INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Quesiti e soluzioni 2018
Tutte le Categorie

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

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

AEIOUHWY	X
BFPV	1
CGJKQSXZ	2
DT	3
L	4
NM	5
R	6

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

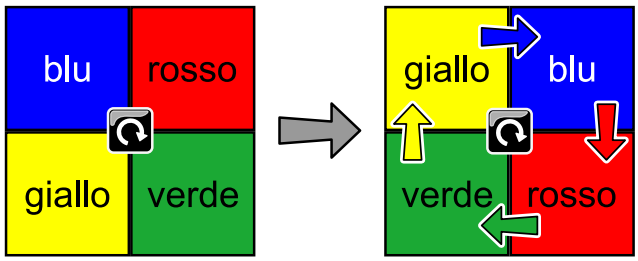
Per ulteriori informazioni:

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

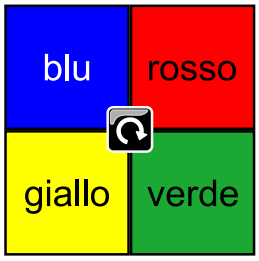
1. Strega comanda color...

Quando la strega preme il bottone al centro, i quadrati colorati ruotano come indicato:

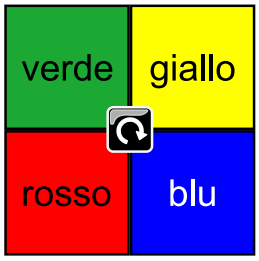


Ora la strega preme il bottone al centro altre due volte ancora. Dove saranno posizionati i quadrati dopo?

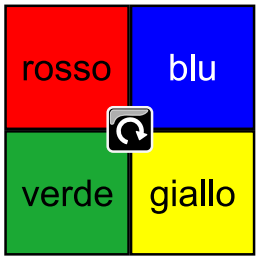
A)



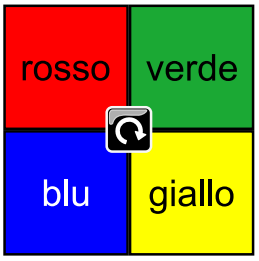
B)



C)

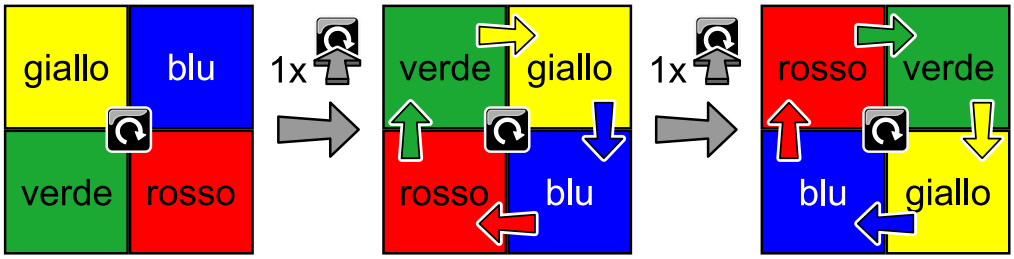


D)



Soluzione

La risposta corretta è D):



Questa è l'informatica!

Il nostro compito descrive una macchina che possiede uno stato specifico per ciascuna delle quattro posizioni colorate: rosso, verde, blu o giallo. Premendo il pulsante si modifica lo stato di ogni posizione, nella successione rosso → blu → giallo → verde → rosso. In informatica un sistema simile è detto *automa a stati finiti*. Nel nostro esempio, (per le persone) è però più facile concepire il cambiamento di stato come una "movimento" dei colori, dunque possiamo dire che i colori ruotano in senso orario.
Da notare che ogni cambiamento di stato nel nostro "automa" per una delle posizioni colorate, segue sempre determinate regole. Innanzitutto, la successione di colori è predefinita (ad esempio, non si può passare direttamente dal blu al verde) e poi il cambiamento avviene solo nel momento in cui il bottone è premuto dalla strega. Molti dispositivi elettronici odierni funzionano con questo principio.

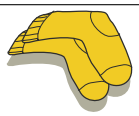
Parole chiave e siti web

automa a stati finiti

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

2. Il mucchio di vestiti

La mamma del piccolo castoro Bruno sistema i vestiti sul tavolo:

Camicia	Canottiera	Pantaloni	Mutande	Bretelle	Calzini	Scarpe
						

Bruno indossa i vestiti nell'ordine in cui sono sul tavolo. Inizia sempre con l'indumento che si trova in cima al mucchio di vestiti. Bruno non vuole assolutamente indossare le bretelle sotto la camicia.
Quale mucchio di vestiti è ordinato in modo giusto per Bruno?

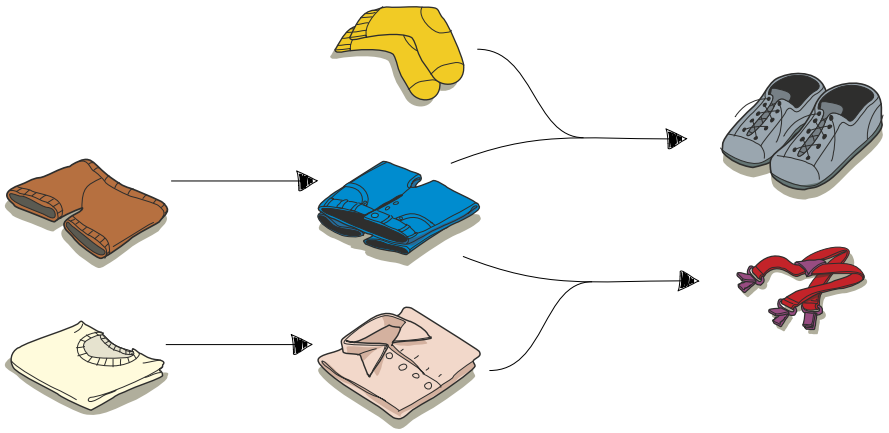
A)	B)	C)	D)
			

Soluzione

La risposta corretta è B).
Per trovare la soluzione bisogna osservare la sequenza di vestiti partendo da quello in alto e verificare che essa segua le regole rispettando le condizioni (la camicia deve essere indossata prima delle bretelle).
Le risposte A), C) e D) sono errate in quanto le bretelle verrebbero indossate prima della camicia.

Questa è l'informatica!

"Prima di entrare in una stanza, devi aprire la porta". In questa semplice azione vediamo un esempio chiaro di condizione: prima di ottenere ciò che vuoi (entrare nella stanza), la porta deve essere aperta.
Il nostro compito può essere risolto controllando quali mucchi soddisfino le condizioni nella forma: "L'indumento X deve essere indossato prima dell'indumento Y". Se un mucchio di vestiti soddisfa tutte le condizioni, è corretto. Se una sola condizione non viene soddisfatta, non è corretto.
Di fatto gli indumenti del compito dovrebbero rispettare più condizioni rispetto a quella relativa alle bretelle: per esempio, le scarpe non possono essere indossate prima dei calzini o dei pantaloni.



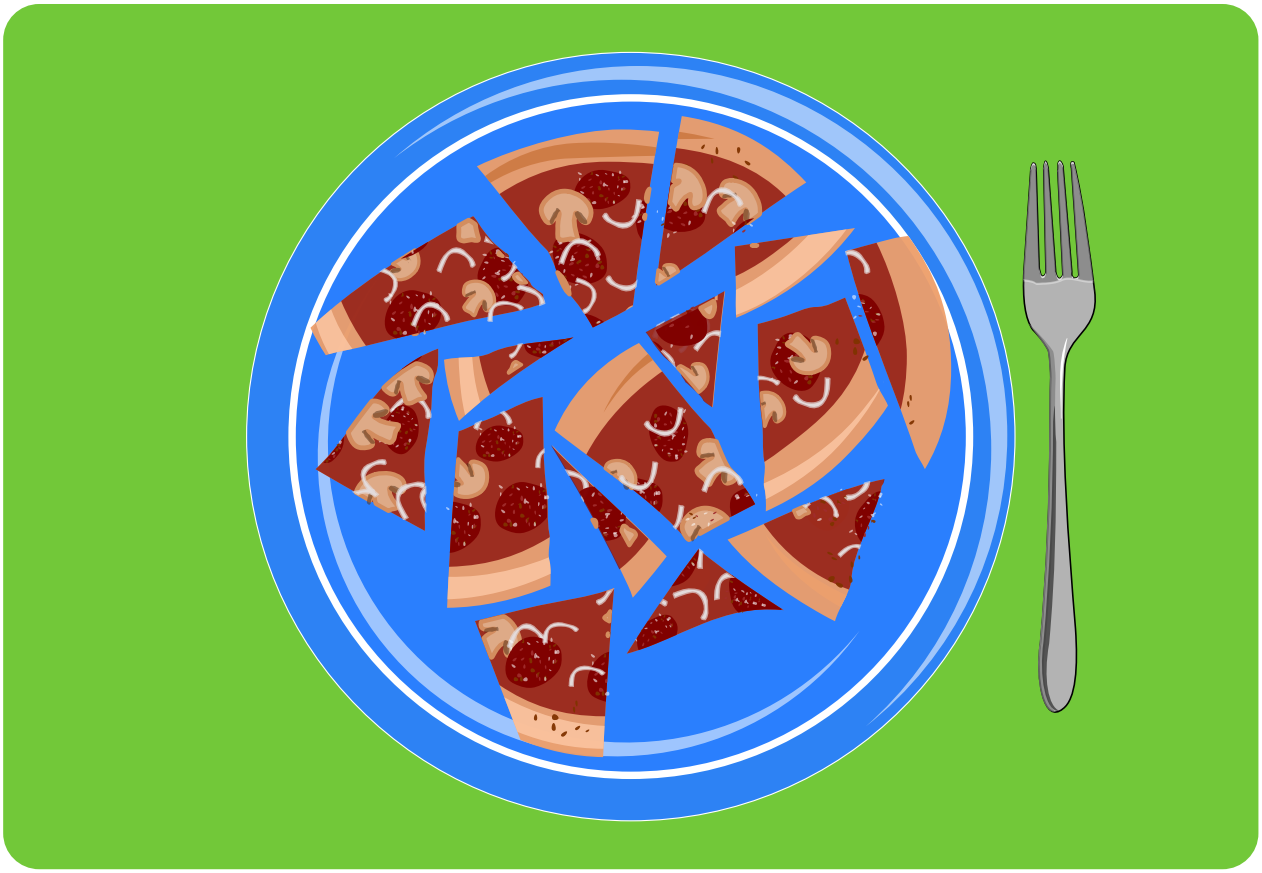
Parole chiave e siti web

Scheduling (pianificazione)

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

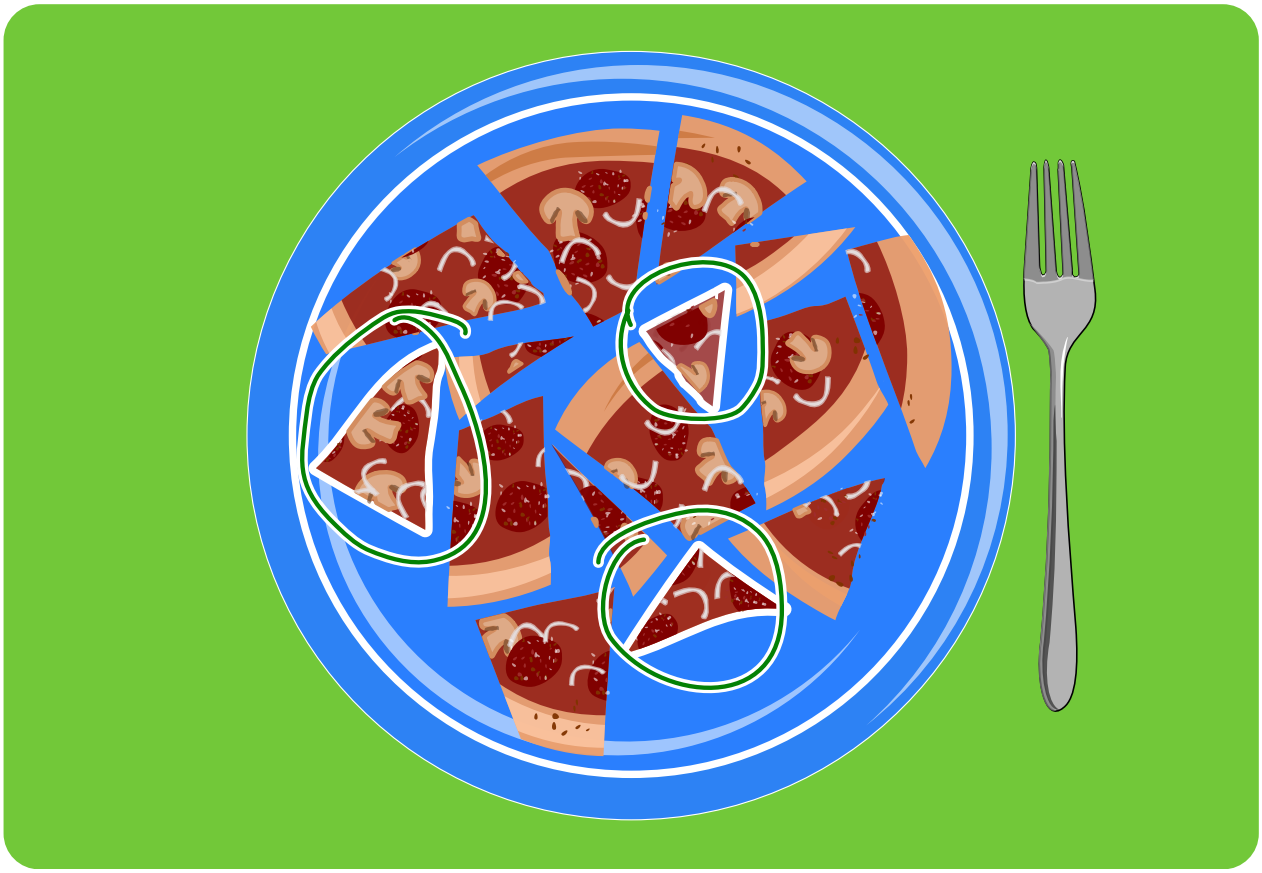
3. Pizza

La mamma di Lucia ha tagliato la pizza a pezzi. Lucia vorrebbe mangiare tutto con le mani. I pezzi di pizza senza crosta possono però essere mangiati solo con la forchetta. Quali sono?



Soluzione

Die richtige Antwort ist:



Questi tre pezzi non hanno la crosta, gli altri si.

Questa è l'informatica!

Lucia deve verificare se ogni pezzo di pizza che vede abbia la crosta oppure no. Decisioni simili devono essere prese anche dai computer per una serie di situazioni diverse: esse sono chiamate selezioni.
Se si dovesse programmare un computer per comportarsi come Lucia, scriveremmo:
SE il pezzo di pizza ha la crosta
    ALLORA mangialo con le mani
    ALTRIMENTI mangialo con la forchetta

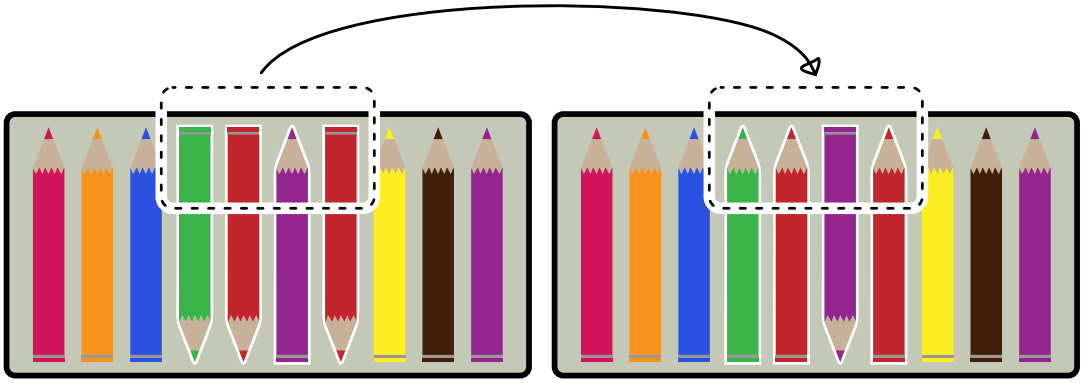
Parole chiave e siti web

selezione, condizione

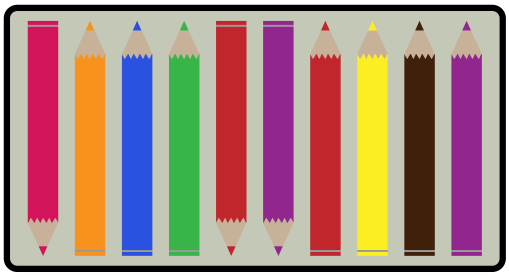
- `https://it.wikipedia.org/wiki/Selezione_(informatica)`

4. Matite colorate

Ada possiede un astuccio con 10 matite colorate. Alcune hanno la punta rivolta verso il basso, altre verso l'alto. Ada vorrebbe che ogni matita fosse rivolta verso l'alto.
Per divertirsi, decide di girare sempre 2 o più matite vicine per volta. Nell'esempio seguente Ada gira la quarta, la quinta, la sesta e la settima matita allo stesso momento.

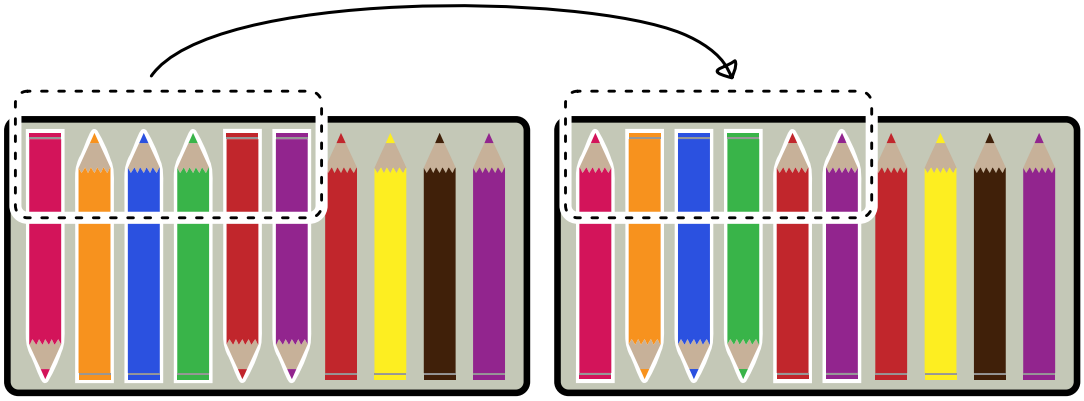


Ada vorrebbe che tutte le matite fossero rivolte verso l'alto nel numero minore di mosse. Quale è il numero minimo di mosse necessario?

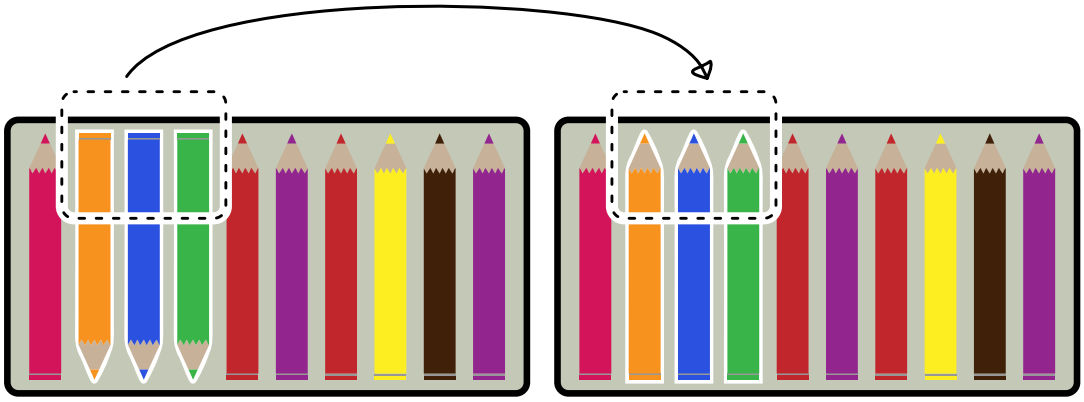


Soluzione

Sono sufficienti 2 mosse. Una non basta, dato che le matite rivolte verso il basso non sono tutte vicine.
Ada deve girare dapprima le matite da 1 a 6...



...e in seguito le matite da 2 a 4...



...così tutte le matite sono rivolte verso l'alto come desidera Ada.

Questa è l'informatica!

Ada ha certamente il tempo per compiere tutte le mosse che vuole, fino a trovare la posizione giusta per tutte le matite. In caso poi perdesse la pazienza, potrebbe anche non rispettare la regola e girare una sola matita per volta.
Ma per i computer non è così facile. Quando si scrive un programma, si impongono delle regole alle quali la macchina deve attenersi. I computer, inoltre, non hanno a che fare con poche matite, bensì con grosse quantità di dati per cui devono essere applicate le stesse regole, nel modo più veloce possibile.
Gli Hard Disk dei computer costituiscono un esempio importante: oggigiorno sono molte diffuse le unità SSD (Solid State Disk). In esse, quando si vogliono cancellare o scrivere delle informazioni, non si può gestire il singolo dato ma si deve sempre considerare un intero blocco composto da molte informazioni. Per rimpiazzare un singolo dato (matita), bisogna dapprima caricare in memoria l'intero blocco, modificare il dato, cancellare il vecchio blocco e quindi riscriverlo sulla SSD. Quando si scrive un blocco, bisogna però accertarsi che esso non sia già utilizzato per altre informazioni. La ricerca di un blocco libero, così come la cancellazione di quelli vecchi, richiede però molto tempo. Per questo i computer utilizzano il tempo di inattività per mappare i blocchi liberi e cancellare quelli non più utilizzati.

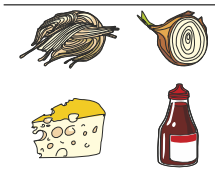
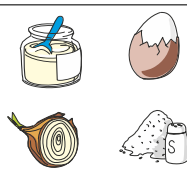
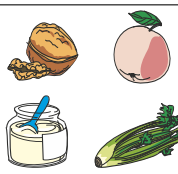
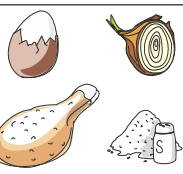
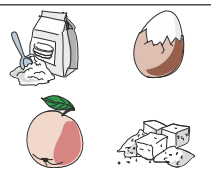
Parole chiave e siti web

Efficienza

- `https://www.codechef.com/problems/ADACRA`
- `https://it.wikipedia.org/wiki/O-grande`
- `https://it.wikipedia.org/wiki/TRIM`

5. Piatti simili

Un cuoco desidera cucinare 2 piatti. Questi piatti non devono però essere simili. Per il cuoco, due piatti sono simili se contengono almeno 2 ingredienti uguali.

Pasta	Insalata con uova	Insalata con noci	Minestra di pollo	Torta
				

Quali sono i piatti simili?

A) Minestra di pollo/Pasta

B) Minestra di pollo/Insalata con noci

C) Minestra di pollo/Insalata con uova

D) Insalata con noci/Torta

Soluzione

La risposta corretta è C) Minestra di pollo/Insalata con uova.
In entrambi i piatti ci sono uova, cipolle e sale.
In tutte le altre coppie di piatti c'è al massimo un solo ingrediente in comune:

- Minestra di pollo/Insalata con noci non hanno alcun ingrediente in comune
- Minestra di pollo/Pasta hanno solo la cipolla in comune
- Insalata con noci/Torta non hanno alcun ingrediente in comune

Questa è l'informatica!

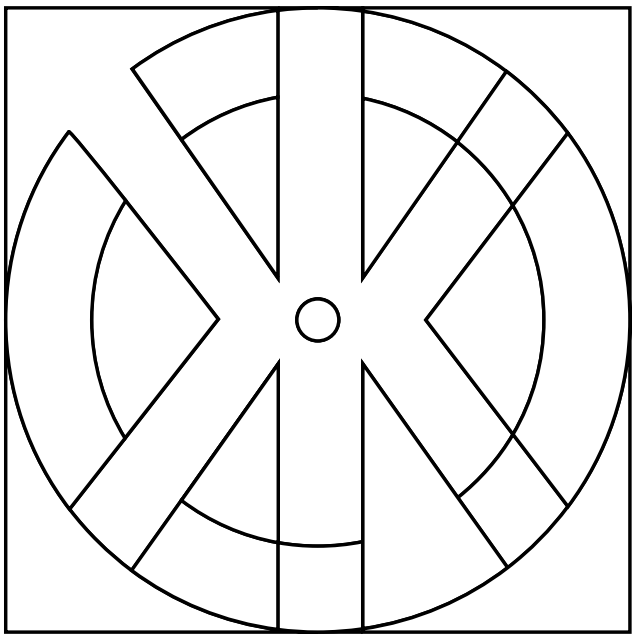
In molti contesti è necessario confrontare degli oggetti per scoprirne differenze e similarità. Ad esempio i biologi confrontano la composizione genetica dei batteri, i chimici le proprietà delle sostanze, gli astronomi la forma delle galassie o la composizione di stelle e pianeti, ecc.
Per poter confrontare degli oggetti bisogna dapprima definire quale proprietà deve essere valutata. Solo così possiamo dire se due oggetti sono simili oppure no. Per esempio, essendo tavolo e sedia fatti entrambi di legno, possiamo affermare che essi sono simili per composizione, ma se ne analizziamo l'utilizzo, tavolo e sedia sono invece diversi.
Nel nostro quesito si confrontano cinque piatti composti da soli quattro ingredienti ciascuno. Biologi, chimici, astronomi e molti altri scienziati devono invece confrontare milioni o addirittura miliardi di oggetti, spesso con numerose proprietà diverse, che ne determinano la definizione di somiglianza. È qui che entra in gioco l'informatica, grazie alla quale si possono confrontare automaticamente grandi quantità di dati sulla base di una misura di similarità predefinita.

Parole chiave e siti web

oggetti, proprietà, misure di distanza e di similarità, big data ("grandi dati")

- `https://en.wikipedia.org/wiki/Similarity_measure`
- `https://it.wikipedia.org/wiki/Big_data`

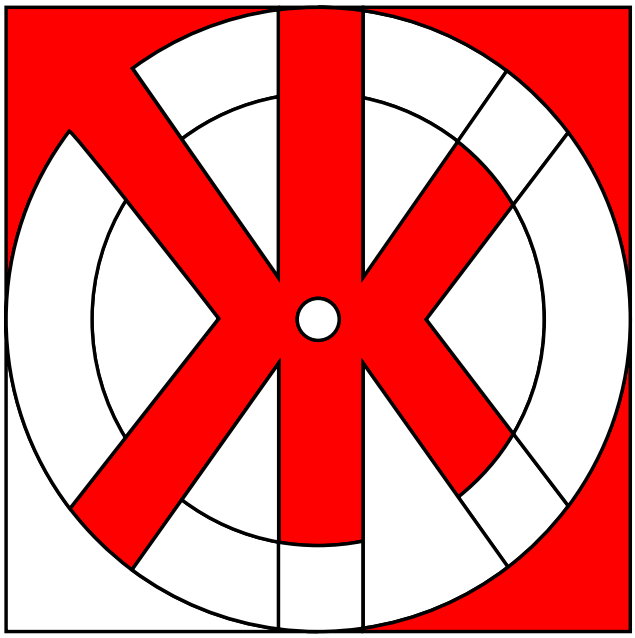
6. Colora le forme



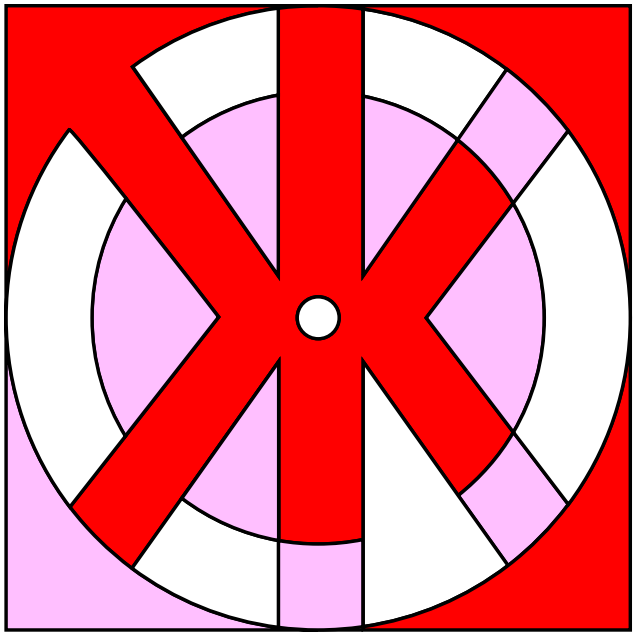
I castori desiderano colorare la forma qui sopra. Aiutali a colorarla, facendo in modo che due superfici vicine ("che si toccano") non abbiano lo stesso colore. Utilizza inoltre il minimo numero di colori diversi.

Soluzione

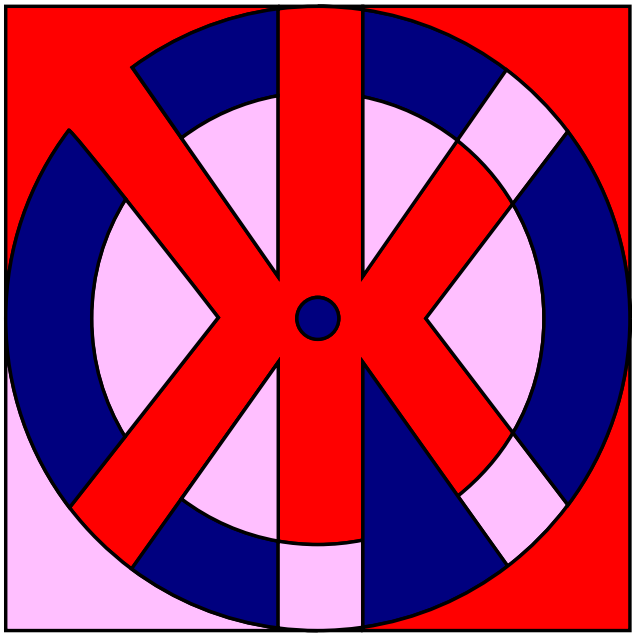
3 colori sono sufficienti per colorare l'intera forma, senza che 2 superfici adiacenti abbiano lo stesso colore.
Naturalmente esistono molte soluzioni a seconda del colore con cui si inizia. Possiamo ad esempio iniziare con il rosso e colorare l'angolo in alto a sinistra. Di seguito coloriamo poi tutte le superfici che non si toccano. La figura appare così:



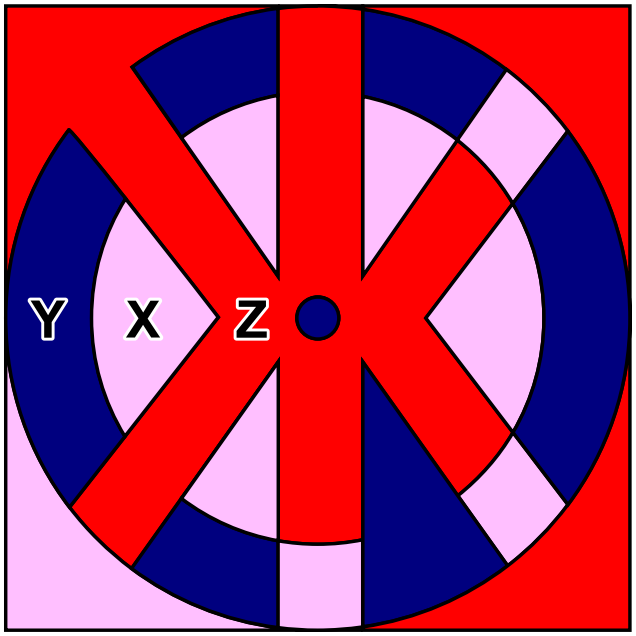
Se ora scegliamo un ulteriore colore (ad esempio il rosa) e lo applichiamo partendo dall'angolo in basso a sinistra e di seguito alle superfici non adiacenti, la figura appare così:



La figura è di fatto già conclusa, poiché con 3 colori (rosso, rosa e bianco) essa è stata completamente "colorata". Possiamo però naturalmente sostituire il bianco con un ulteriore tono, ad esempio il blu:



Non è possibile utilizzare meno di 3 colori. Le superfici X, Y, Z sono reciprocamente adiacenti e dunque devono avere 3 colori differenti.



Questa è l'informatica!

Quanti colori diversi sono necessari per colorare una forma, senza che superfici vicine ("adicenti") abbiano lo stesso colore? La risposta è 4, se non esistono enclavi. Un enclave è una superficie che appartiene ad un'altra, che però non è in contatto con essa. Campione d'Italia, Büsingen am Hochrhein o Baarle (appartenente sia all'Olanda sia al Belgio) sono esempi di enclavi, territori appartenenti ad uno stato, ma inclusi in uno diverso.
La dimostrazione che 4 colori sono sufficienti non è semplice. Già nel 1800 si era dimostrato che 5 colori bastavano per colorare una mappa (geografica), ma solo nel 1976 i matematici Kenneth Appel e Wolfgang Haken, hanno potuto dimostrare che anche 4 sono sufficienti. Per fare ciò hanno però

utilizzato i computer per testare una grossa quantità di eccezioni e controesempi. Dal momento che non è stato possibile controllare ogni situazione manualmente, ci sono ancora oggi molti matematici che mettono in dubbio la validità di questa prova e, in generale, l'utilizzo di computer per dimostrare teoremi.
Il teorema dei quattro colori è utilizzato in molte occasioni, ad esempio per allestire piani di volo (e relative distanze tra i corridoi) o per la copertura delle antenne per cellulari.

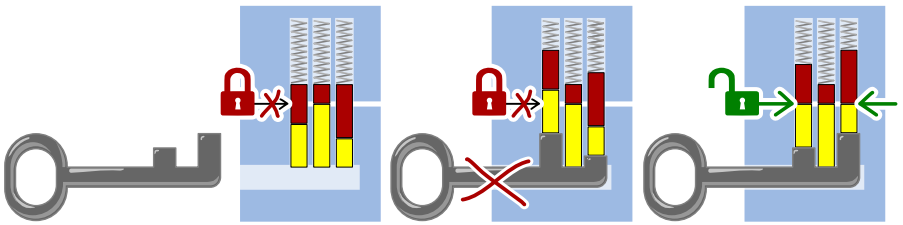
Parole chiave e siti web

Teorema dei quattro colori

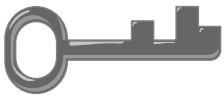
- `https://it.wikipedia.org/wiki/Teorema_dei_quattro_colori`
- `http://www.mathepedia.de/Vier-Farben-Satz.html`
- `https://it.wikipedia.org/wiki/Enclave`
- `https://en.wikipedia.org/wiki/Baarle`

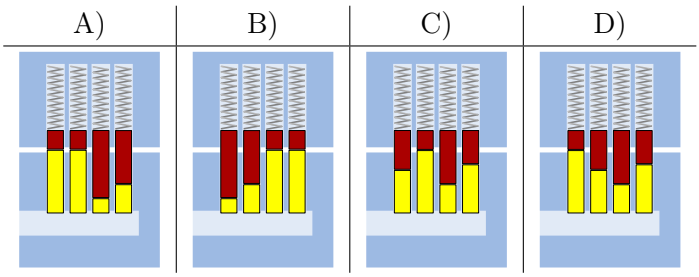
7. Serratura

Enrico lavora per un servizio di serrature. Una serratura funziona così:



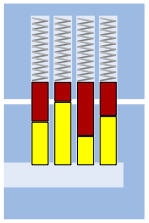
Quale serratura è giusta per la chiave seguente:



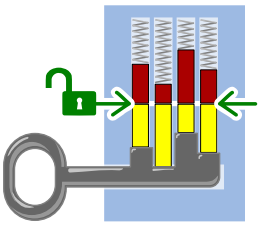


Soluzione

La risposta corretta è C):



Se la chiave viene inserita nella serratura, i pistoncini del cilindro (in giallo) sono allineati e consentono allo stesso di ruotare.



Questa è l'informatica!

Una chiave può aprire o chiudere una serratura, solo se i suoi denti si adattano perfettamente ai pistoncini del "cilindro". Un dente lungo si accoppia ad un pistoncino corto e viceversa, in modo che una volta inserita la chiave tutti i pistoncini siano allineati e permettano al cilindro di ruotare.
Per capire se una chiave si adatta ad un cilindro dobbiamo quindi osservare la "forma" di entrambi e vedere se esse combaciano. In informatica il riconoscimento o la ricerca di particolari forme (o meglio, modelli – pattern) "di dati" è un'attività basilare. Ad esempio si può cercare l'occorrenza di una determinata parola (in tutte le sue "forme") in un testo oppure delle immagini simili.

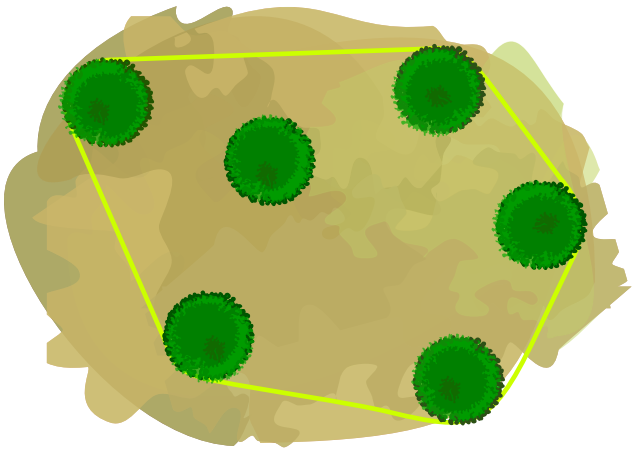
Parole chiave e siti web

riconoscimento di pattern (modelli), serratura (cilindro)

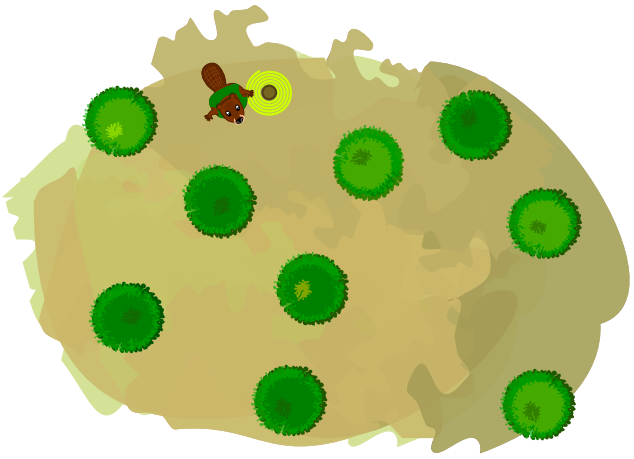
- `https://it.wikipedia.org/wiki/Cilindro_(serratura)`

8. Gruppo di cespugli

I castori avvolgono un lungo nastro attorno ai gruppi di cespugli che vogliono abbattere.
Ieri volevano abbattere un gruppo di sei cespugli. Il nastro è stato avvolto però solo lungo cinque dei cespugli. La situazione, dall'alto, appariva così:



Oggi i castori vogliono abbattere questi nove cespugli:



Lungo quanti cespugli dovrà essere avvolto il nastro?

A) 3 cespugli

B) 4 cespugli

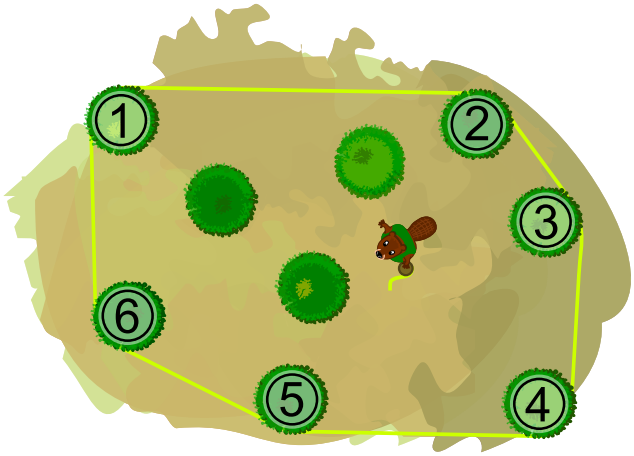
C) 5 cespugli

D) 6 cespugli

E) 9 cespugli

Soluzione

La risposta corretta è D), 6 cespugli. I castori avvolgono il nastro lungo i cespugli in questo modo:



Il nastro è avvolto dunque lungo i 6 cespugli numerati.

Questa è l'informatica!

Il nastro delimita l'area più piccola, nella quale sono inseriti tutti i cespugli che devono essere abbattuti. L'unica limitazione consiste nel fatto che il perimetro dell'area è formato da linee rette. Se i sei arbusti nell'immagine della soluzione fossero dei punti, la superficie che ne deriva sarebbe un esagono (non regolare).
Un poligono più piccolo, che contiene tutti i punti di un determinato insieme, è chiamato in matematica "inviluppo convesso" ("convex hull") di tale insieme di punti. Qui, "convesso" si riferisce al tipo di angolo formato dai lati (in opposizione a "concavo"), mentre pur "inviluppo" si intende "involucro" (la reale definizione matematica è un po' più complessa). Dunque, il nastro viene avvolto attorno ai cespugli formando un "involucro convesso".
In informatica, vengono determinati gli inviluppi convessi di un insieme di punti per diversi scopi:

- Riconoscimento pattern ("modelli"): c'è una faccia in un'immagine?
- Riconoscimento della grafia: il carattere scritto a mano è la lettera B?
- Sistemi di informazione geografica: qual è la dimensione di una zona di piena o di un sistema fluviale?
- Confezionamento: qual è la quantità minima di materiale sufficiente per confezionare un articolo specifico?

L'informatica ha elaborato metodi che possono calcolare in modo efficiente l'inviluppo convesso di un insieme di punti. Questi metodi sono molto efficienti anche con insiemi di punti molto grandi.

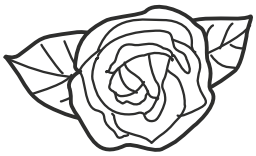
Parole chiave e siti web

grafo, inviluppo convesso

- `https://it.wikipedia.org/wiki/Inviluppo_convesso`
- `https://en.wikipedia.org/wiki/Convex_hull_algorithms`
- `https://brilliant.org/wiki/convex-hull`

9. I fiori di Clara

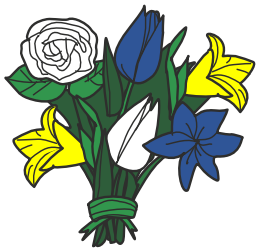
A Clara piacciono i mazzi di fiori colorati e quindi va in un negozio di fiori, dove trova i seguenti tipi:

Gladioli	Gigli	Tulipani	Rose
			

Ogni tipo di fiore può essere ottenuto in questi colori: bianco, blu e giallo.
Clara vuole creare un mazzo composto da sei fiori, che soddisfi le seguenti regole:

1. Ogni colore (bianco, blu e giallo) deve essere impiegato esattamente 2 volte.
2. I fiori dello stesso tipo non devono possedere lo stesso colore.
3. Nel mazzo non devono esserci più di 2 fiori dello stesso tipo.

Quale mazzo di fiori soddisfa tutte le 3 *regole?*

A)	B)	C)	D)
			

Soluzione

La risposta corretta è D). Nel mazzo A) ci sono 3 fiori bianchi (regola 1 non rispettata), nel mazzo B) ci sono 3 rose (regola 3 non rispettata) e nel mazzo C) ci sono 2 fiori dello stesso tipo con lo stesso colore (regola 2 non rispettata).

Questa è l'informatica!

In generale, i problemi informatici sono caratterizzati da una serie di vincoli, ovvero da condizioni o regole, che devono essere rispettate. La sfida consiste nel trovare una soluzione che soddisfi tutti questi limiti o, almeno, il maggior numero possibile.
In informatica si risolvono problemi molto complessi, dove i vincoli sono espressi attraverso operazioni logiche, come ad esempio le operazioni *AND* ("E") e *OR* ("O"): A *AND* B significa che entrambe le condizioni A e B devono essere soddisfatte (come le tre regole nel nostro compito); A *OR* B significa invece che sarà sufficiente che solo una delle condizioni sia soddisfatta.

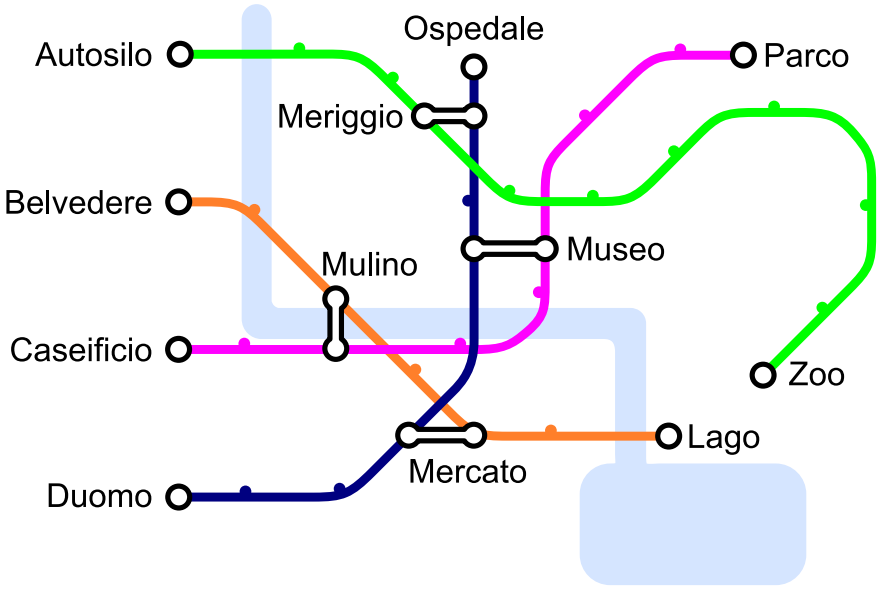
Parole chiave e siti web

condizioni, operatori logici, espressioni logiche

- `https://bookofbadarguments.com/de/?view=allpages`
- `https://it.wikipedia.org/wiki/Algebra_di_Boole`
- `https://www.iep.utm.edu/prop-log/`

10. Linee del tram

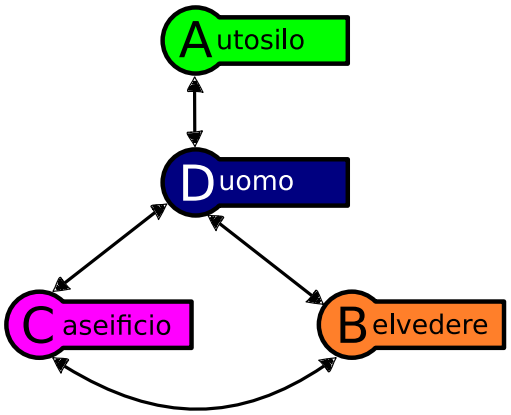
In una città, ci sono quattro linee del tram che partono rispettivamente dai capilinea "Autosilo", "Belvedere", "Caseificio" e "Duomo". Ci sono anche quattro fermate di scambio che permettono di cambiare linea, ossia "Museo", "Mercato", "Mulino" e "Meriggio".



Giovanni desidera andare allo Zoo. Egli sa che deve cambiare linea solo una volta! Quale è il capolinea della sua linea?

Soluzione

La risposta corretta è "Duomo". Se seguiamo a ritroso la linea che arriva allo Zoo, vediamo che esiste una sola fermata di scambio, ossia "Meriggio", nella quale Giovanni deve necessariamente cambiare tram. Questa linea parte dalla fermata "Duomo".
Per trovare la soluzione possiamo anche rappresentare le linee con un grafo, i cui archi indicano le possibilità di cambio tra le linee del tram (i nodi):



Da	A		
Autosilo ↔ Zoo	Duomo ↔ Ospedale		
Belvedere ↔ Lago	Caseificio ↔ Parco	Duomo ↔ Ospedale	
Caseificio ↔ Parco	Belvedere ↔ Lago	Duomo ↔ Ospedale	
Duomo ↔ Ospedale	Belvedere ↔ Lago	Caseificio ↔ Parco	Autosilo ↔ Zoo

Se si vuole cambiare una sola volta per salire sulla linea "Autosilo ↔ Zoo", si può farlo solo dalla linea "Duomo ↔ Ospedale", che parte dalla fermata "Duomo".

Questa è l'informatica!

Probabilmente hai già visto delle reti simili. Molte linee dei trasporti pubblici – bus, tram, metro, ecc. – hanno infatti una struttura simile. Questa rappresentazione è dovuta a Harry Beck, il quale elaborò una piano simile nel 1931 per rappresentare la metropolitana di Londra.
Piani del genere vengono chiamati grafi, dove i nodi sono le diverse fermate (stazioni), mentre gli archi rappresentano le linee (binari o strade). In essi si distinguono in particolare nodi con un solo arco (i capilinea), con due archi (stazioni normali) e quelli con più archi (stazioni di scambio).
I grafi sono molto utilizzati in varie attività che si basano su programmi informatici, come reti sociali, pianificatori di itinerario, consigli per lo shopping on-line, ecc. Per un informatico è dunque importante saperli utilizzare con sicurezza.

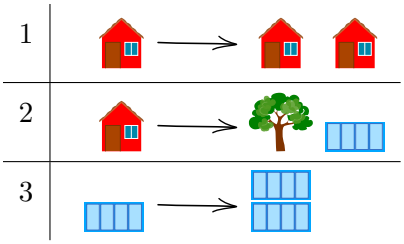
Parole chiave e siti web

Grafi, reti (dei trasporti)

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

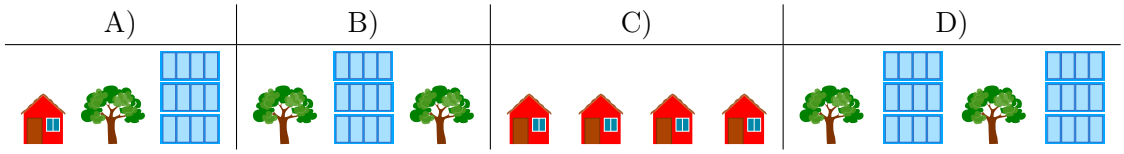
11. Pianeta Z

Gli abitanti del pianeta Z costruiscono le loro città sempre allo stesso modo. Ogni città inizia con la costruzione di una casa. Le varie costruzioni vengono poi rimpiazzate con le regole seguenti:



Applicando dapprima la regola 1, poi la regola 2 e infine due volte la regola 3, si ottiene per esempio la città mostrata a destra dell'immagine seguente:

Attenzione: l'ordine nella sequenza di costruzione non può essere modificata.
Quale delle seguenti città non *può appartenere al pianeta Z?*



Soluzione

La risposta corretta è B). Gli alberi possono essere inseriti in una città solo dalla regola 2 ed essa impone che a destra di una albero debba esserci un blocco: nella città B) non c'è. Siccome non esiste alcuna regola per rimuovere i blocchi, non è possibile che la città B appartenga il Pianeta Z.
La città A) può essere costruita applicando le seguenti regole: 1, 2, 3 e poi ancora 3.
La città C) può essere costruita applicando la regola 1 tre volte.
La città D) può essere costruita in questo modo: per prima cosa si applica la regola 1, quindi si applica la regola 2 per ciascuna casa e infine si applica la regola 3 a ciascun blocco due volte.

Questa è l'informatica!

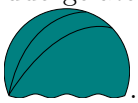
Le regole del nostro quesito sono dette "regole di sostituzione": un simbolo o un oggetto viene sostituito da una sequenza di altri simboli o oggetti. Se viene sostituito un solo simbolo o oggetto alla volta, tale sistema è detto "senza contesto": in altre parole, un simbolo o un oggetto viene sostituito da qualcos'altro senza rispettare il contesto, cioè senza considerare anche ciò che gli sta sulla destra e/o sulla sinistra.
Le regole di sostituzione sono utilizzate in informatica, per definire, ad esempio, la sintassi di un linguaggio di programmazione. I simboli o gli oggetti sono parole chiave e le regole descrivono come trasformarli in un programma (sintatticamente) corretto. Nel nostro quesito, i simboli o gli oggetti sono le case, gli alberi e i blocchi. I simboli e gli oggetti, insieme alle regole di sostituzione, formano quindi una "grammatica" che descrive un linguaggio.
Quando il computer traduce un programma in linguaggio macchina (ossia, compila il programma) o lo esegue direttamente (ossia, "interpretata" uno script), controlla innanzitutto se il testo del programma segue effettivamente le regole del linguaggio di programmazione. Cerca quindi di ricreare le regole di sostituzione con l'aiuto di un albero di sintassi che generi il testo del programma (che nel nostro quesito corrisponderebbe alle quattro opzioni proposte) dal simbolo di inizio (nel nostro quesito, una casa). Questo è molto semplice per noi, poiché le parole (ovvero, le sequenze di simboli o oggetti: per noi una città) diventano sempre più grandi e mai più piccole (gli oggetti non spariscono).

Parole chiave e siti web

regole di sostituzione, grammatica, linguaggi liberi dal contesto

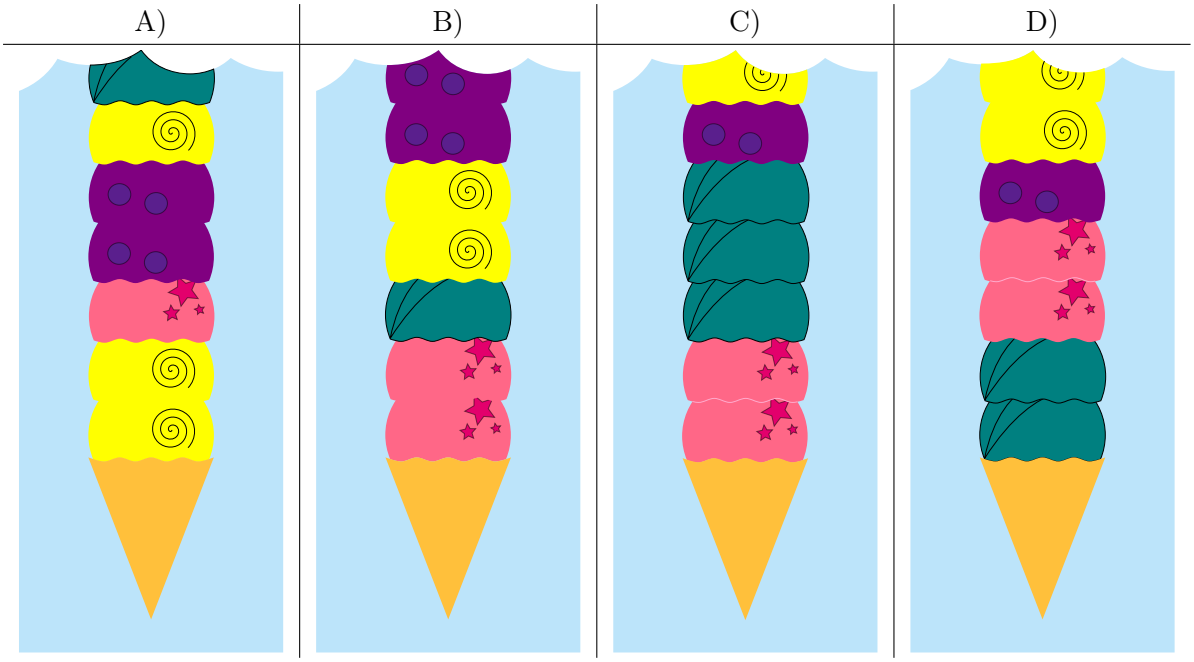
- `https://en.wikipedia.org/wiki/Production_(computer_science)`
- `https://it.wikipedia.org/wiki/Linguaggio_libero_dal_contesto`
- `https://it.wikipedia.org/wiki/Albero_sintattico`

12. Gelateria

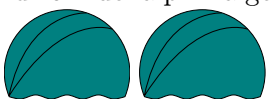
In un villaggio ci sono due gelaterie. Esse offrono le stesse quattro varietà di gelato:  ,  ,  e  .
La prima gelateria utilizza le seguenti istruzioni per un fare i cornetti:

1. Prendi un cornetto vuoto.
2. Scegli una varietà di gelato casuale e aggiungi 2 palline di questa varietà.
3. Aggiungi una pallina di una delle altre tre varietà di gelato rimanenti.
4. Quando il numero desiderato di palline è stato raggiunto, fermati. Altrimenti continua dal punto 2.

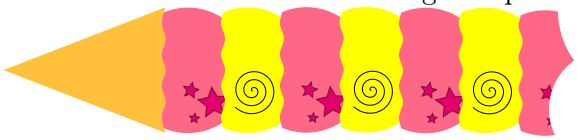
La seconda gelateria, invece, non segue assolutamente nessuna istruzione.
Nelle immagini vengono mostrate solo le parti inferiori di alcuni cornetti. Quale tra questi è stato fatto di sicuro dalla seconda gelateria?



Soluzione

Il cornetto D) è l'unico che non è stato di sicuro realizzato secondo le istruzioni della prima gelateria.
Esso inizia correttamente con due palline della stessa varietà di gelato  seguite da una pallina di un'altra varietà di gelato  . Dopo ci sono però due palline di varietà diversa  , ma secondo le istruzioni della prima gelateria esse dovrebbero essere uguali.
Le risposte A), B) e C) non sono corrette. Tutti questi cornetti, per quanto si possa vedere, seguono le istruzioni della prima gelateria.

Questa è l'informatica!

La struttura (il modello) dei cornetti della prima gelateria può essere descritta attraverso sequenze di istruzioni. La stessa cosa può avvenire per parole (testo) e immagini. Gli informatici sviluppano programmi per computer con i quali possono essere riconosciuti determinati modelli o delle loro deviazioni. Tali modelli vengono spesso creati dalla ripetizione di istruzioni. Questo modello  , ad esempio, è semplicemente creato dalla ripetizione di  e  . Questo particolare modello è molto semplice da riconoscere, quello della gelateria, invece, è un po' più difficile, poiché possiede anche una componente casuale.
In generale, non si può mai sapere con sicurezza se una determinata sequenza sia stata generata a caso o da una serie di istruzioni mirate. Per questo, nel nostro quesito possiamo solo escludere il cornetto che chiaramente non segue le istruzioni, ma non possiamo affermare con assoluta certezza che gli altri tre siano stati fatti dalla prima gelateria: sebbene vi possiamo riconoscere il modello, essi potrebbero benissimo essere il frutto di una composizione casuale.

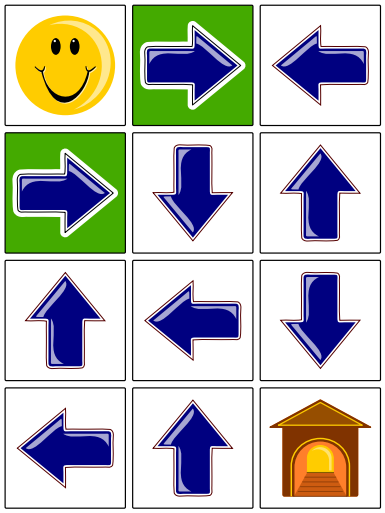
Parole chiave e siti web

pattern recognition ("riconoscimento di modelli o strutture")

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

13. Il labirinto delle frecce

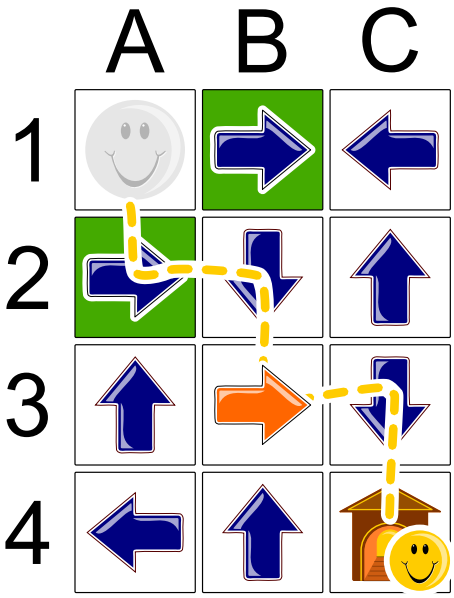
Smiley desidera andare a casa, ma può farlo solo attraversando il labirinto delle frecce. Esso può entrare da uno degli ingressi (caselle verdi). Quando si trova su una casella con una freccia, deve poi spostarsi nella direzione indicata. Con le frecce disposte nel modo seguente è però impossibile che esso arrivi a casa.



A quale singola freccia devi cambiare la direzione, per fare in modo che Smiley giunga a casa?

Soluzione

La freccia la cui direzione deve essere modificata è evidenziata in rosso. Il percorso per arrivare alla casa di Smiley è: A1 – A2 – B2 – B3 – C3 – C4.



Puoi adirittura provare che questa è l'unica soluzione possibile, partendo dalla casella C4 e andando a ritroso. È possibile raggiungere la casa in C4 da due direzioni: da C3 e da B4. La freccia in B4 indica la direzione sbagliata e dovrebbe essere modificata. Ma poiché nessuna casella vicina punta a B4, una seconda freccia dovrebbe essere cambiata e questo non è permesso.
Di conseguenza, la destinazione può essere raggiunta solo attraverso la casella adiacente C3. La freccia in questo campo è già corretta e punta all'obiettivo. Siccome nessuna freccia punta a C3 nelle caselle vicine, una delle due frecce tra B3 e C2 dovrebbe essere ruotata. Non esiste però alcuna freccia che punta a C2 e quindi questa casella non può essere considerata. Dobbiamo dunque necessariamente modificare la freccia in B3. Ora, B3 può essere raggiunta da un ingresso (A2) senza dover modificare altre frecce (A1 – A2 – B2). Questa è dunque l'unica possibilità.

Questa è l'informatica!

In questo esercizio dovevamo identificare il percorso attraverso un labirinto di frecce. Come si trova effettivamente la soluzione a tale problema? Come può trovarla un computer? Nella spiegazione abbiamo utilizzato una tecnica utilizzata in informatica chiamata *backtracking* (monitoraggio a ritroso). In pratica dobbiamo provare a seguire un percorso (partendo dal principio o dalla fine) fino a quando non troviamo una soluzione oppure fino a quando non giungiamo in un vicolo cieco. In questo secondo caso, facciamo un passo indietro (lungo la nostra traccia) e verifichiamo un'altra possibilità (percorso) fino a provarle tutte. Se una soluzione esiste, saremo dunque in grado di trovarla.

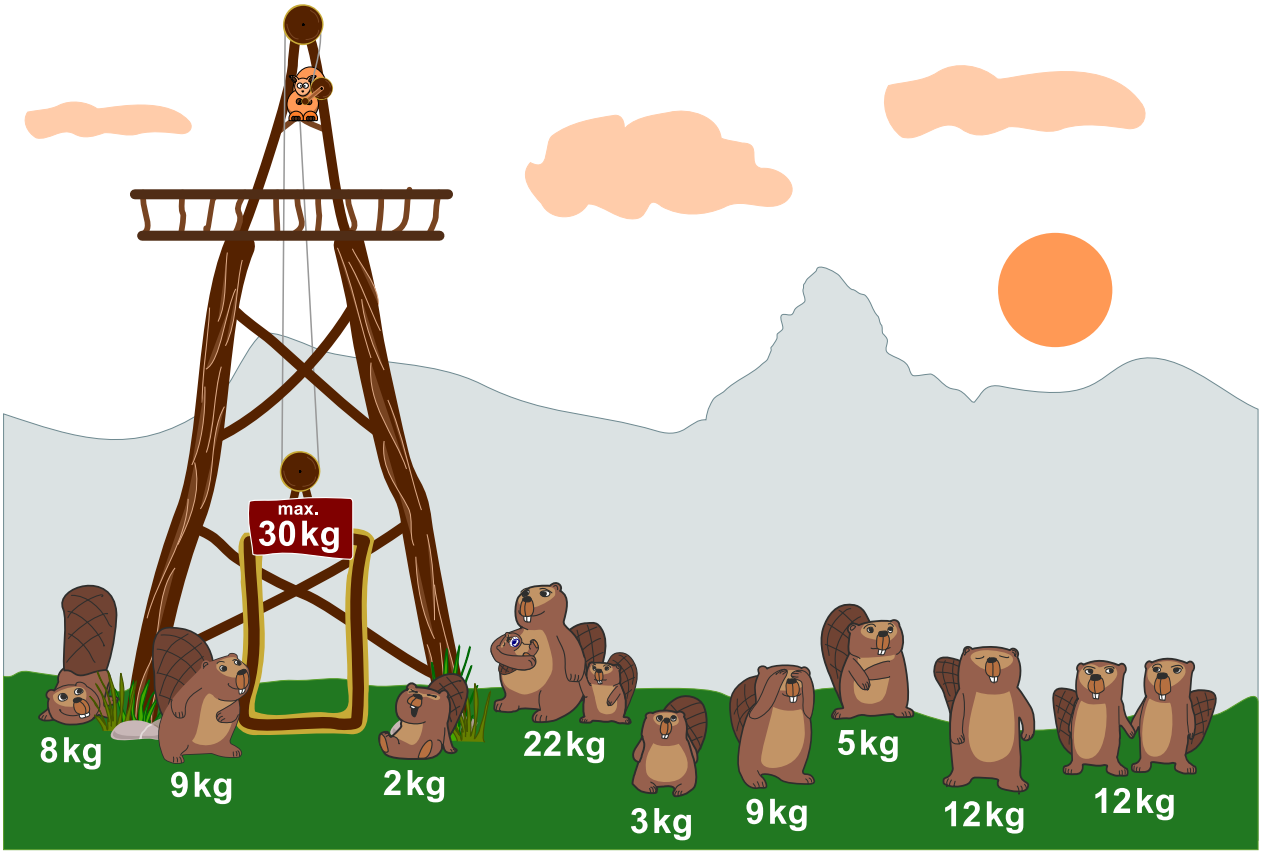
Parole chiave e siti web

labirinto, backtracking ("monitoraggio a ritroso")

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

14. Torre panoramica

Una famiglia di castori fa una gita fino a una torre panoramica. Purtroppo sono arrivati in ritardo e l'ascensore può compiere solo altri due viaggi fino alla cima. L'ascensore può trasportare un massimo di 30 kg per volta. I gemelli possono salire solo insieme sulla torre e la mamma castoro deve salire con il neonato e l'altro piccolo castoro per mano. Sulla torre vorrebbero salire più castori possibile.



Restando poco tempo prima della chiusura, i castori considerano solo una delle seguenti opzioni. Chi deve restare a terra per consentire al maggior numero di castori di salire sulla torre?

A) Nessuno: tutti i castori potranno salire sulla torre.

B) La mamma con il neonato e l'altro piccolo.

C) I gemelli e il castoro di 5 kg.

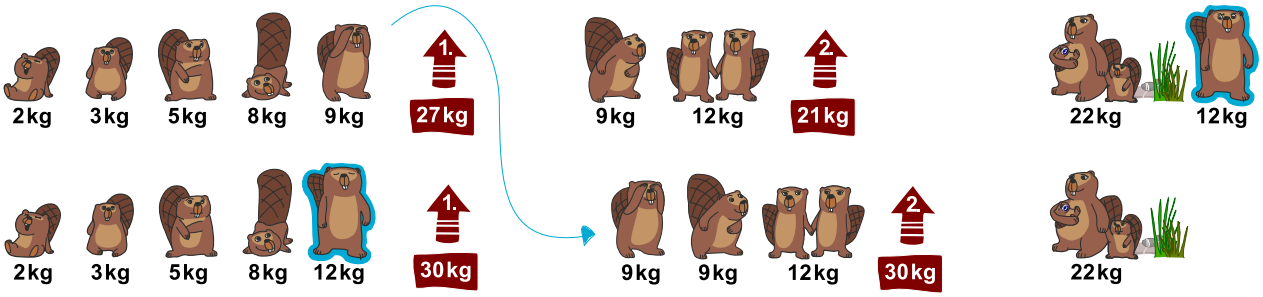
D) La mamma con il neonato e l'altro piccolo e anche i gemelli.

E) La mamma con il neonato e l'altro piccolo e anche il castoro di 12 kg.

Soluzione

La risposta corretta è B).

Per verificare questa soluzione, possiamo riempire dapprima l'ascensore con i castori più leggeri: $2\,\text{kg} + 3\,\text{kg} + 5\,\text{kg} + 8\,\text{kg} + 9\,\text{kg} = 27\,\text{kg}$ e poi con i rimanenti $9\,\text{kg} + 12\,\text{kg} = 21\,\text{kg}$.
Ora possiamo cercare di distribuire i pesi in modo più opportuno ("ottimizzare"):



Spostando il castoro da 9 kg nella seconda corsa, la capacità dell'ascensore è utilizzata al massimo (30 kg) e si trova il posto anche per il castoro di 12 kg nella prima, la quale, a sua volta, raggiunge la capacità massima.
La risposta A) non è corretta, in quanto il peso totale dei castori supera la capacità massima delle due corse (60 kg). Analogamente la risposta C) non è corretta, poiché anche lasciando a terra i gemelli e il castoro di 5 kg il peso totale sarebbe ancora 65 kg. Le risposte D) ed E) non sono corrette poiché distribuendo opportunamente il carico (vedi soluzione B) non è necessario lasciare a terra anche i gemelli o il castoro di 12 kg. Tali soluzioni, quindi, non consentirebbero al maggior numero di castori di salire sulla torre.

Questa è l'informatica!

L'ottimizzazione è uno dei problemi classici dell'informatica. In questo ambito, spesso non è possibile trovare in tempi ragionevoli le soluzioni ottimali ("le migliori"), poiché le possibilità da verificare sono infinite.
Il nostro quesito è una versione del famoso "problema dello zaino", in cui è necessario imballare il maggior numero possibile di oggetti senza superare il peso massimo. Problemi simili hanno la caratteristica di essere *NP-completi*. In pratica, significa che non possono essere risolti in tempi ragionevoli: si può solo trovare una soluzione possibile abbastanza buona, ma non necessariamente la soluzione ottimale. Soluzioni relativamente buone sono ottenute applicando strategie opportune ("euristiche").

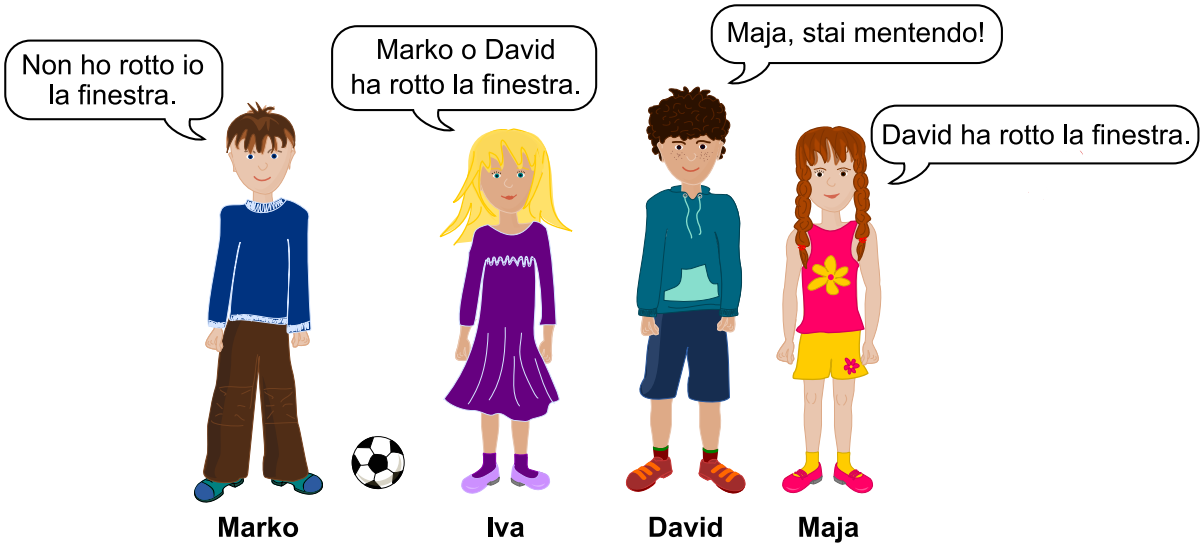
Parole chiave e siti web

ottimizzazione e ricerca operativa, problema dello zaino

- `https://en.wikipedia.org/wiki/Combinatorial_optimization`
- `https://it.wikipedia.org/wiki/Ricerca_operativa`
- `https://it.wikipedia.org/wiki/Problema_dello_zaino`

15. Le bugie hanno le gambe corte

In una giornata soleggiata, Maja, David, Iva e Marko giocano a pallone vicino alla casa di Anna. Qualcuno però rompe improvvisamente una finestra e Anna vuole sapere chi è stato. Anna conosce bene i bambini e sa che tre di loro dicono sempre la verità, mentre il quarto potrebbe anche mentire. I quattro bambini fanno queste affermazioni:



Chi ha rotto la finestra?

Soluzione

David ha rotto la finestra.

Le affermazioni di David e Maja non possono essere entrambe vere. Uno di loro deve quindi mentire.... Se Maja dicesse la verità, allora David mentirebbe, altrimenti se David dicesse la verità, sarebbe Maja a mentire e, di conseguenza, anche Marko oppure Iva. Questo non sarebbe però possibile, poiché tre bambini dicono sempre la verità.

Questa è l'informatica!

Per risolvere questo esercizio bisogna possedere un pensiero logico. I princìpi logici di base furono formulati nel 1854 da George Boole (1815 – 1864), che descrisse le affermazioni logiche riducendole fino alle loro unità.
Secondo questi princìpi, un'affermazione può solo essere vera o falsa (*tertium non datur*). Inoltre, le singole dichiarazioni possono essere combinate con l'aiuto di operatori. Gli operatori logici semplici come AND (*e*) o OR (*oppure*) combinano due affermazioni in una nuova affermazione. Ci sono anche operatori unari (come NOT – *non*) che prendono semplicemente una dichiarazione e ne cambiano il valore logico. Per scoprire il valore (vero o falso) di affermazioni combinate, si possono applicare vari procedimenti tra cui quello della tabella della verità (*truth table*).
L'operatore $\rightarrow$ rappresenta l'implicazione logica ("SE" $\rightarrow$ "ALLORA"), ovvero il processo attraverso cui si traggono delle conclusioni logiche. Proprio questo processo era necessario per risolvere l'esercizio.
I computer sono di principio basati su istruzioni logiche e semplici operatori logici (detti anche *booleani*), la cui semplicità rende molto facile costruirli. Sebbene ci siano alcuni computer basati su altri sistemi (come quelli ternari della fine degli anni '50 in Russia), essi sono sempre rimasti sperimentali o non hanno mai raggiunto grandi numeri.

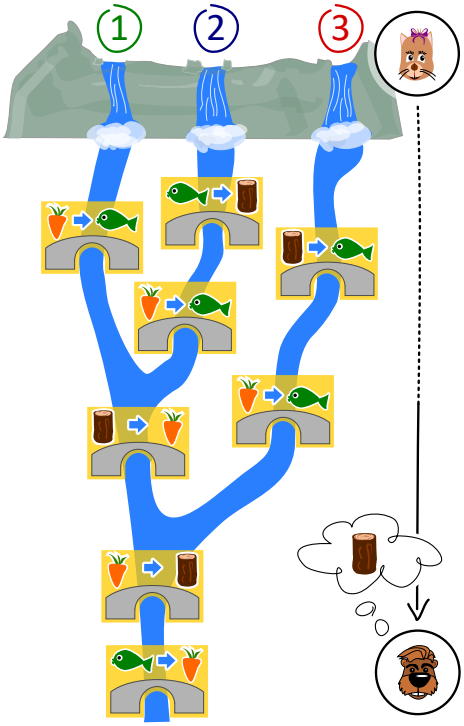
Parole chiave e siti web

logica, logica booleana

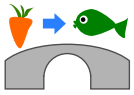
- `https://it.wikipedia.org/wiki/Tertium_non_datur`

- `https://it.wikipedia.org/wiki/George_Boole`
- `https://it.wikipedia.org/wiki/Algebra_di_Boole`
- `https://it.wikipedia.org/wiki/Implicazione_logica`
- `https://it.wikipedia.org/wiki/Calcolatore_ternario`

16. Le cascate



Katja è seduta in cima a una montagna. La montagna ha tre cascate che sfociano in un fiume.
Katja può far cadere una carota o un pesce in una delle cascate. Il fiume è attraversato da diversi ponti su cui vivono dei troll. I troll sostituiscono gli oggetti che passano sotto il ponte.



Ad esempio, quando una carota passa sotto il ponte mostrato, i troll la sostituiscono con un pesce.

Gianni aspetta alla fine del fiume.
Gianni ha bisogno di un tronco di legno. Cosa deve far cadere Katja e in quale cascata, in modo che Gianni possa ricevere alla fine un tronco di legno?

A) Deve far cadere un pesce nella cascata numero 1.

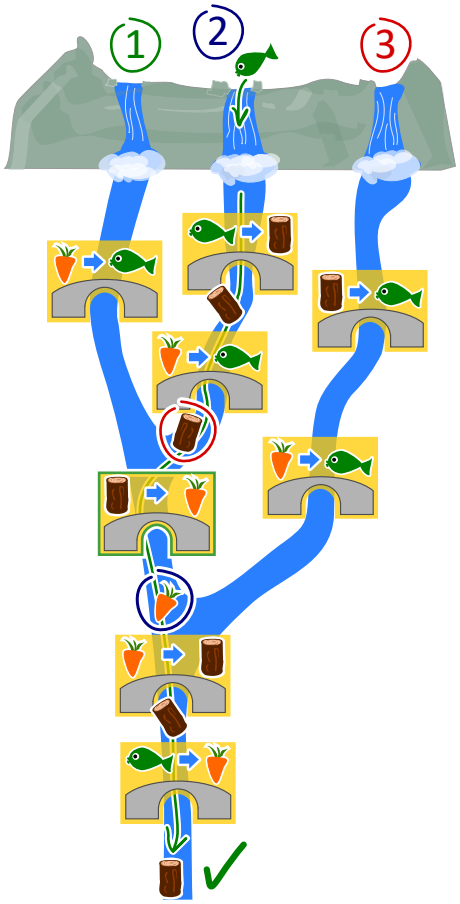
B) Deve far cadere un pesce nella cascata numero 2.

C) Deve far cadere una carota nella cascata numero 2.

D) Deve far cadere una carota nella cascata numero 3.

Soluzione

La risposta corretta è: B) Deve far cadere un pesce nella cascata numero 2.
Questo succede con le diverse possibilità:



A) Un pesce viene lasciato cadere nella cascata 1 e viene scambiato con una carota nell'ultimo ponte. Gianni ottiene quindi una carota.

B) Un pesce viene lasciato cadere nella cascata 2, viene scambiato con un tronco, il quale viene scambiato poi con una carota e infine di nuovo con un tronco. Gianni ottiene quindi correttamente un tronco di legno.

C) Una carota viene lasciata cadere nella cascata 2, viene scambiata con pesce e il pesce con una carota. Gianni ottiene quindi una carota.

D) Una carota viene lasciata cadere nella cascata 3, viene scambiata con un pesce e il pesce con una carota. Gianni ottiene quindi una carota.

Oltre a provare ogni possibilità, un approccio alternativo per trovare la soluzione è quello di tornare indietro:
per poter ottenere alla fine un tronco di legno, una carota deve passare sotto al penultimo ponte. L'unico modo per avere una carota a questo punto è giungervi dalle cascate 1 o 2 (non 3). Al ponte dopo la congiunzione delle cascate 1 e 2 bisogna necessariamente avere un tronco e lo si può ottenere solo lasciando cadere un pesce nella cascata 2.

Questa è l'informatica!

Si può pensare a un computer come a un dispositivo che legge un input (richiesta/dato iniziale), lo elabora e scrive un output (risposta/dato finale). Ma come fa un computer a "sapere" cosa fare? Semplice... alcune persone in precedenza glielo hanno detto, attraverso delle istruzioni racchiuse in programmi. La verifica del corretto funzionamento di questi programmi è detta "collaudo del software".
Esistono molti linguaggi di programmazione e diversi modelli (paradigmi) di programmazione. Uno di questi paradigmi è la programmazione funzionale. Esso si basa su delle funzioni, che -come in matematica- elaborano un input e producono un output. I ponti nel nostro quesito sono delle piccole funzioni e l'intero sistema è un programma scritto con un linguaggio di programmazione funzionale che trasforma il pesce (input) in un tronco (output).

Parole chiave e siti web

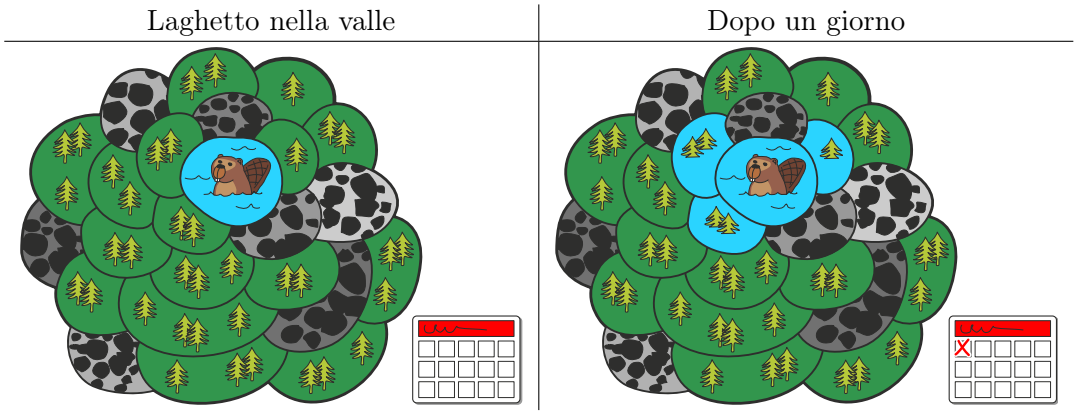
collaudo e test dei software, paradigmi di programmazione, programmazione funzionale, funzioni e parametri

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

- `https://it.wikipedia.org/wiki/Test_strutturale`
- `https://en.wikipedia.org/wiki/Black-box_testing`
- `https://it.wikipedia.org/wiki/Paradigma_di_programmazione`
- `https://it.wikipedia.org/wiki/Programmazione_funzionale`

17. Il laghetto dei castori

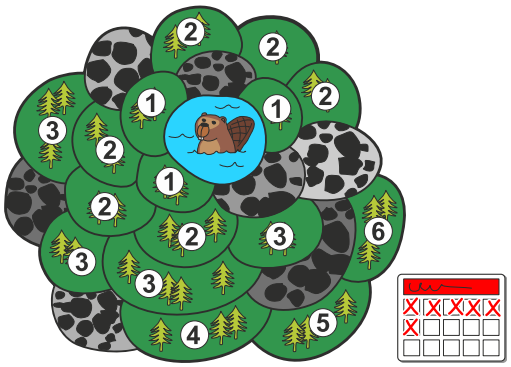
In una valle c'è un piccolo laghetto, circondato da aree boschive o rocciose. Nel laghetto vivono alcuni castori.
Un giorno il laghetto diventa troppo piccolo per i castori e quindi decidono di allagare le aree boschive. Ogni giorno allagano delle nuove aree boschive confinanti con il lago: dopo il primo giorno, sono state dunque allagate 3 nuove aree boschive:



Dopo quanti giorni (incluso il giorno già mostrato nella figura) tutte le aree boschive verrano allagate?

Soluzione

Dopo sei giorni, tutte le aree boschive sono state allagate.
L'immagine mostra quando le diverse aree boschive vengono allagate: quelle inizialmente confinanti con il lago vengono allagate il primo giorno e quindi sono contrassegnate con il numero 1; le aree adiacenti sono allagate il secondo giorno e vengono contrassegnate con il numero 2 e così via. Dopo sei giorni l'intera foresta è stata allagata.



Questa è l'informatica!

Nel nostro quesito, i castori allagano ogni giorno le aree boschive immediatamente vicine. L'intera foresta può essere definita come "connessa" in quanto ogni area boschiva può essere raggiunta da una vicina.
La caratteristica di "connessione" è in determinati ambiti molto importante: nelle immagini digitalizzate, gruppi di pixel sono connessi se racchiusi in una superficie avente un unico colore; nelle reti sociali i vari utenti sono connessi ad esempio da relazioni di amicizia.
In informatica esistono dei metodi (algoritmi) per analizzare aree connesse, come ricerca in ampiezza e in profondità. Tali metodi possono essere utilizzati per sostituire il colore di un'intera superficie o per identificare gruppi sui social networks.

Parole chiave e siti web

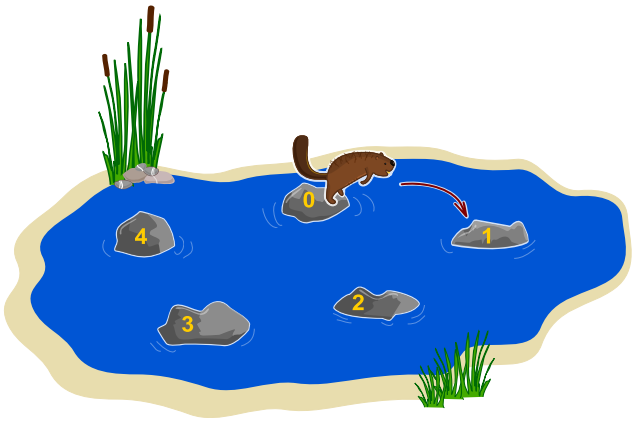
ricerca in ampiezza, algoritmo wavefront ("fronte d'onda")

- `https://it.wikipedia.org/wiki/Grafo_connesso`
- `https://it.wikipedia.org/wiki/Ricerca_in_ampiezza`

18. Il concorso dei castori

Per prepararsi al concorso annuale, i castori si allenano intensivamente. La sessione odierna di esercizi consiste nel saltare di pietra in pietra in senso orario, come mostrato dalla freccia. Se un castoro dovesse saltare 8 volte, partendo dalla pietra numero 0, finirebbe sulla pietra numero 3.

$$0 \to 1 \to 2 \to 3 \to 4 \to 0 \to 1 \to 2 \to 3.$$



Il castoro più in forma ha saltato nella sessione odierna ben 129 *volte, partendo dalla pietra numero* 0. *Su quale pietra ha terminato?*

Soluzione

Ogni 5 salti, il castoro torna sulla pietra dalla quale era partito. Esso ha dunque completato un "giro". Per capire su quale pietra ha terminato la propria serie di salti, dobbiamo dapprima calcolare il numero di giri interi che ha compiuto e quanti salti ulteriori ha poi effettuato. Nel nostro caso, il castoro ha compiuto $129 = 25 \times 5 + 4$ salti (25 giri + ulteriori 4 salti). Dopo 129 salti il castoro termina sulla stessa pietra su cui terminerebbe dopo 4 salti. Dunque, la risposta corretta è la pietra numero 4.

Questa è l'informatica!

Probabilmente avrai già imparato qualcosa di simile nelle lezioni di matematica. Si tratta infatti di calcolare il *resto* di una *divisione intera*, detta anche *divisione euclidea*.
I computer devono spesso effettuare questo tipo di operazione, detta "operazione modulo". Nella programmazione essa viene in genere indicata con l'operatore % o *mod*. Nel nostro esercizio, possiamo quindi scrivere $129\%5 = 4$.
L'operazione modulo è una parte molto importante di molti algoritmi, come quelli di cifratura RSA. Essa è inoltre applicata implicitamente quando il valore massimo rappresentabile in una *variabile* viene superato e si ricomincia a contare partendo da 0, come ad esempio nel nostro esercizio (valore massimo: 4).

Parole chiave e siti web

operazione modulo

- `https://it.wikipedia.org/wiki/Operazione_modulo`
- `https://en.wikipedia.org/wiki/Long_division`
- `https://it.wikipedia.org/wiki/Divisione_euclidea`
- `https://it.wikipedia.org/wiki/RSA`

19. Casa di vacanza Nr. 29

Milo fa uno stage in uno stabilimento con case per vacanze. Oggi dovrebbe assegnare dei numeri alle case vacanza. Alcune di esse sono già state numerate. Per assegnare un nuovo numero, iniziando dalla casa nr. 50, dovrebbe:

- andare a sinistra se il nuovo numero è minore del numero della casa in cui si trova,
- andare a destra se il nuovo numero è maggiore del numero della casa in cui si trova,
- assegnare il nuovo numero di casa, se la casa non è ancora stata contrassegnata.

A quale casa Milo dovrebbe assegnare il nuovo numero 29*?*

Soluzione

La casa a cui assegnare il numero 29 è la terza, partendo da sinistra.

Il numero 29 è minore del numero 50, quindi Milo deve andare dapprima a sinistra. Il 29 viene poi confrontato con il numero 24 e Milo va allora a destra. Analogamente, alla casa 34 Milo va di nuovo a sinistra. La casa successiva non ha un numero e le viene dunque assegnato il numero 29.

Questa è l'informatica!

La numerazione delle case di vacanza corrisponde a un *albero di ricerca binario*, una struttura di dati che viene spesso utilizzata nell'informatica. Con un albero di ricerca binario è possibile trovare rapidamente i dati memorizzati.
Un albero di ricerca binario è costruito in modo tale che ad ogni intersezione (*"nodo"*) venga memorizzato un *"elemento"*. Da ogni nodo, si diramano un massimo di due percorsi (*"archi"*) che portano a ulteriori intersezioni. Per assegnare nuovi elementi a un albero binario, partendo dalla radice (*"root"*), si procede confrontando il nuovo valore con quello del nodo in cui ci si trova e si procede verso sinistra quando il nuovo elemento è minore, verso destra quando è maggiore. Il nuovo elemento viene salvato nella prima intersezione libera.
Quando si cerca un elemento, è semplice scegliere quale arco seguire ad ogni nodo, semplicemente confrontando i valori. Se l'albero di ricerca binario è *"bilanciato"* (un cosiddetto *"albero AVL"*), dopo ogni passo, rimangono solo ca. la metà degli elementi da considerare. Ad esempio in soli 10 passaggi possiamo trovare un valore tra più di 1′000 elementi, in 20 tra più di 1′000′000, in 30 in 1′000′000′000 (ovvero, con n elementi $\log_2(n)$ passi).

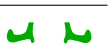
Parole chiave e siti web

Albero di ricerca binario, Albero AVL

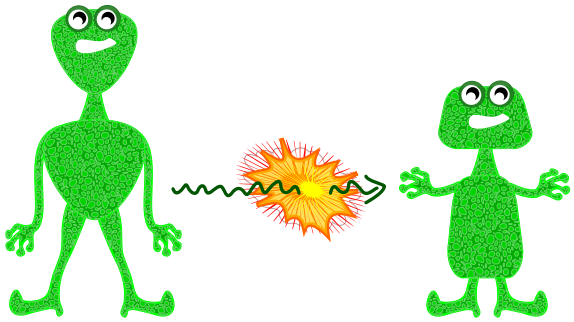
- `https://it.wikipedia.org/wiki/Albero_binario_di_ricerca`
- `https://it.wikipedia.org/wiki/Albero_AVL`

20. Alieni!

Un alieno è formato da un capo, un tronco, due braccia e due gambe. Un alieno può essere modificato attraverso i seguenti comandi, che possono essere applicati anche più volte alla stessa parte del corpo:

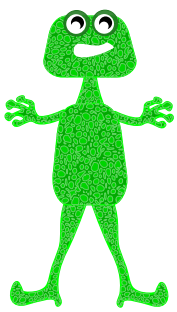
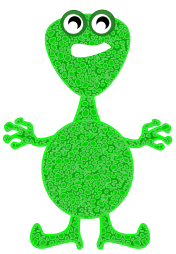
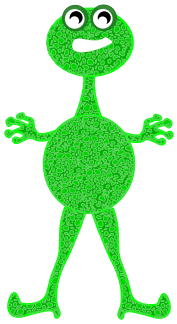
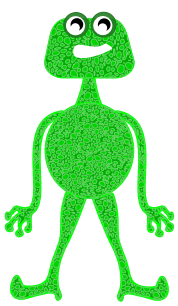
$C(r)$	Il capo è rotondo.	
$C(4)$	Il capo è quadrangolare.	
$C(3)$	Il capo è triangolare.	
$T(r)$	Il tronco è rotondo.	
$T(4)$	Il tronco è quadrangolare.	
$T(3)$	II tronco è triangolare.	
$B(+)$	Le braccia sono lunghe.	
$B(-)$	Le braccia sono corte.	
$G(+)$	La gambe sono lunghe.	
$G(-)$	La gambe sono corte.	

I singoli comandi vengono eseguiti da sinistra verso destra. Per esempio con $C(r)$, $T(4)$, $C(4)$, $B(-)$, $G(-)$ si crea un alieno come quello a destra nella figura seguente:

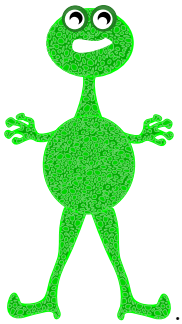


Come appare l'alieno dopo i seguenti comandi?

$$C(3),\ G(+),\ T(3),\ B(+),\ C(r),\ B(-),\ T(r)$$

A)	B)	C)	D)
			

Soluzione

La risposta corretta è C)  .
Per ogni parte del corpo, si applica solo l'ultimo comando. Un comando precedente per la stessa parte del corpo non è riconoscibile nella forma finale dell'alieno, poiché esso viene "sovrascritto" e quindi è inefficace.
Il risultato è un alieno con un tronco rotondo ($T(r)$), le braccia corte ($B(-)$), il capo rotondo ($C(r)$) e le gambe lunghe ($G(+)$). Gli altri alieni differiscono dalla soluzione C) in almeno due parti e quindi sono ovviamente sbagliati.

Questa è l'informatica!

Quando si esegue un programma, i comandi vengono eseguiti in sequenza. I comandi successivi possono modificare l'effetto dei comandi precedenti.
Nei programmi per computer ciò accade spesso con variabili che memorizzano valori che cambiano più volte durante l'esecuzione del programma. Si può quindi pensare alle quattro parti del corpo come a delle variabili in cui è memorizzata la forma corrente. Il comando "$C(r)$" quindi fa in modo che la variabile del capo memorizzi il nuovo valore "r".
La notazione "$C(r)$" utilizzata nel nostro esempio è pensata per essere "funzionale", ossia chiamiamo la funzione "$C()$" e gli passiamo l'argomento "r". Tale notazione è spesso preferita, poiché la funzione "$C()$" permette di controllare che l'argomento passato sia valido. Se ciò non fosse necessario o se la variabile fosse utilizzata solo localmente, sarebbe possibile sovrascriverla direttamente, utilizzando l'operatore di assegnazione "=". Ad esempio, si scriverebbe "$C = r$".

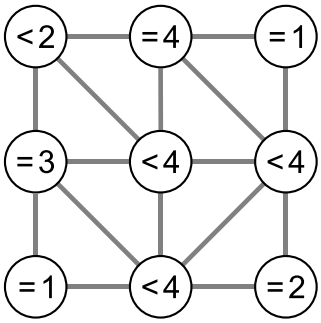
Parole chiave e siti web

variabile, sequenza, programmazione

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

21. Vicini

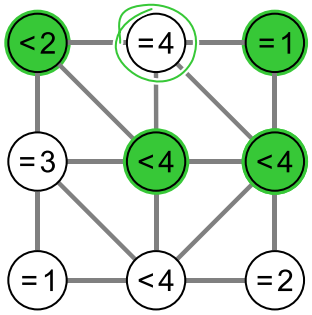
L'immagine qui sotto mostra nove cerchi, in parte connessi. Una connessione (linea) li definisce come "vicini".
Ogni cerchio contiene un'espressione che indica quanti dei suoi vicini devono essere colorati. Per esempio, "$= 3$" significa che esattamente tre dei vicini devono essere colorati; "< 4" significa invece che devono essere colorati al massimo tre dei suoi vicini.



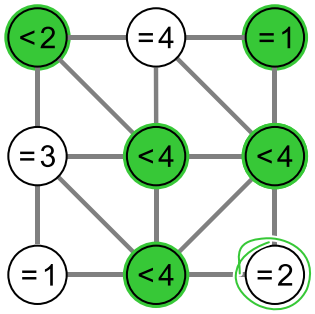
Colora i cerchi in modo che ogni condizione (espressione) sia soddisfatta.

Soluzione

Il cerchio centrale in alto contiene l'espressione "$= 4$" e quindi tutti e quattro i vicini devono essere colorati:



Analogamente, il cerchio in basso a destra contiene l'espressione "$= 2$" e quindi entrambi i vicini devono essere colorati:



A questo punto tutte le condizioni (espressioni) sono già soddisfatte.
Colorando altri cerchi le condizioni non sarebbero soddisfatte:

- Colorando il cerchio "$= 4$" in alto al centro, la condizione del cerchio "$= 1$" in alto a destra non sarebbe più soddisfatta.
- Colorando il cerchio "$= 3$" al centro a sinistra, la condizione del cerchio "< 2" in alto a sinistra non sarebbe più soddisfatta.
- Colorando il cerchio "$= 1$" in basso a sinistra, la condizione del cerchio "$= 3$" al centro a sinistra non sarebbe più soddisfatta.
- Colorando il cerchio "$= 2$" in basso a destra, la condizione del cerchio "< 4" al centro a destra non sarebbe più soddisfatta.

Questa è l'informatica!

Quanti tentativi sono necessari per risolvere il problema? Provando tutte le possibili soluzioni per ciascuno dei 9 cerchi (2 opzioni ciascuno: colorato oppure no) in modo indipendente, si hanno $2^9 = 512$ diverse opzioni. Questo approccio è chiamato "di forza bruta" (brute force), esso consiste nel provare ogni possibilità e verificare se le condizioni sono soddisfatte per ognuna di esse.
L'approccio è però dispendioso in termini di tempo ed è quindi più efficiente procedere in modo logico e coerente. Bisogna dapprima cercare le espressioni (cerchi) che possono essere univocamente soddisfatte. Queste sono, ad esempio, tutti i cerchi contenenti un'uguaglianza "$= n$", con esattamente

n connessioni (vicini). Da quel punto in poi si può procedere con il ragionamento logico: si deve semplicemente verificare se esistono condizioni insoddisfatte che possono essere soddisfatte. Un approccio analitico, in cui vengono prese in considerazione solo le soluzioni più promettenti possibili, è chiamato euristico. Con questo approccio è spesso possibile determinare anche se esista una soluzione oppure no tra le varie possibilità.

Parole chiave e siti web

induzione logica, intorno e vicinanza nei grafi (neighbourhood)

- `https://en.wikipedia.org/wiki/Neighbourhood_(graph_theory)`
- `https://it.wikipedia.org/wiki/Algoritmo_euristico`
- `https://it.wikipedia.org/wiki/Metodo_forza_bruta`

22. 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.

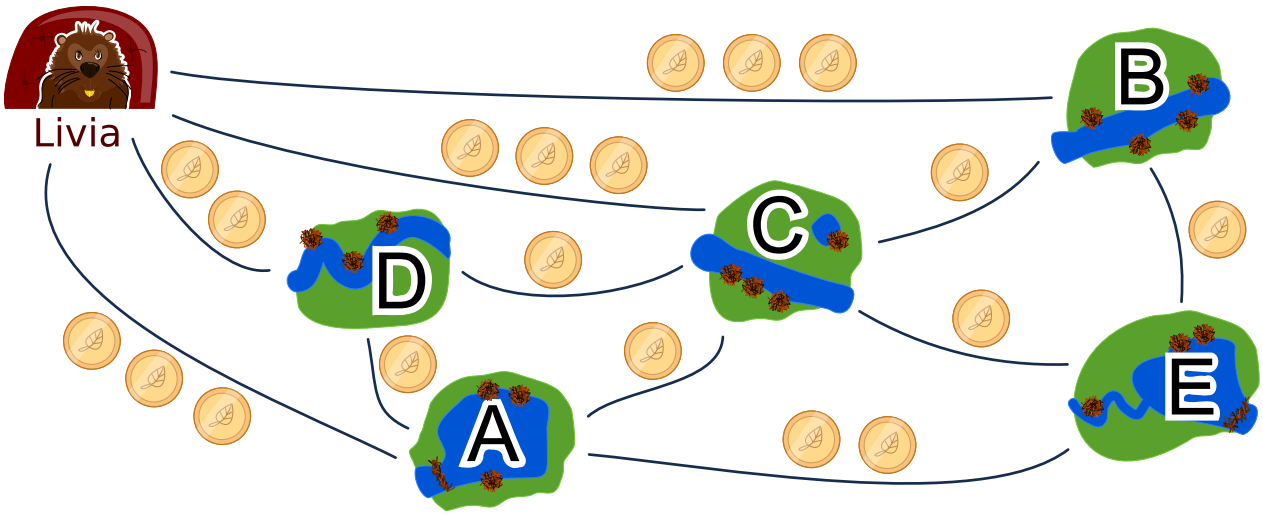
Parole chiave e siti web

analisi, verifica e validazione del software

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

23. 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:

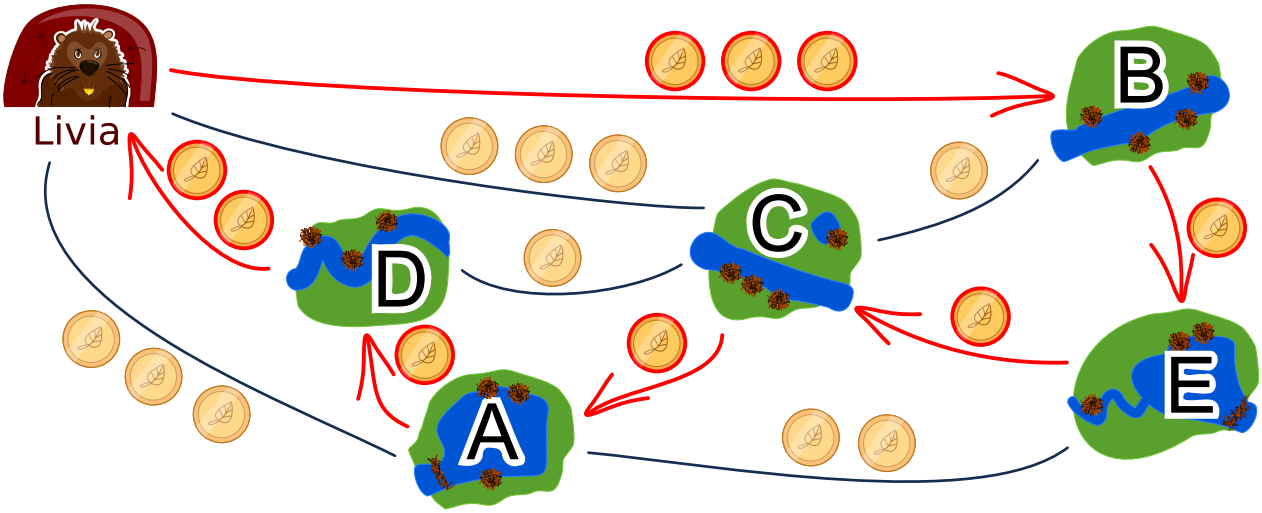
$$\text{Casa} \to \text{B} \to \text{E} \to \text{A} \to \text{D} \to \text{C} \to \text{Casa}.$$

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.

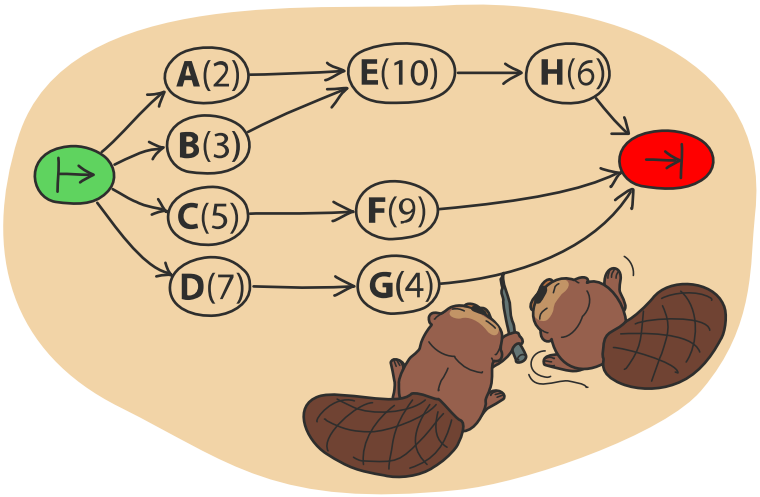
Parole chiave e siti web

ottimizzazione, problema del commesso viaggiatore

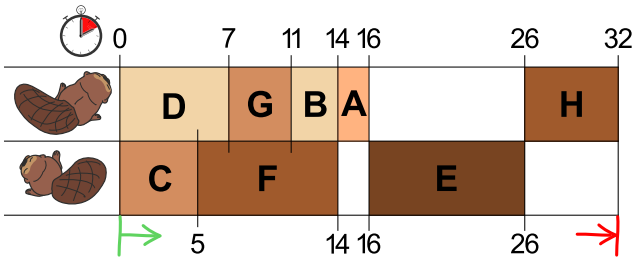
- `https://it.wikipedia.org/wiki/Problema_del_commesso_viaggiatore`
- `https://it.wikipedia.org/wiki/Problema_di_ottimizzazione`

24. 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.



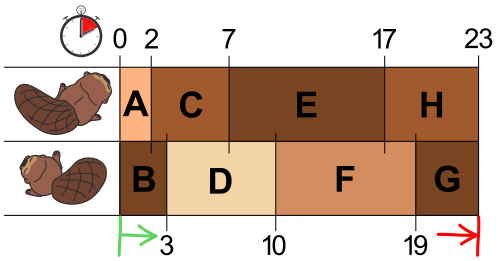
La figura qui sotto mostra un possibile piano di lavoro per i due castori... Si può però di sicuro terminare la diga più velocemente!



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

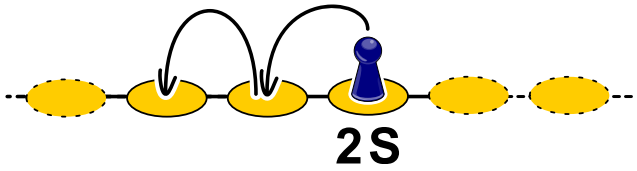
scheduling ("pianificazione"), algoritmo greedy ("avido")

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

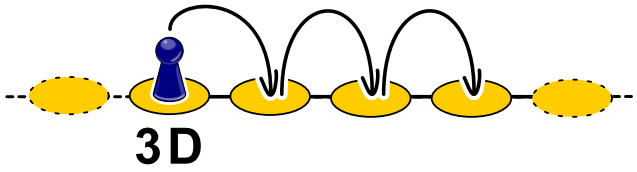
25. Salti

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,

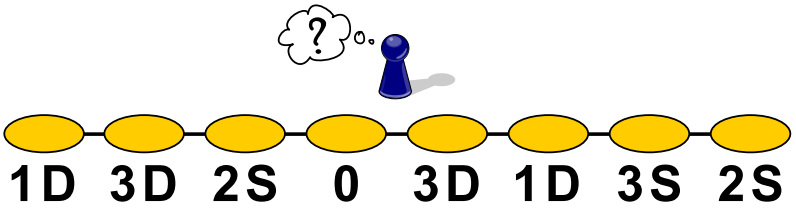


- nD: saltare n volte (posizioni) a destra, quindi 3D significa saltare tre volte a destra,



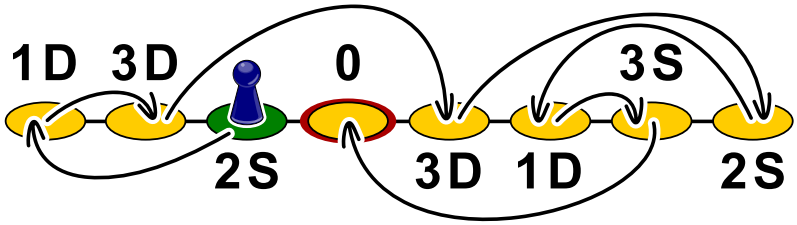
- 0: terminare.

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.

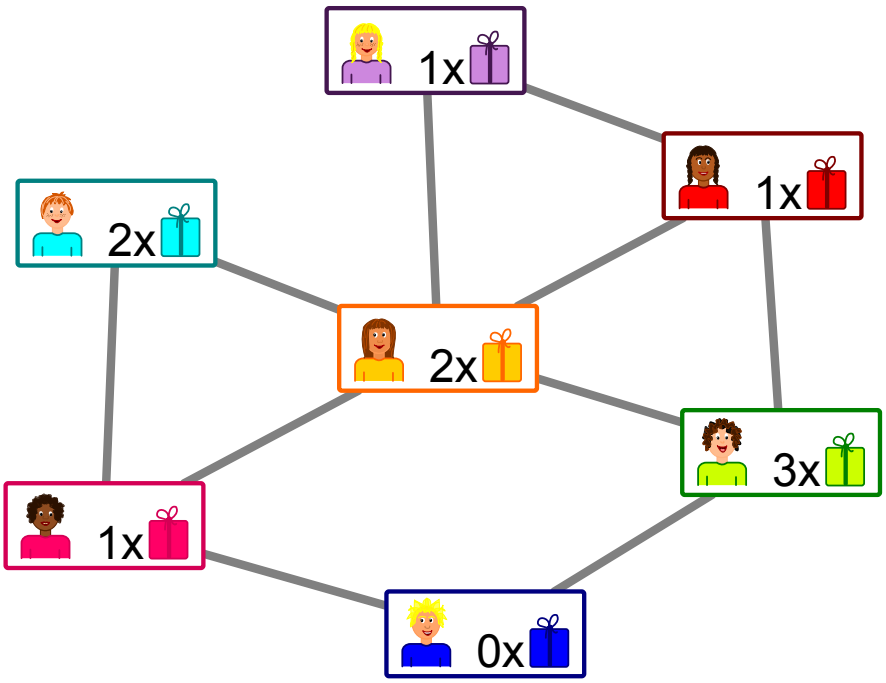
Parole chiave e siti web

lista, gestione della memoria, GOTO

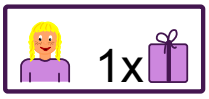
- `https://it.wikipedia.org/wiki/Garbage_collection`
- `https://en.wikipedia.org/wiki/St-connectivity`

26. Regali

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



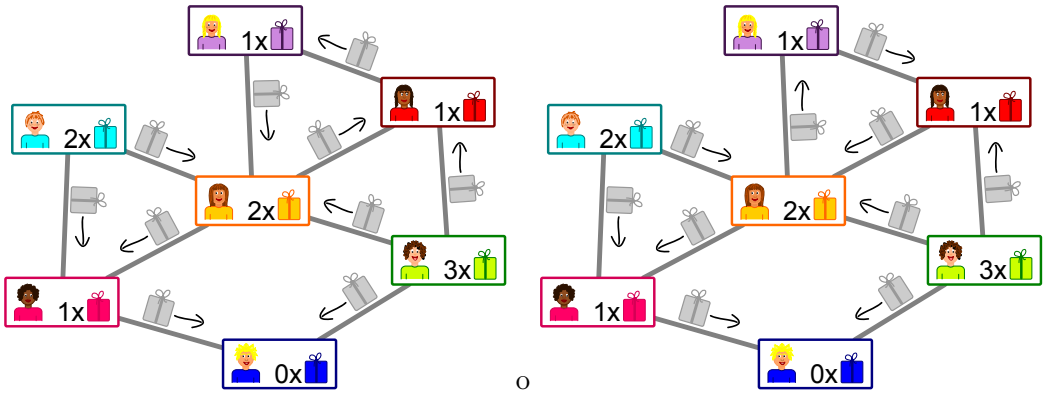
Gli inquilini pianificano una festa con dei regali per i ragazzi: per ogni coppia di amici, uno dei due ragazzi dovrà fare un regalo all'altro.

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

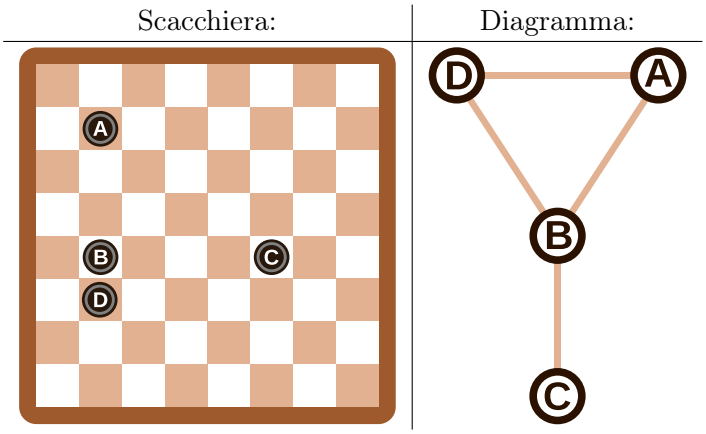
rete di flusso, riduzione di problemi

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

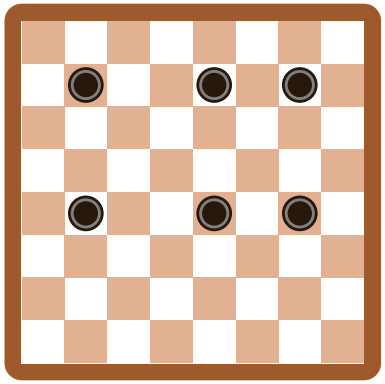
27. Righe e colonne

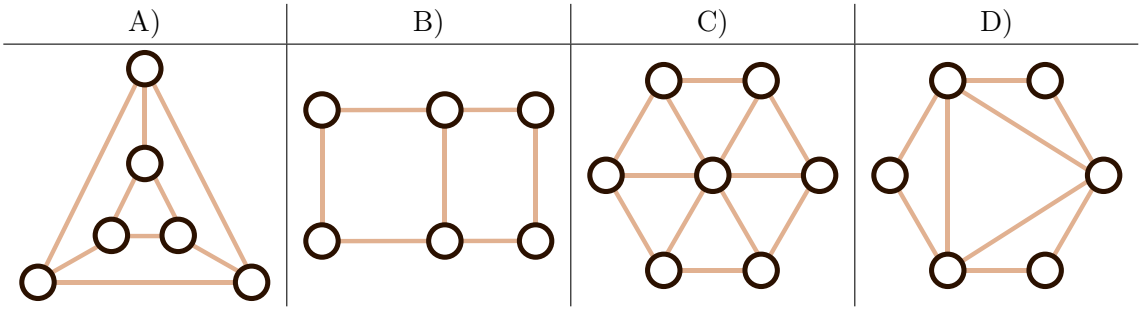
Il diagramma mostrato sulla destra della scacchiera è stato allestito partendo dalla posizione delle pedine in modo che...

- ...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.



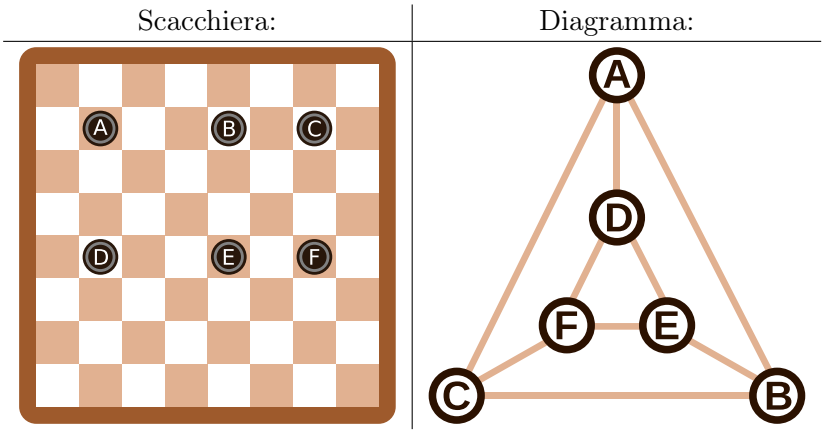
Nel nostro esempio, le pedine sulla scacchiera e i cerchi nel diagramma sono etichettati con una lettera dell'alfabeto, in modo da rendere chiara la spiegazione.
Quale diagramma è associato alla scacchiera seguente, sulla quale sono posizionate 6 pedine?



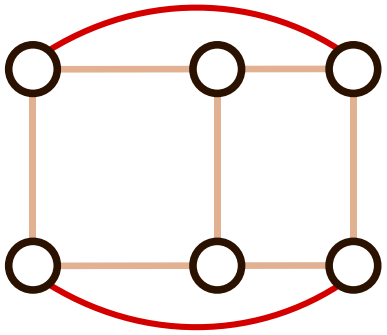


Soluzione

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`

28. 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

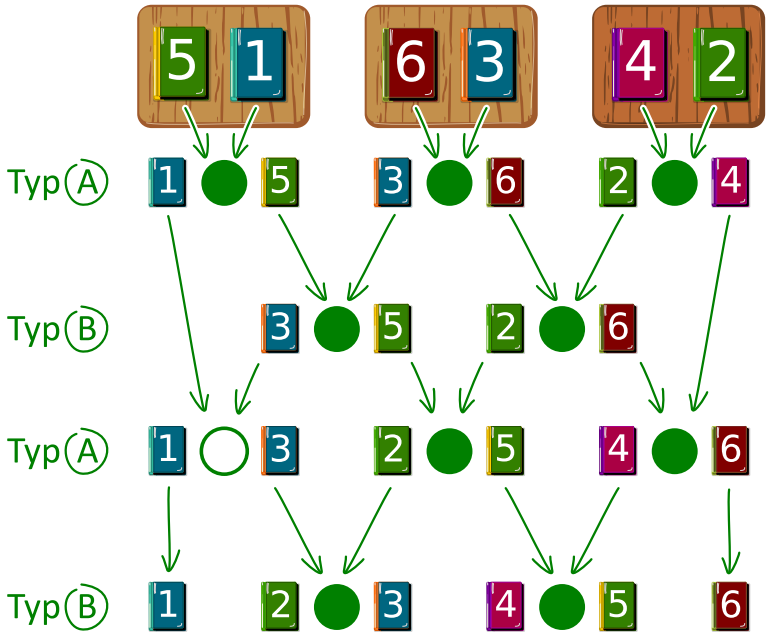
B) quattro turni

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`

29. 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.

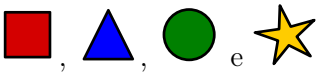
Parole chiave e siti web

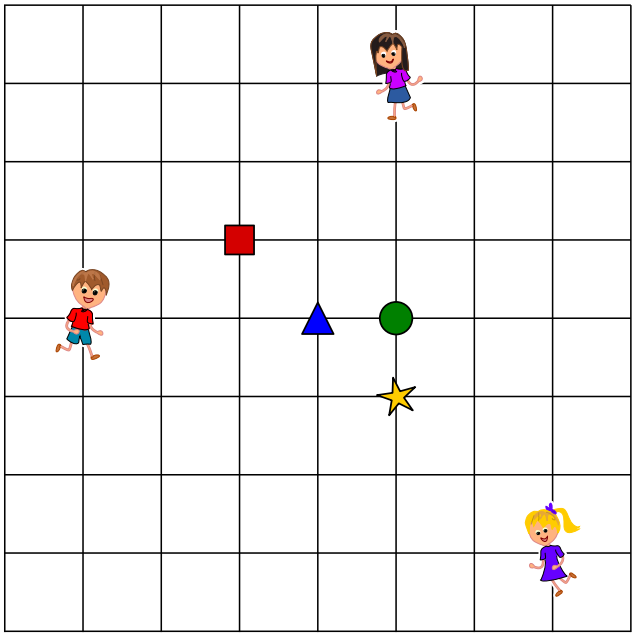
Ricerca fonetica, soundex

- `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`

30. Tre amici

 Alice, Bob  e Séline  vivono a La Chaux-de-Fonds. I tre amici hanno segnato la propria abitazione su una mappa e desiderano concordare un punto di incontro, per cui la somma totale dei tragitti sia la minore possibile. La lunghezza di un singolo tragitto corrisponde al numero di intersezioni tra le diverse linee della mappa che bisogna superare.

 sono i possibili punti d'incontro. Il tragitto più corto per Alice  fino al possibile punto di incontro  ha, ad esempio, lunghezza 4.



Quale punto di incontro è raggiungibile dai tre amici attraverso dei tragitti con somma totale delle lunghezze minore?

A)	B)	C)	D)
			

Soluzione

La risposta corretta è C)  . La somma totale delle lunghezze dei tragitti fino al cerchio verde è: $3 + 4 + 5 = 12$.

 non è corretto. La somma totale delle lunghezze dei tragitti fino al quadrato rosso è infatti: $4 + 3 + 8 = 15$.

 non è corretto. La somma totale delle lunghezze dei tragitti fino al triangolo blu è infatti: $4 + 3 + 6 = 13$.

 non è corretto. La somma totale delle lunghezze dei tragitti fino alla stella gialla è infatti: $4 + 5 + 4 = 13$.

Questa è l'informatica!

Per determinare il migliore dei quattro punti di incontro, la somma delle lunghezze dei tragitti dei tre amici deve essere calcolata per ciascuno di essi. Il luogo di incontro a cui è associata la somma minore è il migliore. Il nostro compito è relativamente semplice e non richiede molto tempo. Se però il numero di amici e il numero di possibili luoghi d'incontro fossero grandi, avremmo un problema di ottimizzazione, la cui soluzione non può essere determinata (in genere) in un tempo "ragionevole".
In informatica, sono però stati sviluppati metodi che trovano delle soluzioni quasi ottimali per questo tipo di problemi in un tempo ragionevole: ad esempio possiamo identificare delle soluzioni che peggiorano solo dell'1% la migliore soluzione possibile. Se invece che in termini di lunghezza ponessimo il nostro problema in termini di tempo e la soluzione migliore fosse 10 minuti, allora una soluzione quasi ottimale, peggiore dell'1%, impiegherebbe solo 6 secondi in più.
Un metodo per trovare soluzioni quasi ottimali è la ricerca locale: per trovare un luogo di incontro quasi perfetto per un gran numero di amici, si parte da una soluzione casuale; da essa si confrontano poi altre soluzioni (somme totali) per luoghi d'incontro vicini, cercando di migliorare il risultato fino ad identificare quello localmente migliore per la "zona" presa in esame. Ciò non garantisce che la soluzione sia la migliore in assoluto, ma la più ottimale in quella zona.
Perché abbiamo scelto La Chaux-de-Fonds per il nostro quesito? La Chaux-de-Fonds, insieme a Le Locle, fa parte del patrimonio UNESCO dal 2009: non solo grazie alla tradizione orologiera, ma anche perché le due città sono state ricostruite "a scacchiara" nel 19º secolo dopo gli incendi che le avevano distrutte.

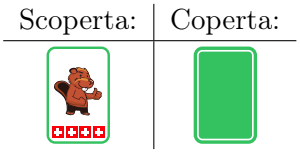
Parole chiave e siti web

problema di ottimizzazione, ricerca locale

- `https://it.wikipedia.org/wiki/Problema_di_ottimizzazione`
- `https://www.okpedia.it/ricerca_locale`
- `https://en.wikipedia.org/wiki/Local_search_(optimization)`
- `https://whc.unesco.org/en/list/1302`

31. Girare le carte

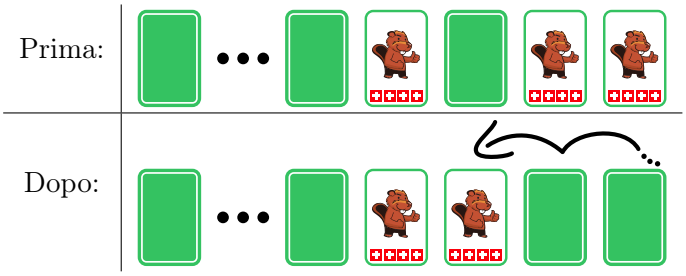
Ti hanno regalato un mazzo di carte uguali. Le carte sono fatte così:



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:

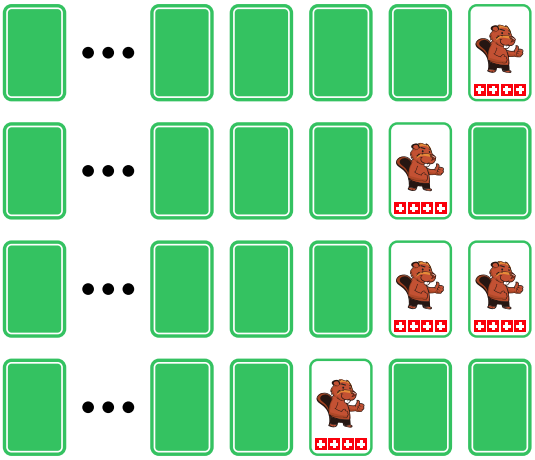
Quante carte sono scoperte dopo 16 sessioni di gioco?

Soluzione

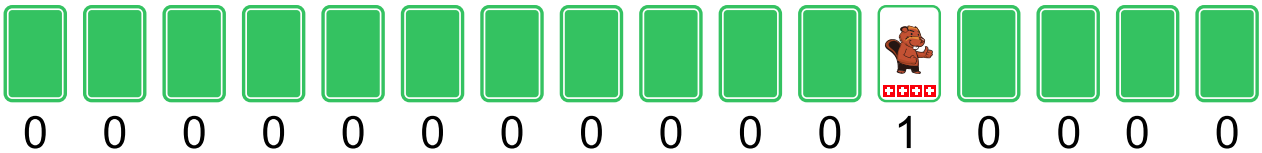
Esattamente una carta è scoperta.
Le carte (scoperte o coperte) possono essere equiparate a dei numeri binari. I numeri binari, infatti, sono composti solo dalle cifre 1 (carta scoperta) e 0 (carta coperta).
Analogamente al sistema decimale, ogni cifra di un numero binario indica se la potenza di 2, 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

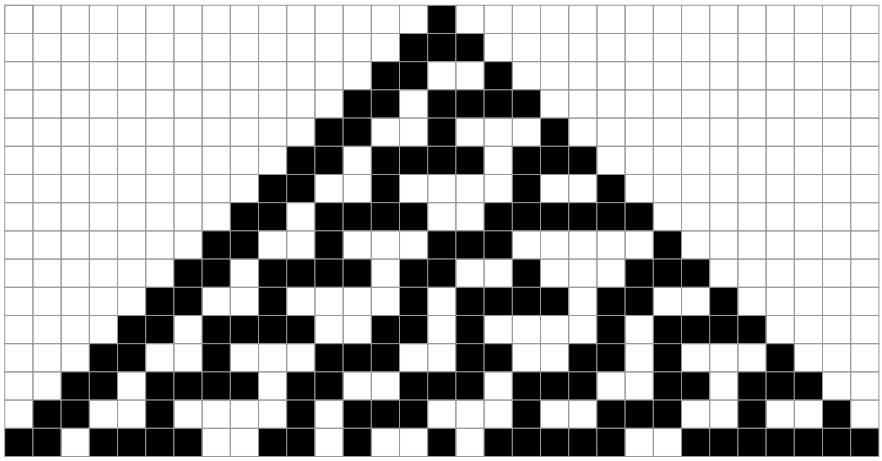
numero binario, codice binario

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

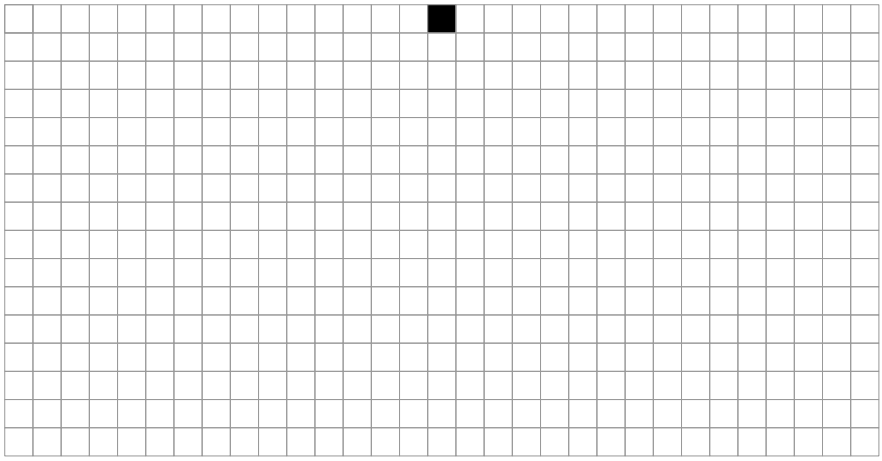
32. Mosaico

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:



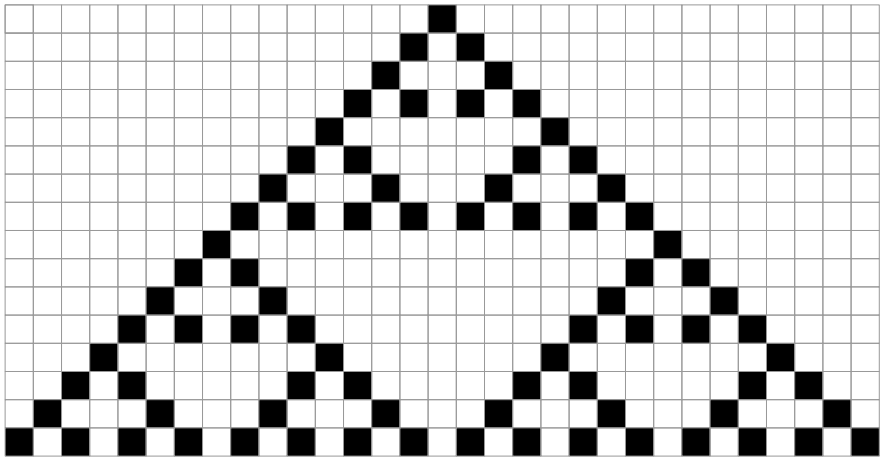
Crea la tua serie di semplici regole, in modo che nell'ultima fila le tessere nere e quelle bianche siano alternate.

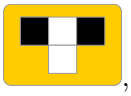


Soluzione

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.

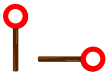
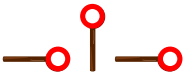
Parole chiave e siti web

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`

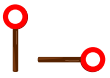
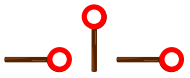
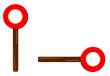
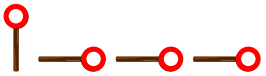
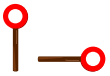
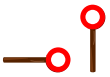
33. Dove è l'aliante?

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
			

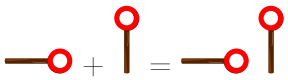
C'è però un problema con questi segnali. Quando ad esempio si danno le indicazioni seguenti

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?

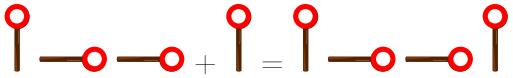
	a sinistra	a destra	verso la collina	verso la valle
A)				
B)				
C)				
D)				

Soluzione

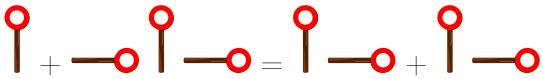
La risposta corretta è C). Si può notare come ogni indicazione inizi con  , seguita poi da 0, 1, 2 o 3  . Il segnale  non viene più utilizzato all'interno dell'indicazione. In questo modo sappiamo che ogni indicazione inizia con  e per interpretarla dobbiamo contare da quanti  è composta.
La risposta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

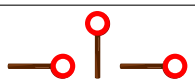
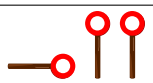


Capire che la risposta A) non è corretta, è un po 'più difficile. L'indicazione "verso la valle" seguita da "a sinistra" corrisponde a due volte "a destra":



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`

34. 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

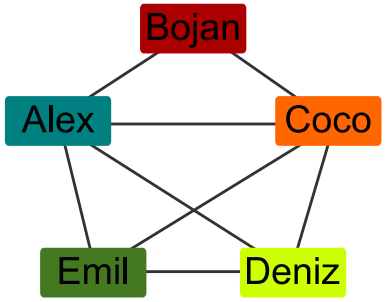
C) Coco

D) Deniz

E) Emil

Soluzione

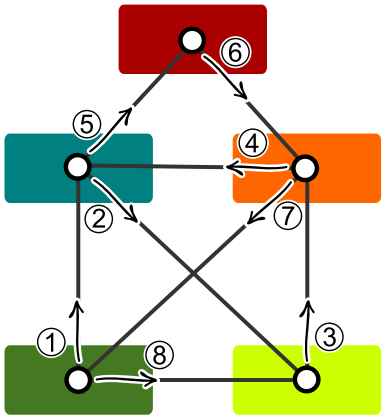
La risposta corretta è B) Bojan.
Nel diagramma seguente ogni rettangolo etichettato con un nome rappresenta un ballerino. Due ballerini sono connessi da una linea se formano una coppia. Un pianificazione valida delle prove è dunque un cammino continuo che attraversi tutte le linee del diagramma esattamente 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.

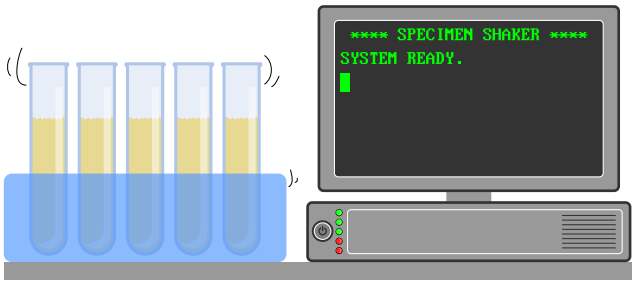
Parole chiave e siti web

grafo, cammino di Eulero (o euleriano)

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

35. 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.

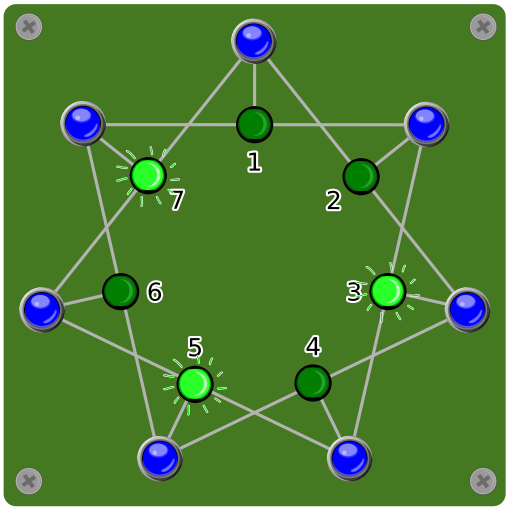
Parole chiave e siti web

struttura di controllo, analisi del programma, `GOTO`

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

36. Accendi la luce!

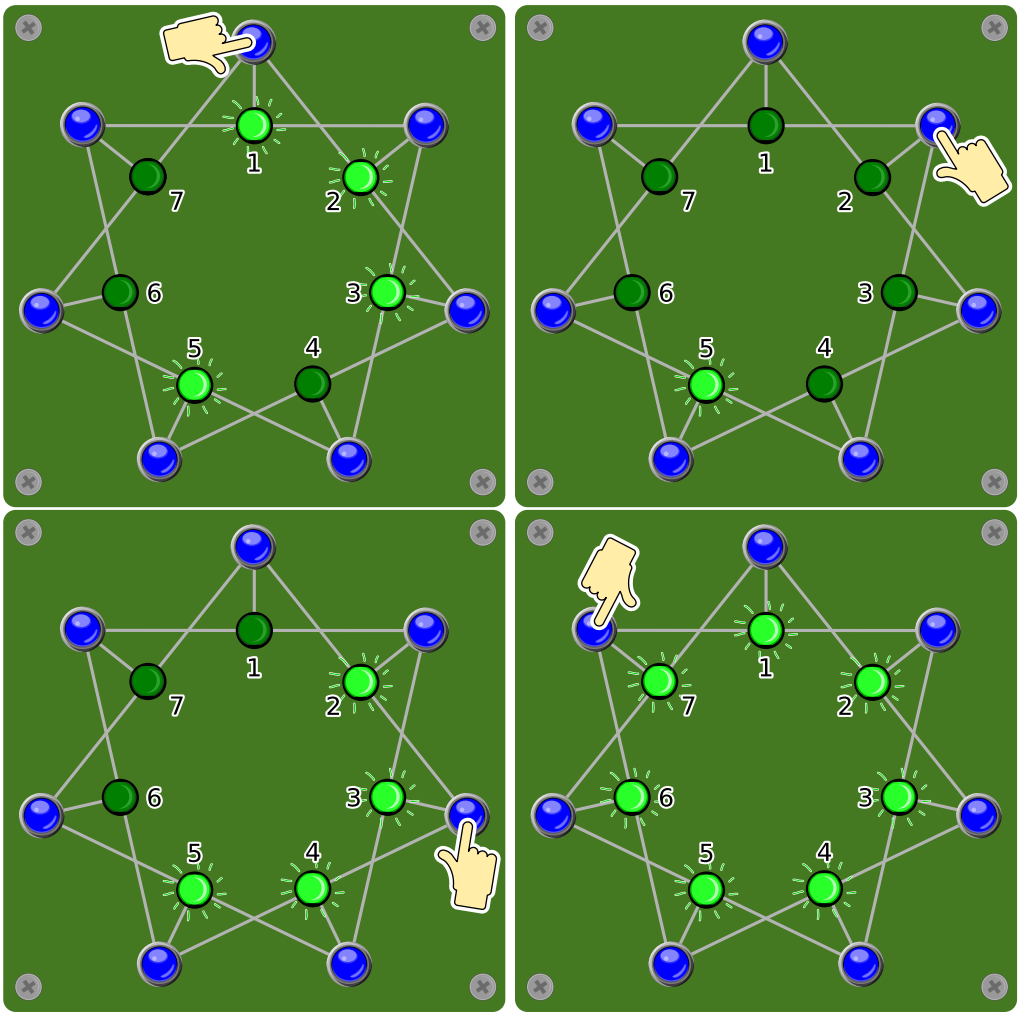
Sette interruttori sono collegati a sette lampade in modo tale che ogni interruttore controlli sempre tre lampade alla volta. Quando viene premuto un interruttore, le lampade sotto il suo controllo che erano spente vengono accese, mentre quelle accese vengono spente.



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

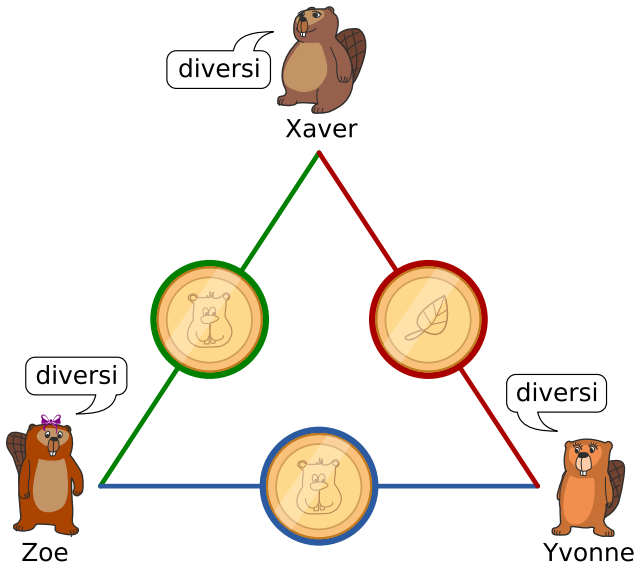
- `https://en.wikipedia.org/wiki/Bidirectional_search`
- `https://it.wikipedia.org/wiki/Automa_a_stati_finiti`

37. Grande segreto

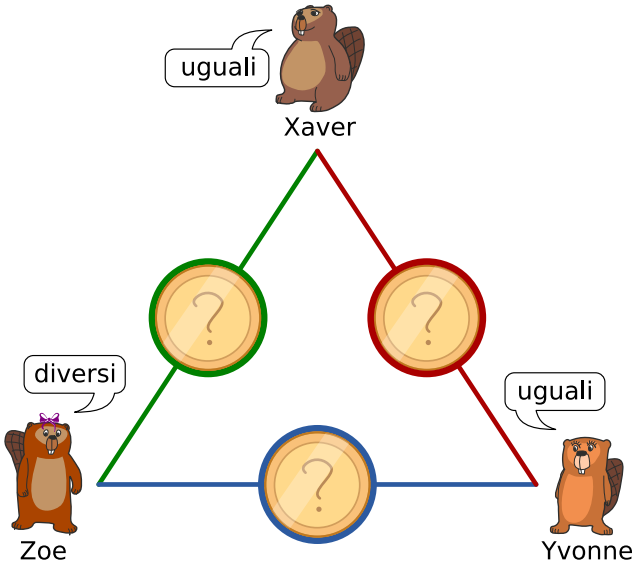
Xaver, Yvonne e Zoe giocano occasionalmente alla lotteria. I media hanno annunciato che il biglietto vincente è stato venduto nella loro città. I tre vorrebbero sapere se uno tra loro è per caso il vincitore, ma d'altro canto il nome esatto dovrebbe rimanere segreto. Ecco quindi come procedono:

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
Carlo Bellettini
Javier Bilbao
Daphne Blokhuis
Laura Briviba
Lucia Budinská
Špela Cerar
William Chan
Kessarapan Charoensueksa
Anton Chukhnov
Kris Coolsaet
Valentina Dagienė
Darija Dasović Rakijašić
Christian Datzko
Susanne Datzko
Dilek Doğan
Marissa Engels
Hanspeter Erni
Veerle Fack
Georgios Fessakis
Gerald Futschek
Ionuț Gorgos
Shuchi Grover
Yasemin Gülbahar
Martin Guggisberg
Bent Halden
Urs Hauser
Wei-fu Hou
Juraj Hromkovič
Takeharu Ishizuka
Svetlana Jakšić
Zhang Jinbao
Emil Kelevedjiev
Dong Yoon Kim
Vaidotas Kinčius
Iryna Kirynovich
Jia-Ling Koh
Regula Lacher
Anh Vinh Le
Dan Lessner
Judith Lin
Violetta Lonati
Nils Mak
Dimitris Mavrovouniotis
Karolína Mayerová
Mattia Monga
Samart Moodleah
Anna Morpurgo
Tom Naughton
Henry Ong
Sanja Pavlovic Šijanović
Péter Piltmann
Zsuzsa Pluhár
Wolfgang Pohl
Ilya Posov
Stavroula Prantsoudi
Nol Premasathian
J.P. Pretti
Doris Reck
Alei Reyes
Chris Roffey
Kirsten Schlüter
Andrea Maria Schmid
Victor Schmidt
Andrea Schrijvers
Eljakim Schrijvers
Vipul Shah
Mohamed El-Sherif
Jacqueline Staub
Allira Storey
Gabrielė Stupurienė
Faisal Al-Sudani
Márta Szabó
Aliaksei Tolstsikau
Peter Tomcsányi
Ahto Truu
Willem van der Vegt
Jiří Vaníček
Troy Vasiga
Rechilda Villame
Eslam Wageed
Pieter Waker
Michael Weigend
Khairul A. Mohamad Zaki
Magdalena Zarach

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.