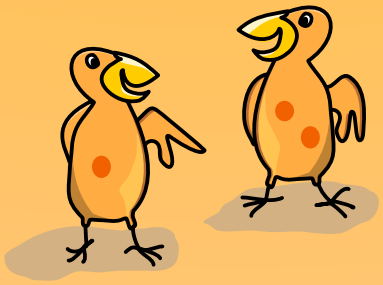
INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Quesiti e soluzioni 2021

7º e 8º anno scolastico



https://www.castoro-informatico.ch/

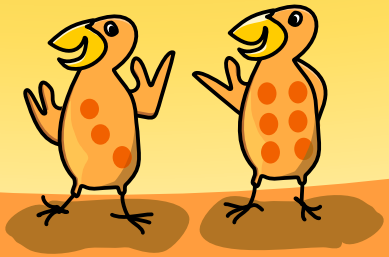
A cura di:

Susanne Datzko, Masiar Babazadeh, Christian Giang,
Fabian Frei, Jean-Philippe Pellet

01010011010101100100100
10000001001011010101001
10101001101001001010001
00101101010100110101001
10100100101001001001000

SSII

www.svia-ssie-ssii.ch
schweizerischerverein für informatikin
derausbildung//société suisse pour l'infor
matiquedans l'enseignement//società sviz
zera per l'informatica nell'insegnamento



Hanno collaborato al Castoro Informatico 2021

Masiar Babazadeh, Susanne Datzko, Fabian Frei, Martin Guggisberg, Gabriel Parriaux, Jean-Philippe Pellet

Capo progetto: Nora A. Escherle

Un particolare ringraziamento per il lavoro sui quesiti del concorso Svizzero va a:
Juraj Hromkovič, Michael Barot, Christian Datzko, Jens Gallenbacher, Dennis Komm, Regula Lacher, Peter Rossmanith: ETH Zürich, Ausbildunges- und Beratungszentrum für Informatikunterricht
Bernadette Spieler: Pädagogische Hochschule Zürich

La scelta dei quesiti è stata svolta in collaborazione con gli organizzatori dei concorsi in Germania, Austria, Ungheria, Slovacchia e Lituania. Ringraziamo specialmente:
Valentina Dagienė, Tomas Šiaulys, Vaidotas Kinčius: Bebras.org
Wolfgang Pohl, Hannes Endreß, Ulrich Kiesmüller, Kirsten Schlüter, Michael Weigend: Bundesweite Informatikwettbewerbe (BWINF), Germania
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, Ungheria
Michal Winzcer: Comenius University, Slovacchia

La versione online del concorso è stata creata su cuttle.org. Ringraziamo per la buona collaborazione:
Eljakim Schrijvers, Justina Dauksaite, Arne Heijenga, Dave Oostendorp, Andrea Schrijvers, Alieke Stijf, Kyra Willekes: cuttle.org, Olanda
Chris Roffey: UK Bebras Administrator, Regno Unito

Per il supporto durante le settimane del concorso ringraziamo:
Hanspeter Erni: Direttore scuola media di Rickenbach

Christoph Frei: Chragokyberneticks (Logo Informatik-Biber Schweiz)
Dr. Andrea Leu, Maggie Winter, Brigitte Manz-Brunner: Senarclens Leu + Partner AG

Questi quaderni sono dedicati alla memoria di Martin Guggisberg.

L'edizione dei quesiti in lingua tedesca è stata utilizzata anche in Germania e in Austria.

La traduzione francese è stata curata da Elsa Pellet mentre quella italiana da Christian Giang.

INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Il Castoro Informatico 2021 è stato organizzato dalla Società Svizzera per l'Informatica nell'Insegnamento SSII con il sostegno della fondazione Hasler.

HASLERSTIFTUNG

Questo quaderno è stato creato il 24 agosto 2022 con il sistema per la preparazione di testi LaTeX. Ringraziamo Christian Datzko per lo sviluppo del sistema di generazione dei testi che ha permesso di generare le 36 versioni di questa brochure (divise per lingua e livello scolastico). Il sistema è stato riprogrammato basandosi sul sistema precedente, sviluppato nel 2014 assieme a Ivo Blöchliger. Ringraziamo Jean-Philippe Pellet per lo sviluppo del sistema `bebras`, utilizzato dal 2020 per la conversione dei documenti sorgente dai formati Markdown e YAML.

Nota: Tutti i link sono stati verificati l'01.12.2021.

Premessa

Il concorso del «Castoro Informatico», presente già da diversi anni in molti paesi europei, ha l'obiettivo di destare l'interesse per l'informatica nei bambini e nei ragazzi. In Svizzera il concorso è organizzato in tedesco, francese e italiano dalla Società Svizzera per l'Informatica nell'Insegnamento (SSII), con il sostegno della fondazione Hasler nell'ambito del programma di promozione «FIT in IT».

Il Castoro Informatico è il partner svizzero del Concorso «Bebras International Contest on Informatics and Computer Fluency» (`https://www.bebras.org/`), situato in Lituania.

Il concorso si è tenuto per la prima volta in Svizzera nel 2010. Nel 2012 l'offerta è stata ampliata con la categoria del «Piccolo Castoro» (3º e 4º anno scolastico).

Il Castoro Informatico incoraggia gli alunni ad approfondire la conoscenza dell'informatica: esso vuole destare interesse per la materia e contribuire a eliminare le paure che sorgono nei suoi confronti. Il concorso non richiede alcuna conoscenza informatica pregressa, se non la capacità di «navigare» in internet poiché viene svolto online. Per rispondere alle domande sono necessari sia un pensiero logico e strutturato che la fantasia. I quesiti sono pensati in modo da incoraggiare l'utilizzo dell'informatica anche al di fuori del concorso.

Nel 2021 il Castoro Informatico della Svizzera è stato proposto a cinque differenti categorie d'età, suddivise in base all'anno scolastico:

- 3º e 4º anno scolastico («Piccolo Castoro»)
- 5º e 6º anno scolastico
- 7º e 8º anno scolastico
- 9º e 10º anno scolastico
- 11º al 13º anno scolastico

Alla categoria del 3º e 4º anno scolastico sono stati assegnati 9 quesiti da risolvere, di cui 3 facili, 3 medi e 3 difficili. Alla categoria del 5º e 6º anno scolastico sono stati assegnati 12 quesiti, suddivisi in 4 facili, 4 medi e 4 difficili. Ogni altra categoria ha ricevuto invece 15 quesiti da risolvere, di cui 5 facili, 5 medi e 5 difficili.

Per ogni risposta corretta sono stati assegnati dei punti, mentre per ogni risposta sbagliata sono stati detratti. In caso di mancata risposta il punteggio è rimasto inalterato. Il numero di punti assegnati o detratti dipende dal grado di difficoltà del quesito:

	Facile	**Medio**	**Difficile**
Risposta corretta	6 punti	9 punti	12 punti
Risposta sbagliata	−2 punti	−3 punti	−4 punti

Il sistema internazionale utilizzato per l'assegnazione dei punti limita l'eventualità che il partecipante possa ottenere buoni risultati scegliendo le risposte in modo casuale.

Ogni partecipante inizia con un punteggio pari a 45 punti (risp., Piccolo Castoro: 27 punti, 5º e 6º anno scolastico: 36 punti).

Il punteggio massimo totalizzabile era dunque pari a 180 punti (risp., Piccolo castoro: 108 punti, 5º e 6º anno scolastico: 144 punti), mentre quello minimo era di 0 punti.

In molti quesiti le risposte possibili sono state distribuite sullo schermo con una sequenza casuale. Lo stesso quesito è stato proposto in più categorie d'età.

Per ulteriori informazioni:

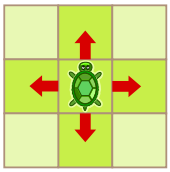
SVIA-SSIE-SSII Società Svizzera per l'Informatica nell'Insegnamento
Castoro Informatico
Lucio Negrini

`https://www.castoro-informatico.ch/it/kontaktieren/`
`https://www.castoro-informatico.ch/`

Indice

1. Cammino della tartaruga

Una tartaruga vuole passeggiare in diversi giardini. Ogni giardino è suddiviso in zone (quadrati) che sono coperte di erba o di pietre. La tartaruga non può passeggiare dove ci sono le pietre. Tuttavia, può spostarsi da un quadrato d'erba a un altro quadrato d'erba proprio accanto.

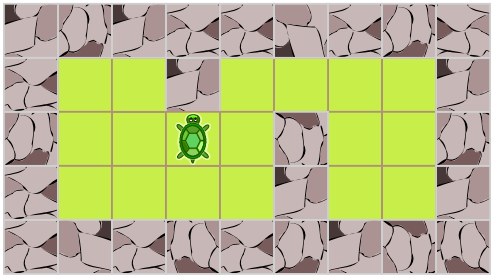


La tartaruga vuole passare per tutte le zone d'erba di ogni giardino. In ogni giardino comincia la sua passeggiata sulla zona dove si trova nel disegno. Alla fine del suo giro, la tartaruga vuole aver visitato tutte le zone del giardino esattamente una volta.

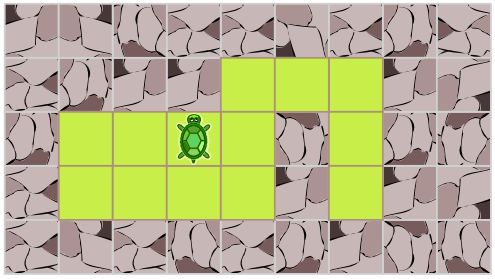
Sfortunatamente, la tartaruga non può visitare tutte le zone d'erba esattamente una volta su uno dei giardini.

Di quale giardino si tratta?

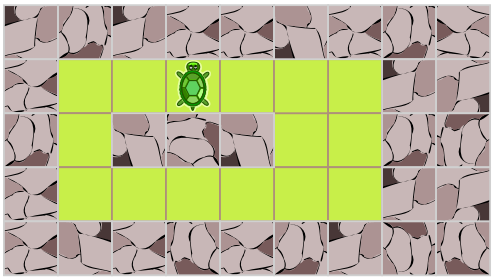
A)



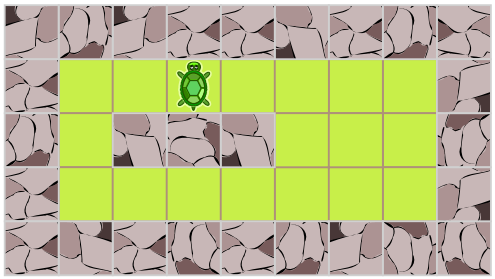
B)



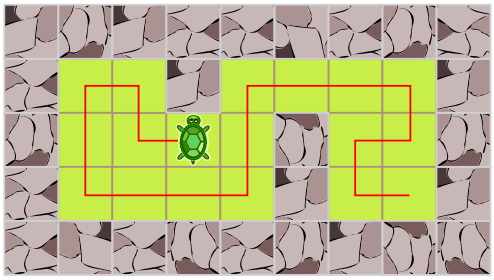
C)

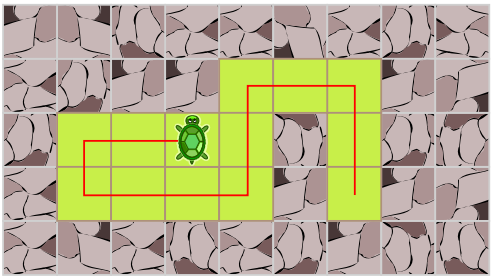


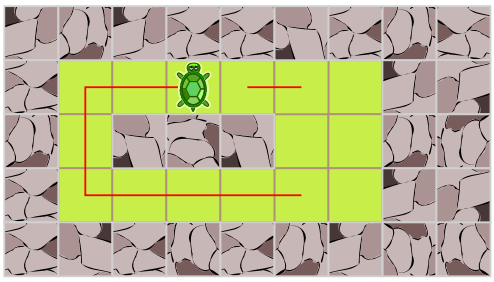
D)

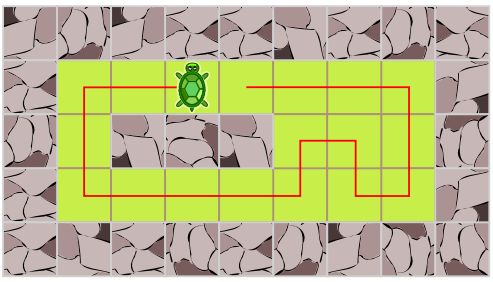


Soluzione

A) 

B) 

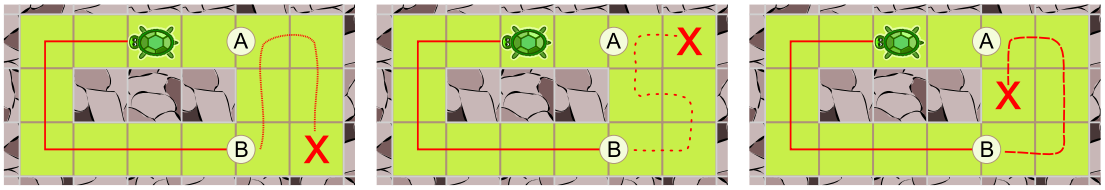
C) 

D) 

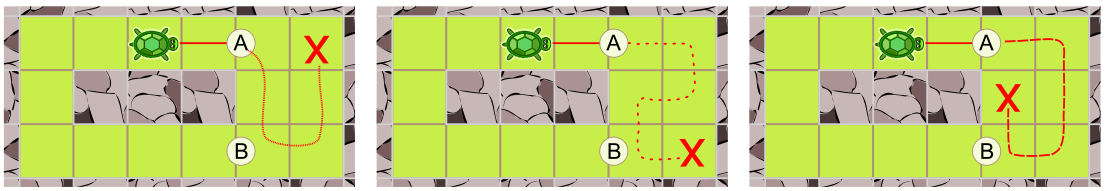
La tartaruga può visitare tutte le zone d'erba dei giardini A, B e D.

La tartaruga non può invece visitare tutte le zone d'erba del giardino C. La tartaruga ha solo 2 opzioni dal suo punto di partenza:

- Se va prima a sinistra, arriverà al punto B. Da lì dovrebbe visitare i 6 campi sulla destra in modo da raggiungere il punto A alla fine. Ma nessuno dei possibili percorsi da B finisce ad A.



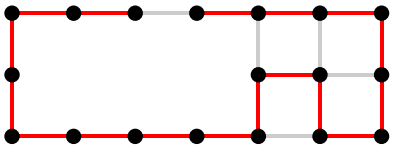
- Se la tartaruga va prima a destra, arriva ad A e dovrebbe visitare i 6 campi in modo da raggiungere il punto B alla fine. Ora si può argomentare come prima, bisogna solo scambiare la parte superiore e inferiore. Quindi non c'è nessun cammino adatto neanche in questo modo.



Questa è l'informatica!

La tartaruga deve trovare un cammino attraverso il suo giardino, visitando ogni campo erboso esattamente una volta. Il problema alla base di questo compito è il cosiddetto *problema del cammino hamiltoniano*.

Il giardino della tartaruga (cioè i quadrati di erba) può essere visto così: Ogni quadrato d'erba è un *vertice* (rappresentato come un nodo). Il giardino D si presenta quindi così:



Per tali strutture (chiamate *grafi* dagli informatici e matematici), Sir William Rowan Hamilton si chiedeva nel XIX secolo se esistesse un cammino lungo i bordi che visita ogni nodo esattamente una volta. Un tale percorso è quindi chiamato un *cammino hamiltoniano*. La questione dell'esistenza o meno di un percorso hamiltoniano è generalmente molto difficile da risolvere. Nessuno conosce un *algoritmo* che possa decidere in modo efficiente (in tempo più o meno utile) per grafi arbitrari se c'è o meno un cammino hamiltoniano nel grafo dato. Non sappiamo nemmeno se un tale algoritmo efficiente possa esistere. Questo è vero per tutti i cosiddetti problemi *NP-completi*, di cui il problema del cammino hamiltoniano è uno dei più famosi.

Parole chiave e siti web

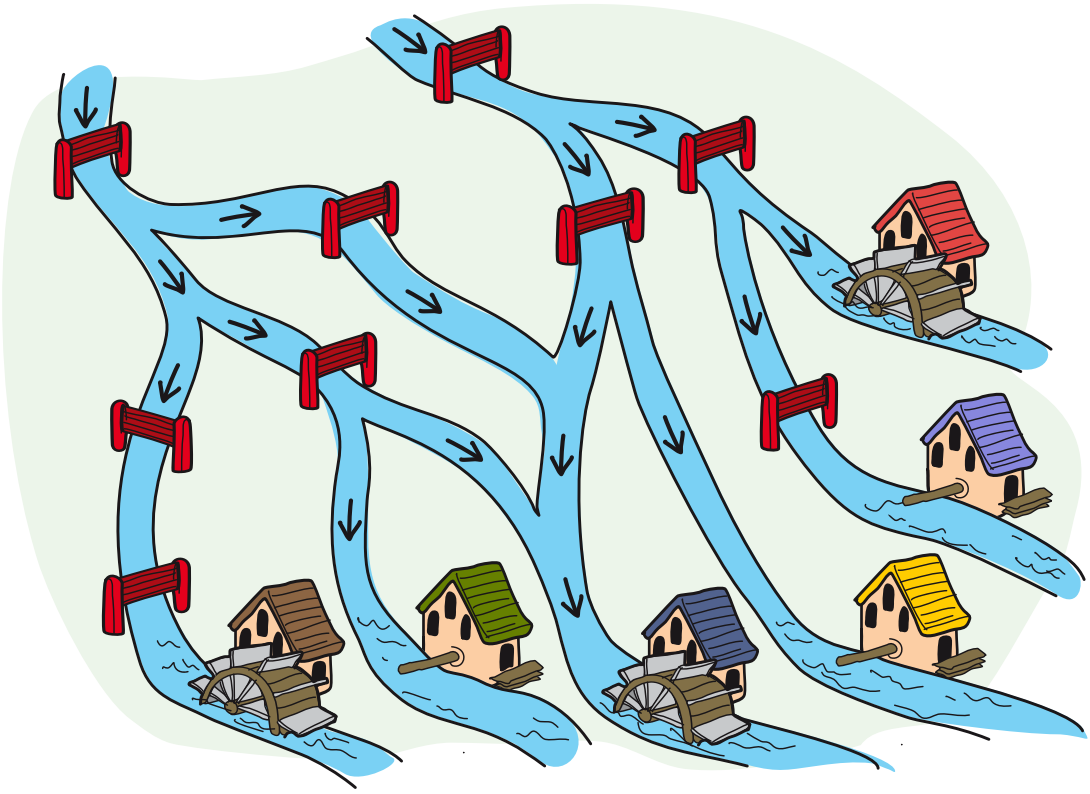
- Cammino hamitoniano: `https://it.wikipedia.org/wiki/Cammino_hamiltoniano`

2. I mulini del castoro Mert

Mert il mugnaio ha sei mulini. Deve ancora installare la ruota del mulino in tre di loro. Per fare questo, deve fermare il flusso della corrente verso questi mulini. Ma l'acqua dovrebbe continuare a scorrere verso gli altri mulini.

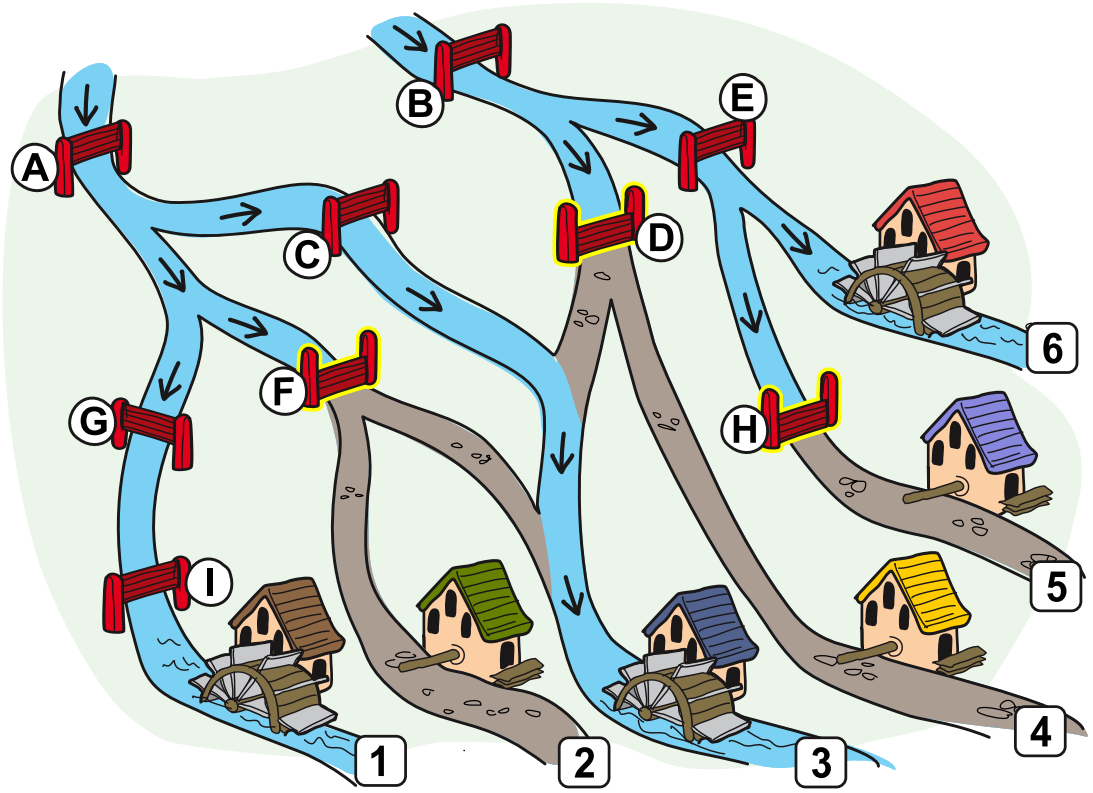
L'acqua può scorrere solo verso il basso. Una valvola a scorrimento chiusa ferma l'acqua.

Quale valvola dovrebbe chiudere Mert?



Soluzione

La risposta corretta è: Ci sono tre valvole da chiudere, che sono state etichettate D, F e H nel seguente disegno.



Questo è l'unico modo per fermare il flusso d'acqua ai mulini 2, 4 e 5 senza una ruota del mulino, mentre i mulini 1, 3 e 6 continuano a ricevere acqua:

- Le vaolvole A, G e I devono rimanere tutte aperte, altrimenti non scorrerebbe più acqua al mulino 1.
- Anche le valvole B ed E devono rimanere aperte, altrimenti non scorrerebbe più acqua al mulino 6.
- Poiché le valvole B ed E rimangono aperte, la valvola H deve essere chiusa, altrimenti l'acqua fluirebbe nel mulino 5.
- Poiché la valvola A rimane aperta, la valvola F deve essere chiusa, altrimenti l'acqua fluirebbe nel mulino 2.
- Poiché la valvola B rimane aperta, la valvola D deve essere chiusa, altrimenti l'acqua fluirebbe nel mulino 4.
- Poiché le valvole D e F sono chiuse, la valvola C deve rimanere aperta, altrimenti non ci sarebbe più acqua nel mulino 3.

Questa è l'informatica!

In questo compito, il flusso dell'acqua è controllato dalle *condizioni*. Per esempio, l'acqua scorre al mulino sul fiume 6 quando entrambe le valvole B ed E sono aperte. Ed ecco un secondo esempio un po' più complicato: l'acqua scorre al mulino 3 esattamente quando almeno una o entrambe le seguenti condizioni sono soddisfatte:

- La valvola A è aperta e una delle due valvole C o F è aperta.
- Entrambe le valvole B e D sono aperte.

Tali condizioni composte si ottengono con gli operatori logici AND (come simbolo: $\wedge$) o OR (come simbolo: $\vee$). Tali operatori collegano valori di verità come vero o falso. Così, se A e B sono due valori di verità, si può indicare quali valori di verità hanno le espressioni composte «A AND B» o «A OR B»:

A	**B**	**A AND B**	**A OR B**
falso	falso	falso	falso
vero	falso	falso	vero
falso	vero	falso	vero
vero	vero	vero	vero

Nell'informatica (e anche nella matematica), l'affermazione «A OR B» è quindi considerata corretta anche se sia A che B sono veri. L'affermazione «L'acqua scorre verso il mulino 6» è equivalente a:

> «la valvola B è aperta» AND «la valvola E è aperta».

Nel secondo esempio, l'affermazione «L'acqua scorre verso il mulino 3» è equivalente a:

> («la valvola A è aperta» AND («la valvola C è aperta» OR «la valvola F è aperta»)) OR («la valvola B è aperta» AND «la valvola D è aperta»).

Quando si programma, è importante formulare correttamente le condizioni. I collegamenti con gli operatori logici sono utili qui per formulare condizioni più complesse. Sia nella selezione (ramificazione con l'aiuto di `if`) che nell'iterazione condizionale (ciclo `while`), le condizioni sono usate per controllare il flusso del programma.

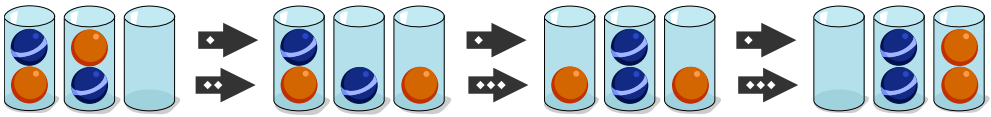
Parole chiave e siti web

- Selezione, struttura condizionale: `https://it.wikipedia.org/wiki/Selezione_(informatica)`
- Variabile booleana: `https://it.wikipedia.org/wiki/Variabile_booleana`
- Operatori booleani: `https://it.wikipedia.org/wiki/Algebra_di_Boole`

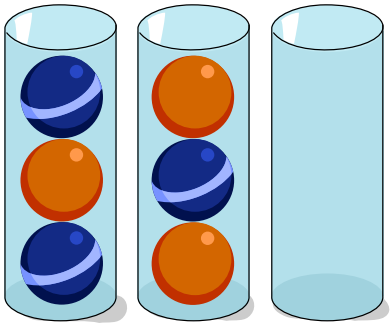
3. Gioco di palline

I castori vogliono ordinare le palline secondo il loro colore. Alla fine, tutte le palline dovrebbero essere in due bicchieri: ogni bicchiere conterrà palline dello stesso colore. Queste tre regole devono essere seguite:

- Regola 1: Solo la pallina superiore di un bicchiere può essere mossa in un passo.
- Regola 2: Una pallina può essere spostata in un bicchiere vuoto.
- Regola 3: Una pallina può essere spostata in un bicchiere se c'è ancora spazio libero e la palla sotto ha lo stesso colore.

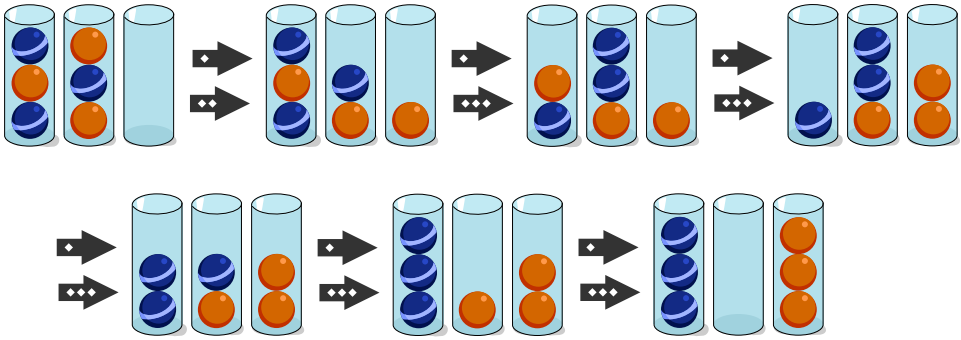


Disponi le palline spostandole secondo le tre regole.



Soluzione

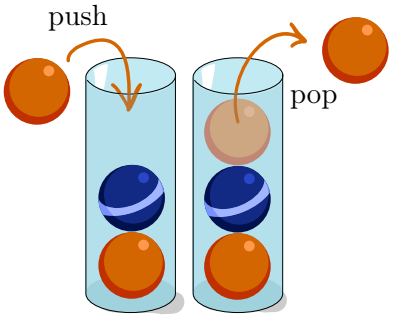
Le palle possono essere spostate nel seguente ordine:



Per disporre le palline, hai bisogno di almeno 6 passi. Ci sono anche altri modi per disporre le palline in soli 6 passi.

Questa è l'informatica!

In questo compito si spostano le palline in modo simile a come il computer gestisce i dati in una *pila*: Può solo *aggiungere un elemento in alto* (*push* in inglese) e solo *rimuovere l'elemento in alto* (*pop* in inglese). L'elemento in questo compito è una pallina.



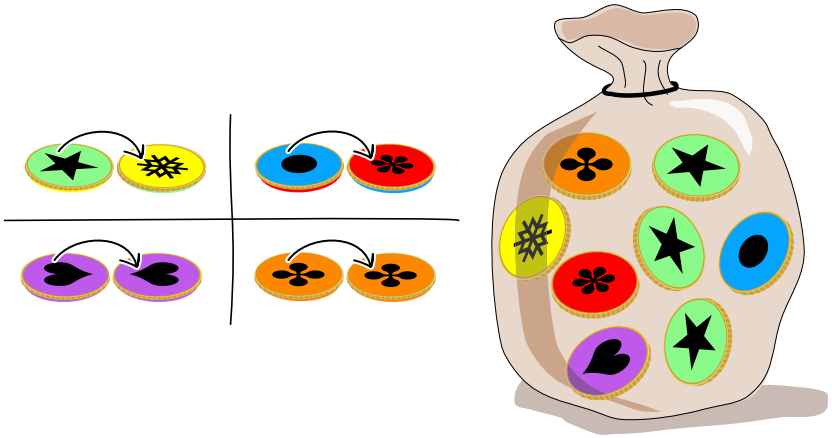
Il computer può accedere agli elementi inferiori solo se prima vengono rimosse gli elementi superiori. E l'elemento che è stato memorizzato per ultimo, il computer lo rimuoverà di nuovo per primo. Gli informatici chiamano questo il principio *Last-in-First-out* (*LIFO* in breve)

Parole chiave e siti web

- Pila: `https://it.wikipedia.org/wiki/Pila_(informatica)`

4. Sacco di monete

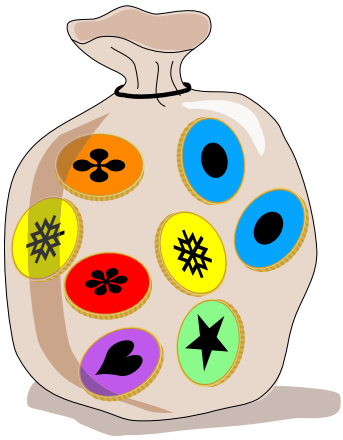
Nel paese di Emil ci sono 4 tipi diversi di monete. Qui puoi vedere i due lati di queste monete e anche il sacco di Emil con le sue monete.



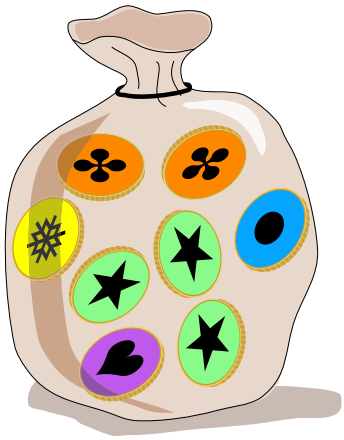
Il suo sacco viene in seguito agitato.

Qual è il sacco di Emil?

A)



B)



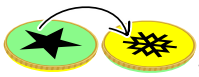
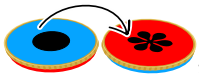
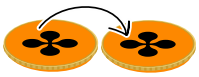
C)

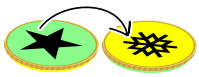
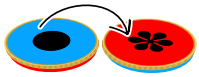
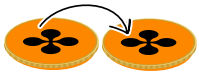
D)

Soluzione

La risposta corretta è C:

Il sacco di Emil ha:

- 4 monete  ,
- 2 monete  ,
- 1 moneta 
- e 1 moneta  .

	Sacco di Emil	Sacco A	Sacco B	**Sacco C**	Sacco D
	**4**	3	4	**4**	2
	**2**	3	1	**2**	4
	**1**	1	2	**1**	1
	**1**	1	1	**1**	1

Solo il sacco C ha lo stesso numero di monete per ogni tipo di moneta del sacco di Emil. Pertanto, è la soluzione.

Questa è l'informatica!

In questo compito devi riconoscere i tipi di monete senza vedere entrambi i lati. Hai dunque solo informazioni incomplete. Gli oggetti del mondo reale sono immagazzinati in un sistema informatico con le loro caratteristiche essenziali. Spesso è sufficiente conoscere solo una parte di queste caratteristiche per poter riconoscere un oggetto. Una telecamera in un veicolo autonomo (per esempio un'automobile)

vede sempre solo parti della realtà e il sistema informatico deve comunque essere in grado di riconoscere i veicoli e le persone sulla strada e reagire correttamente alla rispettiva situazione del traffico. Come gli esseri umani, l'intelligenza artificiale dei sistemi informatici impara gradualmente a riconoscere correttamente gli oggetti a partire dai frammenti.

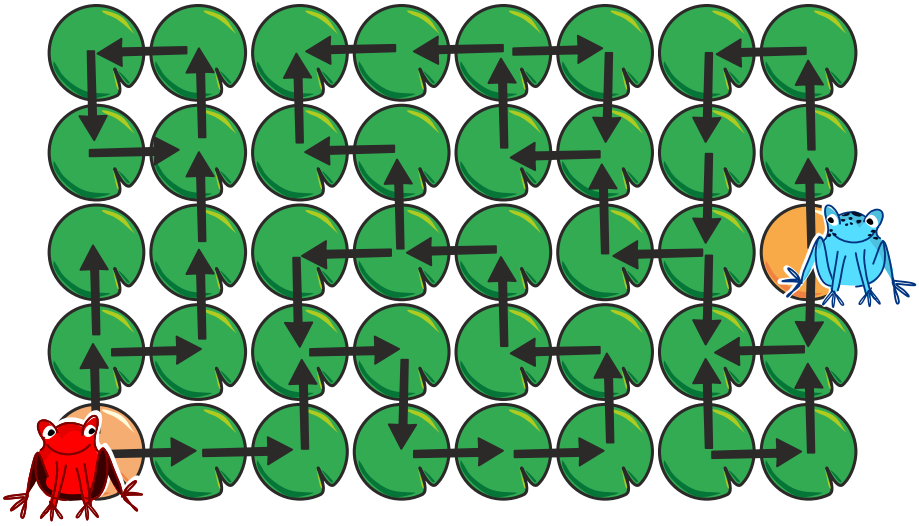
Parole chiave e siti web

- Multiinsieme: `https://it.wikipedia.org/wiki/Multiinsieme`

5. Si incontrano?

Su un lago, due rane possono saltare da una foglia di ninfea all'altra - ma solo lungo le frecce.

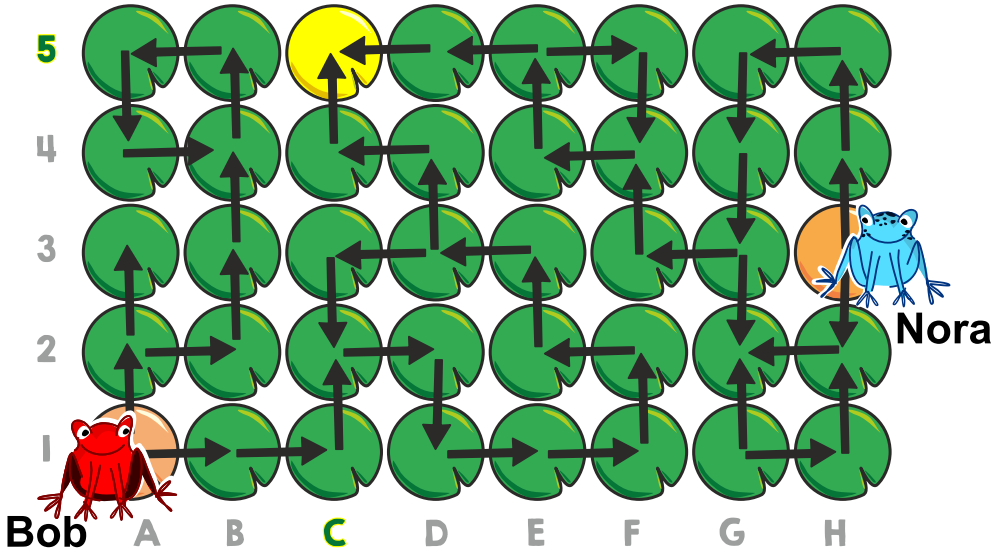
Su quale foglia di ninfea possono incontrarsi?



Man kann auf die Blätter klicken. Klickt man auf ein Blatt, wird dieses ausgewählt und gleichzeitig ein bereits ausgewähltes Blatt wieder deaktiviert.

Soluzione

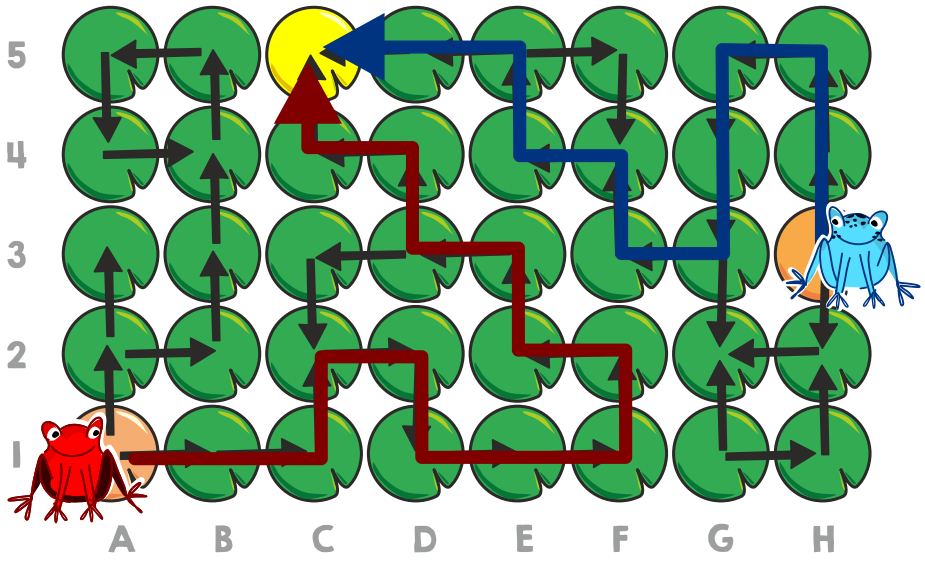
Le rane possono incontrarsi solo sulla foglia C5.



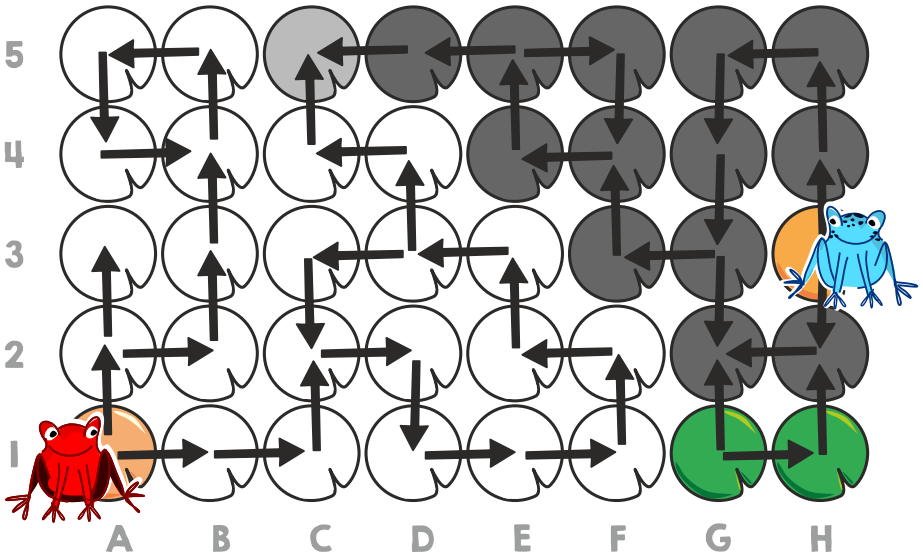
Nella sua posizione di partenza, la rana rossa Bob ha due opzioni: se va «su», o entra nel vicolo cieco A3 o rimane bloccato nel cerchio che inizia in B4. Se inizialmente va «a destra» (a B1), può prima saltare su D3. Lì può saltare «a sinistra» in un cerchio, che lo riporta a D3, o «su», che lo porta ulteriormente a C5 - un altro vicolo cieco.

Anche la rana blu Nora ha due scelte all'inizio. Se va «giù», entra nel vicolo cieco G2. Se parte «in alto», raggiunge prima il G3. Da lì può entrare di nuovo nel vicolo cieco G2 o andare a «sinistra» e raggiungere finalmente la foglia E5. Da lì va di nuovo in un cerchio che la porta alla foglia E5, o al punto morto in C5.

Sappiamo già che anche Bob può raggiungere C5, quindi possono incontrarsi lì. Il disegno mostra i modi in cui entrambi possono arrivare a C5.



Tuttavia, questo non garantisce che non si incontreranno da qualche altra parte. Il prossimo disegno mostra tutte le foglie che Bob (bianco) e Nora (grigio scuro) possono raggiungere se seguono le frecce in ogni modo possibile. Vediamo che solo C5 può essere raggiunto da entrambi.



Questa è l'informatica!

Come si può creare l'ultima immagine? Le foglie che possono essere raggiunte da una rana possono essere trovate con una *ricerca in ampiezza* o una *ricerca in profondità*. Queste sono due delle procedure standard più importanti nell'informatica. Con il loro aiuto, si può determinare le foglie grigie e bianche. Infine, bisogna trovare solo le foglie che possono essere raggiunte da entrambe le rane.

Parole chiave e siti web

- Ricerca in ampiezza: `https://it.wikipedia.org/wiki/Ricerca_in_ampiezza`
- Ricerca in profondità: `https://it.wikipedia.org/wiki/Ricerca_in_profondità`

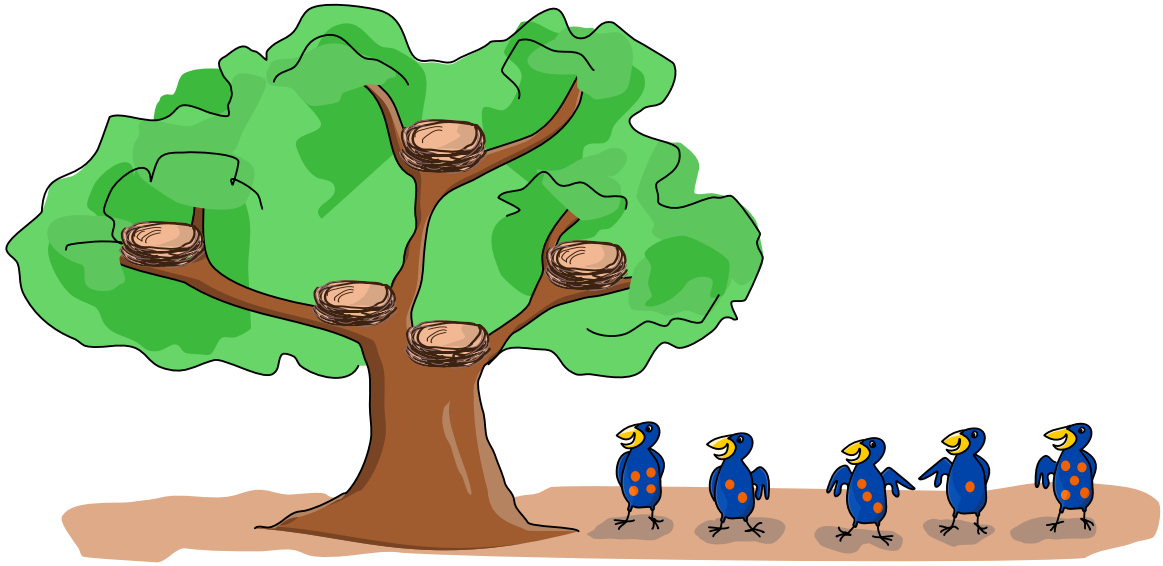
6. Nidi dei Dottuccelli

I Dottuccelli sono uccelli a pois. Ci sono cinque Dottuccelli accanto a un albero. Uno per uno - in ordine da sinistra a destra - salgono sull'albero e si appollaiano nei nidi vuoti. Quello con i quattro punti è il primo. Ogni Dottuccello procede così:

Iniziando dalla parte bassa dell'albero, esegue i seguenti passi finché trova un nido vuoto:

1. Sale fino a trovare un nido.
2. Se il nido è vuoto, si appollaia in quel nido e ci rimane.
3. Altrimenti continua a salire a seconda del numero di pois del Dottuccello già appollaiato nel nido:
   - a sinistra, se quest'ultimo ha più pois del Dottuccello che sta salendo;
   - a destra, se quest'ultimo ha meno o lo stesso numero di pois del Dottuccello che sta salendo.

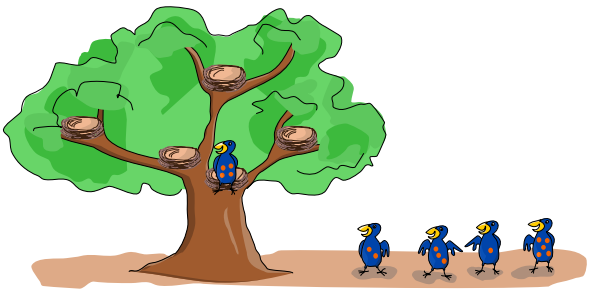
Dove sono i Dottuccelli alla fine? Metti ogni Dottuccello nel nido giusto.



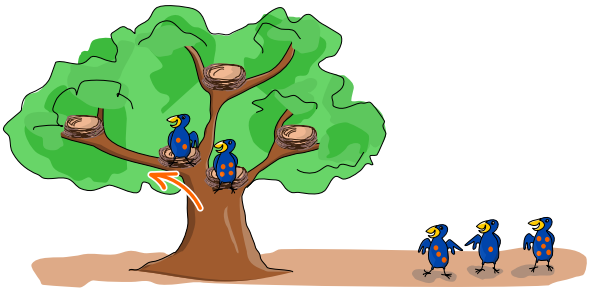
Soluzione

È così che si arriva alla soluzione giusta:

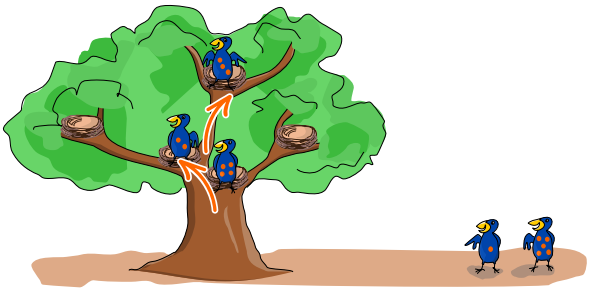
Il primo Dottuccello, quello con 4 pois, si appollaia nel nido più basso e rimane lì.



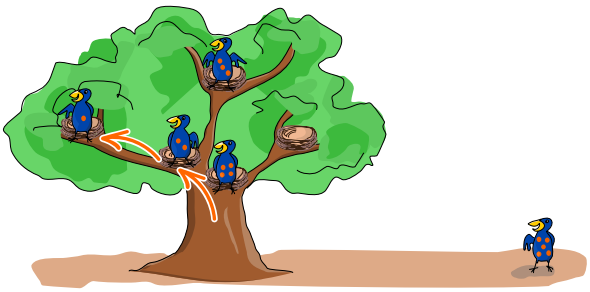
Il secondo Dottuccello ha 2 pois. Il primo Dottuccello con 4 pois si siede nel nido più basso. Poiché 4 è maggiore di 2, il secondo Dottuccello sale più a sinistra e si appollaia nel primo nido libero.



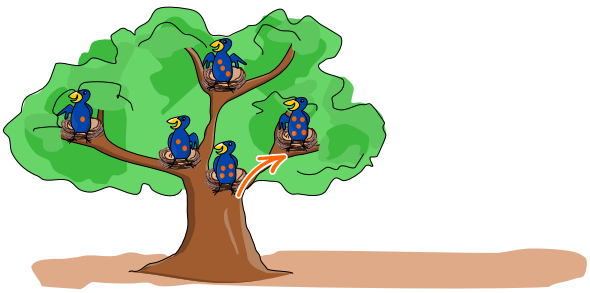
Il terzo Dottuccello ha 3 pois. Sale a sinistra al nido più basso, dove è seduto il Dottuccello con 4 pois, siccome 4 è maggiore di 3. Nel prossimo nido si trova il Dottuccello con 2 pois. Poiché 3 è maggiore di 2, il terzo Dottuccello sale a destra. Poi si appollaia nel prossimo nido libero. Questo è il nido più alto.



Il quarto Dottuccello ha 1 pois. Poiché tutti gli altri Dottuccelli hanno più pois, sale a sinistra ad ogni nido occupato. Poi arriva al nido più a sinistra e rimane lì.



L'ultimo Dottuccello ha 5 pois. Poiché nessun Dottuccello ha più pois, sale a destra ad ogni nido occupato. Lo fa una volta al nido più basso e quindi si appollaia nel nido vuoto all'estrema destra.



Questa è l'informatica!

Con i Dottuccelli appollaiati nei nidi secondo questa procedura, si ha un vantaggio interessante: un certo Dottuccello può essere trovato rapidamente. Se il Dottuccello che cerchi ha meno punti di quello che stai guardando, devi continuare a cercare nella parte sinistra dell'albero. Altrimenti, continua a guardare a destra. Così, ogni volta che controlli un Dottuccello, puoi restringere l'area di ricerca a una delle due metà. Pertanto, troverai rapidamente il Dottuccello che cerchi.

Ci sono molti modi in cui i dati possono essere organizzati; queste sono chiamate diverse *strutture dati*. La struttura dati in questo compito è un *albero binario di ricerca*. La parola *«binario»* deriva dalla parola latina «bis» per «due volte». Questo perché alla fine di un ramo (dove si trova un nido nel compito), al massimo due rami più piccoli portano avanti. Gli alberi binari di ricerca sono utilizzati nei programmi per computer quando si devono trovare rapidamente molti dati. Di solito sono molto più grandi del piccolo albero nel compito. C'è anche un'altra differenza: l'albero nel compito ha un numero fisso di cinque punti. Con un albero binario di ricerca, d'altra parte, è possibile inserire sempre più dati. Per inserire dei dati, un nuovo ramo viene semplicemente aggiunto alla fine di un ramo, allargando così l'albero. Le strutture dati che possono cambiare in questo modo sono chiamate *strutture dati dinamiche*.

Parole chiave e siti web

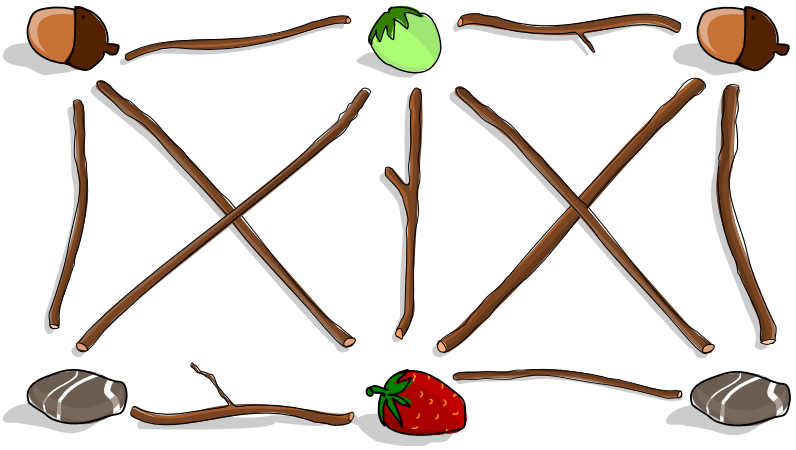
- Albero binario di ricerca: `https://it.wikipedia.org/wiki/Albero_binario_di_ricerca`

7. Ladro di fragole

Anja vuole creare un'opera d'arte in giardino, per farlo ha raccolto diverse cose: ghiande, nocciole, pietre e una fragola. Mette alcune cose sul prato.

Poi Anja mette dei rami tra queste cose. Segue la seguente regola: un ramo non può trovarsi tra due cose identiche - per esempio, non tra due ghiande.

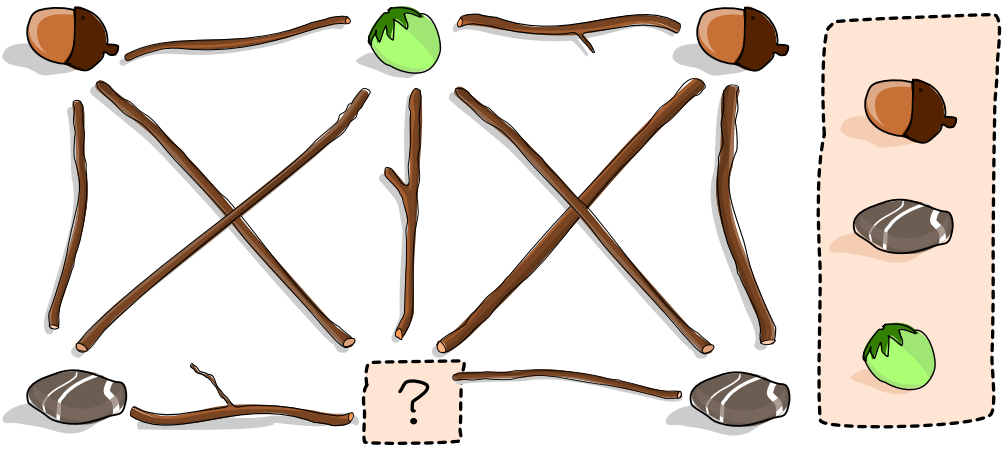
Ecco l'opera d'arte finita:



Mentre Anja è via, suo fratello arriva e mangia la fragola.

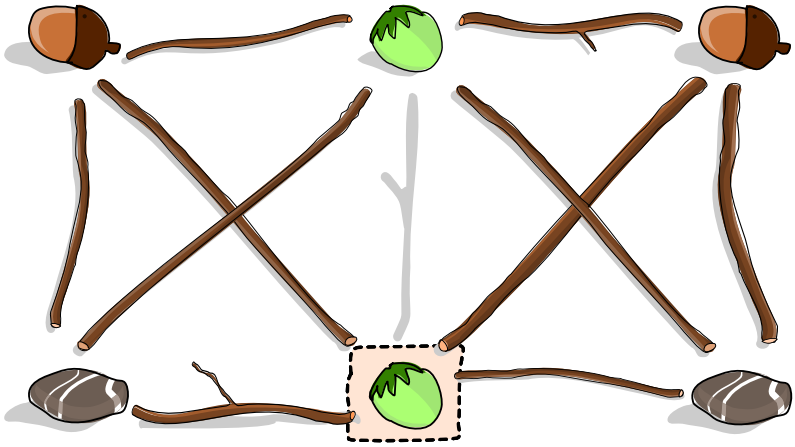
Puoi aiutarlo a coprire il misfatto?

Metti un'altra cosa al posto della fragola e rimuovi esattamente un ramo. Alla fine, la regola di Anja dovrebbe applicarsi anche all'opera d'arte modificata.



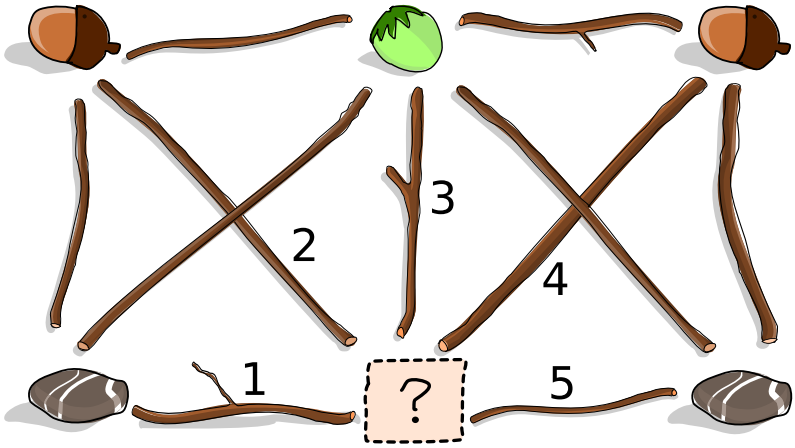
Soluzione

Se si sostituisce la fragola con una nocciola, il ramo 3 al centro vìola la regola di Anja: si trova tra due cose identiche, cioè due nocciole. Pertanto, questo ramo deve essere rimosso.



Per le altre due possibili sostituzioni, è necessario rimuovere più di un ramo:

- Se la fragola è sostituita da una ghianda, è necessario rimuovere i rami 2 e 4.
- Se la fragola è sostituita da una pietra, è necessario rimuovere i rami 1 e 5.



Questa è l'informatica!

L'opera d'arte di Anja può essere rappresentata come un *grafo*. Un grafo è composto da *vertici* (i luoghi delle cose) e da *archi* (i rami), ognuno dei quali collega due vertici. I grafi sono molto versatili e sono utilizzati per la modellazione in molti compiti di informatica. Quando due vertici sono collegati direttamente da un arco, sono *vicini* l'uno all'altro. Un gruppo di vertici in cui ogni vertice è un vicino a ogni altro vertice è chiamato *cricca*. Nel nostro grafo, abbiamo due cricche con quattro vertici: la metà destra e la metà sinistra del grafo (la nocciola sopra e il punto interrogativo appartengono a entrambe le cricche). Segue dalla regola di Anja che tutti i nodi di una cricca debbano essere occupati da cose diverse. Per mantenere la regola, abbiamo bisogno di almeno tante cose diverse quanti sono i vertici di una cricca. Dopo aver rimosso la fragola, abbiamo solo 3 cose diverse.

Così ora possono rimanere cricche con un massimo di 3 vertici per continuare a soddisfare la regola. Quindi un arco (un ramo) deve essere rimosso in modo che entrambe le cricche con quattro vertici siano rotte.

La regola di Anja corrisponde a una regola del cosiddetto *problema della colorazione dei grafi*: Assegniamo un colore ad ogni vertice di un grafo, dove i vicini devono avere colori diversi. (I colori corrispondono a i diversi tipi di cose.) L'obiettivo è di solito quello di usare il minor numero possibile di colori. Il problema di come colorare un grafo con il numero minimo di colori ha molte applicazioni. Alcuni esempi sono la pianificazione di competizioni sportive, la progettazione di un piano di posti a sedere e persino la risoluzione di un Sudoku.

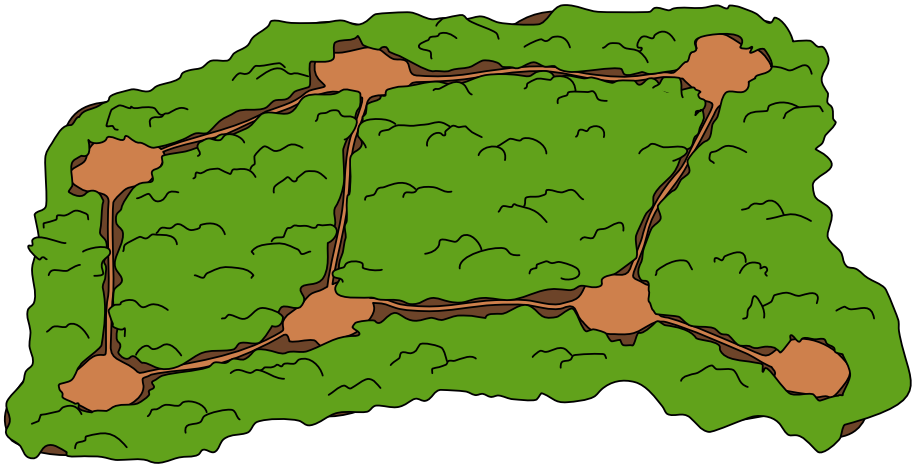
Parole chiave e siti web

- Colorazione dei grafi: `https://it.wikipedia.org/wiki/Colorazione_dei_grafi`
- Cricca: `https://it.wikipedia.org/wiki/Cricca_(teoria_dei_grafi)`

8. Osservazione del bosco

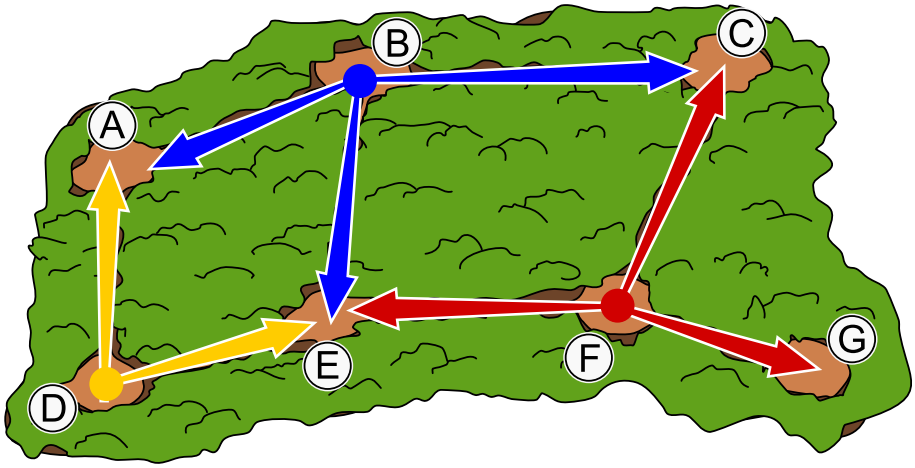
I forestali vogliono osservare gli animali sui sentieri del bosco. Da ogni radura possono osservare tutti i sentieri collegati con la radura successiva. Tutti i sentieri devono essere osservati dal minor numero possibile di forestali.

Scegli il minor numero possibile di radure da cui i forestali possano osservare tutti i sentieri!



Soluzione

L'immagine mostra la soluzione minima in cui i forestali possono stare solo su tre radure e osservare tutti i sentieri.



Ci sono otto sentieri che devono essere osservati. Se solo due forestali potessero osservare tutti i sentieri, dovrebbe esserci una radura da cui partono almeno quattro sentieri. Ma non c'è una tale radura in questo bosco. Pertanto, due forestali non sono sufficienti.

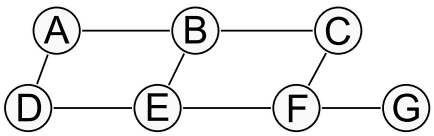
Quindi, sono necessari almeno tre forestali per osservare tutti i sentieri. Di conseguenza, la soluzione data qui è una soluzione con il minor numero possibile di forestali. In effetti, non esiste un'altra soluzione con esattamente tre forestali.

Dal numero di sentieri da osservare e dal fatto che non ci sono radure con più di tre sentieri collegati, possiamo concludere che ogni forestale deve osservare almeno due sentieri che gli altri forestali non osservano.

Per osservare il vicolo cieco tra la radura F e G, un forestale deve essere posizionato sulla radura F. Per osservare il sentiero tra la radura B e C, il secondo forestale deve osservare dalla radura B. Per osservare gli ultimi due sentieri con un solo forestale, quest'ultimo deve essere posizionato sulla radura D. In questo modo, la soluzione data è definitiva e non può esserci altro.

Questa è l'informatica!

Le relazioni tra le cose (per esempio i sentieri tra le radure) possono essere rappresentate come un cosiddetto *grafo*. Un grafo è composto da *vertici* (qui: le radure), rappresentati come cerchi, e *archi* (qui: i sentieri), rappresentati come linee tra i vertici. Il grafo di questo compito si presenta così:



In questo compito, devi trovare il minor numero di vertici nel grafo in modo che ogni arco inizi o finisca in almeno uno di questi vertici. Gli informatici chiamano un tale sottoinsieme di nodi una

copertura minima dei vertici. Troviamo questi problemi di copertura dei vertici nella vita di tutti i giorni, per esempio quando si cercano le migliori posizioni per i lampioni o per il posizionamento intelligente delle telecamere di sorveglianza.

Parole chiave e siti web

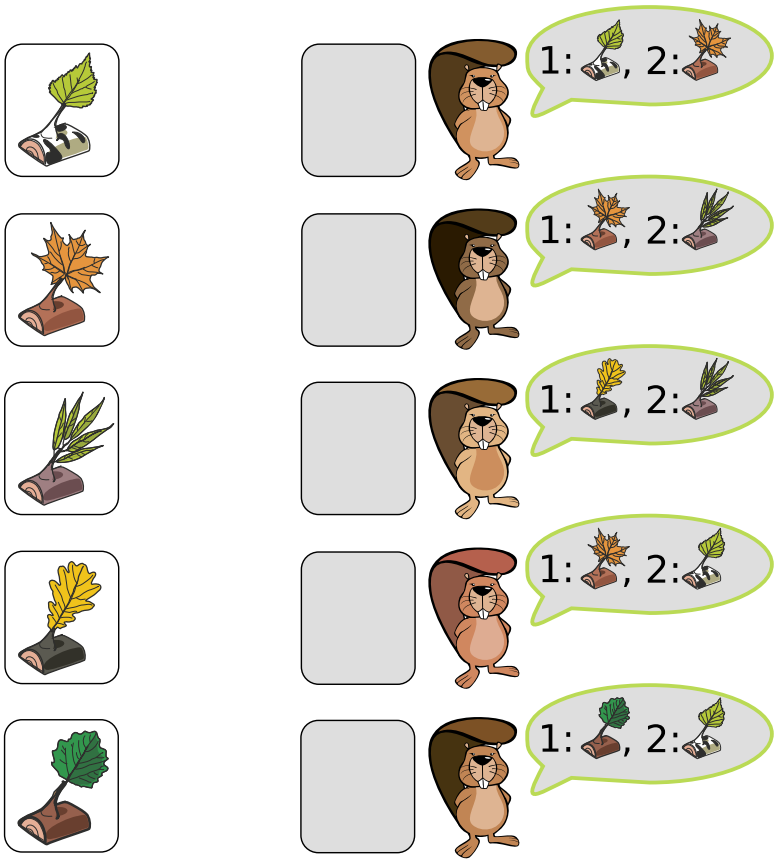
- Grafo: `https://it.wikipedia.org/wiki/Grafo`
- Copertura dei vertici: `https://it.wikipedia.org/wiki/Copertura_dei_vertici`

9. Regalo preferito

La famiglia castoro ha cinque regali per i suoi cinque figli. Ogni castorino nomina prima il suo regalo preferito e poi il secondo preferito. I regali devono essere assegnati correttamente:

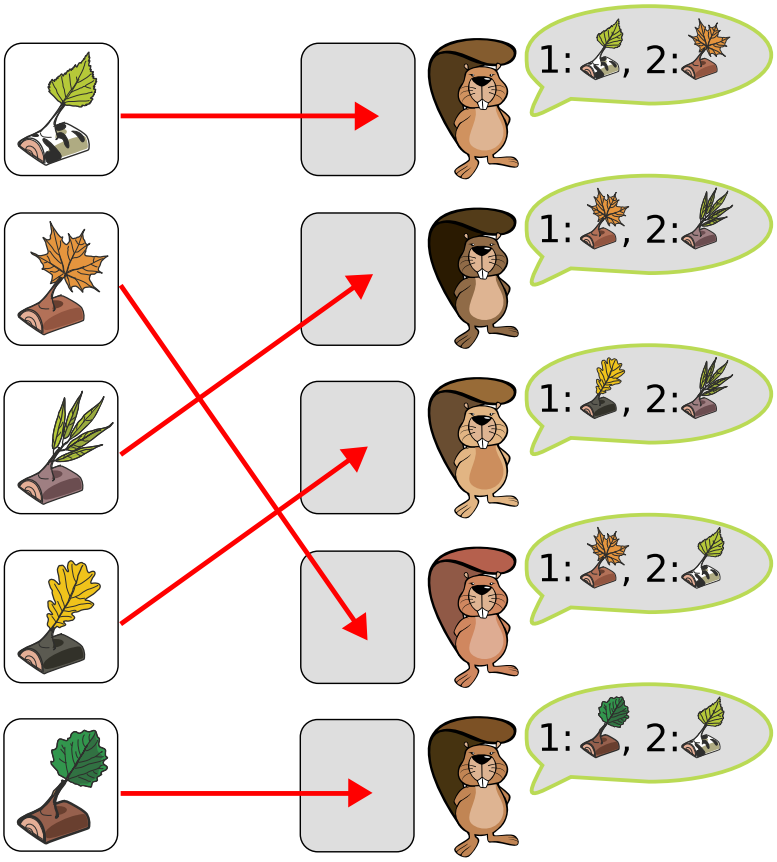
1. Il maggior numero possibile di castorini dovrebbe ricevere il loro regalo preferito.
2. Gli altri dovrebbero ricevere il secondo regalo preferito.

Dai ai castorini i regali giusti.



Soluzione

Questo è l'unico assegnamento di regali che soddisfa entrambe le condizioni.



Il grafico qui sopra assegna a quattro castorini il loro regalo preferito e a un castorino il suo secondo regalo preferito. Non tutti i castorini possono ottenere il loro regalo preferito perché due castorini hanno lo stesso regalo preferito. Pertanto, non è possibile nessun assegnamento dove più castorini ottengono il loro regalo preferito. Nota: se fai l'assegnazione dall'alto verso il basso e assegni il secondo regalo al secondo castoro, allora il quarto castoro non riceverà nessuno dei suoi regali preferiti. Quindi in questo compito non è sufficiente fare la migliore selezione per ogni singolo castorino.

Una strategia di risoluzione è quella di assegnare prima tutti i regali che sono il regalo preferito di un solo castorino. In seguito, solo due castorini rimangono con lo stesso regalo preferito. Si valuta in seguito a quale castorino può essere assegnato il secondo regalo preferito. Si assegna dunque all'ultimo castorino il suo regalo preferito.

Questa è l'informatica!

Questo compito è un chiaro *problema di assegnazione*: vogliamo assegnare i regali in modo che tutti i castorini ricevano un regalo e non ci sia nessun castorino senza regalo. Così facendo, i castorini non hanno un solo desiderio, ma danno una sequenza di preferenze. Tali problemi di assegnazione con ordini di preferenze possono diventare molto complicati. L'informatica ci aiuta a risolvere questi problemi il più rapidamente possibile.

Una possibilità è quella di dare un valore alle assegnazioni: il regalo preferito ha valore 1 e il secondo regalo preferito ha valore 2. Vogliamo minimizzare il valore totale. Un *accopiamento* (in inglese *matching*) è *ottimale* se non c'è un altro accopiamento con più prime selezioni soddisfatte. In informatica, tale assegnazione è chiamata *rank-maximal-matching*. Ci sono molti problemi di corrispondenza. Uno di essi è chiamato il *problema del matrimonio stabile*. Sembra interessante? Allora dovresti studiare informatica!

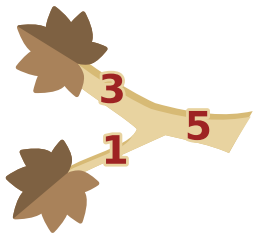
Parole chiave e siti web

- Problema di assegnazione: `https://it.wikipedia.org/wiki/Problema_di_assegnazione`
- Accoppiamento: `https://it.wikipedia.org/wiki/Accoppiamento_(teoria_dei_grafi)`

10. Salvataggio dell'albero

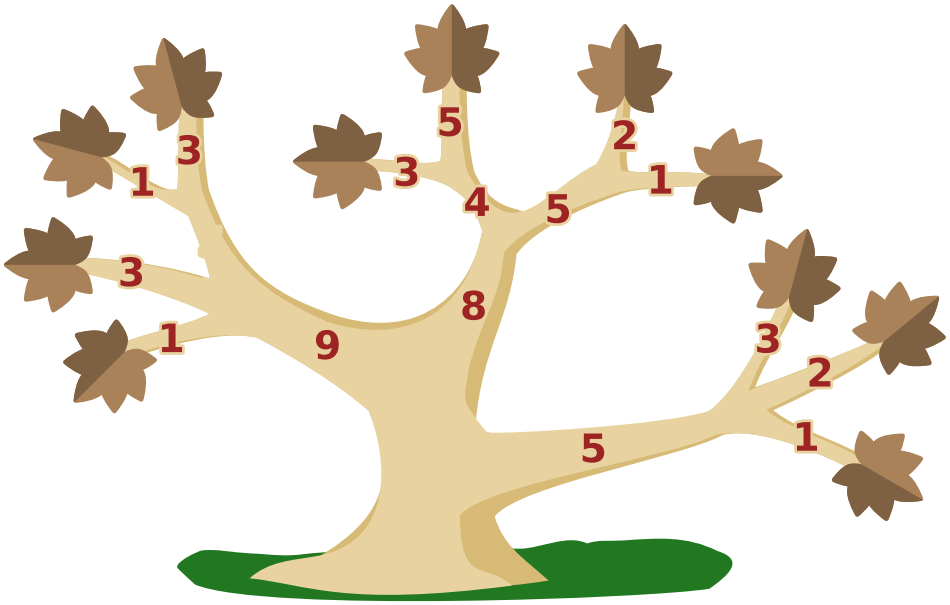
Un albero nel giardino di Bruno è malato e tutte le foglie si sono seccate. Bruno vuole salvare l'albero. Per farlo, deve segare alcuni rami in modo che alla fine tutte le foglie siano rimosse e che possano crescere nuovi rami con nuove foglie.

Bruno vuole finire il prima possibile. L'immagine mostra un esempio:



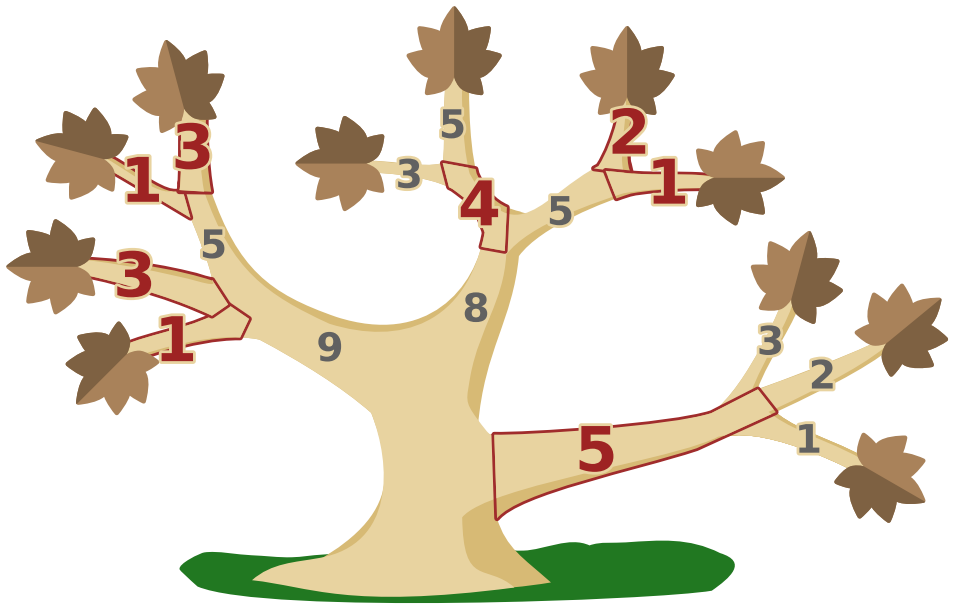
Per tagliare le due foglie, Bruno può segare i due rami con le foglie o solo il ramo da cui si dipartono gli altri due. I numeri indicano per ogni ramo quanto tempo ci vuole per il taglio. Bruno segherà quindi i due rami con le foglie, perché $3 + 1 < 5$. Qui sotto si vede l'intero albero.

Quali rami segherà Bruno per finire il più velocemente possibile?



Soluzione

Questa è la soluzione giusta: Bruno sega i rami segnati in rosso per finire il più velocemente possibile:



Ma perché? Prima di tutto, possiamo calcolare di quanto tempo ha bisogno Bruno se sega solo i rami con le foglie:

$$1 + 3 + 1 + 3 + 3 + 5 + 2 + 1 + 3 + 2 + 1 = 25$$

Ora continuiamo verso il tronco e consideriamo se potrebbe essere più veloce segare il ramo da cui dipartono direttamente o indirettamente i rami precedenti. Dopo il primo di questi passi, il seguente calcolo risulta (la funzione «min» calcola il minimo dei suoi argomenti):

$$\begin{aligned} & 1 + 3 + \min(5, 1 + 3) + \min(4, 3 + 5) + \min(5, 2 + 1) + \min(5, 3 + 2 + 1) \\ = \; & 1 + 3 + (1 + 3) + 4 + (2 + 1) + 5 \\ = \; & 20 \end{aligned}$$

All'inizio non calcoliamo il tempo totale, in modo da vedere meglio quali rami devono essere tagliati. Dopo il prossimo passo abbiamo già raggiunto il tronco:

$$\begin{aligned} & \min(9, 1 + 3 + 1 + 3) + \min(8, 4 + 2 + 1) + 5 \\ = \; & (1 + 3 + 1 + 3) + (4 + 2 + 1) + 5 \\ = \; & 20 \end{aligned}$$

Bruno non può essere più veloce di così.

Questa è l'informatica!

Immaginiamo che i pezzi segati dell'albero di Bruno non cadano direttamente a terra - come succede quando si risolve questo compito sullo schermo. Allora potremmo dire che l'albero si divide in due sole parti segandolo: Una parte contiene tutti i pezzi segati, cioè in particolare tutte le foglie, e l'altra parte contiene il tronco e tutti i rami che si estendono da esso ai punti di taglio. Questa divisione o *taglio* attraverso l'albero è minima in termini di tempo che Bruno deve spendere per segare.

Anche l'informatica conosce gli alberi e li usa per rappresentare oggetti che sono collegati tra loro in un certo modo. Gli oggetti sono chiamati *vertici*, le connessioni *archi*. C'è sempre un solo percorso tra due vertici lungo gli archi - proprio come in un albero reale c'è sempre un solo percorso lungo i rami da una foglia o una biforcazione al tronco. Se rinunciamo a questa condizione, si parla più generalmente di un *grafo*.

In un grafo generale, un *taglio minimo*, cioè la scomposizione in due o più parti con costi minimi, non è così facile da calcolare come abbiamo dimostrato qui per un albero, ma non è nemmeno troppo difficile. Questo è un bene, perché ci sono applicazioni interessanti. I tagli minimi possono essere utilizzati, per esempio, nella scomposizione di file di immagini in parti simili. In grafi speciali, le *reti di flusso*, che possono essere utilizzate per modellare i flussi di dati attraverso le reti, tra le altre cose, il costo di un taglio minimo corrisponde al massimo flusso possibile attraverso l'intera rete.

Parole chiave e siti web

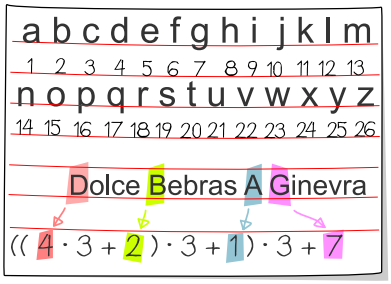
- Albero: `https://it.wikipedia.org/wiki/Albero_(grafo)`
- Taglio: `https://it.wikipedia.org/wiki/Taglio_(teoria_dei_grafi)`
- Rete di flusso: `https://it.wikipedia.org/wiki/Rete_di_flusso`
- Teorema del flusso massimo e taglio minimo:
  `https://it.wikipedia.org/wiki/Teorema_del_flusso_massimo_e_taglio_minimo`

11. Biblioteca

Susi è nella biblioteca dei castori con Tim e vuole prendere in prestito un libro: «Dolce Bebras A Ginevra»

Tim va allo scaffale 1, raggiunge la fila 3, scomparto 6 ed estrae il libro. Susi è impressionata dalla sua velocità! Tim spiega a Susi come determinare la posizione di un libro:

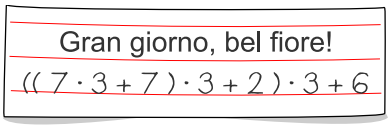
Prendi la prima lettera di ogni parola nel titolo e determina la sua posizione nell'alfabeto. Somma questo valore al valore della lettera precedente, ma prima di ogni somma, il valore totale raggiunto finora viene moltiplicato per 3. Il risultato per il libro desiderato è 136, è quindi chiaro dove si trova il libro.



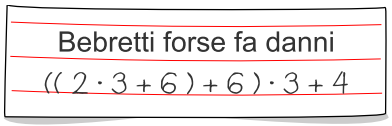
In seguito, Susi prepara i codici corrispondenti per recuperare i suoi libri preferiti. In un caso, tuttavia, ha commesso un errore.

Quale di questi codici è stato calcolato in modo errato?

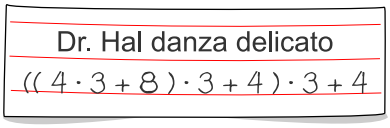
A)



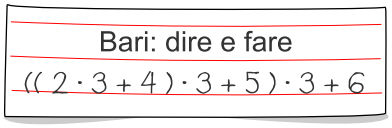
B)



C)

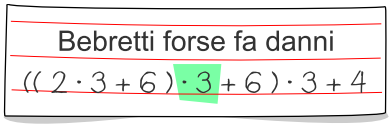


D)



Soluzione

Susi ha fatto quasi tutto giusto: ha sempre aggiunto i valori delle posizioni corretti e ha sempre moltiplicato i risultati intermedi per 3 - con un'eccezione: nella risposta B ha dimenticato la moltiplicazione una volta!



Questa è l'informatica!

Con le «espressioni di localizzazione», la biblioteca consente ai suoi visitatori di determinare l'esatta ubicazione dei libri, cosicché nessuno debba cercare troppo a lugno. Tuttavia, c'è una cosa che la biblioteca e i visitatori devono tenere a mente: le espressioni e quindi i loro risultati possono essere gli stessi per libri diversi. Ad esempio, «Abbattimento di alberi» e «Andiamo da Alberto» si trovano nello stesso scomparto. Gli scomparti quindi non devono essere troppo piccoli, oppure devono essere sufficientemente flessibili per essere adattati.

Anche con i dati archiviati nella memoria del computer si usa spesso la stessa metodologia: è infatti una buona idea se la loro posizione in memoria può essere calcolata direttamente dai dati stessi. A questo scopo sono state sviluppate in informatica le funzioni hash: funzioni matematiche che calcolano un valore dal contenuto dei dati o da parte dei dati e che indica direttamente la posizione di archiviazione - come con i titoli dei libri in questo esercizio. Buone funzioni di hash assicurano che lo stesso valore risulti nel minor numero possibile di casi. Se si verifica tali collisioni, l'informatica conosce buoni metodi per affrontarla.

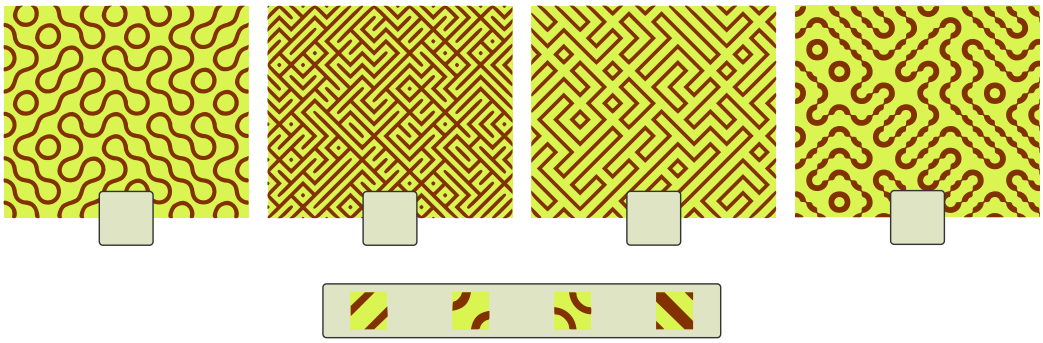
Parole chiave e siti web

- Funzione di hash: `https://it.wikipedia.org/wiki/Funzione_di_hash`
- Hash table: `https://it.wikipedia.org/wiki/Hash_table`

12. Piastrelle Truchet

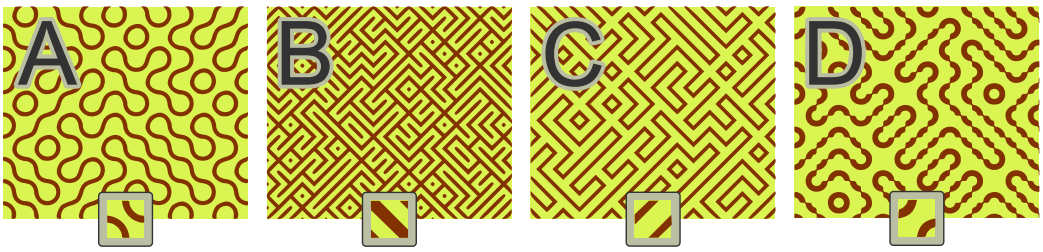
I seguenti modelli sono stati creati ciascuno da una singola piastrella. Le singole piastrelle sono mostrate ingrandite.

Abbina le piastrelle ai loro possibili modelli.

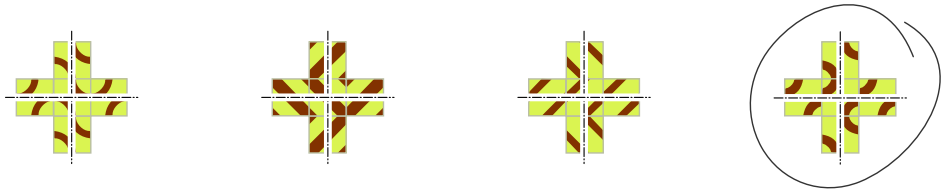


Soluzione

Questa è la classificazione corretta:



Se metti 5 piastrelle una accanto all'altra e le compari più da vicino, puoi vedere delle chiare differenze:



La piastrella  è l'unica delle piastrelle con quattro lati che non si adattano esattamente l'uno all'altro. Questo è l'unico modo per creare linee di larghezza variabile come nel modello D. La piastrella  è l'unica che può creare punti quadrati nel modello B, cioè con quattro triangoli che si incontrano. Inoltre, ha la superficie del marrone più ampia rispetto al giallo, proprio come B; questo dimostra anche che fa parte del modello B. Rimane solo il modello A come possibile risultato per le forme rotonde della piastrella  e solo il modello C per le forme diritte della piastrella  .

Questa è l'informatica!

Queste piastrelle prendono il nome da Sébastien Truchet (* 1657; † 1729), che ha lavorato su diverse varianti di queste piastrelle. Le piastrelle con 4 lati uguali formano un sottogruppo di piastrelle Truchet (ma le piastrelle Truchet non devono necessariamente avere 4 lati uguali, come in 3 degli schemi precedenti). Il fatto che schemi complessi possano essere creati con blocchi di costruzione molto semplici è una proprietà interessante che incontriamo ancora e ancora nell'informatica. Le piastrelle Truchet sono studiate in matematica e in informatica e utilizzate nei giochi di computer per creare labirinti o decorazioni.

Parole chiave e siti web

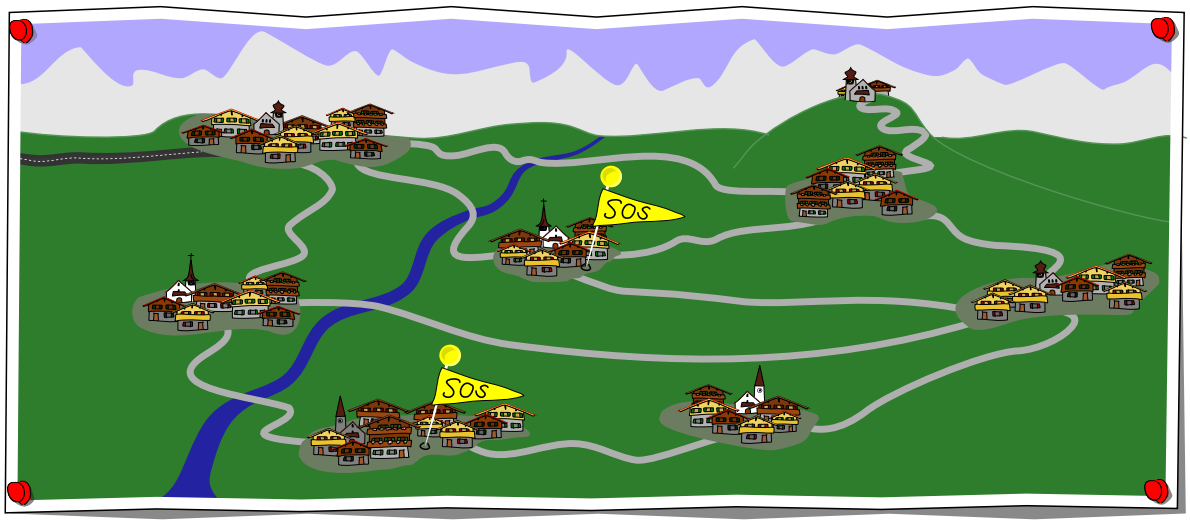
- Piastrelle Truchet: `https://it.wikinew.wiki/wiki/Truchet_tiles`

13. Villaggi isolati

Alcuni villaggi di montagna sono collegati alla grande città attraverso la seguente rete stradale.

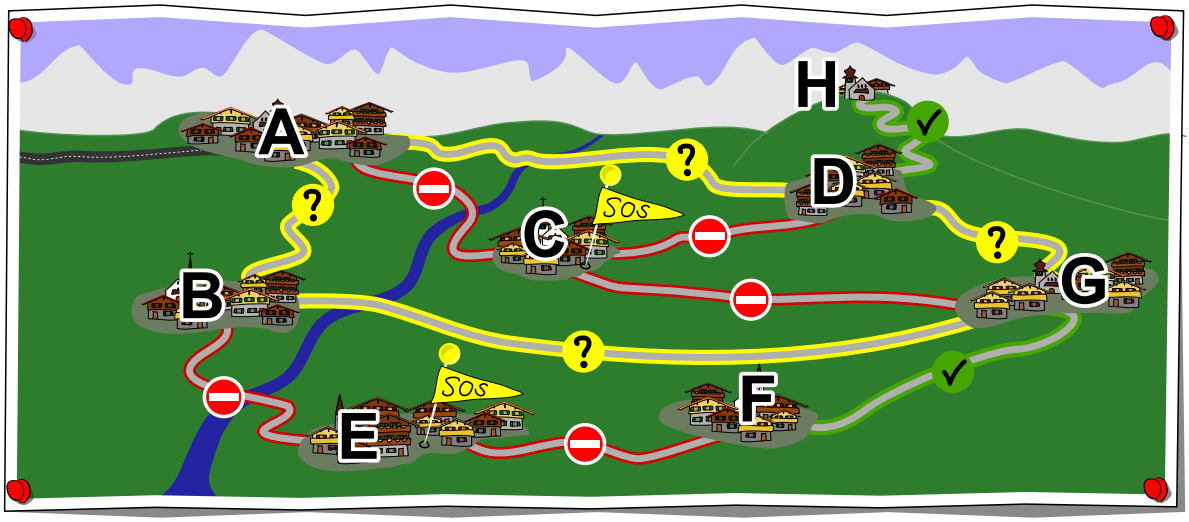
Dopo una tempesta, diversi villaggi segnalano che non sono più accessibili, in particolare quelli con i segnali di SOS. Possiamo concludere che alcune strade sono bloccate.

Per ogni strada tra villaggi in questa rete stradale, segnala se è (1) bloccata ⛔, (2) aperta ✅, o (3) non possiamo essere sicuri senza ulteriori informazioni che la strada sia aperta o bloccata ❓.



Soluzione

La mappa mostra quello che sappiamo sulle connessioni della rete stradale:



Iniziamo a riconoscere le strade bloccate. Le due strade che portano al villaggio E possono essere bloccate, perché altrimenti il villaggio E sarebbe ancora raggiungibile. Allo stesso modo, le tre strade che portano al villaggio C sono bloccate, perché altrimenti il villaggio C sarebbe ancora raggiungibile.

Poi, cerchiamo le strade che devono essere aperte. La strada tra il villaggio G e F deve essere aperta, altrimenti, a causa della strada bloccata tra il villaggio F ed E, il villaggio F non sarebbe raggiungibile. Anche la strada tra la chiesa H e il villaggio D deve essere aperta, poiché H è accessibile e può essere raggiunta solo attraverso D.

Ora rimangono le strade che potrebbero essere aperte. Poiché i villaggi B, G e D sono collegati più volte al villaggio A, non possiamo dire quali delle strade rimanenti sono aperte. Per esempio, il villaggio B potrebbe essere raggiunto attraverso il villaggio A, ma anche attraverso il villaggio G. Lo stesso vale per il villaggio D. Il villaggio G può essere servito attraverso il villaggio B o D. Quindi una qualsiasi delle strade del circuito A - B - G - D - A potrebbe essere bloccata e questi 4 villaggi potrebbero ancora rimanere tutti accessibili.

Questa è l'informatica!

Proprio come nelle reti stradali, le connessioni nelle reti di computer possono essere problematiche, sovraccaricate o completamente difettose. Per prevenire i guasti, sono spesso previste misure di sicurezza, come connessioni multiple ad un luogo. Questo si chiama *ridondanza*.

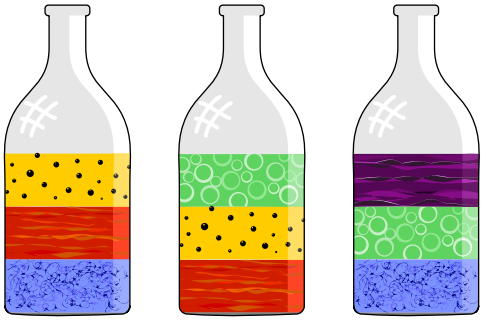
Sistemare i difetti in un sistema è un compito che gli informatici devono fare molto spesso, non solo nelle reti di computer ma anche nello sviluppo di software. Per correggere un errore è necessario identificare la sua fonte esatta, e questo processo è di solito fatto passo dopo passo in diverse fasi. Alcuni programmatori credono che non sia possibile trovare tutti gli errori e bug di un programma.

Parole chiave e siti web

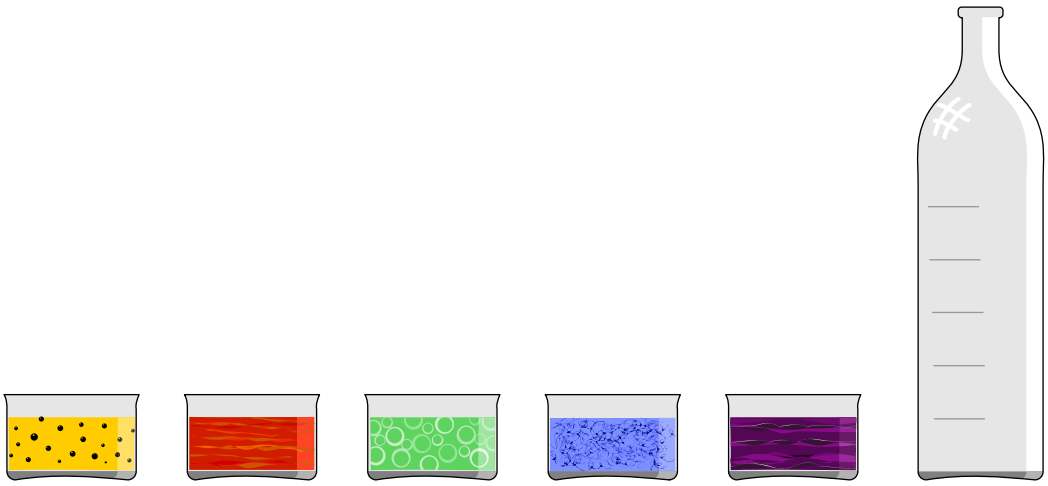
- Ridondanza: `https://it.wikipedia.org/wiki/Ridondanza_(ingegneria)`
- Debugging: `https://it.wikipedia.org/wiki/Debugging`

14. Disposizione dei liquidi

Mark ha delle bottiglie contenenti tre liquidi colorati ciascuna, stratificati l'uno sull'altro. Sa che i liquidi con densità minore si muovono sempre sopra i liquidi con densità maggiore. Ora vuole vedere cosa succede quando si mettono tutti i liquidi colorati in una bottiglia.

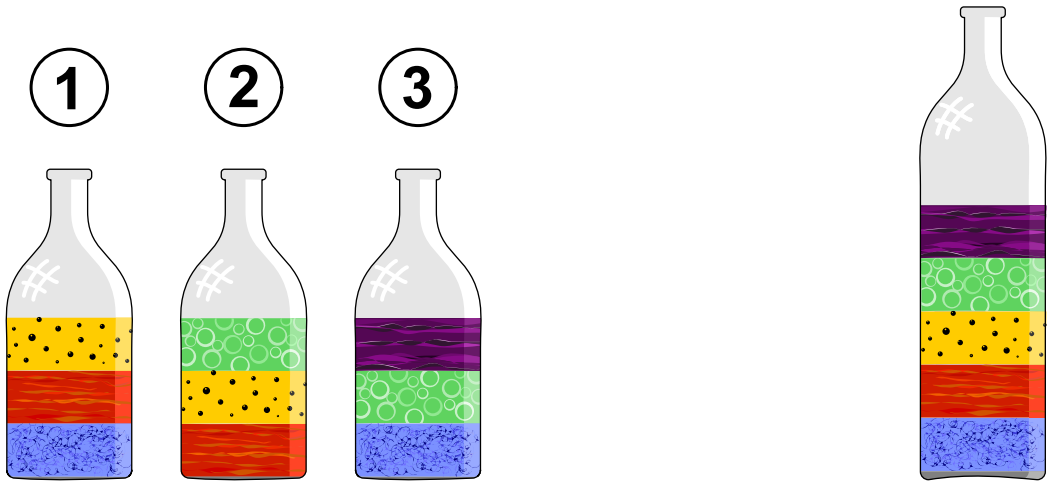


Disponi i cinque liquidi colorati nella bottiglia come apparirebbero dopo averli mischiati tutti!



Soluzione

L'immagine mostra la corretta disposizione dei cinque liquidi colorati nella bottiglia grande.



L'ordine si trova seguendo la seguente procedura: passo dopo passo, rimuovi dalle tre bottiglie date i liquidi che non sono sopra un altro liquido e mettili nella bottiglia grande.

All'inizio, solo le bottiglie 1 e 3 hanno un liquido blu e si trova sul fondo, quindi non si trova da nessuna parte su un altro strato di liquido. Il liquido rosso è sul fondo della bottiglia 2. Ma nella bottiglia 1 è sopra il liquido blu e deve quindi avere una densità inferiore al liquido blu. Quindi la prima cosa da fare è togliere il liquido blu dalle bottiglie e metterlo nella bottiglia grande.

Ora il liquido rosso è l'unico che non si trova sopra un altro liquido. Viene tolto dalle bottiglie 1 e 2 e messo nella bottiglia grande. Poi viene il liquido giallo, poi il liquido verde e infine il liquido viola, che ha la densità più bassa e sopra il quale non si trova nessun altro liquido.

Questa è l'informatica!

Nel risolvere questo compito, hai valutato la disposizione dei liquidi nelle tre bottiglie del compito e hai ordinato i liquidi in base alla loro densità.

Una sostanza ha molte proprietà misurabili: temperatura di ebollizione, temperatura di fusione, conducibilità elettrica e densità. In questo caso, la densità è stata utilizzata come criterio di selezione delle sostanze.

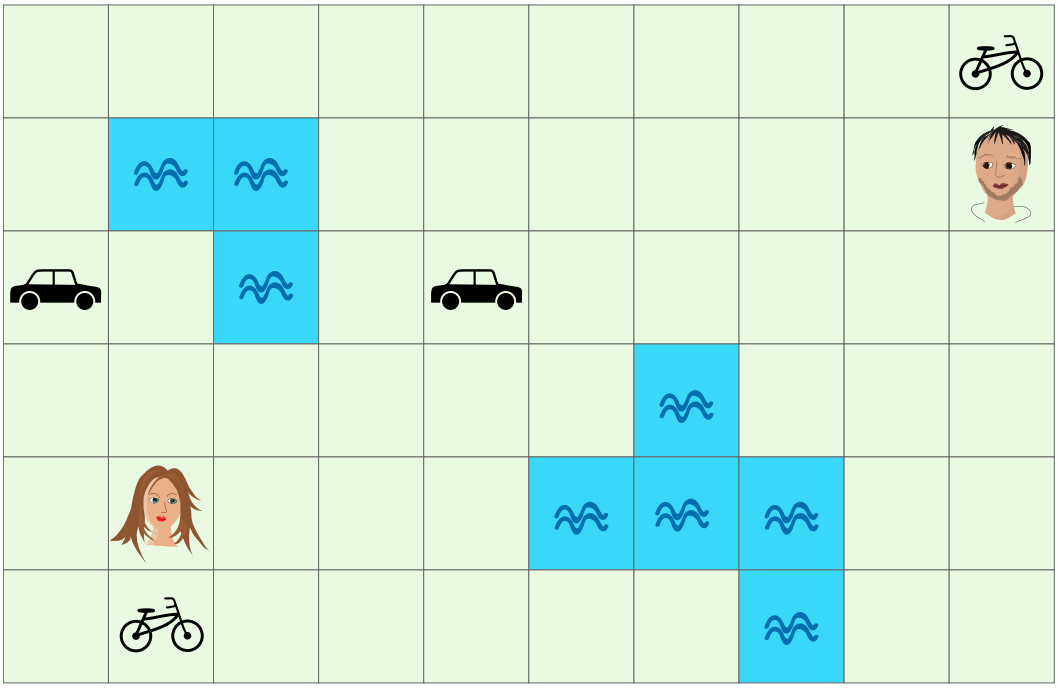
L'ordinamento dei dati gioca un ruolo importante in molti programmi informatici. Il metodo usato in questo compito per determinare l'ordine degli strati liquidi è chiamato *ordinamento topologico*. Si usa per ordinare gli oggetti per i quali sono conosciute relazioni del tipo predecessore/successore.

Parole chiave e siti web

- Algortimo di ordinamento: `https://it.wikipedia.org/wiki/Algoritmo_di_ordinamento`
- Ordinamento topologica: `https://it.wikipedia.org/wiki/Ordinamento_topologico`

15. Riunione veloce

Due amici vogliono incontrarsi il più presto possibile. Possono spostarsi da una casella a una casella adiacente a sinistra, a destra, in alto o in basso. Ci vuole 1 minuto per farlo a piedi. Se raggiungono una casella con un veicolo, possono usarlo. Con una bicicletta possono raggiungere 2 campi in un minuto e con una macchina 5 campi. I cambi di direzione sono possibili. Non possono attraversare le superfici d'acqua.

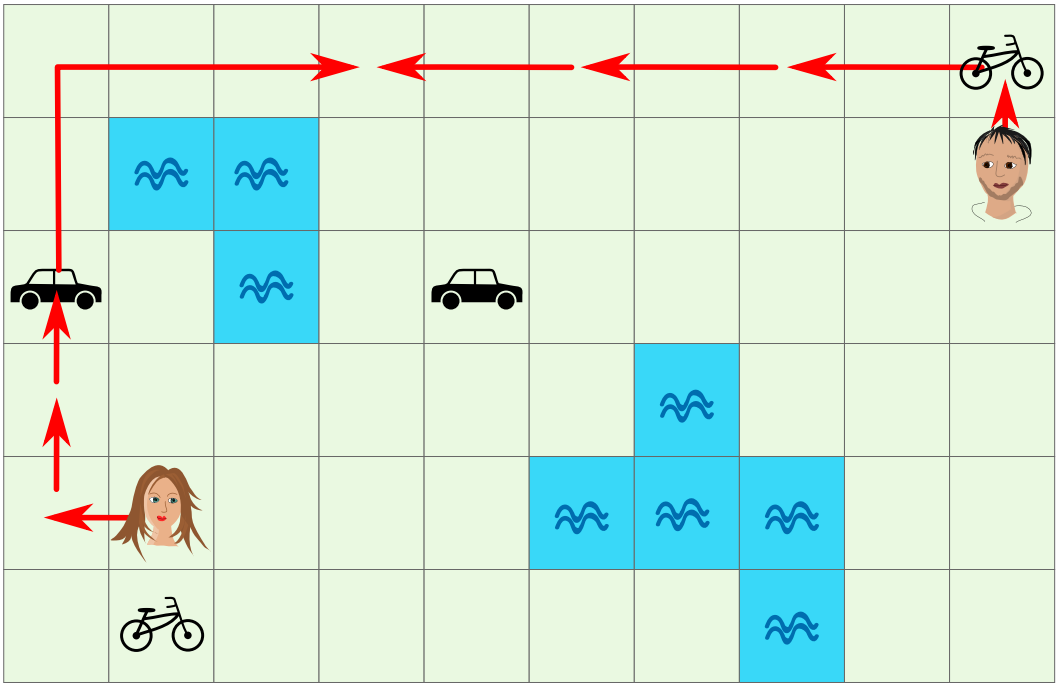


Di quanti minuti hanno bisogno i due amici per incontrarsi su una casella?

A) 1 minuto
B) 2 minuti
C) 3 minuti
D) 4 minuti
E) 5 minuti
F) 6 minuti

Soluzione

La risposta corretta è 4. L'immagine mostra un percorso che permette ai due amici di incontrarsi in una casella in 4 minuti.



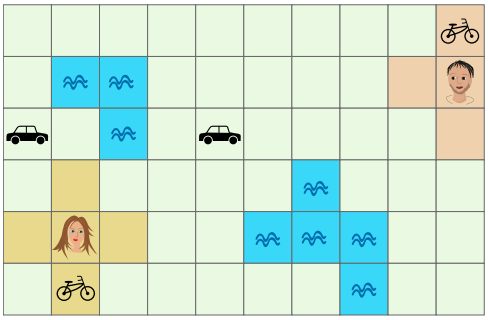
Ora bisogna dimostrare che non possono incontrarsi in 3 minuti: I due amici sono a 11 campi di distanza l'uno dall'altro. In 3 minuti, però, possono avvicinarsi a piedi solo un totale di 6 campi. Se uno ha raggiunto la bicicletta e l'altro sta camminando, allora possono avvicinarsi di 9 caselle l'uno all'altro in 3 minuti, che non è neanche abbastanza. Anche se entrambi camminano verso una bicicletta, non è sufficiente. Perché allora potrebbero avvicinarsi di 12 spazi in 3 minuti, ma le due biciclette sono distanti 13 spazi.

Quindi l'unica opzione è usare una macchina. In 3 minuti, solo la ragazza può raggiungere una macchina. Ma poi non c'è più tempo per usare la macchina. E in 3 minuti il ragazzo non può raggiungere una casella con una macchina.

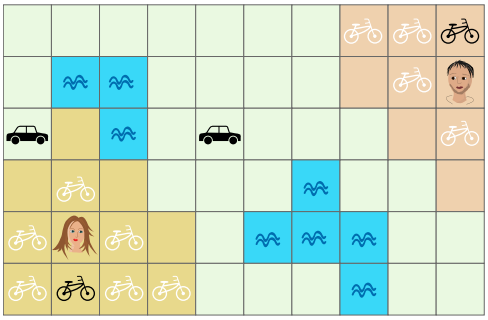
Questa è l'informatica!

Come hai risolto il compito? Hai trovato un percorso veloce per caso e hai sperato che non ce ne fosse uno più veloce? O hai provato molte possibilità e hai scoperto quella più veloce?

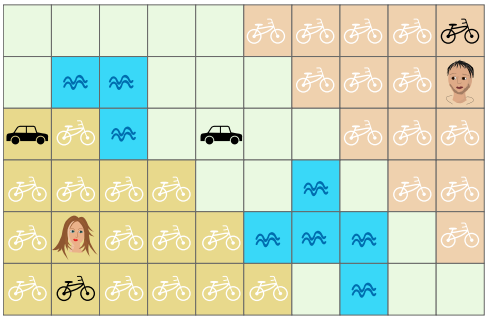
I programmi per computer progettati per questo tipo di problemi di solito usano un metodo chiamato *ricerca in ampiezza*. In questo problema, la ricerca in ampiezza funziona come segue:



1. Segna tutte le caselle che possono essere raggiunte dai due amici in un minuto.

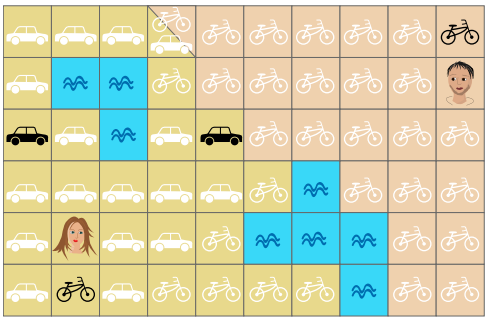


2. Segna tutti i campi che possono essere raggiunti in (al massimo) un minuto dai campi segnati nel passo 1. Annota anche i mezzi di trasporto utilizzati.



3. Segna tutti i campi che possono essere raggiunti in un minuto dai campi che sono stati segnati nel passo 2.

Poiché le due aree che hai segnato finora non si sovrappongono, gli amici non possono incontrarsi dopo 3 minuti.



4. Ora segna tutti i campi che possono essere raggiunti in un minuto dai campi segnati nel passo 3.

Ora le due aree si sovrappongono in un campo. Può essere raggiunto dopo 4 minuti dalla ragazza con una macchina e dal ragazzo con una bicicletta. I sistemi di navigazione trovano il percorso più veloce tra due punti. Si assicurano che il percorso segua strade e sentieri adatti - e non attraverso la campagna e i fiumi. Questo compito è simile al problema della navigazione, tranne che qui due persone devono essere guidate verso una destinazione comune - inizialmente sconosciuta - piuttosto che una sola persona verso una destinazione fissa.

Poiché un computer procede sistematicamente nella ricerca in ampiezza, trova anche soluzioni che non sono immediatamente ovvie. A volte una deviazione con meno semafori è più veloce del percorso

più breve tra la partenza e la destinazione. Un collegamento in treno con un cambio di treno può essere più veloce di un collegamento diretto in autobus. Nell'informatica, ci sono diversi metodi per trovare la migliore soluzione a un problema di questo tipo. Oltre alla ricerca in ampiezza, che è stata appena descritta, c'è anche un approccio chiamato *Branch and Bound* (*ramificazione e limitazione* in inglese).

Nella ricerca in ampiezza, viene considerata ogni soluzione parziale. Con *Branch and Bound*, non si perseguono ulteriormente le soluzioni parziali se si sa che non possono portare alla soluzione ottimale.

Se un problema diventa troppo complesso, ci vorrebbe troppo tempo anche per il computer più veloce del mondo per passare attraverso tutte le possibilità e trovare la soluzione migliore. In pratica, con un sistema di navigazione, è spesso sufficiente trovare un ottimo percorso, anche se non è il migliore possibile (se puoi raggiungere la tua destinazione in 78 minuti, probabilmente non ti importa se potrebbe teoricamente essere raggiunta in 77 minuti).

Parole chiave e siti web

- Ricerca in ampiezza: `https://it.wikipedia.org/wiki/Ricerca_in_ampiezza`
- Algortimo branch and bound: `https://it.wikipedia.org/wiki/Branch_and_bound`

A. Autori dei quesiti

Sarah Atkins

Michael Barot

Liam Baumann

Wilfried Baumann

Javier Bilbao

Sarah Chan

Kris Coolsaet

Valentina Dagienė

Christian Datzko

Susanne Datzko

Margarita Flores-Sicich

Fabian Frei

Gerald Futschek

Thomas Galler

Christian Giang

Yasemin Gülbahar

Ezgi Arzu Güneş

Juraj Hromkovič

Alisher Ikramov

YongJu Jeon

Soojin Jun

Filiz Kalelioğlu

Dong Yoon Kim

Jihye Kim

Vaidotas Kinčius

Regula Lacher

Taina Lehtimäki

Marielle Léonard

Tom Naughton

Mochammad Irfan Noviana

Jean-Philippe Pellet

Zsuzsa Pluhár

Wolfgang Pohl

Peter Rossmanith

Eljakim Schrijvers

Tomas Šiaulys

Timur Sitdikov

Bernadette Spieler

Ahto Truu

Florentina Voboril

Michael Weigend

Kyra Willekes

Hongjin Yeh

Mija Zaļūksne

B. Sponsoring: concorso 2021

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

https://educatec.ch/
educaTEC

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

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

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

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

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

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

`https://www.phzh.ch/`
Pädagogische Hochschule Zürich

C. Ulteriori offerte

Diventate membri della SSII `http://svia-ssie-ssii.ch/verein/mitgliedschaft/` sostenendo in questo modo il Castoro Informatico.

Chi insegna presso una scuola dell'obbligo, media superiore, professionale o universitaria in Svizzera può diventare membro ordinario della SSII.

Scuole, associazioni o altre organizzazioni possono essere ammesse come membro collettivo.