INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Quesiti e soluzioni 2020

7º e 8º 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

SSII

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

Hanno collaborato al Castoro Informatico 2020

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

Capo progetto: Nora A. Escherle

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

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

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

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

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

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

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

INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

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

HASLERSTIFTUNG

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

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

Premessa

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

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

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

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

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

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

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

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

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

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

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

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

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

Per ulteriori informazioni:

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

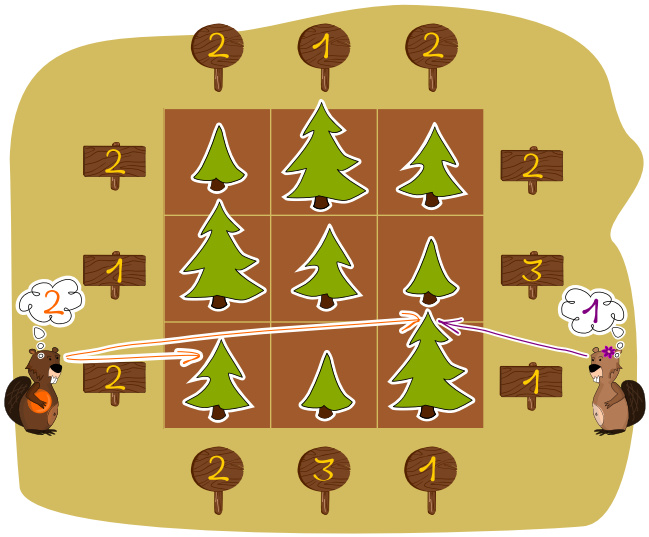
`https://www.castoro-informatico.ch/it/kontaktieren/`
`https://www.castoro-informatico.ch/`

Indice

Hanno collaborato al Castoro Informatico 2020 i
Premessa iii
Indice v
1. 3×3 sudoku con gli alberi 1
2. Prossima fermata, stazione! 5
3. Case colorate 7
4. Dall'alveare ai fiori 11
5. Scale e serpenti 15
6. Comparazioni pesanti 19
7. Braccialetto 23
8. Elettrodomestici 27
9. Viaggio in treno 31
10. Rete ferroviaria 33
11. Sequenza di DNA 37
12. Il castoro testardo 39
13. Auto del ragno 43
14. L'arcipelago dei castori 47
15. Riscaldamento a pavimento 51
A. Autori dei quesiti 54
B. Sponsoring: concorso 2020 56
C. Ulteriori offerte 58

1. 3×3 sudoku con gli alberi

I castori piantano abeti in fila. Gli abeti hanno tre diverse altezze (1 , 2 e 3) e in ogni fila c'è esattamente un abete di ogni altezza.



Quando i castori guardano una fila di abeti da un'estremità, **non** possono vedere gli abeti più bassi nascosti dietro gli abeti più alti.

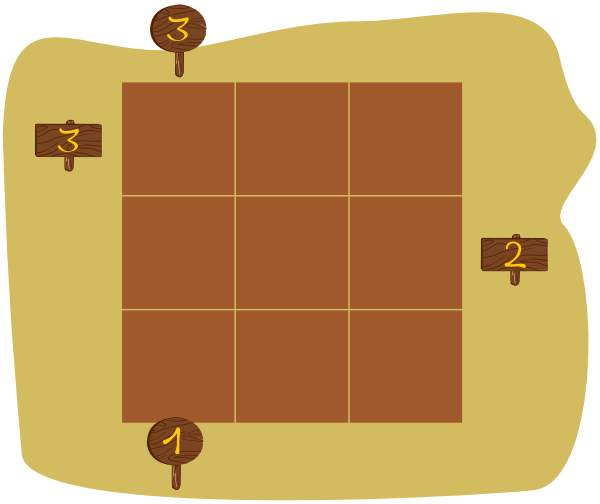
Alla fine di ogni fila di abeti c'è un cartello che indica quanti abeti un castoro può vedere da quel punto.

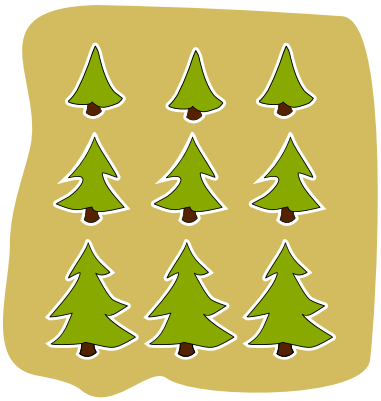
Ora i castori piantano nove abeti in un campo 3×3, come nell'esempio a destra.

Si applicano le seguenti regole:

- in ogni riga (fila orizzontale) c'è esattamente un abete di ogni altezza;
- in ogni colonna (fila verticale) c'è esattamente un abete di ogni altezza;
- i cartelli con il numero di abeti visibili sono posizionati intorno al campo 3×3.

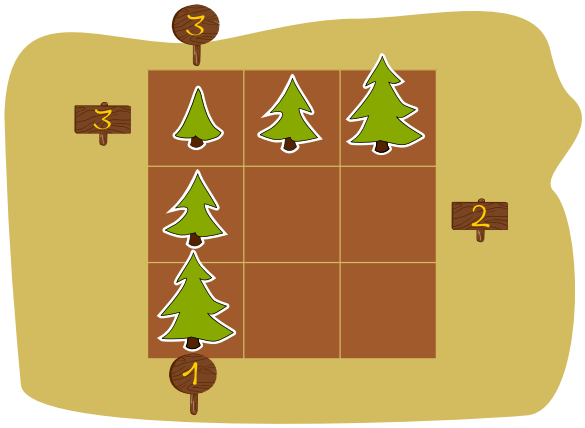
Scrivi in ogni campo l'altezza dell'albero corrispondente.





Soluzione

Sul campo, due cartelli indicano che da quelle posizioni si possono vedere tre abeti. Tutti e tre gli abeti in fila possono essere visti solo se gli abeti sono disposti in modo tale che la loro altezza aumenti, cioè da questa posizione. Questo determina la colonna a sinistra e la riga superiore:

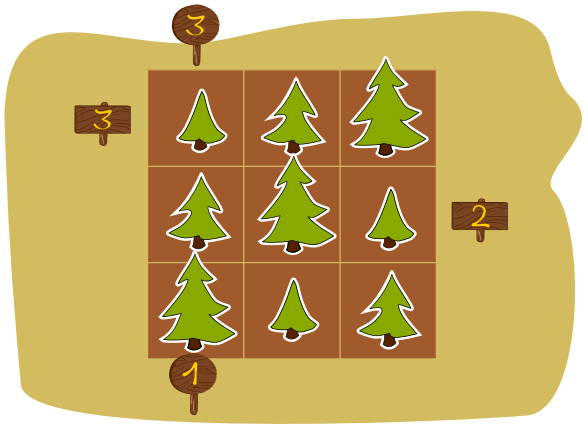


Il cartello a destra con il 2 richiede che da lì siano visibili due abeti, quindi deve esserci un abete di altezza 3 proprio al centro e questa fila centrale è quindi 2 (), 3 (), 1 ().

Gli altri campi sono compilati seguendo la regola del «Sudoku» secondo la quale deve esserci esattamente un abete per ogni altezza in ogni fila.

Al centro della fila inferiore deve esserci un abete di altezza 1 () perché le altre due altezze nella colonna centrale sono già assegnate. Infine, un abete di altezza 2 () deve essere posizionato in basso a destra per completare la fila.

La soluzione completa si presenta così:



Questa è l'informatica!

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

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

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

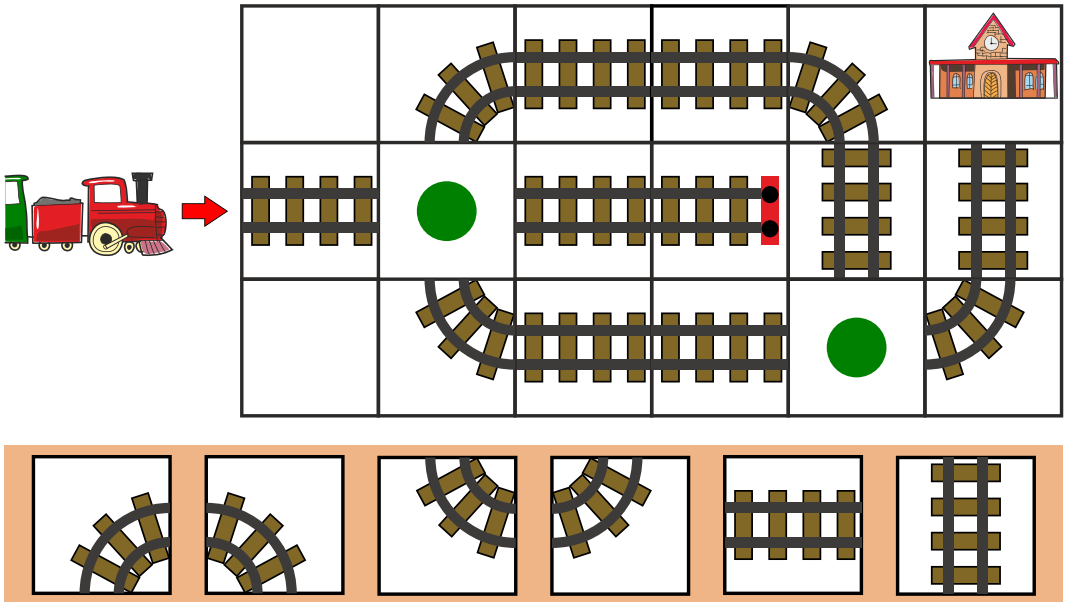
In terzo luogo, tali campi ad albero con cartelli possono essere utilizzati per generare codici autoverificanti. Gli errori che si verificano durante l'immissione o il trasporto delle informazioni possono essere rilevati o persino corretti automaticamente.

Parole chiave e siti web

- Sudoku
- Rilevazione e correzione d'errore:
  `https://it.wikipedia.org/wiki/Rilevazione_e_correzione_d'errore`

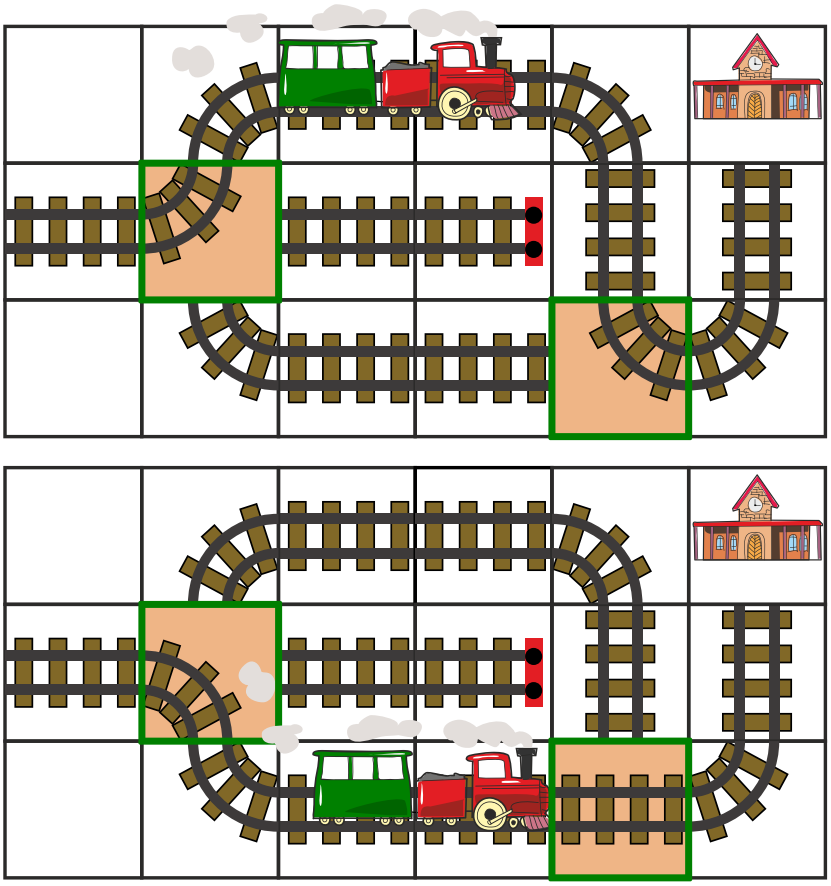
2. Prossima fermata, stazione!

Scegli i binari corretti da mettere nei campi con il punto verde affinché il treno possa raggiungere la stazione.



Soluzione

Per questo problema ci sono le seguenti 2 soluzioni:



Con altre combinazioni il treno deraglia o si scontra con il paraurti.

Questa è l'informatica!

Proprio come un treno viaggia ostinatamente sui binari, un computer esegue ostinatamente le istruzioni di un programma. Non è in grado di riconoscere quando il programma contiene un errore e può bloccarsi, così come un treno può deragliare se i binari sono costruiti in modo errato. Quindi, quando si scrive un programma, bisogna stare molto più attenti che, per esempio, spiegare la strada per la stazione a una persona.

Il compito di cui sopra consiste nell'inserire i comandi mancanti nei punti giusti di un programma in modo da raggiungere l'obiettivo.

Parole chiave e siti web

- Programma
- Instruzione: `https://it.wikipedia.org/wiki/Istruzione_(informatica)`
- `https://it.wikipedia.org/wiki/Algoritmo`

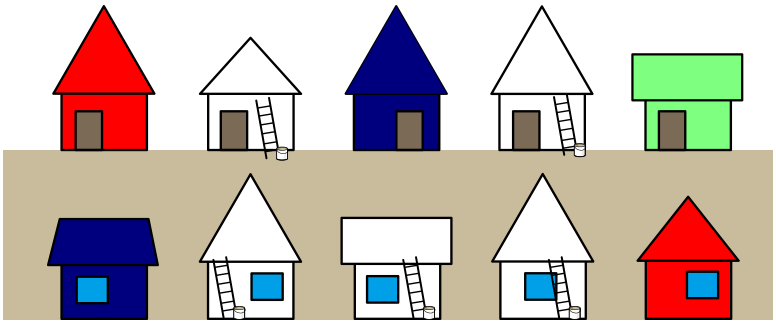
3. Case colorate

Gli abitanti di una strada vogliono dipingere con dei colori le loro case bianche. Ogni casa dovrebbe avere uno dei tre colori: verde chiaro, rosso o blu scuro. Le seguenti regole si applicano per evitare di sembrare noioso:

- Due case che si trovano direttamente l'una accanto all'altra non devono avere lo stesso colore.
- Due case che si trovano direttamente l'una di fronte all'altra non devono avere lo stesso colore.

Alcuni residenti hanno già dipinto le loro case a colori. I restanti residenti devono ora dipingere le loro case in modo che le regole non vengano violate.

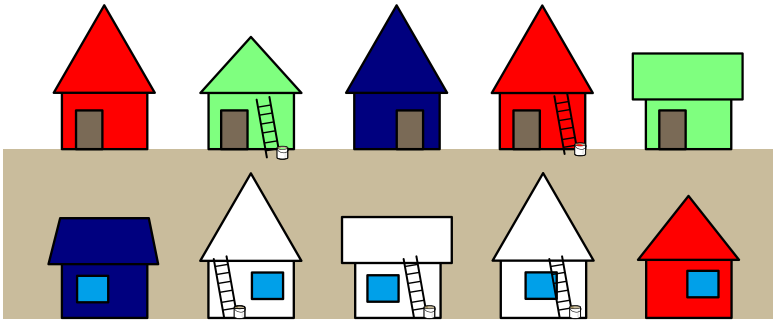
Trova i colori corrispondenti per i residenti.



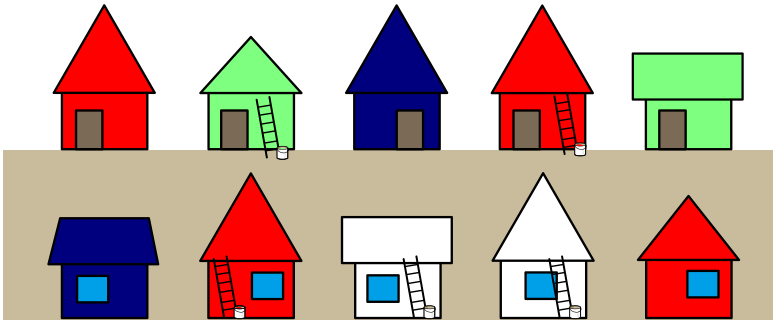
Soluzione

Il modo più semplice per scoprire i colori delle case è scoprirli passo dopo passo.

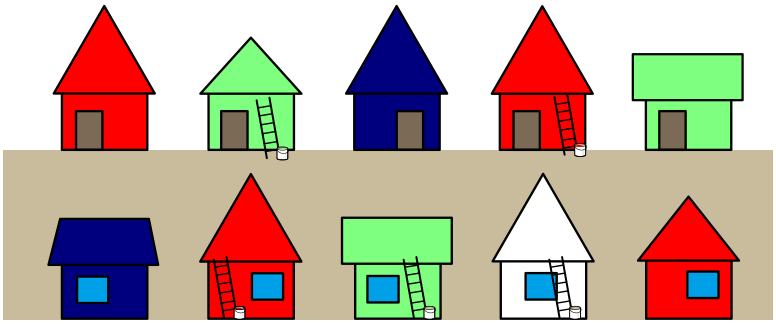
Le due case bianche sul lato superiore della strada sono circondate da due case di colore diverso a sinistra e a destra. Pertanto, possono essere dipinti in un solo colore particolare senza infrangere le regole: la casa bianca in alto a sinistra in verde chiaro e la casa bianca in alto a destra in rosso.



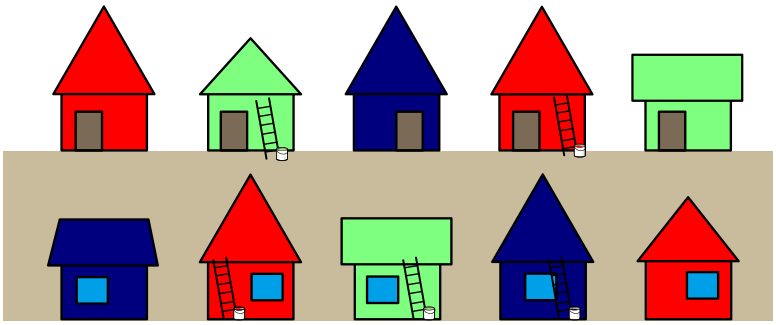
Poi si vede che la casa bianca in basso a sinistra deve essere dipinta di rosso perché la casa direttamente a sinistra è blu scuro e la casa di fronte è verde chiaro:



Quasi la stessa considerazione può essere fatta ora per la casa di mezzo sul lato inferiore della strada: Deve essere dipinta di verde chiaro, perché direttamente alla sua sinistra c'è la casa appena dipinta di rosso e di fronte c'è una casa blu scuro.



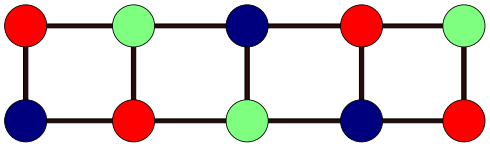
Infine, si può anche scegliere il colore per la casa bianca sul lato destro della strada: La casa direttamente a destra e la casa di fronte sono entrambe rosse, ma poiché la casa direttamente a sinistra è verde chiaro, l'unica opzione rimasta è dipingere la casa di blu scuro:



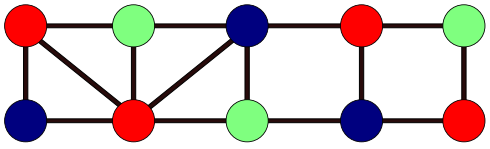
Questa è l'informatica!

In termini astratti, questo compito consiste nel trovare una soluzione che soddisfi le restrizioni date (regole). Questo è un problema molto comune nell'informatica.

Le case e i loro vicini diretti (sia a sinistra che a destra e dall'altra parte della strada) possono essere modellati bene con l'aiuto di un *grafo*, una struttura di dati ampiamente utilizzata nell'informatica. Ogni casa è un *nodo* e ogni vicinato diretto è un *arco*:



Nell' immagine, i nodi sono già colorati come le case corrispondenti. Per le case c'era la regola che le case vicine dovevano avere colori diversi. Nell' immagine la colorazione dei nodi è quindi tale che i nodi collegati direttamente da un arco non hanno mai lo stesso colore. Che ci sia una colorazione valida del grafo con tre colori non è ovvio. Se si aggiungono due archi come nell'immagine successiva, non c'è più una colorazione valida: non importa come si distribuiscono i tre colori in questo grafo, ci sono sempre due nodi direttamente collegati con lo stesso colore.



Ma con quattro colori funziona di nuovo. Forse funziona sempre con quattro colori? La risposta è di nuovo no. Ma almeno un tipo di grafi può sempre essere colorato con quattro colori: i cosiddetti *grafi planari*. Si tratta di grafi che possono essere disegnati in modo che nessun arco si incroci. (Il grafo nell'ultima immagine non è planare, cioè a causa delle connessioni dei quattro nodi all'estrema sinistra). Che i grafi planari hanno una colorazione valida con quattro colori è chiamato il *teorema dei quattro colori*.

Il teorema dei quattro colori è particolarmente interessante per la creazione di mappe. Se si immagina ogni paese come un nodo e poi si collegano i paesi vicini con un arco, si ottiene sempre un grafo planare. (A rigor di termini, per questo dobbiamo escludere l'esistenza delle cosiddette enclavi ed esclavi, cioè parti di un paese che si trovano completamente in un altro paese). Questo grafo può quindi essere colorato con quattro colori validi, e quindi anche i paesi corrispondenti sulla mappa possono essere colorati con quattro colori, in modo che i paesi vicini abbiano sempre colori diversi.

La prova che quattro colori sono sufficienti non è facile. Che cinque colori siano sufficienti era già noto 200 anni fa. I matematici Kenneth Appel e Wolfgang Haken hanno dimostrato nel 1976 che quattro colori sono sufficienti. Hanno usato un computer per controllare un gran numero di eccezioni e controesempi. Ci sono volute più di mille ore al computer per farlo. Controllare tutto a mano sarebbe stato del tutto impossibile. Molti matematici si sono poi chiesti se una tale prova sia valida, perché bisogna fidarsi del computer.

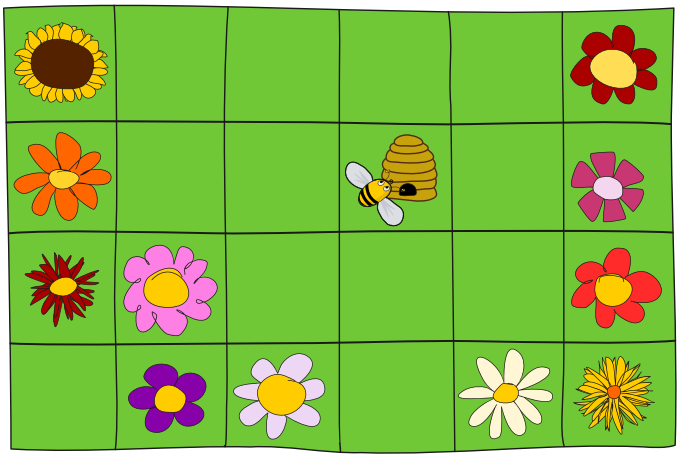
Parole chiave e siti web

- Teorema dei quattro colori: `https://it.wikipedia.org/wiki/Teorema_dei_quattro_colori`
- Grafo: `https://it.wikipedia.org/wiki/Grafo`

4. Dall'alveare ai fiori

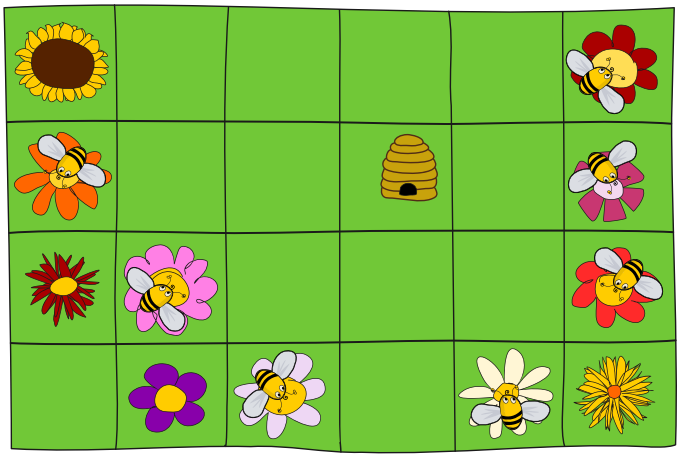
Un'ape vola in su, in giù, a sinistra o a destra. Per volare la distanza di un quadrato ci impiega 10 minuti. Vola dall'alveare, per un massimo di 30 minuti prima di tornare indietro.

Disegna un cerchio attorno ai fiori che si possono raggiungere dall'alveare in massimo 30 minuti.



Soluzione

I fiori con sopra un'ape sono raggiungibili dall'alveare in massimo 30 minuti:



L'immagine sottostante mostra per ogni campo quanti minuti necessita un'ape per raggiungerlo dall'alveare. Così in mezz'ora tutti i campi con un 10, 20 o 30 sono raggiungibili.

Riempire i numeri funziona in questo modo: Nelle caselle accanto all'alveare scriviamo 10 perché l'ape ha bisogno di 10 minuti per volare dall'alveare a lì. Poi scriviamo 20 in tutti i campi vuoti accanto a un campo con 10, perché l'ape ha bisogno di 10 minuti per volare da un campo all'altro. Continueremo a farlo. Così ne scriviamo 30 in tutti i campi vuoti accanto a un campo con 20. Poi ne scriviamo 40 in tutti gli spazi vuoti accanto a uno spazio con 30. Infine, scriviamo 50 in tutti i campi vuoti accanto a un campo con 40.

Questa è l'informatica!

Quando si risolve il compito, si calcola per ogni campo il tempo in cui un'ape può raggiungerlo dall'alveare. Per prima cosa vengono determinati i campi raggiungibili in 10 minuti. Questi vengono poi utilizzati per determinare i campi che si trovano a 20 minuti di distanza. Utilizzando i campi distanti 20 minuti, si trovano poi i campi distanti 30 minuti e così via.

Quindi utilizziamo i risultati già calcolati e memorizzati (i numeri dei campi riempiti) per calcolare ulteriori risultati (i numeri dei campi vicini, ancora vuoti). Questo principio molto generale si chiama *programmazione dinamica*. Di solito è importante l'ordine in cui vengono calcolati i risultati. Anche questo è da considerare con il volo delle api.

Nel compito un'ape vola in 10 minuti solo su, giù, a sinistra o a destra. Questo è un po' insolito, perché in realtà un'ape probabilmente volerebbe anche in diagonale sui campi. Con questa ipotesi più realistica, i campi raggiungibili in mezz'ora sarebbero limitati da un cerchio invece che da un diamante come nel compito.

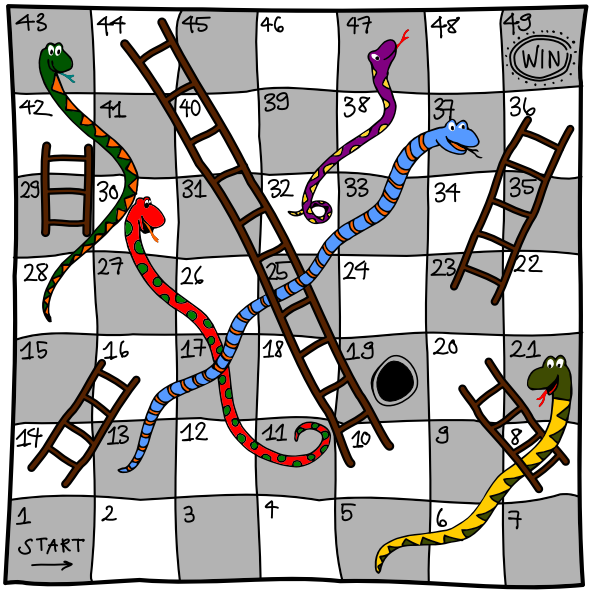
La consueta misura della distanza che porta ad un cerchio è chiamata distanza euclidea. La misura della distanza così come nel compito in cui si è autorizzati a muoversi solo orizzontalmente o verticalmente attraverso i quadrati è chiamata *distanza di Manhattan* (il nome deriva dalle reti stradali a griglia delle città moderne come Manhattan).

Parole chiave e siti web

- Programmazione dinamica: `https://it.wikipedia.org/wiki/Programmazione_dinamica`
- Distanza euclidea: `https://it.wikipedia.org/wiki/Distanza_euclidea`
- Distanza di Manhattan (o geometria del taxi): `https://it.wikipedia.org/wiki/Geometria_del_taxi`

5. Scale e serpenti

Nel gioco delle scale e serpenti tutti i giocatori partono dalla casella 1, e il primo giocatore che raggiunge la casella 49 vince. In ogni turno si tira il dado e si sposta la statuina nel campo corrispondente (tra 1 e 6).



Se si finisce in un campo con la testa di un serpente, si scivola verso il campo con la sua coda. Ma se si finisce ai piedi di una scala, si può salire fino in cima.

Esempio: stai sulla casella 26 e tiri un 3. Puoi passare a 29 e quindi avanzare immediatamente alla casella 42. Nel turno successivo tiri un 5, atterri sulla testa del serpente del campo 47 e devi tornare al campo 32.

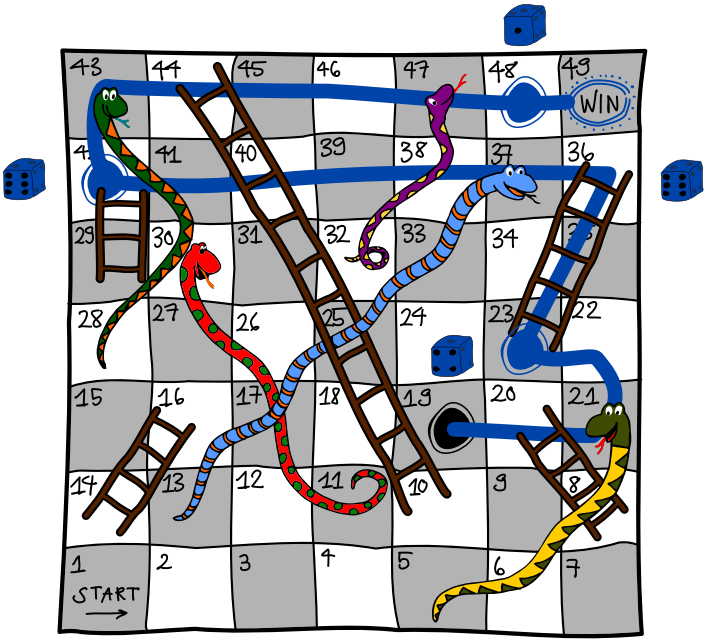
La tua statuina è sul campo 19, di quanti turni hai bisogno minimo per raggiungere il campo 49?

A) 2 turni
B) 3 turni
C) 4 turni
D) 5 turni

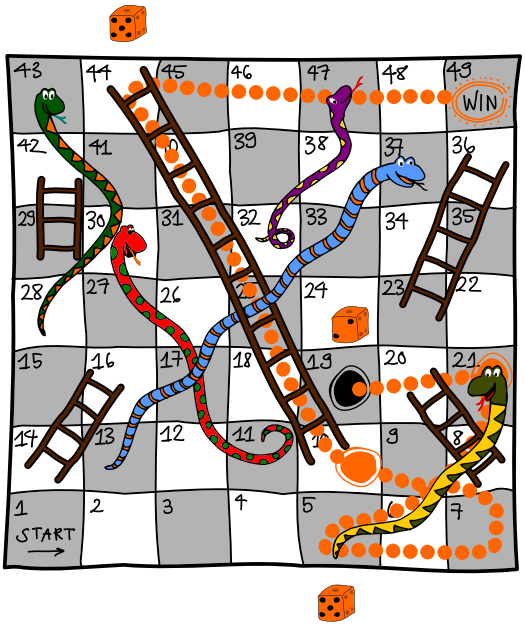
Soluzione

La risposta corretta è B) 3 turni.

Se sei avido e tieni conto solo dei tiri che ti porteranno verso la meta, hai bisogno di almeno 4 turni: Con un 4 arrivi dal 19 al 23 e con la scala al 36, da lì non ci sono più scale e hai bisogno di altri 3 tiri, per esempio 6 – 6 – 1 per raggiungere la meta.



Ma se si accetta una mossa che all'inizio sembra un peggioramento, ci si arriva in 3 turni, con i lanci 2 – 5 – 5. Dal 19 al 21 e attraverso il serpente fino alla casella 5, poi al 10 e fino al 44 e poi fino alla meta.



In 2 turni l'obiettivo non si può raggiungere. Ad un tiro dalla meta si trovano i campi 48, 46, 45, 44 e nessuno di questi campi può essere raggiunto dal 19 in un solo turno.

Questa è l'informatica!

Molti compiti possono essere risolti trovando il cammino più breve tra due punti. Qui, «breve» spesso non ha il significato intuitivo. Qui, per esempio, abbiamo cercato il percorso con il minor numero di tiri e non il percorso che attraversa meno campi. Questo è conosciuto anche dai sistemi di navigazione, che ci propongono il percorso più breve o quello che ci richiede meno tempo. Nelle aziende di logistica, gli stessi dispositivi calcolano il percorso con il pedaggio più basso.

Nell' informatica, le stesse procedure (algoritmi) possono spesso essere utilizzate per compiti molto diversi se sono modellati in modo corrispondente.

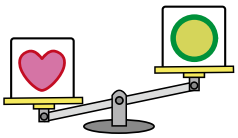
Parole chiave e siti web

- Cammini minimi: `https://it.wikipedia.org/wiki/Algoritmo_di_Dijkstra`
- Scale e serpenti: `https://it.wikipedia.org/wiki/Scale_e_serpenti`

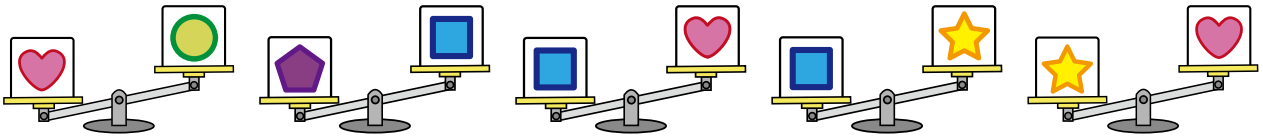
6. Comparazioni pesanti

Cinque scatole sono contrassegnate da cinque diversi simboli:  ,  ,  ,  e  .

Con l'aiuto di una bilancia si comparano due scatole alla volta. La seguente comparazione mostra, ad esempio, che  è più pesante di  :



In totale sono state effettuate cinque comparazioni:



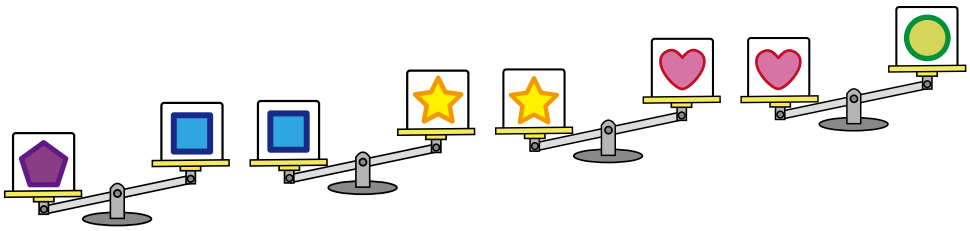
Qual è la scatola più pesante?

A) 
B) 
C) 
D) 
E) 

Soluzione

La scatola C) con il pentagono è la più pesante.

La figura seguente contiene quattro delle cinque comparazioni effettuate e tutte e cinque le scatole.



Così si vede subito: la scatola con il pentagono è più pesante di quella con il quadrato. La scatola con il quadrato è più pesante della scatola con la stella. La scatola con la stella è più pesante di quella con il cuore. E la scatola con il cuore è più pesante della scatola con il cerchio.

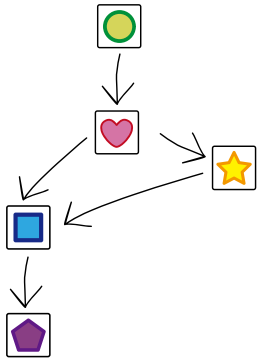
Da questo si può ora concludere che la scatola con il pentagono è più pesante di tutte le altre. Ciò è dovuto ad una speciale proprietà di comparazione dei pesi:

Se A è più pesante di B e B è più pesante di C, allora anche A è più pesante di C. Questa proprietà molto importante si chiama *relazione transitiva*.

A proposito, c'è un modo intelligente per abbreviare questo compito. Poiché si sta cercando la scatola più pesante, è sufficiente cercare la scatola che non è più leggera di qualsiasi altra scatola, e questa è solo la scatola con il pentagono.

Questa è l'informatica!

Fondamentalmente, questo compito riguarda *l'ordinamento* di qualsiasi oggetto. In informatica, per l'ordinamento vengono spesso utilizzati dei *grafi* speciali, che consistono in *nodi* (gli oggetti da ordinare) e *archi* (comparazione di due oggetti). Gli oggetti in questo compito sono le scatole e le comparazioni sono le pesate. Se si disegnano gli archi come frecce che puntano all'oggetto più pesante, il grafo per questo compito si presenta così:



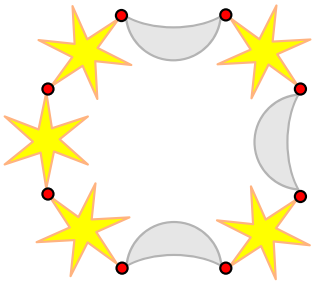
Gli oggetti devono ora essere ordinati in fila, in modo che le frecce partano sempre dagli oggetti più a sinistra verso gli oggetti più a destra. Una tale disposizione è chiamata *ordinamento topologico*. Un ordinamento topologico si ottiene molto semplicemente rimuovendo ripetutamente un nodo dal grafo verso il quale non c'è alcuna freccia puntata e ponendo i nodi rimossi in quest'ordine.

Ma attenzione: non tutti i grafi hanno un ordinamento topologico. Per esempio, non c'è un ordinamento topologico se ci sono tre frecce che puntano in un cerchio.

Parole chiave e siti web

- Relazione transitiva: `https://it.wikipedia.org/wiki/Relazione_transitiva`
- Grafo: `https://it.wikipedia.org/wiki/Grafo`
- Ordinamento topologico: `https://it.wikipedia.org/wiki/Ordinamento_topologico`

7. Braccialetto



Marie vorrebbe avere il braccialetto a destra.

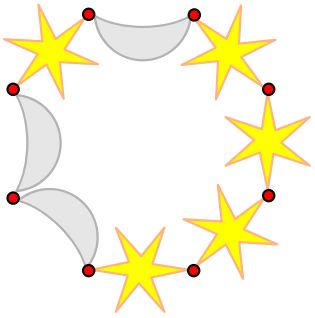
Per questo motivo dà a Jonas le seguenti istruzioni:

- Prendi una stella (★) e una luna (☾) e collegale a formare una coppia. Fallo tre volte in totale, in modo da avere tre coppie.
- Prendi queste tre coppie, girale come vuoi e collegale in una lunga catena.
- Aggiungi altre due stelle ad un'estremità della catena. Ora collega le due estremità della catena per ottenere un braccialetto.

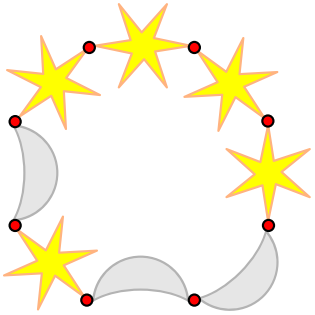
Jonas non ha una foto del braccialetto desiderato. È possibile che ottenga un braccialetto dall'aspetto completamente diverso, anche se Jonas segue esattamente le istruzioni di Marie.

*Uno dei quattro braccialetti **NON** si può ottenere se Jonas segue esattamente le istruzioni di Marie. Quale?*

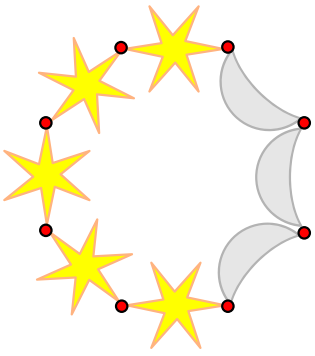
A)



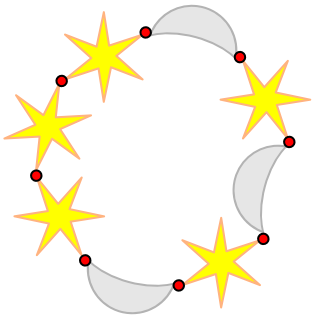
B)



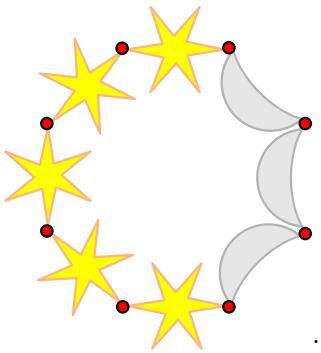
C)



D)

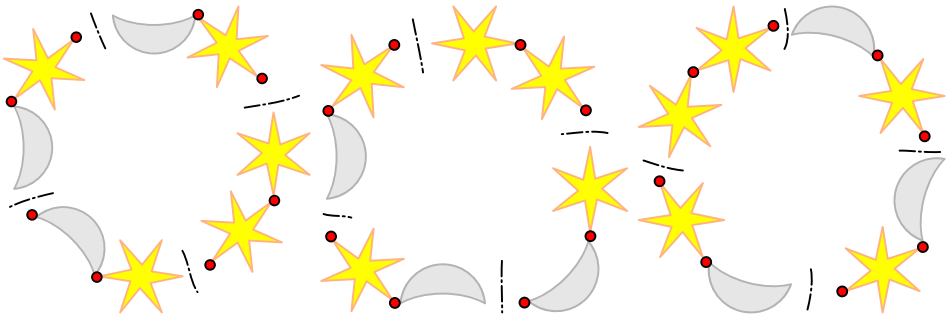


Soluzione

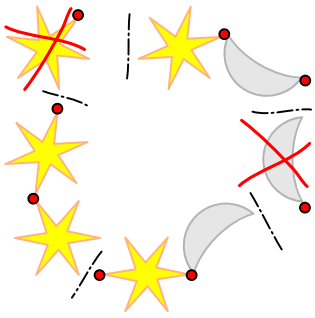
La risposta corretta è C)  .

Solo questo braccialetto non si può ottenere seguendo le istruzioni di Marie.

I braccialetti delle altre tre risposte sono corretti seguendo le istruzioni di Marie. Lo si può vedere, ad esempio, perché ognuno di questi braccialetti può essere diviso in tre coppie stella-luna e una coppia stella-stella, come mostrato nelle foto.



Una luna può essere inserita nel braccialetto solo come parte di una coppia stella-luna. Pertanto, ogni luna ha almeno una stella accanto ad essa. Quindi tre lune di fila come nel braccialetto C non sono possibili. Anche cinque o più stelle di fila sono impossibili.



Questa è l'informatica!

Quando i programmatori danno istruzioni a un computer, è importante che specifichino esattamente ciò che il computer deve fare. Altrimenti si potrebbe ottenere un risultato indesiderato. Per esempio, Marie ha dimenticato di specificare nella sua lista di istruzioni esattamente come le tre coppie stella-luna devono essere collegate tra loro. Nel braccialetto che voleva, una luna è sempre circondata da stelle. Quindi mancava qualcosa, anche se le istruzioni sembrano molto precise. Se ci fosse un

computer che controlla una macchina per fare braccialetti, le istruzioni di Marie non sarebbero abbastanza precise. Fortunatamente, i veri computer di solito si fermano e dicono: «Non so cosa intendi, perché le istruzioni non sono abbastanza chiare».

Nell'informatica ci sono molti meccanismi per descrivere le cose in modo molto preciso. Un meccanismo sono le cosiddette *grammatiche (formali)*. Una grammatica contiene *regole* che descrivono esattamente come creare determinate *parole* (una sequenza di lettere). Ad esempio, è possibile esprimere le istruzioni di Marie in una grammatica come questa:

$$\begin{aligned} &B \to CSS && (1)\\ &C \to PPP && (2)\\ &P \to SL && (3) \end{aligned}$$

Qui B sta per braccialetto, C per catena, P per coppia, S per stella e L per luna. Si inizia con B e poi si possono creare nuove parole applicando le tre regole di sostituzione tutte le volte che si desidera. Lo si fa fino a quando la parola è composta solo dai simboli S e L. Per esempio:

$$\begin{aligned} &B \Rightarrow CSS && \text{per regola (1)}\\ &CSS \Rightarrow PPPSS && \text{per regola (2)}\\ &PPPSS \Rightarrow SLPPSS \Rightarrow SLSLPSS \Rightarrow SLSLSLSS && \text{per regola (3)} \end{aligned}$$

Si può considerare che la grammatica di cui sopra corrisponde esattamente alle istruzioni di Marie.

L'informatica non è solo programmazione. Spesso si tratta di descrivere gli oggetti. Un insieme di regole di creazione (la grammatica o le istruzioni di Marie) può essere utilizzato per descrivere esattamente una classe di oggetti (alcune parole o i possibili braccialetti). La classe contiene esattamente quegli oggetti che possono essere creati con le regole.

Parole chiave e siti web

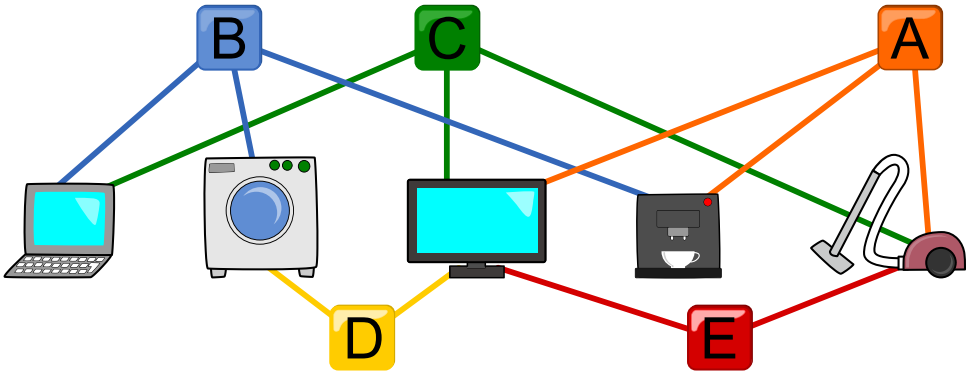
- Grammatica formale: `https://it.wikipedia.org/wiki/Grammatica_formale`

8. 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.

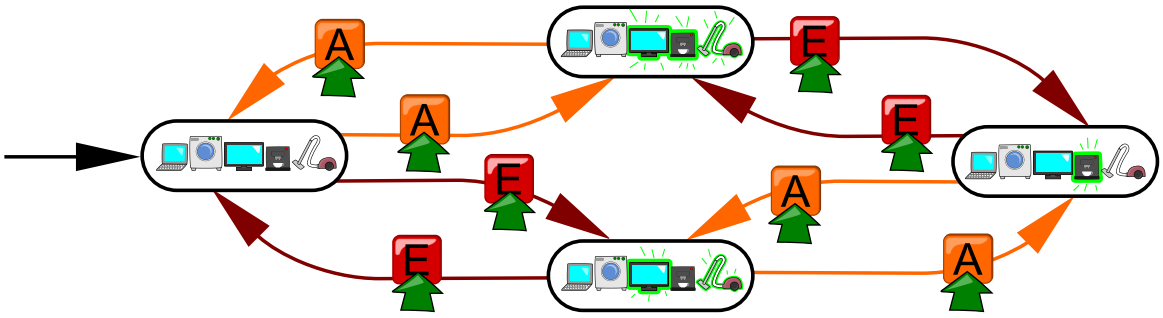
Il sistema dei cinque dispositivi ha molti *stati* diversi. Per esempio, uno stato è quando è acceso solo il televisore. Un altro stato è quando tutti gli apparecchi sono spenti (poiché tutti gli apparecchi sono spenti all'inizio, lo chiamiamo lo *stato iniziale*). E un altro stato è quello in cui sono 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

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.

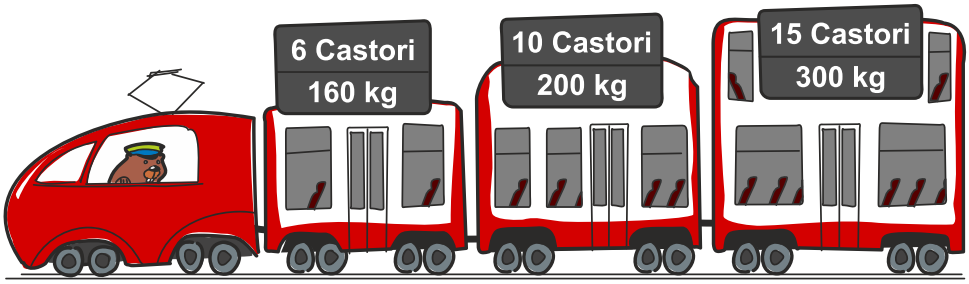
Parole chiave e siti web

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

9. Viaggio in treno

Otto famiglie di castori vorrebbero viaggiare con il treno. Le famiglie sono elencate con il numero dei loro membri e il peso del loro bagaglio nella seguente tabella:

Nome della famiglia	**Numero di membri**	**Peso del bagaglio in kg**
Ammann	3	50
Bernasconi	4	80
Camenzind	5	110
Donetta	4	80
Emery	2	40
Favre	3	70
Gerber	6	130
Huber	5	100



L'immagine mostra per ogni carrozza quanti castori e quanti bagagli possono essere trasportati al massimo. Inoltre, le famiglie devono viaggiare con i loro bagagli in una carrozza e non possono dividersi.

Qual è il numero massimo di famiglie di castori che possono viaggiare con il treno?

A) 1 famiglia di castori
B) 2 famiglie di castori
C) 3 famiglie di castori
D) 4 famiglie di castori
E) 5 famiglie di castori
F) 6 famiglie di castori
G) 7 famiglie di castori
H) 8 famiglie di castori

Soluzione

Possono viaggiare al massimo 7 famiglie di castori. Una delle possibili distribuzioni è:

	Nome della famiglia	Numero di membri	Peso del bagaglio in kg
	Gerber	6	130
	Totale:	**6**	**130**
	Ammann	3	50
	Camenzind	5	110
	Emery	2	40
	Totale:	**10**	**200**
	Bernasconi	4	80
	Donetta	4	80
	Huber	5	100
	Totale:	**13**	**260**

Le 8 famiglie di castori insieme costituiscono un totale di 32 passeggeri, mentre solo 31 posti a sedere sono disponibili sul treno. È quindi impossibile che tutte le 8 famiglie di castori possano viaggiare sul treno.

Questa è l'informatica!

L'informatica si occupa spesso di *problemi di ottimizzazione* dove le risorse limitate - come in questo caso lo spazio e la capacità di peso - devono essere utilizzate al meglio. In realtà, naturalmente, nessun passeggero dovrebbe essere lasciato indietro, ma la compagnia ferroviaria può calcolare, ad esempio, che è meglio trasportare i singoli passeggeri comodamente in taxi piuttosto che utilizzare un treno completo che poi viaggia quasi vuoto.

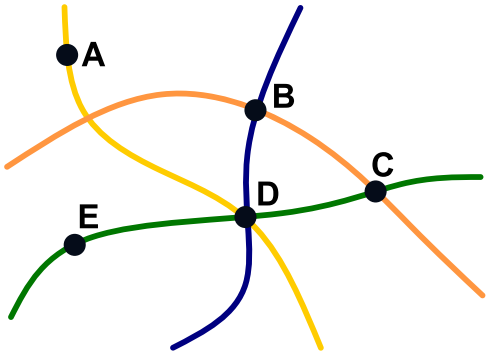
Compiti di questo tipo sono noti come *problemi dello zaino*. A volte tali problemi possono essere ridotti in modo tale da poter essere risolti con l'aiuto della *programmazione dinamica*, cioè individuando prima le possibili soluzioni parziali che possono poi essere ulteriormente sviluppate in una soluzione globale. In molti casi, tuttavia, i compiti appartengono ai cosiddetti *problemi NP-completi*, il che significa che attualmente non c'è soluzione migliore che provare in modo intelligente. La maggioranza delle persone avrà risolto questo compito proprio in questo modo.

Parole chiave e siti web

- Problema dello zaino: `https://it.wikipedia.org/wiki/Problema_dello_zaino`
- Programmazione dinamica: `https://it.wikipedia.org/wiki/Programmazione_dinamica`
- NP-completo: `https://it.wikipedia.org/wiki/NP-completo`

10. 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.

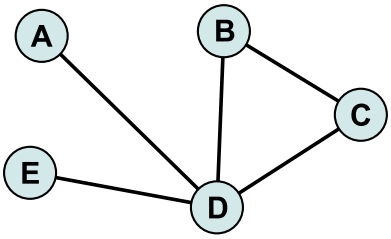


Un diagramma dovrebbe rappresentare questa mappa in modo tale che:

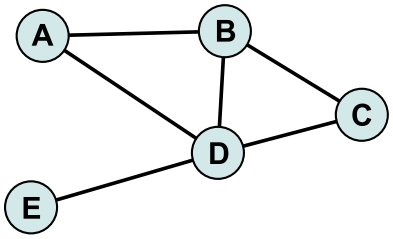
- le città sono rappresentate da cerchi, e
- due città sono collegate da una linea solo quando si trovano sulla stessa linea ferroviaria.

Quale diagramma visualizza correttamente la mappa?

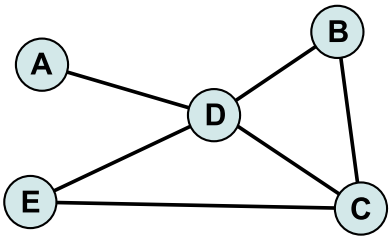
A)



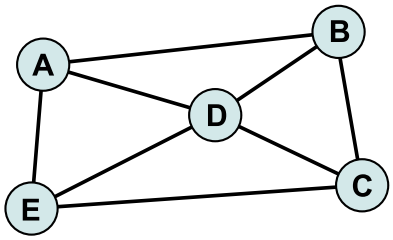
B)



C)

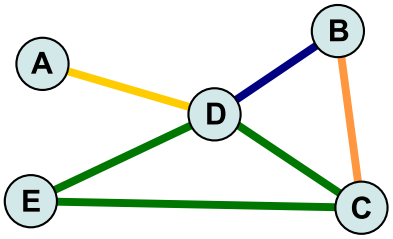


D)



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,

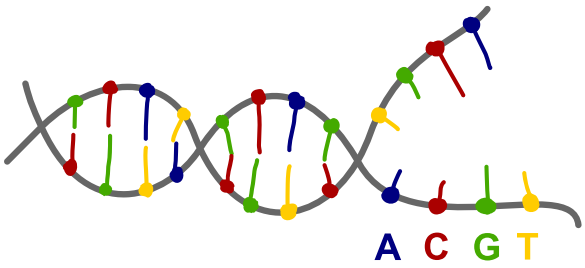
si perde l'informazione che con questa linea ferroviaria si passa per la città D sulla strada dalla città C alla città E.

Parole chiave e siti web

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

11. Sequenza di DNA

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



Consideriamo i seguenti tre tipi di mutazioni:

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

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

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

Soluzione

La risposta corretta è D) GGTAAAC.

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

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

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

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

Non è facile dimostrare che meno mutazioni non sono sufficienti.

Questa è l'informatica!

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

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

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

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

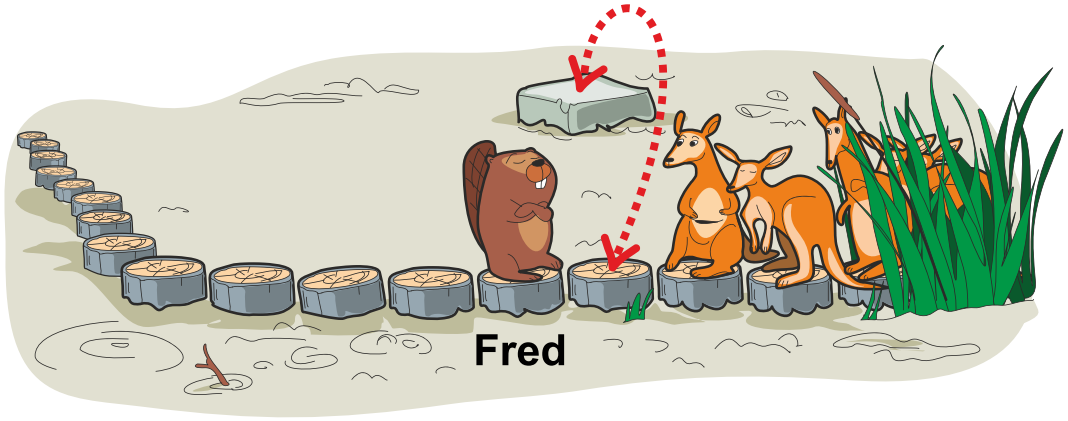
Parole chiave e siti web

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

12. 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?

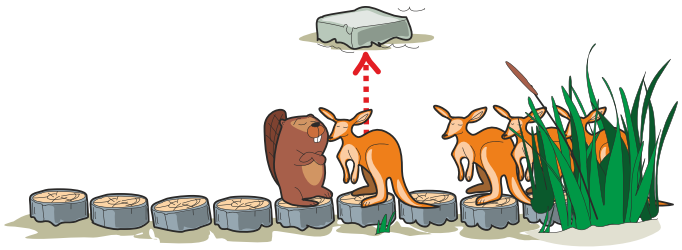
A) Più di 10 canguri.
B) Esattamente 10 canguri.
C) Esattamente 6 canguri.
D) Esattamente 4 canguri.
E) Meno di 4 canguri.
F) È impossibile dirlo con certezza.

Soluzione

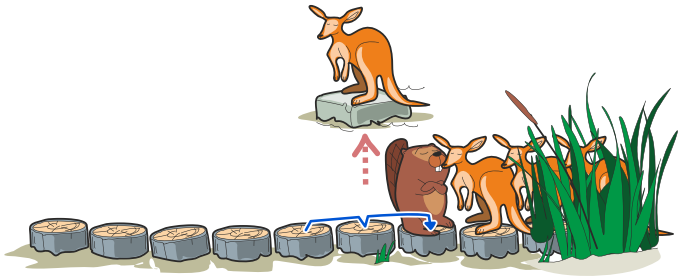
Fred può far passare un massimo di 6 canguri.

Un canguro passa Fred come segue:

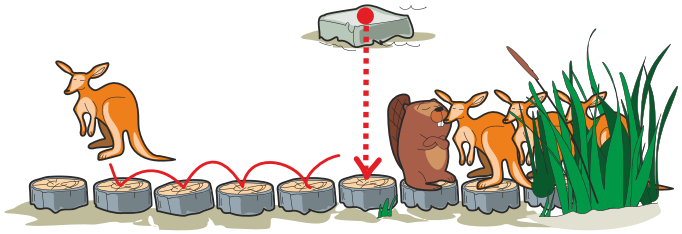
Il canguro salta sulla pietra.



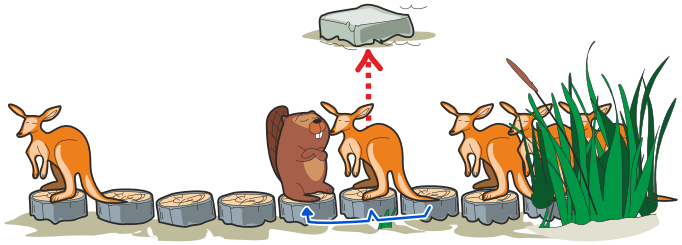
Fred cammina due ceppi d'albero in avanti.



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

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.

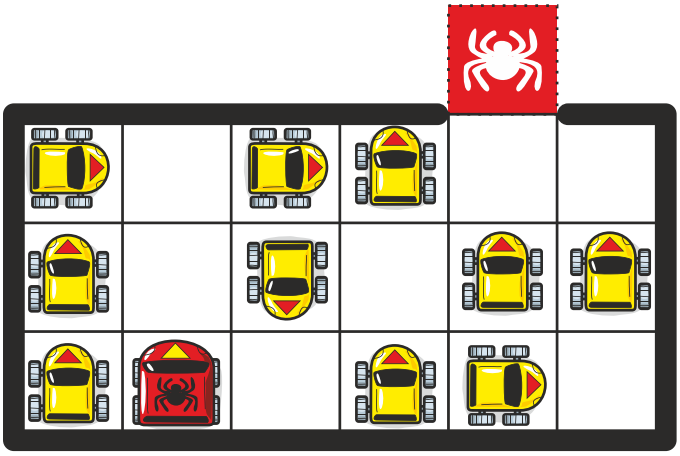
Parole chiave e siti web

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

13. Auto del ragno

11 auto parcheggiano in una piazza circondata da muri con un'uscita. Ogni auto ha le seguenti possibilità di movimento:

- Un campo in avanti
- Un campo all'indietro
- Un quarto di giro a destra o a sinistra nel campo dove si trova



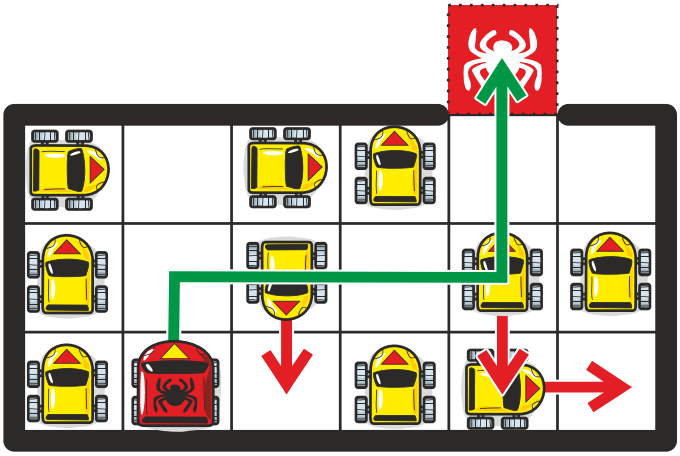
Un'auto può anche eseguire diversi spostamenti. Solo un'auto può essere presente su ogni campo alla volta.

Quanti spostamenti delle auto in totale sono necessari per portare l'auto del ragno rosso nel campo del ragno rosso?

A) 9 spostamenti
B) 11 spostamenti
C) 13 spostamenti
D) 15 spostamenti

Soluzione

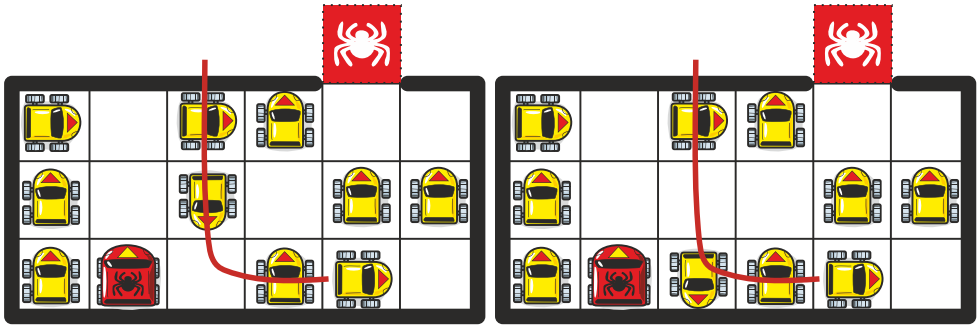
La risposta corretta è: B) 11 spostamenti. L'immagine mostra gli 11 spostamenti per portare l'auto del ragno nel campo del ragno rosso:



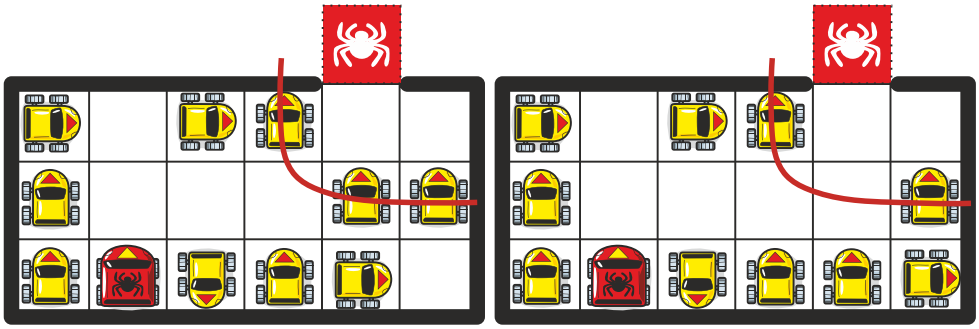
Deve ancora essere dimostrato che 11 è il numero minimo di spostamenti richiesto.

Per questo si parte dal presupposto che l'auto del ragno sia l'unica auto sul percorso. Per arrivare al campo del ragno rosso all'esterno, la macchina del ragno deve spostarsi 3 volte verso l'alto e 3 volte verso destra, deve anche girare 2 volte. Anche se questo può essere realizzato in modi diversi, sono necessari almeno $3 + 3 + 2 = 8$ spostamenti. Ma l'auto del ragno non è l'unica auto sulla piazza e servono più spostamenti per liberare la via.

Prima dobbiamo trovare una via attraverso la barricata a forma di L nella foto successiva. Questo può essere fatto in un unico spostamento come segue:



Poi dobbiamo trovare una via attraverso una seconda barricata a forma di L. Questa barricata non può essere aperta con 1 solo spostamento, ma 2 sono sufficienti come indicato di seguito.



Pertanto il numero minimo di spostamenti è di $8 + 1 + 2 = 11$ spostamenti.

Questa è l'informatica!

Spesso è molto difficile dimostrare che una soluzione trovata è ottimale. Spesso si scopre se esiste o meno una soluzione migliore solo esaminando tutte le soluzioni possibili. Questo metodo si chiama *forza bruta* (in inglese *brute force*) o *ricerca esaustiva* (in inglese *exhaustive search*), perché tutte le possibilità sono esaurite. Anche se questo metodo di solito non è praticabile a mano, per il computer è spesso una strategia facile da implementare.

Ma a volte ci sono così tante soluzioni diverse che anche un computer è sopraffatto da ciò. In questi casi si deve cercare una strategia più adeguata. Ad esempio, vengono spesso utilizzati *algoritmi greedy* (dall'inglese per *avido*) o il principio «*branch and bound*».

Il compito è una variante del gioco *Rush Hour*. Il classico gioco per computer *Sokoban* ha anche molte somiglianze.

Parole chiave e siti web

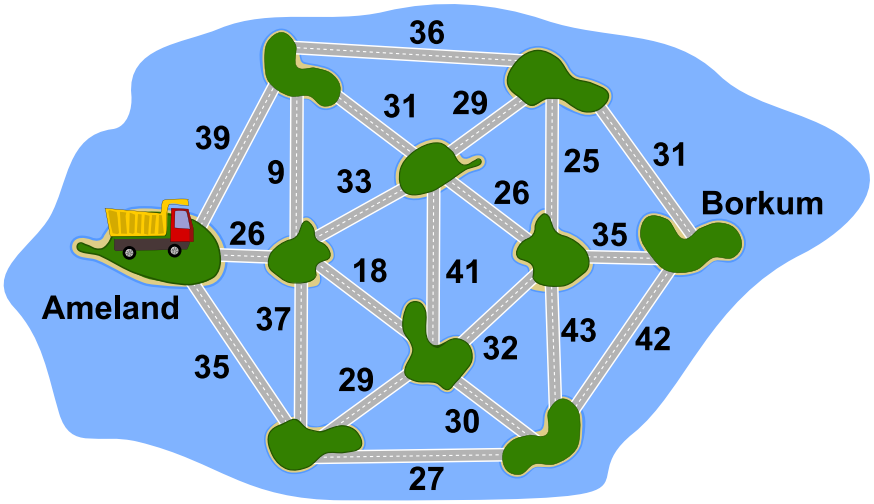
- Forza bruta: `https://it.wikipedia.org/wiki/Metodo_forza_bruta`
- Branch and bound: `https://it.wikipedia.org/wiki/Branch_and_bound`
- Algoritmo greedy: `https://it.wikipedia.org/wiki/Algoritmo_greedy`
- Rush Hour

14. 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.

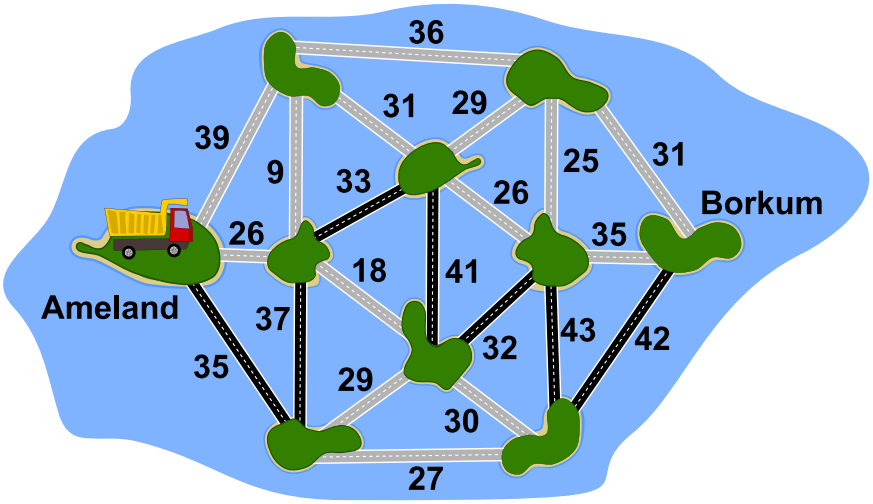
Il castoro Knuth vuole costruire una spiaggia sull'isola di Borkum. Vuole quindi trasportare quanta più sabbia possibile dall'isola di Ameland all'isola di Borkum in un solo viaggio. Non gli interessa la lunghezza del viaggio, ma non vuole passare su nessun ponte più di una volta.

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

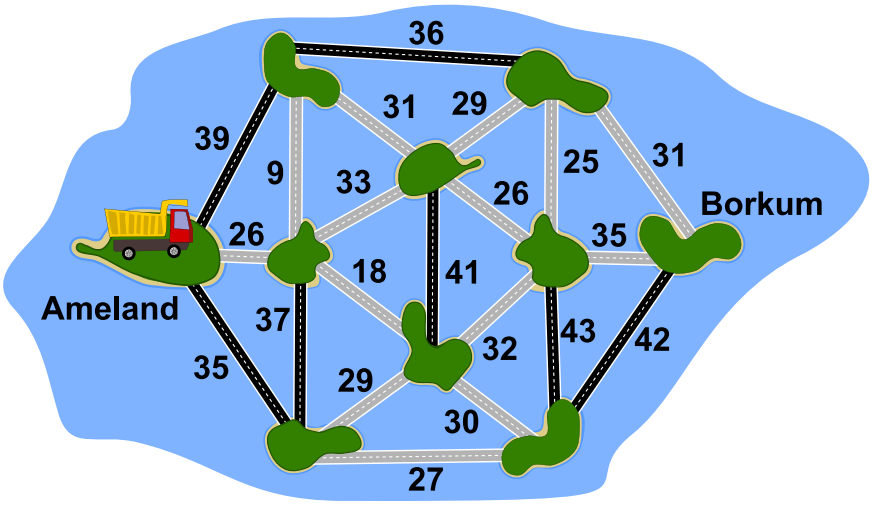


Soluzione

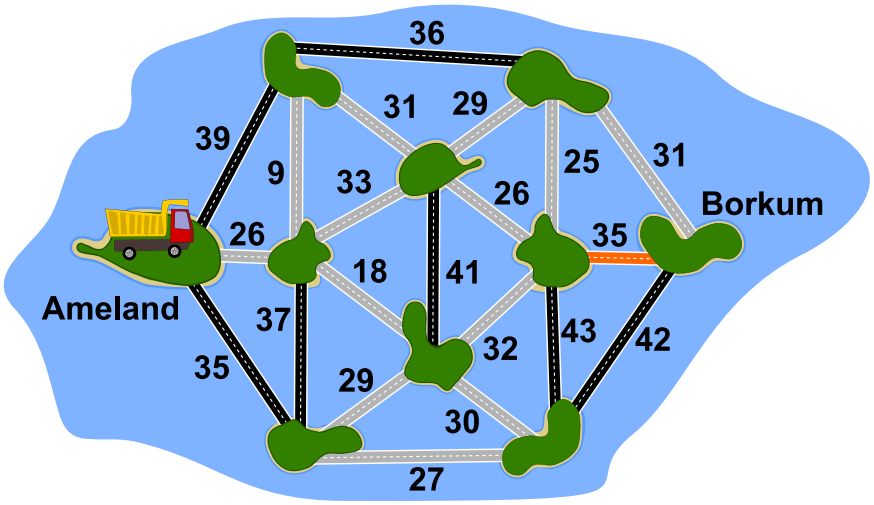
Per il viaggio, il peso totale massimo di un camion è di 32 tonnellate. Si prende il seguente percorso, ad esempio:



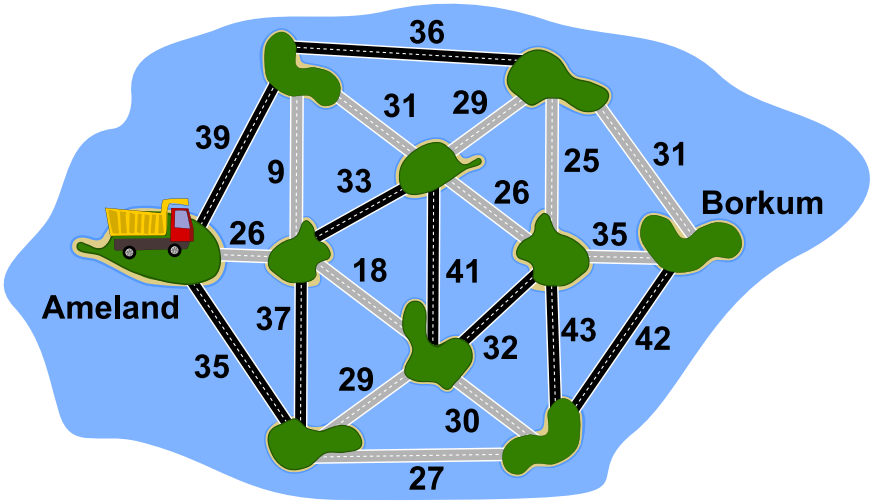
Per determinare questo, possiamo, ad esempio, per prima cosa togliere tutti i ponti dalla mappa e ordinarli in base alla loro capacità di carico. Iniziamo con quelli con la maggiore capacità di carico e li aggiungiamo alla mappa. Poi aggiungiamo quelli con la seconda maggior capacità di carico e così 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.

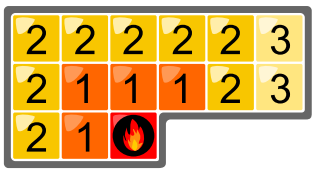
Per una soluzione, la rete può essere prima modellata, cioè semplificata, come qui presentato. Nel nostro caso, l'*algoritmo di Kruskal* crea un albero ricoprente massimo in cui il collo di bottiglia è direttamente visibile.

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`

15. Riscaldamento a pavimento

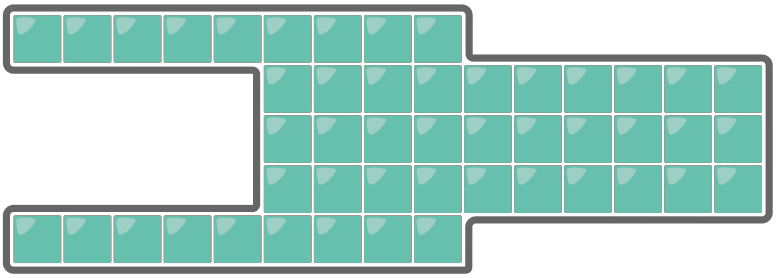
A Luis non piace vestirsi al mattino nel bagno freddo, quindi vuole che nella nuova casa venga installato il riscaldamento a pavimento. Il tecnico del riscaldamento gli consiglia l'innovativo riscaldamento a pavimento «hotspot»: un hotspot viene installato direttamente sotto una piastrella. Se l'hotspot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

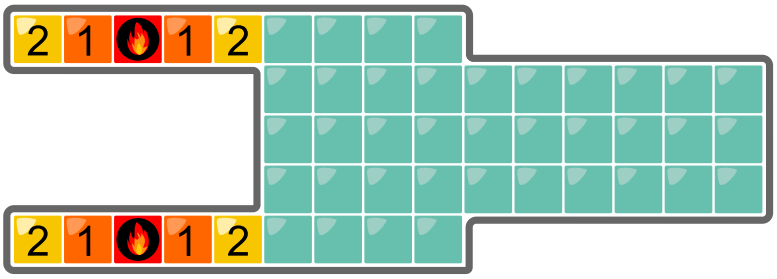


Soluzione

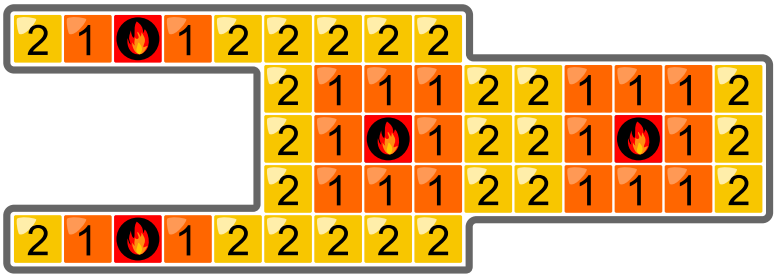
Se i 4 hotspot sono installati come mostrato nell'immagine sottostante, tutte le piastrelle del bagno si riscalderanno entro 2 minuti dall'accensione.

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.

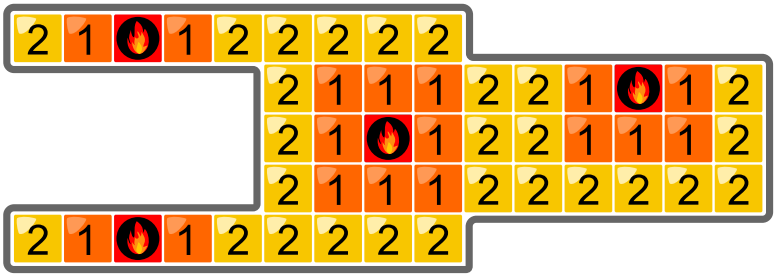
È una buona idea iniziare con i due corridoi a sinistra quando si distribuiscono gli hotspot. Con un punto caldo al centro di ogni corridoio, tutte le piastrelle del corridoio vengono riscaldate in 2 minuti:

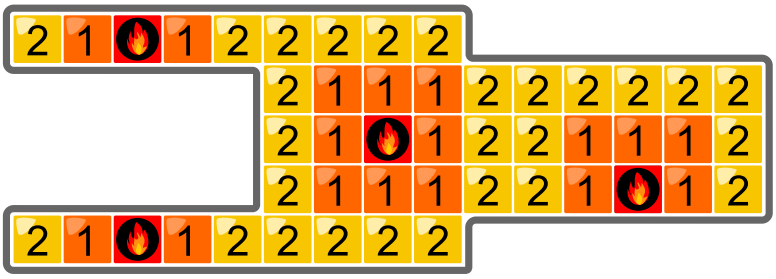


Possiamo poi collocare gli altri due hotspot in questo modo:



Sono possibili anche i seguenti due collocamenti:





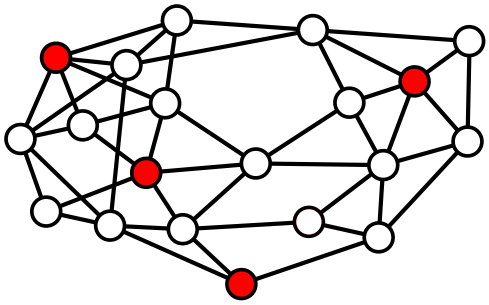
Se il bagno avesse una forma diversa, 2 hotspot potrebbero essere sufficienti per riscaldare l'intero bagno in 2 minuti con la stessa area.

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

A. Autori dei quesiti

Faisal Al-Sudani

Michael Barot

Carlo Bellettini

Linda Björk Bergsveinsdóttir

Maksim Bolonkin

Andrey Brodnik

Lucia Budinská

Špela Cerar

Sarah Chan

Marios O. Choudary

Kris Coolsaet

Valentina Dagienė

Christian Datzko

Susanne Datzko

Hanspeter Erni

Fabian Frei

Gerald Futschek

Jens Gallenbacher

Christian Giang

Yasemin Gulbahar

Mathias Hiron

Juraj Hromkovič

Tiberiu Iorgulescu

Takeharu Ishizuka

Mile Jovanov

Ungyeol Jung

Vaidotas Kinčius

Ritambhra Korpal

Regula Lacher

Marielle Léonard

Hiroki Manabe

Pedro Marcelino

Kwangsik Moon

Anna Morpurgo

Xavier Muñoz

Hiroyuki Nagataki

Vania Natali

Rana R. Natawigena

Andrei Nicolicioiu

Dejan Ozebek

Gabriel Parriaux

Jean-Philippe Pellet

Melinda Phelps

Margot Phillipps

Hannah Piper

Wolfgang Pohl

Prathyush Ponnekanti

Raymond Chandra Putra

Susannah Quidilla

Pedro Ribeiro

Chris Roffey

Peter Rossmanith

 Eljakim Schrijvers

 Vipul Shah

 Maiko Shimabuku

 Timur Sitdikov

 Emil Stankov

 Preethi Sudharsha

 Maciej M. Sysło

 Peter Tomcsányi

 Monika Tomcsányiová

 Troy Vasiga

 Michael Weigend

 Khairul Anwar Mohamad Zaki

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

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.