INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Exercices et solutions 2018
Années scolaires 11/12/13

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

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

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

Ont collaboré au Castor Informatique 2018

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

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

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

INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

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

HASLERSTIFTUNG

Tous les liens ont été vérifiés le 1er novembre 2018. Ce cahier d'exercice a été produit le 26 novembre 2018 avec avec le logiciel de mise en page LaTeX.

Préambule

Très bien établi dans différents pays européens depuis plusieurs années, le concours « Castor Informatique » a pour but d'éveiller l'intérêt des enfants et des jeunes pour l'informatique. En Suisse, le concours est organisé en allemand, en français et en italien par la SSIE, la Société Suisse pour l'Informatique dans l'Enseignement, et soutenu par la Fondation Hasler dans le cadre du programme d'encouragement « FIT in IT ».
Le Castor Informatique est le partenaire suisse du concours « Bebras International Contest on Informatics and Computer Fluency » (`https://www.bebras.org/`), initié en Lituanie.
Le concours a été organisé pour la première fois en Suisse en 2010. Le Petit Castor (5e et 6e HarmoS / Castor 3 et 4) a été organisé pour la première fois en 2012.
Le Castor Informatique vise à motiver les élèves à apprendre l'informatique. Il souhaite lever les réticences et susciter l'intérêt quant à l'enseignement de l'informatique à l'école. Le concours ne suppose aucun prérequis quant à l'utilisation des ordinateurs, sauf de savoir naviguer sur Internet, car le concours s'effectue en ligne. Pour répondre, il faut structurer sa pensée, faire preuve de logique mais aussi de fantaisie. Les exercices sont expressément conçus pour développer un intérêt durable pour l'informatique, au-delà de la durée du concours.
Le concours Castor Informatique 2018 a été fait pour cinq tranches d'âge, basées sur les années scolaires :

- 5e et 6e HarmoS / Castor 3 et 4 (Petit Castor)
- 7e et 8e HarmoS / Castor 5 et 6
- 9e et 10e HarmoS / Castor 7 et 8
- 11e et 12e HarmoS / Castor 9 et 10
- 13e à 15e HarmoS / Castor 11 à 13

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

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

Utilisé au niveau international, ce système de distribution des points est conçu pour limiter le succès en cas de réponses données au hasard.
Chaque participant·e obtenait initialement 45 points (ou 27 pour la tranche d'âge « Petit Castor », et 36 pour les 7e et 8e années HarmoS / Castor 5 et 6).
Le nombre de points maximal était ainsi de 180 (ou 108 pour la tranche d'âge « Petit Castor », et 144 pour les 7e et 8e années HarmoS / Castor 5 et 6). Le nombre de points minimal était zéro.
Les réponses de nombreux exercices étaient affichées dans un ordre établi au hasard. Certains exercices ont été traités par plusieurs tranches d'âge.

Pour de plus amples informations :

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

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

Table des matières

Ont collaboré au Castor Informatique 2018 **i**

Préambule **ii**

1. Jeu vidéo **1**

2. Tournée des castors **3**

3. Deux castors au travail **7**

4. Marelle **9**

5. Cadeaux **11**

6. Rangées et colonnes **13**

7. Classement de livres **17**

8. Soundex **21**

9. Tour de cartes **23**

10. Catelles **27**

11. Où est le planeur ? **31**

12. Horaire de répétition **35**

13. Laboratoire **39**

14. Lumière ! **41**

15. Top secret **45**

A. Auteurs des exercices **48**

B. Sponsoring : Concours 2018 **49**

C. Offres ultérieures **51**

1. Jeu vidéo

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

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

Quelle est la durée maximale d'un jeu ?

A) 4 tours

B) 15 tours

C) 18 tours

D) 19 tours

E) 20 tours

F) La durée du jeu est illimitée.

Solution

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

C'est de l'informatique !

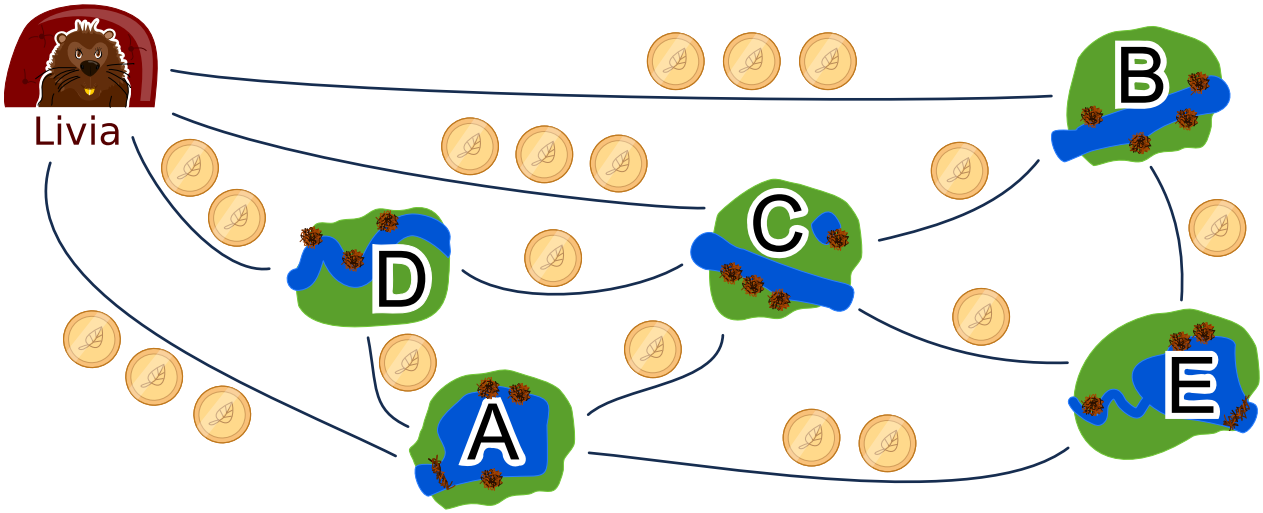
Lors de la programmation d'un jeu, les règles doivent être clairement définies. Les conséquences des règles doivent être bien comprises afin que le jeu permette de gagner et de perdre (le nombre de feuilles doit être suffisant) et que le jeu ne dure ni trop longtemps, ni pas assez.
Un jeu consistant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

Mots clés et sites web

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

2. Tournée des castors

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



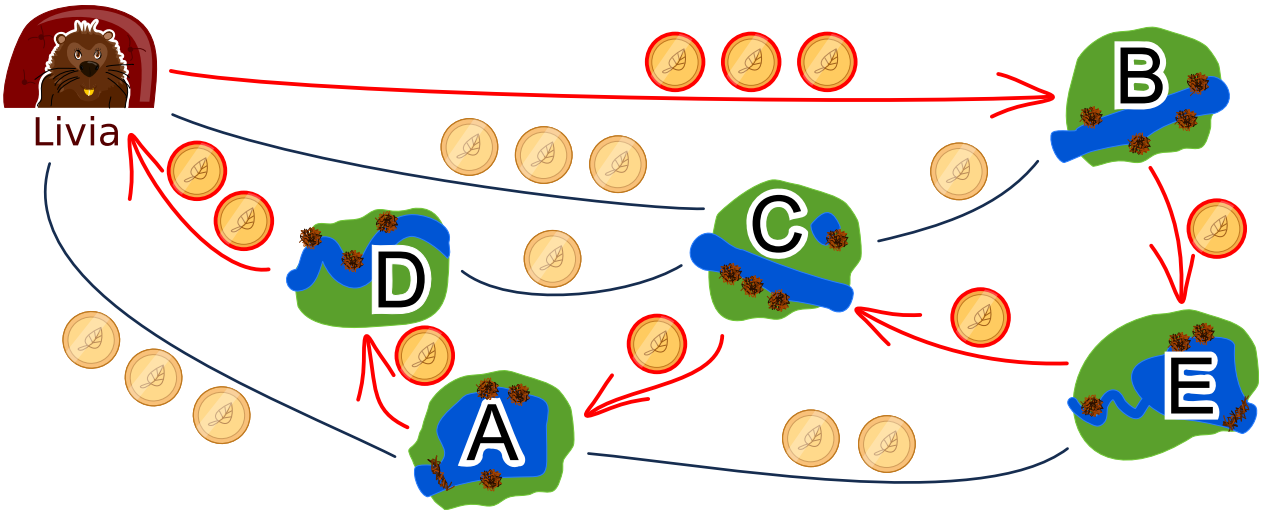
Une des routes possible pour voir ses amis est :

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

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

Solution

Il existe deux solutions optimales :
- départ → B → E → C → A → D → départ
- départ → D → A → C → E → B → départ



Les deux solutions sont semblables, mis à part le sens de trajet, et coûtent 9 francs castor. Il n'y a pas de meilleure solution. Depuis chez elle, Livia peut prendre une fois le chemin à 2 francs castor puis un chemin à 3 francs castor dans l'autre sens. Les quatre autres nœuds à visiter correspondent à quatre autres chemins coûtant au moins un franc castor chacun, ce qui fait déjà 9 francs castor au total.
Toutes les autres solutions sont plus chères :
- Coût de 10 francs castor : départ → A → D → C → E → B → départ
- Coût de 10 francs castor : départ → A → E → B → C → D → départ
- Coût de 10 francs castor : départ → B → C → E → A → D → départ
- Coût de 10 francs castor : départ → B → E → A → C → D → départ
- Coût de 10 francs castor : départ → B → E → C → D → A → départ
- Coût de 10 francs castor : départ → C → B → E → A → D → départ
- Coût de 10 francs castor : départ → D → A → E → B → C → départ
- Coût de 10 francs castor : départ → D → A → E → C → B → départ
- Coût de 10 francs castor : départ → D → C → A → E → B → départ
- Coût de 10 francs castor : départ → D → C → B → E → A → départ
- Coût de 11 francs castor : départ → B → E → A → D → C → départ
- Coût de 11 francs castor : départ → C → D → A → E → B → départ

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

C'est de l'informatique !

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

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

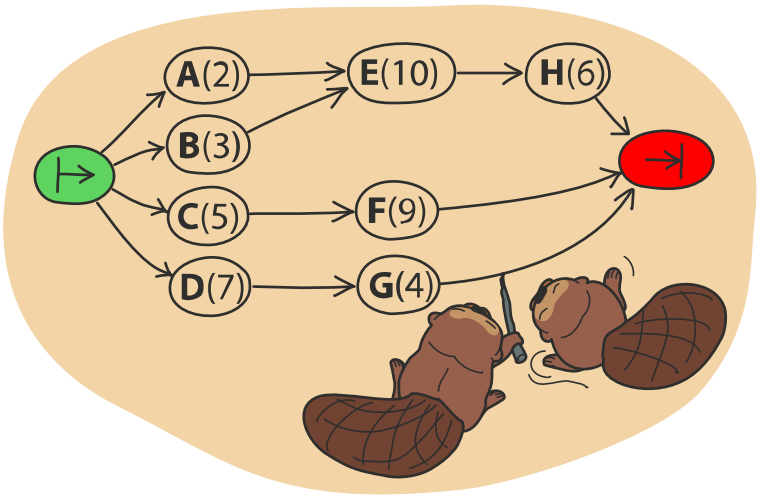
Mots clés et sites web

Optimisation, problème du voyageur de commerce

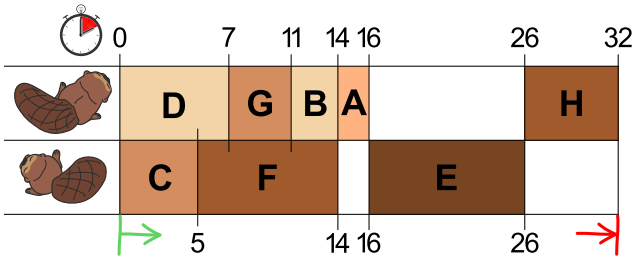
- `https://fr.wikipedia.org/wiki/Problème_du_voyageur_de_commerce`
- `https://en.wikipedia.org/wiki/Optimization_problem`
- `https://fr.wikipedia.org/wiki/Optimisation_(mathématiques)`

3. Deux castors au travail

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



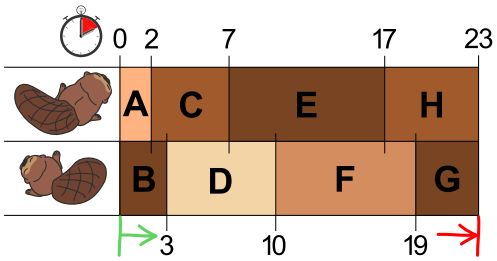
L'image ci-dessous montre un plan de travail possible pour les deux castors qui prévoit 32 heures de travail en tout, mais c'est possible de réaliser le barrage plus rapidement !



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

Solution

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



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

C'est de l'informatique !

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

Mots clés et sites web

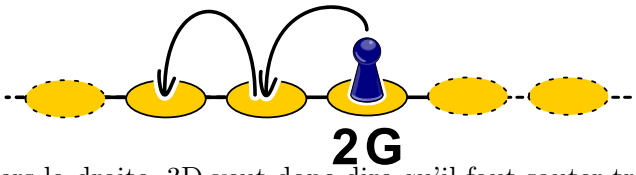
Ordonnancement, algorithme glouton

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

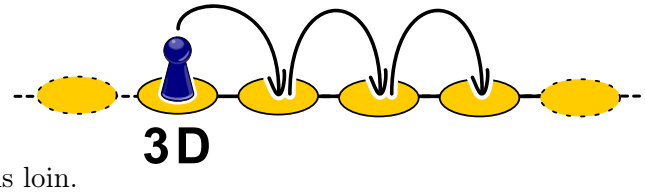
4. Marelle

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

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

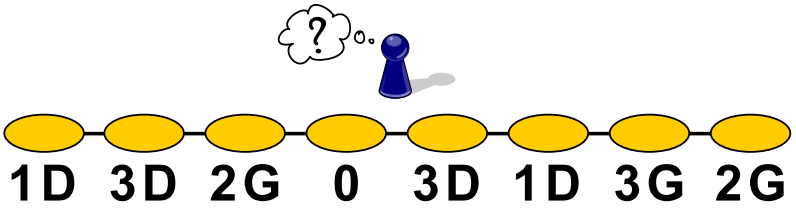


— nD : sauter n cases vers la droite, 3D veut donc dire qu'il faut sauter trois fois vers la droite.



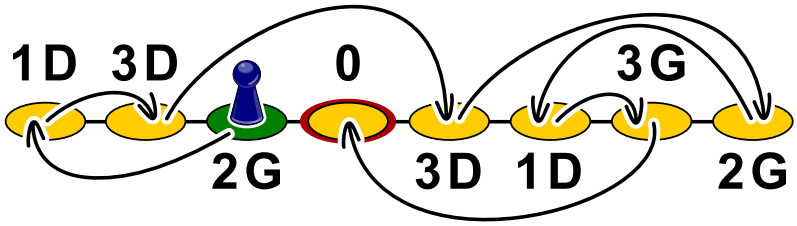
— 0 : ne pas sauter plus loin.

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



Solution

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



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

C'est de l'informatique !

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

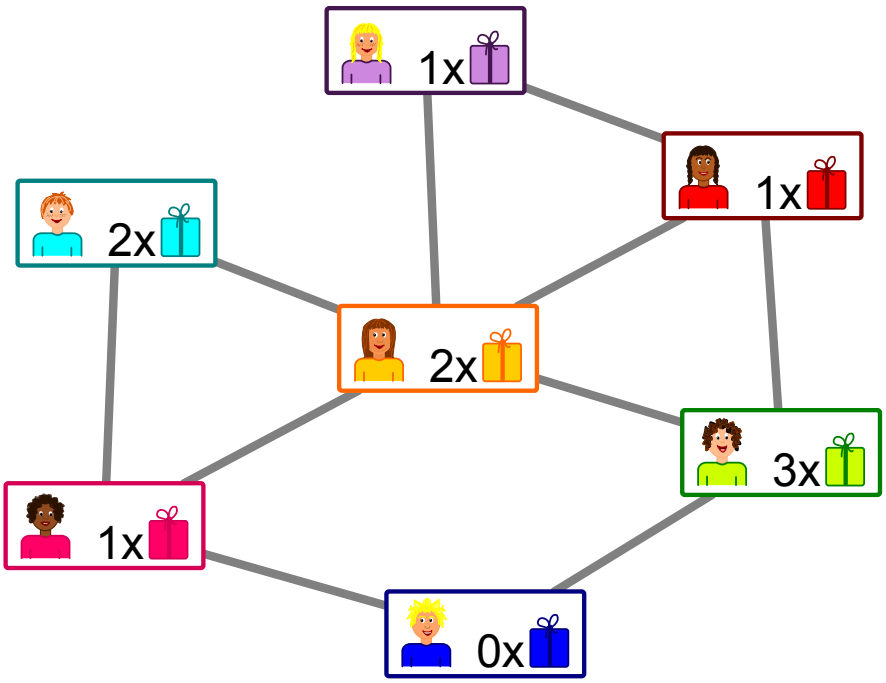
Mots clés et sites web

File, gestion de la mémoire, GOTO

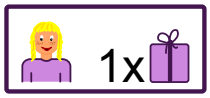
- `https://fr.wikipedia.org/wiki/Ramasse-miettes_(informatique)`
- `https://en.wikipedia.org/wiki/St-connectivity`

5. Cadeaux

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



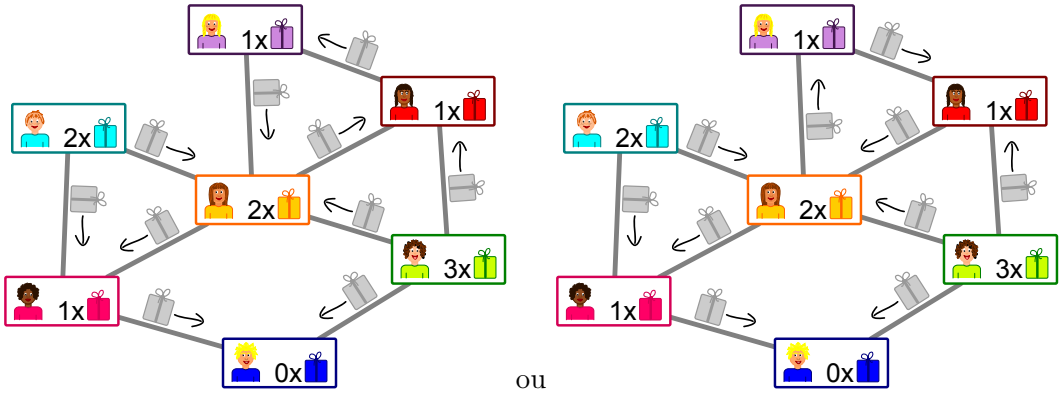
Les habitants de l'immeuble organisent une fête avec des cadeaux pour les enfants. L'un des enfants de chaque paire d'amis doit offrir un cadeau à l'autre.

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

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

Solution

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



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

C'est de l'informatique !

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

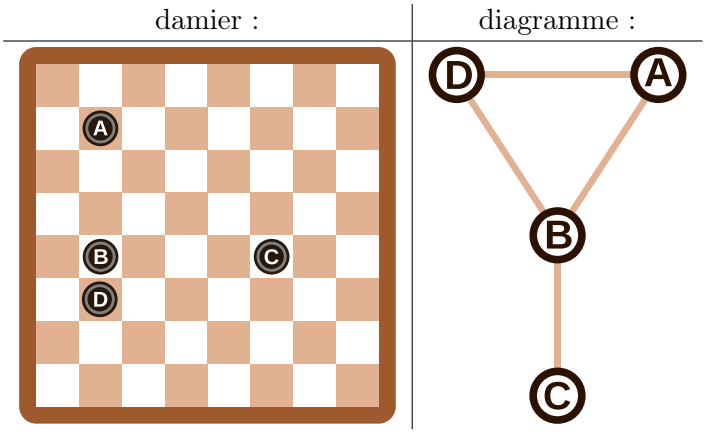
Mots clés et sites web

Débit dans un réseau, réduction de problème

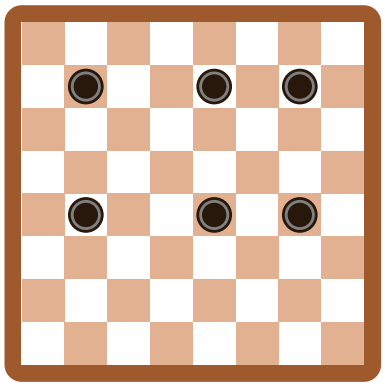
- `https://fr.wikipedia.org/wiki/Problème_de_flot_maximum`
- `https://fr.wikipedia.org/wiki/Réduction_(complexité)`

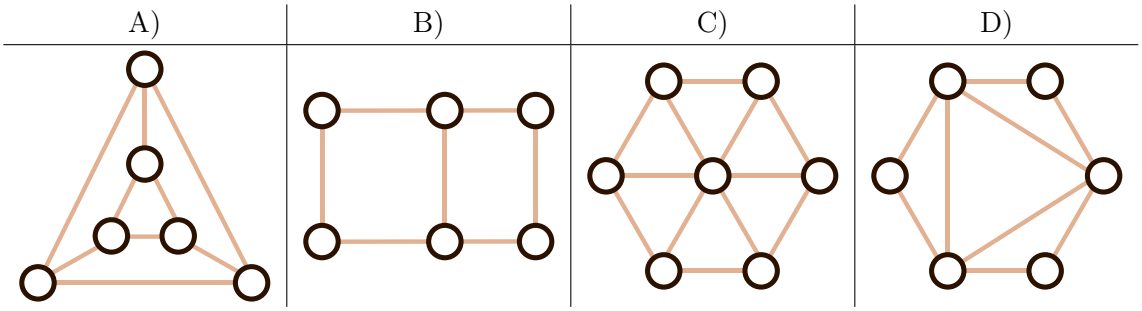
6. Rangées et colonnes

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



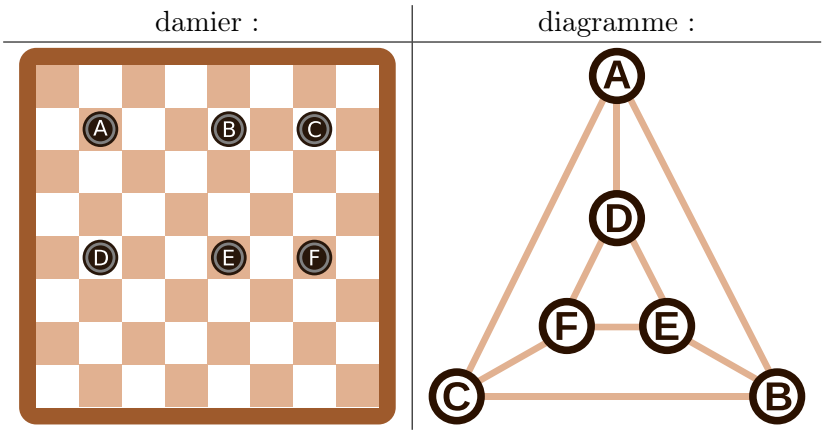
Pour cet exemple, les palets sur le damier et les cercles du diagramme sont annotés d'une lettre afin de mettre leur relation en évidence.
Quel diagramme correspond au damier à six palets suivant ?



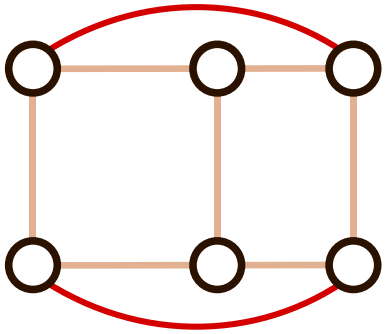


Solution

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



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



C'est de l'informatique !

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

Mots clés et sites web

Graphe

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

7. Classement de livres

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

A. Chaque castor peut (mais ne doit pas) inverser les deux livres sur sa table (exemple A).

B. Chaque livre peut (mais ne doit pas) être échangé avec le livre le plus proche sur une table voisine (exemple B).

Au départ, les livres sont placés comme suit :

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

A) trois tours

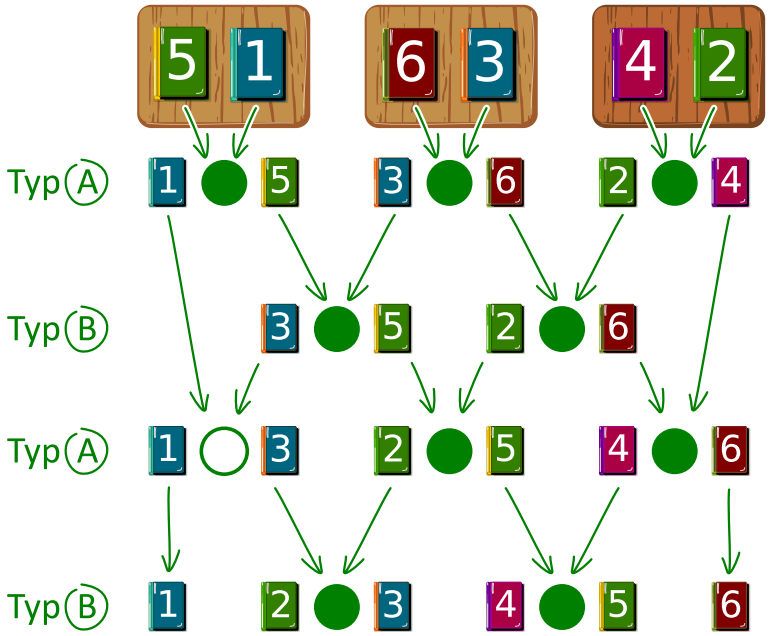
B) quatre tours

C) cinq tours

D) six tours

Solution

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



C'est de l'informatique !

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

Mots clés et sites web

Tri parallèle, réseau de tri

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

8. Soundex

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

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

Les mots suivants sont encodés comme suit :

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

Quel est le code pour le mot « Hilbert » ?

A) H410

B) B540

C) H041

D) H416

Solution

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

C'est de l'informatique !

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

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

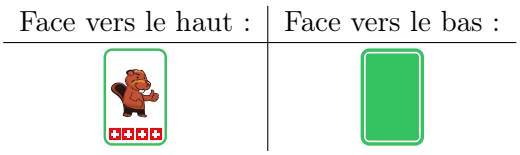
Mots clés et sites web

Recherche phonétique, Soundex

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

9. Tour de cartes

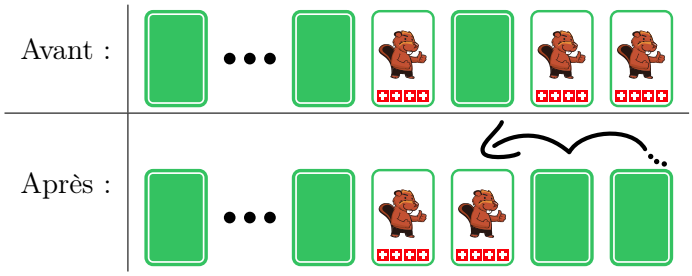
Tu reçois un paquet de cartes à jouer toutes pareilles. Elles sont comme cela :



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

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

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



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

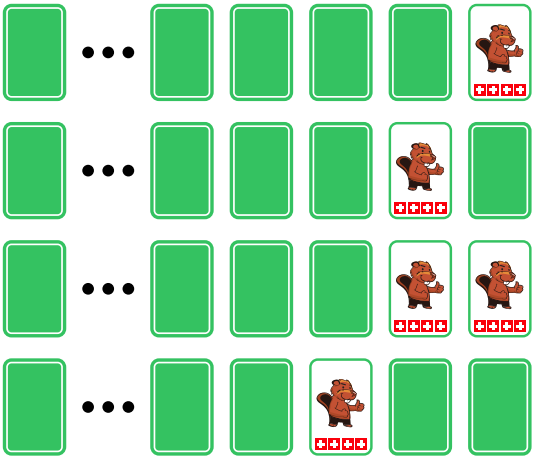
Combien de cartes sont face vers le haut après 16 tours ?

Solution

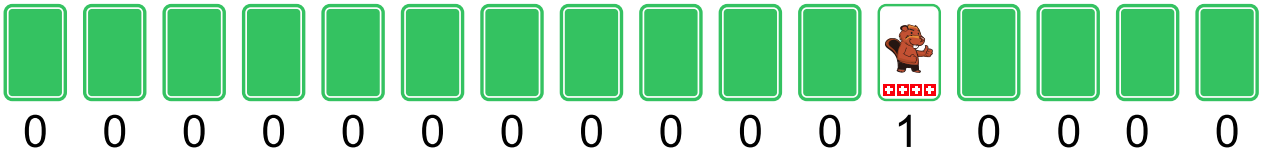
Il y a exactement une carte face vers le haut.
On peut se représenter une rangée de cartes pouvant être face vers le bas ou vers le haut comme un nombre binaire. Les nombres binaires ne sont composés que des chiffres 0 et 1. Par exemple, une carte face vers le bas peut représenter le chiffre 0 et une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

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

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



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



C'est de l'informatique !

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

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

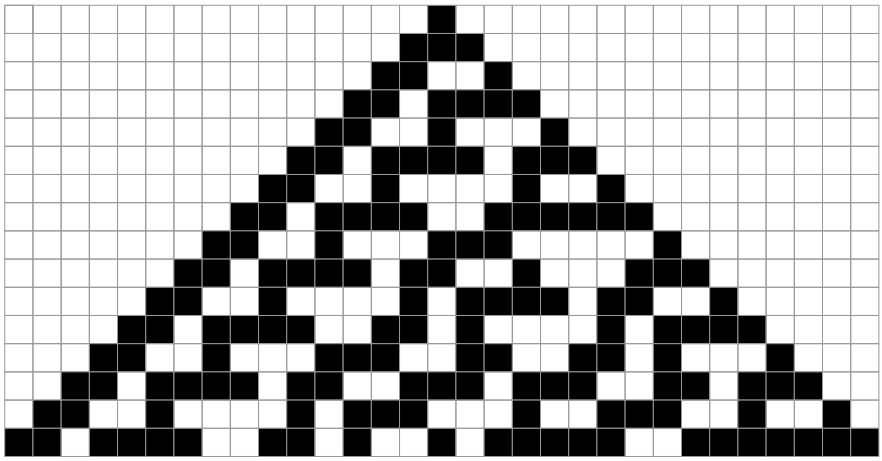
Mots clés et sites web

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

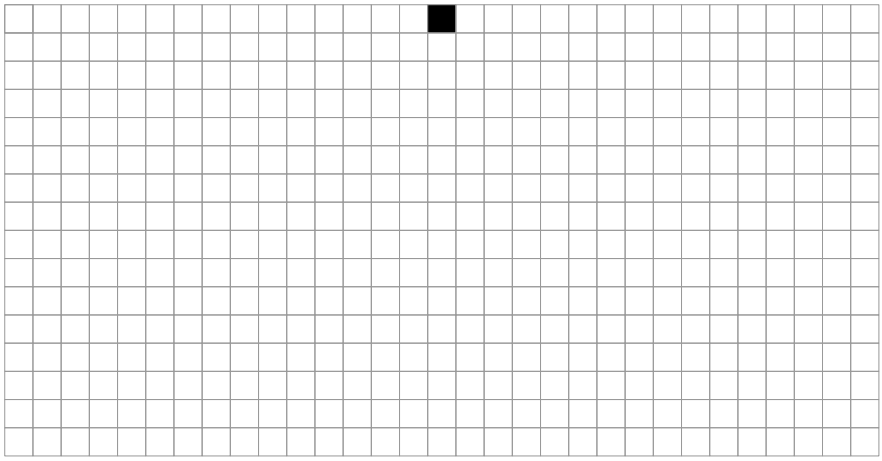
10. Catelles

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

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



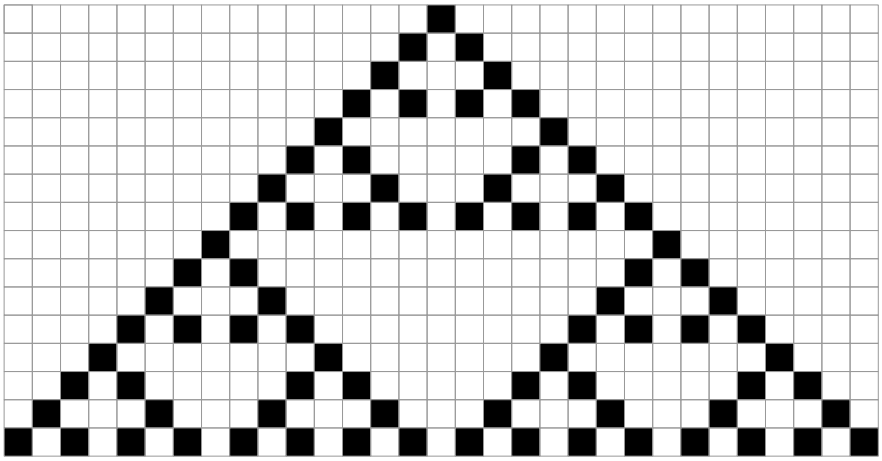
Etablis ton propre set de règles de façon à ce que la rangée du bas de la surface soit constituée tour à tour d'une catelle noire et d'une catelle blanche.

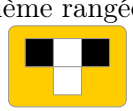


Solution

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

Elle génère le motif suivant :



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

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

C'est de l'informatique !

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

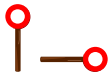
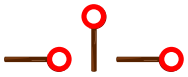
Mots clés et sites web

Motif, automate cellulaire

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

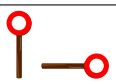
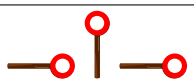
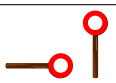
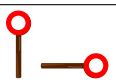
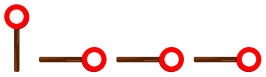
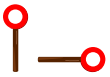
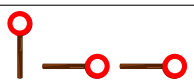
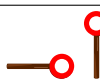
11. Où est le planeur ?

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

gauche	droite	direction colline	direction vallée
			

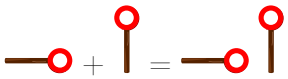
Il y a malheureusement un problème avec ce code. Si l'on envoie par exemple l'instruction suivante…

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

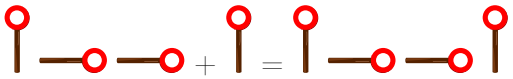
	gauche	droite	direction colline	direction vallée
A)				
B)				
C)				
D)				

Solution

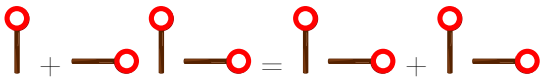
La bonne réponse est C). Cela peut être déterminé, par exemple, en observant que chaque instruction commence par le signal  et est suivie de 0, 1, 2 ou 3 fois  , sans que  ne réapparaisse. De cette manière, on sait qu'à chaque  , une nouvelle instruction commence et on ne doit que compter combien de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



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

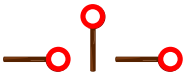
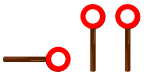


C'est un peu plus compliqué pour la réponse A). Si l'on signale « direction vallée » puis « gauche », c'est la même chose que si l'on signale deux fois « droite » :



C'est de l'informatique !

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

gauche	droite	direction colline	direction vallée
			

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

Mots clés et sites web

Code, code préfixe, disque de signalisation

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

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

12. Horaire de répétition

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

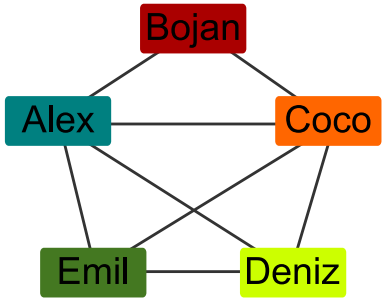
— Alex – Bojan
— Coco – Alex
— Émile – Deniz
— Alex – Émile
— Coco – Deniz
— Bojan – Coco
— Deniz – Alex
— Coco – Émile

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

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

Solution

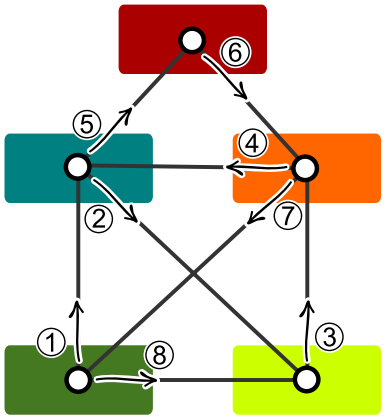
B) Bojan est la bonne réponse.
Chacun des danseurs est représenté par un rectangle encadrant son nom dans le diagramme suivant. Les rectangles représentant des danseurs qui font partie du même binôme sont reliés par une ligne. Un horaire valable peut alors être représenté par un chemin traversant le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



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

C'est de l'informatique !

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



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

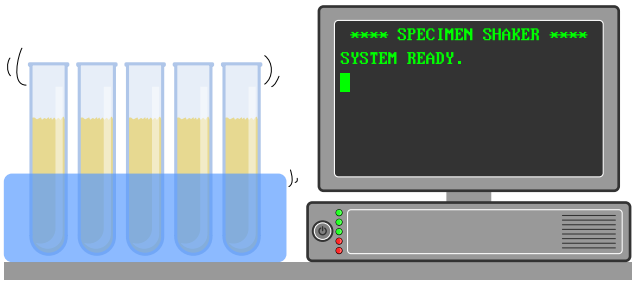
Mots clés et sites web

Graphe, chemin eulérien

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

13. Laboratoire

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




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

Les instructions possibles sont :

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

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

A) L'échantillon n'est pas secoué.

B) L'échantillon est secoué une fois.

C) L'échantillon est secoué 60 fois.

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

Solution

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

C'est de l'informatique !

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

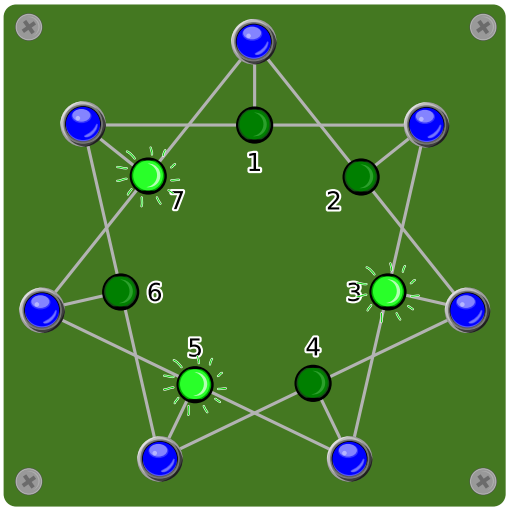
Mots clés et sites web

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

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

14. Lumière !

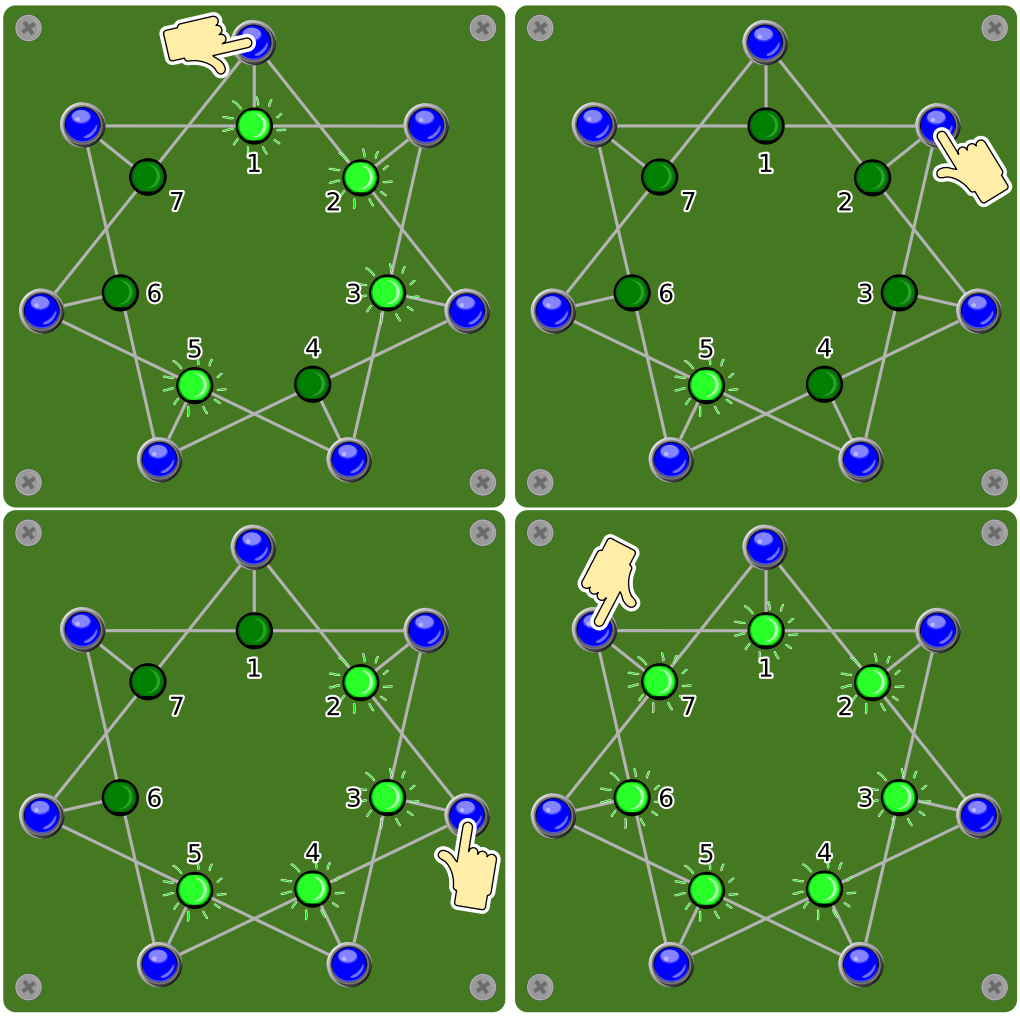
Sept interrupteurs sont reliés à sept lampes, et cela de manière à ce que chaque interrupteur contrôle trois lampes. Lorsque l'on appuie sur un interrupteur, les lampes y étant reliées s'éteignent si elles étaient allumées et s'allument si elles étaient éteintes.



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

Solution

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



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

C'est de l'informatique !

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

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

Mots clés et sites web

Recherche bidirectionnelle, automate fini

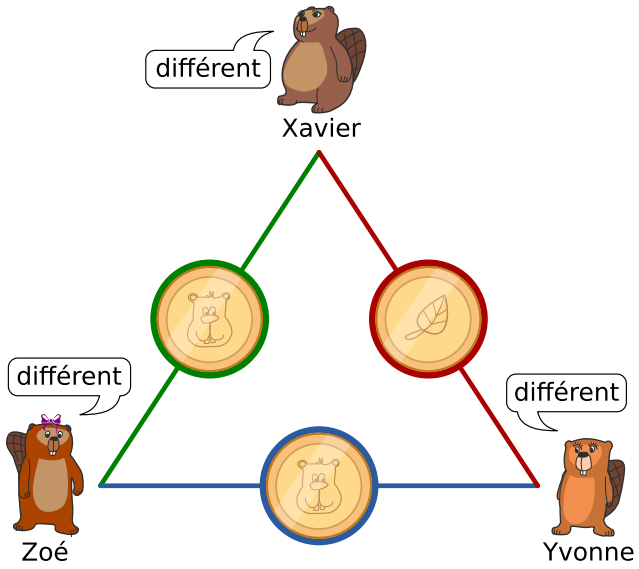
- `https://en.wikipedia.org/wiki/Bidirectional_search`
- `https://fr.wikipedia.org/wiki/Parcours_de_graphe`

15. Top secret

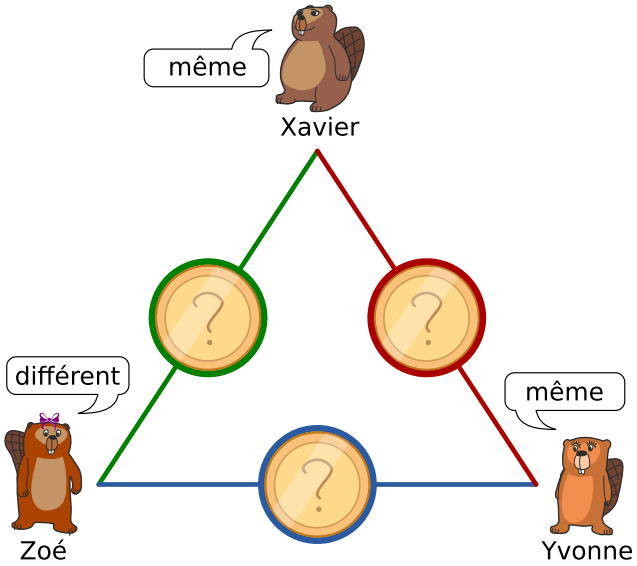
Xavier, Yvonne et Zoé sont amis et jouent de temps en temps à la tombola. Ils viennent de découvrir qu'une personne habitant leur ville a gagné le gros lot. Ils aimeraient savoir si, par hasard, l'un d'entre eux a gagné ; d'un autre côté, l'identité du gagnant doit rester secrète. Ils procèdent de la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

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

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



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



Laquelle des affirmations suivantes est vraie ?

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

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

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

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

Solution

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

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

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

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

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

C'est de l'informatique !

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

Mots clés et sites web

Anonymat, Dining Cryptographers Problem

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

A. Auteurs des exercices

Andrea Adamoli
Jared Asuncion
Wilfried Baumann
Daphne Blokhuis
William Chan
Kessarapan Charoensueksa
Anton Chukhnov
Kris Coolsaet
Valentina Dagienė
Christian Datzko
Susanne Datzko
Marissa Engels
Hanspeter Erni
Veerle Fack
Gerald Futschek
Ionuț Gorgos
Shuchi Grover
Martin Guggisberg
Urs Hauser
Juraj Hromkovič
Svetlana Jakšić
Zhang Jinbao
Emil Kelevedjiev
Dong Yoon Kim
Vaidotas Kinčius
Iryna Kirynovich
Regula Lacher
Judith Lin
Violetta Lonati
Nils Mak
Dimitris Mavrovouniotis
Mattia Monga
Anna Morpurgo
Tom Naughton
Henry Ong
Wolfgang Pohl
Stavroula Prantsoudi
Nol Premasathian
J.P. Pretti
Doris Reck
Chris Roffey
Kirsten Schlüter
Andrea Maria Schmid
Victor Schmidt
Andrea Schrijvers
Eljakim Schrijvers
Vipul Shah
Jacqueline Staub
Allira Storey
Ahto Truu
Willem van der Vegt
Jiří Vaníček
Troy Vasiga
Rechilda Villame
Eslam Wageed
Pieter Waker
Michael Weigend

B. Sponsoring : Concours 2018

http://www.haslerstiftung.ch/

http://www.roborobo.ch/

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

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

Standortförderung beim Amt für Wirtschaft und Arbeit Kanton Zürich

i-factory (Musée des transports, Lucerne)

http://www.ubs.com/

http://www.bbv.ch/

http://www.presentex.ch/

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

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

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

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

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

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

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

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

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

C. Offres ultérieures

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