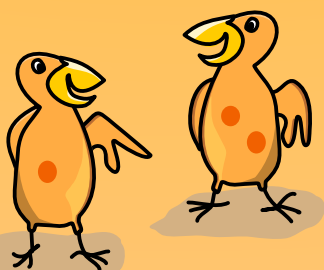




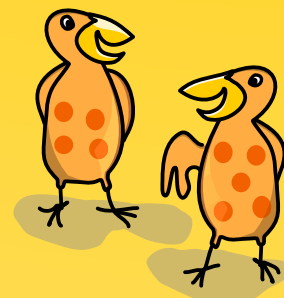
**INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA**

Quesiti e soluzioni 2021

Tutte le Categorie



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



A cura di:

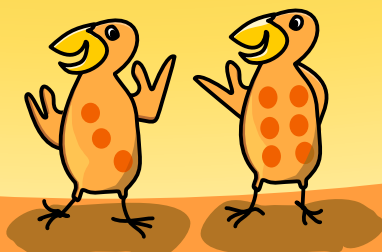
Susanne Datzko, Masiar Babazadeh, Christian Giang,
Fabian Frei, Jean-Philippe Pellet



010100110101011001001001
010000010010110101010011
010100110100100101000101
001011010101001101010011
010010010100100100100001

SS! I

www.svia-ssie-ssii.ch
schweizerischerverein für informatik in d
erausbildung // société suisse pour l'infor
matique dans 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 Rossmann: ETH Zürich, Ausbildungen- 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 Šiaulyš, 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.



I quesiti sono distribuiti con Licenza Creative Commons Attribuzione – Non commerciale – Condividi allo stesso modo 4.0 Internazionale. Gli autori sono elencati a pagina 112.



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.bebas.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» (3^o e 4^o 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:

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

Alla categoria del 3^o e 4^o anno scolastico sono stati assegnati 9 quesiti da risolvere, di cui 3 facili, 3 medi e 3 difficili. Alla categoria del 5^o e 6^o 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, 5^o e 6^o anno scolastico: 36 punti).

Il punteggio massimo totalizzabile era dunque pari a 180 punti (risp., Piccolo castoro: 108 punti, 5^o e 6^o 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

Hanno collaborato al Castoro Informatico 2021	i
Premessa	iii
Indice	v
1. I timbri di Mika	1
2. La maglia giusta	5
3. Costruzione di ponte	7
4. Regalo preferito	9
5. Portachiavi	11
6. Cade l'albero!	15
7. Cammino della tartaruga	17
8. I mulini del castoro Mert	21
9. Gioco di palline	25
10. Sacco di monete	27
11. Si incontrano?	31
12. Nidi dei Dottuccelli	35
13. Ladro di fragole	39
14. Osservazione del bosco	43
15. Puzzle di compleanno	47
16. Regalo preferito	51
17. Salvataggio dell'albero	55
18. Biblioteca	59
19. Piastrelle Truchet	61
20. Villaggi isolati	63
21. Disposizione dei liquidi	67
22. Riunione veloce	69
23. Le ragnatele di Thekla	73



24. Pila di frutta	77
25. La scimmia Coco	81
26. Maledette scrivanie	85
27. Nastro di biglia	89
28. Piano di lavoro dei castori	93
29. Numeri di biglia	95
30. Lavoro di gruppo	99
31. Contare con un cenno	103
32. Beaver Sort	107
33. Clan di Castoria	109
A. Autori dei quesiti	112
B. Sponsoring: concorso 2021	114
C. Ulteriori offerte	116



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?

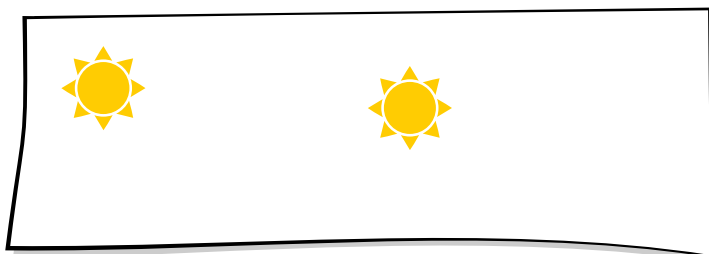
- A)  B)  C)  D) 



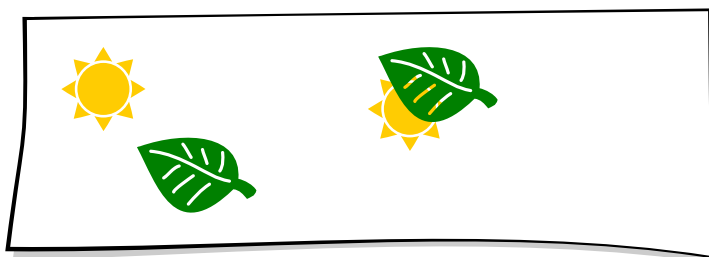
Soluzione

La risposta corretta è C: 

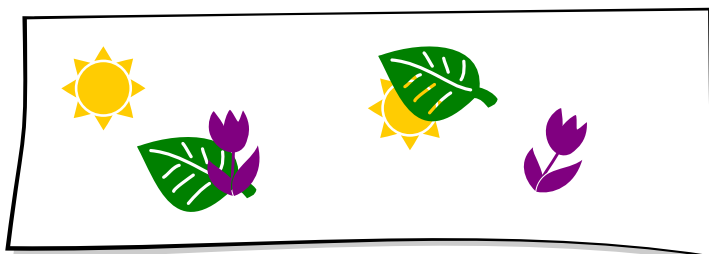
L'ordine in cui Mika usa i timbri può essere identificato da quali figure si sovrappongono ad altre figure. Tutte le figure si sovrappongono al sole. Mika ha quindi usato prima il timbro con il sole.



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?





Soluzione



La risposta B è corretta.



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 negli 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



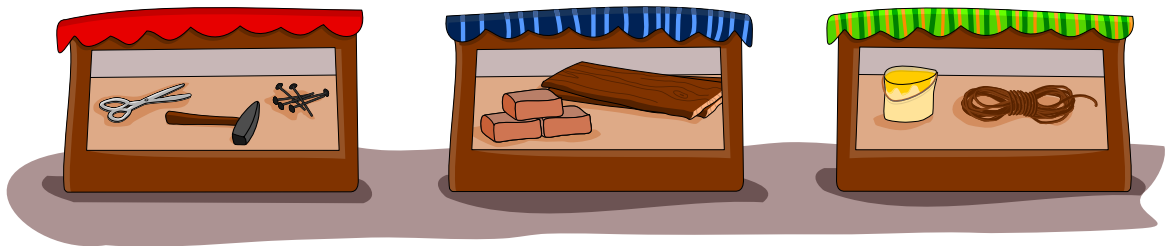
3. Costruzione di ponte

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.

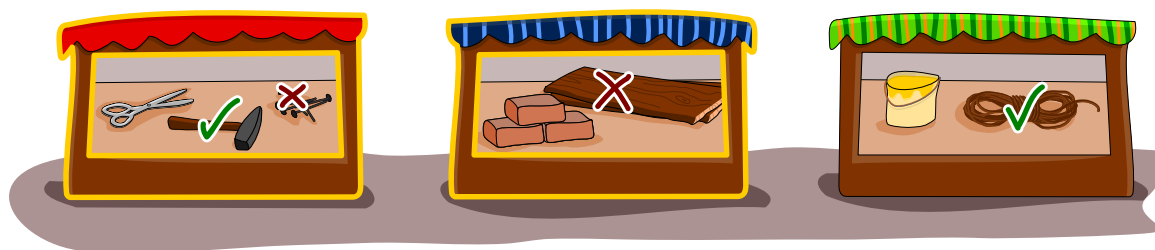
Dove può Bella comprare le altre cose?





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_\(informatica\)](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.



1: , 2: 



1: , 2: 

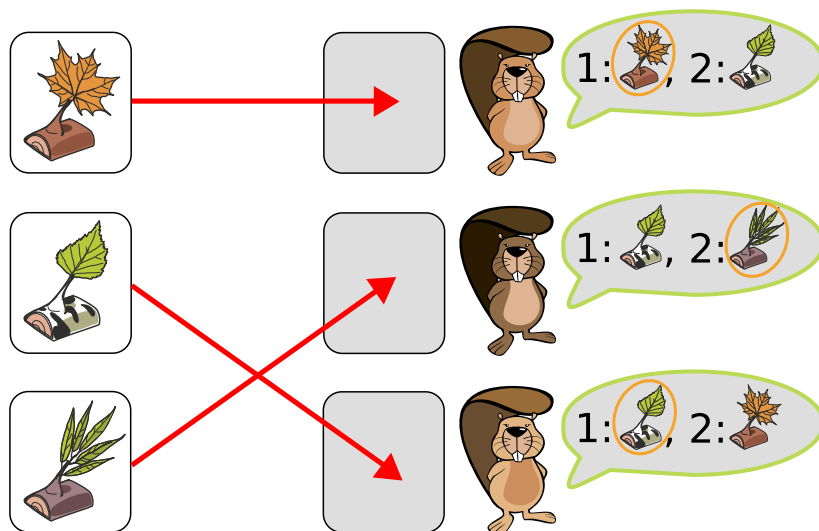


1: , 2: 



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 *accoppiamento* (in inglese *matching*) è *ottimale* se non c'è un altro accoppiamento 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.

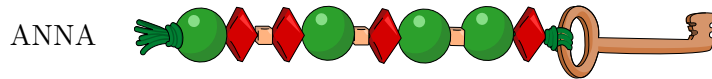
Parole chiave e siti web

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

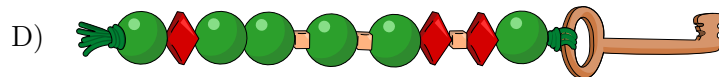
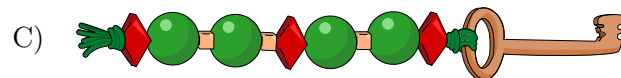
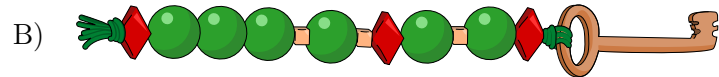
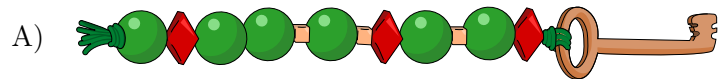


5. Portachiavi

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: ◻.



Quale portachiavi ha fatto LENA?





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 perline . 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 seguenti 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 perline . 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 perline 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 caratteri: https://it.wikipedia.org/wiki/Codifica_di_caratteri
- Codice Morse: https://it.wikipedia.org/wiki/Codice_Morse



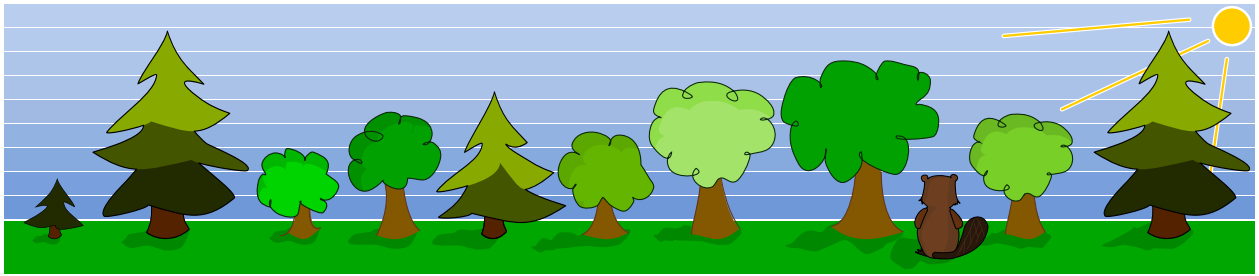


6. Cade l'albero!

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.

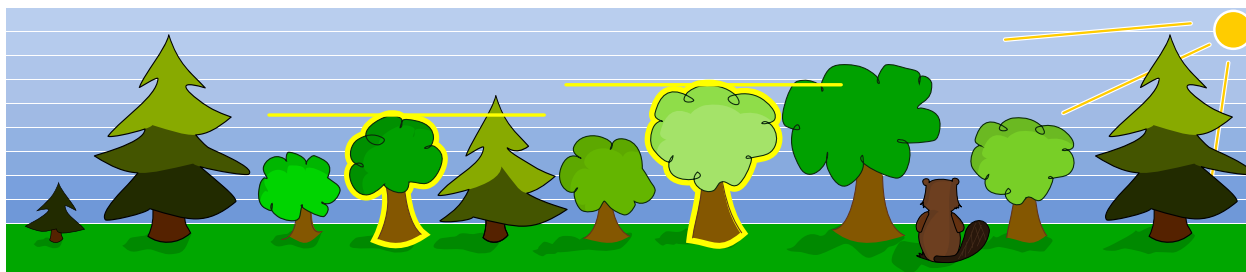
Quali alberi taglierà il castoro?





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.

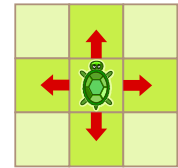
Parole chiave e siti web

- Pensiero algoritmico (algorithmic thinking)
- Operatori logici: [https://it.wikiversity.org/wiki/Operatori_Logici_\(superiori\)](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

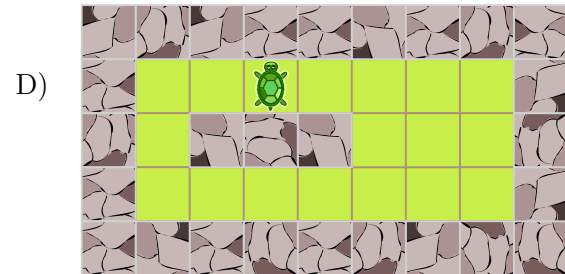
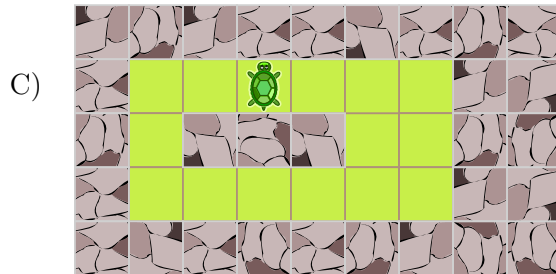
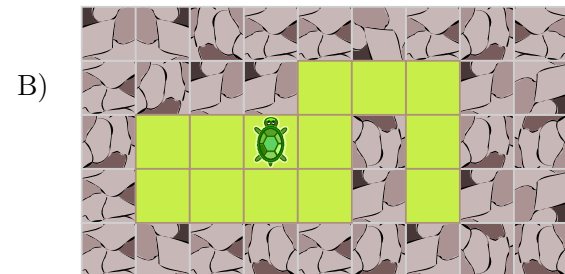
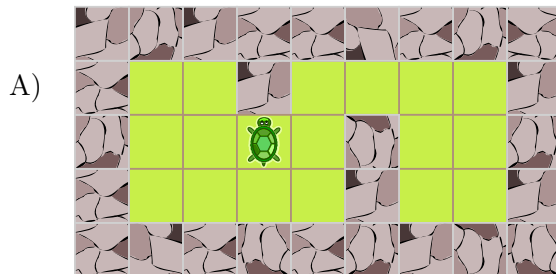
Una tartaruga vuole passeggiare in diversi giardini. Ogni giardino è suddiviso in zone (quadrati) che sono coperte di erba o di pietre. La tartaruga non può passeggiare dove ci sono le pietre. Tuttavia, può spostarsi da un quadrato d'erba a un altro quadrato d'erba proprio accanto.

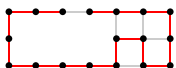


La tartaruga vuole passare per tutte le zone d'erba di ogni giardino. In ogni giardino comincia la sua passeggiata sulla zona dove si trova nel disegno. Alla fine del suo giro, la tartaruga vuole aver visitato tutte le zone del giardino esattamente una volta.

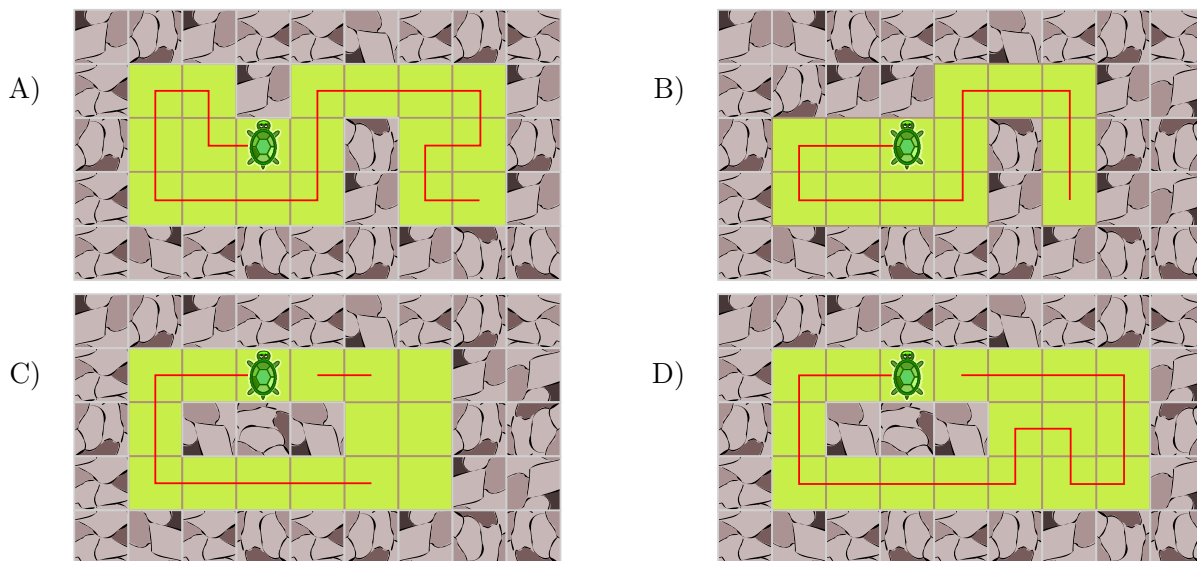
Sfortunatamente, la tartaruga non può visitare tutte le zone d'erba esattamente una volta su uno dei giardini.

Di quale giardino si tratta?





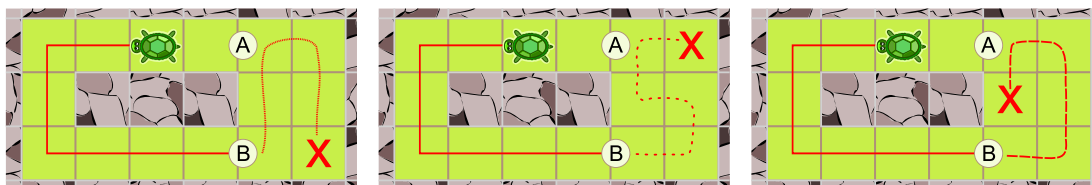
Soluzione



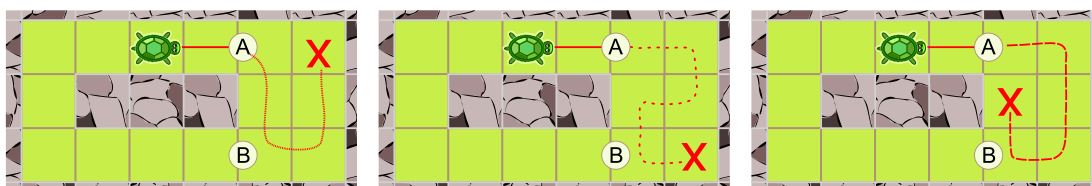
La tartaruga può visitare tutte le zone d'erba dei giardini A, B e D.

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.

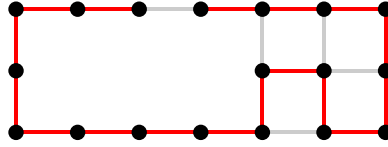


Questa è l'informatica!

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 hamiltoniano: https://it.wikipedia.org/wiki/Cammino_hamiltoniano



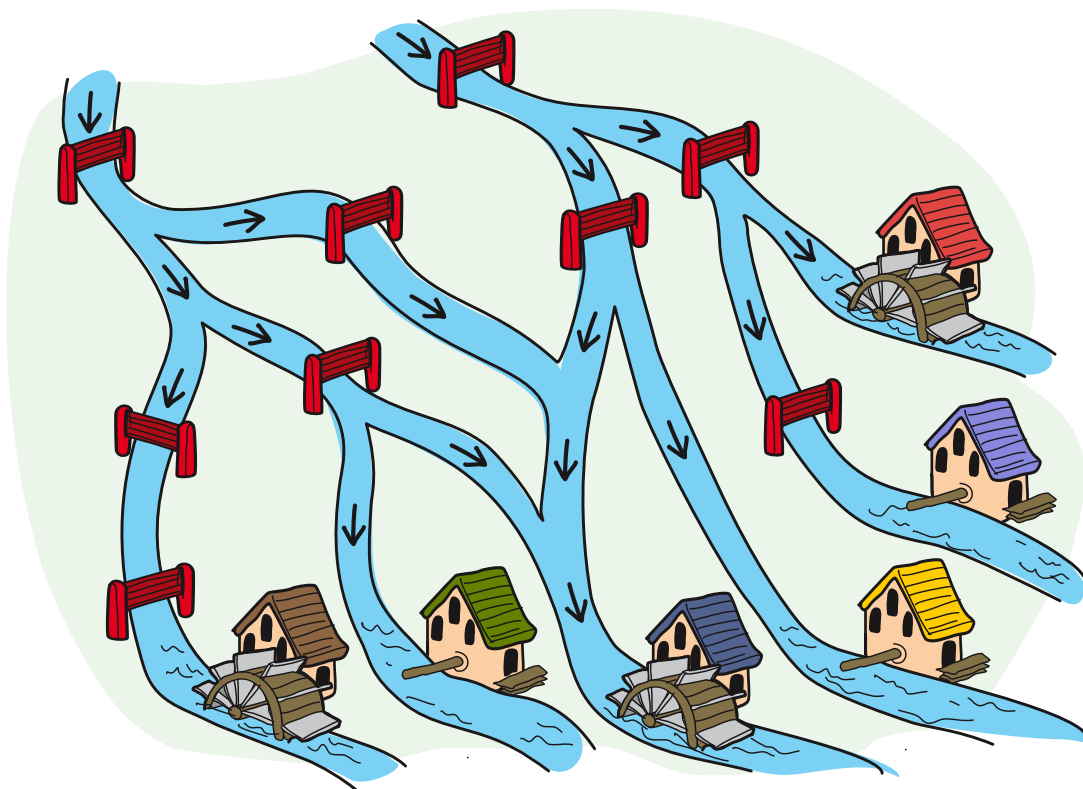


8. I mulini del castoro Mert

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.

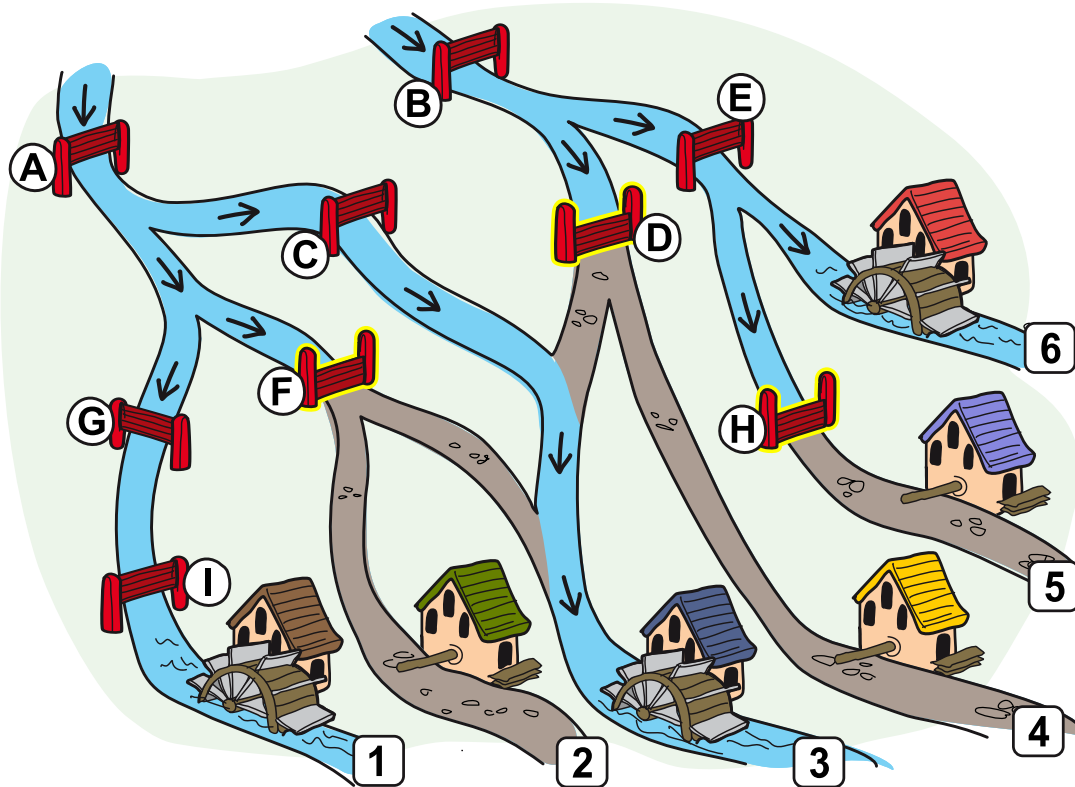
Quale valvola dovrebbe chiudere Mert?





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 valvole 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_\(informatica\)](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

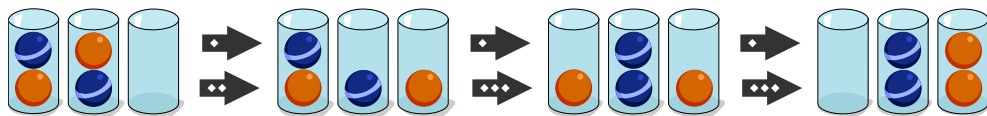




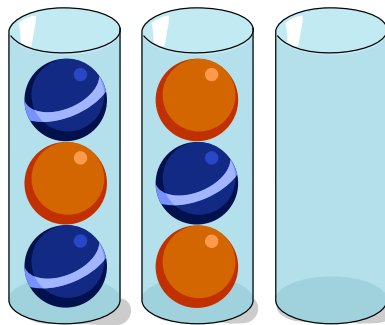
9. Gioco di palline

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.



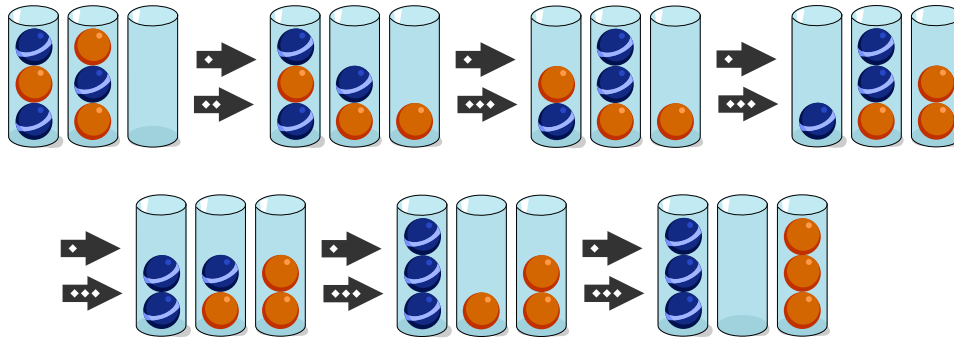
Disponi le palline spostandole secondo le tre regole.





Soluzione

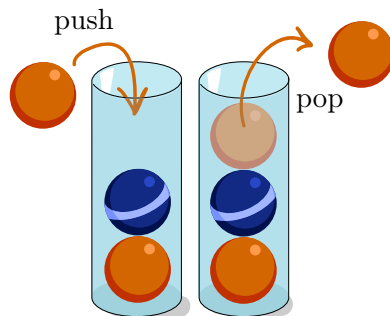
Le palline possono essere spostate nel seguente ordine:



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)

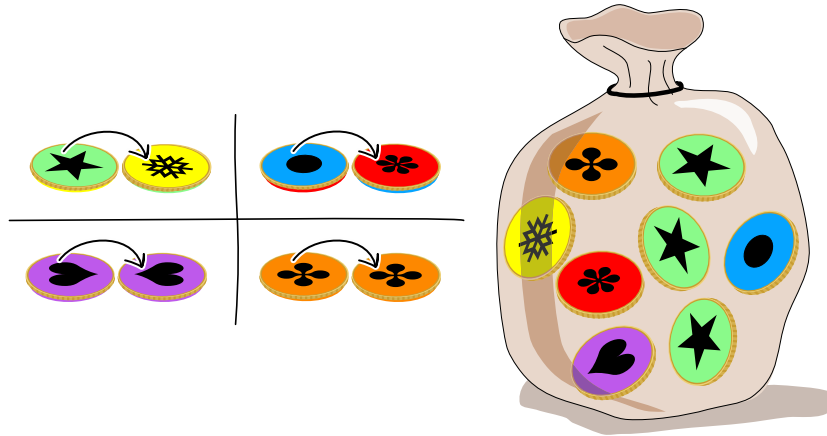
Parole chiave e siti web

- Pila: [https://it.wikipedia.org/wiki/Pila_\(informatica\)](https://it.wikipedia.org/wiki/Pila_(informatica))



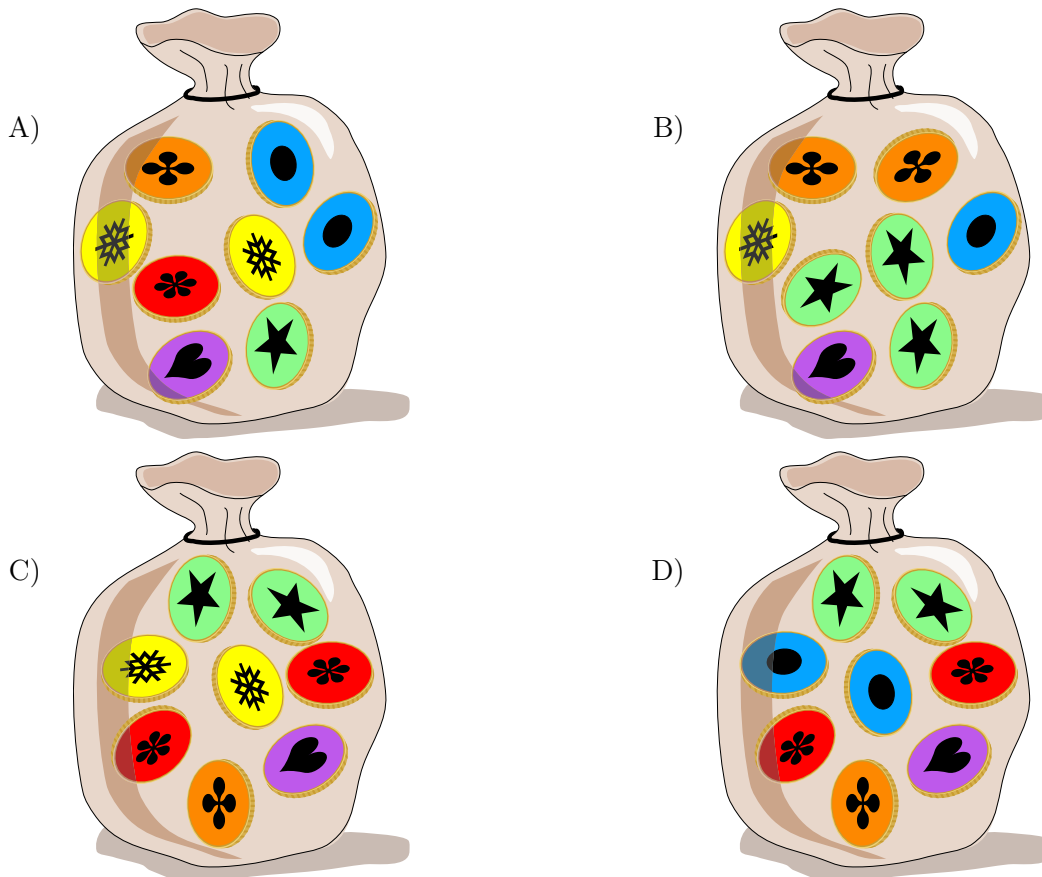
10. Sacco di monete

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



Il suo sacco viene in seguito agitato.

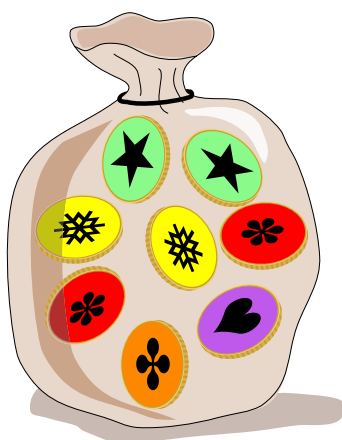
Qual è il sacco di Emil?





















Soluzione

La risposta corretta è C:



Il sacco di Emil ha:

- 4 monete  ,
- 2 monete  ,
- 1 moneta  
- e 1 moneta  .

	Sacco di Emil	Sacco A	Sacco B	Sacco C	Sacco D
 	4	3	4	4	2
 	2	3	1	2	4
 	1	1	2	1	1
 	1	1	1	1	1

Solo il sacco C ha lo stesso numero di monete per ogni tipo di moneta del sacco di Emil. Pertanto, è la soluzione.

Questa è l'informatica!

In questo compito devi riconoscere i tipi di monete senza vedere entrambi i lati. Hai dunque solo informazioni incomplete. Gli oggetti del mondo reale sono immagazzinati in un sistema informatico con le loro caratteristiche essenziali. Spesso è sufficiente conoscere solo una parte di queste caratteristiche per poter riconoscere un oggetto. Una telecamera in un veicolo autonomo (per esempio un'automobile)



vede sempre solo parti della realtà e il sistema informatico deve comunque essere in grado di riconoscere i veicoli e le persone sulla strada e reagire correttamente alla rispettiva situazione del traffico. Come gli esseri umani, l'intelligenza artificiale dei sistemi informatici impara gradualmente a riconoscere correttamente gli oggetti a partire dai frammenti.

Parole chiave e siti web

- Multiinsieme: <https://it.wikipedia.org/wiki/Multiinsieme>

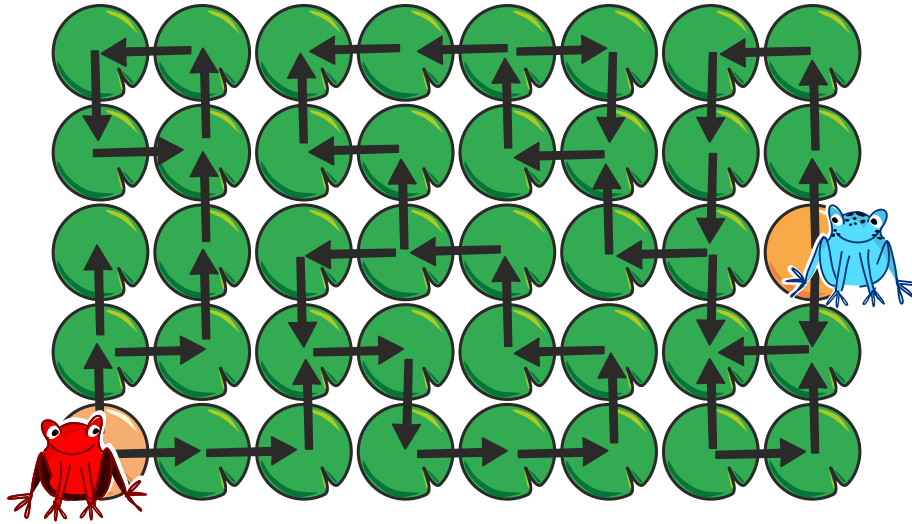




11. Si incontrano?

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

Su quale foglia di ninfea possono incontrarsi?

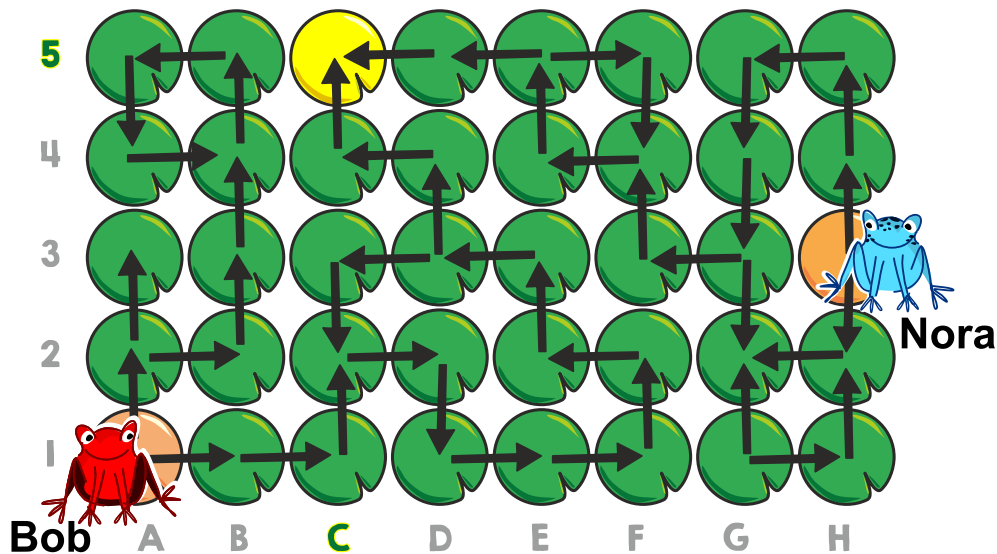


Man kann auf die Blätter klicken. Klickt man auf ein Blatt, wird dieses ausgewählt und gleichzeitig ein bereits ausgewähltes Blatt wieder deaktiviert.



Soluzione

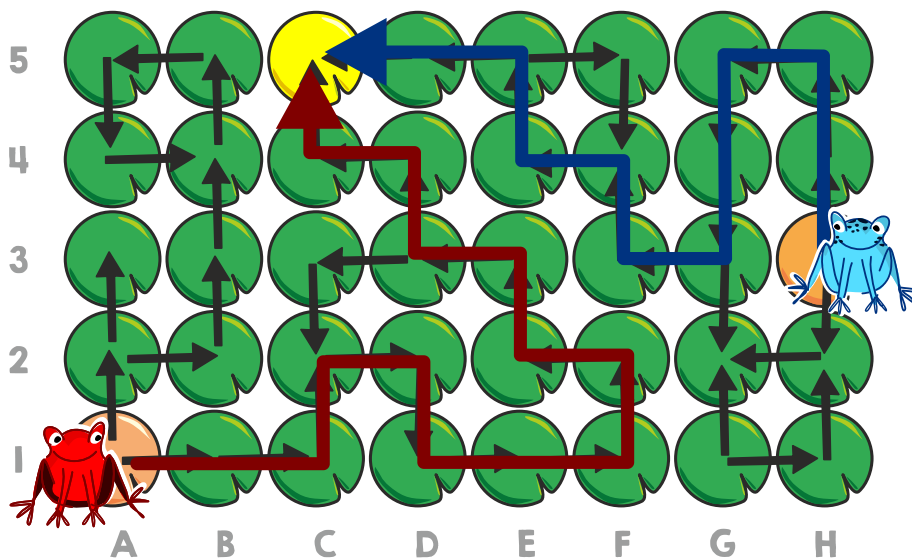
Le rane possono incontrarsi solo sulla foglia C5.



Nella sua posizione di partenza, la rana rossa Bob ha due opzioni: se va «su», o entra nel vicolo cieco A3 o rimane bloccato nel cerchio che inizia in B4. Se inizialmente va «a destra» (a B1), può prima saltare su D3. Lì può saltare «a sinistra» in un cerchio, che lo riporta a D3, o «su», che lo porta ulteriormente a C5 - un altro vicolo cieco.

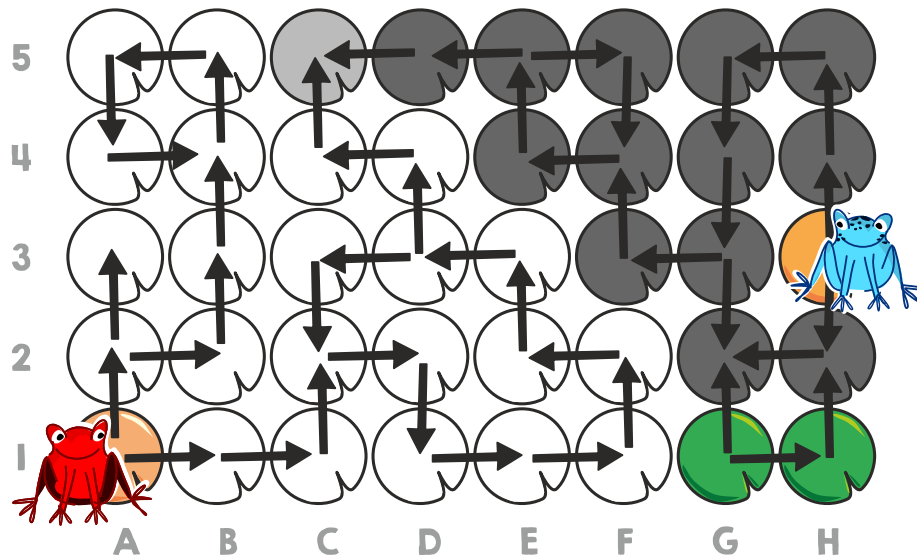
Anche la rana blu Nora ha due scelte all'inizio. Se va «giù», entra nel vicolo cieco G2. Se parte «in alto», raggiunge prima il G3. Da lì può entrare di nuovo nel vicolo cieco G2 o andare a «sinistra» e raggiungere finalmente la foglia E5. Da lì va di nuovo in un cerchio che la porta alla foglia E5, o al punto morto in C5.

Sappiamo già che anche Bob può raggiungere C5, quindi possono incontrarsi lì. Il disegno mostra i modi in cui entrambi possono arrivare a C5.





Tuttavia, questo non garantisce che non si incontreranno da qualche altra parte. Il prossimo disegno mostra tutte le foglie che Bob (bianco) e Nora (grigio scuro) possono raggiungere se seguono le frecce in ogni modo possibile. Vediamo che solo C5 può essere raggiunto da entrambi.



Questa è l'informatica!

Come si può creare l'ultima immagine? Le foglie che possono essere raggiunte da una rana possono essere trovate con una *ricerca in ampiezza* o una *ricerca in profondità*. Queste sono due delle procedure standard più importanti nell'informatica. Con il loro aiuto, si può determinare le foglie grigie e bianche. Infine, bisogna trovare solo le foglie che possono essere raggiunte da entrambe le rane.

Parole chiave e siti web

- Ricerca in ampiezza: https://it.wikipedia.org/wiki/Ricerca_in_ampiezza
- Ricerca in profondità: https://it.wikipedia.org/wiki/Ricerca_in_profondità





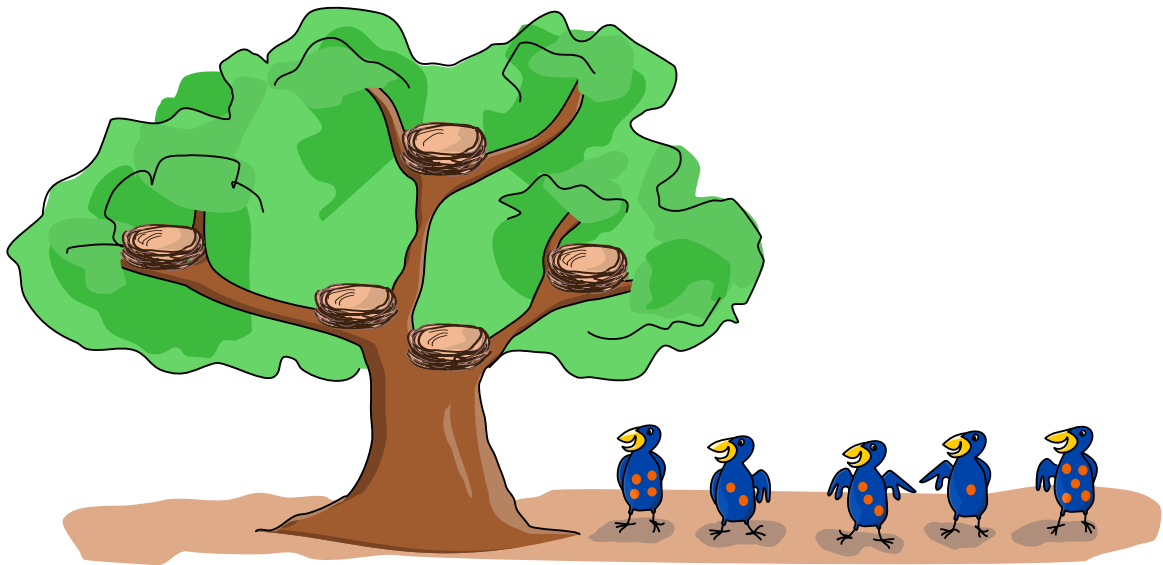
12. Nidi dei Dottuccelli

I Dottuccelli sono uccelli a pois. Ci sono cinque Dottuccelli accanto a un albero. Uno per uno - in ordine da sinistra a destra - salgono sull'albero e si appollaiano nei nidi vuoti. Quello con i quattro punti è il primo. Ogni Dottuccello procede così:

Iniziando dalla parte bassa dell'albero, esegue i seguenti passi finché trova un nido vuoto:

1. Sale fino a trovare un nido.
2. Se il nido è vuoto, si appollaia in quel nido e ci rimane.
3. Altrimenti continua a salire a seconda del numero di pois del Dottuccello già appollaiato nel nido:
 - a sinistra, se quest'ultimo ha più pois del Dottuccello che sta salendo;
 - a destra, se quest'ultimo ha meno o lo stesso numero di pois del Dottuccello che sta salendo.

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





Soluzione

È così che si arriva alla soluzione giusta:

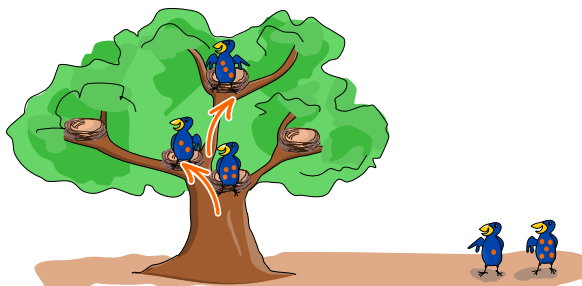
Il primo Dottuccello, quello con 4 pois, si appollaia nel nido più basso e rimane lì.



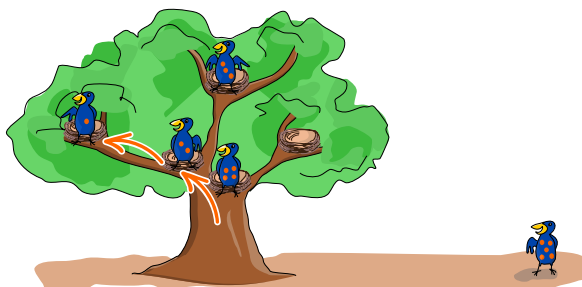
Il secondo Dottuccello ha 2 pois. Il primo Dottuccello con 4 pois si siede nel nido più basso. Poiché 4 è maggiore di 2, il secondo Dottuccello sale più a sinistra e si appollaia nel primo nido libero.



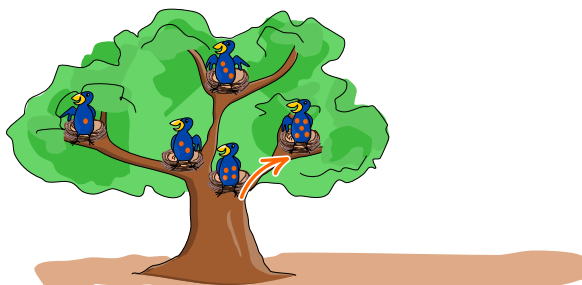
Il terzo Dottuccello ha 3 pois. Sale a sinistra al nido più basso, dove è seduto il Dottuccello con 4 pois, siccome 4 è maggiore di 3. Nel prossimo nido si trova il Dottuccello con 2 pois. Poiché 3 è maggiore di 2, il terzo Dottuccello sale a destra. Poi si appollaia nel prossimo nido libero. Questo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



Il quarto Dottuccello ha 1 pois. Poiché tutti gli altri Dottuccelli hanno più pois, sale a sinistra ad ogni nido occupato. Poi arriva al nido più a sinistra e rimane lì.



L'ultimo Dottuccello ha 5 pois. Poiché nessun Dottuccello ha più pois, sale a destra ad ogni nido occupato. Lo fa una volta al nido più basso e quindi si appollaia nel nido vuoto all'estrema destra.





Questa è l'informatica!

Con i Dottuccelli appollaiati nei nidi secondo questa procedura, si ha un vantaggio interessante: un certo Dottuccello può essere trovato rapidamente. Se il Dottuccello che cerchi ha meno punti di quello che stai guardando, devi continuare a cercare nella parte sinistra dell'albero. Altrimenti, continua a guardare a destra. Così, ogni volta che controlli un Dottuccello, puoi restringere l'area di ricerca a una delle due metà. Pertanto, troverai rapidamente il Dottuccello che cerchi.

Ci sono molti modi in cui i dati possono essere organizzati; queste sono chiamate diverse *strutture dati*. La struttura dati in questo compito è un *albero binario di ricerca*. La parola «binario» deriva dalla parola latina «bis» per «due volte». Questo perché alla fine di un ramo (dove si trova un nido nel compito), al massimo due rami più piccoli portano avanti. Gli alberi binari di ricerca sono utilizzati nei programmi per computer quando si devono trovare rapidamente molti dati. Di solito sono molto più grandi del piccolo albero nel compito. C'è anche un'altra differenza: l'albero nel compito ha un numero fisso di cinque punti. Con un albero binario di ricerca, d'altra parte, è possibile inserire sempre più dati. Per inserire dei dati, un nuovo ramo viene semplicemente aggiunto alla fine di un ramo, allargando così l'albero. Le strutture dati che possono cambiare in questo modo sono chiamate *strutture dati dinamiche*.

Parole chiave e siti web

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



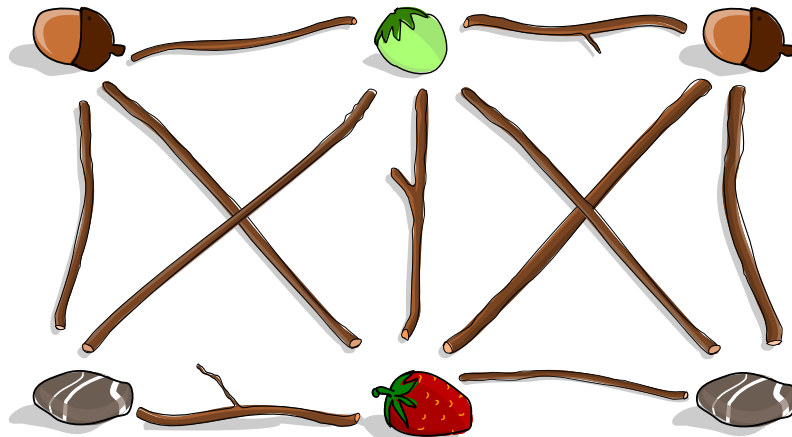


13. Ladro di fragole

Anja vuole creare un'opera d'arte in giardino, per farlo ha raccolto diverse cose: ghiande, nocchie, pietre e una fragola. Mette alcune cose sul prato.

Poi Anja mette dei rami tra queste cose. Segue la seguente regola: un ramo non può trovarsi tra due cose identiche - per esempio, non tra due ghiande.

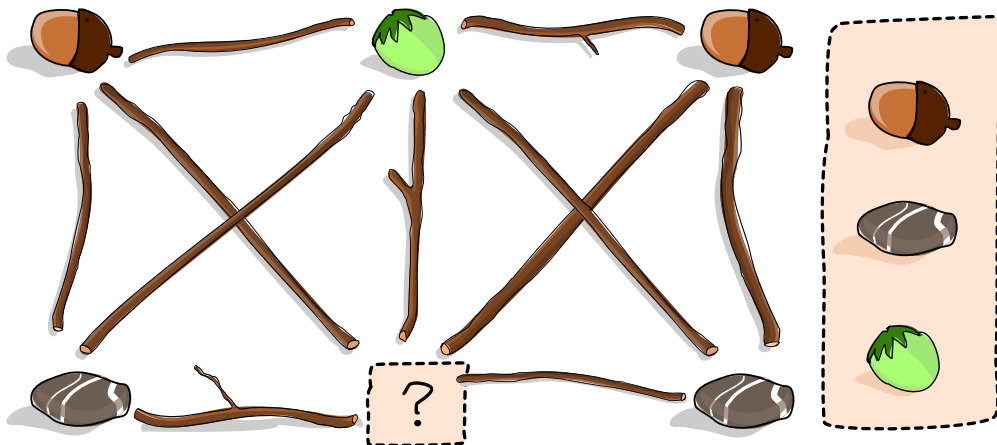
Ecco l'opera d'arte finita:



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

Puoi aiutarlo a coprire il misfatto?

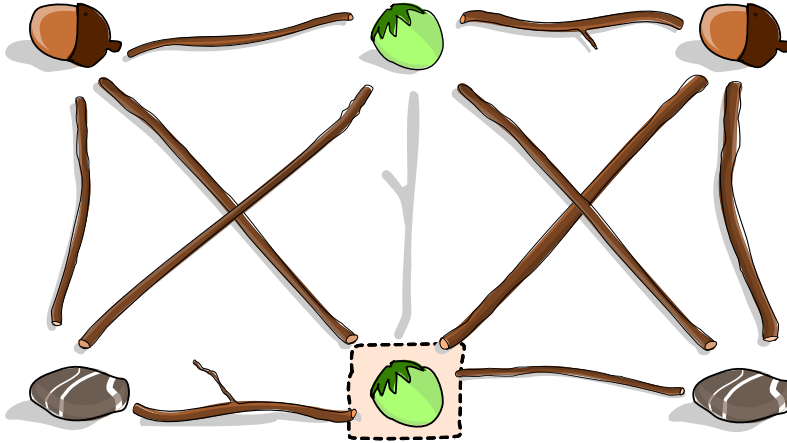
Metti un'altra cosa al posto della fragola e rimuovi esattamente un ramo. Alla fine, la regola di Anja dovrebbe applicarsi anche all'opera d'arte modificata.





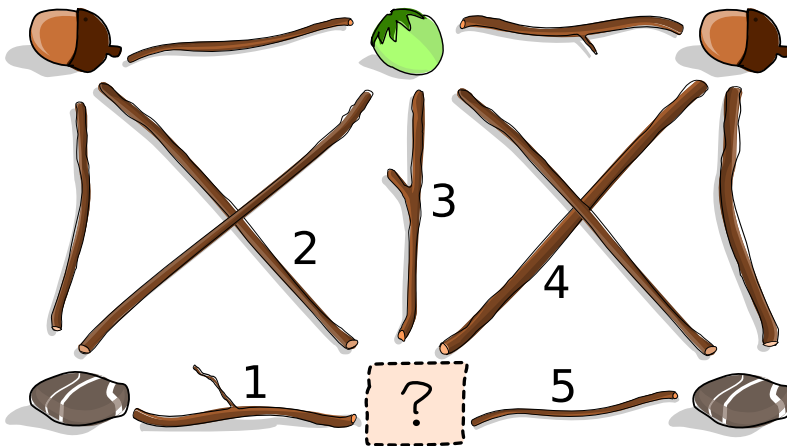
Soluzione

Se si sostituisce la fragola con una nocciola, il ramo 3 al centro viola la regola di Anja: si trova tra due cose identiche, cioè due nocciole. Pertanto, questo ramo deve essere rimosso.



Per le altre due possibili sostituzioni, è necessario rimuovere più di un ramo:

- Se la fragola è sostituita da una ghianda, è necessario rimuovere i rami 2 e 4.
- Se la fragola è sostituita da una pietra, è necessario rimuovere i rami 1 e 5.



Questa è l'informatica!

L'opera d'arte di Anja può essere rappresentata come un *grafo*. Un grafo è composto da *vertici* (i luoghi delle cose) e da *archi* (i rami), ognuno dei quali collega due vertici. I grafi sono molto versatili e sono utilizzati per la modellazione in molti compiti di informatica. Quando due vertici sono collegati direttamente da un arco, sono *vicini* l'uno all'altro. Un gruppo di vertici in cui ogni vertice è un vicino a ogni altro vertice è chiamato *cricca*. Nel nostro grafo, abbiamo due cricche con quattro vertici: la metà destra e la metà sinistra del grafo (la nocciola sopra e il punto interrogativo appartengono a entrambe le cricche). Segue dalla regola di Anja che tutti i nodi di una cricca debbano essere occupati da cose diverse. Per mantenere la regola, abbiamo bisogno di almeno tante cose diverse quanti sono i vertici di una cricca. Dopo aver rimosso la fragola, abbiamo solo 3 cose diverse.



Così ora possono rimanere cricche con un massimo di 3 vertici per continuare a soddisfare la regola. Quindi un arco (un ramo) deve essere rimosso in modo che entrambe le cricche con quattro vertici siano rotte.

La regola di Anja corrisponde a una regola del cosiddetto *problema della colorazione dei grafi*: Assegniamo un colore ad ogni vertice di un grafo, dove i vicini devono avere colori diversi. (I colori corrispondono a i diversi tipi di cose.) L'obiettivo è di solito quello di usare il minor numero possibile di colori. Il problema di come colorare un grafo con il numero minimo di colori ha molte applicazioni. Alcuni esempi sono la pianificazione di competizioni sportive, la progettazione di un piano di posti a sedere e persino la risoluzione di un Sudoku.

Parole chiave e siti web

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

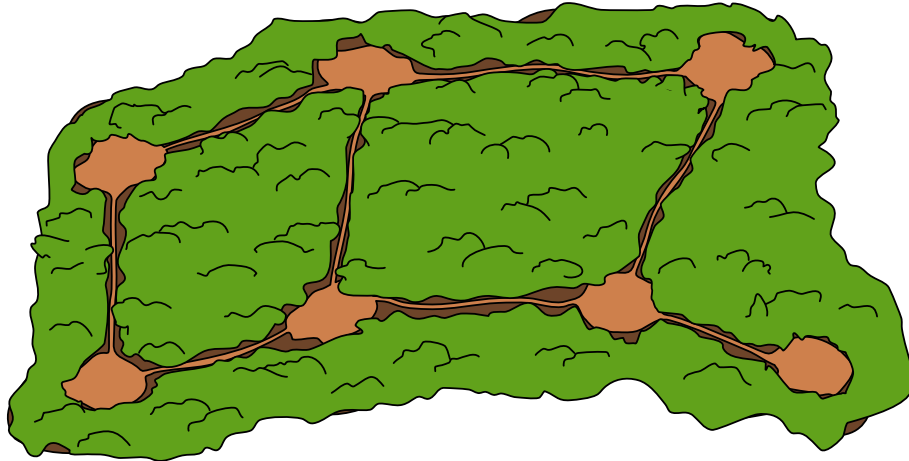




14. Osservazione del bosco

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

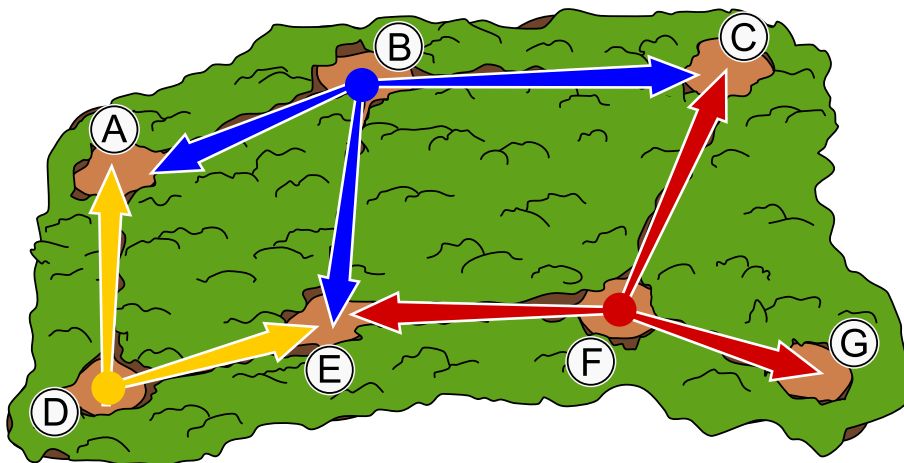
Scegli il minor numero possibile di radure da cui i forestali possano osservare tutti i sentieri!





Soluzione

L'immagine mostra la soluzione minima in cui i forestali possono stare solo su tre radure e osservare tutti i sentieri.



Ci sono otto sentieri che devono essere osservati. Se solo due forestali potessero osservare tutti i sentieri, dovrebbe esserci una radura da cui partono almeno quattro sentieri. Ma non c'è una tale radura in questo bosco. Pertanto, due forestali non sono sufficienti.

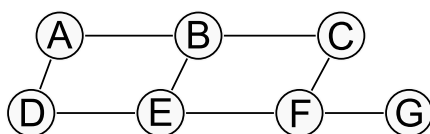
Quindi, sono necessari almeno tre forestali per osservare tutti i sentieri. Di conseguenza, la soluzione data qui è una soluzione con il minor numero possibile di forestali. In effetti, non esiste un'altra soluzione con esattamente tre forestali.

Dal numero di sentieri da osservare e dal fatto che non ci sono radure con più di tre sentieri collegati, possiamo concludere che ogni forestale deve osservare almeno due sentieri che gli altri forestali non osservano.

Per osservare il vicolo cieco tra la radura F e G, un forestale deve essere posizionato sulla radura F. Per osservare il sentiero tra la radura B e C, il secondo forestale deve osservare dalla radura B. Per osservare gli ultimi due sentieri con un solo forestale, quest'ultimo deve essere posizionato sulla radura D. In questo modo, la soluzione data è definitiva e non può esserci altro.

Questa è l'informatica!

Le relazioni tra le cose (per esempio i sentieri tra le radure) possono essere rappresentate come un cosiddetto *grafo*. Un grafo è composto da *vertici* (qui: le radure), rappresentati come cerchi, e *archi* (qui: i sentieri), rappresentati come linee tra i vertici. Il grafo di questo compito si presenta così:



In questo compito, devi trovare il minor numero di vertici nel grafo in modo che ogni arco inizi o finisca in almeno uno di questi vertici. Gli informatici chiamano un tale sottoinsieme di nodi una



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

Parole chiave e siti web

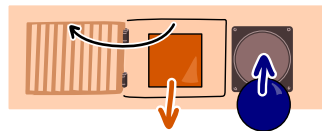
- Grafo: <https://it.wikipedia.org/wiki/Grafo>
- Copertura dei vertici: https://it.wikipedia.org/wiki/Copertura_dei_vertici





15. Puzzle di compleanno

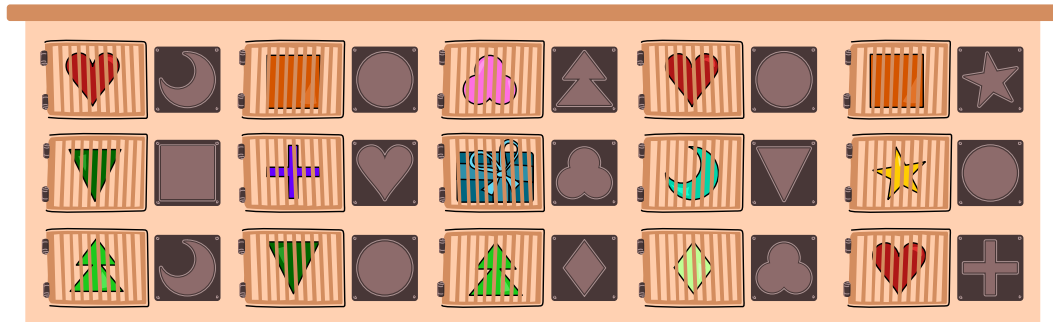
Bastian riceve una scatola con 15 porte per il suo compleanno. Dietro la porta centrale c'è il vero regalo. Dietro le altre porte ci sono blocchetti di varie forme. Ogni porta ha un foro a destra della porta. Bastian può aprire una porta inserendo un blocco della stessa forma nel foro - come una chiave.



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

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

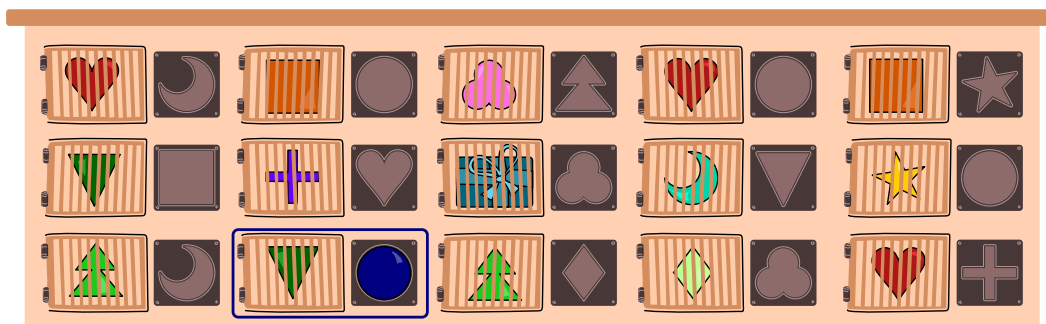
Quale porta deve aprire per prima Bastian?



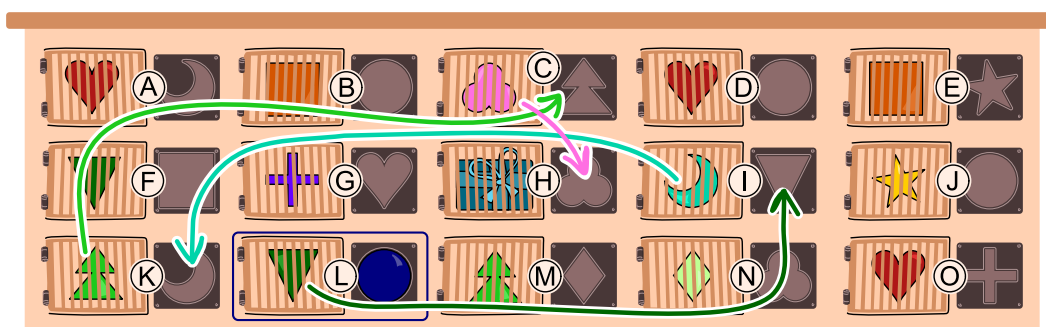


Soluzione

Bastian deve prima aprire la porta segnata in blu:



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



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



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

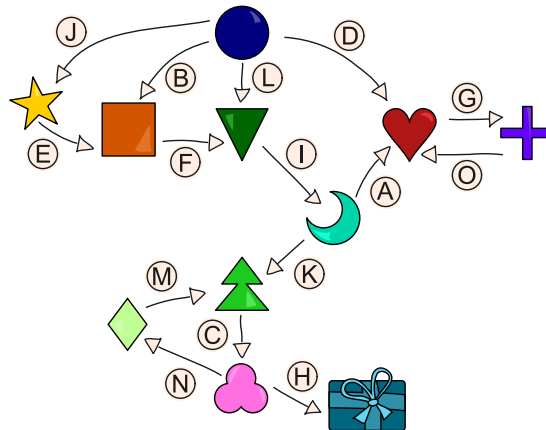


Ma questi modi sono troppo lunghi, bisognerebbe aprire più di cinque porte. Provare tutte le possibilità richiede molto tempo.

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

Questa è l'informatica!



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



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

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

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

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

Parole chiave e siti web

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

























16. Regalo preferito

La famiglia castoro ha cinque regali per i suoi cinque 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.

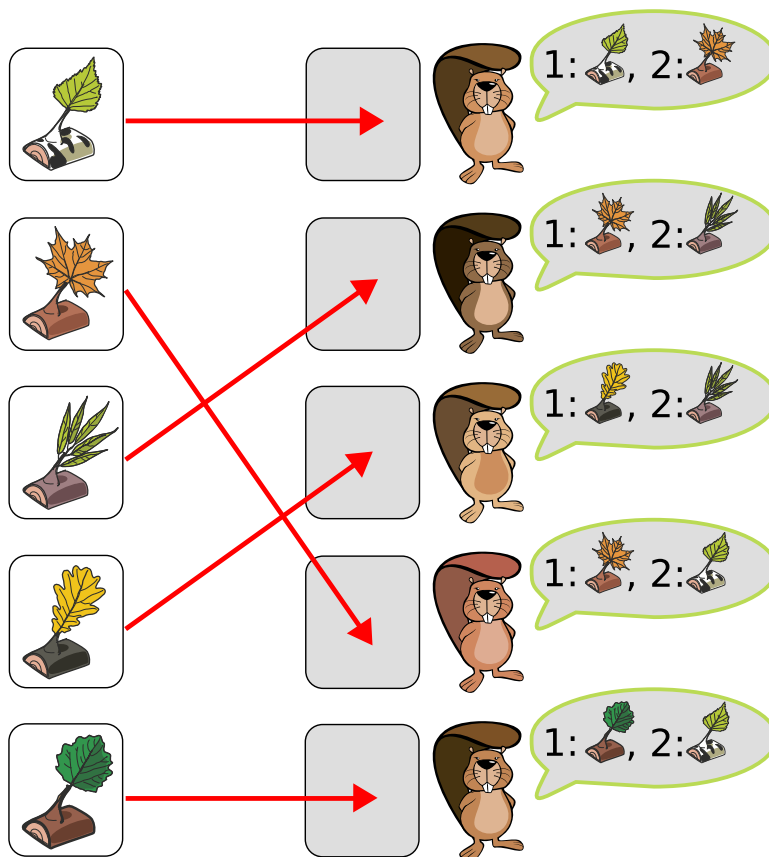
Dai ai castorini i regali giusti.

	<input type="checkbox"/>		1:  , 2: 
	<input type="checkbox"/>		1:  , 2: 
	<input type="checkbox"/>		1:  , 2: 
	<input type="checkbox"/>		1:  , 2: 
	<input type="checkbox"/>		1:  , 2: 



Soluzione

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



Il grafico qui sopra assegna a quattro castorini il loro regalo preferito e a un castorino il suo secondo regalo preferito. Non tutti i castorini possono ottenere il loro regalo preferito perché due castorini hanno lo stesso regalo preferito. Pertanto, non è possibile nessun assegnamento dove più castorini ottengono il loro regalo preferito. Nota: se fai l'assegnazione dall'alto verso il basso e assegni il secondo regalo al secondo castoro, allora il quarto castoro non riceverà nessuno dei suoi regali preferiti. Quindi in questo compito non è sufficiente fare la migliore selezione per ogni singolo castorino.

Una strategia di risoluzione è quella di assegnare prima tutti i regali che sono il regalo preferito di un solo castorino. In seguito, solo due castorini rimangono con lo stesso regalo preferito. Si valuta in seguito a quale castorino può essere assegnato il secondo regalo preferito. Si assegna dunque all'ultimo castorino 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 *accoppiamento* (in inglese *matching*) è *ottimale* se non c'è un altro accoppiamento con più prime selezioni 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? Allora dovrete studiare informatica!

Parole chiave e siti web

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



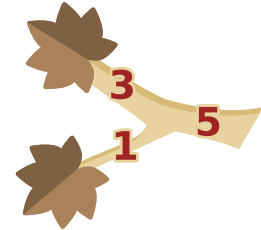


17. Salvataggio dell'albero

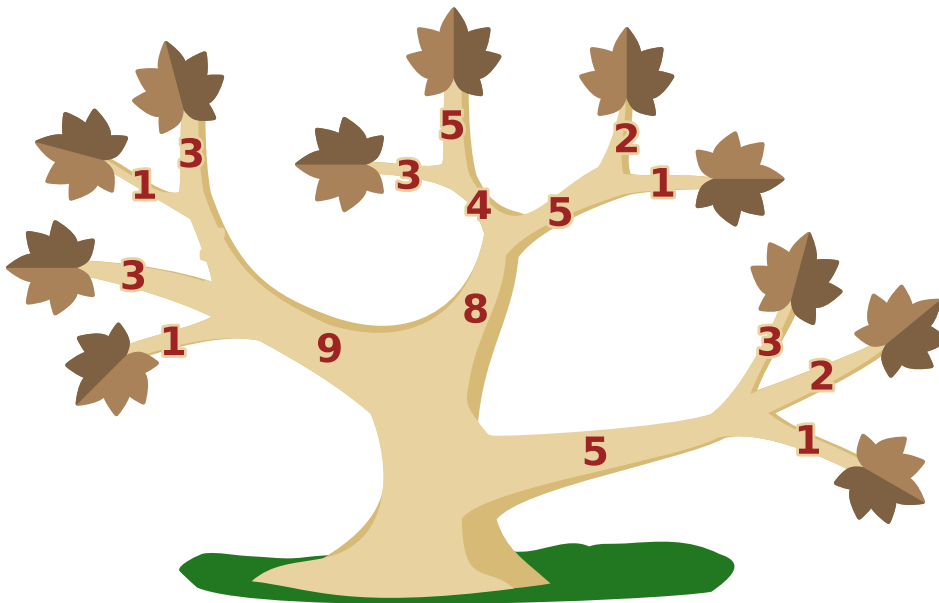
Un albero nel giardino di Bruno è malato e tutte le foglie si sono seccate. Bruno vuole salvare l'albero. Per farlo, deve segare alcuni rami in modo che alla fine tutte le foglie siano rimosse e che possano crescere nuovi rami con nuove foglie.

Bruno vuole finire il prima possibile. L'immagine mostra un esempio:

Per tagliare le due foglie, Bruno può segare i due rami con le foglie o solo il ramo da cui si dipartono gli altri due. I numeri indicano per ogni ramo quanto tempo ci vuole per il taglio. Bruno segherà quindi i due rami con le foglie, perché $3 + 1 < 5$. Qui sotto si vede l'intero albero.



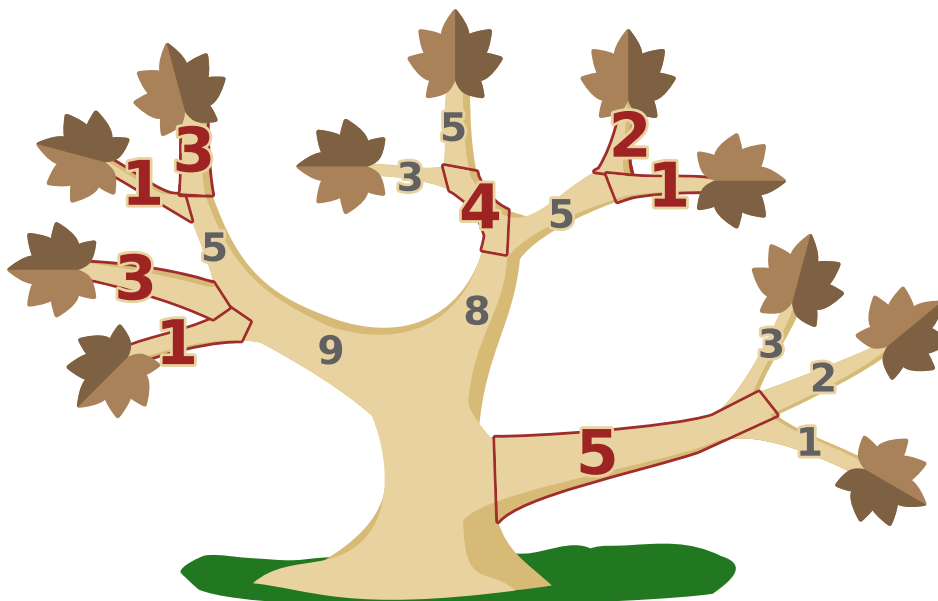
Quali rami segherà Bruno per finire il più velocemente possibile?





Soluzione

Questa è la soluzione giusta: Bruno sega i rami segnati in rosso per finire il più velocemente possibile:



Ma perché? Prima di tutto, possiamo calcolare di quanto tempo ha bisogno Bruno se sega solo i rami con le foglie:

$$1 + 3 + 1 + 3 + 3 + 5 + 2 + 1 + 3 + 2 + 1 = 25$$

Ora continuiamo verso il tronco e consideriamo se potrebbe essere più veloce segare il ramo da cui dipartono direttamente o indirettamente i rami precedenti. Dopo il primo di questi passi, il seguente calcolo risulta (la funzione «min» calcola il minimo dei suoi argomenti):

$$\begin{aligned} & 1 + 3 + \min(5, 1 + 3) + \min(4, 3 + 5) + \min(5, 2 + 1) + \min(5, 3 + 2 + 1) \\ &= 1 + 3 + (1 + 3) + 4 + (2 + 1) + 5 \\ &= 20 \end{aligned}$$

All'inizio non calcoliamo il tempo totale, in modo da vedere meglio quali rami devono essere tagliati. Dopo il prossimo passo abbiamo già raggiunto il tronco:

$$\begin{aligned} & \min(9, 1 + 3 + 1 + 3) + \min(8, 4 + 2 + 1) + 5 \\ &= (1 + 3 + 1 + 3) + (4 + 2 + 1) + 5 \\ &= 20 \end{aligned}$$

Bruno non può essere più veloce di così.



Questa è l'informatica!

Immaginiamo che i pezzi segati dell'albero di Bruno non cadano direttamente a terra - come succede quando si risolve questo compito sullo schermo. Allora potremmo dire che l'albero si divide in due sole parti segandolo: Una parte contiene tutti i pezzi segati, cioè in particolare tutte le foglie, e l'altra parte contiene il tronco e tutti i rami che si estendono da esso ai punti di taglio. Questa divisione o *taglio* attraverso l'albero è minima in termini di tempo che Bruno deve spendere per segare.

Anche l'informatica conosce gli alberi e li usa per rappresentare oggetti che sono collegati tra loro in un certo modo. Gli oggetti sono chiamati *vertici*, le connessioni *archi*. C'è sempre un solo percorso tra due vertici lungo gli archi - proprio come in un albero reale c'è sempre un solo percorso lungo i rami da una foglia o una biforcazione al tronco. Se rinunciamo a questa condizione, si parla più generalmente di un *grafo*.

In un grafo generale, un *taglio minimo*, cioè la scomposizione in due o più parti con costi minimi, non è così facile da calcolare come abbiamo dimostrato qui per un albero, ma non è nemmeno troppo difficile. Questo è un bene, perché ci sono applicazioni interessanti. I tagli minimi possono essere utilizzati, per esempio, nella scomposizione di file di immagini in parti simili. In grafi speciali, le *reti di flusso*, che possono essere utilizzate per modellare i flussi di dati attraverso le reti, tra le altre cose, il costo di un taglio minimo corrisponde al massimo flusso possibile attraverso l'intera rete.

Parole chiave e siti web

- Albero: [https://it.wikipedia.org/wiki/Albero_\(grafo\)](https://it.wikipedia.org/wiki/Albero_(grafo))
- Taglio: [https://it.wikipedia.org/wiki/Taglio_\(teoria_dei_grafi\)](https://it.wikipedia.org/wiki/Taglio_(teoria_dei_grafi))
- Rete di flusso: https://it.wikipedia.org/wiki/Rete_di_flusso
- Teorema del flusso massimo e taglio minimo:
https://it.wikipedia.org/wiki/Teorema_del_flusso_massimo_e_taglio_minimo





18. Biblioteca

Susi è nella biblioteca dei castori con Tim e vuole prendere in prestito un libro: «Dolce Bebras A Ginevra»

Tim va allo scaffale 1, raggiunge la fila 3, scomparto 6 ed estrae il libro. Susi è impressionata dalla sua velocità! Tim spiega a Susi come determinare la posizione di un libro:

Prendi la prima lettera di ogni parola nel titolo e determina la sua posizione nell'alfabeto. Somma questo valore al valore della lettera precedente, ma prima di ogni somma, il valore totale raggiunto finora viene moltiplicato per 3. Il risultato per il libro desiderato è 136, è quindi chiaro dove si trova il libro.

a	b	c	d	e	f	g	h	i	j	k	l	m
1	2	3	4	5	6	7	8	9	10	11	12	13
n	o	p	q	r	s	t	u	v	w	x	y	z
14	15	16	17	18	19	20	21	22	23	24	25	26

D	o	l	c	e	B	e	b	r	a	s	A	G	i	n	e	v	r	a
4					2						1							7

$$((4 \cdot 3 + 2) \cdot 3 + 1) \cdot 3 + 7$$

In seguito, Susi prepara i codici corrispondenti per recuperare i suoi libri preferiti. In un caso, tuttavia, ha commesso un errore.

Quale di questi codici è stato calcolato in modo errato?

- | | | | |
|----|--|----|--|
| A) | Gran giorno, bel fiore!
$((7 \cdot 3 + 7) \cdot 3 + 2) \cdot 3 + 6$ | B) | Bebretti forse fa danni
$((2 \cdot 3 + 6) + 6) \cdot 3 + 4$ |
| C) | Dr. Hal danza delicato
$((4 \cdot 3 + 8) \cdot 3 + 4) \cdot 3 + 4$ | D) | Bari: dire e fare
$((2 \cdot 3 + 4) \cdot 3 + 5) \cdot 3 + 6$ |



Soluzione

Susi ha fatto quasi tutto giusto: ha sempre aggiunto i valori delle posizioni corretti e ha sempre moltiplicato i risultati intermedi per 3 - con un'eccezione: nella risposta B ha dimenticato la moltiplicazione una volta!

Bebretti forse fa danni
$((2 \cdot 3 + 6) \cdot 3 + 6) \cdot 3 + 4$

Questa è l'informatica!

Con le «espressioni di localizzazione», la biblioteca consente ai suoi visitatori di determinare l'esatta ubicazione dei libri, cosicché nessuno debba cercare troppo a lugo. Tuttavia, c'è una cosa che la biblioteca e i visitatori devono tenere a mente: le espressioni e quindi i loro risultati possono essere gli stessi per libri diversi. Ad esempio, «Abbattimento di alberi» e «Andiamo da Alberto» si trovano nello stesso scomparto. Gli scomparti quindi non devono essere troppo piccoli, oppure devono essere sufficientemente flessibili per essere adattati.

Anche con i dati archiviati nella memoria del computer si usa spesso la stessa metodologia: è infatti una buona idea se la loro posizione in memoria può essere calcolata direttamente dai dati stessi. A questo scopo sono state sviluppate in informatica le funzioni hash: funzioni matematiche che calcolano un valore dal contenuto dei dati o da parte dei dati e che indica direttamente la posizione di archiviazione - come con i titoli dei libri in questo esercizio. Buone funzioni di hash assicurano che lo stesso valore risulti nel minor numero possibile di casi. Se si verifica tali collisioni, l'informatica conosce buoni metodi per affrontarla.

Parole chiave e siti web

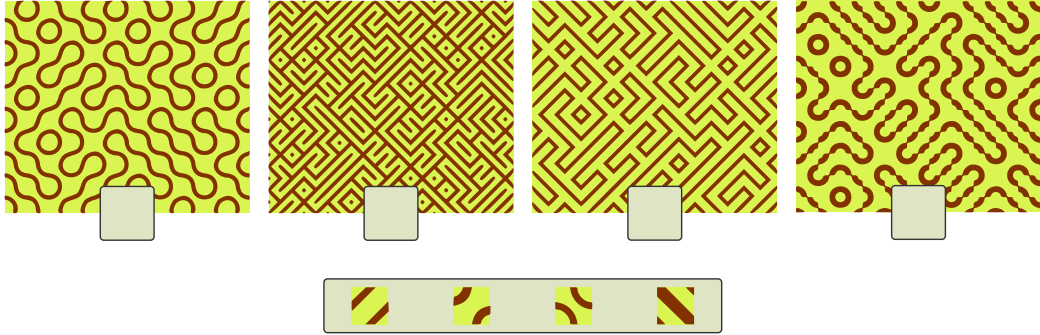
- Funzione di hash: https://it.wikipedia.org/wiki/Funzione_di_hash
- Hash table: https://it.wikipedia.org/wiki/Hash_table



19. Piastrelle Truchet

I seguenti modelli sono stati creati ciascuno da una singola piastrella. Le singole piastrelle sono mostrate ingrandite.

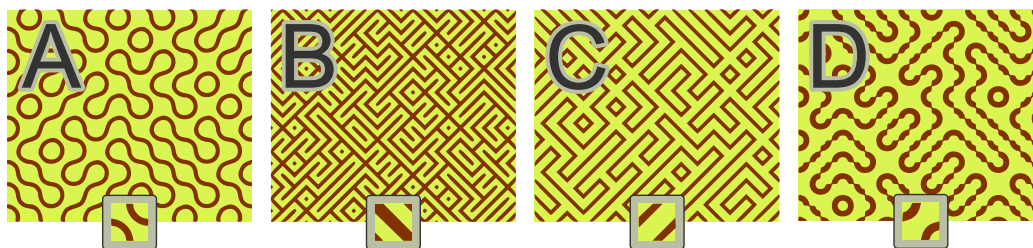
Abbina le piastrelle ai loro possibili modelli.



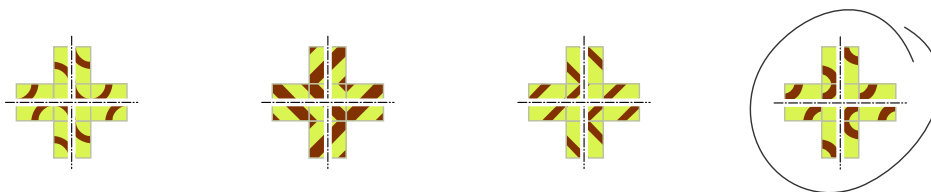






Soluzione

Questa è la classificazione corretta:



Se metti 5 piastrelle una accanto all'altra e le compari più da vicino, puoi vedere delle chiare differenze:



La piastrella  è l'unica delle piastrelle con quattro lati che non si adattano esattamente l'uno all'altro. Questo è l'unico modo per creare linee di larghezza variabile come nel modello D. La piastrella  è l'unica che può creare punti quadrati nel modello B, cioè con quattro triangoli che si incontrano. Inoltre, ha la superficie del marrone più ampia rispetto al giallo, proprio come B; questo dimostra anche che fa parte del modello B. Rimane solo il modello A come possibile risultato per le forme rotonde della piastrella  e solo il modello C per le forme diritte della piastrella .

Questa è l'informatica!

Queste piastrelle prendono il nome da Sébastien Truchet (* 1657; † 1729), che ha lavorato su diverse varianti di queste piastrelle. Le piastrelle con 4 lati uguali formano un sottogruppo di piastrelle Truchet (ma le piastrelle Truchet non devono necessariamente avere 4 lati uguali, come in 3 degli schemi precedenti). Il fatto che schemi complessi possano essere creati con blocchi di costruzione molto semplici è una proprietà interessante che incontriamo ancora e ancora nell'informatica. Le piastrelle Truchet sono studiate in matematica e in informatica e utilizzate nei giochi di computer per creare labirinti o decorazioni.

Parole chiave e siti web




- Piastrelle Truchet: https://it.wikinew.wiki/wiki/Truchet_tiles

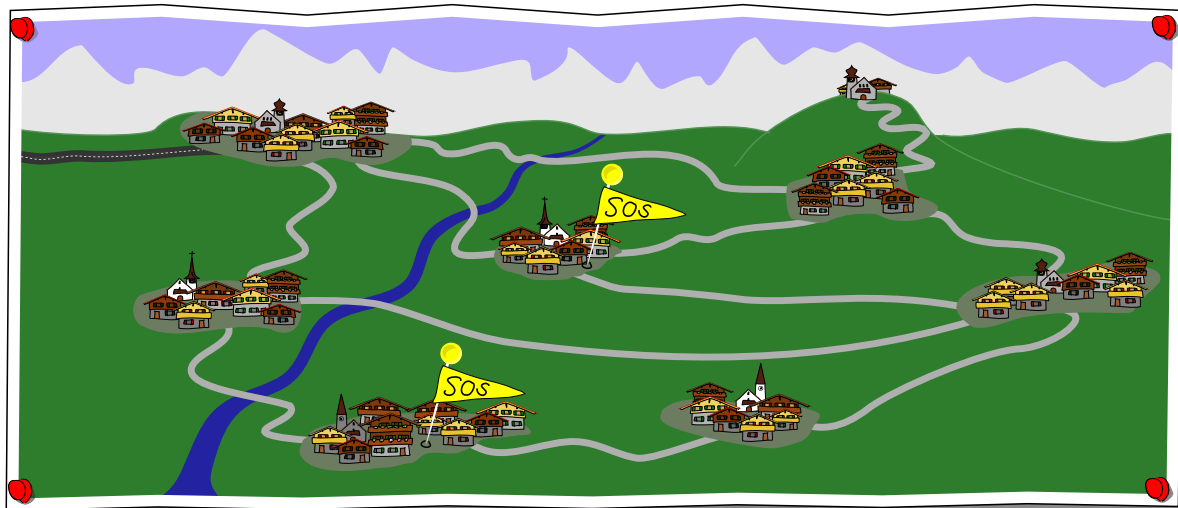


20. Villaggi isolati

Alcuni villaggi di montagna sono collegati alla grande città attraverso la seguente rete stradale.

Dopo una tempesta, diversi villaggi segnalano che non sono più accessibili, in particolare quelli con i segnali di SOS. Possiamo concludere che alcune strade sono bloccate.

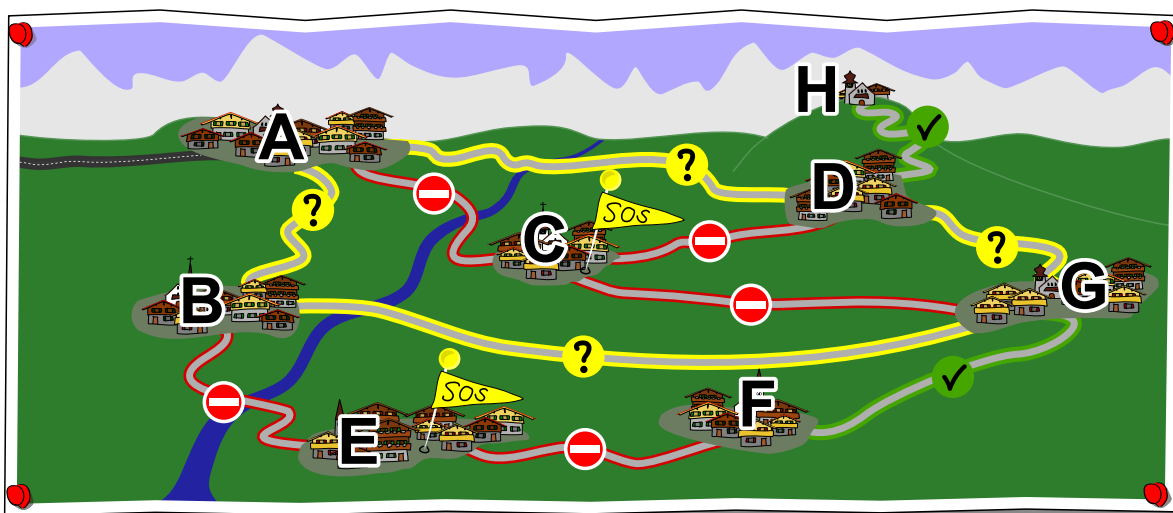
Per ogni strada tra villaggi in questa rete stradale, segnala se è (1) bloccata , (2) aperta , o (3) non possiamo essere sicuri senza ulteriori informazioni che la strada sia aperta o bloccata .





Soluzione

La mappa mostra quello che sappiamo sulle connessioni della rete stradale:



Iniziamo a riconoscere le strade bloccate. Le due strade che portano al villaggio E possono essere bloccate, perché altrimenti il villaggio E sarebbe ancora raggiungibile. Allo stesso modo, le tre strade che portano al villaggio C sono bloccate, perché altrimenti il villaggio C sarebbe ancora raggiungibile.

Poi, cerchiamo le strade che devono essere aperte. La strada tra il villaggio G e F deve essere aperta, altrimenti, a causa della strada bloccata tra il villaggio F ed E, il villaggio F non sarebbe raggiungibile. Anche la strada tra la chiesa H e il villaggio D deve essere aperta, poiché H è accessibile e può essere raggiunta solo attraverso D.

Ora rimangono le strade che potrebbero essere aperte. Poiché i villaggi B, G e D sono collegati più volte al villaggio A, non possiamo dire quali delle strade rimanenti sono aperte. Per esempio, il villaggio B potrebbe essere raggiunto attraverso il villaggio A, ma anche attraverso il villaggio G. Lo stesso vale per il villaggio D. Il villaggio G può essere servito attraverso il villaggio B o D. Quindi una qualsiasi delle strade del circuito A - B - G - D - A potrebbe essere bloccata e questi 4 villaggi potrebbero ancora rimanere tutti accessibili.

Questa è l'informatica!

Proprio come nelle reti stradali, le connessioni nelle reti di computer possono essere problematiche, sovraccaricate o completamente difettose. Per prevenire i guasti, sono spesso previste misure di sicurezza, come connessioni multiple ad un luogo. Questo si chiama *ridondanza*.

Sistemare i difetti in un sistema è un compito che gli informatici devono fare molto spesso, non solo nelle reti di computer ma anche nello sviluppo di software. Per correggere un errore è necessario identificare la sua fonte esatta, e questo processo è di solito fatto passo dopo passo in diverse fasi. Alcuni programmatori credono che non sia possibile trovare tutti gli errori e bug di un programma.



Parole chiave e siti web

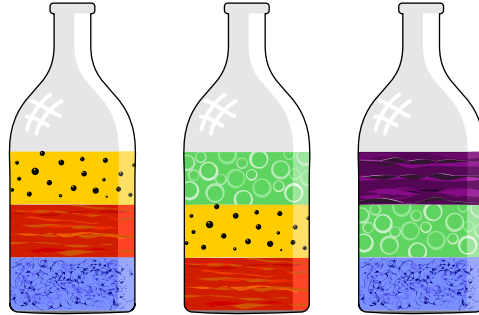
- Ridondanza: [https://it.wikipedia.org/wiki/Ridondanza_\(ingegneria\)](https://it.wikipedia.org/wiki/Ridondanza_(ingegneria))
- Debugging: <https://it.wikipedia.org/wiki/Debugging>



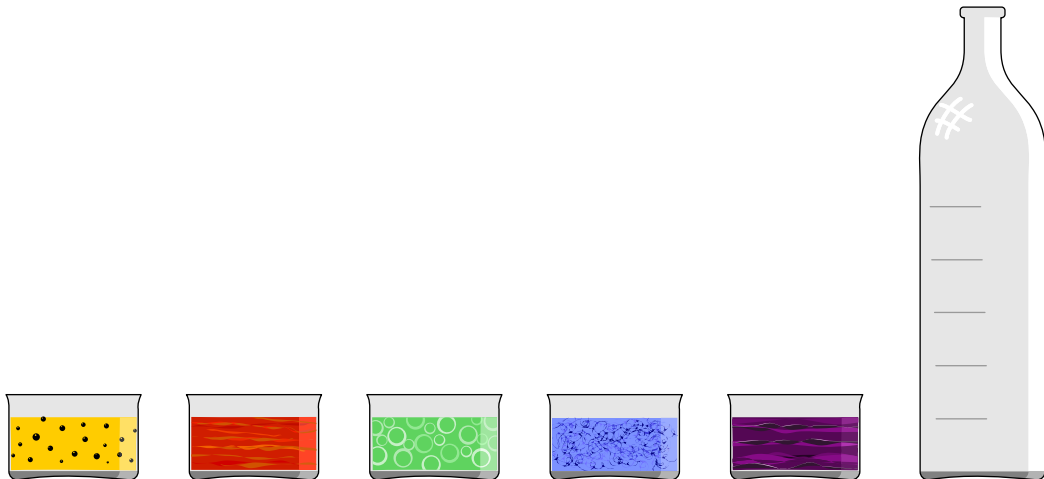


21. Disposizione dei liquidi

Mark ha delle bottiglie contenenti tre liquidi colorati ciascuna, stratificati l'uno sull'altro. Sa che i liquidi con densità minore si muovono sempre sopra i liquidi con densità maggiore. Ora vuole vedere cosa succede quando si mettono tutti i liquidi colorati in una bottiglia.



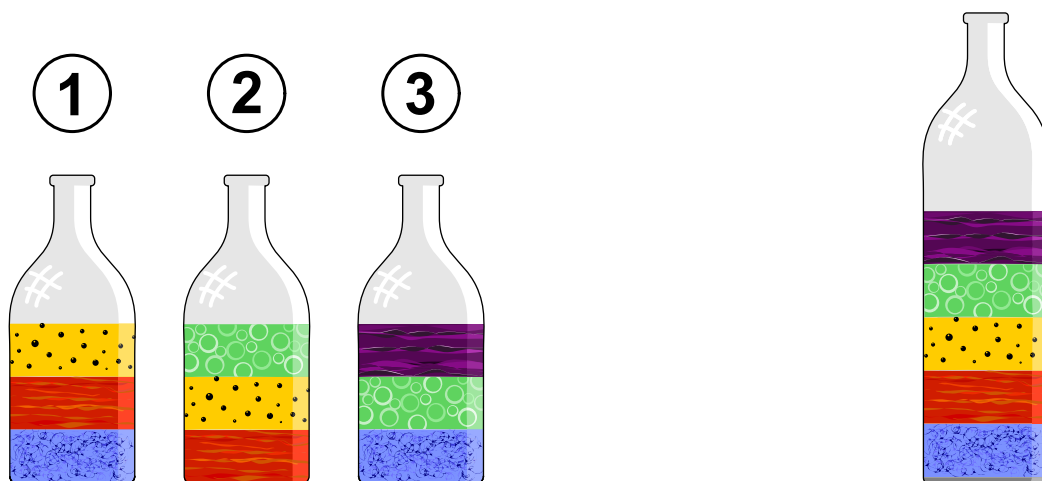
Disponi i cinque liquidi colorati nella bottiglia come apparirebbero dopo averli mischiati tutti!





Soluzione

L'immagine mostra la corretta disposizione dei cinque liquidi colorati nella bottiglia grande.



L'ordine si trova seguendo la seguente procedura: passo dopo passo, rimuovi dalle tre bottiglie date i liquidi che non sono sopra un altro liquido e mettili nella bottiglia grande.

All'inizio, solo le bottiglie 1 e 3 hanno un liquido blu e si trova sul fondo, quindi non si trova da nessuna parte su un altro strato di liquido. Il liquido rosso è sul fondo della bottiglia 2. Ma nella bottiglia 1 è sopra il liquido blu e deve quindi avere una densità inferiore al liquido blu. Quindi la prima cosa da fare è togliere il liquido blu dalle bottiglie e metterlo nella bottiglia grande.

Ora il liquido rosso è l'unico che non si trova sopra un altro liquido. Viene tolto dalle bottiglie 1 e 2 e messo nella bottiglia grande. Poi viene il liquido giallo, poi il liquido verde e infine il liquido viola, che ha la densità più bassa e sopra il quale non si trova nessun altro liquido.

Questa è l'informatica!

Nel risolvere questo compito, hai valutato la disposizione dei liquidi nelle tre bottiglie del compito e hai ordinato i liquidi in base alla loro densità.

Una sostanza ha molte proprietà misurabili: temperatura di ebollizione, temperatura di fusione, conducibilità elettrica e densità. In questo caso, la densità è stata utilizzata come criterio di selezione delle sostanze.

L'ordinamento dei dati gioca un ruolo importante in molti programmi informatici. Il metodo usato in questo compito per determinare l'ordine degli strati liquidi è chiamato *ordinamento topologico*. Si usa per ordinare gli oggetti per i quali sono conosciute relazioni del tipo predecessore/successore.

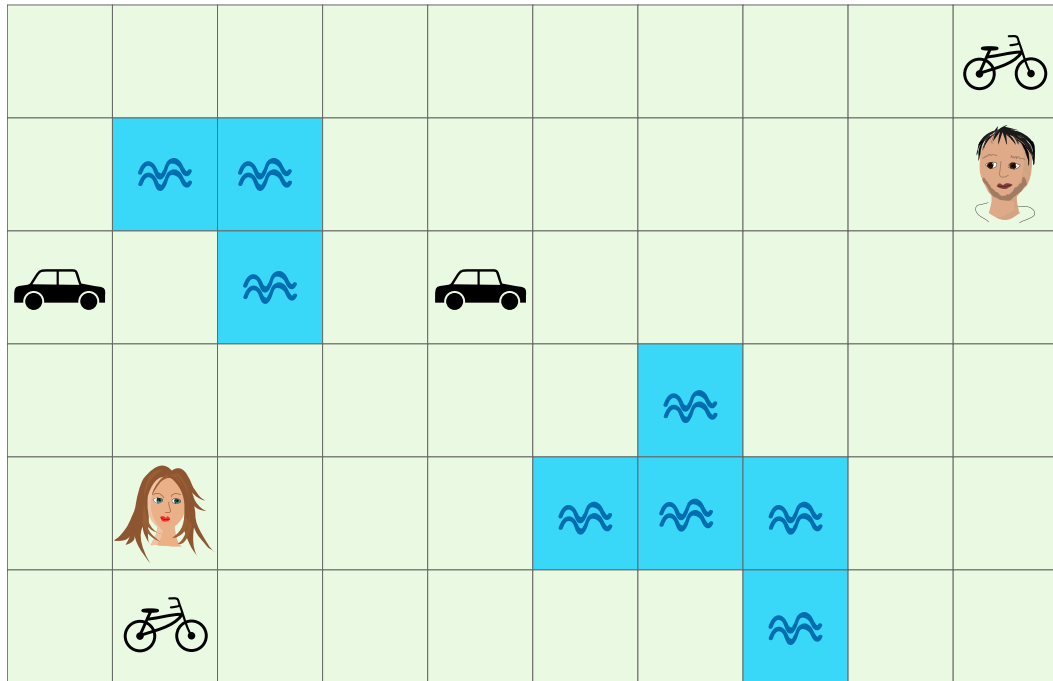
Parole chiave e siti web

- Algoritmo di ordinamento: https://it.wikipedia.org/wiki/Algoritmo_di_ordinamento
- Ordinamento topologica: https://it.wikipedia.org/wiki/Ordinamento_topologico



22. Riunione veloce

Due amici vogliono incontrarsi il più presto possibile. Possono spostarsi da una casella a una casella adiacente a sinistra, a destra, in alto o in basso. Ci vuole 1 minuto per farlo a piedi. Se raggiungono una casella con un veicolo, possono usarlo. Con una bicicletta possono raggiungere 2 campi in un minuto e con una macchina 5 campi. I cambi di direzione sono possibili. Non possono attraversare le superfici d'acqua.



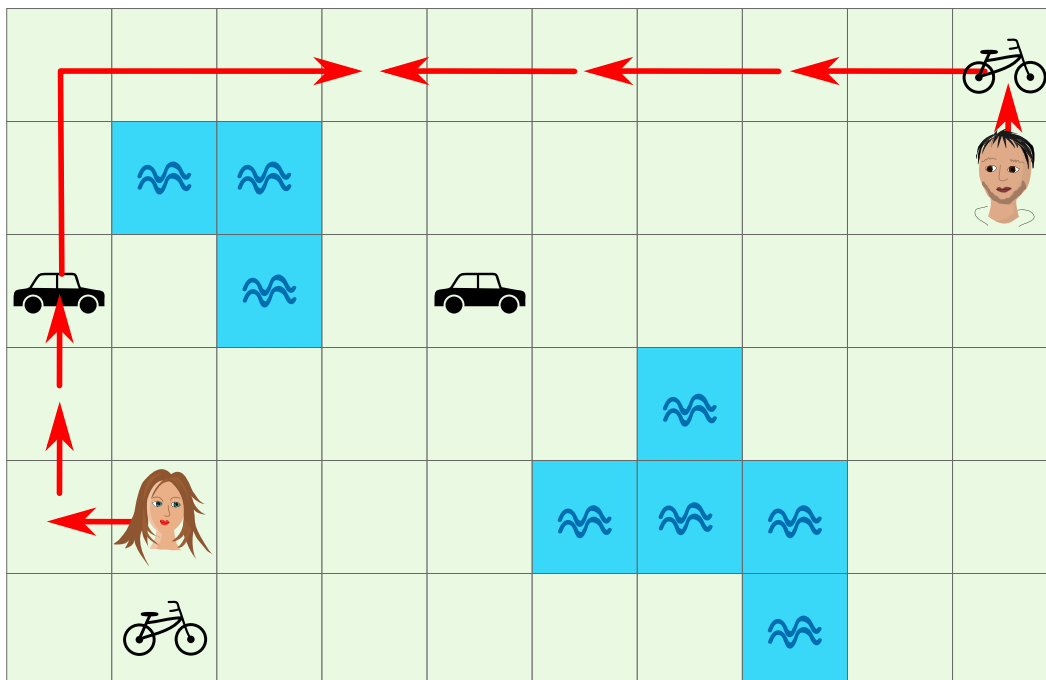
Di quanti minuti hanno bisogno i due amici per incontrarsi su una casella?

- A) 1 minuto
- B) 2 minuti
- C) 3 minuti
- D) 4 minuti
- E) 5 minuti
- F) 6 minuti



Soluzione

La risposta corretta è 4. L'immagine mostra un percorso che permette ai due amici di incontrarsi in una casella in 4 minuti.



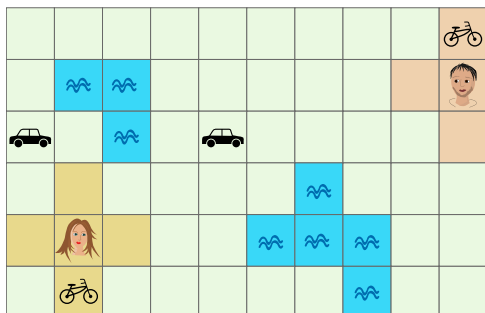
Ora bisogna dimostrare che non possono incontrarsi in 3 minuti: I due amici sono a 11 campi di distanza l'uno dall'altro. In 3 minuti, però, possono avvicinarsi a piedi solo un totale di 6 campi. Se uno ha raggiunto la bicicletta e l'altro sta camminando, allora possono avvicinarsi di 9 caselle l'uno all'altro in 3 minuti, che non è neanche abbastanza. Anche se entrambi camminano verso una bicicletta, non è sufficiente. Perché allora potrebbero avvicinarsi di 12 spazi in 3 minuti, ma le due biciclette sono distanti 13 spazi.

Quindi l'unica opzione è usare una macchina. In 3 minuti, solo la ragazza può raggiungere una macchina. Ma poi non c'è più tempo per usare la macchina. E in 3 minuti il ragazzo non può raggiungere una casella con una macchina.

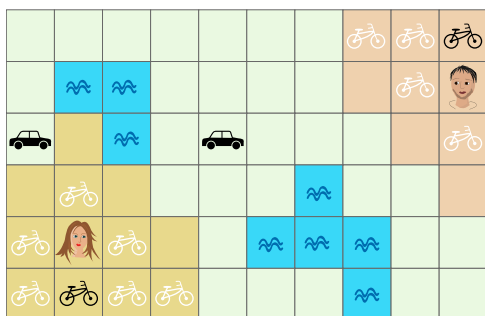
Questa è l'informatica!

Come hai risolto il compito? Hai trovato un percorso veloce per caso e hai sperato che non ce ne fosse uno più veloce? O hai provato molte possibilità e hai scoperto quella più veloce?

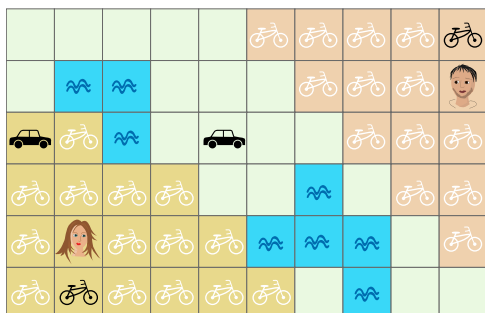
I programmi per computer progettati per questo tipo di problemi di solito usano un metodo chiamato *ricerca in ampiezza*. In questo problema, la ricerca in ampiezza funziona come segue:



1. Segna tutte le caselle che possono essere raggiunte dai due amici in un minuto.

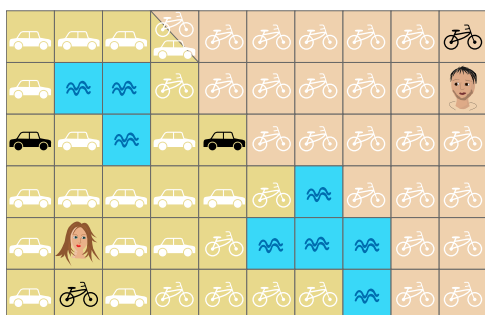


2. Segna tutti i campi che possono essere raggiunti in (al massimo) un minuto dai campi segnati nel passo 1. Annota anche i mezzi di trasporto utilizzati.



3. Segna tutti i campi che possono essere raggiunti in un minuto dai campi che sono stati segnati nel passo 2.

Poiché le due aree che hai segnato finora non si sovrappongono, gli amici non possono incontrarsi dopo 3 minuti.



4. Ora segna tutti i campi che possono essere raggiunti in un minuto dai campi segnati nel passo 3.

Ora le due aree si sovrappongono in un campo. Può essere raggiunto dopo 4 minuti dalla ragazza con una macchina e dal ragazzo con una bicicletta. I sistemi di navigazione trovano il percorso più veloce tra due punti. Si assicurano che il percorso segua strade e sentieri adatti - e non attraverso la campagna e i fiumi. Questo compito è simile al problema della navigazione, tranne che qui due persone devono essere guidate verso una destinazione comune - inizialmente sconosciuta - piuttosto che una sola persona verso una destinazione fissa.

Poiché un computer procede sistematicamente nella ricerca in ampiezza, trova anche soluzioni che non sono immediatamente ovvie. A volte una deviazione con meno semafori è più veloce del percorso



più breve tra la partenza e la destinazione. Un collegamento in treno con un cambio di treno può essere più veloce di un collegamento diretto in autobus. Nell'informatica, ci sono diversi metodi per trovare la migliore soluzione a un problema di questo tipo. Oltre alla ricerca in ampiezza, che è stata appena descritta, c'è anche un approccio chiamato *Branch and Bound* (*ramificazione e limitazione* in inglese).

Nella ricerca in ampiezza, viene considerata ogni soluzione parziale. Con *Branch and Bound*, non si perseguono ulteriormente le soluzioni parziali se si sa che non possono portare alla soluzione ottimale.

Se un problema diventa troppo complesso, ci vorrebbe troppo tempo anche per il computer più veloce del mondo per passare attraverso tutte le possibilità e trovare la soluzione migliore. In pratica, con un sistema di navigazione, è spesso sufficiente trovare un ottimo percorso, anche se non è il migliore possibile (se puoi raggiungere la tua destinazione in 78 minuti, probabilmente non ti importa se potrebbe teoricamente essere raggiunta in 77 minuti).

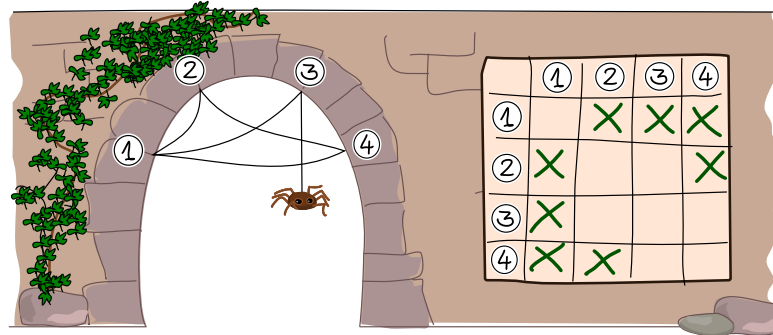
Parole chiave e siti web

- Ricerca in ampiezza: https://it.wikipedia.org/wiki/Ricerca_in_ampiezza
- Algoritmo branch and bound: https://it.wikipedia.org/wiki/Branch_and_bound



23. Le ragnatele di Thekla

Il ragno Thekla vuole costruire quante più ragnatele diverse possibili. Per questo ha inventato un metodo per documentare l'esatta costruzione delle sue ragnatele.

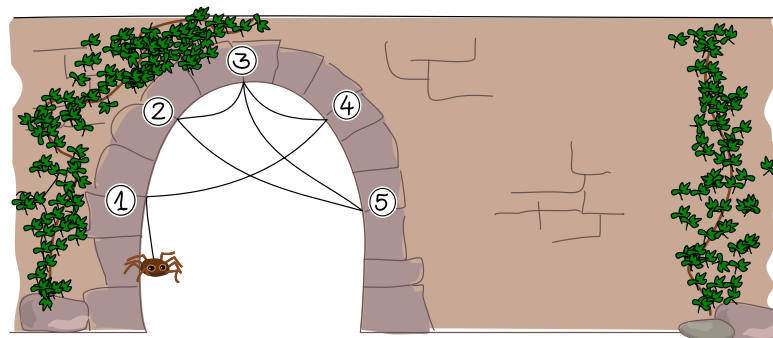


Il metodo funziona così: numera i punti finali della ragnatela da 1 a N e usa i campi in una griglia secondo la seguente regola:

- Se c'è un filo che collega il punto finale x con il punto finale y , allora il campo nella colonna x e nella riga y è segnato con una «x».

Un filo che collega il punto finale x con il punto finale y collega anche il punto finale y con il punto finale x .

Thekla costruisce ora questa ragnatela:



Come documenta Thekla la costruzione di questa ragnatela?



A)

	①	②	③	④	⑤
①				X	
②			X		X
③		X		X	X
④	X		X		
⑤		X	X		

B)

	①	②	③	④	⑤
①		X		X	
②	X		X		
③		X		X	X
④	X		X		
⑤			X		

C)

	①	②	③	④	⑤
①	X			X	
②			X		X
③		X		X	X
④	X		X	X	
⑤		X	X		

D)

	①	②	③	④	⑤
①				X	
②			X		X
③		X		X	X
④	X		X		
⑤			X		



Soluzione

La risposta corretta è A, perché tutti i campi sono segnati correttamente secondo la regola.

	①	②	③	④	⑤
①				X	
②			X		X
③		X		X	X
④	X		X		
⑤		X	X		

Nella risposta B, una connessione aggiuntiva è stata disegnata in modo errato (punto finale 1 al punto finale 2 in entrambe le direzioni) e una è stata dimenticata (punto finale 2 al punto finale 5 in entrambe le direzioni).

	①	②	③	④	⑤
①		XXXX		X	
②	XXXX		X		X
③		X		X	X
④	X		X		
⑤		X	X		

Risposta C: Secondo la regola qui descritta, non ci possono essere segni nella diagonale dall'alto a sinistra al basso a destra. Perché sarebbero connessioni di un punto finale con se stesso. Questo potrebbe essere permesso in alcune ragnatele, ma non si verifica nella nostra ragnatela. Nella risposta C, invece, avremmo 2 connessioni di questo tipo (al punto finale 1 e al punto finale 4).

	①	②	③	④	⑤
①	XXXX			X	
②			X		X
③		X		X	X
④	X		X	XXXX	
⑤		X	X		

Risposta D: Tutte le rappresentazioni di ragnatele devono essere simmetriche rispetto alle diagonali dall'alto a sinistra al basso a destra. In questa risposta, la connessione dal punto finale 2 al punto finale 5 è presente, ma manca la connessione corrispondente di ritorno dal punto finale 5 al punto finale 2.

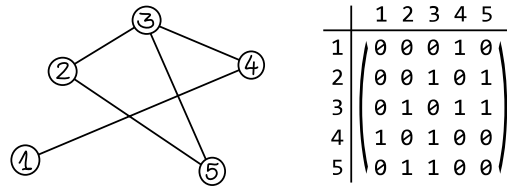
	①	②	③	④	⑤
①				X	
②			X		X
③		X		X	X
④	X		X		
⑤		X	X		

Questa è l'informatica!

La ragnatela può essere considerata un *grafo*, un concetto spesso usato in informatica.

Un grafo è composto da *vertici* (i punti finali della ragnatela) ed *archi* (i fili tra i punti finali). I grafi sono anche usati, per esempio, per rappresentare gli oggetti e le relazioni tra loro. Per esempio, un grafo potrebbe mostrare come le persone sono amiche nelle reti sociali, o i voli tra paesi.

Questo compito mostra come memorizzare la struttura di una ragnatela in una griglia. Alcune proprietà, come l'aspetto esatto della ragnatela, si perdono nel processo. In molti casi, tuttavia, non si è interessati alle proprietà geometriche esatte di una rete, ma solo alla sua struttura. Le informazioni essenziali sono conservate: quanti vertici ci sono? E tra quali coppie di vertici c'è un arco?



La possibilità presentata è solo uno dei tanti modi per registrare la struttura di una rete. Il metodo non è molto economico, perché entrambe le direzioni sono memorizzate per ogni connessione, il che non sarebbe necessario, e i campi diagonali liberi non sarebbero affatto necessari. D'altra parte, questo metodo ha il vantaggio che gli errori di rappresentazione possono essere parzialmente rilevati. La risposta C e la risposta D, per esempio, potrebbero essere riconosciute come sbagliate senza fare riferimento alla rete.

La forma di rappresentazione presentata si chiama *matrice delle adiacenze*.

Parole chiave e siti web

- Matrice delle adiacenze: https://it.wikipedia.org/wiki/Matrice_delle_adiacenze







24. Pila di frutta

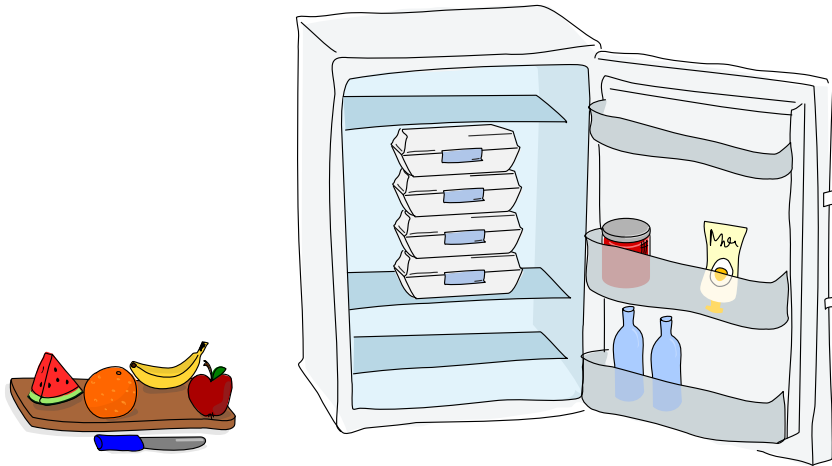
Papà, mamma, Dorie e Ron Castoro preparano la sera quattro scatole per la colazione del giorno dopo, ognuna con un frutto diverso: mela, banana, arancia e anguria. Le scatole sono impilate l'una sull'altra nel frigorifero. Al mattino, i castori sono ancora molto stanchi e quando lasciano la tana prendono semplicemente la scatola più in alto senza guardarla attentamente.

Non sappiamo esattamente in quale ordine i castori lasciano la tana, ma in ogni caso Mamma va sempre prima di Dorie e Papà sempre come l'ultimo.

Ai membri della famiglia piacciono frutti diversi. La tabella mostra cosa piace ad ogni membro della famiglia.

				
Papà	—	—	✓	—
Mamma	✓	—	✓	✓
Dorie	✓	✓	✓	—
Ron	✓	✓	—	✓

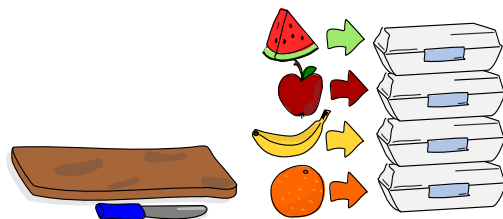
Metti le frutta nelle scatole in modo che tutti i castori prendano una scatola contenente un frutto che gli piace.





Soluzione

C'è solo un modo per distribuire la frutta in modo che ognuno abbia la garanzia di avere qualcosa che gli piace:



A papà piacciono solo le arance ed è l'ultimo a partire. Quindi l'arancia va nella scatola in basso. Ron va per primo, secondo o terzo. Mamma va prima di Dorie, quindi l'ordine è chiaro se si sa quando Ron lascia la tana.

Sono possibili le seguenti tre sequenze:

- | | | | |
|----|-------|-------|-------|
| 1. | Mamma | Mamma | Ron |
| 2. | Dorie | Ron | Mamma |
| 3. | Ron | Dorie | Dorie |
| 4. | Papà | Papà | Papà |

Quindi mamma, Dorie e Ron potrebbero partire per secondi. La seconda scatola deve quindi contenere un frutto che piaccia a tutti e tre, e solo la mela soddisfa questo requisito. Questo lascia solo banane e anguria per la scatola superiore. Dato che a mamma non piacciono le banane, qui dobbiamo assegnare l'anguria. Questo lascia la banana per la terza scatola.

Questa è l'informatica!

Il corretto sequenziamento delle operazioni è molto rilevante in molte aree dell'informatica: molti calcoli richiedono risultati intermedi che devono essere determinati prima di arrivare al risultato finale. Se i passi di calcolo vengono eseguiti su diversi computer, senza un'attenta pianificazione potrebbero presentarsi le cosiddette situazioni di *stallo*. Si tratta di situazioni in cui due o più computer aspettano l'un l'altro e in questo modo il programma non arriva mai alla fine. La sequenza sbagliata di solito porta ad errori (come il dispiacere dei castori per il frutto che hanno preso). Per esempio, se qualcosa deve essere calcolato con la formula $Z = (A + B) \times (A - B)$, questo può essere diviso nei seguenti passi di un programma:

Ingresso A

Ingresso B

Calcolare $X = A + B$

Calcolare $Y = A - B$

Calcolare $Z = X \times Y$

Tuttavia, se si cerca di eseguire il passo di calcolo $Z = X \times Y$, per esempio, prima di aver calcolato X , questo porta a un errore e il programma viene interrotto. Oppure viene usato un valore predefinito



per X , che di solito porta a un risultato sbagliato. Quindi, quando si programma, l'ordine in cui si eseguono le istruzioni è rilevante.

Parole chiave e siti web

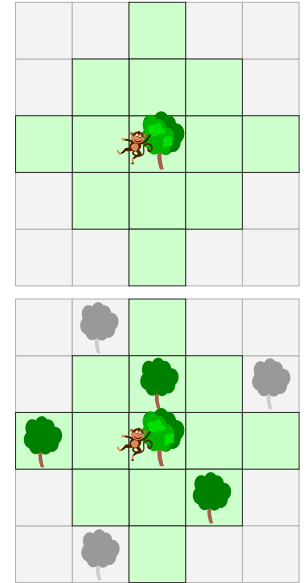
- Stallo: [https://it.wikipedia.org/wiki/Stallo_\(informatica\)](https://it.wikipedia.org/wiki/Stallo_(informatica))





25. La scimmia Coco

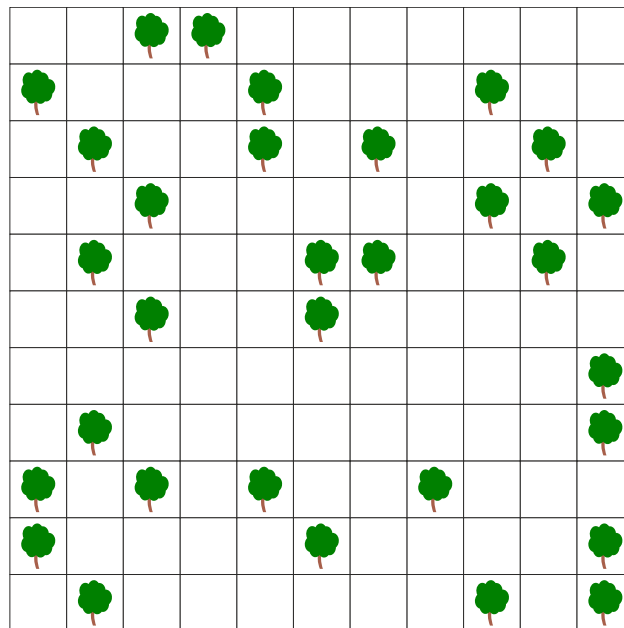
La scimmia arrampicatrice Coco può saltare da un albero e raggiungere qualsiasi luogo nell'area verde.



Nell'esempio seguente, Coco raggiunge gli alberi verdi con un solo salto. Con due salti, può raggiungere anche i due alberi grigi sopra, ma non l'albero grigio sotto.

Ci sono gruppi di alberi tra i quali Coco può muoversi a volontà con diversi salti senza mai toccare il suolo.

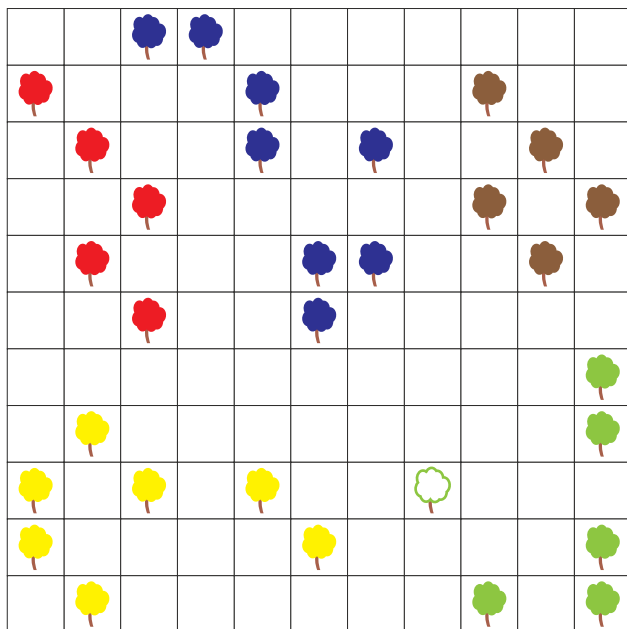
Segna tutti gli alberi del gruppo più grande in cui questo è possibile.





Soluzione

Nell'immagine qui sotto, due alberi sono dello stesso colore se Coco può passare da uno all'altro senza toccare il suolo.

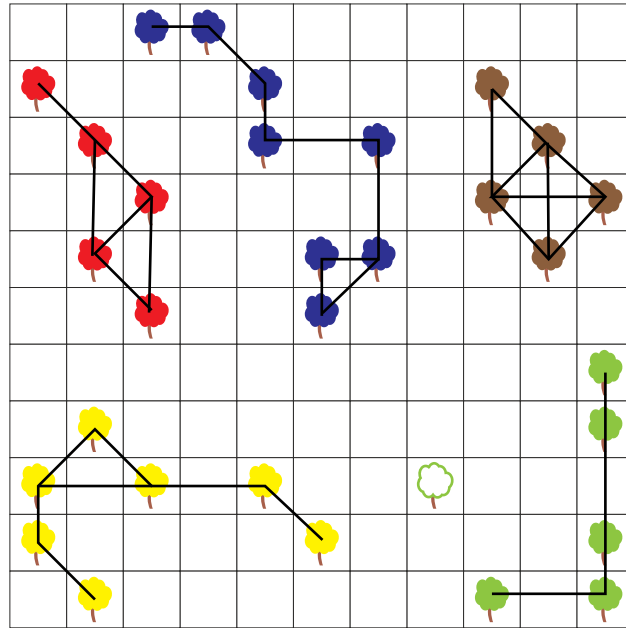


Vediamo che il gruppo di alberi blu con i suoi otto alberi è il più grande.

Questa è l'informatica!

Se Coco può saltare direttamente da un albero all'altro, sono virtualmente collegati. Possiamo rappresentare il salto come una linea tra gli alberi, come mostrato qui sotto. Così abbiamo un grafo con alberi come vertici e archi tra alberi collegati. Coco può saltare da un albero all'altro esattamente quando gli archi formano un percorso tra i due alberi.

Chiamiamo un gruppo di vertici *connessi* se sono tutti collegati da archi. Se non possiamo ingrandire tale gruppo senza perdere la connessione tra loro, allora parliamo di una *componente fortemente connessa*. Un grafico può essere chiaramente diviso in tale componenti fortemente connesse, qui sotto sono segnate con vari colori.



Una componente fortemente connessa può essere facilmente determinata iniziando da qualsiasi nodo e poi cercando tutti i nodi che possono essere raggiunti attraverso gli archi.

Parole chiave e siti web

- Grafo connesso, https://it.wikipedia.org/wiki/Grafo_connesso
- Componente fortemente connessa, https://it.wikipedia.org/wiki/Componente_fortemente_connessa





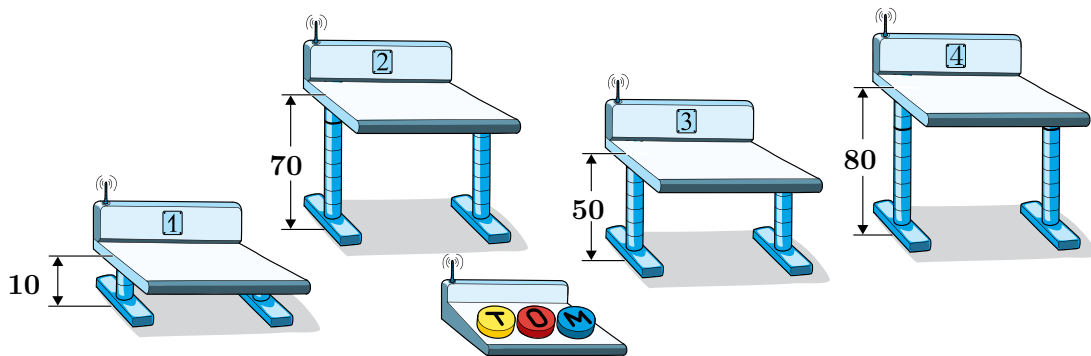
26. Maledette scrivanie

Un'aula ha scrivanie con altezza regolabile elettricamente. Per le lezioni, tutte le scrivanie devono essere impostate a un'altezza di 60 cm. L'altezza delle scrivanie può essere modificata con i pulsanti 🟡, 🔴 e 🔵 di un telecomando. Qualcuno ha giocato con il telecomando e lo ha riprogrammato. Ora i tre pulsanti funzionano come segue:

- 🟡 solleva le scrivanie 1, 2 e 3 di 10 cm.
- 🔴 abbassa le scrivanie 2, 3 e 4 di 10 cm.
- 🔵 solleva le scrivanie 1, 3 e 4 di 10 cm.

Queste azioni vengono eseguite ogni volta che il pulsante viene premuto.

Al momento, le altezze delle scrivanie 1, 2, 3 e 4 sono impostate a 10 cm, 70 cm, 50 cm e 80 cm:



Come si può impostare l'altezza di tutte le quattro scrivanie a 60 cm?

- A) Premere 4 × 🟡, 5 × 🔴 e 1 × 🔵.
 B) Premere 5 × 🟡, 1 × 🔴 e 0 × 🔵.
 C) Premere 3 × 🟡, 4 × 🔴 e 2 × 🔵.
 D) Premere 2 × 🟡, 4 × 🔴 e 6 × 🔵.



Soluzione

La risposta corretta è C) Premere $3 \times \text{🟡}$, $4 \times \text{🔴}$ e $2 \times \text{🟢}$. Sul telecomando, si nota che tutti i tre pulsanti cambiano l'altezza di 10 cm ogni volta che vengono premuti, cioè sempre della stessa misura. Due dei pulsanti sollevano le scrivanie (🟡 e 🟢) e solo un pulsante le abbassa (🔴). Inoltre, tutti i tre tasti cambiano l'altezza di tre scrivanie ciascuno, quindi c'è sempre una scrivania la cui altezza rimane invariata. Il tasto 🔴 non ha alcuna influenza sul tavolo 1, quindi non possiamo abbassare il tavolo 1.

La scrivania 1 è troppo bassa di 50 cm. Da questo concludiamo che dobbiamo premere il tasto 🟡 o 🟢 esattamente 5 volte (il numero di pressioni dei tasti su 🟡 e su 🟢 sommati insieme deve essere esattamente 5). Questo può essere espresso come un'equazione: $T + M = 5$. Con questo, possiamo escludere la soluzione D), perché lì si applica $T + M = 8$. Secondo la sequenza della soluzione D), la scrivania 1 avrebbe l'altezza di $10 + 20 + 60 = 90$ cm, cioè l'altezza iniziale di 10 cm più $2 * 10$ cm per 🟡 più $6 * 10$ cm per 🟢 .

La scrivania 2 è 10 cm troppo alta. 🟢 non ha influenza sulla scrivania 2, quindi la soluzione corretta è $T - O = -1$. Questo significa che possiamo escludere la soluzione B), perché alla fine il banco 2 avrebbe l'altezza $70 + 50 - 10 = 110$.

La scrivania 3 è troppo bassa di 10 cm, quindi $T - O + M = 1$. Questo significa che le soluzioni A) e B) possono essere escluse. Con A) l'altezza della scrivania 3 sarebbe ancora la stessa alla fine, cioè $50 + 40 - 50 + 10 = 50$ cm; con B) l'altezza della scrivania 3 sarebbe $50 + 50 - 10 = 90$ cm alla fine. Ora tutte le soluzioni sono escluse tranne la soluzione C).

Tuttavia, dobbiamo ancora verificare se la soluzione C) dia anche l'altezza corretta per la scrivania 4. La scrivania 4 è 20 cm troppo alta e 🟡 non ha alcuna influenza sull'altezza di essa. Quindi 🔴 deve essere premuto due volte e per ogni pressione di 🟢 è necessaria un'ulteriore pressione di 🔴 . Con la sequenza di tasti della soluzione C), l'altezza della scrivania 4 finisce per essere $80 - 40 + 20 = 60$ cm.

Dato che avevamo già stabilito che la soluzione C) funziona per le scrivanie 1, 2 e 3, siamo ora certi che questa soluzione è quella giusta.

In alternativa, la soluzione può essere cercata con quattro equazioni lineari. Per ogni scrivania, si scrive con un'equazione definendo quali pulsanti cambiano l'altezza della scrivania e quale sia il cambiamento di altezza che stiamo cercando. Per esempio, l'altezza della scrivania 1 cambia solo con 🟡 e 🟢 e la regolazione dell'altezza desiderata è di 50 cm, che può essere ottenuta con 5 pressioni di tasti (perché sono 10 cm per pressione di tasti).

Poiché ci sono quattro banchi e tre chiavi, ci sono quattro equazioni lineari con tre incognite:

$$\begin{array}{l}
T + M = 5 \\
T - O = -1 \\
T - O + M = 1 \\
-O + M = -2
\end{array}$$

Se sottraiamo la terza equazione dalla prima, otteniamo $O = 4$. Inserendolo nella seconda equazione, otteniamo $T = 3$. Solo per $M = 2$ tutte le equazioni sono soddisfatte. Quindi è l'unica soluzione.



Questa è l'informatica!

Questo è un compito tipico del campo dell'*ottimizzazione discreta*, più precisamente della *programmazione lineare*. Questo compito è dato da un insieme di costrizioni. In questo caso speciale, possono essere tutte formulate come equazioni lineari. L'obiettivo è tipico dell'informatica: si cerca una sequenza di azioni che portano a un determinato obiettivo. Si potrebbe anche descrivere l'intero compito come la ricerca di un percorso in uno spazio quadridimensionale con tre azioni di movimento permesse, cioè dal punto (10, 70, 50, 80) al punto (60, 60, 60, 60). In questo compito c'è solo una soluzione, ma tali compiti hanno spesso molte soluzioni, il che permette l'ottimizzazione come obiettivo. Poi si cerca il minimo della funzione lineare $T + M + O$.

Parole chiave e siti web

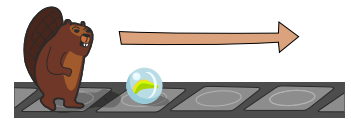
- Ottimizzazione lineare in numeri interi:
https://it.frwiki.wiki/wiki/Optimisation_linéaire_en_nombres_entiers





27. Nastro di biglia

Il castoro si sposta campo per campo da sinistra a destra su un nastro. Lungo il nastro può trovare una biglia.



Ogni volta che il castoro incontra un campo con una biglia e ha le mani libere, raccoglie la biglia e la porta con sé.



Al primo campo libero, il castoro posa la biglia.



Il castoro può portare solo una biglia alla volta e c'è spazio solo per una biglia su ogni quadrato.

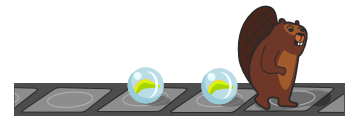
Se il castoro sta già trasportando una biglia quando raggiunge un campo con un'altra biglia, ...



... poi lui le passa accanto ...



... e pone la sua biglia sul prossimo campo libero.



Dopodiché, può raccogliere di nuovo la biglia successiva.

Il castoro sta di fronte a una zona del nastro dove ci sono tre biglie. Su quali quadrati si trovano le biglie quando il castoro ha attraversato l'area?



- A)
- B)
- C)
- D)
- E)

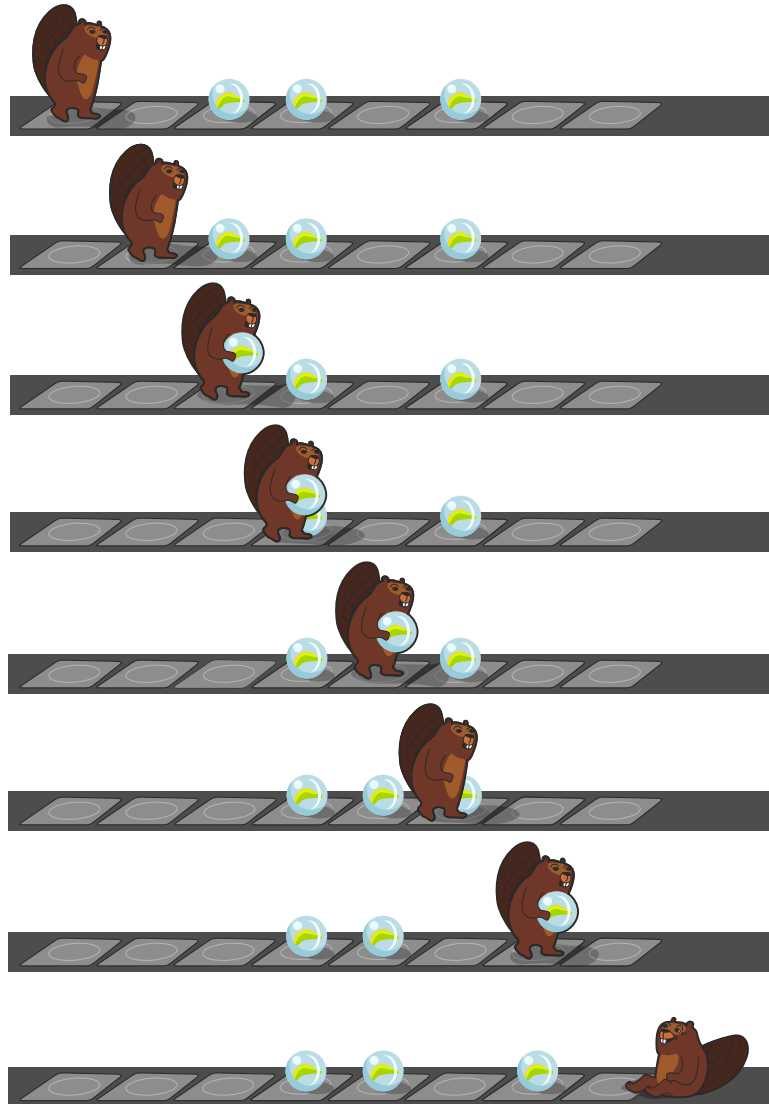


Soluzione

La risposta corretta è C.



La figura seguente mostra il processo:



Questa è l'informatica!

Nell'informatica, anche operazioni relativamente semplici portano spesso a risultati interessanti. Questo compito è un buon esempio. La procedura del castoro è un *algoritmo*. Si basa sul fatto che il castoro può assumere 2 stati diversi (portare una biglia o no) e che può trovare 2 tipi diversi di quadrati sul suo cammino (occupati e vuoti).



Per quanto semplice sia l'esempio in questo compito, contiene alcuni degli elementi essenziali di una *macchina di Turing*. Una macchina di Turing (dal nome del matematico Alan Turing) è un computer speciale con una struttura molto semplice. In teoria, una macchina di Turing può eseguire tutti gli algoritmi che un computer convenzionale può eseguire. In pratica, però, le macchine di Turing non sono usate come computer perché possiamo costruire computer più complicati ma molto più efficienti. Le macchine di Turing sono utili soprattutto in teoria. A causa della loro semplice struttura, è relativamente facile dimostrare affermazioni sulle macchine di Turing e se si può dimostrare che un compito non è risolvibile per le macchine di Turing, allora nessuno dei nostri computer può risolverlo.

Una macchina di Turing è composta da:

- Un *nastro* di qualsiasi lunghezza, composto da singoli *campi*. Ogni campo può contenere un *simbolo*. Nel nostro esempio, questi sono i quadrati su cui si muove il castoro.
- Un insieme finito di *simboli*. Spesso solo 0 e 1 sono usati come simboli. Nel nostro esempio, una biglia sta per 1 e un campo libero per 0.
- Una testina di lettura-scrittura che può muoversi in entrambe le direzioni sul nastro, leggendo i simboli sul nastro e anche scrivendo nuovi simboli. Nel nostro esempio, il castoro ha il ruolo della testa di lettura-scrittura.
- Un insieme finito di cosiddetti *stati*. Il comportamento della testina di lettura-scrittura può cambiare a seconda dello stato. Nel nostro caso, ci sono solo due stati, cioè «portare la biglia» e «non portare la biglia».
- Un insieme di regole: cosa succede, a seconda dello stato, quando un certo simbolo viene letto dal nastro? Le azioni possibili sono: un cambio di stato, la scrittura di un nuovo simbolo sul nastro e lo spostamento della testina di lettura-scrittura di un campo a sinistra o a destra.

Parole chiave e siti web

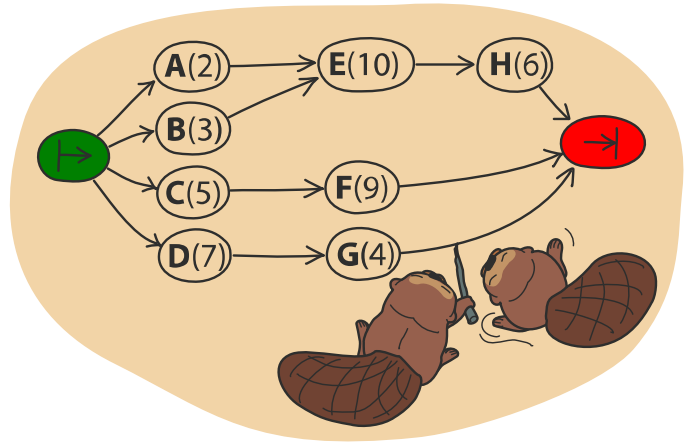
- Macchina di Turing: https://it.wikipedia.org/wiki/Macchina_di_Turing



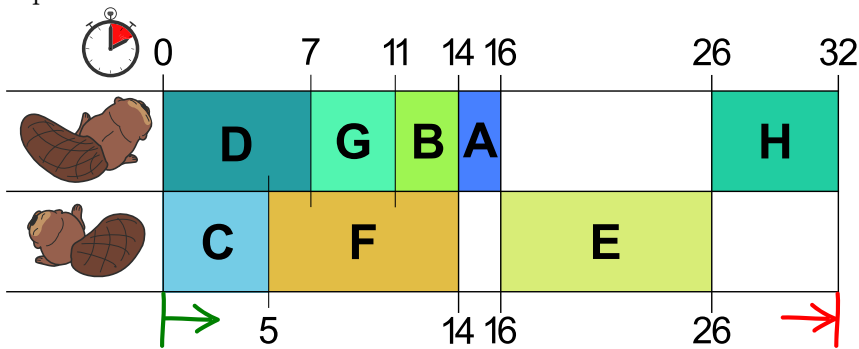


28. Piano di lavoro dei castori

La costruzione di una diga per castori può essere divisa in diversi sottocompiti (abbattimento degli alberi, rimozione dei rami, trasporto dei tronchi verso l'acqua, ecc.). L'immagine a destra mostra tutti gli 8 sottocompiti A, B, C, D, E, F, G, H, ognuno con il numero di ore necessarie per completarli. I sottocompiti non sono completamente indipendenti l'uno dall'altro: una freccia da X a Y significa che il sottocompito X deve essere completamente finito prima di iniziare il sottocompito Y.

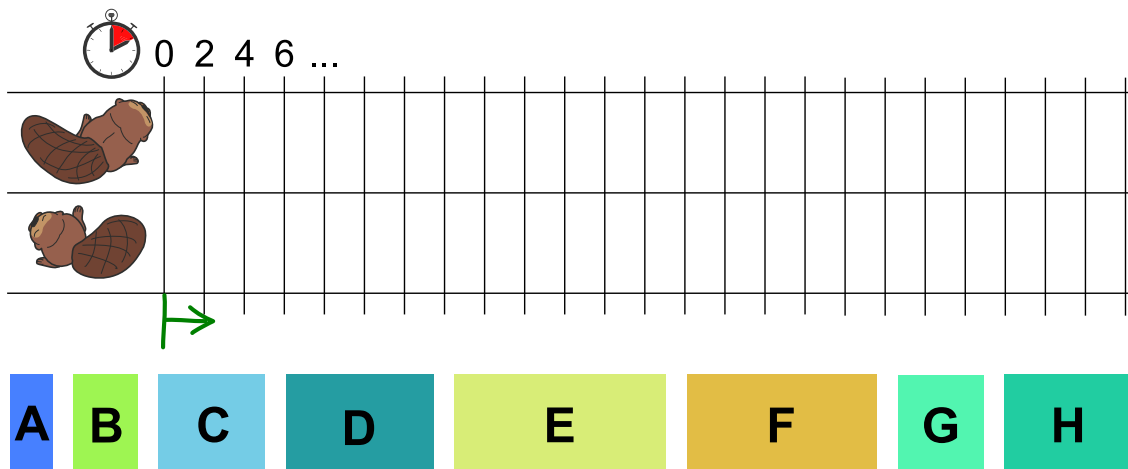


La castora Ulla vuole aiutare il castoro Otso a costruire la diga più velocemente. Dividono i sottocompiti tra di loro e creano il seguente piano di lavoro che soddisfa le dipendenze dell'immagine sopra.



Questo completerebbe la diga in 32 ore. Ma si può fare anche più velocemente!

Crea un piano di lavoro per finire la diga nel minor tempo possibile.

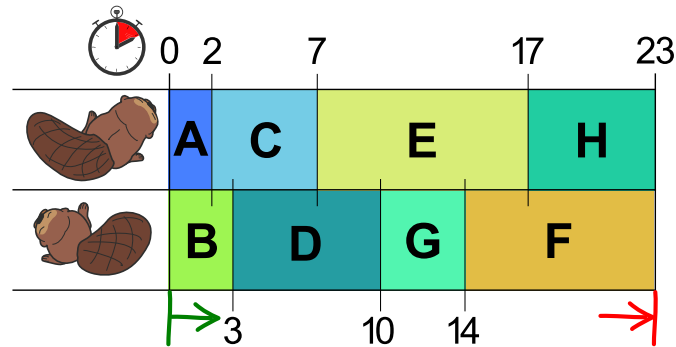




Soluzione

Nel piano di lavoro del compito, il primo castoro ha una lunga pausa (10 ore) e il secondo castoro ha tempo libero per un totale di 8 ore. I due finirebbero più velocemente se lavorassero continuamente.

Si arriva a un piano di lavoro complessivamente più veloce se ci si assicura che i due compiti più grandi E(10) e F(9) non siano eseguiti dallo stesso castoro. Ecco un possibile piano di lavoro che funziona con 23 ore. Non può essere più veloce, perché i due castori lavorano senza una pausa.



Questa è l'informatica!

Gli uomini sono impazienti quindi si chiede spesso che il lavoro sia fatto il più velocemente possibile. Se un lavoro può essere diviso (tra più persone, macchine o castori), allora la distribuzione dei sottocompiti tra i lavoratori gioca un ruolo importante nella velocità. Allo stesso modo, i computer devono spesso dividere i loro compiti computazionali in modo intelligente tra i diversi «core» del processore.

Per tali *problemi di scheduling* ci sono molte strategie diverse che sono state ben studiate dall'informatica. Il primo programma di questo compito è stato creato in modo tale che tra i sottocompiti in sospenso, quello con la durata più lunga è stato assegnato a un castoro attualmente disoccupato - in questo caso rivelatasi una cattiva strategia. Spesso funziona meglio se i sottocompiti brevi sono fatti per primi: La strategia *Shortest-Job-Next* (ingl. per «il sottocompito più breve per primo») minimizza anche il tempo medio di attesa al completamento per sottocompito.

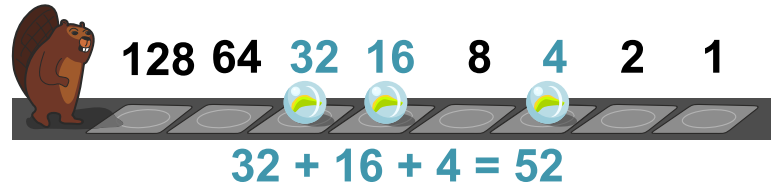
Parole chiave e siti web

- Scheduler: <https://it.wikipedia.org/wiki/Scheduler>
- Shortest-Job-Next: https://it.wikipedia.org/wiki/Shortest_job_first



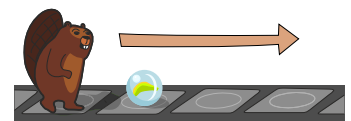
29. Numeri di biglia

I castori hanno un modo speciale di rappresentare i numeri.



I diversi campi hanno pesi diversi e una biglia sul campo determina che il valore viene preso in consegna. Nell'esempio qui sopra, viene mostrato il numero 52.

Il castoro si sposta campo per campo da sinistra a destra lungo un nastro. Le biglie possono trovarsi su alcuni campi del nastro.



Ogni volta che il castoro incontra un campo con una biglia e ha le mani libere, raccoglie la biglia e la porta con sé.

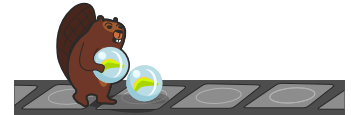


Al primo campo libero, il castoro posa la biglia.



Il castoro può portare solo una biglia alla volta e c'è spazio solo per una biglia su ogni campo.

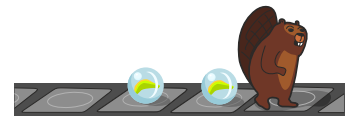
Se il castoro sta già trasportando una biglia quando raggiunge un campo con un'altra biglia, ...



... poi lui le passa accanto ...



... e pone la sua biglia sul prossimo campo libero.



Dopodiché, può raccogliere la biglia successiva.

Quale numero è rappresentato dalle biglie quando il castoro ha attraversato l'area?

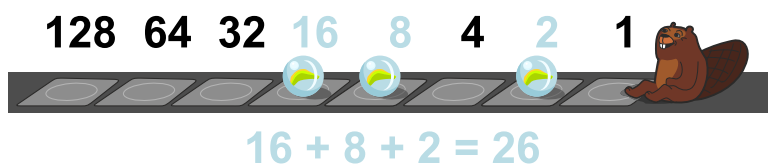


- A) 10
- B) 26
- C) 28
- D) 104

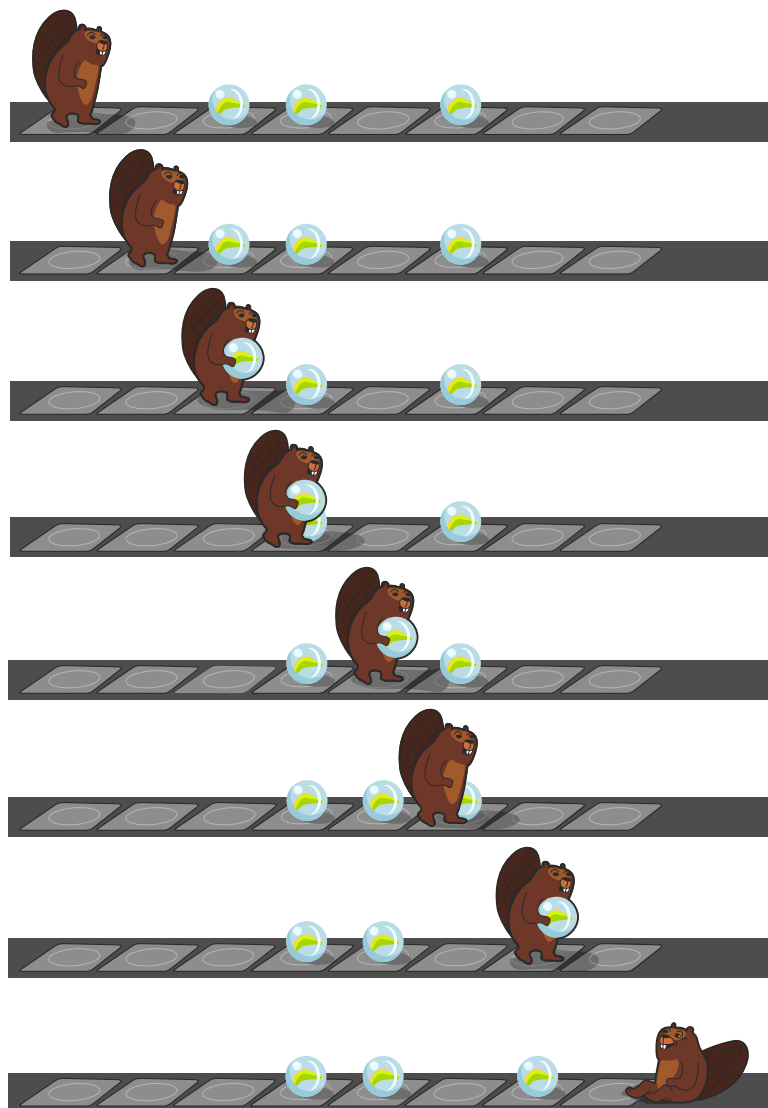


Soluzione

La risposta corretta è B) 26



La figura seguente mostra il processo:



Questa è l'informatica!

Nell'informatica, anche operazioni relativamente semplici portano spesso a risultati interessanti. Questo compito è un buon esempio. La procedura del castoro è un *algoritmo*. Si basa sul fatto che il castoro può assumere 2 stati diversi (portare una biglia o no) e che può trovare 2 tipi diversi di quadrati sul suo cammino (occupati e vuoti).



Nel complesso, dopo aver eseguito l'algoritmo, il risultato è esattamente come se ogni biglia sul nastro fosse stata spostata una a destra. Nella rappresentazione numerica dei castori, questo corrisponde a una divisione per 2. Se il castoro dovesse marciare attraverso il nastro da destra a sinistra, invece, il numero sarebbe moltiplicato per 2. Quando ogni cosa in una serie di zeri e di uno viene fatta avanzare della stessa quantità a sinistra o a destra, questo viene spesso chiamato un *bitshift*.

Per quanto semplice sia l'esempio in questo compito, contiene alcuni degli elementi essenziali di una *macchina di Turing*. Una macchina di Turing (dal nome del matematico Alan Turing) è un computer speciale con una struttura molto semplice. In principio, una macchina di Turing può eseguire tutti gli algoritmi che un computer convenzionale può eseguire. In pratica, però, le macchine di Turing non sono usate come computer perché possiamo costruire computer più complicati ma molto più efficienti. Le macchine di Turing sono utili soprattutto in teoria. A causa della loro semplice struttura, è relativamente facile dimostrare affermazioni sulle macchine di Turing. E se si può dimostrare che un compito non è risolvibile per le macchine di Turing, allora nessuno dei nostri computer può risolverlo.

Una macchina di Turing è composta da:

- Un *nastro* di qualsiasi lunghezza, composto da singoli *campi*. Ogni campo può contenere un *simbolo*. Nel nostro esempio, questi sono i quadrati su cui si muove il castoro.
- Un insieme finito di *simboli*. Spesso solo 0 e 1 sono usati come simboli. Nel nostro esempio, una biglia sta per 1 e un campo libero per 0.
- Una testina di lettura-scrittura che può muoversi in entrambe le direzioni sul nastro, leggendo i simboli sul nastro e anche scrivendo nuovi simboli. Nel nostro esempio, il castoro ha il ruolo della testa di lettura-scrittura.
- Un insieme finito di cosiddetti *stati*. Il comportamento della testina di lettura-scrittura può cambiare a seconda dello stato. Nel nostro caso, ci sono solo due stati, cioè «portare la biglia» e «non portare la biglia».
- Un insieme di regole: Cosa succede, a seconda dello stato, quando un certo simbolo viene letto dal nastro? Le azioni possibili sono: un cambio di stato, la scrittura di un nuovo simbolo sul nastro e lo spostamento della testina di lettura-scrittura di un campo a sinistra o a destra.

Parole chiave e siti web

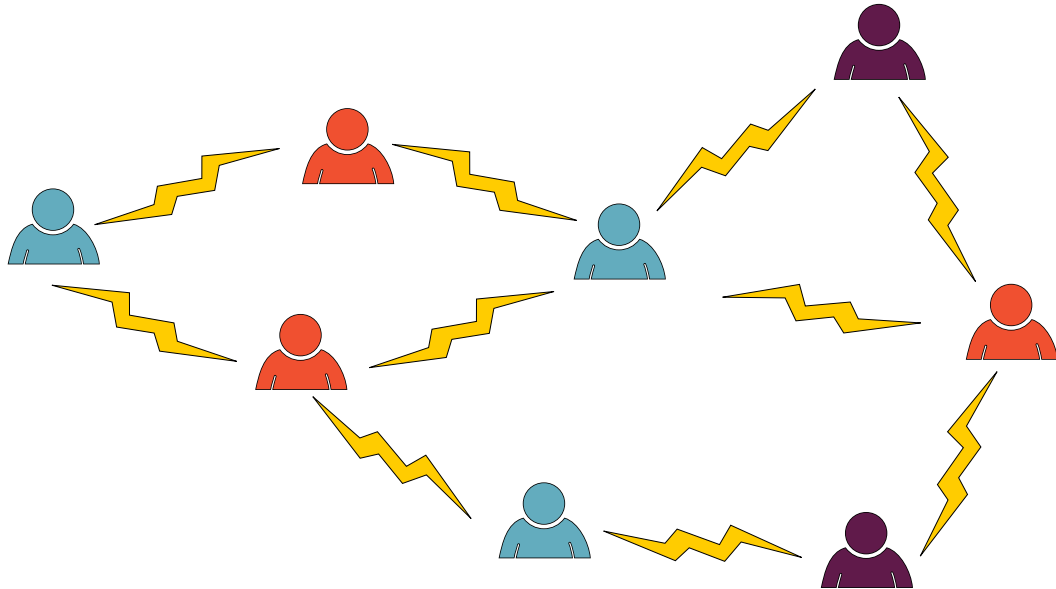
- Macchina di Turing: https://it.wikipedia.org/wiki/Macchina_di_Turing
- Operazione bit a bit: https://it.wikipedia.org/wiki/Operazione_bit_a_bit





30. Lavoro di gruppo

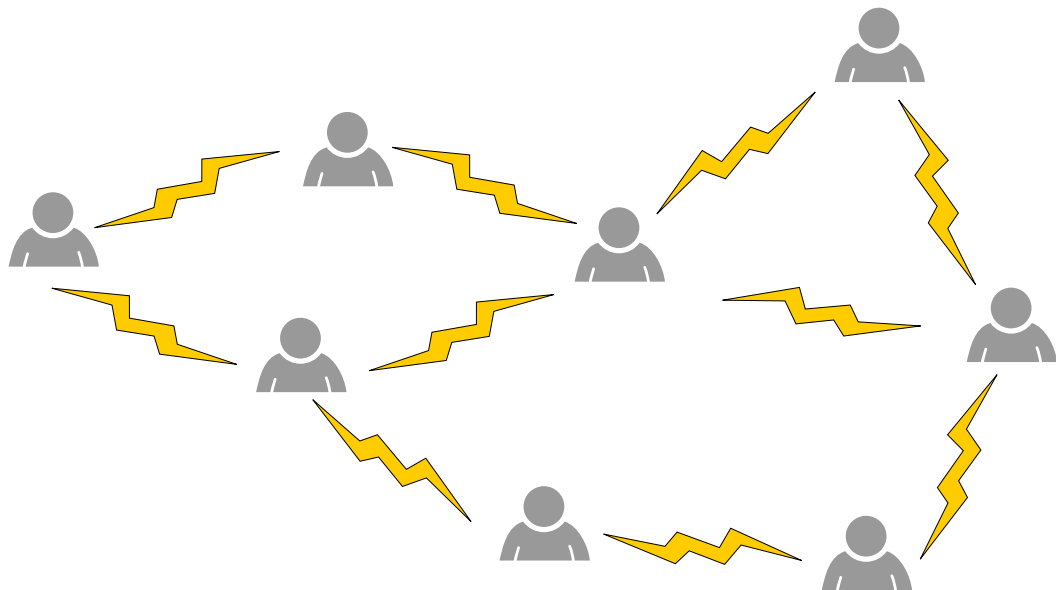
Per un progetto devi dividere otto persone in gruppi. C'è un fulmine tra due persone se non vogliono lavorare insieme. Quindi è meglio non metterli nello stesso gruppo.



Con le antipatie dell'esempio sopra, è possibile una divisione in tre gruppi (rosso, blu, viola). Questo perché non c'è nessun fulmine tra due persone dello stesso colore.

Se si convincono le due persone giuste a collaborare (cioè a togliere un fulmine), allora anche una divisione in due soli gruppi (solo due colori) è possibile.

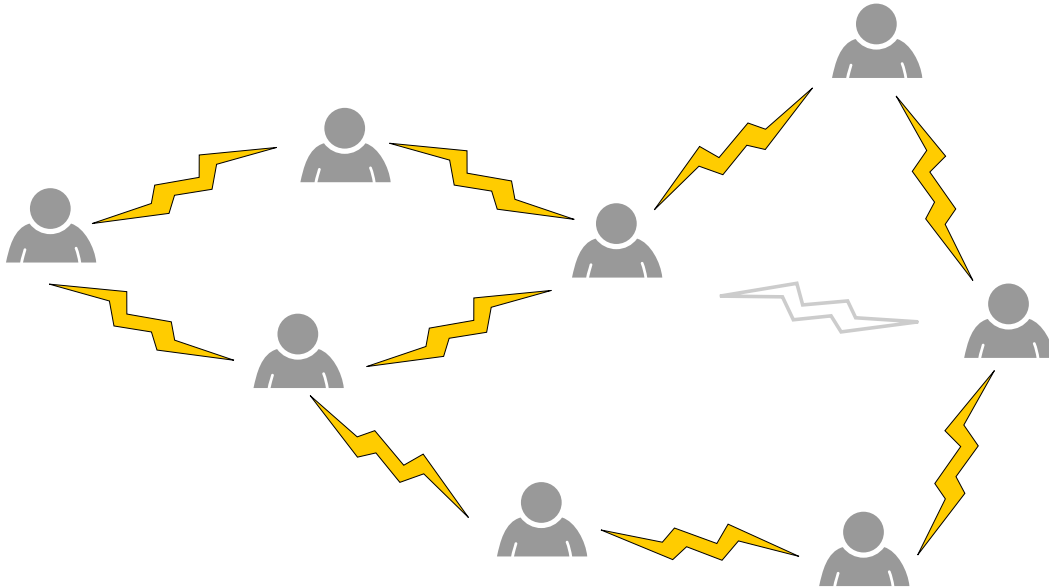
Rimuovi il fulmine giusto.





Soluzione

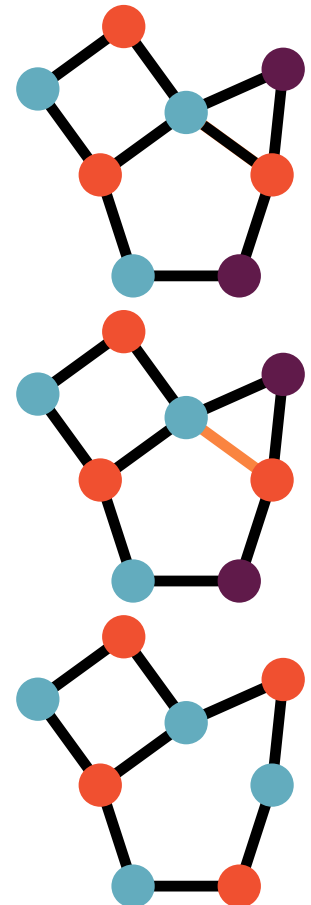
La soluzione corretta è:



Rappresentiamo la situazione un po' più astrattamente come un cosiddetto *grafo*, con le persone come *vertici* (nodi) e i fulmini come *archi* (bordi).

L'unica opzione possibile è l'arco marcato in arancione.

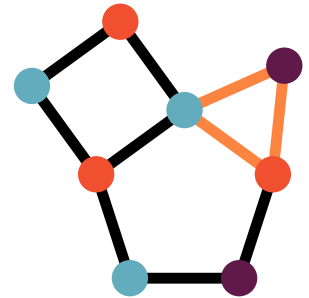
Dopo aver cancellato questo arco, possiamo colorare i nodi con due colori.



Ogni colore rappresenta un gruppo. Vediamo che da nessuna parte due persone dello stesso gruppo si rifiutano di cooperare: i vertici vicini hanno ovunque colori diversi.

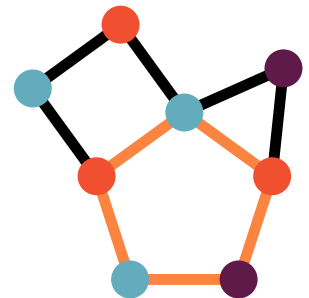


Per vedere che la cancellazione di questo arco è l'unica opzione possibile, guardiamo prima il triangolo segnato in arancione.



Se un arco esterno a questo triangolo viene cancellato, abbiamo ancora bisogno di tre colori solo per i tre vertici del triangolo.

Guardiamo ora il pentagono segnato in arancione:



Se un arco esterno a questo pentagono viene cancellato, rimane intatto ed è impossibile colorare il pentagono con due soli colori: se attraversiamo il pentagono in senso orario, dobbiamo alternare i due colori. Tuttavia, quando raggiungiamo l'ultimo vertice, questo ha lo stesso colore del primo, perché il numero di vertici nel pentagono è dispari, come nel caso del triangolo.

L'unica soluzione possibile è quindi quella di eliminare l'arco comune del triangolo e del pentagono.

Questa è l'informatica!

Molti problemi che si incontrano nella vita quotidiana possono essere formulati come i cosiddetti *problemi di colorazione dei grafi*. Nel nostro compito, i *vertici* di un *grafo* rappresentano le persone e un *arco* tra due persone mostra che si rifiutano di lavorare insieme in un gruppo. Se coloriamo i vertici con k colori, questo può essere interpretato come l'assegnazione delle persone a uno di k gruppi ciascuno. Tale colorazione è detta *permissibile* se per ogni coppia di due nodi direttamente connessi da un arco, questi due nodi hanno colori diversi. Nel nostro caso, quindi, una colorazione è *permessa* proprio quando tutti gli individui di ogni gruppo lavorano insieme. Un arco si dice *critico* se la sua cancellazione permette di colorare il grafico con meno colori di prima (in questo caso, tutti i vertici del grafico possono essere ricolorati). Per noi un arco è quindi critico esattamente quando, secondo la volontà delle due persone corrispondenti per la cooperazione, è possibile una divisione in meno gruppi.

Parole chiave e siti web

- Teoria dei grafi: https://it.wikipedia.org/wiki/Teoria_dei_grafi
- Colorazione dei grafi: https://it.wikipedia.org/wiki/Colorazione_dei_grafi

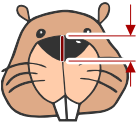
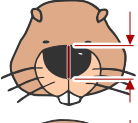



- Grafo critico: https://it.abcdef.wiki/wiki/Critical_graph



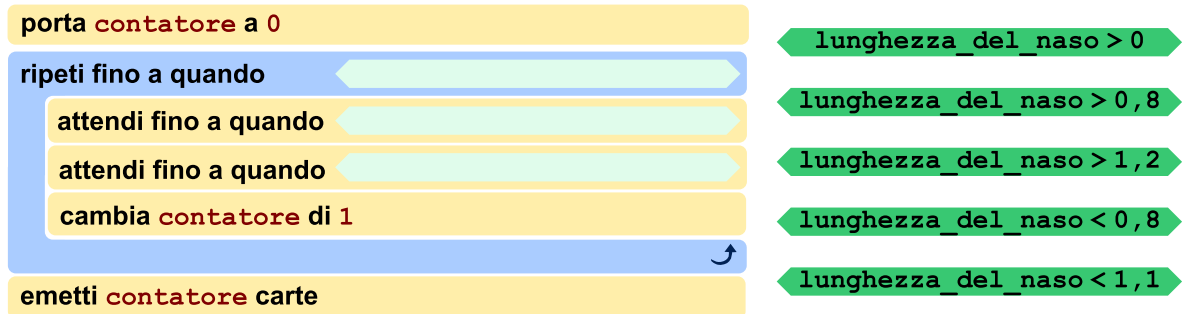
31. Contare con un cenno

Un nuovo distributore automatico di biglietti dovrebbe funzionare così: Un cliente fa un cenno con la testa - cioè abbassa la testa e poi guarda di nuovo dritto - tante volte quanto il numero di biglietti che vuole comprare. Dopodiché, il cliente alza la testa e il distributore automatico distribuisce i biglietti. La macchina ha una telecamera incorporata per questo scopo. Può riconoscere i nasi dei clienti e misurare costantemente la lunghezza dei loro nasi. Il programma di controllo del distributore automatico salva il risultato della misurazione attuale sotto il nome di «lunghezza del naso» e distingue le posizioni della testa dei clienti con l'aiuto di questa tabella:

Misura della telecamera	Valore lunghezza di naso	Postura della testa
	1	Il cliente sta guardando dritto davanti a sé.
	1,3	Il cliente ha abbassato la testa.
	0,7	Il cliente ha alzato la testa.

Il programma di controllo è quasi pronto - vedi sotto.

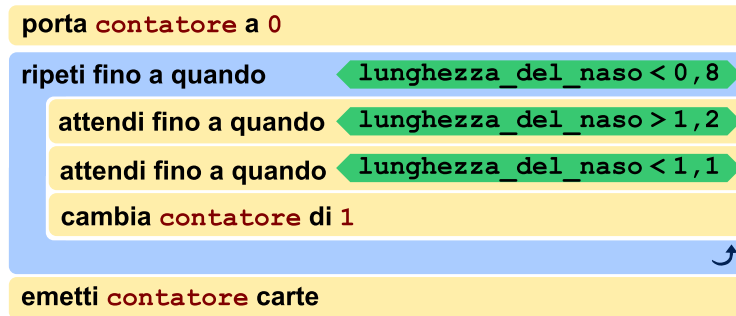
Completa il programma di controllo!





Soluzione

L'unica risposta corretta è:



La struttura del programma è predeterminata: C'è un'istruzione centrale che si ripete, spesso chiamata *ciclo*. L'ultima delle istruzioni ripetute in questo ciclo incrementa il contatore delle carte da emettere. Di conseguenza, le due istruzioni *attendi fino a quando* devono percepire un cenno del cliente: cioè, che il cliente prima abbassa la testa e poi guarda di nuovo dritto davanti a sé. Il valore memorizzato sotto la lunghezza del naso deve quindi essere prima circa 1,3 e poi di nuovo 1. Questo corrisponde alle condizioni **$lunghezza_del_naso > 1,2$** seguito da **<math>lunghezza_del_naso < 1,1</math>**.

Le istruzioni nel ciclo vengono ripetute finché il cliente non alza la testa: cioè viene misurato un valore significativamente più piccolo di 1. L'unica condizione corrispondente è **<math>lunghezza_del_naso < 0,8</math>**.

Forse hai notato che il programma non usa esattamente i valori della tabella. In pratica, non è possibile misurare continuamente, ma solo con una certa frequenza (per esempio 25 volte al secondo). Può succedere, per esempio, che il valore esatto di 1,0 per guardare dritto non sia misurato affatto, perché si misura 0,95 prima e 1,03 dopo.

Questa è l'informatica!

La *visione artificiale* (chiamata anche *machine vision* o *computer vision* in inglese) è un sottocampo dell'informatica in cui si sta svolgendo un'intensa ricerca. Sia le considerazioni teoriche che le applicazioni pratiche sono di grande importanza.

Un'applicazione significativa della visione artificiale è quella di dare alle persone disabili una migliore opportunità di interagire autonomamente con il loro ambiente. Per esempio, a seconda della gravità della disabilità, una persona può essere in grado di controllare solo pochi muscoli. Il fisico di fama mondiale Stephen Hawking (1942–2018) ha usato i movimenti dei suoi muscoli delle guance per controllare un programma di output vocale dopo che aveva perso il controllo della maggior parte dei suoi altri muscoli.

L'esempio concreto, tuttavia, potrebbe essere utilizzato anche per i musicisti: di solito usano entrambe le mani per azionare il loro strumento. Gli strumenti disponibili in commercio offrono un pedale per questo scopo. Tuttavia, alcuni musicisti, come gli organisti, usano anche i loro piedi per suonare e potrebbero, per esempio, girare automaticamente le note con un semplice cenno.



In contrasto con l'esempio di questo compito, dove i valori sono concreti e predeterminati, la visione artificiale è spesso combinata con il *machine learning*. Poi il programma viene addestrato su certi gesti mostrandogli molti esempi e controesempi per ogni gesto. In questo modo, il software costruisce una stima statistica di ciò che deve essere interpretato e come.

Parole chiave e siti web

- Visione artificiale, Machine Vision, Computer Vision:
https://it.wikipedia.org/wiki/Visione_artificiale
- Apprendimento automatico, Machine Learning:
https://it.wikipedia.org/wiki/Apprendimento_automatico
- https://it.wikipedia.org/wiki/Stephen_Hawking
- <https://it.wikipedia.org/wiki/Voltapagine>



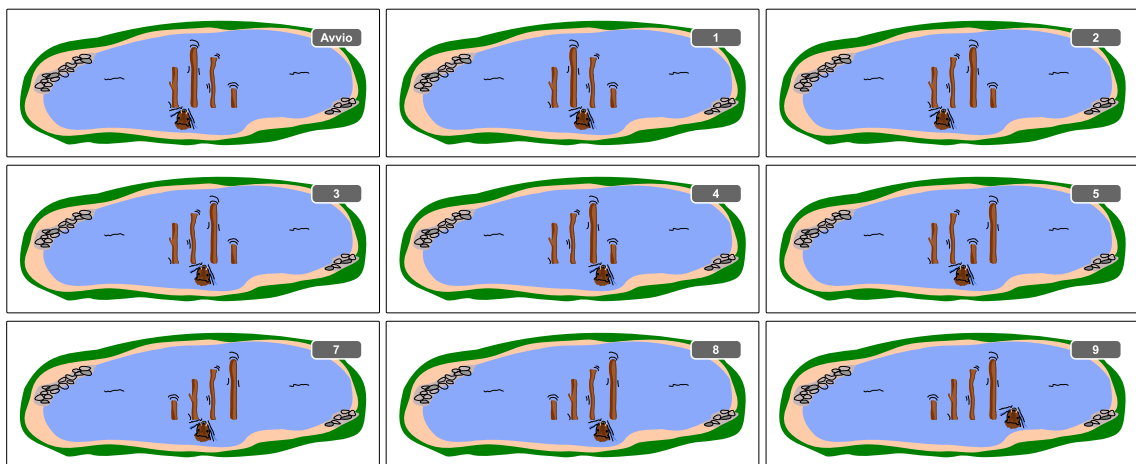


32. Beaver Sort

Il castoro Hamid ordina i tronchi nel lago. Da sinistra a destra, dovrebbero diventare sempre più lunghi.

- Hamid inizia tra i due tronchi all'estrema sinistra.
- Confronta sempre due tronchi vicini:
 - Se il tronco di destra è più lungo di quello di sinistra, Hamid si sposta di uno spazio a destra.
 - Se invece il tronco sinistro è più lungo, allora scambia i due tronchi. Dopo lo scambio, Hamid si sposta di uno spazio a destra se è nella posizione di partenza e uno a sinistra altrimenti.
- Hamid continua in questo modo finché non ha raggiunto la destra di tutti i tronchi. Poi tutti i tronchi sono ordinati correttamente.

L'esempio mostra come Hamid ordina 4 tronchi. Fa un totale di 9 confronti.



Il numero di confronti dipende da come sono disposti i tronchi all'inizio. Per 4 tronchi, Hamid deve fare almeno 3 confronti (se i tronchi sono già ordinati correttamente) e un massimo di 9 confronti (se i tronchi sono tutti ordinati esattamente nel senso contrario). Quindi, per 4 tronchi, Hamid deve fare i conti con 3 a 9 confronti.

Hamid ora deve ordinare 40 tronchi di diverse lunghezze. Quanti confronti deve aspettarsi?

- A) 0 a 20 confronti
- B) 3 a 40 confronti
- C) 39 a 120 confronti
- D) 39 a 1560 confronti



Soluzione

Per 40 tronchi, Hamid ha già bisogno di 39 confronti nel caso migliore, cioè quando tutti i tronchi sono già ordinati e quindi nuota sempre verso destra finché ha finito. Questo significa che le risposte A) e B) non possono essere corrette.

Nel peggiore dei casi, i tronchi sono ordinati esattamente al contrario all'inizio. Qui, il numero di molti confronti è più difficile da determinare. Osserviamo innanzitutto quanto segue: I tronchi d'albero a sinistra di Hamid sono sempre ordinati correttamente. Così, quando Hamid arriva a un tronco che è più corto di quelli k alla sua sinistra, prima lo scambia all'estrema sinistra, facendo k confronti. Poi torna indietro e nuota ancora una volta verso destra, facendo altri $k - 1$ confronti ma senza scambiare nulla. In totale, sono $k + (k - 1) = 2k - 1$ confronti. Questo accade in ognuna delle $n - 1$ posizioni tra due tronchi, e il numero di tronchi alla sinistra di Hamid è $\frac{n}{2}$ in media. Così Hamid fa un totale di $(n - 1) \cdot (2 \cdot \frac{n}{2} - 1) = (n - 1)^2$, cioè $39^2 = 1521$ confronti. Quindi la risposta C) con al massimo 120 confronti è sbagliata e la risposta D) è corretta.

Questa è l'informatica!

L'ordinamento degli elementi è un compito classico dell'informatica. Un ordinamento efficiente fa risparmiare molto tempo. L'algoritmo di ordinamento qui presentato si chiama *Gnome Sort* ed è stato originariamente presentato dall'informatico iraniano Hamid Sarbazi-Azad come *Stupid Sort*. Più tardi, l'informatico olandese Dick Grune ha chiamato questo metodo di ordinamento Gnome Sort, con l'idea che uno gnomo da giardino (ingl. *garden gnome*) potesse ordinare i vasi di fiori in questo modo.

L'analisi del tempo di esecuzione degli algoritmi (cioè scoprire quanto tempo un algoritmo può impiegare fino al suo completamento) è molto importante nell'informatica. Ci si chiede spesso quale sia il tempo di esecuzione nel caso migliore, nel caso medio e nel caso peggiore, a seconda della variabile di ingresso n . (Nel nostro caso, prendiamo il numero di tronchi d'albero per questo.) Per mantenere il confronto tra diversi tempi di esecuzione il più semplice possibile, di solito ci si accontenta di specificarli solo approssimativamente. Nel caso migliore, Gnome Sort ha bisogno solo di un tempo di esecuzione lineare - si scrive $\mathcal{O}(n)$ -, nel caso medio un tempo di esecuzione quadratico - si scrive $\mathcal{O}(n^2)$ - e nel caso peggiore anche un tempo di esecuzione quadratico - si scrive ancora $\mathcal{O}(n^2)$.

Parole chiave e siti web

- Gnome Sort: https://it.wikipedia.org/wiki/Gnome_sort
- Algoritmo di ordinamento: https://it.wikipedia.org/wiki/Algoritmo_di_ordinamento
- Analisi di tempo di esecuzione
- O-grande: <https://it.wikipedia.org/wiki/O-grande>

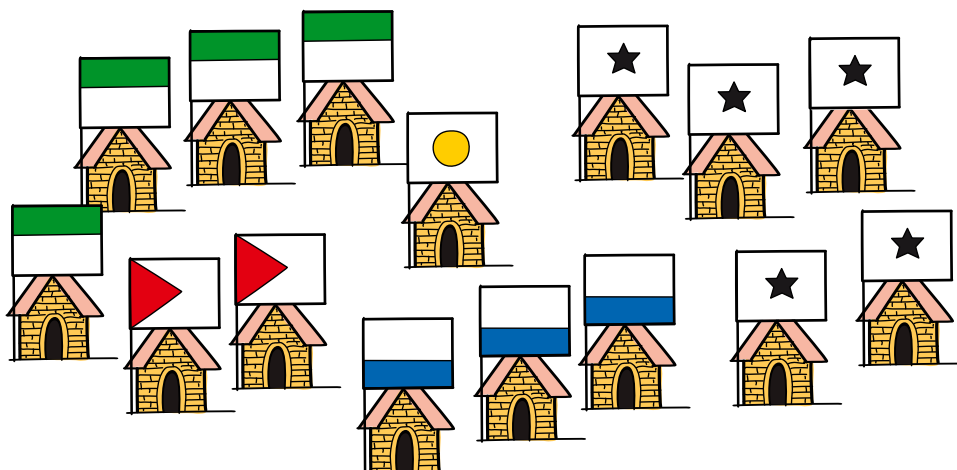


33. Clan di Castoria

Cinque clan ostili vivono a Castoria, ogni clan occupa alcune case come puoi vedere nell'immagine. Si chiamano Mac Intosh, Apfler, Mac Rosoft, Androidiani e Freidosiani. Dato che i tempi sono stati pacifici per molto tempo, decidono di eseguire un rituale di unificazione. Le regole per questo rituale sono le seguenti:

- Solo due clan possono unirsi alla volta.
- In ogni casa dei clan che si uniscono, si tiene una celebrazione di una settimana per suggellare il patto. La durata dell'unione in settimane è quindi uguale al numero di case dei due clan.
- Dopo questo tempo, i due clan sono solo un clan. Ora l'unione dei clan può continuare.

I clan decidono di unirsi nel più breve tempo possibile. Questo può essere fatto solo pianificando attentamente l'ordine di unificazione.



Qual'è il numero minimo di settimane che ci vogliono prima che tutti i clan siano uniti?

- A) 15 settimane
- B) 33 settimane
- C) 35 settimane
- D) 50 settimane
- E) 120 settimane

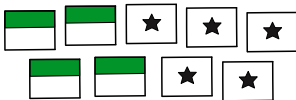



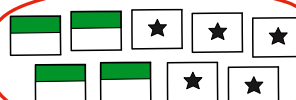



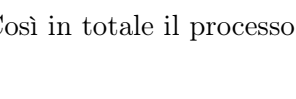



Soluzione

La risposta corretta è B) 33 settimane.

La strategia ottimale per minimizzare il numero totale di settimane necessarie per unire tutti i clan è quella di minimizzare il numero di case coinvolte. Le case dei primi clan uniti vengono poi prese in considerazione anche per le ulteriori unificazioni. Ha quindi senso unire prima i clan più piccoli, in modo che quelli più grandi non debbano partecipare troppo spesso alla cerimonia. Per ottenere ciò, i due clan con il minor numero di case dovrebbero essere uniti in ogni passo.

Questo processo passo dopo passo è mostrato nella tabella seguente. Per motivi di chiarezza, ometteremo i nomi dei clan e citeremo solo le dimensioni dei clan tra parentesi:

(1) e (2) sono uniti in un unico clan (3), il che richiede tre settimane.

(3) e (3) saranno uniti in (6) tra sei settimane.

Ora (4) e (5) saranno uniti in nove settimane per formare (9).

Infine, le rimanenti (6) e (9) sono unite. Questo richiede quindici settimane.

Così in totale il processo è completato dopo $15 + 9 + 6 + 3 = 33$ settimane, che è la risposta B).





Questa è l'informatica!

Questo è un *problema di ottimizzazione* - un compito per minimizzare o massimizzare un valore (in questo caso la durata dell'unificazione). Per questo, si usano spesso gli *algoritmi*, che, per esempio, sono ormai indispensabili per i processi di produzione rispettosi delle risorse e dell'ambiente.

In questo caso, il problema può anche essere risolto in modo ottimale con un semplice *algoritmo greedy*. Con questo algoritmo ad ogni passo si fa ciò che sembra essere meglio secondo una semplice idea. Nel nostro caso, l'idea è di unire prima i piccoli clan, perché questo richiede meno tempo.

Parole chiave e siti web

- Problema di ottimizzazione - https://it.wikipedia.org/wiki/Problema_di_ottimizzazione
- Algoritmo greedy - https://it.wikipedia.org/wiki/Algoritmo_greedy



A. Autori dei quesiti

 Daumilas Ardickas	 Yasemin Gülbahar
 Sarah Atkins	 Ezgi Arzu Güneş
 Michael Barot	 Mathias Hiron
 Liam Baumann	 Benjamin Hirsch
 Wilfried Baumann	 Juraž Hromkovič
 Linda Björk Bergsveinsdóttir	 Andrea Hrušecká
 Javier Bilbao	 Alisher Ikramov
 Lucia Budinská	 Tiberiu Iorgulescu
 Carmel Carroll	 Svetlana Jaksic
 Sarah Chan	 YongJu Jeon
 Anton Chukhnov	 Soojin Jun
 Kris Coolsaet	 Ungyeol Jung
 Valentina Dagiene	 Filiz Kalelioğlu
 Christian Datzko	 Martin Kandlhofer
 Susanne Datzko	 Ulrich Kiesmüller
 Janez Demšar	 Dong Yoon Kim
 Nora A. Escherle	 Jihye Kim
 Lidia Feklistova	 Vaidotas Kinčius
 Margarita Flores-Sicich	 V́ictor Koleszar
 Fabian Frei	 Regula Lacher
 Gerald Futschek	 Taina Lehtimäki
 Jens Gallenbacher	 Marielle Léonard
 Thomas Galler	 Angélica Herrera Loyo
 Christian Giang	 Tom Naughton
 Mark Edward M. Gonzales	 Mochammad Irfan Noviana
 Martin Guggisberg	 Graciela Oyhenard



 Gabriela Lourdes Rodríguez Parada

 Jean-Philippe Pellet

 Hannah Piper

 Jonatan Pipping


 Zsuzsa Pluhár

 Wolfgang Pohl

 Peter Rossmann

 Rodrigo Santamaría

 Eljakim Schrijvers

 Rosario Schunk

 Tomas Šiaulys

 Timur Sitdikov

 Bernadette Spieler

 Cuttle.org Team

 Ezra Templonuevo


 Ahto Truu

 Troy Vasiga


 Florentina Voboril

 Eslam Wageed

 Michael Weigend

 Kyra Willekes

 Hongjin Yeh

 Mija Zaļuksne



B. Sponsoring: concorso 2021

HASLERSTIFTUNG

<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.



hep/ haute
école
pédagogique
vaud

<http://www.hepl.ch/>

Haute école pédagogique du canton de Vaud

PH LUZERN
PÄDAGOGISCHE
HOCHSCHULE

<http://www.phlu.ch/>

Pädagogische Hochschule Luzern

n|w Fachhochschule
Nordwestschweiz

<https://www.fhnw.ch/de/die-fhnw/hochschulen/ph>

Pädagogische Hochschule FHNW

Scuola universitaria professionale
della Svizzera italiana

<http://www.supsi.ch/home/supsi.html>

La Scuola universitaria professionale della Svizzera italiana
(SUPSI)

SUPSI

PÄDAGOGISCHE
HOCHSCHULE
ZÜRICH

PH
ZH

<https://www.phzh.ch/>

Pädagogische Hochschule Zürich



C. Ulteriori offerte

010100110101011001001001
010000010010110101010011
010100110100100101000101
001011010101001101010011
010010010100100100100001

SSII

www.svia-ssie-ssii.ch
schweizerischervereinfürinformatikind
erausbildung//sociétésuissepourl'infor
matique dans l'enseignement//societasviz
zeraperl'informaticanell'insegnamento

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.