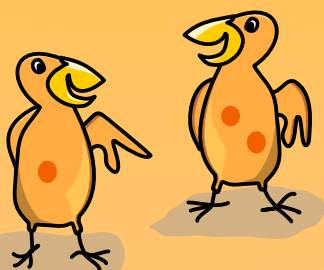




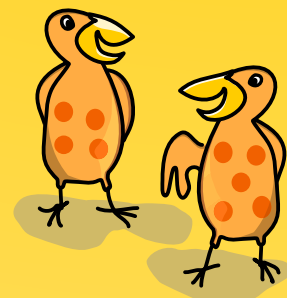
**INFORMATIK-BIBER SCHWEIZ  
CASTOR INFORMATIQUE SUISSE  
CASTORO INFORMATICO SVIZZERA**

**Exercices et solutions 2021**

**Années HarmoS 13/14/15**



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



**Éditeurs :**

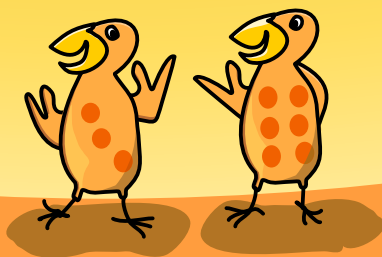
Susanne Datzko, Elsa Pellet, Jean-Philippe Pellet,  
Fabian Frei, Gabriel Parriaux



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 2021

Masiar Babazadeh, Susanne Datzko, Fabian Frei, Martin Guggisberg, 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  
Bernadette Spieler : Pädagogische Hochschule Zürich

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

Valentina Dagienė, Tomas Šiaulyš, Vaidotas Kinčius : Bebras.org

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

Wilfried Baumann, Liam Baumann, Anoki Eischer, Thomas Galler, Benjamin Hirsch, Martin Kandlhofer, Katharina Resch-Schobel : 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 : UK Bebras Administrator, Royaume-Uni

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

Hanspeter Erni : Direction, école secondaire de Rickenbach

Christoph Frei : Chragokyberneticks (Logo Castor Informatique Suisse)

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

*Ces brochures sont dédiées à la mémoire de Martin Guggisberg.*

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 2021 a été réalisé par la Société Suisse pour l'Informatique dans l'Enseignement (SSIE) et soutenu par la Fondation Hasler.

## HASLERSTIFTUNG

Cette brochure a été produite le 24 août 2022 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 2021.



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. 52.



# 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 2021 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 pour 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 2021 . . . . .	i
Préambule . . . . .	iii
Table des matières . . . . .	v
1. Bibliothèque . . . . .	1
2. Pavage de Truchet . . . . .	3
3. Villages isolés . . . . .	5
4. Couches de liquides . . . . .	9
5. Un, deux, trois, partez, feu ! . . . . .	13
6. Toiles d'araignée . . . . .	17
7. Pile de fruits . . . . .	21
8. Petit singe . . . . .	25
9. Sacrés pupitres . . . . .	29
10. Plan de travail . . . . .	33
11. Nombres en billes . . . . .	35
12. Travail d'équipe . . . . .	39
13. Compter avec les muscles . . . . .	43
14. Beaver Sort . . . . .	47
15. Les clans de Castorland . . . . .	49
A. Auteur-e-s des exercices . . . . .	52
B. Sponsoring: Concours 2021 . . . . .	53
C. Offres ultérieures . . . . .	55







# 1. Bibliothèque

Susi est à la bibliothèque des castors avec Tim. Ils veulent emprunter un livre appelé « Construire de grands barrages ».

Tim va vers l'étagère 1, regarde dans la rangée 4 et sort le livre du casier 0. Susi est impressionnée. Tim explique à Susi comment on détermine l'emplacement d'un livre :

On prend la première lettre de chaque mot du titre du livre et détermine sa position dans l'alphabet. On additionne ensuite ces positions après avoir multiplié par 3 la valeur obtenue à l'addition précédente.

Le livre désiré donne 140. L'emplacement du livre est ainsi tout de suite clair.

a	b	c	d	e	f	g	h	i	j	k	l	m
1	2	3	4	5	6	7	8	9	10	11	12	13
n	o	p	q	r	s	t	u	v	w	x	y	z
14	15	16	17	18	19	20	21	22	23	24	25	26
Construire de grands barrages												
((3 × 3 + 4) × 3 + 7) × 3 + 2												

Susi écrit maintenant les calculs correspondant à ses livres préférés, mais elle a fait une erreur pour l'un d'entre eux.

Lequel ?

A)  $((6 \times 3 + 4) \times 3 + 3) \times 3 + 5$

B)  $((1 \times 3 + 4) + 2) \times 3 + 2$

C)  $((7 \times 3 + 4) \times 3 + 7) \times 3 + 6$

D)  $((3 \times 3 + 5) \times 3 + 7) \times 3 + 2$



## Solution

Susi a presque tout fait juste : elle a toujours additionné les bonnes positions et a multiplié les résultats intermédiaires par 3 — avec une exception : elle a oublié une multiplication dans la réponse B.

$$\begin{array}{|c|} \hline \text{ABC du bon bûcheron} \\ \hline ((1 \times 3 + 4) \times 3 + 2) \times 3 + 2 \\ \hline \end{array}$$

## C'est de l'informatique !

Le système d'expressions correspondant à l'emplacement des livres permet aux visiteurs de la bibliothèque de déterminer l'endroit exact où les livres sont rangés. Comme ça, personne ne doit chercher longtemps. La bibliothèque et les visiteurs doivent cependant faire attention à une chose : différents livres peuvent avoir la même expression et donc le même résultat. Par exemple, les livres « Guide des grands fleuves » et « Guide des grandes familles » sont dans le même casier. Les casiers doivent donc être assez grands ou pouvoir être agrandis selon les besoins.

C'est aussi une bonne idée que l'endroit auquel les données sont enregistrées dans un ordinateur puisse être calculé directement à partir des données elles-mêmes. Pour cela, des *fonctions de hachage* ont été développées en informatique : des fonctions mathématiques qui calculent une valeur à partir du contenu des données ou d'une partie des données, valeur qui indique directement l'emplacement mémoire — comme dans cet exercice du castor. De bonnes fonctions de hachage minimisent le nombre de fois où la même valeur est calculée. Si un tel conflit a lieu, l'informatique dispose de bonnes méthodes pour le gérer.

## Mots clés et sites web

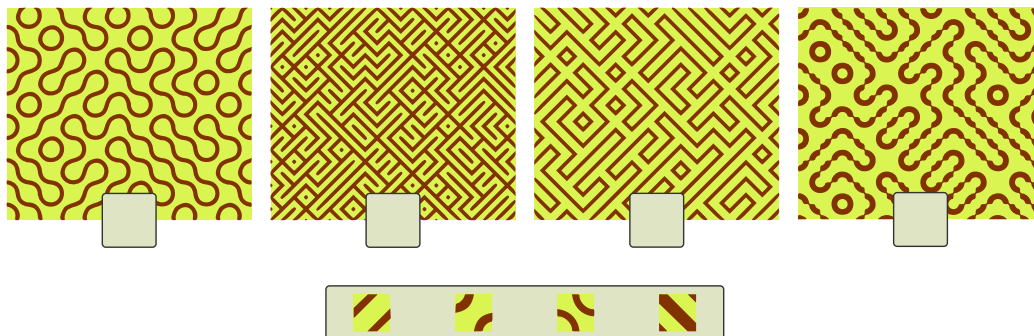
- Fonction de hachage : [https://fr.wikipedia.org/wiki/Fonction\\_de\\_hachage](https://fr.wikipedia.org/wiki/Fonction_de_hachage)
- Table de hachage : [https://fr.wikipedia.org/wiki/Table\\_de\\_hachage](https://fr.wikipedia.org/wiki/Table_de_hachage)



## 2. Pavage de Truchet

Les motifs suivants ont été créés en n'utilisant qu'un seul type de pavé. Les images des pavés individuels sont agrandies.

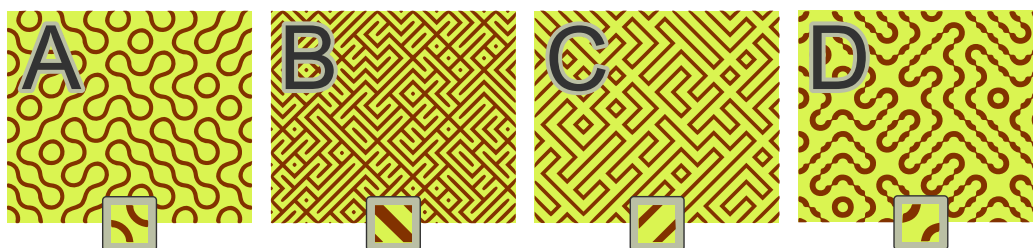
*Assigne chaque pavé au motif correspondant.*



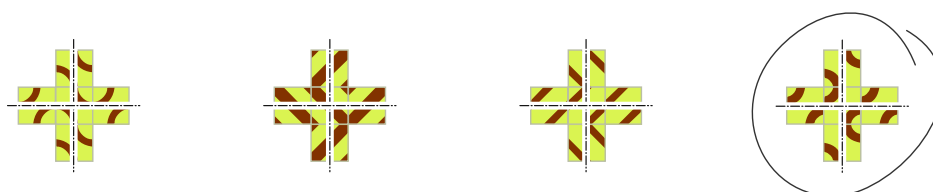






## Solution

Voici la bonne attribution :



En mettant cinq mêmes pavés côte à côte et en comparant les différents pavés, on voit de claires différences :



Le pavé  est le seul pavé dont les quatre côtés ne vont pas exactement ensemble. C'est le seul moyen pour obtenir des lignes de largeurs différentes comme dans le motif D. Le pavé  est le seul avec lequel on peut créer des points carrés comme sur le motif B, en mettant bout à bout quatre coins avec un triangle. De plus, il a la plus grande proportion de brun par rapport au jaune, comme le motif B. Il ne reste donc que le pavé  pour former le motif A aux formes arrondies, alors que les lignes droites du motif C ne peuvent être créées qu'avec le pavé .

## C'est de l'informatique !

Ces pavés sont nommés d'après Sébastien Truchet (\* 1657; † 1729), qui en a développé différentes variantes. Les pavés ayant quatre côtés pareils forment un sous-ensemble des pavés de Truchet (mais les pavés de Truchet ne doivent pas forcément avoir quatre côtés pareils, comme dans trois des motifs de cet exercice). Le fait que des motifs complets peuvent être générés à partir d'éléments très simples est une propriété intéressante que l'on rencontre souvent en informatique. Les pavés de Truchet sont étudiés en mathématiques et en informatique, et sont utilisés dans les jeux vidéo pour créer des labyrinthes et des décors.

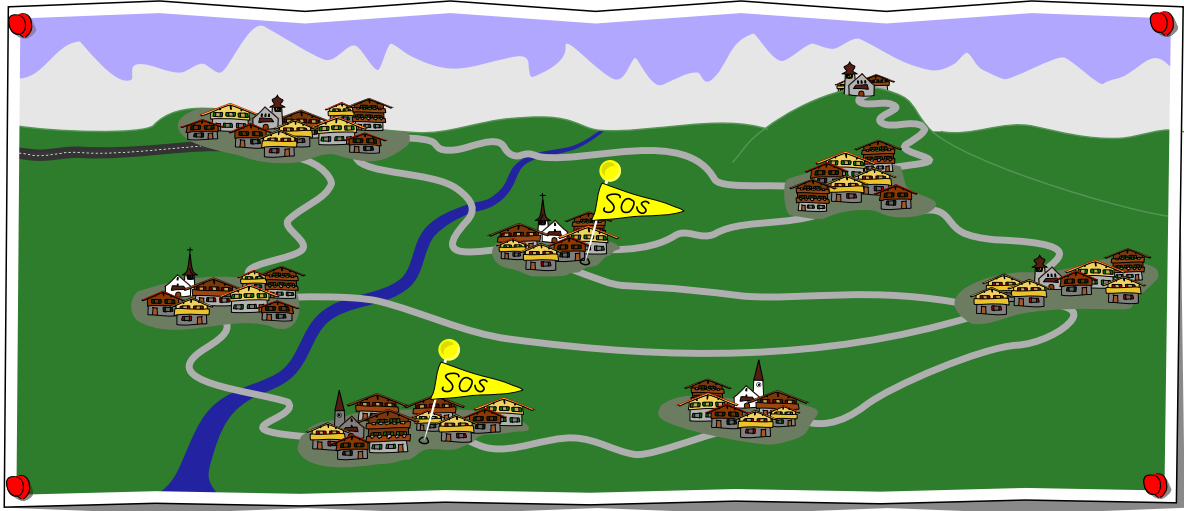
## Mots clés et sites web

- Pavés de Truchet : [https://en.wikipedia.org/wiki/Truchet\\_tiles](https://en.wikipedia.org/wiki/Truchet_tiles)
- Sébastien Truchet : [https://fr.wikipedia.org/wiki/Sébastien\\_Truchet](https://fr.wikipedia.org/wiki/Sébastien_Truchet)






### 3. Villages isolés

Plusieurs villages de montagne sont approvisionnés par la grande ville grâce au réseau de routes suivant :



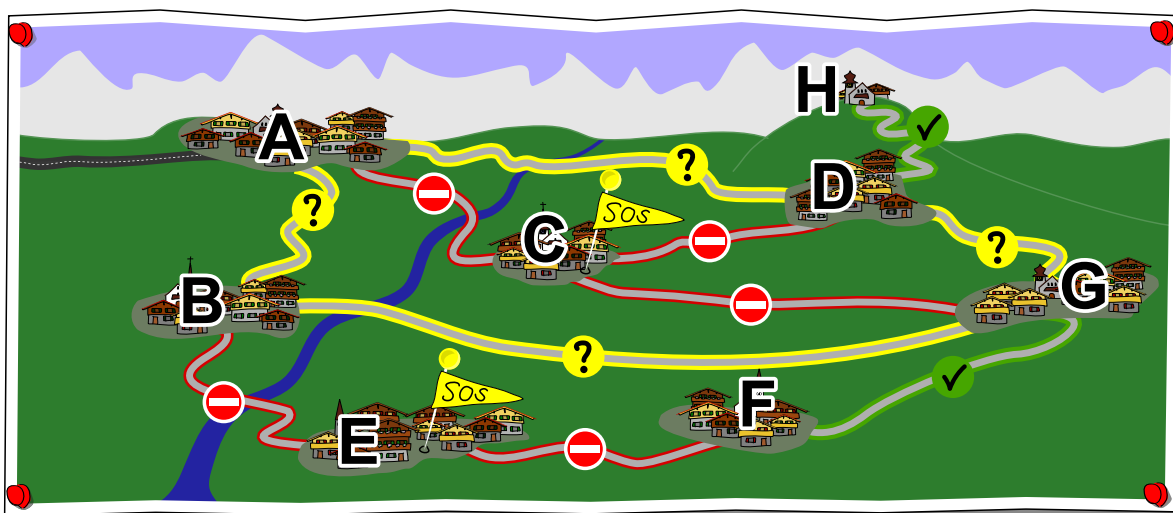
Après une tempête, plusieurs de ces villages ne sont plus du tout accessibles et le signalent avec un drapeau « SOS ». On peut en déduire que certaines des routes sont bloquées.

Indique pour chaque route du réseau entre les villages si elle est (1) bloquée , (2) ouverte  ou (3) si on ne peut pas le savoir sans informations supplémentaires .



## Solution

La carte suivante montre ce que l'on sait sur les connexions dans le réseau routier :



Nous commençons par déterminer quelles routes sont bloquées. Les deux routes menant au village E sont bloquées, car le village E serait accessible si ce n'était pas le cas. De même, les trois routes menant au village C sont bloquées, car il serait accessible sinon.

Ensuite, nous cherchons les routes qui doivent être ouvertes. La route entre les villages G et F doit être ouverte, car le village F ne serait pas accessible autrement, la route entre les villages F et E étant bloquée. La route entre l'église H et le village D doit aussi être ouverte, vu que H est accessible et qu'on ne peut y accéder qu'en passant par D.

Il ne reste que les routes qui sont peut-être accessibles. Comme les villages B, G et D sont reliés plusieurs fois au village A, on ne peut pas déterminer quelles routes parmi celles qui restent sont ouvertes. On pourrait par exemple accéder au village B par le village A, mais aussi par le village G. C'est la même chose pour le village D. On peut accéder au village G par le village B ou par le village D. N'importe laquelle des routes dans le circuit A - B - G - D - A pourrait donc être fermée et les quatre villages resteraient accessibles.

## C'est de l'informatique !

Comme dans les réseaux routiers, il peut aussi y avoir des connexions fautives, surchargées ou défectueuses dans les réseaux informatiques. Pour éviter les pannes, on planifie souvent des mesures de sécurité, comme par exemple plusieurs connexions menant au même endroit. On appelle cela la *redondance*.

La correction de dysfonctionnements d'un système est une tâche que les informaticiens doivent souvent effectuer ; pas seulement dans les réseaux informatiques, mais aussi lors de développement de programmes. Pour corriger un problème, il faut identifier sa source exacte ; c'est un processus qui se fait en général étape par étape. Certains programmeurs disent que l'on ne peut jamais trouver toutes les erreurs et tous les bugs d'un programme.



## Mots clés et sites web

- Redondance : [https://fr.wikipedia.org/wiki/Redondance\\_\(ingénierie\)](https://fr.wikipedia.org/wiki/Redondance_(ingénierie))
- Bug : [https://fr.wikipedia.org/wiki/Bug\\_\(informatique\)](https://fr.wikipedia.org/wiki/Bug_(informatique))

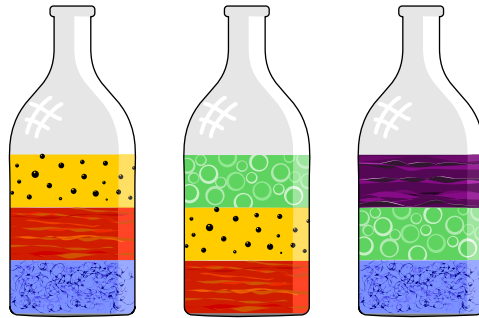




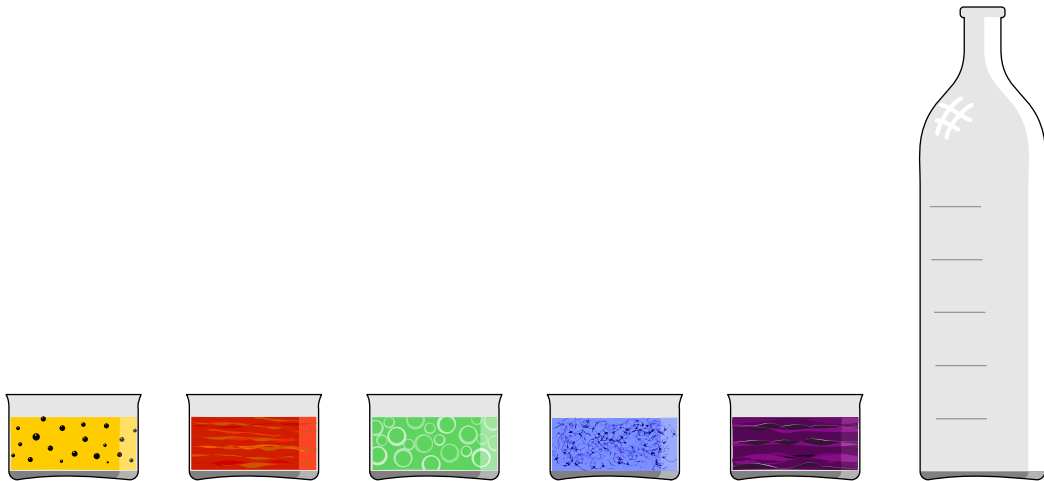


## 4. Couches de liquides

Marc a des des bouteilles qui contiennent chacune trois liquides formant des couches superposées. Il sait que les liquides à densité plus faible se mettent toujours au dessus des liquides à densité plus forte. Il aimerait maintenant voir à quoi une grande bouteille dans laquelle on met tous les liquides colorés ressemble.



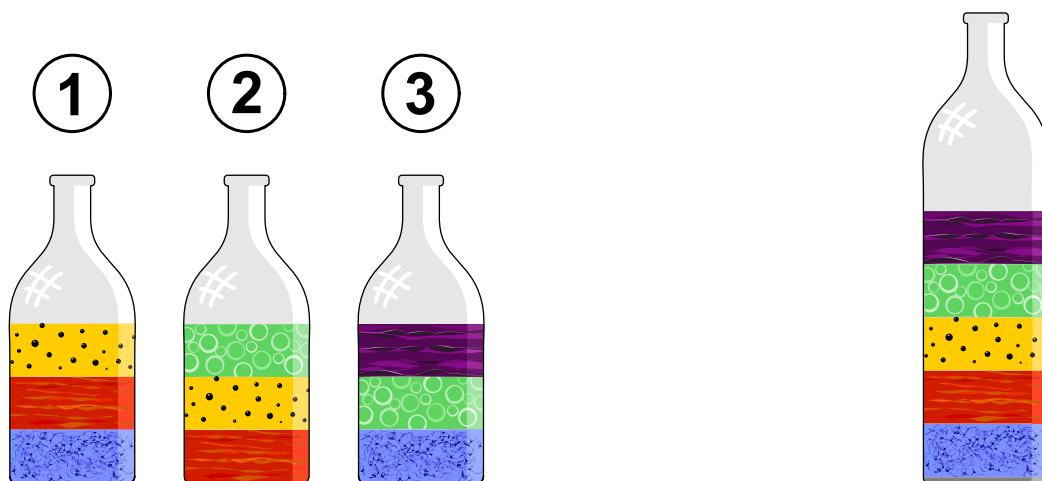
Arrange les cinq couches de liquides colorés dans la bouteille dans leur ordre final.





## Solution

L'image montre le bon arrangement des couches de liquide dans la grande bouteille.



Tu trouves dans quel ordre se trouvent les couches de liquide de la façon suivante : étape par étape, tu enlèves dans ta tête les liquides qui ne sont pas au dessus d'autres liquides dans aucune des trois bouteilles données, et les verses dans la grande bouteille.

Au départ, le liquide bleu est tout au fond des bouteilles 1 et 3 et jamais au dessus d'une autre couche de liquide. Le liquide rouge est tout au fond de la bouteille 2, mais au-dessus du liquide bleu dans la bouteille 1 et doit donc avoir une densité plus faible que le liquide bleu. On enlève donc le liquide bleu des bouteilles et le verse dans la grande bouteille.

La liquide rouge est maintenant le seul qui n'est pas au dessus d'un autre liquide. On l'enlève des bouteilles 1 et 2 et le met dans la grande bouteille. Ensuite viennent le liquide jaune, puis le vert et finalement le violet, qui a la plus faible densité et au dessus duquel ne se trouve aucun autre liquide.

## C'est de l'informatique !

Dans cet exercice, tu as évalué l'arrangement des liquides dans les trois bouteilles et trié les liquides par densité.

Une substance a beaucoup de propriétés mesurables, par exemple la température d'ébullition, la température de fusion, la conductivité et la densité. Dans le cas présent, la densité a été utilisée comme critère pour trier des substances.

Le tri de données joue un rôle important dans beaucoup de programmes informatiques. La méthode utilisée dans cet exercice pour déterminer l'ordre des couches de liquide s'appelle *tri topologique*. Elle est utilisée pour trier des objets lorsque l'on connaît la *relation d'ordre* entre certains des objets (si l'on sait déjà que certains objets en précèdent ou suivent d'autres).



## Mots clés et sites web

- Relation d'ordre : [https://fr.wikipedia.org/wiki/Relation\\_d'ordre](https://fr.wikipedia.org/wiki/Relation_d'ordre)
- Tri topologique : [https://fr.wikipedia.org/wiki/Tri\\_topologique](https://fr.wikipedia.org/wiki/Tri_topologique)



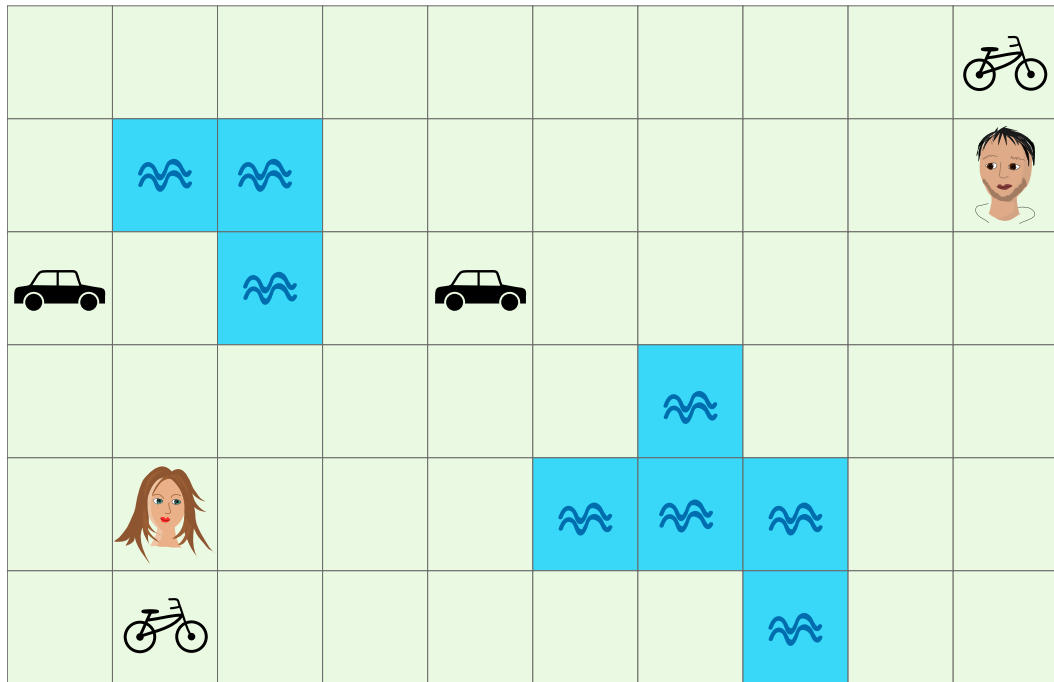


## 5. Un, deux, trois, partez, feu !

Deux amis veulent se voir le plus vite possible. Ils peuvent passer d'une case à la case voisine de droite, de gauche, du haut ou du bas.

La traversée d'une case à une autre prend 1 minute à pied. S'ils arrivent sur une case avec un véhicule, il peuvent l'utiliser. Ils peuvent alors avancer de 2 cases en 1 minute à vélo et de 5 cases en 1 minute en voiture.

Les changements de direction sont toujours possible. Ils ne peuvent pas traverser les étendues d'eau.



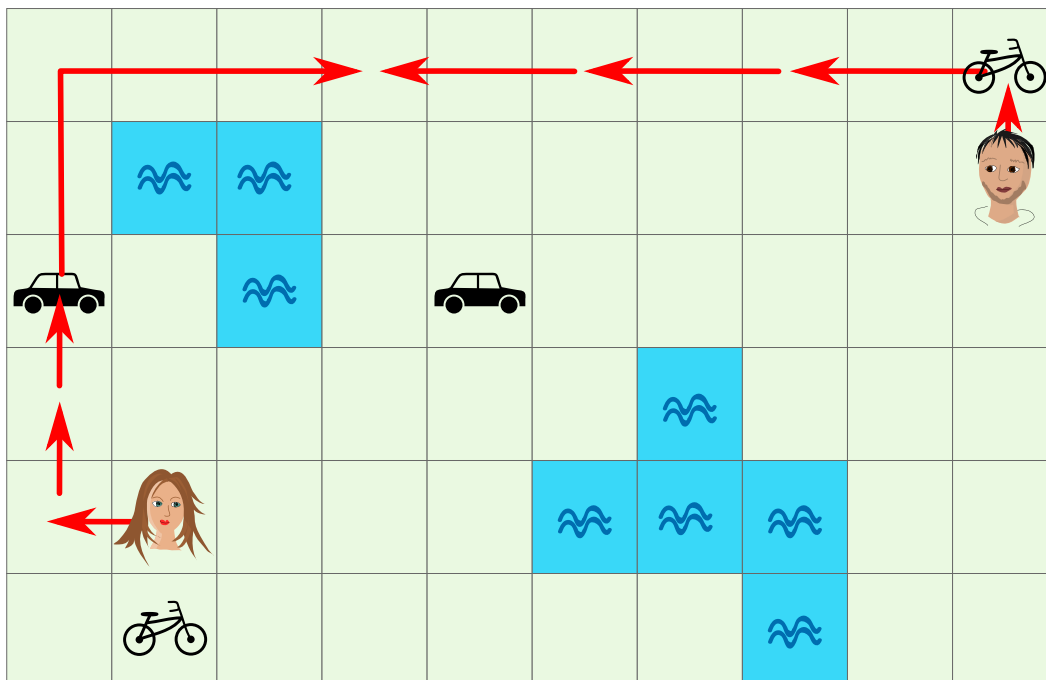
*De combien de minutes au minimum les deux amis ont-ils besoin pour se retrouver sur la même case ?*

- A) 1 minute
- B) 2 minutes
- C) 3 minutes
- D) 4 minutes
- E) 5 minutes
- F) 6 minutes



## Solution

La bonne réponse est 4. L'image montre un chemin permettant aux deux amis de se retrouver sur la même case en 4 minutes.



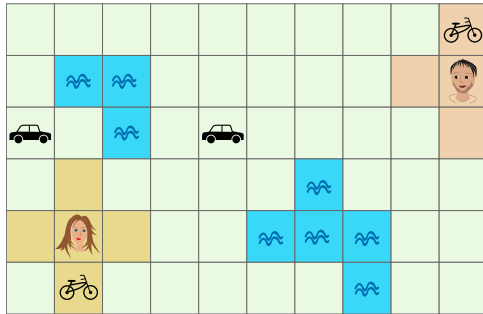
Maintenant, il faut encore prouver qu'ils ne peuvent pas se retrouver en 3 minutes. Les deux amis sont à 11 cases de distance l'un de l'autre. En trois minutes, s'ils se déplacent à pied, il ne peuvent se rapprocher que de 6 cases en tout. Si l'un des deux a atteint un vélo et que l'autre va à pied, ils peuvent se rapprocher de 9 cases, ce qui ne suffit pas non plus. Même s'ils se déplacent les deux à vélo, ils n'y arrivent pas : ils pourraient se rapprocher de 12 cases en 3 minutes, mais les deux vélos sont à 13 cases l'un de l'autre.

Il ne leur reste donc que la possibilité d'utiliser une voiture. En 3 minutes, la fille peut atteindre une voiture, mais n'aurait plus le temps de l'utiliser. Le garçon ne peut pas atteindre de voiture en 3 minutes.

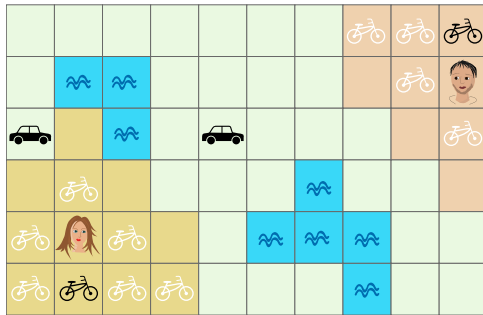
## C'est de l'informatique !

Comment as-tu résolu l'exercice ? As-tu trouvé un chemin rapide par hasard et espéré qu'il n'en existe pas de plus rapide ? Ou as-tu essayé beaucoup de chemins différents en te rappelant du plus rapide ?

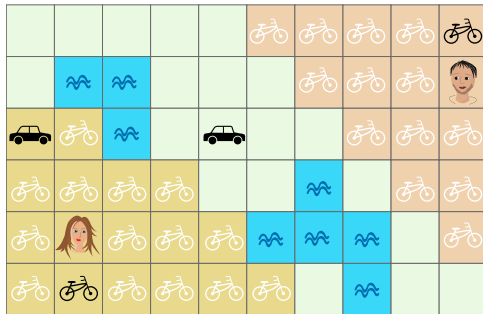
Les programmes informatiques qui ont été développés pour résoudre ce genre de problème travaillent souvent avec une méthode appelée *parcours en largeur*. Dans cet exercice, le parcours en largeur se passe comme cela :



1. Sélectionne toutes les cases que chacun des deux amis peut atteindre en 1 minute.

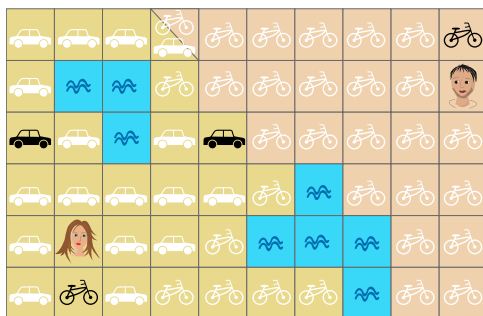


2. Sélectionne toutes les cases qui peuvent être atteinte en (maximum) 1 minute depuis les cases sélectionnées à l'étape 1. Note quel moyen de transport a été utilisé.



3. Sélectionne toutes les cases qui peuvent être atteinte en (maximum) 1 minute depuis les cases sélectionnées à l'étape 2.

Comme les deux régions que tu as sélectionnées ne se chevauchent pas encore, les deux amis ne peuvent pas encore se voir après 3 minutes.



4. Sélectionne toutes les cases qui peuvent être atteinte en (maximum) 1 minute depuis les cases sélectionnées à l'étape 3.

Maintenant, les deux régions se chevauchent sur une case. Elle peut être atteinte en 4 minutes par la fille en voiture et par le garçon en vélo.

Les systèmes de navigation trouvent le chemin le plus rapide entre deux points. Pour cela, ils font attention à ce que le chemin passe par des routes adaptées, et non pas à travers champs ou par des rivières. Cet exercice ressemble à un problème de navigation, mais ici, ce n'est pas une personne qui doit arriver à un but, mais deux personnes qui doivent arriver à un but commun inconnu au départ.

Comme un ordinateur procède de manière systématique lors d'un parcours en largeur, il peut aussi trouver des solutions qui ne sont pas évidentes. Parfois, un détour par une route avec moins de feux



peut être plus rapide que le chemin le plus court entre le départ et l'arrivée. Un voyage en train avec changement peut être plus rapide qu'un voyage direct en bus.

Il existe en informatique plusieurs méthodes pour résoudre des problèmes de ce type. En plus de la méthode de parcours en largeur discutée ici, il existe une méthode appelée *Branch and Bound* (*séparation et évaluation* en français). Le parcours en largeur tient compte de chaque solution partielle obtenue après un certain nombre d'étapes. Avec *Branch and Bound*, les solutions partielles ne menant pas à la solution optimale ne sont plus considérées dans les étapes suivantes.

Lorsqu'un problème devient trop complexe, cela peut durer trop longtemps d'essayer toutes les possibilités pour trouver la meilleure solution, même avec l'ordinateur le plus rapide du monde. En pratique, il suffit à un système de navigation de trouver un très bon chemin, même si ce n'est pas le meilleur chemin possible — si tu peux atteindre ton but en 78 minutes, ça ne te fait probablement rien qu'on puisse théoriquement aussi l'atteindre en 77 minutes.

## Mots clés et sites web

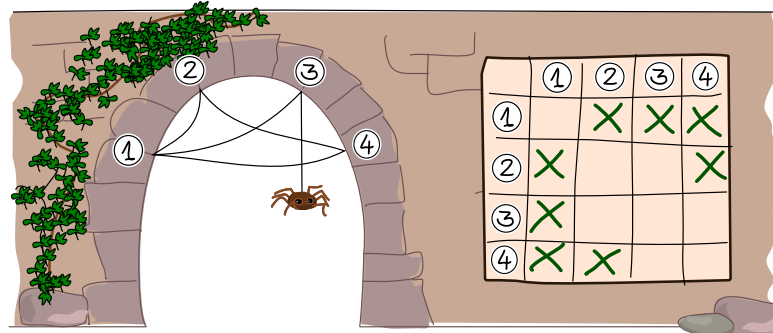
- Parcours en largeur : [https://fr.wikipedia.org/wiki/Algorithme\\_de\\_parcours\\_en\\_largeur](https://fr.wikipedia.org/wiki/Algorithme_de_parcours_en_largeur)
- Séparation et évaluation : [https://fr.wikipedia.org/wiki/Séparation\\_et\\_évaluation](https://fr.wikipedia.org/wiki/Séparation_et_évaluation)





## 6. Toiles d'araignée

Thekla l'araignée aimerait construire autant de toiles différentes que possible. Pour cela, elle a mis au point une méthode lui permettant de décrire la structure exacte de ses toiles.

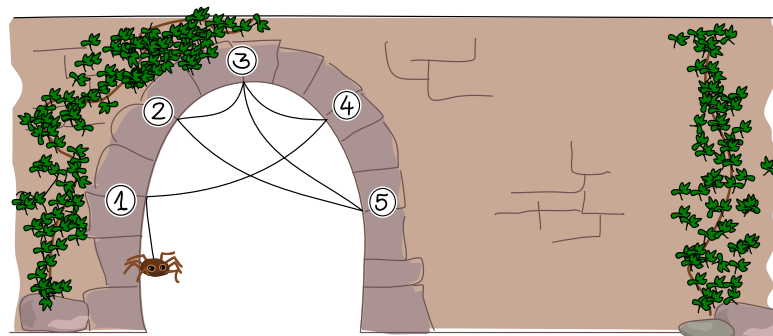


Elle procède de la façon suivante : elle numérote les points d'ancrage de la toile de 1 à  $N$  et utilise les cases d'une grille comme cela :

- S'il y a un fil qui relie le point d'ancrage  $x$  au point d'ancrage  $y$ , elle met un « X » dans la case située dans la colonne  $x$  et dans la ligne  $y$ .

Un fil qui relie le point d'ancrage  $x$  au point d'ancrage  $y$  relie également le point d'ancrage  $y$  au point d'ancrage  $x$ .

Thekla construit à présent cette toile :



*Comment Thekla décrit-elle la structure de cette toile ?*



A)

	①	②	③	④	⑤
①				X	
②			X		X
③		X		X	X
④	X		X		
⑤		X	X		

B)

	①	②	③	④	⑤
①		X		X	
②	X		X		
③		X		X	X
④	X		X		
⑤			X		

C)

	①	②	③	④	⑤
①	X			X	
②			X		X
③		X		X	X
④	X		X	X	
⑤		X	X		

D)

	①	②	③	④	⑤
①				X	
②			X		X
③		X		X	X
④	X		X		
⑤			X		



## Solution

La réponse A est correcte, car toutes les cases sont remplies correctement d'après la règle.

	①	②	③	④	⑤
①				X	
②			X		X
③		X		X	X
④	X		X		
⑤		X	X		

La réponse B comporte une connection supplémentaire erroné (entre le point d'ancrage 1 et le point d'ancrage 2 dans les deux directions) et une connection a été oubliée (entre le point d'ancrage 2 et le point d'ancrage 5 dans les deux directions).

	①	②	③	④	⑤
①		<del>X</del>		X	
②	<del>X</del>		X		<del>X</del>
③		X		X	X
④	X		X		
⑤		<del>X</del>	X		

Concernant la réponse C : d'après la règle, aucun « X » ne peut être présent dans la diagonale allant du haut à gauche au bas à droite. ils représenteraient en effet des connections entre d'un point d'ancrage et lui-même. Ceux-ci peuvent exister dans certains réseaux, mais pas dans notre toile d'araignée. La réponse C comporte cependant deux telles connections (au points d'ancrage 1 et 4).

	①	②	③	④	⑤
①	<del>X</del>			X	
②			X		X
③		X		X	X
④	X		X	<del>X</del>	
⑤		X	X		

Le réponse D n'est pas symétrique par rapport à la diagonale allant du haut à gauche au bas à droite, alors que toutes les descriptions de toiles devraient l'être. Dans cette réponse, le point d'ancrage 2 et le point d'ancrage 5 sont connectés, mais la connection correspondante dans l'autre direction manque.

	①	②	③	④	⑤
①				X	
②			X		X
③		X		X	X
④	X		X		
⑤		<del>X</del>	X		

## C'est de l'informatique !

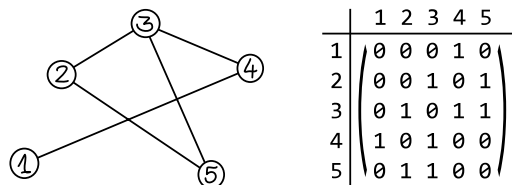
La toile d'araignée peut être considérée comme un *graphe* ; un concept qui apparait fréquemment en informatique.

Un graphe est composé de *nœuds* (les points d'ancrage de la toile) et d'*arêtes* (les fils entre les points d'ancrage). Les graphes peuvent aussi être utilisés pour représenter des objets et les relations entre ces objets. Par exemple, un graphe pourrait représenter la manière dont des personnes sont connectées sur un réseau social ou des vols d'avion entre différents pays.

Cet exercice montre comment la structure d'une toile d'araignée peut être enregistrée dans une grille. Certaines propriétés de la toiles sont perdues dans cette représentation, comme par exemple l'apparence exacte de la toile. Souvent, on ne s'intéresse pas aux propriétés géométriques exactes



d'une toile, mais seulement à sa structure. Les informations essentielles sont conservées : combien y a-t-il de nœuds ? Quelles paires de nœuds sont-elles reliées par une arête ?



La représentation utilisée ici n'est qu'une possibilité parmi beaucoup de représenter la structure d'un réseau. Cette méthode n'est pas très économe, car toutes les connections sont enregistrées dans les deux sens, ce qui ne serait pas nécessaire ; les cases de la diagonale ne seraient pas nécessaire non plus. Cette méthode a par contre l'avantage de mettre en évidence les erreurs de représentation. Les réponses C et D, par exemple, peuvent être éliminées sans même considérer la toile d'araignée.

La méthode de représentation utilisée dans cet exercice s'appelle une *matrice d'adjacence*.

## Mots clés et sites web

- Matrice d'adjacence : [https://fr.wikipedia.org/wiki/Matrice\\_d'adjacence](https://fr.wikipedia.org/wiki/Matrice_d'adjacence)







## 7. Pile de fruits

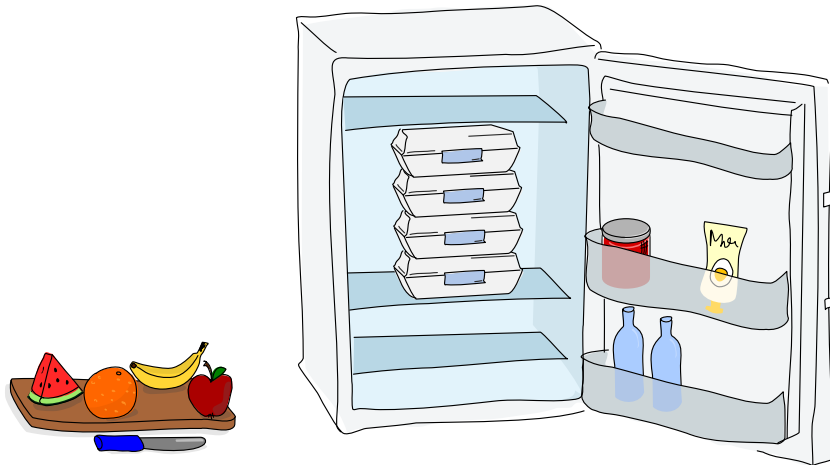
Papa, Maman, Dorie et Ron Castor préparent quatre boîtes avec un fruit différent dans chacune : pomme, banane, orange et pastèque. Les boîtes sont empilées dans le réfrigérateur. Le matin, les castors sont encore très fatigués et prennent simplement la boîte du haut de la pile en quittant le gîte sans la regarder plus en détail.

On ne sait pas exactement dans quel ordre les castors quittent le gîte, mais Maman part dans tous les cas avant Dorie et Papa sort toujours en dernier.

Les membres de la famille aiment des fruits différents. Le tableau suivant indique ce que chaque membre de la famille aime :

				
<b>Papa</b>	—	—	✓	—
<b>Maman</b>	✓	—	✓	✓
<b>Dorie</b>	✓	✓	✓	—
<b>Ron</b>	✓	✓	—	✓

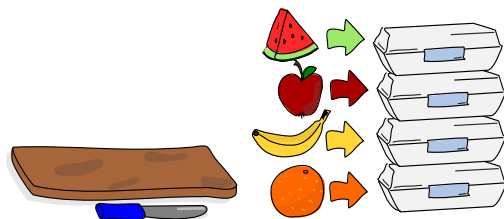
Mets les fruits dans les boîtes de manière à ce que chaque castor prenne une boîte contenant un fruit qu'il aime.





## Solution

Il n'y a qu'une possibilité de répartir les fruits de manière à garantir que chacun ait quelque chose qu'il aime :



Papa n'aime que les oranges et part en dernier. L'orange va donc dans la boîte la plus basse. Ron part en premier, deuxième ou troisième. Comme Maman part avant Dorie, on connaît l'ordre de départ exact des castors si l'on sait quand Ron quitte le gîte. Les trois ordres de départ suivants sont possibles :

1. Maman Maman Ron
2. Dorie Ron Maman
3. Ron Dorie Dorie
4. Papa Papa Papa

Maman, Dorie et Ron peuvent chacun partir en deuxième. Il faut donc mettre un fruit que les trois aiment dans la deuxième boîte, et ce n'est le cas que de la pomme. Il ne reste ainsi que la banane et la pastèque pour la boîte du haut. Comme maman n'aime pas les bananes, il faut y mettre la pastèque. La banane va ainsi dans la troisième boîte.

## C'est de l'informatique !

Le bon ordre d'une séquence est important dans beaucoup de domaines de l'informatique : beaucoup de calculs doivent utiliser des résultats intermédiaires pour arriver au résultat final. Si différentes étapes de calcul sont effectuées sur différents ordinateurs sans planning consciencieux, des *interblocages* (ou *àtreintes fatales*, *deadlock* en anglais) peuvent avoir lieu. Ce sont des situations dans lesquelles plusieurs ordinateurs s'attendent mutuellement et qui empêchent le programme de se terminer. Un mauvais ordre ne mène cependant souvent qu'à des erreurs (comme il peut amener la mauvaise humeurs chez les castors qui découvrent leur fruit). Par exemple, si l'on doit calculer à l'aide de la formule  $Z = (A + B) \times (A - B)$ , on peut partager cela en plusieurs étapes d'un programme comme cela :

Entrée A

Entrée B

Calcule  $X = A + B$

Calcule  $Y = A - B$

Calcule  $Z = X \times Y$

Si l'on essaie maintenant d'effectuer par exemple l'étape  $Z = X \times Y$  avant d'avoir calculé  $X$ , cela cause une erreur et l'interruption du programme. Alternativement, une valeur par défaut va être



utilisée pour  $X$ , ce qui mène à un faux résultat dans la plupart des cas. L'ordre dans lequel les commandes sont effectuées est donc très important en informatique.

## Mots clés et sites web

- Interblocage: <https://fr.wikipedia.org/wiki/Interblocage>

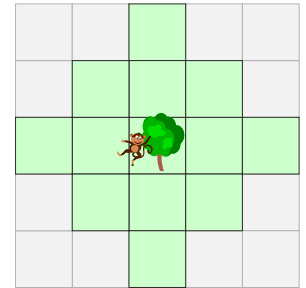




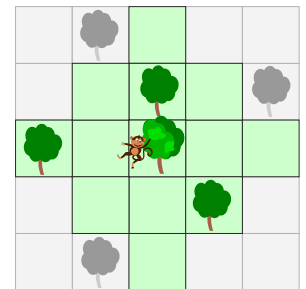


## 8. Petit singe

Coco le petit singe peut sauter depuis un arbre aussi loin que le montre le domaine coloré en vert.

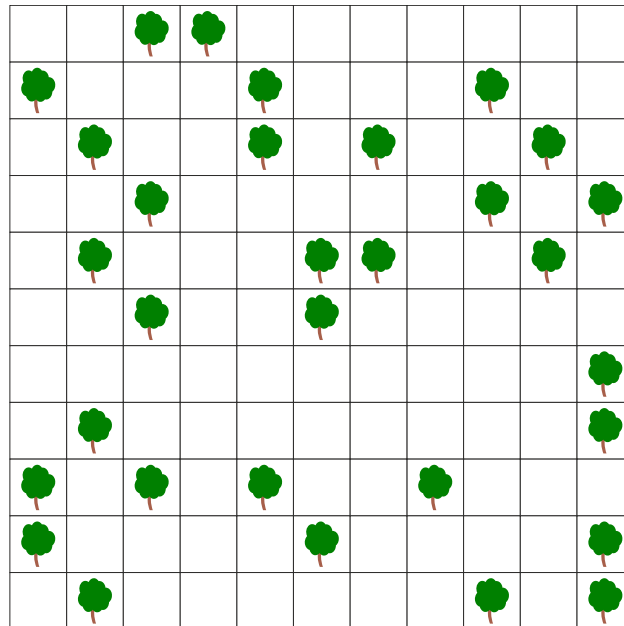


Dans l'exemple suivant, Coco peut atteindre les arbres en couleur d'un seul saut. En deux sauts, il peut aussi atteindre les deux arbres gris du haut, mais pas l'arbre gris du bas.



Il existe des groupes d'arbres dans lesquels Coco peut se déplacer sans toucher le sol une seule fois.

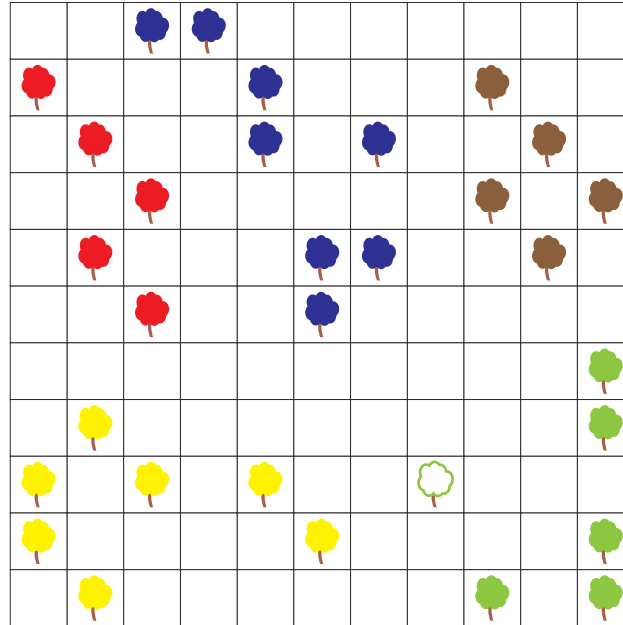
*Sélectionne tous les arbres du plus grand de ces groupes.*





## Solution

Sur l'image ci-dessous, deux arbres ont la même couleur si Coco peut passer de l'un à l'autre sans toucher le sol.

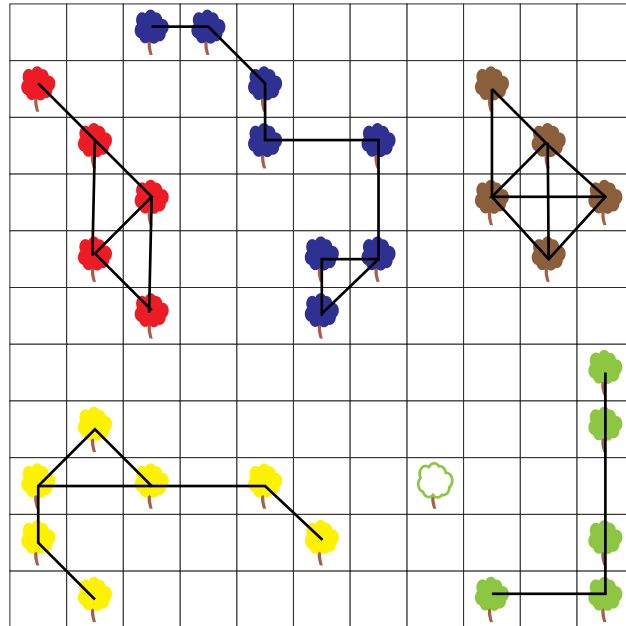


On voit que le groupe d'arbres bleus comptant huit arbres est le plus grand des groupes.

## C'est de l'informatique !

Si Coco peut sauter directement d'un arbre à l'autre, ils sont pour ainsi dire connectés l'un à l'autre. On peut représenter cela à l'aide d'une ligne entre les arbres comme montré plus bas. On obtient donc un graphe dont les arbres sont les nœuds avec des arêtes entre les arbres connectés. Coco peut passer d'un arbre à l'autre en sautant seulement s'il existe un chemin le long des arêtes menant d'un arbre à l'autre.

On appelle un groupe de nœuds *connexe* si tous les nœuds sont connectés ensemble par des arêtes. Lorsqu'un tel groupe ne peut plus être agrandi sans perdre la connection entre les nœuds, on parle d'une *composante connexe*. Un graphe peut être divisé de manière unique en composantes connexes ; elles sont colorées sur l'image ci-dessous.



On peut facilement identifier une composante connexe en partant d'un nœud quelconque puis en cherchant tous les nœuds accessibles par des arêtes.

## Mots clés et sites web

- Graphe connexe : [https://fr.wikipedia.org/wiki/Graphe\\_connexe](https://fr.wikipedia.org/wiki/Graphe_connexe)





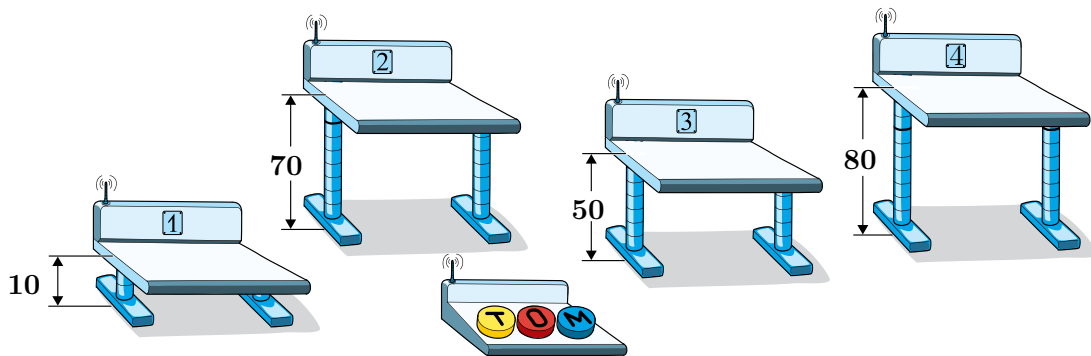
## 9. Sacrés pupitres

Dans la salle de classe, il y a des pupitres dont la hauteur est réglable électriquement. Tous les pupitres doivent être réglés à 60 cm pour les cours. La hauteur peut être changée à l'aide des touches 🟡, 🔴 et 🔵 d'une télécommande. Quelqu'un a joué avec la télécommande et l'a reprogrammée. Maintenant, les trois touches fonctionnent comme cela :

- 🟡 monte les pupitres 1, 2 et 3 de 10 cm chacun.
- 🔴 baisse les pupitres 2, 3 et 4 de 10 cm chacun.
- 🔵 monte les pupitres 1, 3 et 4 de 10 cm chacun.

Ces actions sont exécutées chaque fois que l'on appuie sur le bouton.

En ce moment, les hauteurs des pupitres 1, 2, 3 et 4 sont de 10 cm, 70 cm, 50 cm et 80 cm :



*Comment peut-on régler la hauteur des quatre pupitres à 60 cm ?*

- A) Appuie 4 × sur 🟡, 5 × sur 🔴 et 1 × sur 🔵.  
 B) Appuie 5 × sur 🟡, 1 × sur 🔴 et 0 × sur 🔵.  
 C) Appuie 3 × sur 🟡, 4 × sur 🔴 et 2 × sur 🔵.  
 D) Appuie 2 × sur 🟡, 4 × sur 🔴 et 6 × sur 🔵.



## Solution

La bonne réponse est C) Appuie 3 × sur , 4 × sur et 2 × sur .

Tu peux constater que les trois touches de la télécommande changent la hauteur de 10 cm, donc toujours de la même distance. Deux des touches font monter les pupitres ( et ) et seulement un touche les fait descendre (). De plus, chacune des trois touches modifie la hauteur de trois pupitres ; il y a donc toujours un pupitre dont la hauteur ne change pas. La touche n'a aucun effet sur le pupitre 1, on ne peut donc pas le faire descendre.

Le pupitre 1 est 50 cm trop bas. On peut en déduire que l'on doit appuyer exactement 5 fois sur les touches ou (la somme des nombres de fois où les touches et sont utilisées doit être égale à 5). On peut exprimer cela par l'équation  $T + M = 5$ . On peut donc éliminer la solution D), car  $T + M = 8$  pour cette solution. D'après la suite de touches de la solution D), la pupitre 1 serait haut de  $10 + 20 + 60 = 90$  cm, c'est à dire la hauteur de départ 10 cm plus  $2 * 10$  cm pour plus  $6 * 10$  cm pour .

le pupitre 2 est 10 cm trop haut. n'a pas d'effet sur le pupitre 2. La bonne solution doit donc satisfaire l'équation  $T - O = -1$ . On peut donc éliminer la solution B), car en l'utilisant, le pupitre 2 aurait à la fin une hauteur de  $70 + 50 - 10 = 110$  cm.

Le pupitre 3 est 10 cm trop bas, ce qui donne l'équation  $T - O + M = 1$ . Les solutions A) et B) peuvent donc être éliminées. Avec la solution A), le pupitre aurait la même hauteur à la fin qu'au début :  $50 + 40 - 50 + 10 = 50$  cm ; avec la solution B, la hauteur du pupitre serait  $50 + 50 - 10 = 90$  cm. Toutes les solutions excepté la solution C) ont maintenant été éliminées.

Il faut encore vérifier que la solution C) met aussi le pupitre 4 à la bonne hauteur. Le pupitre 4 est 20 cm trop haut et la touche n'a aucun effet sur sa hauteur. Il faut donc appuyer deux fois sur et une fois de plus pour chaque fois que la touche est utilisée. La suite de touches de la solution C) donne la hauteur  $80 - 40 + 20 = 60$  cm.

Comme on a déjà constaté plus haut que la solution C) permettait de mettre les pupitres 1, 2 et 3 à la bonne hauteur, on est maintenant sûr que cette solution fonctionne.

Un autre moyen de trouver la solution est de résoudre quatre équations linéaires. Pour chaque pupitre, on écrit une équation décrivant quelles touches changent sa hauteur et quelle est le changement de hauteur désiré. Par exemple, la hauteur du pupitre 1 ne change qu'avec les touches et et le changement désiré est de 50 cm, ce que l'on peut obtenir en appuyant sur 5 touches (étant donné que pression sur une touche change la hauteur de 10 cm).

Comme il y a quatre pupitres et trois touches, on obtient quatre équations linéaires avec trois inconnues :

$$\begin{array}{l}
 T + M = 5 \\
 T - O = -1 \\
 T - O + M = 1 \\
 -O + M = -2
 \end{array}$$

En soustrayant la troisième équation de la première, on obtient  $O = 4$ . en substituant dans la deuxième équation, on obtient  $T = 3$ . Toutes les équations sont résolues juste en prenant  $M = 2$ . C'est donc la seule bonne solution.



## C'est de l'informatique !

Cet exercice est un problème typique du domaine de l'*optimisation linéaire en nombres entiers*, aussi appelée *programmation linéaire en nombres entiers*. Le problème est donné par un certain nombre de contraintes. Dans ce cas particulier, on peut toutes les formuler sous la forme d'équations linéaires. L'optimisation est un but typique en informatique : on cherche une suite d'action permettant d'arriver à un but prédéfini. On pourrait même décrire tout l'exercice par la recherche d'un chemin dans un espace quadridimensionnel avec trois déplacements possibles pour passer du point (10, 70, 50, 80) au point (60, 60, 60, 60). Cet exercice n'a qu'une solution, mais de tels problèmes ont souvent de multiples solutions, rendant l'optimisation possible. On cherche alors le minimum de la fonction linéaire  $T + M + O$ .

## Mots clés et sites web

- Optimisation linéaire en nombres entiers :  
[https://fr.wikipedia.org/wiki/Optimisation\\_linéaire\\_en\\_nombres\\_entiers](https://fr.wikipedia.org/wiki/Optimisation_linéaire_en_nombres_entiers)

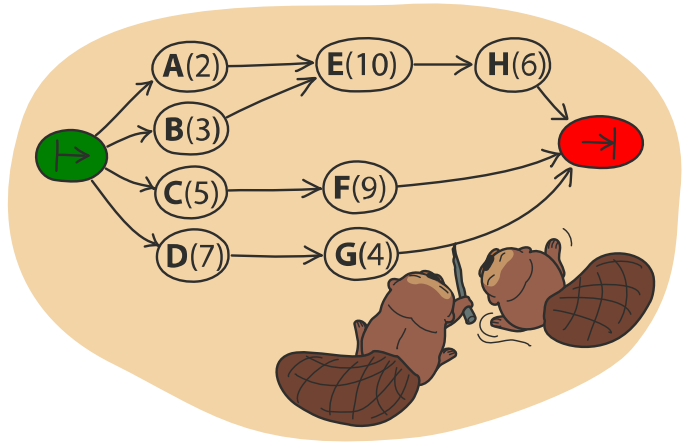




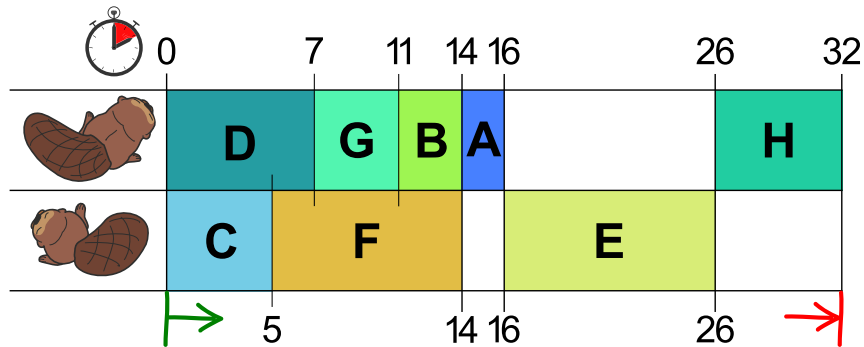


# 10. Plan de travail

La construction d'un barrage par des castors peut être divisée en plusieurs tâches (abattre des arbres, enlever les branches, transporter les troncs jusqu'à la rivière, etc.). L'image de droite montre les huit tâches A, B, C, D, E, F, G et H et le nombre d'heures nécessaires à chacune pour construire un barrage. Les tâches ne sont pas complètement indépendantes les unes des autres : un flèche de X à Y signifie que la tâche X doit être terminée avec de pouvoir commencer la tâche Y.

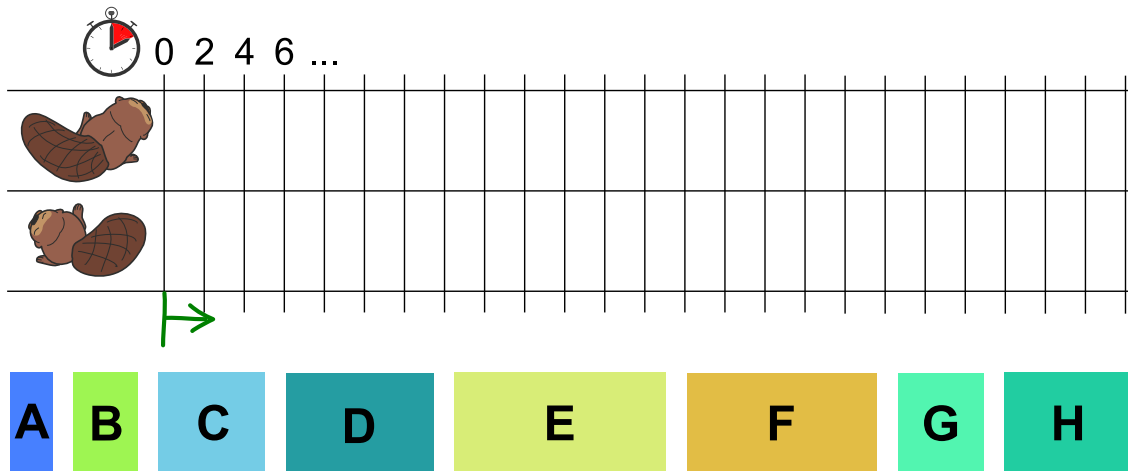


La castore Ulla veut aider le castor Otso à construire le barrage plus vite. Ils se répartissent les tâches et font le plan de travail suivant en tenant compte des dépendances indiquées plus haut.



Le barrage serait ainsi terminé en 32 heures. Mais c'est possible de le faire plus vite!

Fais un plan de travail qui permet de finir le barrage le plus vite possible.

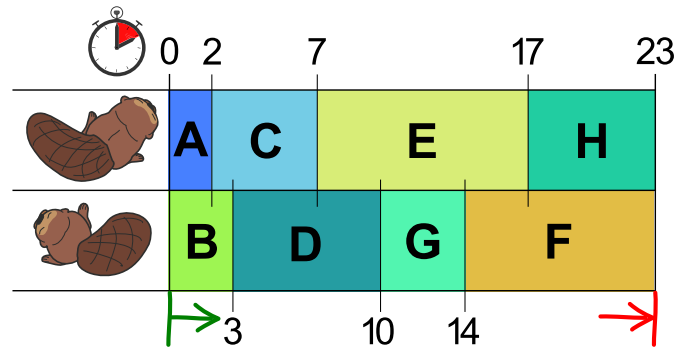




## Solution

Dans le plan de travail de l'exercice, le premier castor a une longue pause (10 heures), et le deuxième castor a en tout 8 heures de libre. Ils auraient terminé plus vite s'ils travaillaient en permanence.

On obtient un plan de travail plus rapide si l'on fait attention à ce que les deux tâches les plus longues E(10) et F(9) ne soient pas effectuées par le même castor. Voici un plan de travail qui ne dure que 23 heures. Ce n'est pas possible de terminer le barrage plus vite, car les deux castors travaillent ici déjà sans pause.



## C'est de l'informatique !

L'être humain est impatient de nature, c'est pourquoi il est souvent souhaitable de finir des travaux aussi rapidement que possible. Lorsqu'un travail peut être réparti (que ce soit entre différentes personnes, machines ou castors), la répartition des différentes tâches entre les travailleurs joue un rôle important dans la durée du travail. Similairement, les ordinateurs doivent souvent répartir leur tâches entre différents processeurs de manière optimale.

Il existe plusieurs stratégies bien étudiées pour de tels problèmes d'*ordonnancement* en informatique. Pour le premier plan de travail de cet exercice, les plus longues tâches parmi toutes celles à réaliser ont été assignées au castor qui ne travaillait pas à ce moment-là – une mauvaise stratégie dans ce cas. Cela fonctionne souvent mieux lorsque les tâches les plus courtes sont effectuées en premier : la stratégie *Shortest job first* (« le plus court processus en premier » en anglais) minimise aussi le temps d'attente moyen pour chaque tâche.

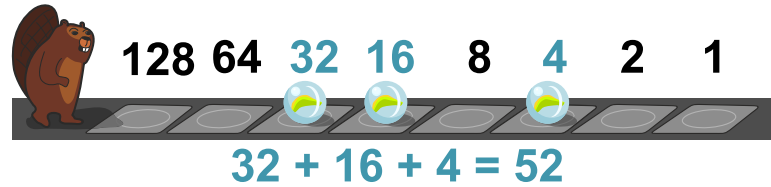
## Mots clés et sites web

- Ordonnancement : [https://fr.wikipedia.org/wiki/Ordonnancement\\_de\\_travaux\\_informatiques](https://fr.wikipedia.org/wiki/Ordonnancement_de_travaux_informatiques)
- Shortest job first : [https://fr.wikipedia.org/wiki/Shortest\\_job\\_first](https://fr.wikipedia.org/wiki/Shortest_job_first)



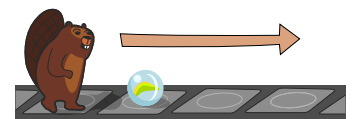
# 11. Nombres en billes

Les castors ont une manière particulière de représenter les nombres.



Les différentes cases ont des poids différents et la présence d'une bille sur la case détermine que la valeur est prise en compte. Le nombre 52 est représenté dans l'exemple ci-dessus.

Le castor se déplace sur un ruban d'une case à la suivante de gauche à droite. Sur chaque case du ruban peut se trouver une bille.



Si le castor arrive sur une case avec une bille et a les mains libres, il la soulève et la prend avec en la portant dans ses bras.



Il dépose la bille sur la prochaine case libre.



Le castor ne peut porter qu'une seule bille à la fois, et il n'y a la place que pour une bille sur chaque case.

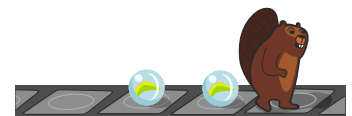
Si le castor porte déjà une bille en arrivant sur une case avec une autre bille...



... il dépasse celle-ci...



... et pose sa bille sur la prochaine case vide.



Il peut ensuite à nouveau soulever la bille suivante.

Quel nombre les billes représentent-elles une fois que le castor a traversé cette partie du ruban ?



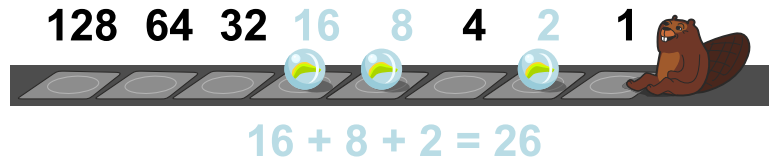


- A) 10
- B) 26
- C) 28
- D) 104

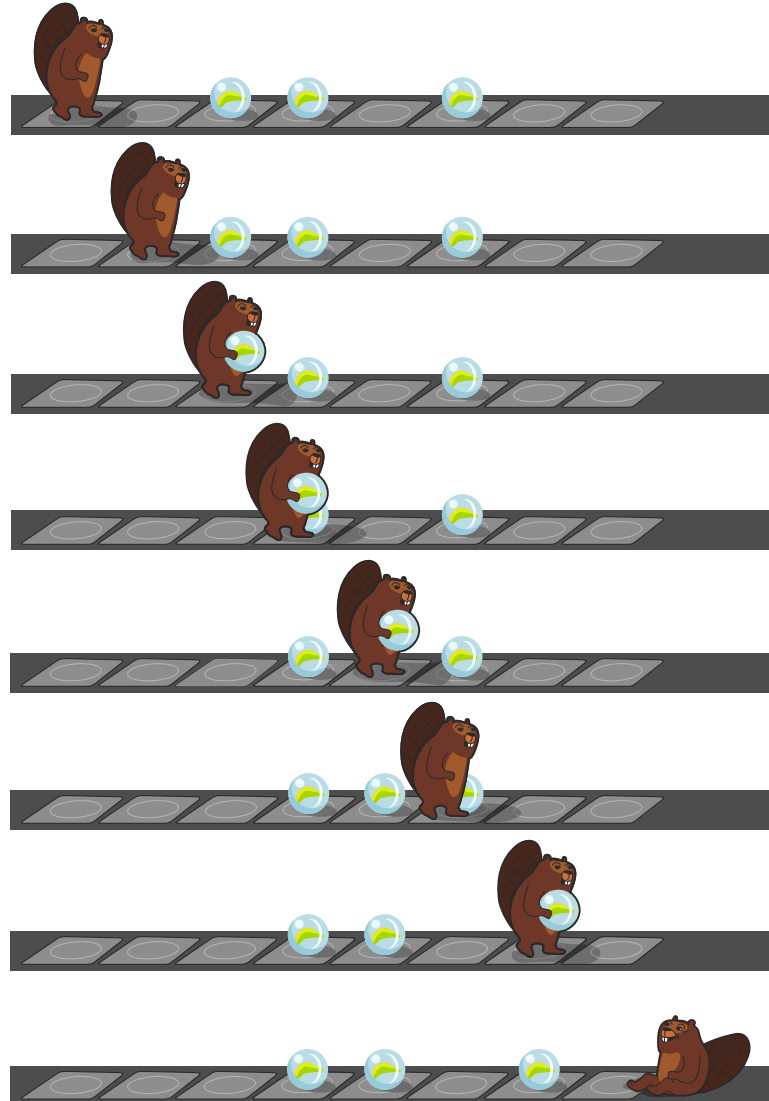


## Solution

La bonne réponse est B) 26.



L'image suivante montre le déroulement :



## C'est de l'informatique !

En informatique, des opérations relativement simples donnent souvent des résultats intéressants. Cet exercice en est un bon exemple. La démarche du castor est un *algorithme*. Il se base sur le fait que le castor peut adopter deux états différents (portant une bille ou n'en portant pas) et qu'il peut trouver deux sortes de cases sur son chemin (occupées ou libres).



Le résultat final de l'algorithme est le même que si l'on avait déplacé chaque bille d'une case vers la droite. Cela représente une division par deux dans le système numérique des castors. Si le castor se déplaçait de droite à gauche sur le ruban, le nombre serait multiplié par deux. Lorsque toute une rangée de uns et de zéros est décalée vers la gauche ou vers la droite, on appelle cela un *décalage de bits* en informatique.

Aussi simple que soit l'exemple dans cet exercice, il contient plusieurs des éléments essentiels d'une *machine de Turing*. Une machine de Turing (qui doit son nom au mathématicien Alan Turing) est un ordinateur particulier qui a une structure très simple. En principe, une machine de Turing peut exécuter tous les algorithmes qu'un ordinateur traditionnel peut exécuter. En pratique, les machines de Turing ne sont pas utilisées comme ordinateur, car nous pouvons construire des ordinateurs qui sont bien plus efficaces, même s'ils sont plus compliqués. Les machines de Turing sont surtout utiles pour la théorie. Leur structure simple permet de prouver des affirmations simples les concernant ; et si l'on peut prouver qu'un exercice ne peut pas être résolu par une machine de Turing, cela veut dire qu'aucun de nos ordinateurs ne peut le résoudre non plus.

Une machine de Turing est composée :

- D'un *ruban* de longueur infinie divisé en *cases*. Chaque case peut contenir un *symbole*. Dans notre cas, il s'agit des cases sur lesquelles le castor se déplace.
- D'une quantité finie de *symboles*. Souvent, on n'utilise que 0 et 1 comme symboles. Dans notre exemple, une bille représente le 1 et une position libre le 0.
- D'une tête de lecture/écriture qui peut se déplacer sur le ruban dans les deux directions tout en lisant les symboles écrits et en écrivant de nouveaux symboles sur le ruban. Dans notre exemple, le castor joue le rôle de la tête de lecture/écriture.
- D'un registre d'*états* de taille finie. Le comportement de la tête de lecture/écriture peut changer en fonction de l'état. Dans notre cas, il n'y a que deux états : « portant une bille » ou « ne portant pas de bille ».
- D'un ensemble de règles, ou *table d'actions* : que ce passe-t-il, en fonction de l'état, lorsqu'un symbole précis est lu sur le ruban ? Les actions possibles sont : un changement d'état, l'écriture d'un nouveau symbole et le déplacement de la tête de lecture d'une case vers la gauche ou vers la droite.

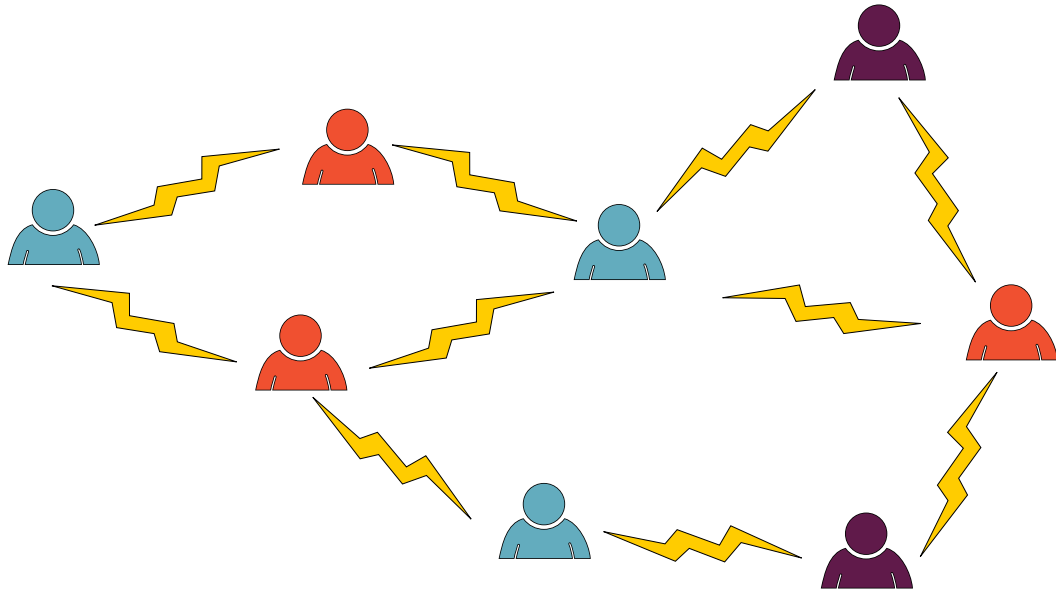
## Mots clés et sites web

- Machine de Turing : [https://fr.wikipedia.org/wiki/Machine\\_de\\_Turing](https://fr.wikipedia.org/wiki/Machine_de_Turing)
- Opération bit à bit, décalage de bit :  
[https://fr.wikipedia.org/wiki/Op%C3%A9ration\\_bit\\_%C3%A0\\_bit#D%C3%A9calages\\_de\\_bit](https://fr.wikipedia.org/wiki/Op%C3%A9ration_bit_%C3%A0_bit#D%C3%A9calages_de_bit)



## 12. Travail d'équipe

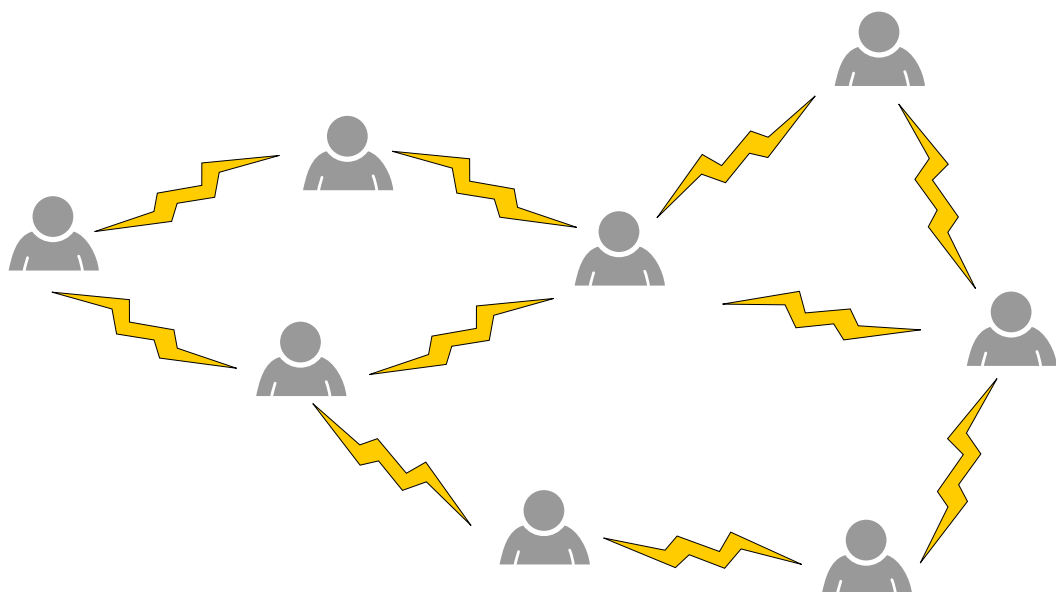
Tu dois répartir huit personnes en groupes de travail pour un projet. Un éclair est dessiné entre deux personnes qui ne veulent pas travailler ensemble. Dans ce cas, tu ne veux pas les mettre dans le même groupe de travail.



Dans l'exemple du haut, une répartition en trois groupes (rouge, bleu, violet) est possible en tenant compte des inimitiés. Il n'y a donc jamais d'éclair entre deux personnes de la même couleur.

Si tu arrives à convaincre les deux bonnes personnes de travailler ensemble, une répartition en seulement deux groupes (deux couleurs) est possible.

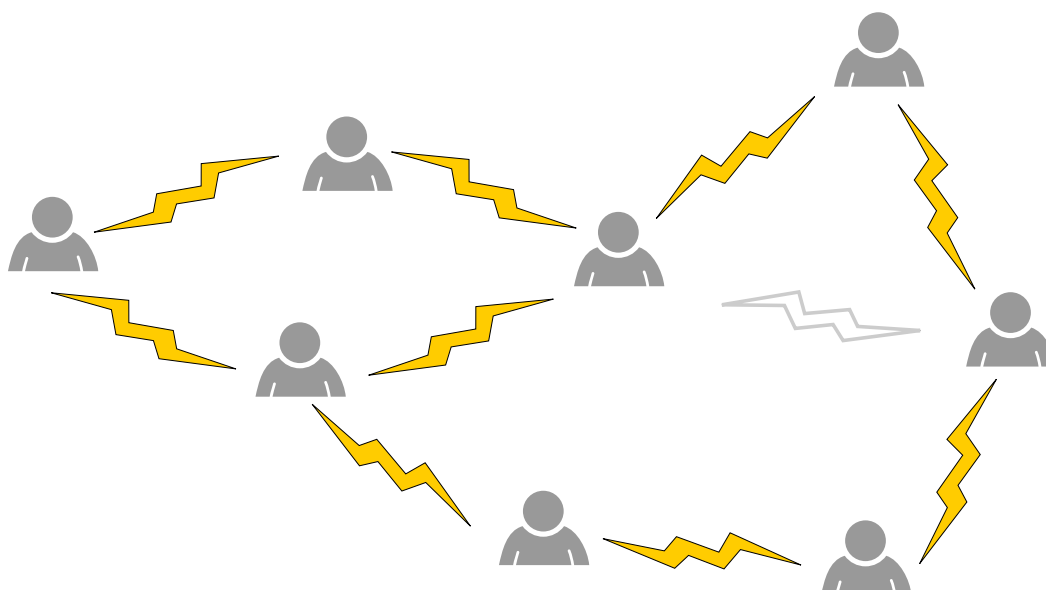
*Enlève le bon éclair.*





## Solution

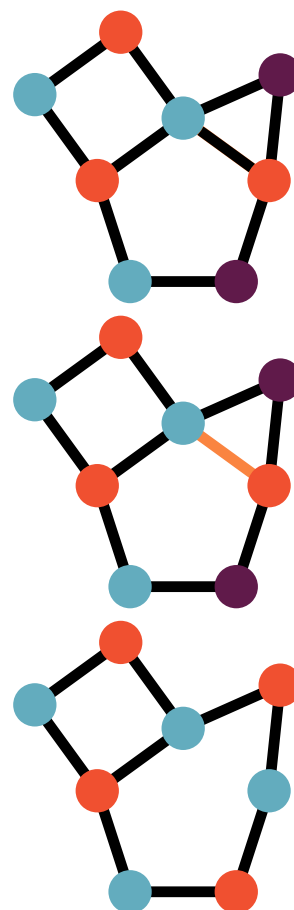
La bonne réponse est :



Nous représentons la situation de manière plus abstraite sous forme de *graphe* dans lequel les personnes sont les *nœuds* (cercles) et les éclairs les *arêtes* (lignes).

La seule option possible est d'enlever l'arête orange.

Après avoir enlevé cette arête, nous pouvons colorer les nœuds de deux couleurs.

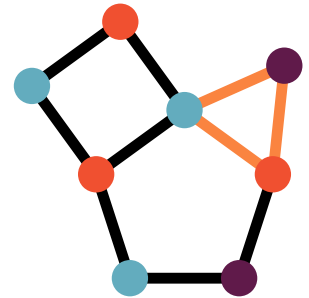


Chaque couleur représente un groupe. On peut voir qu'il n'y a jamais deux personnes du même groupe qui refusent de travailler ensemble : les nœuds voisins ont toujours des couleurs différentes.

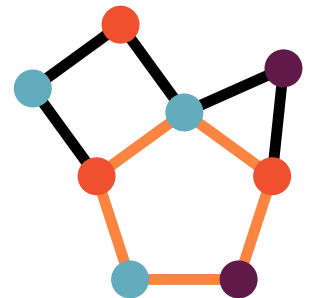




Pour déterminer que la seule option est d'enlever cette ligne-là, nous commençons par considérer le triangle orange.



Si une ligne en dehors de ce triangle est éliminée, nous avons toujours besoin de trois couleurs simplement pour le triangle en question.



Considérons maintenant le pentagone orange :

Si une arête en dehors de ce pentagone est éliminée, il reste intact et c'est donc impossible de le colorer avec deux couleurs : si l'on parcourt le pentagone dans le sens des aiguilles d'une montre, on doit alterner entre les deux couleurs. Lorsque l'on arrive au dernier nœud, celui-ci a alors la même couleur que le premier nœud juste à côté, étant donné que le nombre de nœuds est impair, comme c'est le cas pour le triangle.

La seule solution est donc d'enlever l'arête commune au triangle et au pentagone.

## C'est de l'informatique !

Beaucoup de problèmes du quotidien peuvent être formulés comme des problèmes de *coloration de graphe*. Dans cet exercice du castor, les *nœuds* d'un *graphe* représentent les personnes et une *arête* entre deux personnes montre qu'elles refusent de travailler dans le même groupe. Si nous colorons les nœuds avec  $k$  couleurs, cela représente l'assignation des personnes à l'un des  $k$  groupes de travail. Une telle coloration est valide si deux nœuds reliés par une arête ont toujours deux couleurs différentes. Dans notre cas, une coloration est donc valide lorsque toutes les personnes de chaque groupe travaillent ensemble. Une arête est appelée *critique* quand une coloration valide avec une couleur de moins devient possible en enlevant cette arête (on peut pour cela changer la couleur de tous les nœuds du graphe). Ici, une arête est donc critique quand on peut réduire le nombre de groupes en convainquant les deux personnes correspondantes de travailler ensemble.

## Mots clés et sites web

- Théorie de graphe : [https://fr.wikipedia.org/wiki/Théorie\\_des\\_graphes](https://fr.wikipedia.org/wiki/Théorie_des_graphes)
- Coloration de graphe : [https://fr.wikipedia.org/wiki/Coloration\\_de\\_graphe](https://fr.wikipedia.org/wiki/Coloration_de_graphe)





# 13. Compter avec les muscles

Un nouveau distributeur de billets doit fonctionner de la manière suivante : un client hoche la tête (c'est à dire qu'il baisse la tête puis la remet droite) aussi souvent qu'il veut de billets. Le client lève ensuite la tête, et le distributeur émet les billets. Pour cela, le distributeur a une caméra intégrée. Elle peut reconnaître le nez du client et mesure constamment sa longueur. Le programme de commande du distributeur enregistre le résultat de la mesure actuelle sous le nom `longueur_du_nez` et détermine la position de la tête du client à l'aide du tableau suivant :

Mesure de la caméra	Valeur <code>longueur_du_nez</code>	Position de la tête
	1	Le client a la tête droite.
	1,3	Le client a la tête baissée.
	0,7	Le client a la tête levée.

Le programme de commande est presque fini — regarde plus bas.

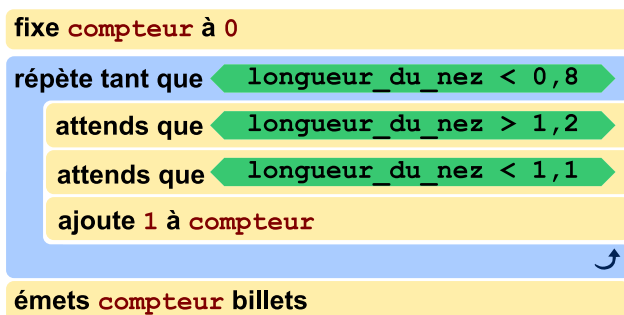
Complète le programme de commande.

fixe <code>compteur</code> à 0	longueur_du_nez > 0
répète tant que	longueur_du_nez > 0,8
attends que	longueur_du_nez > 1,2
attends que	longueur_du_nez < 0,8
ajoute 1 à <code>compteur</code>	longueur_du_nez < 1,1
émets <code>compteur</code> billets	



## Solution

La seule bonne réponse est :



La structure du programme est donnée: il y a un bloc de commande principal qui se répète, souvent appelé *boucle*. La dernière instruction de cette boucle d'instructions répétées augmente le compteur du nombre de billets à émettre. Les deux instructions commençant par **attends que** doivent donc détecter un hochement de tête du client; c'est à dire détecter que le client a d'abord baissé la tête puis l'a remise droite. La valeur enregistrée dans `longueur_du_nez` doit donc d'abord être environ 1,3 puis à nouveau 1. Cela correspond à l'instruction `longueur_du_nez > 1,2` suivie de `longueur_du_nez < 1,1`.

Les instructions de la boucle sont répétées jusqu'à ce que le client lève la tête; donc jusqu'à ce qu'une valeur clairement inférieure à 1 soit mesurée. La seule condition correspondant à cela est `longueur_du_nez < 0,8`.

Tu as peut-être remarqué que le programme n'utilise pas exactement les valeurs du tableau. En pratique, on ne peut en effet pas mesurer de manière continue, mais seulement à une certaine fréquence (par exemple 25 fois par seconde). Il peut donc arriver, par exemple, que la valeur 1 qui correspond à la tête droite ne soit jamais mesurée, parce que la valeur 1,03 a été mesurée juste après 0,95.

## C'est de l'informatique !

La *vision par ordinateur* (*machine vision* ou *computer vision* en anglais) est un domaine de l'informatique très étudié actuellement. Les considérations théoriques et les applications pratiques sont les deux d'une grande importance.

Une application notable de la vision par ordinateur est de permettre aux personnes handicapées de mieux interagir avec leur environnement de manière autonome. Suivant le degré du handicap, il peut arriver qu'une personne ne puisse plus contrôler que très peu de ses muscles, par exemple. Stephen Hawking (1942–2018), physicien mondialement connu, utilisait les mouvements des muscles de ses joues pour contrôler un programme de synthèse vocal après qu'il avait perdu le contrôle de la majeure partie du reste de sa musculature.

Cet exemple concret pourrait aussi être utilisé par les musiciens: ils utilisent en général leurs deux mains pour jouer de leur instrument et ne peuvent donc pas tourner les pages d'une partition. Les appareils usuels disposent pour cette raison d'une pédale. Certains musiciens, comme les organistes,



doivent cependant aussi utiliser leurs pieds pour jouer et pourraient utiliser un simple hochement de tête pour tourner les pages d'une partition automatiquement, par exemple.

Contrairement à l'exemple de cet exercice, dans lequel les valeurs sont concrètes et préprogrammées, la vision par ordinateur est souvent combinée à l'*apprentissage automatique* (*machine learning* en anglais). Le programme s'entraîne alors à reconnaître certains gestes lorsque de nombreux exemples et contre-exemples de chaque geste lui sont présentés. Le programme peut ainsi élaborer un modèle statistique de comment les exemples peuvent être interprétés.

## Mots clés et sites web

- Vision par ordinateur, machine vision, computer vision :  
[https://fr.wikipedia.org/wiki/Vision\\_par\\_ordinateur](https://fr.wikipedia.org/wiki/Vision_par_ordinateur)
- Apprentissage automatique, machine learning :  
[https://fr.wikipedia.org/wiki/Apprentissage\\_automatique](https://fr.wikipedia.org/wiki/Apprentissage_automatique)
- [https://fr.wikipedia.org/wiki/Stephen\\_Hawking#Maladie\\_et\\_poursuite\\_de\\_ses\\_travaux](https://fr.wikipedia.org/wiki/Stephen_Hawking#Maladie_et_poursuite_de_ses_travaux)
- [https://fr.wikipedia.org/wiki/Tourneur\\_de\\_pages](https://fr.wikipedia.org/wiki/Tourneur_de_pages)



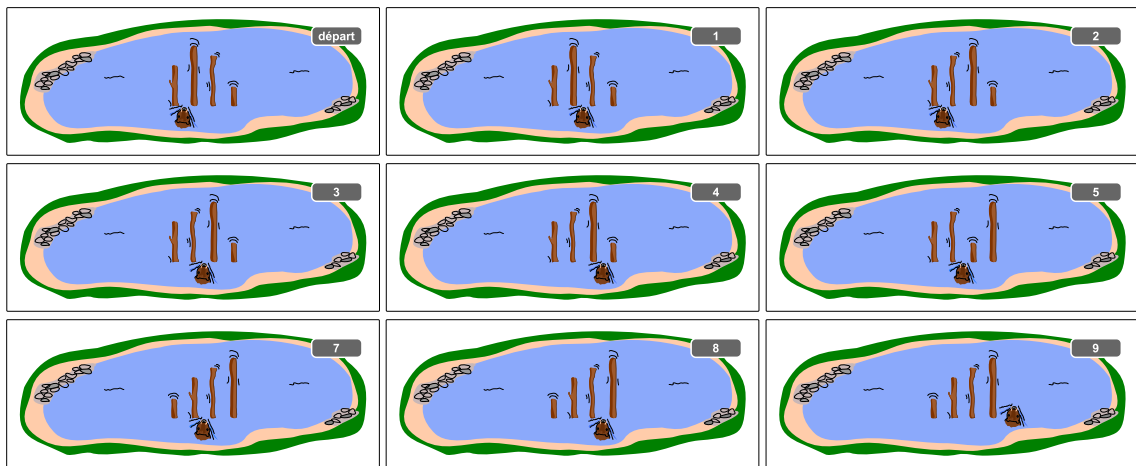


## 14. Beaver Sort

Hamid le castor trie les troncs d'arbre dans le lac. Ils doivent aller du plus petit au plus grand de gauche à droite.

- Hamid commence avec les deux troncs tout à gauche.
- Il compare toujours deux troncs voisins :
  - Si le tronc de droite est plus long que le tronc de gauche, il nage d'un tronc vers la droite.
  - Par contre, si le tronc de gauche est le plus long, il échange les deux troncs. Ensuite, il nage d'un tronc vers la droite s'il se trouve à sa position de départ et d'un tronc vers la gauche sinon.
- Hamid continue comme cela jusqu'à ce qu'il se trouve à la droite des troncs. Les troncs sont alors triés correctement.

L'exemple montre comment Hamid trie 4 troncs. Pour cela, il fait 9 comparaisons en tout.



Le nombre de comparaisons dépend de la configuration de départ des troncs. Pour trier 4 troncs, Hamid doit faire au moins 3 comparaisons (dans le cas où les troncs sont déjà triés correctement) et au maximum 9 comparaisons (dans le cas où les troncs sont dans l'ordre inverse). Hamid doit donc prévoir de faire entre 3 et 9 comparaisons pour trier 4 troncs.

*Hamid doit maintenant trier 40 troncs de longueurs différentes. Combien de comparaisons doit-il faire dans le meilleur et le pire des cas ?*

- A) de 0 à 20 comparaisons
- B) de 3 à 40 comparaisons
- C) de 39 à 120 comparaisons
- D) de 39 à 1521 comparaisons



## Solution

Dans le meilleur des cas, celui où tous les troncs sont déjà triés et où Hamid nage toujours vers la droite, Hamid ne doit faire que 39 comparaisons pour trier 40 troncs. Les réponses A) et B) ne peuvent donc pas être justes.

Dans le pire des cas, les troncs sont exactement dans l'ordre inverse. C'est plus difficile de déterminer le nombre de comparaisons nécessaires ici. On constate d'abord que les troncs à la gauche de Hamid sont toujours bien triés. Si Hamid arrive à un tronc qui est plus petit que les  $k$  troncs à sa gauche, il l'échange jusqu'à arriver tout à gauche, faisant ainsi déjà  $k$  comparaisons. Il revient et nage encore d'un tronc vers la droite, faisant  $k - 1$  comparaisons mais aucun échange. Il fait donc  $k + (k - 1) = 2k - 1$  comparaisons. Ceci a lieu pour chacune des  $n - 1$  positions entre deux troncs et Hamid a en moyenne  $\frac{n}{2}$  troncs à sa gauche. Hamid fait donc  $(n - 1) \cdot (2 \cdot \frac{n}{2} - 1) = (n - 1)^2$ , dans notre cas  $39^2 = 1521$ , comparaisons. Cela signifie que la réponse C) avec 120 comparaisons au maximum est fausse, et la bonne réponse est D).

## C'est de l'informatique !

Le tri d'éléments est une tâche classique en informatique. Une méthode de tri efficace peut permettre de gagner beaucoup de temps.

L'algorithme de tri présenté ici s'appelle *Gnome Sort* et a été introduit par l'informaticien iranien Hamid Sarbazi-Azad sous le nom de *Stupid Sort*. Plus tard, l'informaticien néerlandais Dick Grune a nommé cette méthode de tri Gnome Sort en imaginant qu'un nain de jardin (*garden gnome* en anglais) pouvait trier des pots de fleurs de cette façon.

L'analyse de la durée d'exécution d'un algorithme (c'est à dire du temps dont l'algorithme peut avoir besoin réaliser sa tâche) est très importante en informatique. On veut souvent savoir quelle est la durée dans le meilleur des cas, en moyenne et dans le pire des cas, et cela en fonction de la taille  $n$  de l'entrée (dans notre cas du nombre de troncs). Pour faciliter la comparaison de différentes durées d'exécution, on se contente la plupart du temps de donner un ordre de grandeur. La durée du Gnome Sort dans le meilleur des cas en fonction de  $n$  est linéaire — on écrit cela  $\mathcal{O}(n)$  —, la durée moyenne est de degré deux — on écrit cela  $\mathcal{O}(n^2)$ , et la durée dans le pire des cas est également de degré 2 — donc également  $\mathcal{O}(n^2)$ .

## Mots clés et sites web

- Gnome Sort : <https://de.wikipedia.org/wiki/Gnomesort>
- Algorithme de tri : [https://fr.wikipedia.org/wiki/Algorithme\\_de\\_tri](https://fr.wikipedia.org/wiki/Algorithme_de_tri)
- Théorie de la complexité : [https://fr.wikipedia.org/wiki/Théorie\\_de\\_la\\_complexité\\_\(informatique\\_théorique\)#Relation\\_au\\_coût\\_énergétique](https://fr.wikipedia.org/wiki/Théorie_de_la_complexité_(informatique_théorique)#Relation_au_coût_énergétique)
- O-Notation, Landau-Symbole :  
[https://fr.wikipedia.org/wiki/Comparaison\\_asymptotique](https://fr.wikipedia.org/wiki/Comparaison_asymptotique)



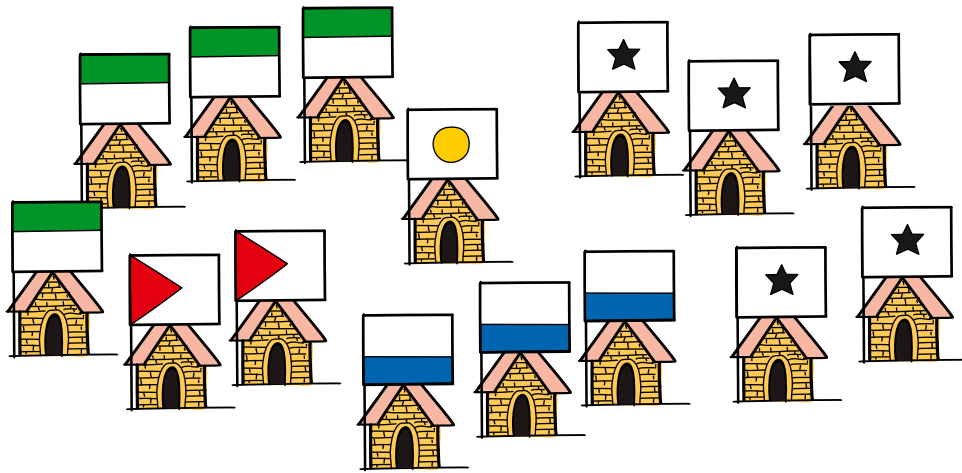


## 15. Les clans de Castorland

À Castorland vivent cinq clans, autrefois ennemis, qui possèdent chacun un certain nombre de maisons, comme on peut le voir dans l'image : les Mac Intosh, les Applondt, les Mac Rosoft, les Androidier et les Libredosset. Comme ils vivent en paix depuis longtemps, ils décident d'accomplir le rituel d'union. Les règles de ce rituel sont les suivantes :

- Seuls deux clans peuvent s'unir en même temps.
- Une fête d'une semaine a lieu dans chacune des maisons appartenant aux clans qui s'unissent afin de sceller le pacte. La durée du rituel de l'union, en semaines, est donc égale au nombre de maisons des deux clans.
- Après ce rituel, les deux clans ne forment plus qu'un clan. L'union des clans peut alors continuer.

Les clans décident de procéder à l'union en le moins de temps possible. Pour cela, il faut bien planifier l'ordre dans lequel les unions se feront.



*Combien de semaines au minimum l'union de tous les clans dure-t-elle ?*

- A) 15 semaines
- B) 33 semaines
- C) 35 semaines
- D) 50 semaines
- E) 120 semaines

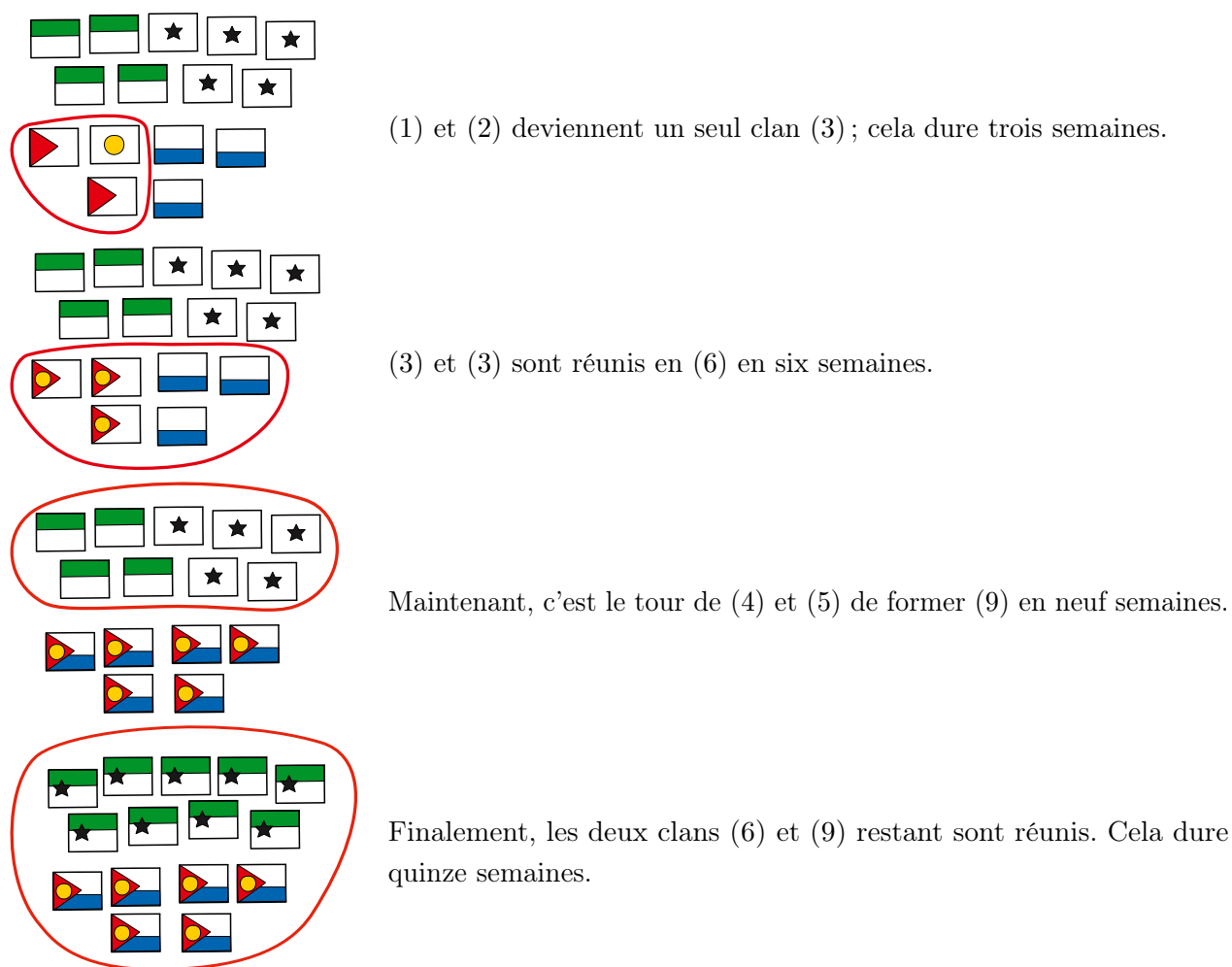


## Solution

La bonne réponse est B) 33 semaines.

La stratégie optimale pour minimiser le nombre total de semaines nécessaire à l'union de tous les clans consiste à minimiser le nombre de maisons y prenant part. Les maisons des clans unis au début doivent aussi être prises en compte dans les unions suivantes, c'est donc raisonnable de commencer avec les petits clans afin que les grands clans ne doivent pas participer au rituel trop souvent. Pour cela, à chaque étape, les deux clans avec le moins de maisons devraient être unis.

Les étapes du processus sont représentées dans le tableau suivant ; pour avoir une meilleure vue d'ensemble, nous laissons les noms des clans de côté et mentionnons seulement la taille des clans entre parenthèses :



Le processus est donc terminé après  $15 + 9 + 6 + 3 = 33$  semaines, ce qui correspond à la réponse B).



## C'est de l'informatique !

Cet exercice est un *problème d'optimisation* — une tâche consistant à minimiser (dans ce cas, la durée du rituel d'union) ou maximiser une valeur. Souvent, des algorithmes sont utilisés pour cela, dont on ne peut aujourd'hui plus se passer pour des méthodes de production écologiques qui économisent les ressources, par exemple

Dans notre cas, le problème peut être résolu de manière optimale à l'aide d'un simple *algorithme glouton*. Celui-ci fait à chaque étape ce qui semble être le mieux d'après un critère simple. Dans notre cas, le critère est que les clans doivent être les plus petits parce que cela prend moins de temps.

## Mots clés et sites web

- Optimisation - [https://fr.wikipedia.org/wiki/Optimisation\\_\(mathématiques\)](https://fr.wikipedia.org/wiki/Optimisation_(mathématiques))
- Algorithme glouton - [https://fr.wikipedia.org/wiki/Algorithme\\_glouton](https://fr.wikipedia.org/wiki/Algorithme_glouton)



## A. Auteur·e·s des exercices

-  Michael Barot
-  Liam Baumann
-  Wilfried Baumann
-  Lucia Budinská
-  Sarah Chan
-  Anton Chukhnov
-  Kris Coolsaet
-  Valentina Dagienė
-  Christian Datzko
-  Susanne Datzko
-  Janez Demšar
-  Fabian Frei
-  Gerald Futschek
-  Jens Gallenbacher
-  Thomas Galler
-  Mark Edward M. Gonzales
-  Martin Guggisberg
-  Mathias Hiron
-  Juraj Hromkovič
-  Svetlana Jaksic
-  Martin Kandlhofer
-  Ulrich Kiesmüller
-  Dong Yoon Kim
-  Vaidotas Kinčius
-  Regula Lacher
-  Taina Lehtimäki
-  Angélica Herrera Loyo
-  Tom Naughton
-  Mochammad Irfan Noviana
-  Gabriela Lourdes Rodríguez Parada
-  Elsa Pellet
-  Jean-Philippe Pellet
-  Hannah Piper
-  Jonatan Pipping
-  Zsuzsa Pluhár
-  Wolfgang Pohl
-  Rodrigo Santamaría
-  Tomas Šiaulys
-  Bernadette Spieler
-  Cuttle.org Team
-  Ezra Templonuevo
-  Eslam Wageed
-  Michael Weigend
-  Mija Zaļuksne



## B. Sponsoring : Concours 2021

**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.phzh.ch/>  
Pädagogische Hochschule Zürich



## C. Offres ultérieures

010100110101011001001001  
010000010010110101010011  
010100110100100101000101  
001011010101001101010011  
010010010100100100100001

**SS!E**

[www.svia-ssie-ssii.ch](http://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.