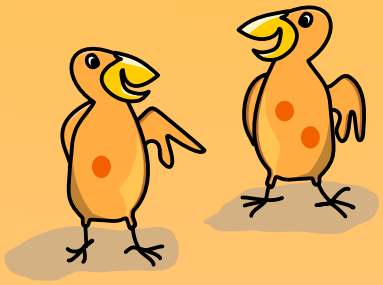
INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Quesiti e soluzioni 2021

5º e 6º anno scolastico



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

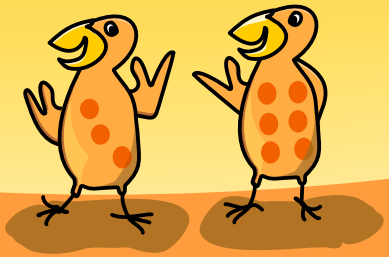
A cura di:

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

01010011010101100100100
10000010010110101010011
01010011010010010100101
00101101010100110101001
01001001010010010010001

SSII

www.svia-ssie-ssii.ch
schweizerischervereinfürinformatikind
erausbildung//sociétésuissepourl'infor
matiquedansl'enseignement//societàsviz
zeraperl'informaticanell'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/`

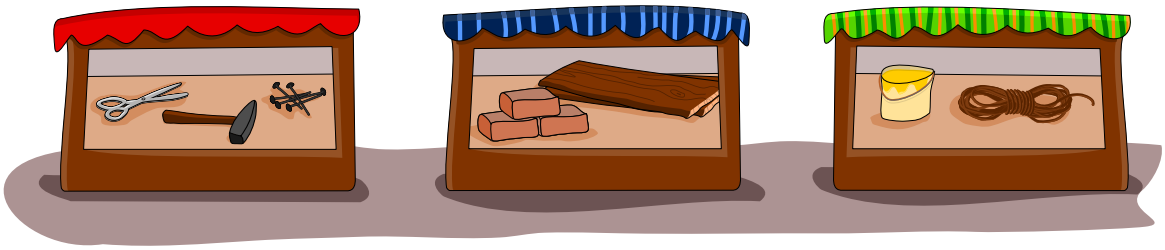
Indice

1. Costruzione di ponte

Bella vuole costruire un ponte su un ruscello. Per farlo ha bisogno di un martello, chiodi, tavole e una corda. Fortunatamente trova un martello e una corda nella cantina.

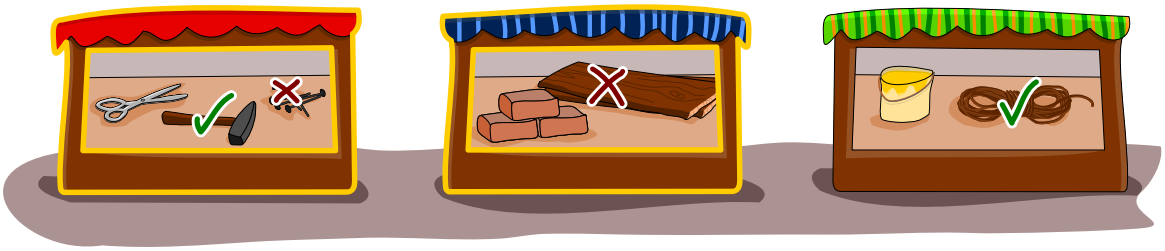
Deve però comprare gli altri materiali. Qui sotto puoi vedere tre negozi e quello che vendono.

Dove può Bella comprare le altre cose?



Soluzione

Questo è corretto:



Questa è l'informatica!

I negozi di questo compito vendono un totale di sette cose: forbici, martelli, chiodi, mattoni, assi, corde e secchi. È un bel po' di roba! Le due cose che Bella deve comprare sono un *sottoinsieme*. Si può disegnare così: si mostrano tutte le cose dell'insieme e si segna per ogni cosa se appartiene o no al sottoinsieme della spesa di Bella:

Allo stesso modo, si può dipingere ciò che i negozi vendono, per esempio il negozio sulla sinistra:

In questo modo puoi vedere a prima vista cosa Bella può comprare nel negozio sulla sinistra: i chiodi hanno un segno di spunta verde nel sottoinsieme di acquisto e nel sottoinsieme di vendita.

Anche i programmi informatici devono spesso confrontare degli *insiemi* (*set* in inglese). Possono farlo come mostrato sopra: per ogni cosa che può accadere, è necessario un *bit*. In un bit, un computer può memorizzare uno dei due valori, come «sì» e «no». In questo caso, memorizza se questa cosa appartiene a un insieme («sì») oppure non vi appartiene («no»). Poi un programma può confrontare due insiemi nella seguente maniera: controlla se il bit per una cosa in un insieme è «sì» e il bit per la stessa cosa nell'altro set è anch'esso «sì». Un computer può eseguire un tale controllo di due bit in modo particolarmente rapido. Nell'informatica, la descrizione degli insiemi con i bit è quindi molto comune.

Parole chiave e siti web

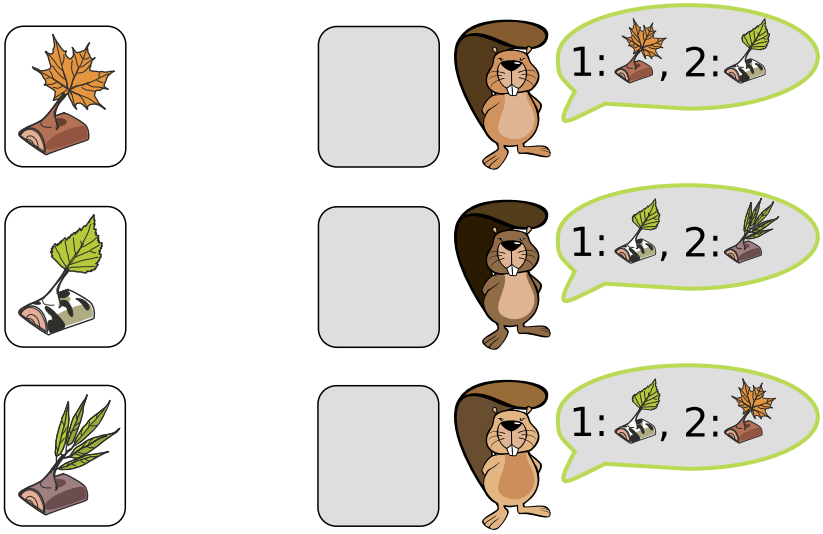
- Set: `https://it.wikipedia.org/wiki/Set_(informatica)`
- Bits: `https://it.wikipedia.org/wiki/Bit`

2. Regalo preferito

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

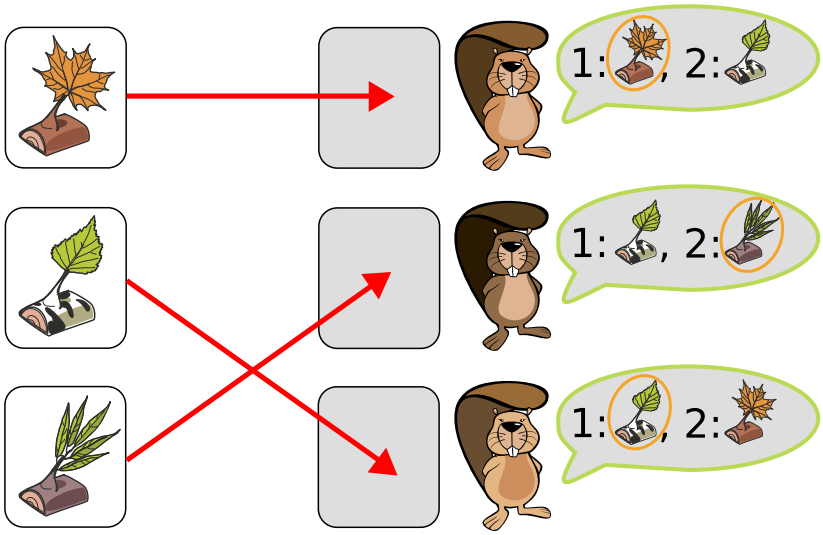
1. Il maggior numero possibile di castorini dovrebbe ricevere il loro regalo preferito.
2. Gli altri dovrebbero ricevere il secondo regalo preferito.

Dai ai castorini i regali giusti.



Soluzione

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



Solo il secondo castoro vuole il terzo regalo, quindi deve ottenerlo. Altrimenti, qualcun altro riceverebbe qualcosa che non è né il regalo preferito né il secondo preferito. Per gli altri due regali, la divisione è allora chiara: ognuno può avere 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.

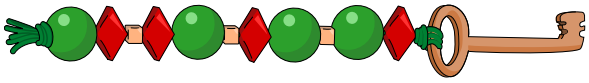
Una possibilità è quella di dare un valore alle assegnazioni: Il regalo preferito ha valore 1 e il secondo regalo preferito ha valore 2. Vogliamo minimizzare il valore totale. Un *accopiamento* (in inglese *matching*) è *ottimale* se non c'è un altro accopiamento con più prime elezioni soddisfatte. In 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? L'informatica è una materia molto versatile.

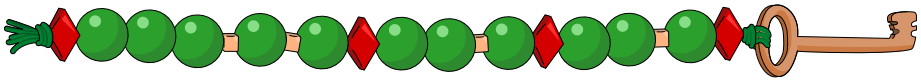
Parole chiave e siti web

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

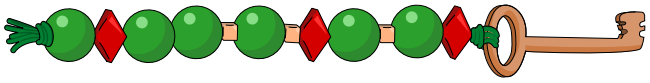
3. Portachiavi

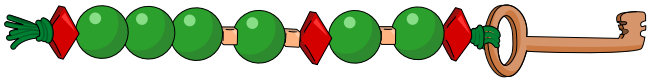
ANNA, BELLA e LENA hanno costruito dei portachiavi con i loro nomi. Hanno usato due tipi di perline per le lettere:  e  . Separano le lettere individuali con questa perlina:  .

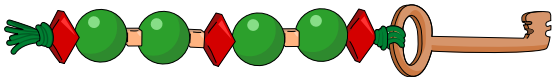
ANNA 

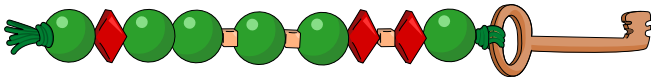
BELLA 

Quale portachiavi ha fatto LENA?

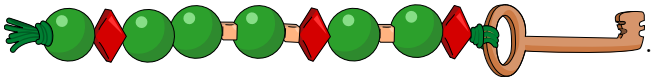
A) 

B) 

C) 

D) 

Soluzione

La risposta corretta è A)  .

La parola LENA inizia con L. In BELLA L è la terza e quarta lettera e riconosciamo in BELLA per L la sequenza di perline  . Solo le risposte A) e D) iniziano con questa lettera e possono essere la soluzione. La seconda lettera di LENA, cioè E, è anche la seconda di BELLA, lì troviamo questa perlina  . Sia in A) che in D) abbiamo  come seconda lettera, quindi potrebbero ancora essere entrambi la soluzione corretta. In seguito vogliamo trovare le perline per la lettera N. Nel portachiavi di ANNA troviamo per N la sequenza di perline  . E queste sono le sguenti perline solo nella soluzione A).

Un altro modo per trovare le perline per il portachiavi di LENA è fare una tabella con le perline per le lettere che già conosciamo. Dal portachiavi di ANNA troviamo per A la sequenza di perline  e per N  . Dal portachiavi di BELLA troviamo la sequenza di perline per B:  , per E:  e per L:  .

Lettera	Perline
A	
N	
B	
E	
L	

Quindi possiamo fare il portachiavi per LENA dalle perline  ,  ,  e  , se separiamo ulteriormente le singole lettere con la perlina  . Ecco come si ottiene questo portachiavi:  , che è la risposta A). Se decodifichiamo gli altri portachiavi dati usando la tabella di codifica, troviamo BENA per B), NENA per C) e LEAN per D).

Questa è l'informatica!

L'informazione è *codificata* per poter trasmettere messaggi in certe condizioni o per trasmettere l'informazione segretamente (*codificata*). In questo compito, la codifica è basata sul codice Morse. Il punto • del codice Morse è rappresentato dalla perlina rotonda  e la barra — da  . La lettera A è in codice Morse • —, quindi nel codice delle perline  . Per codificare qualsiasi testo, abbiamo bisogno di codici per tutte le lettere dell'alfabeto.

Il codice Morse ha avuto origine nel XIX secolo. Samuel Morse ha inventato un semplice telegrafo a scrittura elettromagnetica nel 1837. Il codice usato all'epoca comprendeva solo le dieci cifre da 0 a 9; i numeri trasmessi dovevano essere tradotti in lettere e parole con l'aiuto di una tabella. Alfred Lewis Vail, un impiegato di Morse, ha sviluppato il primo codice che includeva anche le lettere nel 1838. Il codice è stato sviluppato per trasmettere testi acusticamente, otticamente o elettricamente sulle linee telegrafiche. Un punto è un tempo di trasmissione breve e un trattino è tre volte più lungo. Ci deve

essere una pausa tra le lettere. Una pausa più lunga separa le parole. Il codice Morse è usato ancora oggi per il segnale di soccorso SOS. SOS in codice Morse: • • • — — — • • • (3x breve, 3x lungo, 3x breve) può essere trasmesso molto facilmente urlando, bussando, o con una torcia.

Nell'elaborazione elettronica dei dati, i caratteri sono codificati tramite un valore numerico per poterli trasmettere o memorizzare.

Parole chiave e siti web

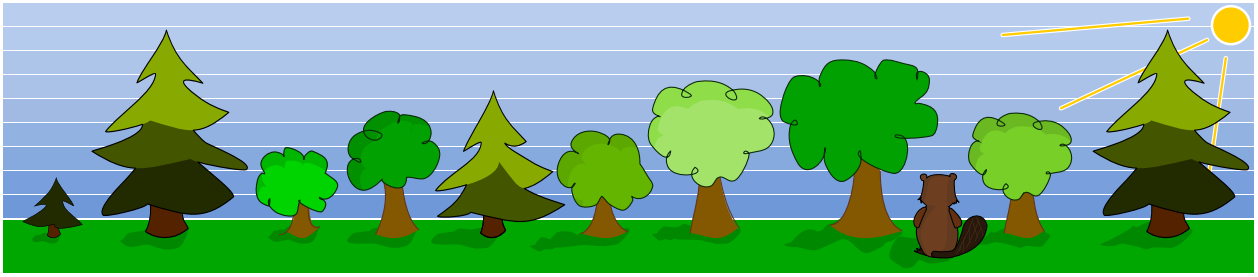
- Codifica di caretteri: `https://it.wikipedia.org/wiki/Codifica_di_caratteri`
- Codice Morse: `https://it.wikipedia.org/wiki/Codice_Morse`

4. Cade l'albero!

Un castoro vuole costruire una diga. Per assicurarsi di tagliare sempre gli alberi giusti ha pensato a due condizioni. Ha deciso di tagliare un albero solo se

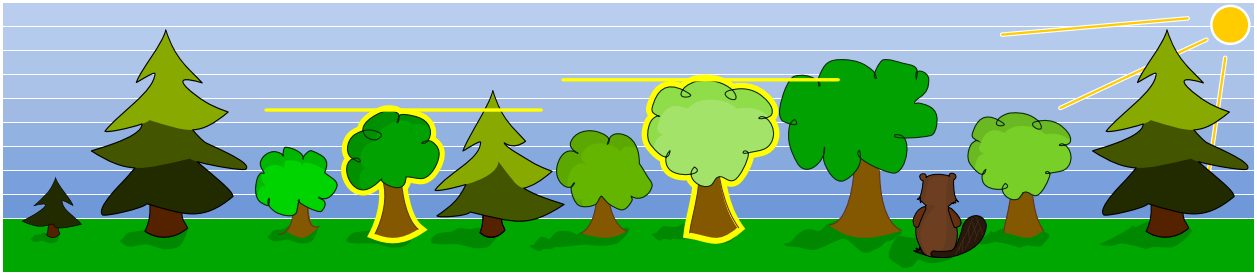
- un albero più piccolo cresce direttamente a sinistra di esso e
- un albero più grande cresce direttamente alla sua destra.

Quali alberi taglierà il castoro?



Soluzione

Solo gli alberi al quarto e settimo posto soddisfano le condizioni date: c'è un piccolo albero direttamente sulla sinistra E un albero più grande direttamente sulla destra.



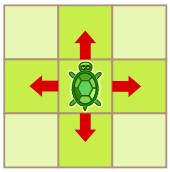
Questa è l'informatica!

In informatica il compito è spesso quello di risolvere problemi che sono specificati da un insieme di vincoli logici. Il compito è quello di trovare una soluzione che soddisfi tutti i vincoli dati. Compiti più complessi di questo possono essere gestiti combinando i vincoli usando gli *operatori logici*. La congiunzione (operatore $\wedge$ o anche AND), per esempio, restituisce «vero» come risultato nell'espressione A $\wedge$ B esattamente quando i due vincoli sono anche veri. In questo compito, questo sarebbe: «l'albero a sinistra è più piccolo» $\wedge$ «l'albero a destra è più grande». Questo principio di base si verifica in quasi tutte le aree dell'informatica, come in molti algoritmi di ordinamento, per esempio il *Bubble Sort*. In questo algoritmo diversi oggetti di una lista sono sempre controllati per certi vincoli, al fine di spostarli successivamente in un'altra posizione della lista. Questo principio viene ripetuto fino a quando la lista è completamente ordinata.

Parole chiave e siti web

- Pensiero algoritmico (algorithmic thinking)
- Operatori logici: `https://it.wikiversity.org/wiki/Operatori_Logici_(superiori)`
- Algoritmo di ordinamento: `https://it.wikipedia.org/wiki/Algoritmo_di_ordinamento`
- Problema di soddisfacimento di vincoli: `https://it.wikipedia.org/wiki/Problema_di_soddisfacimento_di_vincoli`

5. Cammino della tartaruga



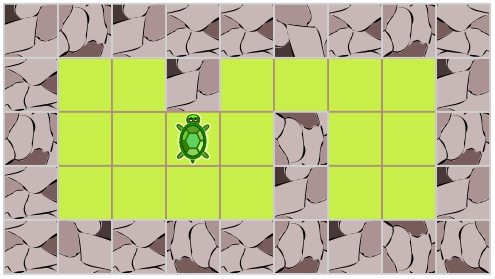
Una tartaruga vuole passeggiare in diversi giardini. Ogni giardino è suddiviso in zone (quadrati) che sono coperte di erba o di pietre. La tartaruga non può passeggiare dove ci sono le pietre. Tuttavia, può spostarsi da un quadrato d'erba a un altro quadrato d'erba proprio accanto.

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.

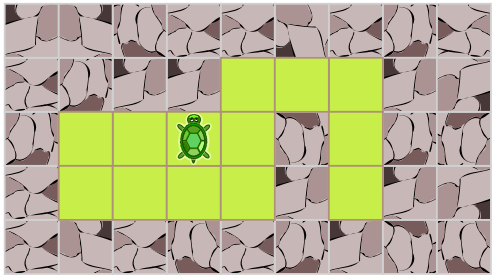
Sfortunatamente, la tartaruga non può visitare tutte le zone d'erba esattamente una volta su uno dei giardini.

Di quale giardino si tratta?

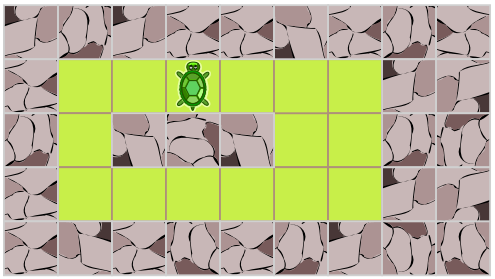
A)



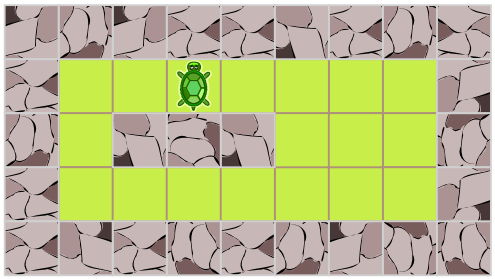
B)



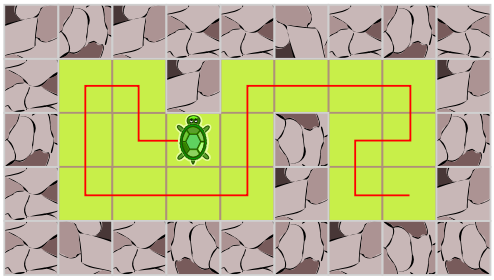
C)

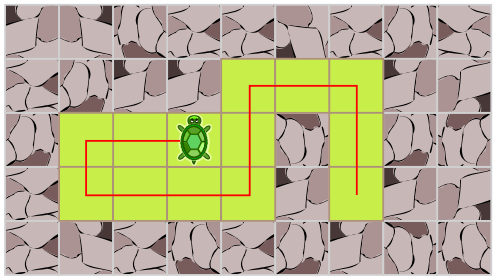


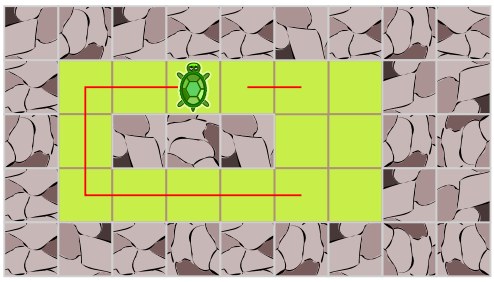
D)

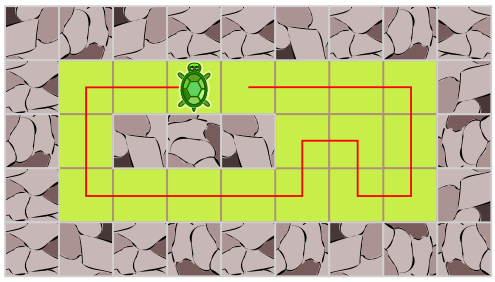


Soluzione

A) 

B) 

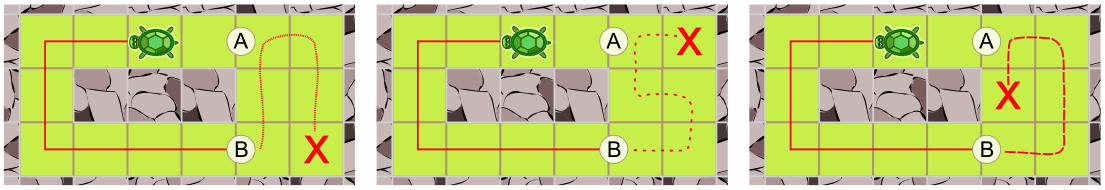
C) 

D) 

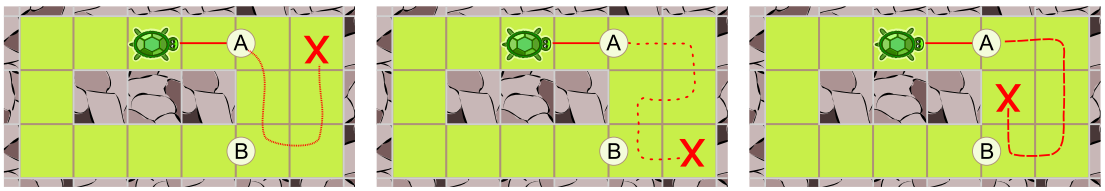
La tartaruga può visitare tutte le zone d'erba dei giardini A, B e D.

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.



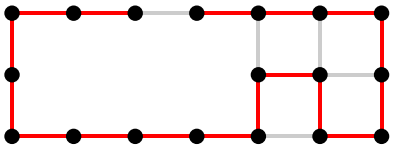
- Se la tartaruga va prima a destra, arriva ad A e dovrebbe visitare i 6 campi in modo da raggiungere il punto B alla fine. Ora si può argomentare come prima, bisogna solo scambiare la parte superiore e inferiore. Quindi non c'è nessun cammino adatto neanche in questo modo.



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

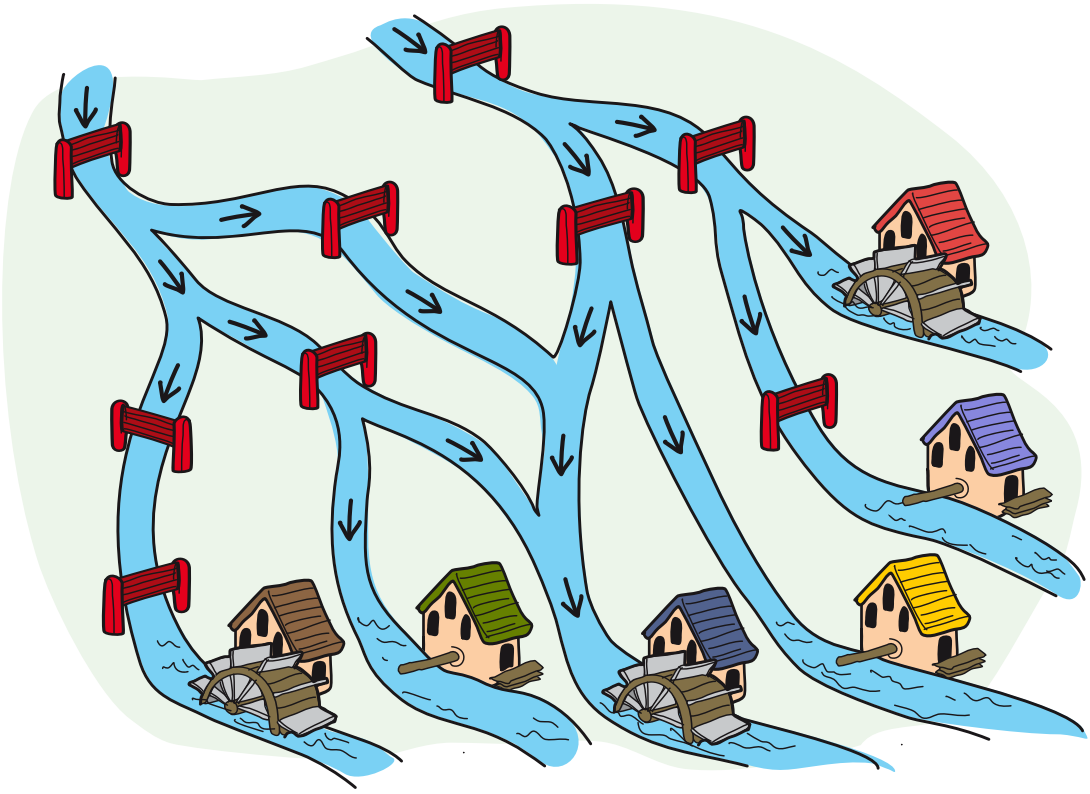
- Cammino hamitoniano: `https://it.wikipedia.org/wiki/Cammino_hamiltoniano`

6. 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.

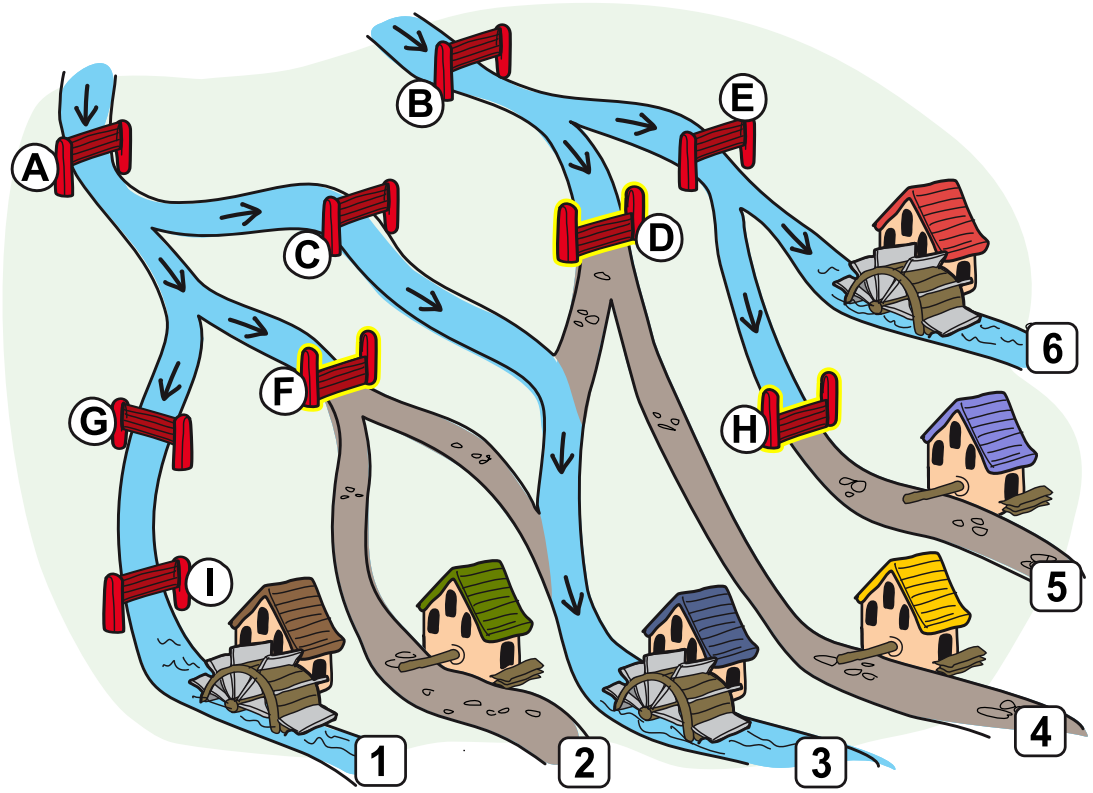
L'acqua può scorrere solo verso il basso. Una valvola a scorrimento chiusa ferma l'acqua.

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»).

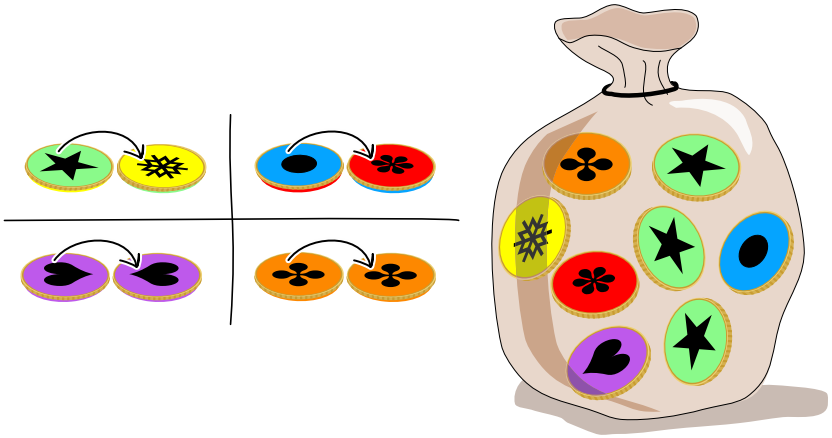
Quando si programma, è importante formulare correttamente le condizioni. I collegamenti con gli operatori logici sono utili qui per formulare condizioni più complesse. Sia nella selezione (ramificazione con l'aiuto di `if`) che nell'iterazione condizionale (ciclo `while`), le condizioni sono usate per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

7. Sacco di monete

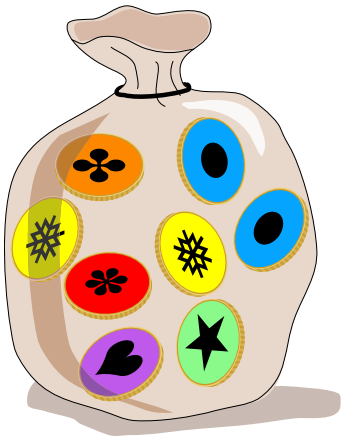
Nel paese di Emil ci sono 4 tipi diversi di monete. Qui puoi vedere i due lati di queste monete e anche il sacco di Emil con le sue monete.



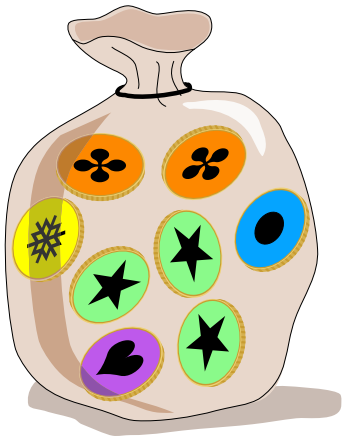
Il suo sacco viene in seguito agitato.

Qual è il sacco di Emil?

A)



B)



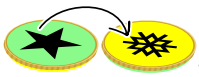
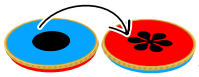
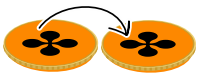
C)

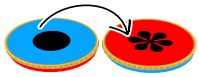
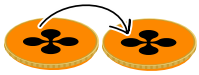
D)

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.

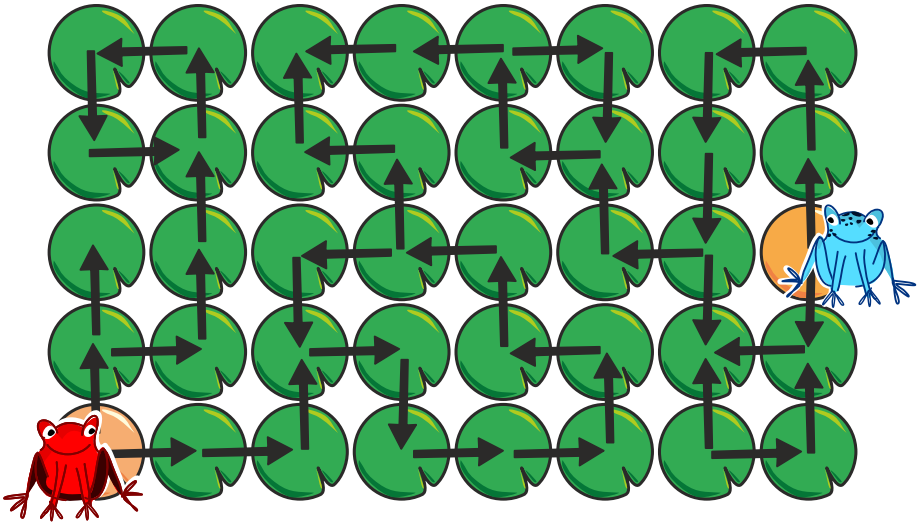
Parole chiave e siti web

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

8. Si incontrano?

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

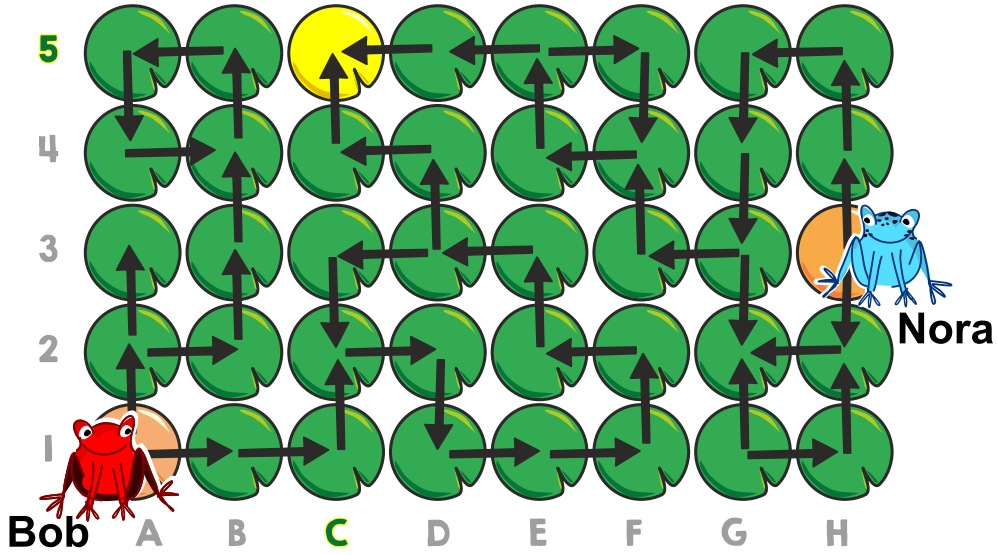
Su quale foglia di ninfea possono incontrarsi?



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

Soluzione

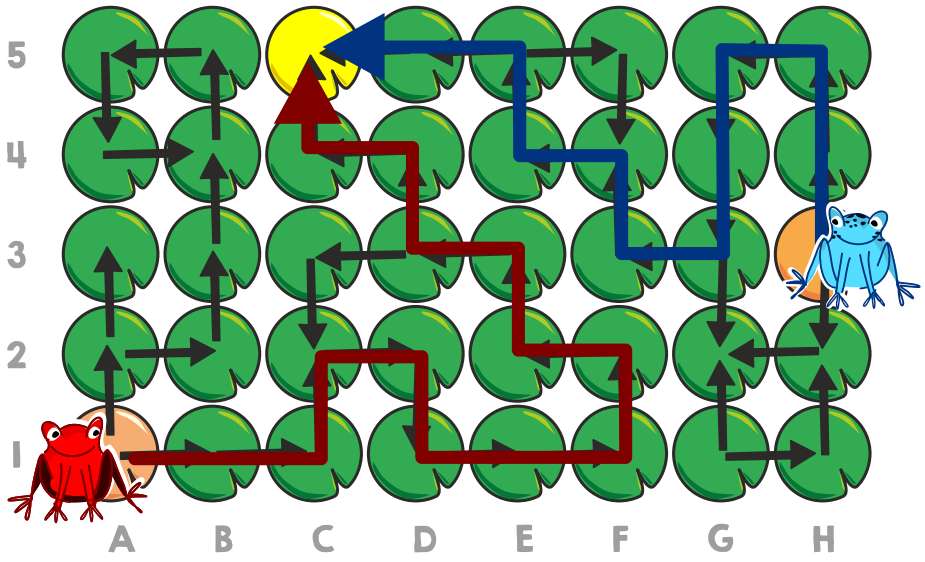
Le rane possono incontrarsi solo sulla foglia C5.



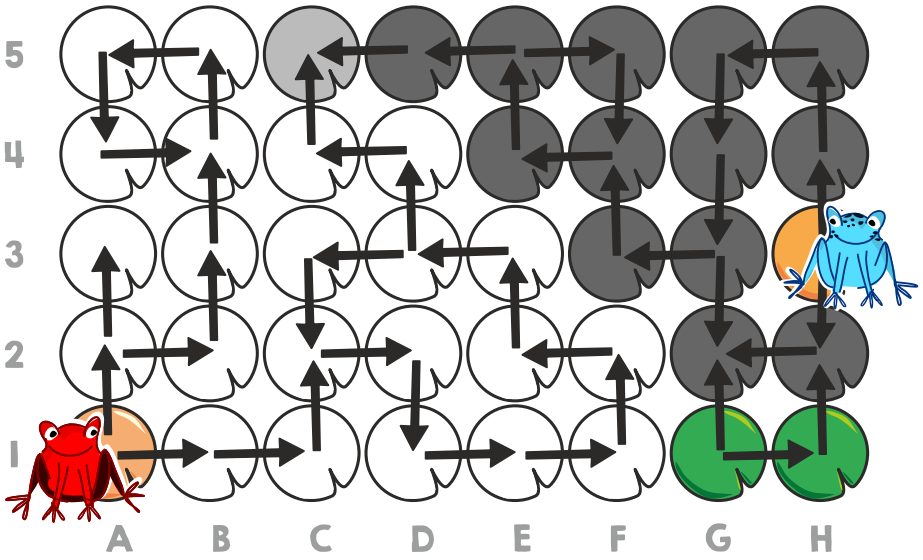
Nella sua posizione di partenza, la rana rossa Bob ha due opzioni: se va «su», o entra nel vicolo cieco A3 o rimane bloccato nel cerchio che inizia in B4. Se inizialmente va «a destra» (a B1), può prima saltare su D3. Lì può saltare «a sinistra» in un cerchio, che lo riporta a D3, o «su», che lo porta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

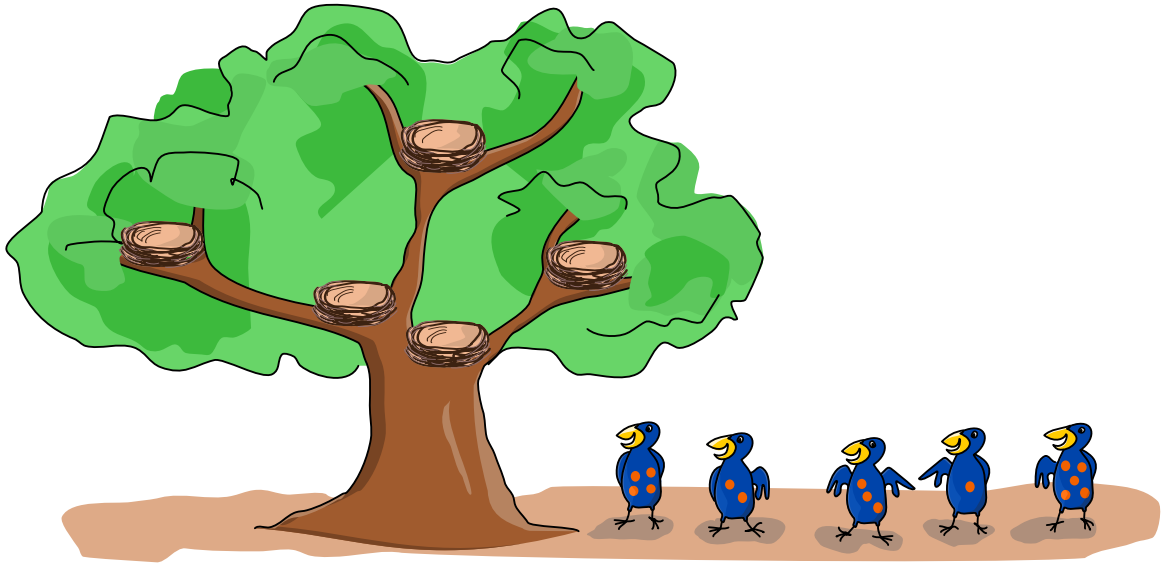
9. Nidi dei Dottuccelli

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.

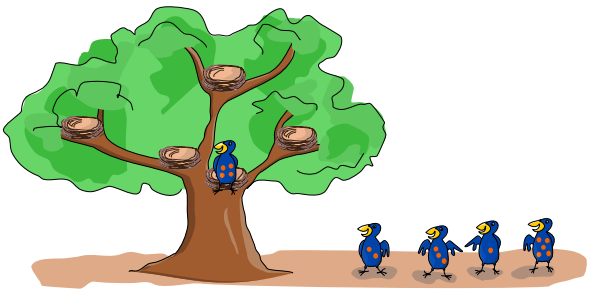
Dove sono i Dottuccelli alla fine? Metti ogni Dottuccello nel nido giusto.



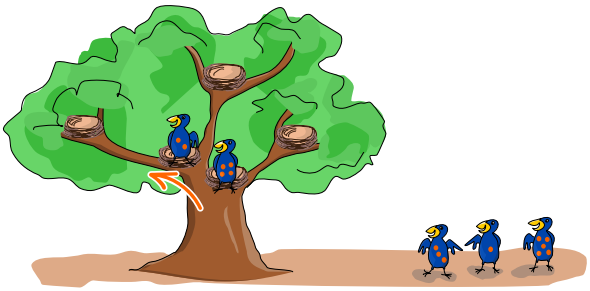
Soluzione

È così che si arriva alla soluzione giusta:

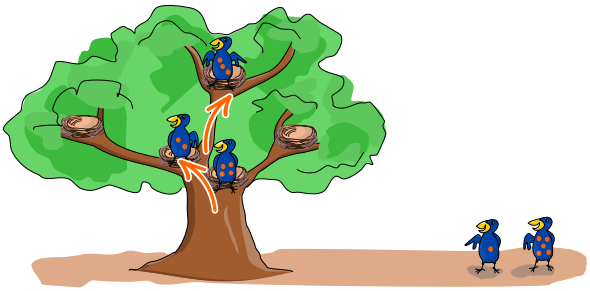
Il primo Dottuccello, quello con 4 pois, si appollaia nel nido più basso e rimane lì.



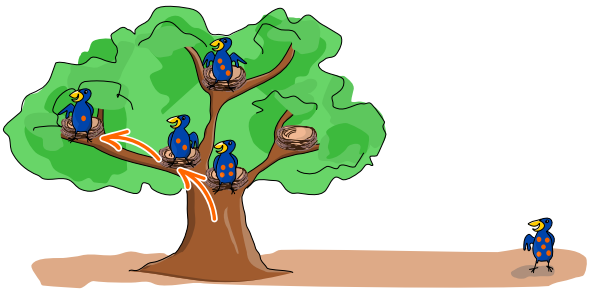
Il secondo Dottuccello ha 2 pois. Il primo Dottuccello con 4 pois si siede nel nido più basso. Poiché 4 è maggiore di 2, il secondo Dottuccello sale più a sinistra e si appollaia nel primo nido libero.



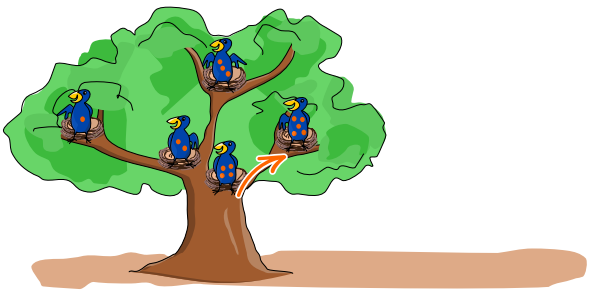
Il terzo Dottuccello ha 3 pois. Sale a sinistra al nido più basso, dove è seduto il Dottuccello con 4 pois, siccome 4 è maggiore di 3. Nel prossimo nido si trova il Dottuccello con 2 pois. Poiché 3 è maggiore di 2, il terzo Dottuccello sale a destra. Poi si appollaia nel prossimo nido libero. Questo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

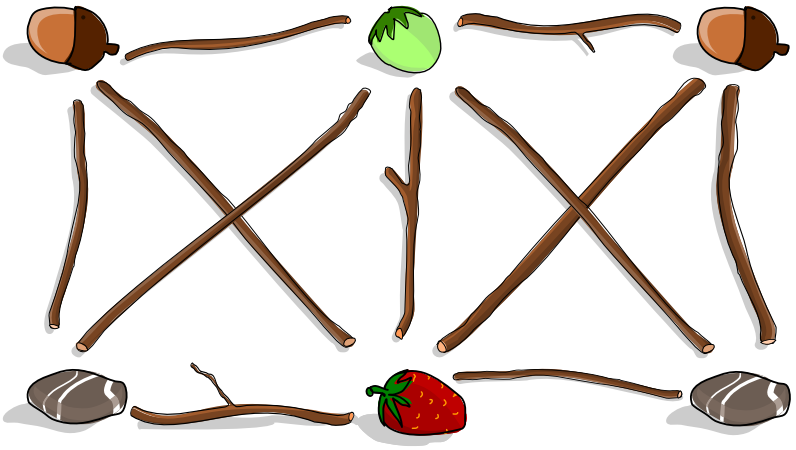
- Albero binario di ricerca: `https://it.wikipedia.org/wiki/Albero_binario_di_ricerca`

10. 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.

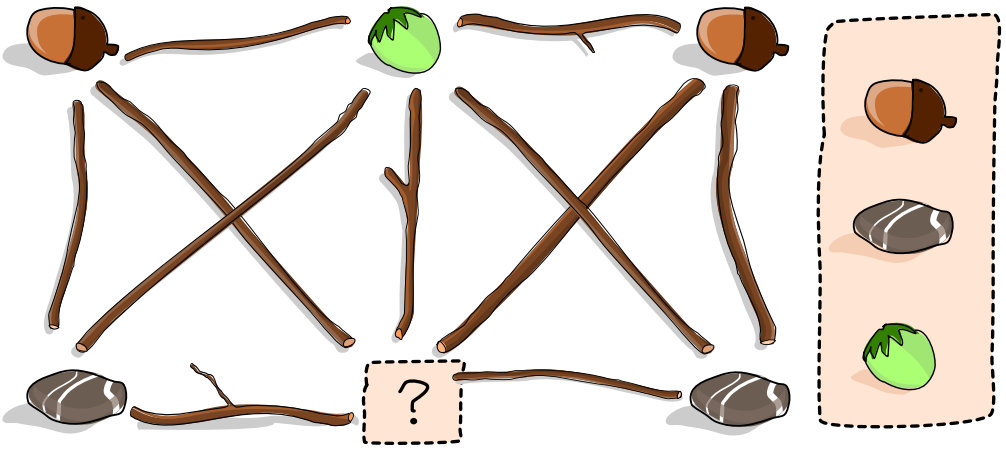
Ecco l'opera d'arte finita:



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

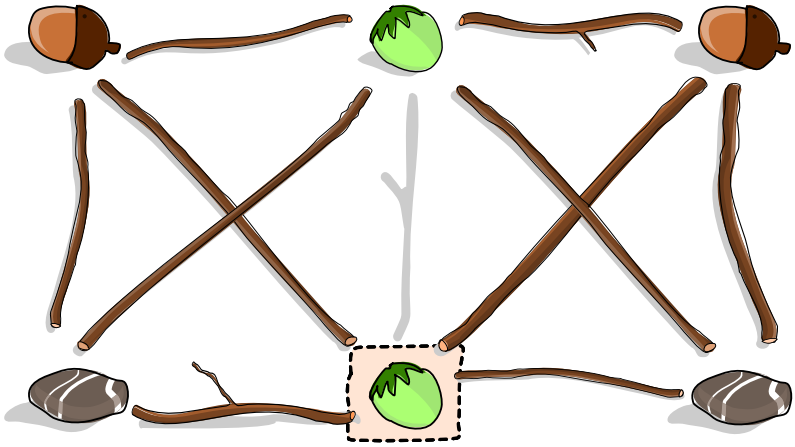
Puoi aiutarlo a coprire il misfatto?

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.



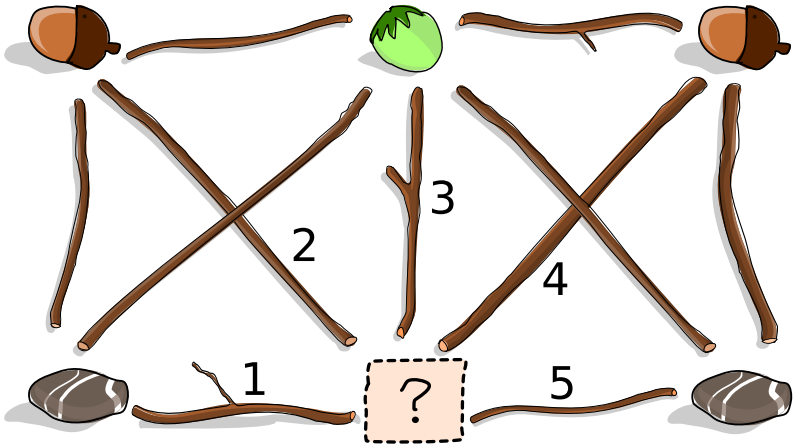
Soluzione

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.

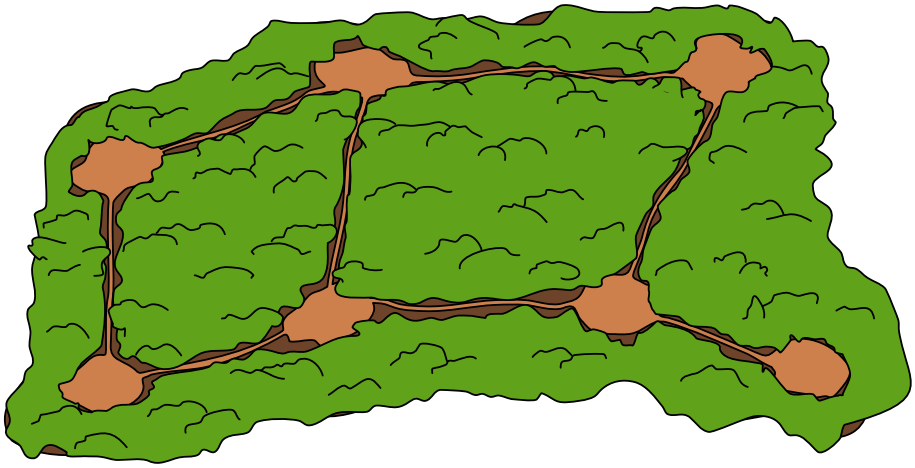
Parole chiave e siti web

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

11. Osservazione del bosco

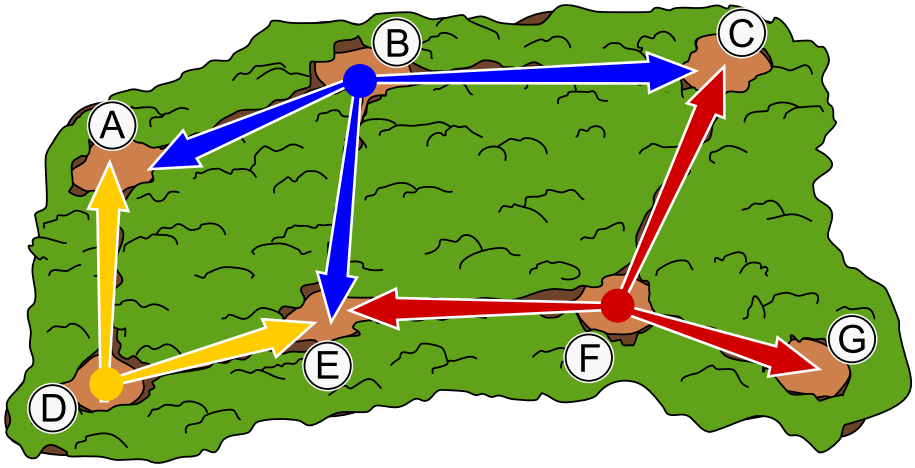
I forestali vogliono osservare gli animali sui sentieri del bosco. Da ogni radura possono osservare tutti i sentieri collegati con la radura successiva. Tutti i sentieri devono essere osservati dal minor numero possibile di forestali.

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.

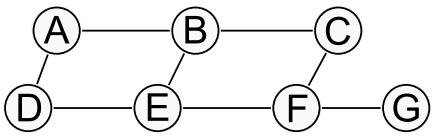
Quindi, sono necessari almeno tre forestali per osservare tutti i sentieri. Di conseguenza, la soluzione data qui è una soluzione con il minor numero possibile di forestali. In effetti, non esiste un'altra soluzione con esattamente tre forestali.

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

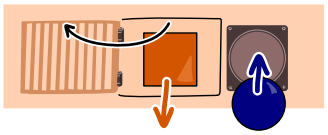
copertura minima dei vertici. Troviamo questi problemi di copertura dei vertici nella vita di tutti i giorni, per esempio quando si cercano le migliori posizioni per i lampioni o per il posizionamento intelligente delle telecamere di sorveglianza.

Parole chiave e siti web

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

12. Puzzle di compleanno

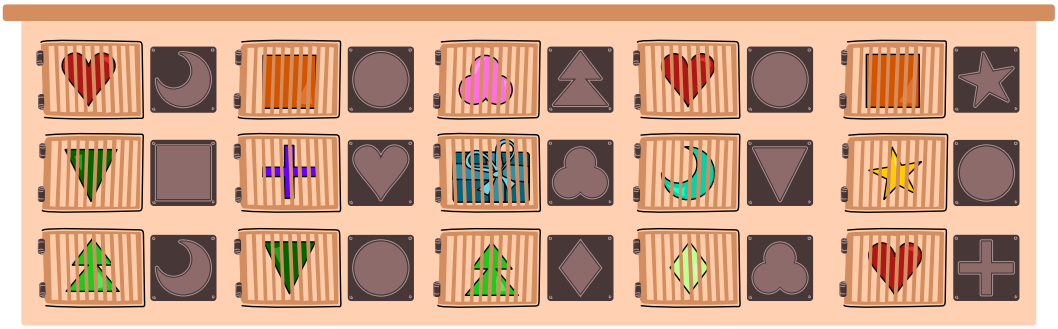
Bastian riceve una scatola con 15 porte per il suo compleanno. Dietro la porta centrale c'è il vero regalo. Dietro le altre porte ci sono blocchetti di varie forme. Ogni porta ha un foro a destra della porta. Bastian può aprire una porta inserendo un blocco della stessa forma nel foro - come una chiave.



All'inizio, Bastian ha questo blocco di costruzione rotondo: 

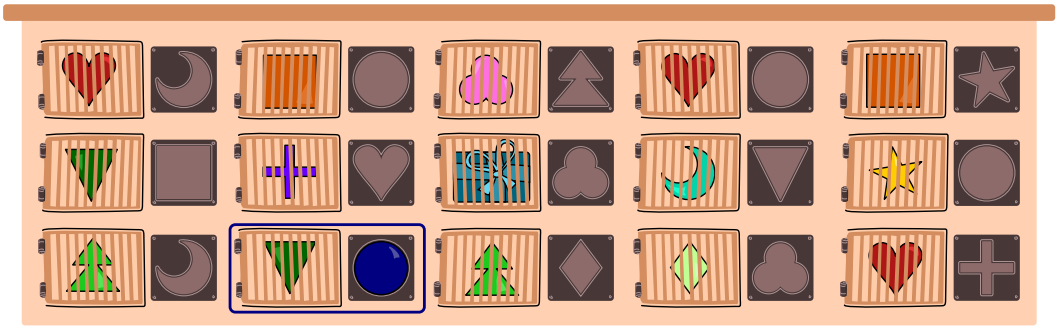
Vuole aprire un massimo di cinque porte per raggiungere il regalo.

Quale porta deve aprire per prima Bastian?

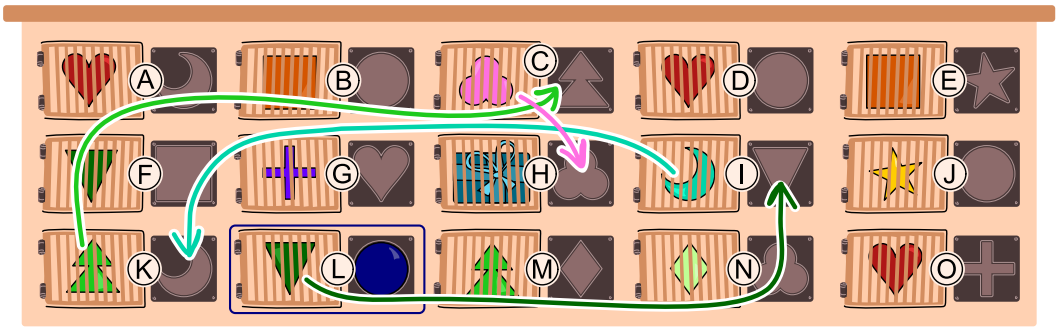


Soluzione

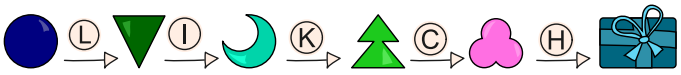
Bastian deve prima aprire la porta segnata in blu:



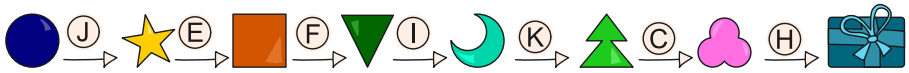
Nell'immagine seguente, le porte sono contrassegnate da lettere e le frecce mostrano come Bastian raggiunge il regalo con un totale di 5 aperture di porte.



Possiamo anche rappresentare l'ordine in cui apre le cinque porte come segue.



Ci sarebbero anche altri modi per il regalo, per esempio il seguente.

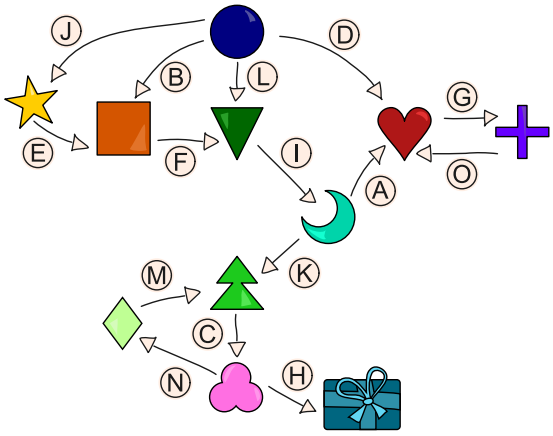


Ma questi modi sono troppo lunghi, bisognerebbe aprire più di cinque porte. Provare tutte le possibilità richiede molto tempo.

In questo caso, il modo più veloce per trovare il percorso più breve e quindi la soluzione corretta è quello di utilizzare una cosiddetta ricerca a ritroso: inizia dalla porta con il regalo e poi guarda quale blocco di costruzione ti serve.

Questa è l'informatica!

Con un po' di impego e tempo, la situazione del compito può anche essere rappresentata come un *grafo*:



Un grafo è generalmente composto da *vertici* (nodi) e da *archi* (bordi) tra i vertici. Nel nostro caso abbiamo un vertice per ogni forma e il regalo. Gli archi qui sono frecce (chiamate anche archi *diretti*) e corrispondono alle porte. Ogni freccia punta dalla forma per aprire la porta alla forma dietro la porta.

L'informatica ama molto lavorare con i grafi. Da un lato, forniscono spesso vivide rappresentazioni di relazioni astratte.

D'altra parte, esistono algoritmi già pronti che rispondono alle nostre domande sui grafi in modo molto efficiente. Per i compiti più complicati, lo sforzo di redigere il grafo può quindi ripagare rapidamente.

Nel compito in questione, stiamo cercando un percorso di lunghezza al massimo 5 dal blocco ricevuto  al regalo  . Un buon algoritmo per questo è la cosiddetta *ricerca in ampiezza*. Questo funziona sia per grafi con archi diretti, come nel compito, sia per grafi con archi non diretti.

Parole chiave e siti web

- Grafo diretto, digrafo: `https://it.wikipedia.org/wiki/Digrafo_(matematica)`
- Ricerca in ampiezza: `https://it.wikipedia.org/wiki/Ricerca_in_ampiezza`

A. Autori dei quesiti

Sarah Atkins

Michael Barot

Liam Baumann

Linda Björk Bergsveinsdóttir

Sarah Chan

Valentina Dagienė

Christian Datzko

Susanne Datzko

Nora A. Escherle

Margarita Flores-Sicich

Fabian Frei

Gerald Futschek

Christian Giang

Yasemin Gülbahar

Ezgi Arzu Güneş

Benjamin Hirsch

Juraj Hromkovič

Andrea Hrušecká

Tiberiu Iorgulescu

Ungyeol Jung

Filiz Kalelioğlu

Martin Kandlhofer

Vaidotas Kinčius

Víctor Koleszar

Regula Lacher

Taina Lehtimäki

Marielle Léonard

Tom Naughton

Graciela Oyhenard

Jean-Philippe Pellet

Zsuzsa Pluhár

Wolfgang Pohl

Peter Rossmanith

Eljakim Schrijvers

Rosario Schunk

Troy Vasiga

Florentina Voboril

Michael Weigend

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.