INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Quesiti e soluzioni 2018
9º e 10º 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

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 9 ottobre 2019 col sistema per la preparazione di testi LaTeX.

Premessa

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

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

Alla categoria del 3º e 4º anno scolastico sono stati assegnati 9 quesiti da risolvere, di cui 3 facili, 3 medi e 3 difficili. Alla categoria del 5º e 6º anno scolastico sono stati assegnati 12 quesiti, suddivisi in 4 facili, 4 medi e 4 difficili. Ogni altra categoria ha ricevuto invece 15 quesiti da risolvere, di cui 5 facili, 5 medi e 5 difficili.
Per ogni risposta corretta sono stati assegnati dei punti, mentre per ogni risposta sbagliata sono stati detratti. In caso di mancata risposta il punteggio è rimasto inalterato. Il numero di punti assegnati o detratti dipende dal grado di difficoltà del quesito:

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

Il sistema internazionale utilizzato per l'assegnazione dei punti limita l'eventualità che il partecipante possa ottenere buoni risultati scegliendo le risposte in modo casuale.
Ogni partecipante ha iniziato con un punteggio pari a 45 punti (risp., Piccolo Castoro: 27 punti, 5º e 6º anno scolastico: 36 punti).
Il punteggio massimo totalizzabile era dunque pari a 180 punti (risp., Piccolo castoro: 108 punti, 5º e 6º anno scolastico: 144 punti), mentre quello minimo era di 0 punti.
In molti quesiti le risposte possibili sono state distribuite sullo schermo con una sequenza casuale. Lo stesso quesito è stato proposto in più categorie d'età.

Per ulteriori informazioni:

SVIA-SSIE-SSII Società Svizzera per l'Informatica nell'Insegnamento
Castoro Informatico
Andrea Adamoli
https://www.castoro-informatico.ch/it/kontaktieren/
https://www.castoro-informatico.ch/
https://www.facebook.com/informatikbiberch

Indice

Hanno collaborato al Castoro Informatico 2018 i

Premessa ii

1. Le cascate 1

2. Il laghetto dei castori 5

3. Il concorso dei castori 7

4. Casa di vacanza Nr. 29 9

5. Alieni! 11

6. Vicini 13

7. Videogioco 17

8. Visitare gli amici 19

9. Due castori al lavoro 23

10. Salti 25

11. Regali 27

12. Righe e colonne 29

13. Ordinare i libri 33

14. Soundex 37

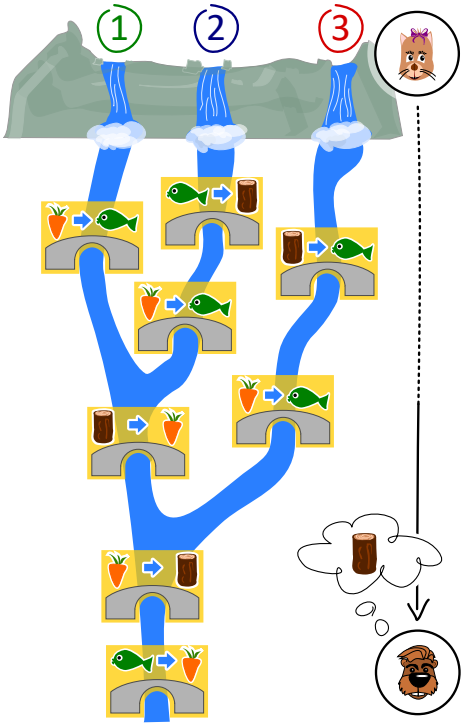
15. Tre amici 39

A. Autori dei quesiti 41

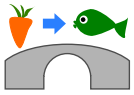
B. Sponsoring: concorso 2018 42

C. Ulteriori offerte 44

1. 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.

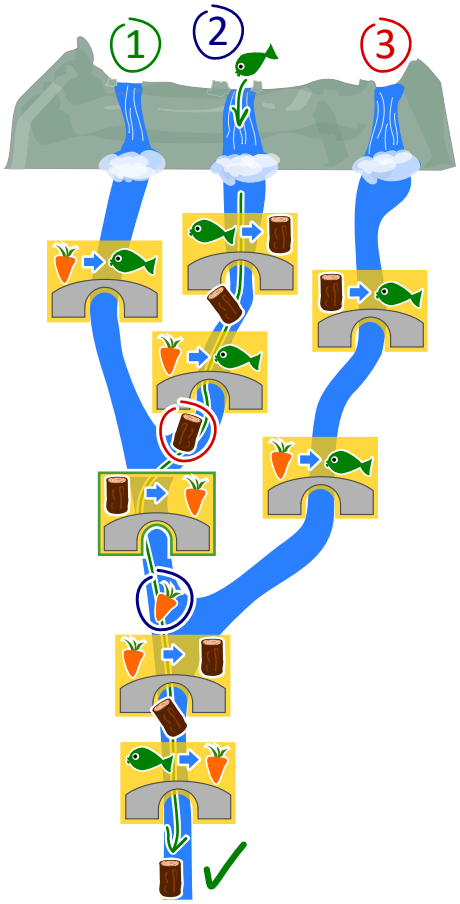
B) Deve far cadere un pesce nella cascata numero 2.

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

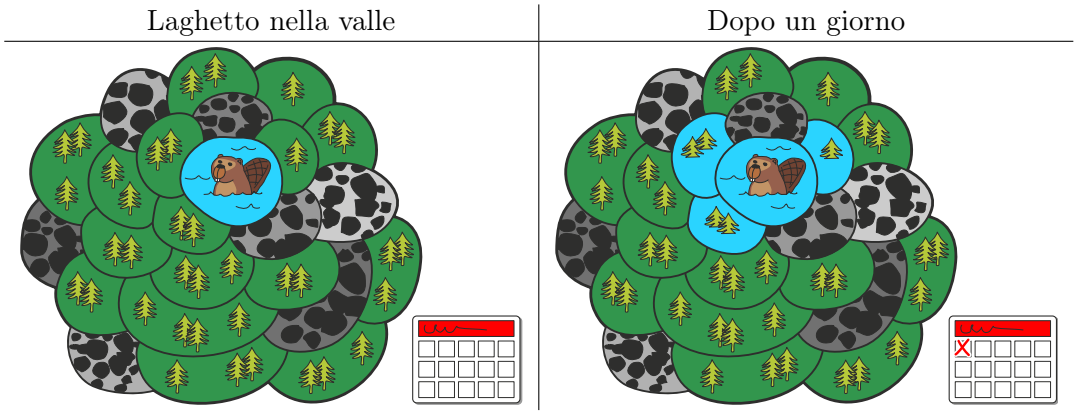
collaudo e test dei software, paradigmi di programmazione, programmazione funzionale, funzioni e parametri

- `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`

2. Il laghetto dei castori

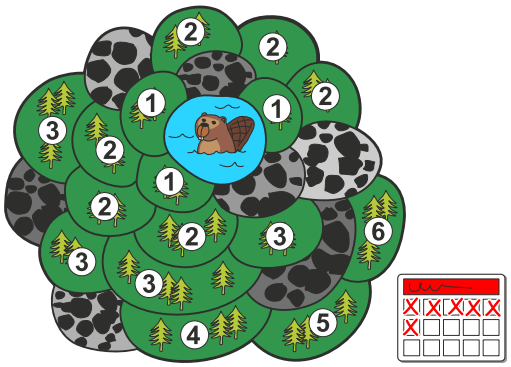
In una valle c'è un piccolo laghetto, circondato da aree boschive o rocciose. Nel laghetto vivono alcuni castori.
Un giorno il laghetto diventa troppo piccolo per i castori e quindi decidono di allagare le aree boschive. Ogni giorno allagano delle nuove aree boschive confinanti con il lago: dopo il primo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

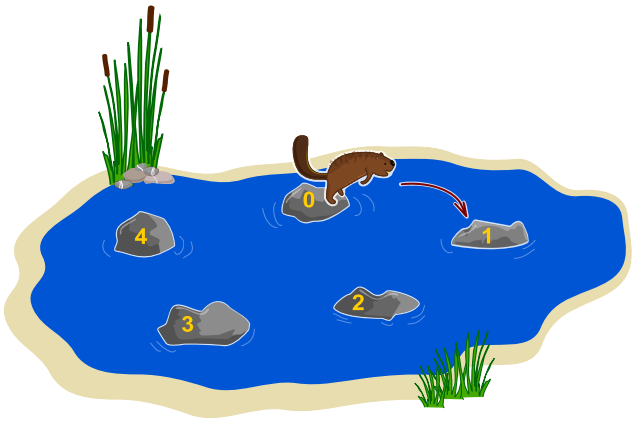
ricerca in ampiezza, algoritmo wavefront ("fronte d'onda")

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

3. 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`

4. 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).

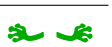
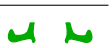
Parole chiave e siti web

Albero di ricerca binario, Albero AVL

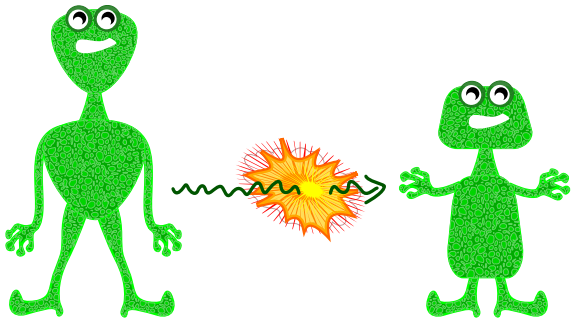
- `https://it.wikipedia.org/wiki/Albero_binario_di_ricerca`
- `https://it.wikipedia.org/wiki/Albero_AVL`

5. Alieni!

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

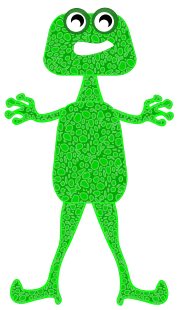
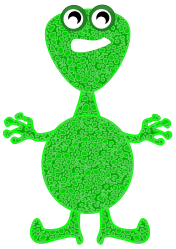
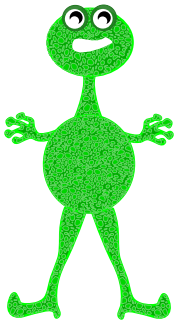
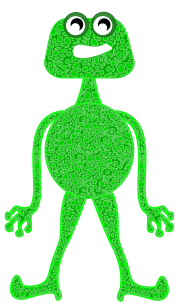
$C(r)$	Il capo è rotondo.	
$C(4)$	Il capo è quadrangolare.	
$C(3)$	Il capo è triangolare.	
$T(r)$	Il tronco è rotondo.	
$T(4)$	Il tronco è quadrangolare.	
$T(3)$	II tronco è triangolare.	
$B(+)$	Le braccia sono lunghe.	
$B(-)$	Le braccia sono corte.	
$G(+)$	La gambe sono lunghe.	
$G(-)$	La gambe sono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

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

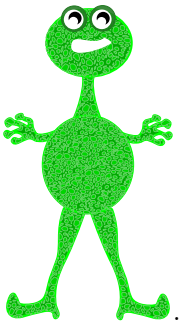


Come appare l'alieno dopo i seguenti comandi?

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

A)	B)	C)	D)
			

Soluzione

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

Questa è l'informatica!

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

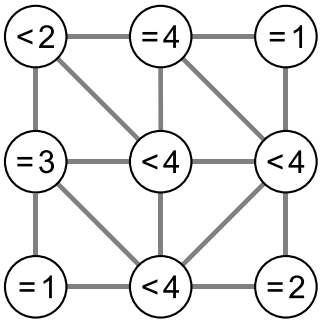
Parole chiave e siti web

variabile, sequenza, programmazione

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

6. Vicini

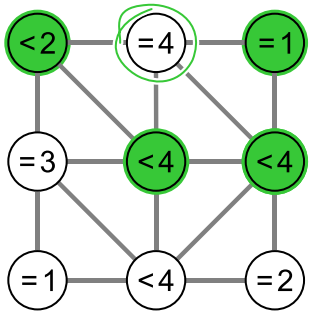
L'immagine qui sotto mostra nove cerchi, in parte connessi. Una connessione (linea) li definisce come "vicini".
Ogni cerchio contiene un'espressione che indica quanti dei suoi vicini devono essere colorati. Per esempio, "$= 3$" significa che esattamente tre dei vicini devono essere colorati; "< 4" significa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



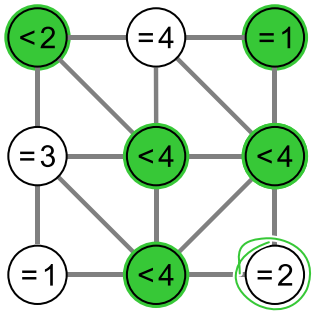
Colora i cerchi in modo che ogni condizione (espressione) sia soddisfatta.

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`

7. 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.

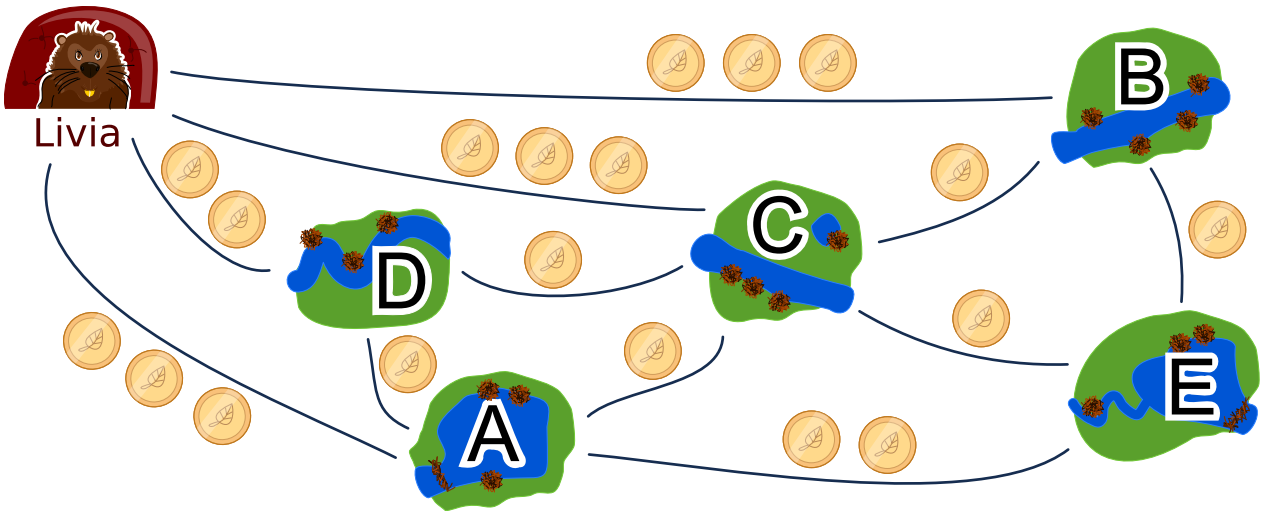
Parole chiave e siti web

analisi, verifica e validazione del software

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

8. 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:

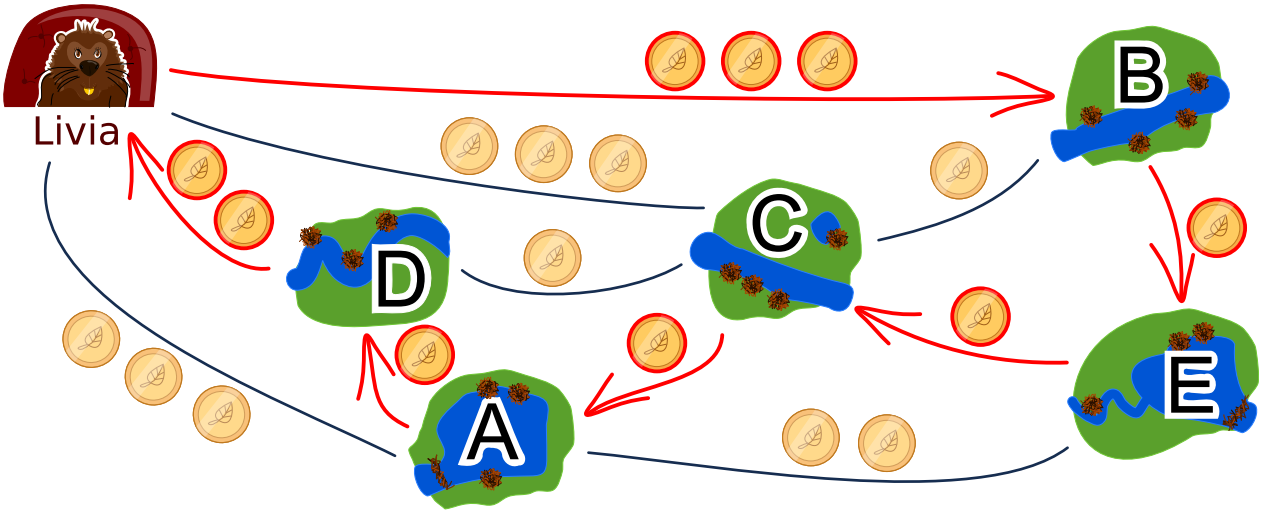
$$\text{Casa} \to \text{B} \to \text{E} \to \text{A} \to \text{D} \to \text{C} \to \text{Casa}.$$

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.

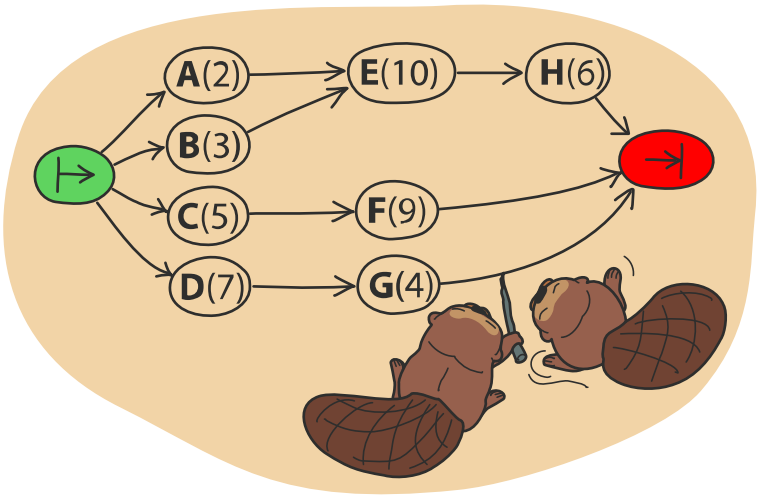
Parole chiave e siti web

ottimizzazione, problema del commesso viaggiatore

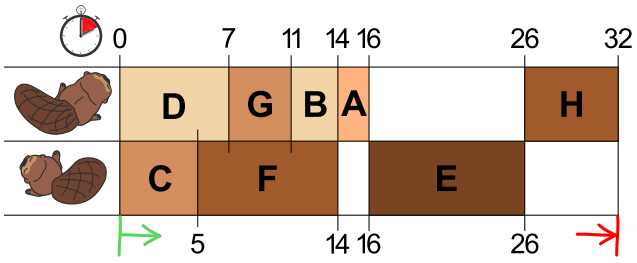
- `https://it.wikipedia.org/wiki/Problema_del_commesso_viaggiatore`
- `https://it.wikipedia.org/wiki/Problema_di_ottimizzazione`

9. 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.



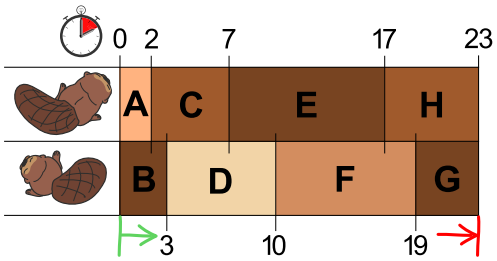
La figura qui sotto mostra un possibile piano di lavoro per i due castori... Si può però di sicuro terminare la diga più velocemente!



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

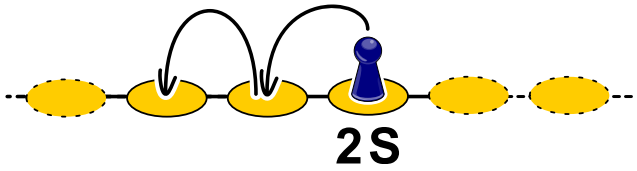
scheduling ("pianificazione"), algoritmo greedy ("avido")

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

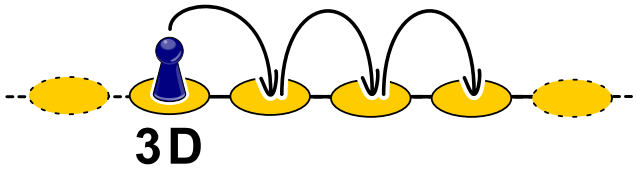
10. Salti

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

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

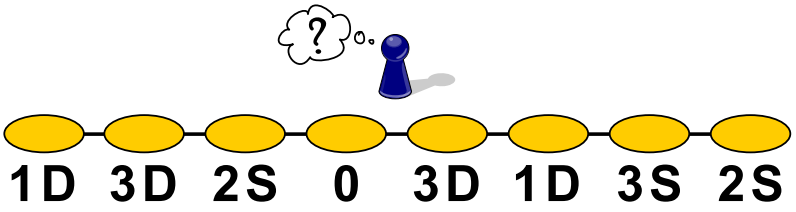


- nD: saltare n volte (posizioni) a destra, quindi 3D significa saltare tre volte a destra,



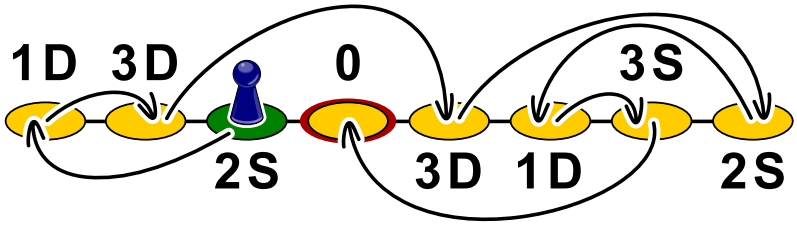
- 0: terminare.

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



Soluzione

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



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

Questa è l'informatica!

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

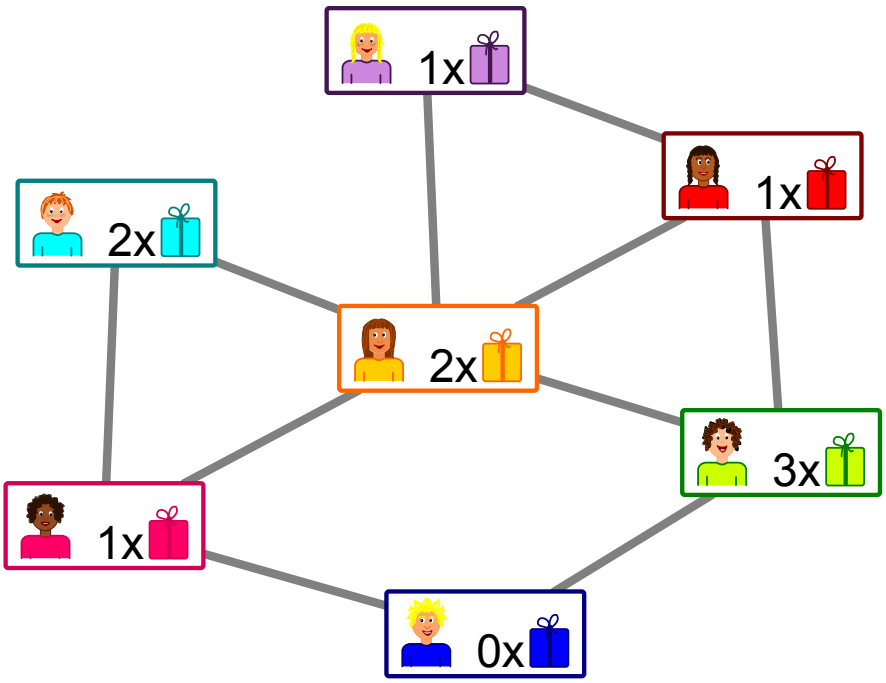
Parole chiave e siti web

lista, gestione della memoria, GOTO

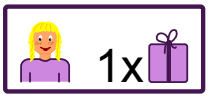
- `https://it.wikipedia.org/wiki/Garbage_collection`
- `https://en.wikipedia.org/wiki/St-connectivity`

11. Regali

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



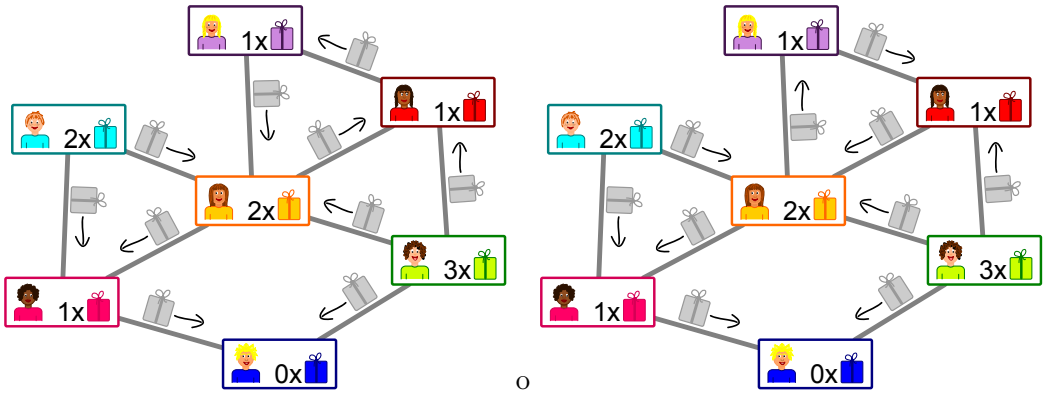
Gli inquilini pianificano una festa con dei regali per i ragazzi: per ogni coppia di amici, uno dei due ragazzi dovrà fare un regalo all'altro.

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

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

Soluzione

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



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

Questa è l'informatica!

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

Parole chiave e siti web

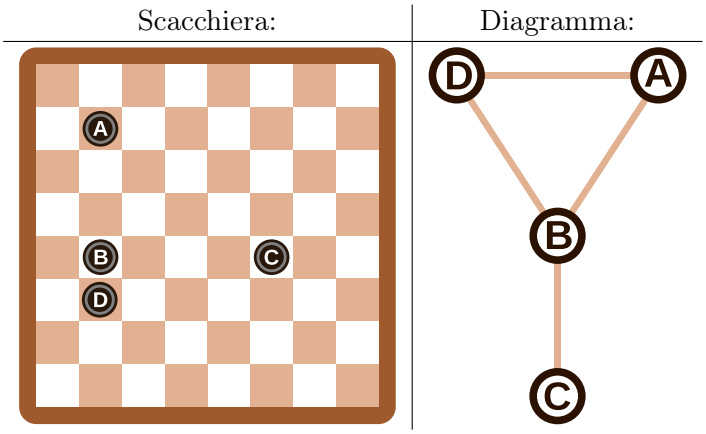
rete di flusso, riduzione di problemi

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

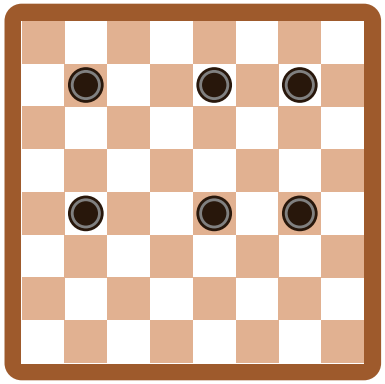
12. Righe e colonne

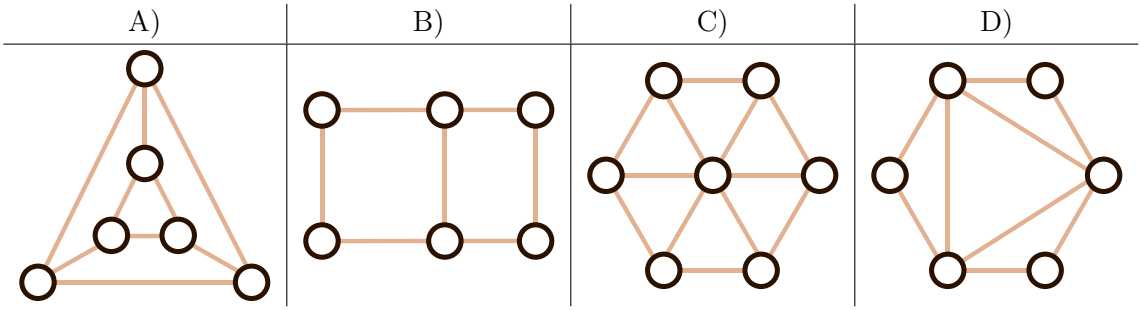
Il diagramma mostrato sulla destra della scacchiera è stato allestito partendo dalla posizione delle pedine in modo che...

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



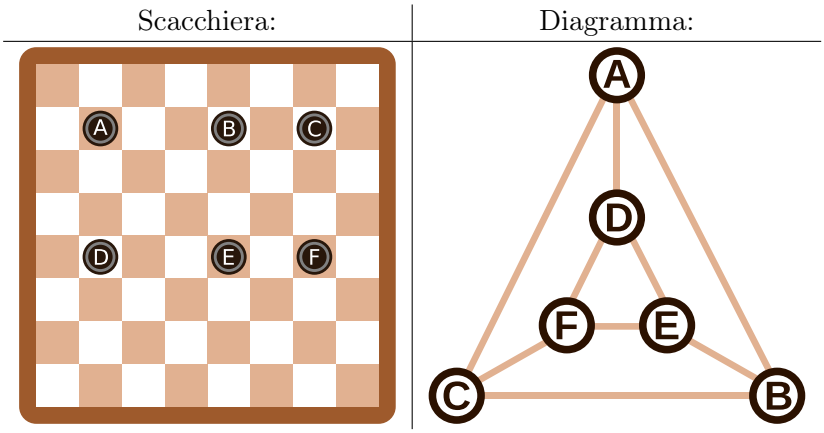
Nel nostro esempio, le pedine sulla scacchiera e i cerchi nel diagramma sono etichettati con una lettera dell'alfabeto, in modo da rendere chiara la spiegazione.
Quale diagramma è associato alla scacchiera seguente, sulla quale sono posizionate 6 pedine?



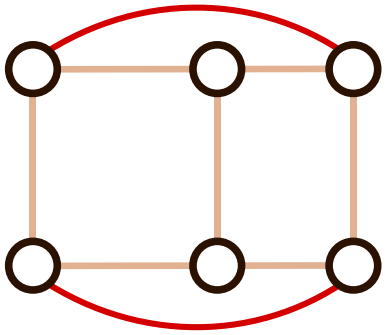


Soluzione

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



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



Questa è l'informatica!

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

Parole chiave e siti web

grafo

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

13. Ordinare i libri

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

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

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

Questa è la situazione iniziale:

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

A) tre turni

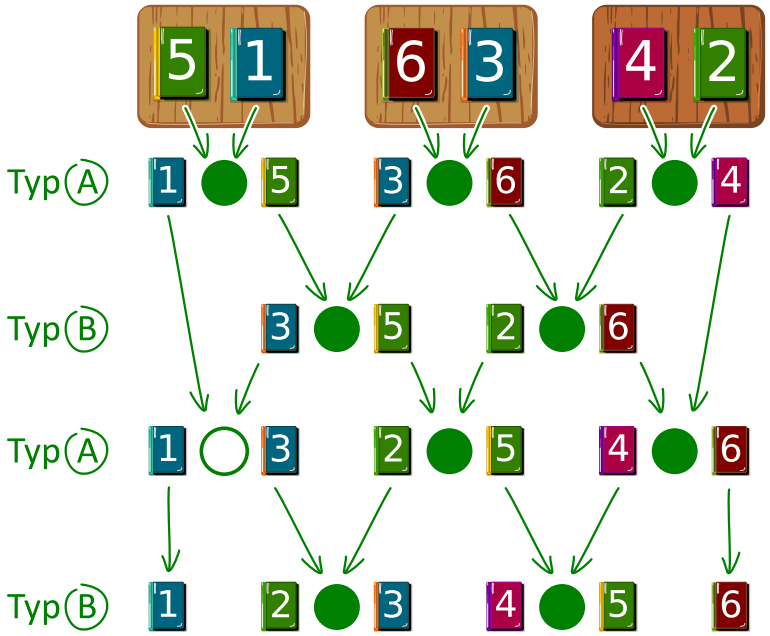
B) quattro turni

C) cinque turni

D) sei turni

Soluzione

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



Questa è l'informatica!

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

Parole chiave e siti web

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

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

14. Soundex

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

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

Le parole seguenti sono state codificate in questo modo:

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

Qual è il codice della parola "Hilbert"?

A) H410

B) B540

C) H041

D) H416

Soluzione

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

Questa è l'informatica!

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

Parole chiave e siti web

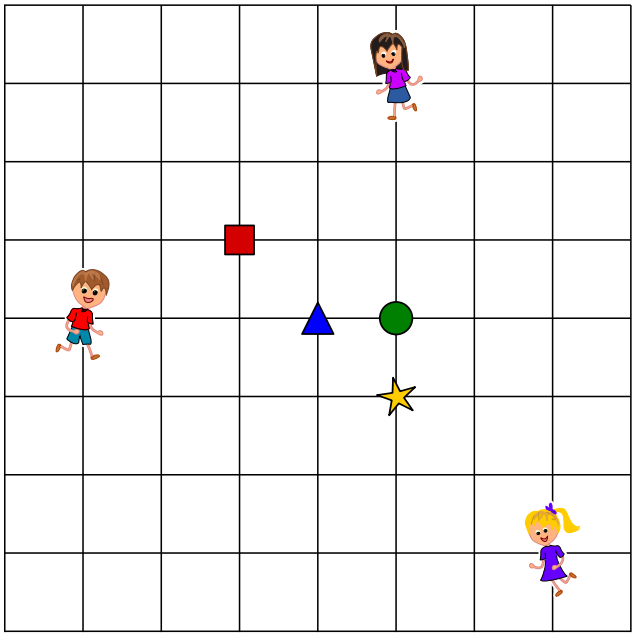
Ricerca fonetica, soundex

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

15. Tre amici

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

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



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

A)	B)	C)	D)
			

Soluzione

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

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

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

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

Questa è l'informatica!

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

Parole chiave e siti web

problema di ottimizzazione, ricerca locale

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

A. Autori dei quesiti

Andrea Adamoli
Jared Asuncion
Daphne Blokhuis
Lucia Budinská
Špela Cerar
Kessarapan Charoensueksa
Kris Coolsaet
Valentina Dagienė
Christian Datzko
Susanne Datzko
Dilek Doğan
Marissa Engels
Hanspeter Erni
Gerald Futschek
Ionuț Gorgos
Shuchi Grover
Yasemin Gülbahar
Martin Guggisberg
Bent Halden
Urs Hauser
Juraj Hromkovič
Takeharu Ishizuka
Svetlana Jakšić
Zhang Jinbao
Dong Yoon Kim
Vaidotas Kinčius
Jia-Ling Koh
Regula Lacher
Anh Vinh Le
Dimitris Mavrovouniotis
Karolína Mayerová
Samart Moodleah
Tom Naughton
Henry Ong
Péter Piltmann
Zsuzsa Pluhár
Wolfgang Pohl
Ilya Posov
Nol Premasathian
J.P. Pretti
Doris Reck
Chris Roffey
Kirsten Schlüter
Andrea Maria Schmid
Jacqueline Staub
Allira Storey
Gabrielė Stupurienė
Peter Tomcsányi
Troy Vasiga
Rechilda Villame
Eslam Wageed
Pieter Waker
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.