INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Quesiti e soluzioni 2022

5º e 6º anno scolastico

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

A cura di:

Susanne Datzko, Nora A. Escherle, Masiar Babazadeh,
Christian Giang, Jean-Philippe Pellet

SSII
www.svia-ssie-ssii.ch
schweizerischer verein für informatik in der ausbildung // société suisse pour l'informatique dans l'enseignement // società svizzera per l'informatica nell'insegnamento

HASLERSTIFTUNG ETH zürich | ABZ AUSBILDUNGS- UND BERATUNGSZENTRUM FÜR INFORMATIKUNTERRICHT hep/ Scuola universitaria professionale della Svizzera italiana SUPSI

Hanno collaborato al Castoro Informatico 2022

Masiar Babazadeh, Susanne Datzko, Jean-Philippe Pellet, Giovanni Serafini, Bernadette Spieler

Capo progetto: Nora A. Escherle

Un particolare ringraziamento per il lavoro sui quesiti del concorso Svizzero va a:
Juraj Hromkovič, Christian Datzko, Jens Gallenbacher, Regula Lacher: ETH Zurich, Ausbildungges- und Beratungszentrum für Informatikunterricht
Tobias Berner: Pädagogische Hoschschule Zürich
Waël Almoman: Collège Voltaire

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, Dave Oostendorp, Alieke Stijf, Kyra Willekes, Jo-Ann Bolten: 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, Lena Frölich: Senarclens Leu + Partner AG

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 2022 è stato organizzato dalla Società Svizzera per l'Informatica nell'Insegnamento (SSII) con il sostegno determinante della fondazione Hasler. Gli sponsor del concorso sono l'Ufficio per l'economia e il lavoro del Cantone di Zurigo e UBS.

Questo quaderno è stato creato il 22 novembre 2023 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'01.12.2022.

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.

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 2022 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

Ogni categoria aveva quesiti classificati in tre livelli di difficoltà: facile, medio e difficile. 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. Questi quesiti presentavano livelli di difficoltà diversi nei vari gruppi di età.

Alcuni quesiti sono indicati come «bonus» per determinate categorie di età: non contano nel totale dei punti, ma vengono utilizzati come spareggio per punteggi identici in caso di qualificazione agli eventuali turni successivi.

Per ulteriori informazioni:

SVIA-SSIE-SSII Società Svizzera per l'Informatica nell'Insegnamento
Castoro Informatico
Masiar Babazadeh

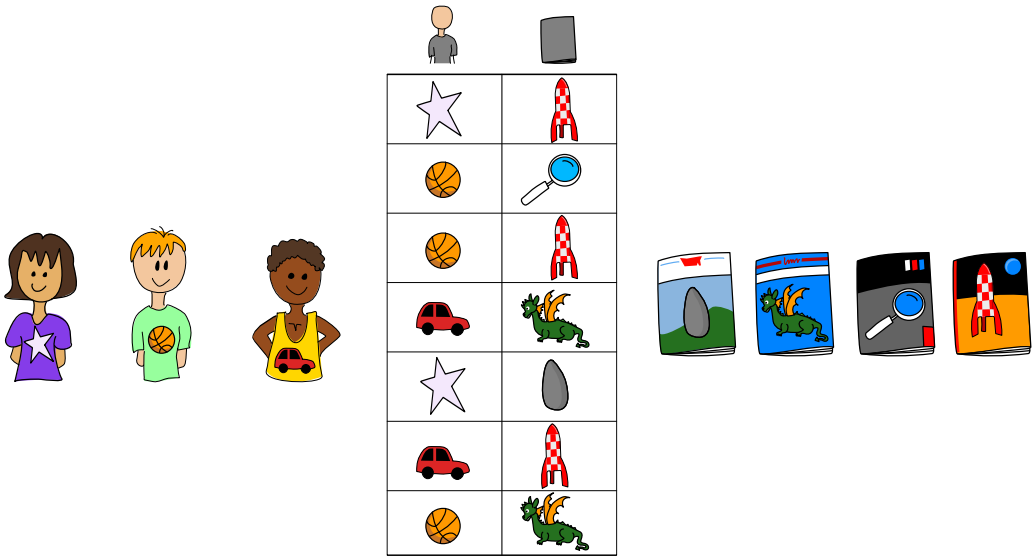
`https://www.castoro-informatico.ch/it/kontaktieren/`
`https://www.castoro-informatico.ch/`

Indice

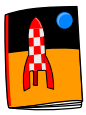
1. Biblioteca

Dei bambini prendono in prestito alcuni libri dalla biblioteca. La biblioteca scrive in una tabella chi ha preso in prestito quale libro.

Quale libro hanno preso in prestito più spesso i bambini?

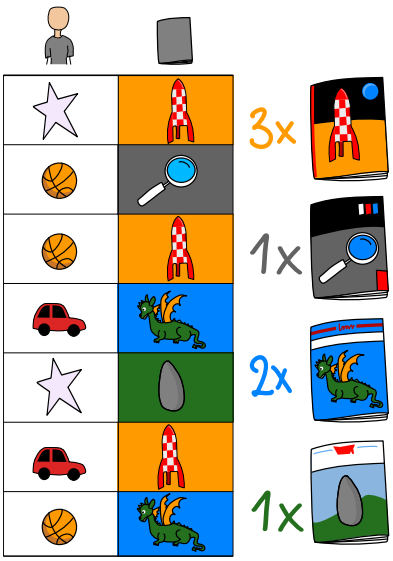


Soluzione

La risposta corretta è C): 

Quanto segue è corretto:

- Tre bambini hanno preso in prestito il libro con il razzo.
- Un bambino ha preso in prestito il libro con la lente d'ingrandimento.
- Due bambini hanno preso in prestito il libro con il drago.
- Un bambino ha preso in prestito il libro con il menhir.

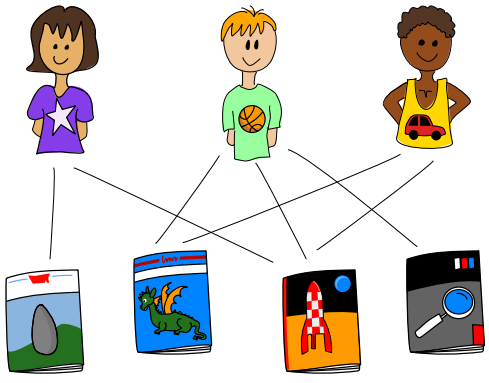


Quindi è il libro con il razzo che è stato preso in prestito più spesso.

Questa è l'informatica!

È fantastico che ai bambini del Concorso Castoro Informatica piaccia leggere libri!

Ma abbiamo davvero bisogno di un tavolo con bambini e libri per rappresentare i desideri dei bambini? Non funzionerebbe anche disegnare semplicemente delle linee?



Sarebbe più facile per gli esseri umani, ma non per i computer. I computer non sono bravi a leggere le righe. Ma sono molto bravi a lavorare con i tavoli. Se vogliamo che i computer ci aiutino a capire, ad esempio, quale bambino ha preso in prestito un libro o quale persona possiede un conto in banca, di solito è una buona idea mostrarlo in tabelle.

Le tabelle sono state introdotte 4000 anni fa a Babilonia per memorizzare informazioni sulle *relazioni*. Questa capacità di memorizzare relazioni rende le tabelle un importante concetto di base dei database relazionali.

Le tabelle rappresentano le relazioni tra le cose (o le persone). Le relazioni determinano il modo in cui rappresentare le informazioni nelle tabelle. Ad esempio, se la regola fosse che ogni bambino può prendere in prestito un solo libro, la tabella avrebbe una sola riga per ogni bambino. Nel nostro esempio della biblioteca, è giusto che i bambini possano prendere in prestito diversi libri, e che possano persino prendere in prestito gli stessi libri degli altri bambini. In questo caso, abbiamo bisogno di una tabella speciale che colleghi i bambini e i libri e che possa elencare lo stesso bambino più volte e anche lo stesso libro più volte.

Il tavolo di circolazione è pratico. Se manca un libro, ad esempio, il bibliotecario può verificare se è stato prestato. La tabella di circolazione ha due colonne e molte righe. Nella prima colonna viene inserito il bambino che prende in prestito un libro e nella seconda colonna il libro. In questo modo, alla domanda su quale libro sia stato prestato di più si può rispondere semplicemente contando il numero nella seconda colonna.

Questo compito potrebbe essere svolto anche da un computer. Se si tratta di una grande biblioteca con molte migliaia di libri, non c'è altro modo! In una biblioteca così grande, non solo il tavolo di circolazione verrebbe mantenuto. Ci sarà anche un archivio clienti (tabella clienti) in cui saranno memorizzate tutte le informazioni sui clienti, come nome, indirizzo e numero di telefono e un indice dei libri (tabella libri) con informazioni sui libri, come autore e titolo. In questo modo, la tabella di circolazione rimane snella perché contiene solo le relazioni (cioè chi ha preso in prestito quale libro) tra i clienti e i libri.

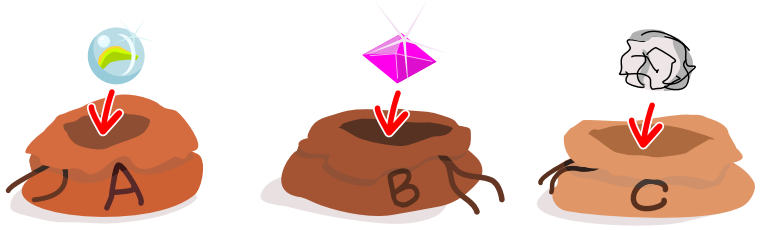
In informatica, tali tabelle sono chiamate *basi di dati* relazionali.

Parole chiave e siti web

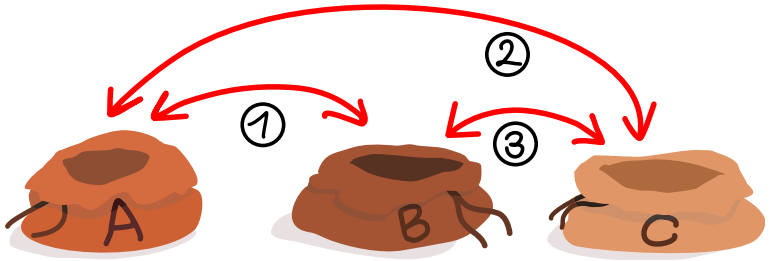
- Base di dati («*database*»): `https://it.wikipedia.org/wiki/Base_di_dati`

2. Permutazioni

Lila mette una biglia nel sacchetto A, una pietra preziosa nel sacchetto B e un pezzo di carta nel sacchetto C.



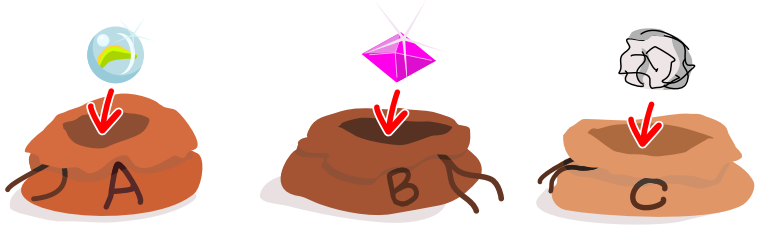
Poi scambia il contenuto del sacchetto A e del sacchetto B, quindi il contenuto di A e C e infine scambia il contenuto di B e C.



Dove sono i tre oggetti?

Soluzione

All'inizio abbiamo questa disposizione delle 3 cose nei sacchetti:



Lila scambia le cose 3 volte. Dopo il primo scambio (A-B) i sachetti hanno questa disposizione:

Dopo il secondo scambio (A-C):

Dopo il terzo e ultimo scambio (B-C):

Pertanto, alla fine, il pezzo di carta si trova in A, la pietra preziosa in B e la biglia in C. Questo risultato si sarebbe potuto ottenere anche in modo più semplice, cioè con un unico scambio dei contenuti di A e C.

Questa è l'informatica!

Qui si tratta di sequenzialità di cose. Questa sequenza di cose viene anche chiamata «disposizione». Un ordine diverso rappresenta una disposizione diversa. Uno scambio cambia l'ordine delle cose e quindi porta a una disposizione diversa. Nel nostro compito, abbiamo la disposizione biglia-pietra-carta all'inizio e la disposizione carta-pietra-biglia dopo le 3 permutazioni.

Una domanda interessante è quante disposizioni diverse possono avere 3 cose. Possiamo semplificare un po' la situazione per ora e fare tutte le disposizioni, eseguendo solo un'azione. Per le altre due cose ci sono solo due disposizioni. Se la biglia si trova al primo posto, le due disposizioni sono:

Biglia-pietra-carta
Biglia-carta-pietra

Pertanto, anche per gli altri due oggetti esistono solo due disposizioni diverse. Quindi ci sono altre 4 disposizioni delle 3 cose:

Pietra-biglia-carta
Pietra-carta-biglia
Carta-biglia-pietra
Carta-pietra-biglia

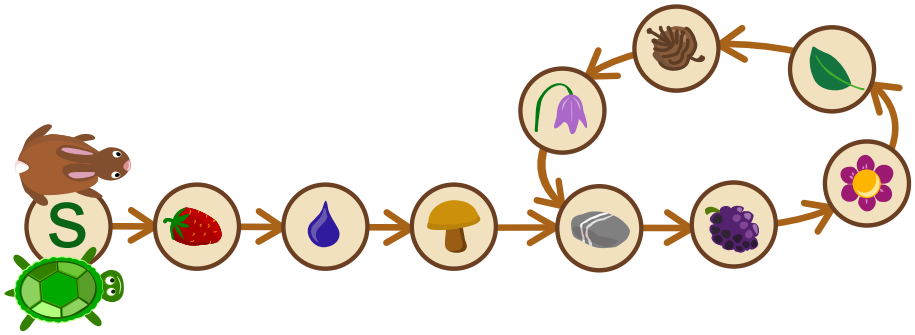
È inoltre interessante notare che è possibile creare qualsiasi disposizione solo con le permutazioni. Ciò richiede al massimo $n - 1$ permutazioni per n cose.

Parole chiave e siti web

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

3. Tartaruga e lepre

Una tartaruga e una lepre stanno facendo una gara. Utilizzano questa pista.



Iniziano nello stesso momento sul campo di partenza. Vanno di campo in campo e seguono le frecce.

In un minuto, ...

- ... la tartaruga avanza di un campo.
- ... la lepre avanza di due campi.

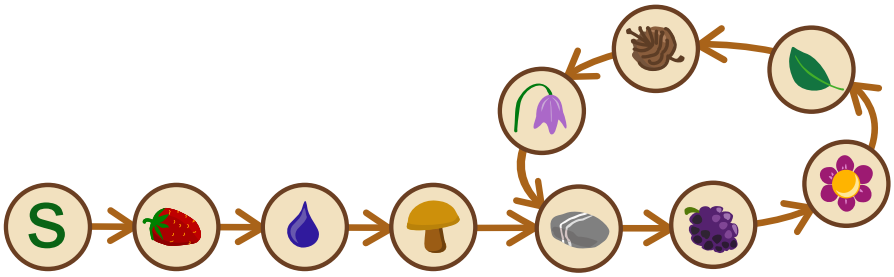
In quale campo la tartaruga e la lepre si incontrano per la prima volta dopo la partenza?

Soluzione

Tartaruga e lepre si incontrano per la prima volta sul campo . Puoi seguirlo facilmente con due dita.

La seguente tabella mostra i campi della tartaruga e della lepre per ogni minuto:

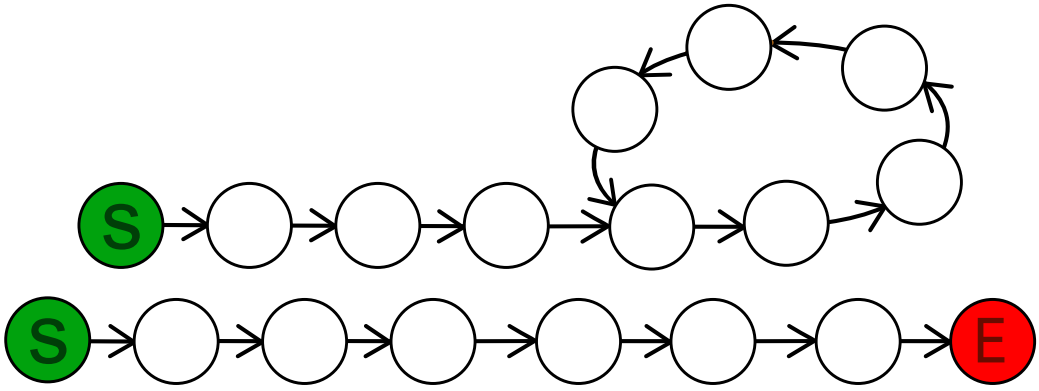
Minuti dopo l'inizio	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	...
	S														...
	S														...



Questa è l'informatica!

In questo compito, la gara si svolge su una pista speciale. È composta da singoli campi e da frecce che indicano il campo successivo. La particolarità è che la pista termina con un cerchio in cui i corridori possono correre all'infinito. La tartaruga e la lepre possono incontrarsi in questo compito solo perché questi 6 campi formano un cerchio o un *ciclo*.

In informatica, una traccia di corsa come quella descritta nel compito verrebbe chiamata *lista*. Un cerchio di campi che si riferiscono l'uno all'altro come nel compito si chiama *ciclo*. In una lista, ogni *vertice* si riferisce al massimo a un altro vertice. Esistono liste con un ciclo, come in questo compito, e liste senza ciclo.



Se una lista non ha cicli, allora la lista consiste in una catena lineare di vertici. Allora deve esistere anche un campo finale dal quale non parte alcuna freccia. Il famoso informatico Robert W. Floyd

(1936−2001) ha ideato un algoritmo in grado di distinguere facilmente se una lista ha un ciclo o consiste in una catena lineare. In modo simile al nostro compito, lascia che la lepre e la tartaruga inizino a correre nel campo di partenza. Se la tartaruga e la lepre arrivano nello stesso campo nello stesso momento, c'è un ciclo. Nel momento in cui la lepre raggiunge il campo finale o il campo precedente, non c'è più nessun ciclo e l'algoritmo è terminato.

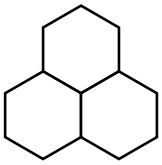
Parole chiave e siti web

- Lista concatenata: `https://it.wikipedia.org/wiki/Lista_concatenata`
- Vertice: `https://it.wikipedia.org/wiki/Vertice_(teoria_dei_grafi)`
- Robert W. Floyd: `https://it.wikipedia.org/wiki/Robert_Floyd`
- Algoritmo: `https://it.wikipedia.org/wiki/Algoritmo`

4. Piramide colorata

Sami mette insieme gli esagoni bianchi. Poi li dipinge con tre colori diversi.

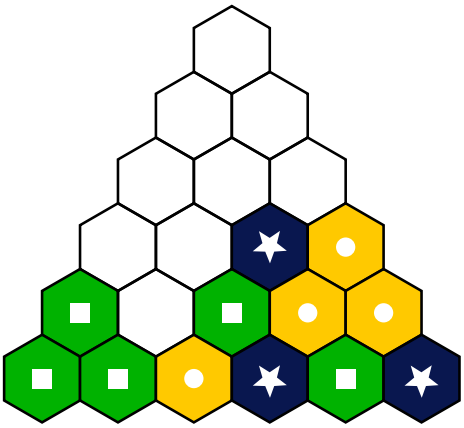
Sami vuole che quando tre esagoni si trovano esattamente insieme in questo modo (due in basso e uno in alto al centro), devono finire ...



- ... tutti e tre dello stesso colore o ...
- ... tutti e tre di colori diversi.

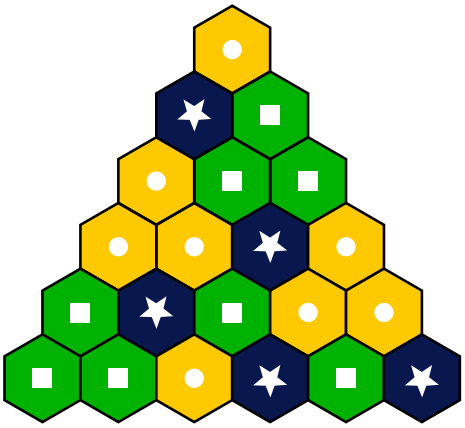
Sami ha messo insieme molti esagoni e ne ha già colorati alcuni.

Colora tutti gli esagoni rimanenti come piace a Sami.



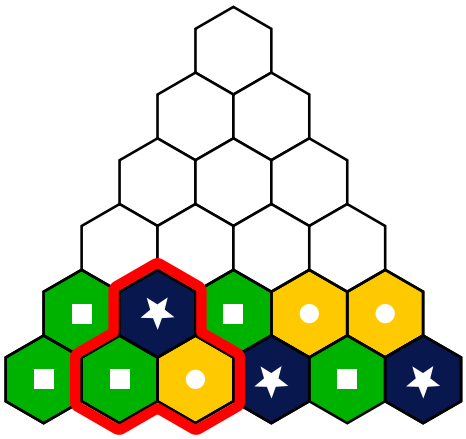
Soluzione

Questa è la soluzione giusta:

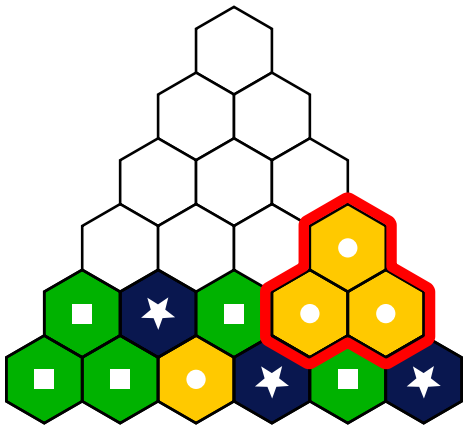


Non appena vengono colorati due esagoni vicini nella piramide di esagoni, il colore dell'esagono superiore viene fissato:

Se entrambi hanno colori diversi, l'esagono sopra riceve il terzo colore. Ad esempio, l'esagono bianco più basso è dipinto di blu



Se entrambi hanno lo stesso colore, anche l'esagono sopra di esso è dipinto di quel colore. Quindi anche l'esagono sopra i due gialli è dipinto di giallo.



In questo modo potrai colorare gli esagoni rimanenti in fila, dal basso verso l'alto, uno dopo l'altro, proprio come piace a Sami.

Questa è l'informatica!

Come si risolve questo compito? Quando colori un esagono, esegui un'azione. Per scegliere l'azione giusta (con il colore giusto), devi guardare gli esagoni sottostanti e verificare quale *condizione* soddisfano: se hanno lo stesso colore o colori diversi. Questo controllo, con le azioni successive, viene *ripetuto*, cioè per ogni esagono ancora bianco che si trova sopra due esagoni già colorati.

Azioni, condizioni, ripetizioni: questi sono gli elementi di base di qualsiasi *algoritmo*, cioè una procedura descritta con precisione che può essere realizzata come programma per un computer. Quindi, per risolvere questo compito, hai inventato un algoritmo. Questo è uno dei compiti più importanti degli informatici: inventare algoritmi o utilizzare algoritmi già inventati e convertirli in programmi per computer al fine di risolvere compiti e problemi di elaborazione automatica delle informazioni.

Parole chiave e siti web

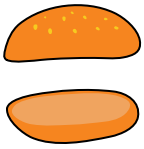
- Algoritmo: `https://it.wikipedia.org/wiki/Algoritmo`
- Selezione: `https://it.wikipedia.org/wiki/Selezione_(informatica)`
- Ciclo: `https://it.wikipedia.org/wiki/Struttura_di_controllo#Ciclo_for`

5. Ricetta hamburger

La castorina Jess prepara gli hamburger. Per farli segue tre regole:

1. la salsa è direttamente sulla carne.
2. la carne e il formaggio sono sotto i cetrioli, la lattuga e le cipolle.
3. le cipolle non toccano il panino.

Ingredienti dell'hamburger:

Panino	**Carne**	**Salsa**	**Cetrioli**	**Lattuga**	**Cipolle**	**Formaggio**
						

Quale hamburger è composto secondo le tre regole?

A) 

B) 

C) 

D) 

Soluzione

La risposta corretta è D. 

Per trovare la soluzione, è necessario controllare ogni hamburger per vedere se è messo insieme in modo da seguire tutte e tre le regole.

A) Questo hamburger segue le regole 1 e 2. Ma le cipolle toccano il panino, quindi non rispetta la regola 3.

B) Questo hamburger segue la regola 1. Ma la lattuga è sotto la carne e il formaggio, quindi la regola 2 non è stata rispettata.

C) Questo hamburger segue la regola 2 perché la carne e il formaggio sono sotto i cetrioli, la lattuga e le cipolle. Inoltre, questo hamburger di castoro segue la regola 3 perché le cipolle non toccano il panino. Tuttavia, la salsa non viene versata direttamente sulla carne. Pertanto, la regola 1 non è stata rispettata.

D) Questo hamburger soddisfa tutte le regole.

Questa è l'informatica!

Gli hamburger in questo compito sono realizzati secondo tre regole. Per ogni hamburger che prepara, la castorina Jess deve seguire ognuna delle tre regole. Se non rispetta una sola delle regole, l'hamburger non è giusto. Ognuna delle tre regole è una condizione che deve essere soddisfatta affinché ogni hamburger sia giusto.

In informatica, il controllo dei vincoli è spesso usato per scoprire se una soluzione segue tutte le regole date. In questo controllo, si collegano tutte le regole (condizioni) con l'operatore *E*. Ciò significa che tutte le regole (condizioni) devono essere soddisfatte contemporaneamente.

Verificare se una determinata soluzione soddisfa tutti i vincoli è un compito fondamentalmente diverso dal trovare una possibile soluzione. Si tratta del cosiddetto *problema di soddisfacimento di vincoli*. Nella maggior parte dei casi, è molto più difficile trovare una soluzione che soddisfi tutti i vincoli che verificare se una soluzione soddisfa tutti i vincoli. Questo vale anche per un computer.

Parole chiave e siti web

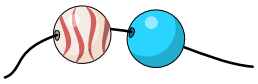
- Programmazione a vincoli: `https://it.wikipedia.org/wiki/Programmazione_a_vincoli`
- Problema di soddisfacimento di vincoli: `https://it.wikipedia.org/wiki/Problema_di_soddisfacimento_di_vincoli`
- Congiunzione logica: `https://it.wikipedia.org/wiki/Congiunzione_logica`

- NP: `https://it.wikipedia.org/wiki/NP_(complessità)`

6. Collana da marinaio

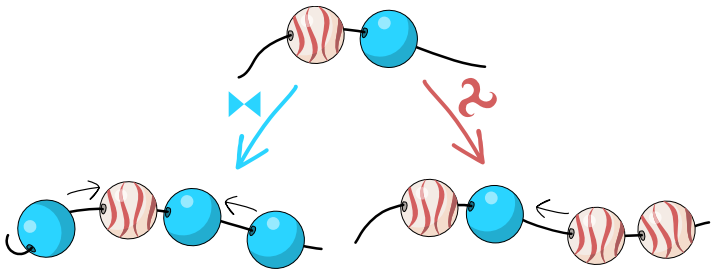
Ecco il manuale per la collana da marinaio di Monika con perline a onda bianche e rosse e perline blu semplici.

Inizia sempre con una perlina a onda e una perlina blu in questo ordine:



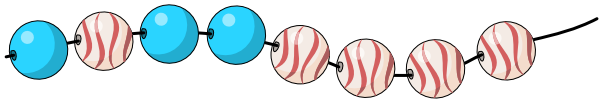
Poi puoi estendere la collana da marinaio,

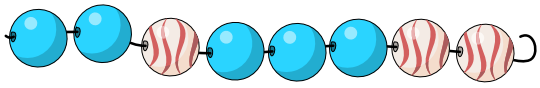
- aggiungendo una perlina blu a ciascuna estremità della stringa (⋈)
- oppure aggiungendo due perline a onda all'estremità destra della stringa (2)

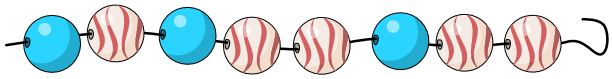


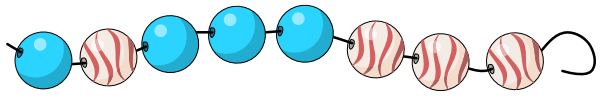
Puoi eseguire queste azioni più volte per creare collane sempre più lunghe.

*Quale delle seguenti collane **non** è una delle collane da marinaio di Monika?*

A) 

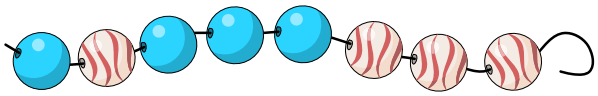
B) 

C) 

D) 

Soluzione

D è la risposta corretta.



Puoi risolvere il problema in diversi modi.

Per esempio, trovando prima le due perline iniziali di ogni collana ed eseguendo poi una serie di azioni ⋈ e ⋈.

- Nella collana A, puoi iniziare con la seconda e la terza perlina e poi eseguire le azioni ⋈ - ∿ - ∿.
- Per la collana B, puoi iniziare con la terza e la quarta perlina e poi eseguire le azioni ⋈ - ⋈ - ∿.
- Per la collana C, puoi iniziare con la seconda e la terza perlina e poi eseguire le azioni ∿ - ⋈ - ∿.
- Tuttavia, se guardi la collana D, la seconda e la terza perlina devono essere l'inizio. L'azione B può essere eseguita una volta, ma dopo di essa non ci sono altre azioni per ottenere il resto della catena.

Questo approccio non funziona bene se la collana è molto lunga e ha molte possibili perline di partenza. In questo caso, un approccio decostruttivo potrebbe funzionare meglio. In questo caso rimuovi ripetutamente le perline eseguendo l'azione B o l'azione W al contrario, finché non rimangono solo due perline.

Una terza strategia si avvale della *parità*. Secondo le istruzioni della collana del marinaio, c'è sempre un numero dispari di perline blu e un numero dispari di perline ondulate rosse e bianche («parità dispari»). Capisci perché?

La collana D ha un numero pari di entrambi i tipi di perline e quindi non può essere una delle collane da marinaio di Monika.

Questa è l'informatica!

In questa attività puoi infilare le perline solo alle estremità della collana. Non puoi inserire una perlina al centro. Inoltre, non puoi rimuovere una perlina dal centro senza aver prima sfilato le perline dall'estremità della collana.

Questo tipo di struttura di memoria, in cui è possibile aggiungere e rimuovere facilmente elementi alle estremità ma non al centro, è chiamata in informatica *coda a doppia estremità* o *coda deque* (deque si pronuncia come «deck»).

Le code deque possono essere utilizzate per memorizzare la cronologia del browser, per programmare i lavori di stampa e anche per verificare la validità delle espressioni matematiche. Ad esempio, il

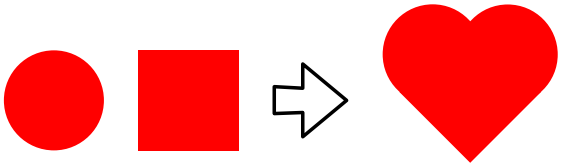
controllo della corrispondenza delle parentesi può essere fatto più o meno nello stesso modo in cui si controlla se una collana è una delle collane di marinaio di Monika.

Parole chiave e siti web

- double-ended queue: `https://it.wikipedia.org/wiki/Deque`

7. Cuore composto

Tina ha due forme: un cerchio e un quadrato. Li trasforma in un cuore.



Per farlo, utilizza queste tre trasformazioni:

- *gira*: gira una forma quanto si vuole.
- *sposta*: sposta una forma quanto si vuole.
- *raddoppia*: raddoppiare una forma in modo che entrambe rimangano nello stesso posto.

Cosa ha fatto e in che ordine?

A) *raddoppia* il cerchio, *gira* il quadrato, *sposta* il cerchio, *sposta* il cerchio

B) *raddoppia* quadrato, *gira* quadrato, *sposta* quadrato, *sposta* cerchio

C) *raddoppia* cerchio, *gira* cerchio, *sposta* cerchio, *sposta* quadrato

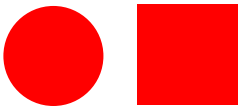
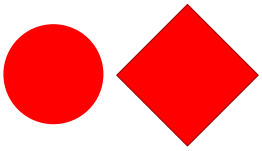
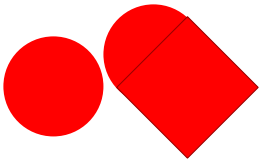
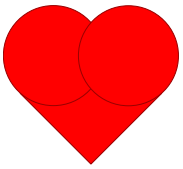
D) *sposta* cerchio, *sposta* cerchio, *raddoppia* cerchio, *sposta* quadrato

Soluzione

Se si osserva attentamente il cuore, si noterà che è composto da due cerchi e da un quadrato ruotato di 1/8. Quindi è necessario un «raddoppia cerchio» nelle trasformazioni, in modo da avere due cerchi, e un «gira quadrato», in modo da poter girare il quadrato di 1/8. Questo elimina le risposte B), C) e D), perché:

- Nella risposta B) viene raddoppiato un quadrato e non un cerchio.
- Nella risposta C) viene ruotato un cerchio, ma non il quadrato.
- Nella risposta D) nessuna forma viene ruotata.

Ma la risposta A) è corretta? Le forme devono ancora essere spostate! Le trasformazioni seguenti sono date:

- Questo: 
- diventa  raddoppiando il cerchio
- diventa  girando il quadrato
- diventa  spostando il cerchio
- diventa  spostando il cerchio

Pertanto, la risposta A) raddoppia il cerchio, gira il quadrato, sposta il cerchio, sposta il cerchio è corretta.

Questa è l'informatica!

Nei programmi di modifica delle immagini è possibile effettuare molte trasformazioni diverse con un'immagine. In questo compito, si tratta di trasformazioni come la rotazione, lo spostamento o il raddoppio. Ma questo da solo non basta: bisogna anche dire al computer, per esempio, di quanto ruotare una forma o dove spostarla.

Si potrebbe descrivere il modo in cui disegnare un cuore da un cerchio e da un quadrato in un testo più lungo. In informatica, tuttavia, è meglio utilizzare il minor numero possibile di trasformazioni di base, che poi si ripetono o si eseguono in modo diverso. Si parla di generalizzazione quando

si sviluppano soluzioni generali a partire da esempi specifici. Tali comandi potrebbero essere, ad esempio:

- Ruotare una forma: ruotare la forma, fino a che punto.
- Spostare una forma: spostare la forma, dove
- Raddoppiare una forma: doppia forma

Il programma di modifica delle immagini di Tina può sembrare insolito: invece di salvare l'immagine come *pixel* come nelle foto, viene salvata una descrizione della forma (ad esempio «cerchio, raggio 2 cm, colore di riempimento rosso»). In questo modo è possibile sovrapporre due forme, come i due cerchi, e spostare successivamente una di esse senza che quella inferiore venga sovrascritta. Questo tipo di grafica si chiama *grafica vettoriale*. Vengono spesso utilizzati quando si devono disegnare forme astratte di alta qualità. Gli altri elementi grafici utilizzano la *grafica a pixel* e spesso sono foto o disegni fotorealistici.

Parole chiave e siti web

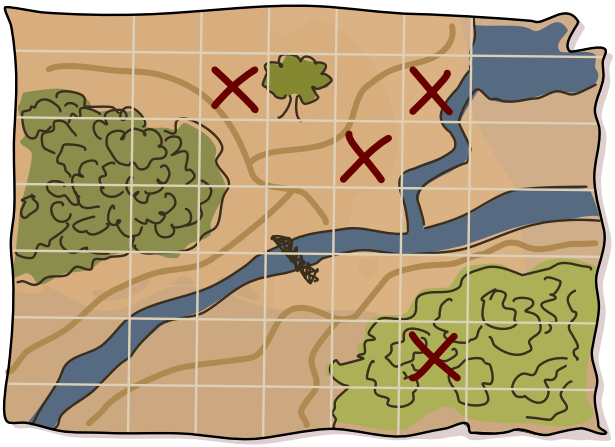
- Pixel: `https://it.wikipedia.org/wiki/Pixel`
- Grafica raster: `https://it.wikipedia.org/wiki/Grafica_raster`
- Grafica vettoriale: `https://it.wikipedia.org/wiki/Grafica_vettoriale`

8. Mappa del tesoro

Il castoro Bilbo ha due buoni nascondigli per il suo cibo. Su una mappa segna i due campi dove si trovano i nascondigli con ✕. Ma cosa succede se altri castori trovano la mappa e quindi i nascondigli?

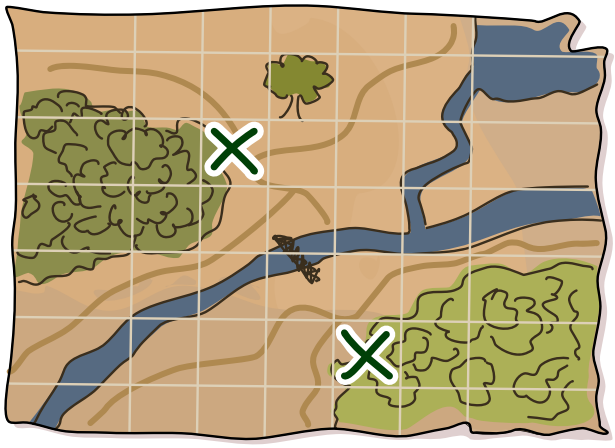
Per confondere le cose, Bilbo segna altri campi con ✕. Lo fa in modo che in ogni riga e colonna della mappa sia segnato un numero pari di caselle. Poi rimuove i due ✕ dai campi con i suoi nascondigli. Di seguito è possibile vedere il risultato.

In quali campi si trovano i nascondigli di Bilbo?

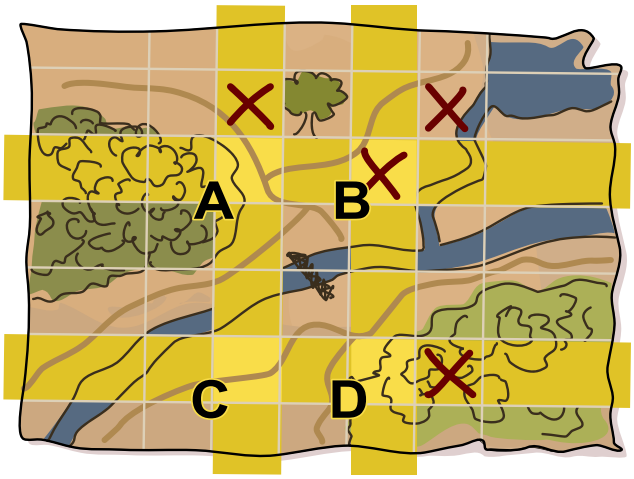


Soluzione

Ecco i due nascondigli:



Per trovarli, osserviamo la mappa originale e notiamo che ci sono due righe e due colonne in cui il numero di ✕ non è pari: le righe 3 e 6 e le colonne 3 e 5.



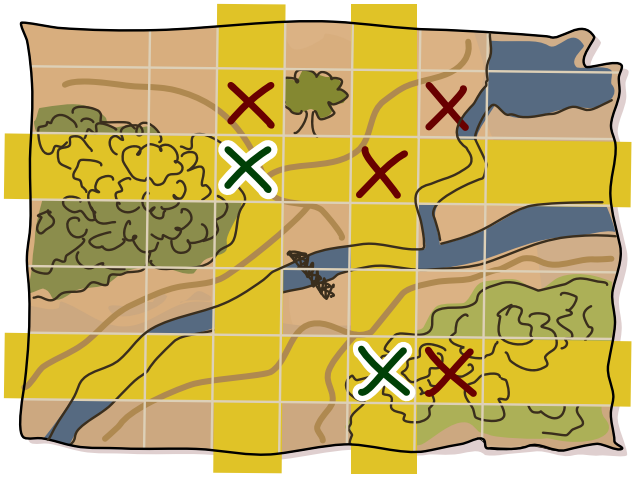
Dopo tutto, i ✕ che segnalano i nascondigli sono stati rimossi. Sappiamo che deve esserci un numero pari di ✕ in tutte le righe e le colonne dopo che le ✕ cancellate sono state reinserite.

Le righe e le colonne interessate si sovrappongono e hanno quattro campi comuni (A, B, C e D). Questi «campi intersecanti» sono di particolare interesse per noi. Se contrassegniamo i campi al di fuori di un campo di intersezione con ✕, possiamo ottenere un numero ✕ pari in una colonna, mentre il numero nella rispettiva riga diventa dispari e viceversa. Pertanto, le ✕ dei due nascondigli devono trovarsi sui campi di intersezione.

Il campo di intersezione B è già contrassegnato da una ✕: non può essere un nascondiglio perché sappiamo che Bilbo ha cancellato la ✕ dei nascondigli.

Quindi, per restituire un numero pari di ✕ nella riga 3, dobbiamo contrassegnare l'intersezione A con una ✕. Lì c'è un nascondiglio. L'altro nascondiglio non può trovarsi nell'intersezione C, perché

allora ci sarebbero tre ✕ in quella colonna. Quindi l'altro nascondiglio si trova all'intersezione D. Ecco la mappa prima che Bilbo cancellasse le ✕, con un numero pari di ✕ in ogni riga e colonna:



Questa è l'informatica!

Bilbo utilizza un trucco spesso usato in informatica: i *bit di parità*. Fanno parte di un insieme di tecniche di *rilevazione e correzione d'errore*. L'idea è che ogni volta che memorizziamo o trasmettiamo dati come una serie di *bits* (che possono essere 0 o 1), aggiungiamo bit supplementari per aiutarci a rilevare se si sono prodotti errori di trasmissione o di memorizzazione, in genere quando un bit è stato distorto, cioè quando un bit è stato inviato come 1 e ricevuto erroneamente come 0, o viceversa.

Ad esempio, se utilizziamo un semplice codice di rilevamento degli errori, viene aggiunto un bit di parità in modo che il numero di uno sia sempre pari. 0110101 viene aggiunto uno 0 per diventare 01101010 (il numero di bit a «1» rimane pari). Se il secondo bit è stato invertito e il messaggio viene ora inviato a 00101010, il messaggio ricevuto non soddisfa il requisito di parità (tre bit sono bit a «1»). È importante notare che questo metodo non è in grado di rilevare un problema se più di un bit è in errore.

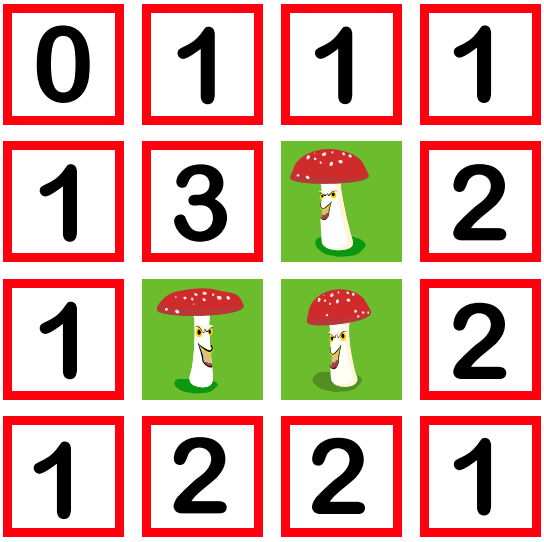
Parole chiave e siti web

- Bit: `https://it.wikipedia.org/wiki/Bit`
- Bit di parità: `https://it.wikipedia.org/wiki/Bit_di_parità`
- Rilevazione e correzione d'errore: `https://it.wikipedia.org/wiki/Rilevazione_e_correzione_d'errore`

9. Attenzione ai funghi

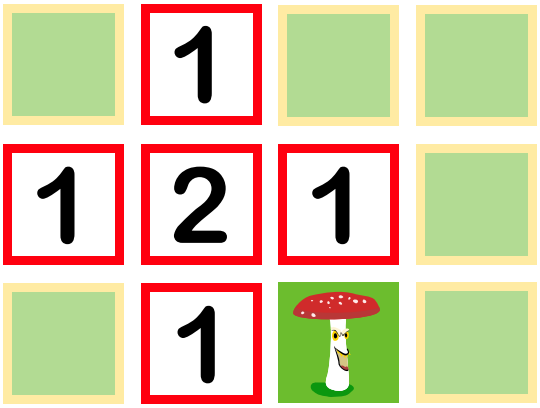
Nel gioco «Attenzione ai funghi», all'inizio è visibile esattamente un fungo. Tutte le altre caselle del tabellone sono coperte. Se si scopre un campo, appare un altro fungo o il numero di funghi sui campi vicini. Se si scoprono tutte le caselle in cui non è nascosto alcun fungo, si vince.

Ecco un esempio di una tavola completamente scoperta:



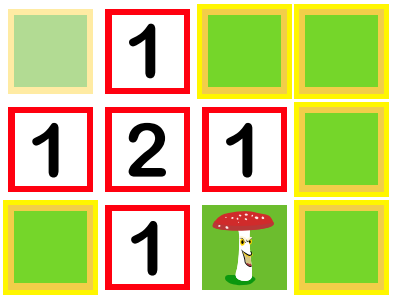
Hai iniziato una nuova partita e hai già scoperto alcune caselle.

Su quale dei campi rimanenti non c'è sicuramente un fungo?

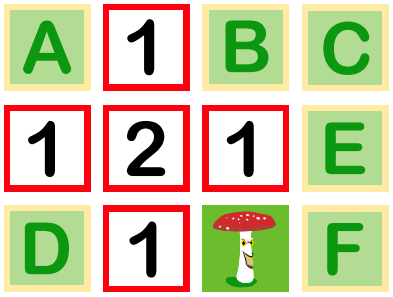


Soluzione

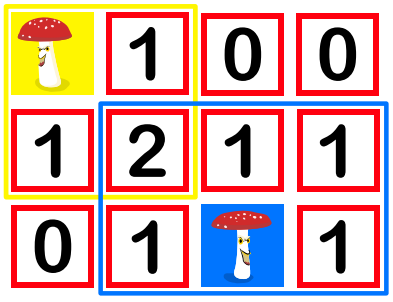
Questa è la soluzione:



Per spiegare la risposta corretta, etichettiamo i quadrati coperti con delle lettere. Inoltre, diciamo che un numero N su un campo è «esaurito» se c'è già un fungo scoperto su ciascuno degli N campi vicini di questo numero; non ci possono quindi essere altri funghi su altri campi vicini.



- Non c'è nessun fungo sulla casella D perché il numero 1 alla sua destra è esaurito.
- Nei campi B, C, E e F non c'è nessun fungo, perché il numero 1 di questi campi, comunemente vicino, è esaurito.
- C'è un fungo sul campo A, perché altrimenti i numeri 1, 2 e 1 non indicherebbero correttamente il numero di funghi sui campi vicini.



Quindi c'è un fungo nascosto nel campo A. I campi B, C, D, E e F possono essere scoperti.

Questa è l'informatica!

Come abbiamo proceduto? A volte è necessario partire da un'ipotesi per poi ragionare in modo logico. Se si trova una contraddizione, si torna indietro e si prosegue l'ipotesi seguente più plausibile. Si tratta di una ricerca «mirata» e non di tentativi ed errori.

Usando un computer come si potrebbe risolvere questo problema? Se si scopre almeno un campo con un rospo, si possono derivare delle semplici regole. Ad esempio, se il campo con il numero 1 copre già un campo vicino con un rospo scoperto, non può esserci un altro rospo lì vicino. Se queste regole sono formulate con precisione per ogni numero, un computer potrebbe eseguirle passo dopo passo come *istruzioni*. In definitiva, avremmo un *algoritmo* che sarebbe necessario eseguire per avere successo nel gioco (con almeno un rospo scoperto).

Parole chiave e siti web

- Campo minato: `https://it.wikipedia.org/wiki/Campo_minato_(videogioco)`
- Algoritmo: `https://it.wikipedia.org/wiki/Algoritmo`

10. Bulloni e dadi

Ben è alla catena di montaggio e lavora i componenti: dadi e bulloni.

Ben segue rigorosamente la seguente procedura:

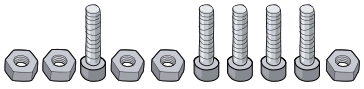
- Ben prende il componente successivo dalla catena di montaggio.
- Quando Ben ha preso un dado dalla catena di montaggio, lo mette nel secchio.
- Quando Ben ha preso un bullone dalla catena di montaggio, prende un dado dal secchio, lo avvita sul bullone e mette il pezzo finito nella scatola.

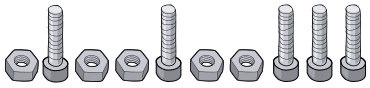
In questa procedura possono verificarsi due errori:

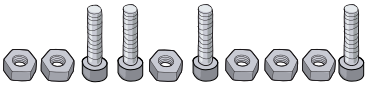
1. Ben prende un bullone dalla catena di montaggio, ma nel secchio non c'è nessun dado da avvitare.
2. Ben ha lavorato tutti i componenti della catena di montaggio, ma ci sono ancora dadi nel secchio.

Il secchio per i dadi è sufficientemente grande e vuoto all'inizio. Quale delle sequenze di dadi e bulloni può essere elaborata da Ben da sinistra a destra senza commettere errori?

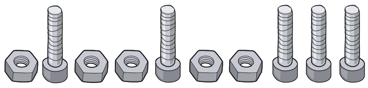
A) 

B) 

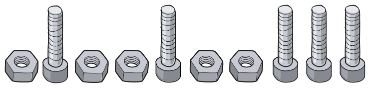
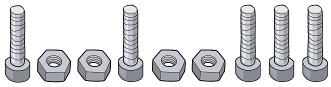
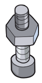
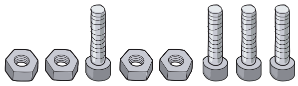
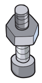
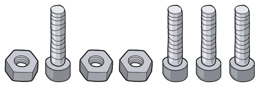
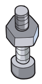
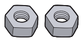
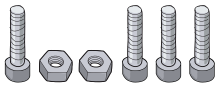
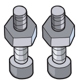
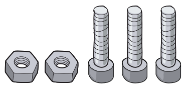
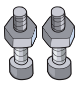
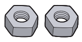
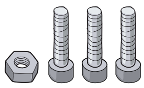
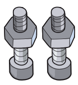
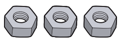
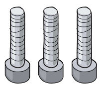
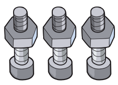
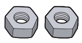
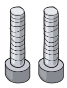
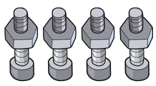
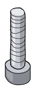
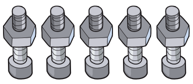
C) 

D) 

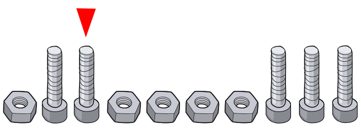
Soluzione

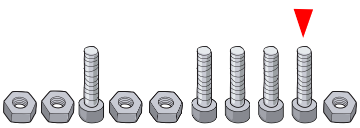
La risposta corretta è C) 

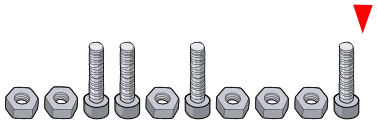
La tabella mostra lo stato della scatola per i pezzi finiti, del secchio per i dadi e della catena di montaggio.

Scatola	Secchio	Catena di montaggio
vuota	*vuoto*	
vuota		
	*vuoto*	
		
		
		
		
		
		
		
	*vuoto*	*vuota*

Perché le altre risposte sono sbagliate?

A)  porta a un errore nella posizione contrassegnata. Poi Ben ha preso un bullone, ma nel secchio non c'è più il dado.

B)  porta a un errore nella posizione contrassegnata. Finora Ben ha avvitato 4 dadi su quattro bullone. Quindi il secchio è vuoto. Ma ora ha preso un quinto bullone per il quale non ha più un dado.

D)  porta a un errore dopo l'elaborazione dell'intera sequenza. Questo perché sono stati avvitati 4 dadi su 4 bulloni e sono rimasti 2 dadi.

Questa è l'informatica!

Ben elabora i componenti che vengono consegnati uno per uno dalla catena di montaggio. Nel processo, utilizza un grande secchio per conservare temporaneamente i dadi. Una disposizione simile viene utilizzata in *informatica teorica* come modello per gli *algoritmi* in grado di risolvere una certa classe di problemi: automi a pila.

Un automa a pila elabora i dati (numeri o caratteri) che riceve in ingresso uno per uno. Ha un'unica memoria infinita, una pila. A differenza del secchio nel compito, gli elementi della pila hanno un certo ordine e si può togliere da una cantina solo l'elemento che si è messo per ultimo («last in first out», LIFO). Un automa di pila può essere utilizzato per riconoscere un *linguaggio libero dal contesto*.

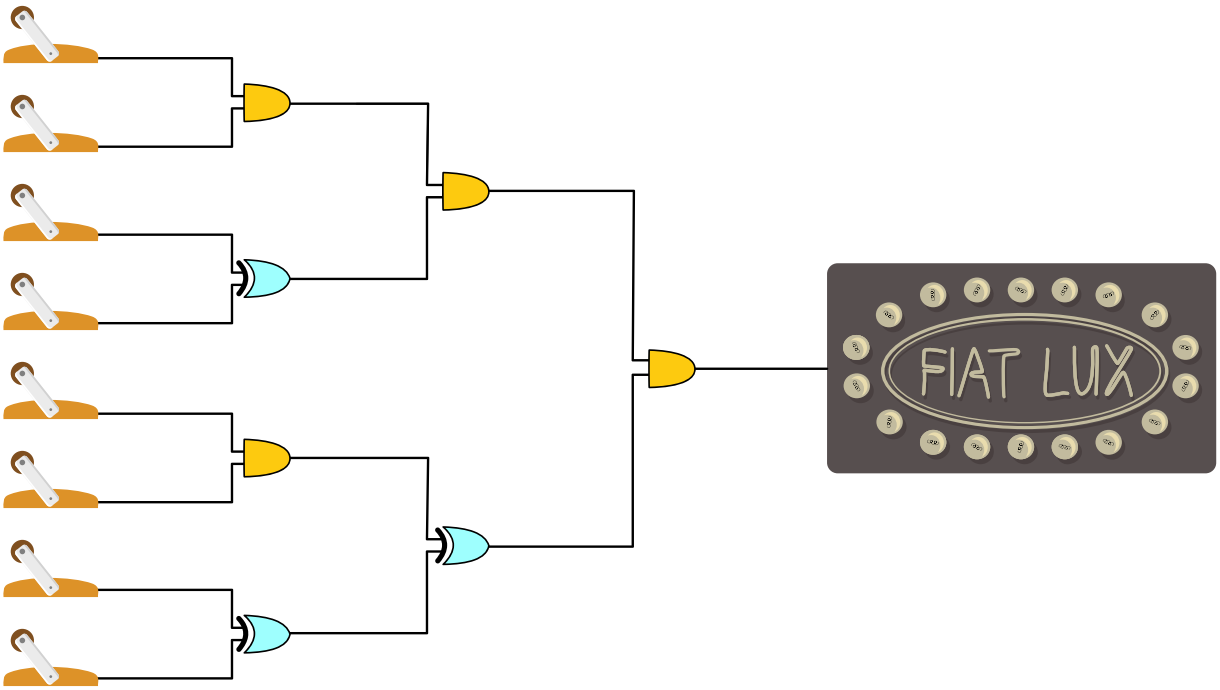
In informatica, un linguaggio è un insieme di stringhe formate secondo determinate regole. Un tipo semplice di linguaggio è il linguaggio libero dal contesto. Un esempio di linguaggio libero dal contesto è costituito da tutte le espressioni ben formate di parentesi. In un'espressione ben formata, ogni parentesi aperta viene richiusa. Le espressioni ben formate sono, ad esempio, ((( ))) e (()()). Non ben formati, invece, sono (((() e ())(. Si può pensare ai dadi e ai bulloni del compito come a delle parentesi di apertura e chiusura. Quindi Ben elabora una sequenza di componenti sulla catena di montaggio senza errori solo se rappresenta un'espressione di parentesi ben formata. La verifica delle espressioni di parentesi è un compito importante di un compilatore che traduce i testi dei programmi in programmi eseguibili. Questo perché le chiamate di funzione annidate e le espressioni aritmetiche con parentesi sono presenti nei testi dei programmi della maggior parte dei linguaggi di programmazione.

Parole chiave e siti web

- Informatica teorica: `https://it.wikipedia.org/wiki/Informatica_teorica`
- Automa a pila: `https://it.wikipedia.org/wiki/Automa_a_pila`
- Linguaggio libero dal contesto: `https://it.wikipedia.org/wiki/Linguaggio_libero_dal_contesto`

11. FIAT LUX!

Il gioco «FIAT LUX!» ha 8 interruttori che possono essere attivati o disattivati. Da questi interruttori, i fili passano attraverso alcuni componenti e infine a un'insegna al neon.

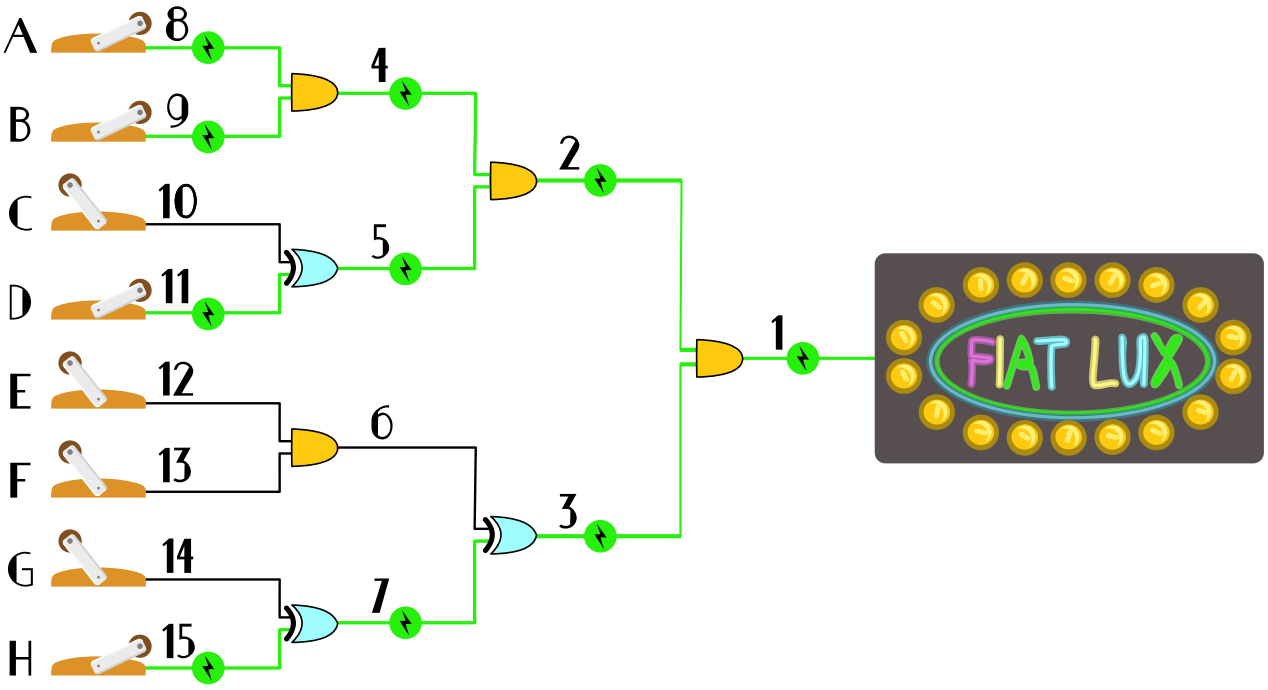


L'uscita del componente  è attiva solo quando entrambi i fili in ingresso sono attivi. L'uscita del componente  è attiva quando è attivo esattamente uno dei fili in ingresso.

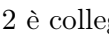
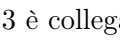
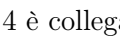
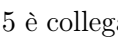
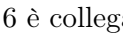
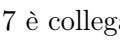
Quali interruttori  *devono essere attivati per accendere l'insegna al neon?*

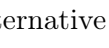
Soluzione

Una possibile soluzione è la seguente:



È possibile trovarla facilmente risolvendo il problema da dietro. Il filo collegato 1 è collegato a un componente  . Affinché l'uscita sia _on, entrambi i fili in ingresso 2 e 3 devono essere *ON*.

- Il filo 2 è collegato a un componente  . Affinché l'uscita sia *ON*, entrambi i fili 4 e 5 in ingresso devono essere *ON*.
- Il filo 3 è collegato a un componente  . Affinché l'uscita sia *ON*, esattamente uno dei due fili in ingresso deve essere *ON*, ad esempio il filo 7. Quindi il filo 6 deve essere *OFF*.
- Il filo 4 è collegato a un componente  . Affinché l'uscita sia *ON*, entrambi i fili in ingresso 8 e 9 devono essere *ON*, quindi anche gli interruttori A e B devono essere *ON*:  .
- Il filo 5 è collegato a un componente  . Affinché questo sia *ON* all'uscita, esattamente uno dei due fili in ingresso deve essere *ON*, ad esempio il filo 11. Quindi il filo 10 deve essere *OFF*. Quindi l'interruttore C deve essere *off*  e l'interruttore D deve essere *ON*  .
- Il filo 6 è collegato a un componente  . Affinché l'uscita sia *OFF*, almeno uno dei fili in ingresso 12 e 13 deve essere *OFF*, quindi anche entrambi gli interruttori E e F possono essere *OFF*:  .
- Il filo 7 è collegato a un componente  . Affinché l'uscita sia *ON*, esattamente uno dei due fili in ingresso deve essere *ON*, ad esempio il filo 15. Quindi il filo 14 deve essere *OFF*. Quindi l'interruttore G deve essere *OFF*  e l'interruttore H deve essere *ON*  .

Esistono alternative con i componenti  , perché in questo caso è possibile decidere quale dei due fili in ingresso è *ON*. Inoltre, al componente  con il filo 6 come uscita si può decidere se

nessuno o uno dei due è *on*, perché in entrambi i casi l'uscita rimane *OFF*. Affinché l'uscita del componente  con il filo 6 sia *ON*, anche entrambi gli ingressi devono essere *ON*. In questo caso, i due ingressi del componente  con il filo 7 come uscita devono essere entrambi *ON* o entrambi *OFF*, in modo che il filo 7 sia *OFF*. In questo modo si ottengono 16 diverse combinazioni possibili:

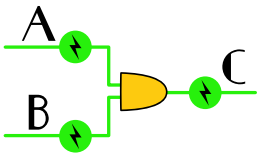
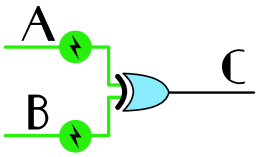
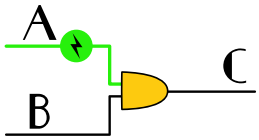
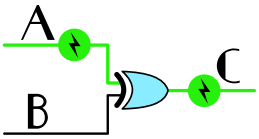
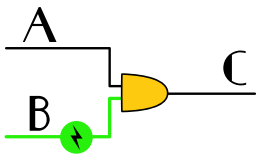
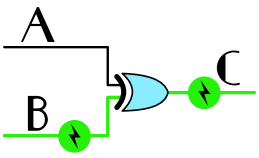
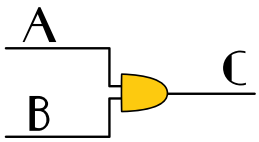
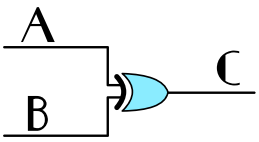
Interruttore								**Filo**	
A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**6**	**7**
sempre *ON*		**esattamente uno *ON***		**entrambi *ON*, se filo 6 è *ON*, altrimenti massimo uno *ON***		**esattament uno *ON*, se filo 7 è *ON*, altrimenti entrambi *ON* o *OFF***		**esattamente uno *ON***	
ON	ON	ON	OFF	ON	ON	ON	ON	ON	OFF
ON	ON	OFF	ON	ON	ON	ON	ON	ON	OFF
ON	ON	ON	OFF	ON	ON	OFF	OFF	ON	OFF
ON	ON	OFF	ON	ON	ON	OFF	OFF	ON	OFF
ON	ON	ON	OFF	ON	OFF	ON	OFF	OFF	ON
ON	ON	OFF	ON	ON	OFF	ON	OFF	OFF	ON
ON	ON	ON	OFF	ON	OFF	OFF	ON	OFF	ON
ON	ON	OFF	ON	ON	OFF	OFF	ON	OFF	ON
ON	ON	ON	OFF	OFF	ON	ON	OFF	OFF	ON
ON	ON	OFF	ON	OFF	ON	ON	OFF	OFF	ON
ON	ON	ON	OFF	OFF	ON	OFF	ON	OFF	ON
ON	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON
ON	ON	ON	OFF	OFF	OFF	ON	OFF	OFF	ON
ON	ON	OFF	ON	OFF	OFF	ON	OFF	OFF	ON
ON	ON	ON	OFF	OFF	OFF	OFF	ON	OFF	ON
ON	ON	OFF	ON	OFF	OFF	OFF	ON	OFF	ON

Questa è l'informatica!

La corrente può passare o meno attraverso i fili di questo compito, quindi gli interruttori sono accesi o spenti. In informatica, tali stati rappresentano il valore di una *variabile booleana*. Questi sono spesso chiamati *vero* o *falso*, rispettivamente *1* o *0*.

I computer di oggi funzionano di solito solo con questi due stati. Uno dei motivi è che nel nucleo del computer sono incorporati miliardi di *transistor*, i cui ingressi e uscite sono anch'essi solo accesi o spenti.

Si possono quindi costruire *reti logici* a partire da diversi transistor. In questo compito sono presenti due reti di questo tipo: il componente  è una *porta AND* la cui uscita è attiva solo quando entrambi gli ingressi sono attivi. Il componente  è una *porta XOR* la cui uscita è attiva quando è attivo esattamente uno dei due ingressi. Si può anche scrivere questo come una *tabella della verità*:

Ingressi		Porta AND		Porta XOR	
Ingresso A	**Ingresso B**	**Immagine**	**Uscita C**	**Immagine**	**Uscita C**
ON	ON		ON		OFF
ON	OFF		OFF		ON
OFF	ON		OFF		ON
OFF	OFF		OFF		OFF

Altre porte comune sono la *porta OR*, la cui uscita è attiva quando almeno uno dei due ingressi è attivo, e l'*l'invertitore*, la cui uscita è attiva esattamente quando l'ingresso non è attivo. Spesso si utilizza una combinazione di una porta AND e di un invertitore, che può essere realizzata con un numero molto ridotto di transistor. Le tabelle di verità sono:

Ingresso A	Ingresso B	Uscita porta OR	Uscita invertitore
ON	ON	ON	OFF
ON	OFF	ON	ON
OFF	ON	ON	ON
OFF	OFF	OFF	ON

Ingresso	Uscita invertitore
ON	OFF
OFF	ON

Grazie ad abili combinazioni di *porte logiche*, un computer può eseguire calcoli complicati in modo molto rapido.

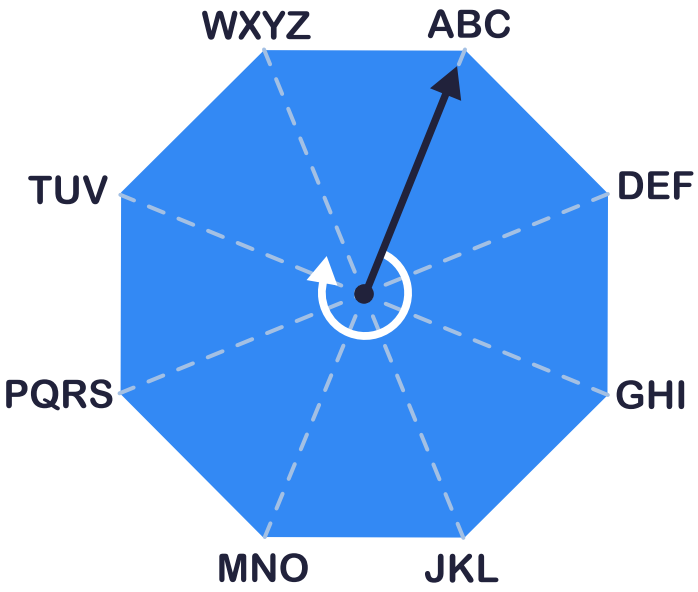
A un livello superiore, le porte logiche sono utilizzate anche nella programmazione: se l'esecuzione di una parte di un programma si basa su diverse condizioni, queste condizioni possono essere combinate con l'aiuto di operatori logici che funzionano esattamente nello stesso modo. Questo si riscontra anche nei programmi informatici. A volte un programma deve prendere «decisioni» su cosa fare dopo, a seconda che una cosa (o a volte diverse cose) sia già accaduta in precedenza.

Parole chiave e siti web

- Variabile booleana: `https://it.wikipedia.org/wiki/Variabile_booleana`
- Transistor: `https://it.wikipedia.org/wiki/Transistor`
- Rete logica: `https://it.wikipedia.org/wiki/Elettronica_digitale`
- Porta AND: `https://it.wikipedia.org/wiki/Porta_AND`
- Porta XOR: `https://it.wikipedia.org/wiki/Algebra_di_Boole#XOR`
- Tabella della verità: `https://it.wikipedia.org/wiki/Tabella_della_verità`
- Porta OR: `https://it.wikipedia.org/wiki/Porta_OR`
- Invertitore: `https://it.wikipedia.org/wiki/Invertitore`
- Porta logica: `https://it.wikipedia.org/wiki/Porta_logica`

12. Codice 8

Questo disco viene utilizzato per crittografare i testi in chiaro in testi cifrati:



All'inizio, il puntatore del disco è impostato su «ABC».

Ogni lettera viene crittografata singolarmente. A tal fine, vengono determinate due cifre:

- La prima cifra indica di quante posizioni è ruotato il puntatore in senso orario. Poi il puntatore viene posizionato sul blocco con la lettera da criptare.
- La seconda cifra indica la posizione della lettera da cifrare nel blocco puntato.

Ad esempio, la parola «RETE» è codificata come $53 - 42 - 51 - 32$.

Come si decifra il codice 52-12-43-54?

A) CASA

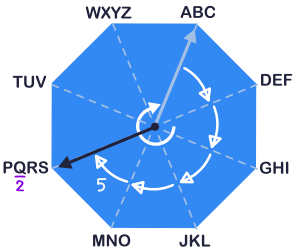
B) QUIZ

C) ROBOT

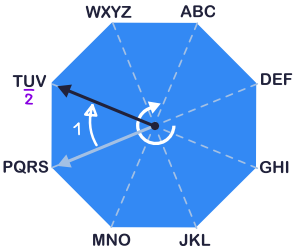
D) JAZZ

E) LUCE

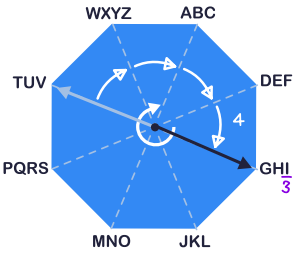
Soluzione



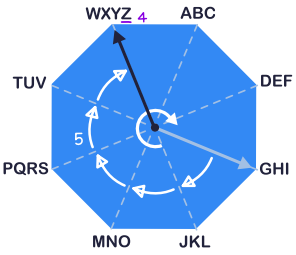
52 significa che il puntatore viene spostato dal blocco «ABC» al blocco «PQRS» (prima cifra 5) e che viene presa la seconda lettera «Q» (seconda cifra 2).



12 significa che il puntatore viene spostato dal blocco «PQRS» al blocco «TUV» (prima cifra 1) e che viene presa la seconda lettera «U» (seconda cifra 2).



43 significa che il puntatore viene spostato dal blocco «TUV» al blocco «GHI» (prima cifra 4) e che viene presa la terza lettera «I» (seconda cifra 3).



54 significa che il puntatore viene spostato dal blocco «GHI» al blocco «WXYZ» (prima cifra 5) e che viene presa la quarta lettera «Z» (seconda cifra 4).

Ciò significa che la risposta B) «QUIZ» è corretta.

Avresti potuto trovare questa soluzione più rapidamente: La risposta C) ROBOT non è possibile, perché è composta da cinque lettere, ma il testo cifrato ne rappresenta solo quattro. Poiché l'ultima lettera è codificata con un 4 come seconda cifra, può essere solo «S» o «Z». Solo le risposte B) e D) soddisfano questo requisito. La lettera che la precede deve provenire dal blocco di lettere di cinque giri in senso antiorario, cioè dal blocco «GHI». Ciò significa che la risposta può essere solo B) «QUIZ».

Questa è l'informatica!

Per migliaia di anni, l'uomo ha cercato di nascondere le informazioni in modo che solo i destinatari potessero decifrarle. Ciò che è iniziato con strisce di carta avvolte intorno a un bastone si è sviluppato attraverso cifrari a trasposizione come il «codice Cesare» e procedure di *crittografia polialfabetica* (come la «procedura Vigenére») fino alla moderna *crittografia a chiave pubblica* (come «GnuPG», che utilizza la «procedura RSA», tra le altre).

Il metodo di crittografia di questo compito è un metodo di crittografia polialfabetico, perché la stessa lettera non è necessariamente crittografata con lo stesso testo cifrato: la lettera «E» nell'esempio è crittografata come 42 all'inizio, ma come 32 alla fine. In linea di massima, tutti questi metodi di crittografia possono essere decifrati in modo semplice e veloce con l'aiuto dei computer.

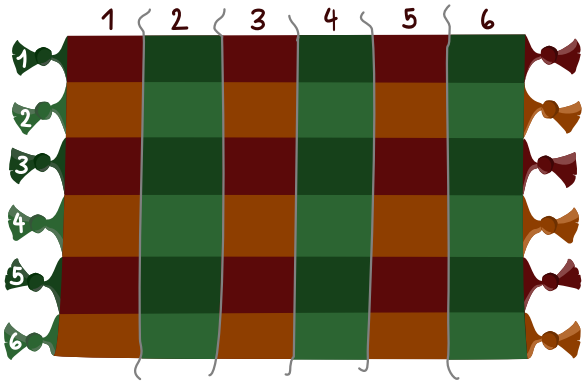
In questo caso, però, la decifrazione è semplicissima: esiste una sola chiave per criptare un testo. Anche se si potesse far partire la posizione iniziale del puntatore non dall'ABC ma da un qualche blocco, si avrebbero solo otto chiavi diverse... persino il codice Cesare, che ha più di 2000 anni, è «più sicuro». Ora si può ancora sostenere che il segreto non è la chiave ma il metodo di crittografia. Ma il *Principio di Kerckhoffs*, che Auguste Kerckhoffs (1835−1903) ha formulato nel 1883 e che è tuttora valido, chiarisce che la sicurezza di un *crittosistema* non deve basarsi sul mantenimento del segreto di un metodo di crittografia, perché questo potrebbe diventare troppo facilmente noto ad altri.

Parole chiave e siti web

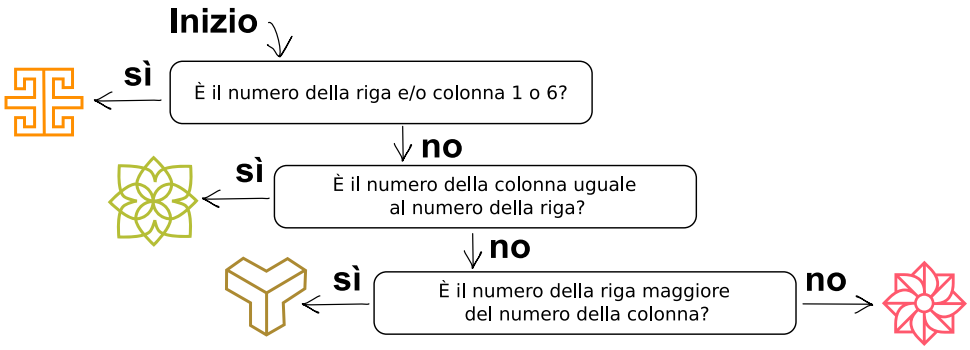
- Cifrario di Cesare: `https://it.wikipedia.org/wiki/Cifrario_di_Cesare`
- Cifrario polialfabetico: `https://it.wikipedia.org/wiki/Cifrario_polialfabetico`
- Cifrario: `https://it.wikipedia.org/wiki/Cifrario`
- Cifrario di Vigenère: `https://it.wikipedia.org/wiki/Cifrario_di_Vigenère`
- Crittografia asimmetrica: `https://it.wikipedia.org/wiki/Crittografia_asimmetrica`
- GNU Privacy Guard: `https://it.wikipedia.org/wiki/GNU_Privacy_Guard`
- RSA: `https://it.wikipedia.org/wiki/RSA_(crittografia)`
- Principio di Kerckhoffs: `https://it.wikipedia.org/wiki/Principio_di_Kerckhoffs`
- Crittosistema: `https://it.wikipedia.org/wiki/Crittosistema`
- Crittografia: `https://it.wikipedia.org/wiki/Crittografia`

13. Motivo del tappeto

Hale è un artista turco. Disegna un tappeto con una griglia di sei righe e sei colonne.

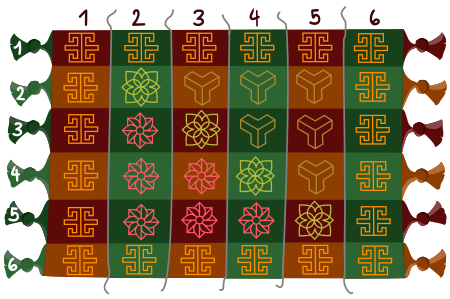


Hale numera le righe e le colonne. Quindi per ogni campo della griglia c'è il numero della riga e il numero della colonna. I commessi di Hale devono inserire un simbolo in ogni casella. Hale ha dato loro queste istruzioni per farlo:

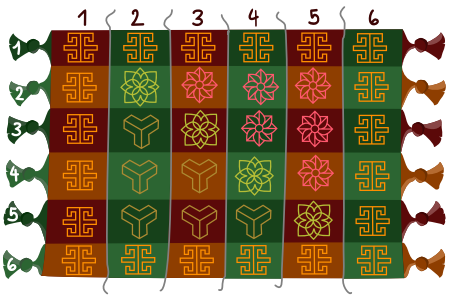


Come sarà il tappeto?

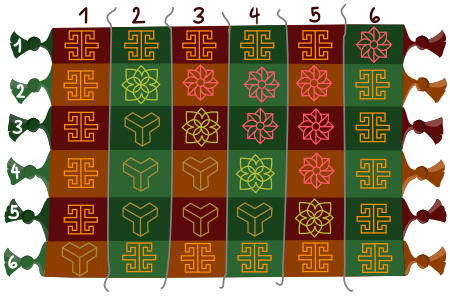
A)



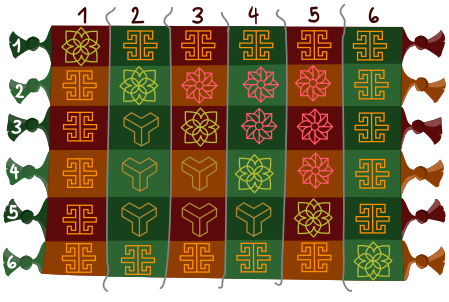
B)



C)



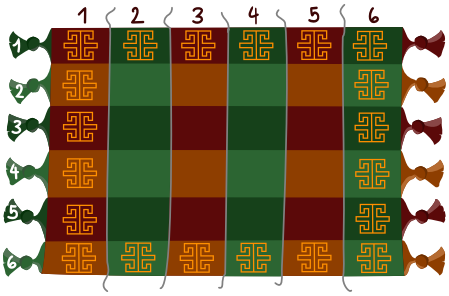
D)



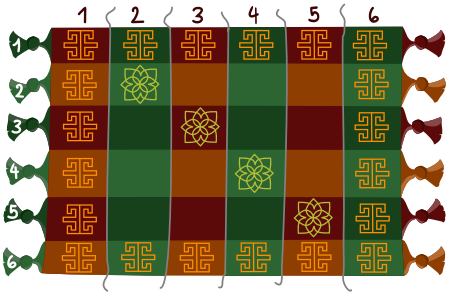
Soluzione

La risposta corretta è B).

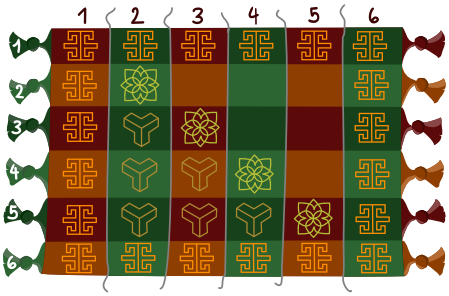
Alla prima domanda dell'immagine si risponde «sì» per tutti i quadrati sul bordo della griglia. Questo perché ogni campo del bordo si trova nella prima o nella sesta colonna o nella prima o nella sesta riga. A questi campi viene assegnato il simbolo  e si ottiene la seguente disposizione:



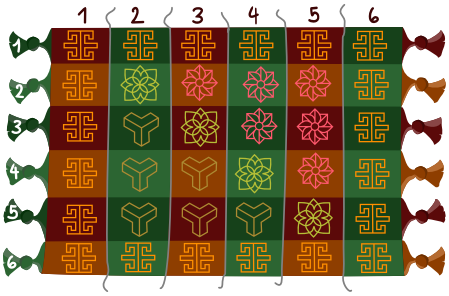
Alla seconda domanda si risponde con «sì» per tutti i campi sulla diagonale, perché sulla diagonale i numeri di colonna e di riga sono gli stessi. Questi campi ricevono il simbolo  e lo schema del tappeto si presenta come segue:



Secondo la terza domanda, tutti i campi il cui numero di riga è maggiore del numero di colonna ricevono il simbolo  .

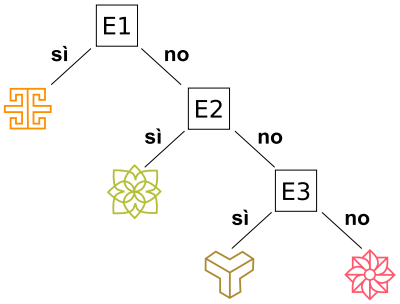


Per i campi rimanenti, alla terza domanda si risponde con «No». Ciò significa che il numero della riga non è maggiore del numero della colonna. Tutti questi campi sono riempiti con il simbolo  . In questo modo si ottiene il modello di tappeto della risposta B.



Questa è l'informatica!

L'immagine che l'artista Hale ha sviluppato come guida è chiamata *albero di decisione* in informatica. Come un vero e proprio albero, un albero di decisione è composto da rami. In ogni ramo (E1 - E3) c'è una domanda a cui si risponde con «Sì» o «No». Percorrendo l'albero da cima a fondo, rispondendo alle domande e seguendo le linee di corrispondenza, si arriva a una decisione.



Nel compito, l'albero di decisione è il fulcro delle istruzioni per tessere un tappeto. Ogni persona che utilizza queste istruzioni per la tessitura realizza esattamente lo stesso tappeto. In linea di principio, anche una macchina potrebbe produrre il tappeto, a patto che sia in grado di leggere e comprendere le istruzioni.

In informatica, un'istruzione unica di questo tipo è chiamata *algoritmo*. Se un algoritmo è scritto in un *linguaggio di programmazione* e può essere eseguito da un computer, si chiama *programma informatico*.

Nella vita di tutti i giorni, spesso si ha a che fare con programmi informatici che prendono decisioni: Il controllo del semaforo decide quando il semaforo pedonale diventa verde. Il sistema operativo del cellulare decide quando passare alla modalità di risparmio energetico. Il controllo automatico dei passaporti in aeroporto decide se il passaporto è valido.

Alla base di tutti questi programmi ci sono gli alberi di decisione.

Parole chiave e siti web

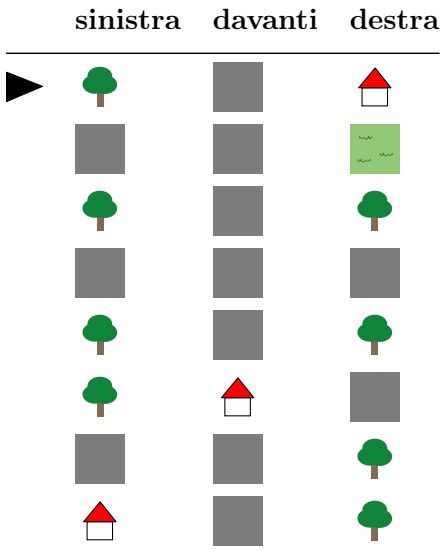
- Albero di decisione: `https://it.wikipedia.org/wiki/Albero_di_decisione`
- Algorithmo: `https://it.wikipedia.org/wiki/Algoritmo`
- Linguaggio di programmazione: `https://it.wikipedia.org/wiki/Linguaggio_di_programmazione`
- Programma: `https://it.wikipedia.org/wiki/Programma_(informatica)`

14. La posta robotizzata

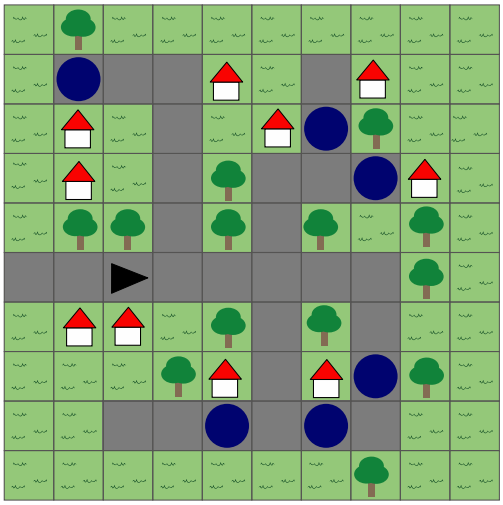
Il robot Tina consegna la posta. Tina utilizza una mappa suddivisa in campi. Tina si sposta lungo la strada verso una strada adiacente a sinistra, a destra o davanti (cioè non in diagonale).

Tina ha tre sensori per la navigazione. Non appena Tina entra in una strada (e prima che Tina possa girarsi), i sensori rilevano ciò che si trova a sinistra, a destra e di fronte a Tina.

La tabella documenta ciò che i sensori di Tina hanno rilevato in ogni casella del suo percorso. Tina inizia sulla casella ▶, in direzione della freccia.

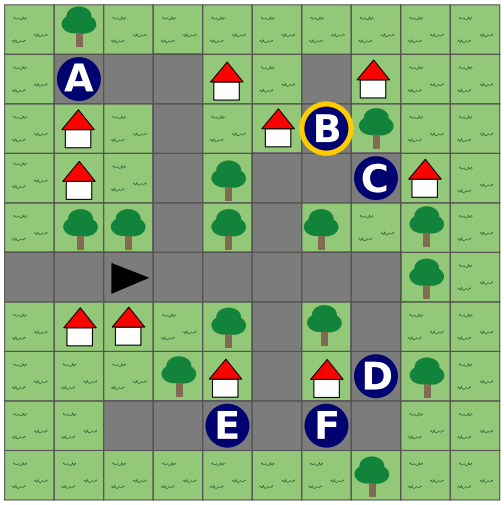


Quale dei punti blu scuro ● Tina raggiungerà alla fine del suo percorso?



Soluzione

La risposta corretta è il punto B.

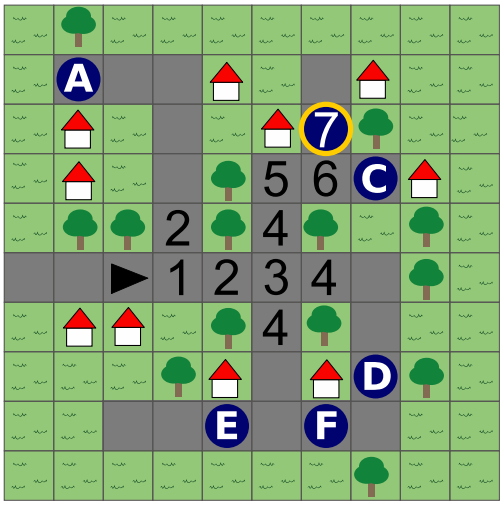


Passo	sinistra	davanti	destra
▶			
1			
2			
3			
4			
5			
6			
7			

In questo caso è sufficiente concentrarsi sui sei punti di destinazione e vedere se le indicazioni del sensore del passaggio 7 «  » possono essere adatte. In questo modo è possibile escludere C, E e F. Le specifiche del sensore del passo 6 sono «  », quindi è possibile escludere A e D.

In alternativa, si può provare a seguire il percorso documentato nella tabella. Il percorso verso il punto B è l'unico che corrisponde.

Se si traccia il percorso di Tina utilizzando le informazioni dei sensori, non è sempre possibile decidere immediatamente dove Tina si è spostata. Nel passo 4, Tina vedeva gli alberi a sinistra e a destra, indipendentemente dalle tre direzioni in cui si muoveva. In questa situazione, è necessario prendere in considerazione anche le informazioni del sensore dopo il movimento successivo per poter determinare chiaramente il punto 4.



Questa è l'informatica!

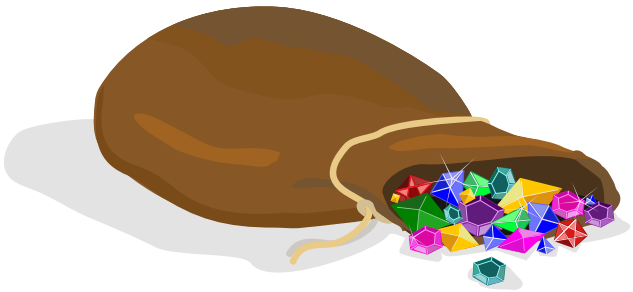
In questo compito incontriamo il *robot* Tina. I robot sono computer appositamente attrezzati che raccolgono informazioni dall'ambiente circostante con l'aiuto di *sensori*, le elaborano automaticamente (cioè con un programma) e, in base al risultato, eseguono autonomamente un'azione nel loro ambiente attraverso i cosiddetti *attuatori*. I sensori di Tina rilevano innanzitutto il contenuto delle caselle sinistra, davanti e destra. Nello specifico, potremmo immaginare che i sensori scattino foto e che dall'analisi automatizzata di queste immagini vengano estratti dati geometrici che il computer può assegnare a una casa, un albero o una strada. Il corpo di Tina, cioè gli attuatori, potrebbero essere controllati per evitare campi con alberi o una casa.

Le auto a guida autonoma sono esempi famosi di questi robot. Sono dotati di numerosi sensori che non solo misurano la velocità o la posizione corrente, ma anche la distanza dal ciglio della strada e rilevano gli oggetti presenti sulla strada o a bordo strada e molto altro ancora. Queste informazioni vengono elaborate da programmi a volte molto complessi che possono, ad esempio, riconoscere i bambini che potrebbero attraversare la strada e distinguerli da un cartello stradale. In molti di questi scenari, il cosiddetto *apprendimento automatico* è la tecnologia chiave. Nel caso delle auto a guida autonoma, i computer imparano, sulla base di numerosi esempi, a distinguere i bambini dai segnali stradali. Gli attuatori sono quindi, ad esempio, i freni, che vengono attivati in modo indipendente o senza l'intervento umano.

Parole chiave e siti web

- Robot: `https://it.wikipedia.org/wiki/Robot`
- Sensore: `https://it.wikipedia.org/wiki/Sensore`
- Attuatore: `https://it.wikipedia.org/wiki/Attuatore`
- Apprendimento automatico: `https://it.wikipedia.org/wiki/Apprendimento_automatico`

15. Pietre preziose



Pietro ha delle pietre preziose. Hanno tutte un valore diverso.

Sarah conosce le pietre preziose di Pietro, ma non il loro valore. Vuole sapere qual è la pietra più preziosa.

A tal fine, esegue tre volte la seguente procedura:

- Sceglie quattro pietre di Pietro e gli chiede quale sia la più preziosa.

Ogni volta sceglie le quattro pietre a caso e Pietro le dà ogni volta una risposta sincera.

Dopodiché, Sarah sa qual è la pietra più preziosa.

Quante pietre preziose può avere al massimo Pietro?

A) 8 pietre preziose

B) 10 pietre preziose

C) 11 pietre preziose

D) 12 pietre preziose

Soluzione

La risposta B) è corretta: 10 pietre preziose

Se Pietro ha 10 pietre, Sarah può scegliere un totale di otto pietre diverse nelle prime due domande. Le due «vincitrici» delle singole domande (cioè le pietre più preziose tra le quattro scelte) possono anche essere «vincitrici complessive», cioè la pietra di maggior valore in assoluto. Le altre sei pietre vengono eliminate. Per l'ultima domanda, sceglie i due vincitori e le due pietre non ancora scelte. Il vincitore di questa domanda deve essere la pietra con il maggior valore.

Quindi, per 10 pietre, Sarah può (tra le altre cose) procedere in questo modo per trovare la pietra più preziosa. Se Pietro ha 11 pietre, purtroppo non può farlo.

Se, come sopra, Sarah confronta un totale di otto pietre diverse nelle prime due domande, le rimangono le due pietre con il valore più alto e altre tre pietre, una di troppo, per trovare il vincitore assoluto con la terza domanda. Se, invece, Sarah confronta il vincitore della prima domanda con le 3 «nuove» pietre della seconda domanda, allora conosce la più preziosa delle sette pietre scelte. Deve confrontare questa pietra con le altre quattro. Anche questa è una pietra di troppo per la terza domanda.

Se Sarah sceglie solo sei o anche meno pietre diverse per le prime due domande con 11 pietre, o se Pietro ha più di 12 pietre, Sarah non può sapere quale pietra è la più preziosa dopo tre domande.

Questa è l'informatica!

Questo compito riguarda un *algoritmo* vincolato da condizioni. Nel nostro caso, Sarah può porre solo tre domande e ogni domanda può contenere solo 4 elementi.

Nonostante questa restrizione, l'algoritmo funziona bene per raccolte di dimensioni inferiori a 11, ma non funziona altrimenti.

Le ragioni per imporre restrizioni agli algoritmi possono essere varie. Ad esempio, si potrebbe richiedere che un'operazione venga completata in un tempo fisso, come avviene nei sistemi operativi in tempo reale. Un altro motivo potrebbe essere che le operazioni possono comportare costi esterni o danneggiare un componente.

Non è un problema se l'algoritmo fallisce al di sopra di una certa soglia, purché sia garantito che tale soglia non venga mai raggiunta. Ad esempio, la strategia ristretta di questo compito non deve mai essere utilizzata per collezioni superiori a 10.

Parole chiave e siti web

- Algoritmo: `https://it.wikipedia.org/wiki/Algoritmo`
- Complessità temporale: `https://it.wikipedia.org/wiki/Complessità_temporale`

A. Autori dei quesiti

Gulgun Afacan
Esraa Almajhad
Waël Almoman
Leo Barichello
Liam Baumann
Wilfried Baumann
Linda Björk Bergsveinsdóttir
Tobias Berner
Sarah Chan
Byeonggyu Cho
Christian Datzko
Susanne Datzko
Justina Dauksaite
Nora A. Escherle
Gerald Futschek
Christian Giang
Mark Edward M. Gonzales
Adam Grodeck
Yasemin Gülbahar
Benjamin Hirsch
Alisher Ikramov
Dauksaite Justina
Dong Yoon Kim
Hakin Kim
Jihye Kim
Seulki Kim
Vaidotas Kinčius
Lidija Kralj
Regula Lacher
Taina Lehtimäki
Karolína Miková
Jelena Milojkovic
Ágnes Erdősné Németh
Jean-Philippe Pellet
Margot Phillipps
Zsuzsa Pluhár
Wolfgang Pohl
John-Paul Pretti
Le Quang Quan
Susannah Quidilla
Chris Roffey
Kirsten Schlüter
Giovanni Serafini
Yeh Yi Shan
Bernadette Spieler
Alieke Stijf
Goran Sukovic
Monika Tomcsányiová
Ahto Truu
Troy Vasiga
Michael Weigend
Kyra Willekes

B. Sponsoring: concorso 2022

HASLERSTIFTUNG

http://www.haslerstiftung.ch/

Kanton Zürich
Volkswirtschaftsdirektion
Amt für Wirtschaft und Arbeit

Standortförderung beim Amt für Wirtschaft und Arbeit Kanton Zürich

UBS

http://www.ubs.com/

verkehrshaus.ch

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

i-factory (Musée des transports, Lucerne)

senarclens
leu+partner
strategische kommunikation

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

ABZ
AUSBILDUNGS- UND BERATUNGSZENTRUM
FÜR INFORMATIKUNTERRICHT

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

hep/ haute école pédagogique vaud

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

Scuola universitaria professionale
della Svizzera italiana
SUPSI

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

C. Ulteriori offerte

La Fiamma IT: `https://it-feuer.ch/it/`
In Svizzera, numerose organizzazioni si impegnano per la formazione delle giovani leve nell'ambito dell'informatica. L'iniziativa «La Fiamma IT» vuole unire queste forze e contribuire insieme a diffondere il tema nell'opinione pubblica in tutta la Svizzera. La fiamma IT presenta numerose offerte rivolte sia ai docenti che agli studenti.

CoetryLab: `https://www.coetry-lab.org/`
Il team del CoetryLab (Zürich) vuole dare ai bambini e ai giovani l'accesso alla programmazione e ai media. Il CoetryLab vuole essere il luogo di sperimentazione e progettazione extrascolastica e aprire il mondo del coding a tutti. Le loro idee possono essere realizzate in modo creativo e siti web, applicazioni, giochi e molto altro possono essere sviluppati in team o da soli.

Roteco: `https://www.roteco.ch/it/`
Il progetto Roteco consiste in una comunità di insegnanti desiderosi di preparare gli allievi per la società digitale. In questa comunità gli insegnanti trovano, sviluppano e si scambiano attività didattiche inerenti la robotica educativa e più in generale le scienze informatiche pronte da essere utilizzate in classe e vengono informati con le ultime novità e corsi in questi campi.

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.