INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Quesiti e soluzioni 2019
9º e 10º anno scolastico

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

A cura di:
Lucio Negrini, Christian Datzko, Susanne Datzko, Juraj Hromkovič, Regula Lacher

01010011010101100100100101000001001011010101001101010011010010010100010100101010100110101001100100101001001010000100001

SSII

www.svia-ssie-ssii.ch
schweizerischervereinfürinformatikinderausbildung//sociétésuissepourl'informatiquedansl'enseignement//societàsvizzeraperl'informaticanell'insegnamento

Hanno collaborato al Castoro Informatico 2019

Christian Datzko, Susanne Datzko, Olivier Ens, Hanspeter Erni, Nora A. Escherle, Martin Guggisberg, Saskia Howald, Lucio Negrini, Gabriel Parriaux, Elsa Pellet, Jean-Philippe Pellet, Beat Trachsler.

Un particolare ringraziamento va a:
Juraj Hromkovič, Michelle Barnett, Michael Barot, Anna Laura John, Dennis Komm, Regula Lacher, Jacqueline Staub, Nicole Trachsler: ETHZ
Gabriel Thullen: Collège des Colombières
Valentina Dagienė: Bebras.org
Wolfgang Pohl, Hannes Endreß, Ulrich Kiesmüller, Kirsten Schlüter, Michael Weigend: Bundesweite Informatikwettbewerbe (BWINF), Germania
Chris Roffey: University of Oxford, Regno Unito
Carlo Bellettini, Violetta Lonati, Mattia Monga, Anna Morpurgo: ALaDDIn, Università degli Studi di Milano, Italia
Gerald Futschek, Wilfried Baumann, Florentina Voboril: Oesterreichische Computer Gesellschaft, Austria
Zsuzsa Pluhár: ELTE Informatikai Kar, Ungheria
Eljakim Schrijvers, Justina Dauksaite, Arne Heijenga, Dave Oostendorp, Andrea Schrijvers, Kyra Willekes, Saskia Zweerts: Cuttle.org, Paesi Bassi
Christoph Frei: Chragokyberneticks (Logo Castoro Informatico Svizzera)
Andrea Leu, Maggie Winter, Brigitte Manz-Brunner: Senarclens Leu + Partner

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 Veronica Ostini.

INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Il Castoro Informatico 2019 è stato organizzato dalla Società Svizzera per l'Informatica nell'Insegnamento SSII con il sostegno della fondazione Hasler.

HASLERSTIFTUNG

Nota: Tutti i link sono stati verificati l'01.11.2019. Questo quaderno è stato creato il 2 gennaio 2020 col sistema per la preparazione di testi LaTeX.

Premessa

Il concorso del "Castoro Informatico", presente già da diversi anni in molti paesi europei, ha l'obiettivo di destare l'interesse per l'informatica nei bambini e nei ragazzi. In Svizzera il concorso è organizzato in tedesco, francese e italiano dalla Società Svizzera per l'Informatica nell'Insegnamento (SSII), con il sostegno della fondazione Hasler nell'ambito del programma di promozione "FIT in IT".
Il Castoro Informatico è il partner svizzero del Concorso "Bebras International Contest on Informatics and Computer Fluency" (`https://www.bebras.org/`), situato in Lituania.
Il concorso si è tenuto per la prima volta in Svizzera nel 2010. Nel 2012 l'offerta è stata ampliata con la categoria del "Piccolo Castoro" (3º e 4º anno scolastico).
Il "Castoro Informatico" incoraggia gli alunni ad approfondire la conoscenza dell'informatica: esso vuole destare interesse per la materia e contribuire a eliminare le paure che sorgono nei suoi confronti. Il concorso non richiede alcuna conoscenza informatica pregressa, se non la capacità di "navigare" in internet poiché viene svolto online. Per rispondere alle domande sono necessari sia un pensiero logico e strutturato che la fantasia. I quesiti sono pensati in modo da incoraggiare l'utilizzo dell'informatica anche al di fuori del concorso.
Nel 2019 il Castoro Informatico della Svizzera è stato proposto a cinque differenti categorie d'età, suddivise in base all'anno scolastico:

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

Alla categoria del 3º e 4º anno scolastico sono stati assegnati 9 quesiti da risolvere, di cui 3 facili, 3 medi e 3 difficili. Alla categoria del 5º e 6º anno scolastico sono stati assegnati 12 quesiti, suddivisi in 4 facili, 4 medi e 4 difficili. Ogni altra categoria ha ricevuto invece 15 quesiti da risolvere, di cui 5 facili, 5 medi e 5 difficili.
Per ogni risposta corretta sono stati assegnati dei punti, mentre per ogni risposta sbagliata sono stati detratti. In caso di mancata risposta il punteggio è rimasto inalterato. Il numero di punti assegnati o detratti dipende dal grado di difficoltà del quesito:

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

Il sistema internazionale utilizzato per l'assegnazione dei punti limita l'eventualità che il partecipante possa ottenere buoni risultati scegliendo le risposte in modo casuale.
Ogni partecipante ha iniziato con un punteggio pari a 45 punti (risp., Piccolo Castoro: 27 punti, 5º e 6º anno scolastico: 36 punti).
Il punteggio massimo totalizzabile era dunque pari a 180 punti (risp., Piccolo castoro: 108 punti, 5º e 6º anno scolastico: 144 punti), mentre quello minimo era di 0 punti.
In molti quesiti le risposte possibili sono state distribuite sullo schermo con una sequenza casuale. Lo stesso quesito è stato proposto in più categorie d'età.

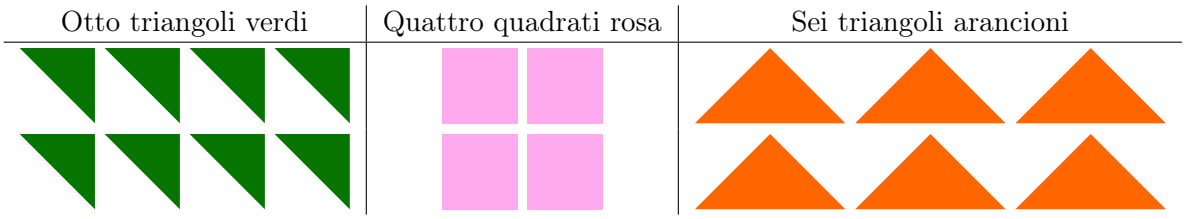
Per ulteriori informazioni:

SVIA-SSIE-SSII Società Svizzera per l'Informatica nell'Insegnamento
Castoro Informatico
Lucio Negrini
https://www.castoro-informatico.ch/it/kontaktieren/
https://www.castoro-informatico.ch/
https://www.facebook.com/informatikbiberch

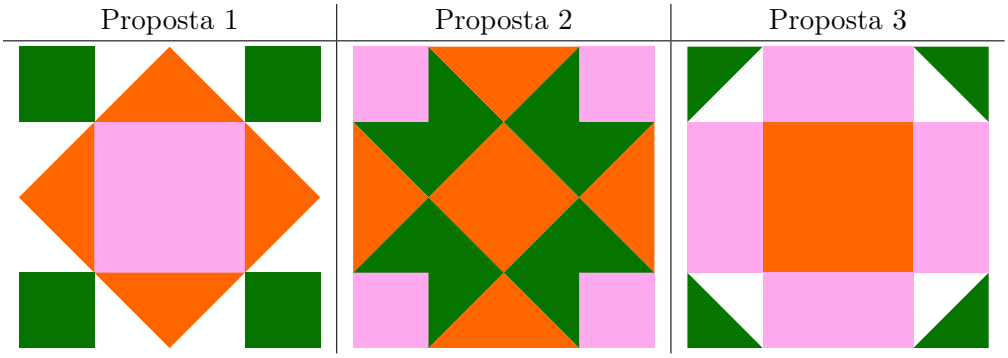
Indice

1. Rangoli

Il rangoli è una forma d'arte proveniente dall'India. Vengono posati dei motivi sul pavimento. Questi motivi sono per lo più simmetrici.
Per il suo rangoli Priya ha delle pietre di tre forme diverse: otto triangoli verdi, quattro quadrati rosa e sei triangoli arancioni. Le pietre dello stesso colore hanno anche la stessa grandezza:



Priya trova le seguenti proposte per dei rangoli su un sito internet (le superfici bianche rimangono libere):



Quali delle tre proposte per dei rangoli può posare Priya con le sue pietre?

A) Solo la proposta 1.

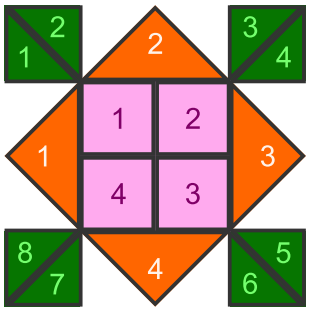
B) Solo la proposta 2.

C) Solo la proposta 3.

D) Tutte e tre le proposte.

Soluzione

Priya può posare A) solo la proposta 1 con le sue pietre.
La grafica seguente conta le pietre di diverso tipo nella proposta 1. Poiché ha bisogno al massimo di tante pietre di ogni tipo quante ne ha a sua disposizione, può posare la proposta 1:



Per la proposta 2 avrebbe bisogno in totale di dodici triangoli verdi, poiché ognuna delle quattro figure verdi nella proposta 2 necessita tre triangoli verdi. Priya ha però solo otto triangoli verdi a disposizione, quindi non può posare la proposta 2.
Per la proposta 3 avrebbe bisogno in totale di otto quadrati rosa, poiché ognuna delle quattro figure rosa nella proposta 3 necessita 2 quadrati rosa. Priya ha però solo quattro quadrati rosa a disposizione, quindi non può posare la proposta 3.
Siccome non può posare né la proposta 2 né la proposta 3, neanche la risposta D) può essere corretta

Questa è l'informatica!

Il *rangoli* è una forma d'arte che è tradizionalmente realizzata in India con riso e farina colorati, ma anche con sabbia colorata o fiori. I rangoli hanno principalmente uno scopo decorativo, ma sono anche legati a tradizioni regionali o di famiglia e agli auguri di buon auspicio. Anche alcune tradizioni religiose sono legate ai rangoli.
In questo problema bisognava scomporre una forma complessa in forme più piccole, che si potevano poi confrontare con le forme di base disponibili. Poi si sa quante di tutte le forme di base si hanno bisogno. Questo processo si chiama *decomposizione*, e viene usato spesso in informatica.
Confrontare le forme scomposte con le forme di base si chiama *Pattern Matching* (ingl. per *correlazione di motivi* o *confronto di motivi*). Nell'informatica il Pattern Matching è di grande importanza, dove non si cercano solo motivi grafici ma anche per esempio parole in testi o nomi di file nei sistemi di file, o anche per il confronto delle sequenze del genoma nella ricerca dei criminali.

Parole chiave e siti web

Decomposizione, Pattern Matching

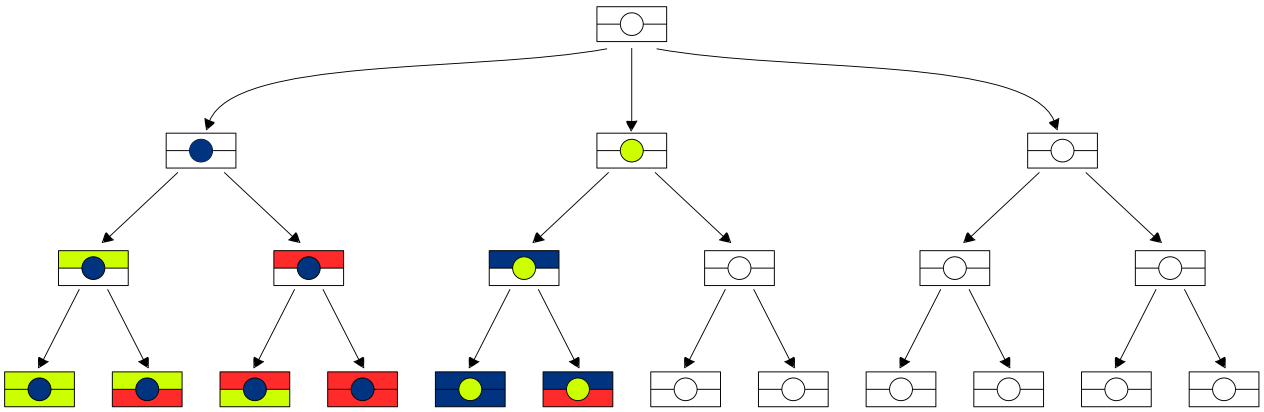
- `https://en.wikipedia.org/wiki/Rangoli`
- `https://it.wikipedia.org/wiki/Pattern_matching`
- `https://en.wikipedia.org/wiki/Decomposition_(computer_science)`

2. Bandiere variopinte

Il costruttore di barche dei castori costruisce barche eccellenti. Ogni castoro vuole avere una barca di quel tipo. Ma: come fanno a distinguere le barche se hanno tutte lo stesso aspetto?
I castori decidono di contrassegnare ogni barca con una bandiera. Una bandiera dei castori ha questo aspetto:

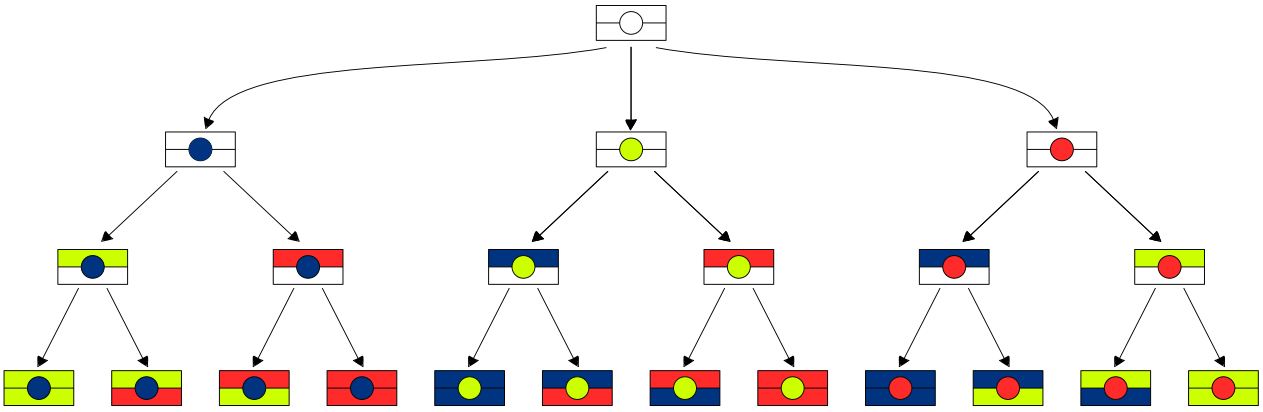
Si mettono d'accordo di usare tre colori differenti per le diverse superfici della bandiera: rosso, verde chiaro e blu scuro. È permesso che le strisce abbiano lo stesso colore, il cerchio in mezzo però deve avere un colore diverso da quello delle strisce:

Per non perdere la visione d'insieme, i castori disegnano un diagramma di tutte le possibili combinazioni di colori per le bandiere. Ma non hanno ancora finito.
Completa il diagramma per i castori. Ci sono più soluzioni corrette, è abbastanza dare una soluzione. Colora le superfici libere nel diagramma completamente.



Soluzione

Una soluzione possibile è:



Generalmente tutte le combinazioni di colori sono giuste, finché ...

- ... nella seconda fila il cerchio a destra è rosso, ...
- ... nella terza riga, a dipendenza del colore del cerchio, la striscia superiore ha un colore diverso per ogni colore del cerchio (l'ordine non importa), ...
- ... nella quarta riga, a dipendenza del colore del cerchio, la striscia inferiore ha un colore diverso dal colore del cerchio (l'ordine non importa).

Questa è l'informatica!

A volte bisogna risolvere dei problemi complicati. In questi casi aiuta enumerare tutte le soluzioni possibili. Soprattutto nell'informatica è importante sapere enumerare tutte le soluzioni possibili in modo efficiente.
In molti casi aiuta avere un metodo sistematico per enumerare, in modo da non dimenticare nessuna possibile soluzione o che una possibile soluzione viene considerata due volte. Le strutture dati, come l'*albero*, che usano i castori, aiutano a trovare sistematicamente tutte le soluzioni. In ogni fila vengono segnati vicino per una parte dell'oggetto (quindi per una superficie della bandiera) tutti i valori possibili (quindi tutti i colori permessi). In questo caso le bandiere superiori (ancora non riempite) vengono indicate come *radice*, e le bandiere riempite completamente sotto come *foglie*. Una diramazione viene chiamata *ramo*. Poiché tutti i rami corrispondono a tutti i valori possibili per l'area da riempire, si può essere sicuri che le foglie contengono tutte le soluzioni possibili.

Parole chiave e siti web

Albero

- `https://it.wikipedia.org/wiki/Albero_(informatica)`
- `https://it.wikipedia.org/wiki/Enumerazione_(matematica)`

3. Caratteri cinesi variopinti

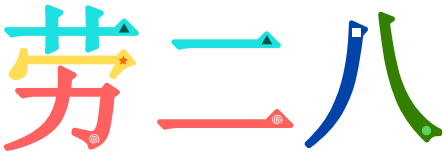
La struttura dei caratteri cinesi ci appare estranea. Per capire meglio la composizione di alcuni caratteri cinesi si può pensare a questo schema, nel quale si distinguono cinque parti, sopra , sotto , sinistra , destra e centro :

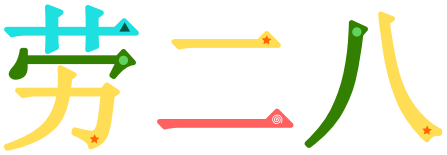
Queste parti possono essere disposte come quattro strutture:

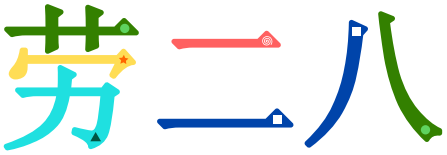
Struttura	sinistra-centro-destra	sinistra-destra	sopra-centro-sotto	sopra-sotto
Esempio di carattere	川	儿	三	吕
Esempio di analisi				

Quale analisi mostra la disposizione corretta secondo lo schema dei tre caratteri cinesi 劳, 二, *e* 八*?*

A) 

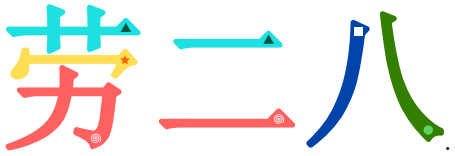
B) 

C) 

D) 

Soluzione

Il primo carattere 劳 corrisponde alla struttura sopra-centro-sotto, quindi il tratto superiore è azzurro, il tratto centrale è giallo e il tratto inferiore è rosa.
Il secondo carattere 二 corrisponde alla struttura sopra-sotto, quindi il tratto superiore è azzurro e quello inferiore è rosa.
Il terzo carattere 八 corrisponde alla struttura sinistra-destra, quindi il tratto a sinistra è blu scuro e il tratto a destra è verde.

Quindi la risposta corretta è la B)  .
Nella risposta A) il secondo carattere 二 viene analizzato correttamente, ma ad entrambi i caratteri 劳 e 八 vengono assegnati dei colori sbagliati: per 劳 il colore superiore è sbagliato, per 八 i due colori sono scambiati.
Nella risposta C) tutti i caratteri sono analizzati erroneamente. Nel primo carattere il colore centrale e quello inferiore sono stati scelti erroneamente, nel secondo carattere il colore superiore è stato scelto erroneamente e per il terzo carattere entrambi i colori sono stati scelti erroneamente.
Nella risposta D) il carattere 八 è stato analizzato correttamente, ma per 劳 il colore superiore e quello inferiore sono sbagliati e per 二 entrambi i colori sono stati scelti erroneamente.

Questa è l'informatica!

La scrittura cinese è composta da complessi caratteri compositi. Anche nelle varianti semplificate ci sono oltre 200 elementi di base (*radicali*), dai quali vengono assemblati i caratteri. Questi vengono scritti l'uno a fianco all'altro o uno sotto l'altro, in modo da formare effettivamente delle strutture come spiegato nel problema. In questo modo possono essere combinati migliaia di caratteri diversi. Se si devono imparare questi caratteri bisogna imparare la loro composizione. Per farlo vengono spesso usati dei colori. L'alfabeto latino usato da noi funziona diversamente: una *lettera* corrisponde ad un suono (con eccezioni come sc seguito da "i" o "e", che viene pronunciato [ʃ] e non [khe]).
Cosa ha a che fare questo con l'informatica? Da un lato questi segni devono poter essere rappresentati da un computer. Per questo ci sono diversi approcci, un approccio sfrutta i radicali descritti in questo problema. Dall'altro bisogna essere in grado di poter cercare delle parole, ad esempio nei dizionari o nelle enciclopedie. I radicali più comunemente usati al giorno d'oggi provengono da un dizionario elaborato dal 1710 al 1716 sotto l'imperatore Kangxi. È ordinato secondo il numero di tratti in ogni radicale.

Parole chiave e siti web

Caratteri cinesi

- `https://it.wikipedia.org/wiki/Radicali_(cinese)`
- `https://it.wikipedia.org/wiki/Caratteri_cinesi_semplificati`
- `https://en.wikipedia.org/wiki/Chinese_character_encoding`
- `https://en.wikipedia.org/wiki/Chinese_input_methods_for_computers`
- `https://it.wikipedia.org/wiki/Dizionario_di_Kangxi`

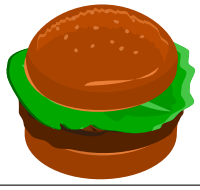
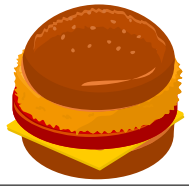
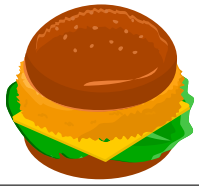
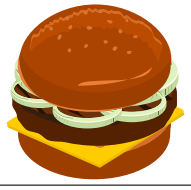
- https://it.wikipedia.org/wiki/Alfabeto_latino
- https://it.wikipedia.org/wiki/Digrammi_e_trigrammi_della_lingua_italiana

Le lettere cinesi sono:

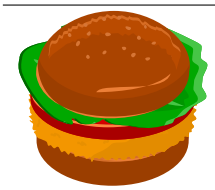
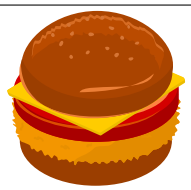
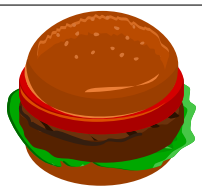
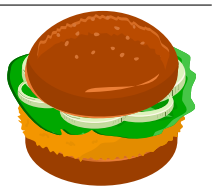
- 川: https://en.wikipedia.org/wiki/Radical_47
- 儿: https://en.wikipedia.org/wiki/Radical_10
- 吕: https://en.wikipedia.org/wiki/L%C3%BC_(surname)
- 二: https://en.wikipedia.org/wiki/Radical_7
- 三: https://en.wikipedia.org/wiki/3
- 八: https://en.wikipedia.org/wiki/Radical_12
- 劳: https://en.wiktionary.org/wiki/%E5%8A%B3

4. Ingredienti degli hamburger

BeaverBurger offre sei ingredienti (A, B, C, D, E e F) per i suoi hamburger fatti in casa. La tabella seguente mostra gli ingredienti per i quattro esempi di hamburger, dove gli ingredienti non sono ordinati per forza come negli esempi di hamburger:

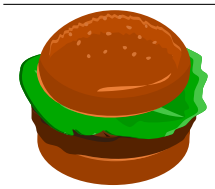
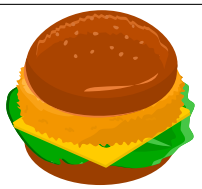
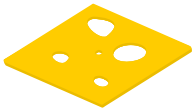
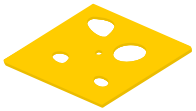
Burger				
Ingredienti	C, F	A, B, E	B, E, F	B, C, D

Quale hamburger contiene gli ingredienti A, E e F?

A)	B)	C)	D)
			

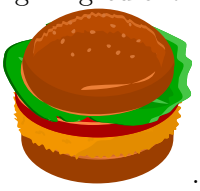
Soluzione

Per scoprire quale ingrediente è assegnato a quale lettera bisogna sempre confrontare due hamburger uno con l'altro:

Hamburger confrontati		Lettera in comune	Ingrediente in comune
		F	
		C	
		B	
		B (appena identificato)	
		E	

Due ingredienti si trovano solo in un hamburger. Siccome conosciamo già tutte le altre lettere possiamo quindi identificare gli ingredienti corrispondenti:

Hamurger particolare	Lettera particolare	Ingrediente particolare
	A	
	D	

Quindi l'hamburger che cerchiamo con gli ingredienti A, E e F, composto dagli ingredienti  ,  e  è l'hamburger della risposta A)  .

Questa è l'informatica!

L'*inferenza logica* è la base per molti ragionamenti, anche nell'informatica. Per risolvere questo problema bisogna applicarla in modo intensivo: attraverso il confronto degli hamburger con gli stessi ingredienti si possono dedurre informazioni che prima erano sconosciute (quale ingrediente corrisponde a quale lettera).
Nel caso di questo problema gli ingredienti in comune di due hamburger corrispondono all'*intersezione* degli ingredienti dei due hamburger. Contiene solo gli ingredienti che sono contenuti in entrambi gli hamburger. Per il primo confronto si scriverebbe quindi $\{C, F\} \cap \{B, E, F\} = \{F\}$. Il contrario dell'intersezione sarebbe d'altronde l'*unione* $\{C, F\} \cup \{B, E, F\} = \{B, C, E, F\}$, contiene tutti gli elementi che sono contenuti almeno in un hamburger.
Per scoprire gli ingredienti che sono contenuti solo in un hamburger si può usare la *differenza*. Contiene solo gli ingredienti del primo insieme che non sono contenuti nel secondo insieme. Per il primo hamburger particolare si potrebbe ad esempio scrivere: $\{A, B, E\} \setminus (\{C, F\} \cup \{B, E, F\} \cup \{B, C, D\}) = \{A, B, E\} \setminus \{B, C, D, E, F\} = \{A\}$.
La teoria degli insiemi si conosce magari dalle lezioni di matematica. Nell'informatica viene usata ad esempio nelle banche dati. Ma si può anche convertire la teoria degli insiemi $1 : 1$ in logica, chiamata anche algebra di Boole, e anche essa viene usata in molti ambiti dell'informatica.

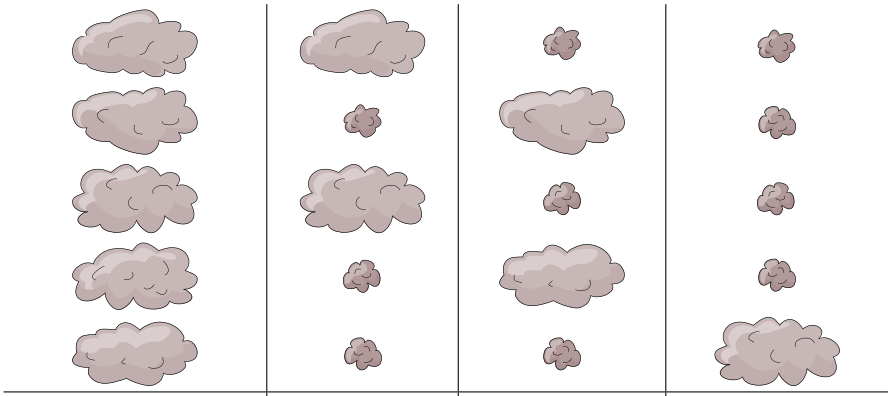
Parole chiave e siti web

Inferenza logica, Teoria degli insiemi, Logica

- `https://it.wikipedia.org/wiki/Inferenza`
- `https://it.wikipedia.org/wiki/Insieme`
- `https://it.wikipedia.org/wiki/Algebra_di_Boole`

5. Segnali di fumo

Un castoro si siede sempre sulla montagna e osserva il tempo. Trasmette ai castori nella valle come sarà il tempo. Usa segnali di fumo che consistono in cinque nuvole di fumo successive. Una nuvola di fumo o è piccola o è grande. I castori hanno concordato i seguenti segnali di fumo:



In un giorno ventoso i castori nella valle non riescono a riconoscere bene le nuvole di fumo. Interpretano il messaggio seguente:

Siccome questo non corrisponde a nessuno dei messaggi concordati, suppongono che hanno interpretato male una nuvola di fumo: una nuvola di fumo piccola dovrebbe in realtà essere grande o una nuvola di fumo grande dovrebbe in realtà essere piccola.
Se fosse stata interpretata male esattamente una nuvola di fumo, quale sarebbe il significato?

A) Sarà temporalesco.

B) Sarà piovoso.

C) Sarà nuvoloso.

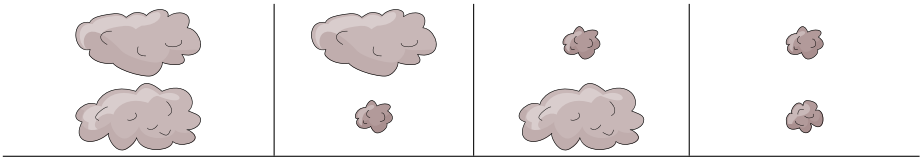
D) Sarà soleggiato.

Soluzione

Se esattamente una nuvola fosse stata interpretata male potrebbero risultare esattamente cinque segnali di fumo diversi. Interpretare diversamente la prima, la seconda, la quarta o la quinta nuvola di fumo non porta tuttavia a nessuno dei segnali di fumo concordati. Interpretare la terza nuvola di fumo come nuvola di fumo piccola risulta nella risposta corretta, il segnale di fumo C) "Sarà nuvoloso".
Si può anche confrontare il segnale di fumo interpretato con i quattro segnali di fumo concordati e guardare quante nuvole di fumo sono diverse. Per il segnale di fumo "Sarà temporalesco" sono due (quella più in alto e quella più in basso), per il segnale di fumo "Sarà piovoso" sono tre nuvole di fumo (le due più in alto e la seconda più in basso), per il segnale "Sarà nuvoloso" è una nuvola di fumo (quella in mezzo, per questo è la soluzione giusta, come descritto sopra) e per il segnale di fumo "Sarà soleggiato" sono quattro nuvole di fumo (tutte tranne quella più in alto).

Questa è l'informatica!

Quando si vuole trasmettere un messaggio si vuole che arrivi correttamente al destinatario. In questo problema il messaggio viene trasmesso con l'aiuto di nuvole di fumo grandi e piccole. Nel caso generale si parla di *simboli*. È quindi sensato scegliere una sequenza di simboli in modo che il messaggio trasmesso sia comprensibile anche se viene danneggiato durante la trasmissione. Ciò si può ottenere comunicando più informazioni di quelle strettamente necessarie. Queste informazioni aggiuntive vengono chiamate *ridondanti*.
Quando si riesce a ricostruire l'informazione danneggiata con al massimo n errori, si parla di codificazione n-autoregolante. Rappresentare messaggi come sequenze di simboli in modo da potere ricostruire il messaggio, anche quando la sua rappresentazione viene danneggiata durante la trasmissione, è un compito tipico degli informatici. Ad esempio rendono possibile riprodurre correttamente musica da dei CD o video da dei DVD anche quando nella trasmissione si presentano alcuni errori. D'altronde, per questo problema sarebbero abbastanza due nuvole di fumo per trasmettere i quattro diversi messaggi:



I castori, però, usano cinque nuvole di fumo. Ciò permette loro di capire i messaggi correttamente anche in casi dove due nuvole di fumo, o addirittura in certi casi tre nuvole di fumo, sono "illegibili". D'altronde, i castori hanno pensato ai messaggi in modo tale che ogni due messaggi si differenziano in almeno tre posizioni.

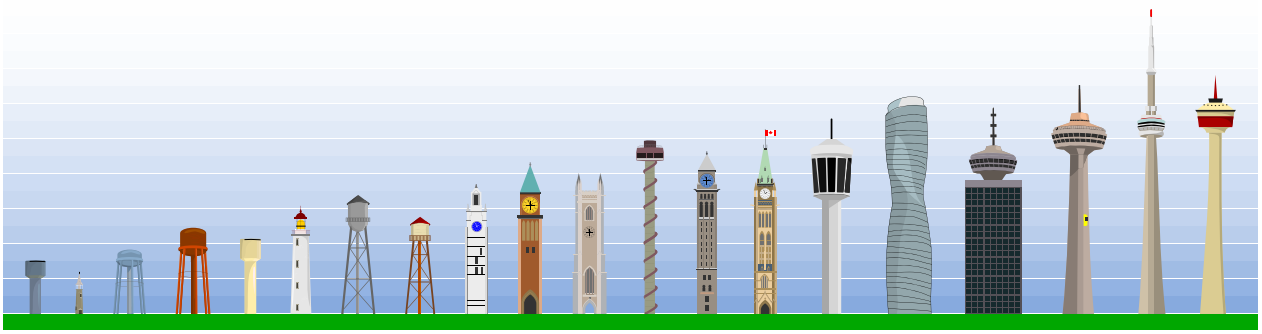
Parole chiave e siti web

Rilevazione e correzione di errore

- `https://it.wikipedia.org/wiki/Rilevazione_e_correzione_d'errore`

6. Torri speciali

Una torre è speciale quando tutte le torri alla sua sinistra sono più piccole e tutte le torri alla sua destra sono più grandi.



Quante torri nel grafico sopra sono speciali?

A) 4

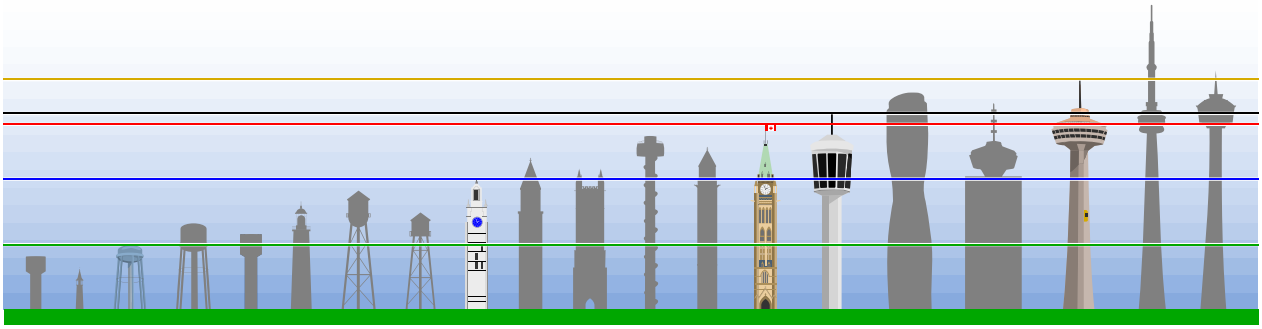
B) 5

C) 6

D) 7

Soluzione

Le seguenti cinque torri sono speciali, come si può vedere dalle linee, quindi la risposta B) è corretta:



Questa è l'informatica!

In questo problema le torri vengono confrontate in base alla loro altezza. Questo tipo di confronti si trovano anche nella *ricerca e ordinamento*, una branca dell'informatica che è stata molto studiata. Ci sono molti algoritmi di ordinamento diversi, i quali sono adatti per diverse applicazioni. L'*algoritmo quicksort* è un algoritmo di ordinamento conosciuto e veloce. Un elemento essenziale dell'algoritmo quicksort è l'identificazione di valori, per i quali tutti i valori alla loro sinistra sono più piccoli e quelli alla sua destra più grandi. Un tale elemento divide il campo da ordinare in due parti e quindi divide il problema di ordinamento originale in due problemi di ordinamento più piccoli. L'elemento tra i due si chiama *perno*. A differenza di questo problema, nell'algoritmo quicksort gli elementi a sinistra non sono più piccoli già in partenza e quelli a destra più grandi: questo deve essere fatto per mezzo dello scambio. Questo processo viene poi ripetuto per ogni parte del campo fino alla fine, quando tutte le parti di campo contengono solo un elemento . . . e il campo è già stato ordinato. Questo modo di procedere *ricorsvio* di scomporre un grande problema in problemi più piccoli e risolverli si chiama *dividi e domina*. È molto diffuso per risolvere problemi difficili.
L'algoritmo quicksort è più veloce rispetto a molti altri algoritmi di ordinamento, da lì anche il nome. Questo perché nel caso usuale attraverso la scelta del perno la grandezza della parte del campo da ordinare viene dimezzata. Un campo con 1000 elementi necessita nel caso usuale circa di 10 piani di divisione (scritti matematicamente sono $\log_2(1000)$ piani di divisione). Siccome inoltre ogni elemento deve essere ancora confrontato con il perno ci sono 10′000 confronti. Altri algoritmi diffusi hanno bisogno in questo stesso caso di, in ordine di grandezza, 1′000′000 di confronti!

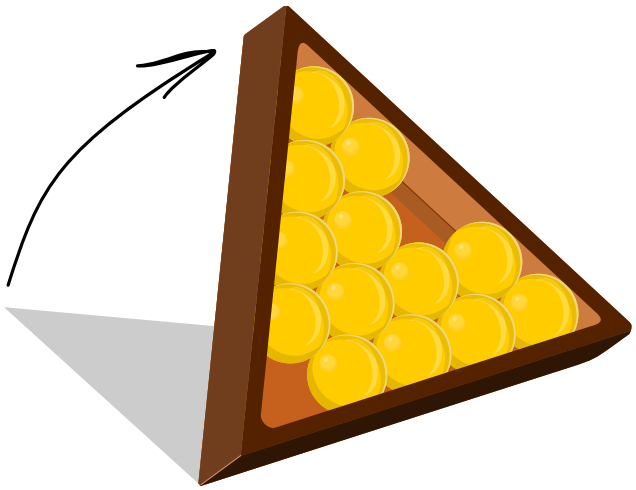
Parole chiave e siti web

Qucksort, Perno, Dividi e domina (divide and conquer)

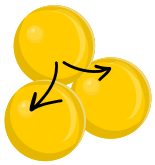
- `https://it.wikipedia.org/wiki/Quicksort`
- `https://it.wikipedia.org/wiki/Divide_et_impera_(informatica)`
- `https://www.youtube.com/watch?v=ywWBy6J5gz8`

7. Biglie traballanti

In una scatola rettangolare sono inserite quindici biglie della stessa dimensione. Due biglie vengono rimosse come mostrato nel disegno. La scatola viene ora inclinata.



Inclinando la scatola alcune biglie possono diventare "traballanti". Una biglia è traballante, se ...



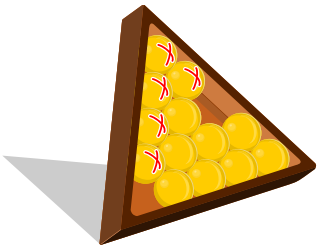
- ... la biglia a sinistra sotto di essa o a destra sotto di essa è rimossa, ...
- ... o la biglia a sinistra sotto di essa o a destra sotto di essa è traballante.

Le biglie della fila più in basso non sono traballanti.
Quante delle tredici biglie sono traballanti?

A) Nessuna biglia
B) 1 biglia
C) 2 biglie
D) 3 biglie
E) 4 biglie
F) 5 biglie
G) 6 biglie
H) 7 biglie
I) 8 biglie
J) 9 biglie
K) 10 biglie
L) 11 biglie
M) 12 biglie
N) Tutte le biglie

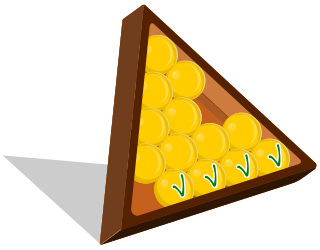
Soluzione

Cinque biglie sono traballanti. Sono contrassegnate nel seguente disegno:

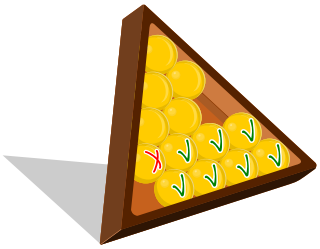


Il modo più semplice di ragionarci è dal basso verso l'alto:

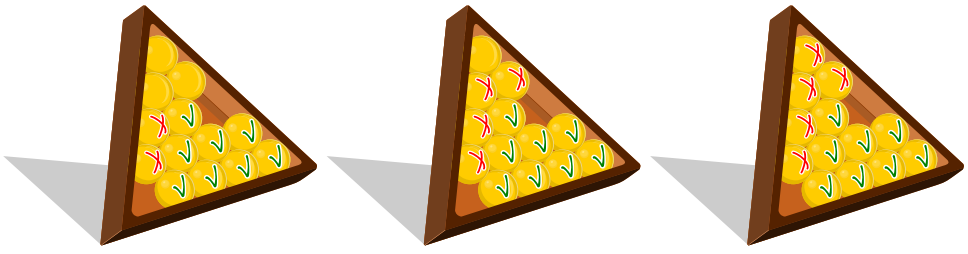
- Tutte le biglie della fila più in basso non sono traballanti.



- Tutte le biglie della seconda riga più in basso, sotto le quali ci sono due biglie non traballanti, sono anche non traballanti, tutte le altre sono traballanti.



- Questo ragionamento viene ripetuto per le file superiori.



Questa è l'informatica!

Ci sono due condizioni per le quali una biglia si può considerare traballante. La prima condizione può essere controllata direttamente. Per controllare la seconda condizione si deve prima sapere se nella fila subito sotto si trova una biglia traballante. Nella fila più in basso è facile perché lì tutte le biglie non sono traballanti, siccome non c'è nessun altra fila sotto di esse. Come è spiegato nella soluzione si può in seguito controllare la fila superiore e scoprire quali biglie di quella fila sono traballanti. In

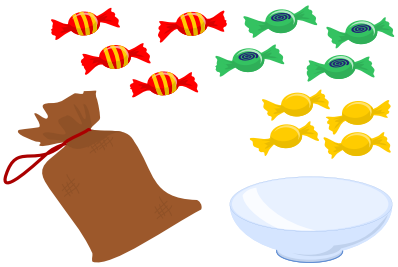
questo modo è possibile fare passare tutte le file dal basso all'alto e scoprire per tutte le biglie quali sono traballanti.
Il principio per cui una condizione è dipendente dal risultato di un'altra condizione dello stesso tipo si chiama *ricorsione*. Le condizioni ricorsive sono costruite in modo che il loro risultato o è dato (*fine della ricorsione*, in questo caso tutte le biglie dell'ultima fila non sono traballanti) o dipende dal risultato di condizioni ricorsive successive (*passo ricorsivo*, in questo caso per tutte le biglie che non sono nell'ultima fila possiamo avere il risultato solo se prima abbiamo controllato la fila inferiore ad esse).
Il principio della ricorsione viene usato spesso in informatica. Con esso si possono risolvere molto facilmente ed elegantemente molti problemi complessi. È anche possibile convertire soluzioni ricorsive in soluzioni passo a passo (*iterative*). Un classico esempio dove una soluzione ricorsiva è molto facile è la torre di Hanoi.

Parole chiave e siti web

Ricorsione, Torre di Hanoi

- `https://it.wikipedia.org/wiki/Algoritmo_ricorsivo`
- `https://it.wikipedia.org/wiki/Torre_di_Hanoi`

8. Un sacchetto pieno di caramelle



Petra ha in un sacchetto opaco quattro caramelle rosse, quattro verdi e quattro gialle. Inoltre ha una ciotola vuota.
Petra e Marco giocano a un gioco. Marco durante i tre round può estrarre dal sacchetto una caramella. Per ogni caramella estratta valgono le regole seguenti:

- Finché la caramella estratta è verde può metterla nella ciotola e può estrarre un'ulteriore caramella nello stesso round.
- Se la caramella estratta è rossa, Marco la può mettere nella ciotola e finisce il suo round.
- Se la caramella estratta è gialla, Marco la mangia direttamente senza metterla nella ciotola e finisce il suo round.

Alla fine del gioco quante caramelle ha Marco al massimo nella ciotola?

A) 0
B) 1
C) 2
D) 3
E) 4
F) 5
G) 6
H) 7
I) 8
J) 9
K) 10
L) 11
M) 12

Soluzione

La risposta corretta è H) 7.
Nel caso migliore vengono estratte tutte le quattro caramelle verdi. Ciò significa che, da un lato, i quattro dolci verdi sono nella ciotola e, dall'altro lato, Marco ha potuto estrarre un altro dolce quattro volte nel corso dei tre turni, per un totale di sette.
Per le restanti tre caramelle, nel caso migliore, Marco estrae una caramella rossa, le quali alla fine si trovano anche nella ciotola. In totale ci sono quattro caramelle verdi e tre caramelle rosse, ci sono quindi in totale sette caramelle nella ciotola.
Non ci possono essere più di sette caramelle. Dopo ogni estrazione ci può essere al massimo una caramella nella ciotola e siccome ci sono solo quattro caramelle verdi, dopo le quali si può estrarre un'ulteriore caramella, ci sono al massimo sette caramelle.
L'ordine nel quale vengono estratte le caramelle nel caso migliore è relativamente indifferente, finché l'ultima caramella estratta è una rossa, poiché si può sempre estrarre un'altra caramella dopo una caramella verde.

Questa è l'informatica!

Due delle tre regole del problema sono formulate come una *diramazione*: *se* vale una condizione, *allora* viene effettuata una determinata azione. Tali diramazioni sono molto frequenti nella programmazione. Per questo scopo vengono spesso utilizzate le parole chiavi inglesi *if* (ingl. per "se") e *then* (ingl. per "allora"). Una regola è formulata in modo che qualcosa è ripetuto *fino a quando* una determinata condizione non vale più. Questo viene chiamato un *ciclo*, per cui spesso viene usata la parola chiave inglese *while* (ingl. per "finché"). Cicli di questo tipo possono anche essere formulati come cicli induttivi, che specificano un certo numero di ripetizioni.
Si potrebbe formulare il gioco di Petra anche così:

fissa i round a tre
finché c'è ancora un round:
    riduci i round di 1
    estrai una caramella
    finché la caramella è verde, *allora* mettila nella ciotola ed estrai una caramella
    se la caramella è rossa, *allora* mettila nella ciotola
    se la caramella è gialla, *allora* mangiala.

Per risolvere il problema bisogna *analizzare* il programma. In un caso semplice come questo si potrebbe naturalmente provare tutti i possibili ordini di caramelle. Questo potrebbe perfino essere eseguito automaticamente da un computer. La spiegazione data nella soluzione, al contrario, si basa sulla comprensione delle relazioni, in modo da dimostrare che un determinato risultato è vero, senza che il programma sia eseguito. Tali analisi non sono eseguibili per ogni caso da un computer, come può mostrare la *teoria della calcolabilità*. Donald Knuth, uno dei più grandi informatici del 20esimo secolo, l'ha espresso in poche parole: *"Attenzione agli errori nel codice; ho solo dimostrato che è corretto, non l'ho provato"*.

Parole chiave e siti web

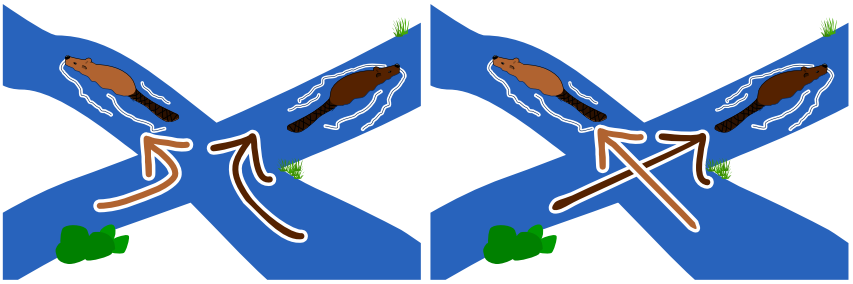
Diramazione, Ciclo, Teoria della calcolabilità

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

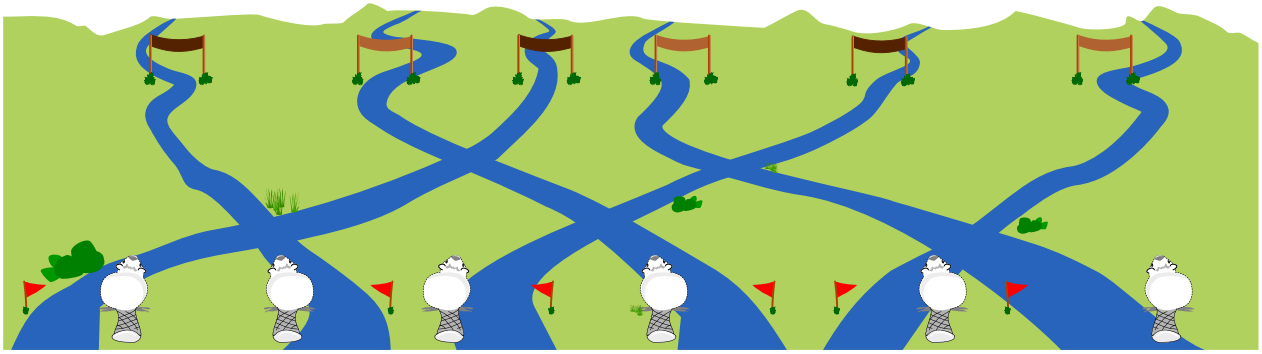
- `https://it.wikipedia.org/wiki/Struttura_di_controllo`
- `https://it.wikipedia.org/wiki/Teoria_della_calcolabilit%C3%A0`
- `https://en.wikiquote.org/wiki/Donald_Knuth`

9. La rete dei castori

Tre castori marroni chiari e tre castori marroni scuri nuotano attraverso un sistema di canali dal basso verso l'alto. Ad ogni incrocio di due canali si incontrano due castori. Se questi castori sono di colori diversi il castoro marrone chiaro nuota verso sinistra e il castoro marrone scuro nuota verso destra. Altrimenti nuotano semplicemente uno a sinistra e uno a destra.

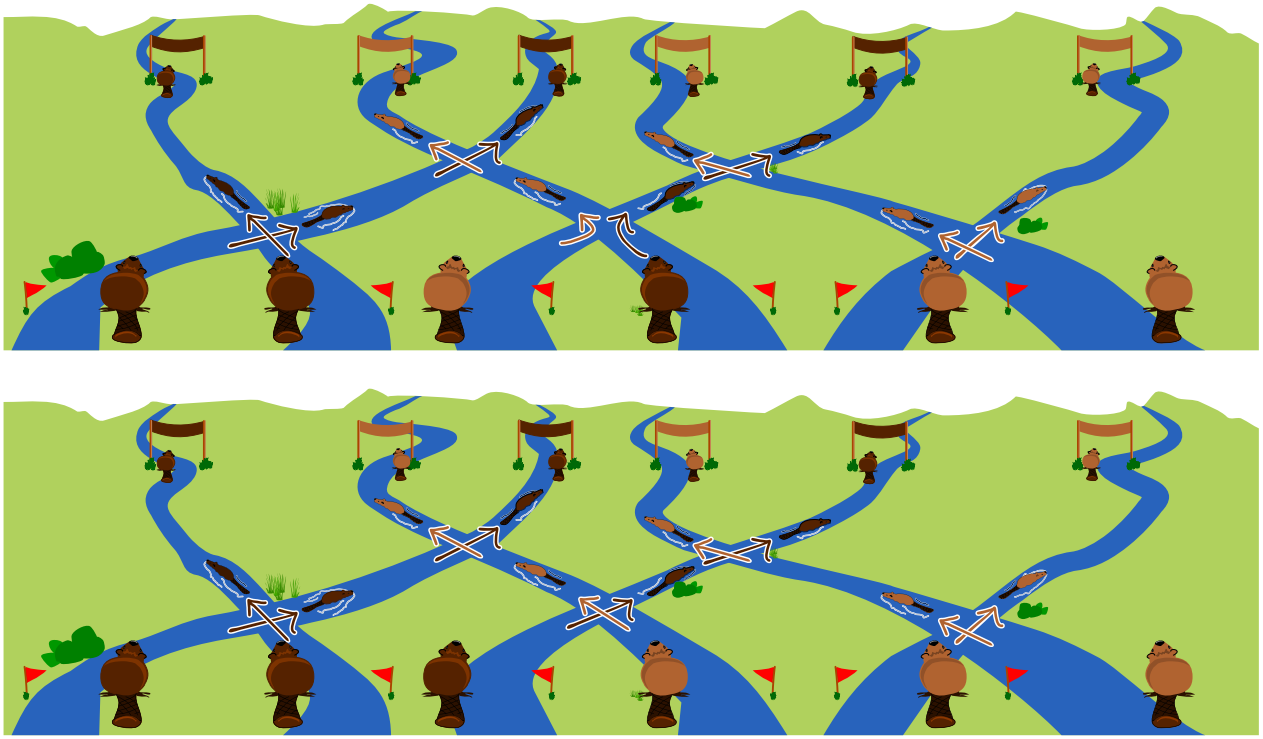


Alla fine i castori dovrebbero arrivare nell'ordine seguente: marrone scuro, marrone chiaro, marrone scuro, marrone chiaro, marrone scuro e marrone chiaro.
Come devono partire i tre castori marroni chiari e i tre castori marroni scuri in modo che l'arrivo sia corretto?



Soluzione

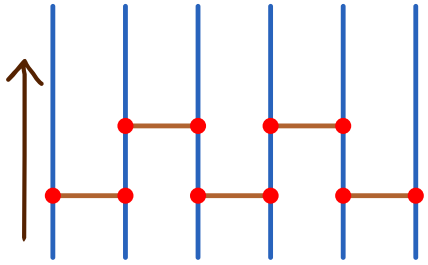
Ci sono due risposte corrette:



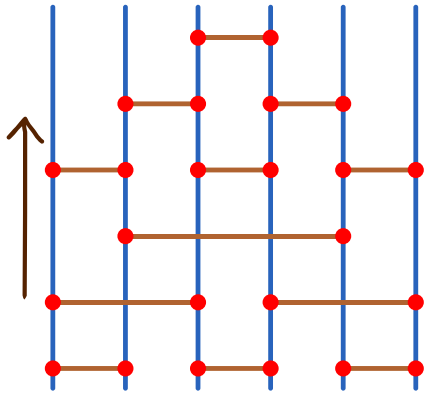
Queste sono anche le uniche due risposte corrette. Infatti per far sì che possa arrivare un castoro marrone scuro al traguardo a sinistra, al primo incrocio non può nuotare da sinistra nessun castoro marrone chiaro, perché se no quest ultimo dovrebbe nuotare verso sinistra. Quindi le due posizioni di partenza a sinistra devono essere occupate da due castori marroni scuri.
Lo stesso vale per il traguardo a destra del castoro marrone chiaro: Infatti per far sì che possa arrivare un castoro marrone chiaro tutto a destra al traguardo, al primo incrocio da destra si devono incontrare due castori marroni chiari. Quindi le due posizioni di partenza a destra devono essere occupate da due castori marroni chiari.
Per i castori in mezzo è indifferente se il terzo castoro marrone chiaro sta a sinistra e il terzo castoro marrone scuro sta a destra o al contrario, siccome dopo l'incrocio in mezzo, il castoro marrone chiaro nuota in ogni caso verso sinistra e il castoro marrone scuro verso destra.

Questa è l'informatica!

Il sistema di canali con le regole su chi nuota a sinistra e chi a destra rappresenta una parte di una *rete di ordinamento*. In una rete di ordinamento i dati viaggiano su una linea (i canali di questo problema) e per ogni connessione (gli incroci in questo problema) viene verificato se si dovrebbe scambiare o no. Un castoro scuro può quindi essere immaginato come il numero 0 e un castoro chiaro come il numero 1. La risultante rete di ordinamento assomiglia a questa:



Una rete di ordinamento completa e minimale per questo problema apparirebbe così, si vede bene come la parte di una rete di ordinamento viene integrata in questo problema:



Le reti di ordinamento sono particolarmente efficienti se si possono eseguire i confronti parallelamente. Perciò è difficile trovare reti di ordinamento ottimali per insiemi di dati più grandi.
Generalmente ci si può immaginare il sistema di canali dei castori anche come sistema di cavi in una rete di computer come internet. Qui i canali rappresentano connessioni di cavi dirette tra due router, gli incroci. Di regola, in tali router vengono programmate tabelle di routing fisse, con il cui aiuto i pacchetti di dati vengono inviati nella direzione della loro destinazione.

Parole chiave e siti web

Rete di ordinamento, Rete di computer, Router, tabelle di routing

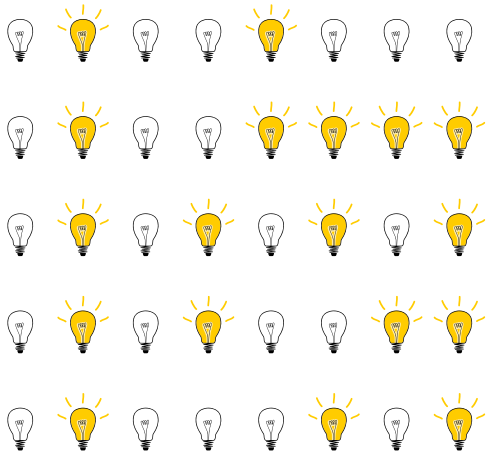
- `https://en.wikipedia.org/wiki/Sorting_network`
- `http://www.inf.fh-flensburg.de/lang/algorithmen/sortieren/networks/optimal/optimal-sorting-networks.htm`
- `https://www.computernetworkingnotes.com/ccna-study-guide/basic-routing-concepts-and-protocols-explained.html`
- `https://it.wikipedia.org/wiki/Instradamento`
- `https://it.wikipedia.org/wiki/Tabella_di_routing`

10. Segnali luminosi

Sina ha collegato otto lampade con interruttori e cavi. Può quindi mandare dei messaggi. Usa la seguente tabella di codificazione, nella quale `0` significa che la lampada corrispondente è spenta (💡) e `1`, che la lampada corrispondente è accesa (💡):

`A`: 01000001	`J`: 01001010	`S`: 01010011
`B`: 01000010	`K`: 01001011	`T`: 01010100
`C`: 01000011	`L`: 01001100	`U`: 01010101
`D`: 01000100	`M`: 01001101	`V`: 01010110
`E`: 01000101	`N`: 01001110	`W`: 01010111
`F`: 01000110	`O`: 01001111	`X`: 01011000
`G`: 01000111	`P`: 01010000	`Y`: 01011001
`H`: 01001000	`Q`: 01010001	`Z`: 01011010
`I`: 01001001	`R`: 01010010	

Sina ora manda i segnali luminosi seguenti:



Cosa significano i segnali luminosi di Sina?

A) `HOUSE`

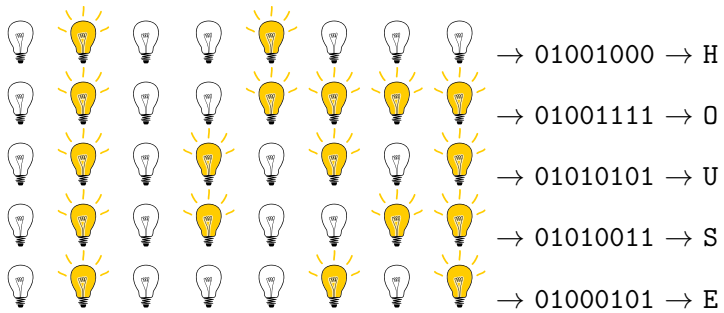
B) `HAPPY`

C) `HORSE`

D) `HONEY`

Soluzione

I segnali luminosi significano:



Quindi la parola della soluzione A) `HOUSE` è corretta.
D'altronde si può trovare questa risposta molto velocemente: la lettera in mezzo è diversa per ogni parola: A) `U`, B) `P`, C) `R` e D) `N`. Siccome il terzo segnale luminoso significa `U` solo la risposta A) può ancora essere corretta.

Questa è l'informatica!

La codificazione di Sina non è scelta in modo casuale. Usa una parte del cosiddetto codice ASCII, il quale fu già sviluppato oltre cinquant'anni fa per lo scambio di messaggi. Si basa sul principio del codice binario, il quale fu descritto da Gottfried Leibnitz (1646–1716) appena nel 1679 e nel 1703 sulla base dei sistemi precursori indiani e cinesi per la rappresentazione di numeri e il calcolo con questi numeri. Claude Shannon (1916–2001) li applicò poi allo sviluppo dei computer.
Oggi i computer utilizzano sviluppi successivi del codice ASCII. Siccome il primo codice ASCII conteneva solamente 95 caratteri stampabili (lettere latine maiuscole e minuscole, le cifre da 0 a 9 così come due segni di interpunzione) e i restanti 33 erano caratteri di controllo (ad esempio per stampanti), si necessitavano delle espansioni per gli Umlaut e altri alfabeti. Questo è accaduto prima sotto forma di codice ANSI e successivamente nell'Unicode, oggi quasi usato universalmente. Le lettere di Sina sono ancora codificate nello stesso modo nella variante Unicode più diffusa, la UFT-8.
A proposito il primo blocco di segni (identico in ASCII, ANSI e Unicode) è (i caratteri di controllo sono lasciati vuoti, ␣ sta per lo spazio):

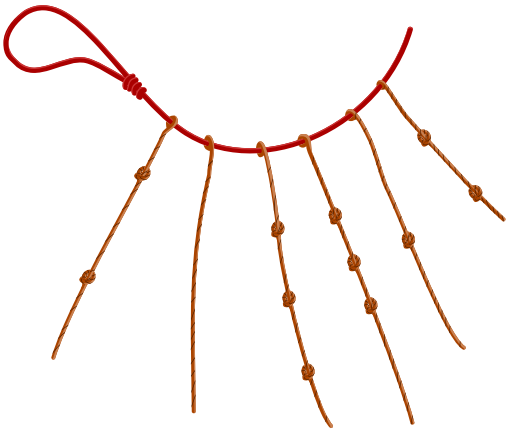
	000 ...	001 ...	010 ...	011 ...	100 ...	101 ...	110 ...	111 ...
...0000			␣	0	@	P	‘	p
...0001			!	1	A	Q	a	q
...0010			"	2	B	R	b	r
...0011			#	3	C	S	c	s
...0100			$	4	D	T	d	t
...0101			%	5	E	U	e	u
...0110			&	6	F	V	f	v
...0111			’	7	G	W	g	w
...1000			(	8	H	X	h	x
...1001			)	9	I	Y	i	y
...1010			*	:	J	Z	j	z
...1011			+	;	K	[	k	{
...1100			,	<	L	\	l	|
...1101			-	=	M	]	m	}
...1110			.	>	N	^	n	~
...1111			/	?	O	_	o	

Parole chiave e siti web

ASCII, Unicode, codificazione

- https://it.wikipedia.org/wiki/ASCII
- https://en.wikipedia.org/wiki/Binary_code
- https://it.wikipedia.org/wiki/Gottfried_Wilhelm_von_Leibniz
- https://it.wikipedia.org/wiki/Claude_Shannon
- https://it.wikipedia.org/wiki/Unicode
- https://it.wikipedia.org/wiki/UTF-8
- https://www.unicode.org/charts/PDF/U0000.pdf

11. Quipu



Gli Inca usavano in precedenza dei nodi per la trasmissione di messaggi. Ad una corda principale sono appese delle corde secondarie, alle quali erano fatti dei nodi. Questi cosiddetti Quipu erano grandi e dispendiosi da produrre. Immaginati che deve essere sviluppata una versione semplificata dei Quipu. Le condizioni sono:

- Alla corda principale devono essere attaccate sempre lo stesso numero di corde secondarie.
- Le corde secondarie si differenziano soltanto nel numero di nodi.
- Una corda secondaria ha 0, 1, 2 o 3 nodi.
- L'ordine delle corde secondarie è stabilito da un nodo nella corda principale.
- Dovrebbero essere possibili 30 Quipu unici e distinguibili per diversi messaggi.

Quante corde secondarie ha al meno la versione semplificata del Quipu sotto queste condizioni?

A) 2

B) 3

C) 4

D) 5

E) 8

F) 10

Soluzione

La risposta B) 3 è corretta.
Ogni corda secondaria può memorizzare uno di 4 diversi valori (0, 1, 2 o 3). Con due corde si hanno $4 \cdot 4 = 16$ possibili combinazioni, con tre corde si hanno $4 \cdot 4 \cdot 4 = 64$ possibili combinazioni e così via. Quindi sono abbastanza tre corde secondarie, più corde secondarie contraddirebbero la condizione per la quale ci devono essere il minor numero di corde secondarie possibili. Poiché l'ordine dei valori è determinato dal nodo della corda principale, non è necessario assicurarsi di poter leggere la corda in una o nell'altra direzione.

Questa è l'informatica!

I *Quipu* furono usati effettivamente dagli Inca in Sud America. I Quipu grigi erano utilizzati per la contabilità e la riscossione delle imposte. Con l'aiuto di corde colorate si presume che si potevano codificare fino a 95 sillabe diverse e che quindi potesse avvenire della corrispondenza. In contrasto alla variante semplice come in questo compito, c'erano anche diversi tipi di nodi e in alcuni casi corde terziarie che erano legate alle corde secondarie.
L'esempio del problema è una variante semplificata. Siccome l'ordine è stabilito dai nodi nella corda principale, i valori singoli (0, 1, 2 o 3) generano una *notazione posizionale*, in questo caso in base 4. Le notazioni posizionali sono ampiamente diffuse: di regola viene usata la notazione posizionale in base 10, i computer usano la notazione posizionale in base 2 (chiamati anche *numeri binari*). Nei primi tempi dei computer c'erano anche tentativi di costruire computer basati sul *sistema ternario* con base 3 (in quel caso interpretati come -1, 0 e $+1$). Con una notazione posizionale in base b si possono memorizzare esattamente b^n valori diversi in n posizioni. Un byte (8 bit, che possono essere o 0 o 1) può quindi memorizzare $2^8 = 256$ valori diversi (da 0 fino a 255), il Quipu in questo problema può memorizzare $4^3 = 64$ valori diversi.
Per gli Inca, tra l'altro, sarebbe bastata una singola corda secondaria per memorizzare i valori da 1 a 30. Anche loro usavano una notazione posizionale in base 10, come noi per scrivere i numeri, semplicemente con nodi distinti su una corda. Ad esempio, le unità sarebbero state codificate con un doppio nodo e le decine con il numero corrispondente di nodi di chiusura. Tuttavia, avrebbero avuto bisogno fino a 4 nodi, e poi anche di tipi diversi di nodi.

Parole chiave e siti web

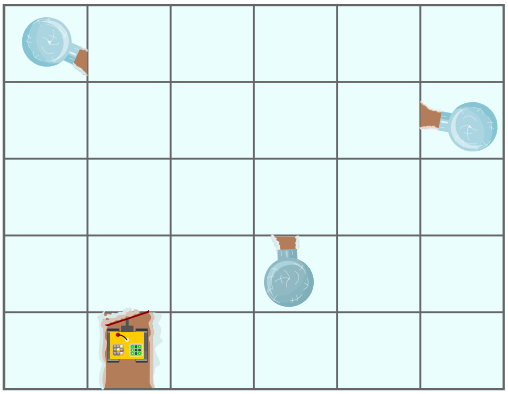
Quipu, Notazione posizionale

- `https://it.wikipedia.org/wiki/Quipu`
- `https://en.wikipedia.org/wiki/Double_overhand_knot`
- `https://en.wikipedia.org/wiki/Stopper_knot`
- `https://it.wikipedia.org/wiki/Notazione_posizionale`
- `https://it.wikipedia.org/wiki/Calcolatore_ternario`

12. Bufera di neve

Dopo una forte bufera di neve ci sono ammassi di neve dappertutto e gli abitanti dei tre igloo sono isolati. Gli abitanti possono sgombrare i sentieri con l'aiuto dei loro spazzaneve telecomandati. Funziona così:

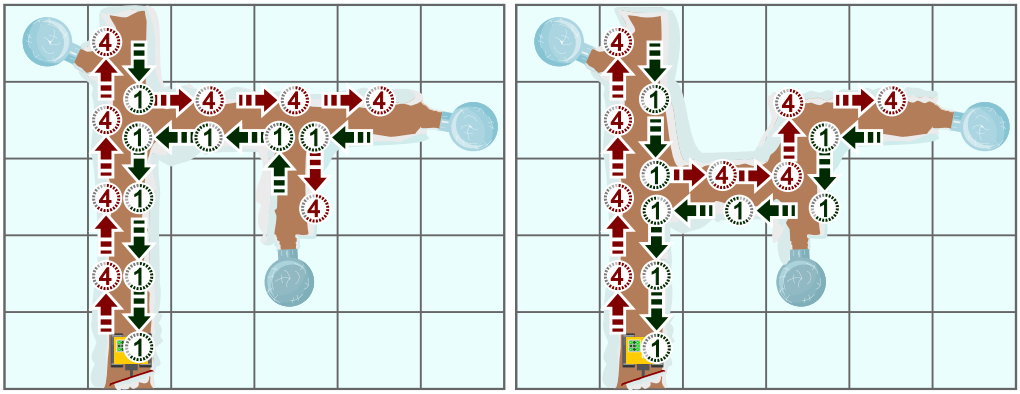
- Lo spazzaneve necessita di 4 minuti per andare da un quadrato a un altro quadrato innevato adiacente e per sgombrarlo.
- Lo spazzaneve necessita di 1 minuto per andare da un quadrato a un altro quadrato adiacente senza neve.
- I quadrati adiacenti sono sempre solo i quadrati sulla cartina che sono direttamente sopra, sotto, a sinistra o a destra di un altro quadrato, lo spazzaneve non può spostarsi diagonalmente.
- Appena il quadrato davanti all'entrata di un igloo è sgombrato gli abitanti dell'igloo possono liberare l'entrata e non sono più isolati.



Quanti minuti necessita lo spazzaneve per liberare tutti gli igloo dall'isolamento e tornare al suo quadrato di partenza nel caso ideale?

Soluzione

La risposta giusta è 40 minuti. La grafica seguente mostra entrambi i percorsi ottimali dello spazzaneve:



Perché non c'è un modo più veloce? Per raggiungere l'igloo in alto a sinistra devono essere sgombrati 4 quadrati. Questi sono 16 minuti. Per raggiungere l'igloo a destra devono essere sgombrati altri tre quadrati. Questi sono altri 12 minuti. Per raggiungere l'igloo in basso deve essere sgombrato un altro quadrato, perché o si deve sgombrare un vicolo cieco del percorso trasversale o bisogna modificare verso il basso il percorso trasversale. Questi sono altri 4 minuti. Per tornare indietro lo spazzaneve deve percorrere quattro quadrati verso il basso e tre quadrati verso sinistra. Questi sono altri 7 minuti. Per il giro attraverso il vicolo cieco o per il percorso trasversale modificato, necessita 1 ulteriore minuto. In totale quindi necessita almeno 40 minuti.
Se lo spazzaneve sgombrasse più velocemente i quadrati sarebbe forse più efficiente se sulla strada del ritorno dall'igloo più in basso andasse sul quadrato innevato a sinistra e lo sgombrasse. Ma questo gli costa 4 minuti per lo sgombramento e 1 minuto per proseguire sul quadrato senza neve, quindi 5 minuti. Il giro passante dai quadrati già sgombrati gli costa soltanto 4 minuti.

Questa è l'informatica!

In questo problema si cerca una rete di sentieri che colleghi tutti i luoghi (gli igloo e il quadrato di partenza dello spazzaneve) con costi minimi (il tempo che necessità lo spazzaneve). Queste reti di sentieri non contengono per forza i sentieri più corti tra tutti i nodi, ma i costi per costruire queste reti di sentieri sono i più bassi possibili. Queste reti di sentieri sono chiamati *alberi di Steiner*. Vengono usati ad esempio per la costruzione di pannelli di computer o la costruzione di reti ferroviarie per merci poco usate. Trovare alberi di Steiner è uno dei problemi di ottimizzazione più difficili e dispendiosi a livello di tempo in informatica, spesso si usano algoritmi che trovano una soluzione sufficientemente buona, ma non necessariamente la migliore.
Nel caso di questo problema i costi vengono calcolati in modo particolare, perché non sono solo calcolati i costi fissi per la costruzione di un sentiero (i 4 minuti per sgombrare un quadrato), bensì bisogna considerare anche i costi per il ritorno della macchina. Quindi questo problema è una generalizzazione del problema dell'albero di Steiner.

Parole chiave e siti web

Problema dell'albero di Steiner (Steiner tree problem)

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

13. Che bello che ci sono gli alberi

Sergio ha scritto una canzone che descrive come da un albero possono nascere diversi oggetti. Un verso fa così:

> Che bello che ci sono gli alberi.
> Su un albero crescono le foglie,
> Su un albero crescono i fiori,
> Dai fiori crescono frutti,
> Con le foglie e con i fiori posso fare delle corone.

Per Sergio era importante che dopo la prima riga del verso venissero usati solo oggetti che aveva già menzionato prima.
Quale dei seguenti versi è sbagliato per Sergio?

A) Che bello che ci sono gli alberi.
Su un albero crescono i fiori,
Su un albero crescono le foglie,
Con le foglie e con i fiori posso fare delle corone.
Dai fiori crescono frutti.

B) Che bello che ci sono gli alberi.
Su un albero crescono i fiori,
Su un albero crescono le foglie,
Dai fiori crescono frutti,
Con le foglie e con i fiori posso fare delle corone.

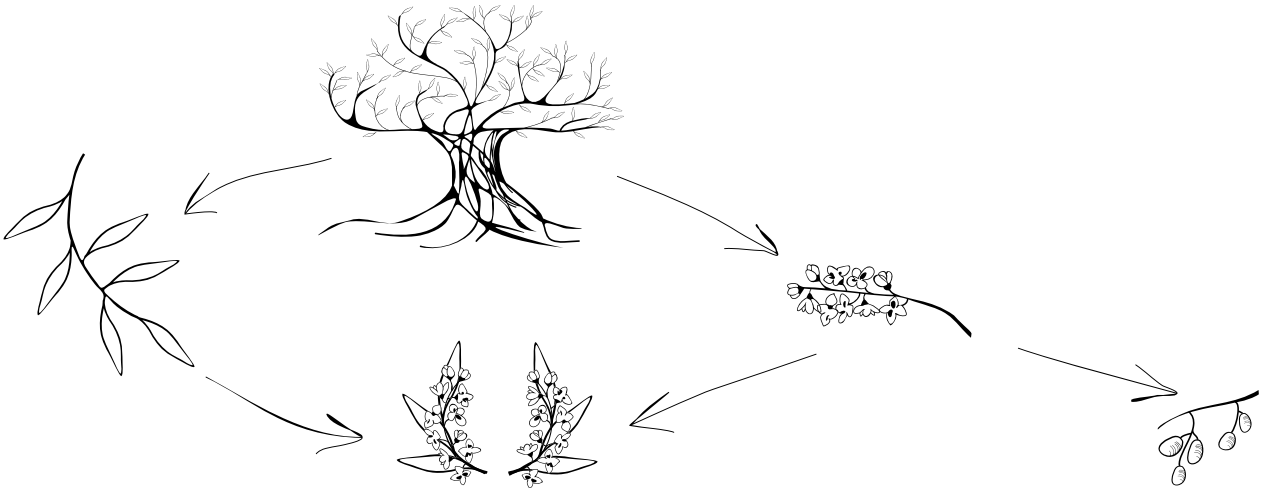
C) Che bello che ci sono gli alberi.
Su un albero crescono le foglie,
Dai fiori crescono frutti,
Su un albero crescono i fiori,
Con le foglie e con i fiori posso fare delle corone.

D) Che bello che ci sono gli alberi.
Su un albero crescono i fiori,
Dai fiori crescono frutti,
Su un albero crescono le foglie,
Con le foglie e con i fiori posso fare delle corone.

E) Che bello che ci sono gli alberi.
Su un albero crescono le foglie,
Su un albero crescono i fiori,
Con le foglie e con i fiori posso fare delle corone,
Dai fiori crescono frutti.

Soluzione

Si può descrivere la dipendenza degli oggetti "albero", "foglie", "fiori", "corone" e "frutti" con l'aiuto di un grafo, dove una freccia significa che uno è necessario per l'altro:



Dunque prima dei frutti devono essere menzionati i fiori e prima delle corone, foglie e fiori.
Le risposte hanno il seguente ordine degli oggetti:

A) Albero, fiori, foglie, corone, frutti

B) Albero, fiori, foglie, frutti, corone

C) Albero, foglie, *frutti*, *fiori*, corone

D) Albero, fiori, frutti, foglie, corone

E) Albero, foglie, fiori, corone, frutti

Nella risposta C) i frutti vengono cantati prima dei fiori (accentuato sopra), ciò è una contraddizione, siccome i frutti necessitano dei fiori. In tutti gli altri versi le condizioni sono rispettate.

Questa è l'informatica!

Nel 1974 il musicista italiano Sergio Endrigo (1933–2005) scrive la canzone per bambini "Ci vuole un fiore" su un testo di Gianni Rodari (1920–1980). In questa canzone canta che, se si vuole un tavolo, prima si ha bisogno del legno, per il legno un albero, per un albero un seme, per un seme un frutto e per un frutto un fiore. Descrive anche che per avere un fiore bisogna passare da un ramo, da un albero, da un bosco, da un monte e dalla terra, per la quale si ha anche bisogno di un fiore. Conclude dicendo che alla fine si ha bisogno di un fiore per tutto.
La precedenza di un oggetto su un altro può essere descritta con l'aiuto di un *grafo orientato*. Un tale grafo si trova nella spiegazione della risposta. È un *grafo orientato aciclico*, che descrive gli *ordini permessi di un insieme*. Se si desidera un nodo (uno degli oggetti), bisogna già avere tutti gli oggetti che lo puntano. Lo stesso vale per questi oggetti, in modo che bisogna tornare indietro *ricorsivamente* fino a quando si giunge su degli oggetti verso i quali non punta nessuna freccia. Questi possono essere usati come oggetti di partenza.
Del resto, la canzone di Sergio Endrigo non si lascia descrivere con l'aiuto di un grafo orientato e aciclico. Nella seconda parte descritta sopra canta che per un fiore alla fine si ha bisogno di un fiore.

Questa è una contraddizione, siccome il grafo deve essere aciclico, quindi non può contenere nessuna conclusione circolare. Con questa rottura logica rende ancora più chiara la sua affermazione: "Ci vuole un fiore"!

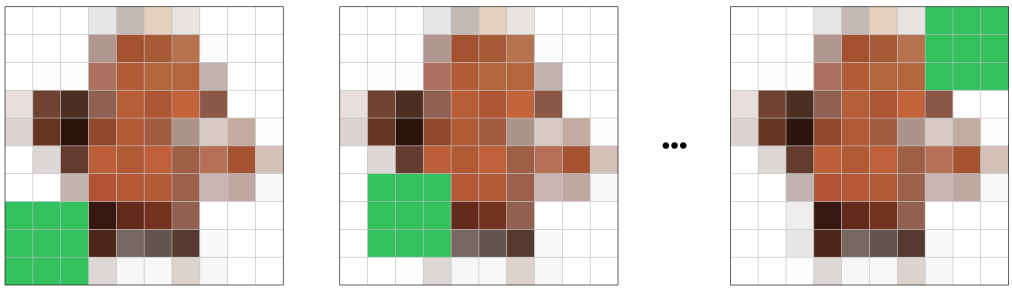
Parole chiave e siti web

Grafo orientato e aciclico, Ordinamento topologico

- `https://www.filastrocche.it/contenuti/ci-vuole-un-fiore/`
- `https://www.youtube.com/watch?v=9ht4tIot8XY`
- `https://en.wikipedia.org/wiki/Precedence_graph`
- `https://it.wikipedia.org/wiki/Ordinamento_topologico`

14. Compressione video

I video hanno bisogno di molto spazio di archiviazione. Allo stesso tempo, tuttavia, due immagini fisse consecutive di un video sono spesso molto simili.
Il video seguente è grande 10×10 punti d'immagine. Il quadrato verde nell'angolo in basso a sinistra è grande 3×3 punti d'immagine. Si muove da un'immagine fissa all'altra di un punto d'immagine a sinistra e uno in alto, fino a quando arriva nell'angolo in alto a destra.

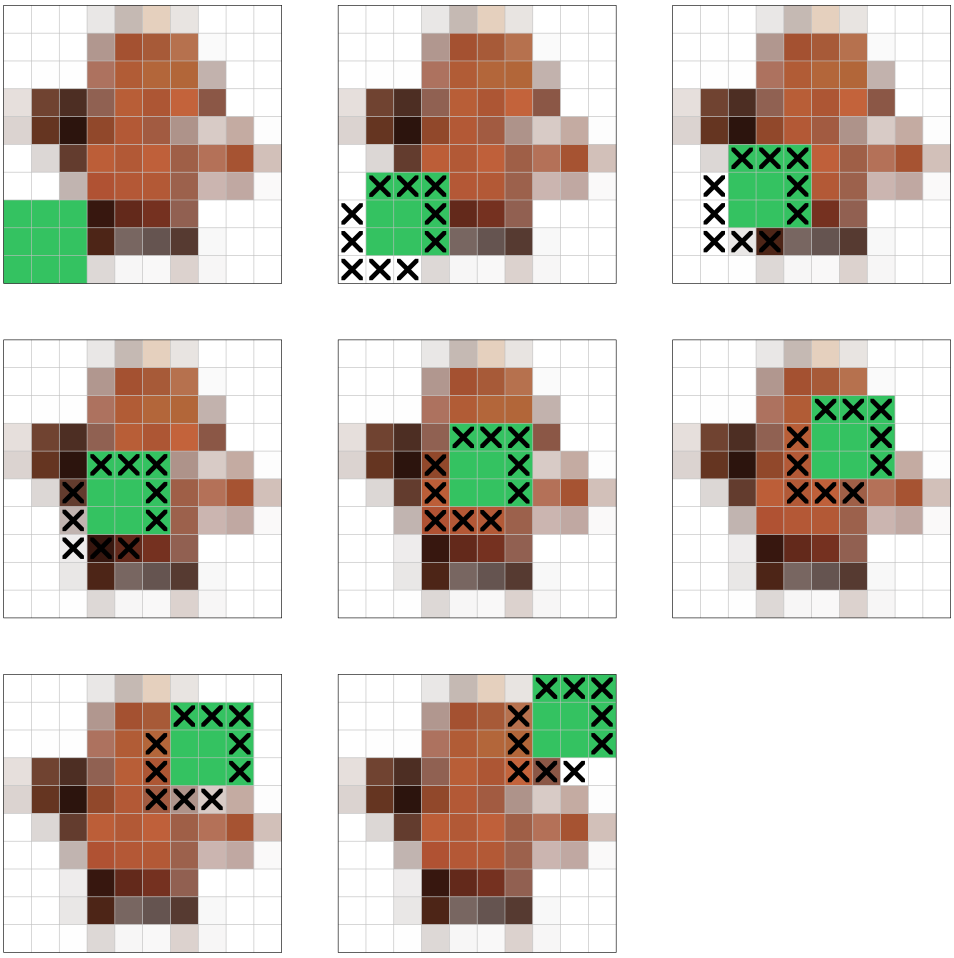


Per risparmiare spazio di archiviazione a partire dalla seconda immagine fissa vengono memorizzati solamente i punti d'immagine che sono cambiati.
Quanti punti dell'immagine devono essere memorizzati per l'intero video?

A) 100
B) 135
C) 140
D) 170
E) 180
F) 700
G) 800
H) 1000

Soluzione

Le singole immagini fisse del video appaiono così, quando si marca ogni punto che è cambiato:



Prima si stabilisce che la prima immagine fissa contiene $10 \cdot 10 = 100$ punti dell'immagine. Per ogni ulteriore immagine fissa, devono essere memorizzati solo i punti d'immagine che sono cambiati. Questi sono i cinque punti d'immagine sotto a sinistra del quadrato che sono sostituiti da dei punti d'immagine dello sfondo, così come i cinque punti d'immagine sopra a destra del quadrato, i quali costituiscono il nuovo quadrato. Quindi per immagine fissa vengono modificati 10 punti d'immagine. Il quadrato ha bisogno di ulteriori sette immagini fisse per muoversi da in basso a sinistra a in alto a destra, quindi devono essere inseriti $10 \cdot 7 = 70$ punti d'immagine per i punti d'immagine modificati oltre agli originali 100 punti d'immagine, in modo che la risposta D) 170 è corretta.

Questa è l'informatica!

Come è stato appena descritto nel problema, al giorno d'oggi la compressione video digitale gioca un grande ruolo. La procedura descritta è solo uno dei diversi approcci alla compressione dei video. Un altro approccio è tralasciare determinate informazioni che non sono percepite dagli esseri umani. Il formato immagine JPEG sfrutta tali connessioni. In immagini particolarmente compresse, questo può essere riconosciuto dalla formazione di blocchi, perché per tale blocco la somiglianza del colore è stata erroneamente interpretata come impercettibile. Ulteriori possibilità sono la riduzione dello spazio colorato.

Lo standard MPEG si basa su queste idee. Come in questo problema distingue tra diversi tipi di immagini fisse. Un tipo di immagini fisse (cosiddette "immagini intra") rappresentano una completa immagine fissa (simile alla nostra prima immagine fissa). Un altro tipo di immagini fisse si basano sulle immagini fisse precedenti ("immagini P", come le nostre ulteriori immagini fisse) o perfino addizionalmente sulle immagine fisse successive ("immagini D", non si presentano in questo problema). Le immagini Intra vengono inserite ad intervalli regolari per ridurre al minimo lo sforzo del buffer e per poter "rientrare" in caso di errori di trasmissione. Nel caso di video altamente compressi, le immagini P e D possono essere riconosciute quando uno sfondo poco luminoso "salta" improvvisamente, anche se la scena si è mossa lentamente nel corso di un certo periodo di tempo.
A proposito, il fabbisogno di memoria non è così formidabile come suggerito nel compito: oltre ai valori di colore, deve essere salvata anche la posizione dei pixel modificati. Questo dà forse un fattore 2 per il fabbisogno di memoria di un pixel modificato. Ma anche 240 unità di archiviazione sarebbero comunque un impressionante risparmio di spazio rispetto a 800 unità di archiviazione, soprattutto perché la procedura descritta nel compito è senza perdite, a differenza di MPEG!

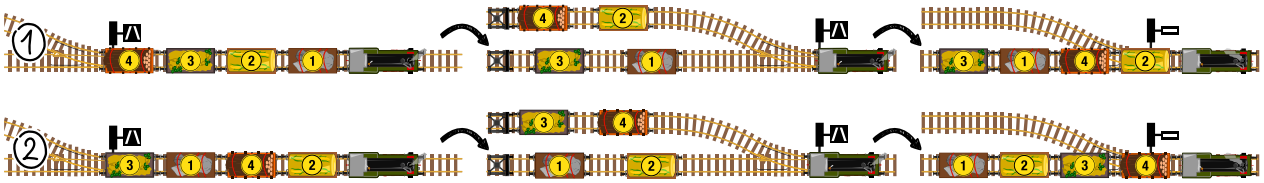
Parole chiave e siti web

Compressione video digitale

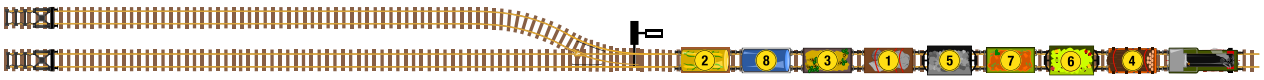
- `https://it.wikipedia.org/wiki/Compressione_video_digitale`
- `https://it.wikipedia.org/wiki/JPEG`
- `https://it.wikipedia.org/wiki/MPEG-1`

15. Stazione di smistamento

Un treno merci deve consegnare i singoli vagoni merci ai binari di raccordo lungo la linea principale. Per risparmiare tempo ed evitare manovre sulla linea principale, i vagoni merci nell'area di smistamento dovrebbero essere ordinati secondo i loro numeri in modo che il vagone merci numero 1  si trovi all'estrema sinistra.
Nella stazione di smistamento c'è una collina di decorso sopra la quale i vagoni merci vengono spinti da destra a sinistra. Sulla collina di decorso viene deciso per ogni singolo vagone merci in quale dei due binari morti deve passare. Poi la locomotiva li estrae di nuovo: prima tutti quelli di un binario morto e poi tutti quelli dall'altro. Questo processo viene indicato come processo di spinta.
Quando ad esempio quattro vagoni merci devono essere ordinati sono sufficienti due processi di spinta (passaggio ① e passaggio ②):



Non è possibile ordinare i quattro vagoni merci in un processo di spinta.
Se i vagoni sono nell'ordine 2 – 8 – 3 – 1 – 5 – 7 – 6 – 4 *quanti processi di spinta sono necessari come minimo, in modo che il treno merci sia ordinato?*



A) 3

B) 4

C) 5

D) 6

E) 7

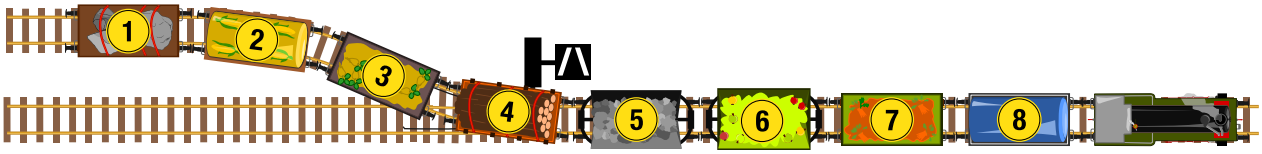
F) 8

Soluzione

La risposta corretta è che sono necessari A) 3 processi di spinta.
Naturalmente si possono smistare i vagoni merci con diversi metodi, ma uno dei migliori è di prima spingere i vagoni 1, 3, 5 e 7 nel binario superiore e poi i vagoni 2, 4, 6 nel binario inferiore, ed estrarre prima i vagoni merci dal binario inferiore e poi quelli dal binario superiore:

Quindi per ogni coppia (1 e 2, 3 e 4, 5 e 6, 7 e 8) di vagoni merci quello con il numero più piccolo è a sinistra di quello con il numero più grande.
Successivamente, ha senso spingere i vagoni merci 1, 2, 5 e 6 nel binario superiore e i vagoni merci 3, 4, 7 e 8 nel binario inferiore, ed estrarre prima i vagoni merci dal binario inferiore e poi estrarre i vagoni merci dal binario superiore:

Quindi l'ordine della coppia di prima non è cambiato, siccome una coppia è sempre spinta nello stesso binario. Inoltre, solo i vagoni merci da 1 a 4 e 5 a 8 sono ordinati relativamente tra loro, ma i due gruppi sono ancora mischiati.
Infine, basta spingere i vagoni da 1 a 4 nel binario superiore e i vagoni da 5 a 8 nel binario inferiore ed estrarre prima i vagoni dal binario inferiore e poi i vagoni dal binario superiore:



L'ordine di entrambi i gruppi non viene cambiato, siccome tutti i vagoni merce del gruppo dal 1 al 4 sono spinti in un binario e tutti i vagoni merce del gruppo dal 5 al 8 sono spinti sull'altro. Ora entrambi i gruppi sono costituiti da vagoni merci ordinati e tutti i vagoni merci in un gruppo hanno un numero inferiore a quello dei vagoni merci dell'altro gruppo.
Gli otto vagoni non si possono ordinare più velocemente. Una dimostrazione completa qui sarebbe troppo dispendiosa, ma l'idea di base è questa: In un processo di spinta si può soltanto cambiare la posizione di un sottoinsieme rispetto agli altri sottoinsiemi, ma non all'interno del sottoinsieme stesso. Perciò in un primo processo di spinta possono essere ordinati al massimo due vagoni merci in posizioni sfavorevoli. Ogni altro processo di spinta raddoppia il numero di vagoni merci in posizioni sfavorevoli che possono essere ordinati. Gli otto vagoni merci nel problema sono scelti in modo che siano in un ordine sfavorevole, quindi non sono sufficienti due processi di spinta.

Questa è l'informatica!

Ferrovieri di tutto il mondo devono risolvere un problema simile quotidianamente, poiché smistare i vagoni merci è un lavoro che richiede molto tempo e manodopera: i vagoni merci devono essere accoppiati e disaccoppiati ogni volta, il che è ancora lavoro manuale. Ciò costa tempo e blocca la

linea principale, soprattutto quando alcuni vagoni merci devono essere fissati e disaccoppiati sulla linea principale. Quindi i ferrovieri hanno sviluppato già molto presto grandi stazioni di smistamento con molti binari morti. In svizzera ci sono stazioni di smistamento a Muttenz vicino a Basilea, a Buchs SG, tra Spreitenbach e Dietikon vicino a Zurigo, a Denges vicino a Losanna e a Chiasso. In questo problema la stazione di smistamento ha solo due binari morti, una sfida per grandi treni merci, ma una situazione tipica per le linee ferroviarie secondarie, soprattutto per le ferrovie a scartamento ridotto che non hanno collegamenti diretti con le grandi compagnie ferroviarie.
Al fine di ordinare treni merci in modo efficiente, l'informatica può aiutare molto. In questo caso il principio di risolvere lo stesso problema più e più volte semplifica enormemente il problema: un metodo che è conosciuto come "dividi e domina" (ingl. *divide & conquer*) nell'informatica. In questo caso vengono prima ordinati due vagoni merci, poi quattro vagoni merci e poi otto vagoni merci.
I binari morti per i vagoni merci funzionano come una *pila*, un tipo di dati astratto, il quale viene applicato intensivamente nell'informatica. Le uniche operazioni permesse sono: *rimuovere l'elemento superiore* (ingl. *pop*) e *inserire un elemento da sopra* (ingl. *push*). A volte nelle pile si può anche guardare *l'elemento superiore* (ingl. *top*) e guardare *se la pila è vuota* (ingl. *empty*).

Parole chiave e siti web

Dividi e impera (Divide & Conquer), Pila

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

A. Autori dei quesiti

Tony René Andersen
Michelle Barnett
Michael Barot
Wilfried Baumann
Jan Berki
Špela Cerar
Mony Chanroath
Marios Choudary
Anton Chukhnov
Kris Coolsaet
Allira Crowe
Christian Datzko
Maria Suyana Datzko
Susanne Datzko
Guillaume de Moffarts
Lanping Deng
Gerald Futschek
Sonali Gogate
Arnheiður Guðmundsdóttir
Martin Guggisberg
Juraj Hromkovič
Alisher Ikramov
Thomas Ioannou
Takeharu Ishizuka
Anna Laura John
Mile Jovanov
Ungyeol Jung
Injoo Kim
Jihye Kim
Mária Kiss
Sophie Koh
Dennis Komm
Bohdan Kudrenko
Regula Lacher
Inggriani Liem
Judith Lin
Violetta Lonati
Mattia Monga
Samart Moodleah
Madhavan Mukund
Tom Naughton
Pia Niemelä
Tomohiro Nishida
Assylkan Omashev
Zsuzsa Pluhár
Sergei Pozdniakov
Nol Premasathian
J.P. Pretti
Milan Rajković
Chris Roffey
Andrea Schrijvers
Eljakim Schrijvers
Humberto Sermeno
Daigo Shirai
Jacqueline Staub
Nikolaos Stratis
Maciej M. Sysło
Bundit Thanasopon
Nicole Trachsler
Jiří Vaníček
Florentina Voboril
Michael Weigend
Jing-Jing Yang
Xing Yang
Khairul A. Mohamad Zaki

B. Sponsoring: concorso 2019

http://www.haslerstiftung.ch/

http://www.roborobo.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.bbv.ch/

http://www.presentex.ch/

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

http://www.diartis.ch/
Diartis AG

https://educatec.ch/
educaTEC

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

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

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

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

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

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

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

C. Ulteriori offerte

Diventate membri della SSII `http://svia-ssie-ssii.ch/verein/mitgliedschaft/` sostenendo in questo modo il Castoro Informatico.
Chi insegna presso una scuola dell'obbligo, media superiore, professionale o universitaria in Svizzera può diventare membro ordinario della SSII.
Scuole, associazioni o altre organizzazioni possono essere ammesse come membro collettivo.