INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Quesiti e soluzioni 2018
7º e 8º anno scolastico

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

01010011010101100100100101000001001011010101001101010011010010010100010100110101010011010010010100001

SSII

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

Hanno collaborato al Castoro Informatico 2018

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

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

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

INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

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

HASLERSTIFTUNG

Nota: Tutti i link sono stati verificati l'01.11.2018. Questo quaderno è stato creato il 16 novembre 2018 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

Hanno collaborato al Castoro Informatico 2018 i

Premessa ii

1. I fiori di Clara 1

2. Linee del tram 3

3. Pianeta Z 5

4. Gelateria 7

5. Il labirinto delle frecce 9

6. Torre panoramica 11

7. Le bugie hanno le gambe corte 13

8. Le cascate 17

9. Il laghetto dei castori 21

10. Il concorso dei castori 23

11. Casa di vacanza Nr. 29 25

12. Vicini 27

13. Videogioco 31

14. Visitare gli amici 33

15. Due castori al lavoro 37

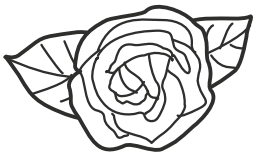
A. Autori dei quesiti 39

B. Sponsoring: concorso 2018 40

C. Ulteriori offerte 42

1. I fiori di Clara

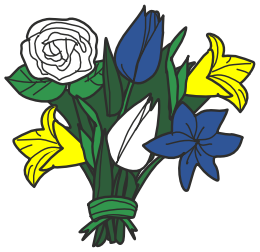
A Clara piacciono i mazzi di fiori colorati e quindi va in un negozio di fiori, dove trova i seguenti tipi:

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.

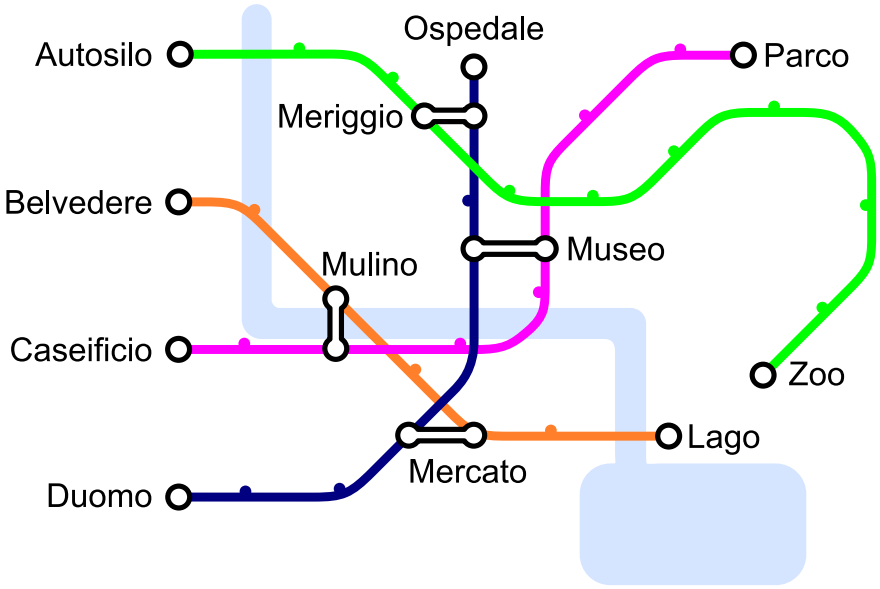
Parole chiave e siti web

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/`

2. Linee del tram

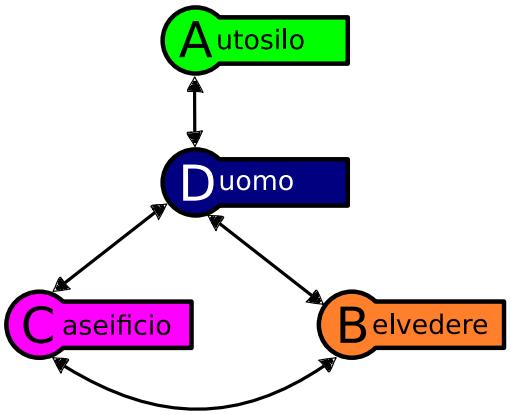
In una città, ci sono quattro linee del tram che partono rispettivamente dai capilinea "Autosilo", "Belvedere", "Caseificio" e "Duomo". Ci sono anche quattro fermate di scambio che permettono di cambiare linea, ossia "Museo", "Mercato", "Mulino" e "Meriggio".



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.

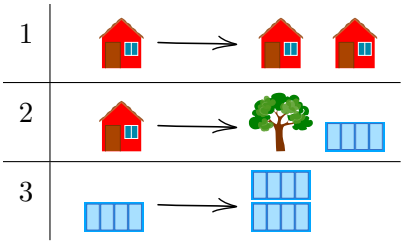
Parole chiave e siti web

Grafi, reti (dei trasporti)

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

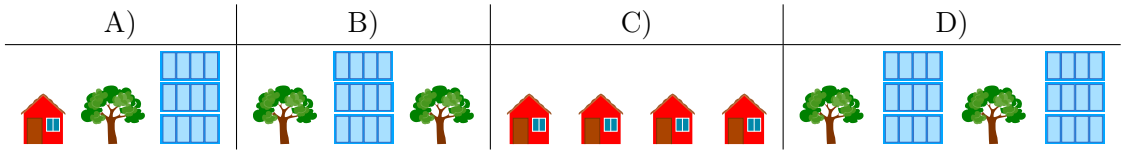
3. Pianeta Z

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!

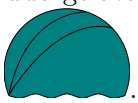
Le regole del nostro quesito sono dette "regole di sostituzione": un simbolo o un oggetto viene sostituito da una sequenza di altri simboli o oggetti. Se viene sostituito un solo simbolo o oggetto alla volta, tale sistema è detto "senza contesto": in altre parole, un simbolo o un oggetto viene sostituito 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

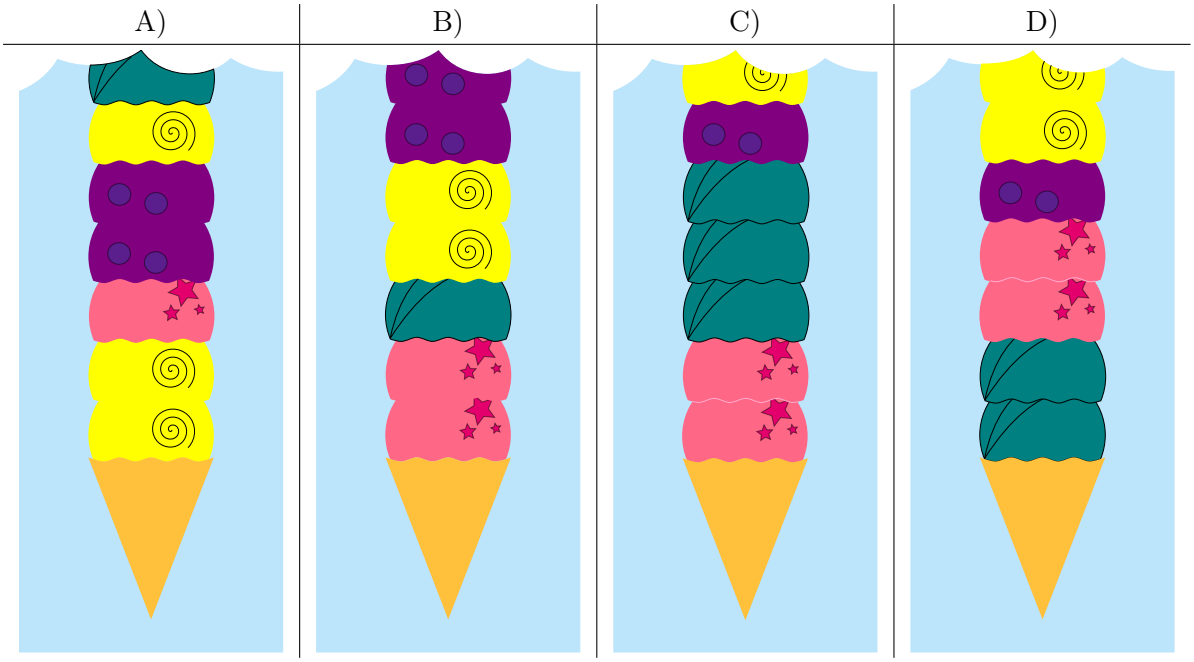
- `https://en.wikipedia.org/wiki/Production_(computer_science)`
- `https://it.wikipedia.org/wiki/Linguaggio_libero_dal_contesto`
- `https://it.wikipedia.org/wiki/Albero_sintattico`

4. Gelateria

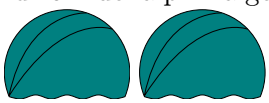
In un villaggio ci sono due gelaterie. Esse offrono le stesse quattro varietà di gelato:  ,  ,  e  .
La prima gelateria utilizza le seguenti istruzioni per un fare i cornetti:

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.

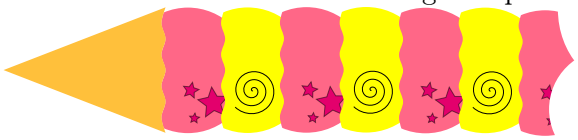
La seconda gelateria, invece, non segue assolutamente nessuna istruzione.
Nelle immagini vengono mostrate solo le parti inferiori di alcuni cornetti. Quale tra questi è stato fatto di sicuro dalla seconda gelateria?



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.

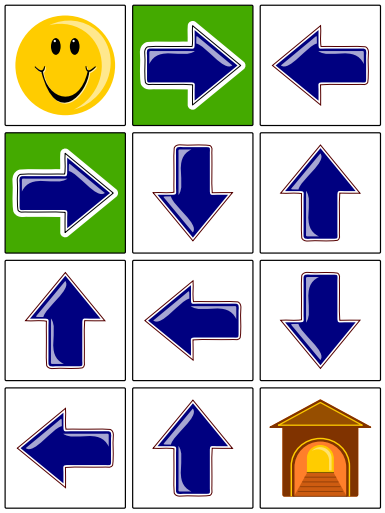
Parole chiave e siti web

pattern recognition ("riconoscimento di modelli o strutture")

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

5. Il labirinto delle frecce

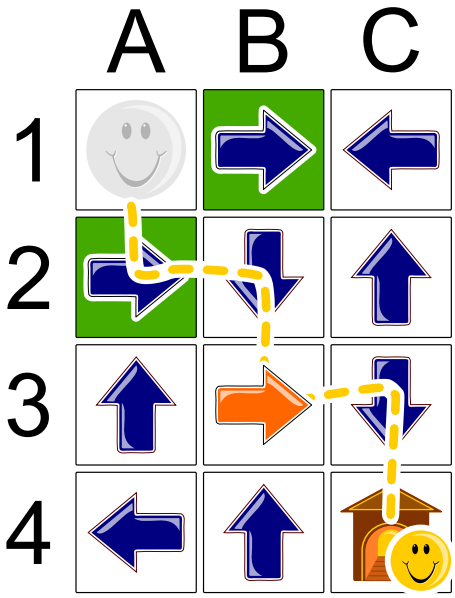
Smiley desidera andare a casa, ma può farlo solo attraversando il labirinto delle frecce. Esso può entrare da uno degli ingressi (caselle verdi). Quando si trova su una casella con una freccia, deve poi spostarsi nella direzione indicata. Con le frecce disposte nel modo seguente è però impossibile che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

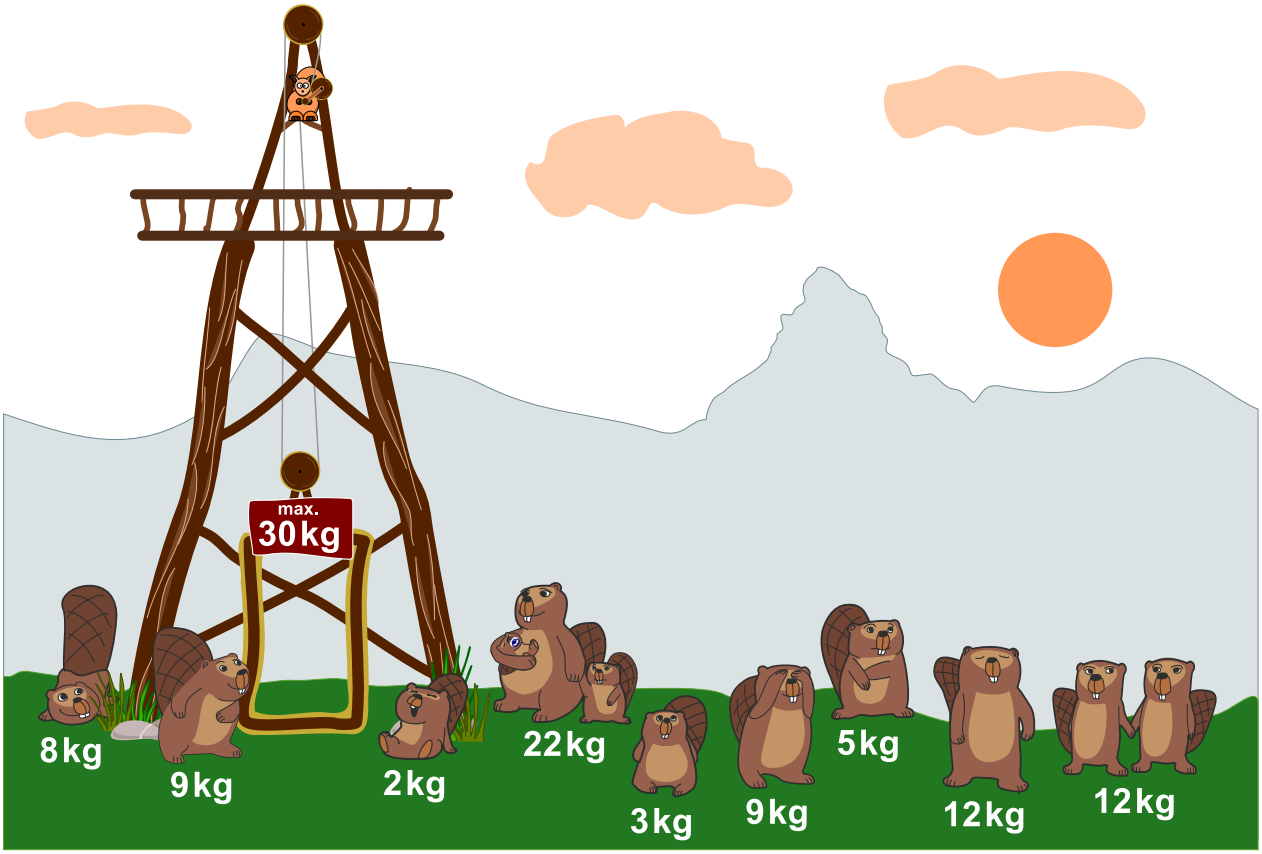
Parole chiave e siti web

labirinto, backtracking ("monitoraggio a ritroso")

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

6. 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.

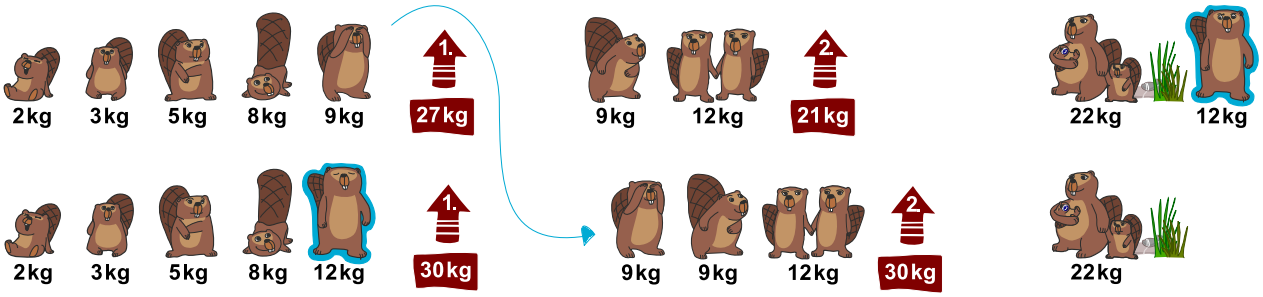
D) La mamma con il neonato e l'altro piccolo e anche i gemelli.

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`

7. 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 → rappresenta l'implicazione logica ("SE" → "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.

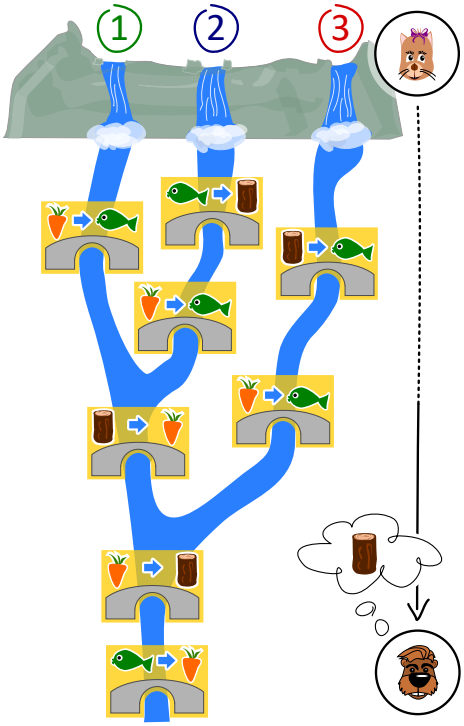
Parole chiave e siti web

logica, logica booleana

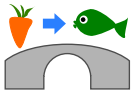
- `https://it.wikipedia.org/wiki/Tertium_non_datur`

- `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`

8. 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?

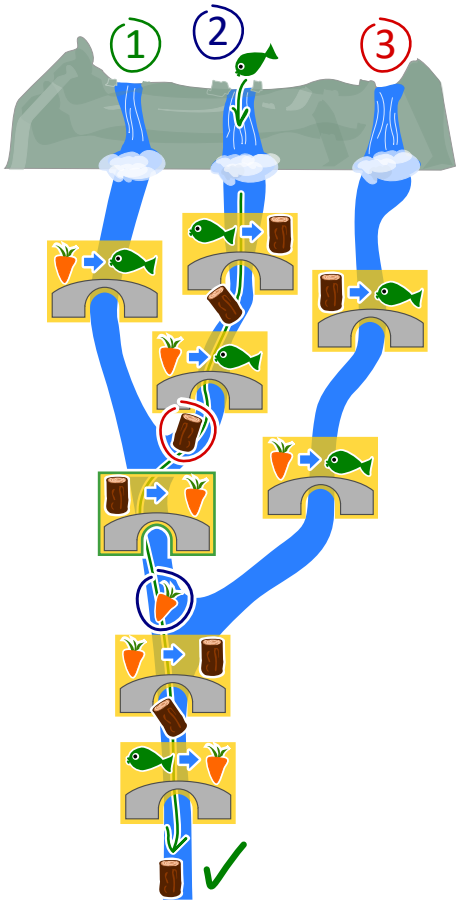
A) Deve far cadere un pesce nella cascata numero 1.

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

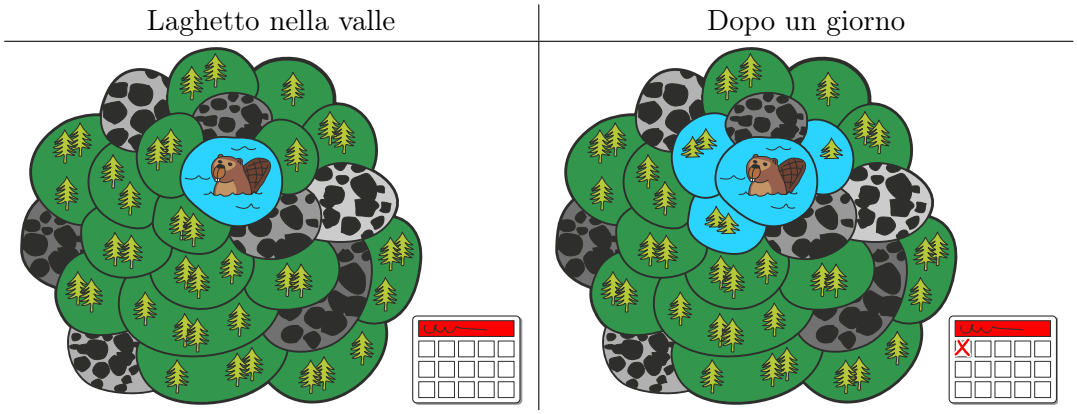
collaudo e test dei software, paradigmi di programmazione, programmazione funzionale, funzioni e parametri

- `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`

9. Il laghetto dei castori

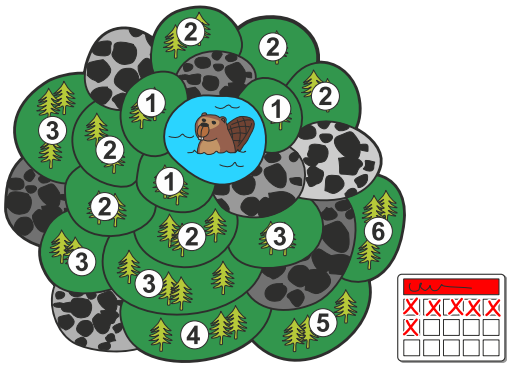
In una valle c'è un piccolo laghetto, circondato da aree boschive o rocciose. Nel laghetto vivono alcuni castori.
Un giorno il laghetto diventa troppo piccolo per i castori e quindi decidono di allagare le aree boschive. Ogni giorno allagano delle nuove aree boschive confinanti con il lago: dopo il primo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

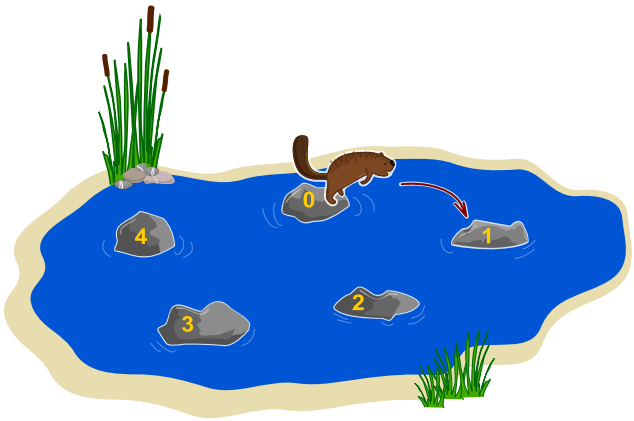
ricerca in ampiezza, algoritmo wavefront ("fronte d'onda")

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

10. 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`

11. 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).

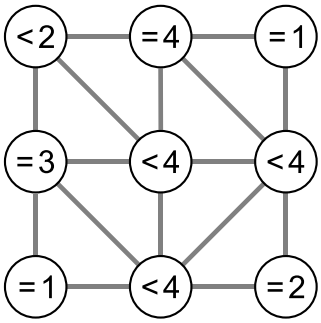
Parole chiave e siti web

Albero di ricerca binario, Albero AVL

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

12. Vicini

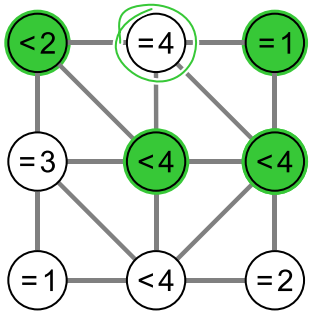
L'immagine qui sotto mostra nove cerchi, in parte connessi. Una connessione (linea) li definisce come "vicini".
Ogni cerchio contiene un'espressione che indica quanti dei suoi vicini devono essere colorati. Per esempio, "$= 3$" significa che esattamente tre dei vicini devono essere colorati; "< 4" significa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



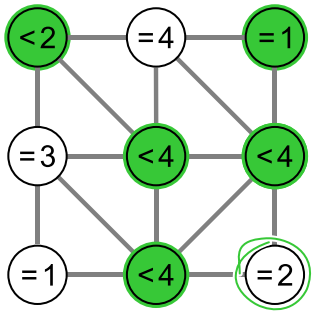
Colora i cerchi in modo che ogni condizione (espressione) sia soddisfatta.

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`

13. 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.

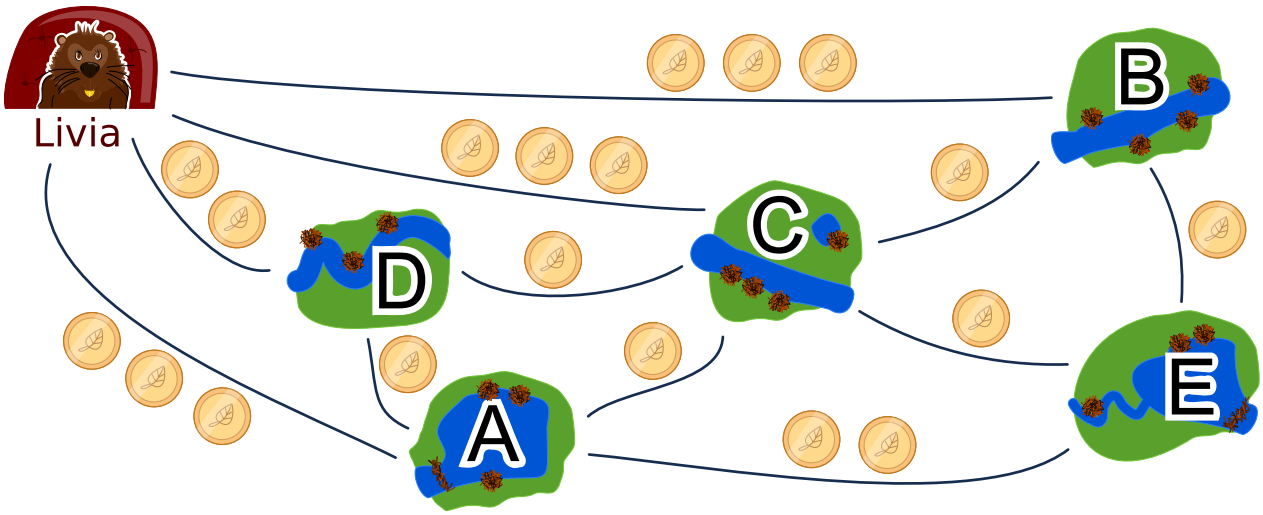
Parole chiave e siti web

analisi, verifica e validazione del software

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

14. 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:

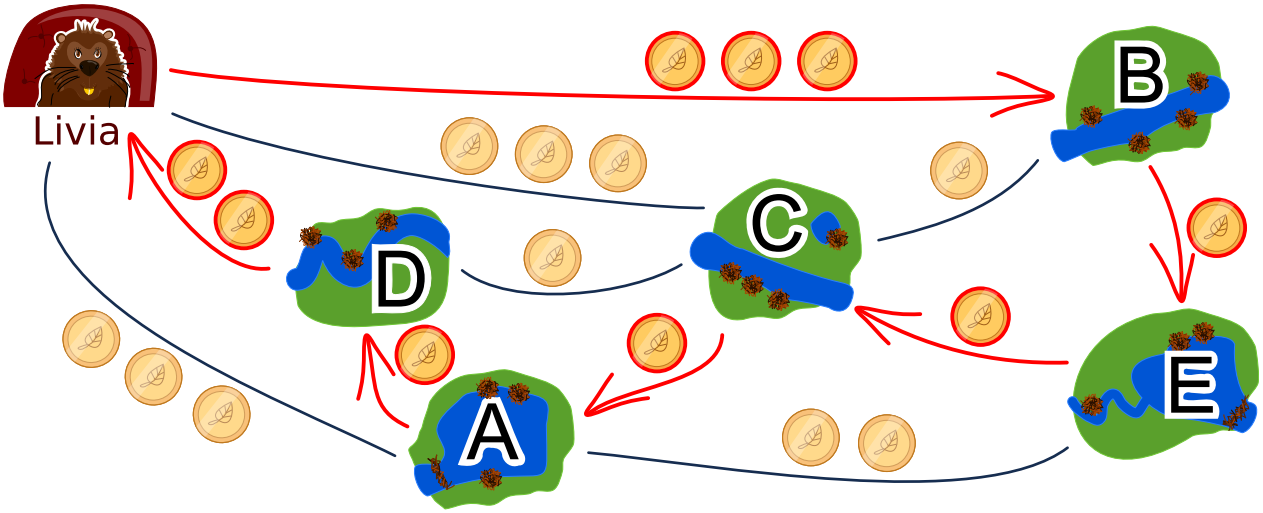
$$\text{Casa} \to \text{B} \to \text{E} \to \text{A} \to \text{D} \to \text{C} \to \text{Casa}.$$

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.

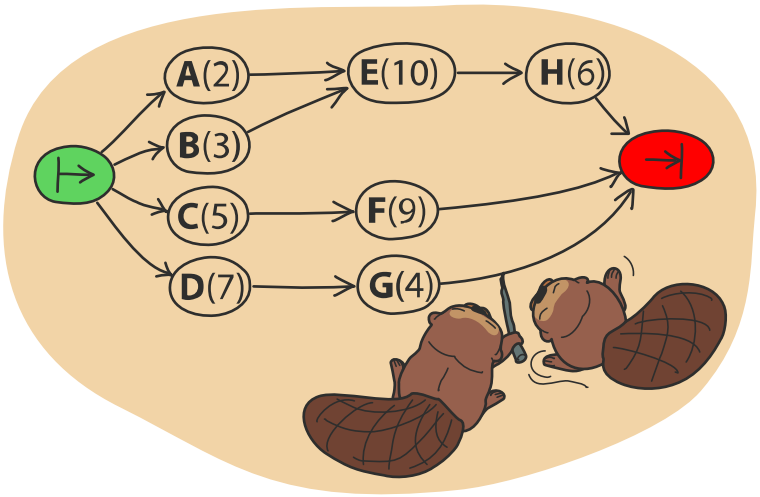
Parole chiave e siti web

ottimizzazione, problema del commesso viaggiatore

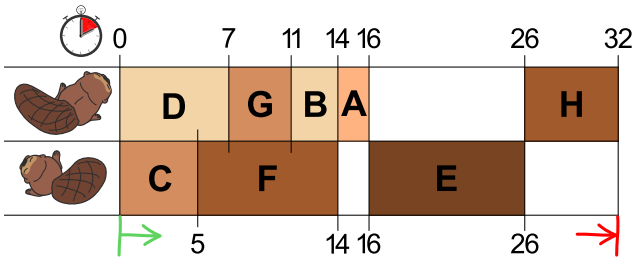
- `https://it.wikipedia.org/wiki/Problema_del_commesso_viaggiatore`
- `https://it.wikipedia.org/wiki/Problema_di_ottimizzazione`

15. 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.



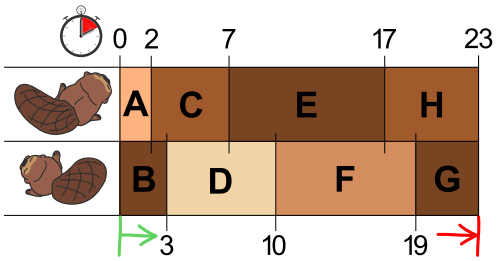
La figura qui sotto mostra un possibile piano di lavoro per i due castori... Si può però di sicuro terminare la diga più velocemente!



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`

A. Autori dei quesiti

Andrea Adamoli
Jared Asuncion
Javier Bilbao
Lucia Budinská
Špela Cerar
Kris Coolsaet
Valentina Dagienė
Darija Dasović Rakijašić
Christian Datzko
Susanne Datzko
Marissa Engels
Hanspeter Erni
Georgios Fessakis
Gerald Futschek
Martin Guggisberg
Bent Halden
Urs Hauser
Wei-fu Hou
Juraj Hromkovič
Takeharu Ishizuka
Svetlana Jakšić
Dong Yoon Kim
Vaidotas Kinčius
Jia-Ling Koh
Regula Lacher
Dan Lessner
Dimitris Mavrovouniotis
Karolína Mayerová
Samart Moodleah
Tom Naughton
Sanja Pavlovic Šijanović
Péter Piltmann
Zsuzsa Pluhár
Wolfgang Pohl
Ilya Posov
Nol Premasathian
J.P. Pretti
Doris Reck
Kirsten Schlüter
Andrea Maria Schmid
Mohamed El-Sherif
Jacqueline Staub
Allira Storey
Peter Tomcsányi
Willem van der Vegt
Jiří Vaníček
Troy Vasiga
Michael Weigend
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.