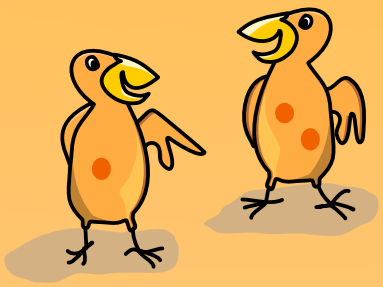
INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Quesiti e soluzioni 2021

11º al 13º anno scolastico



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

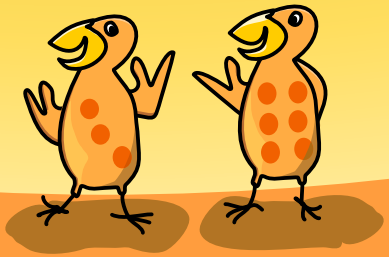
A cura di:

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

01010011010101100100100
10000001001011010101001
10101001101001001010001
00101101010100110101001
10100100101001001001001

SSII

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



Hanno collaborato al Castoro Informatico 2021

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

Capo progetto: Nora A. Escherle

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

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

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

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

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

Questi quaderni sono dedicati alla memoria di Martin Guggisberg.

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

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

INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

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

HASLERSTIFTUNG

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

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

Premessa

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

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

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

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

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

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

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

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

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

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

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

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

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

Per ulteriori informazioni:

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

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

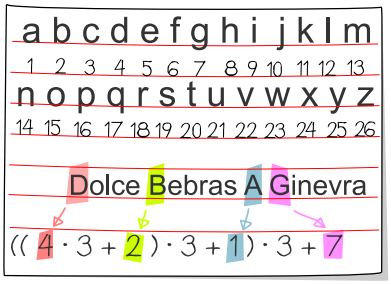
Indice

1. Biblioteca

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

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

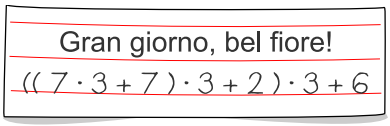
Prendi la prima lettera di ogni parola nel titolo e determina la sua posizione nell'alfabeto. Somma questo valore al valore della lettera precedente, ma prima di ogni somma, il valore totale raggiunto finora viene moltiplicato per 3. Il risultato per il libro desiderato è 136, è quindi chiaro dove si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



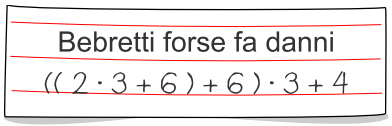
In seguito, Susi prepara i codici corrispondenti per recuperare i suoi libri preferiti. In un caso, tuttavia, ha commesso un errore.

Quale di questi codici è stato calcolato in modo errato?

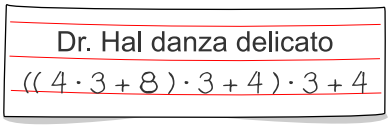
A)



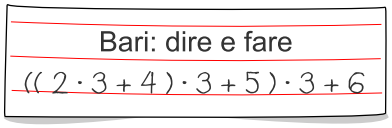
B)



C)

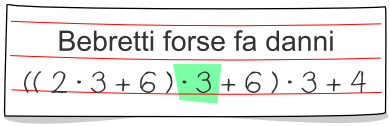


D)



Soluzione

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



Questa è l'informatica!

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

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

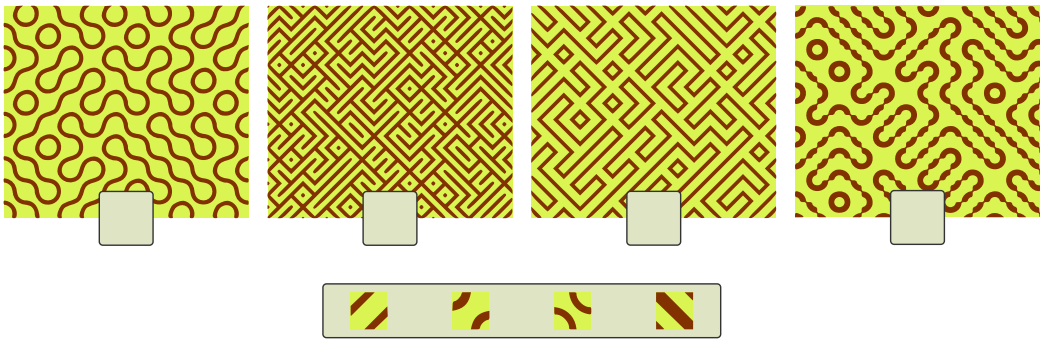
Parole chiave e siti web

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

2. Piastrelle Truchet

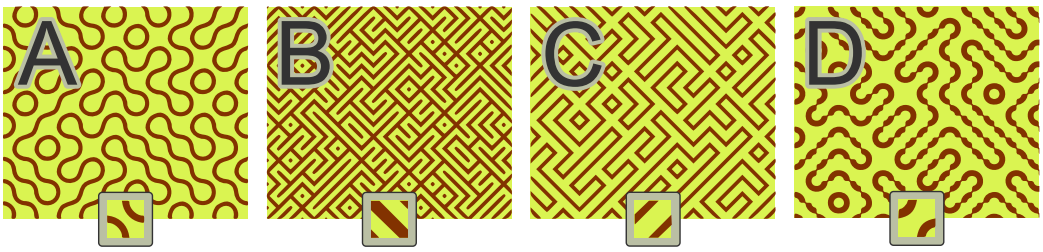
I seguenti modelli sono stati creati ciascuno da una singola piastrella. Le singole piastrelle sono mostrate ingrandite.

Abbina le piastrelle ai loro possibili modelli.

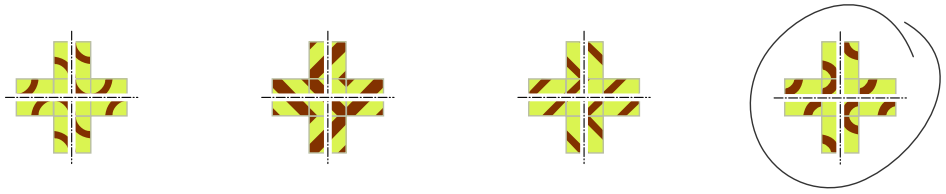


Soluzione

Questa è la classificazione corretta:



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



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

Questa è l'informatica!

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

Parole chiave e siti web

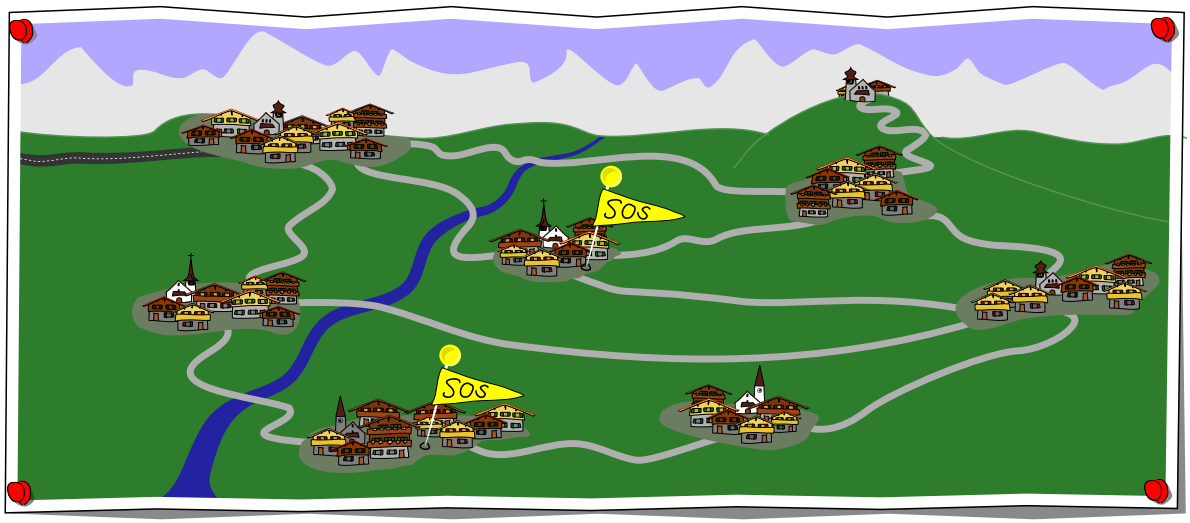
- Piastrelle Truchet: `https://it.wikinew.wiki/wiki/Truchet_tiles`

3. Villaggi isolati

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

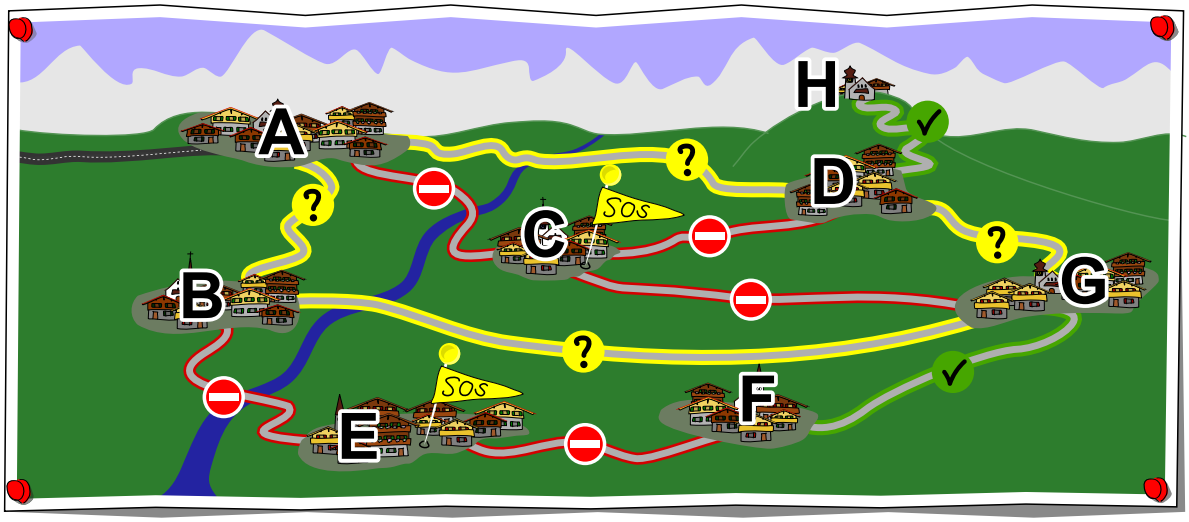
Dopo una tempesta, diversi villaggi segnalano che non sono più accessibili, in particolare quelli con i segnali di SOS. Possiamo concludere che alcune strade sono bloccate.

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



Soluzione

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



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

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

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

Questa è l'informatica!

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

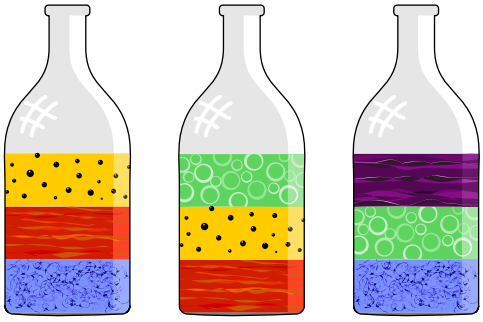
Sistemare i difetti in un sistema è un compito che gli informatici devono fare molto spesso, non solo nelle reti di computer ma anche nello sviluppo di software. Per correggere un errore è necessario identificare la sua fonte esatta, e questo processo è di solito fatto passo dopo passo in diverse fasi.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

Parole chiave e siti web

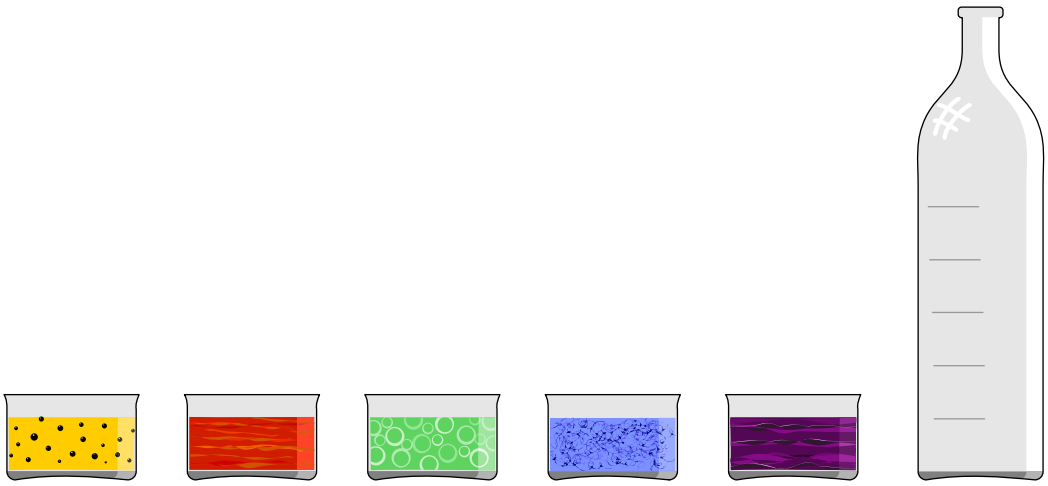
- Ridondanza: `https://it.wikipedia.org/wiki/Ridondanza_(ingegneria)`
- Debugging: `https://it.wikipedia.org/wiki/Debugging`

4. Disposizione dei liquidi

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

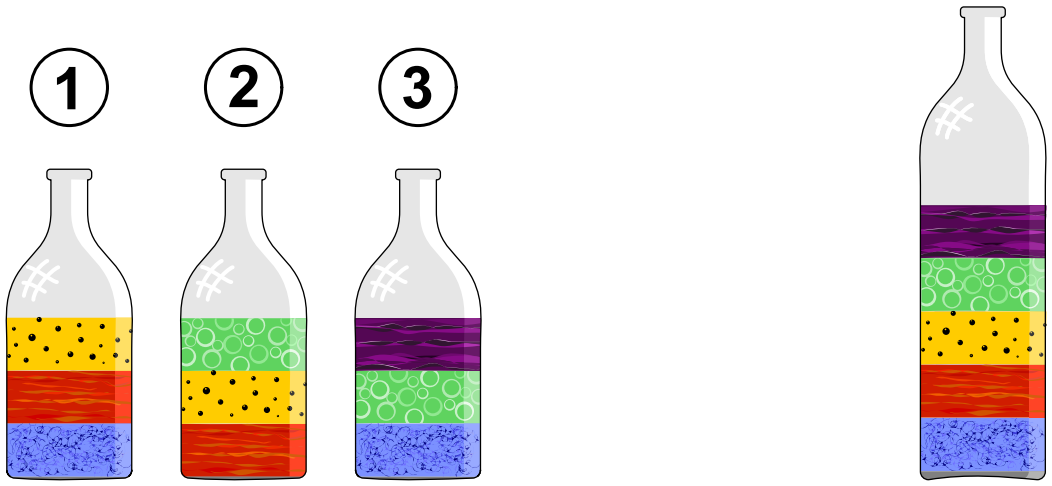


Disponi i cinque liquidi colorati nella bottiglia come apparirebbero dopo averli mischiati tutti!



Soluzione

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



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

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

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

Questa è l'informatica!

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

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

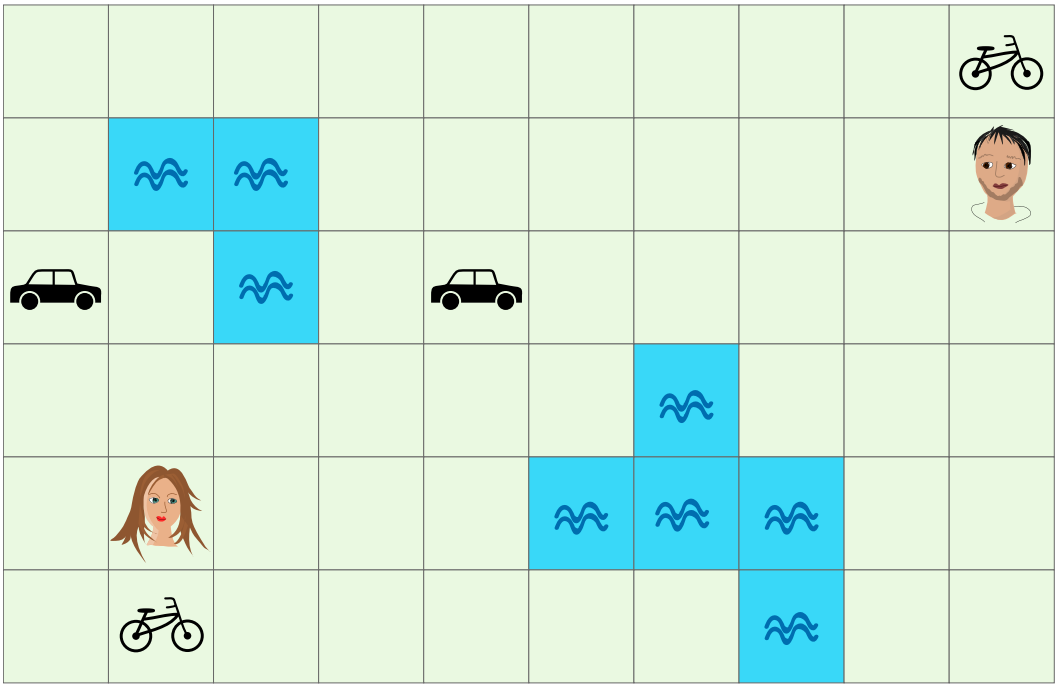
L'ordinamento dei dati gioca un ruolo importante in molti programmi informatici. Il metodo usato in questo compito per determinare l'ordine degli strati liquidi è chiamato *ordinamento topologico*. Si usa per ordinare gli oggetti per i quali sono conosciute relazioni del tipo predecessore/successore.

Parole chiave e siti web

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

5. Riunione veloce

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

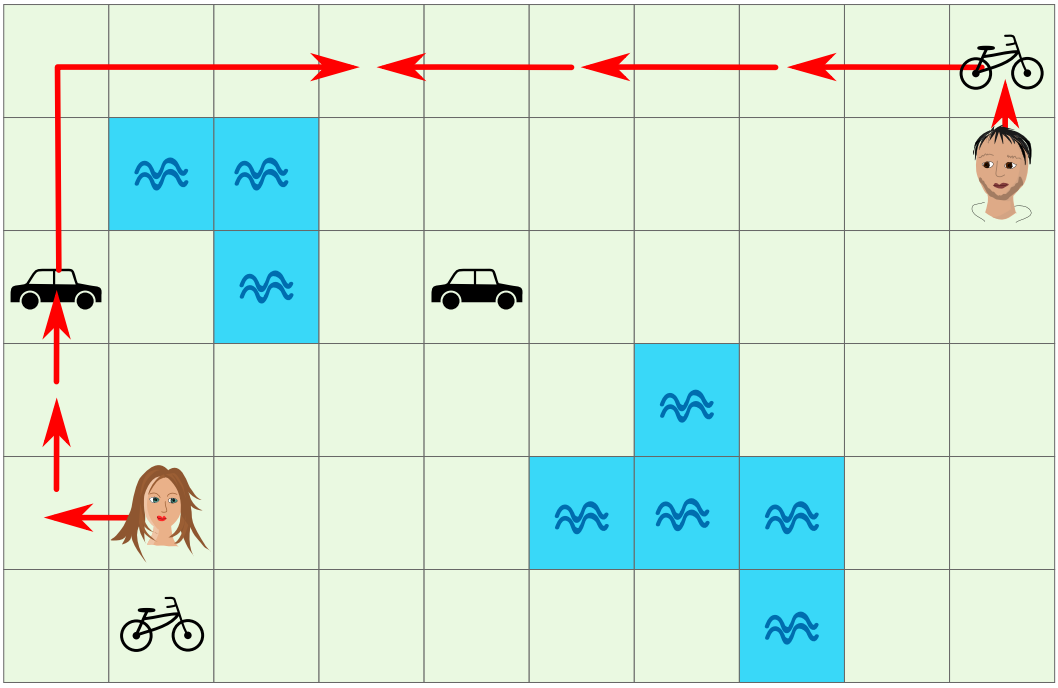


Di quanti minuti hanno bisogno i due amici per incontrarsi su una casella?

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

Soluzione

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



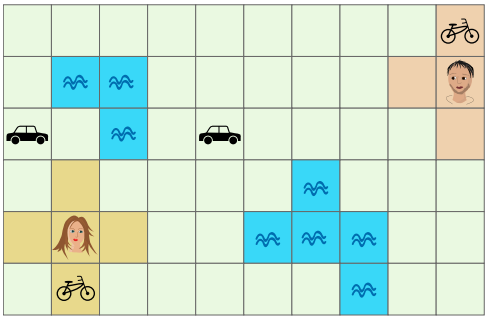
Ora bisogna dimostrare che non possono incontrarsi in 3 minuti: I due amici sono a 11 campi di distanza l'uno dall'altro. In 3 minuti, però, possono avvicinarsi a piedi solo un totale di 6 campi. Se uno ha raggiunto la bicicletta e l'altro sta camminando, allora possono avvicinarsi di 9 caselle l'uno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

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

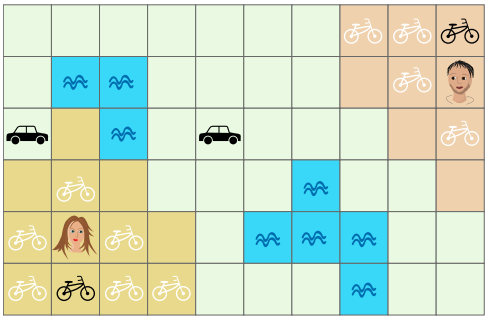
Questa è l'informatica!

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

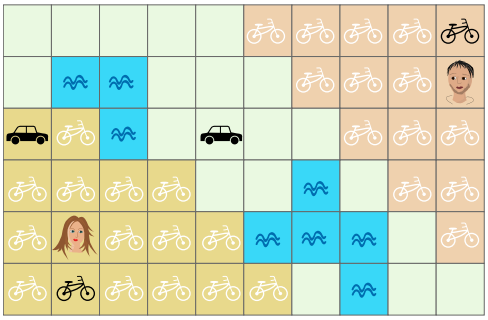
I programmi per computer progettati per questo tipo di problemi di solito usano un metodo chiamato *ricerca in ampiezza*. In questo problema, la ricerca in ampiezza funziona come segue:



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

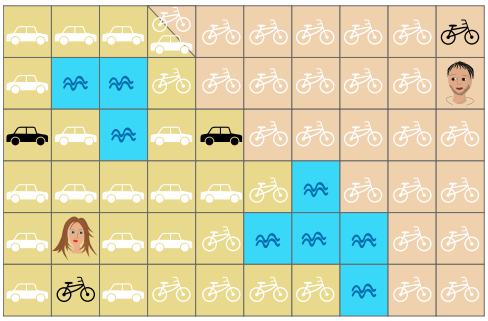


2. Segna tutti i campi che possono essere raggiunti in (al massimo) un minuto dai campi segnati nel passo 1. Annota anche i mezzi di trasporto utilizzati.



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

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



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

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

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

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

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

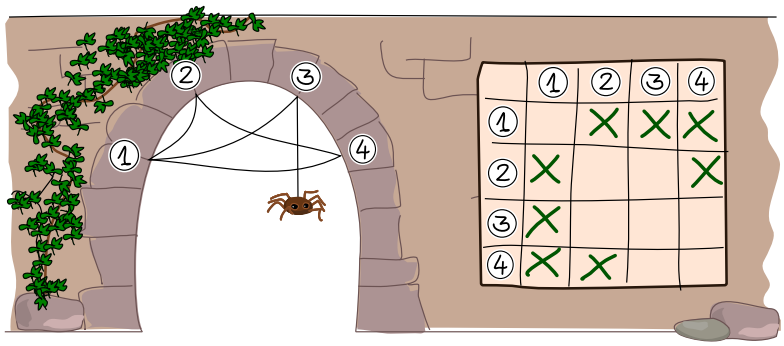
Se un problema diventa troppo complesso, ci vorrebbe troppo tempo anche per il computer più veloce del mondo per passare attraverso tutte le possibilità e trovare la soluzione migliore. In pratica, con un sistema di navigazione, è spesso sufficiente trovare un ottimo percorso, anche se non è il migliore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

Parole chiave e siti web

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

6. Le ragnatele di Thekla

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

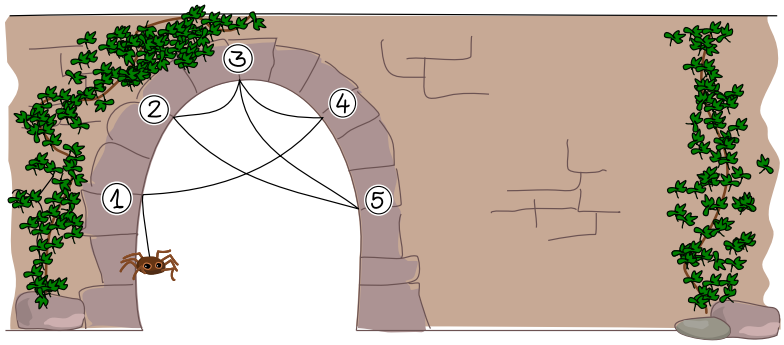


Il metodo funziona così: numera i punti finali della ragnatela da 1 a N e usa i campi in una griglia secondo la seguente regola:

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

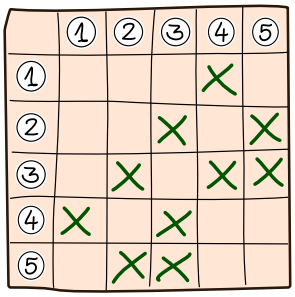
Un filo che collega il punto finale x con il punto finale y collega anche il punto finale y con il punto finale x.

Thekla costruisce ora questa ragnatela:

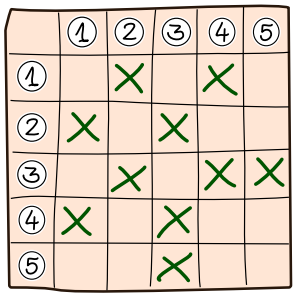


Come documenta Thekla la costruzione di questa ragnatela?

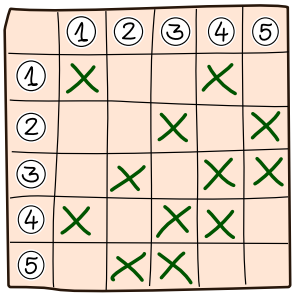
A)



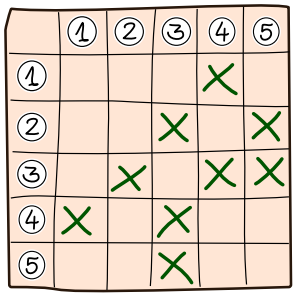
B)



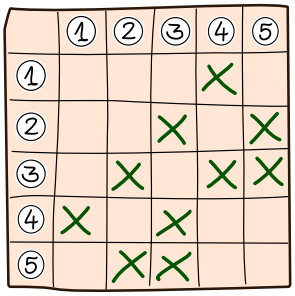
C)



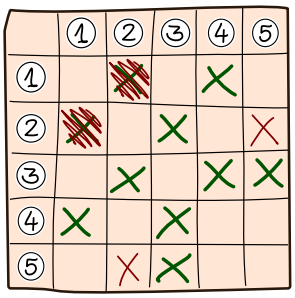
D)



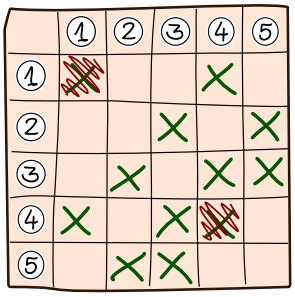
Soluzione



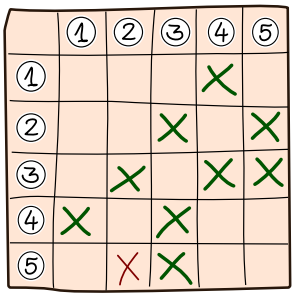
La risposta corretta è A, perché tutti i campi sono segnati correttamente secondo la regola.



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



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



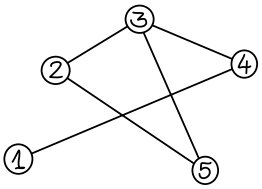
Risposta D: Tutte le rappresentazioni di ragnatele devono essere simmetriche rispetto alle diagonali dall'alto a sinistra al basso a destra. In questa risposta, la connessione dal punto finale 2 al punto finale 5 è presente, ma manca la connessione corrispondente di ritorno dal punto finale 5 al punto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

Questa è l'informatica!

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

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

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



	1	2	3	4	5
1	0	0	0	1	0
2	0	0	1	0	1
3	0	1	0	1	1
4	1	0	1	0	0
5	0	1	1	0	0

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

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

Parole chiave e siti web

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

7. Pila di frutta

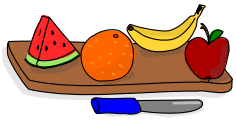
Papà, mamma, Dorie e Ron Castoro preparano la sera quattro scatole per la colazione del giorno dopo, ognuna con un frutto diverso: mela, banana, arancia e anguria. Le scatole sono impilate l'una sull'altra nel frigorifero. Al mattino, i castori sono ancora molto stanchi e quando lasciano la tana prendono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

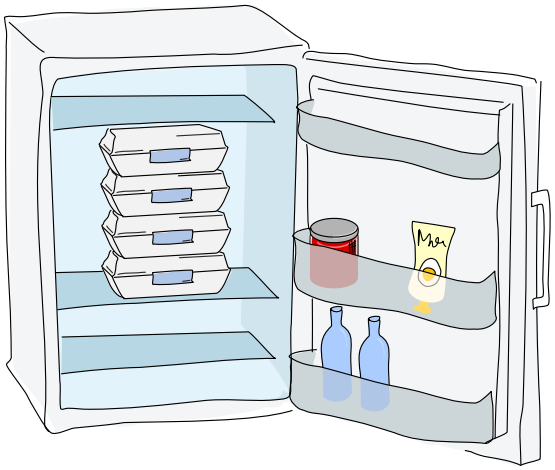
Non sappiamo esattamente in quale ordine i castori lasciano la tana, ma in ogni caso Mamma va sempre prima di Dorie e Papà sempre come l'ultimo.

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

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

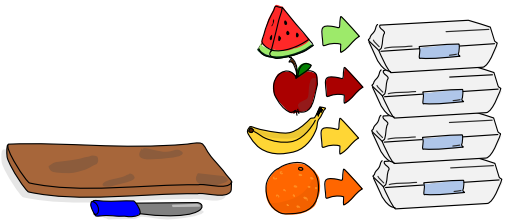
Metti le frutte nelle scatole in modo che tutti i castori prendano una scatola contenente un frutto che gli piace.





Soluzione

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



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

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

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

Questa è l'informatica!

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

Ingresso A
Ingresso B
Calcolare $X = A + B$
Calcolare $Y = A - B$
Calcolare $Z = X \times Y$

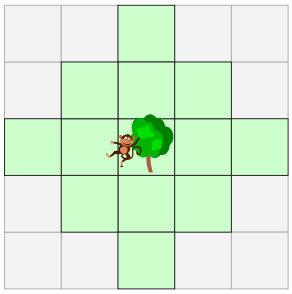
Tuttavia, se si cerca di eseguire il passo di calcolo $Z = X \times Y$, per esempio, prima di aver calcolato X, questo porta a un errore e il programma viene interrotto. Oppure viene usato un valore predefinito

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

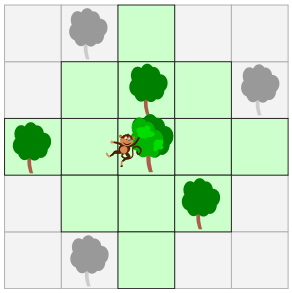
Parole chiave e siti web

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

8. La scimmia Coco



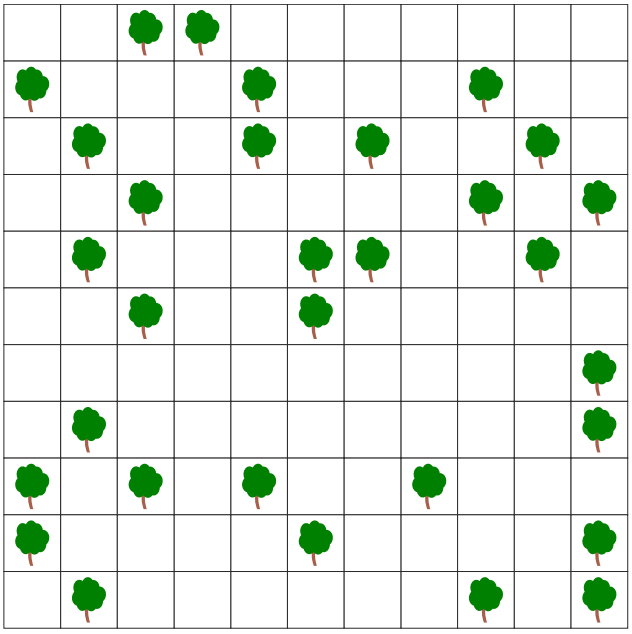
La scimmia arrampicatrice Coco può saltare da un albero e raggiungere qualsiasi luogo nell'area verde.



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

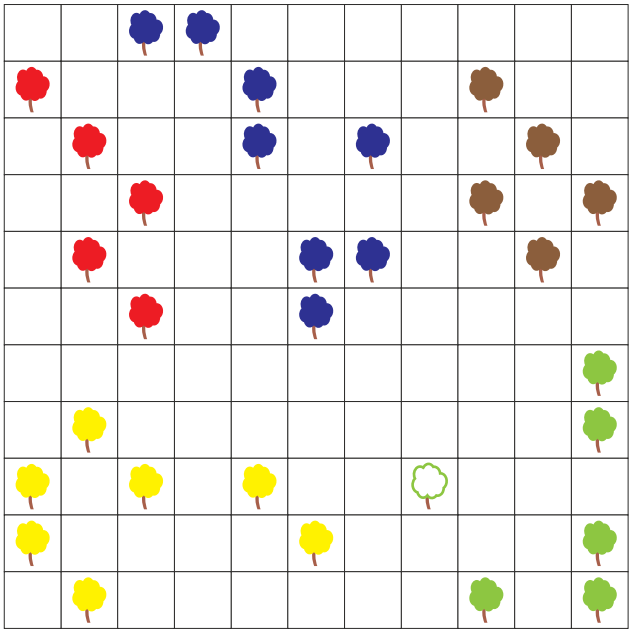
Ci sono gruppi di alberi tra i quali Coco può muoversi a volontà con diversi salti senza mai toccare il suolo.

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



Soluzione

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

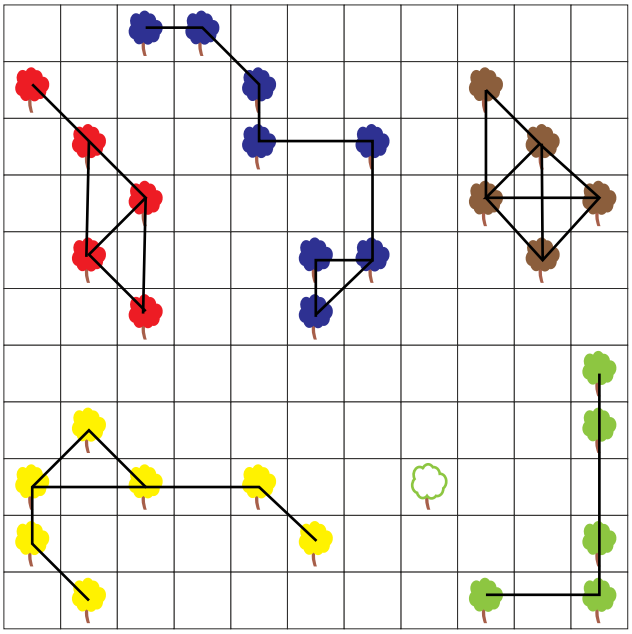


Vediamo che il gruppo di alberi blu con i suoi otto alberi è il più grande.

Questa è l'informatica!

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

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



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

Parole chiave e siti web

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

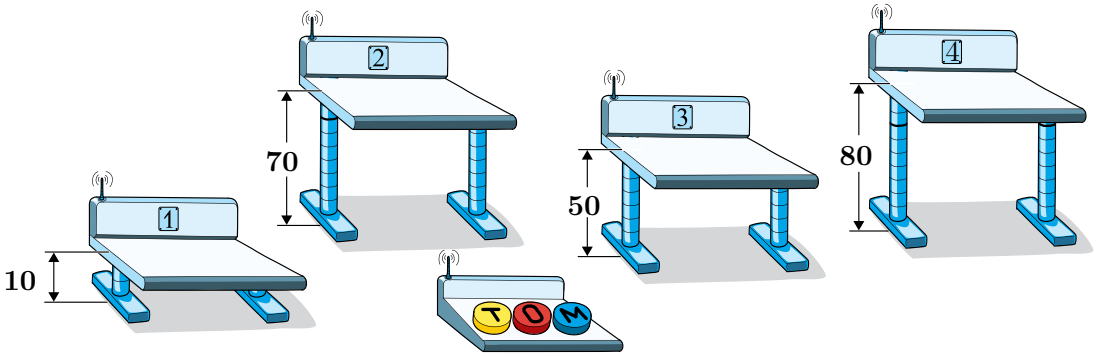
9. Maledette scrivanie

Un'aula ha scrivanie con altezza regolabile elettricamente. Per le lezioni, tutte le scrivanie devono essere impostate a un'altezza di 60 cm. L'altezza delle scrivanie può essere modificata con i pulsanti T, O e M di un telecomando. Qualcuno ha giocato con il telecomando e lo ha riprogrammato. Ora i tre pulsanti funzionano come segue:

- T solleva le scrivanie 1, 2 e 3 di 10 cm.
- O abbassa le scrivanie 2, 3 e 4 di 10 cm.
- M solleva le scrivanie 1, 3 e 4 di 10 cm.

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

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



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

A) Premere 4 × T, 5 × O e 1 × M.
B) Premere 5 × T, 1 × O e 0 × M.
C) Premere 3 × T, 4 × O e 2 × M.
D) Premere 2 × T, 4 × O e 6 × M.

Soluzione

La risposta corretta è C) Premere 3 × T, 4 × O e 2 × M. Sul telecomando, si nota che tutti i tre pulsanti cambiano l'altezza di 10 cm ogni volta che vengono premuti, cioè sempre della stessa misura. Due dei pulsanti sollevano le scrivanie (T e M) e solo un pulsante le abbassa (O). Inoltre, tutti i tre tasti cambiano l'altezza di tre scrivanie ciascuno, quindi c'è sempre una scrivania la cui altezza rimane invariata. Il tasto O non ha alcuna influenza sul tavolo 1, quindi non possiamo abbassare il tavolo 1.

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

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

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

Tuttavia, dobbiamo ancora verificare se la soluzione C) dia anche l'altezza corretta per la scrivania 4. La scrivania 4 è 20 cm troppo alta e T non ha alcuna influenza sull'altezza di essa. Quindi O deve essere premuto due volte e per ogni pressione di M è necessaria un'ulteriore pressione di O. Con la sequenza di tasti della soluzione C), l'altezza della scrivania 4 finisce per essere $80 - 40 + 20 = 60$ cm.

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

In alternativa, la soluzione può essere cercata con quattro equazioni lineari. Per ogni scrivania, si scrive con un'equazione definendo quali pulsanti cambiano l'altezza della scrivania e quale sia il cambiamento di altezza che stiamo cercando. Per esempio, l'altezza della scrivania 1 cambia solo con T e M e la regolazione dell'altezza desiderata è di 50 cm, che può essere ottenuta con 5 pressioni di tasti (perché sono 10 cm per pressione di tasti).

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

$$\begin{aligned} T + M &= 5 \\ T - O &= -1 \\ T - O + M &= 1 \\ -O + M &= -2 \end{aligned}$$

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

Questa è l'informatica!

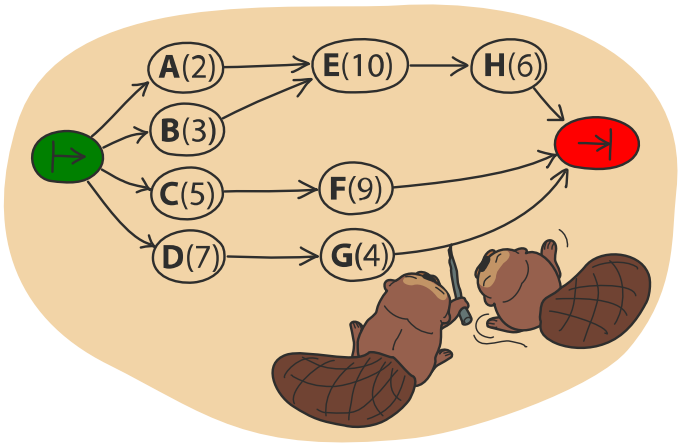
Questo è un compito tipico del campo dell'*ottimizzazione discreta*, più precisamente della *programmazione lineare*. Questo compito è dato da un insieme di costrizioni. In questo caso speciale, possono essere tutte formulate come equazioni lineari. L'obiettivo è tipico dell'informatica: si cerca una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

Parole chiave e siti web

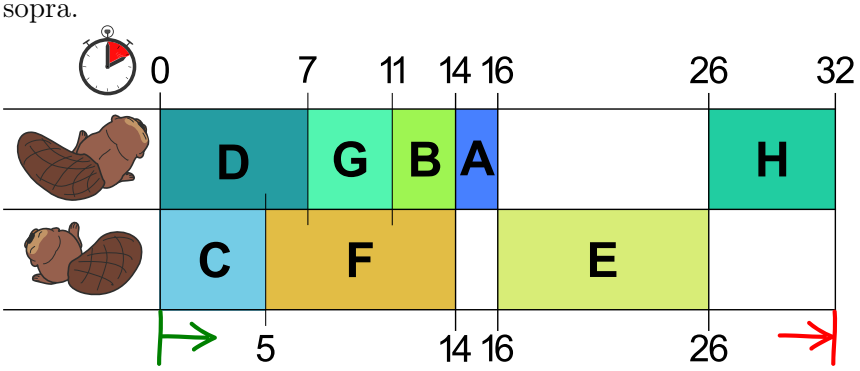
- Ottimizzazione lineare in numeri interi:
  `https://it.frwiki.wiki/wiki/Optimisation_linéaire_en_nombres_entiers`

10. Piano di lavoro dei castori

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

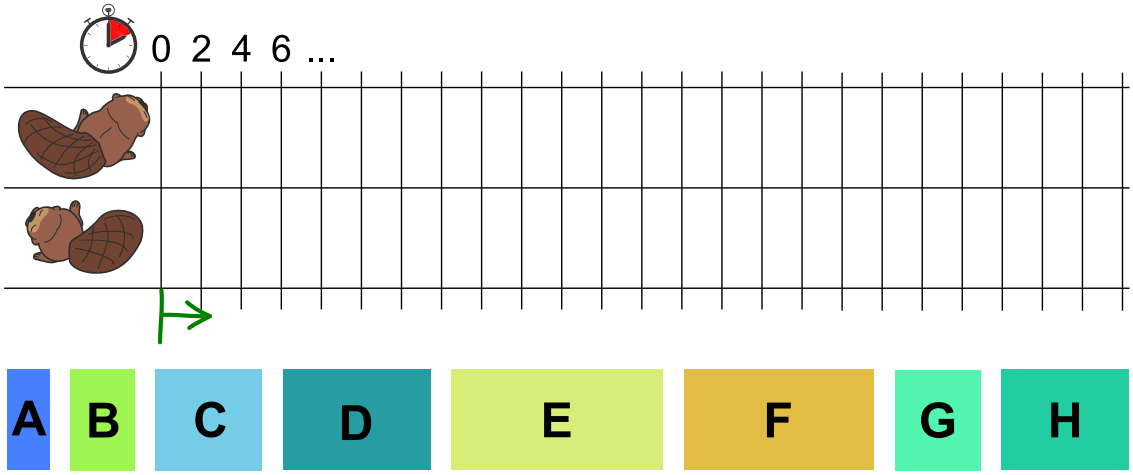


La castora Ulla vuole aiutare il castoro Otso a costruire la diga più velocemente. Dividono i sottocompiti tra di loro e creano il seguente piano di lavoro che soddisfa le dipendenze dell'immagine sopra.



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

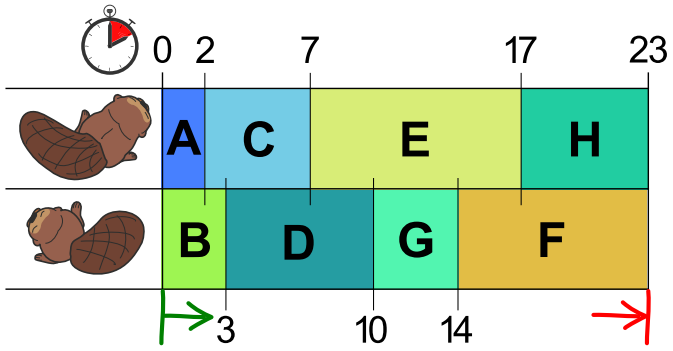
Crea un piano di lavoro per finire la diga nel minor tempo possibile.



Soluzione

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

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



Questa è l'informatica!

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

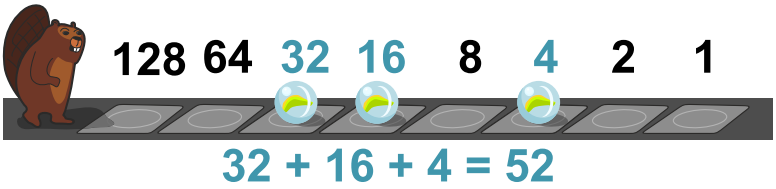
Per tali *problemi di scheduling* ci sono molte strategie diverse che sono state ben studiate dall'informatica. Il primo programma di questo compito è stato creato in modo tale che tra i sottocompiti in sospeso, quello con la durata più lunga è stato assegnato a un castoro attualmente disoccupato - in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

Parole chiave e siti web

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

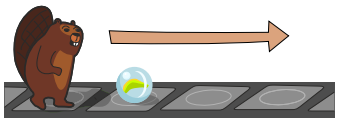
11. Numeri di biglia

I castori hanno un modo speciale di rappresentare i numeri.



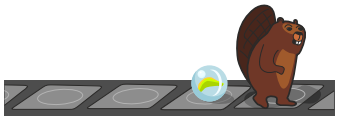
I diversi campi hanno pesi diversi e una biglia sul campo determina che il valore viene preso in consegna. Nell'esempio qui sopra, viene mostrato il numero 52.

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



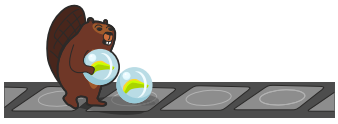
Ogni volta che il castoro incontra un campo con una biglia e ha le mani libere, raccoglie la biglia e la porta con sé.

Al primo campo libero, il castoro posa la biglia.



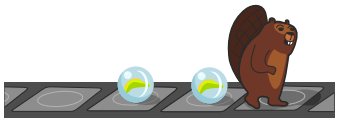
Il castoro può portare solo una biglia alla volta e c'è spazio solo per una biglia su ogni campo.

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



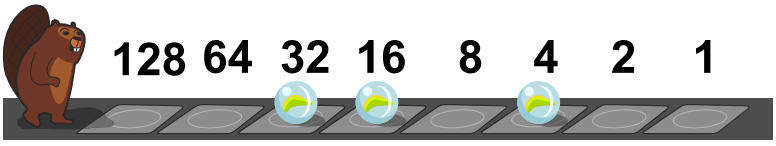
... poi lui le passa accanto ...

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



Dopodiché, può raccogliere la biglia successiva.

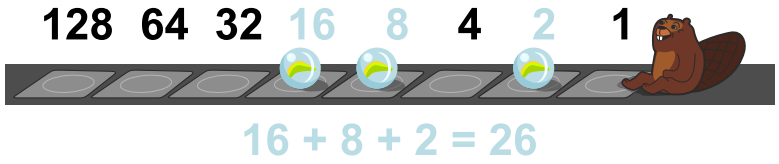
Quale numero è rappresentato dalle biglie quando il castoro ha attraversato l'area?



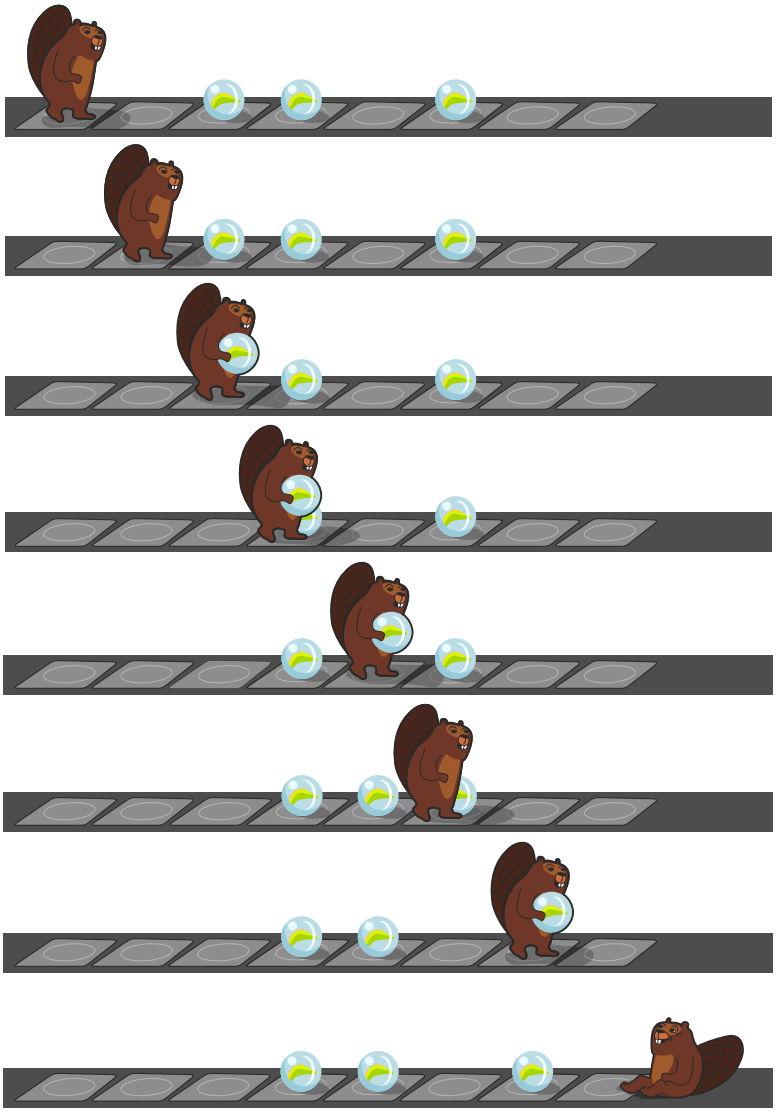
A) 10
B) 26
C) 28
D) 104

Soluzione

La risposta corretta è B) 26



La figura seguente mostra il processo:



Questa è l'informatica!

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

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

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

Una macchina di Turing è composta da:

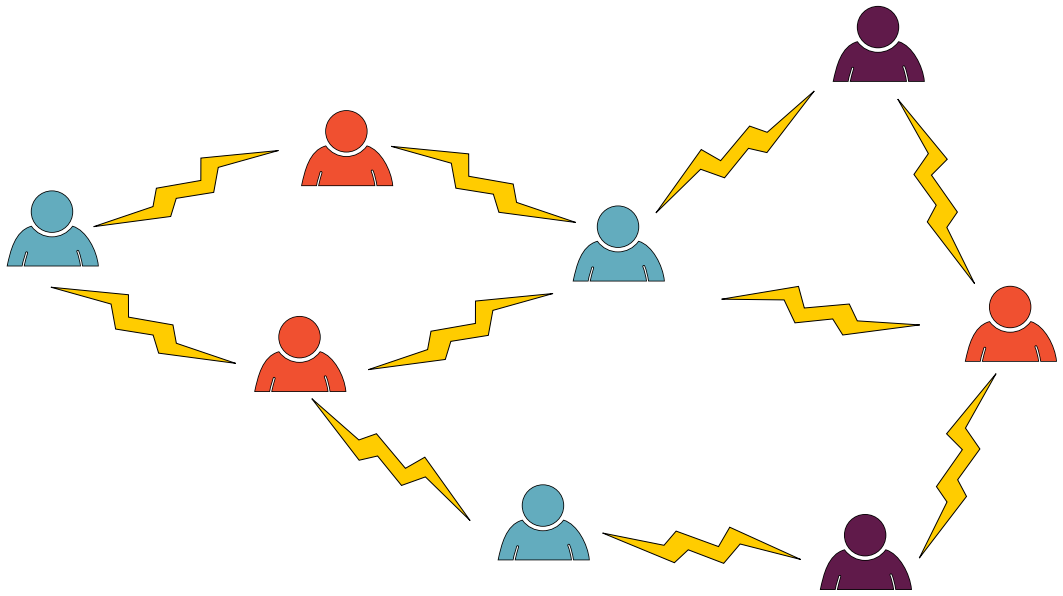
- Un *nastro* di qualsiasi lunghezza, composto da singoli *campi*. Ogni campo può contenere un *simbolo*. Nel nostro esempio, questi sono i quadrati su cui si muove il castoro.
- Un insieme finito di *simboli*. Spesso solo `0` e `1` sono usati come simboli. Nel nostro esempio, una biglia sta per `1`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

Parole chiave e siti web

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

12. Lavoro di gruppo

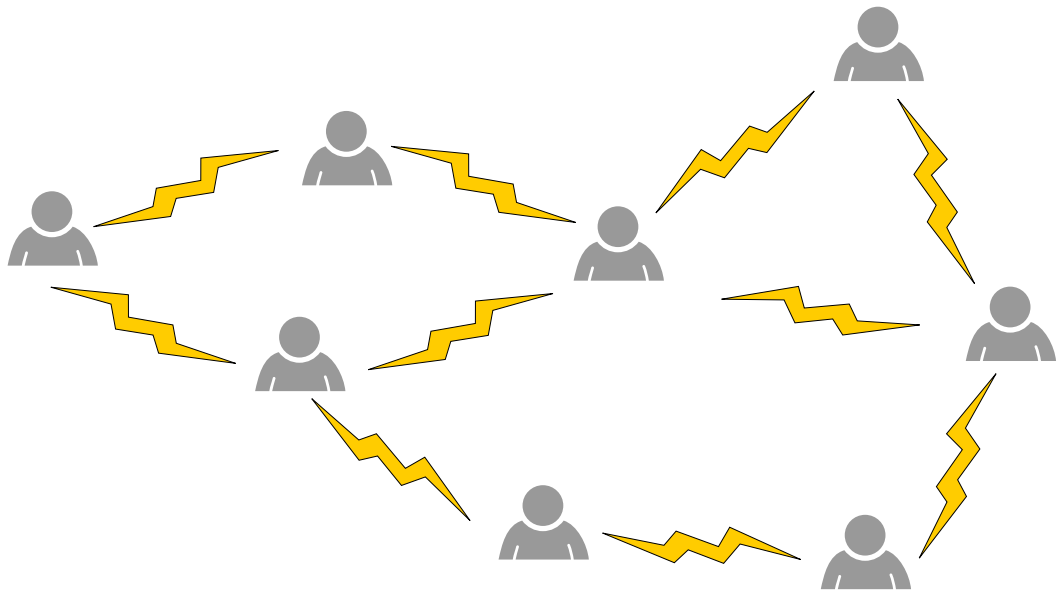
Per un progetto devi dividere otto persone in gruppi. C'è un fulmine tra due persone se non vogliono lavorare insieme. Quindi è meglio non metterli nello stesso gruppo.



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

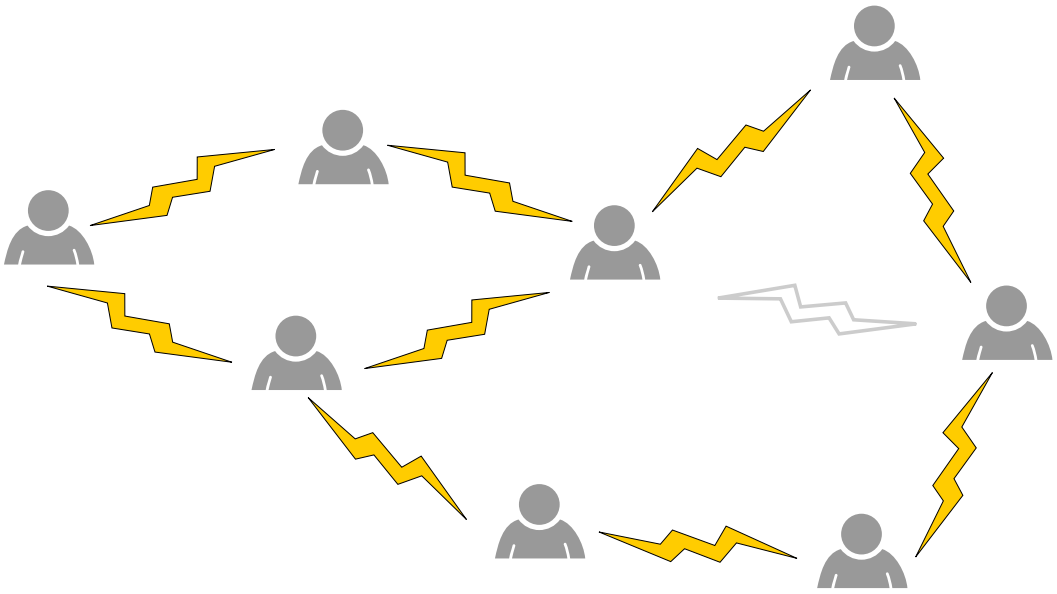
Se si convincono le due persone giuste a collaborare (cioè a togliere un fulmine), allora anche una divisione in due soli gruppi (solo due colori) è possibile.

Rimuovi il fulmine giusto.

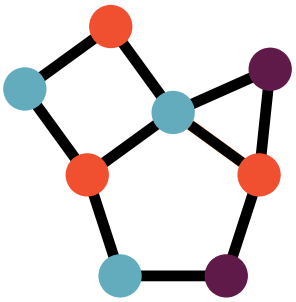


Soluzione

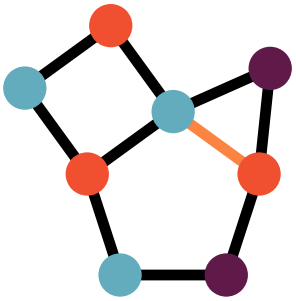
La soluzione corretta è:



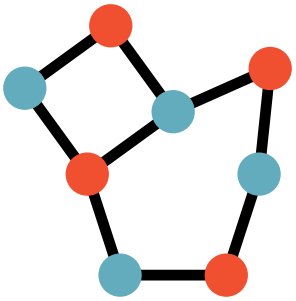
Rappresentiamo la situazione un po' più astrattamente come un cosiddetto *grafo*, con le persone come *vertici* (nodi) e i fulmini come *archi* (bordi).



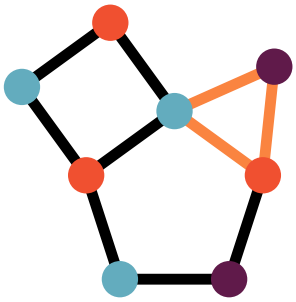
L'unica opzione possibile è l'arco marcato in arancione.



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

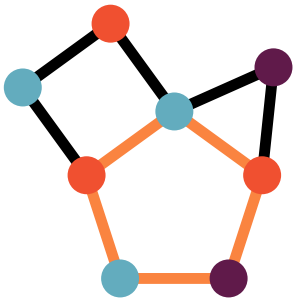


Ogni colore rappresenta un gruppo. Vediamo che da nessuna parte due persone dello stesso gruppo si rifiutano di cooperare: i vertici vicini hanno ovunque colori diversi.



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

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



Guardiamo ora il pentagono segnato in arancione:

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

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

Questa è l'informatica!

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

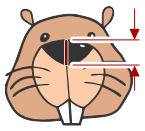
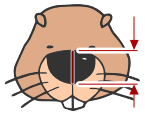
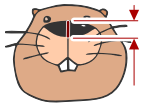
Parole chiave e siti web

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

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

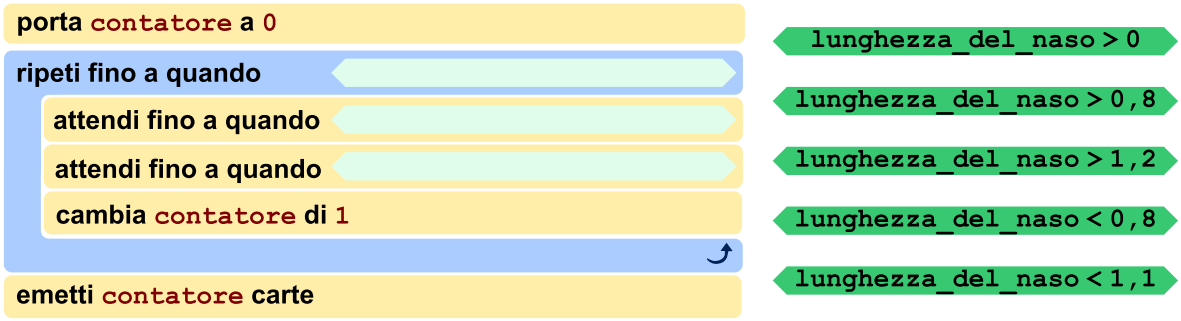
13. Contare con un cenno

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

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

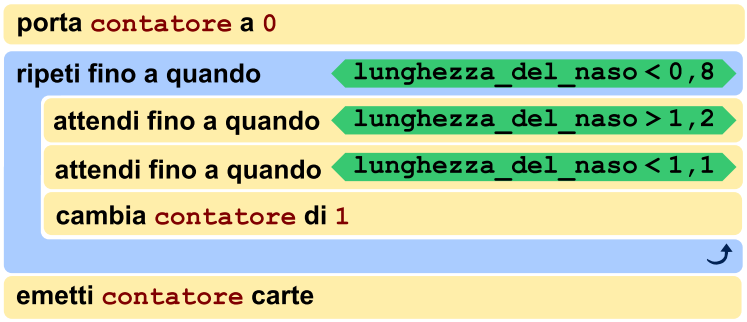
Il programma di controllo è quasi pronto - vedi sotto.

Completa il programma di controllo!



Soluzione

L'unica risposta corretta è:



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

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

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

Questa è l'informatica!

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

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

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

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

Parole chiave e siti web

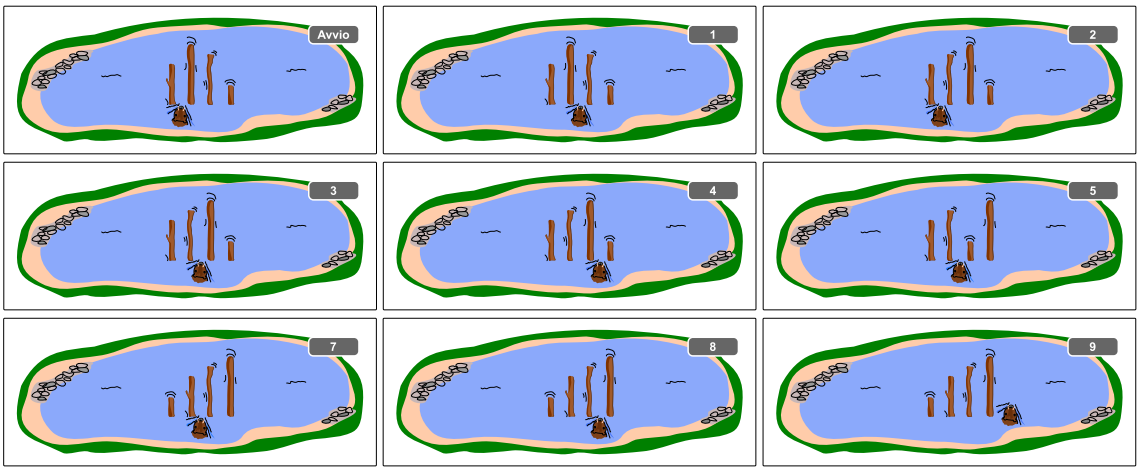
- Visione artificiale, Machine Vision, Computer Vision:
  `https://it.wikipedia.org/wiki/Visione_artificiale`
- Apprendimento automatico, Machine Learning:
  `https://it.wikipedia.org/wiki/Apprendimento_automatico`
- `https://it.wikipedia.org/wiki/Stephen_Hawking`
- `https://it.wikipedia.org/wiki/Voltapagine`

14. Beaver Sort

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

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

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



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

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

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

Soluzione

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

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

Questa è l'informatica!

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

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

Parole chiave e siti web

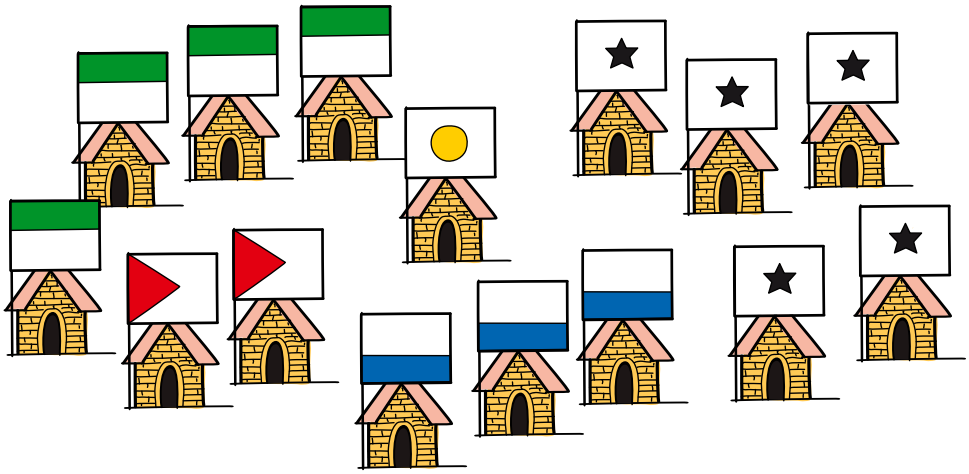
- Gnome Sort: `https://it.wikipedia.org/wiki/Gnome_sort`
- Algoritmo di ordinamento: `https://it.wikipedia.org/wiki/Algoritmo_di_ordinamento`
- Analisi di tempo di esecuzione
- O-grande: `https://it.wikipedia.org/wiki/O-grande`

15. Clan di Castoria

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

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

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



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

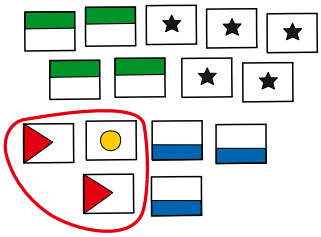
A) 15 settimane
B) 33 settimane
C) 35 settimane
D) 50 settimane
E) 120 settimane

Soluzione

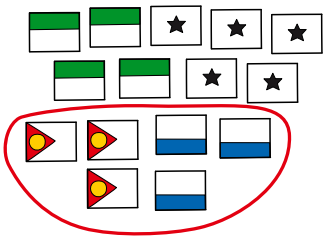
La risposta corretta è B) 33 settimane.

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

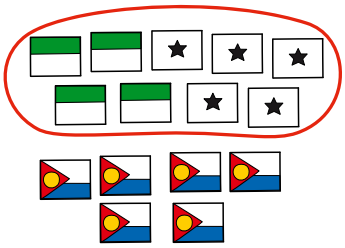
Questo processo passo dopo passo è mostrato nella tabella seguente. Per motivi di chiarezza, ometteremo i nomi dei clan e citeremo solo le dimensioni dei clan tra parentesi:



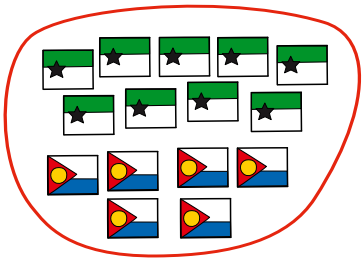
(1) e (2) sono uniti in un unico clan (3), il che richiede tre settimane.



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



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



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

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

Questa è l'informatica!

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

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

Parole chiave e siti web

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

A. Autori dei quesiti

Michael Barot
Liam Baumann
Wilfried Baumann
Lucia Budinská
Sarah Chan
Anton Chukhnov
Kris Coolsaet
Valentina Dagienė
Christian Datzko
Susanne Datzko
Janez Demšar
Fabian Frei
Gerald Futschek
Jens Gallenbacher
Thomas Galler
Christian Giang
Mark Edward M. Gonzales
Martin Guggisberg
Mathias Hiron
Juraj Hromkovič
Svetlana Jaksic
Martin Kandlhofer
Ulrich Kiesmüller
Dong Yoon Kim
Vaidotas Kinčius
Regula Lacher
Taina Lehtimäki
Angélica Herrera Loyo
Tom Naughton
Mochammad Irfan Noviana
Gabriela Lourdes Rodríguez Parada
Jean-Philippe Pellet
Hannah Piper
Jonatan Pipping
Zsuzsa Pluhár
Wolfgang Pohl
Rodrigo Santamaría
Tomas Šiaulys
Bernadette Spieler
Cuttle.org Team
Ezra Templonuevo
Eslam Wageed
Michael Weigend
Mija Zaļūksne

B. Sponsoring: concorso 2021

http://www.haslerstiftung.ch/

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

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

Standortförderung beim Amt für Wirtschaft und Arbeit Kanton Zürich

i-factory (Musée des transports, Lucerne)

http://www.ubs.com/

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

https://educatec.ch/
educaTEC

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

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

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

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

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

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

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

C. Ulteriori offerte

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

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

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