INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Quesiti e soluzioni 2020

9º e 10º anno scolastico

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

A cura di:

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

0101001101010110010010010000001001011010101001101010010100100101000010100101

SSII

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

Hanno collaborato al Castoro Informatico 2020

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

Capo progetto: Nora A. Escherle

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

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

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

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

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

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

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

© Castoro Informatico 2020, SSII

INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Il Castoro Informatico 2020 è stato organizzato dalla Società Svizzera per l'Informatica nell'Insegnamento SSII con il sostegno della fondazione Hasler.

HASLERSTIFTUNG

Questo quaderno è stato creato il 9 settembre 2021 con il sistema per la preparazione di testi LaTeX. Ringraziamo Christian Datzko per lo sviluppo del sistema di generazione dei testi che ha permesso di generare le 36 versioni di questa brochure (divise per lingua e livello scolastico). Il sistema è stato riprogrammato basandosi sul sistema precedente, sviluppato nel 2014 assieme a Ivo Blöchliger. Ringraziamo Jean-Philippe Pellet per lo sviluppo del sistema `bebras`, utilizzato dal 2020 per la conversione dei documenti sorgente dai formati Markdown e YAML.

Nota: Tutti i link sono stati verificati l'01.12.2020.

I quesiti sono distribuiti con Licenza Creative Commons Attribuzione – Non commerciale – Condividi allo stesso modo 4.0 Internazionale. Gli autori sono elencati a pagina 58.

Premessa

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

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

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

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

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

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

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

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

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

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

© Castoro Informatico 2020, SSII

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/`

© Castoro Informatico 2020, SSII

Indice

© Castoro Informatico 2020, SSII

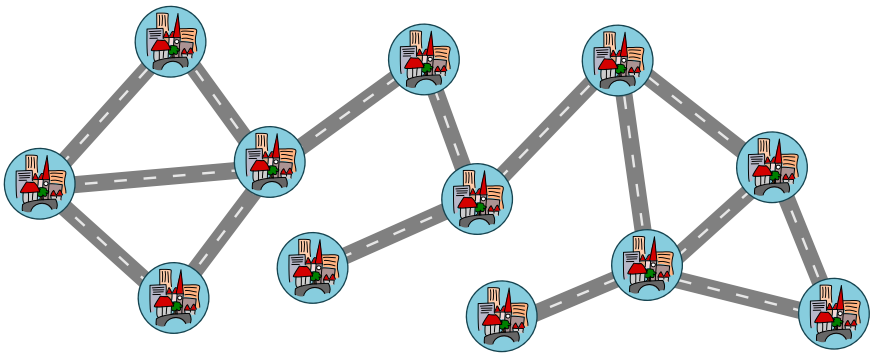
1. Considerazioni epidemiologiche

Biberland è composta da 12 città, che sono collegate da strade. Le città che sono direttamente o indirettamente collegate da strade formano una comunità commerciale. La mappa mostra quindi nella sua forma attuale un'unica comunità commerciale di 12 città.

Per contenere un'epidemia, il traffico deve essere ridotto. Il parlamento di Biberland decide di chiudere esattamente due strade per dividere le città in tre comunità commerciali separate.

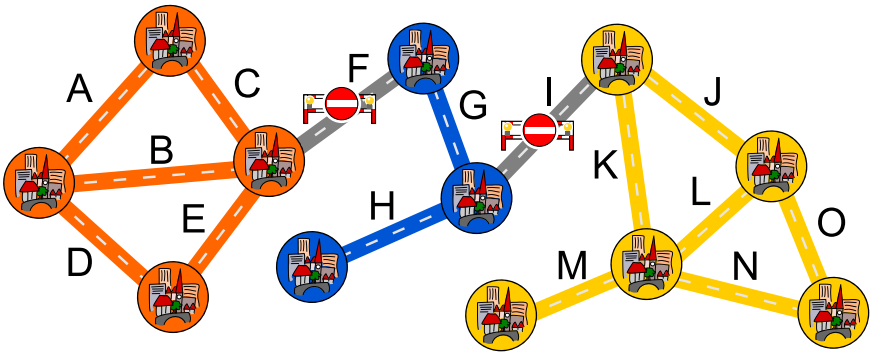
Per non isolare nessuno più del necessario, la più piccola comunità commerciale dovrebbe essere composta dal maggior numero possibile di città.

Quali due strade dovrebbero essere chiuse? Indicale.



Soluzione

La risposta corretta è: le strade F e I nella foto qui sotto devono essere bloccate. Questo crea comunità commerciali di 3, 4 e 5 città.



E' ovvio che dobbiamo guardare solo alle strade che, se sono bloccate, causeranno anche la divisione della comunità commerciale perché sono l'unico collegamento. Dopo tutto, abbiamo bisogno di due vere e proprie divisioni per raggiungere tre unità. Ad esempio, non ha senso chiudere la strada B, perché si possono comunque raggiungere tutte le città via A e C. Rimangono quindi solo le candidate F, G, H, I e M per il blocco.

Se si provano tutte le 10 possibilità di chiudere due delle cinque strade, si otterrà la risposta di cui sopra. Come essere umano, si vede subito che il blocco H o M taglierebbe solo una singola città ed è quindi fuori questione. Ciò limita ulteriormente il numero di possibilità da prendere in considerazione.

Questa è l'informatica!

Nell'informatica, una data rete è spesso suddivisa in cosiddette *componenti connesse*. In una componente connessa, tutte le parti sono collegate tra loro attraverso percorsi diretti o indiretti, mentre non c'è alcun collegamento tra le diverse componenti connesse. Ovviamente, l'applicazione è in reti di computer dove è rilevante quali computer possono essere raggiunti da quali altri. Ma anche, ad esempio, nel riconoscimento ottico dei caratteri (OCR), è importante sapere quali punti sono «connessi».

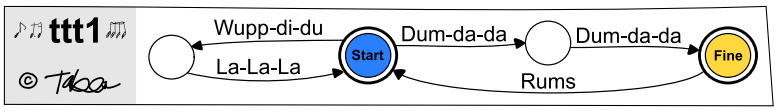
Parole chiave e siti web

- Componente connessa:
  `https://it.wikipedia.org/wiki/Componente_connessa_(teoria_dei_grafi)`

 © Castoro Informatico 2020, SSII

2. Il ritmo di Tabea

Tabea ha molto successo nel creare testi di canzoni con il marchio ttt: Tabea's Tactful Texts. I testi possono essere prodotti con il seguente diagramma ttt1:



Per produrre una canzone, Tabea inizia da «Start» (Start) e segue una delle frecce in uscita. Se ci sono diverse possibilità, può scegliere quella che preferisce. Canta le sillabe corrispondenti lungo il percorso nell'ordine dato. Se raggiunge «Fine» (Fine), la canzone può finire ma può anche continuare.

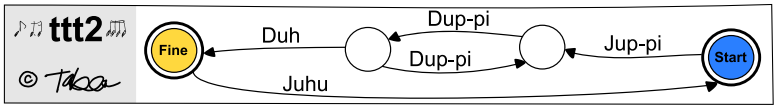
Possibili canzoni possono essere:

«Wupp-di-du La-La-La Wupp-di-du La-La-La
Dum-da-da Dum-da-da Rums Dum-da-da Dum-da-da»

Oppure

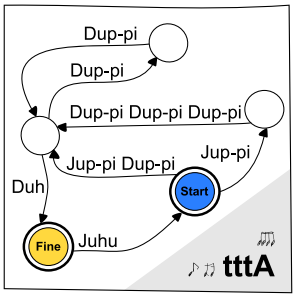
«Dum-da-da Dum-da-da Rums Wupp-di-du La-La-La
Dum-da-da Dum-da-da Rums Wupp-di-du La-La-La
Dum-da-da Dum-da-da Rums Dum-da-da Dum-da-da»

Nel novembre 2020 Tabea produce nuovi testi con il diagramma ttt2:

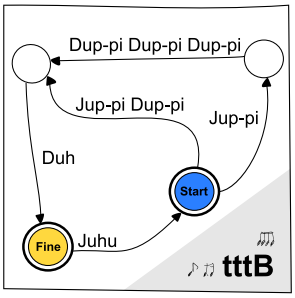


Con quali dei seguenti diagrammi si possono creare esattamente gli stessi testi come con il diagramma ttt2?

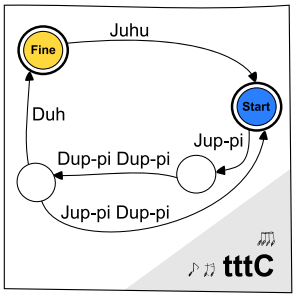
A)



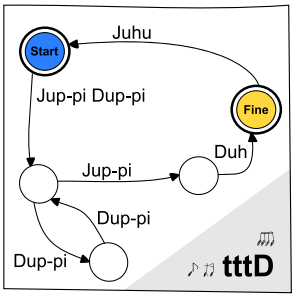
B)



C)

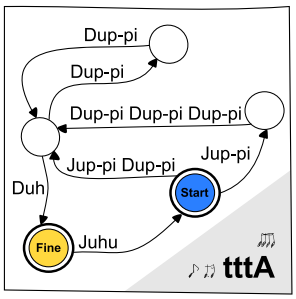


D)

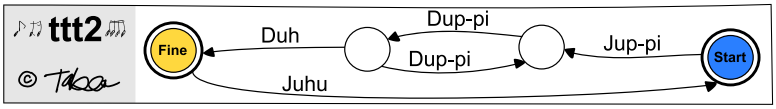


Soluzione

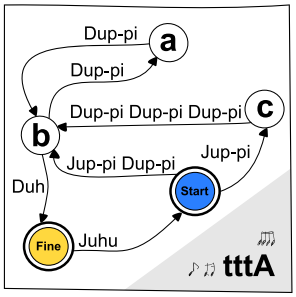
La risposta corretta è A) ossia il diagramma tttA.



Se si produce un brano con il diagramma ttt2, esso inizierà sempre con «Jup-pi» e seguirà almeno un «Dup-pi». Ora continua o direttamente con «Duh» o con un numero pari di ulteriori «Dup-pi» e poi «Duh». Ora la canzone può essere terminata o continuare con un «Juhu» e ricominciare dall'inizio.



Il diagramma tttA raggiunge esattamente lo stesso risultato: Dallo «Start», il brano può partire direttamente in b e quindi con «Jup-pi Dup-pi» o via c con «Jup-pi Dup-pi Dup-pi Dup-pi Dup-pi Dup-pi». Dopo di che, una deviazione tramite a permette di aggiungere un numero pari qualsiasi di «Dup-pi», poi si può usare «Duh» per arrivare alla fine della canzone. Proprio come in ttt2 si può ricominciare dopo «Juhu».



Sia il diagramma ttt2 che il diagramma tttA possono produrre dopo «Jup-pi» un numero dispari a volontà di «Dup-pi». Il diagramma tttB invece può produrre solo 1 o 3 «Dup-pi» di fila e il diagramma tttC solo 1 o 2. Il diagramma tttD crea invece un numero dispari di Dup-pi, ma precede il «Duh» finale con un altro «Jup-pi», che ttt2 non può creare. Quindi tttA è l'unica risposta possibile.

Questa è l'informatica!

Un compito importante dell'informatica è quello di riconoscere le strutture nei dati, ad esempio nel linguaggio, come il testo di una canzone. I diagrammi rappresentano i cosiddetti automi finiti con i quali si possono definire regole molto severe per la generazione e il riconoscimento di determinate lingue. Questo a sua volta è cruciale nello sviluppo dei linguaggi di programmazione. Nel nostro esempio,

l'automa finito descrive l'insieme di canzoni che possono essere create con esso. Il riconoscimento del modello è importante anche in molti altri settori, come l'elaborazione del linguaggio naturale.

Parole chiave e siti web

- Automi finiti: `https://it.wikipedia.org/wiki/Automa_a_stati_finiti_deterministico`
- Linguaggi formali: `https://it.wikipedia.org/wiki/Linguaggio_formale`
- `https://sites.google.com/isabc.ca/computationalthinking/pattern-recognition`

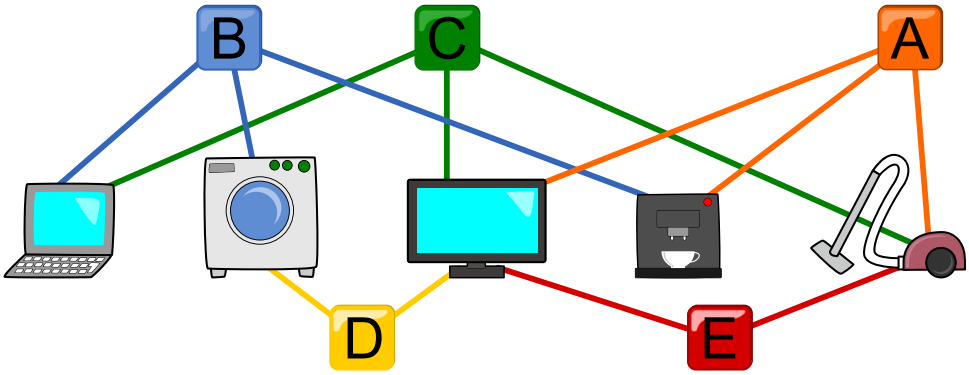
© Castoro Informatico 2020, SSII

3. Elettrodomestici

Nella casa del castoro Bruno ci sono cinque elettrodomestici (computer, lavatrice, televisione, macchina per il caffè e aspirapolvere) e cinque pulsanti (A, B, C, D ed E) per accendere e spegnere. Tuttavia, il cablaggio è molto insolito. Ogni pulsante è collegato a diversi dispositivi, come mostrato nella figura sotto. Ogni volta che si preme un tasto, esso commuta tutti i dispositivi collegati: Quelli che sono spenti vengono accesi e quelli che sono accesi vengono spenti.

All'inizio tutti gli apparecchi sono spenti. Ad esempio, se si premono i pulsanti A, C ed E, l'aspirapolvere si accende perché il primo pulsante lo accende, il secondo lo spegne e il terzo lo riaccende.

Quali pulsanti deve premere Bruno affinché alla fine si accendano solo il televisore e la macchina del caffè?



Soluzione

Se si premono i pulsanti B, C, D, E (in qualsiasi ordine), si accendono solo il televisore e la macchina del caffè.

Possiamo anche scoprire sistematicamente come accendere e spegnere ogni apparecchio separatamente. Iniziamo con due semplici combinazioni:

- A + E (premendo A ed E) si comanda la macchina del caffè da sola.
- C + E (premendo C ed E) si comanda il computer da solo.

Osserviamo poi che la lavatrice può essere comandata individualmente premendo prima B e poi riportando immediatamente il computer e la macchina da caffè al punto di partenza premendo A + E e C + E. Così, tutto sommato, la lavatrice è controllata individualmente da B + A + E + C + E. Qui E appare due volte. Premere due volte lo stesso interruttore è come non averlo premuto affatto. Pertanto, la lavatrice può essere comandata anche singolarmente da B + A + C. Con questo metodo si ottiene la seguente lista di combinazioni di pulsanti per il controllo dei singoli apparecchi:

- Computer: C + E
- Macchina del caffé: A + E
- Lavatrice: A + B + C
- Televisione: A + B + C + D
- Aspirapolvere: A + B + C + D + E

Per accendere la televisione e la macchina del caffè, dobbiamo quindi premere A + B + C + D + A + E, il che semplifica a B + C + D + E, in quanto le due A si annullano a vicenda.

Questa è l'informatica!

Il sistema di dispositivi e pulsanti per l'accensione e lo spegnimento può essere modellato come un cosiddetto *automa a stati finiti*. Questo avviene come segue.

Il sistema dei cinque dispositivi ha molti *stati* diversi. Per esempio, uno stato è quando è acceso solo il televisore. Un altro stato è quando tutti gli apparecchi sono spenti (poiché tutti gli apparecchi sono spenti all'inizio, lo chiamiamo lo *stato iniziale*). E un altro stato è quello in cui sono accese solo la TV e la macchina del caffè (nel nostro esempio questo è lo *stato finale* perché è lo stato che vogliamo).

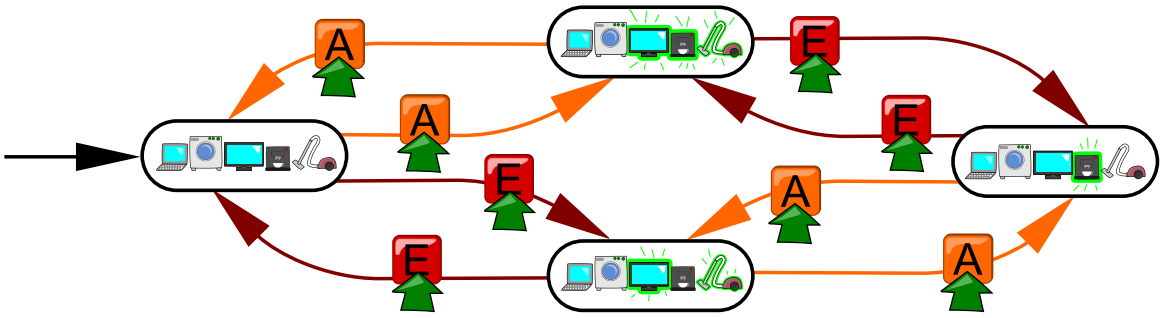
Premendo un pulsante si sposta il sistema da uno stato all'altro.

Per esempio: Se il sistema è nello stato iniziale, premendo E si passa allo stato in cui sono accesi solo il televisore e l'aspirapolvere. Un tale cambiamento di stato è anche chiamato *transizione*.

Se si disegnano gli stati del sistema come cerchi e le transizioni come frecce, si ottiene un'immagine come quella qui sotto (per ragioni di spazio, sono disegnati solo quattro stati e solo le transizioni tra di essi.) Lo stato iniziale è contrassegnato da una freccia speciale. In informatica questo si chiama

 © Castoro Informatico 2020, SSII

automa a stati finiti (a proposito, un automa a stati finiti è semplicemente un grafo speciale; gli stati sono i *nodi* e le transizioni sono gli *archi*). Nella foto, ora possiamo facilmente vedere in quale stato ci troviamo quando vengono premuti diversi pulsanti.



Il compito consiste nel passare dallo stato iniziale (tutti i dispositivi spenti) allo stato di destinazione (solo TV e macchina del caffè accesa). Quindi si tratta di trovare un modo per passare dallo stato iniziale allo stato di destinazione. Trovare percorsi nei grafi è un compito fondamentale dell'informatica.

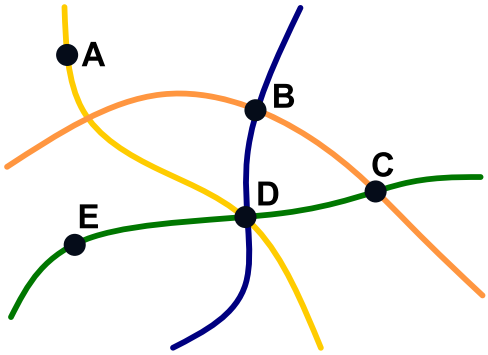
Parole chiave e siti web

- Automa a stati finiti: `https://it.wikipedia.org/wiki/Automa_a_stati_finiti`

© Castoro Informatico 2020, SSII

4. Rete ferroviaria

Questa è una mappa di 5 città e 4 linee ferroviarie. I punti neri sono le città, le linee colorate sono linee ferroviarie.

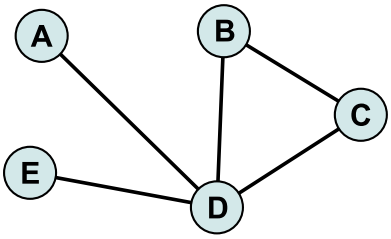


Un diagramma dovrebbe rappresentare questa mappa in modo tale che:

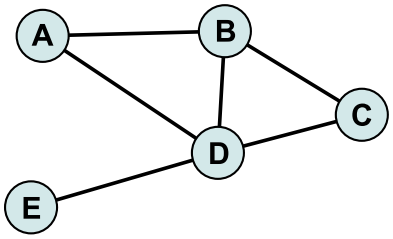
- le città sono rappresentate da cerchi, e
- due città sono collegate da una linea solo quando si trovano sulla stessa linea ferroviaria.

Quale diagramma visualizza correttamente la mappa?

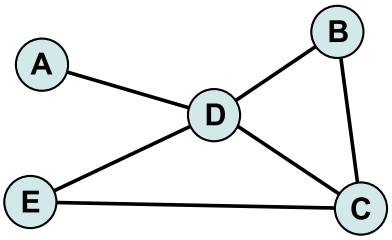
A)



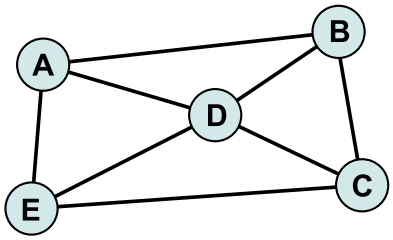
B)



C)

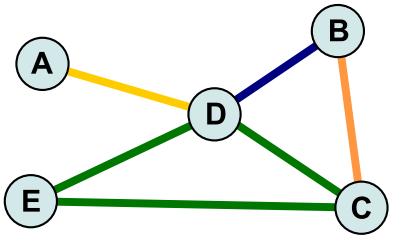


D)



Soluzione

La risposta corretta è C).



Un'analisi più approfondita della mappa dimostra che:

- Le città A e D si trovano insieme sulla linea ferroviaria gialla,
- le città B e C sono insieme sulla linea ferroviaria arancione,
- le città B e D si trovano insieme sulla linea ferroviaria blu, e
- le città C, D ed E si trovano insieme sulla linea ferroviaria verde.

Tutte le altre risposte sono sbagliate:

- Nella risposta A, manca la linea tra le città C ed E, che deve esistere a causa della linea ferroviaria verde.
- La risposta B ha lo stesso problema della risposta A e in aggiunta c'è una linea tra le città A e B, anche se non sono insieme sulla stessa linea ferroviaria.
- Nella risposta D, ci sono due linee dalla città A alla città B e dalla città A alla città E, anche se la città A non è su una linea ferroviaria comune con la città B o la città E.

I due punti seguenti meritano un'attenzione particolare:

- Anche se si può andare dalla città A alla città B se si utilizzano varie linee ferroviarie, le due città non sono sulla stessa linea ferroviaria.
- Anche se c'è una terza città sulla linea ferroviaria verde tra C ed E, C ed E sono ancora sulla stessa linea ferroviaria.

Questa è l'informatica!

Ci sono molti modi diversi di rappresentare la realtà. Ad esempio, la mappa qui sopra con le linee ferroviarie colorate è già una rappresentazione piuttosto astratta della situazione reale. Un tipo di rappresentazione molto importante è un *grafo* - un diagramma costituito da *nodi* (piccoli cerchi) e *archi* (linee tra i nodi). Questo tipo di rappresentazione viene utilizzato nella soluzione.

Molte cose diventano più facili se si sceglie un buon metodo di rappresentazione. Pertanto, è importante conoscere molte modalità di visualizzazione durante la programmazione. Spesso non è possibile affermare che una modalità di visualizzazione sia migliore dell'altra. A seconda dell'applicazione, l'uno o l'altro è più adatto. Ad esempio, il grafo della soluzione è pratico perché si può vedere direttamente che si può andare da C a E con una sola linea ferroviaria. Rispetto alla mappa, tuttavia,

si perde l'informazione che con questa linea ferroviaria si passa per la città D sulla strada dalla città C alla città E.

Parole chiave e siti web

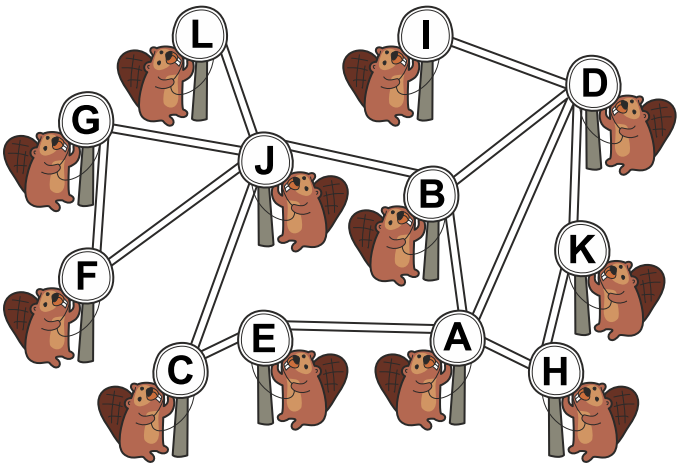
- Grafo: `https://it.wikipedia.org/wiki/Grafo`
- Teoria dei grafi: `https://it.wikipedia.org/wiki/Teoria_dei_grafi`

© Castoro Informatico 2020, SSII

5. Rete di comunicazione

Ai castori piace diffondere notizie tra di loro. A tale scopo utilizzano la rete di comunicazione qui sotto. Quando un castoro riceve un nuovo messaggio, lo inoltra a tutti coloro con cui è collegato da un canale di comunicazione diretta (una linea bianca). L'invio dei messaggi si effettua a turni. C'è sempre un turno tra l'invio e la ricezione.

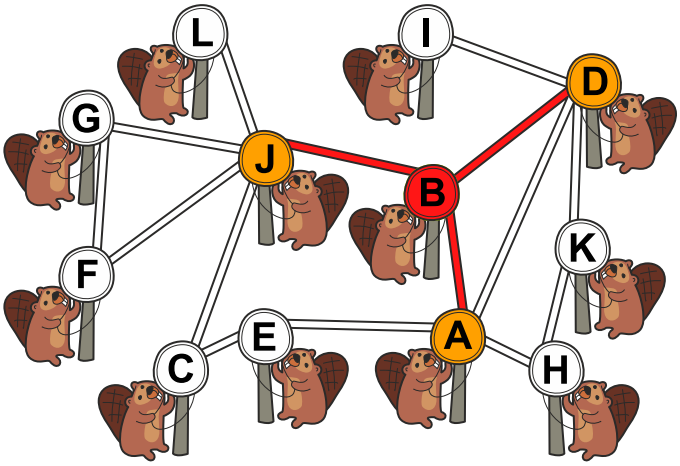
Da quale castoro un messaggio raggiunge tutti gli altri castori più velocemente, cioè nel minor numero di turni?



Soluzione

La risposta corretta è il castoro B. Può diffondere un messaggio a tutti gli altri castori in due turni.

Nel primo turno, il castoro B invia il messaggio ai suoi vicini, cioè i castori A, D e J collegati ad un canale di comunicazione diretta. L'immagine sottostante mostra chi ha il messaggio dopo questo primo turno.

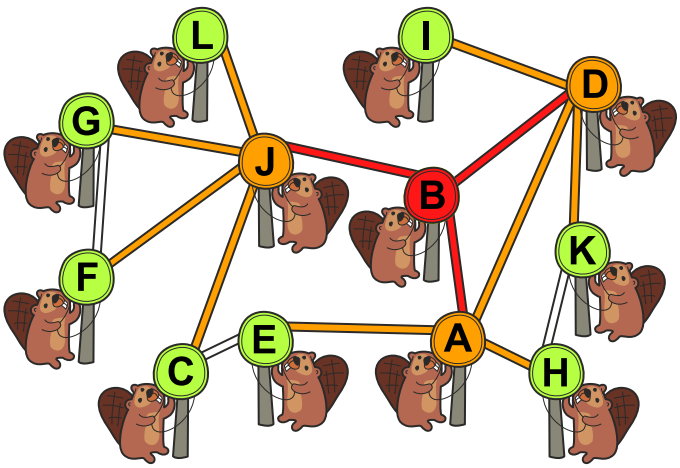


Nel secondo turno i castori A, D e J inviano il messaggio ai loro vicini: - Il castoro A invia il messaggio ai castori E e H;

- Il castoro D invia il messaggio ai castori I e K;
- Il castoro J invia il messaggio ai castori C, F, G e L.

Inoltre, il castoro B riceve il messaggio tre volte, perché anche lui è un vicino dei tre castori A, D e J. Poiché questo non è un messaggio nuovo per lui, il castoro B non lo invierà nei prossimi turni. Anche i castori A e D si invieranno di nuovo il messaggio attraverso il loro canale di comunicazione diretta, ma dopo di che non lo invieranno più perché non è più nuovo per loro.

L'immagine sottostante mostra la situazione dopo il secondo turno.



Così la notizia ha raggiunto tutti i castori in soli due turni.

Non c'è un modo più veloce, perché altrimenti un castoro dovrebbe essere collegato a tutti gli altri castori con una linea bianca per inviare il messaggio direttamente a tutti gli altri castori in un unico turno.

Il castoro B è anche l'unico dal quale un messaggio raggiunge tutti gli altri castori in soli due turni: per i castori C, E, F, G, H, J e L, il castoro I non sarebbe raggiungibile in due turni. E per i castori A, D, E, E, H, I e K, il castoro L non può essere raggiunto in due turni.

Questa è l'informatica!

La rete di comunicazione dei castori può essere descritta da un *grafo*. Ogni castoro si trova in un cosiddetto *nodo*, che in questo caso è nominato con una lettera. Le linee bianche sono chiamate *archi*, ognuna di esse collega due nodi. I messaggi si diffondono nella rete di comunicazione attraverso turni *sincronizzati*, cioè tutti i castori inviano contemporaneamente. In un turno, ogni castoro invia nuovi messaggi a tutti i suoi vicini. Quello che i castori fanno qui è quello che gli informatici chiamano *broadcasting* (inglese per «trasmettere») in una *rete di comunicazioni*. Nel compito di cui sopra, abbiamo studiato la velocità con cui una tale trasmissione può essere completata, cioè la velocità con cui un nuovo messaggio può raggiungere tutti i partecipanti.

Un compito ancora più complesso è quello di progettare le reti in modo tale che sia possibile una trasmissione veloce da tutti i nodi, ma con un numero di connessioni non troppo elevato. Il nodo del ricercato castoro B è chiamato poi il centro del grafo. In astratto, il centro è un nodo che riduce al minimo la distanza dai nodi più lontani. Non c'è quindi nessun altro nodo che avrebbe una distanza minore rispetto a tutti gli altri nodi. Nel presente compito c'è un solo centro. Tuttavia, a seconda del grafo, possono esserci diversi nodi, in modo che ognuno di essi minimizzi la distanza dai nodi più lontani da esso; quindi, un grafo può avere diversi centri.

Trovare un centro non è sempre facile come in questo compito. Per prima cosa, potrebbe essere che ci vogliano diversi turni per il trasferimento tra alcuni nodi direttamente collegati. D'altra parte, i grafi possono essere semplicemente molto più grandi e complessi. Per tali grafi, si potrebbe, ad esempio, utilizzare l'algoritmo di Floyd-Warshall per trovare in modo efficiente un centro.

Parole chiave e siti web

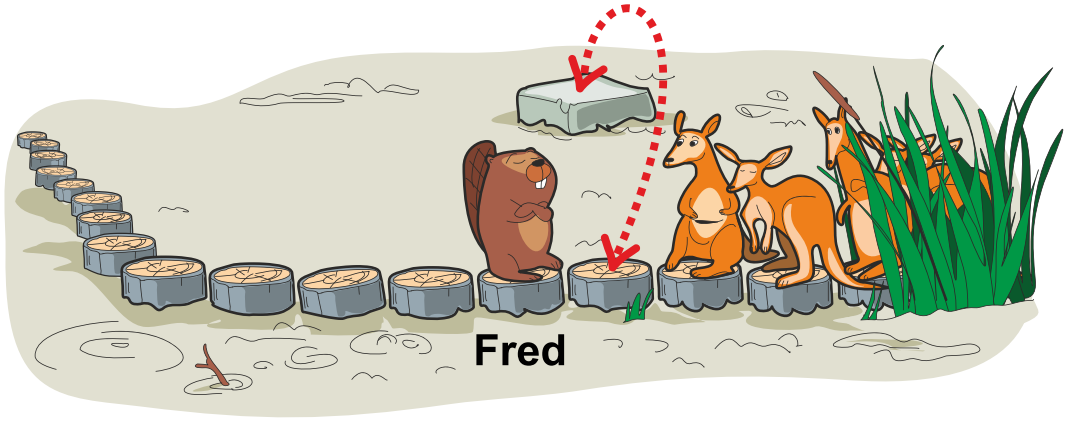
- Grafo: `https://it.wikipedia.org/wiki/Grafo`
- Algoritmo di Floyd-Warshall: `https://it.wikipedia.org/wiki/Algoritmo_di_Floyd-Warshall`

© Castoro Informatico 2020, SSII

6. Il castoro testardo

Il castoro Fred incontra i canguri su un percorso di ceppi di albero. Il percorso è piuttosto stretto, così che lui e i canguri non possano passare allo stesso tempo. Ma c'è uno specifico ceppo di albero dal quale i canguri possono saltare su una pietra e da lì tornare a questo ceppo, come mostrato nella foto. Solo un animale alla volta può stare su ogni ceppo di albero e sulla pietra.

Fred vuole andare avanti. È abbastanza testardo e disposto a tornare indietro di un ceppo solo 10 volte al massimo. In avanti, invece, può andare tutte le volte che vuole.



Qual è il numero massimo di canguri che Fred può far passare?

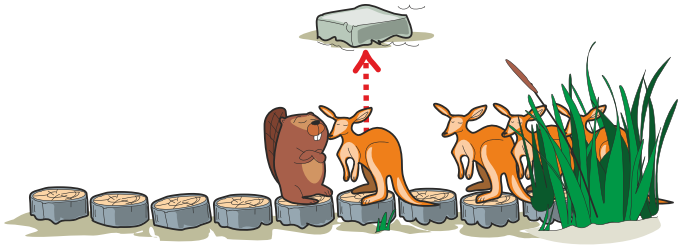
A) Più di 10 canguri.
B) Esattamente 10 canguri.
C) Esattamente 6 canguri.
D) Esattamente 4 canguri.
E) Meno di 4 canguri.
F) È impossibile dirlo con certezza.

Soluzione

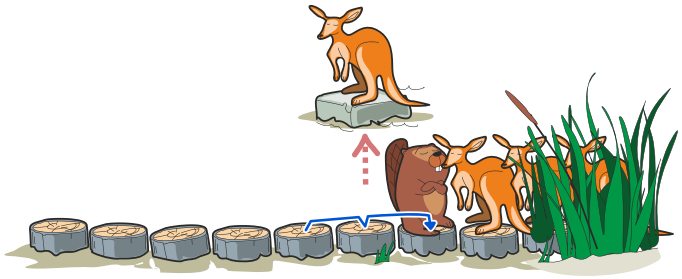
Fred può far passare un massimo di 6 canguri.

Un canguro passa Fred come segue:

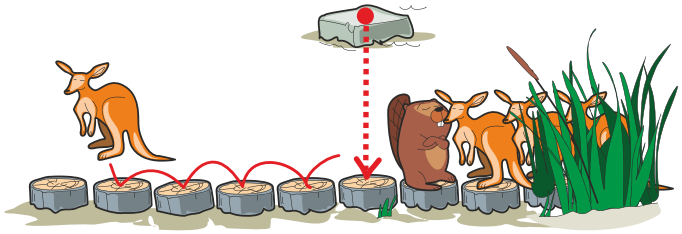
Il canguro salta sulla pietra.



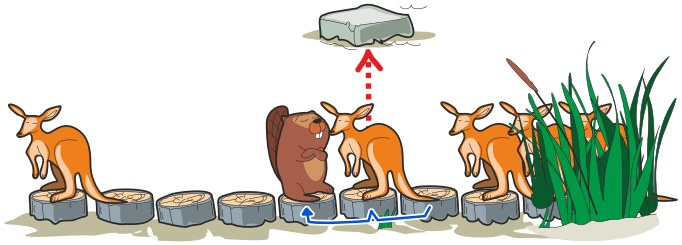
Fred cammina due ceppi d'albero in avanti.



Il canguro salta indietro e continua il suo cammino.



Se Fred ora torna indietro di due ceppi, è tornato alla posizione di partenza e può ripetere la procedura per far passare un altro canguro.



Poiché torna indietro di un massimo di 10 ceppi, può farlo cinque volte e insieme al primo canguro può far passare un massimo di 6 canguri.

Questa è l'informatica!

Nell' informatica i compiti vengono risolti, tra l'altro, con l'ausilio di algoritmi: seguendo semplici *istruzioni* e *comandi* che vengono eseguiti passo dopo passo - proprio come «Fred cammina un ceppo di albero in avanti» o «un canguro salta su una pietra».

In una *iterazione* (o «*loop*» in inglese), le sequenze di istruzioni possono essere ripetute. In questo modo, i compiti uniformi possono essere eseguiti in modo affidabile più volte. Di solito è vantaggioso creare la stessa situazione ad ogni passaggio d'iterazione - la cosiddetta *invariante*. Nel nostro caso

 © Castoro Informatico 2020, SSII

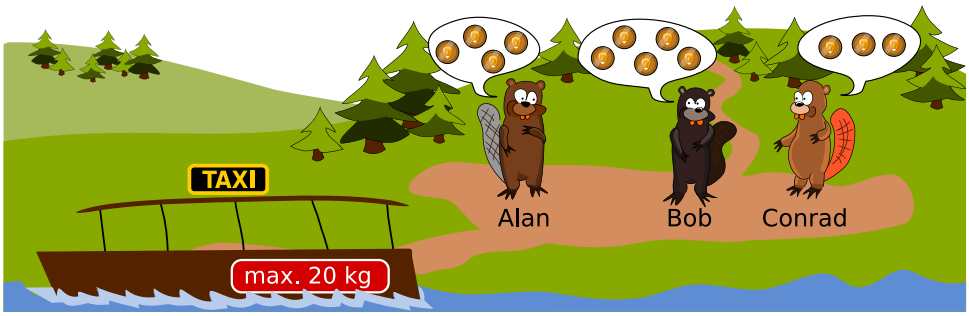
Fred deve tornare alla sua posizione di partenza più e più volte, in modo che la stessa procedura funzioni di nuovo per il canguro successivo.

Parole chiave e siti web

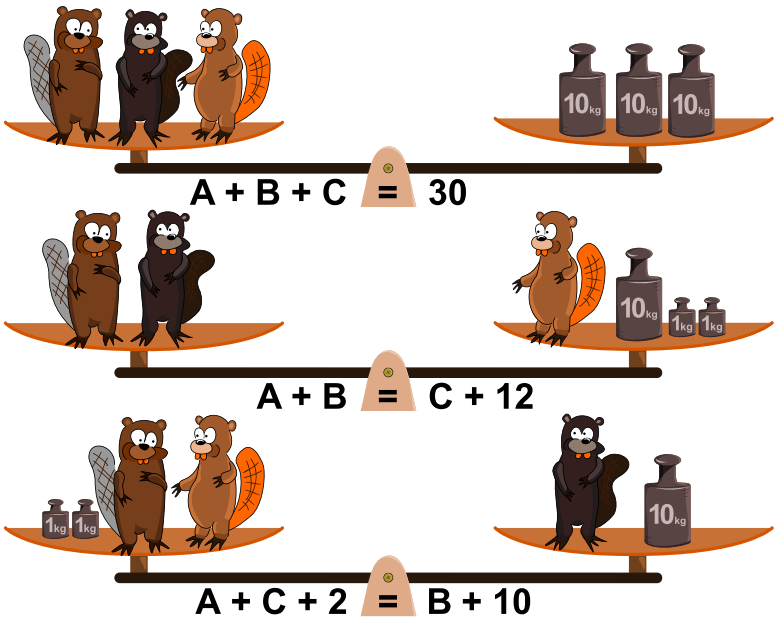
- Algoritmo: `https://it.wikipedia.org/wiki/Algoritmo`
- `https://it.wikipedia.org/wiki/Programmazione_strutturata`
- Iterazione: `https://it.wikipedia.org/wiki/Struttura_di_controllo#Iterazione`

© Castoro Informatico 2020, SSII

7. Taxi acquatico



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



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

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

Soluzione

La risposta corretta è: C) Bob e Conrad

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

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

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

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

Così il tassista può:

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

Pertanto la risposta corretta è C).

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

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

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

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

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

La differenza I. – III. dà:

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

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

 © Castoro Informatico 2020, SSII

Questa è l'informatica!

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

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

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

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

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

Parole chiave e siti web

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

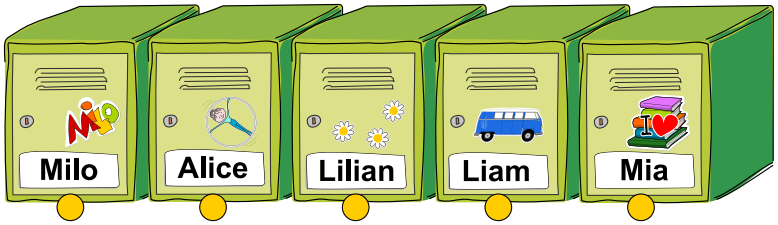
© Castoro Informatico 2020, SSII

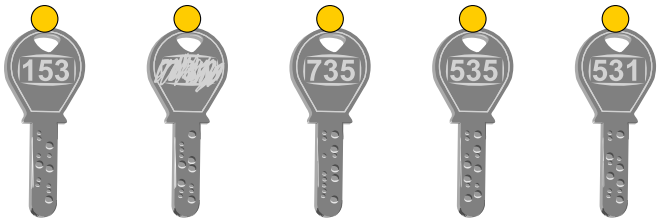
8. Armadietti

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

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

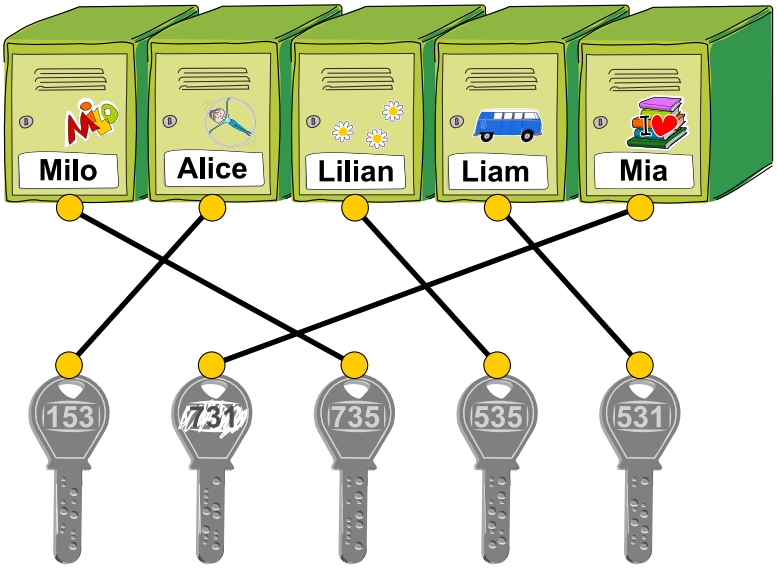
Assegna le chiavi agli armadietti corretti. Traccia delle linee tra i punti gialli.





Soluzione

Di seguito viene mostrata la soluzione corretta:



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

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

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

Questa è l'informatica!

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

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

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

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

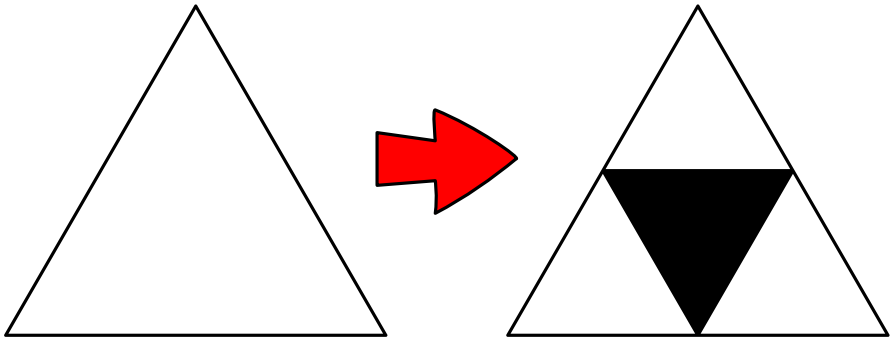
Parole chiave e siti web

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

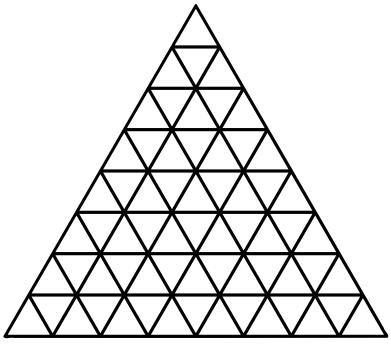
© Castoro Informatico 2020, SSII

9. Triangolo di Sierpiński

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

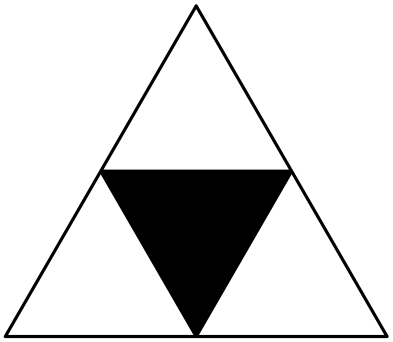


Disegna la figura che emerge dopo tre passi. Colora di nero i triangoli corretti.

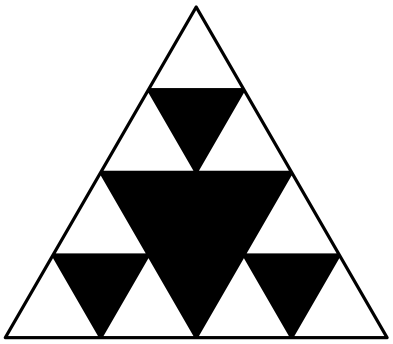


Soluzione

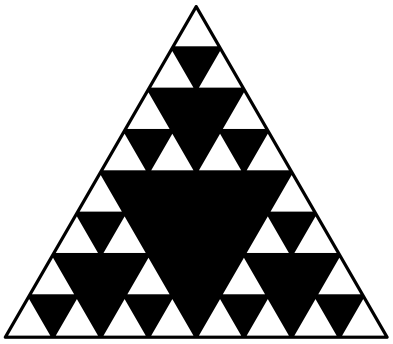
Dopo il primo passo, il triangolo centrale è nero e rimangono tre triangoli bianchi:



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



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



Questa è l'informatica!

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

 © Castoro Informatico 2020, SSII

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

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

Parole chiave e siti web

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

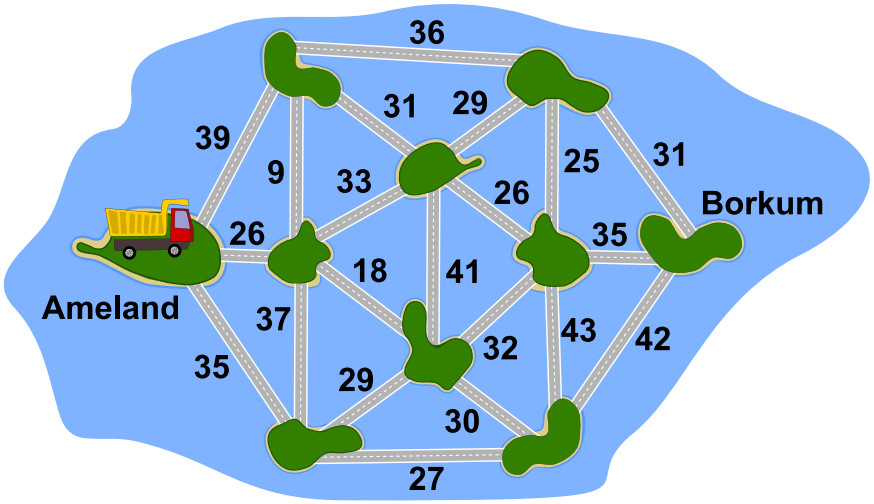
© Castoro Informatico 2020, SSII

10. L'arcipelago dei castori

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

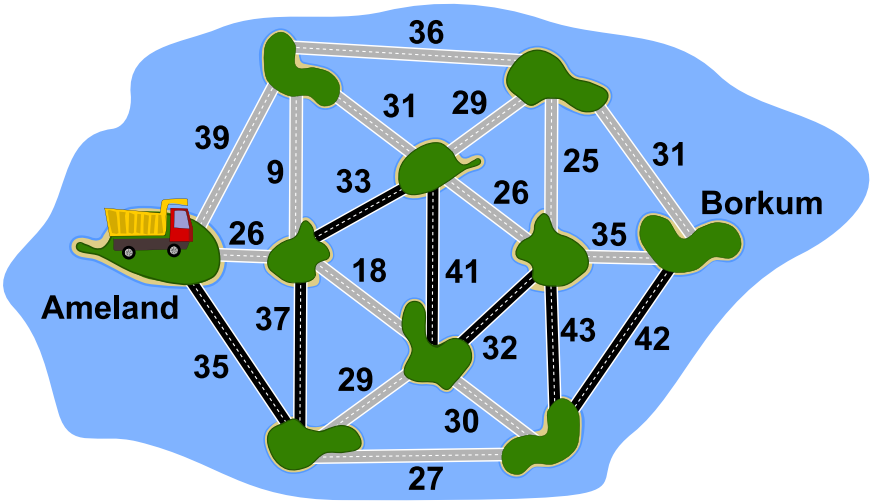
Il castoro Knuth vuole costruire una spiaggia sull'isola di Borkum. Vuole quindi trasportare quanta più sabbia possibile dall'isola di Ameland all'isola di Borkum in un solo viaggio. Non gli interessa la lunghezza del viaggio, ma non vuole passare su nessun ponte più di una volta.

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

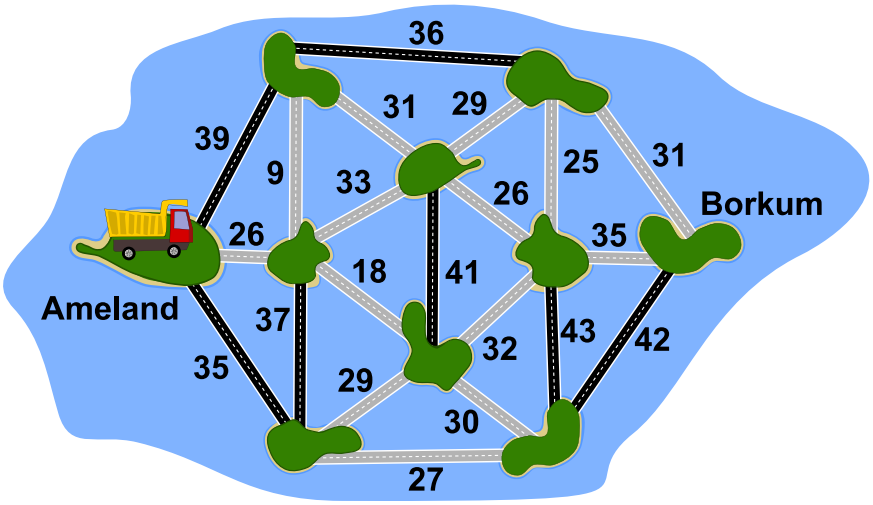


Soluzione

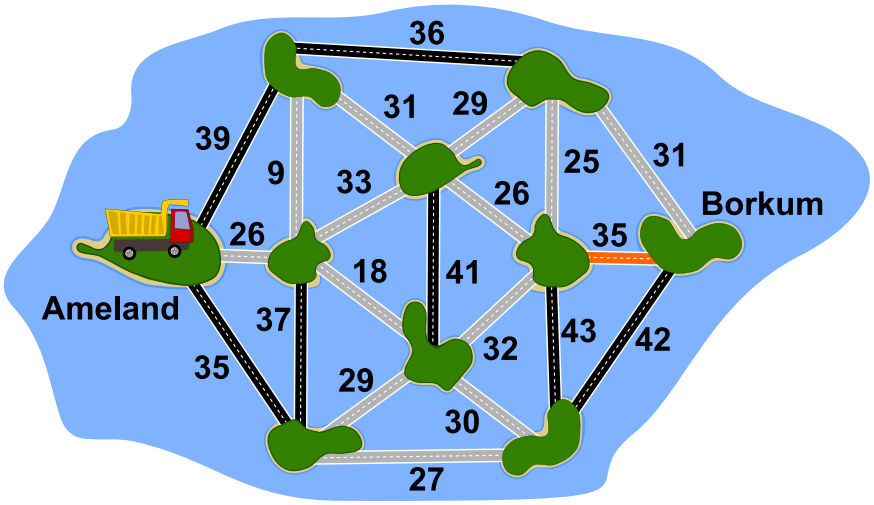
Per il viaggio, il peso totale massimo di un camion è di 32 tonnellate. Si prende il seguente percorso, ad esempio:



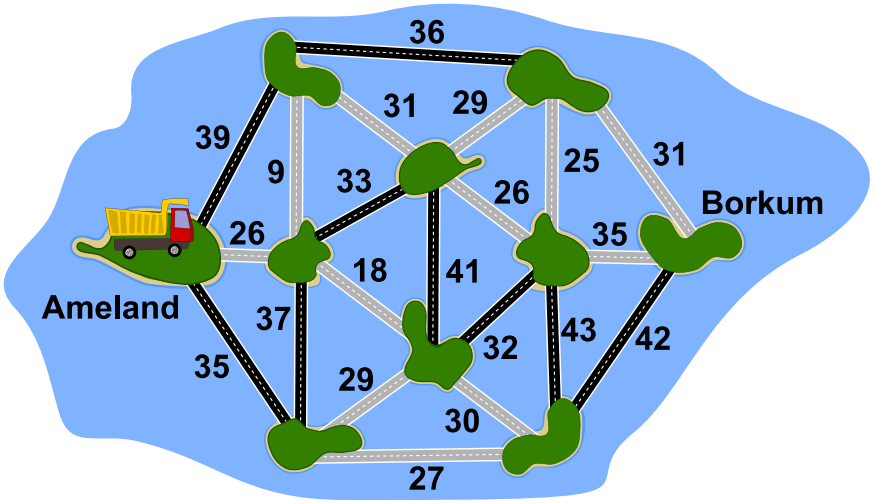
Per determinare questo, possiamo, ad esempio, per prima cosa togliere tutti i ponti dalla mappa e ordinarli in base alla loro capacità di carico. Iniziamo con quelli con la maggiore capacità di carico e li aggiungiamo alla mappa. Poi aggiungiamo quelli con la seconda maggior capacità di carico e così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



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



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



Questa è l'informatica!

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

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

Parole chiave e siti web

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

 © Castoro Informatico 2020, SSII

11. Lavagna rovinata

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

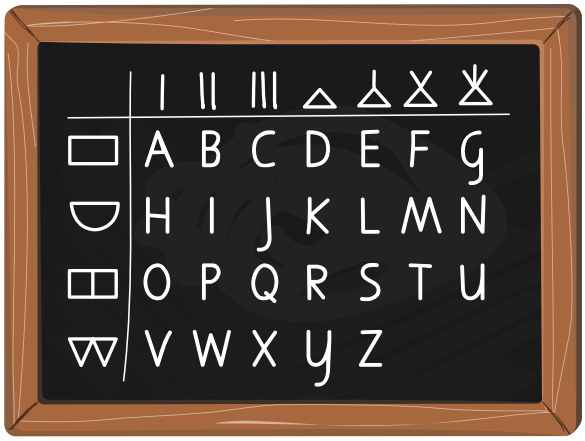
Ricostruisci il testo originale a partire dal testo cifrato attuale (decifra il testo cifrato). Quale delle 4 soluzioni è corretta?

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

Soluzione

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

Ecco la lavagna di cifratura completa:



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

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

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

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

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

Questa è l'informatica!

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

 © Castoro Informatico 2020, SSII

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

Parole chiave e siti web

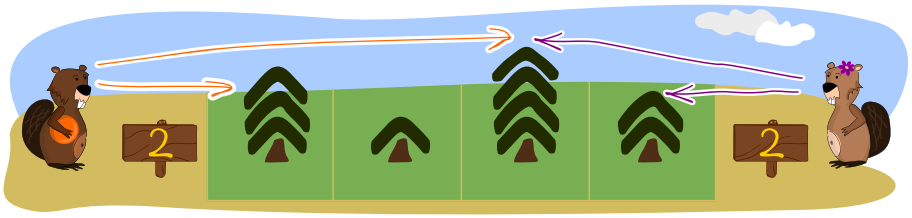
- Crittografia: `https://it.wikipedia.org/wiki/Crittografia`

© Castoro Informatico 2020, SSII

12. 4×4 sudoku con gli alberi

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

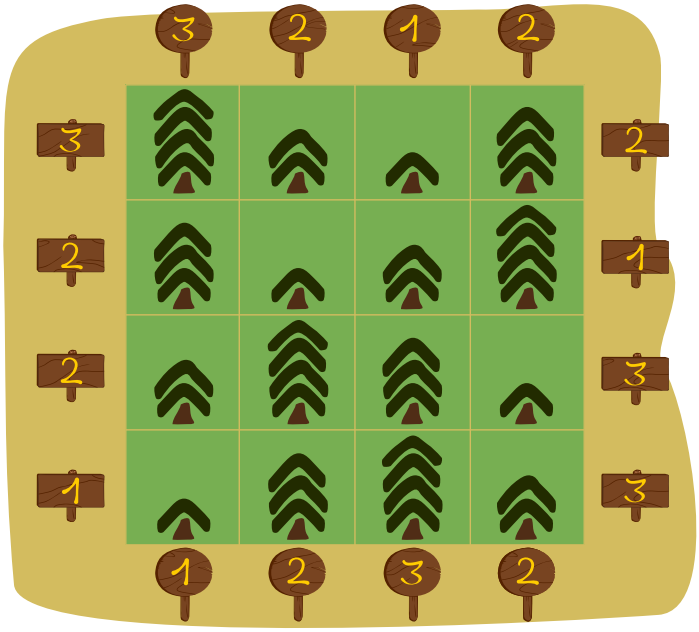
- In ogni riga (riga orizzontale) c'è esattamente un albero di ogni altezza;
- In ogni colonna (riga verticale) c'è esattamente un albero di ogni altezza.



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

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

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



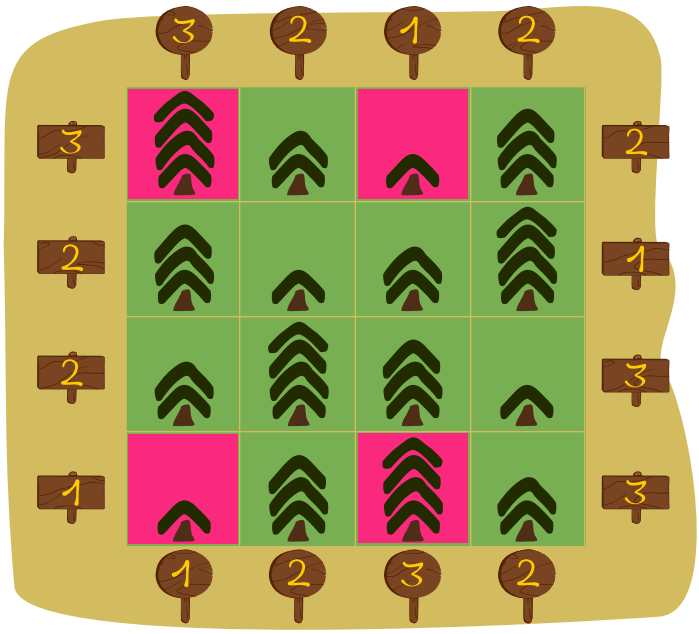
Soluzione

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

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

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

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



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

Questa è l'informatica!

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

 © Castoro Informatico 2020, SSII

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

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

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

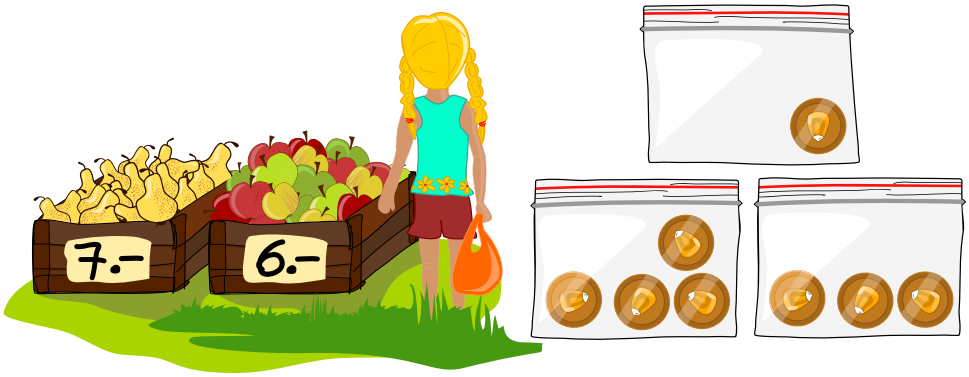
Parole chiave e siti web

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

© Castoro Informatico 2020, SSII

13. Sacchetto per i soldi

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



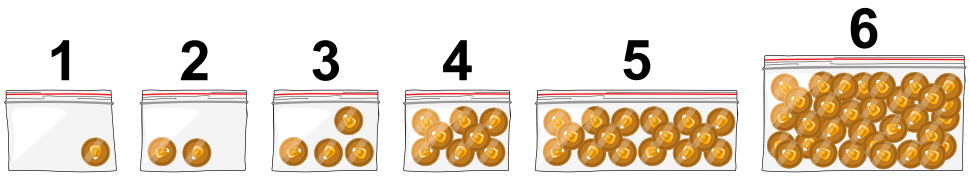
Oggi Bina ha con sé 63 monete identiche. Vuole dividerle in sacchetti diversi in modo da poter pagare qualsiasi importo compreso tra 1 e 63 monete senza dover aprire i sacchetti.

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

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

Soluzione

La risposta corretta è C) 6 sacchetti:



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

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

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

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

 © Castoro Informatico 2020, SSII

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

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

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

Questa è l'informatica!

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

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

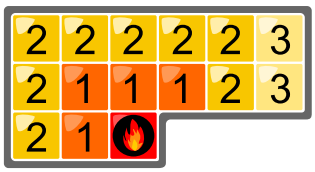
Parole chiave e siti web

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

 © Castoro Informatico 2020, SSII

14. Riscaldamento a pavimento

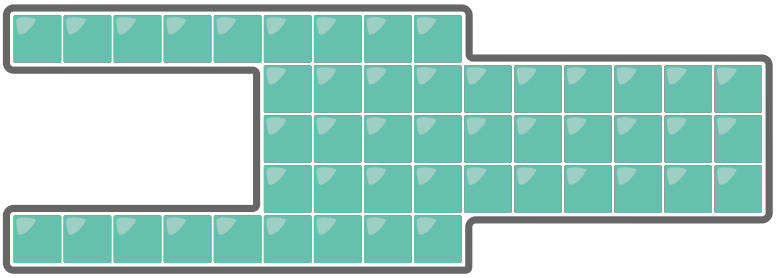
A Luis non piace vestirsi al mattino nel bagno freddo, quindi vuole che nella nuova casa venga installato il riscaldamento a pavimento. Il tecnico del riscaldamento gli consiglia l'innovativo riscaldamento a pavimento «hotspot»: un hotspot viene installato direttamente sotto una piastrella. Se l'hotspot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



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

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

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

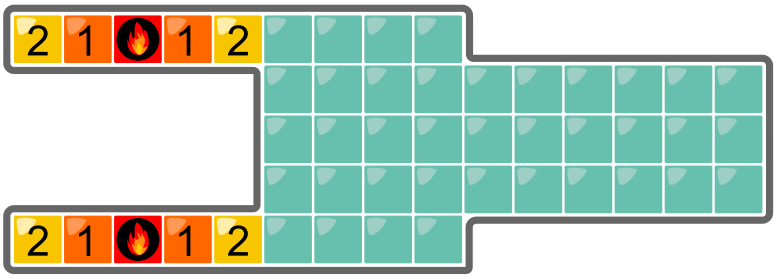


Soluzione

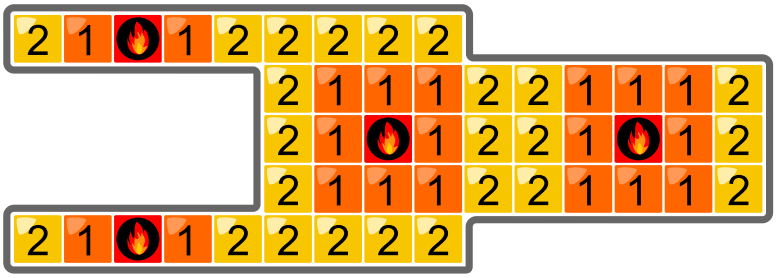
Se i 4 hotspot sono installati come mostrato nell'immagine sottostante, tutte le piastrelle del bagno si riscalderanno entro 2 minuti dall'accensione.

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

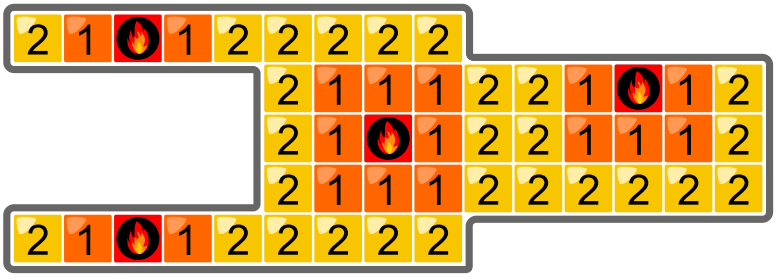
È una buona idea iniziare con i due corridoi a sinistra quando si distribuiscono gli hotspot. Con un punto caldo al centro di ogni corridoio, tutte le piastrelle del corridoio vengono riscaldate in 2 minuti:

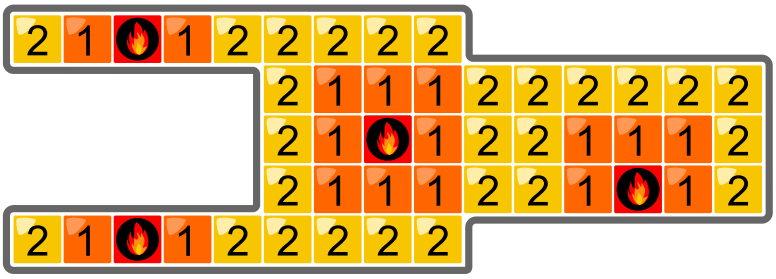


Possiamo poi collocare gli altri due hotspot in questo modo:



Sono possibili anche i seguenti due collocamenti:





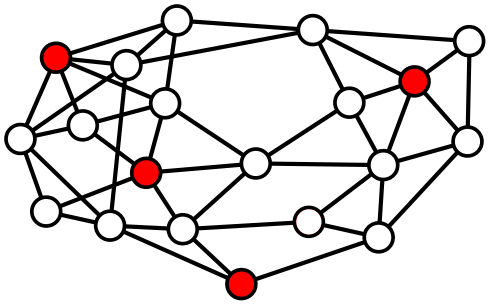
Se il bagno avesse una forma diversa, 2 hotspot potrebbero essere sufficienti per riscaldare l'intero bagno in 2 minuti con la stessa area.

Questa è l'informatica!

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

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

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



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

Parole chiave e siti web

- Insieme dominante

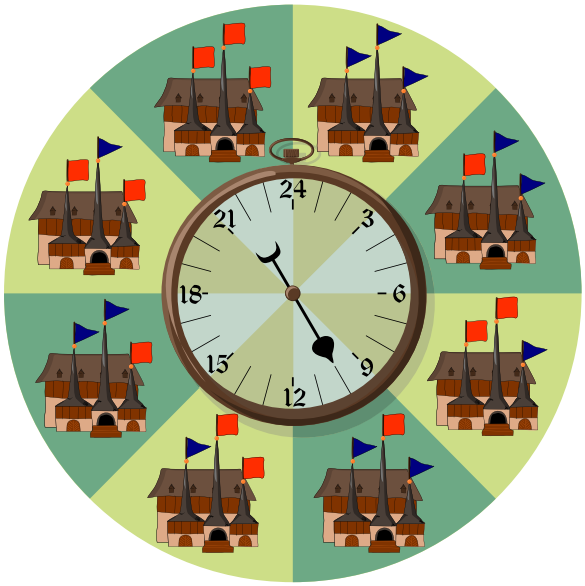
© Castoro Informatico 2020, SSII

15. Castori rilassati

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

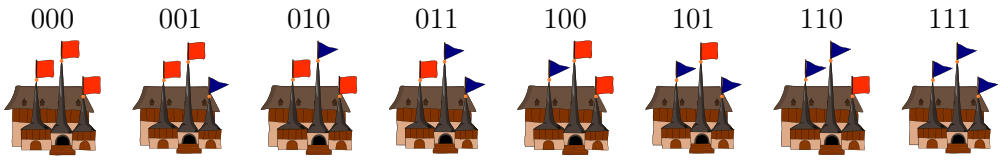
La sistemazione attuale richiede un solo cambio di bandiera a quasi tutte le transizioni. Solo a mezzanotte devono essere cambiate tre bandiere contemporaneamente. I castori vogliono introdurre una sistemazione più conveniente, dove bisogna cambiare una sola bandiera alla volta.

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



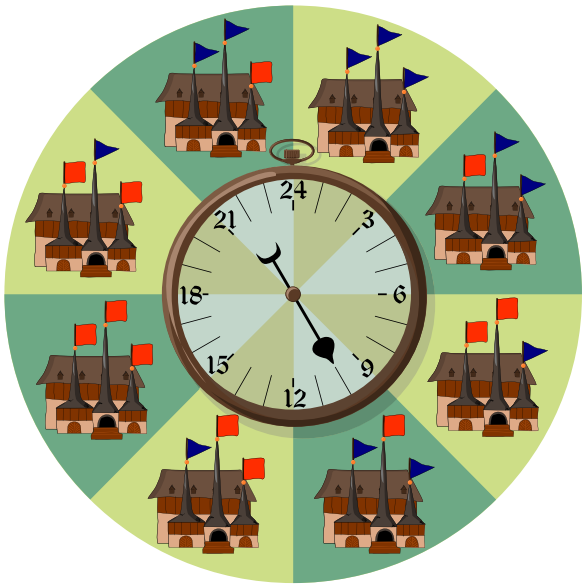
Soluzione

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



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

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



Si trova sistematicamente una soluzione con il seguente metodo:

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

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

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

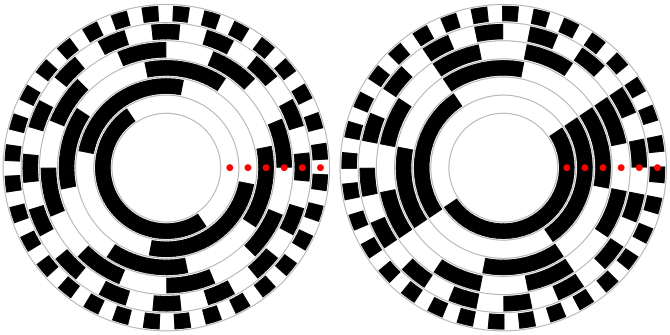
Questo metodo (invertendo la sequenza di numeri esistente e cambiando la cifra successiva più alta da 0 a 1) può essere continuato per tutto il tempo che si desidera per ottenere tali composizioni per

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

Questa è l'informatica!

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

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



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

Parole chiave e siti web

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

A. Autori dei quesiti

Michael Barot
Maksim Bolonkin
Andrey Brodnik
Lucia Budinská
Marios O. Choudary
Kris Coolsaet
Valentina Dagienė
Christian Datzko
Susanne Datzko
Amirmohammad Djazbi
Lidia Feklistova
Fabian Frei
Jens Gallenbacher
Christian Giang
Tom Grubb
Yasemin Gulbahar
Mathias Hiron
Juraj Hromkovič
Alisher Ikramov
Thomas Ioannou
Mile Jovanov
Ungyeol Jung
Vaidotas Kinčius
Sophie Koh
Dennis Komm
Ritambhra Korpal
Chia-Yi Ku
Regula Lacher
Marielle Léonard
Judith Lin
Lynn Liu
Matija Lokar
Vu Van Luan
Pedro Marcelino
Hamed Mohebbi
Kwangsik Moon
Anna Morpurgo
Xavier Muñoz
Ágnes Erdősné Németh
Andrei Nicolicioiu
Jean-Philippe Pellet
Peter Rossmanith
Eljakim Schrijvers
Vipul Shah
Maiko Shimabuku
Timur Sitdikov
Emil Stankov
Maciej M. Sysło
Monika Tomcsányiová
Meng-ting Tsai
Jiří Vaníček
Khairul Anwar Mohamad Zaki

© Castoro Informatico 2020, SSII

B. Sponsoring: concorso 2020

http://www.haslerstiftung.ch/

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

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

Standortförderung beim Amt für Wirtschaft und Arbeit Kanton Zürich

i-factory (Musée des transports, Lucerne)

http://www.ubs.com/

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

https://educatec.ch/
educaTEC

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

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

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

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

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

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

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

© Castoro Informatico 2020, SSII

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.