INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Quesiti e soluzioni 2020

11º al 13º anno scolastico

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

A cura di:

Lucio Negrini, Christian Giang, Susanne Datzko, Fabian Frei,
Juraj Hromkovič, Regula Lacher, Jean-Philippe Pellet

0101001101010110010010010000001001011010101001101010010100100101000010

SSII

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

Hanno collaborato al Castoro Informatico 2020

Susanne Datzko, Fabian Frei, Martin Guggisberg, Lucio Negrini, Gabriel Parriaux, Jean-Philippe Pellet

Capo progetto: Nora A. Escherle

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

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: Bebras.org
Wolfgang Pohl, Hannes Endreß, Ulrich Kiesmüller, Kirsten Schlüter, Michael Weigend: Bundesweite Informatikwettbewerbe (BWINF), Germania
Wilfried Baumann, Anoki Eischer: Österreichische Computer Gesellschaft
Gerald Futschek, Florentina Voboril: Technische Universität Wien
Zsuzsa Pluhár: ELTE Informatikai Kar, Ungheria
Michal Winzcer: Comenius University, Slovacchia

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

Per il supporto durante le settimane del concorso ringraziamo:
Hanspeter Erni: Direttore scuola media di Rickenbach
Gabriel Thullen: Collège des Colombières
Beat Trachsler: Scuola cantonale di Kreuzlingen

Christoph Frei: Chragokyberneticks (Logo Informatik-Biber Schweiz)
Dr. Andrea Leu, Maggie Winter, Brigitte Manz-Brunner: 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 2020 è stato organizzato dalla Società Svizzera per l'Informatica nell'Insegnamento SSII con il sostegno della fondazione Hasler.

HASLERSTIFTUNG

Questo quaderno è stato creato il 9 settembre 2021 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'01.12.2020.

Premessa

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

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

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

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

Nel 2020 il Castoro Informatico della Svizzera è stato proposto a cinque differenti categorie d'età, suddivise in base all'anno scolastico:

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

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

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

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

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

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

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

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

Per ulteriori informazioni:

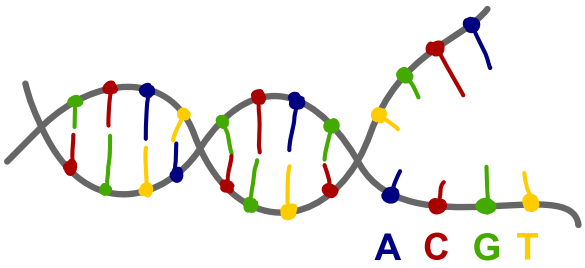
SVIA-SSIE-SSII Società Svizzera per l'Informatica nell'Insegnamento
Castoro Informatico
Lucio Negrini

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

Indice

1. Sequenza di DNA

Il nostro materiale genetico è immagazzinato in sequenze di DNA. Una sequenza di DNA è essenzialmente una sequenza di basi che si presentano nei quattro tipi A, C, G e T.



Consideriamo i seguenti tre tipi di mutazioni:

Tipo di mutazione	**Descrizione**	**Esempio**
Sostituzione	Una singola base viene sostituita da un'altra.	AT**G**GT → AT**A**GT
Cancellazione	Una singola base viene eliminata senza sostituzione.	ATG**G**T → ATGT
Inserimento	Una singola base è inserita da qualche parte.	ATGGT → A**C**TGGT

*Esattamente una delle quattro sequenze di DNA seguenti **non** può essere creata se la sequenza GTATCG subisce tre mutazioni. Qual è?*

A) GCAATG
B) ATTATCCG
C) GAATGC
D) GGTAAAC

Soluzione

La risposta corretta è D) GGTAAAC.

Il modo migliore per ottenere questa risposta è la procedura di esclusione, perché 3 mutazioni sono sufficienti per tutte le altre sequenze.

Risposta A: GTATCG ⇒ G**C**ATCG ⇒ GCA**A**CG ⇒ GCAATG
Risposta B: GTATCG ⇒ **A**TATCG ⇒ AT**T**ATCG ⇒ ATTAT**C**CG
Risposta C: GTATCG ⇒ G**A**ATCG ⇒ GAAT**G**G ⇒ GAATG**C**

Invece, sono necessarie 4 mutazioni per ottenere la sequenza dalla risposta D, per esempio le seguenti:

GTATCG ⇒ G**G**TATCG ⇒ GGT**A**ATCG ⇒ GGTAA**A**CG ⇒ GGTAAAC

Non è facile dimostrare che meno mutazioni non sono sufficienti.

Questa è l'informatica!

La rappresentazione di informazioni con *stringhe di caratteri* (sequenze di lettere) e il lavoro con esse è un compito centrale dell'informatica.

Una domanda importante è quanto le due stringhe di caratteri differiscano l'una dall'altra. Esistono diversi metodi per misurare la differenza tra due stringhe. Un metodo di misura frequentemente usato è la cosiddetta *distanza di Levenshtein*, che è definita dai tre *tipi di mutazioni* sopra descritti: la distanza di Levenshtein tra due stringhe è il numero minimo di mutazioni che può essere usato per convertire una stringa nell'altra.

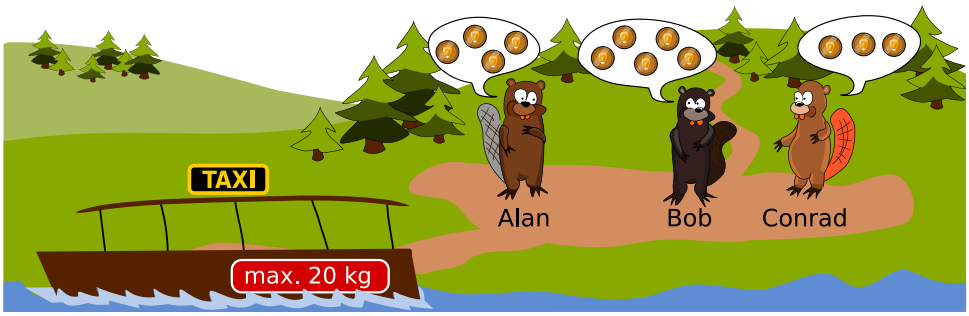
Il consueto algoritmo per il calcolo della distanza Levenshtein tra due parole si basa sulla *programmazione dinamica*: qui le distanze Levenshtein tra i prefissi sempre più lunghi delle due parole vengono scritte in una tabella fino a quando alla fine i due prefissi corrispondono alle parole intere e il risultato può essere letto.

Se la correttezza dell'algoritmo è dimostrata, si può calcolare che la distanza di Levenshtein tra la sequenza di DNA originale e quella della risposta D) è esattamente 4. Questo dimostra che meno mutazioni non sono sufficienti.

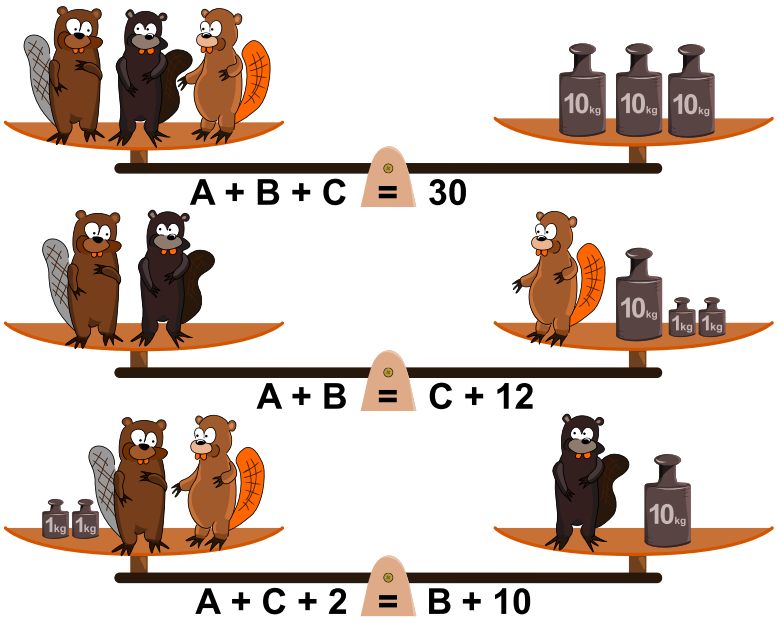
Parole chiave e siti web

- Distanza di Levenshtein: `https://it.wikipedia.org/wiki/Distanza_di_Levenshtein`

2. Taxi acquatico



I tre castori Alan, Bob e Conrad vogliono prendere un taxi acquatico. C'è solo un taxi acquatico. Alan pagherebbe 4 talleri (4×), Bob invece 5 talleri (5×) e Conrad solo 3 talleri (3×). Il taxi può trasportare un massimo di 20 kg. Pertanto il tassista effettua le seguenti pesate:



Quali castori trasporta il tassista se vuole guadagnare il più possibile?

A) Solo Bob
B) Alan e Bob
C) Bob e Conrad
D) Alan e Conrad
E) Tutti e tre: Alan, Bob e Conrad

Soluzione

La risposta corretta è: C) Bob e Conrad

Per poter elencare e poi valutare tutte le possibili soluzioni, dobbiamo prima sapere quanto pesa ogni castoro.

Sappiamo che tutti e tre insieme pesano 30 kg e quindi non tutti possono essere presi dal tassista. Se mettiamo di nuovo una copia di C(onrad) sul lato destro e sinistro della seconda bilancia, otteniamo $A + B + C = 30$ kg a sinistra e $C + C + 12$ kg a destra. Pertanto deve essere applicato $2C = 18$ kg e quindi $C = 9$ kg.

Se mettiamo di nuovo una copia di B(ob) sul lato destro e sinistro della terza bilancia, otteniamo $A + B + C + 2$ kg $= 32$ kg a sinistra e $2B + 10$ kg a destra. Quindi $2B = 22$ kg e quindi $B = 11$ kg.

Poiché $A + B + C = 30$ kg, A deve essere di 10 kg.

Così il tassista può:

- Trasportare Alan e Conrad, e guadagnare $4 + 3 = 7$ talleri.
- Trasportare Bob e Conrad, e guadagnare $5 + 3 = 8$ talleri.
- Trasportare Alan e Bob, allora lui guadagnerebbe 9 talleri, ma purtroppo i due insieme pesano 21 kg e quindi sovraccaricano il taxi d'acqua.

Pertanto la risposta corretta è C).

Ma questo non è l'unico modo per determinare il peso dei castori. Così come si sarebbe potuto sostituire $A + B$ con $C + 12$ sulla prima bilancia a sinistra nel primo passo. Si ottengono quindi $2C + 12$ kg sul lato sinistro, pari a 30 kg. Così si conclude ancora una volta che $C = 9$ kg.

Più formalmente, le tre pesate possono essere scritte come un sistema di equazioni:

I. $A + B + C = 30$ kg
II. $A + B - C = 12$ kg
III. $A - B + C = 8$ kg

Queste equazioni possono poi essere sottratte l'una dall'altra. Così la differenza I – II fornisce l'equazione:

$$2C = 18 \text{ kg} \rightarrow C = 9 \text{ kg}$$

La differenza I. – III. dà:

$$2B = 22 \text{ kg} \rightarrow B = 11 \text{ kg}$$

Da I. segue quindi $A = 10$ kg.

Questa è l'informatica!

Tutti i problemi di ottimizzazione discreti di NP possono essere rappresentati nel linguaggio delle equazioni lineari e delle disuguaglianze. Le equazioni e le disuguaglianze sono le cosiddette restrizioni, che i valori delle variabili devono soddisfare. Si ottimizza quindi il valore di una funzione delle variabili, mentre le restrizioni devono essere rispettate. Nel presente lavoro abbiamo tre variabili booleane, x_A, x_B, x_C. Si $x_A = 1$, il castoro A viene preso a bordo, altrimenti $x_A = 0$. Si ottimizza la funzione lineare $4x_A + 5x_B + 3x_C$, cercando il valore massimo. L'unica restrizione è:

$$Peso(A) \cdot x_A + Peso(B) \cdot x_B + Peso(C) \cdot x_C \leq 20.$$

Il compito può essere formulato in modo completo solo attraverso la determinazione del peso dei castori. Questo esempio di problema è un caso del *problema dello zaino*. Si cerca di mettere più valore possibile nello zaino senza superare il peso totale.

Solo 80 anni fa, tali questioni erano compito dei matematici, ma man mano che si rendevano disponibili computer sempre più potenti, sono stati sviluppati metodi di soluzione per tali problemi (ad esempio i metodi «*branch-and-bound*» o «*cutting-plane*»). Oggi questi metodi di soluzione vengono utilizzati, ad esempio, per ottimizzare la produzione, nella logistica o nelle reti di trasporto locale.

Tuttavia, la soluzione dei problemi di ottimizzazione nella pratica è ancora un compito difficile, che richiede una modellazione abile e algoritmi appositamente sviluppati, a seconda delle dimensioni e della struttura del problema. Spesso vengono combinati diversi metodi di soluzione.

Parole chiave e siti web

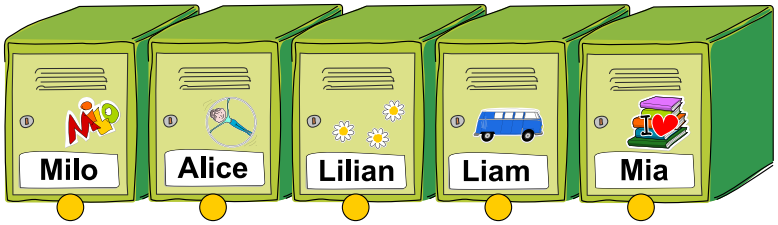
- Programmazione lineare: `https://it.wikipedia.org/wiki/Programmazione_lineare`
- Restrizione di una funzione: `https://it.wikipedia.org/wiki/Restrizione_di_una_funzione`
- Branch- and bound: `https://it.wikipedia.org/wiki/Branch_and_bound`

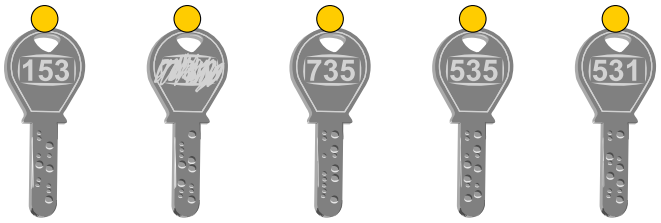
3. Armadietti

Cinque bambini hanno ciascuno un armadietto etichettato nella loro scuola. Le cinque chiavi corrispondenti hanno numeri a tre cifre. Sfortunatamente, una chiave ha un numero graffiato.

Ogni numero a tre cifre rappresenta le prime tre lettere di un nome. Una cifra sta per la stessa lettera ovunque, ad esempio 8 sta sempre per «C» o «c».

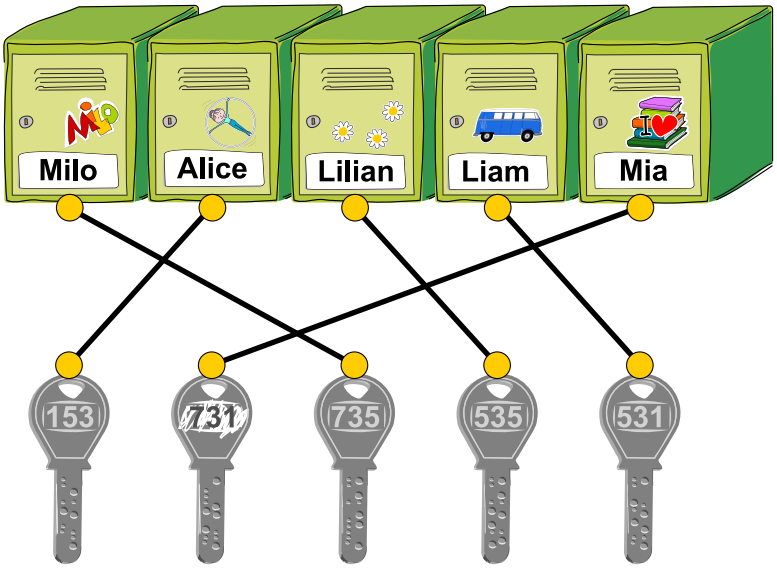
Assegna le chiavi agli armadietti corretti. Traccia delle linee tra i punti gialli.





Soluzione

Di seguito viene mostrata la soluzione corretta:



I quattro numeri conosciuti sono: 153, 735, 535, 735. Le prime tre lettere dei cinque nomi sono MIL, ALI, LIL, LIA, LIA, MIA.

Solo LIL inizia e finisce con la stessa lettera. Quindi deve includere un numero a tre cifre che inizia e finisce con la stessa cifra, e ci può essere solo un numero di questo tipo. Il numero 535 si adatta a questo modello, quindi deve appartenere a LIL. Quindi 5 sta per L e I per 3. Ora possiamo vedere che 531 deve stare per LIA, perché altrimenti non ci sono nomi che iniziano per L. Quindi 1 sta per A. Inoltre, 153 deve stare per ALI, perché altrimenti nessun nome ha una L al secondo posto. Ora solo il numero 7 e la lettera M non sono assegnati. Quindi devono stare insieme. Quindi abbiamo la seguente chiara assegnazione: 1 = A, 3 = I, 5 = L e 7 = M. Quindi 735 sta per MIL e 531 per LIA. Ora vediamo anche che la chiave con il numero graffiato appartiene a Mia e che il numero graffiato deve essere 731.

Un'altra idea per trovare l'assegnazione corretta è quella di contare la frequenza di lettere e numeri. In MIL, ALI, LIL, LIA, LIA, MIA le due lettere A e M appaiono due volte ciascuna e le lettere I e L appaiono cinque volte ciascuna. Purtroppo, questo non è ancora sufficiente per una chiara assegnazione delle lettere ai numeri. È quindi necessario fare ulteriori osservazioni, ad esempio quelle sopra descritte.

Questa è l'informatica!

Nell'informatica, i nomi e i testi sono spesso codificati con numeri.

La dichiarazione del compito indica che i numeri sulle chiavi possono essere ricavati in modo univoco dalle prime tre lettere dei rispettivi nomi. Questo funziona assegnando esattamente una cifra ad ogni lettera come codifica e utilizzando solo poche lettere. Si parla di una codifica *monoalfabetica*, poiché ogni lettera è sostituita ovunque dallo stesso carattere. D'altra parte, non è stato specificato quale

cifra è effettivamente assegnata a quale lettera. Ma la soluzione mostra come si possa trovare la corretta assegnazione con l'aiuto di alcuni indizi strutturali.

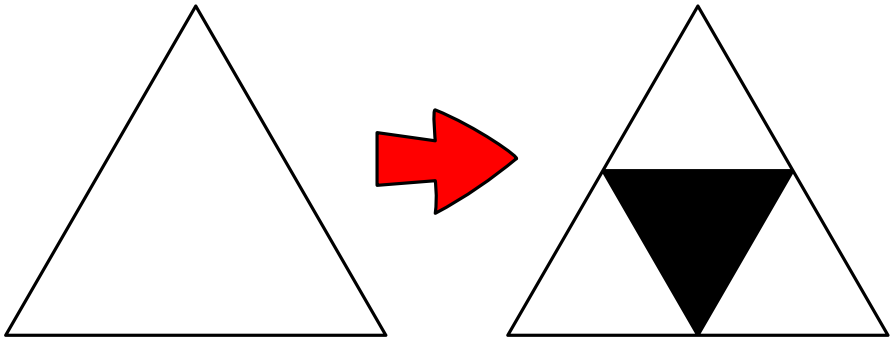
Se per la codifica non si usano solo 10 cifre, ma un simbolo per ogni lettera, allora si può utilizzare una tale sostituzione monoalfabetica come semplice codice segreto. Purtroppo il metodo di cifratura monoalfabetico non è molto sicuro, perché spesso è possibile scoprire il codice rapidamente con qualche trucco. Il compito è un esempio di questo. Fortunatamente ci sono molti sistemi di crittografia migliori. La *crittografia* è un importante sottocampo dell'informatica, in cui vengono sviluppati e analizzati vari codici segreti.

Parole chiave e siti web

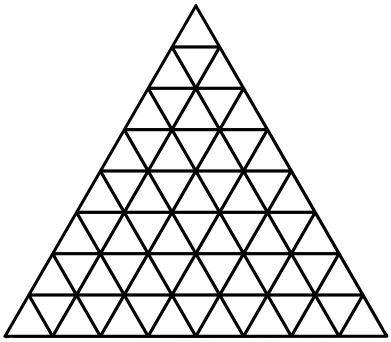
- Cifrario a sostituzione: `https://it.wikipedia.org/wiki/Cifrario_a_sostituzione`
- Crittografia: `https://it.wikipedia.org/wiki/Crittografia`

4. Triangolo di Sierpiński

Per ottenere il cosiddetto triangolo di Sierpiński, si deve prima disegnare un triangolo bianco equilatero. Poi si procede passo dopo passo. In ogni passo, ogni triangolo bianco esistente è diviso in quattro più piccoli e quello centrale è colorato di nero, come mostrato nella figura seguente:

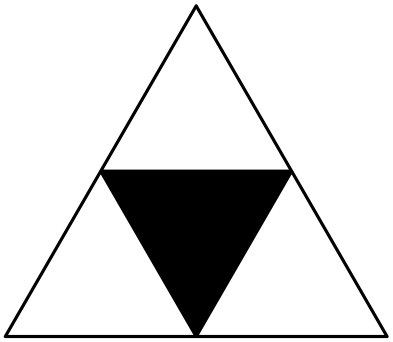


Disegna la figura che emerge dopo tre passi. Colora di nero i triangoli corretti.

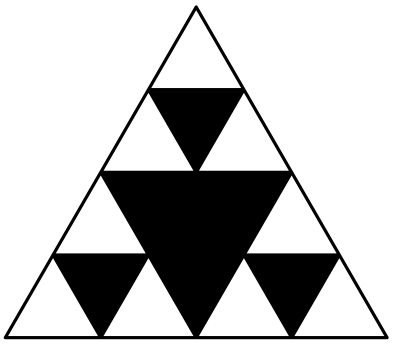


Soluzione

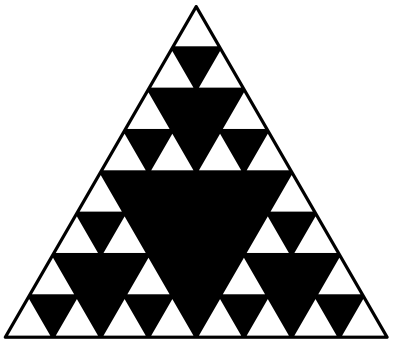
Dopo il primo passo, il triangolo centrale è nero e rimangono tre triangoli bianchi:



Nel secondo passo, questi tre triangoli parziali sono nuovamente suddivisi in quattro triangoli parziali più piccoli, di cui quello centrale è colorato di nero. Questo lascia $3 \cdot 3 = 9$ triangoli parziali bianchi più piccoli:



Nel terzo e ultimo passo, questi 9 triangoli bianchi sono divisi in quattro triangoli più piccoli e quello centrale è dipinto. La seguente figura con $9 \cdot 3 = 27$ triangoli parziali bianchi viene creata:



Questa è l'informatica!

Il triangolo di Sierpiński è un *frattale* descritto per la prima volta dal matematico polacco Wacław Franciszek Sierpiński (1882–1969) nel 1915. I frattali sono figure in cui compaiono parti sempre più piccole, simili all'intera figura. Disegnare immagini esatte dei frattali è estremamente complesso. Quando nel XX secolo apparvero i computer che potevano fare i calcoli necessari, i frattali divennero molto popolari. I frattali più noti sono la *curva di Koch* e l'insieme di *Mandelbrot*.

La costruzione del triangolo di Sierpiński è *ricorsiva* (dal latino *re-currere*: correre indietro, tornare indietro). Questo significa quanto segue: Le istruzioni per la costruzione contengono una dichiarazione che dice che bisogna rifare l'intera istruzione. Nell'esempio, questa istruzione dice: «Dividi il triangolo bianco in quattro triangoli più piccoli, colora quello centrale di nero e ripeti questa istruzione per gli altri tre triangoli.» Un passaggio attraverso le istruzioni è chiamato *passo ricorsivo*, e le istruzioni per passare attraverso le istruzioni di nuovo sono le chiamate ricorsive. (Nell'esempio, ci sono tre chiamate ricorsive per ogni passo ricorsivo.) Poiché ci sono nuove chiamate ricorsive in ogni chiamata ricorsiva, il passo ricorsivo deve essere eseguito più e più volte, il che richiede un tempo infinito. È possibile evitarlo utilizzando una *condizione di terminazione*. Nell'esempio le chiamate di ricorsione si fermano quando i triangoli diventano troppo piccoli.

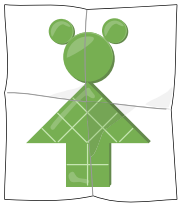
Il concetto di *algoritmo ricorsivo* è ampiamente utilizzato nell' informatica. Molti oggetti complessi — per esempio i frattali — possono essere descritti in modo compatto con la ricorsione e molti compiti complicati — per esempio il problema delle *torri di Hanoi* — possono essere risolti con algoritmi ricorsivi molto semplici.

Parole chiave e siti web

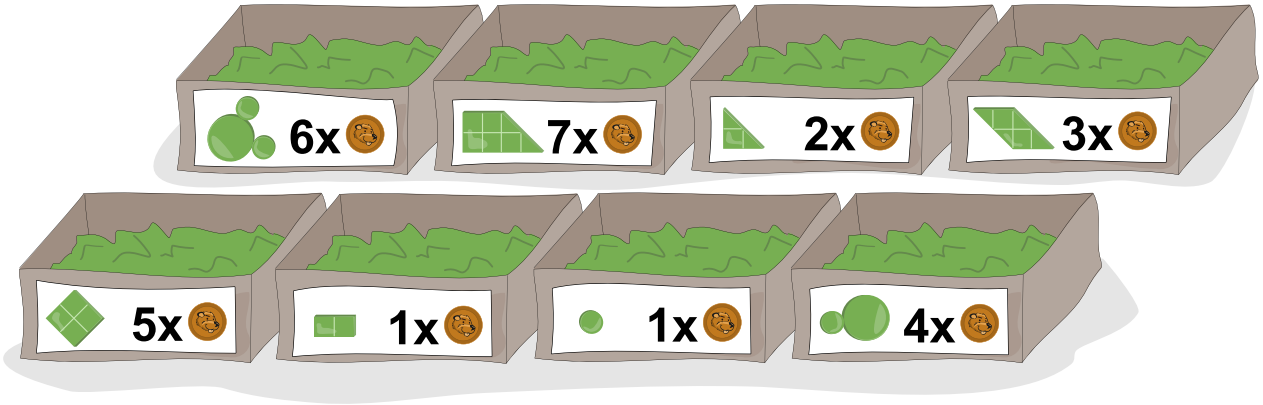
- Triangolo di Sierpiński: `https://it.wikipedia.org/wiki/Triangolo_di_Sierpiński`
- Algoritmo ricorsivo: `https://it.wikipedia.org/wiki/Algoritmo_ricorsivo`
- Frattale: `https://it.wikipedia.org/wiki/Frattale`
- Torre di Hanoi: `https://it.wikipedia.org/wiki/Torre_di_Hanoi`
- Curva di Koch: `https://it.wikipedia.org/wiki/Curva_di_Koch`
- Insieme di Mandelbrot: `https://it.wikipedia.org/wiki/Insieme_di_Mandelbrot`

5. Gioco con le tessere

Giulia vuole comprare delle tessere per realizzare questa figura:



Il negozio di giocattoli offre diverse tessere in qualsiasi quantità. I prezzi per tessera variano da 1 a 7 monete.



Le tessere possono essere girate e capovolte a piacere, ma non devono sovrapporsi.

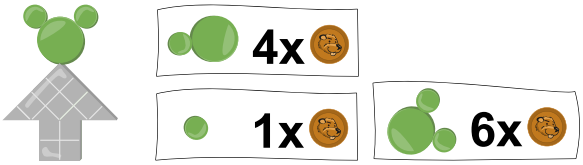
Quante monete deve spendere Giulia se sceglie l'opzione più economica?

A) 13 monete
B) 14 monete
C) 16 monete
D) 20 monete

Soluzione

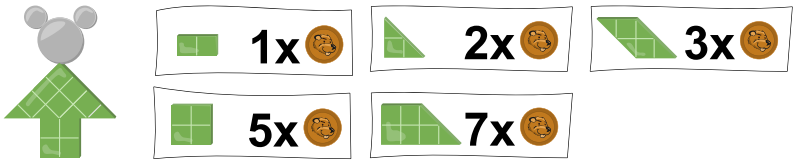
La risposta corretta è 13 monete.

Una soluzione è quella di guardare le singole parti della figura separatamente. Il modo più semplice per iniziare è quello di iniziare con la testa, che può essere posizionata solo con tessere rotonde:

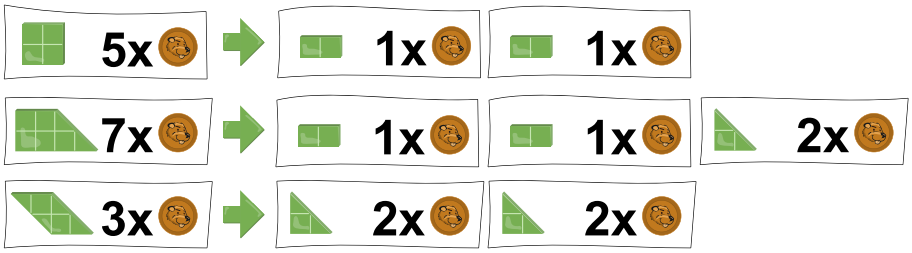


Ci sono solo due possibilità per la testa: O si usa la tessera corrispondente per 6 monete direttamente o la si mette insieme dalle altre due tessere rotonde, che insieme costano $4 + 1 = 5$ monete. La seconda opzione è più economica, quindi usiamo questa.

Il resto della figura può essere assemblato solo da tessere spigolose



Ora è possibile provare tutti i modi possibili per realizzare la figura e calcolare il prezzo per tutti. Ma questo è molto complesso. Le seguenti osservazioni sulle tessere quadrate portano più velocemente alla soluzione:



- Una tessera quadrata per 5 monete può sempre essere sostituita da due rettangoli per $1 + 1 = 2$ monete. Questo lo rende sempre più economico.
- Si potrebbe anche sostituire una tessera quadrata con due triangoli, ma sarebbe un po' più costoso con $2 + 2 = 4$ monete, quindi questa è l'opzione peggiore.

Ecco perché Giulia non compra mai un quadrato, anche se ci starebbe bene, ma sempre due rettangoli.

- Un trapezio per 7 monete può essere composto da un quadrato e da un triangolo. Se sostituiamo il quadrato con due rettangoli, $1 + 1 + 2 = 4$ monete sono sufficienti per questo trapezio.

Così Giulia non compra mai un trapezio direttamente, anche se uno ci starebbe bene, ma lo mette sempre insieme con due rettangoli e un triangolo.

- Il parallelogramma per 3 monete potrebbe essere sostituito da due triangoli più piccoli per $2 + 2 = 4$ monete. Ma questo lo rende solo più costoso, quindi non è una buona opzione.

Un parallelogramma potrebbe essere utile per Giulia, ma se questo è davvero il caso sarà rivelato solo da un approfondimento.

Versione A		**Versione B**	
	• Testa per 5 monete • Corpo di 4 rettangoli e 2 triangoli: $1+1+1+1+2+2=8$ monete		• Testa per 5 monete • Corpo di 1 parallelogrammo, 2 rettangoli e 2 triangoli: $3+1+1+2+2=9$ monete

Se Giulia non usa il parallelogramma, ha bisogno di due triangoli per posizionare i punti triangolari a sinistra e a destra della figura. Può poi spendere il resto con rettangoli, come nella versione A, che costa complessivamente $5+8=13$ monete.

Il parallelogramma si inserisce nella figura in un solo modo, come mostrato nella versione B (o la versione specchiata). Se si posiziona un parallelogramma in questo modo e il resto viene riempito di rettangoli e triangoli, la figura costa $5+9=14$ monete. Tutti gli altri posizionamenti del parallelogramma darebbero lacune che non possono essere colmate.

È quindi chiaro che 13 monete sono la soluzione più economica.

Questa è l'informatica!

Il compito di realizzare una certa figura con determinate tessere può essere estremamente complicato anche per pochissime parti. Un esempio è il gioco Tangram.

Il problema a disposizione è ancora più complicato, perché anche il prezzo totale delle tessere deve essere ottimizzato. Nell'informatica un tale problema si chiama *problema di ottimizzazione*.

Il problema è stato risolto con un importante principio dell'informatica: dividere un problema in sottoproblemi più piccoli che possono essere risolti indipendentemente l'uno dall'altro e le cui soluzioni possono poi essere messe insieme per formare una soluzione globale. In concreto, il problema è stato diviso in due sotto-problemi che possono essere risolti indipendentemente, uno per le tessere rotonde e uno per le tessere quadrate. Con le tessere spigolose, la combinazione di tessere più favorevole per un quadrato può essere riutilizzata ovunque senza doverci pensare più e più volte. Lo stesso vale per il parallelogramma.

La suddivisione in sottoproblemi indipendenti è molto importante nella programmazione. Il riutilizzo di soluzioni per problemi secondari che si verificano più di una volta può far risparmiare molto tempo. Questo si chiama principio di *modularità*. La suddivisione in sottoproblemi più piccoli è anche la base per i programmi basati sul principio «*divide et impera*» (in inglese «divide and conquer»).

Parole chiave e siti web

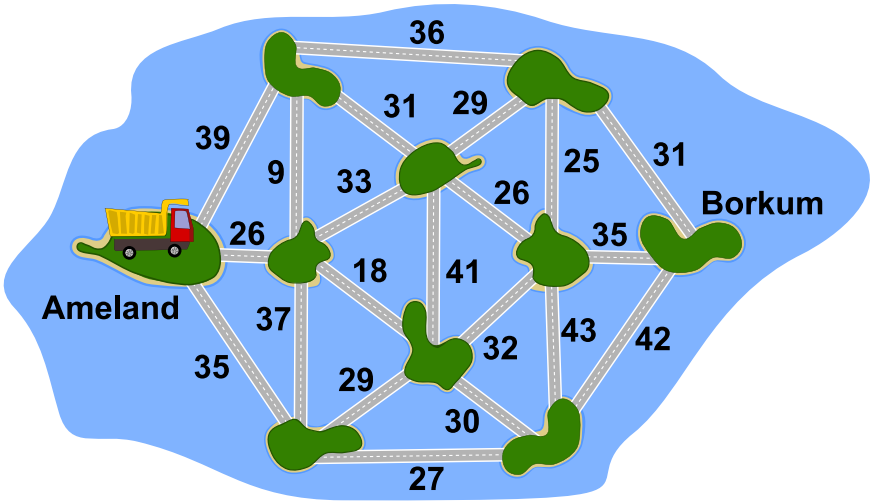
- Problema di ottimizzazione:
  `https://it.wikipedia.org/wiki/Problema_di_ottimizzazione`
- Divide et impera: `https://it.wikipedia.org/wiki/Divide_et_impera_(informatica)`
- Tangram: `https://it.wikipedia.org/wiki/Tangram`

6. L'arcipelago dei castori

L'arcipelago dei castori è composto da dieci isole collegate da ponti. Qui sotto c'è una mappa. Il numero su ogni ponte indica il peso totale massimo ammissibile in tonnellate per un camion che vuole attraversare quel ponte.

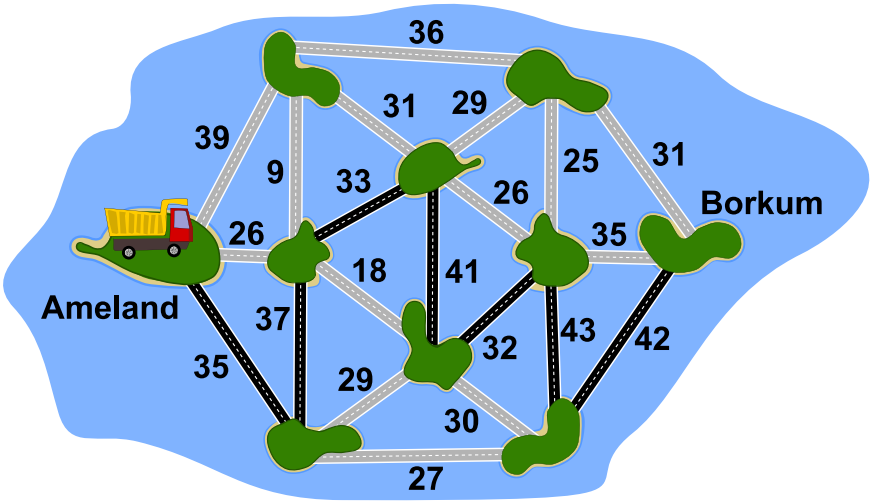
Il castoro Knuth vuole costruire una spiaggia sull'isola di Borkum. Vuole quindi trasportare quanta più sabbia possibile dall'isola di Ameland all'isola di Borkum in un solo viaggio. Non gli interessa la lunghezza del viaggio, ma non vuole passare su nessun ponte più di una volta.

Che strada deve prendere con il suo camion per arrivare a Borkum? Disegnala sulla mappa.

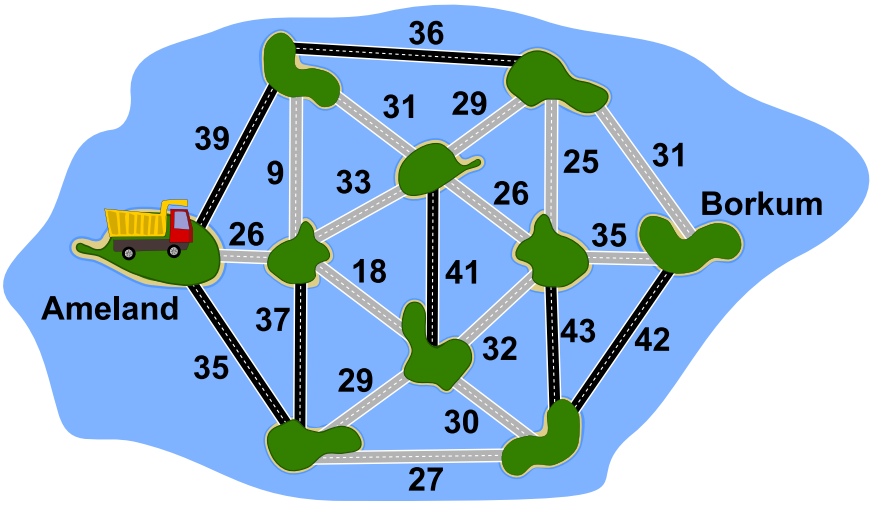


Soluzione

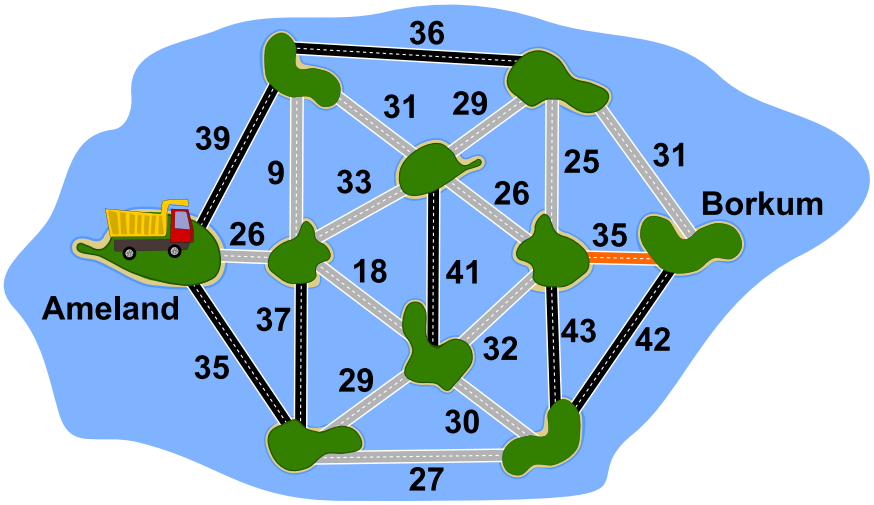
Per il viaggio, il peso totale massimo di un camion è di 32 tonnellate. Si prende il seguente percorso, ad esempio:



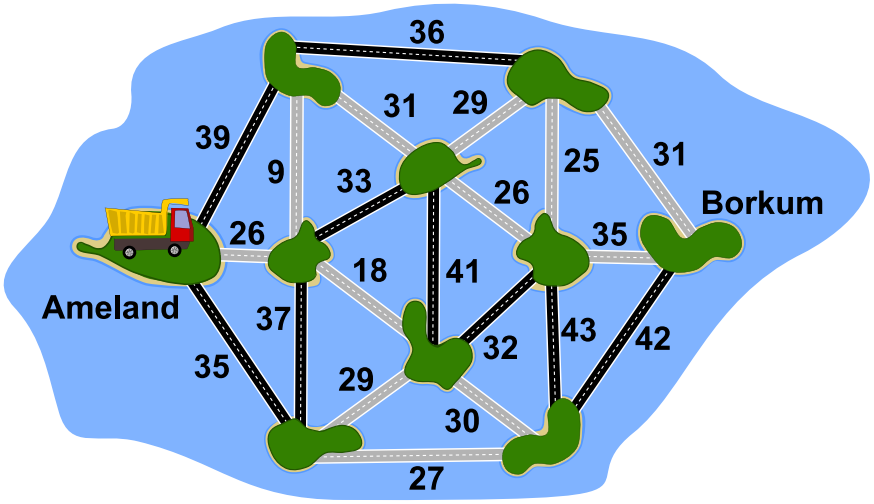
Per determinare questo, possiamo, ad esempio, per prima cosa togliere tutti i ponti dalla mappa e ordinarli in base alla loro capacità di carico. Iniziamo con quelli con la maggiore capacità di carico e li aggiungiamo alla mappa. Poi aggiungiamo quelli con la seconda maggior capacità di carico e così via. Nel seguente diagramma i ponti inseriti con le portate 43, 42, 41, 39, 39, 37, 36, 35 sono contrassegnati in nero.



Tuttavia, se inserendo un ponte dovessimo creare un cosiddetto ciclo, cioè un percorso circolare, non lo metteremmo in quanto le isole di questo ciclo sarebbero già accessibili da ponti di maggiore capacità. Nel seguente diagramma, il ponte con una capacità di carico di 35 verrebbe inserito, ma non farebbe altro che abbreviare un percorso già esistente.



Lo faremo fino a quando tutte le isole saranno collegate. Ora c'è solo una strada possibile tra ogni paio d'isole e il ponte con la capacità più piccola dà il massimo peso che stiamo cercando.



Questa è l'informatica!

Una vera e propria applicazione per la soluzione di questo compito è l'identificazione del «collo di bottiglia» (in inglese «bottleneck») nelle reti di computer, cioè la maggiore velocità di trasmissione possibile tra due computer della rete. Il compito qui riguarda il peso totale massimo di un camion in viaggio tra due isole come un collo di bottiglia. Questo è determinato dalla capacità di carico del ponte più debole. Nelle reti di computer questo sarebbe il collegamento con la larghezza di banda più bassa.

Per una soluzione, la rete può essere prima modellata, cioè semplificata, come qui presentato. Nel nostro caso, l'*algoritmo di Kruskal* crea un albero ricoprente massimo in cui il collo di bottiglia è direttamente visibile.

Parole chiave e siti web

- Albero ricoprente: `https://it.wikipedia.org/wiki/Albero_ricoprente`
- Algoritmo di Kruskal: `https://it.wikipedia.org/wiki/Algoritmo_di_Kruskal`

7. Lavagna rovinata

I castori utilizzano un codice segreto in cui ogni lettera è sostituita da un carattere completamente nuovo. Come creare i nuovi caratteri è descritto nella lavagna sottostante. Purtroppo la lavagna non è completa perché alcune parti sono state cancellate.

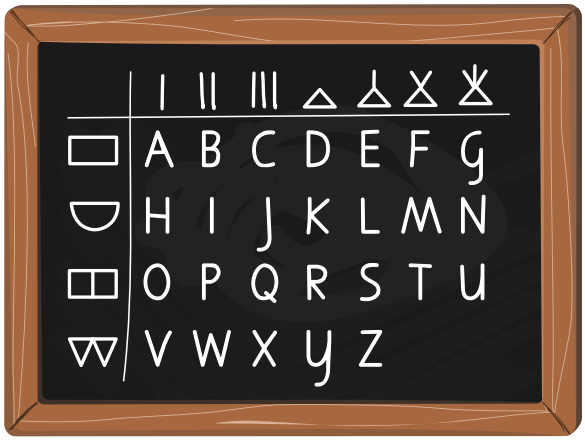
Ricostruisci il testo originale a partire dal testo cifrato attuale (decifra il testo cifrato). Quale delle 4 soluzioni è corretta?

A) INFORMATICA BELLA
B) MATEMATICA È BELLA
C) INFORMAZIONE VERA
D) INFORMAZIONI VERE

Soluzione

La risposta corretta è A), il testo decifrato è: INFORMATICA BELLA.

Ecco la lavagna di cifratura completa:



Si può facilmente ricostruire la lavagna. Le lettere dell'alfabeto latino sono disposte riga per riga da sinistra a destra. Noterete che i nuovi caratteri sono composti in modo tale che la designazione della riga corrisponde alla parte inferiore e la designazione della colonna corrisponde alla parte superiore. L'unica parte inferiore mancante nel testo cifrato è il ▭. Quindi questo carattere è il simbolo mancante della prima riga. I tre caratteri mancanti per le colonne possono essere determinati altrettanto rapidamente.

Ma non è necessario ripristinare completamente la lavagna. Potete utilizzare le lettere che potete leggere direttamente dalla tabella danneggiata. In questo modo si ottiene il seguente testo con gli spazi vuoti:

I N _ O _ _ _ _ I _ _ _ _ L L _

Con questo testo con gli spazi è possibile escludere tutte le soluzioni tranne A): B) inizia con «MA», C) termina con «ERA», D) termina con «ERE».

Un'altra soluzione è riconoscere che il testo cifrato contiene due caratteri uguali alla penultima e terzultima posizione. Quindi solo A) e B) entrano in discussione. Il primo carattere può essere chiaramente identificato come «I» nella lavagna danneggiata, il che rende chiaro che la soluzione corretta è A).

Questa è l'informatica!

Mantenere la segretezza delle informazioni e proteggere i dati è un compito che risale a 4000 anni fa. A questo scopo sono stati sviluppati e utilizzati innumerevoli linguaggi segreti. Oggi la sicurezza dei dati è uno dei temi centrali dell'informatica. Uno dei metodi per proteggere i dati da letture non autorizzate è la *crittografia*. La cifratura trasforma un testo in chiaro in un *testo cifrato*. Ricostruire il testo in chiaro dal testo cifrato si chiama *decifrare*. La scienza del testo cifrato si chiama *crittologia*.

Le culture antiche utilizzavano per lo più scritture segrete, che venivano create codificando le lettere con altre lettere o con caratteri completamente nuovi. Il cifrario seguente è stato sviluppato specialmente per la competizione del castoro informatico, ma si basa su un concetto dell'antica Palestina. La regola di sicurezza dell'epoca era che si usavano solo codici segreti che si potevano imparare facilmente a memoria. Mantenere una descrizione scritta del codice segreto era considerato un rischio troppo grande. Una tabella, come si usa qui, è facile da imparare a memoria. Il famoso codice segreto dei massoni si basa su questo principio.

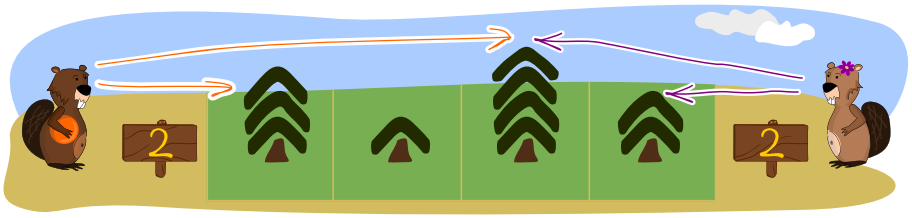
Parole chiave e siti web

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

8. 4×4 sudoku con gli alberi

I castori piantano sedici alberi (quattro alberi di altezza 4, quattro alberi di altezza 3, quattro alberi di altezza 2 e quattro alberi di altezza 1) in un campo di alberi 4 x 4, seguendo le seguenti regole:

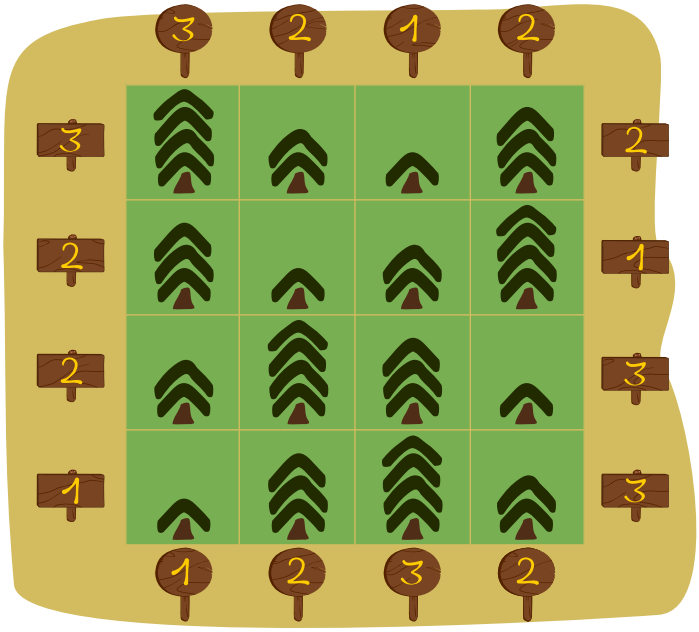
- In ogni riga (riga orizzontale) c'è esattamente un albero di ogni altezza;
- In ogni colonna (riga verticale) c'è esattamente un albero di ogni altezza.



Quando i castori guardano una fila di alberi da un lato, **non** possono vedere gli alberi più bassi nascosti dietro gli alberi più alti. Alla fine di ogni fila di alberi c'è un cartello che indica quanti alberi un castoro può vedere da quel punto. Questi cartelli con il numero di alberi visibili sono posizionati intorno al campo di alberi.

Kubko ha cercato di trasferire la descrizione del campo su un foglio di carta. Ha trasferito correttamente i numeri dei cartelli, ma si è sbagliato con quattro alberi.

Cerchia le quattro posizioni con gli alberi inseriti in modo errato e annota di lato l'altezza corretta che l'albero dovrebbe avere



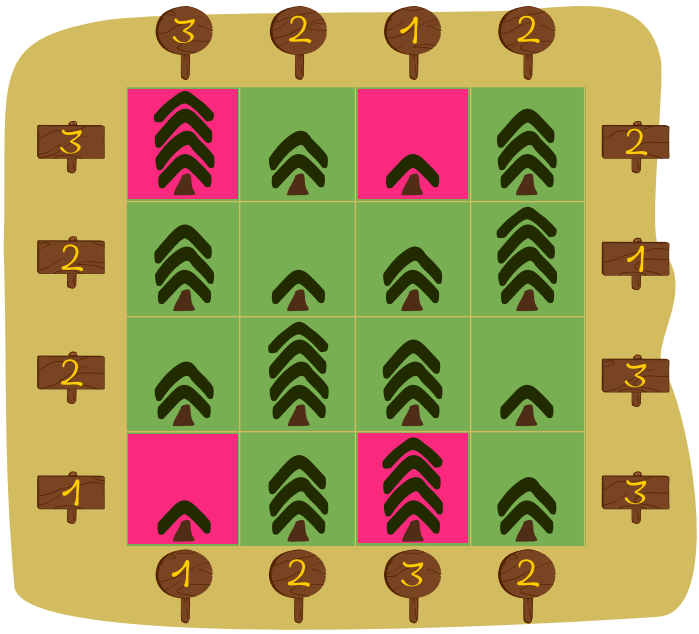
Soluzione

Prima di tutto si nota che entrambe le regole del «Sudoku» sono state seguite: C'è esattamente un albero di ogni altezza in ogni fila.

Poi si può vedere per quali righe i numeri sui pannelli sono corretti e per quali no. Si può vedere che i numeri sono corretti per le righe 2 e 3 e per le colonne 2 e 4. Per le altre righe i numeri non sono corretti, chiamiamo queste *righe problematiche*.

Ma questo non basta. Vogliamo sapere quali sono le posizioni che causano i numeri sbagliati. Per fare questo, si nota che ci sono esattamente quattro posizioni che sono contemporaneamente in una riga problematica e in una colonna problematica. Queste sono le quattro posizioni in cui le righe problematiche (cioè 1 e 4) si intersecano con le colonne problematiche (cioè 1 e 3).

Se si scambiano le coppie di alberi in questi quattro incroci problematici (contrassegnati in rosso sotto) tra le righe o le colonne, si ottiene la soluzione corretta.



Che questa sia effettivamente l'unica soluzione possibile può essere vista come segue: Esattamente quattro alberi sono sbagliati. Se un albero viene cambiato in una posizione, è necessario cambiarne almeno altri due per mantenere la regola del Sudoku soddisfatta, ovvero: un albero nella riga corrispondente e uno nella colonna. Quindi ci sono già tre cambiamenti. Le ultime due modifiche forzano di nuovo un'altra modifica nella riga o colonna corrispondente. Poiché in totale possono essere apportate solo quattro modifiche, queste due devono ora coincidere. Ciò è possibile solo se le quattro posizioni con le modifiche sono disposte in un rettangolo. Poiché è necessario apportare almeno una modifica in ogni riga problematica, la soluzione di cui sopra è l'unica possibile.

Questa è l'informatica!

Questo compito si concentra su tre competenze di base degli informatici.

La prima è quella di trovare una soluzione che rispetti i vincoli indicati, o di correggere una soluzione proposta, se necessario.

In secondo luogo, si tratta della capacità di ricostruire gli oggetti a partire da informazioni parziali attraverso la loro rappresentazione. Questo è legato alla generazione di oggetti (*rappresentazioni di oggetti*) a partire da informazioni limitate disponibili, se si conosce la regolarità dell'oggetto. Tali procedure possono essere utilizzate anche per la *compressione*.

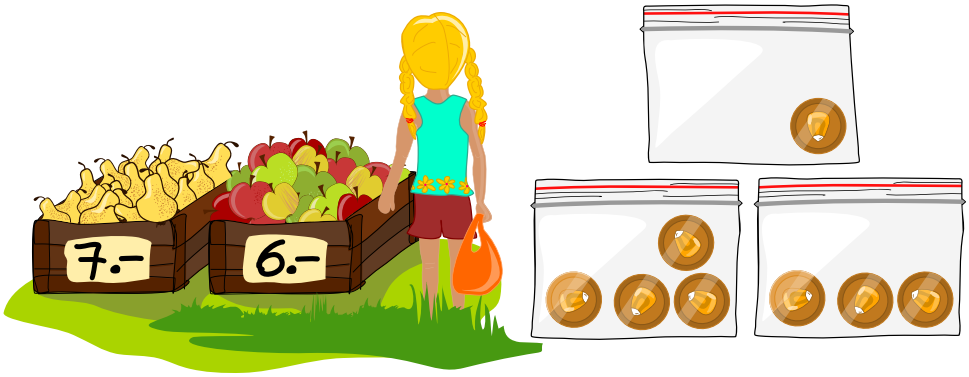
In terzo luogo, tali campi ad albero con etichette possono essere utilizzati per generare *codici autoverificanti*. Eventuali errori che si verificano durante l'inserimento dei dati o il trasporto delle informazioni possono poi essere rilevati automaticamente o addirittura corretti.

Parole chiave e siti web

- Sudoku
- Rappresentazioni di oggetti
- Compressione: `https://it.wikipedia.org/wiki/Compressione_dei_dati`
- Rilevazione e correzione d'errore: `https://it.wikipedia.org/wiki/Rilevazione_e_correzione_d'errore`

9. Sacchetto per i soldi

A Bina piace andare a nuotare. Mette i suoi soldi in sacchetti impermeabili in modo che il metallo non inizi ad arrugginire. Ieri Bina aveva con sé tre sacchetti con 1, 3 e 4 monete. Con queste monete poteva pagare una pera in modo esatto (cioè senza resto) tenendo i sacchetti chiusi, ma non una mela.



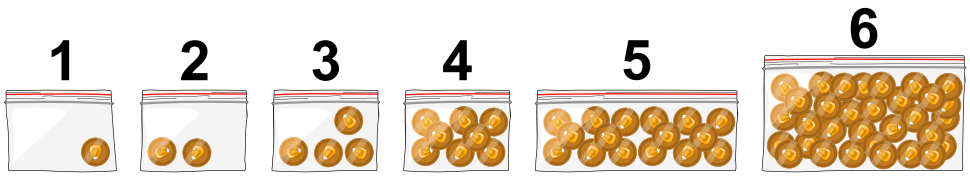
Oggi Bina ha con sé 63 monete identiche. Vuole dividerle in sacchetti diversi in modo da poter pagare qualsiasi importo compreso tra 1 e 63 monete senza dover aprire i sacchetti.

Qual è il numero più piccolo di sacchetti di cui Bina ha bisogno?

A) 4 sacchetti
B) 5 sacchetti
C) 6 sacchetti
D) 7 sacchetti
E) 8 sacchetti
F) 15 sacchetti
G) 16 sacchetti
H) 31 sacchetti
I) 32 sacchetti o di più

Soluzione

La risposta corretta è C) 6 sacchetti:



Bina può dividere le monete tra i 6 sacchetti come segue:

- Sachetto 1: 1 moneta
- Sachetto 2: 2 monete
- Sachetto 3: 4 monete
- Sachetto 4: 8 monete
- Sachetto 5: 16 monete
- Sachetto 6: 32 monete

Bina ha quindi un totale di $1 + 2 + 4 + 8 + 16 + 32 = 63$ monete nei sacchetti e può pagare qualsiasi importo totale da 1 a 63 monete in sacchetti chiusi.

Per pagare 13 monete, ad esempio, può pagare con i sacchetti 1, 3 e 4.

La tabella seguente mostra come ogni importo totale può essere pagato adeguatamente con la giusta selezione dei sacchetti. Una cella contiene un 1 se Bina usa il sacchetto corrispondente per pagare, e 0 in caso contrario.

Importo	**32**	**16**	**8**	**4**	**2**	**1**	**Importo**	**32**	**16**	**8**	**4**	**2**	**1**
0	0	0	0	0	0	0	**32**	1	0	0	0	0	0
1	0	0	0	0	0	1	**33**	1	0	0	0	0	1
2	0	0	0	0	1	0	**34**	1	0	0	0	1	0
3	0	0	0	0	1	1	**35**	1	0	0	0	1	1
4	0	0	0	1	0	0	**36**	1	0	0	1	0	0
5	0	0	0	1	0	1	**37**	1	0	0	1	0	1
6	0	0	0	1	1	0	**38**	1	0	0	1	1	0
7	0	0	0	1	1	1	**39**	1	0	0	1	1	1
8	0	0	1	0	0	0	**40**	1	0	1	0	0	0
9	0	0	1	0	0	1	**41**	1	0	1	0	0	1
10	0	0	1	0	1	0	**42**	1	0	1	0	1	0
11	0	0	1	0	1	1	**43**	1	0	1	0	1	1
12	0	0	1	1	0	0	**44**	1	0	1	1	0	0
13	0	0	1	1	0	1	**45**	1	0	1	1	0	1
14	0	0	1	1	1	0	**46**	1	0	1	1	1	0
15	0	0	1	1	1	1	**47**	1	0	1	1	1	1
16	0	1	0	0	0	0	**48**	1	1	0	0	0	0
17	0	1	0	0	0	1	**49**	1	1	0	0	0	1
18	0	1	0	0	1	0	**50**	1	1	0	0	1	0
19	0	1	0	0	1	1	**51**	1	1	0	0	1	1
20	0	1	0	1	0	0	**52**	1	1	0	1	0	0
21	0	1	0	1	0	1	**53**	1	1	0	1	0	1
22	0	1	0	1	1	0	**54**	1	1	0	1	1	0
23	0	1	0	1	1	1	**55**	1	1	0	1	1	1
24	0	1	1	0	0	0	**56**	1	1	1	0	0	0
25	0	1	1	0	0	1	**57**	1	1	1	0	0	1
26	0	1	1	0	1	0	**58**	1	1	1	0	1	0
27	0	1	1	0	1	1	**59**	1	1	1	0	1	1
28	0	1	1	1	0	0	**60**	1	1	1	1	0	0
29	0	1	1	1	0	1	**61**	1	1	1	1	0	1
30	0	1	1	1	1	0	**62**	1	1	1	1	1	0
31	0	1	1	1	1	1	**63**	1	1	1	1	1	1

Con meno di 6 sacchi Bina non può raggiungere il suo obiettivo. Può usare o meno ogni sacchetto quando paga, quindi ci sono esattamente due possibilità per ogni sacchetto. Con solo 5 o anche meno sacchi, avrebbe al massimo $2^5 = 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 32$ combinazioni possibili. Quindi potrebbe pagare al massimo 32 diversi importi totali, che non è sufficiente per tutti gli importi totali fino a 63 monete.

Questa è l'informatica!

Questo compito riguarda i *numeri binari*. I numeri binari sono studiati in matematica e informatica in diversi modi. La matematica si occupa principalmente delle loro proprietà, mentre l'informatica si occupa maggiormente delle loro applicazioni. I computer utilizzano numeri binari per rappresentare informazioni di tipo molto diverso: Documenti, immagini, voci, video e numeri, anche i programmi e le applicazioni che tutti noi utilizziamo sono codificati come numeri binari. L'unità è un *bit* (dall'inglese «*binary digit*»), che può assumere il valore 0 o 1. Quindi un bit da solo può distinguere solo due possibilità. Con due bit, tuttavia, si possono distinguere quattro possibilità: 00, 01, 10 e 11. In questo compito, Bina utilizza 6 bit (sacchetti) per rappresentare $2^6 = 64$ importi diversi.

Nei computer, i bit sono di solito raggruppati in gruppi di otto; tale gruppo di otto è chiamato *byte*. Un byte può rappresentare $2^8 = 256$ numeri diversi, da 0 a 255.

Parole chiave e siti web

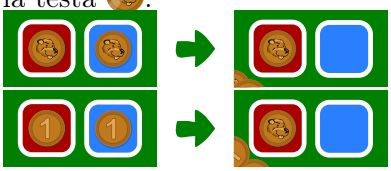
- Sistema numerico binario: `https://it.wikipedia.org/wiki/Sistema_numerico_binario`

10. Las Bebras

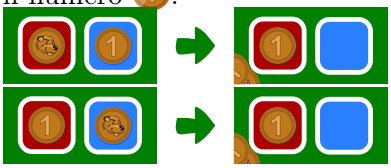
Al Casinò «Las Bebras» Gloria può giocare da John utilizzando delle monete. Gloria ha 4 monete con rappresentato sul davanti una testa, e sul retro un numero 1. Gloria lancia le prime 2 monete e ne mette una sul quadrato rosso e l'altra sul quadrato blu.

John scambia le due monete con una nuova moneta che mette sul campo rosso.

- Se le due monete sono uguali, John mette la nuova moneta sul campo rosso lasciando visibile la testa.



- Se le due monete sono diverse, John mette la nuova moneta sul campo rosso lasciando visibile il numero 1.



Gloria ora lancia un'altra moneta e la mette sul quadrato blu, John la sostituisce di nuovo secondo le regole di cui sopra e così via fino a quando Gloria non avrà giocato tutte e 4 le monete. La partita finisce quando John mette l'ultima moneta sul campo rosso. Se si vede il numero 1 Gloria vince!

Gloria gioca le 4 monete in ordine da sinistra a destra. In quale caso vince Gloria?

A) 

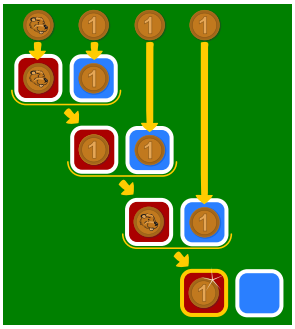
B) 

C) 

D) 

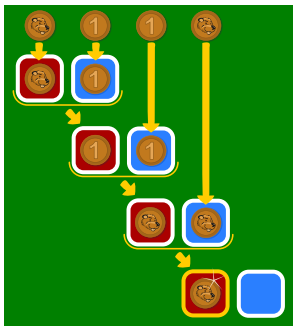
Soluzione

La risposta corretta è C). Solo per il caso C) alla fine si ottiene una moneta nel campo rosso con il numero  visibile.

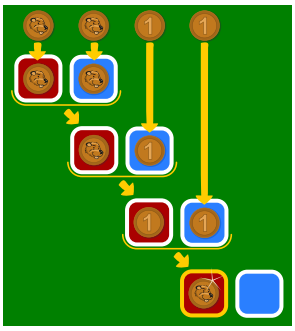


In tutti gli altri casi alla fine si ottiene una moneta nel campo rosso con visibile la testa  .

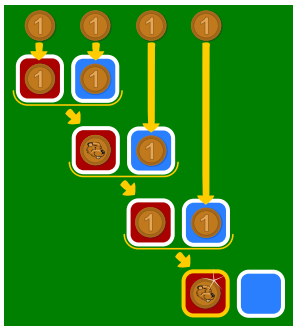
A)



B)



D)



Per ognuna delle 4 monete che Gloria gioca, ci sono 2 modi per piazzarle ( oppure ), quindi con 4 monete si può giocare un totale di $2^4 = 16$ sequenze. Se c'è un numero pari di monete con testa (o numero) in fila, alla fine del gioco la moneta viene posta con visibile la testa  sul campo rosso. Se c'è un numero dispari di monete con testa (o numero) in fila, la moneta viene messa sul campo rosso alla fine della partita con il numero visibile  . Un numero dispari di monete con testa (o numero) in fila rappresenta quindi la «sequenza vincente». Ci sono esattamente 8 sequenze con un numero dispari e 8 sequenze con un numero pari.

Questa è l'informatica!

Poiché i computer sono macchine elettroniche, l'elettricità viene utilizzata per rappresentare le informazioni che immettiamo al loro interno. Quando l'elettricità scorre, diciamo che è accesa. Se non c'è elettricità, diciamo che è spenta. Gli informatici di solito rappresentano questi due stati con i due numeri 0 e 1, che chiamiamo «rappresentazione binaria». Un'unità di informazione è chiamata «bit».

Possiamo eseguire operazioni su tali bit e combinarli, così come con le due posizioni delle monete (testa o numero).

Una di queste operazioni si chiama disgiunzione esclusiva (oppure XOR per «eXclusive OR» in inglese) ed è quella presentata in questo compito. Funziona così:

0 XOR 0 = 0
0 XOR 1 = 1
1 XOR 0 = 1
1 XOR 1 = 0

Un esempio d'uso quotidiano: su entrambi i lati di una scala ci sono due interruttori della luce che accendono o spengono la stessa luce. Se entrambi gli interruttori della luce sono alzati, la luce è accesa e se entrambi sono abbassati, la luce è anche accesa. Se uno è alzato e l'altro è abbassato, la luce è spenta.

Un gate XOR è un'implementazione elettronica del funzionamento XOR nei computer. Un gate XOR da un risultato di 1 alla sua uscita se esattamente uno dei suoi due ingressi è 1. Se entrambi gli ingressi sono uguali, il risultato in uscita è 0.

Nell' informatica, l'operazione XOR ha diverse applicazioni come per esempio:

- Ci dice se due bit sono uguali o diversi.
- Ci dice se il numero di bit di 1 è pari o dispari (lo XOR di una sequenza di bit è «vero» esattamente quando un numero dispari è «vero»).
- Nella crittografia, l'operazione XOR viene utilizzata nella crittografia simmetrica con i cosiddetti «one-time pad».

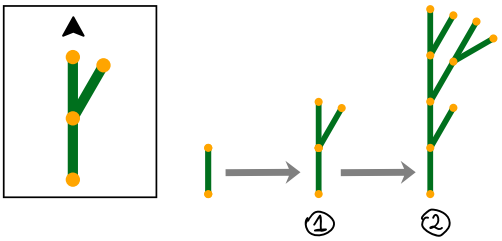
Parole chiave e siti web

- Operazione binaria: `https://it.wikipedia.org/wiki/Operazione_binaria`
- XOR: `https://it.wikipedia.org/wiki/Porta_logica#EXOR`
- Porta logica: `https://it.wikipedia.org/wiki/Porta_logica`

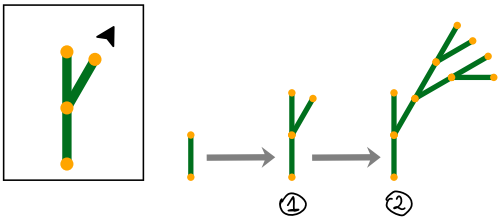
11. Alberi digitali

Un albero digitale cresce dal seguente pezzo di albero singolo:  .

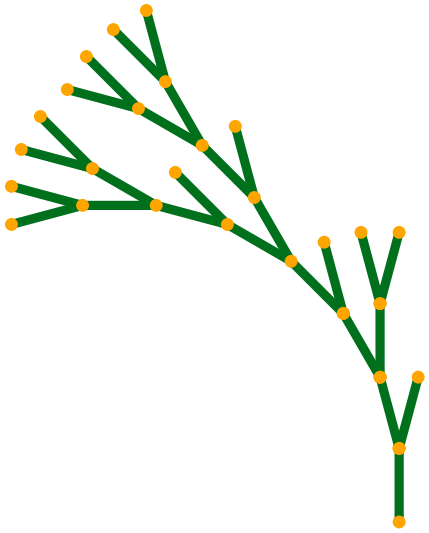
Cresce gradualmente secondo una regola di crescita predeterminata.
La regola della crescita specifica come un pezzo di albero può essere sostituito da una struttura di nuovi pezzi di albero. In ogni passo, ogni pezzo di albero viene sostituito in questo modo. La punta di una freccia indica dove e in quale direzione vengono messi insieme i pezzi dell'albero.



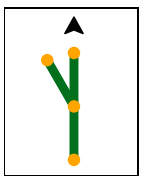
A destra ci sono due esempi di una regola di crescita e le corrispondenti prime due fasi di crescita.

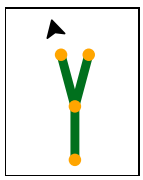


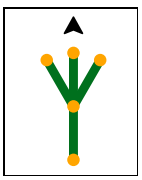
Il seguente albero digitale è cresciuto in 3 fasi di crescita:

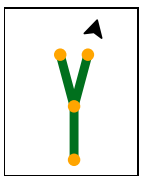


Secondo quale regola di crescita è cresciuto l'albero digitale?

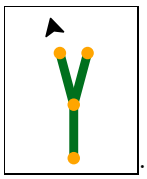
A) 

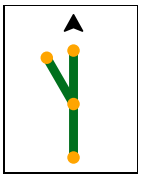
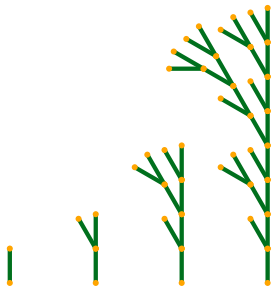
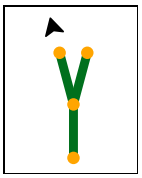
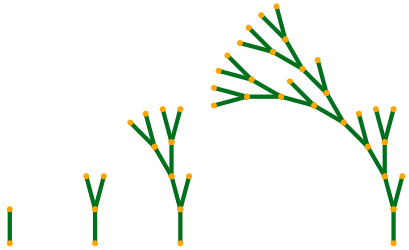
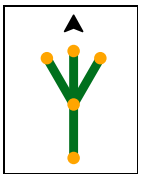
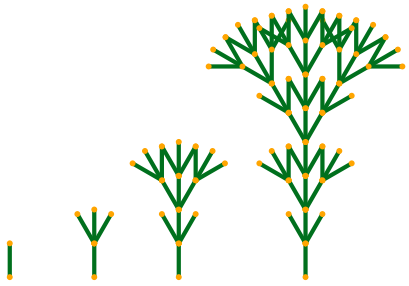
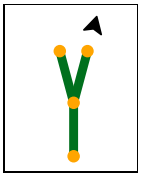
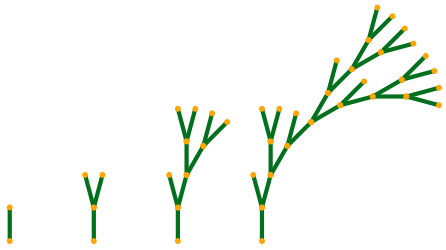
B) 

C) 

D) 

Soluzione

La risposta corretta è B)  .

Regola di crescita	3 fasi di crescita	Descrizione
		Il resto dell'albero è sempre aggiunto al ramo che punta verso l'alto, in linea retta. Forma così un tronco dritto con rami che puntano solo a sinistra.
		Il resto dell'albero viene sempre aggiunto al ramo superiore sinistro. L'albero quindi si inclina a sinistra.
		Il resto dell'albero è sempre aggiunto al centro, in linea retta. I due rami a sinistra e a destra formano una struttura uniforme e simmetrica.
		Il resto dell'albero viene sempre aggiunto al ramo superiore destro. L'albero quindi si inclina a destra.

Questa è l'informatica!

Nel compito si può vedere come l'applicazione ripetuta di una regola di crescita molto semplice possa creare figure complicate. Tali figure, che consistono di parti simili all'intera figura, sono chiamate *frattali*. I frattali sono molto spesso utilizzati dai computer, ad esempio per creare paesaggi o effetti speciali per i film.

In biologia, i cosiddetti *sistemi di Lindenmayer* (dal nome del biologo Aristid Lindenmayer) vengono utilizzati per simulare la crescita delle piante. Anche i frattali vengono creati in questo processo. Nel compito abbiamo visto quattro esempi molto semplici di un sistema di Lindenmayer.

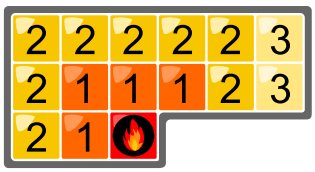
Gli alberi nel compito sono creati applicando una regola ad ogni pezzo di albero, e poi applicandola di nuovo ai pezzi di albero risultanti e così via. Tali processi sono chiamati *ricorsivi*. Il concetto degli algoritmi ricorsivi è importante nell'informatica. Con questo tipo di algoritmo è possibile descrivere molte cose complicate in modo molto semplice.

Parole chiave e siti web

- Frattale: `https://it.wikipedia.org/wiki/Frattale`
- Sistema di Lindenmayer: `https://it.wikipedia.org/wiki/Sistema_di_Lindenmayer`
  `http://paulbourke.net/fractals/lsys/`
- Algoritmo ricorsivo: `https://it.wikipedia.org/wiki/Algoritmo_ricorsivo`

12. Riscaldamento a pavimento

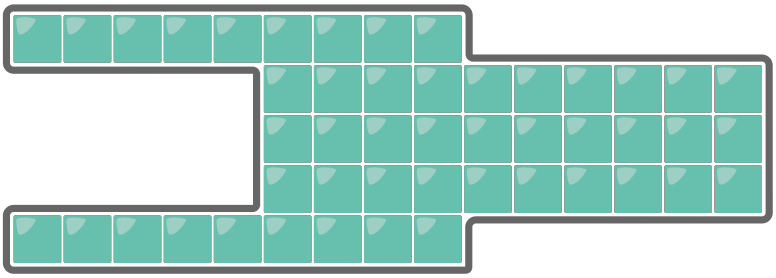
A Luis non piace vestirsi al mattino nel bagno freddo, quindi vuole che nella nuova casa venga installato il riscaldamento a pavimento. Il tecnico del riscaldamento gli consiglia l'innovativo riscaldamento a pavimento «hotspot»: un hotspot viene installato direttamente sotto una piastrella. Se l'hotspot è acceso, la piastrella è immediatamente calda.



In un minuto il calore si diffonde su tutte le piastrelle adiacenti, cioè tutte le piastrelle che toccano la piastrella già riscaldata su un bordo o un angolo. I numeri su ogni piastrella indicano dopo quanti minuti è calda.

Luis vuole far installare 4 hotspot nel suo nuovo bagno in modo che tutte le piastrelle si riscaldino il più velocemente possibile quando vengono accese.

Sotto quali 4 piastrelle il tecnico del riscaldamento deve installare i 4 hotspots ?

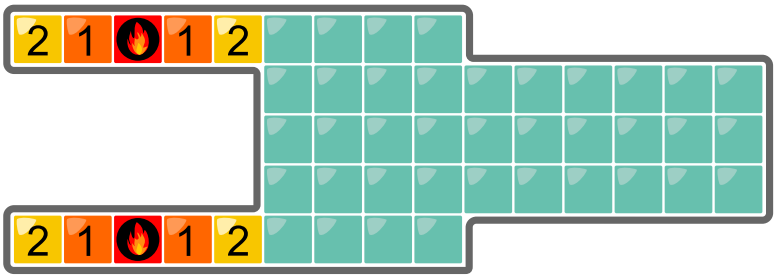


Soluzione

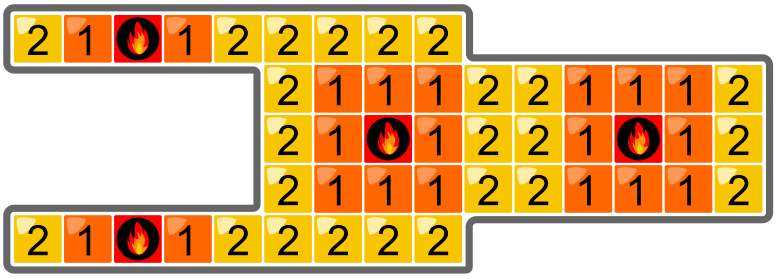
Se i 4 hotspot sono installati come mostrato nell'immagine sottostante, tutte le piastrelle del bagno si riscalderanno entro 2 minuti dall'accensione.

Questo è ottimale, perché è impossibile riscaldare tutte le piastrelle in 1 minuto con 4 punti caldi. Ogni hotspot può riscaldare infatti un massimo di 9 piastrelle nel primo minuto, cioè la propria piastrella e fino a 8 piastrelle intorno ad essa. Quindi 4 punti caldi insieme riscaldano un massimo di $4 \cdot 9 = 36$ piastrelle nel primo minuto. Il bagno ha 48 piastrelle in totale. Quindi 1 minuto non è sufficiente. Ma con 2 minuti potrebbe funzionare, visto che teoricamente fino a $4 \cdot 25 = 100$ piastrelle potrebbero essere riscaldate.

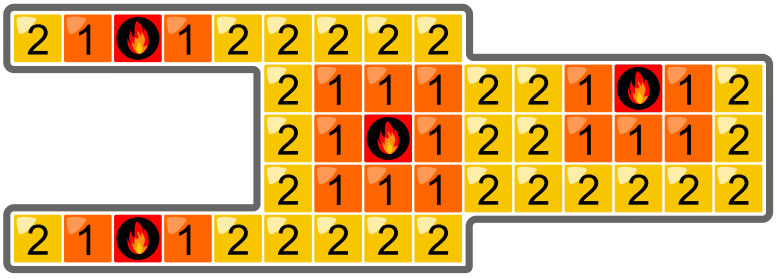
È una buona idea iniziare con i due corridoi a sinistra quando si distribuiscono gli hotspot. Con un punto caldo al centro di ogni corridoio, tutte le piastrelle del corridoio vengono riscaldate in 2 minuti:

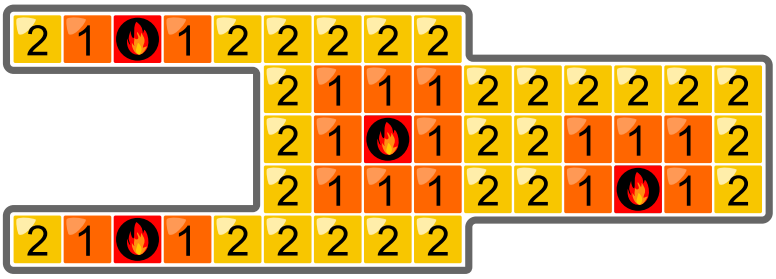


Possiamo poi collocare gli altri due hotspot in questo modo:



Sono possibili anche i seguenti due collocamenti:





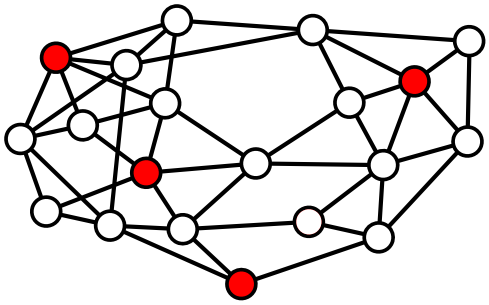
Se il bagno avesse una forma diversa, 2 hotspot potrebbero essere sufficienti per riscaldare l'intero bagno in 2 minuti con la stessa area.

Questa è l'informatica!

Il problema risolto in questo compito è legato ad un ben noto problema di ottimizzazione: Qui cerchiamo un piccolo gruppo di *nodi* in un *grafo* chiamato *insieme dominante*.

Un insieme dominante è definito come segue: Ogni nodo del grafo deve essere contenuto nel insieme dominante o avere un vicino che è contenuto nel insieme dominante. Le piastrelle del bagno possono essere interpretate come nodi. I nodi sono collegati con archi quando la piastrella vicina viene riscaldata dopo un minuto. Un set dominante del grafo risultante indica quindi i luoghi in cui è possibile posizionare gli hotspot per riscaldare il bagno in 2 minuti.

In generale è molto difficile trovare un insieme dominante minimo. Per i grafi speciali esistono algoritmi efficienti. Il disegno seguente mostra un esempio. Come si può vedere, ogni nodo bianco è vicino ad almeno un nodo rosso. Quindi i nodi rossi sono un insieme dominante.



Un'applicazione tipica è il posizionamento di hotspot WiFi in un grande edificio. I nodi del grafo sono le singole stanze. Due di esse sono adiacenti nel grafo se entrambe le stanze si trovano nel raggio d'azione di un hotspot. Le stanze che formano un insieme dominante minimo sono luoghi adatti per gli hotspot.

Parole chiave e siti web

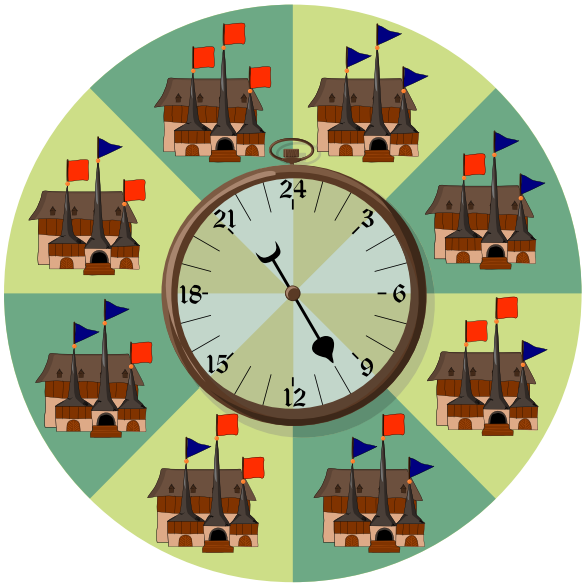
- Insieme dominante

13. Castori rilassati

In un villaggio idilliaco, i castori sono molto rilassati nel gestire il loro tempo. Suddividono la giornata in soli 8 periodi di 3 ore ciascuno. Il periodo di tempo attuale è indicato dal municipio con tre bandiere, come mostrato nell'immagine sottostante. Si utilizzano due diversi tipi di bandiere, un quadrato rosso e un triangolo blu.

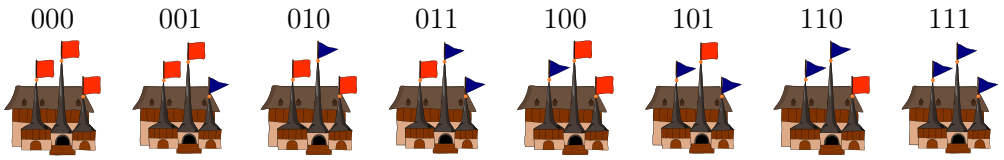
La sistemazione attuale richiede un solo cambio di bandiera a quasi tutte le transizioni. Solo a mezzanotte devono essere cambiate tre bandiere contemporaneamente. I castori vogliono introdurre una sistemazione più conveniente, dove bisogna cambiare una sola bandiera alla volta.

Trova una sistemazione più conveniente. Disegna gli schemi delle tre bandiere vicino ad ogni orario.



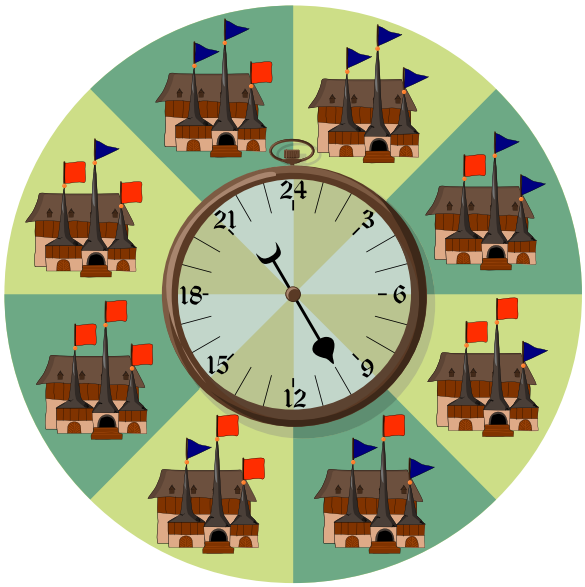
Soluzione

Gli 8 schemi possono essere rappresentati anche da numeri binari a tre cifre: 0 sta per un quadrato rosso e 1 per un triangolo blu.



Quindi gli 8 schemi sono 000, 001, 010, 011, 011, 100, 101, 101, 110, 111, ecc. Ora dobbiamo organizzare questi numeri in modo tale che tutti i numeri adiacenti differiscano solo in un posto, così come il primo e l'ultimo numero.

Questo può essere ottenuto per tentativi ed errori. Una possibile soluzione è 111, 011, 001, 001, 101, 101, 100, 000, 010, 110. Ecco l'orologio corrispondente:



Si trova sistematicamente una soluzione con il seguente metodo:

Per prima cosa, guardiamo solo i numeri che iniziano con due zeri, cioè 000 e 001. Qui ci sono due possibili configurazioni, che soddisfano entrambe la condizione sopra descritta. Scegliamo 000, 001.

Ora scriviamo di nuovo questi due numeri in ordine inverso (001, 000), ma cambiamo la seconda cifra da 0 a 1 (011, 010). Si ottiene così la sequenza dei numeri 000, 001, 011, 010, che soddisfa di nuovo la condizione.

Scriviamo questa nuova sequenza di numeri di nuovo all'indietro, ma cambiamo la prima cifra da 0 a 1 ovunque, così otteniamo 000, 001, 011, 011, 010, 110, 111, 111, 101, 100, che soddisfa di nuovo la condizione. Quindi abbiamo la soluzione desiderata.

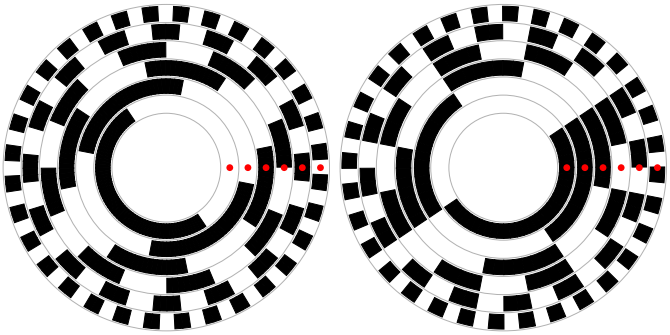
Questo metodo (invertendo la sequenza di numeri esistente e cambiando la cifra successiva più alta da 0 a 1) può essere continuato per tutto il tempo che si desidera per ottenere tali composizioni per

quante bandiere si desidera, invece che solo tre. Si può considerare il motivo per cui questo metodo funziona sempre e perché vengono utilizzati tutti i possibili modelli.

Questa è l'informatica!

Una tale composizione di numeri binari si chiama *codice di Gray* e ha molte applicazioni. Ad esempio, il fatto che solo un bit cambi tra numeri adiacenti può aiutare a risparmiare energia. In ogni caso, cambiare più bit richiede più energia, e con la normale enumerazione ascendente dei numeri binari, molti bit cambiano molto spesso allo stesso tempo.

Una famosa applicazione del codice di Gray in ingegneria è la misura degli angoli di un giradischi. Disegniamo il codice di Gray sul disco come mostrato nella figura in basso a sinistra, bianco per 0 e nero per 1. I punti rossi sono sensori posizionati su una linea e possono distinguere tra bianco e nero. I sensori possono così leggere un numero binario (una parola in codice) che codifica l'angolo di rotazione attuale del disco.



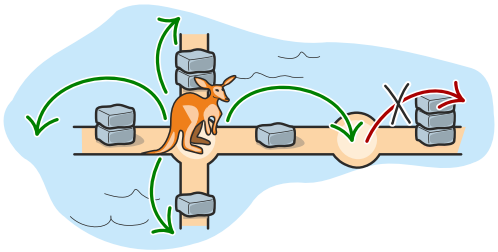
Nella figura a sinistra vediamo che il quarto sensore si trova esattamente al limite tra il bianco e il nero. Quindi il sensore legge o 001010 o 001110, entrambe le opzioni sono accettabili, in quanto l'angolo reale è esattamente al centro. Se non abbiamo un codice di Gray, il tutto sembra molto peggio. Guardiamo il normale codice binario nella figura a destra. Qui le parole di codice 111010 e 111001 si susseguono. Se i sensori sono esattamente nel mezzo, gli ultimi due sensori non possono decidere tra il bianco e il nero, quindi si potrebbe leggere il numero 111011, che è già a una certa distanza. Nel caso peggiore, i sensori si troverebbero al confine tra la parola di codice completamente bianca 000000 e la parola di codice completamente nera 111111, nel qual caso ogni sensore può passare arbitrariamente da 0 a 1, il che rende la misura dell'angolo completamente inutilizzabile.

Parole chiave e siti web

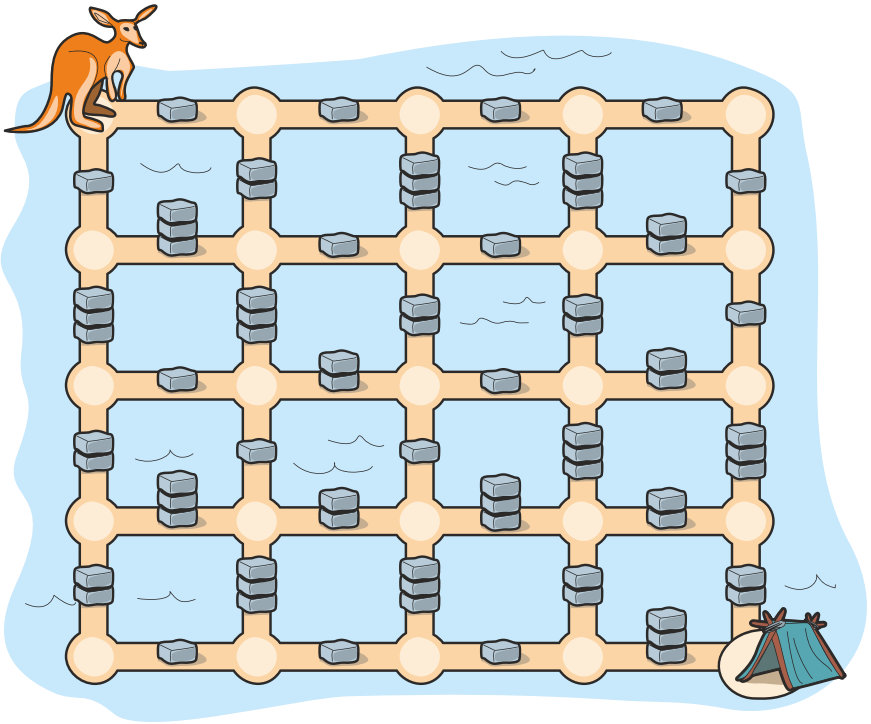
- Codice di Gray: `https://it.wikipedia.org/wiki/Codice_Gray`

14. Canguro salterino

Un canguro vuole tornare a casa . Può solo saltare seguendo il percorso e raggiungere l'incrocio successivo con un unico grande salto. Ad un incrocio può saltare a destra, a sinistra, in alto o in basso. Non può saltare sopra una pila di 3 pietre.



Il canguro vuole tornare a casa nel modo più corto.

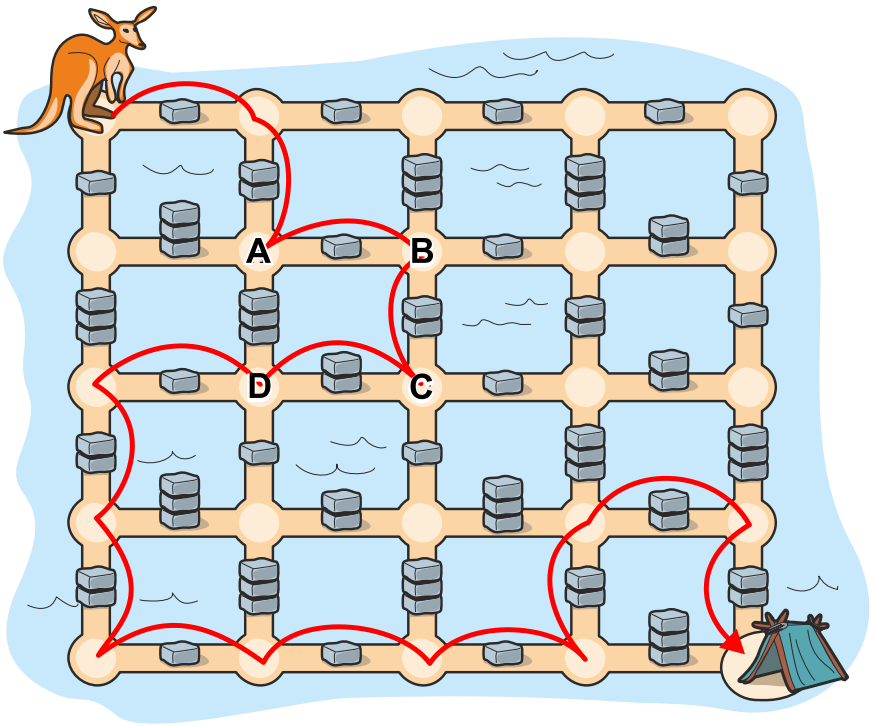


Quanti salti deve fare il canguro per tornare a casa con il percorso più corto?

A) 10 salti
B) 11 salti
C) 12 salti
D) 13 salti
E) 14 salti
F) 15 salti
G) 16 salti
H) 17 salti
I) 18 salti
J) 19 salti
K) 20 salti

Soluzione

La risposta corretta è E) 14 salti:



Il modo più semplice per iniziare la ricerca è quello di partire dalla destinazione. Si vedrà presto che c'è solo un percorso possibile di 9 salti dalla destinazione fino al punto D. Ora si deve solo trovare la via più breve dall'inizio al punto D. Con due passi il canguro raggiunge il punto A, un punto vicino al punto D. Ma non può saltare direttamente da A a D, perché c'è una pila di 3 pietre in mezzo. La deviazione più corta da A a D è via B e C, che richiede 3 salti. In totale il canguro ha bisogno di $2 + 3 + 9 = 14$ salti e tutti gli altri percorsi sono più lunghi.

Questa è l'informatica!

Per trovare un percorso, si può procedere come segue: Si cammina passo dopo passo seguendo il percorso che si vuole. Una volta raggiunto un vicolo cieco, dove tutte le direzioni sono bloccate o che portano a un punto già visitato del sentiero, si torna indietro fino a quando si trova una scelta alternativa di direzione, e poi si continua a provare.

Questo approccio è conosciuto nell' informatica come «*backtracking*» (inglese per tornare indietro). Viene utilizzato nell' informatica in vari algoritmi. Può essere utilizzato per trovare soluzioni di puzzle, sudoku o altri problemi di ottimizzazione combinatoria.

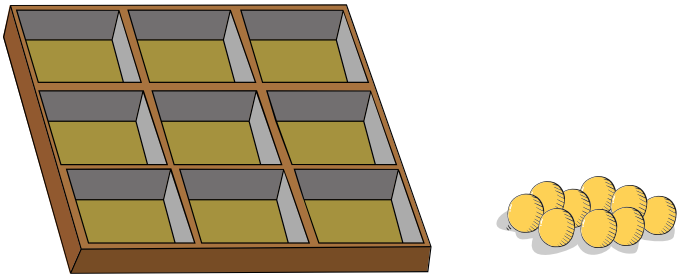
Il compito dimostra che a volte è più efficiente cercare una soluzione da dietro. Questa si chiama *ricerca all'indietro*. In questo caso, è necessario un minore backtracking perché alla fine non ci sono più opzioni. In generale non è possibile dire se una *ricerca in avanti* o all'indietro sia migliore, dipende dal problema concreto.

Parole chiave e siti web

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

15. Scomparti e biglie

Hira ha una scatola divisa in 9 scomparti e un numero illimitato di biglie:



Hira mette le biglie negli scomparti della scatola. Rispetta le seguenti regole:

- In ogni scomparto mette al massimo una biglia.
- In ogni riga e in ogni colonna il numero di biglie alla fine è pari.

Quanti schemi diversi può creare Hira con la scatola e le biglie?

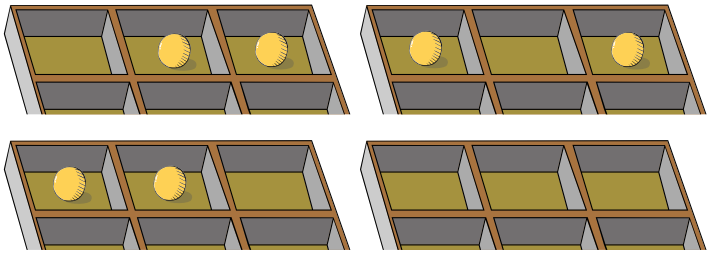
(La scatola non può essere ruotata. Lo schema con una sola biglia nell'angolo superiore sinistro è diverso da quello con una sola biglia nell'angolo superiore destro.)

A) 12
B) 16
C) 64
D) 512

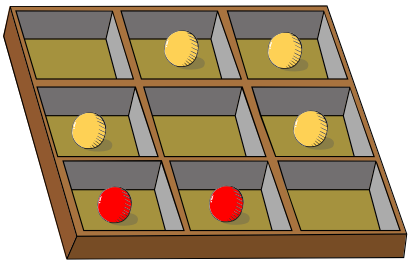
Soluzione

La risposta corretta è: B) 16.

In quanti modi Hira può riempire la prima riga? Ci deve essere un numero pari di biglie nella prima riga, cioè 0 o 2, quindi ci sono 4 modi per riempire la prima riga:



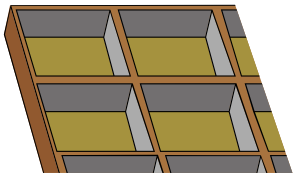
Allo stesso modo Hira ha 4 possibilità per riempire la seconda riga. Ma dopo non può scegliere nulla, perché anche le tre colonne devono contenere un numero pari di biglie. Se c'è un numero dispari di biglie nelle due righe superiori (cioè esattamente una biglia), Hira deve collocare una biglia nella terza riga di questa colonna, come avviene nelle prime due colonne dell'esempio seguente (biglie rosse):



Se le prime due righe di una colonna contengono un numero pari di biglie (cioè 0 o 2), non deve mettere una biglia nella terza riga di questa colonna, come avviene nella terza colonna dell'esempio sopra.

Poiché la scelta della prima riga è completamente indipendente dalla scelta della seconda, Hira ha 4 scelte per la prima riga e per ognuna di queste scelte ha di nuovo 4 scelte per la seconda riga. Pertanto ha un totale di $4 \cdot 4 = 16$ possibilità.

Una seconda opzione per il conteggio delle possibilità è la seguente: Si inizia guardando una parte 2×2 della scatola.

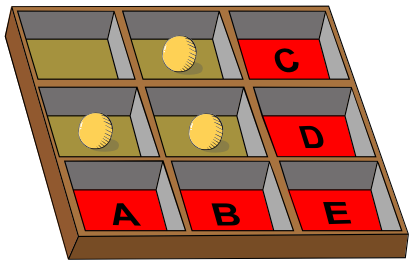


In questa parte ci sono 4 scomparti e ognuno può contenere una biglia o no. Quindi ci sono $2^4 = 16$ modi diversi per riempire questa parte.

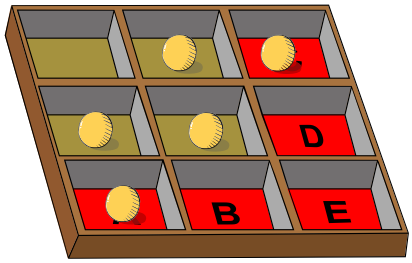
Un'osservazione importante è la seguente: Dopo che le biglie sono state collocate in questa parte della scatola, Hira non ha altra scelta per riempire gli scompartimenti rimanenti. Per ogni scomparto

sul bordo destro o nella riga inferiore, Hira deve posizionare una biglia o lasciarla fuori in modo che il numero sia pari.

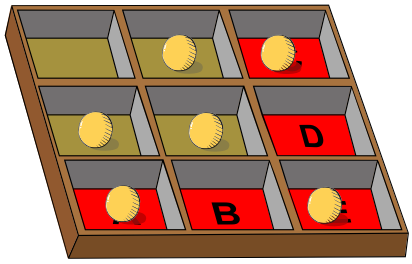
Ad esempio, Hira potrebbe riempire la parte 2×2 considerata come segue:



Poiché la prima colonna contiene una sola biglia, Hira deve mettere una biglia nello scomparto A in modo che il numero di biglie nella prima colonna sia pari. Nella seconda colonna c'è già un numero pari di biglie e quindi Hira non deve mettere una biglia nello scomparto B. Con argomenti simili si può vedere che lo scomparto D deve rimanere vuoto e Hira deve mettere una biglia in C.



Il numero di biglie in $A + B$ è quindi pari esattamente quando il numero di biglie nella parte 2×2 è pari. Esattamente lo stesso vale per la somma $C + D$. Se queste due somme sono pari, lo scomparto E può e deve rimanere vuoto; se entrambe sono dispari, Hira può e deve mettere una biglia nello scomparto E.



Questo dimostra che Hira può mettere le biglie negli scomparti della scatola in 16 modi diversi.

Questa è l'informatica!

Un importante compito dell'informatica è quello di trasferire i dati in modo sicuro. Un modo per garantire la sicurezza della trasmissione dei dati contro gli errori di trasmissione è quello di *verificare la parità*.

Alla fine del messaggio viene calcolato un bit di parità in base al messaggio da trasmettere e viene aggiunto al messaggio. Quando il messaggio viene ricevuto, il bit di parità può essere ricalcolato. Se non corrisponde al bit di parità trasmesso, è noto che c'è stato un errore di trasmissione.

In questo compito, gli scomparti dell'ultima riga e dell'ultima colonna servono come bit di parità. Se i numeri delle biglie negli scomparti sono stati trasmessi come messaggio, il destinatario può calcolare i totali delle righe e i totali delle colonne. Se questi non sono pari, il destinatario può dire a Hira che c'è stato un errore di trasmissione.

Un'altra competenza dell'informatica è la capacità di enumerare tutte le soluzioni con specifiche proprietà e quindi di determinarne il numero.

Parole chiave e siti web

- Bit di parità: `https://it.wikipedia.org/wiki/Bit_di_parità`

A. Autori dei quesiti

Tony René Andersen
Michael Barot
Wilfried Baumann
Maksim Bolonkin
Andrey Brodnik
Sarah Chan
Marios O. Choudary
Valentina Dagienė
Tolmantas Dagys
Christian Datzko
Susanne Datzko
Amirmohammad Djazbi
Nora A. Escherle
Lidia Feklistova
Fabian Frei
Gerald Futschek
Jens Gallenbacher
Christian Giang
Tom Grubb
Mathias Hiron
Juraj Hromkovič
Alisher Ikramov
Thomas Ioannou
Ungyeol Jung
Vaidotas Kinčius
Ritambhra Korpal
Regula Lacher
Vu Van Luan
Pedro Marcelino
Hamed Mohebbi
Kwangsik Moon
Xavier Muñoz
Vania Natali
Rana R. Natawigena
Ágnes Erdősné Németh
Andrei Nicolicioiu
Jean-Philippe Pellet
Wolfgang Pohl
Raymond Chandra Putra
Peter Rossmanith
Vipul Shah
Fei Shang
Wenpan Sheng
Timur Sitdikov
Maciej M. Sysło
Congyu Tian
Jiří Vaníček
Troy Vasiga
Fan Wang
Yang Xing
Binru Zhi

B. Sponsoring: concorso 2020

HASLERSTIFTUNG

http://www.haslerstiftung.ch/

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

verkehrshaus.ch

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

Kanton Zürich
Volkswirtschaftsdirektion
Amt für Wirtschaft und Arbeit

Standortförderung beim Amt für Wirtschaft und Arbeit Kanton Zürich

i-factory (Musée des transports, Lucerne)

UBS

http://www.ubs.com/

OXOCARD

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

educaTEC

https://educatec.ch/
educaTEC

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.

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

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

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

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

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

C. Ulteriori offerte

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

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

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