INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Quesiti e soluzioni 2020

3º e 4º anno scolastico

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

A cura di:

Lucio Negrini, Christian Giang, Susanne Datzko, Fabian Frei, Juraj Hromkovič, Regula Lacher, Jean-Philippe Pellet

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

Hanno collaborato al Castoro Informatico 2020

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

Capo progetto: Nora A. Escherle

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

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

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

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

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

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

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

INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

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

HASLERSTIFTUNG

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

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

Premessa

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

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

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

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

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

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

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

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

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

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

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

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

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

Per ulteriori informazioni:

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

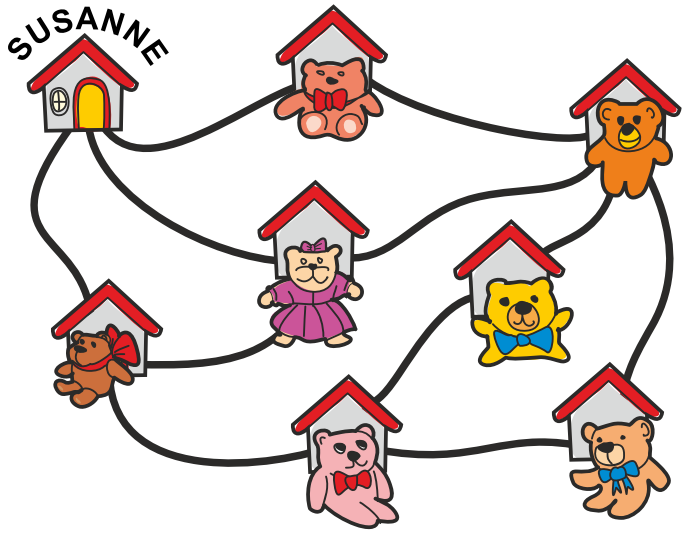
`https://www.castoro-informatico.ch/it/kontaktieren/`
`https://www.castoro-informatico.ch/`

Indice

Hanno collaborato al Castoro Informatico 2020 i
Premessa iii
Indice v
1. Caccia agli orsacchiotti 1
2. Spettacolo teatrale 5
3. Annaffiare i fiori 9
4. Anno di costruzione del castello 13
5. 3×3 sudoku con gli alberi 15
6. Visita al museo 19
7. Castoro al castello 23
8. Prossima fermata, stazione! 27
9. Tronchi e pile 29
A. Autori dei quesiti 32
B. Sponsoring: concorso 2020 33
C. Ulteriori offerte 35

1. Caccia agli orsacchiotti

Nel quartiere di Susanne si trovano i seguenti orsacchiotti di peluche davanti alle case.



Da casa sua, Susanne ha fatto una passeggiata passando esattamente davanti ad altre quattro case. Non è mai passata due volte su un percorso che collega una casa ad un'altra. In una casa le è sfuggito l'orsacchiotto. Gli altri tre orsacchiotti che ha visto erano:

Quale orsacchiotto è sfuggito a Susanne?

A) 

B) 

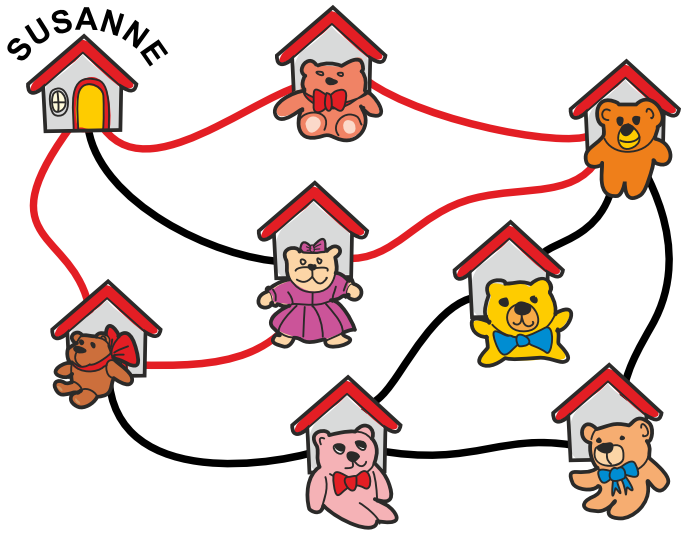
C) 

D) 

Soluzione

La risposta corretta è C).

Durante il suo giro Susanne deve essere passata davanti alle case con i tre orsacchiotti , e Questi tre orsacchiotti sono direttamente collegati da un sentiero. Al primo orsacchiotto Susanne ci arriva direttamente da casa sua. Alla fine di questo percorso si trova il terzo orsacchiotto . Da lì c'è solo una strada per tornare a casa sua, che è la strada per l'orsacchiotto . Altri modi possibili passano almeno da altri due orsacchiotti. Ma Susanne è passata davanti a solo quattro case. La seguente mappa mostra la strada fatta da Susanne:



Susanne può fare il giro in entrambe le direzioni possibili.

Questa è l'informatica!

Cloze in lezioni di lingua, problemi di matematica con campi vuoti o una caccia all'orsacchiotto con un'immagine mancante: sono tutti compiti in cui si cercano informazioni mancanti. Tuttavia, i compiti sono impostati (o *strutturati*) in modo tale che queste informazioni mancanti possano essere trovate con il *pensiero logico* o il *ragionamento*.

Nell'informatica, queste situazioni si trovano spesso. Possono per esempio presentarsi durante il trasferimento dei dati o durante il salvataggio dei dati. Pertanto, le procedure vengono utilizzate per *rilevare gli errori* o anche per *correggerli*. Se l'errore non è troppo grande, questo si ottiene salvando deliberatamente più informazioni di quelle effettivamente necessarie. In questo compito si tratta della mappa e del fatto che Susanne ha superato esattamente quattro orsacchiotti. Così si possono trovare le informazioni mancanti, cioè quale orsacchiotto le è sfuggito.

A proposito, in alcuni paesi del mondo nel 2020 sono state offerte delle «cacce all'orsacchiotto» (in inglese «teddy bear hunts»), dove gli orsacchiotti erano nascosti in diverse case, e potevano essere visti dall'esterno. Questo ha permesso ai bambini di divertirsi a nascondere e cercare insieme gli

orsacchiotti, nonostante le regole della distanza dovute al Coronavirus. L'idea di rievocare una caccia all'orso è nata dal libro illustrato «We're Going on a Bear Hunt» di Michael Rosen (1989).

Parole chiave e siti web

- Rilevazione e correzione d'errore:
  `https://it.wikipedia.org/wiki/Rilevazione_e_correzione_d'errore`
- `https://www.insider.com/coronavirus-pandemic-sparked-worldwide-bear-hunt-to-entertain-kids-2020-4`
- `https://www.youtube.com/watch?v=0gyI6ykDwds`
- `https://www.tio.ch/rubriche/ti-mamme/1431131/bambini-caccia-mondo-idea-ti-mamme`

2. Spettacolo teatrale

Uno spettacolo teatrale presenta una saggia principessa, un nobile cavaliere, un re bello e un drago malvagio. All'inizio il palco è vuoto. Durante lo spettacolo queste quattro figure entrano ed escono dal palco nel seguente ordine:

Primo atto		**Atto secondo**
Il re entra in scena		Il drago entra in scena
La principessa entra in scena		Il cavaliere entra in scena
Il re esce di scena	PAUSA	Il drago esce di scena
Il drago entra in scena		La principessa entra in scena
La principessa esce di scena		Il cavaliere esce di scena
Il drago esce di scena		La principessa esce di scena
Pausa		**Fine**

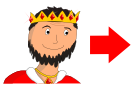
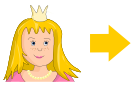
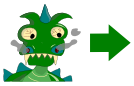
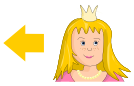
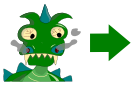
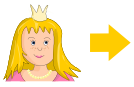
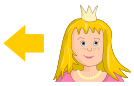
Cosa non succederà?

A) La principessa e il cavaliere sono sul palco insieme.
B) Il re e il drago sono sul palco insieme.
C) Il cavaliere entra in scena dopo la pausa.
D) Il cavaliere e il drago sono sul palco insieme.

Soluzione

La risposta corretta è B) «Il re e il drago sono sul palco insieme», perché questa affermazione non è mai vera durante tutto lo spettacolo.

Si può analizzare passo per passo:

Azione	Re sul palco?	Principessa sul palco?	Drago sul palco?	Cavaliere sul palco?	Risposte corrispondenti
Atto primo					
	Sì	No	No	No	
	Sì	Sì	No	No	
	No	Sì	No	No	
	No	Sì	Sì	No	
	No	No	Sì	No	
	No	No	No	No	
Pausa					
Atto secondo					
	No	No	Sì	No	
	No	No	Sì	Sì	C), D)
	No	No	No	Sì	
	No	Sì	No	Sì	A)
	No	Sì	No	No	
	No	No	No	No	
Fine					

Per ogni risposta è possibile verificare se l'affermazione in essa contenuta è vera o meno, esaminando le righe delle tabelle.

Nella risposta A), si cerca una riga in cui sia la principessa che il cavaliere sono sul palco insieme. Questo è il caso della quarta riga del secondo atto, perché la principessa entra nel palco dove il cavaliere si trova dalla seconda riga e rimane fino alla quinta riga. L'affermazione della risposta A) è quindi corretta almeno in un certo momento.

Nella risposta D), si cerca una riga in cui il cavaliere e il drago sono sul palco insieme. Questo è il caso della seconda riga del secondo atto, perché il cavaliere entra sul palco nella seconda riga, mentre il drago è già entrato sul palco nella prima riga e rimane fino alla terza riga. L'affermazione della risposta D) è quindi corretta almeno in un certo momento.

Nella risposta C), l'affermazione è di natura diversa. Se questo è vero, il cavaliere non deve essere entrato in scena durante tutto il primo atto. Qui bisogna guardare la colonna del cavaliere per il primo atto. C'è scritto «No» ovunque, quindi il cavaliere in realtà non è entrato in scena durante tutto il primo atto. Ma poi entra nella seconda riga del secondo atto, quindi anche l'affermazione della risposta C) è vera.

Se l'affermazione della risposta B) fosse vera, il re e il drago dovrebbero stare insieme sul palco in qualche riga. Ma in nessuna delle dodici righe c'è un «Sì» in entrambe le colonne. Il re esce di scena già nella terza riga del primo atto e non entra più in scena fino alla fine. Il drago, invece, non entra in scena fino alla quarta riga del primo atto. Forse i due si incontrano dietro il palco, ma sul palco non sono mai insieme. Pertanto l'affermazione della risposta B) non è corretta. Quindi B) è la risposta corretta.

Questa è l'informatica!

Anche se si può immaginare vividamente una storia basata sul corso dello spettacolo, solo una caratteristica è importante per ogni personaggio in questo compito: è o non è sul palco in un certo momento? Questa limitazione della visione a certe caratteristiche è chiamata *astrazione*.

Nell' informatica, tali astrazioni possono essere formulate molto bene. Per ognuna delle quattro figure si definisce una cosiddetta *variabile*, che risponde alla domanda se la figura è attualmente in scena. Le quattro variabili sono: «Re sul palco?», «Principessa sul palco?», «Drago sul palco?» e «Cavaliere sul palco?». Durante lo spettacolo, le risposte a queste domande cambiano continuamente; per ogni domanda la risposta è a volte «sì» e a volte «no». Nell' informatica, chiamiamo la risposta attuale ad una domanda il *valore* attuale della variabile associata. Il valore di una variabile può quindi cambiare continuamente (in matematica è diverso, le variabili non cambiano i loro valori nel tempo). La tabella nella spiegazione della risposta mostra le quattro variabili e i valori corrispondenti in qualsiasi momento.

Esiste anche un altro modo di considerare lo spettacolo. In ogni momento guardiamo quali personaggi sono sul palco in quel momento (quindi guardiamo i valori attuali delle quattro variabili.) Ogni possibile combinazione di figure la chiamiamo uno *stato* del palco. Così, quando una figura entra o esce dal palco, lo stato del palco cambia. Possiamo chiamarla quindi una *transizione* del palco da uno stato all'altro. Se si prende un pezzo di carta e si disegna un cerchio separato per ogni possibile stato (cioè ogni combinazione di figure), si può considerare il tutto come un'astrazione del palco.

Inoltre, è possibile disegnare le possibili transizioni come frecce che portano da uno stato all'altro. Se facciamo anche questo, abbiamo quello che in informatica chiamiamo *diagramma di stato* del palco.

All'inizio dello spettacolo il palco è vuoto. Per questo motivo chiamiamo lo stato corrispondente *stato iniziale*. Ora possiamo disegnare il percorso del gioco come un percorso nel diagramma di stato. Il percorso inizia nello stato iniziale e poi segue le frecce che corrispondono all'azione.

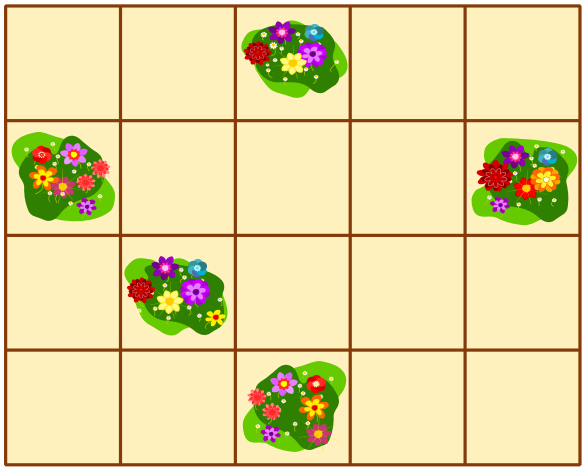
I diagrammi di stato sono molto importanti nell'informatica. Con quasi tutti i sistemi complicati, ad un certo punto bisogna pensare al diagramma di stato. Ma per gli esseri umani è spesso molto noioso lavorare con questi stati e transizioni astratti. I computer, invece, sono estremamente bravi per questo. Pertanto, vale la pena se le persone possono rappresentare i loro problemi con i diagrammi di stato in modo tale che i computer possano poi risolverli.

Parole chiave e siti web

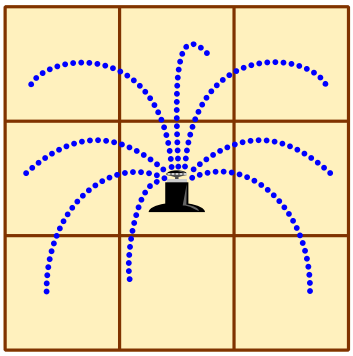
- Variabili: `https://it.wikipedia.org/wiki/Variabile_(informatica)`
- Diagramma di stato: `https://it.wikipedia.org/wiki/Diagramma_di_stato_(informatica)`

3. Annaffiare i fiori

Il giardino di Daniel è costituito da campi quadrati. In alcuni di questi campi ha piantato fiori:



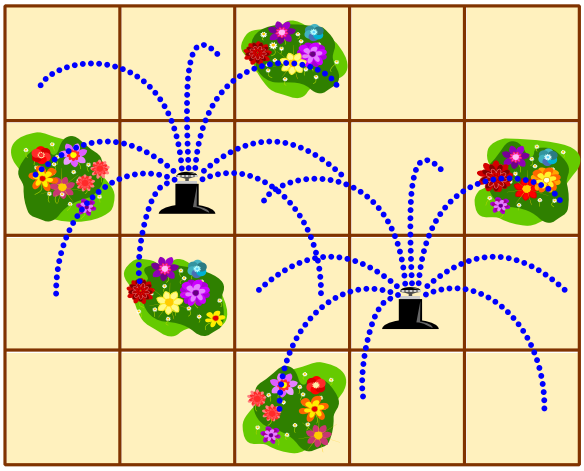
In estate vuole annaffiare i fiori con l'irrigatore a prato. Non può mettere un irrigatore nei campi con i fiori. Un irrigatore annaffia tutti i fiori negli 8 campi intorno a lui.:



Posiziona il minor numero possibile di irrigatori per annaffiare tutti i campi di fiori. Inseriscili nei campi del giardino di Daniel

Soluzione

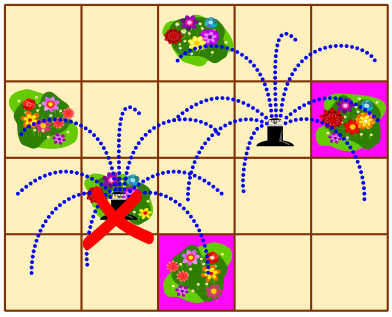
La seguente soluzione richiede due irrigatori per annaffiare tutti i campi di fiori:



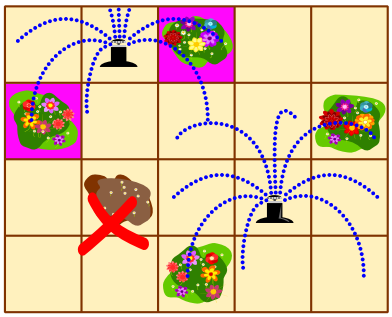
Tra il campo di fiori all'estrema sinistra e il campo di fiori all'estrema destra ci sono tre campi. Un solo irrigatore non può annaffiare campi così distanti tra loro.

Non c'è nemmeno un'altra soluzione con due soli irrigatori.

Per annaffiare il campo di fiori all'estrema destra e quello in basso al centro allo stesso tempo, un irrigatore deve essere posizionato esattamente dove si trova nella soluzione. Se si dovesse alzarne uno più in alto per annaffiare il campo di fiori in alto al centro, non si potrebbe più annaffiare il campo di fiori in basso al centro e non si potrebbero annaffiare i rimanenti tre campi di fiori con un irrigatore, perché nessun irrigatore può essere posizionato su un campo di fiori.



Per annaffiare il campo di fiori all'estrema sinistra e quello in alto al centro, si dovrebbe posizionare un irrigatore come mostrato nella soluzione o un campo sopra di esso. Ma se questo irrigatore deve annaffiare anche il campo di fiori nella seconda colonna da sinistra e quello nella terza fila dall'alto, non può essere posizionato in alto.



Questa è l'informatica!

Questo compito è un tipico problema di ottimizzazione: mentre è chiaro che tutti i campi di fiori devono essere annaffiati, il numero di irrigatori da prato necessari è variabile e deve essere il più piccolo possibile. Problemi di ottimizzazione simili sorgono quando, ad esempio, si vogliono rendere sicuri i villaggi con le caserme dei pompieri o fornire ai piazzali la ricezione dei cellulari.

In informatica si parla anche di *problemi di copertura*. Questi appartengono a una classe di problemi molto difficili in informatica. Il corretto posizionamento di un numero minimo di irrigatori era ancora abbastanza semplice. Ma la difficoltà aumenta talmente tanto con il numero di campi di fiori che presto non è possibile trovare una soluzione ottimale in un tempo ragionevole, anche con l'assistenza informatica.

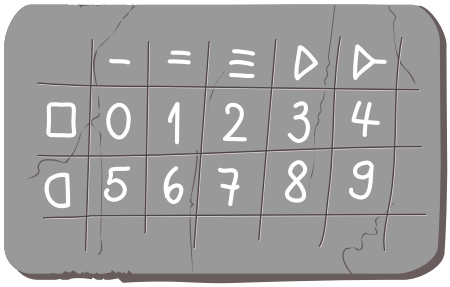
Una possibilità in questi casi è quindi quella di accontentarsi di soluzioni che possono non essere ottimali, ma che sono comunque buone. Non fa molta differenza se si posizionano 101 invece di appena 100 caserme dei pompieri o 1000 antenne di trasmettitori per cellulari invece di appena 990, ma il problema è spesso molto più facile da risolvere.

Parole chiave e siti web

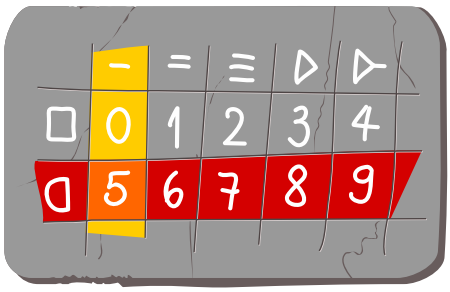
- Problema di ottimizzazione:
  `https://it.wikipedia.org/wiki/Problema_di_ottimizzazione`
- Problema di copertura:
  `https://it.wikipedia.org/wiki/Problema_di_copertura_dei_vertici`

4. Anno di costruzione del castello

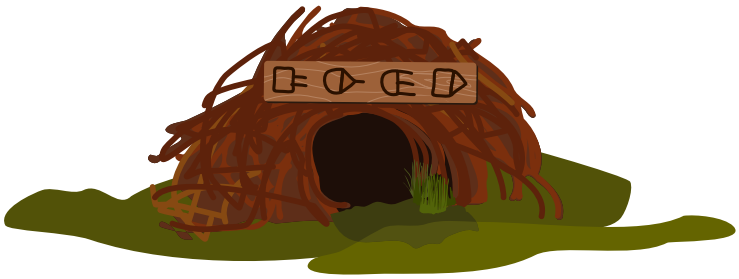
Sul cartello sopra l'ingresso di ogni castello dei castori è indicato l'anno di costruzione. I castori usano i loro caratteri per i numeri. La seguente tabella mostra come le cifre possono essere utilizzate per comporre i caratteri dei castori:



Ad esempio, i castori utilizzano la cifra «5» per formare il nuovo carattere  , che è assemblato nel seguente modo:



Questo è il castello di Cleveria:

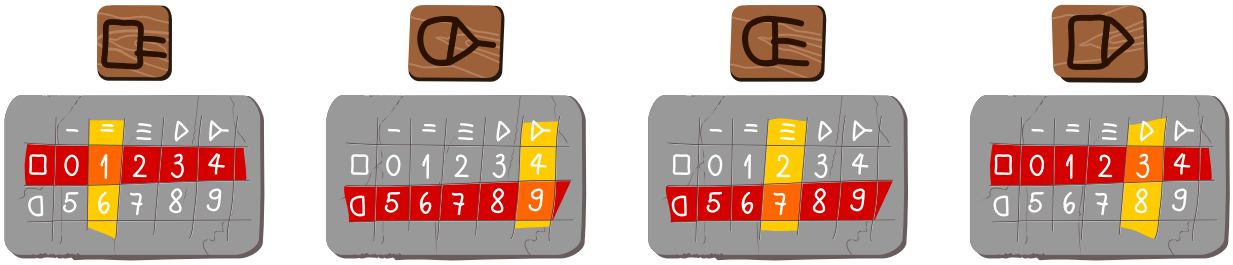


In quale anno è stato costruito il castello di Cleveria?

A) 0978
B) 1574
C) 1923
D) 1973
E) 1993
F) 2973
G) 6378

Soluzione

È possibile scoprire l'anno di costruzione del castello scegliendo la riga e la colonna appropriata per ogni simbolo. All'intersezione tra la colonna e la riga troverete il numero che cercate.



Le quattro cifre nell'ordine corretto danno il numero 1973.

Questa è l'informatica!

Mantenere la segretezza delle informazioni e proteggere i dati è un compito che risale a 4000 anni fa. A questo scopo sono stati sviluppati e utilizzati innumerevoli linguaggi segreti. Oggi la sicurezza dei dati è uno dei temi centrali dell'informatica. Uno dei metodi per proteggere i dati da letture non autorizzate è la *crittografia*. La cifratura trasforma un testo in chiaro in un testo cifrato. Ricostruire il testo in chiaro dal *testo cifrato* si chiama *decifrare*. La scienza del testo cifrato si chiama *crittologia*.

Le culture antiche utilizzavano per lo più scritture segrete, che venivano create codificando le lettere con altre lettere o con caratteri completamente nuovi. Il cifrario seguente è stato sviluppato specialmente per la competizione del castoro informatico, ma si basa su un concetto dell'antica Palestina. La regola di sicurezza dell'epoca era che si usavano solo codici segreti che si potevano imparare facilmente a memoria. Mantenere una descrizione scritta del codice segreto era considerato un rischio troppo grande. Una tabella, come si usa qui, è facile da imparare a memoria. Il famoso codice segreto dei massoni si basa su questo principio.

Invece di limitarsi a mettere insieme nuovi caratteri per le cifre, si possono anche inventare nuovi codici segreti per i testi. Per fare questo, si scrivono tutte le lettere in una tabella e si inventano nuovi simboli per le colonne e le righe, ottenendo così nuovi caratteri per tutte le lettere.

Parole chiave e siti web

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

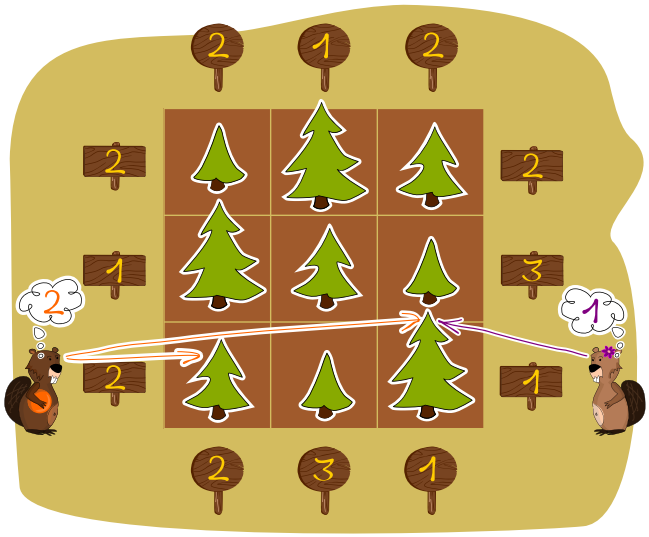
5. 3×3 sudoku con gli alberi

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

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

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

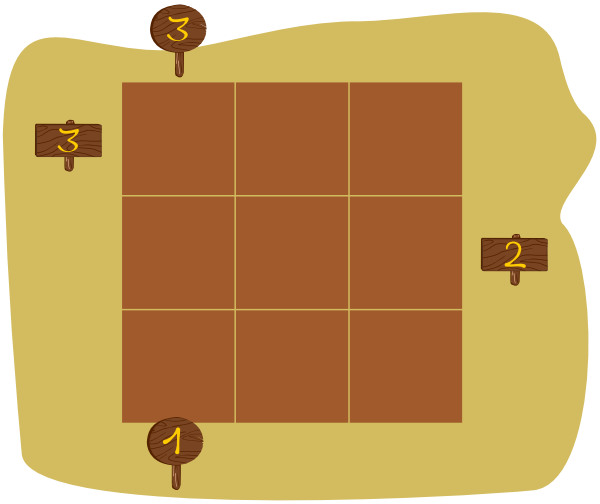
Ora i castori piantano nove abeti in un campo 3×3, come nell'esempio a destra.

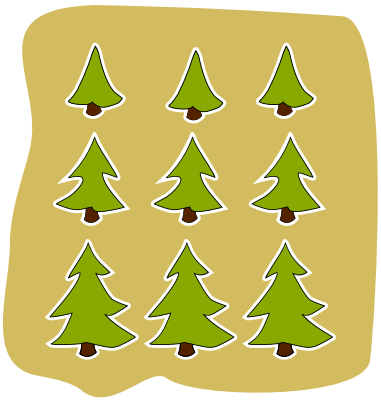


Si applicano le seguenti regole:

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

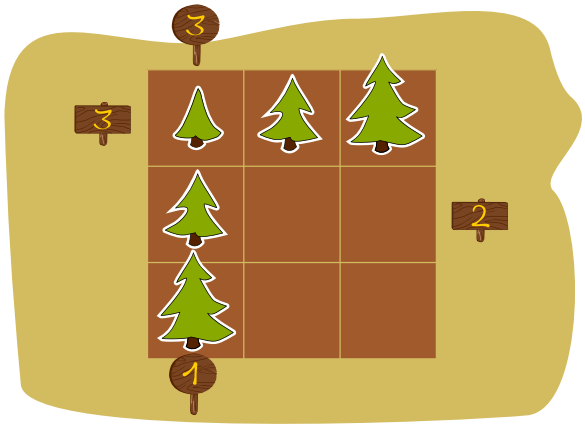
Scrivi in ogni campo l'altezza dell'albero corrispondente.





Soluzione

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

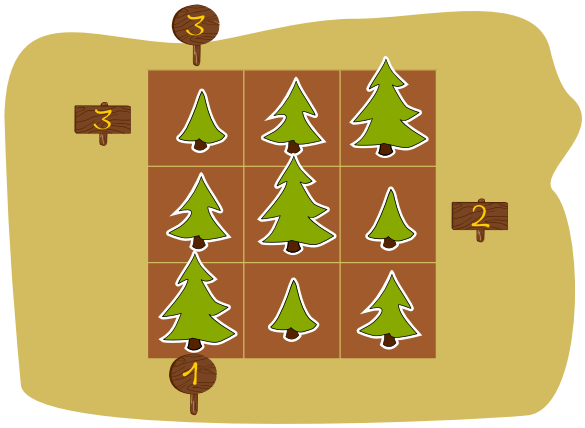


Il cartello a destra con il 2 richiede che da lì siano visibili due abeti, quindi deve esserci un abete di altezza 3 proprio al centro e questa fila centrale è quindi 2 (), 3 (), 1 ().

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

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

La soluzione completa si presenta così:



Questa è l'informatica!

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

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

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

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

Parole chiave e siti web

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

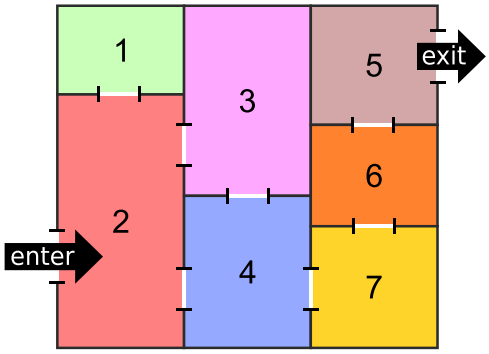
6. Visita al museo

Per un nuovo museo vengono proposte quattro planimetrie per le stanze. Ogni piano contiene le sette stanze da 1 a 7, e le stanze dovrebbero essere progettate in modo tale che i visitatori possano visitare tutte le stanze senza entrare due volte in una stanza.

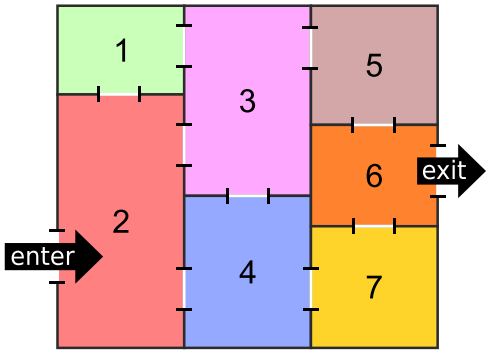
I visitatori iniziano la visita da «enter» e escono dal museo da «exit».

Quale piantina permette ai visitatori di entrare e uscire da ogni stanza esattamente una volta?

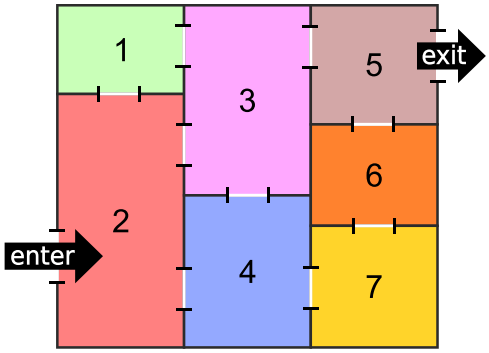
A)



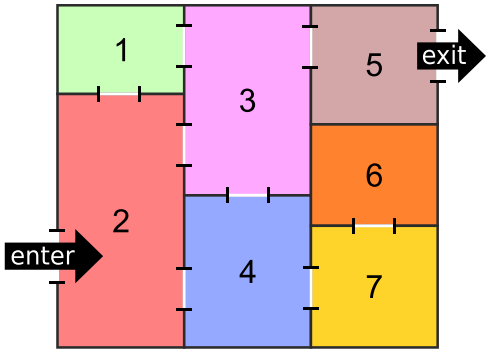
B)



C)

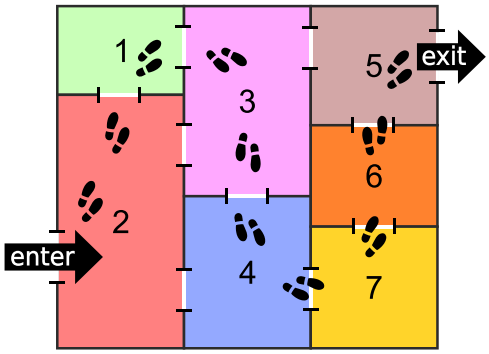


D)



Soluzione

Solo la piantina C permette ai visitatori di entrare e uscire da ogni stanza esattamente una volta. L'ordine delle stanze è: 2, 1, 3, 4, 7, 6, 5.



In generale, un tour di questo tipo è sempre impossibile se una delle stanze ha un solo ingresso. La spiegazione è la seguente: Se un visitatore entra in questa stanza, quando esce deve tornare nella stanza da cui è venuto, infrangendo la regola di entrare in ogni stanza una sola volta.

Nella piantina A, la camera 1 ha un solo ingresso.

Nella piantina D, la camera 6 ha un solo ingresso.

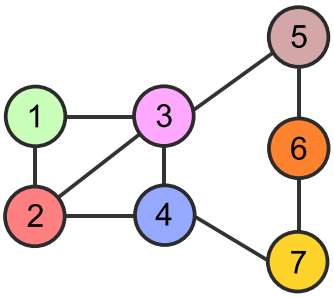
Nella piantina B, l'ultima stanza 6 è raggiungibile dalla stanza 5 o dalla stanza 7. Se il visitatore arriva dalla stanza 5, può entrare nella stanza 7, ma dopo può raggiungere l'uscita solo attraverso la stanza 6 (o viceversa), che ancora una volta infrange le regole.

Questa è l'informatica!

La maggior parte dei bambini o degli adolescenti risolve il problema per tentativi senza ulteriori rappresentazioni astratte. In questo modo utilizzano in una certa misura la strategia generale del *backtracking*. Per lo meno si rendono conto che si può imparare dai tentativi falliti e in questo caso si può tornare indietro per provare un'altra opzione. Allo stesso tempo, si confrontano con l'importante concetto di *non determinismo*, perché spesso hanno diverse opzioni tra cui scegliere.

Il compito è un esempio di un problema ben noto nell'informatica, la ricerca di un *cammino hamiltoniano*. In una rappresentazione astratta sotto forma di grafo, ogni stanza corrisponde ad un *nodo* e ogni porta tra due stanze corrisponde ad un *arco* tra i due nodi corrispondenti.

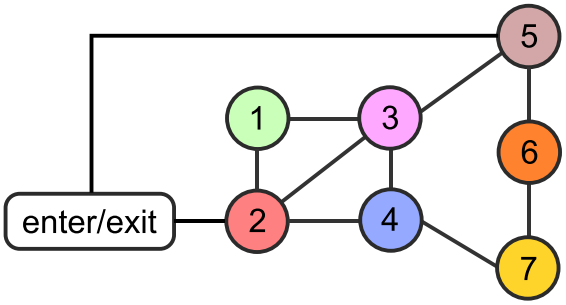
In termini astratti, il compito si presenta così:



Il compito ora è quello di trovare un percorso in questo grafo con le seguenti proprietà:

1. Il percorso inizia in 2 («ingresso»)
2. Il percorso termina in 5 («uscita»).
3. Ogni nodo viene visitato esattamente una volta.

Se si combina lo spazio esterno in un nodo, allora il tutto corrisponde alla ricerca di un *ciclo hamiltoniano* (un giro), dove ogni nodo viene anche attraversato esattamente una volta e si finisce di nuovo al nodo di partenza.



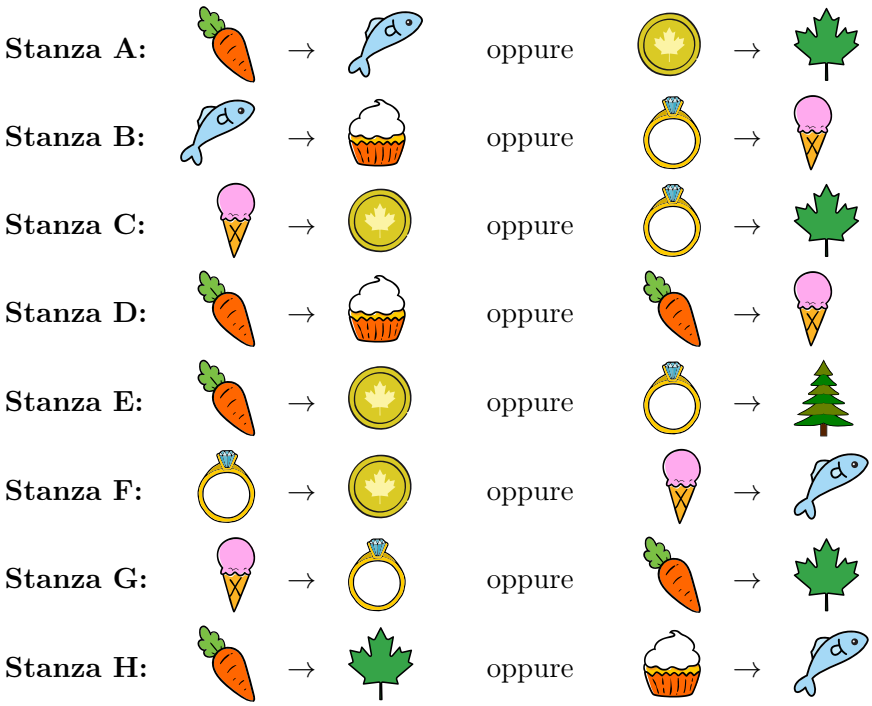
Parole chiave e siti web

- Teoria dei grafi: `https://it.wikipedia.org/wiki/Teoria_dei_grafi`
- Cammino hamiltoniano: `https://it.wikipedia.org/wiki/Cammino_hamiltoniano`

7. Castoro al castello

Un castoro intelligente ha bisogno di un abete per costruire una diga nel fiume. Purtroppo ha solo una carota. Oggi è giorno di mercato nel castello e il castoro vuole scambiare la sua carota con un abete.

Ogni stanza del castello offre due offerte di scambio. La tabella mostra queste offerte:



Per esempio, nella stanza B, il castoro può ottenere un cono per un anello, ma non viceversa.

In quale ordine il castoro intelligente deve attraversare le stanze per possedere finalmente l'abete desiderato?

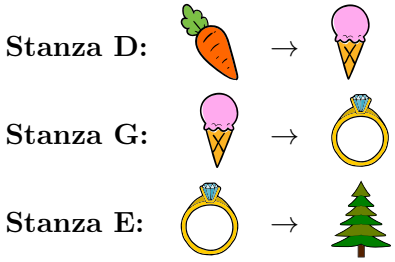
A) DGE: Prima la stanza D, poi la stanza G e infine la stanza E.
B) GGE: Prima la stanza G, poi di nuovo la stanza G e infine la stanza E.
C) AGE: Prima la stanza A, poi la stanza G e infine la stanza E.
D) DBC: Prima la stanza D, poi la stanza B e infine la stanza C.

Soluzione

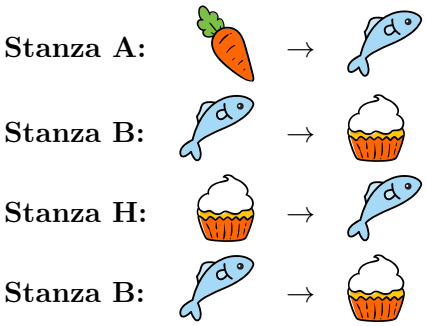
La risposta corretta è A) DGE: prima la stanza D, poi la stanza G e infine la stanza E.

Nella stanza D il castoro scambia la sua carota con un cono. Poi va nella stanza G, dove scambia il cono con un anello. Alla fine il castoro va nella stanza E per scambiare l'anello con un abete.

Questa sequenza complessiva si presenta così:



Per trovare un ordine adeguato delle stanze, due diverse strategie sono utili. La prima strategia cerca di considerare tutte le possibilità di scambio. Si inizia con il primo scambio, dove si può scambiare la carota in cinque stanze (A, D, E, G e H) con 6 oggetti diversi. In seguito, tutte le possibilità di scambio per questi 6 oggetti vengono nuovamente considerate. Questo è complesso e si può anche girare in cerchio, come nell'esempio seguente, dove il castoro può visitare le stanze B e H tutte le volte che vuole:

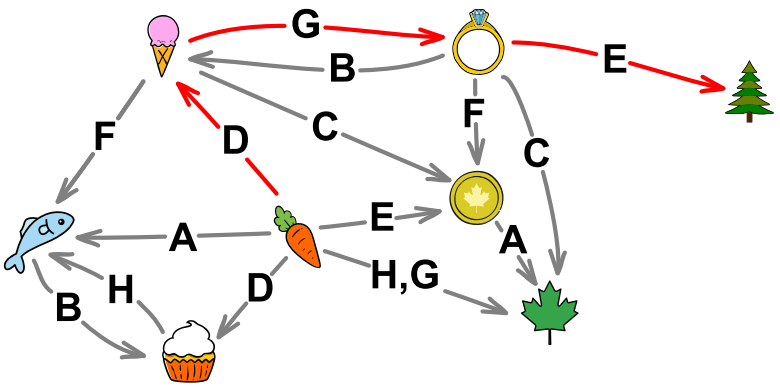


Pertanto questa prima strategia è molto complessa e solo con un po' di fortuna si può avere successo in breve tempo.

La seconda strategia porta rapidamente all'obiettivo in questo compito concreto. Si basa sull'avvio della ricerca dall'obiettivo desiderato, l'abete. Solo nella stanza E il castoro può ottenere l'abete desiderato e l'abete può essere ottenuto solo in cambio di un anello. Il prossimo oggetto desiderato è quindi un anello! L'anello può essere ottenuto solo in una stanza, la stanza G, in cambio di un cono. È possibile ottenere il cono sia nella stanza B scambiandolo con un anello, che nella stanza D scambiandolo con una carota. Quindi il castoro intelligente deve iniziare il suo scambio nella stanza D.

Per una migliore visione d'insieme, la tabella degli scambi possibili può essere visualizzata sotto forma di grafo orientato con archi. Ogni nodo del grafo rappresenta un oggetto di scambio e ogni

arco in uscita rappresenta una possibilità di scambio. Inoltre, l'arco indica lo spazio in cui esiste questa possibilità di scambio.



Questa rappresentazione visiva degli oggetti di scambio, delle possibilità di scambio e delle stanze permette di scoprire facilmente come arrivare dalla carota all'abete, ovvero su un percorso nel grafo orientato: Prima la stanza D, poi la stanza G e infine la stanza E.

Questa è l'informatica!

I *processi di calcolo* possono essere considerati a livello generale come *conseguenze di trasformazioni* (qui si tratta di processi di scambio) o, in modo equivalente, come conseguenze di *stati* di un sistema. Lo stato iniziale del sistema nel nostro caso è la carota e la trasformazione (*la transizione*) da carota a cono cambia questo stato in cono.

Una transizione porta così da uno stato all'altro. Una sequenza di transizioni è anche chiamata *calcolo*.

Questo compito si occupa quindi anche di calcoli a livello molto generale. In questo caso, il sistema *non è deterministico*; ciò significa che ci sono a volte diverse possibili fasi di calcolo, cioè, come nel compito, diversi possibili scambi. Il non determinismo è un altro importante concetto di modellazione nell'informatica. Le possibili fasi di calcolo sono descritte da *regole di trasformazione* (la tabella con possibilità di scambio). Determinare se il castoro può scambiare una carota con un abete, cioè se un certo stato obiettivo del sistema può essere raggiunto da un certo stato di partenza, è il famoso STCON (*st-connettività*) con numerose applicazioni.

Il compito di cui sopra mostra che a volte è una buona idea cercare lo stato di partenza dallo stato di destinazione invece che il contrario. Questa strategia è chiamata anche *ricerca inversa*.

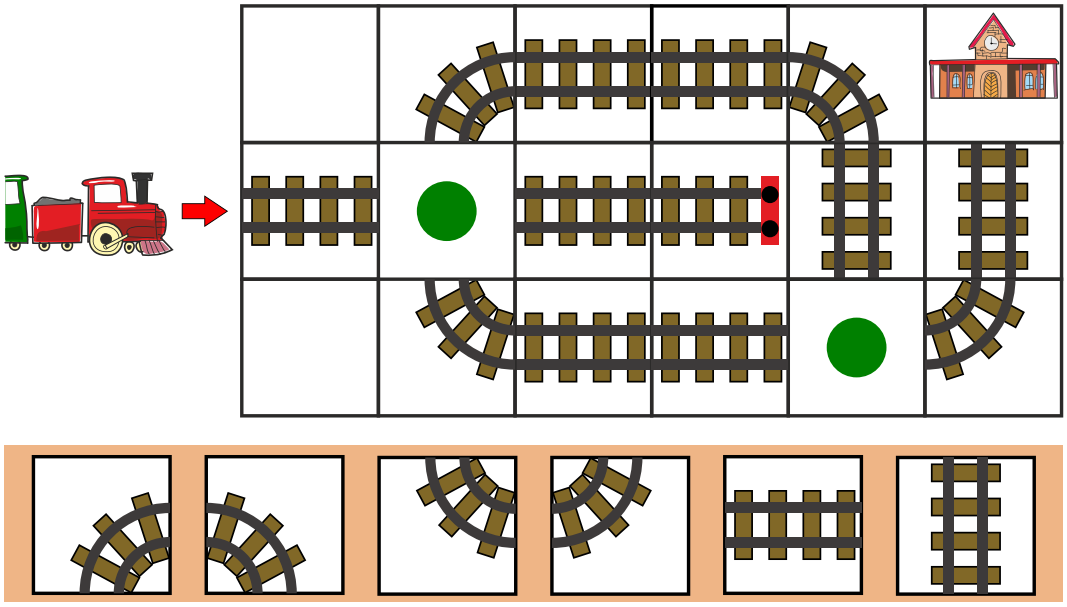
Quando si confrontano le diverse strategie di soluzione, si può notare che il grafo orientato è un modo illustrativo di rappresentare un cosiddetto *spazio di stato* di un sistema con tutte le possibili transizioni tra gli stati. In questo modello di base si potrebbero affrontare i ben noti *algoritmi di ricerca* di base nei grafi, cioè la *ricerca in ampiezza* e la *ricerca in profondità*.

Parole chiave e siti web

- Grafi: `https://it.wikipedia.org/wiki/Grafo`
- Ricerca in profondità: `https://it.wikipedia.org/wiki/Ricerca_in_profondità`
- Ricerca in ampienzza: `https://it.wikipedia.org/wiki/Ricerca_in_ampiezza`

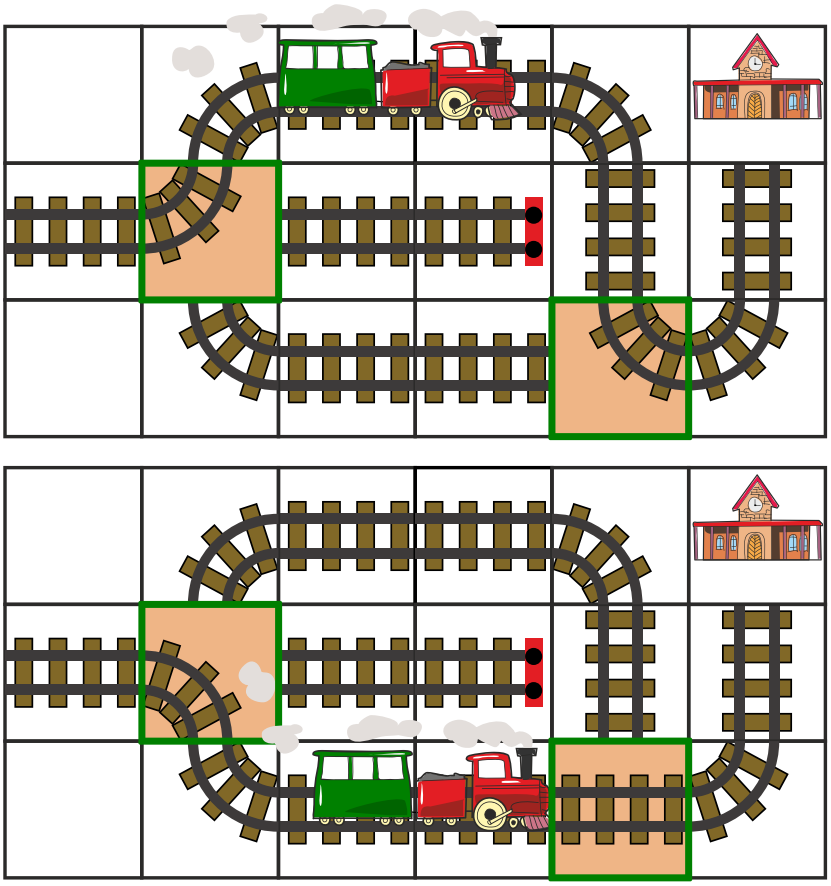
8. Prossima fermata, stazione!

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



Soluzione

Per questo problema ci sono le seguenti 2 soluzioni:



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

Questa è l'informatica!

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

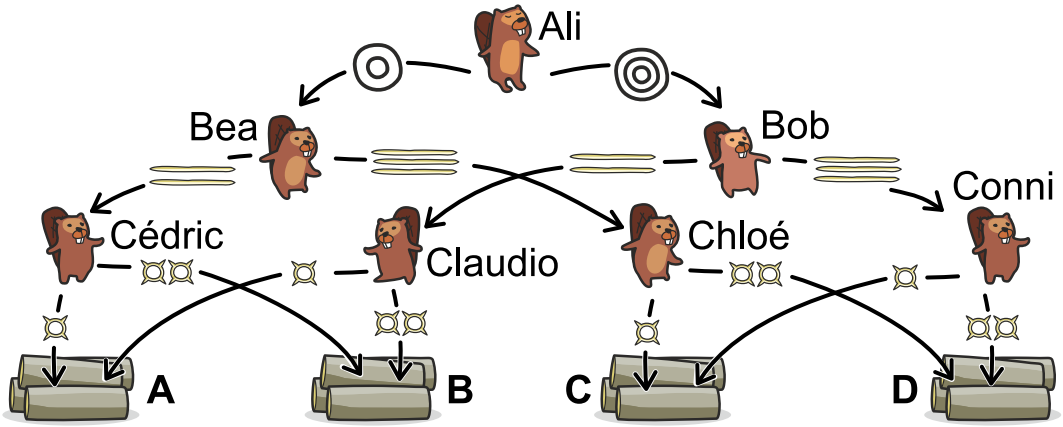
Il compito di cui sopra consiste nell'inserire i comandi mancanti nei punti giusti di un programma in modo da raggiungere l'obiettivo.

Parole chiave e siti web

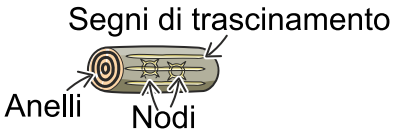
- Programma
- Instruzione: `https://it.wikipedia.org/wiki/Istruzione_(informatica)`
- `https://it.wikipedia.org/wiki/Algoritmo`

9. Tronchi e pile

Nel villaggio dei castori i tronchi sono divisi in quattro gruppi (A, B, C, D) secondo tre caratteristiche (numero di anelli del tronco, segni di trascinamento sulla corteccia e numero di nodi). Il seguente diagramma di decisione mostra come si dividono fra i gruppi.

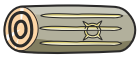


Per esempio, questo tronco viene inserito nella pila D in seguito alle seguenti decisioni:



- Ali vede tre anelli e dà il tronco a Bob;
- Bob vede tre segni di trascinamento e dà il tronco a Conni;
- Conni vede due nodi e mette il tronco sulla pila D.

Su quale pila è collocato questo tronco?



A) Pila A
B) Pila B
C) Pila C
D) Pila D

Soluzione

La risposta corretta è la pila C. Questo perché Ali vede due anelli e dà il tronco a Bea. Bea vede tre segni di trascinamento e dà il tronco a Chloé. Chloé vede un nodo e mette il tronco sulla pila C.

Se lo si desidera, è possibile determinare per ogni pila quali tronchi appartengono alla rispettiva pila. Ci sono due tipi di tronchi su ogni pila.

Sulla pila **A**:

- Tronchi con 2 anelli, 2 segni di trascinamento e 1 nodo.
- Tronchi con 3 anelli, 2 segni di trascinamento e 1 nodo.

Sulla pila **B**:

- Tronchi con 2 anelli, 2 segni di trascinamento e 2 nodi.
- Tronchi con 3 anelli, 2 segni di trascinamento e 2 nodi.

Sulla pila **C**:

- Tronchi con 2 anelli, 3 segni di trascinamento e 1 nodo.
- Tronchi con 3 anelli, 3 segni di trascinamento e 1 nodo.

Sulla pila **D**:

- Tronchi con 2 anelli, 3 segni di trascinamento e 2 nodi.
- Tronchi con 3 anelli, 3 segni di trascinamento e 2 nodi.

Questa è l'informatica!

Questo compito tocca diversi concetti dell'informatica.

In primo luogo, viene affrontato il concetto di diagrammi decisionali, che hanno applicazioni molto versatili nel campo dell'informatica. Qui vengono utilizzati per classificare gli oggetti in categorie selezionate (molto spesso si tratta di alberi di decisione, un tipo speciale di diagrammi decisionali. Il diagramma decisionale del compito non è un albero decisionale in questo caso, perché al livello più basso due gruppi sono posizionati sulla stessa pila).

Qui si può anche pensare al diagramma decisionale come a una rappresentazione astratta dei valori di una funzione di diverse variabili. A livello terminologico, in informatica si parla di «branching programs» (inglese per «programmi di ramificazione»).

Si riferisce anche al concetto di attributi (caratteristiche o proprietà) degli oggetti. Qui gli oggetti hanno tre attributi (anelli del tronco, segni di trascinamento, nodi), con ogni attributo che ha due possibili valori (due o tre anelli annuali o segni di trascinamento e uno o due nodi).

Ci sono molte possibili applicazioni per tali diagrammi decisionali. Uno di questi è la classificazione dei pacchetti di dati quando vengono inviati attraverso una rete (con router o switch).

Parole chiave e siti web

- Albero di decisione: `https://it.wikipedia.org/wiki/Albero_di_decisione`

A. Autori dei quesiti

Serge Adam

Wilfried Baumann

Carlo Bellettini

Linda Björk Bergsveinsdóttir

Daniela Bezáková

Lucia Budinská

Sarah Chan

Marios O. Choudary

Valentina Dagienė

Christian Datzko

Susanne Datzko

Lidia Feklistova

Fabian Frei

Christian Giang

Husnul Hakim

Juraj Hromkovič

Alisher Ikramov

Ungyeol Jung

Vaidotas Kinčius

Regula Lacher

Vu Van Luan

Hamed Mohebbi

Kwangsik Moon

Xavier Muñoz

Tom Naughton

Gabriel Parriaux

Jean-Philippe Pellet

Margot Phillipps

Wolfgang Pohl

Pedro Ribeiro

Peter Rossmanith

Vipul Shah

Peter Tomcsányi

Monika Tomcsányiová

Jiří Vaníček

Michael Weigend

Jonas Winckler

Michal Winczer

B. Sponsoring: concorso 2020

http://www.haslerstiftung.ch/

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

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

Standortförderung beim Amt für Wirtschaft und Arbeit Kanton Zürich

i-factory (Musée des transports, Lucerne)

http://www.ubs.com/

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

https://educatec.ch/
educaTEC

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

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

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

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

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

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

`https://www.zhdk.ch/`
Zürcher Hochschule der Künste

C. Ulteriori offerte

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

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

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