doit donc y avoir un sapin de hauteur 3 au milieu et la ligne centrale est ainsi 2 () , 3 () , 1 () .

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

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

Voici la solution complète :



## C'est de l'informatique !

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

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



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

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

## Mots clés et sites web

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



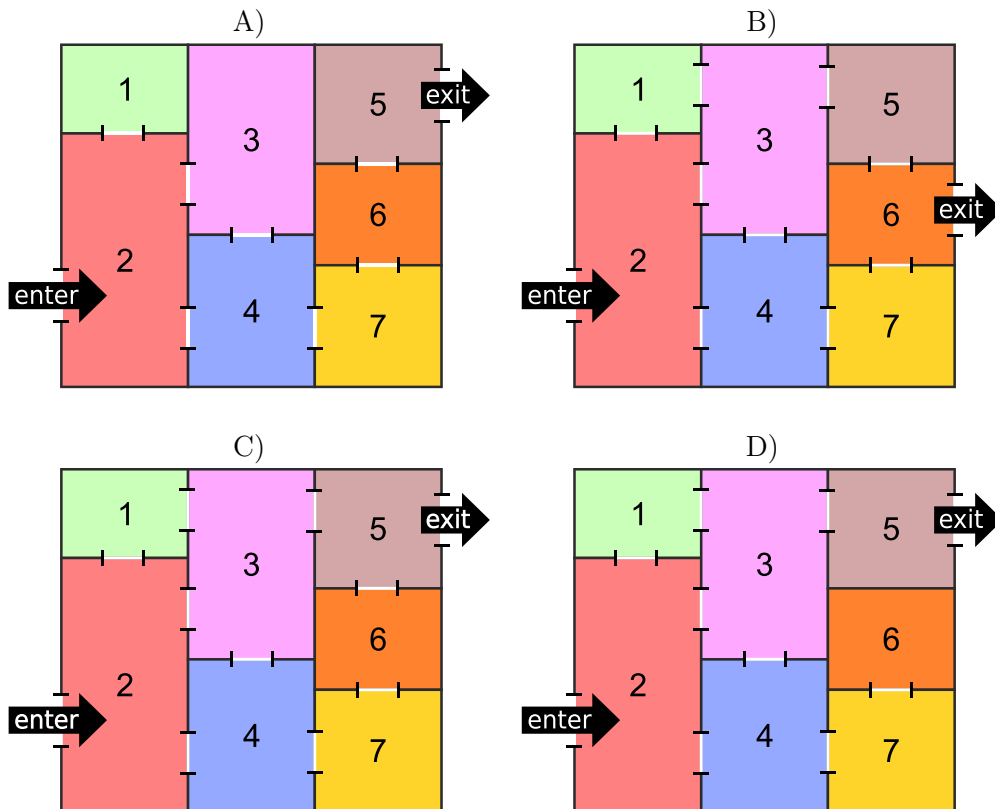


## 6. Visite de musée

Quatre plans au sol sont proposés pour la construction d'un nouveau musée. Chaque plan comporte les sept pièces 1 à 7. Les pièces doivent être arrangées de façon à ce que les visiteurs puissent visiter toutes les pièces sans passer deux fois par la même pièce.

Les visiteurs commencent la visite à l'entrée « enter » et quittent le musée par la sortie « exit ».

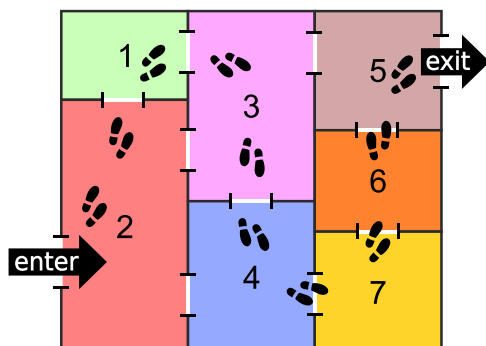
Quel plan au sol permet aux visiteurs d'entrer et de sortir de chaque pièce exactement une fois ?





## Solution

Seul le plan C permet aux visiteurs de n'entrer et de ne sortir qu'une seule fois de chaque pièce. L'ordre des pièces visitées est 2, 1, 3, 4, 7, 6, 5.



De manière générale, une telle visite n'est pas possible si n'importe laquelle des pièces ne possède qu'une entrée. L'explication est la suivante : si un visiteur entre dans cette pièce, il est obligé de retourner dans la pièce de laquelle il vient en ressortant, et ne respecte donc pas la règle de n'entrer qu'une fois dans chaque pièce.

Le plan A n'a qu'une entrée à la pièce 1.

Le plan D n'a qu'une entrée à la pièce 6.

Le plan B ne permet l'entrée dans la dernière pièce, la 6, que par la pièce 5 ou la pièce 7. Si le visiteur vient de la pièce 5, il peut entrer dans la pièce 7, mais ne peut atteindre la sortie qu'en retournant dans la pièce 6 (ou l'inverse), ce qui va à l'encontre des règles.

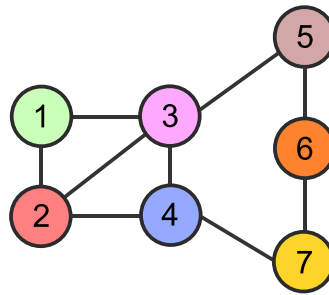
## C'est de l'informatique !

La plupart des enfants et des jeunes résolvent ce problème par tâtonnement sans utiliser de représentation abstraite supplémentaire. Pour cela, ils utilisent à un certain niveau la stratégie générale appelée *retour sur trace*. Ils reconnaissent tout au moins que l'on peut apprendre d'essais infructueux et que l'on peut, dans ce cas, revenir en arrière pour essayer une autre possibilité. Ils sont également confrontés au concept du *non-déterminisme*, car il y a souvent plusieurs possibilités à choix.

Cet exercice est un exemple d'un problème connu en informatique : la recherche d'un *chemin hamiltonien*. Dans une représentation discrète du plan au sol par un *graphe*, chaque pièce est représentée par un *nœud* et chaque porte entre deux pièces par une *arête* entre les deux nœuds correspondants.



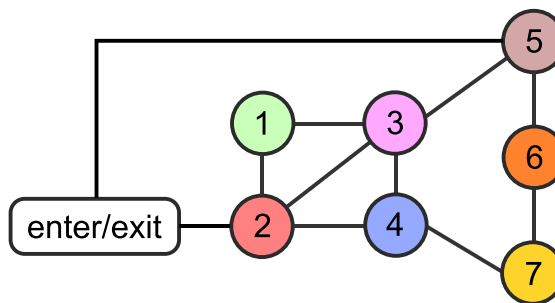
Voici la représentation abstraite du plan de l'exercice :



Il s'agit à présent de trouver dans ce graphe un chemin ayant les propriétés suivantes :

1. Le chemin commence au nœud 2 (« enter »).
2. Le chemin finit au nœud 5 (« exit »).
3. Le chemin passe exactement une fois par chaque nœud.

Si l'on représente l'espace extérieur par un nœud, ceci correspond à la recherche d'un cycle hamiltonien (un tour) dans lequel on passe également une seule fois par chaque nœud avant de terminer au nœud de départ.



## Mots clés et sites web

- Théorie de graphes, nœud, arête : [https://fr.wikipedia.org/wiki/Théorie\\_des\\_graphes](https://fr.wikipedia.org/wiki/Théorie_des_graphes)
- Chemin hamiltonien : [https://fr.wikipedia.org/wiki/Graphe\\_hamiltonien](https://fr.wikipedia.org/wiki/Graphe_hamiltonien)





































## 7. Troc au château

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

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

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

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

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

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

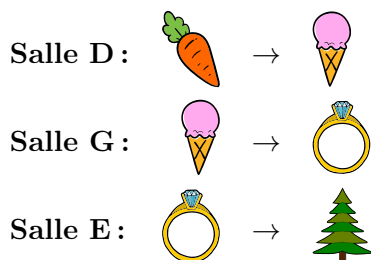


## Solution

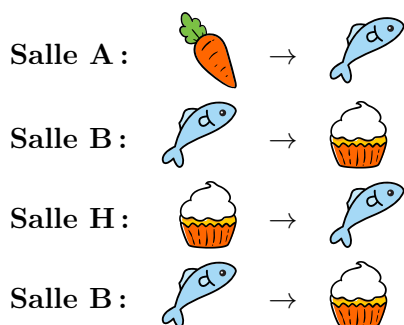
La bonne réponse est A) DGE. D'abord la salle D, puis la salle G et finalement la salle E.

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

Voici le déroulement complet :



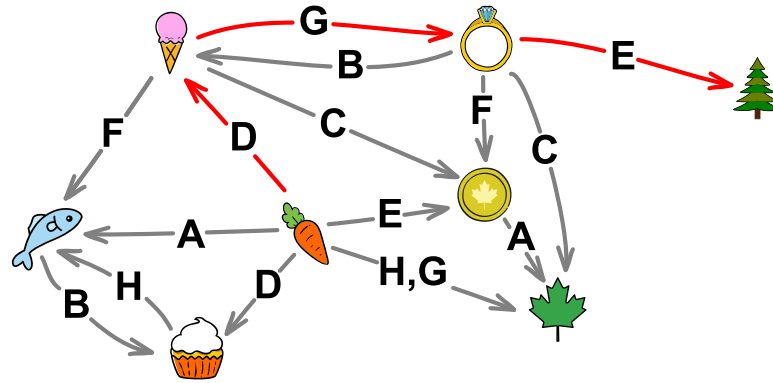
Il y a deux stratégies appropriées pour trouver le bon ordre dans lequel visiter les salles. La première essaie de prendre toutes les possibilités de troc en considération. Elle commence avec le premier troc, lors duquel on peut échanger la carotte dans cinq salles (A, D, E, G et H) contre six objets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



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

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

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



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

## C’est de l’informatique !

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

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

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

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

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

## Mots clés et sites web



- Théorie des graphes : [https://fr.wikipedia.org/wiki/Théorie\\_des\\_graphes](https://fr.wikipedia.org/wiki/Théorie_des_graphes)
- Problème d’accessibilité : [https://fr.wikipedia.org/wiki/Problème\\_d’accessibilité](https://fr.wikipedia.org/wiki/Problème_d’accessibilité)

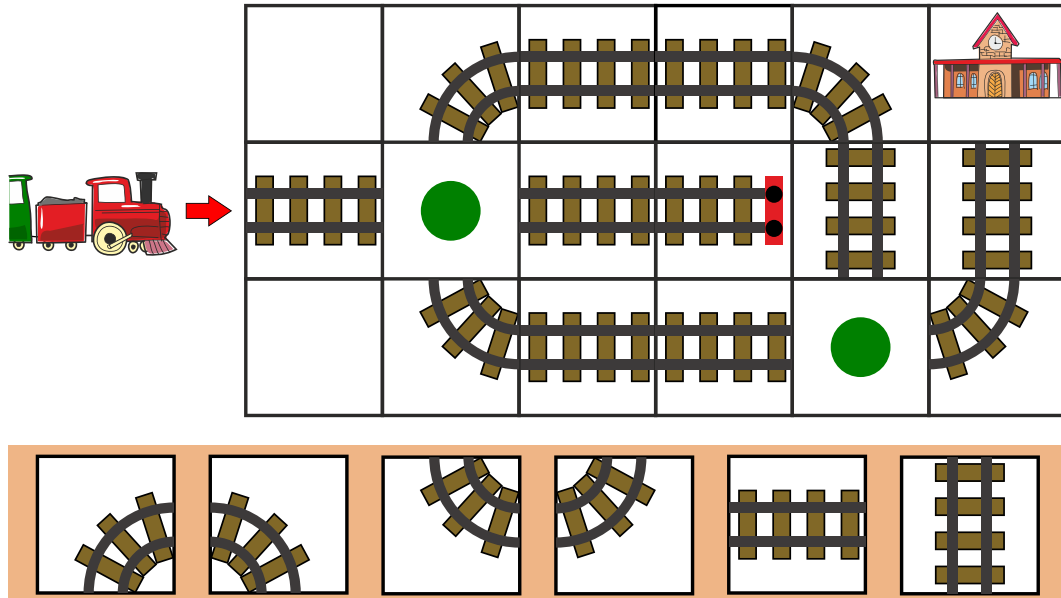


- Parcours en profondeur :  
[https://fr.wikipedia.org/wiki/Algorithme\\_de\\_parcours\\_en\\_profondeur](https://fr.wikipedia.org/wiki/Algorithme_de_parcours_en_profondeur)
- Parcours en largeur :  
[https://fr.wikipedia.org/wiki/Algorithme\\_de\\_parcours\\_en\\_largeur](https://fr.wikipedia.org/wiki/Algorithme_de_parcours_en_largeur)



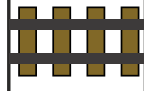
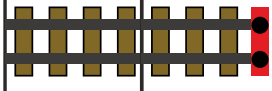

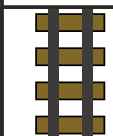




## 8. Prochain arrêt, gare !

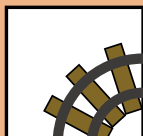
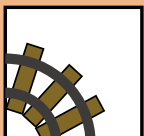
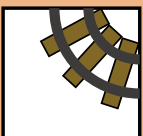
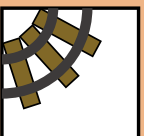
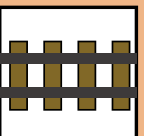
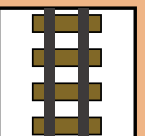
Choisis les bons rails pour les deux cases avec les points verts de façon à ce que le train  puisse aller à la gare .



The puzzle consists of a 3x5 grid. The train starts on the left, moving right. The goal is to reach the station on the right. The grid contains various track pieces and two green circles indicating missing pieces. A legend below shows the available track pieces.

					
		●			
				●	

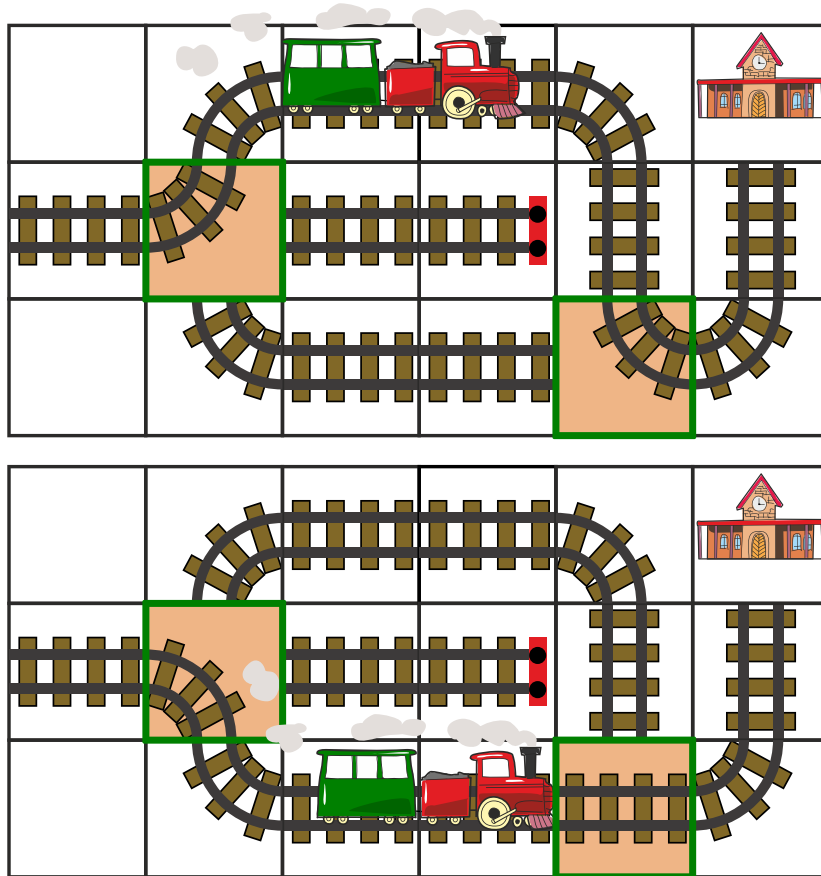
Legend:

					
------------------------------------------------------------------------------------	------------------------------------------------------------------------------------	------------------------------------------------------------------------------------	------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------



## Solution

Ce problème a les deux solutions suivantes :



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

## C'est de l'informatique !

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

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

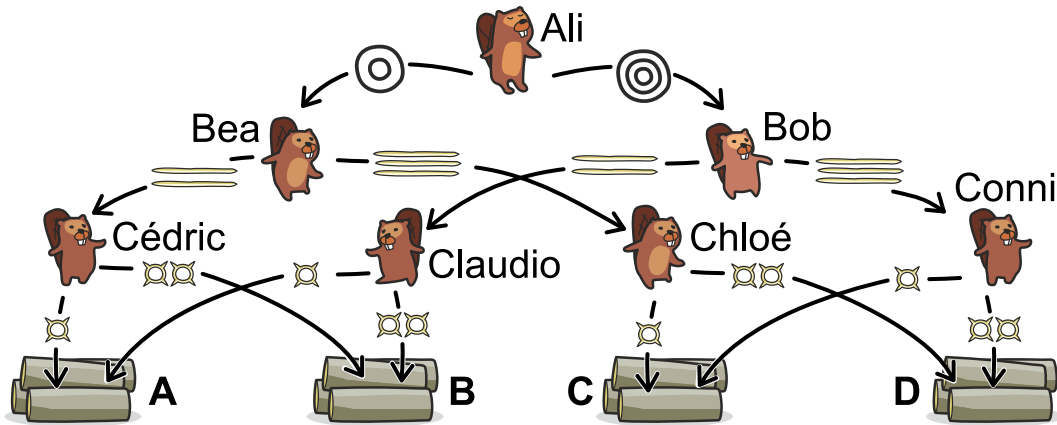
## Mots clés et sites web

- Programme
- Instruction : [https://fr.wikipedia.org/wiki/Instruction\\_informatique](https://fr.wikipedia.org/wiki/Instruction_informatique)
- <https://fr.wikipedia.org/wiki/Algorithme>



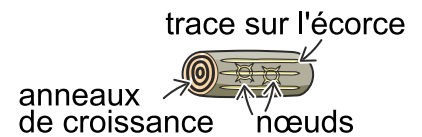
## 9. Piles de troncs d'arbres

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



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

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



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



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



## Solution

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

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

Sur la pile **A** :

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

Sur la pile **B** :

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

Sur la pile **C** :

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

Sur la pile **D** :

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

## C'est de l'informatique !

Cet exercice touche à plusieurs concepts informatiques.

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

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

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

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



## Mots clés et sites web

- Arbre de décision : [https://fr.wikipedia.org/wiki/Arbre\\_de\\_d%C3%A9cision](https://fr.wikipedia.org/wiki/Arbre_de_d%C3%A9cision)
- Classification : [https://fr.wikipedia.org/wiki/Classement\\_automatique](https://fr.wikipedia.org/wiki/Classement_automatique)



## A. Auteur·e·s des exercices

- |                                                                                                                |                                                                                                         |
|----------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------|
|  Serge Adam                   |  Vu Van Luan           |
|  Wilfried Baumann             |  Hamed Mohebbi         |
|  Carlo Bellettini             |  Kwangsik Moon         |
|  Linda Björk Bergsveinsdóttir |  Xavier Muñoz          |
|  Daniela Bezáková             |  Tom Naughton          |
|  Lucia Budinská               |  Gabriel Parriaux      |
|  Sarah Chan                   |  Elsa Pellet           |
|  Marios O. Choudary           |  Jean-Philippe Pellet  |
|  Valentina Dagienė            |  Margot Phillipps      |
|  Christian Datzko             |  Wolfgang Pohl         |
|  Susanne Datzko             |  Pedro Ribeiro       |
|  Lidia Feklistova           |  Peter Rossmann      |
|  Fabian Frei                |  Vipul Shah          |
|  Husnul Hakim               |  Peter Tomcsányi     |
|  Juraj Hromkovič            |  Monika Tomcsányiová |
|  Alisher Ikramov            |  Jiří Vaníček        |
|  Ungyeol Jung               |  Michael Weigend     |
|  Vaidotas Kinčius           |  Jonas Winckler      |
|  Regula Lacher              |  Michal Winczer      |



## B. Sponsoring : Concours 2020

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

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



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



**Kanton Zürich**  
Volkswirtschaftsdirektion  
Amt für Wirtschaft und Arbeit

Standortförderung beim Amt für Wirtschaft und Arbeit Kanton Zürich



i-factory (Musée des transports, Lucerne)



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



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



<https://educatec.ch/>  
educaTEC



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



AUSBILDUNGS- UND BERATUNGSZENTRUM  
FÜR INFORMATIKUNTERRICHT

<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

Scuola universitaria professionale  
della Svizzera italiana



<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

010100110101011001001001  
010000010010110101010011  
010100110100100101000101  
001011010101001101010011  
010010010100100100100001

**SS!E**

[www.svia-ssie-ssii.ch](http://www.svia-ssie-ssii.ch)  
schweizerischerverein fürinformatikind  
erausbildung//sociétésuissepourl'infor  
matique dansl'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.