INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Quesiti e soluzioni 2022

Tutte le Categorie

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

A cura di:

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

010100110101011001001001
010000010010110101010011
010100110100101010001001
001011010101001101010011
010010010100100101000001

SSII

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

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.

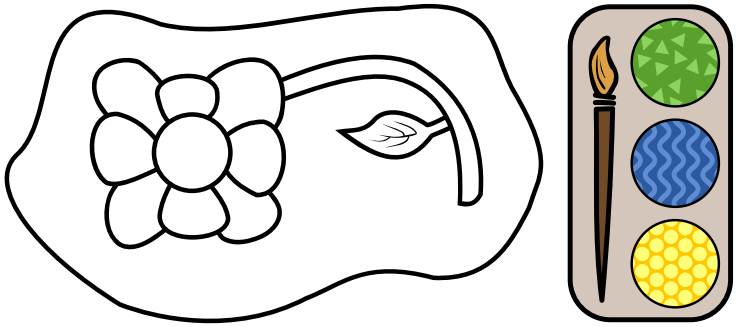
Per ulteriori informazioni:

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

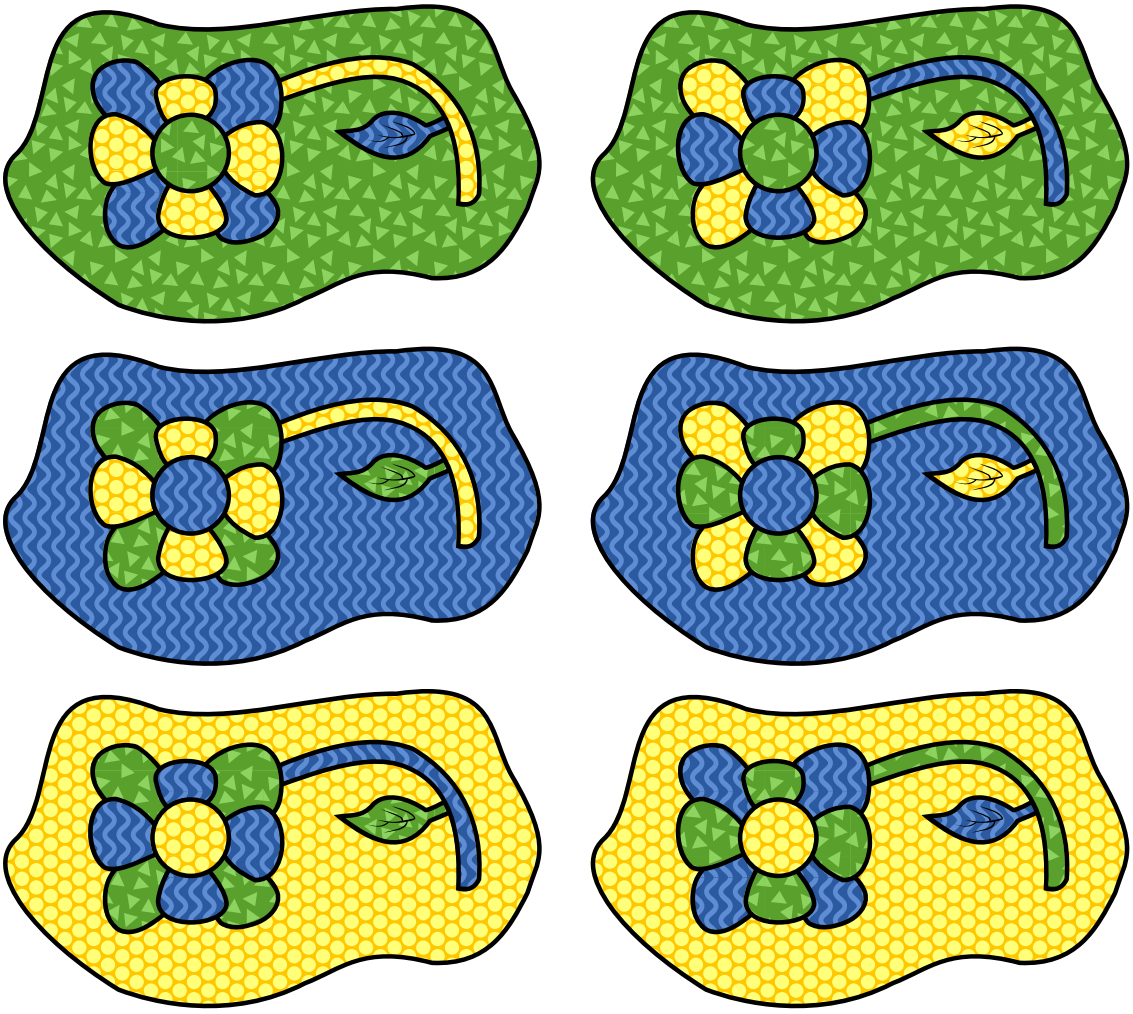
1. Colorazione



Colora l'immagine di verde, giallo e blu in modo che non ci siano due aree dello stesso colore che si toccano.

Soluzione

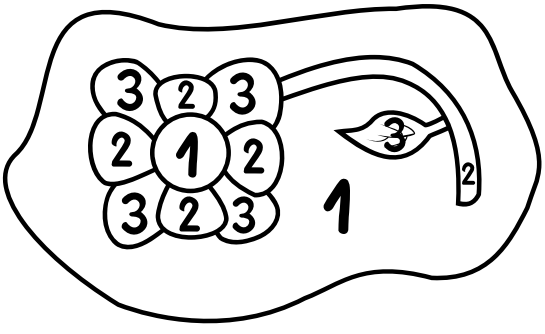
Qui si possono vedere tutte le possibilità di come colorare l'immagine.



Come si può trovare una delle soluzioni?

Si sceglie innanzitutto un colore per la superficie più esterna, ad esempio il giallo. La maggior parte delle altre superfici tocca la superficie gialla. Quindi devono essere colorati di blu e verde. Si inizia con una di queste aree e si alternano i colori. Il centro del fiore può essere quindi giallo.

Questa immagine numerata mostra questa strategia indipendentemente dai colori:



Questa è l'informatica!

Questo compito consiste nell'assegnare i colori alle aree, rispettando determinati vincoli. In informatica e in matematica, tali problemi sono noti come *colorazione di grafi*.

I problemi di colorazione di grafi hanno molte applicazioni e spesso è importante utilizzare il minor numero possibile di colori. Tra gli esempi ci sono la divisione delle squadre in un torneo sportivo, la suddivisione delle persone in gruppi o l'assegnazione delle frequenze delle stazioni radio. Nelle applicazioni dei problemi di colorazione di grafi, di solito ci sono più superfici che in questo compito. Spesso non è più possibile trovare una colorazione con pochi colori a mano. Gli informatici hanno bisogno di un computer per risolvere questi compiti.

Il problema della colorazione di grafi può essere utilizzato anche per le mappe. In questo caso, l'obiettivo è trovare una colorazione in modo che i paesi vicini non siano colorati allo stesso modo. Per ogni mappa è possibile trovare una tale colorazione con solo quattro colori ma non è facile da dimostrare. Solo nel 1976 i matematici Kenneth Appel e Wolfgang Haken ci riuscirono. Per farlo, hanno utilizzato i computer per verificare un gran numero di eccezioni e controesempi. Altre persone non possono riprodurlo senza computer, ecco perché ci sono altri matematici che criticano l'uso dei computer in questa prova o nelle prove in generale.

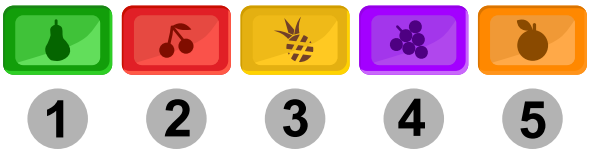
Parole chiave e siti web

- Colorazione di grafi: `https://it.wikipedia.org/wiki/Colorazione_dei_grafi`
- Teorema dei quattro colori: `https://it.wikipedia.org/wiki/Teorema_dei_quattro_colori`

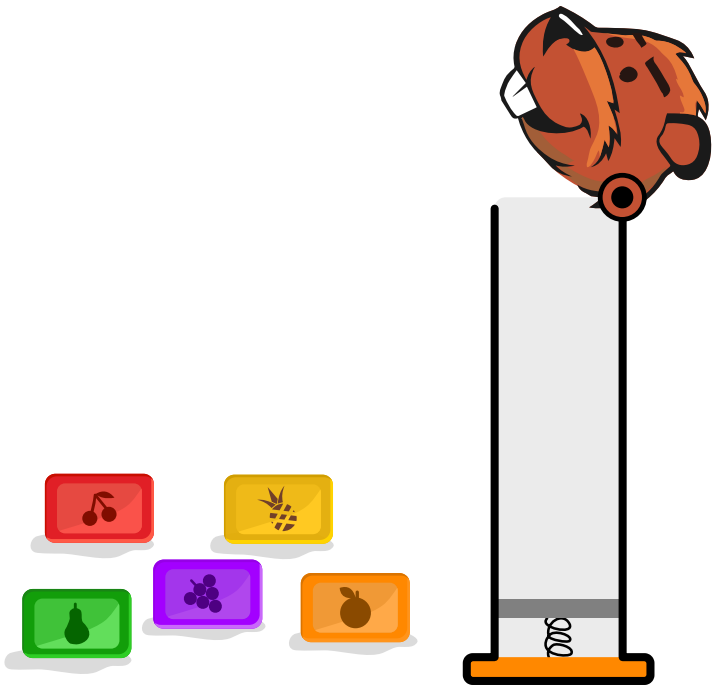
2. Caramelle preferite

Anna riempie un distributore con cinque caramelle. Poi potrà mangiare le caramelle una dopo l'altra mentre escono dalla parte superiore del distributore.

Vuole mangiare le caramelle una dopo l'altra in quest'ordine:



Come deve fare Anna per riempire le caramelle nel distributore?



Soluzione

Affinché le caramelle escano dal distributore nell'ordine corretto, è importante capire che la caramella che viene inserita per prima esce per ultima. Ciò significa che il distributore deve essere riempito nel seguente modo:

Questa è l'informatica!

Se Anna si limita a inserire le caramelle nel distributore nell'ordine in cui vuole mangiarle, queste usciranno esattamente nell'ordine opposto. Pertanto, Anna decide prima l'ordine che desidera! Poi immagina come deve riempire il distributore per ottenere i dolci nell'ordine desiderato.

Per gli informatici è spesso importante pensare anche all'ordine. L'ordine utilizzato in questo compito è chiamato «ordine di pila». Le *pile* sono strutture di memoria che aggiungono e tolgono oggetti in un certo ordine: Lavorano secondo il principio *«Last in First out» (LIFO)*. Ciò significa che gli oggetti aggiunti per ultimi vengono eliminati per primi.

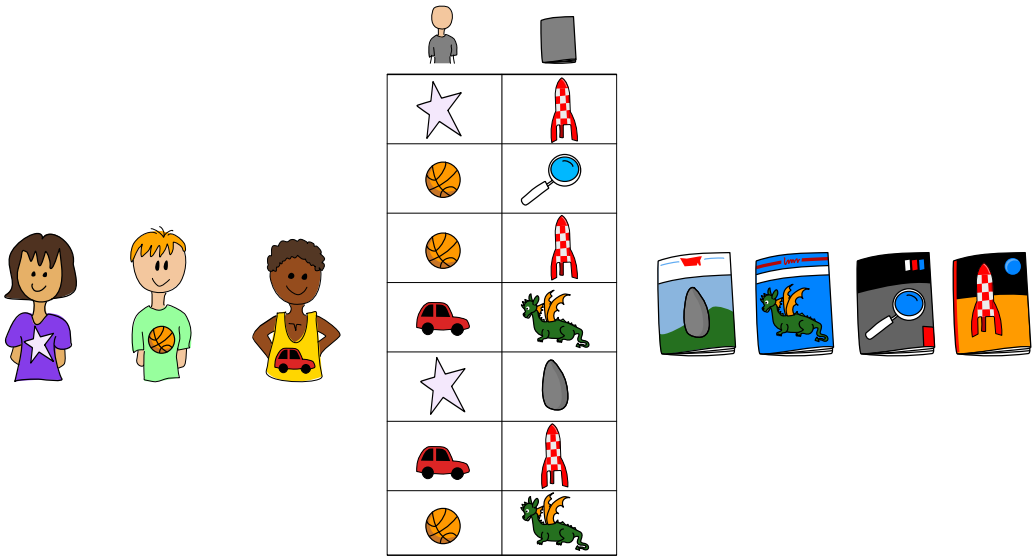
Parole chiave e siti web

- Pila: `https://it.wikipedia.org/wiki/Pila_(informatica)`
- Last in First out (LIFO): `https://it.wikipedia.org/wiki/LIFO`

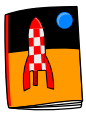
3. Biblioteca

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?

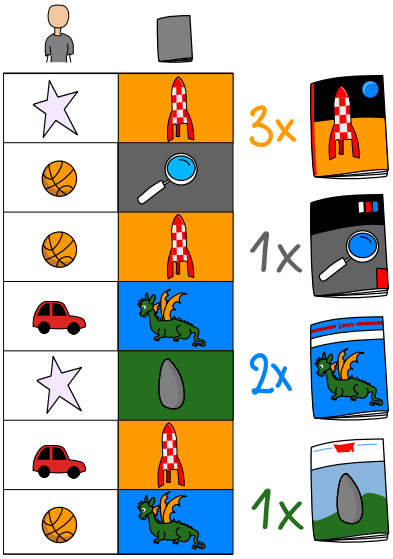


Soluzione

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.

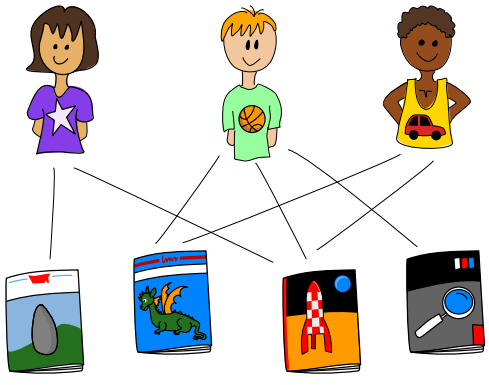


Quindi è il libro con il razzo che è stato preso in prestito più spesso.

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.

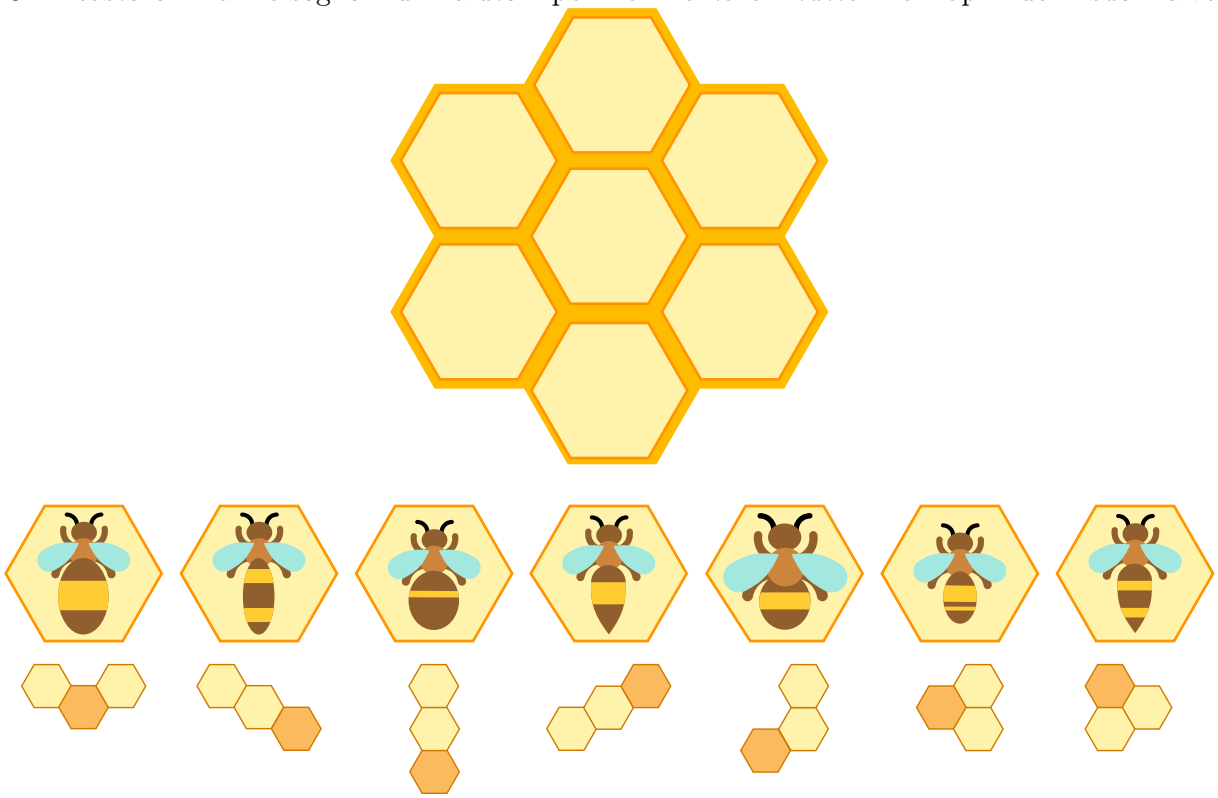
In informatica, tali tabelle sono chiamate *basi di dati* relazionali.

Parole chiave e siti web

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

4. Alveare

Un castoro ha bisogno di aiuto per alimentare tutte le api del suo alveare.

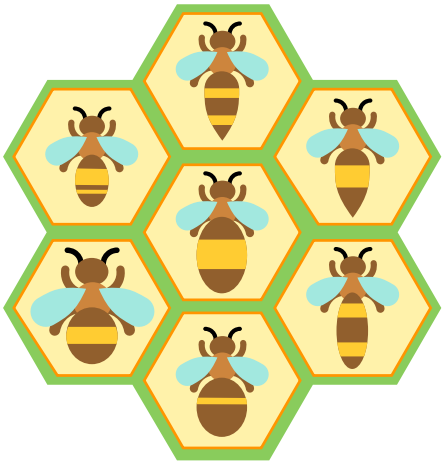


Sotto ogni ape, una regola indica in quale cella può essere alimentata l'ape.

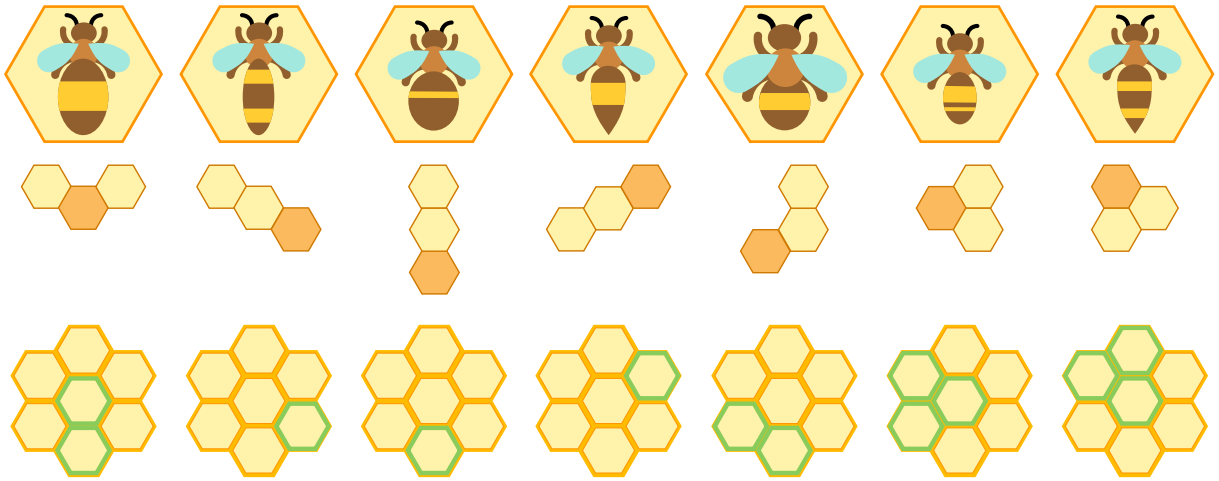
Alimenta le api nell'alveare. Rispetta le regole sotto le api.

Soluzione

Le api possono essere alimentate nell'alveare solo in questo modo:



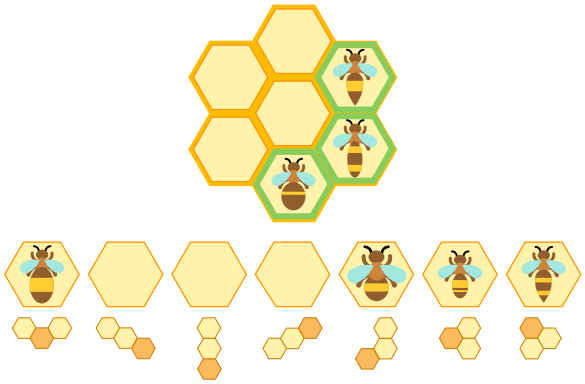
Il compito può essere risolto per tentativi ed errori. Ma questo può richiedere molto tempo. Per trovare un modo più veloce, si può osservare più da vicino le regole delle api. Nella figura seguente è possibile vedere ogni ape e la relativa regola. Le celle in cui l'ape può essere alimentata secondo la sua regola sono delineate in verde.



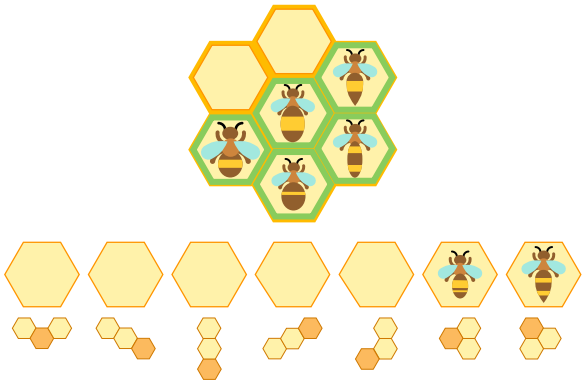
Si può notare che per alcune api ci sono più possibilità di alimentarle secondo le regole, per altre ce ne sono meno. Per tre api c'è solo un modo per alimentarle.

Per risolvere il compito in modo più rapido rispetto ai tentativi, si può procedere come segue:

Per prima cosa alimentare le api per le quali è possibile utilizzare una sola cella dell'alveare.



Allora c'è solo una cella possibile per le due api successive.



Allo stesso modo, ci si occupa delle ultime due api.

Questa è l'informatica!

In questo compito, sette api devono essere alimentate in sette celle diverse. Ci sono molti modi per prendersi cura delle api. Se si tiene conto delle regole, il numero di possibilità si riduce già di molto, ma è ancora così alto che provare tutte le possibilità richiederebbe uno sforzo considerevole. La chiave per una rapida soluzione del compito è l'ordine giusto. In questo caso, abbiamo iniziato con gli elementi più ristretti, cioè le api che hanno una sola cella possibile, per limitare il numero di casi da studiare.

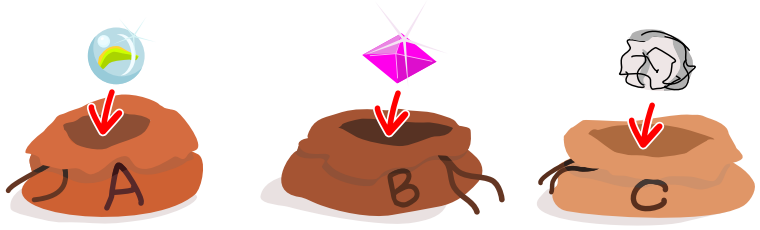
Un approccio risolutivo di questo tipo è chiamato *euristico* in informatica. Grazie a un'ingegnosa sequenza di soluzioni, la soluzione esatta può essere trovata con pochi passaggi. Tuttavia, per alcuni problemi, come la pianificazione di un percorso tra diverse località in un sistema di navigazione, un approccio euristico va a scapito della precisione. Questo perché le soluzioni possibili sono numerosissime. Per avere la garanzia di trovare il percorso migliore, bisognerebbe calcolare e confrontare tutti i percorsi possibili attraverso l'intera rete di percorsi, il che comporterebbe un'enorme quantità di calcoli. Provando prima le possibilità che hanno maggiori probabilità di portare a una buona soluzione, lo sforzo computazionale può essere ridotto in modo significativo. In questo modo, è possibile determinare un buon percorso in pochi secondi invece di calcolare il migliore in anni.

Parole chiave e siti web

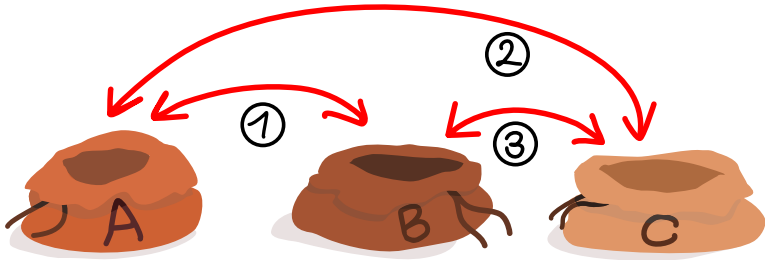
- Euristica: `https://it.wikipedia.org/wiki/Euristica`
- Problema del commesso viaggiatore:
  `https://it.wikipedia.org/wiki/Problema_del_commesso_viaggiatore`

5. Permutazioni

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



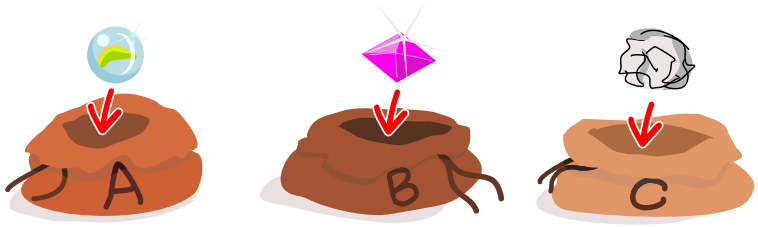
Poi scambia il contenuto del sacchetto A e del sacchetto B, quindi il contenuto di A e C e infine scambia il contenuto di B e C.



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

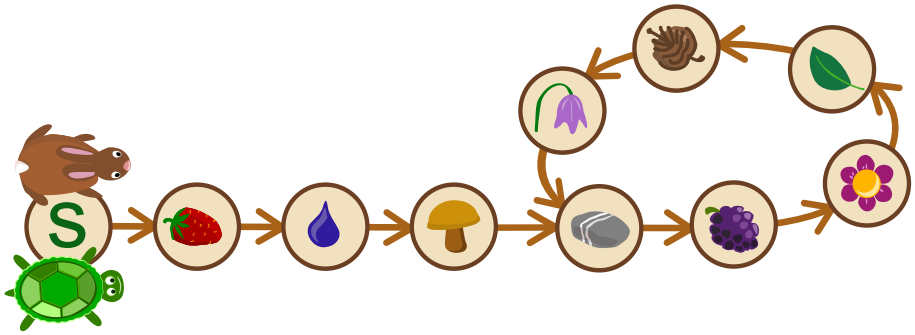
È inoltre interessante notare che è possibile creare qualsiasi disposizione solo con le permutazioni. Ciò richiede al massimo $n - 1$ permutazioni per n cose.

Parole chiave e siti web

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

6. 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.

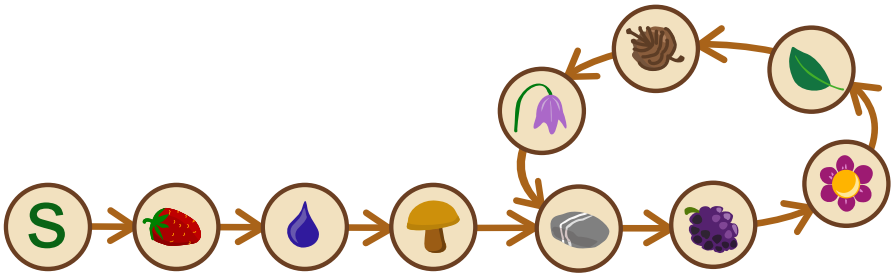
In quale campo la tartaruga e la lepre si incontrano per la prima volta dopo la partenza?

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:

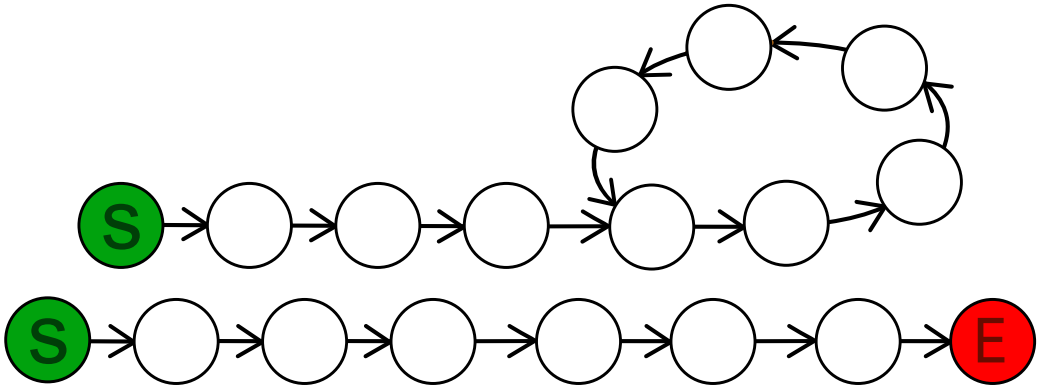
Minuti dopo l'inizio	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	...
	S														...
	S														...



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.

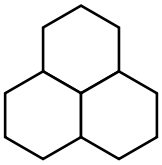
Parole chiave e siti web

- 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`

7. Piramide colorata

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

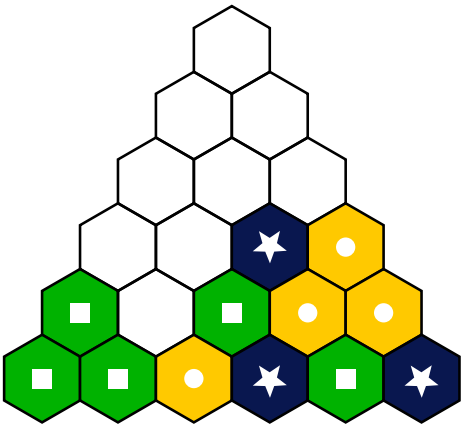
Sami vuole che quando tre esagoni si trovano esattamente insieme in questo modo (due in basso e uno in alto al centro), devono finire ...



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

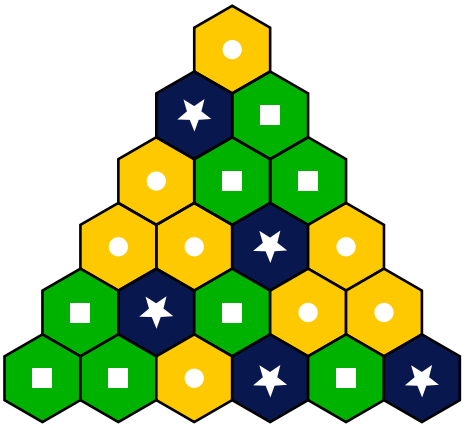
Sami ha messo insieme molti esagoni e ne ha già colorati alcuni.

Colora tutti gli esagoni rimanenti come piace a Sami.



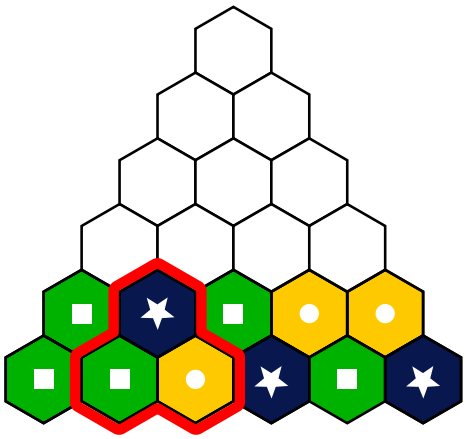
Soluzione

Questa è la soluzione giusta:

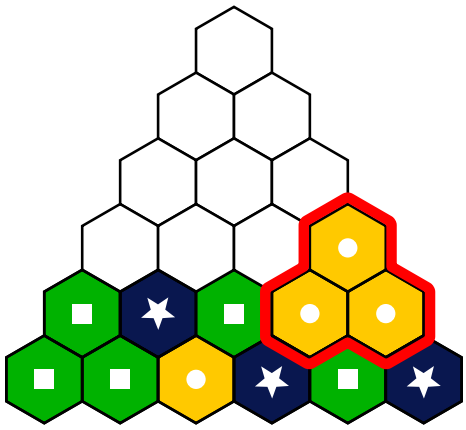


Non appena vengono colorati due esagoni vicini nella piramide di esagoni, il colore dell'esagono superiore viene fissato:

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

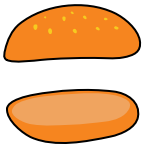
- Algoritmo: `https://it.wikipedia.org/wiki/Algoritmo`
- Selezione: `https://it.wikipedia.org/wiki/Selezione_(informatica)`
- Ciclo: `https://it.wikipedia.org/wiki/Struttura_di_controllo#Ciclo_for`

8. 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

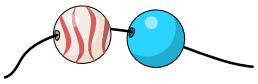
- Programmazione a vincoli: `https://it.wikipedia.org/wiki/Programmazione_a_vincoli`
- Problema di soddisfacimento di vincoli: `https://it.wikipedia.org/wiki/Problema_di_soddisfacimento_di_vincoli`
- Congiunzione logica: `https://it.wikipedia.org/wiki/Congiunzione_logica`

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

9. Collana da marinaio

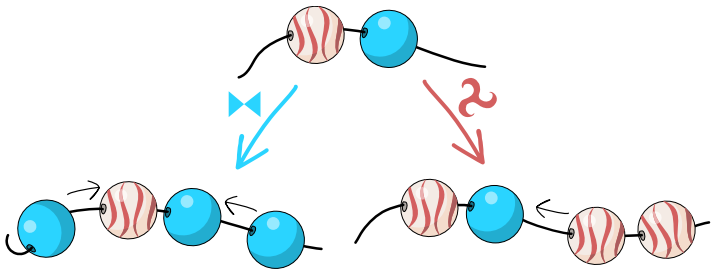
Ecco il manuale per la collana da marinaio di Monika con perline a onda bianche e rosse e perline blu semplici.

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



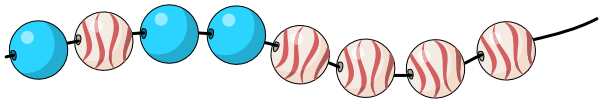
Poi puoi estendere la collana da marinaio,

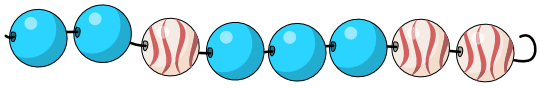
- aggiungendo una perlina blu a ciascuna estremità della stringa (⋈)
- oppure aggiungendo due perline a onda all'estremità destra della stringa (ᔓ)

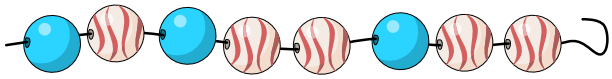


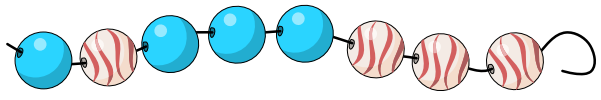
Puoi eseguire queste azioni più volte per creare collane sempre più lunghe.

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

A) 

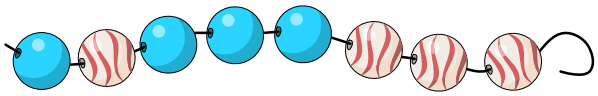
B) 

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

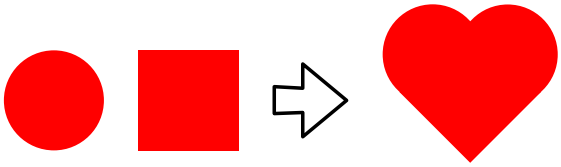
controllo della corrispondenza delle parentesi può essere fatto più o meno nello stesso modo in cui si controlla se una collana è una delle collane di marinaio di Monika.

Parole chiave e siti web

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

10. 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

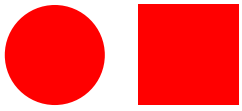
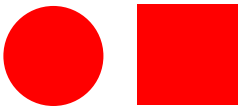
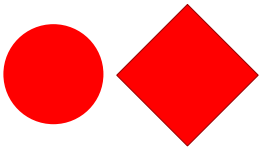
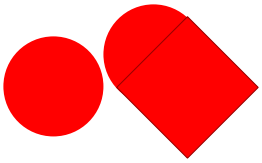
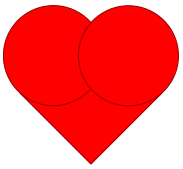
D) *sposta* cerchio, *sposta* cerchio, *raddoppia* cerchio, *sposta* quadrato

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

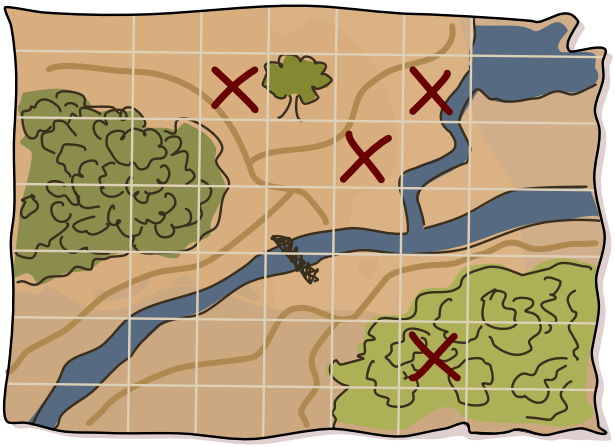
- Pixel: `https://it.wikipedia.org/wiki/Pixel`
- Grafica raster: `https://it.wikipedia.org/wiki/Grafica_raster`
- Grafica vettoriale: `https://it.wikipedia.org/wiki/Grafica_vettoriale`

11. 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?

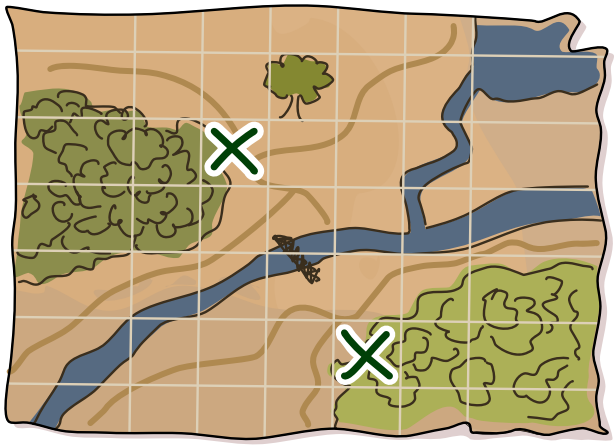
Per confondere le cose, Bilbo segna altri campi con ✕. Lo fa in modo che in ogni riga e colonna della mappa sia segnato un numero pari di caselle. Poi rimuove i due ✕ dai campi con i suoi nascondigli. Di seguito è possibile vedere il risultato.

In quali campi si trovano i nascondigli di Bilbo?

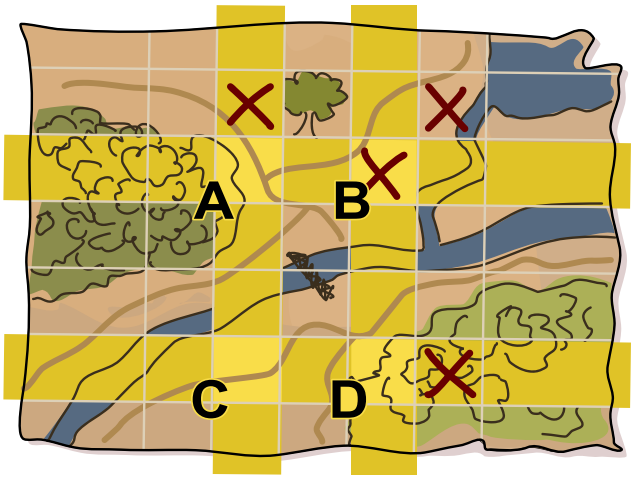


Soluzione

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.



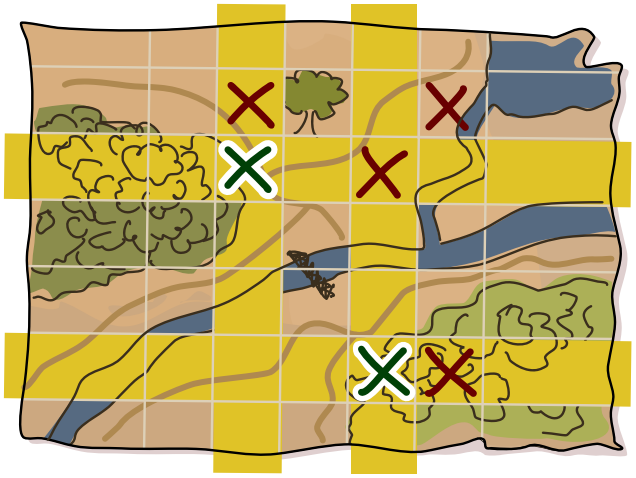
Dopo tutto, i ✕ che segnalano i nascondigli sono stati rimossi. Sappiamo che deve esserci un numero pari di ✕ in tutte le righe e le colonne dopo che le ✕ cancellate sono state reinserite.

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.

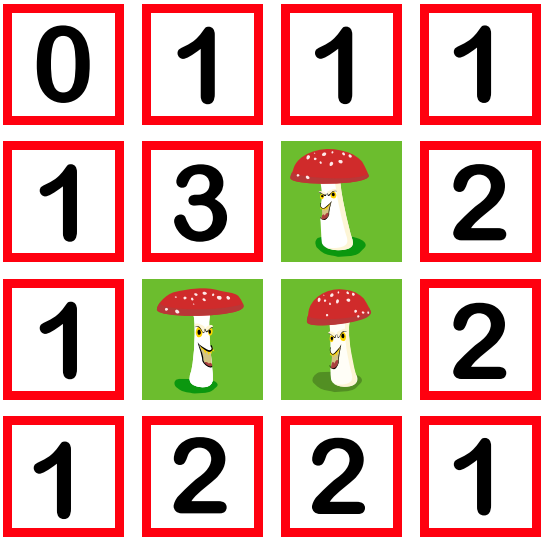
Parole chiave e siti web

- 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`

12. Attenzione ai funghi

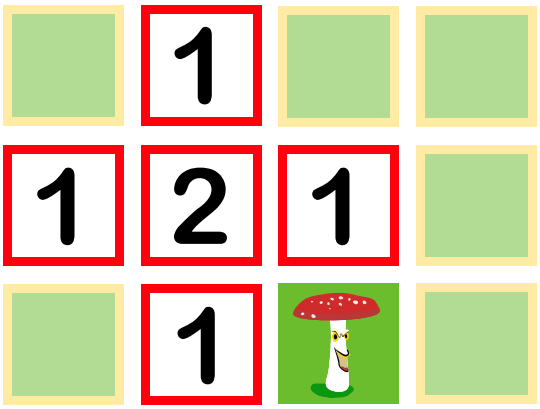
Nel gioco «Attenzione ai funghi», all'inizio è visibile esattamente un fungo. Tutte le altre caselle del tabellone sono coperte. Se si scopre un campo, appare un altro fungo o il numero di funghi sui campi vicini. Se si scoprono tutte le caselle in cui non è nascosto alcun fungo, si vince.

Ecco un esempio di una tavola completamente scoperta:



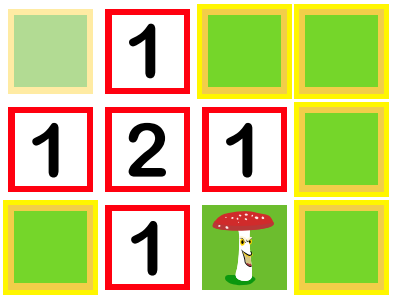
Hai iniziato una nuova partita e hai già scoperto alcune caselle.

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

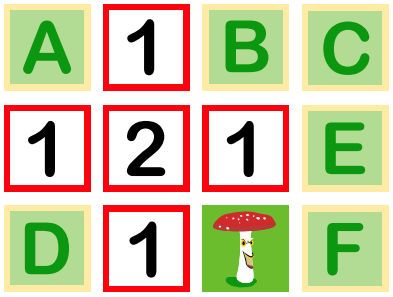


Soluzione

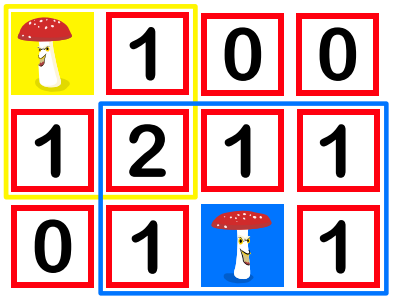
Questa è la soluzione:



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`

13. Ricamo

Lana ha una macchina da ricamo programmabile. La macchina può ricamare due tipi di motivi:  o  . Per creare questo motivo composto  sono necessari entrambi i motivi  e  . Nel mezzo, il tessuto deve essere spinto indietro di un punto.

Lana può programmare la macchina da ricamo con i seguenti tre pulsanti:

 La macchina da ricamo ricama  .

 La macchina da ricamo ricama  .

 Il tessuto viene spostato indietro di un punto.

Un programma viene creato con i tasti ed eseguito ripetutamente dalla macchina da ricamo.

Ad esempio, la macchina da ricamo creerà...

- ... con questo programma  ...
- ... questo motivo: 

Quale dei seguenti programmi ha utilizzato Lana per creare questo motivo?

A) 

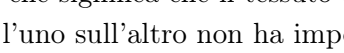
B) 

C) 

D) 

Soluzione

La risposta corretta è C). 

Per determinare il programma di Lana, cerchiamo innanzitutto la parte dello schema che si ripete:  . I primi 2 punti devono essere un  . Per questo utilizza  . All'inizio del programma di Lana devono esserci 2 . Il programma D non è corretto perché inizia con un . Il punto successivo dello schema è una stella  . Per ricamare una stella la macchina deve cucire  e  uno sopra l'altro, il che significa che il tessuto deve essere spostato nel mezzo. L'ordine in cui  e  sono ricamati l'uno sull'altro non ha importanza. A questo scopo puoi utilizzare le due seguenti varianti di programma:  o .

I quattro programmi producono i seguenti schemi:

	programma	**schema generato**
A		
B		
C		
D		

Nel programma B e D i punti non sono nell'ordine corretto. I programmi A e C sono uguali fino al quinto punto ricamato. Il Programma A aggiunge altre due croci dietro la seconda stella. Quindi, quando il programma A viene ripetuto, ci sono quattro croci tra la seconda stella e quella successiva, invece di due sole croci.

Ecco perché solo il programma C è corretto.

Questa è l'informatica!

In questo compito, uno schema ricorrente viene creato da una sequenza di istruzioni. Anche nell'informatica, i problemi più grandi e complicati vengono spesso suddivisi in problemi più piccoli, più facili da comprendere, risolvere e, ad esempio, programmare. Un'abilità importante in questo processo è riconoscere queste sequenze di modelli ricorrenti per riutilizzare una soluzione. Questo può essere fatto, ad esempio, sotto forma di *cicli*.

Il programma generato dalla macchina da ricamo è un elenco di istruzioni scritte in un linguaggio di programmazione. In sostanza, una macchina da ricamo programmabile non è altro che un robot o un computer che esegue istruzioni. Proprio come una macchina da ricamo ricama i punti esatti, un computer esegue le istruzioni esatte di un programma. Seguire esattamente le istruzioni è un concetto importante nell'informatica. L'ordine delle istruzioni è altrettanto importante. Se cambiamo l'ordine, di solito cambia anche l'output del programma. Nel nostro caso, questo significa che una

sequenza diversa di istruzioni darà luogo a una sequenza diversa di punti e quindi a un disegno diverso (eccezione: stiamo ricamando una stella).

Parole chiave e siti web

- Linguaggio di programmazione: `https://it.wikipedia.org/wiki/Linguaggio_di_programmazione`
- Algoritmo: `https://it.wikipedia.org/wiki/Algoritmo`

14. Bulloni e dadi

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

Ben segue rigorosamente la seguente procedura:

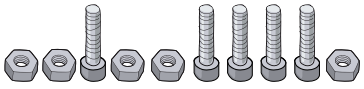
- Ben prende il componente successivo dalla catena di montaggio.
- Quando Ben ha preso un dado dalla catena di montaggio, lo mette nel secchio.
- Quando Ben ha preso un bullone dalla catena di montaggio, prende un dado dal secchio, lo avvita sul bullone e mette il pezzo finito nella scatola.

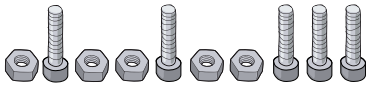
In questa procedura possono verificarsi due errori:

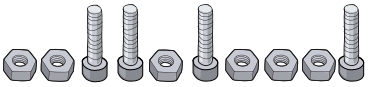
1. Ben prende un bullone dalla catena di montaggio, ma nel secchio non c'è nessun dado da avvitare.
2. Ben ha lavorato tutti i componenti della catena di montaggio, ma ci sono ancora dadi nel secchio.

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?

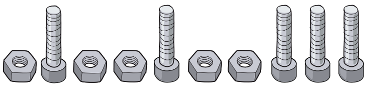
A) 

B) 

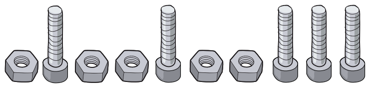
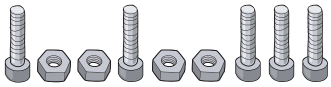
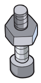
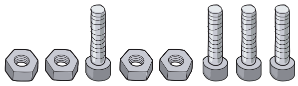
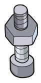
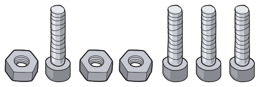
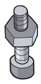
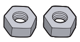
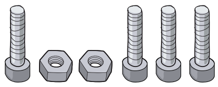
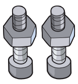
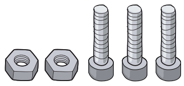
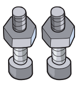
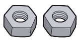
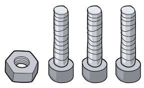
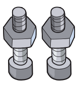
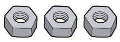
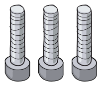
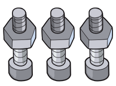
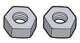
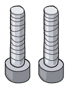
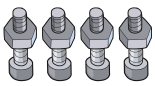
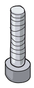
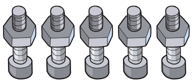
C) 

D) 

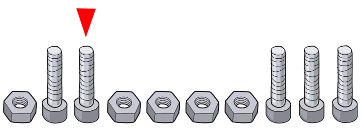
Soluzione

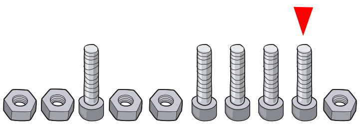
La risposta corretta è C) 

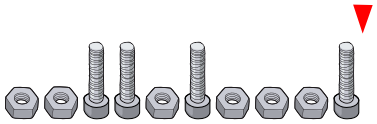
La tabella mostra lo stato della scatola per i pezzi finiti, del secchio per i dadi e della catena di montaggio.

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*.

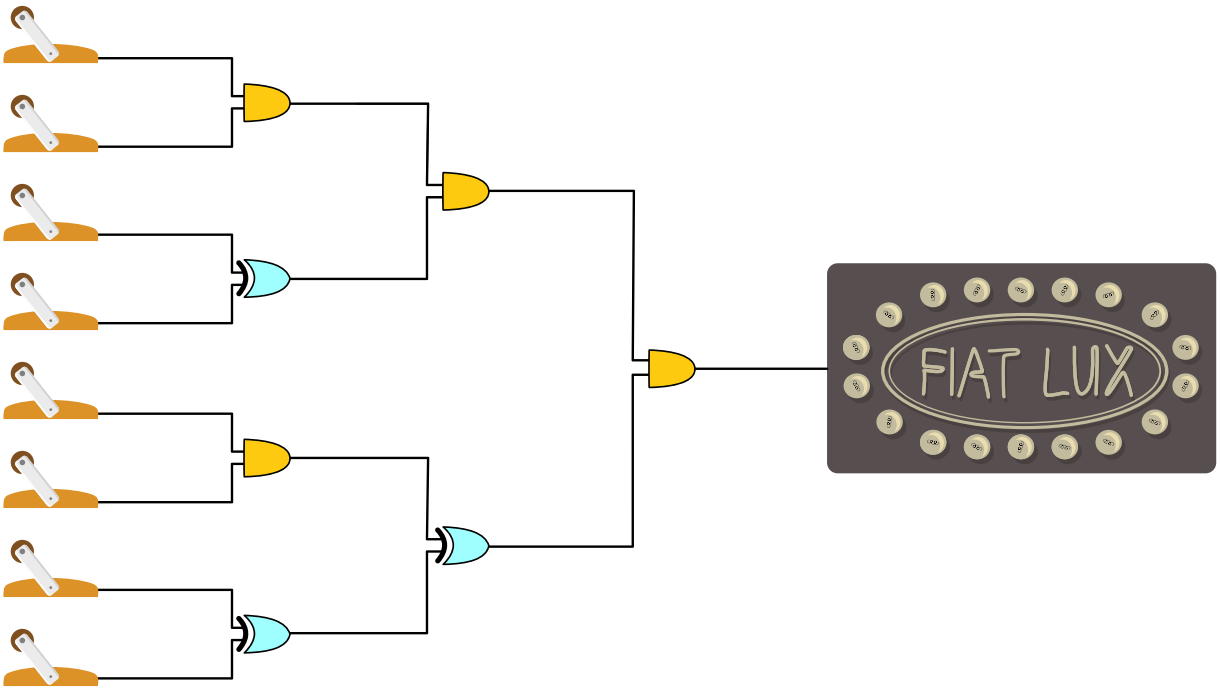
In informatica, un linguaggio è un insieme di stringhe formate secondo determinate regole. Un tipo semplice di linguaggio è il linguaggio libero dal contesto. Un esempio di linguaggio libero dal contesto è costituito da tutte le espressioni ben formate di parentesi. In un'espressione ben formata, ogni 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`

15. 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.

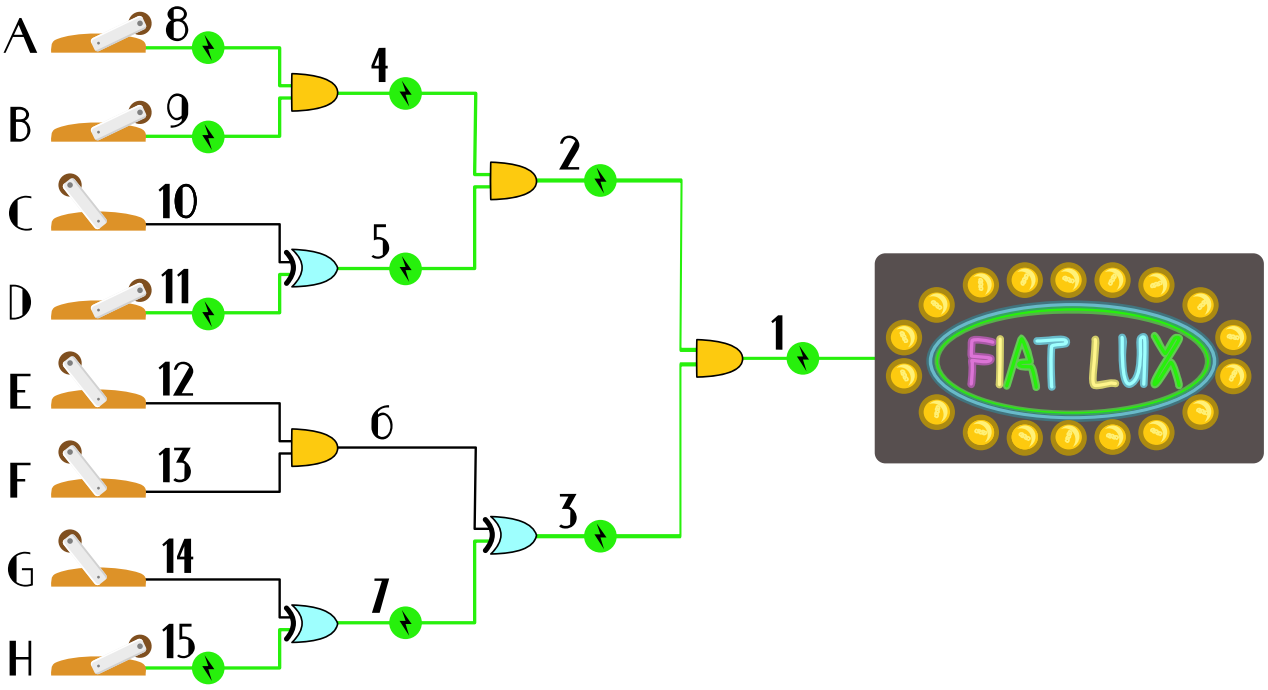


L'uscita del componente  è attiva solo quando entrambi i fili in ingresso sono attivi. L'uscita del componente  è attiva quando è attivo esattamente uno dei fili in ingresso.

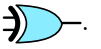
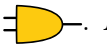
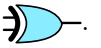
Quali interruttori  *devono essere attivati per accendere l'insegna al neon?*

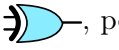
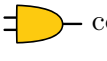
Soluzione

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:

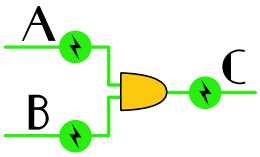
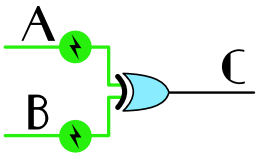
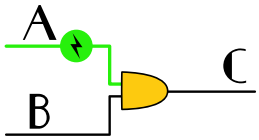
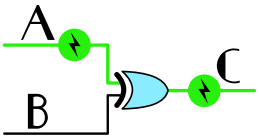
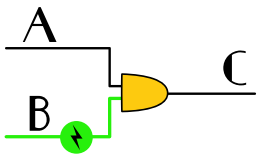
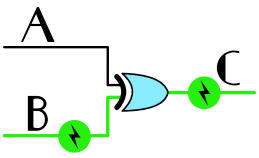
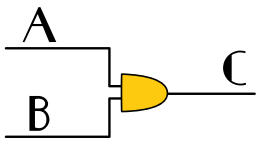
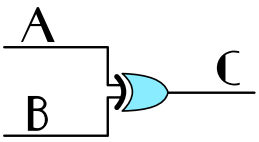
Interruttore								**Filo**	
A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**6**	**7**
sempre *ON*		**esattamente uno *ON***		**entrambi *ON*, se filo 6 è *ON*, altrimenti massimo uno *ON***		**esattament uno *ON*, se filo 7 è *ON*, altrimenti entrambi *ON* o *OFF***		**esattamente uno *ON***	
ON	ON	ON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

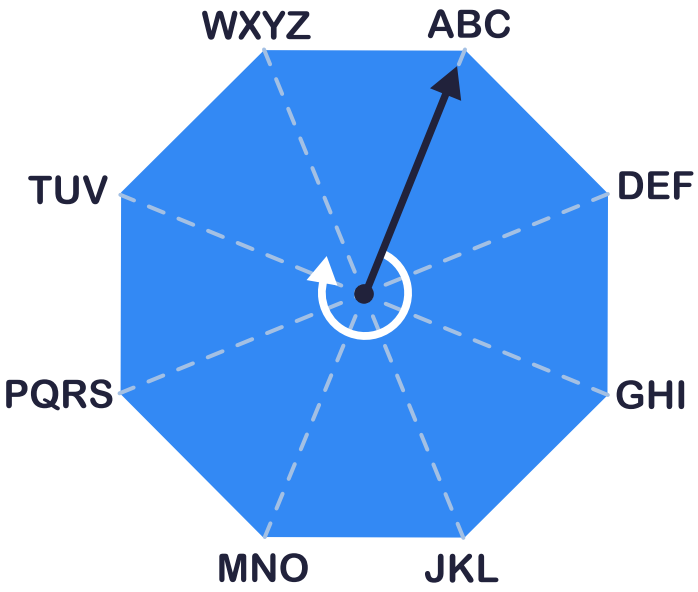
A un livello superiore, le porte logiche sono utilizzate anche nella programmazione: se l'esecuzione di una parte di un programma si basa su diverse condizioni, queste condizioni possono essere combinate con l'aiuto di operatori logici che funzionano esattamente nello stesso modo. Questo si riscontra 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`

16. 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

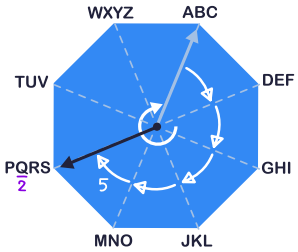
B) QUIZ

C) ROBOT

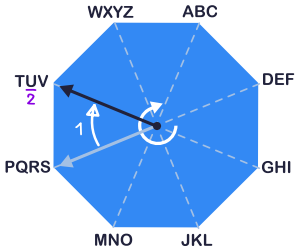
D) JAZZ

E) LUCE

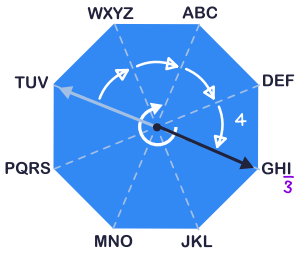
Soluzione



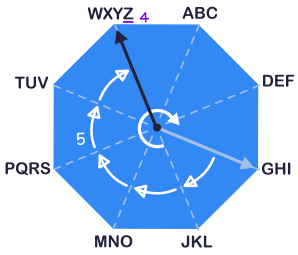
52 significa che il puntatore viene spostato dal blocco «ABC» al blocco «PQRS» (prima cifra 5) e che viene presa la seconda lettera «Q» (seconda cifra 2).



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.

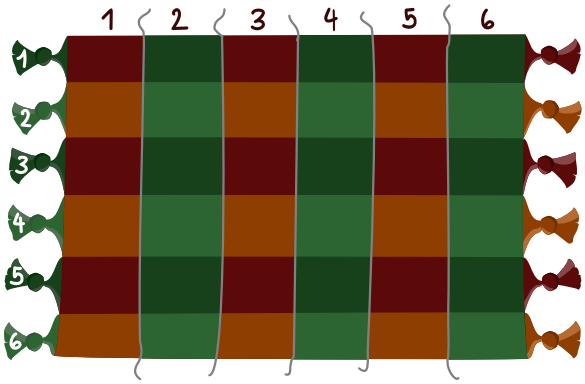
In questo caso, però, la decifrazione è semplicissima: esiste una sola chiave per criptare un testo. Anche se si potesse far partire la posizione iniziale del puntatore non dall'ABC ma da un qualche blocco, si avrebbero solo otto chiavi diverse... persino il codice Cesare, che ha più di 2000 anni, è «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

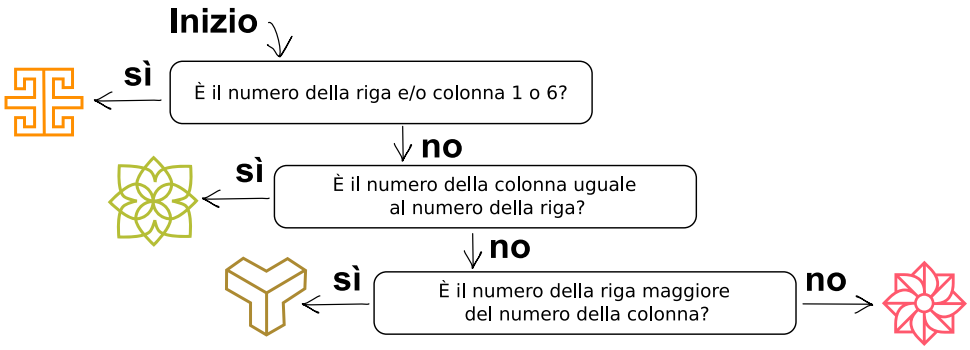
- Cifrario di Cesare: `https://it.wikipedia.org/wiki/Cifrario_di_Cesare`
- Cifrario polialfabetico: `https://it.wikipedia.org/wiki/Cifrario_polialfabetico`
- Cifrario: `https://it.wikipedia.org/wiki/Cifrario`
- Cifrario di Vigenère: `https://it.wikipedia.org/wiki/Cifrario_di_Vigenère`
- Crittografia 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`

17. Motivo del tappeto

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

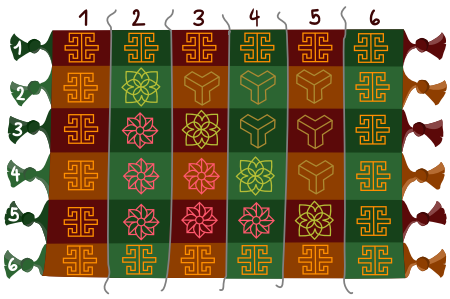


Hale numera le righe e le colonne. Quindi per ogni campo della griglia c'è il numero della riga e il numero della colonna. I commessi di Hale devono inserire un simbolo in ogni casella. Hale ha dato loro queste istruzioni per farlo:

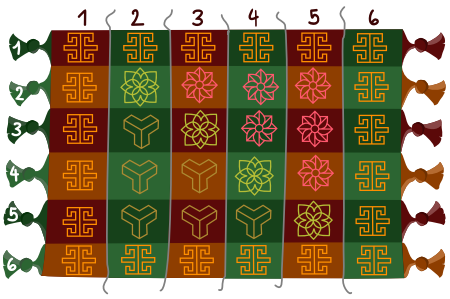


Come sarà il tappeto?

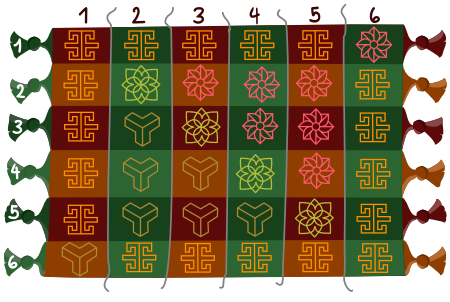
A)



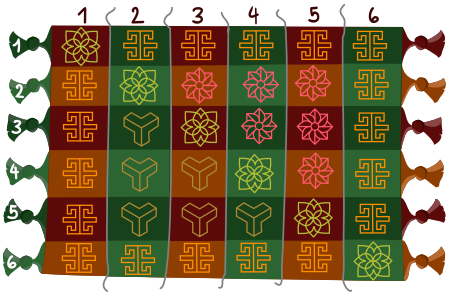
B)



C)



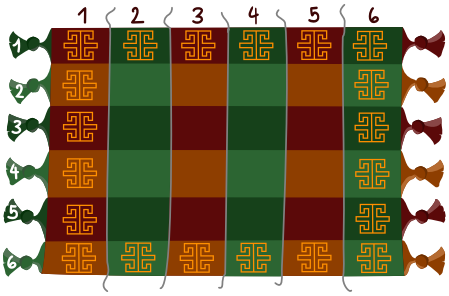
D)



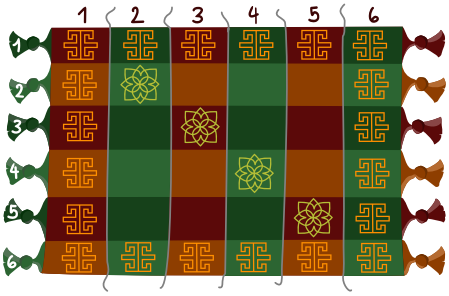
Soluzione

La risposta corretta è B).

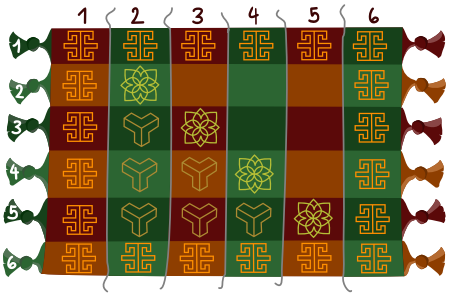
Alla prima domanda dell'immagine si risponde «sì» per tutti i quadrati sul bordo della griglia. Questo perché ogni campo del bordo si trova nella prima o nella sesta colonna o nella prima o nella sesta riga. A questi campi viene assegnato il simbolo  e si ottiene la seguente disposizione:



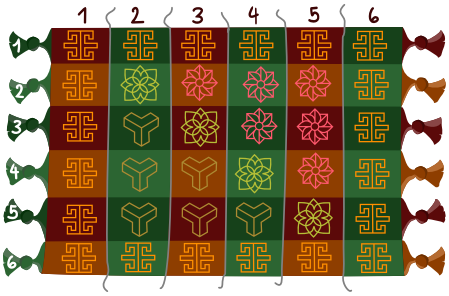
Alla seconda domanda si risponde con «sì» per tutti i campi sulla diagonale, perché sulla diagonale i numeri di colonna e di riga sono gli stessi. Questi campi ricevono il simbolo  e lo schema del tappeto si presenta come segue:



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

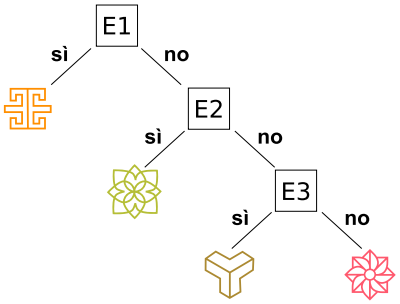


Per i campi rimanenti, alla terza domanda si risponde con «No». Ciò significa che il numero della riga non è maggiore del numero della colonna. Tutti questi campi sono riempiti con il simbolo  . In questo modo si ottiene il modello di tappeto della risposta B.



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

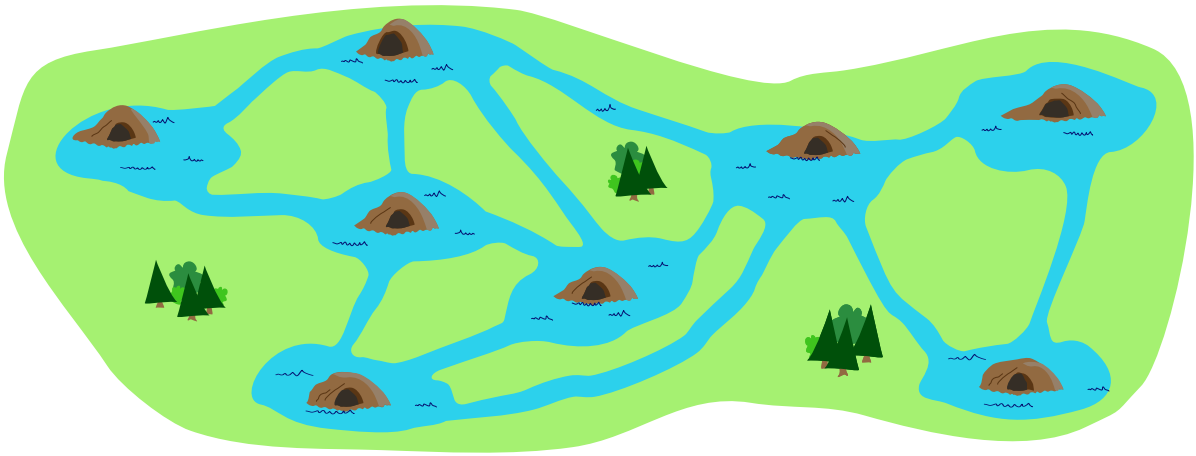
- Albero di decisione: `https://it.wikipedia.org/wiki/Albero_di_decisione`
- Algorithmo: `https://it.wikipedia.org/wiki/Algoritmo`
- Linguaggio di programmazione: `https://it.wikipedia.org/wiki/Linguaggio_di_programmazione`
- Programma: `https://it.wikipedia.org/wiki/Programma_(informatica)`

18. I vicini di Lili

Sulla mappa si possono vedere i castelli di otto castori. Due castori sono vicini di casa se un canale collega i loro castelli.

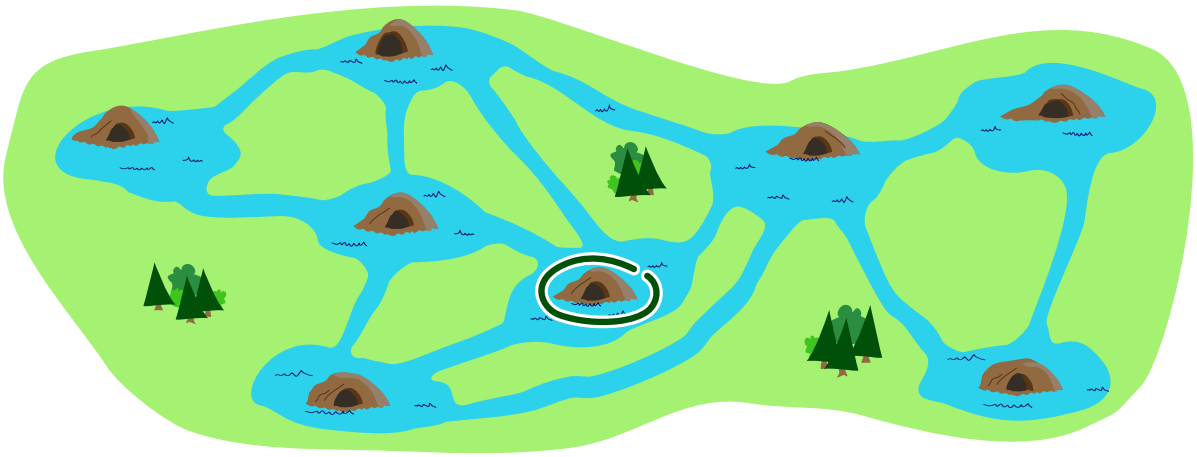
- Lili, Simon e Peter hanno ciascuno quattro vicini.
- Simon e Peter sono gli unici vicini di Nina.

In quale castello vive Lili?

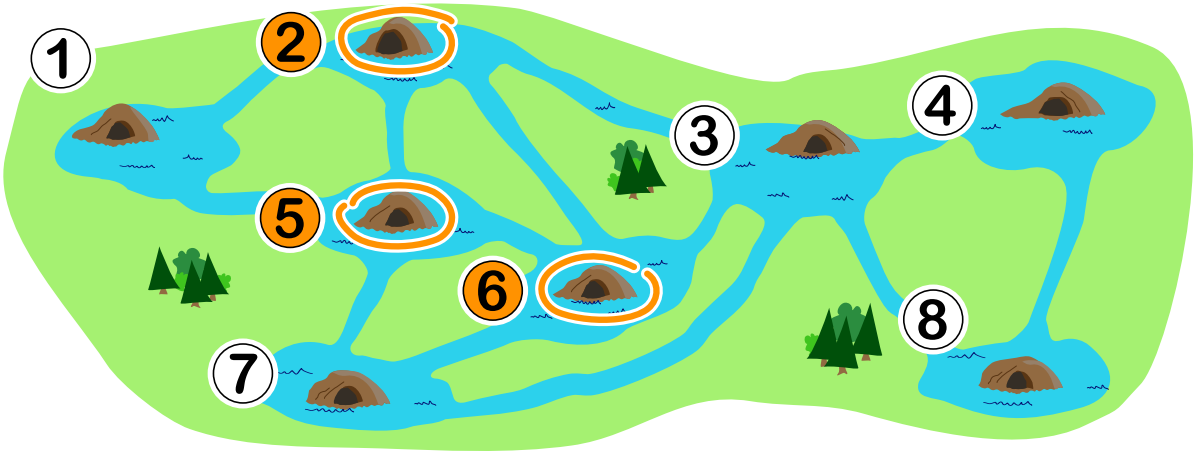


Soluzione

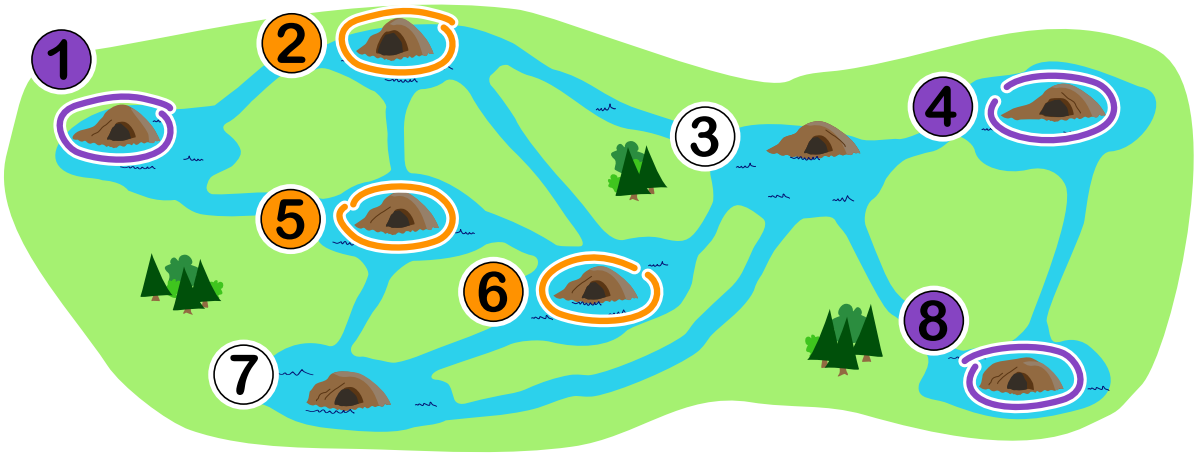
La soluzione corretta è:



Per risolvere il problema, è necessario concentrarsi sui canali tra i castelli. Dobbiamo identificare i castelli in cui vivono Lili, Peter o Simon. Poiché tutti hanno 4 vicini, ci devono essere esattamente quattro canali che partono da ciascuno dei loro castelli. Ci sono tre castelli di questo tipo: 2, 5 e 6.



Di conseguenza, Lili, Peter e Simon vivono ciascuno in uno di questi tre castelli. Ora dobbiamo scoprire in quale dei tre castelli vive Lili. Le altre due informazioni si riferiscono al castello di Nina. Da questi possiamo concludere che dal suo castello partono esattamente due canali. Quindi Nina vive in uno di questi castelli: 1, 4 o 8.



Siccome sappiamo che Simone e Pietro sono i due vicini di casa di Nina, possiamo concludere che

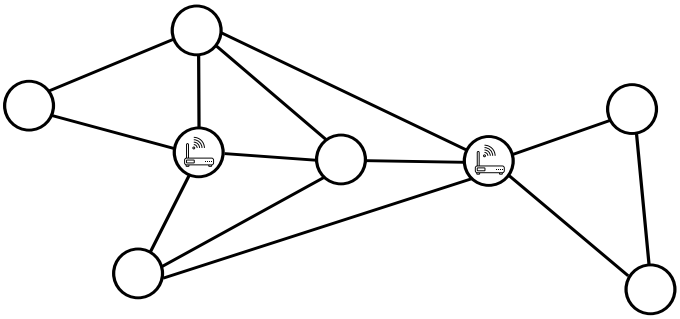
- Nina vive nel castello 1
- Simon e Peter vivono nei castelli 5 e 7 (o viceversa).

Quindi c'è solo un castello da cui partono quattro canali, che può essere il castello di Lili. È il castello 6!

Questa è l'informatica!

In questo compito, due castelli sono collegati da un canale. L'insieme dei castelli e dei canali forma una rete che mostra le relazioni tra tutti i castelli. Una tale rete di relazioni tra oggetti è chiamata *grafo* in informatica e matematica. Un grafo può essere considerato come un *insieme* di *vertici* collegati da *archi*. In questo compito, i castelli rappresentano i vertici e i canali gli archi.

Lo studio dei grafi è chiamato *teoria dei grafi*. Può essere utilizzato per modellare le relazioni a coppie tra gli oggetti. I grafi sono modelli matematici di strutture simili a reti in natura e in tecnologia. Ne sono un esempio le strutture sociali, le reti stradali, le reti informatiche, i circuiti elettrici, le reti di distribuzione o le molecole chimiche. I grafi possono essere utili per descrivere e risolvere i *problemi di rete*, come ad esempio trovare un buon posto per un router in un edificio o assicurarsi che ogni stanza di una casa abbia un segnale Wi-Fi forte.



Parole chiave e siti web

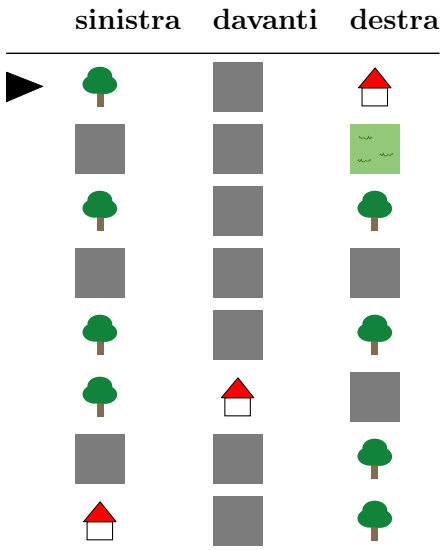
- Grafo: `https://it.wikipedia.org/wiki/Grafo`
- Insieme: `https://it.wikipedia.org/wiki/Insieme`
- Vertice: `https://it.wikipedia.org/wiki/Vertice_(teoria_dei_grafi)`
- Arco: `https://it.wikipedia.org/wiki/Glossario_di_teoria_dei_grafi#Arco`

19. La posta robotizzata

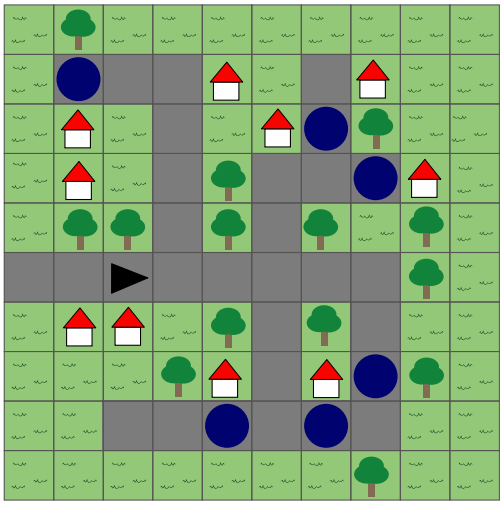
Il robot Tina consegna la posta. Tina utilizza una mappa suddivisa in campi. Tina si sposta lungo la strada verso una strada adiacente a sinistra, a destra o davanti (cioè non in diagonale).

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.

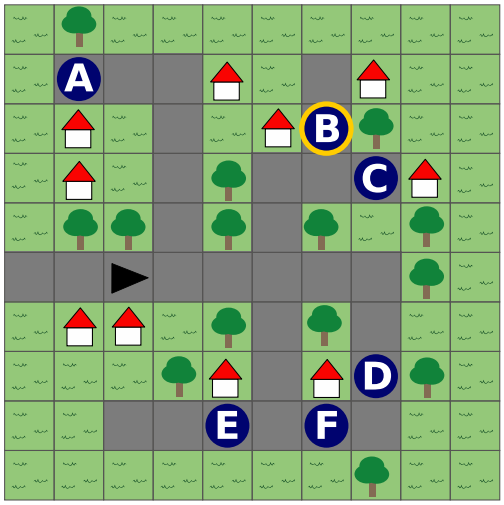


Quale dei punti blu scuro ● Tina raggiungerà alla fine del suo percorso?



Soluzione

La risposta corretta è il punto B.

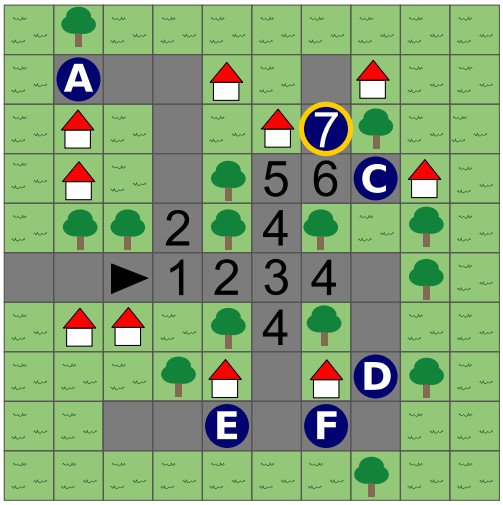


Passo	sinistra	davanti	destra
▶	🌳	■	🏠
1	■	■	
2	🌳	■	🌳
3	■	■	■
4	🌳	■	🌳
5	🌳	🏠	■
6	■	■	🌳
7	■	🌳	🌳

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.

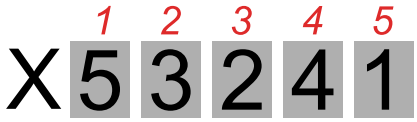
Le auto a guida autonoma sono esempi famosi di questi robot. Sono dotati di numerosi sensori che non solo misurano la velocità o la posizione corrente, ma anche la distanza dal ciglio della strada e rilevano gli oggetti presenti sulla strada o a bordo strada e molto altro ancora. Queste informazioni 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`

20. Sequenze di dati

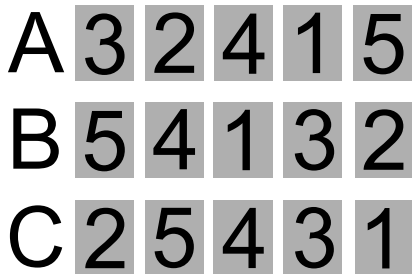
Qui possiamo vedere una sequenza di numeri di nome X. Nelle posizioni da 1 a 5 della sequenza X si trovano i seguenti numeri: 5, 3, 2, 4, 1.



Descriviamo il numero in una certa posizione mettendo tra parentesi il nome e la posizione. Un esempio: descriviamo il numero in posizione 2 della sequenza X in questo modo: (X 2). Attualmente, (X 2) = 3.

Un numero nella sequenza così descritta può essere esso stesso una posizione. Ad esempio, (X (X 2)) = (Xv3) = 2.

Ecco altre tre sequenze: A, B e C.



Quale numero descriviamo in questo modo: (A (B (C 3))) ?

A) 1

B) 2

C) 3

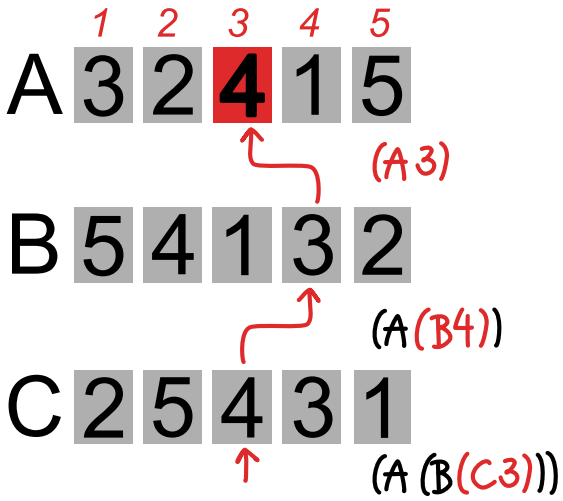
D) 4

E) 5

Soluzione

La risposta corretta è D) 4.

La descrizione (A (B (C 3)) dice: il numero descritto si trova nella sequenza A alla posizione (B (C 3)); la posizione si trova quindi nella sequenza B alla posizione (C 3); e questa posizione si trova a sua volta nella sequenza C alla posizione 3. Complicato!



È più facile se valutiamo la descrizione «dall'interno verso l'esterno», come con un'espressione aritmetica - e come è già stato dimostrato nell'esempio del compito: (A (B (C 3))) = (A (B 4)) = (A 3) = 4

Questa è l'informatica!

Non molto tempo fa si parlava di *elaborazione dei dati* quando si parlava dell'uso dei computer. Giustamente, perché i computer elaborano tutti i tipi di dati, come numeri, testi, immagini, suoni, ecc. La maggior parte dei dati interessanti memorizzati nei computer sono di natura complessa e hanno una struttura: le temperature misurate nel corso della giornata in una stazione meteorologica, ad esempio, possono essere memorizzate come una sequenza di coppie, ciascuna composta dall'ora della misurazione e dalla temperatura misurata. Quindi c'è una struttura a coppie e una struttura a sequenze.

I dati possono avere un'ampia varietà di strutture e per questo gli informatici hanno sviluppato un'ampia varietà di cosiddette *strutture di dati* per memorizzare i dati in modo intelligente e (altrettanto importante) per accedere ai dati in modo efficiente. Una semplice struttura di dati è l'*array*, che svolge il ruolo principale in questo compito. Un array può memorizzare un numero fisso di dati (compresi i numeri) in posizioni successive. A causa delle posizioni, i dati nell'array hanno una struttura ordinata - un array sarebbe quindi adatto per le coppie tempo/temperatura menzionate sopra. A causa della loro dimensione fissa, gli array appartengono alle strutture dati *statiche* dell'informatica. Per le sequenze di dati, esistono anche strutture di dati *dinamiche* come le liste, la cui dimensione può cambiare a seconda delle necessità.

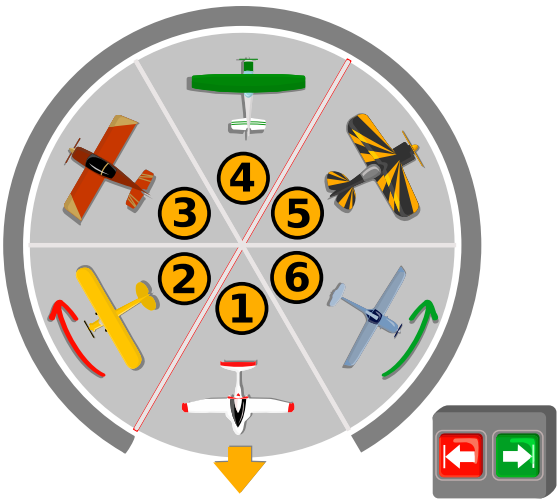
Statico o dinamico: se una struttura di dati di una sequenza contiene numeri, questi numeri possono anche indicare posizioni nella stessa o in un'altra sequenza. Questo viene spesso utilizzato in informatica per il cosiddetto indirizzamento indiretto: L'indirizzo o la posizione in una sequenza non è specificato direttamente come numero, ma indirettamente da un valore (numerico) di una sequenza, che a sua volta può essere indirizzata di nuovo indirettamente - e così via.

Parole chiave e siti web

- Elaborazione dati: `https://it.wikipedia.org/wiki/Elaborazione_dati`
- Struttura dati: `https://it.wikipedia.org/wiki/Struttura_dati`
- Array: `https://it.wikipedia.org/wiki/Array`
- Metodo di indirizzamento: `https://it.wikipedia.org/wiki/Metodo_di_indirizzamento`

21. Capannone rotante

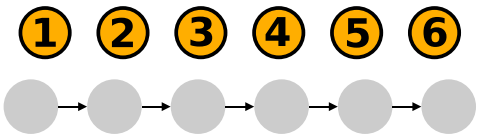
Nel campo di aviazione di Beavertown, sei aerei sono parcheggiati in un capannone. Sono su una piattaforma rotante, parcheggiati in sei posizioni diverse. All'esterno sono presenti due tasti freccia ⬅️ ➡️. Con un solo tasto è possibile ruotare l'unità di rotazione esattamente di una posizione di parcheggio a sinistra o a destra.



Al mattino, quando i piloti ritirano i loro aerei, la posizione di parcheggio 1 è sempre sulla porta del capannone e l'aereo su di essa può uscire. Nel migliore dei casi, i tasti freccia devono essere premuti altre cinque volte, in modo che anche tutti gli altri aerei possano uscire. Ad esempio, se i piloti vogliono accedere alle posizioni di parcheggio nell'ordine 1, 6, 5, 4, 3, 2, è sufficiente premere cinque volte il tasto ➡️.

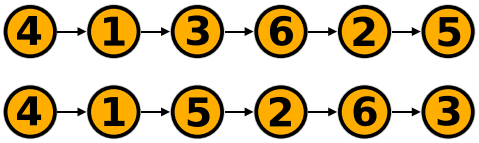
Ma qual è il caso peggiore? In quale ordine devono essere premuti più spesso i tasti?

Fornisci un esempio di un ordine di questo tipo.



Soluzione

Ci sono due risposte corrette:



Per trovare la soluzione, viene sempre selezionato l'aereo che si trova nella posizione di parcheggio con la distanza maggiore dalla porta del capannone.

«➡ ➡ ➡ 4» significa che dopo aver premuto tre tasti l'aeromobile si parcheggerà alla posizione 4

4 1 3 6 2 5:

4 1 5 2 6 3:

In entrambi i casi, sono necessari un totale di 16 passi.

I passi non possono essere più di 16, perché solo all'inizio possono susseguirsi due passaggi con tre pressioni dei tasti freccia. Dopodiché, si possono alternare al massimo due e tre passi.

Questa è l'informatica!

Il capannone rotante ha il vantaggio di poter parcheggiare gli aerei in modo molto poco ingombrante. Tuttavia, la raccolta di solito richiede più tempo rispetto a un normale capannone.

L'*efficienza* di una procedura è un argomento centrale in informatica perché è un importante criterio di valutazione per gli *algoritmi*. Molto spesso l'efficienza riguarda il tempo di esecuzione, ma non è sempre così. Nella definizione generale di efficienza di un algoritmo, si tratta di tutte le risorse necessarie, quindi anche, ad esempio, della dimensione della memoria necessaria (*efficienza della memoria*).

Come nell'esempio del capannone, il risparmio di una risorsa porta a un aumento della domanda di un'altra risorsa. Dipende dal contesto specifico e dalla disponibilità delle risorse a quale risorsa viene data maggiore importanza.

Ad esempio, *bubblesort* e *timsort* sono entrambi algoritmi per ordinare un elenco di elementi. Bubblesort ordina l'elenco in un tempo proporzionale al numero di elementi al quadrato ($\mathcal{O}(n^2)$), ma richiede poca memoria aggiuntiva, costante rispetto alla lunghezza dell'elenco.

Timsort ordina molto più velocemente di bubblesort ($\mathcal{O}(n \log n)$), ma ha un requisito di spazio che aumenta linearmente con la dimensione dell'elenco. Se per una particolare applicazione è necessario

ordinare ad alta velocità elenchi di grandi dimensioni, Timsort è la scelta migliore; se invece è più importante ridurre al minimo i requisiti di memoria dell'ordinamento, Bubblesort è la scelta migliore.

Parole chiave e siti web

- Algoritmo: `https://it.wikipedia.org/wiki/Algoritmo`
- Bubblesort: `https://it.wikipedia.org/wiki/Bubble_sort`
- Timsort: `https://it.wikipedia.org/wiki/Timsort`
- O grande: `https://it.wikipedia.org/wiki/O-grande`

22. Serata cinematografica

Alcuni amici vogliono vedere un film insieme. È possibile scegliere tra sette film. Per prendere una decisione, ogni persona valuta ogni film come bello, così così o brutto.

I risultati sono visibili qui sotto. Purtroppo non c'è un vincitore, o un film «preferito», per la serata cinematografica.

Un film è un «preferito» se ogni persona ha dato a quel film la sua valutazione migliore. Ad esempio, il film 1 non è il preferito perché Niklaus ha dato il suo voto migliore a un altro film, il film 4.

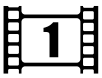
Ora Ada vuole convincere il minor numero possibile di amici a cambiare la propria valutazione, in modo che alla fine ci sia un preferito.

Aiuta Ada e modifica il minor numero possibile di valutazioni in modo che ci sia un preferito.

	1	2	3	4	5	6	7
Ada	🙂	🙂	🙂	🙂	🙂	🙂	🙂
Nancy	😐	🙂	🙂	😐	😐	🙂	🙂
Niklaus	☹	☹	☹	😐	☹	☹	☹
Grace	☹	😐	😐	😐	☹	😐	☹
Edsger	🙂	😐	☹	☹	😐	🙂	🙂
Rozsa	😐	☹	😐	☹	🙂	😐	😐

Soluzione

All'inizio non c'è un film preferito. Per ogni film troviamo amici che valutano meglio altri film.

Film	Amici che valutano meglio altri film
	4: Nancy, Niklaus, Grace e Rozsa
	3: Niklaus, Edsger e Rozsa
	3: Niklaus, Edsger e Rozsa
	3: Nancy, Edsger e Rozsa
	3: Nancy, Grace e Edsger
	2: Niklaus e Rozsa
	3: Niklaus, Grace e Rozsa

Per il film 6, ci sono solo due amici che valutano meglio altri film. Per tutti gli altri film, ce ne sono più di due. Se un solo amico modifica una valutazione, non è possibile creare un preferito. Ada deve quindi convincere Niklaus e Rozsa a migliorare il loro voto per il film 6. Pertanto, Ada ha creato un preferito con due modifiche.

Il miglioramento delle valutazioni è una delle possibili strategie. Niklaus e Rozsa potrebbero ancora decidere di abbassare alcune valutazioni: se Niklaus peggiora la sua valutazione per il film 4 e Rozsa peggiora la sua valutazione per il film 5, il film 6 diventa il preferito. Anche in questo caso sono necessarie due modifiche.

È abbastanza plausibile che Rozsa peggiori la sua valutazione per il film 5 e che Niklaus migliori la sua valutazione per il film 6. Allo stesso modo, Rzosa potrebbe migliorare la sua valutazione per il film 6 e Niklaus potrebbe peggiorare la sua valutazione per il film 4. In entrambi gli scenari, il film 6 diventa il preferito. In entrambi i casi, sono sufficienti due modifiche.

In totale, quindi, ci sono quattro modi per modificare solo due valutazioni, in modo da avere un preferito.

Questa è l'informatica!

Come possiamo risolvere questo compito? Un'idea è quella di verificare per ogni film e persona singolarmente se quella persona ha valutato altri film meglio o peggio. Nel nostro caso, si ottiene la tabella qui sopra. Questa tabella ci aiuta a capire quali persone devono modificare le loro valutazioni, in modo da arrivare a un preferito con il minor numero possibile di modifiche.

Ada può effettivamente usare questo *algoritmo* per risolvere il suo problema.

Tuttavia, questo algoritmo è *efficiente*? Ada potrebbe essere ancora più veloce?

Di seguito indichiamo il numero di film con M e il numero di amici con F. Ada deve considerare singolarmente tutte le $M \times F$ valutazioni e per ogni valutazione deve considerare tutte le altre $M - 1$ valutazioni della stessa persona. In totale, Ada deve considerare $M \times (M - 1) \times F$ valutazioni.

Per scoprire se una delle valutazioni è problematica, Ada deve solo confrontare quella valutazione con la migliore che quella persona ha dato. Se quella persona pensa che un altro film sia migliore, allora il film che Ada ha appena guardato potrebbe non essere affatto il suo preferito.

In altre parole, se Ada scopre prima le migliori valutazioni complessive per ogni persona (esaminando tutte le $M \times F$ valutazioni), può determinare per tutte le $M \times F$ valutazioni se sono peggiori della migliore valutazione di quella persona.

Nel complesso, questo algoritmo alternativo con un pre-calcolo mirato delle migliori valutazioni porta Ada a considerare le valutazioni di $2 \times M \times F$. Con M = 7 e F = 6, si tratta di 84 accessi alla tabella, mentre il primo algoritmo richiede 252 accessi alla tabella. Anche il secondo algoritmo risolve correttamente il problema di Ada, ma è più efficiente del primo.

Uno dei compiti più importanti dell'informatica è quello di risolvere i problemi non solo in modo corretto, ma anche nel modo più efficiente possibile. Con i computer più veloci le soluzioni vengono calcolate più rapidamente. Tuttavia, se non si conoscono algoritmi efficienti per risolvere un problema, anche i computer più veloci possono raggiungere i loro limiti.

Parole chiave e siti web

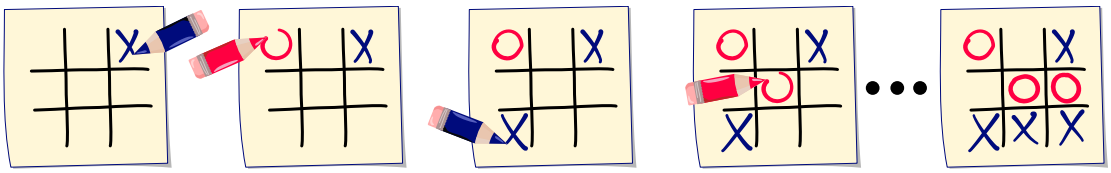
- Algoritmo: `https://it.wikipedia.org/wiki/Algoritmo`

23. Tris

Il tris è un gioco per due persone.

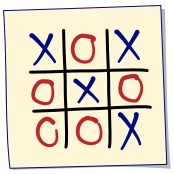
In una griglia con 3×3 caselle, i due giocatori riempiono a turno un simbolo in una casella vuota: un giocatore X, l'altro O. Il primo giocatore che riempie tre caselle in fila, in colonna o in diagonale con il proprio simbolo vince e la partita è finita. Se tutte le caselle sono riempite e nessuno ha vinto, la partita termina con un pareggio.

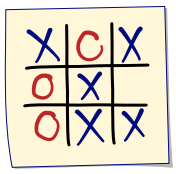
Qui si possono vedere i punteggi di una possibile partita: le prime 4 mosse e l'ultima mossa. Il giocatore con X vince.

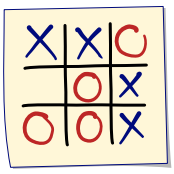


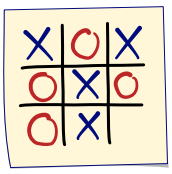
Il punteggio alla fine di una partita è chiamato punteggio finale. Le regole del gioco specificano esattamente come possono essere compilati i campi con X e O e quando il gioco termina.

Solo una delle quattro immagini mostra un punteggio finale di tris. Quale?

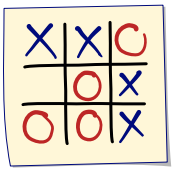
A) 

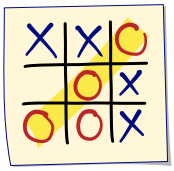
B) 

C) 

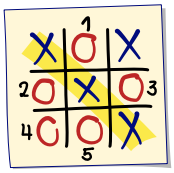
D) 

Soluzione

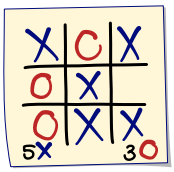
La risposta C è corretta: 



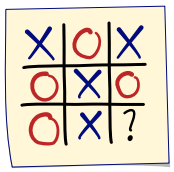
La risposta C è corretta perché un giocatore aveva vinto (tre O in una diagonale) e poi non sono state fatte altre mosse.



La risposta A non è corretta. Il giocatore X ha vinto la partita, ma il giocatore O ha riempito più caselle. Poiché il vincitore riempie sempre l'ultimo campo, non potrà mai avere meno caratteri del perdente.



La risposta B non è corretta perché 5 campi sono riempiti con X ma solo 3 campi con O. Questo non è possibile, perché il numero di caratteri O- e il numero di caratteri X- possono differire al massimo di 1.



La risposta D non è corretta, perché non mostra un punteggio finale. Non c'è ancora un vincitore e i campi non sono completamente riempiti.

Questa è l'informatica!

Nel risolvere il compito, abbiamo verificato se le quattro immagini delle opzioni di risposta documentano una posizione finale valida. Dalle regole del gioco del tris si possono ricavare nuove regole sulle posizioni finali valide, ad esempio queste:

1. La differenza tra il numero di X e il numero di O deve essere pari a 0, –1 o 1.
2. Se nessun giocatore ha vinto, tutte le caselle devono essere riempite.
3. Il perdente può compilare al massimo tanti campi quanti ne ha compilati il vincitore.
4. Nel documento di un gioco finito, può esserci al massimo una sequenza di tre caratteri uguali.

Queste nuove regole non sono regole del gioco, ma servono solo a verificare se la griglia completata è un punteggio finale. Se un'immagine è in conflitto con una di queste regole, non può costituire un punteggio finale.

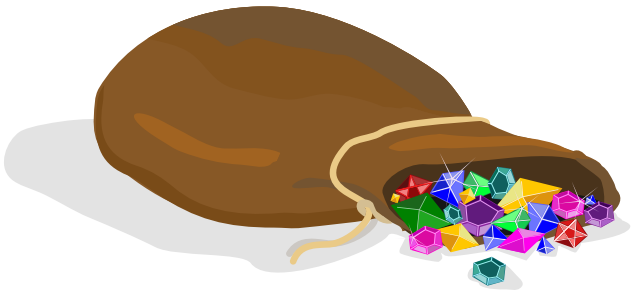
Le regole sono molto importanti nella tecnologia informatica. Un interprete che esegue un programma controlla se il testo inserito è conforme alle regole di sintassi del linguaggio di programmazione.

Nella programmazione, le regole vengono utilizzate nelle cosiddette assicurazioni per verificare la correttezza di un programma durante la sua esecuzione.

Parole chiave e siti web

- Tris: `https://it.wikipedia.org/wiki/Tris_(gioco)`
- Interprete: `https://it.wikipedia.org/wiki/Interprete_(informatica)`
- Linguaggio di programmazione:
  `https://it.wikipedia.org/wiki/Linguaggio_di_programmazione`

24. 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.

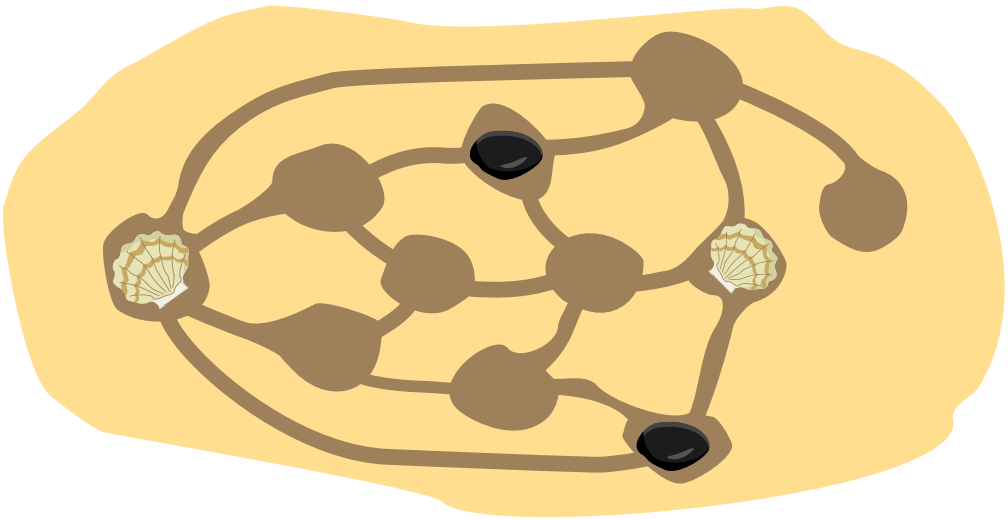
Parole chiave e siti web

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

25. Ciottoli e conchiglie

Ann e Bob giocano sulla spiaggia. Scavano delle cavità e ne collegano alcune con solchi disegnati sulla sabbia. Le pedine di Ann sono conchiglie. Quelle di Bob sono ciottoli.

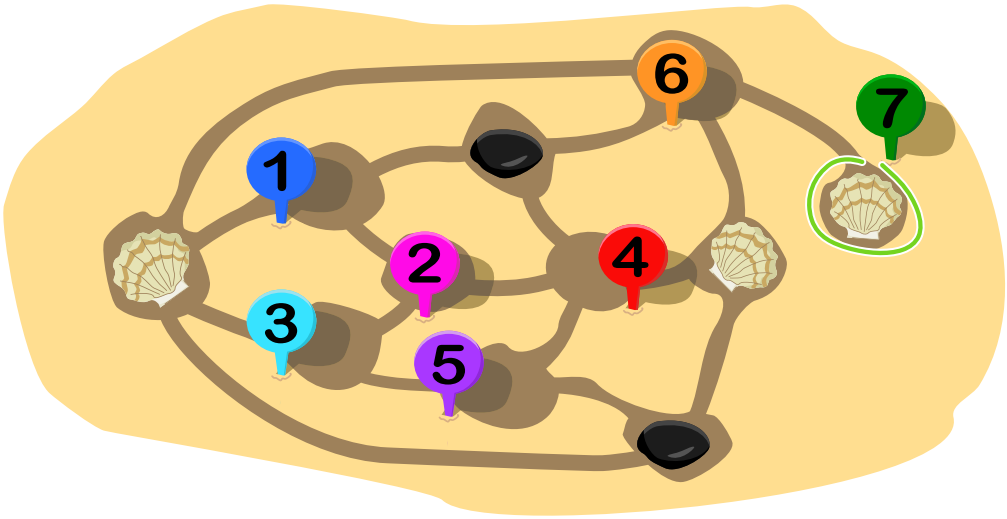
A turno, i giocatori posizionano uno delle loro pedine in uno spazio libero. Il primo giocatore che posiziona due proprie pedine in due cavità direttamente collegate perde. L'immagine mostra il punteggio dopo alcune mosse.



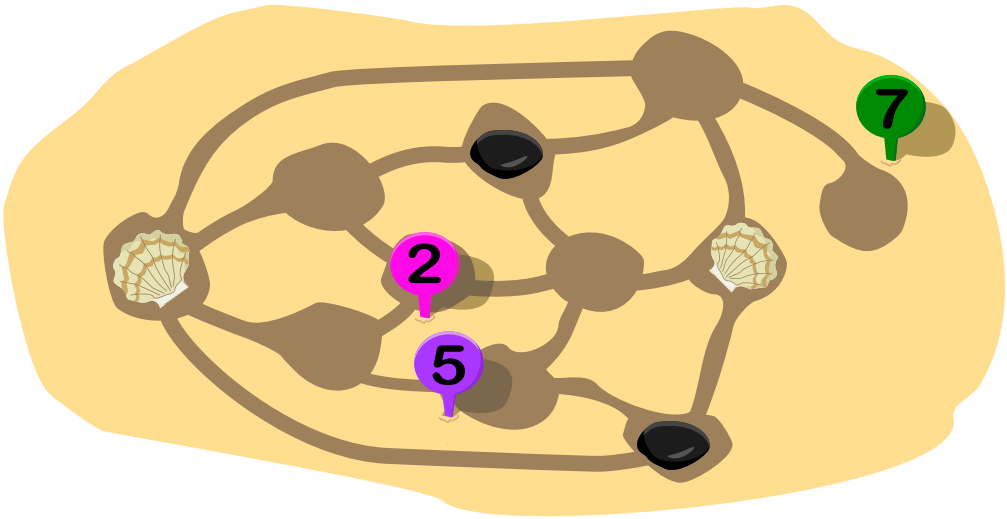
È il turno di Ann. In quale delle cavità libere deve posizionare la sua prossima conchiglia per assicurarsi la vittoria?

Soluzione

La risposta corretta è la cavità 7.



È il turno di Ann. Per lei, le cavità 1, 3, 4 e 6 sono fuori discussione, quindi restano la 2, la 5 e la 7.



Vede che per Bob le cavità 1, 4, 5 e 6 sono fuori discussione. Quindi per lui rimangono 2, 3 e 7.

Se Ann gioca 7, Bob può giocare 2 o 3; in entrambi i casi Ann può comunque giocare 5 e Bob perde.

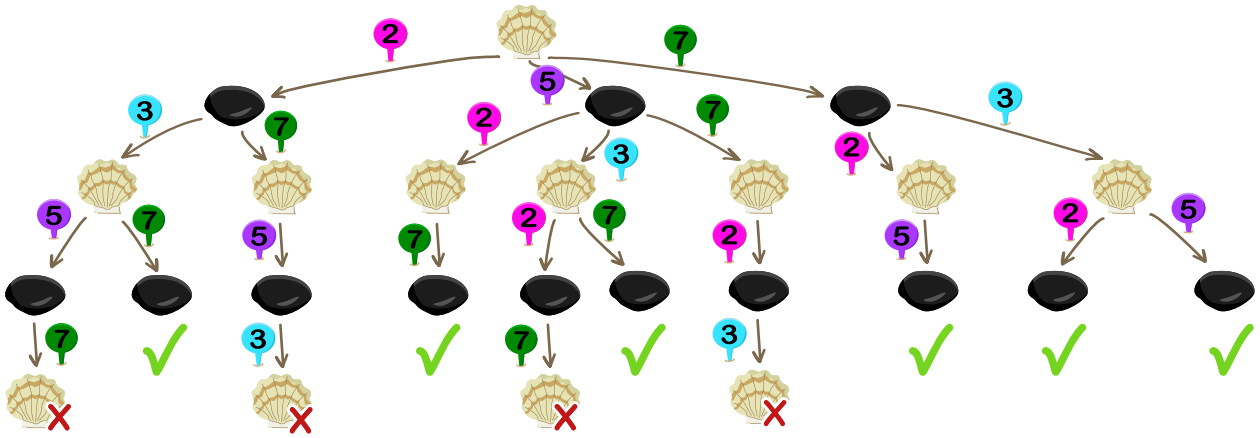
Se Ann giocasse 2 al punteggio della figura, Bob potrebbe giocare 7 al prossimo colpo. Dopo di che, Ann avrebbe dovuto giocare 5, Bob avrebbe dovuto giocare 3 e Ann avrebbe perso.

Se Ann giocasse 5, Bob potrebbe giocare 7, Ann dovrebbe giocare 2, Bob giocherebbe 3 e di nuovo Ann perderebbe.

Tra l'altro, Bob non potrebbe vincere nemmeno se fosse il suo turno al punteggio nella foto.

Questa è l'informatica!

Per visualizzare sistematicamente le possibili mosse di Ann e Bob, si può utilizzare un cosiddetto albero di gioco:



In questo albero di gioco si può vedere con quale mossa Ann può assicurarsi la vittoria: nel ramo di destra, che inizia con Ann che gioca 7, sono possibili solo situazioni in cui vince. Nella cosiddetta *teoria dei giochi*, un campo speciale della matematica, si considerano le affermazioni sull'esito dei giochi in cui interagiscono due o più giocatori. L'informatica si occupa di algoritmi per la valutazione di tali alberi di gioco. I computer con una potenza di calcolo sufficiente possono già competere con gli esseri umani in giochi come gli scacchi e vincere. Tuttavia, la teoria dei giochi fornisce anche alla psicologia, all'economia e ad altre materie modelli per sistemi complessi in cui i «giocatori» interagiscono, ad esempio per il comportamento di acquisto dei clienti quando i prezzi cambiano o per la selezione del percorso nel traffico stradale.

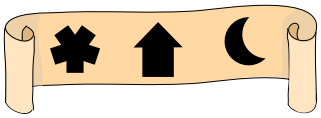
Il gioco di Ann e Bob è un'istanza di «COL». Si tratta di un gioco per due giocatori introdotto da Colin Vout e descritto nel noto libro «On Numbers and Games» del matematico John Horton Conway.

Parole chiave e siti web

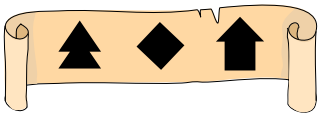
- Teoria dei giochi: `https://it.wikipedia.org/wiki/Teoria_dei_giochi`
- John Horton Conway: `https://it.wikipedia.org/wiki/John_Conway`

26. Maria alla caccia del tesoro

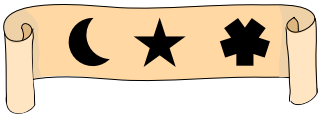
Maria trova una scatola misteriosa. Purtroppo la scatola è chiusa a chiave. Per aprirla, Maria deve scoprire la «chiave»: la giusta combinazione di tre simboli. Per fortuna, accanto alla scatola trova anche gli indizi di alcune combinazioni sbagliate:



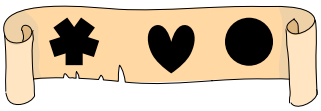
1) Uno dei simboli fa parte della chiave e si trova nella posizione giusta.



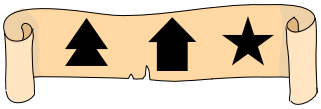
2) Nessuno dei simboli fa parte della chiave.



3) Due simboli fanno parte della chiave. Ma entrambi sono nella posizione sbagliata.



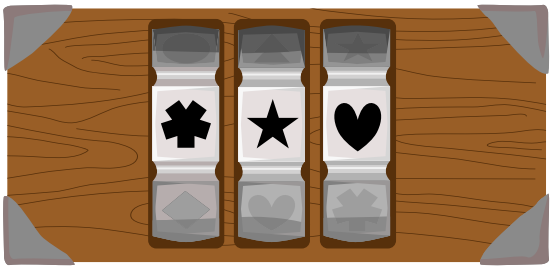
4) Un simbolo fa parte della chiave. Ma questo simbolo è nella posizione sbagliata.



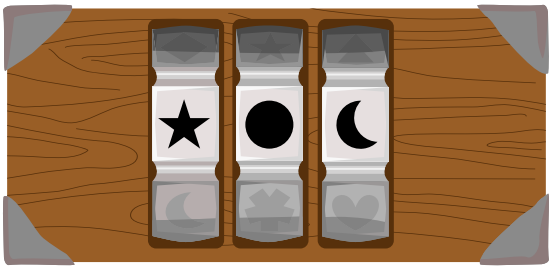
5) Un simbolo fa parte della chiave. Ma è nella posizione sbagliata.

Una delle seguenti combinazioni è la chiave della scatola. Quale?

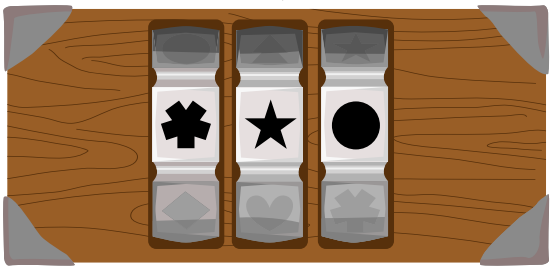
A)



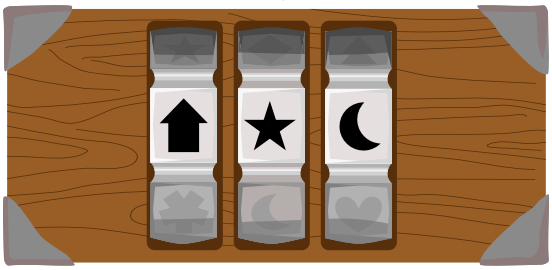
B)



C)



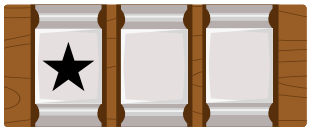
D)



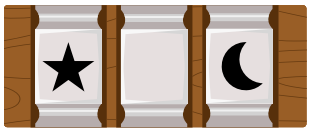
Soluzione

La risposta corretta è B).

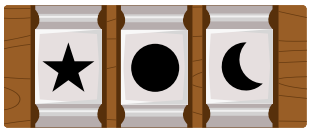
Cominciamo con l'individuare i simboli che possono essere presenti nella chiave. Dopo l'indizio 2) possiamo eliminare i simboli che non possono far parte della chiave: l'abete ▲, il diamante ◆ e la casa ⌂. Il suggerimento 5) indica che un simbolo fa parte della chiave ma si trova nella posizione sbagliata. Poiché l'abete ▲ e la casa ⌂ non possono essere presenti nella chiave, sappiamo che la stella ★ fa parte della chiave ma è nella posizione sbagliata. La nota 3) esclude che la stella ★ possa trovarsi al centro. In questo modo conosciamo la posizione finale della stella:



Poiché esiste una sola risposta possibile, che inizia con la stella, abbiamo già trovato la chiave. Per convincerci, continuiamo a cercare i due simboli mancanti. L'indizio 1) mostra che un simbolo compare nella chiave ed è già nella posizione giusta. La casa `house` e la prima posizione potrebbero già essere escluse. Pertanto, sappiamo che la luna si trova nella posizione corretta. Il risultato è la seguente immagine:



La nota 4) indica che un simbolo fa parte della chiave ma si trova nella posizione sbagliata. Possiamo escludere il simbolo ♣. Inoltre, solo lo spazio centrale è ancora libero. Pertanto, anche il cuore ♥ non può far parte della chiave. Ne consegue che il cerchio assume la posizione centrale.



Il risultato corretto può essere determinato anche in un altro modo. Tuttavia, ogni possibilità porta allo stesso risultato.

Questa è l'informatica!

Questo compito può essere risolto logicamente, ad esempio con l'aiuto del «metodo dell'esclusione». Nel nostro caso, siamo partiti dall'indizio 2) e abbiamo escluso tre simboli, che ci hanno portato rapidamente alla chiave. Le priorità dell'indizio 2) potrebbero essere viste come strategie mentali, regole o scorciatoie che ci hanno aiutato a prendere una decisione con conoscenze e tempo limitati. In informatica, tali regole sono chiamate *euristiche*, che possono anche essere programmate e automatizzate.

Ogni giorno prendiamo molte piccole decisioni basate su indizi o sulla necessità di comprendere vari *vincoli* di un problema per risolverlo. In questo compito, abbiamo seguito gli indizi e risolto il problema passo dopo passo per aprire la scatola.

Come potrebbe un computer risolvere questo problema? Questi otto simboli possono essere disposti in tre posizioni in un totale di 336 modi. Un computer le proverebbe tutte. In informatica si parla di *ricerca completa*. Una ricerca completa (chiamata anche *brute force* o *recursive backtracking*) è un metodo di risoluzione di un problema in cui viene attraversato l'intero spazio di ricerca. A noi questa soluzione può sembrare molto *inefficiente*, perché avremmo bisogno di molto tempo per provare tutte le possibilità (e dimenticare quelle già provate). Tuttavia, un computer può risolvere tali compiti in modo molto rapido e quindi efficiente. I simboli dell'esempio potrebbero anche rappresentare una password. Inoltre, la password deve essere sempre scelta in modo che contenga il maggior numero possibile di caratteri diversi, in modo che anche una ricerca completa non produca la chiave in un tempo ragionevole.

Iniziare con il suggerimento 2), quindi minimizzare le possibili soluzioni, è chiamato in informatica *backtracking*. Ad ogni *vertice* di un *albero*, vengono eliminate le possibilità che ovviamente non possono verificarsi nella chiave. In questo modo, a ogni profondità dell'albero, le possibilità si riducono.

Parole chiave e siti web

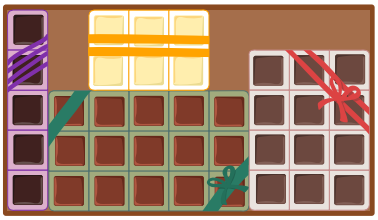
- Euristica: `https://it.wikipedia.org/wiki/Euristica`
- Vertice: `https://it.wikipedia.org/wiki/Vertice_(teoria_dei_grafi)`
- Albero: `https://it.wikipedia.org/wiki/Albero_(grafo)`

27. Incarto di cioccolatini

La fabbrica di cioccolato «Castocolat» invia quattro scatole di cioccolatini a ciascuno dei suoi clienti per una campagna pubblicitaria.

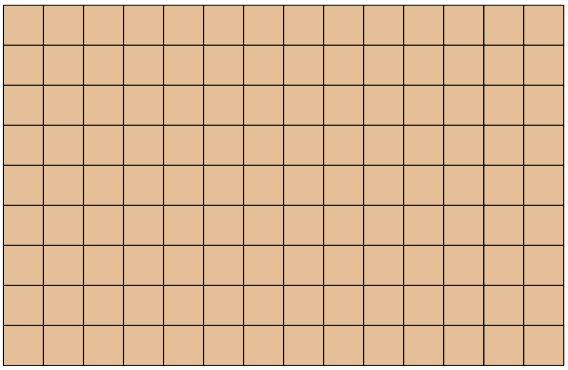
Per risparmiare spese postali e materiale, Linus deve mettere le quattro scatole diverse una accanto all'altra in una cassetta più piccola possibile. Le scatole non devono essere impilate l'una sull'altra, altrimenti le praline si schiacciano.

Linus ha disposto le praline in questo modo, in una cassetta per $5 \times 9 = 45$ praline individuali.



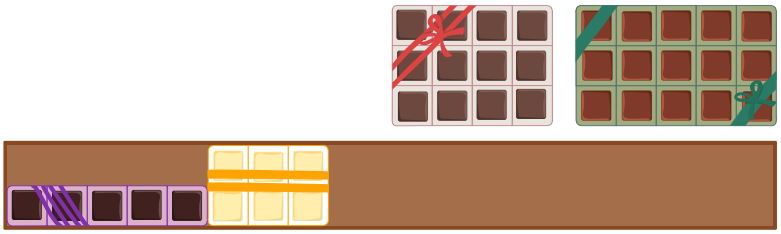
Lina però dice a Linus: «Se metti le scatole in modo diverso, entreranno in una cassetta più piccola.»

Posiziona le scatole in modo che entrino in una cassetta più piccola possibile.



Soluzione

In totale ci sono $12 + 15 + 6 + 5 = 38$ cioccolatini che Linus deve mettere in una cassetta. Una cassetta in cui si possono inserire 38 praline singole senza alcuno spazio vuoto deve avere una dimensione di 1×38 o 2×19 (2 e 19 sono gli unici divisori di 38). Le due scatole di praline di dimensioni 3×4 e 3×5 non entrerebbero in nessuna di queste cassette.



Se Linus sceglie una cassetta per 39 praline (cioè con uno spazio vuoto per esattamente un'altra pralina), essa ha le dimensioni 1×39 o 3×13. Le cassette 3×5, 3×4, 3×2 entrano nella cassetta, ma la scatola 1×5 non entra nello spazio libero rimanente di dimensione 2×3.

Una cassetta per 40 cioccolatini può avere le seguenti dimensioni 1×40, 2×20, 4×10, 5×8. Non tutte le scatole entrano nelle cassette con le dimensioni 1×40 o 2×20. Nelle altre due caselle si inseriscono tutte e quattro le caselle, ad esempio in questo modo:

Puoi aggiungere le scatole ad altre composizioni che entrano in una cassetta per 40 praline. Quindi queste quattro scatole di praline non possono essere confezionate in modo più salvaspazio che con lo spazio vuoto per 2 cioccolatini.

Questa è l'informatica!

In questo compito, i rettangoli devono essere disposti in modo tale che il rettangolo che li racchiude abbia l'area minima. Questo problema è noto in informatica anche come «impacchettamento dei rettangoli», uno dei tanti cosiddetti problemi di impacchettamento. Per alcuni rettangoli possiamo trovare la soluzione *ottimale* in modo relativamente semplice (in questo caso la cassetta più piccola possibile). Per quantità maggiori, è necessario automatizzare il processo; quindi è necessario un algoritmo che possa essere realizzato come programma informatico. Sfortunatamente, l'" impacchettamento dei rettangoli ", come molti altri problemi di impacchettamento, è *NP-completo*. Questo significa che molto probabilmente non esiste un *algoritmo efficiente* per il problema che trovi

soluzioni ottimali. In informatica vengono quindi sviluppati algoritmi efficienti per i problemi NP-completi, che non garantiscono di trovare soluzioni ottimali, ma possono trovarne di sufficientemente buone.

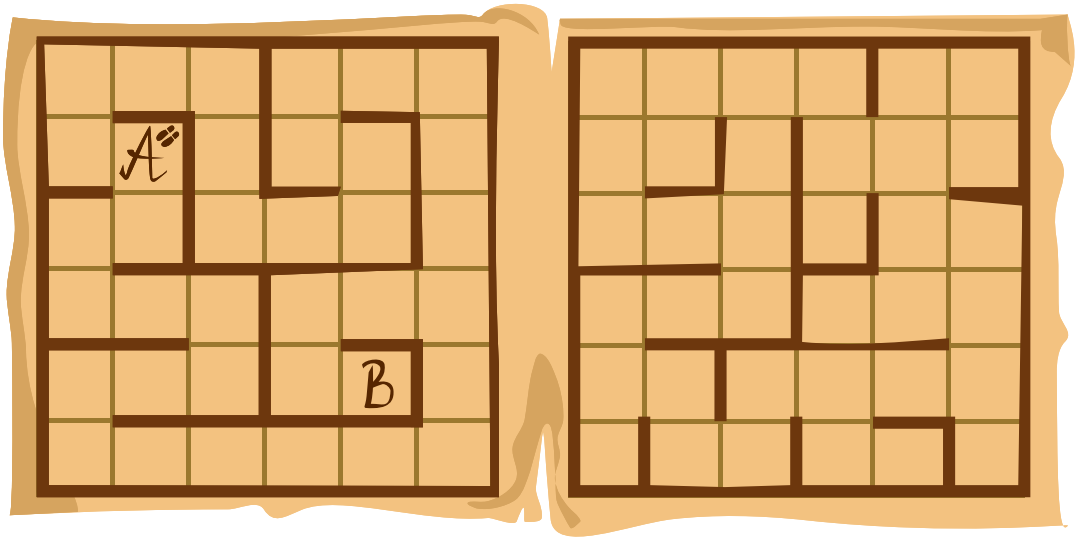
Per le aziende di logistica, tra le altre cose, le soluzioni efficienti per problemi di questo tipo sono di grande importanza, ad esempio per lo stoccaggio in scaffali alti, per l'imballaggio salvaspazio delle merci o per la distribuzione delle merci nei container. Inoltre, problemi apparentemente diversi possono essere descritti come problemi di impacchettamento. Ad esempio, un processo di lavoro che N lavoratori possono gestire in M ore può essere rappresentato come un rettangolo $N \times M$. In questo modo è possibile gestire diversi processi con il minor dispendio di persone e di tempo possibile se il problema dell' «impacchettamento dei rettangoli» viene risolto in modo ottimale per i rettangoli corrispondenti.

Parole chiave e siti web

- NP-completo: `https://it.wikipedia.org/wiki/NP-completo`
- Ottimizzazione : `https://it.wikipedia.org/wiki/Ottimizzazione_(matematica)`

28. Labirinto

La scuola di magia ha due piani. I piani sono esattamente uno sopra l'altro. Entrambi sono divisi in campi e tra alcuni di essi ci sono dei muri:



Lo studente mago Ron ha bisogno di 1 secondo per passare da una casella all'altra sullo stesso piano. Purtroppo Ron ha dimenticato come attraversare i muri. Tuttavia, può passare da un piano al quadrato corrispondente dell'altro piano, impiegando 5 secondi.

Ron vuole passare dalla casella A alla casella B il più velocemente possibile.

Di quanti secondi ha bisogno Ron al minimo per farlo?

A) 6 secondi

B) 16 secondi

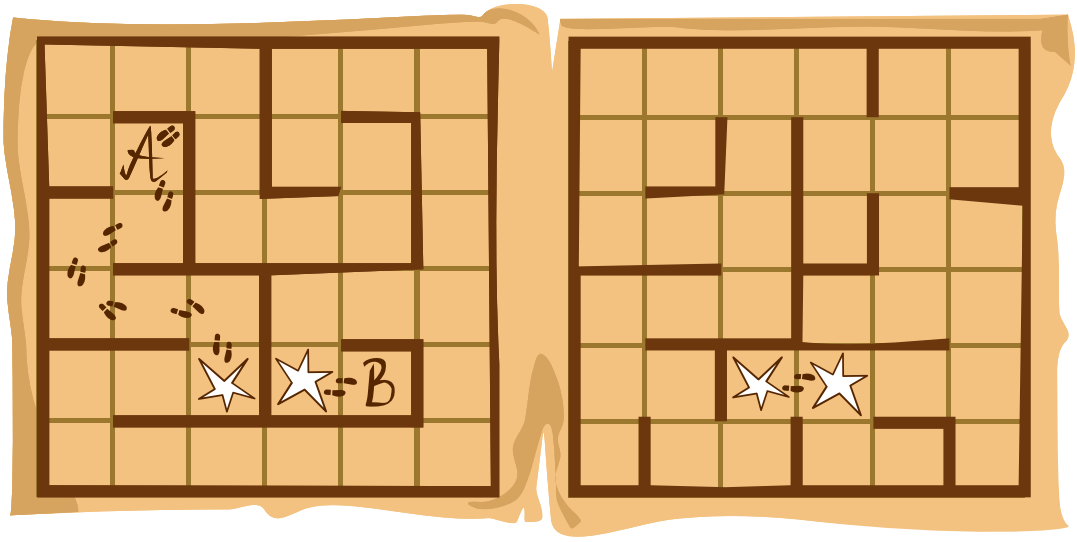
C) 18 secondi

D) 20 secondi

Soluzione

La risposta C) 18 secondi è corretta.

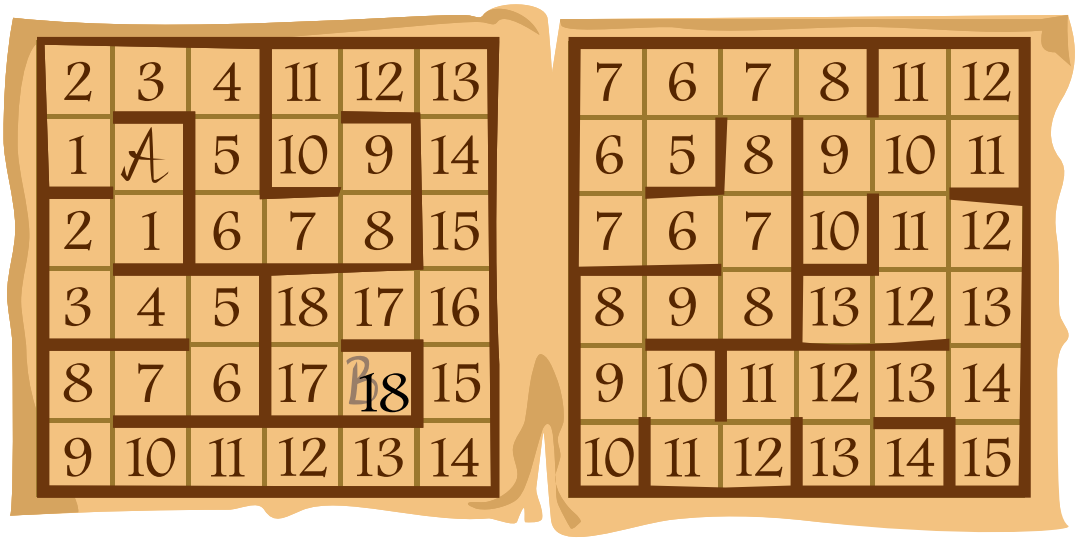
Quindi Ron può andare da A a B in 18 secondi:



È questo il modo più veloce? Il «tempo più breve» che Ron impiega per andare dal campo A a qualsiasi altro campo può essere calcolato passo dopo passo in questo modo:

Per il campo A, il tempo più breve è ovviamente 0 secondi. Poi continua passo dopo passo in questo modo: tra tutti i campi per i quali è già stato inserito il tempo più breve, scegli quello con il valore più basso. A partire da questo campo scelto, si esaminano tutti i possibili campi successivi e si considera come arrivarci più velocemente dal campo scelto; si inseriscono i tempi calcolati nei campi successivi. Può accadere che un tempo precedentemente inserito venga migliorato. In seguito, il campo selezionato non potrà più essere preso in considerazione e quindi non potrà più essere selezionato nelle fasi successive.

Ecco i tempi più brevi calcolati con questo metodo, partendo dal campo A:



Quindi Ron ha bisogno di almeno 18 secondi per andare dal campo A al campo B. 6 secondi (risposta A) sarebbe la durata del percorso più breve se non ci fossero muri tra i campi. Se poi Ron passasse da un piano all'altro, ci vorrebbero 16 secondi (risposta B). Se ci fosse solo il piano con i campi A e B, 20 secondi (risposta D) sarebbe il tempo più breve per il percorso da A a B.

Questa è l'informatica!

I percorsi più veloci o più brevi devono essere calcolati molto spesso; un esempio ovvio è la pianificazione del percorso nelle moderne app di mappe. Il problema si semplifica notevolmente se i percorsi sono costituiti da singoli passaggi tra punti vicini e se per tutti questi passaggi è noto il loro «costo»: Tempo, denaro, consumo di energia - qualunque sia la quantità importante per il problema attuale. In questo caso, i punti, i passi e i costi dei passi possono essere astratti in un *grafo* in cui i passi possono essere messi insieme per formare dei percorsi. Per i grafi, sono noti molti algoritmi in informatica con cui è possibile calcolare in modo efficiente i *cammini minimi*. Uno di questi è stato inventato dall'informatico Edsger Dijkstra; questo *algoritmo di Dijkstra* è stato utilizzato sopra nella spiegazione della risposta corretta.

I percorsi più brevi giocano un ruolo importante anche nella progettazione di circuiti per computer. I punti di commutazione devono essere cablati insieme al minor costo possibile. I circuiti moderni sono costituiti da più livelli e il cablaggio tra due livelli è più costoso rispetto al cablaggio (altrimenti comparabile) sullo stesso livello - simile al passaggio da un piano all'altro in questo compito, che è più costoso di un passo sullo stesso piano.

Parole chiave e siti web

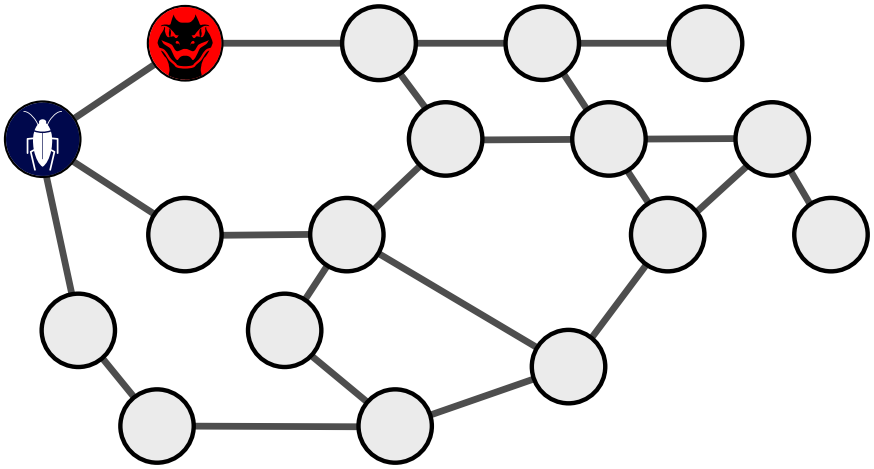
- Grafo: `https://it.wikipedia.org/wiki/Grafo`
- Cammino minimo: `https://it.wikipedia.org/wiki/Cammino_minimo`
- Edsger Dijkstra: `https://it.wikipedia.org/wiki/Edsger_Dijkstra`
- Algoritmo di Dijkstra: `https://it.wikipedia.org/wiki/Algoritmo_di_Dijkstra`

29. Virus

In una rete di computer, due nodi della rete sono stati infettati da virus informatici: uno con il virus BlueBug, l'altro con il virus RedRaptor. Entrambi i virus si diffondono sempre al mattino. Ogni virus infetta poi tutti i nodi che sono direttamente collegati ai nodi che ha già infettato. Se un nodo è infettato da entrambi i virus, si spegne dopo qualche ora a causa del sovraccarico. I virus non possono quindi diffondersi ulteriormente nei giorni successivi.

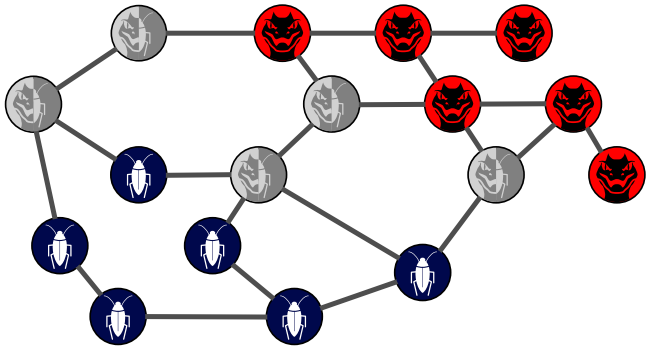
Qui sotto puoi vedere la rete di computer con i nodi e le loro connessioni dirette. I due nodi infettati all'inizio sono contrassegnati. Dopo qualche giorno, tutti i nodi vengono infettati da un virus o addirittura spenti.

Quali nodi vengono poi infettati da quale virus o spenti? Scegli il marcatore corretto per ogni nodo.

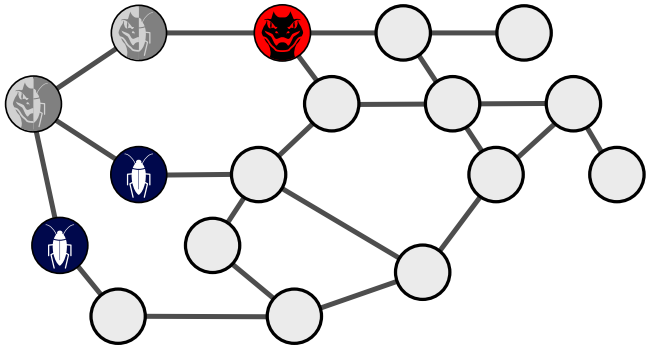


Soluzione

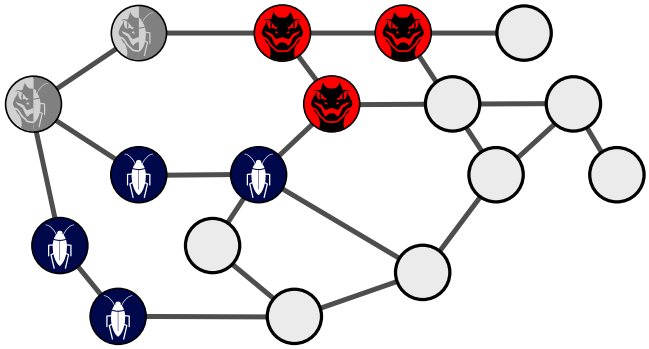
Dopo 5 giorni, tutti i nodi della rete vengono infettati o spenti. Questa è la soluzione corretta:



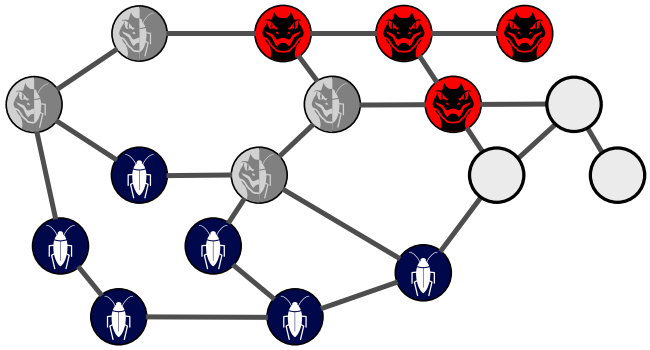
Dopo 1 giorno, cinque nodi della rete sono stati infettati. I due nodi infettati all'inizio sono ora infettati da entrambi i virus e quindi spenti:



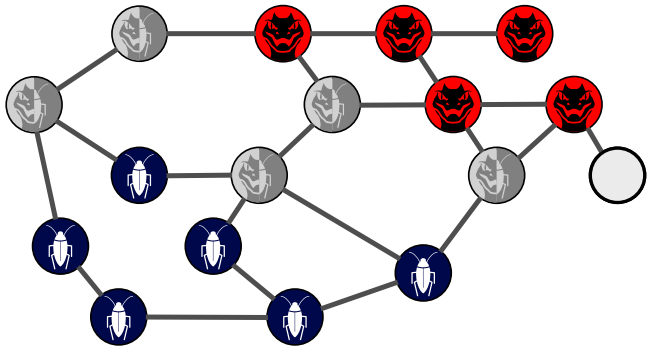
Dopo 2 giorni, vengono infettati altri quattro nodi:



Dopo 3 giorni, due nodi sono doppiamente infetti e ora sono anche spenti. Inoltre, altri tre nodi sono stati infettati da «BlueBug» e due da «RedRaptor»:



Dopo 4 giorni, un altro nodo della rete viene spento. «BlueBug» non può più diffondersi ulteriormente.



Il 5° giorno, l'ultimo nodo viene infettato da «RedRaptor».

Questa è l'informatica!

I virus e le altre minacce informatiche rappresentano una grande sfida per le reti informatiche. Non solo influiscono sulle prestazioni dei computer colpiti, ma spesso hanno un «carico aggiuntivo» (*payload*) che causa ulteriori danni. In alcuni casi, ad esempio, i dati trasmessi vengono letti e quindi informazioni sensibili come password o dati utente vengono scoperti e trasmessi a un cliente. In alcuni casi, il virus cripta i dati presenti sul computer infetto. Se l'utente vuole accedere nuovamente ai suoi dati, deve prima trasferire una somma di denaro su un conto anonimo. A volte gruppi di computer infetti sono controllati a distanza da criminali per effettuare attacchi ad altri computer (*botnet*).

Il fatto che un virus paralizzi completamente un computer di solito non è voluto dal creatore del virus, perché questo blocca la diffusione del virus. Tuttavia, alcuni virus sono stati sviluppati appositamente per il sabotaggio e la guerra informatica. Questo può anche danneggiare in modo permanente i computer colpiti.

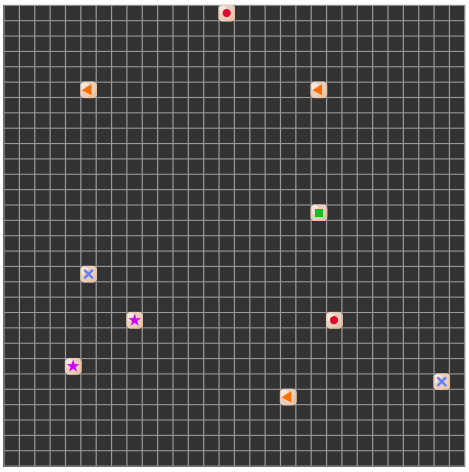
L'installazione degli ultimi aggiornamenti di sicurezza è un requisito importante per difendersi dai virus. I programmi antivirus possono migliorare la protezione, ma sono già inclusi in alcuni sistemi operativi, quindi un programma aggiuntivo potrebbe non essere necessario. Tuttavia, sono indispensabili backup regolari dei dati e un'attenta vigilanza sui comportamenti insoliti del sistema.

Parole chiave e siti web

- Rete di computer: `https://it.wikipedia.org/wiki/Rete_di_computer`
- Virus: `https://it.wikipedia.org/wiki/Virus_(informatica)`
- Botnet: `https://it.wikipedia.org/wiki/Botnet`
- Guerra cibernetica: `https://it.wikipedia.org/wiki/Guerra_cibernetica`

30. Colorazione del pavimento

Il pavimento di una stanza quadrata è diviso in 30×30 caselle. Su dieci caselle si trovano i gettoni con tali simboli colorati:  ,  ,  ,  e  .



Un robot deve dipingere il pavimento con questi simboli, campo per campo. A tal fine, sono previste quattro regole diverse. Su un campo dove non ci sono gettoni, si dipinge ...

1 ... il simbolo del gettone più vicino a lui.

2 ... il simbolo del gettone più lontano da lui.

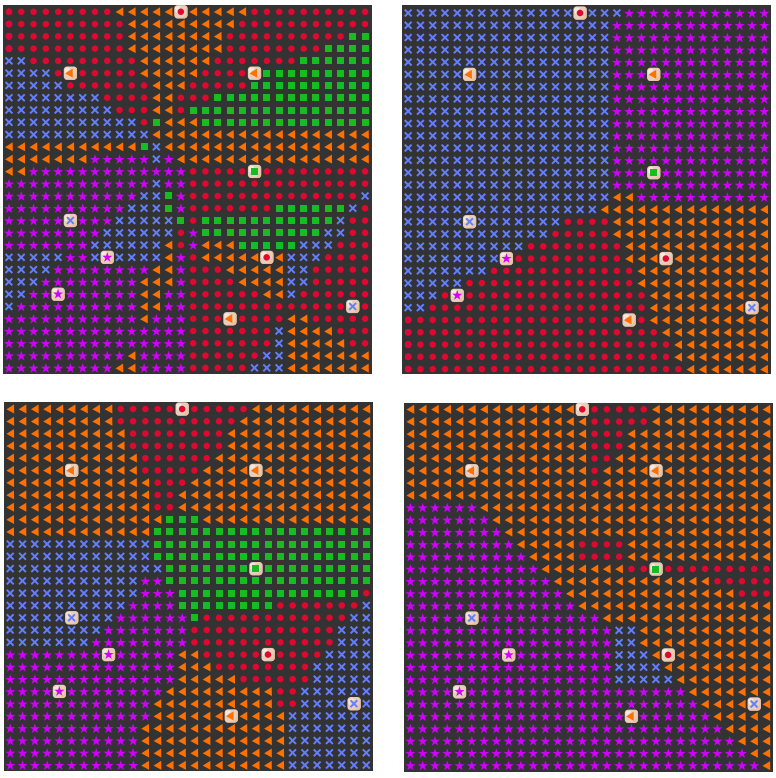
3 ... il simbolo del gettone che è il secondo più vicino a lui.

4 ... il simbolo che si ripete più frequentemente tra i 6 gettoni più vicini.

Il robot dipinge tutti i quadrati seguendo la stessa regola. Se la regola di un campo fornisce diversi simboli possibili, il robot ne sceglie uno a caso.

Di seguito è possibile vedere come viene dipinto il pavimento alla fine per ogni regola.

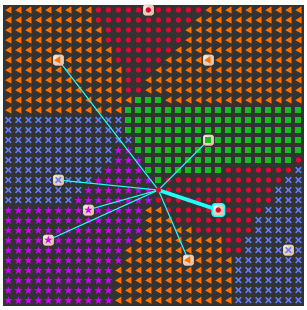
Quale pavimento corrisponde a quale regola? Abbina le regole ai pavimenti.



Soluzione

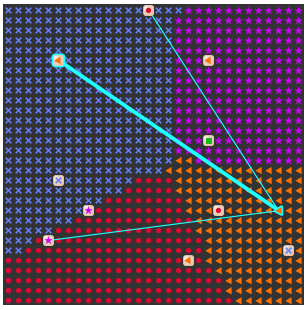
Poiché tutti i campi di un pavimento sono dipinti secondo la stessa regola, è sufficiente controllare un campo alla volta. Per ogni pavimento esaminiamo un campo diverso:

Regola 1



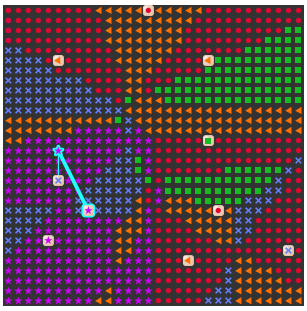
Il campo è dipinto con ● perché un gettone è il più vicino.

Regola 2



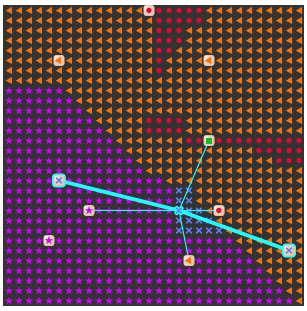
Il campo è dipinto con ◀ perché un gettone è il più lontano.

Regola 3



Il campo è dipinto con ★ perché un gettone è il secondo più vicino.

Regola 4



Il campo viene dipinto con ✕ perché questo è il più comune tra i 6 gettoni più vicini.

Questa è l'informatica!

Le divisioni di un piano e la loro costruzione *algoritmica* svolgono un ruolo importante in varie aree dell'informatica, ad esempio nelle simulazioni e nella computer grafica.

I *diagrammi di Voronoi*, che prendono il nome dal matematico ucraino Georgi Feodosyevich Voronoi (*1868 - †1908), dividono un piano in regioni attorno ai cosiddetti *centri*. Tutti i punti di una regione non sono più vicini a nessun altro centro del proprio. Il risultato della regola 1 è un diagramma di Voronoi. Questi diagrammi si trovano spesso in situazioni reali, ad esempio nella copertura della telefonia mobile. Oppure vengono utilizzati nell'analisi delle partite di calcio o di altri comportamenti socio-economici, come le relazioni tra la popolazione e le scuole, gli ospedali o alcuni fornitori di servizi più vicini.

Nel 1911, il meteorologo Alfred H. Thiessen (*1872 - †1956) ha sviluppato un metodo per determinare i valori medi (ad esempio le quantità di precipitazioni) delle aree in modo più realistico con l'aiuto

dei diagrammi di Voronoi. Non determina la media dei valori misurati dalle stazioni di misura solo in base al numero di stazioni di misura, ma utilizza il diagramma di Voronoi per determinare innanzitutto l'area a cui si riferiscono i valori misurati. Ciò comporta una diversa ponderazione dei valori misurati a livello locale.

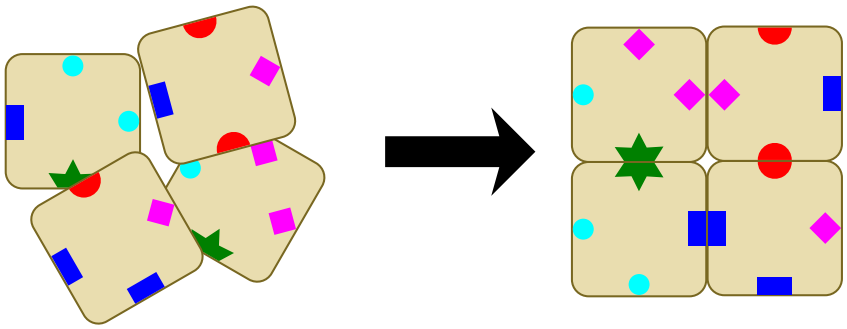
Parole chiave e siti web

- Algoritmo: `https://it.wikipedia.org/wiki/Algoritmo`
- Diagramma di Voronoi: `https://it.wikipedia.org/wiki/Diagramma_di_Voronoi`

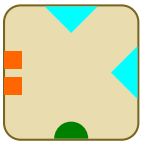
31. Mosaico

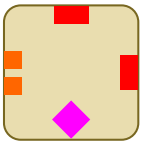
Devi disporre quattro carte in un quadrato in modo che due bordi che si toccano abbiano lo stesso simbolo.

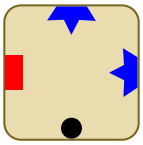
Ad esempio, le seguenti quattro carte possono essere posizionate come un tale quadrato:

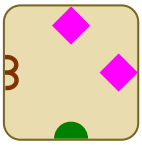


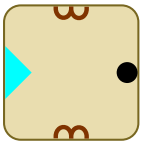
È possibile creare un quadrato di questo tipo con quattro delle cinque carte seguenti. Quale non si può usare?

A) 

B) 

C) 

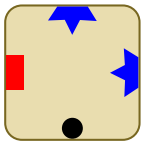
D) 

E) 

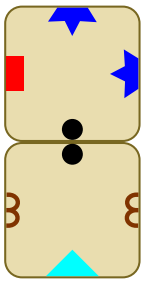
Soluzione

Ci sono $\binom{5}{4} \cdot 4! \cdot 4^4 = 30720$ modi diversi di scegliere e posare quattro delle cinque carte. Anche considerando che ci sono almeno quattro soluzioni dovute alla simmetria rotazionale, sono decisamente troppe per provarle tutte.

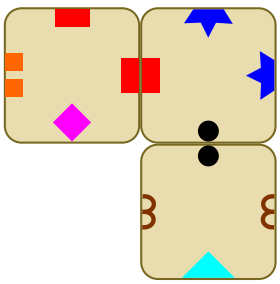
Pertanto, è opportuno osservare innanzitutto la distribuzione dei simboli sulle carte. Si noterà che la mezza stella blu appare solo sulla carta C). Poiché appare anche due volte, cioè attraverso l'angolo, le altre due carte possono essere posizionate solo a sinistra e in basso, se non si cambia l'orientamento (che è inutile visto che è la prima carta).



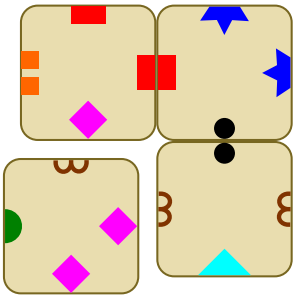
Inoltre, c'è solo un'altra carta con un cerchio nero che deve essere posizionata sotto, cioè la carta E):



Anche per il rettangolo rosso esiste una sola carta, cioè la carta B). Contiene rettangoli rossi in due punti, ma poiché non esiste un'altra carta con rettangoli rossi, deve essere ruotata in modo che il secondo rettangolo rosso sia in alto e non in basso. Altrimenti, si dovrebbe aggiungere una carta con un rettangolo rosso al di sotto di essa, e questa carta non esiste:

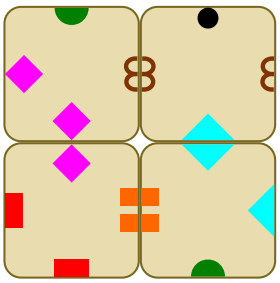


L'ultima cosa necessaria è una carta con un quadrato magenta e due semicerchi marroni. La carta D) presenta anche questi simboli, ma nell'ordine sbagliato: i due semicerchi marroni dovrebbero venire in senso orario dopo il quadrato magenta, ma è esattamente il contrario.



Poiché non c'erano alternative per tutte le carte, si dimostra che con la carta C) con le due mezze stelle blu non si può posare un quadrato che soddisfi le condizioni.

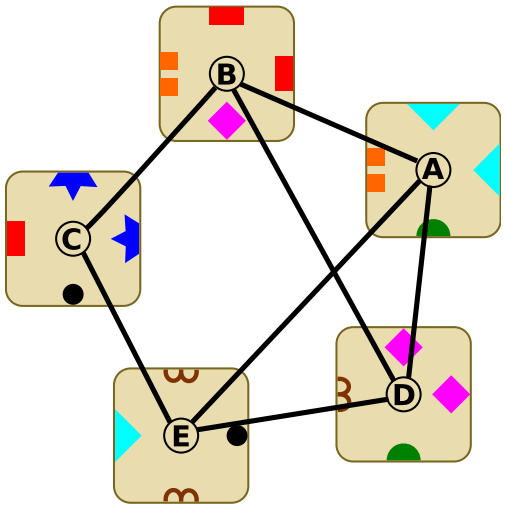
Tuttavia, tale quadrato può essere posato dalle altre carte:



Questo dimostra che la carta C) è l'unica con la quale non è possibile posare un quadrato di questo tipo e C) è la risposta corretta.

Questa è l'informatica!

Se due carte hanno lo stesso simbolo, possono essere messe una accanto all'altra. Questo può essere rappresentato con l'aiuto di un *grafo*: le carte rappresentano i *vertici*, se c'è un simbolo comune, allora c'è un *arco*. Per le cinque carte di questo compito, il grafo si presenta come segue:

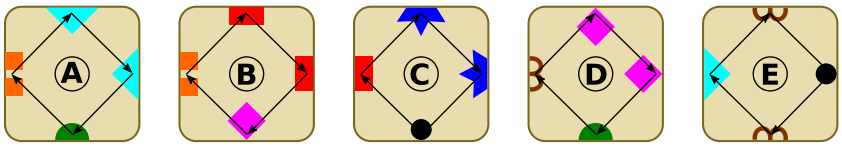


Se quattro carte sono disposte a quadrato in modo che esattamente due carte abbiano sempre lo stesso simbolo, significa che nel grafico è possibile seguire un percorso circolare con esattamente quattro

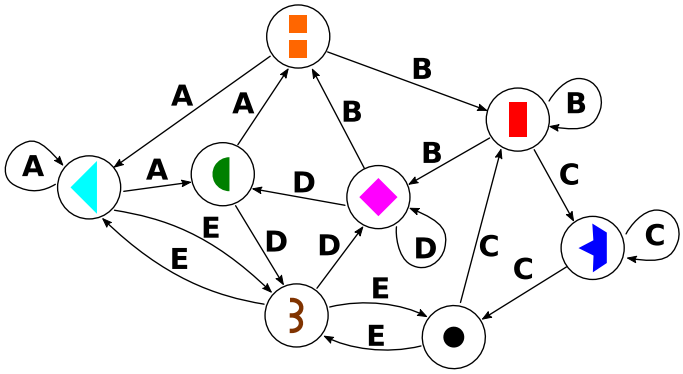
carte, per esempio da A a E a C a B e di nuovo ad A. Abbreviamo tale percorso come A-E-C-B-A e intendiamo tutti gli altri percorsi circolari che seguono questo ordine, cioè anche E-C-B-A-E, C-B-A-E-C e B-A-E-C-B.

Nel grafo ci sono esattamente tre percorsi circolari con quattro mappe: il già citato percorso circolare A-E-C-B-A, A-E-D-B-A e B-D-E-C-B. Questo riduce il numero di soluzioni possibili da $\binom{5}{4} \cdot 4! \cdot 4^4 = 30720$ (rispettivamente 7680 se si omette la simmetria rotazionale) a 12 (rispettivamente 3 se si omette la simmetria rotazionale). Questi possono essere controllati rapidamente per trovare la soluzione corretta (A-E-D-B-A).

È anche possibile utilizzare un grafo in cui i simboli rappresentano i vertici. A tal fine, è necessario definire l'ordine dei simboli all'interno di una carta in modo uniforme, ad esempio in senso orario:



Nel grafo, un bordo direzionato esiste tra due simboli se sono adiacenti su una mappa in senso orario:



La soluzione del compito è data esattamente se si trova un percorso circolare che passa esattamente su quattro archi. Si tratta del percorso circolare ■■ - ▲ - ᔓ - ◆ - ■■, che passa sopra gli archi A, E, D e B e corrisponde quindi alla nostra soluzione.

Questo processo di concentrazione sull'essenziale e di esclusione del non importante si chiama astrazione. In questo caso, l'astrazione consente un riconoscimento molto più rapido della soluzione corretta.

Parole chiave e siti web

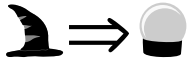
- Grafo: `https://it.wikipedia.org/wiki/Grafo`
- Vertice: `https://it.wikipedia.org/wiki/Vertice_(teoria_dei_grafi)`
- Arco: `https://it.wikipedia.org/wiki/Glossario_di_teoria_dei_grafi#Arco`

32. Oggetti magici

Nella terra magica ci sono quattro diversi oggetti magici:

Cappelli magici, Sfere di cristallo, Libri magici e Pozioni magiche.

I cappelli magici e le sfere di cristallo possono essere trasformati in due modi diversi. La tabella mostra cosa viene creato dagli oggetti in questo processo - esattamente nel posto in cui si trovavano prima e nella disposizione indicata:

da . . .	**nasce**
	
	
	
	

Le trasformazioni possono avvenire un numero qualsiasi di volte e in qualsiasi ordine. Così, da un unico oggetto magico può nascere una lunga disposizione di oggetti.

Quale disposizione NON può nascere da un singolo cappello magico?

A) 

B) 

C) 

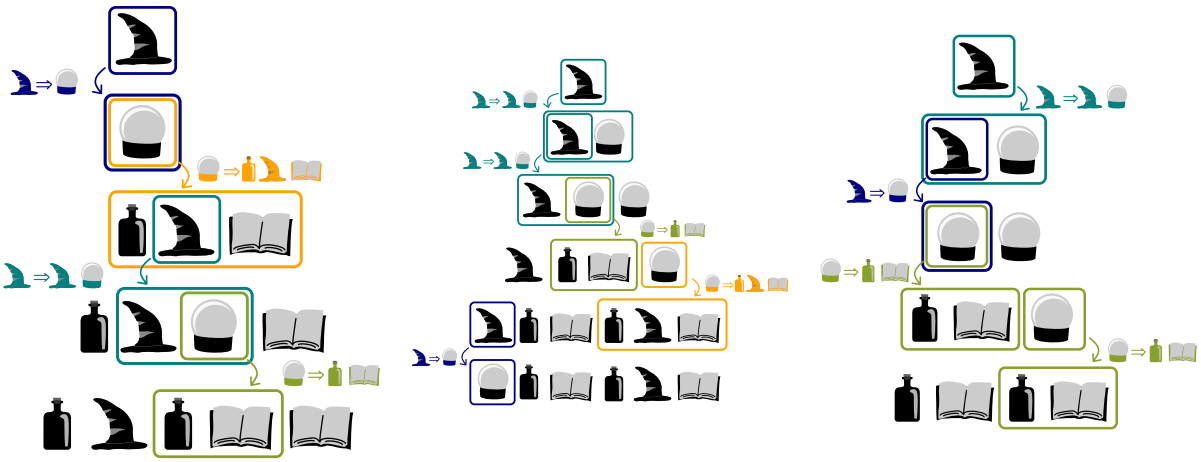
D) 

Soluzione

La risposta B) è corretta: 

Le disposizioni delle risposte A, C e D possono nascere da un unico cappello magico:

Risposta A **Risposta C** **Risposta D**



Contiene un numero diverso di libri ($2 \times$ 📖) e pozioni ($3 \times$ 🧪). Ma ogni volta che viene creata una pozione in una trasformazione, viene creato contemporaneamente un libro (🔮 ⇒ 🧪📖 e 🔮 ⇒ 🧪🎩📖). Quindi, in ogni disposizione creata nella terra magica dalle trasformazioni, ci devono essere esattamente tanti libri di incantesimi quante sono le pozioni. Quindi la disposizione della risposta B) non può nascere né da un cappello magico né da una sfera di cristallo.

Questa è l'informatica!

Quando le sfere di cristallo e i cappelli magici di questo compito del castoro vengono trasformati di volta in volta, si creano disposizioni diverse. Poiché le trasformazioni producono sempre nuove sfere di cristallo e cappelli magici, è possibile creare un numero infinito di disposizioni. Non è quindi possibile scrivere tutte le disposizioni che possono nascere con l'aiuto delle trasformazioni. Ma non è nemmeno necessario, perché l'insieme delle disposizioni è determinato proprio dalle trasformazioni stesse.

Anche prima dell'esistenza dei computer, è nata l'idea di descrivere infiniti insiemi di disposizioni con l'aiuto di un insieme relativamente piccolo e comunque finito di trasformazioni. In informatica, le trasformazioni sono chiamate *regole di sostituzione*, gli insiemi di regole sono chiamati *grammatiche* e gli insiemi che le definiscono sono di conseguenza chiamati *lingue*. Un ruolo centrale in questi *linguaggi formali* dell'informatica è svolto dal problema della decisione: una disposizione di oggetti appartiene al linguaggio (cioè: può essere creata applicando le regole) oppure no?

Nel rispondere a questo compito, hai dovuto risolvere questo problema per quattro disposizioni. Fortunatamente, la grammatica di questo compito rientra nella classe delle grammatiche *libere dal contesto*: le sfere di cristallo e i cappelli magici possono trasformarsi senza considerare ciò che li

circonda, cioè nel loro contesto. Per le grammatiche libere da contesto, il *problema della decisione* è generalmente facile da risolvere, ed è per questo che sono molto popolari in informatica, ad esempio per descrivere i linguaggi di programmazione.

Parole chiave e siti web

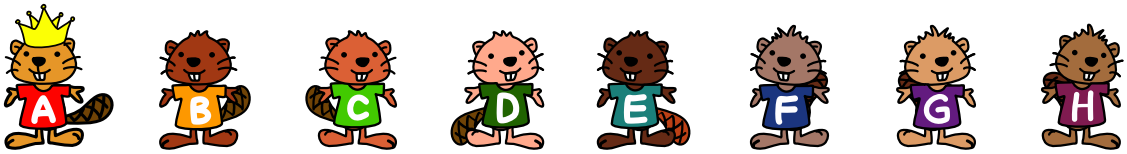
- Linguaggio libero dal contesto: `https://it.wikipedia.org/wiki/Linguaggio_libero_dal_contesto`
- Grammatica libera dal contesto: `https://it.wikipedia.org/wiki/Grammatica_libera_dal_contesto`
- Linguaggio formale: `https://it.wikipedia.org/wiki/Linguaggio_formale`
- Decidibilità: `https://it.wikipedia.org/wiki/Decidibilità`

33. Campionato dei Castori

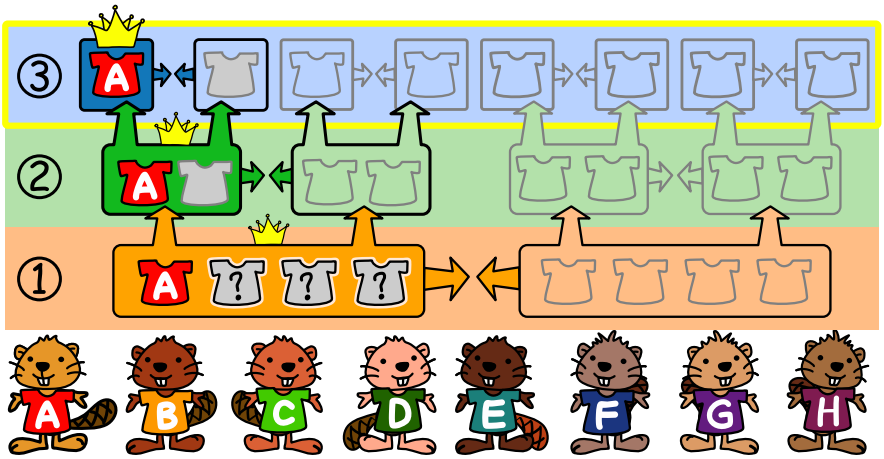
8 castori partecipano al Campionato dei castori. Ci sono tre turni. In ogni turno, ogni castoro raccoglie punti.

- Turno ①: si formano 2 squadre casuali di 4 castori ciascuna. I punti dei singoli castori vengono sommati. La squadra con il maggior numero di punti vince e passa al turno ②. I perdenti continuano a giocare e si accordano tra loro per i posti dal 5 all'8.
- Turno ②: Questo viene condotto con le stesse regole. Le squadre sono ora composte da 2 castori. I vincitori passano alla finale. I perdenti continuano a giocare e a stabilire i posti d'onore tra di loro.
- Turno ③: La finale! Non ci sono squadre, ma 2 castori singoli che si sfidano tra loro.

La castorina Ada è la vincitrice del campionato dei castori. Di seguito sono riportati i punti ottenuti da ciascun castoro in ogni turno.



Nome	Ada	Brown	Candy	Daisy	Eden	Funny	George	Hugh
①	15	16	19	18	17	20	19	19
②	20	27	30	24	28	24	30	30
③	10	14	11	15	16	13	9	12



Quali tre castori facevano parte della squadra di Ada nel primo turno?

Soluzione

I tre compagni di squadra di Ada nel turno ① erano Daisy, Funny e George.

La finale sarà giocata individualmente. George è l'unico castoro con meno punti di Ada. Nel turno ② devono quindi aver fatto parte della stessa squadra.

Nel turno ② hanno raggiunto insieme 50 punti. Questo valore deve essere superiore al punteggio totale dell'altra squadra di due persone. I due castori Daisy e Funny sono l'unica coppia il cui totale dei punti è inferiore a 50. Pertanto, devono aver fatto parte della stessa squadra di Ada e George nel turno ①.

Poiché ora abbiamo trovato tutti e quattro i castori della squadra di Ada nel turno ①, conosciamo anche la composizione della seconda squadra nel turno ①.

Nel turno ① la squadra (Ada, Daisy, Funny, George) ha totalizzato 72 punti. L'altra squadra (Brown, Candy, Eden, Hugh) ha totalizzato solo 71 punti. La squadra di Ada ha vinto.

Nel turno ②, (Ada, George) ha totalizzato 50 punti, mentre (Daisy, Funny) ha ottenuto solo 48 punti. Nel round ③, cioè la finale, Ada vince con 10 punti contro 9 contro George. Ada vince il campionato dei castori.

Questa è l'informatica!

Per risolvere questo compito, possiamo formare sistematicamente tutte le squadre possibili al primo turno. Se conosciamo una delle due squadre, anche la seconda è nota. In totale, esistono $\binom{7}{3} = 35$ combinazioni. Per ognuna di queste combinazioni, dobbiamo esaminare i risultati nel turno ①, nel turno ② e nella finale prima di poter decidere quali sono stati effettivamente i compagni di squadra di Ada nel primo turno. Questo richiede molto tempo.

Per risolvere un compito come questo, gli informatici cercano approcci più efficienti. Invece di procedere in avanti, cioè dal primo al terzo round, si può anche ricavare la soluzione corretta a ritroso. Questo può essere fatto anche molto rapidamente, come abbiamo già visto nella spiegazione precedente.

Questo metodo è chiamato *ricerca a ritroso*. Si utilizza in situazioni in cui si cerca una soluzione che deve soddisfare determinate condizioni. In alcuni casi, per trovare una soluzione si combinano una ricerca *in avanti* e una *all'indietro*.

La ricerca in avanti e la ricerca all'indietro sono due strategie generali di risoluzione dei problemi. Vengono utilizzati per risolvere problemi in tutte le discipline, non solo in informatica.

Parole chiave e siti web

- Algoritmo di Dijkstra: `https://it.wikipedia.org/wiki/Algoritmo_di_Dijkstra`
- Algoritmo A*: `https://it.wikipedia.org/wiki/Algoritmo_A*`

A. Autori dei quesiti

Gulgun Afacan
Esraa Almajhad
Waël Almoman
Leo Barichello
Liam Baumann
Wilfried Baumann
Linda Björk Bergsveinsdóttir
Tobias Berner
Daniela Bezáková
Graeme Buckie
Marta J. Burzanska
Sarah Chan
Byeonggyu Cho
Kris Coolsaet
Darija Dasović
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
Thomas Ioannou
Sangsu Jeong
Mile Jovanov
Dauksaite Justina
Dong Yoon Kim
Hakin Kim
Jihye Kim
Seulki Kim
Vaidotas Kinčius
Víctor Koleszar
Lidija Kralj
Regula Lacher
Taina Lehtimäki
Marielle Léonard
Inggriani Liem
Monika Maneva
Karolína Miková
Zoran Milevski
Jelena Milojkovic
Madhavan Mukund
Ágnes Erdősné Németh
Ilze Nilandere
Veronika Ognjanovska
Mārtiņš Opmanis
Elsa Pellet
Jean-Philippe Pellet

Margot Phillipps
Zsuzsa Pluhár
Wolfgang Pohl
John-Paul Pretti
Le Quang Quan
Susannah Quidilla
Lorenzo Repetto
Chris Roffey
Kirsten Schlüter
Giovanni Serafini
Yeh Yi Shan
Timur Sitdikov
Bernadette Spieler
Emil Stankov
Veronika Stefanovska
Alieke Stijf
Goran Sukovic
Monika Tomcsányiová
Ahto Truu
Jiří Vaníček
Troy Vasiga
Willem van der Vegt
Rechilda Villame
Florentina Voboril
Michael Weigend
Kyra Willekes
Hongjin Yeh

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.