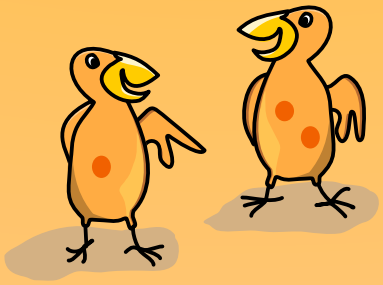
INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Quesiti e soluzioni 2021

3º e 4º anno scolastico



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

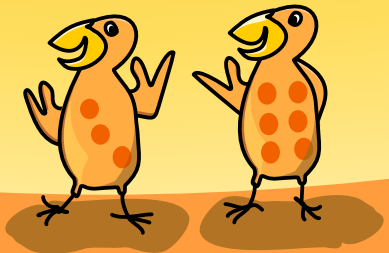
A cura di:

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

01010011010101100100100
10000010010110101010011
01010011010010010100101
00101101010100110101001
01001001010010010010001

SSII

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



Hanno collaborato al Castoro Informatico 2021

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

Capo progetto: Nora A. Escherle

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

La scelta dei quesiti è stata svolta in collaborazione con gli organizzatori dei concorsi in Germania, Austria, Ungheria, Slovacchia e Lituania. Ringraziamo specialmente:
Valentina Dagienė, Tomas Šiaulys, Vaidotas Kinčius: Bebras.org
Wolfgang Pohl, Hannes Endreß, Ulrich Kiesmüller, Kirsten Schlüter, Michael Weigend: Bundesweite Informatikwettbewerbe (BWINF), Germania
Wilfried Baumann, Liam Baumann, Anoki Eischer, Thomas Galler, Benjamin Hirsch, Martin Kandlhofer, Katharina Resch-Schobel: Österreichische Computer Gesellschaft
Gerald Futschek, Florentina Voboril: Technische Universität Wien
Zsuzsa Pluhár: ELTE Informatikai Kar, Ungheria
Michal Winzcer: Comenius University, Slovacchia

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

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

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

Questi quaderni sono dedicati alla memoria di Martin Guggisberg.

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

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

INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

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

HASLERSTIFTUNG

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

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

Premessa

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

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

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

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

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

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

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

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

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

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

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

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

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

Per ulteriori informazioni:

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

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

Indice

1. I timbri di Mika

Mika ha quattro timbri con figure diverse. Uno per uno, Mika fa due timbri su un foglio di carta e mette via il timbro usato. Risulta il seguente disegno.

Quale timbro ha usato per primo Mika?

A) 

B) 

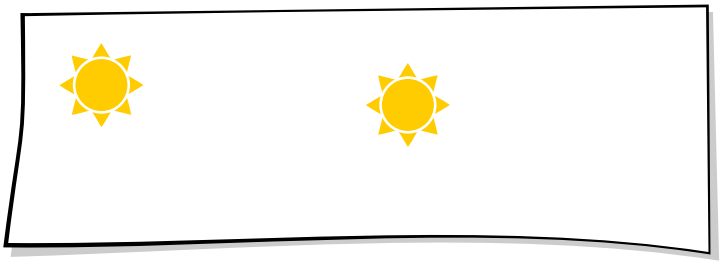
C) 

D) 

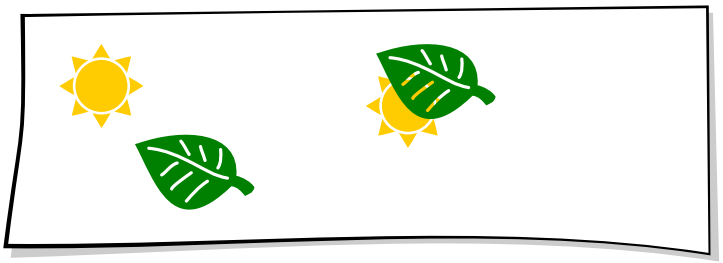
Soluzione

La risposta corretta è C: 

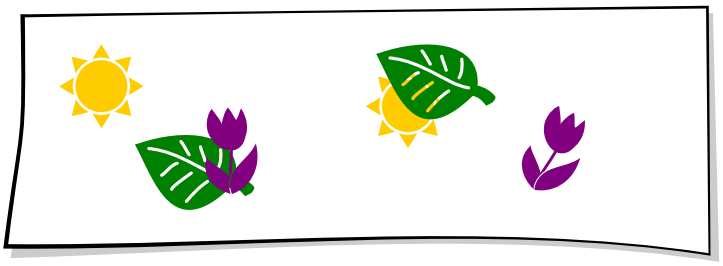
L'ordine in cui Mika usa i timbri può essere identificato da quali figure si sovrappongono ad altre figure. Tutte le figure si sovrappongono al sole. Mika ha quindi usato prima il timbro con il sole.



La foglia si sovrappone al sole, sia il fiore che la casa sono sovrapposte alla foglia. Mika ha quindi utilizzato il timbro con la foglia per secondo:



Il fiore a sua volta sovrappone sia la foglia che il sole.



Tuttavia, non è possibile che Mika abbia usato il timbro con il fiore per ultimo: in due punti la casa è chiaramente visibile sovrapposta al fiore. Così è chiaro che Mika ha usato il timbro con il fiore per terzo e quello con la casa per ultimo.

Questa è l'informatica!

Il disegno di Mika è, in un certo senso, un'immagine o una *modellazione* della realtà per mezzo di figure impresse su quattro livelli diversi: un livello con soli, un livello con foglie, un livello con fiori e un livello con case. La sovrapposizione delle figure di diversi piani permette a Mika di creare l'illusione della profondità e della tridimensionalità (tridimensionale) su una superficie (bidimensionale) di carta.

La modellazione di solito rappresenta solo gli aspetti che sono necessari per svolgere un certo compito o funzione. La realtà è così rappresentata in modo semplificato. La modellazione è un principio importante nell'informatica.

Per esempio, gli informatici creano alcuni programmi per computer con lo scopo di esaminare parti del mondo reale nel modo più veloce e preciso possibile e ottenere conoscenze su di esse. Per creare un tale programma, gli informatici devono prima pensare a un modello di pensiero che contenga gli elementi del mondo reale che sono essenziali per loro. Questo modello di pensiero permette di creare un modello reale sotto forma di schizzo o di programma informatico. Con l'aiuto di questo programma per computer, cioè un modello di una parte del mondo reale, è possibile acquisire conoscenze sul mondo reale.

Parole chiave e siti web

- Modellazione: `https://it.wikipedia.org/wiki/Teoria#Modelli`

2. La maglia giusta

Anne prepara la borsa per andare alla partita. Oggi deve portare la maglia con maniche chiare e colletto nero, ma senza strisce.

Quale maglia mette in borsa?

A) 

B) 

C) 

D) 

Soluzione

La risposta B è corretta.


e


Le maglie A e D non vanno bene oggi perché hanno le maniche nere, e il nero non è chiaro.

La maglia C ha delle strisce, quindi non va bene per la partita di oggi.

La maglia B è perfetta per oggi: ha maniche chiare e un colletto nero e nessuna striscia.

Questa è l'informatica!

In questo compito, bisogna trovare la cosa che soddisfa o non soddisfa certe *condizioni* da un insieme di cose.

Diverse *condizioni parziali*, come il colore delle maniche e il disegno del tessuto, sono state definite qui e collegate per formare una *condizione totale*. Gli informatici usano gli *operatori logici* per questi collegamenti.

Se tutte le condizioni parziali devono essere soddisfatte contemporaneamente, si usa l'operatore *E*: «Il colore delle maniche è chiaro» *E* «il colore del colletto è nero». Se è sufficiente che almeno una delle diverse condizioni parziali sia soddisfatta, si usa l'operatore *O*. Se una condizione parziale non deve essere soddisfatta, si può usare l'operatore *NON*, per esempio: NON (la maglia ha le strisce).

Per la ricerca nelli dati, condizioni molto complesse possono essere formulate con l'aiuto dei linguaggi di interrogazione.

Le condizioni stesse devono essere chiaramente definite per questo. Per esempio, la condizione che le maniche devono essere di colore chiaro può essere poco chiara. In questo caso, gli informatici scrivono un programma o una funzione che controlla se un colore è chiaro o no. Per questo, gli informatici hanno bisogno di una definizione precisa di quando esattamente un colore è «chiaro», altrimenti non è possibile scrivere un programma funzionante.

Parole chiave e siti web

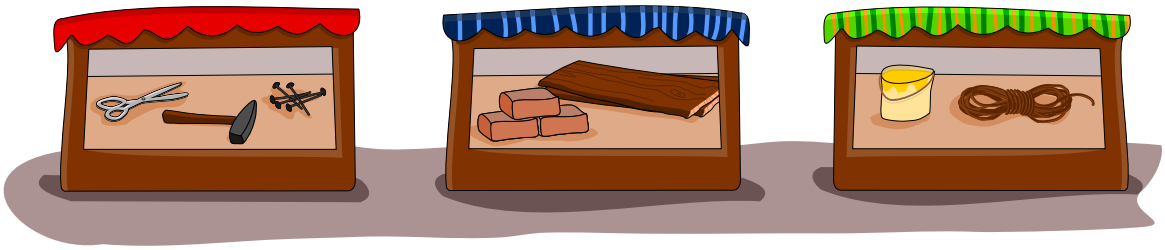
- Algebra di Boole: `https://it.wikipedia.org/wiki/Algebra_di_Boole`
- Connettivo logico: `https://it.wikipedia.org/wiki/Connettivo_logico`

3. Costruzione di ponte

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

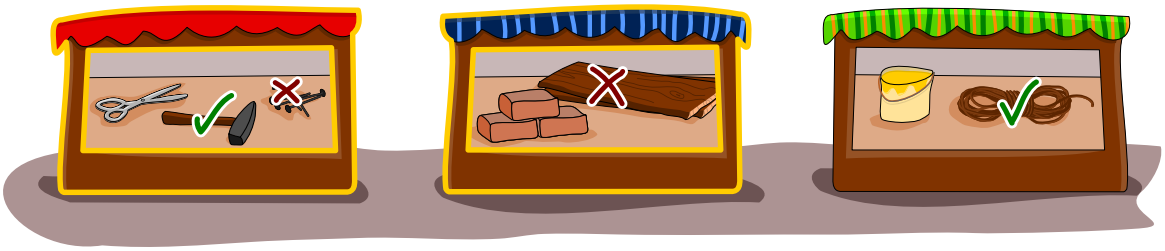
Deve però comprare gli altri materiali. Qui sotto puoi vedere tre negozi e quello che vendono.

Dove può Bella comprare le altre cose?



Soluzione

Questo è corretto:



Questa è l'informatica!

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

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

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

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

Parole chiave e siti web

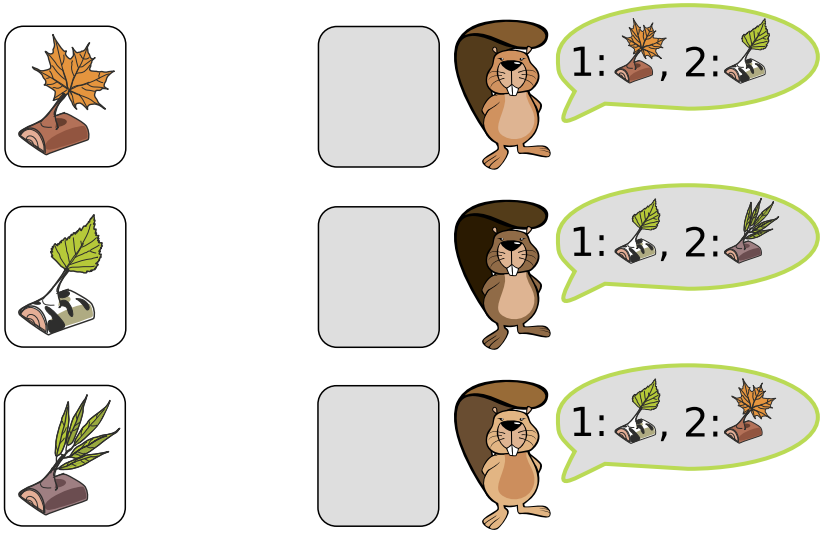
- Set: `https://it.wikipedia.org/wiki/Set_(informatica)`
- Bits: `https://it.wikipedia.org/wiki/Bit`

4. Regalo preferito

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

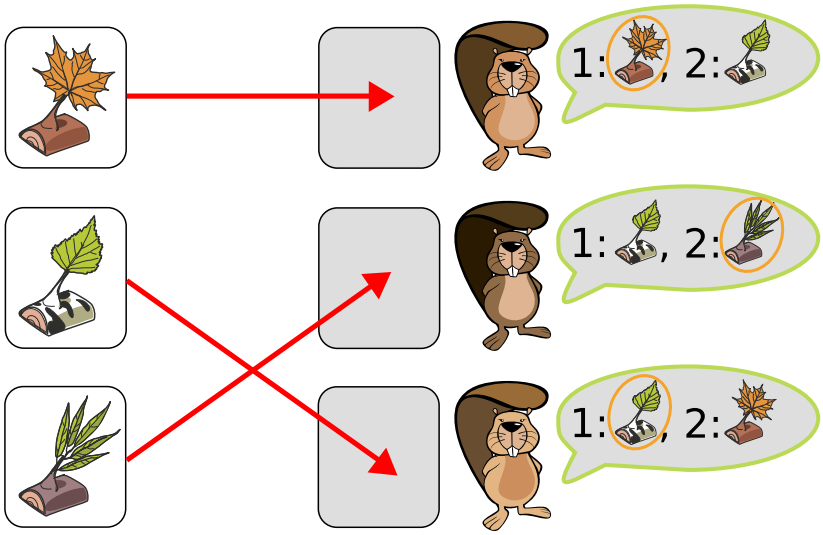
1. Il maggior numero possibile di castorini dovrebbe ricevere il loro regalo preferito.
2. Gli altri dovrebbero ricevere il secondo regalo preferito.

Dai ai castorini i regali giusti.



Soluzione

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



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

Questa è l'informatica!

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

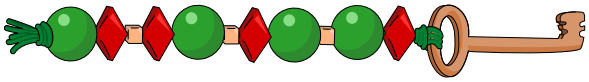
Una possibilità è quella di dare un valore alle assegnazioni: Il regalo preferito ha valore 1 e il secondo regalo preferito ha valore 2. Vogliamo minimizzare il valore totale. Un *accopiamento* (in inglese *matching*) è *ottimale* se non c'è un altro accopiamento con più prime elezioni soddisfatte. In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

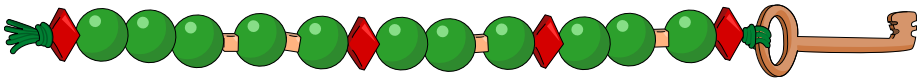
Parole chiave e siti web

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

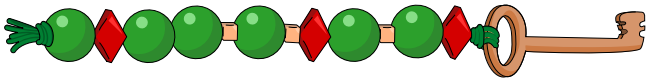
5. Portachiavi

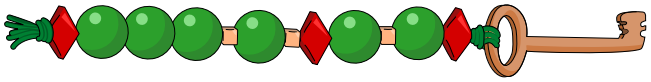
ANNA, BELLA e LENA hanno costruito dei portachiavi con i loro nomi. Hanno usato due tipi di perline per le lettere:  e  . Separano le lettere individuali con questa perlina: .

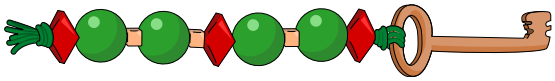
ANNA 

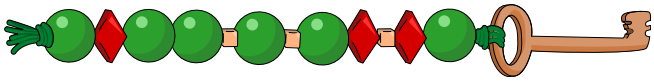
BELLA 

Quale portachiavi ha fatto LENA?

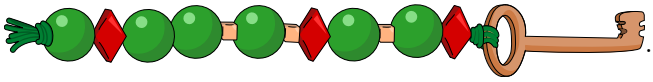
A) 

B) 

C) 

D) 

Soluzione

La risposta corretta è A)  .

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

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

Lettera	Perline
A	
N	
B	
E	
L	

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

Questa è l'informatica!

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

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

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

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

Parole chiave e siti web

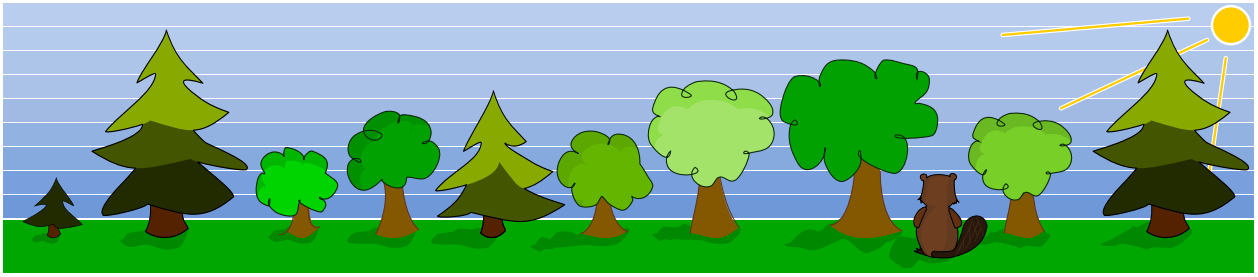
- Codifica di caretteri: `https://it.wikipedia.org/wiki/Codifica_di_caratteri`
- Codice Morse: `https://it.wikipedia.org/wiki/Codice_Morse`

6. Cade l'albero!

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

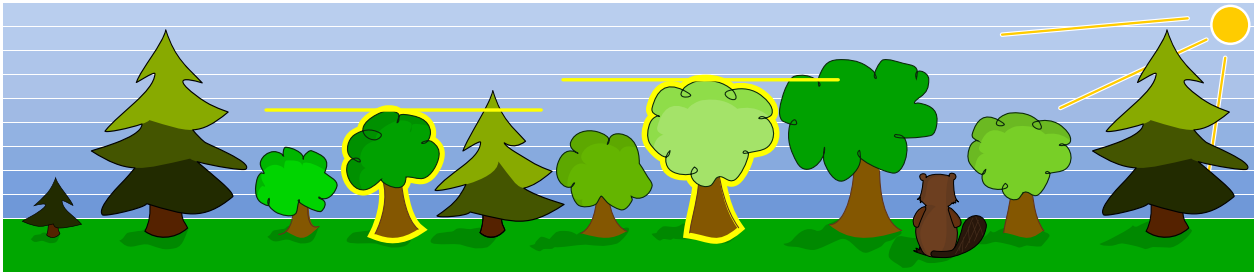
- un albero più piccolo cresce direttamente a sinistra di esso e
- un albero più grande cresce direttamente alla sua destra.

Quali alberi taglierà il castoro?



Soluzione

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



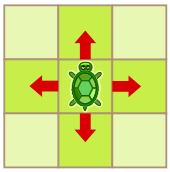
Questa è l'informatica!

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

Parole chiave e siti web

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

7. Cammino della tartaruga



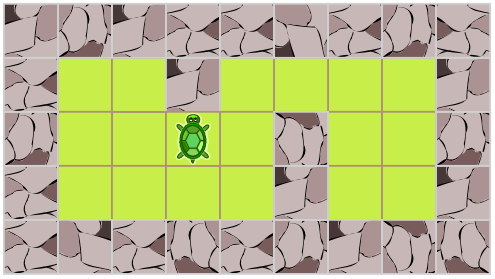
Una tartaruga vuole passeggiare in diversi giardini. Ogni giardino è suddiviso in zone (quadrati) che sono coperte di erba o di pietre. La tartaruga non può passeggiare dove ci sono le pietre. Tuttavia, può spostarsi da un quadrato d'erba a un altro quadrato d'erba proprio accanto.

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

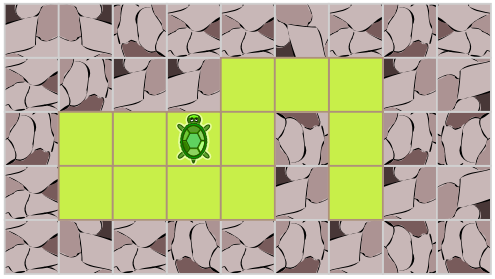
Sfortunatamente, la tartaruga non può visitare tutte le zone d'erba esattamente una volta su uno dei giardini.

Di quale giardino si tratta?

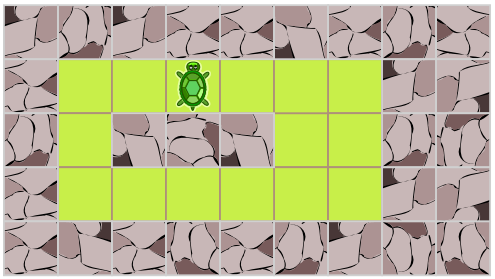
A)



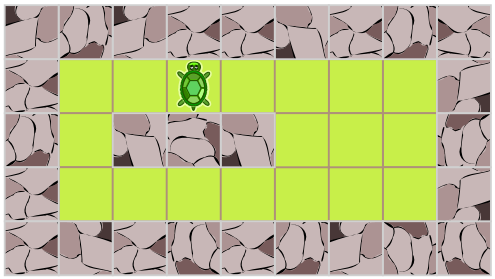
B)



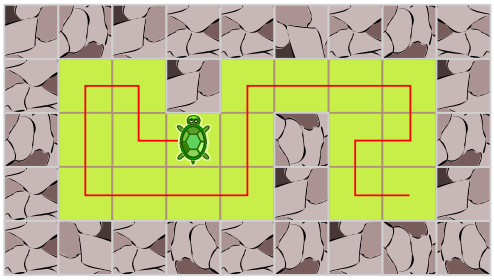
C)

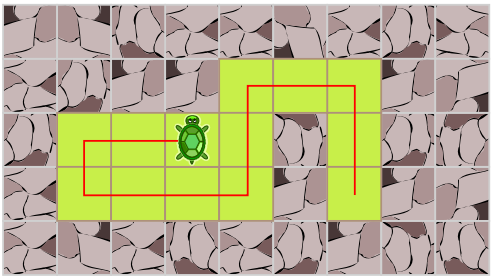


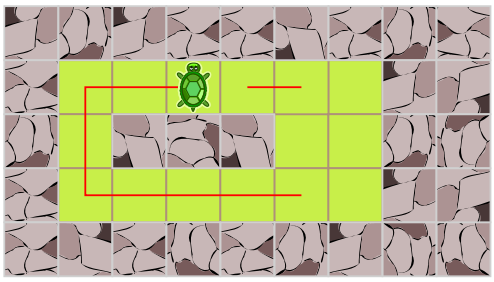
D)

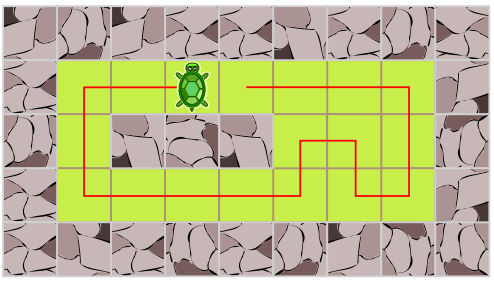


Soluzione

A) 

B) 

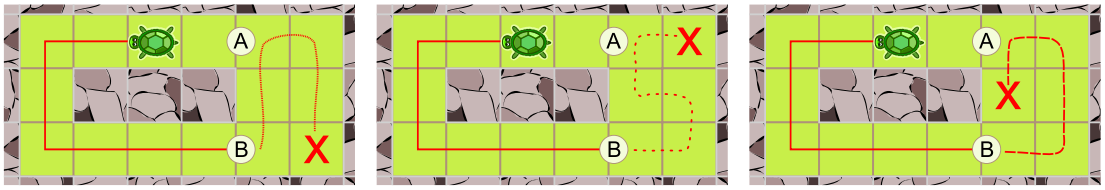
C) 

D) 

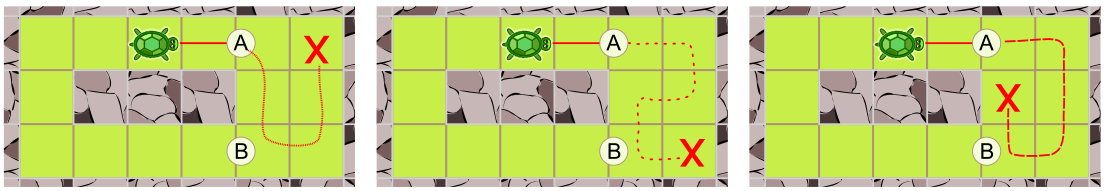
La tartaruga può visitare tutte le zone d'erba dei giardini A, B e D.

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

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



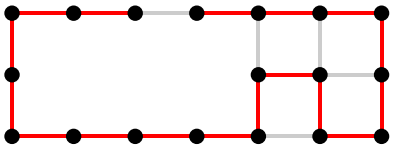
- Se la tartaruga va prima a destra, arriva ad A e dovrebbe visitare i 6 campi in modo da raggiungere il punto B alla fine. Ora si può argomentare come prima, bisogna solo scambiare la parte superiore e inferiore. Quindi non c'è nessun cammino adatto neanche in questo modo.



Questa è l'informatica!

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

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



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

Parole chiave e siti web

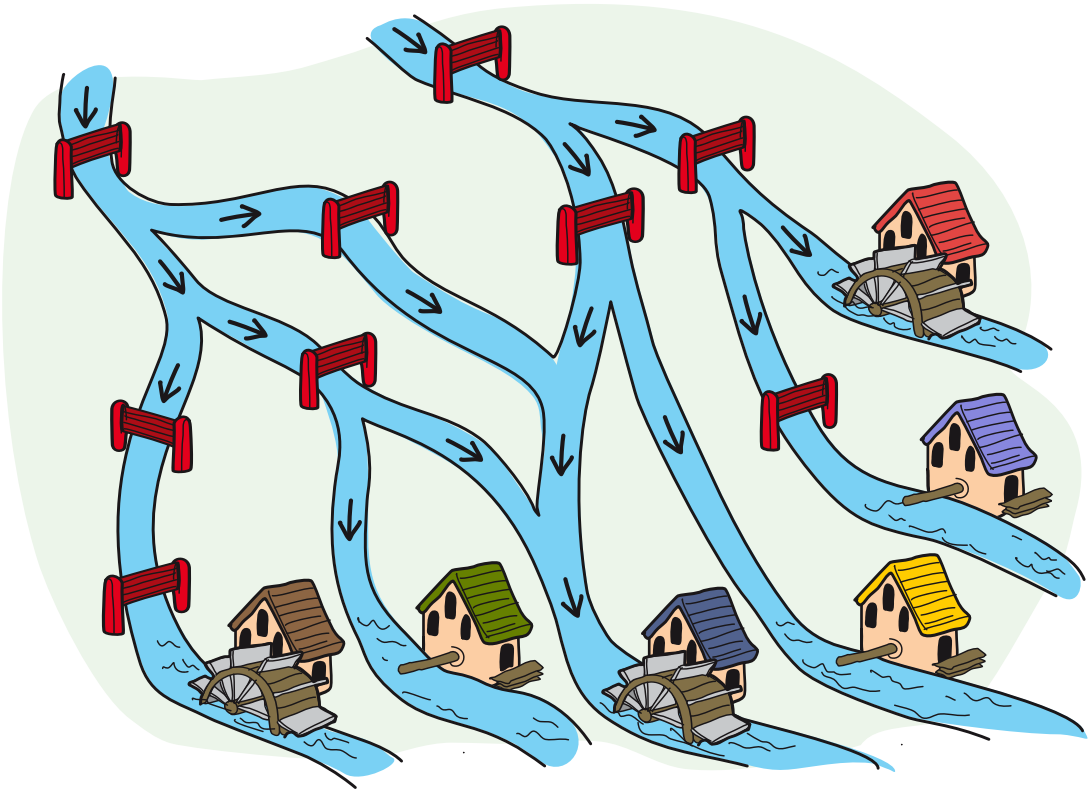
- Cammino hamitoniano: `https://it.wikipedia.org/wiki/Cammino_hamiltoniano`

8. I mulini del castoro Mert

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

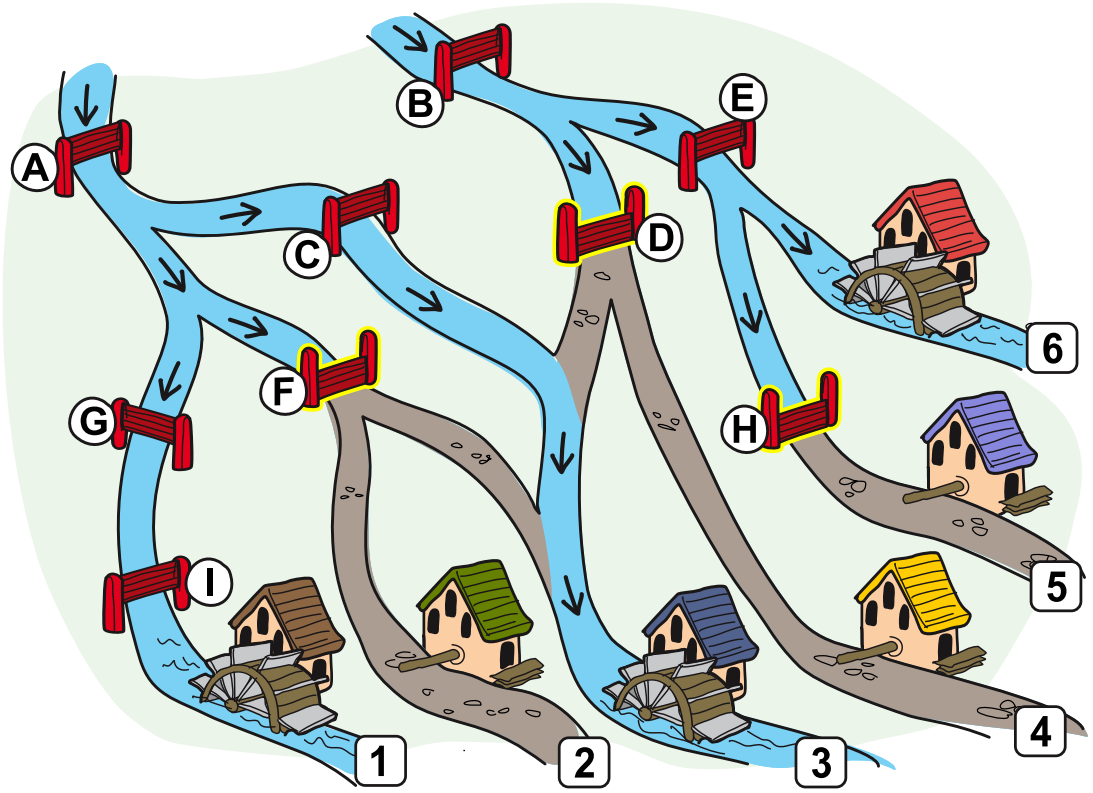
L'acqua può scorrere solo verso il basso. Una valvola a scorrimento chiusa ferma l'acqua.

Quale valvola dovrebbe chiudere Mert?



Soluzione

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



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

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

Questa è l'informatica!

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

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

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

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

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

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

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

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

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

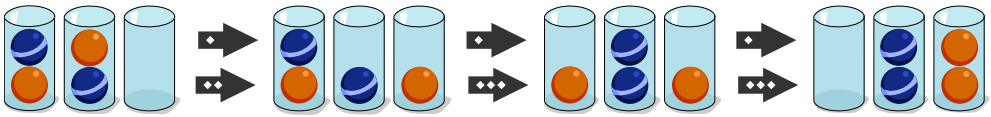
Parole chiave e siti web

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

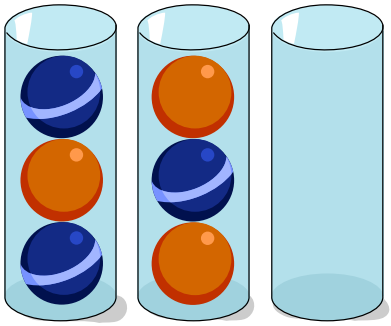
9. Gioco di palline

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

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

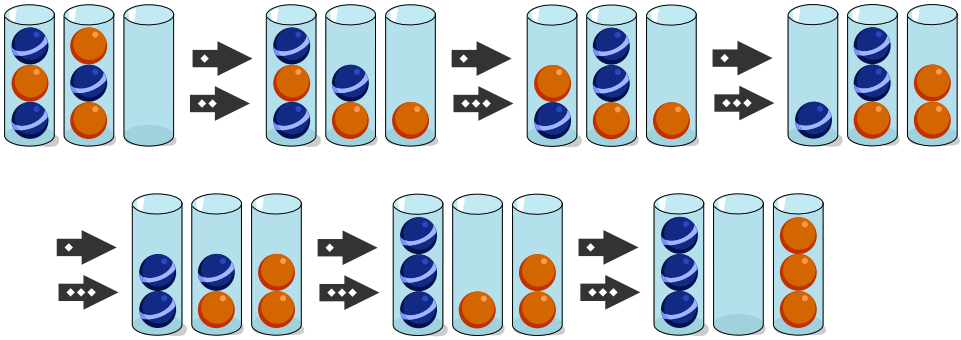


Disponi le palline spostandole secondo le tre regole.



Soluzione

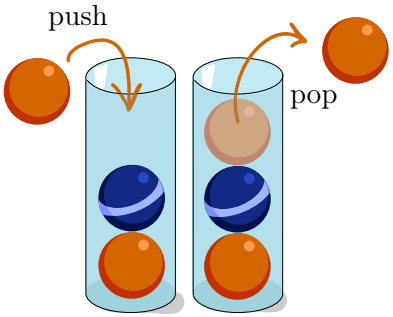
Le palle possono essere spostate nel seguente ordine:



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

Questa è l'informatica!

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



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

Parole chiave e siti web

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

A. Autori dei quesiti

Daumilas Ardickas
Michael Barot
Liam Baumann
Wilfried Baumann
Carmel Carroll
Christian Datzko
Susanne Datzko
Nora A. Escherle
Lidia Feklistova
Fabian Frei
Gerald Futschek
Christian Giang
Yasemin Gülbahar
Ezgi Arzu Güneş
Benjamin Hirsch
Andrea Hrušecká
Tiberiu Iorgulescu
YongJu Jeon
Soojin Jun
Ungyeol Jung
Filiz Kalelioğlu
Martin Kandlhofer
Dong Yoon Kim
Jihye Kim
Vaidotas Kinčius
Víctor Koleszar
Taina Lehtimäki
Tom Naughton
Graciela Oyhenard
Jean-Philippe Pellet
Zsuzsa Pluhár
Wolfgang Pohl
Rosario Schunk
Bernadette Spieler
Troy Vasiga
Florentina Voboril
Kyra Willekes
Hongjin Yeh

B. Sponsoring: concorso 2021

http://www.haslerstiftung.ch/

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

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

Standortförderung beim Amt für Wirtschaft und Arbeit Kanton Zürich

i-factory (Musée des transports, Lucerne)

http://www.ubs.com/

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

https://educatec.ch/
educaTEC

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

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

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

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

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

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

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

C. Ulteriori offerte

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

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

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