INFORMATIK-BIBER SCHWEIZ
CASTOR INFORMATIQUE SUISSE
CASTORO INFORMATICO SVIZZERA

Quesiti e soluzioni 2023

5º e 6º anno scolastico

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

A cura di:

Susanne Datzko-Thut, Nora A. Escherle, Masiar Babazadeh,
Christian Giang, Jean-Philippe Pellet

SSII
www.svia-ssie-ssii.ch
schweizerischerverein für informatikin
erausbildung//société suisse pour l'infor
matiquedans l'enseignement//società sviz
zera per l'informatica nell'insegnamento

HASLERSTIFTUNG ETH zürich ABZ AUSBILDUNGS- UND BERATUNGSZENTRUM FÜR INFORMATIKUNTERRICHT hep/ Scuola universitaria professionale della Svizzera italiana SUPSI

Hanno collaborato al Castoro Informatico 2023

Masiar Babazadeh, Susanne Datzko-Thut, Jean-Philippe Pellet, Giovanni Serafini, Bernadette Spieler

Capo progetto: Nora A. Escherle

Un particolare ringraziamento per il lavoro sui quesiti del concorso Svizzero va a:
Juraj Hromkovič, Angélica Herrera Loyo, Regula Lacher und Manuel Wettstein: ETH Zürich, Ausbildunges- und Beratungszentrum für Informatikunterricht
Tobias Berner: Pädagogische Hochschule Zürich
Christian Datzko: Wirtschaftgymnasium und Wirtschaftsmittelschule, Basel
Fabian Frei: CISPA - Helmholtz-Zentrum für Informationssicherheit
Sebastian Knüsli: Gymnasium Kirschgarten, Basel

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, Vaidotas Kinčius: Bebras.org, Lituania
Wolfgang Pohl, Jakob Schilke: Bundesweite Informatikwettbewerbe (BWINF), Germania
Hannes Endreß: Materna Information & Communications SE, Germania
Ulrich Kiesmüller: Simon-Marius-Gymnasium Gunzenhausen, Germania
Kirsten Schlüter: Bayerisches Staatsminiserium für Unterricht und Kultus, Germania
Margareta Schlüter: Universität Tübingen, Germania
Jacqueline Staub: Universität Trier, Germania
Michael Weigend: WWU Münster, Germania
Wilfried Baumann, Liam Baumann, Josefine Hiebler: Österreichische Computer Gesellschaft, Austria
Gerald Futschek: Technische Universität Wien, Austria
Zsuzsa Pluhár: ELTE Informatikai Kar, Ungheria

La versione online del concorso è stata creata su cuttle.org. Ringraziamo per la buona collaborazione:
Eljakim Schrijvers, Justina Dauksaite, Arjan Huijsers, Dave Oostendorp, Alieke Stijf, Kyra Willekes: cuttle.org, Olanda
Chris Roffey: UK Bebras Administrator, Regno Unito

Per il supporto durante le settimane del concorso ringraziamo:
Hanspeter Erni: Direttore scuola media di Rickenbach
Gabriel Thullen: Collège des Colombières, Versoix

Ringraziamo l'ETH per l'organizzazione e lo svolgimento della finale del Castoro:
Dennis Komm, Hans-Joachim Bückenhauer, Jan Lichensteiger, Moritz Stocker: ETH di Zurigo, Ausbildunges- und Beratungszentrum für Informatikunterricht
Per la correzione dei compiti finali:
Fiona Binder, Joel Birrer, Marlene Bötschi, Danny Camenisch, Gianluca Danieletto, Alexander Frey, Sven Grübel, Laure Guerrini, Charlotte Knierim, Richard Královič, Yanik Künzi, Kenli Lao, Sandro Marchon, Zoé Meier, Dario Näpfer, Kai Zürcher

Per la traduzione dei compiti finali in francese:
Jan Schönbächler: Lycée-Collège de l'Abbaye de St-Maurice

Christoph Frei: Chragokyberneticks (Logo Informatik-Biber Schweiz)
Andrea Leu, Maggie Winter, Lena Frölich: Senarclens Leu + Partner AG

Un ringraziamento speciale va ai nostri grandi sponsor Juraj Hromkovič, Dennis Komm, Gabriel Parriaux e la Fondazione Hasler. Senza di loro, questo concorso non esisterebbe.

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 2023 è stato organizzato dalla Società Svizzera per l'Informatica nell'Insegnamento (SSII) con il sostegno determinante della fondazione Hasler. Gli sponsor del concorso sono l'Ufficio per l'economia e il lavoro del Cantone di Zurigo e UBS.

Questo quaderno è stato creato il 10 gennaio 2024 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'01.12.2023.

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.

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 2023 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

Ogni categoria aveva quesiti classificati in tre livelli di difficoltà: facile, medio e difficile. 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 inizia 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. Questi quesiti presentavano livelli di difficoltà diversi nei vari gruppi di età.

Alcuni quesiti sono indicati come «bonus» per determinate categorie di età: non contano nel totale dei punti, ma vengono utilizzati come spareggio per punteggi identici in caso di qualificazione agli eventuali turni successivi.

Per ulteriori informazioni:

SVIA-SSIE-SSII Società Svizzera per l'Informatica nell'Insegnamento
Castoro Informatico
Masiar Babazadeh

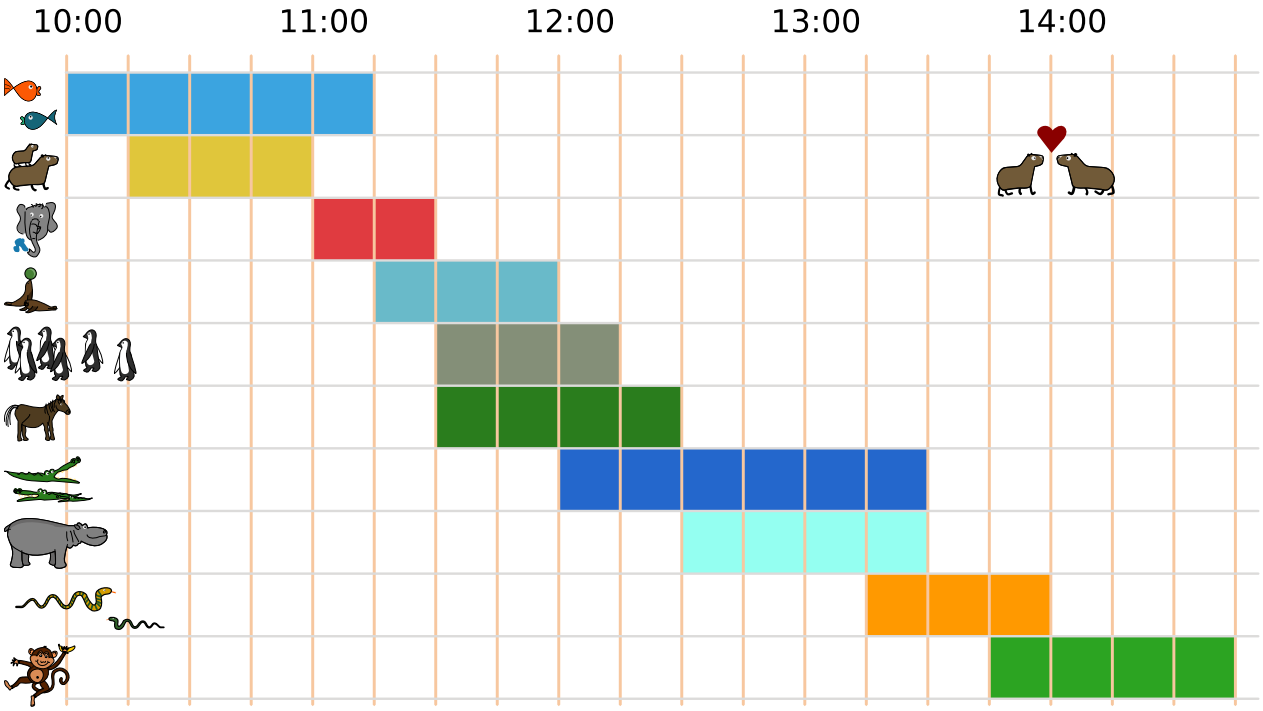
`https://www.castoro-informatico.ch/it/kontaktieren/`
`https://www.castoro-informatico.ch/`

Indice

1. Divertimento allo zoo

Oggi Anja è allo zoo. Vuole visitare il maggior numero possibile di spettacoli diversi.

Ecco un piano con tutti gli spettacoli. Ad esempio, dall'immagine possiamo vedere che lo spettacolo delle scimmie inizia alle 13:45 e termina alle 14:45.

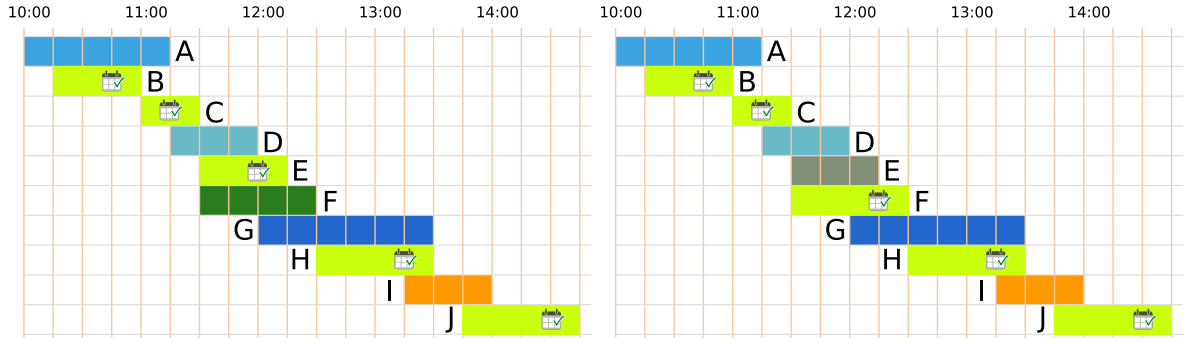


Anja assiste sempre a uno spettacolo dall'inizio alla fine. Puoi aiutare Anja?

Scegli il maggior numero possibile di spettacoli a cui Anja può partecipare uno dopo l'altro.

Soluzione

Anja può assistere a un massimo di 5 spettacoli consecutivi. Queste sono le due risposte corrette:



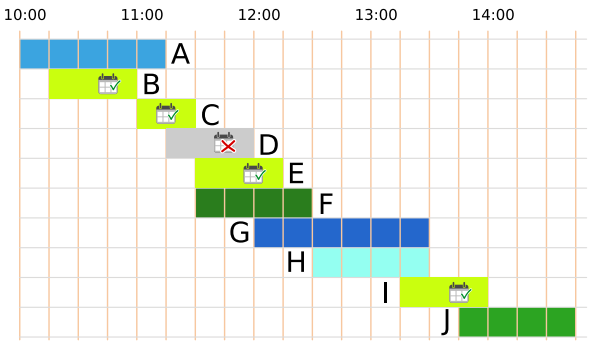
Ci sono diversi modi per trovare le risposte giuste.

Un piano di azione per Anja è una selezione di spettacoli ai quali può assistere uno dopo l'altro. Un modo per ottenere le risposte giuste è elencare tutti i piani di azione. In questo elenco, i piani con il maggior numero di spettacoli sono le risposte corrette. Purtroppo, trovare tutti i piani richiede molto tempo.

Ma non potrebbe esserci anche un programma di visite con 6 presentazioni? Cercheremo di crearne uno. Innanzitutto, diamo un'occhiata più da vicino alla durata degli spettacoli: Nel programma, l'intera giornata è suddivisa in 19 unità temporali di un quarto d'ora ciascuna. Gli spettacoli durano 2, 3, 4, 5 o 6 unità di tempo.

Unità di tempo	**Presentazioni**
2	C
3	B, D, E, I
4	F, H, J
5	A
6	G

Al fine di racchiudere il maggior numero possibile di presentazioni in un unico programma di visita, scegliamo presentazioni che siano il più breve possibile. Le 6 presentazioni più brevi durano complessivamente 18 unità di tempo $(2+3+3+3+4)$. Queste prestazioni brevi comprendono anche gli spettacoli C, D ed E. Tuttavia, poiché gli spettacoli C ed E sono esattamente uno dopo l'altro, Anja non può assistere allo spettacolo D nel mezzo.



Quindi dobbiamo sostituire la presentazione D con un'altra il più breve possibile. Rimangono solo le presentazioni con almeno 4 unità di tempo. Senza la presentazione D, abbiamo quindi bisogno di un totale di almeno 19 unità di tempo per 6 presentazioni: $2+3+3+3+4+4$. Ma entrambi gli spettacoli con 4 unità di tempo si sovrappongono sempre a uno spettacolo con 3 unità di tempo. Dovremmo sostituire anche questo con uno spettacolo di almeno 4 unità di tempo e quindi avremmo bisogno di almeno 20 unità di tempo per 6 spettacoli, ma ci sono solo 19 unità di tempo disponibili! Concludiamo che non può esistere un programma di visite che contenga più di 5 spettacoli.

Questa è l'informatica!

Questo compito contiene un programma di spettacoli allo zoo. Produrre programmi di questo tipo non è facile e in informatica si chiama *problema di programmazione*. Naturalmente, lo zoo vuole permettere ai suoi visitatori di vedere il maggior numero possibile di presentazioni, ma bisogna tenere conto anche di altre condizioni. Per esempio, le presentazioni possono essere offerte solo se i guardiani hanno tempo, se le arene disponibili sono libere e se gli spettacolli sono compatibili con i ritmi di vita degli animali.

Ci sono molti problemi simili nella vita ai quali si possono applicare le stesse considerazioni. Un esempio è la creazione di un orario scolastico o l'assegnazione dei film al cinema. La creazione di questi orari richiede così tanto tempo che anche per esempi relativamente piccoli (gli orari della vostra scuola) è spesso impossibile lavorare a mano. Anche i *processori* del computer devono svolgere molti compiti ed elaborarli uno dopo l'altro. Il sistema operativo crea, alla velocità della luce e senza che l'utente se ne accorga, la programmazione di quando un processore fa cosa. La programmazione è uno dei grandi temi dell'informatica, di cui la ricerca si occupa ancora oggi.

Parole chiave e siti web

- Scheduler: `https://it.wikipedia.org/wiki/Scheduler`
- Sistema operativo: `https://it.wikipedia.org/wiki/Sistema_operativo`

2. L'Ombrello di Anna

Questo è l'ombrello di Anna:

Una delle quattro immagini mostra l'ombrello di Anna. Quale?

A)

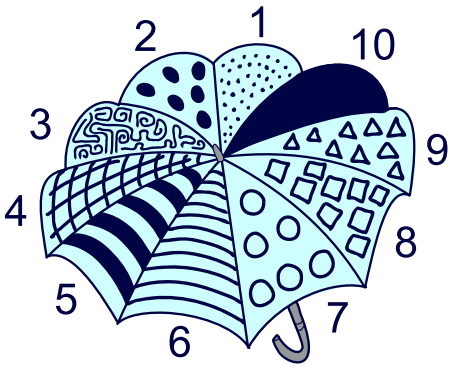
B)

C)

D)

Soluzione

Ogni motivo sull'ombrello di Anna si ripete esattamente una volta.



Per trovare l'immagine corretta, confrontiamo a turno ciascuna immagine con l'ombrello di Anna:

- scegliamo il motivo più a sinistra e troviamo la sua posizione sull'ombrello di Anna.
- verifichiamo che i motivi adiacenti siano uguali a quelli dell'ombrello di Anna.

	A)	**B)**	**C)**	**D)**
Immagini di risposta				
Ombrello di Anna				

Ciascuna delle quattro immagini mostra una sequenza di soli cinque motivi e non di tutti e dieci. Non possiamo sapere se la sequenza di motivi di una delle quattro immagini corrisponde alla sequenza completa di tutti e dieci i motivi dell'ombrello di Anna.

L'immagine C è l'unica con una sequenza di cinque motivi che corrisponde completamente a quelli dell'ombrello di Anna. Dunque solo l'immagine C può mostrare l'ombrello di Anna. Tutte le altre immagini hanno sequenze di motivi che non corrispondono o corrispondono solo parzialmente a quelli dell'ombrello di Anna. Quindi queste immagini non possono rappresentare l'ombrello di Anna.

Questa è l'informatica!

In ciascuna delle opzioni di risposta è mostrata solo una parte della sequenza di modelli. Anche se contengono solo *informazioni parziali*, possiamo determinare quale delle quattro immagini mostra l'ombrello di Anna: un'immagine mostra l'ombrello di Anna solo se la sequenza di modelli si verifica completamente nella sequenza di modelli dell'ombrello di Anna.

Per la ricerca in un documento di testo si applica lo stesso principio della «ricerca a ombrello». Il computer cerca le *stringhe di caratteri* corrispondenti nel documento con le informazioni parziali

fornite (parola di ricerca). Una stringa è una sequenza di caratteri (ad esempio lettere, numeri, caratteri speciali). Per la ricerca vale quanto segue:

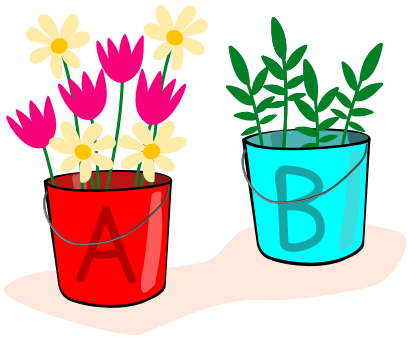
- Più lunga è la parola di ricerca, minore è il numero di possibili corrispondenze e maggiore è la possibilità di trovare il punto del documento che si sta cercando.
- Più breve è la parola chiave, maggiore è il numero di possibili corrispondenze che la ricerca produrrà e meno accurata sarà la ricerca.

Per migliorare la ricerca, sono state sviluppate diverse procedure di ricerca (o *Algoritmi di ricerca*). Sono progettate per eseguire una ricerca accurata nel più breve tempo possibile e fornire un risultato adeguato. Questi algoritmi di ricerca vengono costantemente sviluppati e sono in grado di cercare enormi quantità di dati in tempi molto brevi (ad esempio, i motori di ricerca su Internet utilizzano tali algoritmi di ricerca).

Parole chiave e siti web

- Stringa: `https://it.wikipedia.org/wiki/Stringa_(informatica)`
- Algoritmo di ricerca: `https://it.wikipedia.org/wiki/Algoritmo_di_ricerca`

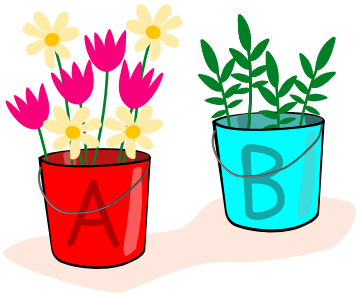
3. Bouquet

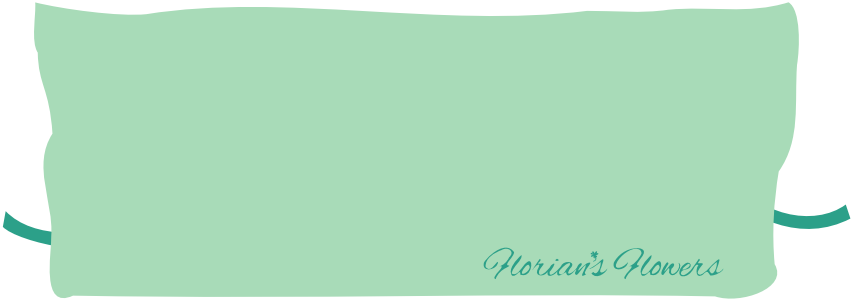


Florian vende mazzi di fiori. Florian lega ogni bouquet secondo queste istruzioni:

1. Prendere un primo fiore dal secchio A.
2. Se il primo fiore è una margherita, prendere un'altra margherita.
3. Poi prendere un rametto dal secchio B fino a formare un bouquet di 4 parti. Fatto!

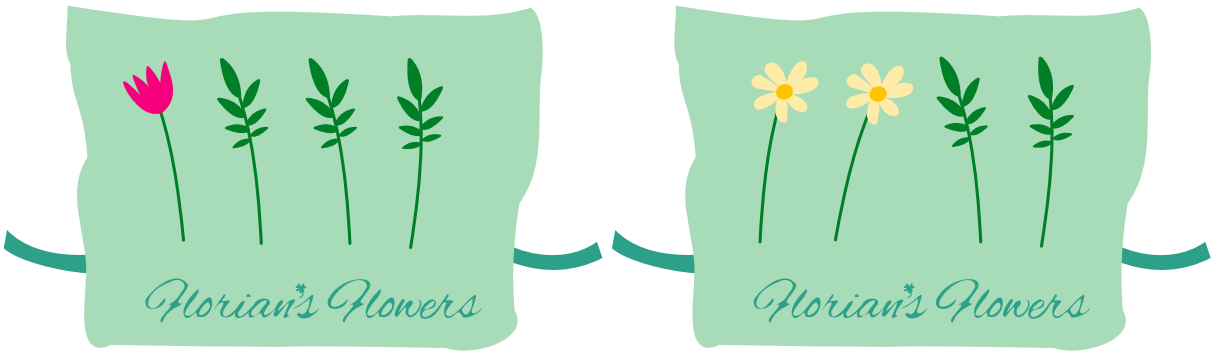
Aiuta Florian: segui le istruzioni e scegli fiori e rami per un bouquet.



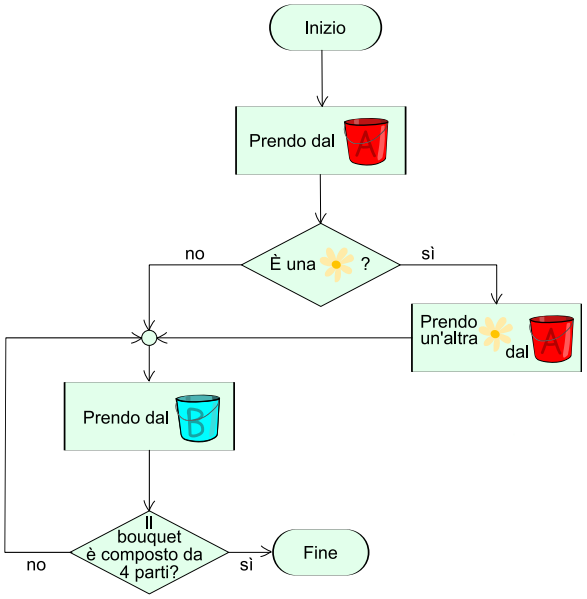


Soluzione

Esistono due soluzioni corrette:



Per legare correttamente i bouquet, Florian deve seguire le istruzioni. Possiamo anche descrivere le istruzioni con un diagramma:



Dopo che Florian ha scelto il primo fiore dal secchio A, segue una decisione che dipende dal primo fiore. O prende un'altra margherita (), o segue la freccia «no» e prende un rametto .

Poi controlla se ha già quattro parti. In caso contrario, segue la freccia «no» e deve prendere un altro rametto e poi controllare di nuovo il numero di parti.

Quindi, se prende prima una margherita , prenderà un'altra margherita e poi prenderà due volte un rametto . Ma se prende prima un tulipano , andrà direttamente a «scegliere dal secchio B» e prenderà un rametto dal secchio B fino ad avere 4 pezzi - quindi prenderà 3 rametti in totale.

Questa è l'informatica!

Le *istruzioni* per legare il bouquet sono chiare e potrebbero essere eseguite da una macchina. In informatica si parla di *algoritmo*. Le istruzioni utilizzano alcune istruzioni che sono comuni anche nei programmi per computer:

- La prima istruzione è una selezione casuale da un insieme di oggetti.
- La seconda istruzione si chiama *struttura condizionale* o *selezione*: perché si deve scegliere tra due o più possibilità.
- La terza istruzione sembra relativamente semplice, ma deve essere ben strutturata in un programma per computer. La parte interna dell'istruzione (essa stessa un'istruzione: «Prendi un rametto dal secchio B») deve essere eseguita più volte finché il bouquet non è composto da 4 parti. L'esecuzione dell'istruzione interna viene quindi ripetuta finché non viene soddisfatta la condizione «Il bouquet è composto da 4 parti». Una tale *iterazione* è chiamata anche *loop*.

Un algoritmo può essere rappresentato in modi diversi. In questo compito, l'algoritmo del «bouquet di fiori» di Florian è formulato come istruzioni in linguaggio naturale. Nella spiegazione della soluzione, viene presentato come un «diagramma di flusso del programma».

Il fiorista è un mestiere. Esistono tradizioni e regole su come legare un bouquet o una corona di fiori. Questo è un esempio di come le istruzioni o gli algoritmi esistano in molti settori della vita, non solo nell'informatica.

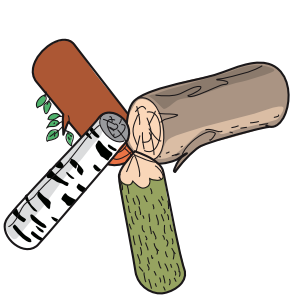
Parole chiave e siti web

- Selezione: `https://it.wikipedia.org/wiki/Selezione_(informatica)`
- Iterazione: `https://it.wikipedia.org/wiki/Iterazione`
- Pseudocodice: `https://it.wikipedia.org/wiki/Pseudocodice`
- Diagramma di flusso: `https://it.wikipedia.org/wiki/Diagramma_di_flusso`

4. Foto

Il castoro ha scattato una foto.

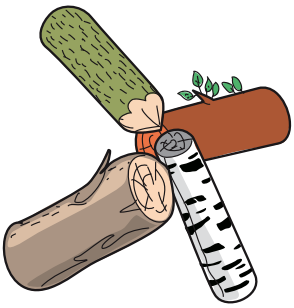
Di quale delle quattro foto si tratta?



A)

B)

C)



D)

Soluzione

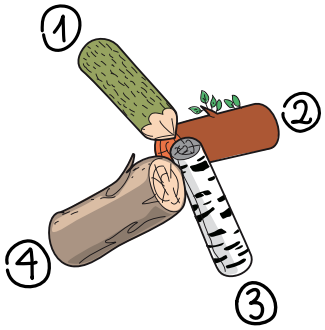
La risposta corretta è D). 

I tronchi d'albero fotografati dal castoro sono disposti in cerchio. Per scoprire quale foto è quella giusta, osserviamo l'ordine dei tronchi in questa disposizione. Scegliamo un tronco (ad esempio quello appuntito) e gli diamo il numero 1. Poi determiniamo quale tronco si trova a sinistra e gli diamo il numero 2. Procediamo così finché tutti i tronchi hanno un numero. Nella situazione fotografata dal castoro, i tronchi sono in quest'ordine: 1 (tronco appuntito) - 2 (tronco marrone con foglie) - 3 (tronco di betulla) - 4 (tronco marrone spesso).

Ora guardiamo la sequenza dei tronchi nelle foto da A a D. Iniziamo come sopra con il tronco affilato 1 e andiamo sempre verso sinistra:

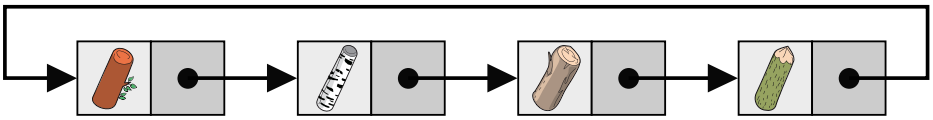
- Foto A: 1 – 3 – 2 – 4
- Foto B: 1 – 4 – 3 – 2
- Foto C: 1 – 3 – 4 – 2
- Foto D: 1 – 2 – 3 – 4

Solo la foto D mostra l'ordine corretto.



Questa è l'informatica!

In questo compito, si considera l'ordine dei tronchi d'albero. Ciò che è possibile con pochi *elementi* (qui quattro tronchi d'albero) semplicemente «guardando» e confrontando le coppie vicine, richiede una procedura automatizzata per problemi con molti più elementi. In un programma informatico che deve elaborare elementi vicini, gli elementi possono essere memorizzati in una struttura di dati adatta, come una lista concatenata:



In una *lista concatenata*, ogni dato è memorizzato in un singolo nodo. Inoltre, in ogni nodo è memorizzato un *riferimento* al nodo successivo della lista. Se l'ultimo nodo contiene un riferimento al primo nodo, si tratta di una struttura di dati anulare. Questo è importante nell'esempio, in modo da poter iniziare da qualsiasi tronco d'albero e scorrere la lista.

Parole chiave e siti web

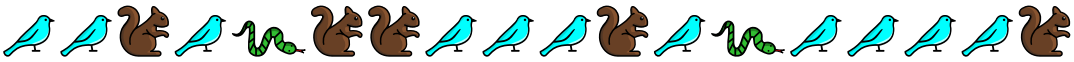
- Lista concatenata: `https://it.wikipedia.org/wiki/Lista_concatenata`

5. L'albero magico

Ben ha un albero di mele speciale in giardino:

- Se un uccello  atterra sull'albero, crescono immediatamente due nuove mele.
- Se uno scoiattolo  si arrampica sull'albero, cade una mela. Se non c'è nessuna mela appesa all'albero, non succede nulla.
- Se un serpente  visita l'albero, tutte le mele scompaiono immediatamente.

Questa mattina ci sono 25 mele appese all'albero. Poi alcuni animali visitano l'albero uno dopo l'altro, per ultimo uno scoiattolo. Ben ha scritto esattamente il loro ordine:



Quante mele sono rimaste appese all'albero?

A) 3 mele

B) 7 mele

C) 17 mele

D) 31 mele

Soluzione

La risposta B è corretta. Dopo che l'ultimo scoiattolo si è arrampicato sull'albero, ci sono ancora 7 mele appese all'albero.

È possibile calcolare per ogni visita animale quante mele sono appese all'albero in quel momento:

Animale:	**Inizio**					
Istruzione:	–	+2	+2	−1	+2	reset
Mele:	25	27	29	28	30	0

Animale:	**Riporto**								
Istruzione:	–	–	–	+2	+2	+2	−1	+2	reset
Mele:	0	0	0	2	4	6	5	7	0

Animale:	**Riporto**					
Istruzione:	–	+2	+2	+2	+2	−1
Mele:	0	2	4	6	8	**7**

Poiché tutte le mele scompaiono immediatamente quando un serpente visita l'albero, possiamo ignorare tutto ciò che accade prima dell'arrivo del secondo (e ultimo) serpente. Come mostrato nella tabella, quattro uccelli atterrano sull'albero dopo la visita dell'ultimo serpente. Successivamente, ci sono $4 \times 2 = 8$ mele appese all'albero. Poi uno scoiattolo si arrampica sull'albero e fa cadere una mela, lasciando $8 - 1 = 7$ mele.

Questa è l'informatica!

La visita di un animale cambia le condizioni dello speciale albero di mele, ma solo in un modo molto specifico: cambia solo il numero di mele appese all'albero. La visita di un animale non ha alcuna influenza su altre proprietà dell'albero, come il numero di foglie, la lunghezza dei singoli rami o la forma della chioma. Per questo compito è quindi sufficiente osservare il numero di mele.

Anche un programma per computer ha uno stato che viene modificato dalle singole istruzioni del programma. I dati memorizzati da un programma sono solitamente considerati lo stato; il programma memorizza questi dati nelle *variabili* introdotte durante la programmazione.

La sequenza di visite degli animali all'albero in questo compito del castoro è come un programma per computer: ogni visita degli animali è un'istruzione che cambia lo stato dell'albero di mele. Questo stato - cioè il numero di mele, vedi sopra - può essere memorizzato in un'unica variabile.

Durante la risoluzione del compito, avrete notato che non avete dovuto esaminare l'intero «programma», ma solo la parte successiva all'ultima occorrenza del serpente. Osservando da vicino gli effetti delle singole istruzioni sullo stato del programma, siete riusciti a scoprire una proprietà speciale

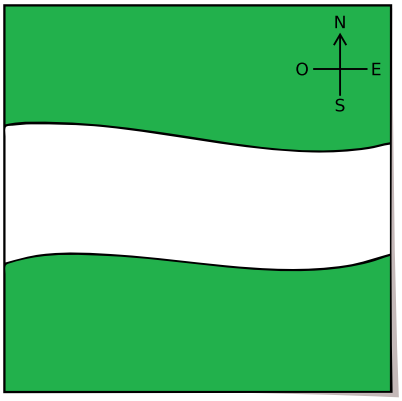
del programma. Questa analisi dei programmi (informatici) è una delle attività più frequenti degli informatici.

Parole chiave e siti web

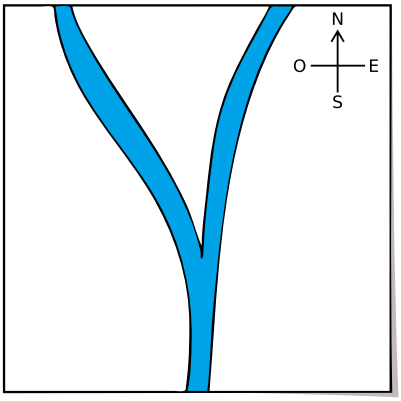
- Variabile: `https://it.wikipedia.org/wiki/Variabile_(informatica)`

6. La casa dei sogni di Karla

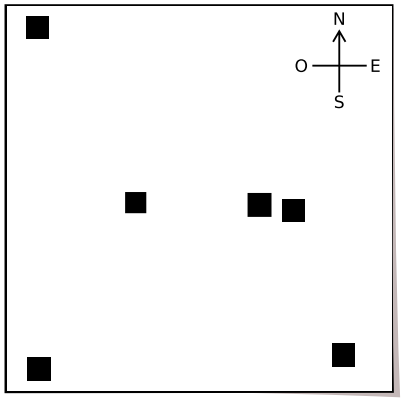
Karla ha tre mappe che mostrano esattamente la stessa area. Una mappa mostra le foreste, una i fiumi e una le case della zona. La casa dei sogni di Karla si trova nella foresta e vicino a un fiume.



Mappa delle foreste



Mappa dei fiumi

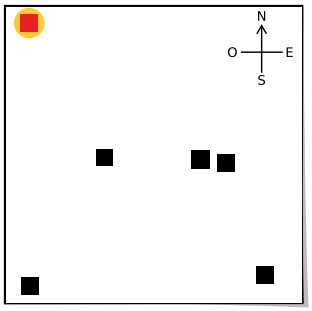


Mappa delle case

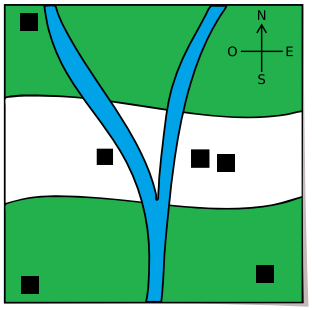
Qual è la casa dei sogni di Karla?

Soluzione

La casa in alto a sinistra della mappa è la casa dei sogni di Karla:



Per trovare la casa dei sogni di Karla, è necessario valutare le informazioni contenute in tutte e tre le mappe. La casa dei sogni deve trovarsi in una zona boscosa e vicino a un fiume. Questo vale solo per la casa in alto a sinistra. Questo è facile da vedere quando le carte vengono messe una sopra l'altra:



Questa è l'informatica!

Quando le informazioni sulle foreste, sui fiumi e sulle case sono presentate su un'unica mappa è facile trovare la casa che si sta cercando.

Un *geoinformation system* (GIS, sistema informativo geografico) riunisce una serie di informazioni spaziali (ad esempio, foreste, strade, confini nazionali, stazioni di servizio, municipi, pianure alluvionali, ecc.). Un GIS serve quindi alla visualizzazione e all'analisi dei cosiddetti *geodati*. Con l'aiuto di un GIS è possibile, ad esempio, per gli addetti al controllo delle catastrofi, compilare informazioni per i piani di evacuazione.

L'uso di più livelli con diverse informazioni sull'immagine è noto anche nei programmi di grafica. Una questione importante è sempre quale livello, con gli oggetti che contiene, è il più alto e quindi viene visualizzato in primo piano. Nell'esempio, la mappa delle case dovrebbe essere il livello superiore, in modo che le case non siano nascoste dalle aree boschive.

Parole chiave e siti web

- GIS: `https://it.wikipedia.org/wiki/Geographic_information_system`

7. Ricca

Evelyn ha cinque foto dei Ricca. Descrive con delle frasi il loro aspetto.

La sua amica Lydia le mostra una sesta foto di una Ricca:

Ora Evelyn si rende conto di una cosa: una delle sue frasi sui Ricca è sicuramente sbagliata.

Quale di queste frasi sui Ricca è sicuramente sbagliata?

A) Tutti i Ricca hanno i denti.

B) Alcuni Ricca hanno le ali.

C) I Ricca hanno o corna o tre occhi, ma mai corna *e* tre occhi.

D) Se i Ricca hanno esattamente due braccia, allora hanno anche esattamente due gambe.

Soluzione

La risposta D) è corretta: *Se i Ricca hanno esattamente due braccia, allora hanno anche esattamente due gambe.*

La risposta A) contiene un'affermazione che deve valere per tutti i Ricca. Se anche un solo Ricca non avesse i denti, l'affermazione sarebbe falsa. Tuttavia, tuttii e sei i Ricca che Evelyn conosce ora hanno i denti. Quindi la frase di Evelyn non può essere sicuramente sbagliata.

La risposta B) contiene un'affermazione che dovrebbe valere solo per alcuni Ricca. Poiché uno dei sei Ricca che Evelyn conosce ora ha le ali, la frase è corretta per i sei Ricca. Ma anche se nessuno dei sei Ricca avesse le ali, altri Ricca potrebbero averle e la frase sarebbe comunque vera. La frase può essere sicuramente falsa solo se Evelyn conosceva tutti i Ricca e nessuno aveva le ali.

La risposta C) collega due affermazioni con «o» e «o». L'affermazione collegata è vera se è vera esattamente una delle due affermazioni. Questo è il caso di tutti e sei i Ricca: quattro Ricca hanno le corna ma non tre occhi,gli altre due Ricca non hanno le corna ma tre occhi. Affinché la frase sia falsa, dovrebbe esserci almeno un Ricca con tre occhi e corna o un Ricca senza corna e con un numero diverso da tre occhi. Tra i sei Ricca che Evelyn conosce ora, non c'è nessun Ricca di questo tipo. Quindi la frase è corretta per i sei Ricca, e non certamente sbagliata.

Rimane la frase della risposta D). È formulata nella forma di un'affermazione «se» e «allora». Se la condizione «se» è vera, deve essere vera anche l'affermazione «allora». La condizione è vera per tutte le sei Ricca che Evelyn conosce: tutti hanno esattamente due bracci. Anche tutte le Ricche delle prime cinque immagini di Evelyn hanno esattamente due gambe; quindi per loro la frase di Evelyn è vera. Tuttavia, la Ricca nella foto di Lydia ha più di due gambe, cioè cinque. Pertanto, la frase è sicuramente sbagliata.

Questa è l'informatica!

Il numero di ali, braccia gambe e occhi e il fatto che i Ricca abbiano o meno i denti o le ali sono *caratteristiche* dei Ricca. Quando si descrivono i Ricca, si formulano delle «affermazioni» su queste proprietà. Questo porta a un *modello* di ciò che i Ricca sono.

Anche i computer hanno molti modelli. Alcuni sono formulati esplicitamente, come ad esempio un modello di studenti composto da nome, data di nascita e indirizzo di casa in un database. Altri modelli sono formati dai computer a partire dai dati, ad esempio quando vengono fornite immagini da confrontare per l'addestramento di una rete neurale.

Le frasi di Evelyn - cioè il suo modello dei Ricca in questo compito - sono formulate come *espressioni logiche*. Alcune hanno dei *quantificatori* («(per) tutti» o «ci sono»/«alcuni»), altre usano degli *operatori logici* («o»-«o» o «se»-«allora»). Queste espressioni logiche sono *formalizzate*: cioè, c'è una specificazione di come usarle e di cosa significano.

Sulla base di queste specifiche

- Le espressioni (semplici) possono essere collegate a espressioni più complesse con l'aiuto di quantificatori e operatori.
- il significato delle espressioni più complesse può essere calcolato a partire da quelle più semplici.

Le espressioni logiche sono un metodo comune per descrivere i modelli in informatica.

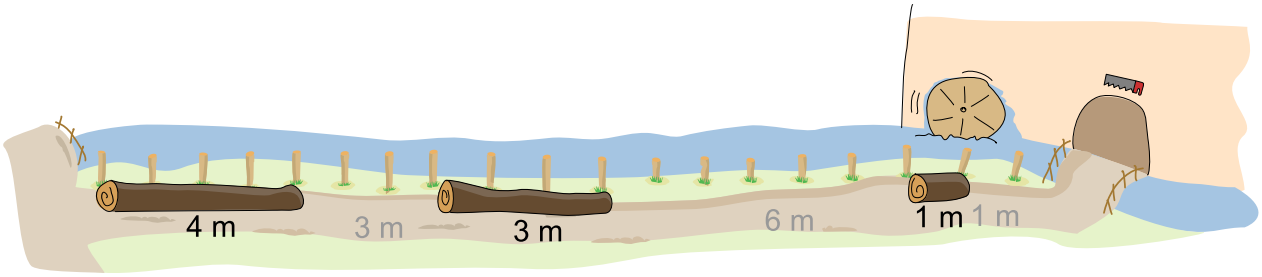
Parole chiave e siti web

Apprendimento automatico: `https://it.wikipedia.org/wiki/Apprendimento_automatico`

8. La Segheria di Timea

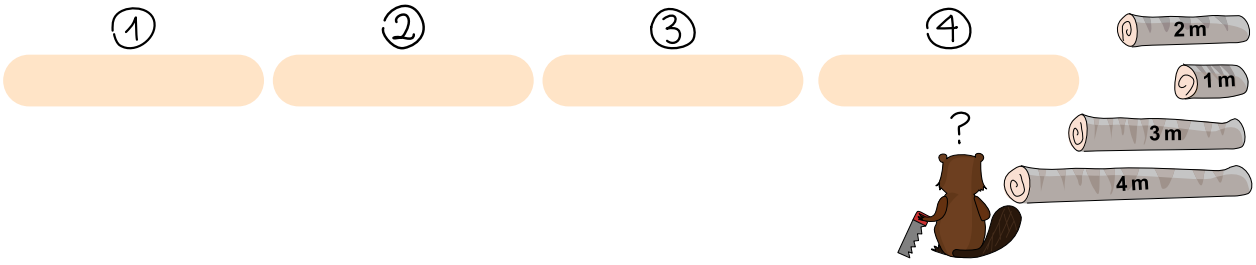
La castora Timea taglia tronchi di diverse lunghezze e poi li vende. Non appena taglia un tronco, lo posa sul sentiero lungo 18 metri. Timea osserva la seguente regola: colloca il tronco nel primo spazio da sinistra nel quale il tronco si inserisce.

Vende alcuni tronchi. Dopo di che, ci sono tre spazi vuoti sul sentiero:



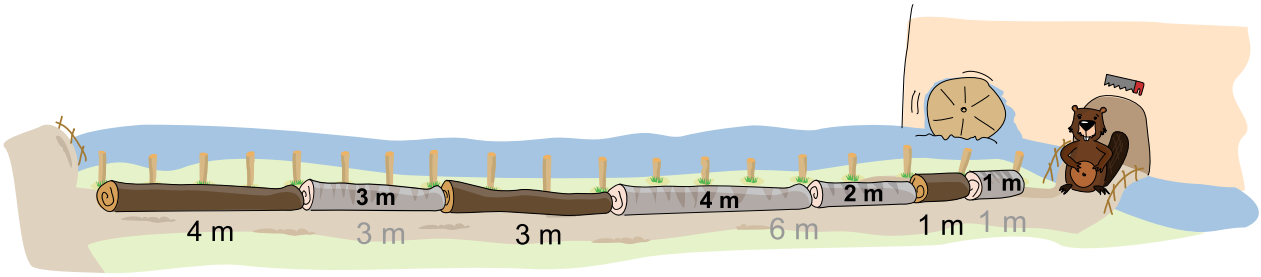
Ora Timea vuole tagliare quattro tronchi di lunghezza pari a 1, 2, 3 e 4 metri.

In quale ordine Timea deve tagliare i tronchi per poterli inserire tutti e quattro negli spazi vuoti?



Soluzione

La risposta corretta:



Se Timea taglia i tronchi nell'ordine (3 m, 4 m, 2 m, 1 m), tutti si inseriscono nel percorso: per il tronco di 3 m, la fessura di 3 m all'estrema sinistra è la prima fessura libera da sinistra in cui il tronco si inserisce; Timea lo colloca lì. Il tronco di 4 m va poi nella fessura di 6 m a sinistra. La fessura rimanente di 2 m è la prima fessura libera da sinistra; il tronco successivo vi si inserisce e Timea colloca l'ultimo tronco nella fessura di 1 m.

Altre sequenze corrette sono (3 m, 2 m, 4 m, 1 m) e (4 m, 3 m, 2 m, 1 m).

Tutte le altre sequenze fanno sì che Timea non riesca a posare tutti i tronchi: Il tronco da 1 m deve essere sempre l'ultimo della fila perché solo questo tronco può riempire l'ultimo spazio libero. Il tronco da 2 m non deve precedere il tronco da 3 m, perché altrimenti verrebbe posizionato nello spazio di 3 m, creando un secondo spazio di 1 m. Oltre alle tre sequenze citate, non esistono sequenze che soddisfino queste condizioni.

Questa è l'informatica!

Questo compito è un caso speciale del problema *dei contenitori*. Nel problema dei contenitori, oggetti di dimensioni diverse devono essere collocati in un certo numero di contenitori, che a loro volta possono avere dimensioni diverse. In questo caso gli oggetti sono i tronchi d'albero, i contenitori gli spazi vuoti del sentiero.

Il problema si presenta in ambiti molto diversi della vita. Alcuni esempi: (a) In un magazzino di mobili, i mobili piccoli e grandi devono essere immagazzinati in modo da risparmiare spazio. (b) Un'azienda di trasporti può risparmiare denaro se ha bisogno di meno camion per trasportare le merci grazie a un imballaggio intelligente. (c) Il sistema operativo di un computer deve memorizzare file di dimensioni diverse sul disco rigido. Quando i file vengono cancellati, sul disco fisso compaiono degli spazi vuoti. Questi spazi vuoti devono essere riempiti in modo da non sprecare spazio di archiviazione, proprio come avviene sul sentiero in questo compito.

In informatica, il problema dei contenitori è considerato uno dei problemi più difficili; soluzioni ottimali garantite possono essere risolte da programmi informatici solo per casi piccoli con pochi oggetti e pochi contenitori. Tuttavia, esistono diversi metodi e strategie che possono essere utilizzati per determinare buone soluzioni al problema dei contenitori. In questo compito, la strategia è data dalla regola di Timea. Essa colloca sempre ogni tronco nel primo spazio da sinistra in cui si inserisce.

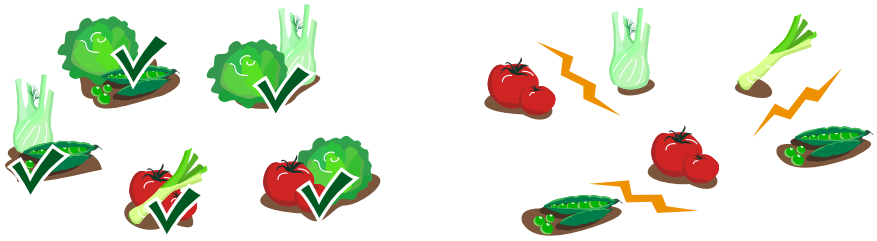
Questa strategia è chiamata *First Fit*. Dall'esempio di questo compito si può vedere che questa strategia può portare a cattivi risultati: solo se i tronchi vengono posizionati in un certo ordine è possibile riempire il tutto.

Parole chiave e siti web

- Gestione della memoria: `https://it.wikipedia.org/wiki/Gestione_della_memoria`
- Frammentazione: `https://it.wikipedia.org/wiki/Frammentazione_(informatica)`

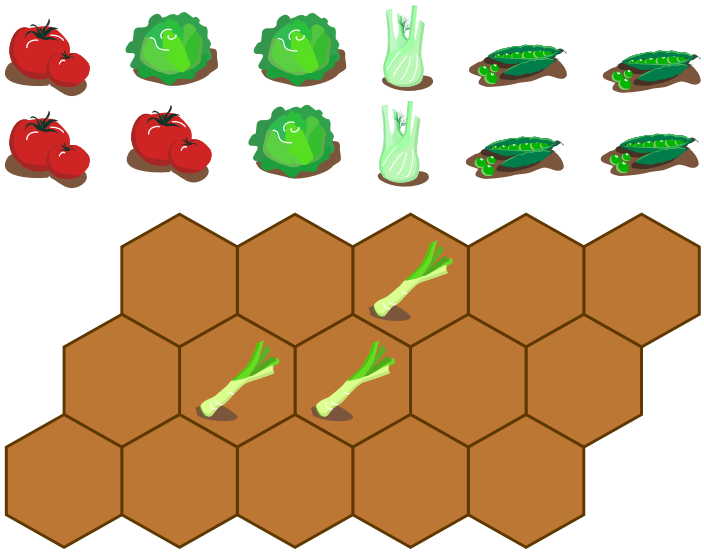
9. Orto di Lisa

Lisa crea un orto. Vuole piantare cinque ortaggi diversi. Alcuni ortaggi vanno d'accordo tra loro ✔, altri no ↯:



Lisa ha diviso l'orto in aree esagonali. Vuole piantare esattamente un ortaggio in ogni area.

Lisa ha già piantato porri  in tre aree.

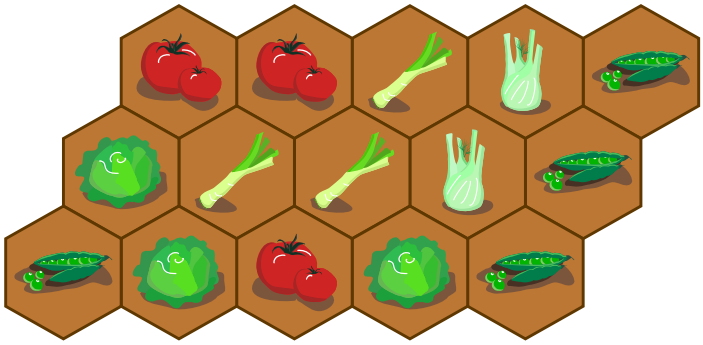


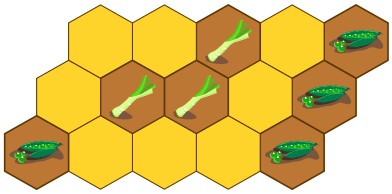
Quando si pianta, Lisa osserva la seguente regola: gli ortaggi che non vanno d'accordo non devono essere piantati in zone che si toccano.

Pianta tutte le aree ancora libere seguendo la regola di Lisa!

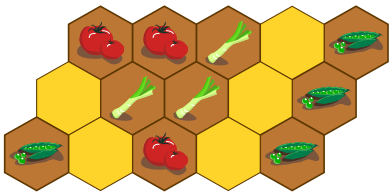
Soluzione

La risposta gista:

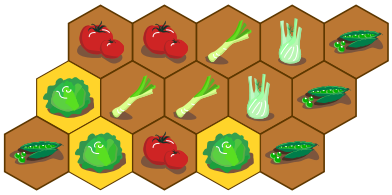




Poiché i piselli non vanno d'accordo con i porri, Lisa non pianta i piselli nelle aree chiare. Solo le aree rimanenti rimangono per i piselli.



Poiché i pomodori non vanno d'accordo con i piselli, Lisa non pianta i pomodori nelle aree chiare. Può piantare i pomodori nelle altre zone; i pomodori vanno d'accordo con i porri.



Poiché i pomodori non vanno d'accordo con i finocchi, Lisa non li pianta nelle aree chiare. Può piantare il finocchio nelle due aree tra i porri e i piselli. Può piantare la lattuga nelle aree chiare: Lisa non è a conoscenza di alcuna discordanza tra gli ortaggi vicini e la lattuga.

Questa è l'informatica!

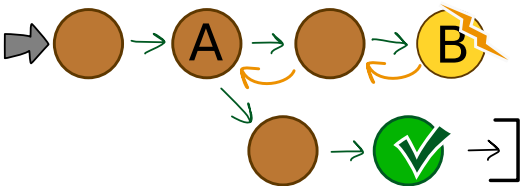
Se si vuole piantare ortaggi in modo che il raccolto sia il più abbondante possibile, si deve osservare molte *condizioni*: Ad esempio, le singole varietà hanno esigenze diverse in termini di spazio, nutrienti e luce. In questo compito consideriamo solo un tipo di condizione: la compatibilità tra le varietà di ortaggi.

Per trovare un piano per l'orto di Lisa che rispetti tutte le condizioni di compatibilità, si potrebbe procedere in questo modo: si provano sistematicamente tutte le combinazioni per disporre gli ortaggi sull'orto. Solo quando l'orto è pieno, si verifica se questa combinazione soddisfa tutte le condizioni ed è una soluzione al problema di Lisa. In informatica, tale prova di tutte le combinazioni è nota come metodo *forza bruta*. Per problemi con molte combinazioni e poche soluzioni, procedere secondo questo metodo può richiedere molto tempo.

Pertanto, di solito è meglio procedere per gradi e considerare tutte le condizioni a ogni passo. In questo modo possiamo trovare la soluzione al problema di Lisa, una combinazione o una piantumazione «sbagliata» dell'orto infatti non può verificarsi.

Fortunatamente, la soluzione si può trovare in modo diretto: ci sono sempre aree in cui possiamo piantare alcuni degli ortaggi rimasti. Questo di solito non funziona sempre.

Se si cerca di assemblare la soluzione passo dopo passo, ci possono essere diverse possibilità di soddisfare tutte le condizioni in un unico passo A.



A seconda della scelta, in una fase successiva B potrebbero non esserci più opzioni. Quindi si fanno gli ultimi passi indietro fino ad arrivare al passo A con diverse possibilità. A questo punto si sceglie un'altra possibilità e si cerca di trovare una soluzione.

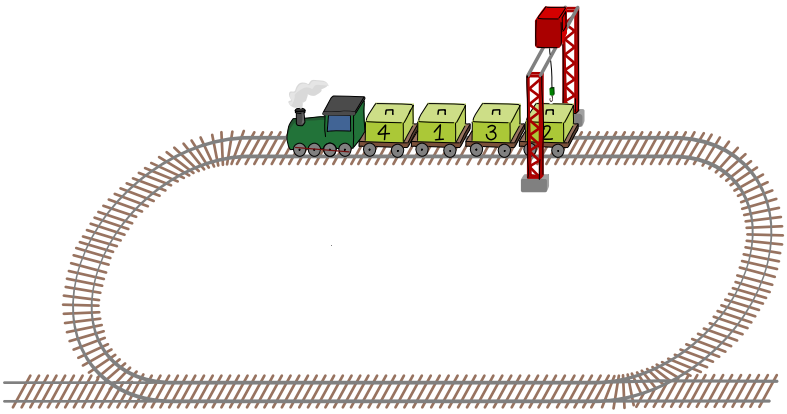
In informatica, questo ritorno sui propri passi è noto come *backtracking*.

Parole chiave e siti web

- Metodo forza bruta: `https://it.wikipedia.org/wiki/Metodo_forza_bruta`
- Backtracking: `https://it.wikipedia.org/wiki/Backtracking`

10. Scarico del treno

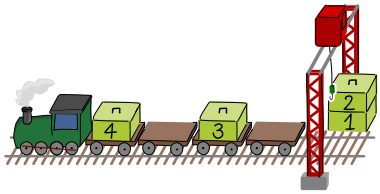
Un treno traina vagoni con casse numerate. La gru si trova in una posizione fissa e scarica le casse. Per scaricare una cassa, questa deve essere posizionata direttamente sotto la gru.



La gru deve scaricare le casse, partendo da 1, in ordine crescente. Il treno può andare solo in avanti. Quando è passato sotto la gru, deve fare un giro per poter scaricare altre casse.

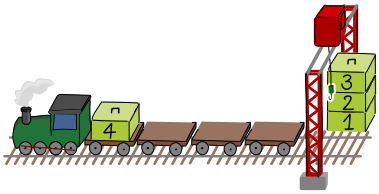
In questo modo la gru scarica le casse 1, 2, 3 e 4 nell'ordine corretto:

Turno 1:



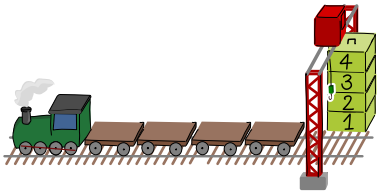
Salta la cassa 4, scarica la cassa 1, salta la cassa 3 e scarica la cassa 2 2.

Turno 2:



Salta la cassa 4 e scarica la cassa 3.

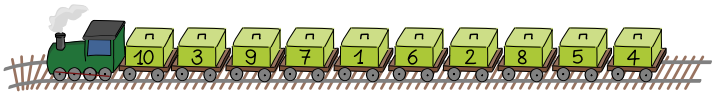
Turno 3:



Scarica la cassa 4.

Quindi il treno deve percorrere tre giri affinché tutte le casse siano scaricate nell'ordine corretto.

Quanti turni sono necessari per scaricare il seguente treno?



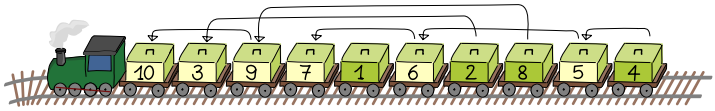
A) 1 turno
B) 2 turni
C) 3 turni
D) 4 turni
E) 5 turni
F) 6 turni
G) 7 turni
H) 8 turni
I) 9 turni
J) 10 turni

Soluzione

La risposta corretta è 7 turni.

L'ordine prescritto per lo scarico è 1, 2, 3, 4, 5, 6, 7, 8, 9, 10. Al primo turno, la gru scarica le casse 1 e 2 insieme. Nel secondo turno, la gru scarica insieme 3 e 4, poi 5, poi 6, poi 7 e 8 insieme, poi 9 e infine 10. Questo corrisponde a 7 turni.

In alternativa, si può sfruttare il fatto che ogni volta che viene richiesto il numero di casella successivo a sinistra di uno dei numeri di casella della sequenza, è necessario un ulteriore giro di scarico.



Ad esempio, poiché il 3 si trova a sinistra del 2, viene saltato per scaricare il 2, quindi è necessario un giro supplementare per portare il 3 sotto la gru. Nella mossa data ci sono sei coppie di questo tipo (2,3), (4,5), (5,6), (6,7), (8,9) e (9,10), quindi sono necessari altri 6 turni, per un totale di 7 turni.

Questa è l'informatica!

Se per un numero qualsiasi della sequenza 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 la cassa con il numero successivo più grande si trova più a sinistra sul treno, si parla di *inversione*. Ogni inversione di questo tipo richiede un giro in più. Se contiamo il numero di inversioni, otteniamo la risposta.

Il conteggio delle inversioni rispetto a una sequenza desiderata ha molte applicazioni. In alcuni algoritmi di ordinamento, come il *bubble sort*, il numero di inversioni ci dice quante permutazioni sono necessarie per ordinare una particolare sequenza. Quando due clienti classificano lo stesso insieme di articoli, il numero di inversioni nelle loro classifiche ci dice quanto le loro preferenze siano simili. Questa funzione viene utilizzata dai negozi online per identificare i clienti «simili» e consigliare loro i prodotti.

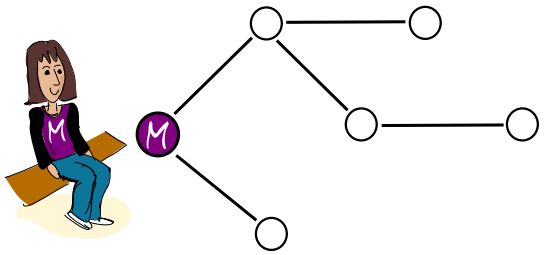
Parole chiave e siti web

- Algoritmo di ordinamento: `https://it.wikipedia.org/wiki/Algoritmo_di_ordinamento`
- Bubble sort: `https://it.wikipedia.org/wiki/Bubble_sort`

11. Il villaggio di Martina

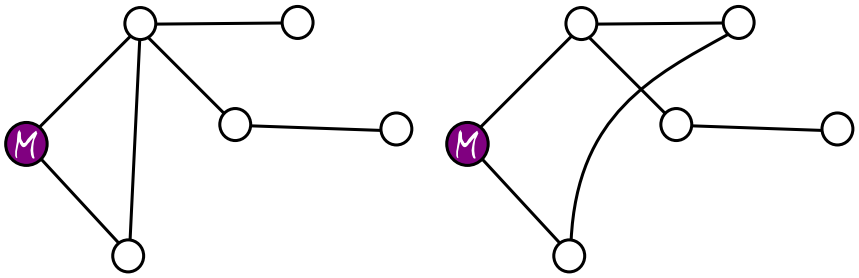
Nel villaggio di Martina ci sono sei case. Ci sono anche sentieri che possono essere utilizzati per camminare da una casa all'altra. Martina ha bisogno della stessa quantità di tempo per tutti questi percorsi.

Martina ha disegnato una mappa speciale del villaggio. In essa ha disegnato i percorsi esatti che può utilizzare per raggiungere le altre case.



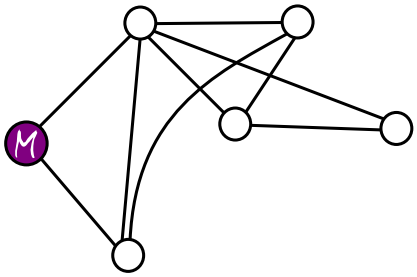
C'è anche una vera e propria mappa del villaggio, con tutti i sentieri.

Quale di questi disegni non può essere la mappa corretta?

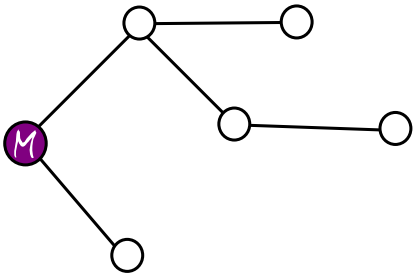


A)

B)

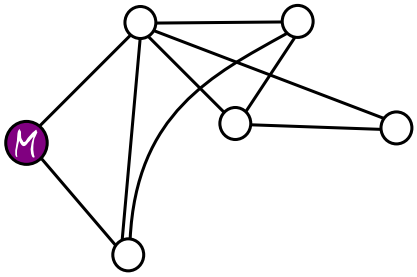


C)



D)

Soluzione

La risposta C è corretta: 

La mappa di Martina mostra che il modo più veloce per raggiungere la casa all'estrema destra è attraverso tre percorsi. Se C fosse la mappa giusta del villaggio, Martina potrebbe raggiungere questa casa più velocemente, cioè attraverso due sentieri. Quindi C non può essere la mappa giusta del villaggio.

Con le mappe A, B e D non c'è modo di raggiungere una delle altre case più velocemente che attraverso i percorsi della mappa speciale di Martina. Quindi queste mappe potrebbero essere vere e proprie mappe del villaggio.

Questa è l'informatica!

Martina è un'esperta di informatica. Ha disegnato la sua mappa come un *grafo*. I grafi sono costituiti da *nodi* (qui le case) che possono essere collegati da *bordi* (qui i percorsi). Sono adatti a modellare la realtà in molte aree dell'informatica e anche in questo compito.

Martina sa che esiste un'intera gamma di algoritmi per i grafi, ad esempio la cosiddetta ricerca in ampiezza, per risolvere compiti come «Qual è il modo più veloce per raggiungere un'altra casa?». Forse ha creato la sua particolare mappa del villaggio utilizzando una ricerca in ampiezza su un grafo più grande, la vera mappa del villaggio con tutti i percorsi.

Nella teoria dei grafi, che si occupa di grafi e algoritmi di grafi, la mappa di Martina corrisponde a un sottografo della mappa complessiva del villaggio. Il sottografo di Martina ha due caratteristiche particolari:

- Tutti i nodi sono collegati direttamente (tramite un bordo) o indirettamente (tramite più bordi).
- Non importa quali due nodi si scelgano a caso, esiste sempre un solo percorso tra i due.

Un grafo con queste caratteristiche è chiamato *albero* in informatica. La casa di Martina rappresenta la *radice* dell'albero. Dalla radice, Martina può raggiungere tutti gli altri nodi (le altre case del villaggio) lungo un unico percorso. Il sottografo di Martina è quindi un albero; inoltre, contiene tutti i nodi dell'intero grafo (l'intera mappa del villaggio) - ma forse non tutti i bordi. Un sottografo con queste proprietà è chiamato *albero ricoprente* dell'intero grafo.

In informatica ci sono molte applicazioni per gli algoritmi a grafo, soprattutto nel contesto delle reti (reti di traffico, reti di telecomunicazione, ...), ad esempio nel calcolo dei percorsi nei sistemi di

navigazione. Gli alberi ricoprenti possono essere utilizzati per la costruzione di reti a basso costo e possono essere utili per risolvere problemi particolarmente difficili.

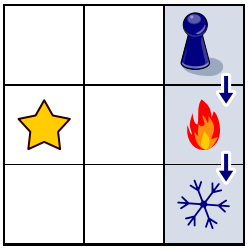
Parole chiave e siti web

- Grapho: `https://it.wikipedia.org/wiki/Grafo`
- Albero: `https://it.wikipedia.org/wiki/Albero_(grafo)`
- Ricerca in ampiezza: `https://it.wikipedia.org/wiki/Ricerca_in_ampiezza`
- Albero ricoprente: `https://it.wikipedia.org/wiki/Albero_ricoprente`

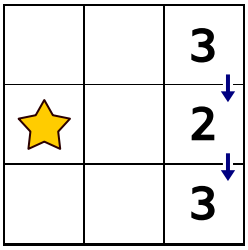
12. Più caldo, più freddo

Nina e Daniel giocano alla caccia al tesoro. Su una tavola con quadrati, Nina seleziona un quadrato e lo tiene a mente. Il tesoro è nascosto lì.

Daniel sceglie un campo di partenza. Da lì, sposta il suo pezzo da gioco di uno spazio alla volta: a sinistra, a destra, in alto o in basso.



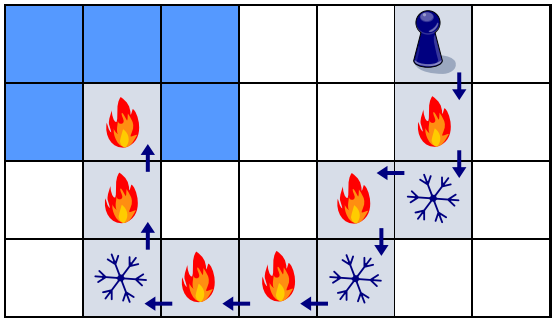
Al primo tentativo, prendono un piccolo tabellone di gioco. Nina nasconde il tesoro nella casella con la stella. Daniel inizia in alto a destra e fa due passi lungo le frecce. Dopo ogni passo, Nina dice se Daniel è più vicino al tesoro o più lontano dal tesoro rispetto a prima del passo.



L'immagine a destra mostra le distanze di Daniel dal tesoro. La distanza dal tesoro è il minor numero di passi che Daniel potrebbe attualmente compiere per raggiungere il tesoro.

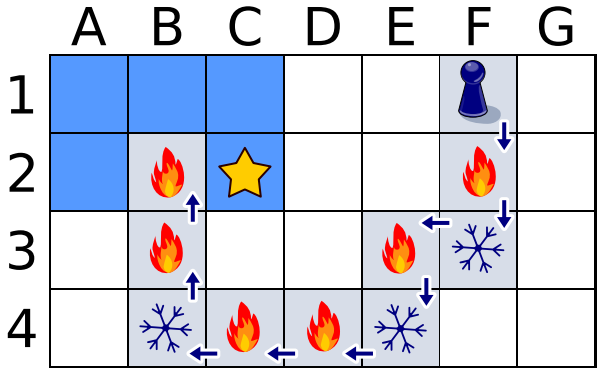
Adesso prendono una tavola più grande. Nina nasconde il tesoro in uno dei campi contrassegnati in blu. L'immagine mostra nuovamente i passi di Daniel e ciò che Nina dice dopo ogni passo.

Dove è nascosto il tesoro?



Soluzione

La risposta corretta:



Seguiamo il percorso di Daniel e il riscontro di Nina. Daniel inizia nella riga 1 del tabellone. Dopo il primo passo si trova nella riga 2 e più vicino al tesoro rispetto alla riga 1. Dopo il passo successivo si trova nella riga 3 e di nuovo più lontano dal tesoro. Dato che è rimasto nella stessa colonna, il tesoro deve trovarsi su una casella della riga 2, infatti non importa in quale colonna sia nascosto il tesoro: la via più breve per raggiungere il tesoro da un'altra colonna è quella di trovarsi nella stessa riga.

Ma in quale colonna è nascosto il tesoro? Continuando il suo cammino, Daniel si avvicina inizialmente al tesoro della riga 4 facendo qualche passo verso sinistra; in particolare, è più vicino al tesoro della colonna 3 che a quello della colonna 4. Ma dopo l'ultimo passo della riga, Daniel si trova più lontano dal tesoro della colonna 2 che da quello della colonna 3. Quindi il tesoro deve trovarsi in un quadrato della colonna 3, siccome quanto detto sopra per le colonne vale anche per le righe: la via più breve per raggiungere il tesoro da un'altra riga è quella di trovarsi nella stessa colonna.

Questa è l'informatica!

Daniel cammina (con il suo pezzo di gioco) attraverso il tabellone. Da ogni casella su cui si trova attualmente, Nina misura la distanza dalla casella con il tesoro e la utilizza per il suo feedback. Di solito, la distanza tra due punti viene misurata come la lunghezza del collegamento rettilineo tra i punti (distanza euclidea). Ma i due campi non sono, in senso stretto, dei punti. Pertanto, Nina misura la distanza tra due campi nel numero di passi che Daniel dovrebbe fare per il percorso più breve da un campo all'altro. Questa *misura* può essere generalmente applicata alle griglie ed è nota in informatica come *distanza di Manhattan*, derivata dalla pianta a griglia del quartiere newyorkese di Manhattan.

Gli informatici scelgono il modo di calcolare la distanza tra due oggetti a seconda del quesito che vogliono risolvere. Ad esempio, se si vuole misurare la distanza tra due parole della stessa lunghezza in un linguaggio naturale, si può contare il numero di punti in cui le parole differiscono; si tratta quindi della *distanza di Hamming*. Se le parole sono di lunghezza diversa, si può usare la *distanza di Levenshtein*. Le distanze giocano spesso un ruolo nell'informatica quando si tratta di trovare soluzioni ottimali a un problema. Non importa se la soluzione a un problema deve essere la più veloce, la

più breve o la più economica: spesso non è necessario cambiare l'algoritmo, ma solo la misura della distanza: durata, lunghezza o costo.

Parole chiave e siti web

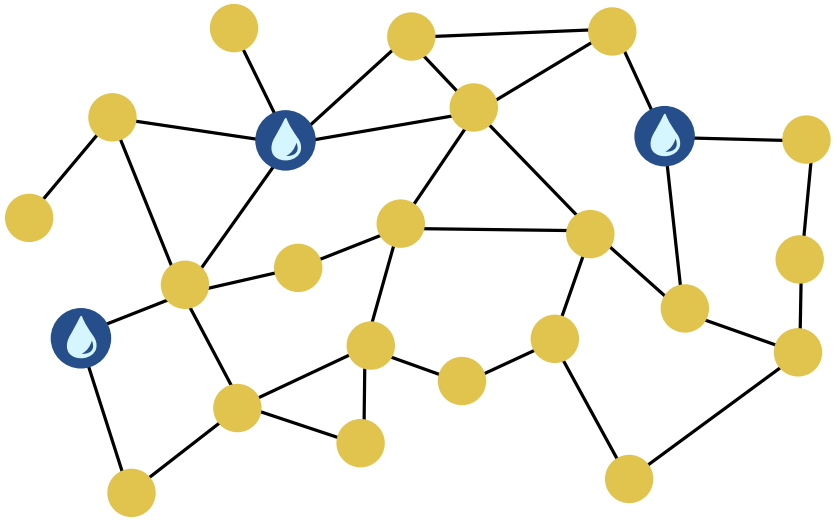
- Distanza di Manhattan: `https://it.wikipedia.org/wiki/Geometria_del_taxi`
- Distanza di Hamming: `https://it.wikipedia.org/wiki/Distanza_di_Hamming`
- Distanza di Levenshtein: `https://it.wikipedia.org/wiki/Distanza_di_Levenshtein`

13. Fontana

L'estate è calda in città. Per questo il sindaco ha fatto installare delle fontane con acqua potabile.

Le fontane devono essere posizionate in modo tale che per raggiungerle non si debbano percorrere più di due segmenti di strada da ogni angolo di strada. Solo in quel caso il sindaco sarà soddisfatto.

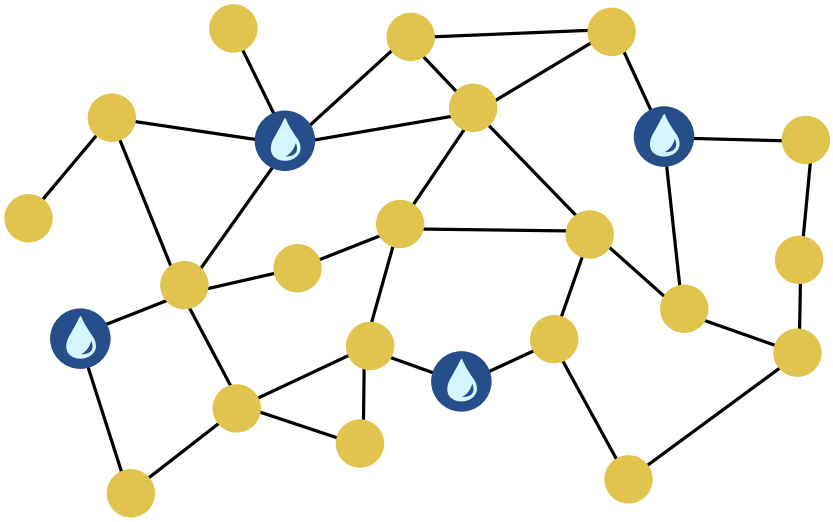
Ecco una mappa della città. Le linee sono segmenti di strada e i punti sono angoli di strada. In tre angoli ci sono già delle fontane .



Colloca un'altra fontana in modo che il sindaco sia soddisfatto.

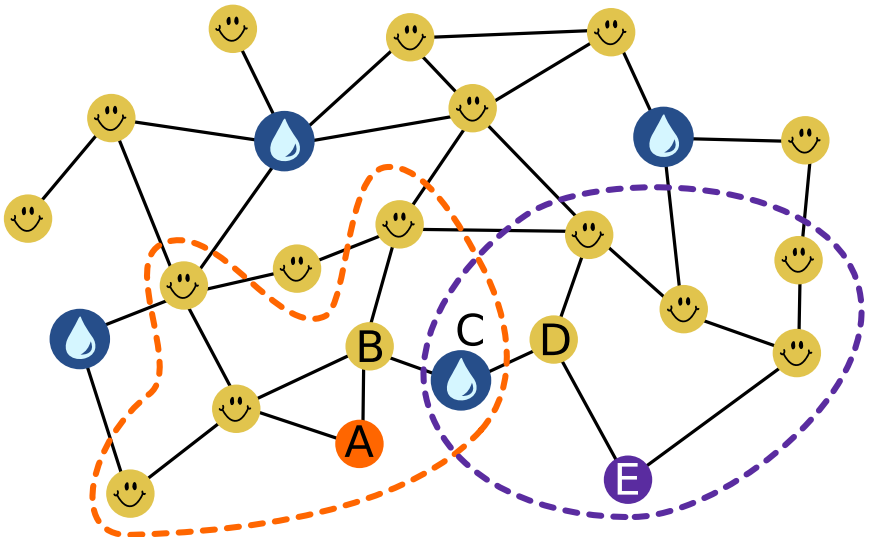
Soluzione

La risposta corretta:



Collocando un'altra fontana in basso al centro, per raggiungere una fontana si devono percorrere al massimo due segmenti di strada da ogni angolo di strada. Così facendo il sindaco sarà soddisfatto.

Come possiamo scoprire a quale angolo della strada si prevede l'installazione di un'altra fontana? Nella mappa della città segniamo tutti gli angoli delle strade con un , che non si trovino a più di due tratti stradali di distanza da una delle fontane già esistenti. Per quanto riguarda questi angoli, il sindaco può già ritenersi soddisfatto.



Per i cinque angoli di strada rimanenti, A, B, C, D ed E, posizioniamo un'altra fontana in C. In questo modo, da questi angoli alla fontana successiva si devono percorrere al massimo due segmenti di strada.

L'angolo C è l'unica posizione per una nuova fontna che lo consente. Se consideriamo per gli angoli A ed E rispettivamente tutti gli altri angoli che possono essere raggiunti attraverso due tratti di

strada (delineati con linee tratteggiate nella figura), l'angolo di strada C è l'unico che soddisfa questa condizione per A *e* E.

Questa è l'informatica!

La mappa della città può essere modellata come un *grafo*. Si tratta di uno strumento importante per l'informatica per modellare le relazioni tra gli oggetti e rispondere alle domande relative a queste relazioni. In questo caso, gli angoli delle strade possono essere intesi come oggetti e quindi come *nodi* del grafo. La relazione tra due oggetti è modellata nel grafo da *bordi*, che sono rappresentati come linee di collegamento. In questo caso, un bordo tra due angoli di strada significa che sono collegati da un segmento di strada. Questa relazione può essere chiamata vicinato. Tuttavia, i bordi possono modellare anche altre relazioni, come per esempio l'amicizia.

In questo compito, si deve trovare un sottoinsieme di nodi (per la creazione delle fontane) tale che ogni nodo al di fuori di questo sottoinsieme sia collegato tramite un percorso a un «nodo fontana» lungo al massimo due bordi. Nella terminologia informatica, questo si chiama trovare un «insieme dominante a distanza 2». In generale (per tutti i percorsi di lunghezza $k \geq 1$), la ricerca del più piccolo sottoinsieme possibile è uno dei problemi più difficili dell'informatica.

Questi «insiemi dominanti a distanza minima k» hanno assunto un ruolo sempre più importante negli ultimi tempi, soprattutto nel campo della *Social Computing* (in italiano anche *socioinformatica*): Per il trattamento automatico dei dati attraverso le reti sociali (ad esempio, per rilevare la diffusione di fake news) le relazioni di fan o follower tra gli utenti sono modellate come un grafo. Questi grafi possono essere così grandi che è possibile visualizzare solo una selezione rappresentativa di utenti (la più piccola possibile) - ad esempio, un «insieme dominante a distanza minima 3». Poiché la selezione veramente minima non può essere calcolata in modo efficiente, l'informatica sviluppa delle procedure, che calcolano le selezioni più piccole possibili, ma non garantite, in un tempo breve.

Parole chiave e siti web

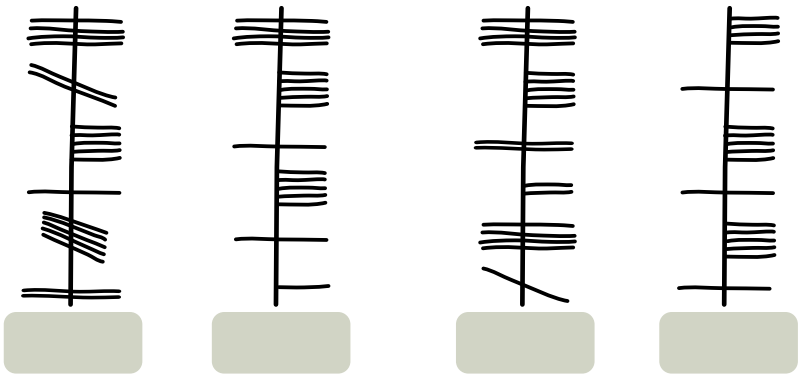
- Socioinformatica: `https://it.wikipedia.org/wiki/Socioinformatica`

14. Ogham

Sue conosce l'antico alfabeto irlandese Ogham. Ogni lettera è composta da uno o più tratti disposti su una lunga linea. Due lettere consecutive sono separate da uno spazio.

Sue usa l'Ogham come codice. Codifica quattro parole (i suoi tipi di frutta preferiti in tedesco): ANANAS, BANANE, MELONE e ORANGE.

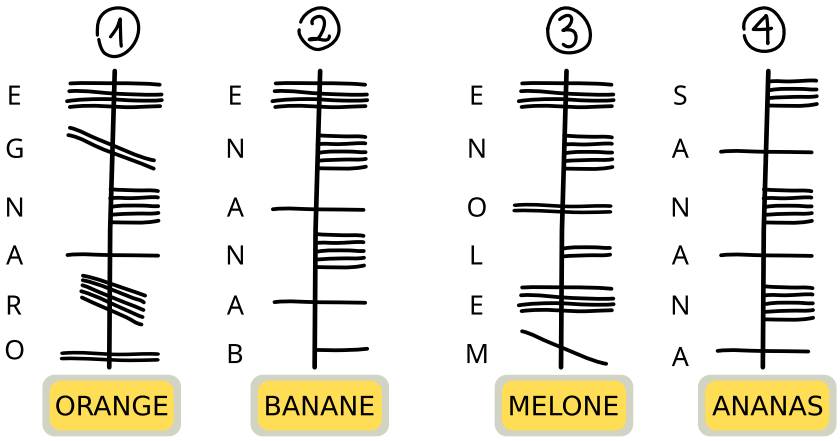
Quale parola corrisponde a quale codice Ogham?



ANANAS BANANE MELONE ORANGE

Soluzione

La risposta corretta:



Esistono vari modi per determinare l'assegnazione corretta. In ogni caso, però, bisogna scoprire in quale direzione sono scritte le lettere lungo la linea verticale. A questo proposito ci viene in aiuto la parola ANANAS, particolarmente suggestiva. In essa la lettera A ricorre tre volte, con una lettera diversa tra l'una e l'altra.

Solo nel codice Ogham 4 una lettera ricorre tre volte, e anche lì c'è una lettera in mezzo. Il codice 4 è quindi l'unico a cui si adatta la parola ANANAS. Questo dimostra che nell'Ogham le parole sono scritte dal basso verso l'alto e che la lettera A, che ricorre tre volte nell'Ogham, è scritta come una linea orizzontale che attraversa la linea verticale.

La lettera A in Ogham ricorre solo due volte nel codice 2. Anche a causa della codifica di N (cinque linee orizzontali a destra della linea) scoperta da ANANAS e della disposizione delle altre lettere, solo BANANE si adatta a questo codice. ORANGE si adatta solo al codice 1 perché la lettera A in Ogham si trova esattamente una volta. Ora rimane solo il codice 3; deve quindi essere la parola Ogham per MELONE e contiene le lettere Ogham E e N scoperte dalle altre parole nei posti appropriati.

Questa è l'informatica!

In questo compito, un testo sconosciuto deve essere decodificato o decifrato. Non si tratta di un compito molto difficile, perché il testo originale è noto. Inoltre, il testo sconosciuto è suddiviso in lettere e parole allo stesso modo del testo noto. Quando si decifra un testo segreto o un testo in una scrittura sconosciuta di cui non si conosce il testo in chiaro, spesso è utile pensare alla frequenza delle lettere e delle parole e su questa base cercare di trovarle nel testo. Alcuni alfabeti e scritture antiche sono stati decifrati in questo modo. Diventa difficile, tuttavia, quando i caratteri del testo sconosciuto non sono così facili da assegnare alle lettere e alle parole della lingua conosciuta, come nel caso dell'Ogham. In questi casi, l'unico modo per aiutarsi è confrontare il testo con testi o scritture note, come in questo compito. Per esempio, i geroglifici egiziani non sono stati decifrati per secoli finché, per caso, è stata trovata una pietra con geroglifici e due scritture conosciute, la Stele di Rosetta. Lo stesso testo è stato trovato tre volte sulla pietra. Era scritto in lingue diverse, ma conteneva sempre

gli stessi nomi. In questo modo è stato possibile decifrare elementi essenziali dei geroglifici. Tuttavia, questo non vale per tutte le scritture: I circa 650 caratteri della cultura Maya non sono ancora stati completamente decifrati, così come le scritture Lineare A e Lineare B della regione mediterranea.

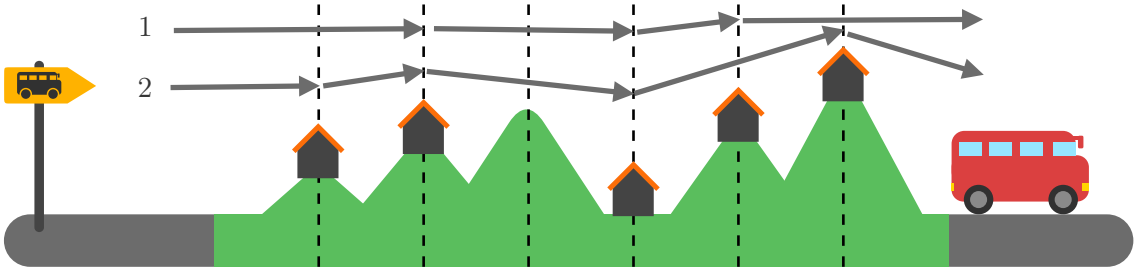
Anche in informatica i caratteri e i testi vengono decodificati, dopo essere stati precedentemente criptati per una trasmissione di dati a prova di intercettazione. Tuttavia, si utilizzano procedure completamente diverse rispetto alla codifica di parole in altre scritture. Codifiche così semplici sono troppo facili da decodificare, specialmente con l'aiuto dei computer, grazie alle considerazioni già citate sulla frequenza delle lettere e delle parole.

Parole chiave e siti web

- Crittografia: `https://it.wikipedia.org/wiki/Crittografia`
- Crittoanalisi: `https://it.wikipedia.org/wiki/Crittoanalisi`
- Alfabeto ogamico: `https://it.wikipedia.org/wiki/Alfabeto_ogamico`

15. Escursioni

A Mia piacciono le vacanze a piedi, in cui soggiorna ogni notte in un posto diverso. Per la sua prossima vacanza, Mia ha una mappa della regione. La mappa mostra il punto di partenza di Mia, la sua destinazione e tutti i luoghi in cui può soggiornare.



Mia ha diviso la regione in sezioni con linee tratteggiate. Può percorrere solo uno o due tratti alla volta in un giorno. Ha già messo sulla mappa due diverse passeggiate che può fare:

- L'escursione 1 prevede tre soggiorni
- L'escursione 2 prevede quattro soggiorni.

Mia può però fare altre escursioni.

Quante escursioni diverse può fare Mia in totale? Conta anche le escursioni 1 e 2.

A) 2 escursioni

B) 3 escursioni

C) 4 escursioni

D) 5 escursioni

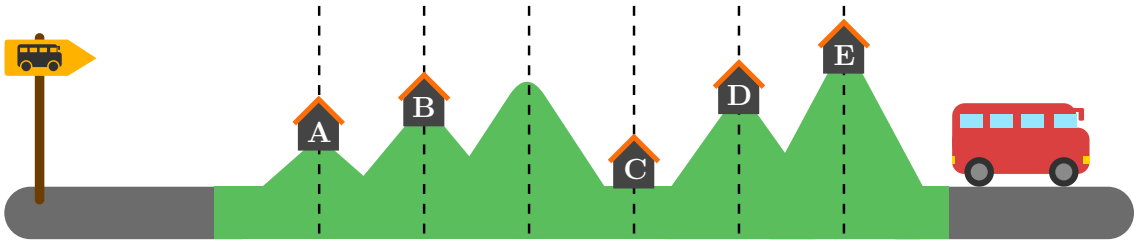
E) 6 escursioni

F) 7 escursioni

G) 8 escursioni

Soluzione

La risposta corretta è E) 6 escursioni.



Innanzitutto ci rendiamo conto che Mia deve pernottare a **B** e **C** perché la distanza tra questi due luoghi è la massima distanza (2) che può percorrere in un solo giorno. Quindi, per il tragitto da **B** a **C**, Mia ha solo un'opzione.

Ora possiamo determinare le possibilità per le altre parti del cammino: Dal punto di partenza ( ) a **B**, Mia può percorrerlo tutto d'un fiato o pernottare in **A** tra un tratto e l'altro; queste sono due possibilità (come nelle escursioni 1 e 2). Da **C** alla destinazione ( ) Mia deve percorrere tre tratti, e può pernottare dopo ogni tratto. Pertanto, può dividere l'intera camminata in tutte e tre le combinazioni di tratti 1 e 2:

- **C** → **D** → **E** → ;
- **C** → **E** → ;
- **C** → **D** → .

Quindi il numero totale di tutte le escursioni che Mia può fare è $2 \times 1 \times 3 = 6$.

Questa è l'informatica!

A volte il numero di possibilità per eseguire un determinato compito può essere molto grande. Ad esempio, esistono circa 14 milioni di modi per scegliere 6 numeri diversi da 1 a 49. E ci sono circa mezzo miliardo di modi per scrivere i numeri da 1 a 12 in diverse sequenze. Anche questo richiede al computer un po' di tempo.

È fortuito che in questo compito non ci sia un soggiorno dopo la terza sezione e che il conteggio di tutte le passeggiate che Mia può fare possa essere diviso in tre parti. Il problema del conteggio viene scomposto in tre problemi di conteggio più piccoli, per così dire. In informatica, la tecnica della *scomposizione del problema* è spesso utilizzata nella progettazione di algoritmi. Questo principio di soluzione è noto anche come *divide et impera*.

Alcuni importanti algoritmi di ordinamento, ad esempio, funzionano secondo questo principio. Anche la programmazione dinamica, un metodo per la soluzione algoritmica di problemi di ottimizzazione (descritto nel 1957 da Richard Bellman), si basa su questo principio: se si riconosce che le soluzioni ottimali di un problema sono composte dalle soluzioni ottimali di sottoproblemi, si può usare questo principio per «iniziare in piccolo», per così dire: In primo luogo, le soluzioni dei sottoproblemi più piccoli vengono calcolate direttamente e poi combinate per formare le soluzioni dei successivi

sottoproblemi più grandi. Questa operazione viene ripetuta fino a trovare la soluzione ottimale per il problema completo. Poiché le soluzioni parziali trovate spesso contribuiscono alle soluzioni di molte parti più grandi, vengono memorizzate per evitare di ripetere calcoli identici. La programmazione dinamica può anche essere molto utile per contare le possibilità, come in questo problema.

Parole chiave e siti web

- Decomposizione del problema, scomposizione
- Divide et impera: `https://it.wikipedia.org/wiki/Divide_et_impera_(informatica)`
- Programmazione dinamica: `https://it.wikipedia.org/wiki/Programmazione_dinamica`

16. Le commissioni di Emma

Emma è a casa . Deve svolgere tre compiti e tornare:

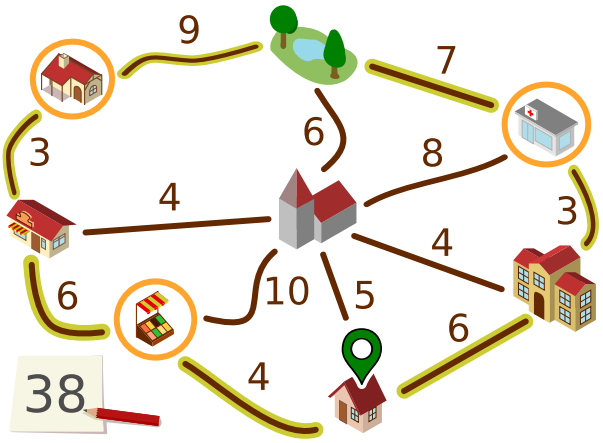
- ritirare un pacco al chiosco
- comprare frutta al mercato e
- Andare in farmacia per prendere una medicina.

Emma non sa quanto tempo impiegherà in ogni negozio. Ma il viaggio dovrebbe essere il più breve possibile.

Emma ha scritto su una mappa quanti minuti dedicherà all'attività di spostamento tra le singole località della città.

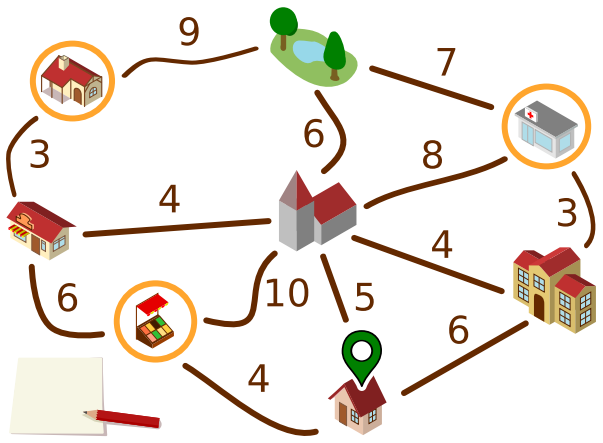
Ha anche segnato sulla planimetria il percorso che sta facendo.

Emma ha bisogno in questo caso di un totale di $6 + 3 + 7 + 9 + 3 + 6 + 4 = 38$ minuti per completare il percorso.



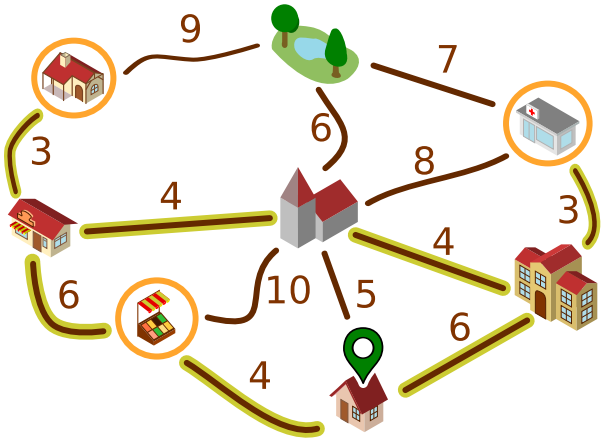
Emma si chiede se si può essere ancora più veloci. Forse è utile percorrere alcune strade più di una volta?

Determina il percorso più breve che Emma può intraprendere per completare i suoi tre compiti.

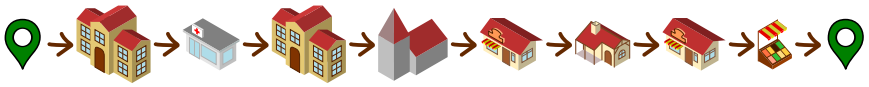


Soluzione

Questa è la soluzione:

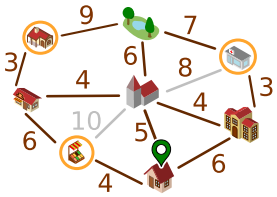


Emma può camminare lungo i percorsi selezionati (o nella direzione opposta):

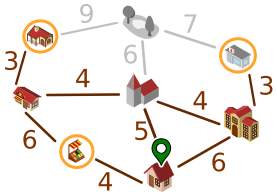


Per percorrere questa distanza ha bisogno di $6 + 3 + 3 + 4 + 4 + 3 + 3 + 6 + 4 = 36$ minuti.

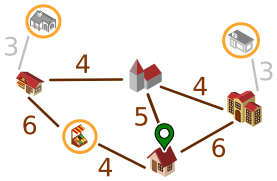
Vogliamo giustificare perché non può esistere un percorso ancora più breve. Per farlo, utilizziamo una rappresentazione semplificata del piano.



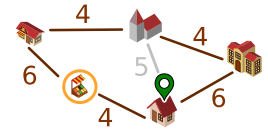
Possiamo ignorare i percorsi disegnati in grigio. Esistono percorsi più brevi tra i luoghi collegati dai percorsi, ossia attraverso altri luoghi.



Possiamo anche ignorare il parco, Emma infatti non deve andare al parco. Inoltre, per ogni percorso che passa per il parco, esiste un'alternativa più breve.



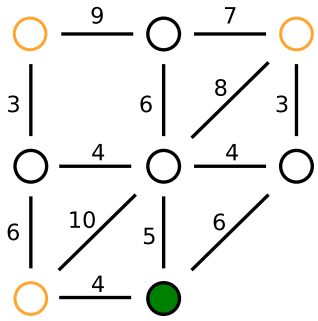
Emma deve andare in farmacia e al chiosco. Può arrivarci solo dal panificio o dalla scuola. Deve percorrere a piedi la distanza tra questi luoghi. Ci vogliono $3 + 3 = 6$, ovvero 12 minuti in totale. Ricordiamolo e combiniamo i due luoghi di cui sopra con quelli di cui sotto in uno solo.



Ora rimane solo il grafico a destra. L'inizio e la fine del percorso sono qui. Questi tre luoghi (, ,) devono essere visitati. Il percorso più breve che soddisfa questo requisito passa attraverso tutti e cinque i luoghi e lungo tutti i percorsi tranne quello grigio e richiede $4 + 6 + 4 + 6 = 24$ minuti. Con i 12 minuti di cui sopra, fanno 36 minuti. Le considerazioni precedenti dimostrano che non può esistere un percorso più breve.

Questa è l'informatica!

Per giustificare la risposta corretta è stata utilizzata una rappresentazione semplificata della mappa. Sarebbe stato possibile presentare la mappa in modo molto più astratto:



Questa rappresentazione contiene tutte le informazioni importanti per il percorso di Emma, ovvero

- Oggetti: i luoghi, con segnati i luoghi importanti per il percorso;
- e relazioni tra gli oggetti: le distanze tra i luoghi per ognuno dei quali è indicata una lunghezza.

Uno strumento importante per modellare le relazioni tra gli oggetti sono i *grafi*. I grafi sono costituiti da nodi (per gli oggetti) e da bordi (coppie di oggetti, per le relazioni). La mappa di Emma può essere modellata come un *grafo pesato*, dove alle singole relazioni vengono assegnati dei valori numerici (i *pesi*).

L'informatica è interessata a domande che possono essere poste in relazione ai grafi, e per gli algoritmi che possono essere utilizzati per rispondere alle domande. Una domanda importante per i grafi pesati è: Qual è il percorso più breve (o più veloce) tra due nodi? La «domanda del grafico» in questo compito è simile: Qual è il viaggio di andata e ritorno più breve da un nodo che visita molti altri nodi? L'informatica conosce molti algoritmi in grado di determinare in modo efficiente i percorsi più brevi nei grafi. Tali algoritmi sono implementati in software per, ad esempio, la pianificazione di itinerari.

Parole chiave e siti web

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

A. Autori dei quesiti

Somayah Albaradei

Laila Alharthi

Aldrich Ellis Catapang Asuncion

Leonardo Barichello

Liam Baumann

Wilfried Baumann

Javier Bilbao

Diego César

Sarah Chan

Marios Omar Choudary

Eimear Colreavy

Kris Coolsaet

Valentina Dagienė

Darija Dasović

Christian Datzko

Justina Dauksaite

Nora A. Escherle

Gerald Futschek

Bence Gaál

Emily Gates

Christian Giang

Juan Gutiérrez

Josefine Hiebler

Mathias Hiron

Alisher Ikramov

Hyun-seok Jeon

Merel Kämper

David Khachatryan

Vaidotas Kinčius

Mhairi King

Jia-Ling Koh

Sophie Koh

Víctor Koleszar

Taina Lehtimäki

Angélica Herrera Loyo

Carlos Luna

Yong Mao

Yoshiaki Matsuzawa

Madhavan Mukund

Natalia Natalia

Tom Naughton

Marika Parviainen

Jean-Philippe Pellet

Zsuzsa Pluhár

Wolfgang Pohl

Estela Ramić

Chris Roffey

Kirsten Schlüter

Eljakim Schrijvers

Giovanni Serafini

Alieke Stijf

Marianne Thut

Monika Tomcsányiová

Svetlana Unković

Florentina Voboril

Michael Weigend

Manuel Wettstein

Kyra Willekes

B. Partner accademici

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

C. Sponsoring

http://www.haslerstiftung.ch/

Standortförderung beim Amt für Wirtschaft und Arbeit Kanton Zürich

http://www.ubs.com/

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

i-factory (Musée des transports, Lucerne)

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

D. Ulteriori offerte

La Fiamma IT: `https://it-feuer.ch/it/`
In Svizzera, numerose organizzazioni si impegnano per la formazione delle giovani leve nell'ambito dell'informatica. L'iniziativa «La Fiamma IT» vuole unire queste forze e contribuire insieme a diffondere il tema nell'opinione pubblica in tutta la Svizzera. La fiamma IT presenta numerose offerte rivolte sia ai docenti che agli studenti.

CoetryLab: `https://www.coetry-lab.org/`
Il team del CoetryLab (Zürich) vuole dare ai bambini e ai giovani l'accesso alla programmazione e ai media. Il CoetryLab vuole essere il luogo di sperimentazione e progettazione extrascolastica e aprire il mondo del coding a tutti. Le loro idee possono essere realizzate in modo creativo e siti web, applicazioni, giochi e molto altro possono essere sviluppati in team o da soli.

Roteco: `https://www.roteco.ch/it/`
Il progetto Roteco consiste in una comunità di insegnanti desiderosi di preparare gli allievi per la società digitale. In questa comunità gli insegnanti trovano, sviluppano e si scambiano attività didattiche inerenti la robotica educativa e più in generale le scienze informatiche pronte da essere utilizzate in classe e vengono informati con le ultime novità e corsi in questi campi.

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.