INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Quesiti e soluzioni 2018
11º al 13º anno scolastico

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

A cura di:
Andrea Adamoli, Christian Datzko, Susanne Datzko, Hanspeter Erni

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

Hanno collaborato al Castoro Informatico 2018

Andrea Adamoli, Christian Datzko, Susanne Datzko, Olivier Ens, Hanspeter Erni, Martin Guggisberg, Carla Monaco, Gabriel Parriaux, Elsa Pellet, Jean-Philippe Pellet, Julien Ragot, Beat Trachsler.

Un particolare ringraziamento va a:
Juraj Hromkovič, Urs Hauser, Regula Lacher, Jacqueline Staub: ETHZ
Andrea Maria Schmid, Doris Reck: PH Luzern
Gabriel Thullen: Collège des Colombières
Valentina Dagienė: Bebras.org
Hans-Werner Hein, Ulrich Kiesmüller, Wolfgang Pohl, Kirsten Schlüter, Michael Weigend: Bundesweite Informatikwettbewerbe (BWINF), Germania
Chris Roffey: University of Oxford, Regno Unito
Anna Morpurgo, Violetta Lonati, Mattia Monga: ALaDDIn, Università degli Studi di Milano, Italia
Gerald Futschek, Wilfried Baumann: Oesterreichische Computer Gesellschaft, Austria
Zsuzsa Pluhár: ELTE Informatikai Kar, Ungheria
Eljakim Schrijvers, Daphne Blokhuis, Arne Heijenga, Dave Oostendorp, Andrea Schrijvers: Eljakim Information Technology bv, Paesi Bassi
Roman Hartmann: hartmannGestaltung (Flyer Castoro Informatico Svizzera)
Christoph Frei: Chragokyberneticks (Logo Castoro Informatico Svizzera)
Andrea Adamoli (pagina web)
Andrea Leu, Maggie Winter, Brigitte Maurer: Senarclens Leu + Partner

L'edizione dei quesiti in lingua tedesca è stata utilizzata anche in Germania e in Austria.
La traduzione francese è stata curata da Nicole Müller e Elsa Pellet mentre quella italiana da Andrea Adamoli.

INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Il Castoro Informatico 2018 è stato organizzato dalla Società Svizzera per l'Informatica nell'Insegnamento SSII. Il Castoro Informatico è un progetto della SSII con il prezioso sostegno della fondazione Hasler.

HASLERSTIFTUNG

Nota: Tutti i link sono stati verificati l'01.11.2018. Questo quaderno è stato creato il 9 ottobre 2019 col sistema per la preparazione di testi LaTeX.

Premessa

Il concorso del "Castoro Informatico", presente già da diversi anni in molti paesi europei, ha l'obiettivo di destare l'interesse per l'informatica nei bambini e nei ragazzi. In Svizzera il concorso è organizzato in tedesco, francese e italiano dalla Società Svizzera per l'Informatica nell'Insegnamento (SSII), con il sostegno della fondazione Hasler nell'ambito del programma di promozione "FIT in IT".
Il Castoro Informatico è il partner svizzero del Concorso "Bebras International Contest on Informatics and Computer Fluency" (`https://www.bebras.org/`), situato in Lituania.
Il concorso si è tenuto per la prima volta in Svizzera nel 2010. Nel 2012 l'offerta è stata ampliata con la categoria del "Piccolo Castoro" (3º e 4º anno scolastico).
Il "Castoro Informatico" incoraggia gli alunni ad approfondire la conoscenza dell'Informatica: esso vuole destare interesse per la materia e contribuire a eliminare le paure che sorgono nei suoi confronti. Il concorso non richiede alcuna conoscenza informatica pregressa, se non la capacità di "navigare" in Internet poiché viene svolto online. Per rispondere alle domande sono necessari sia un pensiero logico e strutturato che la fantasia. I quesiti sono pensati in modo da incoraggiare l'utilizzo dell'informatica anche al di fuori del concorso.
Nel 2018 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
Andrea Adamoli
https://www.castoro-informatico.ch/it/kontaktieren/
https://www.castoro-informatico.ch/
https://www.facebook.com/informatikbiberch

Indice

1. Videogioco

Andrea ha programmato un videogioco a scuola. Le regole sono molto semplici.
Il gioco consiste in diversi turni. Ad ogni turno cade una foglia. Il castoro prova a prendere la foglia prima che cada a terra. Per vincere, il castoro deve prendere 15 foglie prima che 4 foglie possano cadere a terra.
La durata del gioco è misurata dal numero di turni.
Nell'esempio seguente, il castoro perde dopo 6 turni di gioco, perché viene raggiunto il numero massimo di 4 foglie cadute. La durata del gioco è dunque di 6 turni.

Turno	Esito	Punteggio – Numero totale di foglie	
		Prese	Cadute
1	presa	1	0
2	caduta	1	1
3	presa	2	1
4	caduta	2	2
5	caduta	2	3
6	caduta	2	4

Quanti turni può durare al massimo una partita?

A) 4 turni

B) 15 turni

C) 18 turni

D) 19 turni

E) 20 turni

F) Non esiste un limite di turni.

Soluzione

Per trovare la partita più lunga possibile, dobbiamo combinare tutte le situazioni in cui il gioco continua. Combiniamo quindi il numero massimo di foglie prese (14 turni) con il numero massimo di foglie lasciate cadere (3 turni) senza terminare il gioco. A questo punto, sia che catturiamo una foglia (totale 15), sia che la lasciamo cadere (totale 4) il gioco termina (vittoria, risp., sconfitta). Pertanto, la durata massima è $15 + 3 = 14 + 4 = 18$ turni e la risposta corretta è C).
La risposta A) "4 turni" sarebbe la durata minima del gioco in assoluto (nel caso che tutte le foglie cadessero a terra).
La risposta B) "15 turni" sarebbe la durata minima per vincere la partita (tutte le foglie sono prese).
Le risposte D), E) e F) sono errate, poiché il massimo delle foglie prese o il massimo delle foglie lasciate cadere viene raggiunto prima.

Questa è l'informatica!

Quando si programma un gioco, le regole devono essere chiaramente definite e i loro effetti devono essere pienamente compresi. Solo così possiamo garantire che si possa sempre giungere a una conclusione (vittoria o sconfitta) in un tempo opportuno. Ad esempio, nel nostro gioco dobbiamo essere sicuri di poter gestire un sufficiente numero di foglie.
Un gioco composto da diversi turni è di fatto un processo. Gli informatici sono gli specialisti della modellazione e nella descrizione dei processi. Uno dei compiti fondamentali è proprio capire cosa può accadere in ogni situazione e quanto tempo può richiedere un processo.

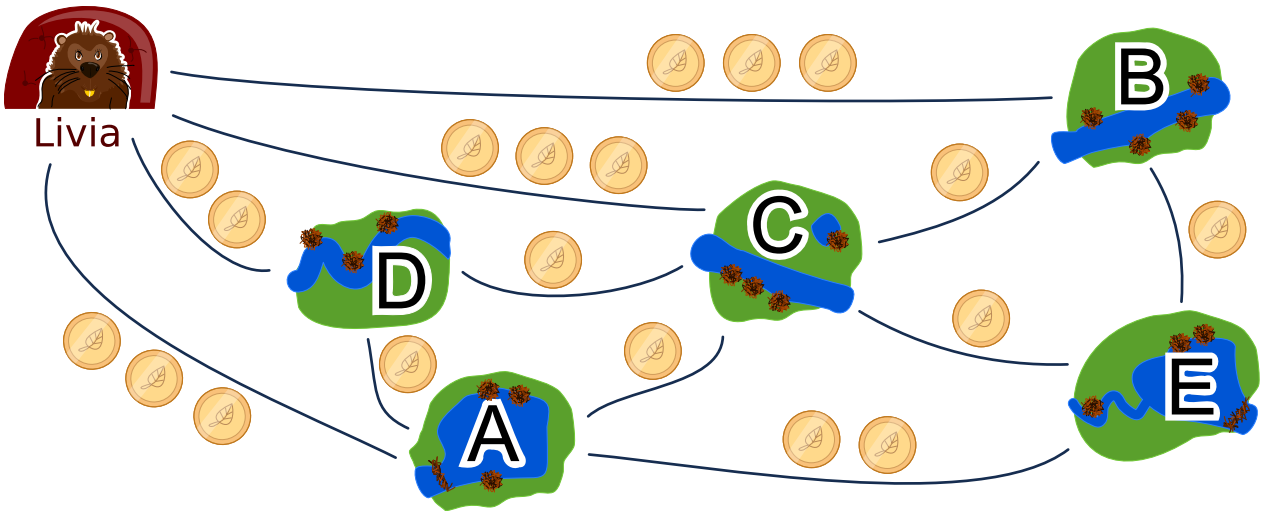
Parole chiave e siti web

analisi, verifica e validazione del software

- `https://en.wikipedia.org/wiki/Software_verification`
- `https://en.wikipedia.org/wiki/Verification_and_validation`

2. Visitare gli amici

Livia desidera visitare tutti i suoi amici nei villaggi A, B, C, D ed E con i mezzi pubblici. Livia vuole poter visitare tutti in un unico viaggio, senza dover passare dallo stesso villaggio più di una volta. Naturalmente, alla fine del suo viaggio, deve anche tornare a casa. La tariffa di ciascuna linea dei mezzi pubblici è mostrata nella figura.



Un possibile viaggio per visitare i suoi amici è:

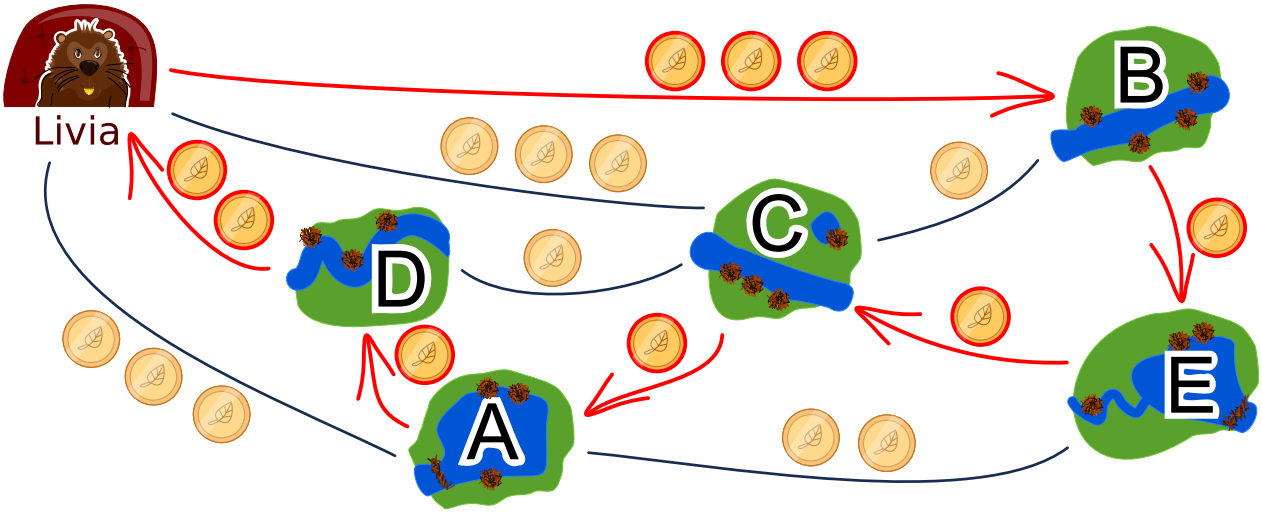
$$\text{Casa} \to \text{B} \to \text{E} \to \text{A} \to \text{D} \to \text{C} \to \text{Casa}.$$

Questo viaggio costerebbe $3 + 1 + 2 + 1 + 1 + 3 = 11$ monete.
Trova il viaggio più economico per Livia. Se esiste più di una soluzione ottimale, basta indicarne una.
In quale ordine Livia deve visitare gli amici?

Soluzione

Esistono due soluzioni ottimali:

- Casa → B → E → C → A → D → Casa
- Casa → D → A → C → E → B → Casa



Le due soluzioni costano 9 monete. Non esiste soluzione migliore, poiché dalla casa di Livia si può prendere solo una linea al costo di due monete e poi bisogna prenderne una da tre monete. Gli altri quattro villaggi sono raggiungibili da linee che costano solo una moneta e dunque il totale minimo è di 9 monete.
Tutte le altre soluzioni costano di più:

- Costo 10 monete: Casa → A → D → C → E → B → Casa
- Costo 10 monete: Casa → A → E → B → C → D → Casa
- Costo 10 monete: Casa → B → C → E → A → D → Casa
- Costo 10 monete: Casa → B → E → A → C → D → Casa
- Costo 10 monete: Casa → B → E → C → D → A → Casa
- Costo 10 monete: Casa → C → B → E → A → D → Casa
- Costo 10 monete: Casa → D → A → E → B → C → Casa
- Costo 10 monete: Casa → D → A → E → C → B → Casa
- Costo 10 monete: Casa → D → C → A → E → B → Casa
- Costo 10 monete: Casa → D → C → B → E → A → Casa
- Costo 11 monete: Casa → B → E → A → D → C → Casa
- Costo 11 monete: Casa → C → D → A → E → B → Casa

Un modo per trovare il percorso più economico consiste scegliere un viaggio che costi poco e poi cercare di modificarlo per ottenere una soluzione ottimale.

Questa è l'informatica!

Cercare soluzioni valide o addirittura ottimali è uno dei compiti fondamentali dell'informatica. Il nostro problema di ottimizzazione può essere descritto attraverso un grafo in cui gli amici sono i nodi, mentre le strade sono gli archi. L'obiettivo è quello di visitare tutti i nodi esattamente una volta in modo che la somma dei pesi degli archi (il costo in monete) sia minima. Il nostro compito è simile al famoso Travelling Salesman Problem (TSP), ovvero il "problema del commesso viaggiatore". Questi tipi di problemi sono solitamente molto difficili da risolvere con un computer. Per evitare di dover provare ogni singola soluzione, è possibile utilizzare una buona euristica (ad esempio, un'euristica è quella di intraprendere dapprima il percorso più breve) ed eliminare tutte le soluzioni che peggiorano il risultato fin lì ottenuto.
Nel nostro caso, permettiamo a ciascun nodo di essere visitato solo una volta. Se fossimo autorizzati a visitare un nodo più di una volta, il problema diventerebbe molto più difficile perché dovremmo considerare molte più alternative.

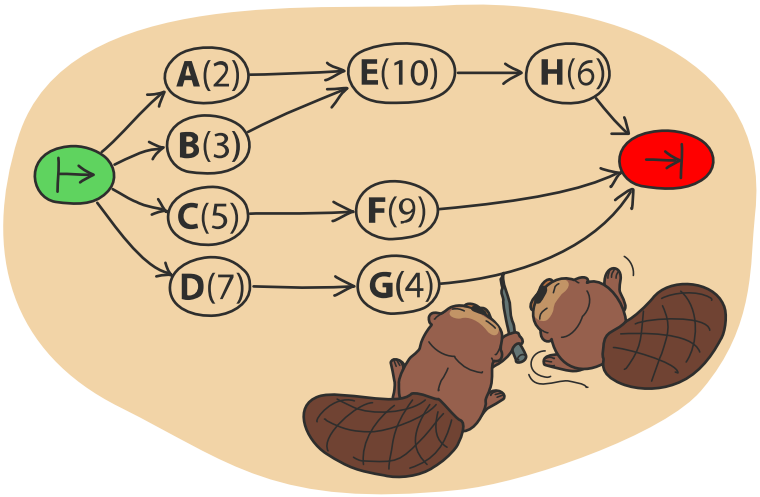
Parole chiave e siti web

ottimizzazione, problema del commesso viaggiatore

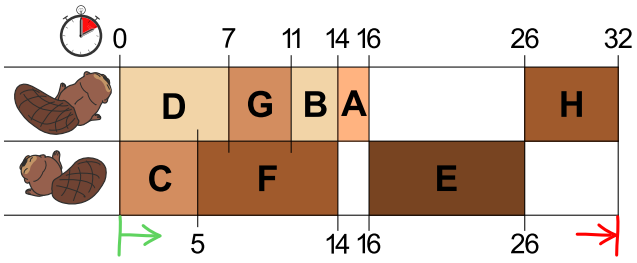
- `https://it.wikipedia.org/wiki/Problema_del_commesso_viaggiatore`
- `https://it.wikipedia.org/wiki/Problema_di_ottimizzazione`

3. Due castori al lavoro

Per costruire una diga, due castori devono terminare otto compiti: abbattere gli alberi, rimuovere i rami dai tronchi, portare i tronchi nell'acqua e così via. Ogni compito è etichettato con una lettera dell'alfabeto e un numero tra parentesi che indica le ore di lavoro richieste.
Alcuni compiti possono essere iniziati solo quando alcuni altri sono stati terminati. Tale dipendenza è rappresentata dalle frecce. I castori possono lavorare contemporaneamente su compiti diversi, ma solo uno di loro può lavorare su un determinato compito alla volta.



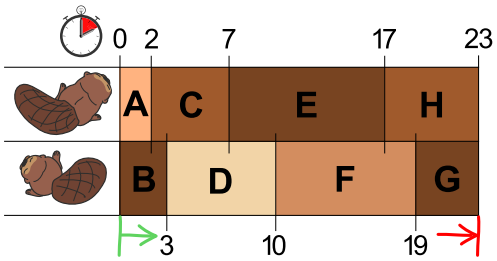
La figura qui sotto mostra un possibile piano di lavoro per i due castori... Si può però di sicuro terminare la diga più velocemente!



Quale è il minor tempo possibile per costruire la diga?

Soluzione

Ci vogliono almeno 23 ore.
La figura nel quesito mostra un possibile piano di lavoro. In essa il primo castoro ha una lunga pausa di 10 ore e il secondo castoro due pause per un totale di 8 ore. Se entrambi lavorassero tutto il tempo, la diga potrebbe essere costruita più velocemente.



Se si smistano i due compiti più grandi, E(10) e F(9) in modo opportuno cosicché non siano eseguiti dallo stesso castoro, è facile stabilire un piano di lavoro che si concluda in 23 ore. Non è possibile lavorare più veloce, poiché i due castori lavorano sempre senza sosta.

Questa è l'informatica!

Per trovare un piano di lavoro più breve, possiamo attenerci alla seguente regola: "scegliere sempre quello con il maggior tempo di lavoro tra i compiti ancora disponibili". In informatica si definisce una tale strategia "greedy" ("avida") e consiste nel terminare dapprima i compiti che comportano un maggior progresso verso la soluzione complessiva del problema.
In molti casi, la strategia "greedy" è buona, ma a volte -come nel nostro quesito- non funziona così bene. Nel nostro caso, abbiamo progettato questa domanda appositamente per fare in modo che essa non funzioni: è infatti importante considerare anche dei casi "limite". Ad esempio, nell'informatica teorica si considerano sempre i casi peggiori ("worst case") per risolvere determinati problemi, al fine di stimare il tempo massimo necessario per trovare la soluzione o compiere un lavoro.
In realtà, c'è solo un modo sicuro per trovare la soluzione migliore: provare tutti i piani di lavoro concepibili che soddisfano le regole date. Per i progetti enormi, tuttavia, il numero di possibilità può essere così grande che l'identificazione della soluzione migliore richiede troppo tempo. In questi casi, una strategia del tipo "greedy" può essere considerata, perché con essa si può velocemente trovare soluzioni sufficientemente buone (ma non necessariamente le migliori). Partendo dall'assunto che i due compiti più lunghi E(10) e F(9) non devono essere eseguiti dallo stesso castoro, è relativamente semplice identificare la migliore soluzione di 23 ore.

Parole chiave e siti web

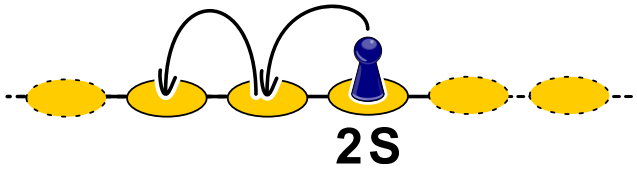
scheduling ("pianificazione"), algoritmo greedy ("avido")

- `https://it.wikipedia.org/wiki/Scheduler`
- `https://it.wikipedia.org/wiki/Ordinamento_topologico`
- `https://it.wikipedia.org/wiki/Algoritmo_greedy`

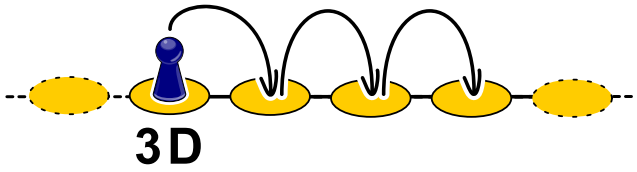
4. Salti

In questo gioco di salti, si devono osservare determinate regole. Esse sono definite in questo modo:

- nS: saltare n volte (posizioni) a sinistra, quindi 2S significa saltare due volte a sinistra,

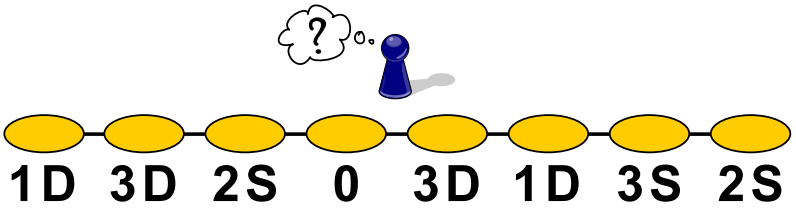


- nD: saltare n volte (posizioni) a destra, quindi 3D significa saltare tre volte a destra,



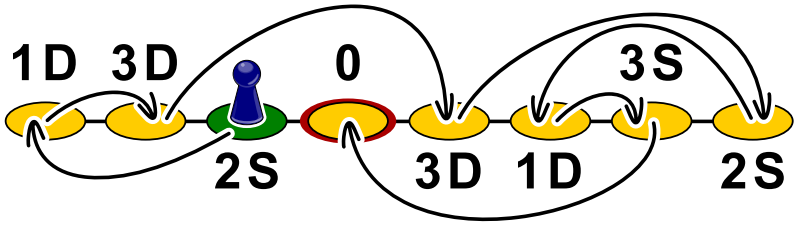
- 0: terminare.

Da quale posizione dobbiamo iniziare per poter saltare su tutte le aree mostrate, prima di terminare?



Soluzione

Partendo dalla terza postazione da sinistra ("2S"), seguendo le regole salteremo su tutte le aree prima di terminare.
Un metodo semplice per trovare la soluzione consiste nel partire postazione "0" e cercare a ritroso da quale area possiamo giungervi. Nel nostro caso, la seconda area da destra ("3S"). Essa può a sua volta può essere raggiunta dalla terza area da destra ("1D"), quindi da quella tutta a destra ("2S"), dalla quarta da destra ("3D"), dalla seconda da sinistra ("3D"), da quella più a sinistra ("1D") e infine rimane solo la terza da sinistra ("2S"), che dunque è la posizione da cui dobbiamo partire.



Se per ogni posizione indichiamo con delle frecce l'area di arrivo, basterà seguire a ritroso dalla posizione "0" il percorso per individuare facilmente l'area di partenza.

Questa è l'informatica!

In informatica, una struttura per la memorizzazione di dati molto diffusa è la *linked list* ("lista collegata"), la quale funziona in modo simile alle postazioni di salto del nostro quesito: nella memoria, i dati sono memorizzati assieme al riferimento all'indirizzo di memoria del dato successivo. In questo modo è possibile spezzettare una determinata informazione in varie parti collegate tra loro, senza dover necessariamente cercare in memoria un'area contigua sufficientemente grande per poter contenere tutto in un solo blocco. Molti sistemi applicano questo principio senza che l'utente se ne accorga, l'importante è solo che ci sia spazio di archiviazione a sufficienza.
Ma cosa succede quando i dati non sono più necessari? Un tempo era necessario che i programmatori liberassero la memoria "manualmente" (procedura all'origine di diversi errori e bug nei programmi), ma i moderni linguaggi di programmazione utilizzano un metodo automatico detto *Garbage Collection* ("raccolta di rifiuti"), che controlla regolarmente se i singoli dati in memoria sono ancora referenziati (ossia raggiungibili, partendo dall'inizio di tutte le liste memorizzate) oppure no. La memoria non più raggiungibile viene liberata e messa a disposizione per eventuali nuovi dati.

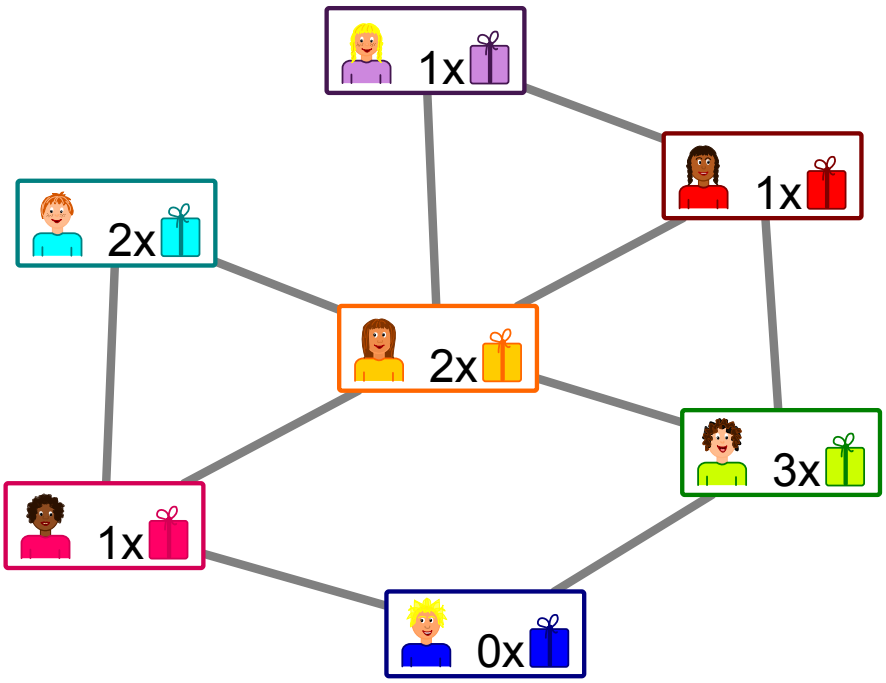
Parole chiave e siti web

lista, gestione della memoria, GOTO

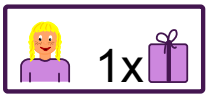
- `https://it.wikipedia.org/wiki/Garbage_collection`
- `https://en.wikipedia.org/wiki/St-connectivity`

5. Regali

L'immagine mostra le relazioni tra i ragazzi di un palazzo. Una linea tra due ragazzi rappresenta un legame d'amicizia.



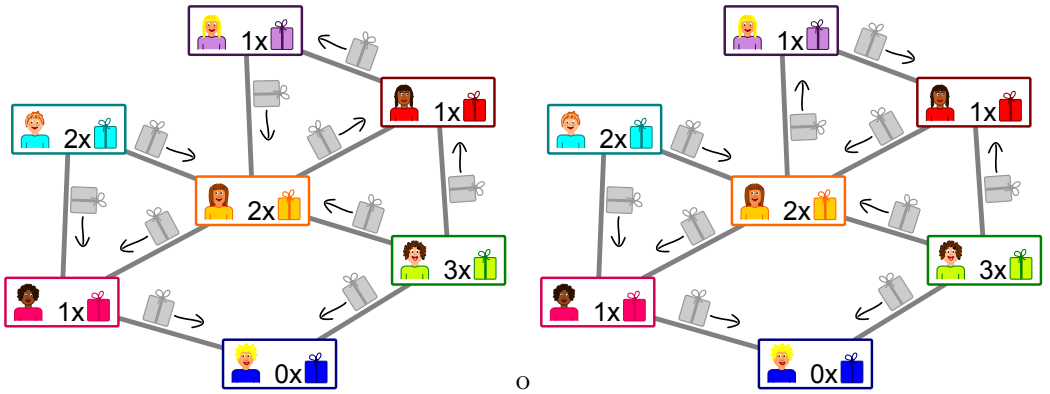
Gli inquilini pianificano una festa con dei regali per i ragazzi: per ogni coppia di amici, uno dei due ragazzi dovrà fare un regalo all'altro.

Nell'immagine vediamo anche quanti regali può preparare ogni ragazzo: 

significa che il ragazzo può preparare un solo regalo.
Indica per ogni coppia di amici chi deve fare un regalo all'altro.

Soluzione

Ci due soluzioni valide per indicare chi nella coppia di amici debba fare il regalo, in modo da non superare la disponibilità massima di ogni ragazzo.



È meglio iniziare con il ragazzo in basso: egli non prepara regali e quindi deve necessariamente ricevere un regalo da ciascuno dei suoi amici a destra e a sinistra. La sua amica a sinistra esaurisce quindi la disponibilità a fare regali e deve dunque riceverli dagli altri amici. A questo punto la scelta per tutti gli altri è abbastanza chiara. L'unica opzione rimasta è se far dare ai tre bambini in alto a destra i regali in "senso orario" o "antiorario".

Questa è l'informatica!

Le relazioni di amicizia tra i ragazzi formano una rete di nodi (i bambini) e archi (le amicizie), esattamente come avviene nei "social network" costituiti da milioni di utenti. Tuttavia i legami all'interno di un network possono avere caratteristiche differenti: in alcuni, come nel nostro caso, ci sono "amicizie" reciproche in cui le connessioni non hanno direzione, mentre in altri ci sono "follower" con connessioni unidirezionali. In tal caso, "seguire" (follow) un personaggio/utente famoso non significa necessariamente essere seguiti da lui.
Nel nostro quesito dobbiamo indicare per ogni amicizia la "direzione" del regalo. Questo è un aspetto importante, considerando che la disponibilità di regali è limitata e dunque influisce sulla scelta. L'obiettivo è quello di fare un regalo in ogni amicizia senza superare la capacità di ogni ragazzo. In informatica esistono problemi molto simili: in una rete (come le connessioni Internet) la capacità di traffico è limitata. Essa dovrebbe sempre essere utilizzata al massimo, senza però superarla.
Il problema del flusso massimo nelle reti può essere risolto in modo efficiente. Poiché esso è alla base anche del nostro quesito, i metodi di soluzione possono anche essere applicati da noi. Anche questo è tipico dell'informatica: un problema viene spesso trasformato (ridotto) in un altro problema con la stessa struttura, per cui sono già state trovate delle soluzioni efficienti.

Parole chiave e siti web

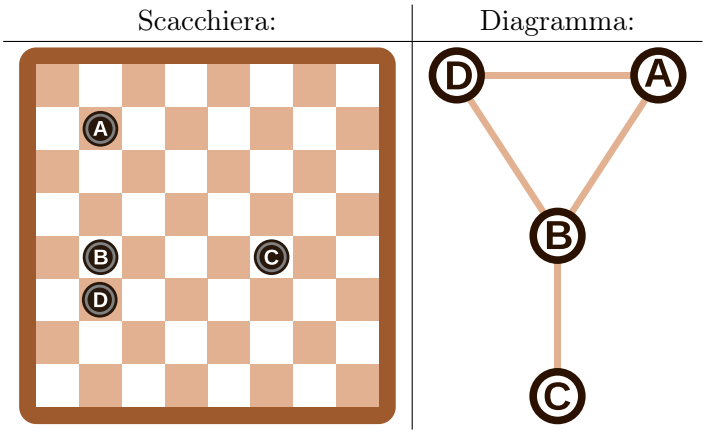
rete di flusso, riduzione di problemi

- `https://it.wikipedia.org/wiki/Problema_del_flusso_massimo`
- `https://it.wikipedia.org/wiki/Rete_di_flusso`

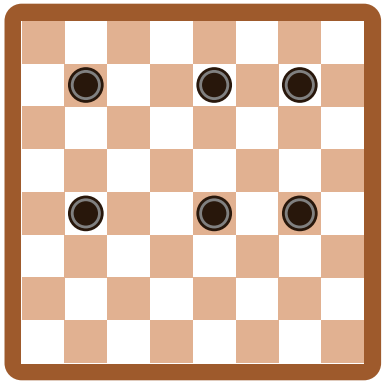
6. Righe e colonne

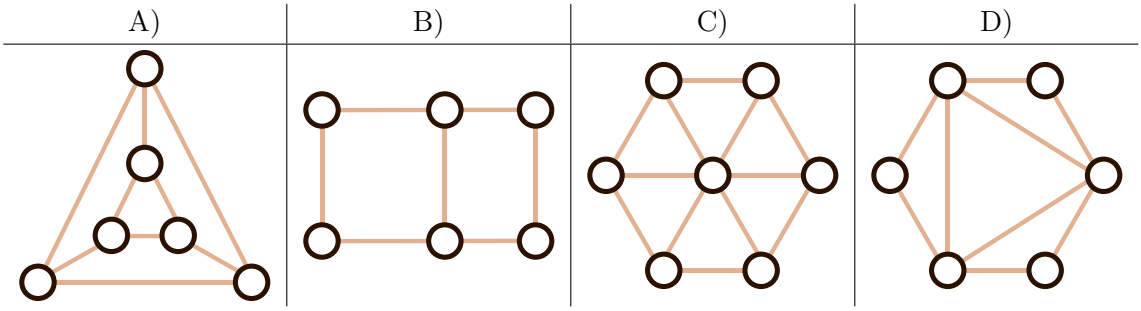
Il diagramma mostrato sulla destra della scacchiera è stato allestito partendo dalla posizione delle pedine in modo che...

- ...ogni pedina è rappresentata da un cerchio e...
- ...2 pedine nel diagramma sono collegate da una linea se sono sulla stessa riga o sulla stessa colonna della scacchiera.



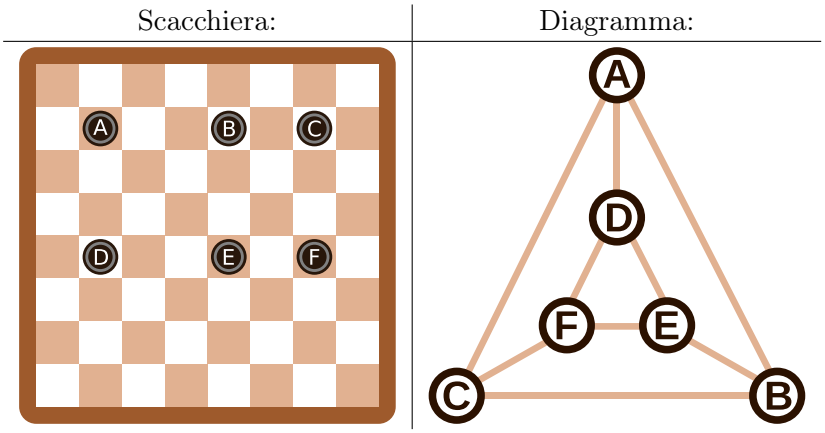
Nel nostro esempio, le pedine sulla scacchiera e i cerchi nel diagramma sono etichettati con una lettera dell'alfabeto, in modo da rendere chiara la spiegazione.
Quale diagramma è associato alla scacchiera seguente, sulla quale sono posizionate 6 pedine?



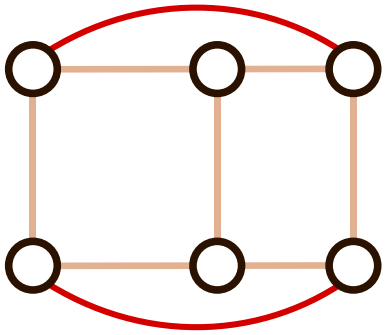


Soluzione

La risposta corretta è A). Nella figura seguente le pedine sono state etichettate con lettere dell'alfabeto in modo da rendere evidente la soluzione:



Le risposte B), C) e D) possono essere escluse in seguito a questa osservazione: ogni pedina ha altre 2 pedine sulla stessa riga e un'ulteriore pedina sulla stessa colonna. Ciò significa che nel diagramma ogni cerchio deve essere connesso esattamente ad altri $2 + 1 = 3$ cerchi. Nei diagrammi B), C) e D) questo non avviene, inoltre nel diagramma C) le pedine sono addirittura 7.
Se pur simile nella disposizione delle pedine sulla scacchiera, il digramma B) non è corretto in quanto i 4 cerchi esterni sono connessi solo a 2 cerchi. Per rendere il diagramma corretto sarebbe necessario aggiungere altre 2 linee come mostrato qui di seguito:



Questa è l'informatica!

In informatica diagrammi simili (i *grafi*) sono spesso usati per rappresentare determinati problemi in modo essenziale. I cerchi sono detti più precisamente *nodi* e le linee *archi*.
Per la rappresentazione di determinate situazioni attraverso grafi è importante solo conoscere quali nodi sono collegati tra loro tramite archi. La loro disposizione o forma non ha alcuna importanza. Lo stesso grafo può quindi avere forme diverse, come abbiamo già visto sopra: sia il grafo in A) sia quello nell'ultima immagine nella spiegazione della risposta sono soluzioni corrette. Entrambi sono grafi equivalenti, in cui cambia solo la disposizione spaziale dei nodi.
Il grafi sono un'astrazione e rappresentano solo l'essenza di un problema. Grazie ad essi è possibile rispondere a diverse domande, come "Qual è il numero minimo di pedine da rimuovere in modo che non ci siano mai 2 tessere nella stessa riga o colonna?". Una parte importante del lavoro di un informatico è trovare una buona rappresentazione di un determinato problema, in modo da poter trovare la soluzione con efficienza.

Parole chiave e siti web

grafo

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

7. Ordinare i libri

Ad ogni persona di un gruppo di tre viene assegnato un tavolo, su cui ci sono 2 libri ciascuno. Il gruppo deve ordinare tutti e 6 i libri, potendo scambiare ad ogni turno solo quelli adiacenti (vicini) e lavorando assieme. A ogni turno un libro può essere spostato al massimo una volta.
Esistono due diversi tipi di turno e vengono sempre eseguiti in modo alternato (prima uno e poi l'altro):

A. Le tre persone possono (ma non devono) scambiare i due libri sul proprio tavolo (esempio A).

B. Le due persone più a sinistra possono (ma non devono) scambiare il libro a destra sul proprio tavolo con quello a sinistra del tavolo vicino sulla loro destra (esempio B).

Questa è la situazione iniziale:

Il primo turno è di tipo A).
Quanti turni sono necessari in totale per ordinare i libri, ossia disporli nella successione 1, 2, 3, 4, 5, 6*?*

A) tre turni

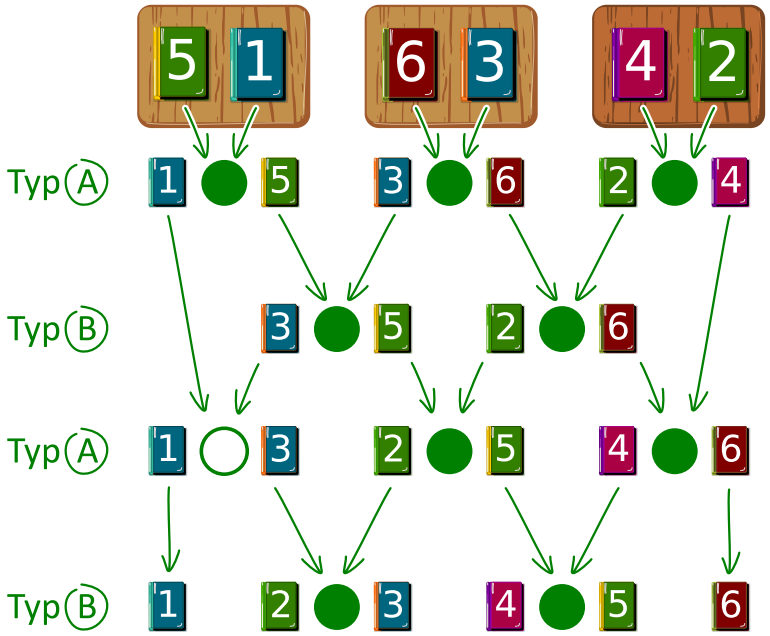
B) quattro turni

C) cinque turni

D) sei turni

Soluzione

La risposta corretta è B), quattro turni.
La figura mostra come sia possibile ordinare i libri scambiandoli opportunamente. Nell'esempio, le persone utilizzano una strategia detta *greedy* ("avido"). Ciò significa che ad ogni turno cercano di ottenere un ordine parziale più vicino possibile alla soluzione finale: in pratica confrontano i libri vicini (a dipendenza del turno, sullo stesso tavolo o nel tavolo sulla destra) e se quello di sinistra ha un numero maggiore, allora si effettua lo scambio.
Nel primo turno (tipo A) tutti i libri sullo stesso tavolo vengono scambiati tra loro. Nel secondo turno (tipo B) tutti i libri adiacenti vengono scambiati. Al terzo turno (tipo A) si scambiano solo i libri nei tavoli più a destra e infine, nel quarto turno (tipo B), tutti i libri adiacenti sono scambiati. In questo modo tutti i libri sono ordinati.
Non è possibile procedere più velocemente, in quanto il libro 5, ad esempio, deve "risalire" ben 4 posizioni: potendo risalirne solo una per ogni turno, dobbiamo necessariamente compiere almeno quattro turni.



Questa è l'informatica!

Nel nostro quesito abbiamo potuto osservare un metodo di ordinamento *parallelo*, in cui diversi "attori" (le persone) lavorano allo stesso tempo per risolvere un problema. L'ordinamento parallelo può essere rappresentato da una rete di *ordinamento* ("sorting network") come mostrato nella figura precedente. Un sorting network è composto da archi diretti (rappresentati da frecce) e nodi rappresentati da cerchi.
In ogni turno, i due libri contrassegnati da un cerchio vengono confrontati e scambiati se necessario. Il confronto dei vari cerchi ad ogni turno può avvenire parallelamente (allo stesso tempo). Seguendo un libro lungo le frecce, dall'alto verso il basso, si può notare come gradualmente esso prenda la sua giusta posizione nell'ordine desiderato.

Parole chiave e siti web

ordinamento in parallelo (parallel sorting), rete di ordinamento (sorting network)

- `https://en.wikipedia.org/wiki/Sorting_network`

8. Soundex

Donald desidera codificare le parole in base al loro suono nel modo seguente:

- Conserva la prima lettera.
- Elimina tutte le lettere A, E, I, O, U, H, W, Y.
- Sostituisci le lettere rimanenti in questo modo:
  - B, F, P, V → 1
  - C, G, J, K, Q, S, X, Z → 2
  - D, T → 3
  - L → 4
  - M, N → 5
  - R → 6
- Se la stessa cifra dovesse apparire due o più volte, come conseguenza della codifica della stessa lettera vicina (es. doppia) nella sequenza, allora conservane solo una. Questo vale anche se la prima lettera non è stata codificata da una cifra, ma ha conservato il carattere originale.
- Alla fine vengono annotati solo i primi 4 caratteri (inclusa la lettera iniziale). Se necessario si completa la parola codificata aggiungendo degli zeri fino a raggiungere i 4 caratteri.

Le parole seguenti sono state codificate in questo modo:

Euler → E460
Gauss → G200
Heilbronn → H416
Kant → K530
Lloyd → L300
Lissajous → L222

Qual è il codice della parola "Hilbert"?

A) H410

B) B540

C) H041

D) H416

Soluzione

La prima lettera è H, quindi anche il codice deve iniziare per H.
In seguito si eliminano tutte le lettere A, E, I, O, U, H, W, Y e dunque bisogna solo trasformare *Hlbrt*.
La codifica di tali lettere porta a *H4163*. Dato che non ci sono cifre uguali, nulla deve essere eliminato.
Infine eliminiamo i caratteri in eccesso e otteniamo H416, ovvero la risposta al nostro quesito.

Questa è l'informatica!

Il procedimento Soundex, più precisamente il Soundex americano, è stato sviluppato e patentato oltre un secolo fa da Rover C. Russel e Margaret King Odell. Esso è stato usato per trovare parole dal suono simile nella lingua inglese, in particolare nomi simili di persone. Esso si basa sul fatto che i gruppi di lettere assegnati allo stesso codice suonano in modo simile: B, F, P e V sono suoni labiali, C, G, J, K, Q, S, X e Z sono suoni "palatali" e sibilanti, D e T sono dentali, L è un suono linguale lungo, M e N sono nasali e infine R è un linguale breve.
Dato che questo procedimento è molto semplice e dà risultati relativamente buoni, non solo in lingua inglese, è spesso usato per la ricerca fonetica, quindi per cercare parole dal suono simile. Esso è anche integrato come standard in molti database.
Gli esempi citati sono di Donald Knuth, uno dei grandi scienziati informatici del 20º secolo, che ancora lavora sul suo libro "The Art of Computer Programming". Il volume 3 "Ordinamento e ricerca" contiene il metodo descritto.

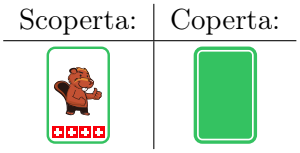
Parole chiave e siti web

Ricerca fonetica, soundex

- `https://www.functions-online.com/soundex.html`
- `https://en.wikipedia.org/wiki/Soundex`
- `https://www-cs-faculty.stanford.edu/~knuth/taocp.html`
- `http://www.highprogrammer.com/alan/numbers/soundex.html`

9. Girare le carte

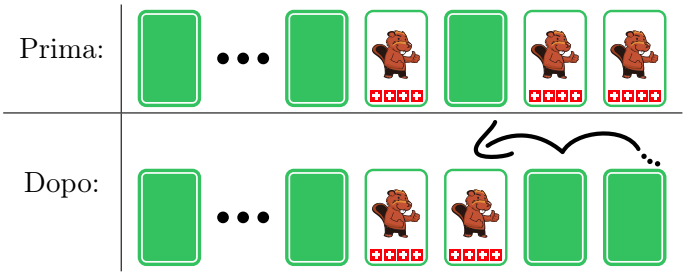
Ti hanno regalato un mazzo di carte uguali. Le carte sono fatte così:



Con queste carte si possono inventare giochi particolari, come questo:
Disponi le carte lungo una fila. In ogni sessione di gioco, partendo da destra verso sinistra, esegui le seguenti operazioni:

- Se la carta è scoperta, la giri.
- Se la carta è coperta, la giri. Termini poi immediatamente la sessione di gioco corrente, lasciando le carte rimanti invariate.

Ad esempio, una sessione completa consiste in:



Le due carte scoperte a destra vengono entrambe girate. La carta seguente è invece coperta e dunque viene scoperta, terminando così la sessione.
Ora cominci un gioco con 16 carte coperte:

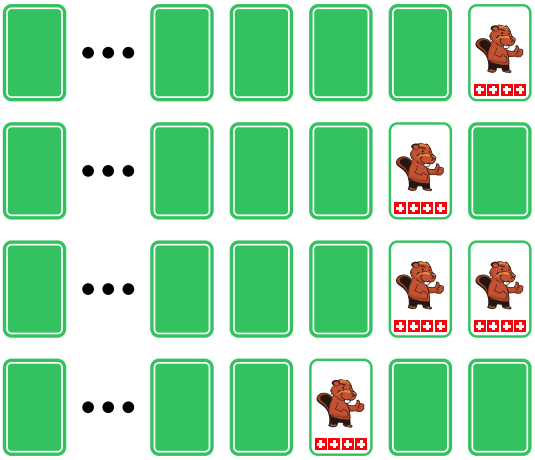
Quante carte sono scoperte dopo 16 sessioni di gioco?

Soluzione

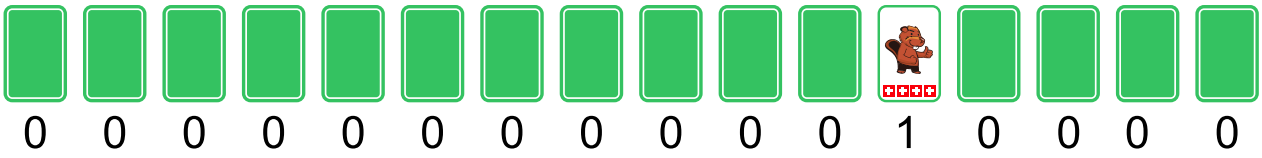
Esattamente una carta è scoperta.
Le carte (scoperte o coperte) possono essere equiparate a dei numeri binari. I numeri binari, infatti, sono composti solo dalle cifre 1 (carta scoperta) e 0 (carta coperta).
Analogamente al sistema decimale, ogni cifra di un numero binario indica se la potenza di 2, a cui è associata la posizione, deve essere inclusa nel valore del numero o no. Ad esempio, se la terza cifra (da destra) di un numero binario è occupata da un 1, la terza potenza di due deve essere aggiunta al valore del numero. Quindi il codice binario 100 è $1^2 + 0 \times 2^1 + 0 \times 2^0 = 4$.
I numeri binari vengono incrementati di 1 in questo modo. Si inizia con il numero all'estrema destra:

- Se la cifra corrente è 0, la si imposta a 1. In questo modo il valore è incrementato di 1.
- Se la cifra corrente è 1, la si imposta a 0 procede verso la cifra seguente a sinistra per il riporto.

Queste regole corrispondono esattamente al nostro gioco di carte. Od ogni sessione il numero binario rappresentato dalle carte viene incrementato di 1. L'inizio del gioco, con tutte le carte coperte, corrisponde ad un numero binario composto solo da cifre 0 e dunque ha un valore totale pari a 0.
L'immagine seguente mostra i risultati delle prime quattro sessioni, che corrispondono ai numeri da 1 a 4. Possiamo vedere come per i numeri 1, 2 e 4 (cioè, le potenze di due 2^0, 2^1 e 2^2) esattamente solo una carta è scoperta. Ogni numero binario con valore pari a una potenza di due possiede infatti sono una cifra pari a 1, mentre tutte le altre sono 0.



Giocare 16 sessioni corrisponde dunque a rappresentare il numero 16 in codice binario, dunque $16 = 2^4 = 10000$. Le cifre più a sinistra sono tutte pari a 0 e possono essere ignorate, oppure possiamo scrivere 0000000000010000.



Questa è l'informatica!

La più piccola unità di archiviazione del computer distingue solo due valori: 0 o 1 (designati anche come FALSE o TRUE). Tutti i dati memorizzati ed elaborati da computer sono in realtà una serie di cifre binarie. I numeri binari e le relative operazioni binarie stanno dunque alla base dell'informatica.

L'esecuzione di tali operazioni è pianificata nel modo più efficiente possibile. Vi sono operazioni che combinano due diversi numeri binari, come le operazioni aritmetiche di addizione (OR) o moltiplicazione (AND). Vi sono anche operazioni che agiscono solo su un singolo numero binario, come lo spostamento di tutte le cifre di una posizione a sinistra (LEFT SHIFT) o semplicemente incrementano il valore di 1 (come nel nostro esercizio).
I processori migliori sono caratterizzati dalla loro capacità di eseguire tali operazioni in modo rapido ed efficiente, milioni di volte al secondo. È però compito del programmatore ottimizzarne l'utilizzo, in modo da rendere "fluida" l'esecuzione delle applicazioni.

Parole chiave e siti web

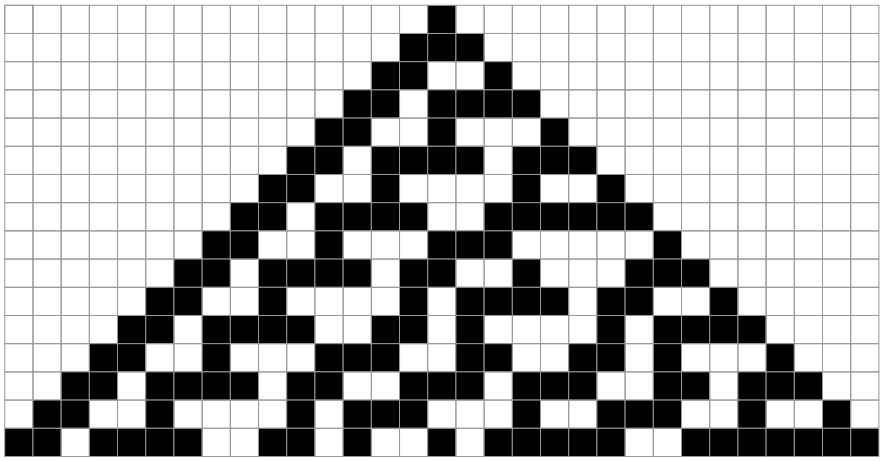
numero binario, codice binario

- `https://it.wikipedia.org/wiki/Sistema_numerico_binario`
- `https://it.wikipedia.org/wiki/Contatores#Contatore_elettronico`

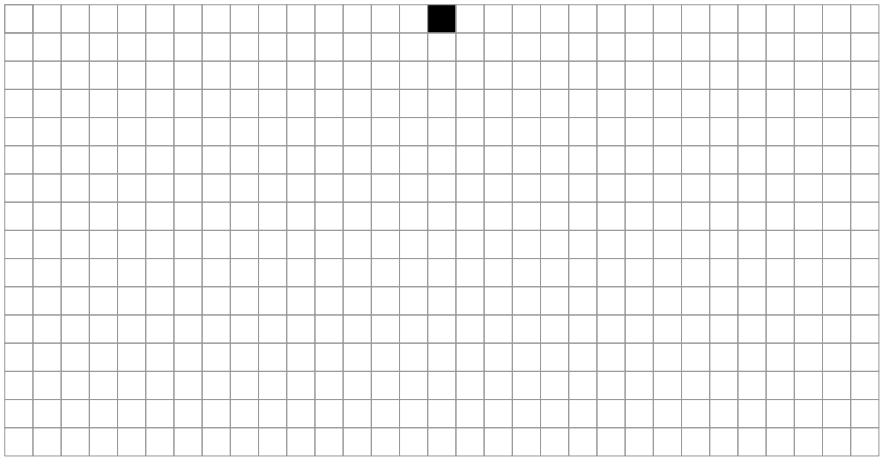
10. Mosaico

Tina deve creare un mosaico su un pavimento largo 31 tessere e alto 16 tessere. Tina desidera che le tessere siano posate seguendo semplici regole.
Tali regole sono definite in modo che 3 tessere contigue su una riga, determinino la tessera centrale che sta direttamente sotto. Con una serie di 8 regole che considerano ogni combinazione possibile per le tre tessere (sui bordi la tessere "mancanti" vengono considerate bianche):

è possibile definire l'intero mosaico. Tina ha iniziato dal centro in alto con una tessera nera, mentre tutte le altre tessere sulla prima riga erano bianche. Dopo aver applicato le regole riga dopo riga, ha ottenuto il seguente mosaico:



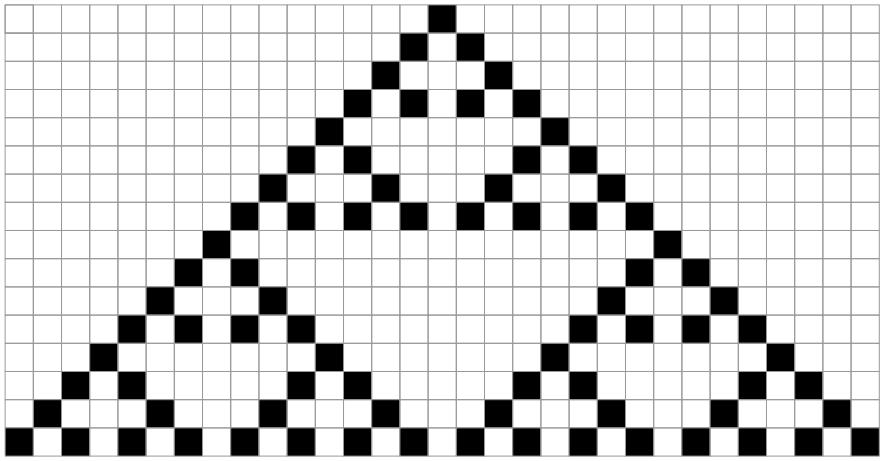
Crea la tua serie di semplici regole, in modo che nell'ultima fila le tessere nere e quelle bianche siano alternate.

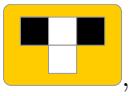


Soluzione

Esistono molte soluzioni per questo esercizio. Una soluzione possibile è la seguente:

Questa serie di regole determina il seguente mosaico:



Osservando il mosaico attentamente, notiamo 9 "triangoli", la cui base (la terza linea) è sempre costituita da tessere bianche e nere alternate. Ciò impone quasi sempre l'utilizzo della regola  , ad eccezione dei bordi, da dove vengono "costruiti" nuovi triangoli.
Se indichiamo una tessera bianca come B e una nera come N, possiamo compire come sia possibile definire ben 256 serie di regole diverse. Ogni singola regola può avere 2 risultati (B o N) ed essendocene 8 per ogni serie in totale abbiamo $2^8 = 256$ serie. L'esempio iniziale di regole (quello mostrato nel testo della domanda) avrebbe dunque il codice: BBBNNNB.
La seguente lista di 16 serie di regole creerebbe un mosaico con tessere bianche e nere nell'ultima linea:

SWSSWWSS
SSSSSWSW
WSSSSWSW
SWSSSWSW
WWSSSWSW
SSWSSWSW
WSWSSWSW
SWWSSWSW
WWWSSWSW
SSSSWWSW
WSSSWWSW
SWSSWWSW
WWSSWWSW
SSWSWWSW
WSWSWWSW
SWWSWWSW
WWWSWWSW (soluzione discussa)

Questa è l'informatica!

Le regole del nostro quesito sono analoghe a quelle del *Game of life* ("gioco della vita") *di Conway*, un matematico inglese che lo pubblicò nel 1970. Esso si basa su un "automa cellulare" bidimensionale, una specie di mosaico nel quale il colore (o meglio: "stato") di ogni tessera (o meglio: "cellula") è determinato da quelle adiacenti (il suo "intorno"). Il nuovo stato di ogni cellula viene determinato in base allo stato precedente delle cellule del suo intorno. Seguendo delle semplici regole possiamo quindi simulare la vita e la morte di individui in una certa area. Nel nostro caso, le regole sono ancora più semplici, poiché lo stato di ogni tessera dipende solo da quello di quelle immediatamente sopra.

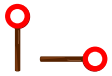
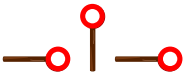
Parole chiave e siti web

Modelli (pattern), automi cellulari

- `http://web.stanford.edu/~cdebs/GameOfLife/`
- `https://en.wikipedia.org/wiki/Rule_90`
- `https://it.wikipedia.org/wiki/Automa_cellulare`
- `https://it.wikipedia.org/wiki/Gioco_della_vita`

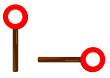
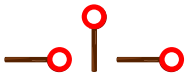
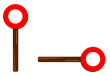
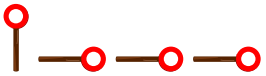
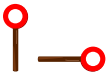
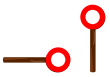
11. Dove è l'aliante?

Jana e Robin giocano con il loro aliante. Uno di loro lo lancia da una piccola collina e l'altro lo riprende dopo ogni atterraggio sul prato. Sfortunatamente, l'erba non è stata falciata da tempo e quindi è possibile vedere dove è atterrato l'aliante solo dalla collina e non dal prato. Jana e Robin hanno quindi concordato dei segnali per indicarne la posizione.

a sinistra	a destra	verso la collina	verso la valle
			

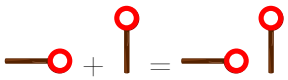
C'è però un problema con questi segnali. Quando ad esempio si danno le indicazioni seguenti

esse possono essere interpretate come "a sinistra – verso la collina – a sinistra", ma anche come "a sinistra – a destra – a sinistra – a sinistra".
Jana e Robin hanno quindi concordato dei nuovi segnali. Quali tra quelli mostrati non possono essere mal interpretati?

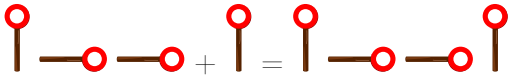
	a sinistra	a destra	verso la collina	verso la valle
A)				
B)				
C)				
D)				

Soluzione

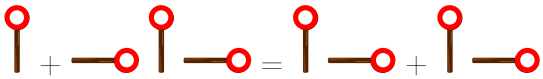
La risposta corretta è C). Si può notare come ogni indicazione inizi con  , seguita poi da 0, 1, 2 o 3  . Il segnale  non viene più utilizzato all'interno dell'indicazione. In questo modo sappiamo che ogni indicazione inizia con  e per interpretarla dobbiamo contare da quanti  è composta.
La risposta B) non è corretta, in quanto l'indicazione "a sinistra" seguita da "a destra" corrisponde a "verso la collina":



La risposta D) non è corretta, in quanto l'indicazione "a sinistra" seguita da "verso la valle" corrisponde a "verso la collina":

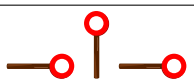
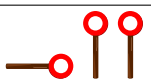


Capire che la risposta A) non è corretta, è un po 'più difficile. L'indicazione "verso la valle" seguita da "a sinistra" corrisponde a due volte "a destra":



Questa è l'informatica!

I computer inviano dati, via cavo o via etere, attraverso veloci successioni di segnali binari (ossia con due soli possibili valori: 1 o 0). Questo è esattamente ciò che fanno anche Jana e Robin, i quali utilizzano 2 diversi segnali per ogni indicazione.
Un messaggio, un'indicazione o un comando convertito in segnali di questo tipo viene detto *codice (binario)*. Nel nostro esempio utilizziamo un codice di *lunghezza variabile* poiché il numero di segnali utilizzati per un'indicazione non è sempre lo stesso.
È importante che il destinatario di un messaggio codificato sia in grado di tradurre i segnali nel messaggio originale senza possibilità di errore. Quando si progettano tali codici si deve quindi prestare molta attenzione. I codici che diciamo "buoni", sono detti *codici univocamente decodificabili*.
Un particolare tipo di codice univocamente decodificabile è il *codice prefisso*. In questo tipo di codice ogni indicazione (detta "parola") non può iniziare con la sequenza completa di segnali di un'altra indicazione, ad esempio:

a sinistra	a destra	verso la collina	verso la valle
			

I codici prefissi hanno queste ottime caratteristiche: possono essere piuttosto brevi e facili da decifrare. Essi non vengono utilizzati solo per la comunicazione, ma anche in diversi algoritmi di compressione.

Parole chiave e siti web

codice prefisso, segnali

- `https://it.wikipedia.org/wiki/Codice_prefisso`
- `https://it.wikipedia.org/wiki/Telegrafo#Il_telegrafo_ottico_Chappe`

12. Pianificazione delle prove

Cinque ballerini fanno delle prove per uno spettacolo: Alex, Bojan, Coco, Deniz ed Emil.
Durante lo spettacolo, i ballerini formeranno queste coppie in successione:

- Alex – Bojan
- Coco – Alex
- Emil – Deniz
- Alex – Emil
- Coco – Deniz
- Bojan – Coco
- Deniz – Alex
- Coco – Emil

Le coppie vorrebbero provare una dopo l'altra senza pause. La pianificazione dovrebbe quindi fare in modo che un membro di una coppia appartenga anche alla coppia successiva e possa continuare direttamente. Per non stancarsi troppo, nessuno dovrebbe però provare tre volte di seguito. Ad esempio, dopo la coppia Alex – Bojan, potrebbe provare una coppia che includa Alex o Bojan, ovvero Coco – Alex, Alex – Emil, Bojan – Coco oppure Deniz – Alex.
Uno dei ballerini capisce che potrà sicuramente venire alle prove più tardi: non sarà sicuramente tra i primi in programma.
Chi è il ballerino che potrà presentarsi più tardi?

A) Alex

B) Bojan

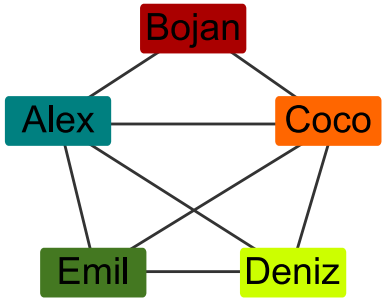
C) Coco

D) Deniz

E) Emil

Soluzione

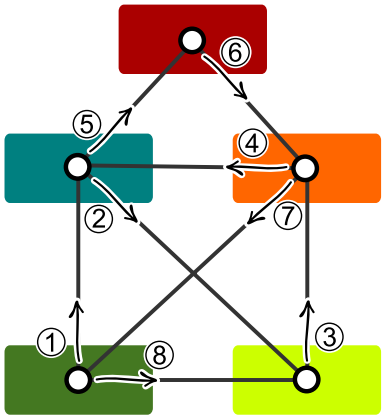
La risposta corretta è B) Bojan.
Nel diagramma seguente ogni rettangolo etichettato con un nome rappresenta un ballerino. Due ballerini sono connessi da una linea se formano una coppia. Un pianificazione valida delle prove è dunque un cammino continuo che attraversi tutte le linee del diagramma esattamente una volta. Non esiste una linea di ritorno diretta, poiché essa implicherebbe che un ballerino provi per tre volte di seguito.



È quindi necessario che ogni volta che si "arriva" ad un rettangolo nel mezzo del cammino, lo si possa anche lasciare attraverso una linea non ancora percorsa. Ogni nome, eccetto due, possiede un numero pari di linee (ovvero, i ballerini fanno parte dunque un numero pari di coppie). Il nostro cammino *continuo* deve necessariamente iniziare e finire da quei nomi che possiedono un numero dispari di linee.
Deniz ed Emil sono i ballerini che formano un numero dispari di coppie. Solo loro possono appartenere alla prima o all'ultima coppia. Essendo Bojan l'unico ballerino che non fa coppia con loro, egli non può in alcun caso iniziare le prove e dunque potrà presentarsi più tardi.

Questa è l'informatica!

L'immagine appena vista mostra come sia possibile rappresentare le diverse coppie attraverso un *grafo* formato da *nodi* (i ballerini) e *archi* (le linee che identificano una coppia). Il grafo è una struttura molto versatile che viene utilizzata in molte attività connesse all'informatica che necessitano di modelli semplici per poter essere elaborate. Ad esempio, possiamo utilizzare grafi per analizzare reti stradali o di comunicazione.
In alcune reti, potrebbe essere necessario disattivare tutte le connessioni su un cammino da un nodo iniziale a uno finale diverso. In questo casi, per motivi di efficienza, potrebbe essere necessario trovare un cammino tale che ogni connessione venga percorsa una volta sola. Tale cammino, che attraversa ogni arco esattamente una volta, è detto cammino di Eulero (in onore di Leonhard Euler, studioso della teoria dei grafi). Eulero ha dimostrato che in un grafo connesso esiste un cammino euleriano se esattamente due nodi possiedono un numero dispari di archi, mentre tutti gli altri ne hanno un numero pari. Questi due nodi, inoltre, devono essere l'inizio, rispettivamente, la fine del cammino di Eulero.



Probabilmente avete già visto un grafo simile a quello del nostro quesito prima d'ora. Esso corrisponde infatti al famoso indovinello che vi sfida a "disegnare una casetta (o una busta), senza staccare la matita dal foglio". Chiunque ci riesca disegna, di fatto, un cammino euleriano.

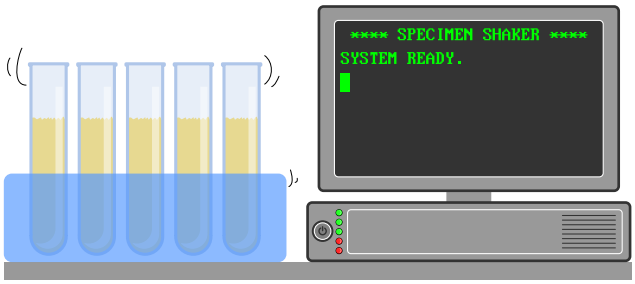
Parole chiave e siti web

grafo, cammino di Eulero (o euleriano)

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

13. Laboratorio medico

In un laboratorio medico, i campioni dei pazienti devono essere agitati regolarmente per poterli analizzare. Per questo, il laboratorio medico utilizza una macchina che esegue un programma. Il programma viene eseguito riga per riga.
La macchina è programmata così:




```
1 SALVA 0 IN N
2 INCREMENTA N DI 1
3 VAI ALLA LINEA 6
4 SE N È UGUALE A 60, ALLORA VAI ALLA LINEA 8
5 SALVA 0 IN N
6 INCREMENTA N DI 1
7 VAI ALLA LINEA 2
8 RIPETI N VOLTE AGITA
9 FINE
```

I comandi sono:

- `SALVA` *`valore`* `IN` *`nome`*: memorizza il *`valore`* nella variabile *`nome`*.
- `INCREMENTA` *`nome`* `DI 1`: legge il valore memorizzato nella variabile *`nome`*, aggiunge 1 e salva il risultato di nuovo in *`nome`*.
- `VAI ALLA LINEA` *`numero-linea`*: continua il programma dalla linea indicata con *`numero linea`*.
- `SE` *`nome`* `È UGUALE A` *`valore`*`, ALLORA` *`comando`*: confronta il valore salvato in *`nome`* con quello indicato con *`valore`*. Se entrambi sono uguali, esegue il comando *`comando`* altrimenti no.
- `RIPETI` *`nome`* `VOLTE` *`comando`*: esegue il comando *`comando`* tante volte quanto indicato dal valore contenuto in *`nome`*.
- `AGITA`: agita il campione una volta.
- `FINE`: termina il programma.

Quante volte un campione verrà agitato dalla macchina?

A) Il campione non sarà mai agitato.

B) Il campione sarà agitato una volta.

C) Il campione sarà agitato 60 volte.

D) La macchina non terminerà mai di agitare il campione.

Soluzione

Il programma salta continuamente dalla linea 3 alla linea 6 e dalla linea 7 alla linea 2. Solo all'inizio viene eseguita la linea 1 e poi le linee 2, 3, 6 e 7 senza fine. Il campione verrebbe agitato solo se fosse eseguita la linea 8, ma ciò non avviene. Questo significa che il programma non agiterà mai il campione. Siccome la riga 9 non viene mai eseguita, la macchina non terminerà mai il programma.

Questa è l'informatica!

Il programma utilizza il comando "`VAI ALLA LINEA`" (in inglese, "`GO TO`") come struttura di controllo per passare ad altre parti del programma. La struttura di controllo `GOTO` è intrinsecamente connessa al funzionamento del hardware ed è stata spesso utilizzata nei primi linguaggi di programmazione (fino agli anni '80) per reagire agli input dell'utente o ad altre condizioni. Tuttavia essa rende il codice meno comprensibile per un essere umano e per questo spesso è all'origine di errori di programmazione. I più moderni linguaggi di programmazione, sviluppati fin dagli anni '50, tendono ad eliminare questa struttura in favore dell'utilizzo di cicli (come `RIPETI`, utilizzato nel nostro esempio), esecuzione condizionale di blocchi di programma o sub-routine.

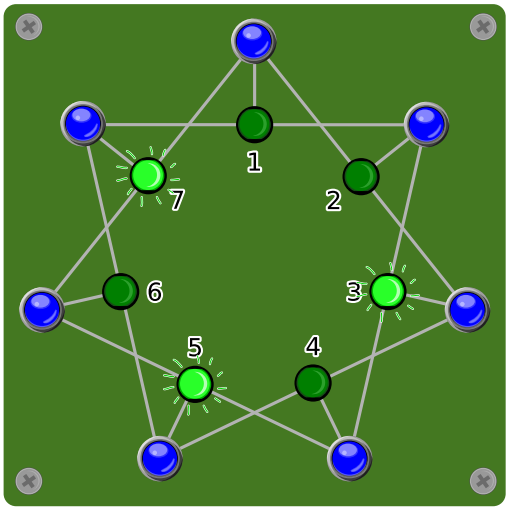
Parole chiave e siti web

struttura di controllo, analisi del programma, `GOTO`

- `https://homepages.cwi.nl/~storm/teaching/reader/Dijkstra68.pdf`
- `https://it.wikipedia.org/wiki/GOTO`

14. Accendi la luce!

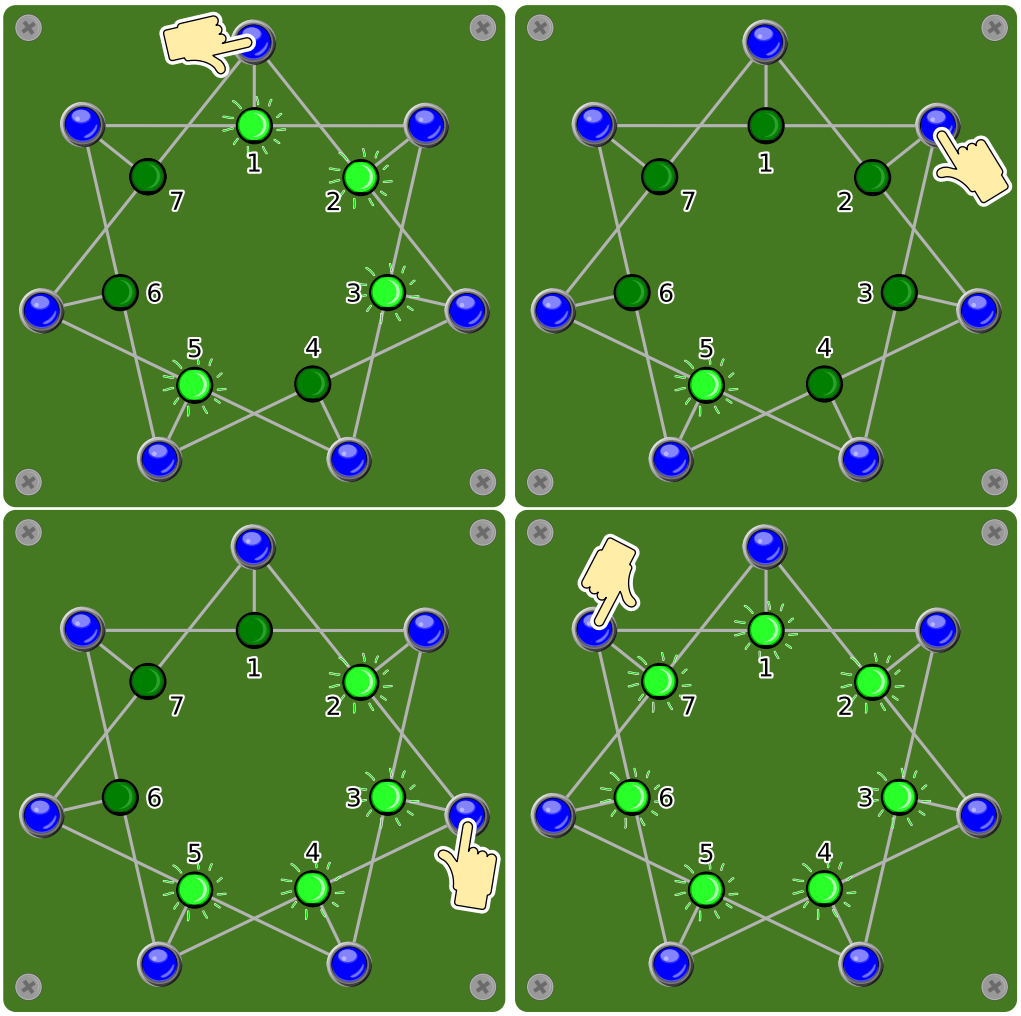
Sette interruttori sono collegati a sette lampade in modo tale che ogni interruttore controlli sempre tre lampade alla volta. Quando viene premuto un interruttore, le lampade sotto il suo controllo che erano spente vengono accese, mentre quelle accese vengono spente.



Quali interruttori devono essere premuti per far in modo che alla fine tutte le lampade siano accese?

Soluzione

Premendo l'interruttore vicino alla lampada 1 (o, risp., 2), entrambe le lampade 1 e 2 vengono accese, mentre la 7 (oppure la 3) viene spenta. In questo modo 3 lampade vicine sono accese. Ora, premendo l'interruttore al centro (risp., l'interruttore 2 o 1), le tre lampade vengono spente. A questo punto solo la lampada 5 è accesa. Visto che le altre 6 sono spente, basta premere gli interruttori al centro di un blocco di 3 lampade (il 3 e il 7) per accenderle tutte.



Qualsiasi successione nel premere questi interruttori (1, 2, 3, 7) porta allo stesso risultato. Premere una seconda volta lo stesso interruttore, invece, annulla la prima operazione. Dunque per trovare la soluzione, bisogna solo chiedersi quale interruttore deve essere premuto e quale no. Con 7 interruttori ci sono $2^7 - 1 = 127$ possibilità di combinazioni diverse (premere o no un dato interruttore), Non premerne nessuno è ovviamente sbagliato.

Questa è l'informatica!

Questo esercizio consiste nel condurre un sistema da un determinato stato iniziale (lampade 3, 5 e 7 accese) in un altro finale (tutte le lampade accese), osservando determinate regole.
In informatica ci si confronta con questi problemi molto spesso. Ingenuamente si potrebbero provare tutte le 127 combinazioni, ma spesso è importante trovare la soluzione rapidamente. In questo caso è utile lavorare contemporaneamente su entrambi gli stati, da quello iniziale in avanti e quello da

finale a ritroso, fino a trovare una convergenza. Più il sistema è grande, più questo metodo di ricerca bidirezionale consente di risparmiare tempo.

Parole chiave e siti web

ricerca bidirezionale, automa a stati finiti

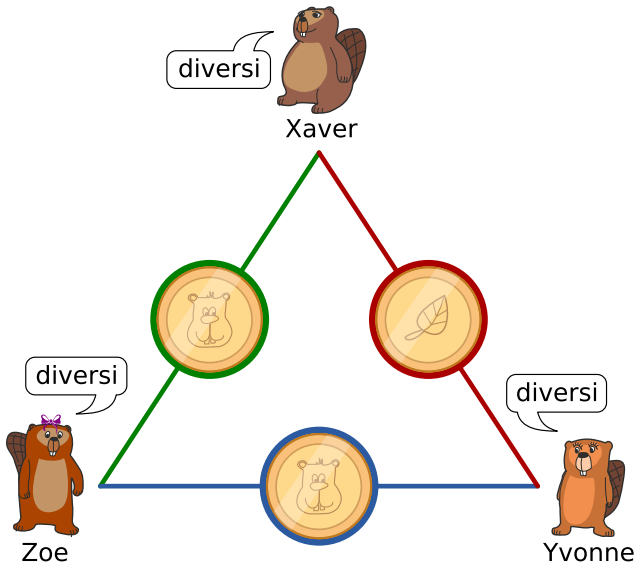
- `https://en.wikipedia.org/wiki/Bidirectional_search`
- `https://it.wikipedia.org/wiki/Automa_a_stati_finiti`

15. Grande segreto

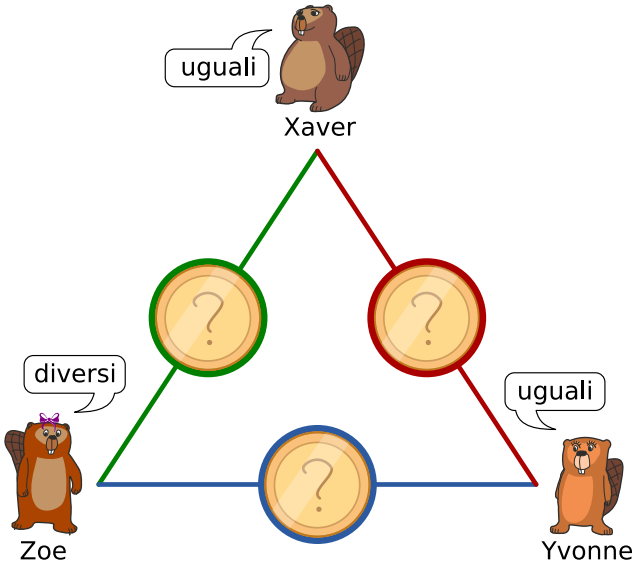
Xaver, Yvonne e Zoe giocano occasionalmente alla lotteria. I media hanno annunciato che il biglietto vincente è stato venduto nella loro città. I tre vorrebbero sapere se uno tra loro è per caso il vincitore, ma d'altro canto il nome esatto dovrebbe rimanere segreto. Ecco quindi come procedono:

1. Xaver e Yvonne lanciano una moneta, solo loro ne conoscono il risultato.
2. Xaver e Zoe lanciano una moneta, solo loro ne conoscono il risultato.
3. Yvonne e Zoe lanciano una moneta, solo loro ne conoscono il risultato.
4. Ognuno di loro poi dice se i propri due rispettivi lanci sono stati "uguali" o "diversi".
    - Chi non ha vinto il premio dovrebbe rispondere sinceramente.
    - Chi ha vinto il premio dovrebbe mentire (dicendo "uguali", se i suoi due lanci erano diversi e viceversa).

Qui sotto vediamo un esempio del lancio di monete e le relative affermazioni, con l'ipotesi che Zoe sia la vincitrice.



Considera la seguente situazione in cui vengono fatti i lanci di monete, senza però che tu ne conosca l'esito:



Quale delle seguenti affermazioni è vera?

A) Nessuno dei tre amici ha vinto il premio.

B) Uno dei tre amici ha vinto, ma non si può sapere chi.

C) Uno dei tre amici ha vinto e si può sapere chi.

D) Non sappiamo se qualcuno ha vinto il primo premio.

Soluzione

La risposta corretta è B). Uno dei tre amici ha vinto, ma non si può sapere chi.
Esistono diverse situazioni che possano descrivere il nostro risultato, ma per il lancio delle monete possono esserci solo queste due possibilità:

- Tutte e tre le monete mostrano lo stesso risultato.
- Una moneta mostra un risultato diverso rispetto alle altre due.

Supponendo che nessuno abbia vinto il premio principale, si applica quanto segue:

- Se tutte le monete sono uguali, tutti e tre gli amici dicono "uguali".
- Se una moneta mostra un risultato diverso, uno degli amici dice "uguali" e gli altri due dicono "diversi".

Dal momento che due amici dicono "uguali" e uno "diversi", uno di loro deve mentire. Quindi uno di loro ha vinto il premio.
Ma non possiamo dire chi: se uno dei due amici che dicono "uguali" ha mentito, non possiamo distinguere il vincitore tra loro, mentre se a mentire è stato chi ha detto "diversi", non potremmo comunque escludere la prima possibilità.

Questa è l'informatica!

L'approccio scelto da Xaver, Yvonne e Zoe è stato descritto per la prima volta come il problema dei "Dining Cryptographers" ("criptografi a cena").
Questo metodo è particolarmente interessante per gli informatici, poiché sia il mittente (vincitore) che il destinatario (gli altri) del messaggio rimangono anonimi: se il processo viene eseguito correttamente, alla fine si saprà solamente se il vincitore è tra loro. Nessuno, ad eccetto del vincitore stesso, saprà da dove viene quell'informazione. Allo stesso modo, anche chi non ha vinto resta anonimo.

Parole chiave e siti web

Anonimità, problema dei "Dining Cryptographers"

- `https://en.wikipedia.org/wiki/Dining_cryptographers_problem`

A. Autori dei quesiti

Andrea Adamoli
Jared Asuncion
Wilfried Baumann
Daphne Blokhuis
William Chan
Kessarapan Charoensueksa
Anton Chukhnov
Kris Coolsaet
Valentina Dagienė
Christian Datzko
Susanne Datzko
Marissa Engels
Hanspeter Erni
Veerle Fack
Gerald Futschek
Ionuț Gorgos
Shuchi Grover
Martin Guggisberg
Urs Hauser
Juraj Hromkovič
Svetlana Jakšić
Zhang Jinbao
Emil Kelevedjiev
Dong Yoon Kim
Vaidotas Kinčius
Iryna Kirynovich
Regula Lacher
Judith Lin
Violetta Lonati
Nils Mak
Dimitris Mavrovouniotis
Mattia Monga
Anna Morpurgo
Tom Naughton
Henry Ong
Wolfgang Pohl
Stavroula Prantsoudi
Nol Premasathian
J.P. Pretti
Doris Reck
Chris Roffey
Kirsten Schlüter
Andrea Maria Schmid
Victor Schmidt
Andrea Schrijvers
Eljakim Schrijvers
Vipul Shah
Jacqueline Staub
Allira Storey
Ahto Truu
Willem van der Vegt
Jiří Vaníček
Troy Vasiga
Rechilda Villame
Eslam Wageed
Pieter Waker
Michael Weigend

B. Sponsoring: concorso 2018

http://www.haslerstiftung.ch/

http://www.roborobo.ch/

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

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

Standortförderung beim Amt für Wirtschaft und Arbeit Kanton Zürich

i-factory (Musée des transports, Lucerne)

http://www.ubs.com/

http://www.bbv.ch/

http://www.presentex.ch/

http://www.zubler.ch/
Zubler & Partner AG Informatik

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

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

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

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.