CONCURSO BEBRAS · ESPAÑA

Bebras 2024

3.º y 4.º de ESO · Grupo V (14–16 años)

Cuadernillo con las 15 tareas del concurso, sus soluciones explicadas y las secciones «Esto es informática» y «Esto es pensamiento computacional».

bebras.ehu.es

Presentación

Bebras es un desafío internacional de informática y pensamiento computacional que se celebra cada curso en más de 80 países. Sus tareas son problemas breves, resolubles sin conocimientos previos de programación, que acercan al alumnado a ideas fundamentales de la informática: algoritmos, estructuras de datos, lógica, grafos o representación de la información.

Este cuadernillo recoge las 15 tareas que resolvió el alumnado de 3.º y 4.º de ESO en la edición de 2024, realizada en la plataforma ViLLE (curso «2024 Spanish V»). Cada tarea valía 10 puntos, con un máximo de 150. Para cada tarea se incluye:

- **Enunciado**, en una página que se puede fotocopiar para trabajar en clase.
- **Solución** con la respuesta correcta y su explicación.
- **Esto es informática**: el concepto informático que hay detrás de la tarea.
- **Esto es pensamiento computacional**: las destrezas que se ponen en juego al resolverla.

La dificultad indicada es la que asignó el taller internacional Bebras para la franja de 14 a 16 años; cuando una tarea no tenía asignada esa franja, se indica entre paréntesis la dificultad en la franja más próxima. Las opciones de respuesta aparecen en el orden de los materiales originales; en ViLLE el orden pudo ser distinto.

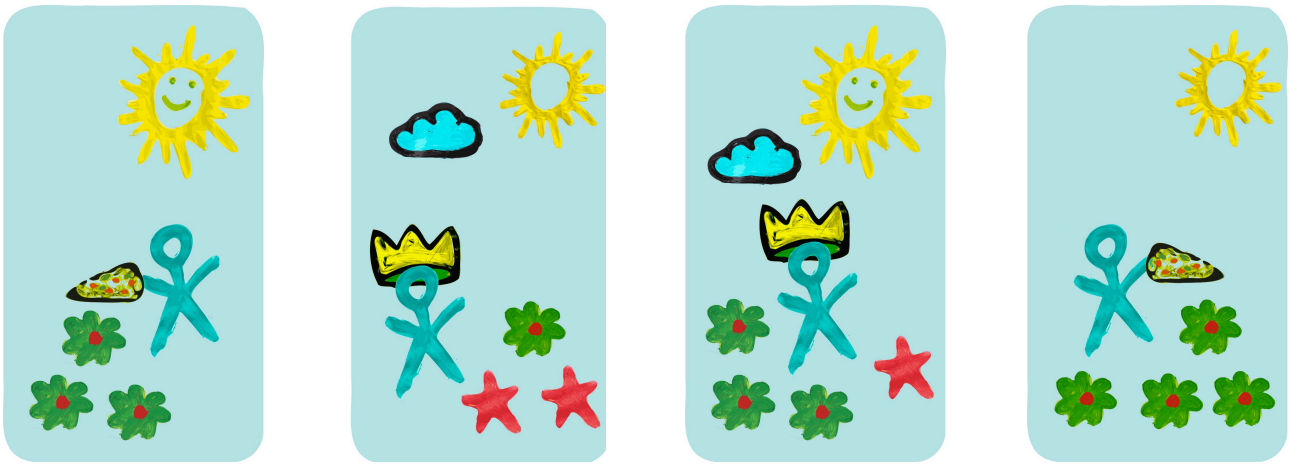
Resumen de tareas y respuestas

N.º	Tarea	Código	Dificultad	Respuesta
1	Cartas dibujadas	2024-IN-04	— (fácil en 10–12)	C) Si el hombre se come una pizza, entonces no lleva corona.
2	Préstamos de canicas	2024-AT-03	— (media en 12–14)	C) Tim.
3	Red ferroviaria	2024-PK-03	Difícil	A) 13.
4	Rutas de árboles	2024-DE-08	— (media en 12–14)	 (roble, abedul, abeto, frutal, árbol claro).
5	Bolas rojas y azules	2024-BG-01b	Difícil	azul, roja, roja, azul, azul, azul, roja, roja.
6	Alfred en el baile	2024-HU-02	— (fácil en 10–12)	B) Alfred puede que llevara corbata o puede que llevara bastón.
7	Amigos	2024-LT-05	Media	Dave y Fred.
8	Superbebras	2024-DE-07	Fácil	C.
9	La puntuación más alta	2024-SK-04	Media	A B B C **C** B **B B** B A A A → 2 + 2 + 4 + 3 = 11 puntos.
10	En busca del tesoro	2024-HU-03	Fácil	4.
11	El mapa de la comida	2024-IT-01b	Difícil	A2, A4, B1, B5, C5, D1 y G3.
12	La máquina de globos	2024-DE-06a	Difícil	Por ejemplo, B E B C A C B D B (hay 4 respuestas válidas).
13	Días soleados	2024-DE-04a	— (fácil en 12–14)	B) ningún castor nadó en el estanque de la cascada helada.
14	Cadenas de palabras	2024-CA-02	Media	C) BAT → CAT → COT.
15	La pulsera Bebras	2024-TW-03	Fácil	D) 6 cuentas.

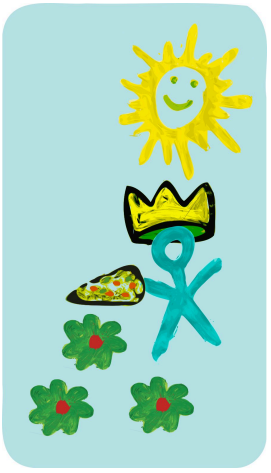
1 Cartas dibujadas

2024-IN-04 · India
Dificultad: — (fácil en 10–12)

Manas hace dibujos en sus cartas siguiendo una única regla secreta. Ha creado cuatro cartas que cumplen la regla:



Anil observa las cartas y crea una carta nueva, pero Manas le dice que no cumple la regla.



¿Cuál de las siguientes podría ser la regla secreta que sigue Manas para dibujar las cartas?

A) Si está nublado, entonces no hay flores.
B) Tiene que haber una estrella roja o el hombre se come una pizza.
C) Si el hombre se come una pizza, entonces no lleva corona.
D) Tiene que haber un sol sonriente o una nube.

Respuesta correcta: C) Si el hombre se come una pizza, entonces no lleva corona.

Explicación

La respuesta correcta es la **C**.

La regla secreta tiene que cumplirse en las cuatro cartas de Manas y no cumplirse en la carta de Anil.

- **A** no puede ser: en la 2.ª y la 3.ª carta hay una nube y también flores.
- **D** no puede ser: la 4.ª carta no tiene ni sol sonriente ni nube.
- **B** no puede ser: las cuatro cartas la cumplen, pero la carta de Anil también (en ella el hombre se come una pizza), y Manas dice que esa carta no cumple la regla.
- **C** sí puede ser: ninguna de las cuatro cartas tiene a la vez pizza y corona, y la carta de Anil tiene las dos cosas, así que no cumple la regla.

Esto es informática

La tarea muestra la **lógica** básica que usa la informática para representar y procesar decisiones y observaciones. Los ordenadores siguen reglas claras para decidir qué hacer, de forma parecida a como observamos patrones y tomamos decisiones en la vida diaria.

Las reglas de las opciones son **condiciones** como las de los programas: «si... entonces...» (implicación) y «... o ...» (disyunción). Un detalle importante: una regla «si A, entonces B» solo se incumple cuando A es verdadero y B es falso. Por eso, las cartas en las que el hombre no come pizza cumplen siempre la regla C, lleve o no corona. En programación, este tipo de reglas sirve, por ejemplo, para validar datos de entrada: un formulario puede rechazar un dato que no cumpla una condición.

Esto es pensamiento computacional

Descomposición: cada opción aísla una condición concreta y su consecuencia, y se puede comprobar carta a carta. **Abstracción**: hay que fijarse solo en los elementos que menciona cada regla (nube, flores, pizza, corona...) e ignorar el resto del dibujo. **Evaluación**: para descartar una regla basta con encontrar un contraejemplo, es decir, una carta de Manas que la incumpla o la carta de Anil si la cumple.

Palabras clave y enlaces: Lógica condicional, lógica booleana, implicación · es.wikipedia.org/wiki/Condicional_material

2 Préstamos de canicas

2024-AT-03 · Austria
Dificultad: — (media en 12–14)

Tres castores amigos juegan a menudo con canicas y se las prestan unos a otros. Para saber quién debe cuántas canicas a quién, decidieron apuntar todos los préstamos en una única hoja de papel:

Ejemplo de lista compartida (no es relevante para la pregunta):

Presta	Recibe	Cantidad
Anna	Tim	5
Tim	Bernd	3
…	…	…

Con el tiempo empezaron a desconfiar. Se acusaban unos a otros de hacer trampa y pensaban que algunos préstamos nunca habían ocurrido.

Así que decidieron no depender de una sola lista a la que todos tuvieran acceso: cada castor llevaría su propia lista para detectar intentos de trampa. Al cabo de una semana comparan sus listas. ¿Ha vuelto a hacer trampa alguien?

Lista de Anna

Presta	Recibe	Cant.
Tim	Anna	2
Bernd	Tim	1
Bernd	Anna	3
Anna	Tim	2

Lista de Tim

Presta	Recibe	Cant.
Tim	Anna	2
Bernd	Tim	1
Tim	Anna	3
Anna	Tim	2

Lista de Bernd

Presta	Recibe	Cant.
Tim	Anna	2
Bernd	Tim	1
Anna	Tim	2
Bernd	Anna	3

¿Quién ha hecho trampa?

A) Anna
B) Bernd
C) Tim
D) Nadie

Respuesta correcta: C) Tim.

Explicación

La respuesta correcta es **Tim**.

Si solo nos fijamos en la columna de cantidades, (2, 1, 3, 2), parece que el tramposo es Bernd, porque es el único cuya lista tiene otra secuencia (2, 1, 2, 3). Pero no es así: la lista de Bernd solo tiene los préstamos en otro orden, y el resultado total es el mismo.

Quien hizo trampa fue Tim: cambió el préstamo (Bernd → Anna, 3) por (Tim → Anna, 3). Si nadie lo hubiera notado, habría ganado 3 canicas a costa de Bernd. Su lista es la única que no coincide con las otras dos en el conjunto de préstamos.

Esto es informática

La idea de no guardar todas las transacciones en un único servidor central de una entidad de confianza, sino distribuirlas por una red entre iguales (*peer-to-peer*), es una de las ideas básicas de la tecnología **blockchain** (cadena de bloques) o de los **registros distribuidos**.

El método de la tarea solo funciona si la mayoría de los participantes dice la verdad. Blockchain va un paso más allá con un **algoritmo de consenso** que, por ejemplo, exigiría que quien quiera hacer trampa controle más capacidad de cálculo que todos los demás participantes juntos, algo imposible si la red es lo bastante grande. Aun así, la tarea ya muestra cómo se puede reforzar la confianza en un entorno sin confianza repartiendo el poder entre todos, en lugar de depender de una única fuente de verdad.

Esto es pensamiento computacional

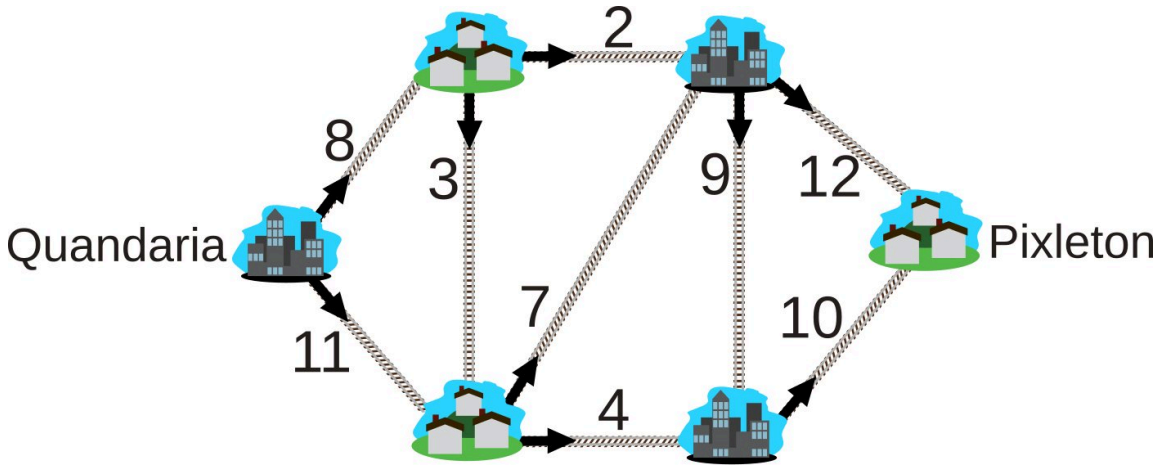
Hay que **abstraer** lo esencial de cada lista: lo que importa es el conjunto de préstamos y su efecto, no el orden en que aparecen. Después se **evalúan** las tres listas comparándolas entre sí para localizar la que discrepa de la mayoría, evitando la conclusión precipitada a la que lleva fijarse solo en un patrón superficial (la columna de cantidades).

Palabras clave y enlaces: Blockchain, registro distribuido, consenso · es.wikipedia.org/wiki/Cadena_de_bloques

3 Red ferroviaria

2024-PK-03 · Pakistán
Dificultad: Difícil

En el país de Bebravia, los pueblos vecinos están unidos por una red de vías de tren. Por cada vía puede circular un número limitado de trenes al día, indicado en el esquema. Trenes distintos pueden compartir una vía.



Quandaria se ha ofrecido a enviar materiales a Pixleton. En este sentido, los trenes deben seguir siempre las flechas.

¿Cuál es el número máximo de trenes que pueden salir cada día de Quandaria y llegar a Pixleton?

A) 13
B) 15
C) 19
D) 22

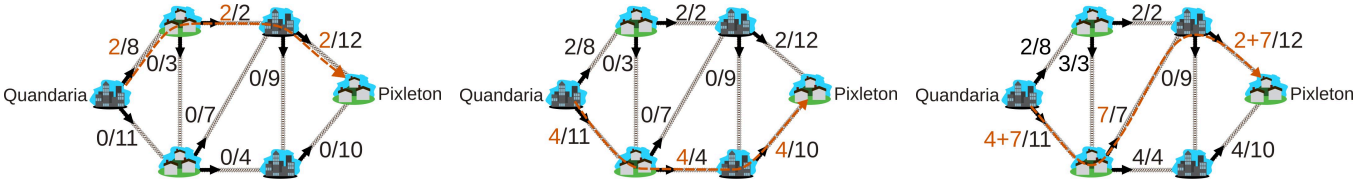
Respuesta correcta: A) 13.

Explicación

La respuesta correcta es la **A) 13**.

Lo primero que se observa es que Quandaria (Q) no puede enviar a Pixleton (P) más de 8 + 11 = 19 trenes, que es el máximo de trenes que pueden salir de Q. Así que la respuesta D) 22 es imposible.

Una forma de resolver el problema es buscar un camino lo más corto posible de Q a P (aquí, de longitud 3) y calcular cuántos trenes admite como máximo. Después se repite el proceso en el resto de la red, con la capacidad que queda en cada vía. Los esquemas muestran tres caminos encontrados así, con una capacidad conjunta de 2 + 4 + 7 = 13 trenes. Hay otras formas de llevar 13 trenes de Q a P, algunas con caminos más largos.



Para comprobar que 13 es el máximo, fíjate en las tres vías del centro de la red, con capacidades 2, 7 y 4: todos los caminos pasan por una de ellas, así que no se puede superar 2 + 7 + 4 = 13.

Esto es informática

Esta tarea es un ejemplo del **problema del flujo máximo**. Es como un juego de tuberías: hay una red de tubos con distintas capacidades y queremos saber cuánta agua puede circular desde un origen hasta un destino.

El problema se puede resolver con varios algoritmos: Ford-Fulkerson, Edmonds-Karp, Dinic y otros. Las vías centrales por las que hay que pasar forzosamente forman lo que se llama un **corte mínimo**; el teorema de flujo máximo–corte mínimo dice que ambos valores coinciden.

Cada algoritmo tiene ventajas e inconvenientes en eficiencia, facilidad de programación y tipo de redes a las que se aplica. Los algoritmos de flujo máximo se usan para optimizar el tráfico, en redes de distribución de agua o electricidad, en asignación de recursos, en redes de ordenadores y en muchos otros campos.

Esto es pensamiento computacional

Resolver la tarea requiere **pensamiento algorítmico** y **evaluación**. La solución se **descompone** en una secuencia de pasos, cada uno de los cuales consiste en encontrar un camino entre Q y P con la máxima capacidad disponible. El algoritmo termina cuando ya no queda ningún camino posible. Además hay que evaluar el resultado: el corte de 13 demuestra que no se puede hacer mejor.

Palabras clave y enlaces: Grafo, flujo máximo, corte mínimo, algoritmo de Ford-Fulkerson · es.wikipedia.org/wiki/Algoritmo_de_Ford-Fulkerson · es.wikipedia.org/wiki/Problema_de_flujo_máximo

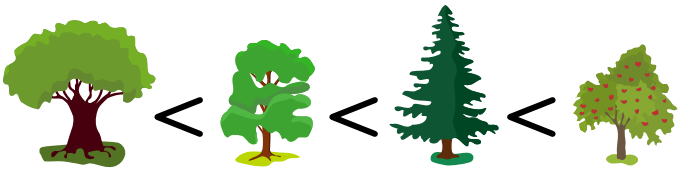
4 Rutas de árboles

2024-DE-08 · Alemania
Dificultad: — (media en 12–14)

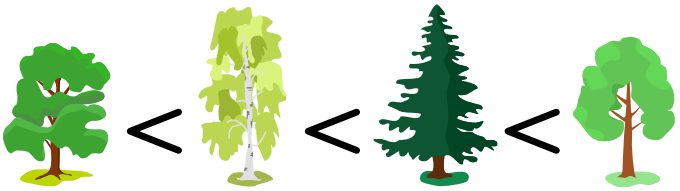
La bióloga Bea organiza rutas por el Bosque de los Castores. En cada ruta enseña algunos árboles especiales.

De tres rutas anteriores recuerda qué árboles gustaron más que otros a los visitantes (árbol 1 < árbol 2 significa que, en esa ruta, el árbol 2 gustó más que el árbol 1):

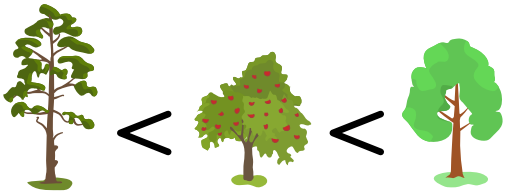
Ruta 1



Ruta 2



Ruta 3



En su próxima ruta, Bea quiere enseñar más tarde los árboles que más gustan. Quiere encontrar un **buen orden**, que no contradiga el orden de popularidad de ninguna de las rutas anteriores.

Por ejemplo, si Bea quiere enseñar los árboles  y  en un buen orden, debe enseñar  antes que . Enseñar  antes que  contradiría la ruta 1, en la que  gustó más que .

En la nueva ruta, Bea quiere enseñar estos árboles: 

Ayuda a Bea: coloca estos árboles en un buen orden.



Escribe el orden (1.º a 5.º):

Respuesta correcta:  (roble, abedul, abeto, frutal, árbol claro).

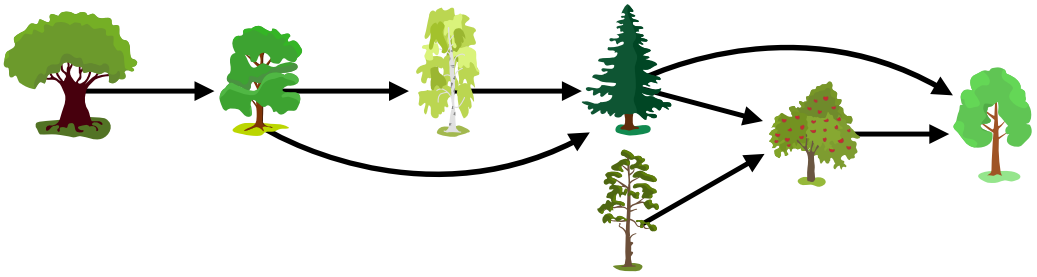
Explicación

La respuesta es:



Un buen orden no debe contradecir ninguno de los órdenes de popularidad de las tres rutas anteriores. Para evitarlo, Bea tiene que deducir el orden de la nueva ruta a partir de esos órdenes.

Entre los cinco árboles,  es el único que no aparece como más popular que ningún otro, así que debe ser el primero. En la ruta 1,  < ... <  < , así que  debe ir después de . En la ruta 2,  <  < , así que  va antes que  y  después. Y en la ruta 3,  < . Todo ello se resume en este diagrama:



Por tanto, el orden  es el único que respeta el diagrama.

Esto es informática

En esta tarea hay que ordenar un conjunto de árboles por popularidad creciente. Se sabe cómo se comparan algunas parejas de árboles, pero no todas: por ejemplo, el roble y el árbol claro no se comparan directamente en ninguna ruta. ¿Seguro que aun así se pueden ordenar?

Hay dos propiedades importantes:

1. La comparación de popularidad es **transitiva**: si árbol 1 < árbol 2 y árbol 2 < árbol 3, entonces árbol 1 < árbol 3.
2. No hay **conflictos** entre las rutas: si árbol 1 < árbol 2 en una ruta, no hay ninguna otra en la que árbol 2 < árbol 1.

Gracias a ellas, la comparación define un **orden**. A primera vista parece un orden **parcial**, porque no todas las parejas se comparan directamente. Convertir un conjunto parcialmente ordenado en una secuencia totalmente ordenada es la **ordenación topológica**, y en informática se conoce un algoritmo sencillo para hacerlo (que solo funciona si no hay conflictos). De hecho, si se mira con atención, el orden de los siete árboles es **total**: por transitividad se sabe cómo se comparan dos árboles cualesquiera, así que la ordenación de cualquier subconjunto tiene una única solución.

Esto es pensamiento computacional

Hay que **abstraer** la información de las tres rutas en una sola estructura (un diagrama de relaciones «va antes que»), usar el **razonamiento lógico** (la transitividad) para deducir comparaciones que no aparecen directamente y aplicar un **algoritmo**: elegir en cada paso el árbol que no tiene ningún otro pendiente por delante.

Palabras clave y enlaces: Ordenación topológica, orden parcial, transitividad · es.wikipedia.org/wiki/Ordenamiento_topológico

5 Bolas rojas y azules

2024-BG-01b · Bulgaria
Dificultad: Difícil

Tenemos una fila de bolas rojas y azules:



Contamos el número de bolas azules de izquierda a derecha empezando por la primera bola, después empezando por la segunda, y así sucesivamente. Obtenemos esta secuencia:

```
3, 3, 2, 1, 1, 1
```

Ahora escribimos 0 si el número es par y 1 si es impar, y obtenemos:

```
1 1 0 1 1 1
```

Esta es otra secuencia de ceros y unos (par e impar): `0 1 1 1 0 1 0 0`. Construye la fila de bolas que le corresponde.

Colorea cada bola de rojo o de azul:



Respuesta correcta: azul, roja, roja, azul, azul, azul, roja, roja.

Explicación

Respuesta correcta:



Hay dos algoritmos sencillos para llegar a la solución: uno recorre la secuencia de izquierda a derecha y otro de derecha a izquierda.

De derecha a izquierda (el más sencillo). El último número dice si la última bola es azul (1) o roja (0). Hacia la izquierda, el número de bolas azules cambia exactamente cuando la secuencia cambia de 0 a 1 o de 1 a 0: si dos valores consecutivos son distintos, en esa posición hay una bola azul; si son iguales, una roja.

0	1	1	1	0	1	0	0
4 azules	3 azules	3 azules	3 azules	2 azules	1 azul	0 azules	0 azules

Así, el número de bolas azules desde cada posición es 4 3 3 3 2 1 0 0, y de ahí sale la fila.

De izquierda a derecha. Se recorren los valores de dos en dos con dos variables, `c` (valor actual) y `cc` (siguiente). Si son iguales, al pasar a la siguiente posición el número de azules no cambia, así que en la posición actual hay una bola roja; si son distintos, hay una azul. La última bola es roja si el último valor es 0 (cero azules) y azul si es 1.

```
leer c
repetir n-1 veces:
    leer cc
    si c == cc: escribir 'roja'  si no: escribir 'azul'
    c ← cc
si c == '0': escribir 'roja'  si no: escribir 'azul'
```

Esto es informática

En informática se usan muchas **codificaciones** distintas para describir una situación a un ordenador. A menudo se puede omitir información, simplificar la codificación y aun así no perder lo original. Aquí no se guardan los números exactos de bolas azules, sino solo su paridad con ceros y unos, y aun así se puede reconstruir la fila completa. Por eso, para usar una codificación hay que saber cómo se creó, es decir, qué algoritmo se aplicó.

Esta idea está en la base de los **algoritmos de compresión**, que reducen el tamaño de un archivo sin perder información; por ejemplo, el formato de imagen PNG combina varios de ellos.

La fila se puede reconstruir con un **algoritmo de una sola pasada**, que procesa los datos una sola vez, leyéndolos y descartándolos sin volver atrás. Este enfoque ahorra tiempo y memoria al cifrar, comprimir, descomprimir y, en general, al procesar grandes volúmenes de datos.

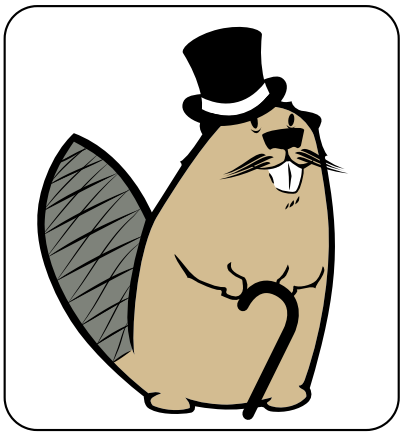
Esto es pensamiento computacional

Hay que descubrir la regla que relaciona dos posiciones consecutivas (**reconocimiento de patrones**) y convertirla en un procedimiento que se repite en **bucle** (**pensamiento algorítmico**). Elegir el sentido de recorrido adecuado simplifica mucho el trabajo: es un buen ejemplo de cómo pensar en la eficiencia de un algoritmo y en cómo gestionar los datos con los mínimos recursos.

Palabras clave y enlaces: Compresión de datos, algoritmo de una pasada, paridad · es.wikipedia.org/wiki/Compresión_de_datos

6 Alfred en el baile

2024-HU-02 · Hungría
Dificultad: — (fácil en 10–12)



Cuando Alfred, un castor famoso, fue invitado al ayuntamiento, le hicieron esta foto.

Cuatro *influencers* escribieron en las redes sociales distintas afirmaciones sobre él. No todo lo que leemos en las redes sociales es verdad.

¿Cuál de las siguientes afirmaciones de los *influencers* es verdadera?

A) Alfred llevaba chistera pero no llevaba bastón

B) Alfred puede que llevara corbata o puede que llevara bastón

C) Alfred llevaba chistera y pajarita

D) Alfred llevaba pajarita o no llevaba bastón

Respuesta correcta: B) Alfred puede que llevara corbata o puede que llevara bastón.

Explicación

La afirmación verdadera es la **B**.

Hay que decidir, a partir de la foto, si cada afirmación es verdadera o falsa. Cada frase contiene dos afirmaciones unidas por los operadores lógicos Y (en la A, «pero» funciona como Y) y O.

p	q	p Y q
F	F	F
F	V	F
V	F	F
V	V	V

p	q	p O q
F	F	F
F	V	V
V	F	V
V	V	V

Una afirmación con Y solo es verdadera si las dos partes lo son; una con O es verdadera si al menos una parte lo es. En la foto, Alfred lleva chistera y bastón, pero no lleva corbata ni pajarita.

- **A)** llevaba chistera (V) pero no llevaba bastón (F): V Y F = **falso**.
- **B)** llevaba corbata (F) o llevaba bastón (V): F O V = **verdadero**.
- **C)** llevaba chistera (V) y pajarita (F): V Y F = **falso**.
- **D)** llevaba pajarita (F) o no llevaba bastón (F): F O F = **falso**.

Esto es informática

Y y O son **operadores lógicos** que permiten tomar decisiones combinando varias condiciones. Por ejemplo: si una persona tiene 18 años o más Y no lleva gafas, se le puede conceder el carné de conducir sin restricciones.

Los ordenadores son, en esencia, sistemas binarios. Los operadores lógicos (con 1 para verdadero y 0 para falso) les permiten realizar operaciones lógicas y matemáticas complejas, imprescindibles en muchos campos de la informática: la programación, el diseño de circuitos de control o las consultas a bases de datos. Las tablas que muestran el resultado de un operador para todas las combinaciones posibles se llaman **tablas de verdad**.

Esto es pensamiento computacional

Se usa el **razonamiento lógico**: entender el problema, aplicar de forma sistemática un procedimiento (un pequeño **algoritmo**) para evaluar cada parte de la afirmación a partir de la foto y combinar los resultados según el operador. Es una forma de **evaluación**: comprobar con datos si una afirmación es cierta, algo muy útil también para no creer todo lo que se lee en las redes.

Palabras clave y enlaces: Lógica, operadores lógicos, tabla de verdad · es.wikipedia.org/wiki/Tabla_de_verdad

7 Amigos

2024-LT-05 · Lituania
Dificultad: Media

En una clase hay siete castores: Alex, Bess, Cora, Dave, Eric, Fred y Gigi. Algunos se conocen y otros no.



En la tabla hay una marca entre dos castores si se conocen.

	Alex	Bess	Cora	Dave	Eric	Fred	Gigi
Alex		✓	✓				
Bess	✓						✓
Cora	✓				✓		
Dave						✓	
Eric			✓				
Fred				✓			
Gigi		✓					

Cada castor recibió un mensaje distinto de alguien de fuera de la clase y lo compartió con todos los castores de la clase a los que conoce. Cada vez que un castor recibe un mensaje nuevo de alguien de la clase, lo comparte inmediatamente con todos los castores de la clase a los que conoce.

¿Qué castor o castores de la clase recibieron menos mensajes en total?

A) Alex
B) Bess
C) Cora
D) Dave
E) Eric
F) Fred
G) Gigi

Puedes seleccionar más de una respuesta.

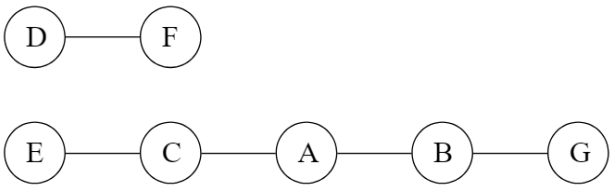
Respuesta correcta: Dave y Fred.

Explicación

La respuesta correcta es **Dave y Fred**.

Podemos dibujar un diagrama de las relaciones: cada castor es un círculo y, si dos castores se conocen, sus círculos se unen con una línea.

Dave solo conoce a Fred y Fred solo conoce a Dave, así que los unimos y no los conectamos con nadie más. Eric solo conoce a Cora; Cora conoce también a Alex; Alex conoce también a Bess y Bess conoce también a Gigi.



En el diagrama se ve que Dave y Fred solo pueden recibir 2 mensajes cada uno, mientras que Eric, Cora, Alex, Bess y Gigi reciben 5 cada uno. Por tanto, Dave y Fred son los que menos mensajes reciben.

Esto es informática

Los grafos son una herramienta de abstracción fundamental en informática. En esta tarea el grafo se presenta como una **matriz de adyacencia**: una matriz cuadrada cuyos elementos indican si cada par de vértices está conectado. El grafo es **no dirigido** (todas las aristas valen en los dos sentidos), y por eso la matriz es simétrica.

El problema está relacionado con la **conexidad** de un grafo: un mensaje llega exactamente a los castores de la misma **componente conexa**. La matriz de adyacencia es una forma habitual de representar grafos en los programas; la principal alternativa es la **lista de adyacencia**.

Esto es pensamiento computacional

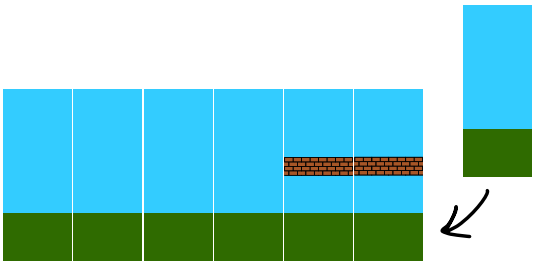
Hay que encontrar una forma de **modelar** las conexiones de los datos: pasar de la tabla a un dibujo de red (cambio de representación) hace evidente la estructura. También hay que **generalizar**: darse cuenta de que el número de mensajes que recibe un castor coincide con el tamaño de su grupo conectado, sin necesidad de simular la difusión de cada mensaje.

Palabras clave y enlaces: Grafo, matriz de adyacencia, componente conexa · es.wikipedia.org/wiki/Matriz_de_adyacencia · es.wikipedia.org/wiki/Conectividad_(teoría_de_grafos)

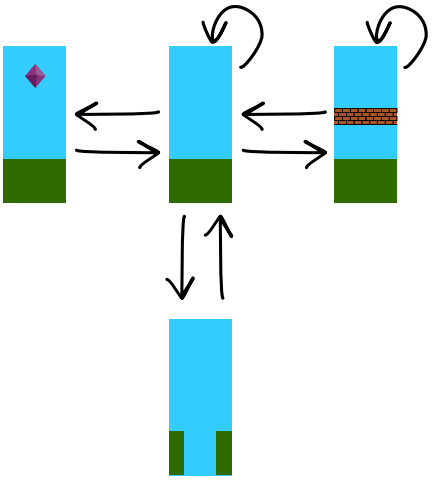
8 Superbebras

2024-DE-07 · Alemania
Dificultad: Fácil

En el videojuego Superbebras, el fondo es una secuencia de baldosas. El ordenador añade continuamente una baldosa nueva por la derecha y, a la vez, quita una por la izquierda. Así crea la ilusión de movimiento.



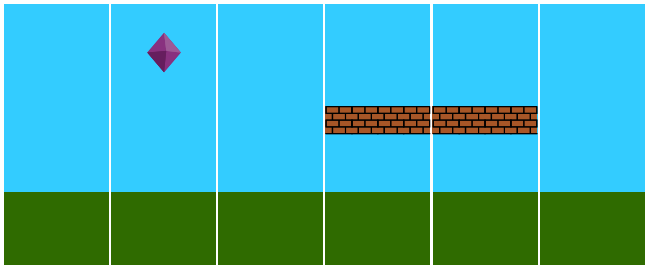
Para elegir la baldosa nueva, el ordenador usa el diagrama de la derecha: mira la baldosa anterior, comprueba las flechas que salen de ella y elige al azar una de las baldosas a las que apuntan.



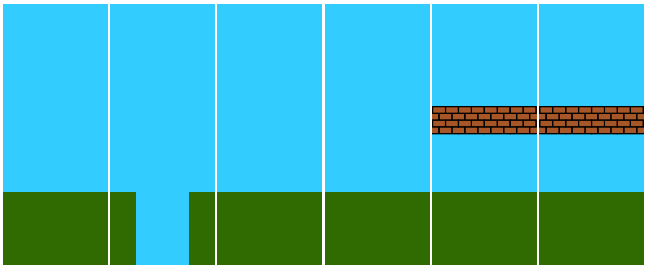
Por ejemplo, después de la baldosa  el ordenador puede elegir la baldosa  o la baldosa .

Una de las siguientes imágenes NO es un fondo válido de Superbebras. ¿Cuál?

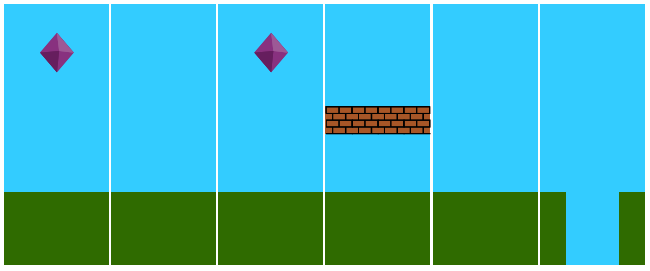
A)



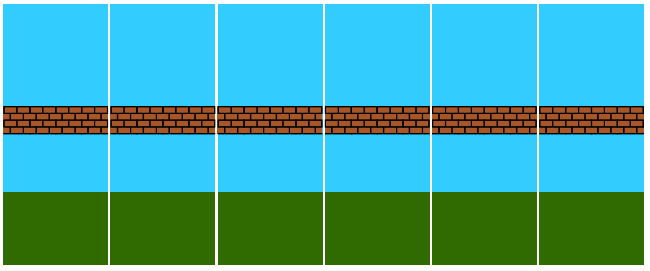
B)



C)



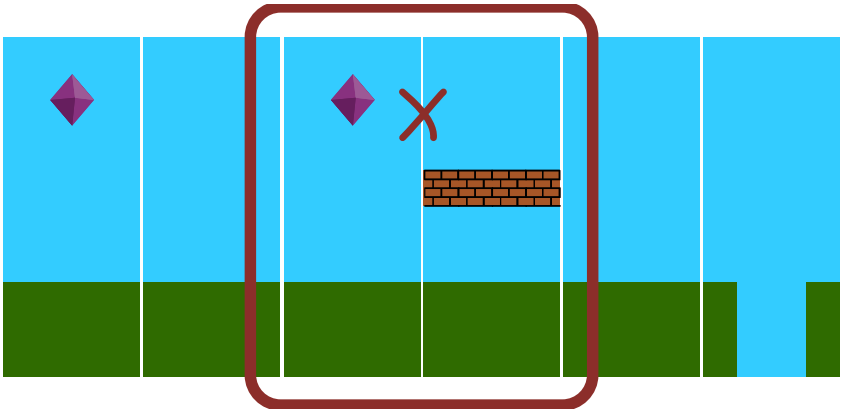
D)



Respuesta correcta: C.

Explicación

El fondo de la respuesta **C** no está construido del todo según el diagrama:



Una baldosa con un diamante solo puede ir seguida de , y aquí no es así.

Hay varias formas de encontrar la solución. Una estrategia directa es revisar cada fondo y comprobar cada baldosa con el diagrama. La más rápida es esta: empezar por las baldosas del diagrama de las que sale una sola flecha. Esas flechas son **restricciones**: A → B significa «a la baldosa A le tiene que seguir la baldosa B». Después se comprueba si cada fondo cumple esas restricciones.

Esto es informática

Algunos videojuegos, como los *endless runners* (juegos de carrera infinita), tienen un fondo que se desplaza horizontalmente y crea la ilusión de que el jugador avanza por un mundo de fantasía. A veces el fondo no es una imagen fija, sino que lo crea automáticamente el ordenador: es la **generación procedimental**. Normalmente se combinan al azar elementos pequeños para crear una enorme variedad de fondos.

Pero la combinación no puede ser totalmente aleatoria: debe cumplir ciertas reglas para no crear situaciones imposibles para el jugador. En la tarea, esas reglas se representan con un diagrama de baldosas y flechas, que es un **grafo dirigido**. Los grafos se usan para modelar todo tipo de situaciones; este en concreto funciona como un autómata que describe qué secuencias son válidas.

Esto es pensamiento computacional

Hay que leer y usar el grafo dirigido, que determina cómo se compone el fondo y es una **abstracción** del mecanismo que genera las imágenes. También hace falta **reconocimiento de patrones** para dividir visualmente cada fondo en baldosas y detectar las transiciones incorrectas. Por tanto, la tarea pone en juego el **pensamiento algorítmico**, la abstracción y el reconocimiento de patrones.

Palabras clave y enlaces: Grafo dirigido, generación procedimental, juegos de plataformas · es.wikipedia.org/wiki/Generación_por_procedimientos

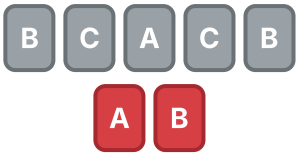
9 La puntuación más alta

2024-SK-04 · Eslovaquia
Dificultad: Media

En un juego se modifican secuencias de cartas con las letras A, B y C. El objetivo es conseguir la mayor puntuación posible sustituyendo algunas letras.

Por cada grupo de 2 letras iguales seguidas se obtienen 2 puntos; por cada grupo de 3 letras iguales seguidas, 3 puntos; y así sucesivamente.

Por ejemplo, esta secuencia de cinco cartas grises vale 0 puntos, porque no hay letras iguales juntas. Pero hay dos cartas rojas disponibles para mejorar la puntuación:

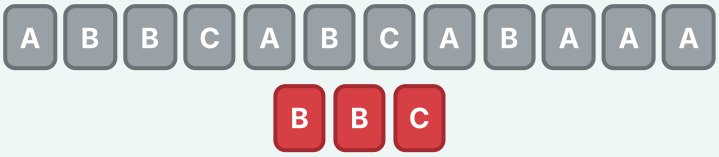


Podríamos tapar la primera C con la carta roja A y la segunda C con la carta roja B:



Ahora la puntuación es 2 + 2 = 4 puntos, por el grupo de dos A y el grupo de dos B. La primera B queda sola y no suma.

Tienes esta secuencia y tres cartas extra. Coloca las cartas rojas B, B y C sobre tres letras para conseguir la secuencia con la mayor puntuación posible.



Escribe encima de cada carta tapada la letra roja que pondrías.

Respuesta correcta: A B B C **C** B **B B** B A A A → 2 + 2 + 4 + 3 = 11 puntos.

Explicación

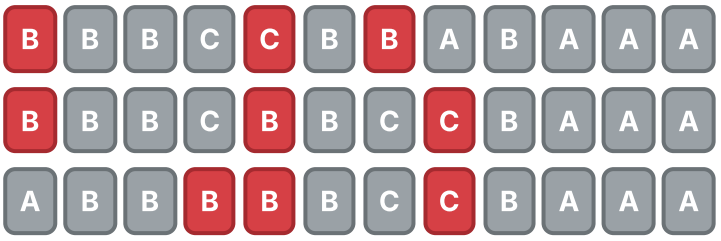
Respuesta correcta:



Esta secuencia vale 2 + 2 + 4 + 3 = **11 puntos**. No se puede hacer mejor, por estas razones:

- La A del extremo izquierdo no puede formar parte de ningún grupo, porque no hay cartas rojas con A.
- Todas las demás letras forman parte de un grupo y, por tanto, puntúan.
- El tamaño del grupo no importa: cada letra de un grupo aporta exactamente 1 punto.

Para comprobar que es la única forma de conseguir 11 puntos, se pueden repasar sistemáticamente las demás posibilidades: se colocan las tres cartas rojas de modo que cada una cree un grupo nuevo sin estropear uno existente. Salen estas secuencias:



Todas valen solo 10 puntos.

Esto es informática

Desde el punto de vista informático, es un **problema de optimización**: conseguir que algo sea lo más eficaz posible dentro de unas restricciones, es decir, encontrar la mejor de todas las soluciones posibles.

Por ejemplo, un compilador optimiza el código de un programa para que sea más rápido y ocupe menos sin cambiar su resultado. La compresión de datos también es una optimización: codificar la información con menos bits que la representación original para ahorrar espacio o transmitir menos datos.

En esta tarea colocamos las letras B, B y C para alargar un grupo de letras iguales o crear uno nuevo; calculamos la puntuación de cada secuencia y elegimos la mejor. Como en informática, buscamos la «mejor» solución dentro de las restricciones fijadas.

Esto es pensamiento computacional

Se usa la **descomposición**: dividir el problema en partes más manejables. Primero se identifican las secuencias que pueden ser relevantes, después se calcula la puntuación de cada una y, por último, se **evalúan** y se elige la mejor. La evaluación también sirve para decidir qué secuencias merece la pena considerar. Darse cuenta de que cada letra agrupada vale exactamente un punto (**generalización**) permite acotar el máximo sin probarlo todo.

Palabras clave y enlaces: Optimización, secuencias, búsqueda exhaustiva · es.wikipedia.org/wiki/Optimización_(matemática)

10 En busca del tesoro

2024-HU-03 · Hungría
Dificultad: Fácil



El capitán Boris busca un tesoro escondido en una isla.

Tiene un mapa que muestra dónde está el tesoro. El mapa está dividido en 16 regiones, marcadas con las letras de la A a la P. El capitán puede introducir en un aparato especial cualquier cantidad de regiones, y el aparato le dice si el tesoro está o no en alguna de ellas (por ejemplo, si el aparato responde «sí» para A + C + D, el tesoro está en la región A, en la C o en la D).



¿Cuál es el menor número de veces que el capitán Boris tendría que usar el aparato para estar seguro de encontrar siempre el tesoro?

Respuesta: _____

Respuesta correcta: 4.

Explicación

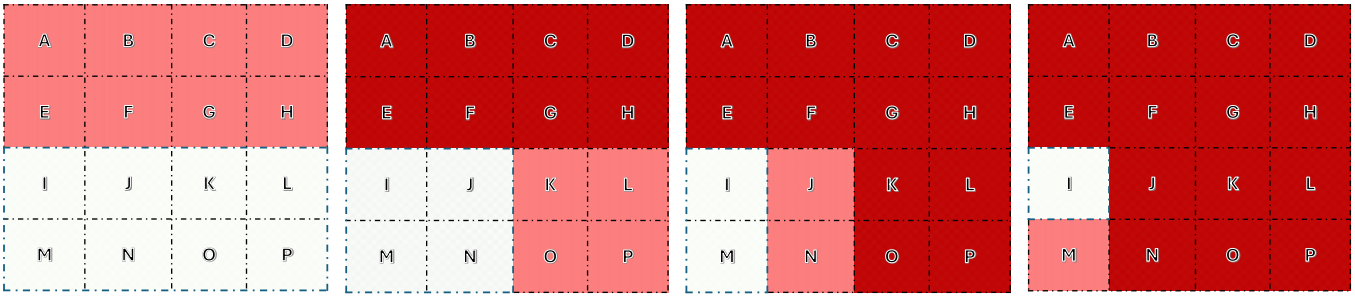
La respuesta es **4**.

Primero, el capitán debe introducir la mitad de las regiones. Puede ocurrir:

a. Que el aparato diga que el tesoro está en esas regiones: entonces divide esas regiones en dos mitades e introduce una de ellas.
b. Que el aparato diga que no está: entonces divide las regiones restantes en dos mitades e introduce una de ellas.

Repite el proceso hasta encontrar la región del tesoro. Cada consulta reduce a la mitad las posibilidades: 16 → 8 → 4 → 2 → 1, es decir, 4 consultas. Con menos no basta: con 3 respuestas sí/no solo se distinguen $2^3 = 8$ casos, y hay 16 regiones.

Por ejemplo, si introduce A, B, C, D, E, F, G, H y responde «no», puede introducir K, L, O, P (valdrían cualesquiera 4 de las restantes). Si vuelve a responder «no», introduce 2 de las que quedan (por ejemplo J y N). Si responde «no», el tesoro está en I o en M. Si introduce M y responde «no», sabe con seguridad que está en I.



Esto es informática

Para encontrar el tesoro de la forma más eficiente hay que dividir las regiones por la mitad. En informática se usa una técnica parecida para encontrar un elemento en una lista ordenada: dividir repetidamente por la mitad el espacio de búsqueda. Así se descarta la mitad de los elementos en cada paso y se encuentra el elemento en el menor número de pasos. Es la **búsqueda binaria**, que en una lista de n elementos necesita unos $\log_2 n$ pasos.

La búsqueda binaria tiene muchas aplicaciones: buscar una palabra en un diccionario, localizar un registro en una base de datos, encontrar en qué versión de un programa apareció un error (como hace `git bisect`), o decidir con el menor número de preguntas sí/no, como en el juego de «¿Quién es quién?».

Esto es pensamiento computacional

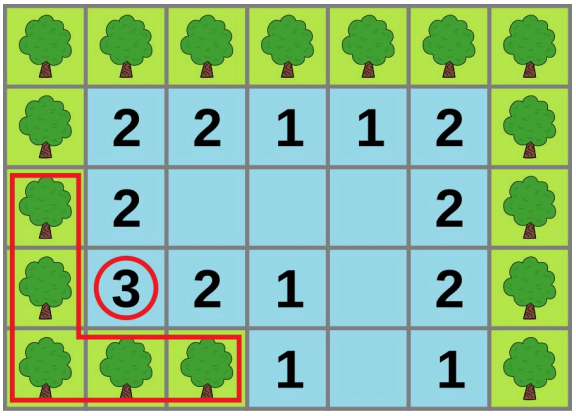
Es una tarea de **descomposición**: dividir el espacio de búsqueda en mitades para reducir el problema a otro más pequeño del mismo tipo (estrategia de «divide y vencerás»). Además hay que **evaluar** la estrategia en el peor caso: no basta con que funcione si hay suerte, debe garantizar el resultado siempre, y hay que argumentar que no existe una estrategia con menos consultas.

Palabras clave y enlaces: Búsqueda binaria, divide y vencerás, logaritmo · es.wikipedia.org/wiki/Búsqueda_binaria

11 El mapa de la comida

2024-IT-01b · Italia
Dificultad: Difícil

El castor Nico escondió sus provisiones bajo 9 de los 17 árboles que rodean el estanque. Hizo un mapa y escribió en las casillas de la orilla cuántos árboles de los que tocan cada casilla esconden comida.



Por ejemplo, la casilla con el número «3» indica que hay comida escondida bajo exactamente 3 de los 5 árboles que toca esa casilla, marcados en rojo en la imagen.

La castora Bella, amiga de Nico, no cree que sea buena idea. Piensa que así Nico no podrá saber dónde están todas sus provisiones ni reconocer los 9 árboles con comida. ¡Y tiene razón!

De hecho, Nico solo podrá identificar con seguridad 7 de los 9 árboles bajo los que escondió comida. Selecciona esos 7 árboles.

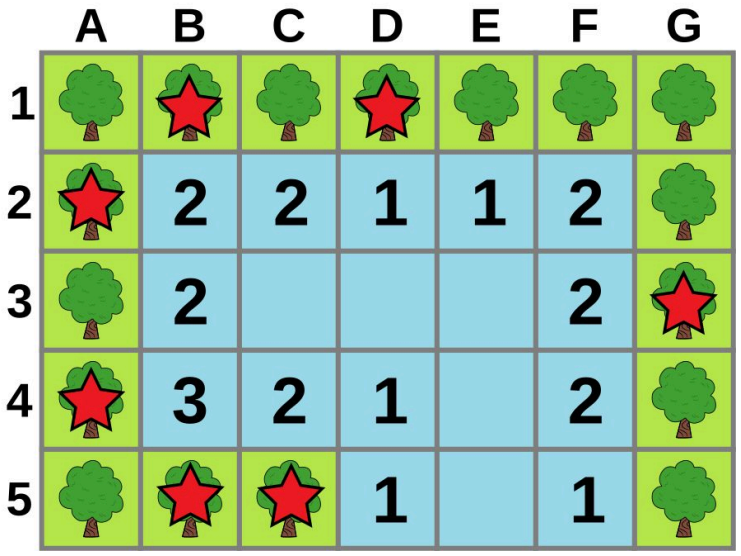
Rodea en este mapa los 7 árboles:

	A	B	C	D	E	F	G
1	🌳	🌳	🌳	🌳	🌳	🌳	🌳
2	🌳	2	2	1	1	2	🌳
3	🌳	2				2	🌳
4	🌳	3	2	1		2	🌳
5	🌳	🌳	🌳	1		1	🌳

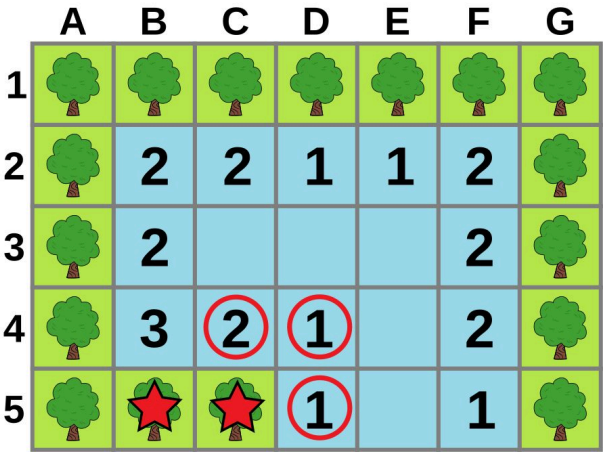
Respuesta correcta: A2, A4, B1, B5, C5, D1 y G3.

Explicación

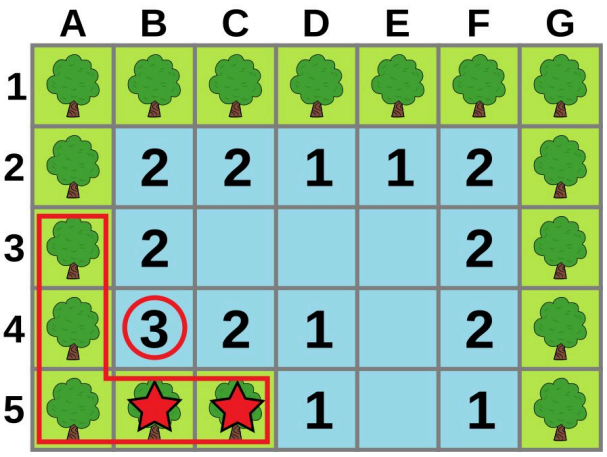
Los siete árboles con comida segura son los marcados con una estrella roja. Se han añadido coordenadas para facilitar la explicación.



Empezamos por la fila de abajo. El árbol C5 debe tener comida para cumplir los «1» de las casillas D4 y D5. El árbol B5 también, para cumplir el «2» de C4.



Ahora el «3» de B4 ya toca dos árboles con comida (B5 y C5), así que entre sus otros vecinos (A3, A4 y A5) tiene que haber exactamente uno con comida.



Dicho de otro modo, cada casilla obliga a elegir un número exacto de árboles de un conjunto. Recorriendo el mapa en el sentido de las agujas del reloj desde B4:

- B4: exactamente 1 de {A3, A4, A5}
- B3: exactamente 2 de {A2, A3, A4}
- B2: exactamente 2 de {A1, A2, A3, B1, C1}
- C2: exactamente 2 de {B1, C1, D1}
- D2: exactamente 1 de {C1, D1, E1}
- E2: exactamente 1 de {D1, E1, F1}
- F2: exactamente 2 de {E1, F1, G1, G2, G3}
- F3: exactamente 2 de {G2, G3, G4}
- F4: exactamente 2 de {G3, G4, G5}
- F5: exactamente 1 de {G4, G5}

Explorando las posibilidades se ve que no se puede elegir A5 en el primer conjunto: habría que elegir A2 en el segundo y además A3 o A4, lo que contradice el primero. Siguiendo así, quedan exactamente dos soluciones posibles para los árboles restantes:

```
{A2, A4, B1, D1, G1, G3, G4} o {A2, A4, B1, D1, G2, G3, G5}
```

Los árboles que están en las dos son A2, A4, B1, D1 y G3. Junto con B5 y C5, forman los **7 árboles seguros**: {A2, A4, B1, B5, C5, D1, G3}. Bella tenía razón: el lado derecho del estanque no se puede resolver con seguridad.

Esto es informática

La tarea se inspira en un famoso juego de lógica conocido con distintos nombres (Buscaminas, *Flower Field*), en el que también interviene la suerte: aunque se descubran zonas grandes, puede ocurrir que no haya información suficiente para seguir jugando sin riesgo. Es justo lo que pasa en esta tarea: el lado derecho del estanque no se puede resolver con certeza.

Si hay n árboles alrededor del estanque, hay 2^n configuraciones posibles. En general es mucho más fácil **comprobar** si una solución dada cumple las restricciones que **encontrarla**. Lo más interesante desde el punto de vista informático es la **formalización** del problema con una estructura de datos adecuada (como la lista de conjuntos de la explicación). Esa formalización sugiere que este tipo de problemas es, en general, computacionalmente difícil (de hecho, el Buscaminas generalizado es NP-completo) y que puede tener varias soluciones.

Esto es pensamiento computacional

La lógica es fundamental: para llegar a la respuesta hay que razonar sobre las restricciones y explorar unas cuantas posibilidades hasta sus últimas consecuencias, aunque a veces el problema no tenga una única solución. El **razonamiento lógico**, la **generación de soluciones candidatas** y la **satisfacción de restricciones** son procesos característicos del pensamiento computacional. También se usa la **descomposición**: el problema se resuelve por partes, empezando por las zonas más restringidas.

Palabras clave y enlaces: Programación lógica con restricciones, NP-completitud, Buscaminas · es.wikipedia.org/wiki/Programación_con_restricciones

12 La máquina de globos

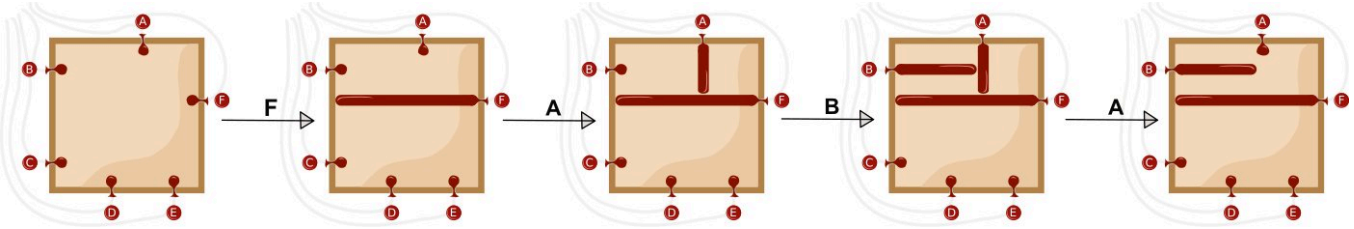
2024-DE-06a · Alemania
Dificultad: Difícil

Los castores tienen una máquina que crea imágenes inflando globos dentro de un marco cuadrado. Los globos están marcados con las letras A, B, C, D, E y F.

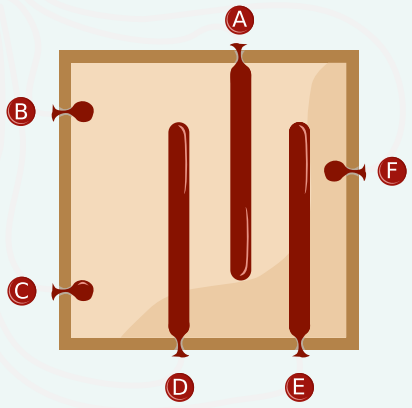
La máquina lee letras de una en una. Cuando lee una letra:

- Si el globo con esa letra está desinflado, lo infla hasta que toca otro globo o el borde opuesto del marco.
- Si no, desinfla el globo con esa letra.

Por ejemplo, si al principio todos los globos están desinflados y la máquina lee F, A, B y después A, hace lo siguiente:



Al principio todos los globos están desinflados y la máquina lee una secuencia de nueve letras. El resultado es la imagen de abajo. Escribe esa secuencia de nueve letras.



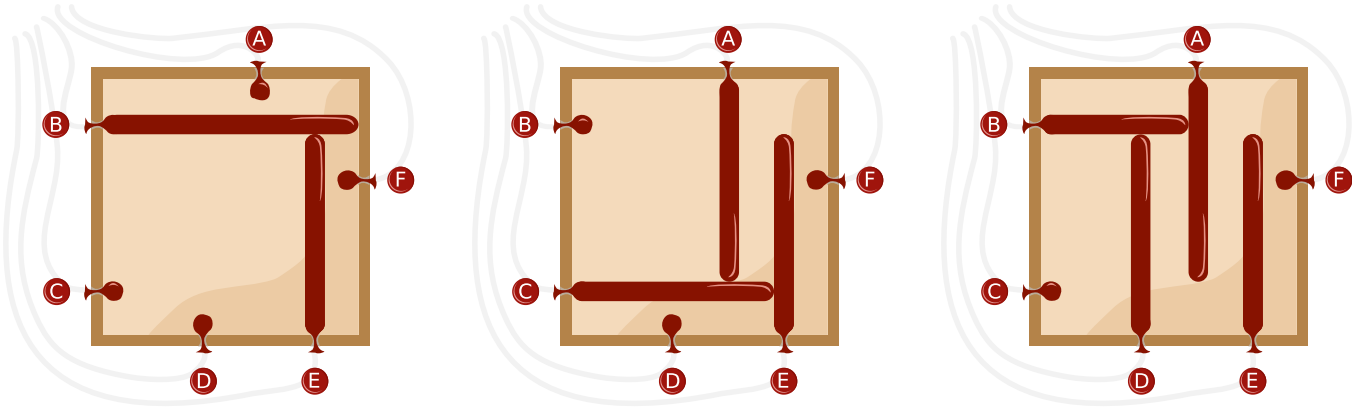
Respuesta correcta: Por ejemplo, B E B C A C B D B (hay 4 respuestas válidas).

Explicación

Hay **4 respuestas correctas** de nueve letras:

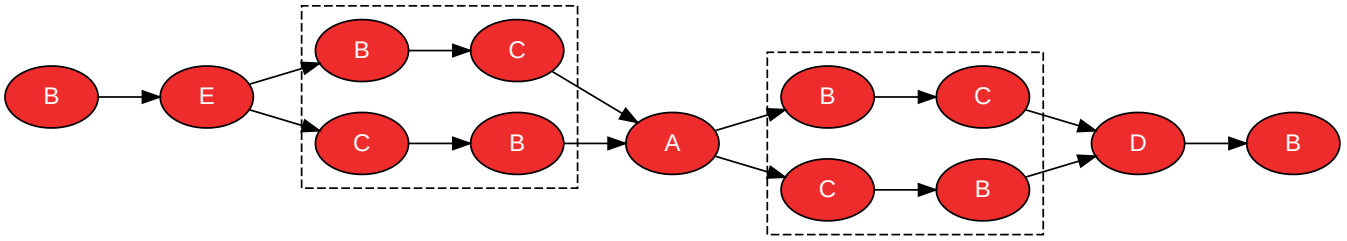
B E B C A C B D B · B E C B A C B D B · B E B C A B C D B · B E C B A B C D B

Explicamos en detalle la primera, B E B C A C B D B. Las imágenes muestran estados intermedios:



- Primera imagen: estado después de B E. Hay que inflar B antes que E para que B sirva de tope al inflar E.
- Segunda imagen: después de B E B C A. El globo C sirve de tope para A, así que debe inflarse antes que A. Además, B debe desinflarse antes de inflar A; si no, A no llegaría a la longitud necesaria.
- Tercera imagen: después de B E B C A C B D. B sirve de tope para D, y C debe desinflarse antes de inflar D.

Para encontrar todas las respuestas ayuda representar el problema con un **grafo dirigido**: una flecha de B a E indica que B debe inflarse antes que E, etc. Hay dos lugares en los que el orden de B y C puede variar (marcados con un rectángulo discontinuo).



Cada respuesta correcta es una **ordenación topológica** de este grafo; como hay 4, hay cuatro soluciones. Un truco para comprobar una respuesta: los globos que acaban inflados (A, D, E) deben aparecer un número impar de veces, y los que acaban desinflados (B, C, F), un número par (incluido 0).

Esto es informática

La secuencia de letras es un **programa** que controla una máquina. Cada letra es una instrucción que infla o desinfla un globo. Como en casi todos los programas de este tipo, el **orden de las instrucciones** es esencial: la secuencia B E produce una imagen distinta de E B.

El grafo dirigido de la explicación no tiene ciclos, por eso se llama **grafo acíclico dirigido** (DAG). Todo DAG tiene al menos una ordenación topológica y se conocen algoritmos rápidos para construirla. Los DAG tienen muchas aplicaciones en informática, por ejemplo en la planificación de tareas o en la compilación de programas (qué hay que compilar antes de qué).

Esto es pensamiento computacional

Hace falta **pensamiento algorítmico**: crear un algoritmo, un conjunto de instrucciones para la máquina, en el que el orden es esencial. El **reconocimiento de patrones** ayuda a deducir cuántas veces debe aparecer cada letra. Y es importante **evaluar** la solución: se puede ejecutar el algoritmo mentalmente, pero también usar atajos para descartar secuencias que nunca pueden ser correctas (por ejemplo, contando cuántas veces aparece cada letra). Si el recuento es correcto, todavía hay que comprobar que el resultado es el pedido.

Palabras clave y enlaces: Programación, secuencia, grafo acíclico dirigido, ordenación topológica · es.wikipedia.org/wiki/Ordenamiento_topológico · es.wikipedia.org/wiki/Grafo_acíclico_dirigido

13 Días soleados

2024-DE-04a · Alemania
Dificultad: — (fácil en 12–14)



El castor Tom dice: «Los días soleados, en todos los estanques hay al menos un castor nadando».

Su hermana Kim responde: «¡Eso no es verdad! El domingo pasado es un ejemplo que demuestra que te equivocas».

Supón que Kim tiene razón. Completa la frase:
El domingo pasado hizo sol y _____.

A) todos los estanques estaban llenos de castores nadando
B) ningún castor nadó en el estanque de la cascada helada
C) el castor Miguel nadó por todos los estanques
D) el castor Miguel no nadó nada

Respuesta correcta: B) ningún castor nadó en el estanque de la cascada helada.

Explicación

La solución es la **B**: ningún castor nadó en el estanque de la cascada helada.

En ese caso no todos los estanques tienen un castor nadando, porque el de la cascada helada no lo tiene. Además, sabemos que el domingo pasado hizo sol. Así que el domingo pasado es un ejemplo de día soleado con al menos un estanque sin castores nadando. Por tanto, Tom se equivoca al decir que todos los días soleados hay al menos un castor nadando en cada estanque.

Las demás opciones no demuestran que Tom esté equivocado:

- **A**: «Hizo sol y todos los estanques estaban llenos de castores nadando»: en todos los estanques había castores, lo que coincide con lo que dice Tom.
- **C**: «Hizo sol y el castor Miguel nadó por todos los estanques»: en todos había al menos un castor (Miguel), de nuevo de acuerdo con Tom.
- **D**: «Hizo sol y el castor Miguel no nadó nada»: puede que otros castores nadaran en todos los estanques, y entonces lo que dice Tom se seguiría cumpliendo.

Esto es informática

Tom y Kim discuten sobre el valor de verdad de la afirmación de Tom. Entran así en el terreno de la **lógica booleana**, en la que cada afirmación es verdadera o falsa. La lógica permite construir afirmaciones complejas con operadores como Y, O, NO y con **cuantificadores** como «para todo» y «existe al menos uno».

Para refutar una afirmación del tipo «para todo…» basta con un **contraejemplo**: un caso concreto que no la cumpla. La lógica ordena las discusiones y ayuda a evitar malentendidos. Además es la base de los ordenadores: los circuitos están hechos de puertas lógicas como NO e Y, y los lenguajes de programación incluyen condiciones lógicas como «SI es verano Y hace sol, ENTONCES enciende el aire acondicionado».

Esto es pensamiento computacional

Como los ordenadores se basan en la lógica, trabajar con ellos exige manejarla: traducir una situación del mundo real a una condición lógica, o encontrar la negación exacta de una condición para que un programa contemple todos los casos. Esta tarea entrena justamente eso, dentro del **análisis e interpretación de datos**. También hay algo de **descomposición**, porque conviene analizar cada opción por separado.

Palabras clave y enlaces: Lógica, cuantificador, contraejemplo · es.wikipedia.org/wiki/Cuantificador

14 Cadenas de palabras

2024-CA-02 · Canadá
Dificultad: Media



El profesor Castor está enseñando a leer a su alumnado. Para ayudarles crea **cadenas de palabras**: secuencias de palabras en las que, para obtener la palabra siguiente, se cambia exactamente una letra. Por ejemplo, MUG → MUD → MAD → FAD es una cadena de palabras.

El profesor Castor tiene estas nueve palabras: BOT, SAD, BAT, CAB, COT, BAD, COB, CAT y SAT. Las agrupa en tres cadenas de tres palabras cada una, de modo que cada una de las nueve palabras aparece en una y solo una de las cadenas.

(Las palabras están en inglés, como en la tarea original; para resolverla solo importan las letras.)

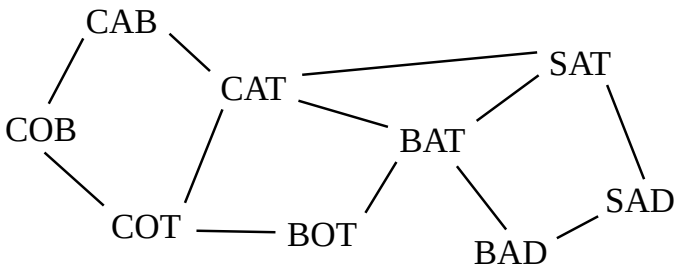
Ninguna de las siguientes secuencias incumple las reglas de las cadenas, pero una de ellas hace imposible formar tres cadenas sin repetir palabras. ¿Cuál no podría ser una de las cadenas del profesor Castor?

A) SAD → BAD → BAT
B) COT → COB → CAB
C) BAT → CAT → COT
D) CAT → CAB → COB

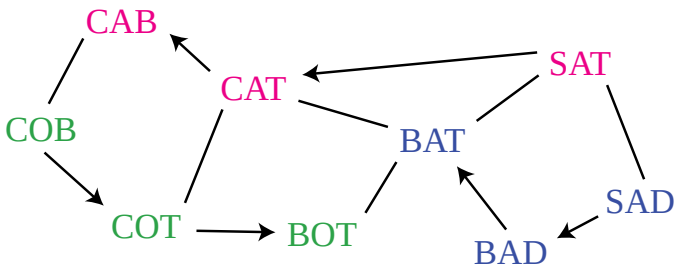
Respuesta correcta: C) BAT → CAT → COT.

Explicación

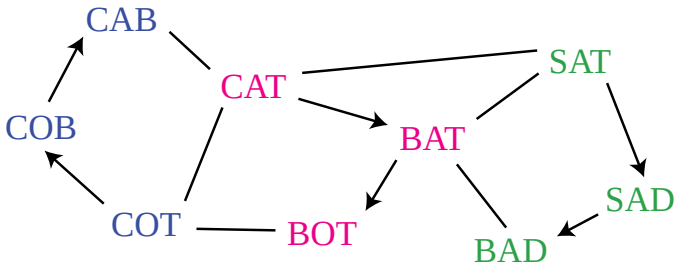
La respuesta es la **C) BAT → CAT → COT**. Para resolverlo ayuda dibujar un diagrama en el que dos palabras se unen con una línea si pueden ir seguidas en una cadena.



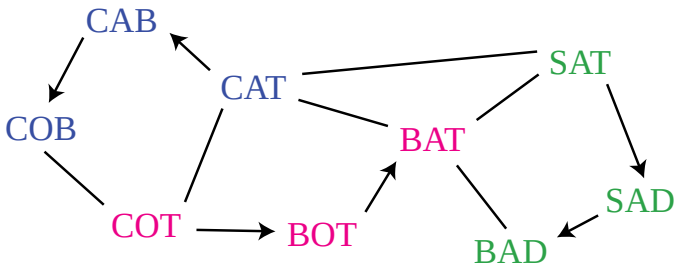
A es posible: las otras dos cadenas podrían ser SAT → CAT → CAB y COB → COT → BOT.



B es posible: las otras dos podrían ser CAT → BAT → BOT y SAT → SAD → BAD.



D es posible: las otras dos podrían ser COT → BOT → BAT y SAT → SAD → BAD.



C no es posible: en el diagrama, BOT solo está unida a BAT y a COT, así que tiene que estar en una cadena con al menos una de ellas. Como BAT y COT están las dos en la cadena de la opción C y BOT no, no queda ninguna cadena posible para BOT.

Esto es informática

En informática, este tipo de diagrama se llama **grafo** y sirve para mostrar conexiones. Está formado por un conjunto de vértices y un conjunto de aristas, cada una de las cuales une dos vértices. Aquí los vértices

son las palabras y hay una arista entre dos de ellas si se diferencian en exactamente una letra.

Una cadena de palabras es un **camino** por aristas conectadas. Como hay un camino entre dos vértices cualesquiera, el grafo es **conexo**. Agrupar las nueve palabras en tres cadenas de tres equivale a quitar aristas hasta obtener un grafo no conexo con tres **componentes** de tres vértices cada una (una partición en caminos). La pregunta consiste en comprobar si un conjunto dado de tres palabras puede ser una de esas componentes.

Esto es pensamiento computacional

Hay que **abstraer** el problema a un grafo para verlo de un vistazo y **evaluar** cada opción: comprobar si, tras usar esa cadena, las seis palabras restantes se pueden repartir en dos cadenas válidas. La clave para descartar rápido es fijarse en las palabras con pocas conexiones (como BOT), que son las que más restringen las soluciones.

Palabras clave y enlaces: Grafo, camino, grafo conexo, componente conexa · es.wikipedia.org/wiki/Grafo_conexo

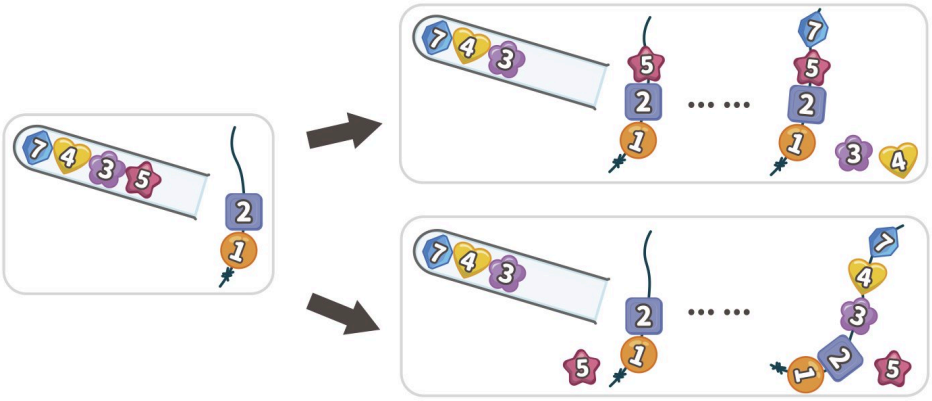
15 La pulsera Bebras

2024-TW-03 · Taiwán
Dificultad: Fácil

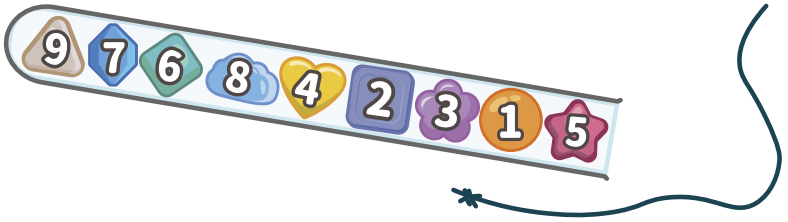
Castor está haciendo una pulsera Bebras. Saca de un tubo cuentas numeradas, de una en una. Con cada cuenta decide si la ensarta en el hilo o la aparta y no la usa. Pero solo puede ensartar una cuenta si:

- el hilo está vacío, o
- el número de la cuenta es mayor que el de la última cuenta ensartada.

En este ejemplo, la última cuenta del hilo es la n.º 2. Así que Castor puede ensartar la cuenta n.º 5 o apartarla.



Castor va a hacer una pulsera nueva con las cuentas de este tubo (la primera que sale es la 5):



¿Cuál es el mayor número de cuentas que puede ensartar?

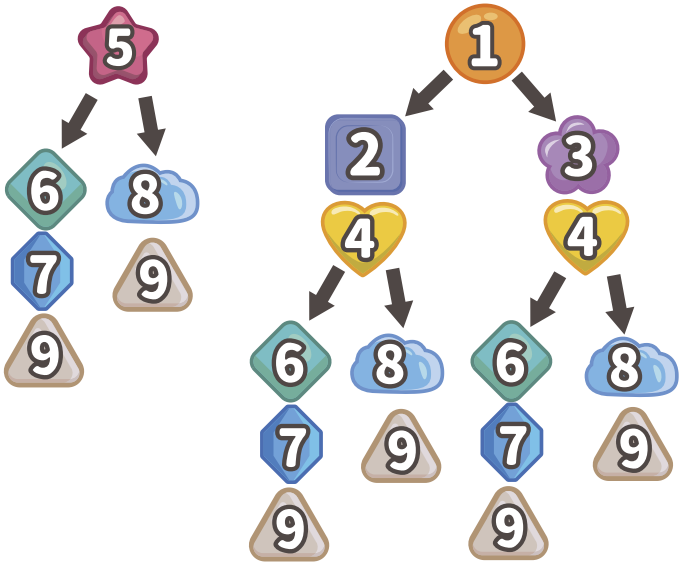
A) 3 cuentas
B) 4 cuentas
C) 5 cuentas
D) 6 cuentas
E) 7 cuentas

Respuesta correcta: D) 6 cuentas.

Explicación

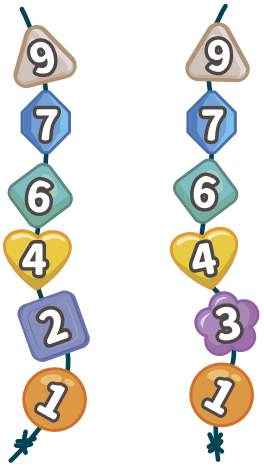
La respuesta correcta es la **D) 6 cuentas**.

Podemos construir todas las pulseras posibles con este tubo y ver cuáles tienen más cuentas. La primera cuenta que sale es la 5. Si se ensarta, después solo se pueden ensartar la 6, la 7, la 8 y la 9, y las secuencias más largas posibles son 5 6 7 9 y 5 8 9. Si se aparta la 5 y se ensarta primero la siguiente (la 1), hay más secuencias y más largas:



Si se empieza por cualquier otra cuenta, la secuencia más larga posible ya está incluida en una de las anteriores, que empiezan por la 1 o por la 5. Por ejemplo, empezar por la 2 lleva a secuencias como 2 4 6 7 9 o 2 4 8 9, incluidas en 1 2 4 6 7 9 y 1 2 4 8 9.

Por tanto, las pulseras con más cuentas empiezan por la 1 y tienen 6 cuentas: **1 2 4 6 7 9** y **1 3 4 6 7 9**.



Esto es informática

En informática, una **secuencia** es una serie ordenada de datos, como una cadena de caracteres o una secuencia de números. Cada miembro es un elemento, y una **subsecuencia** se forma eligiendo elementos de la secuencia original sin cambiar su orden.

Tanto las cuentas del tubo (en el orden en que salen) como las pulseras son secuencias de números si ignoramos la forma y el color. La tarea pide encontrar la **subsecuencia creciente más larga** (*longest increasing subsequence*) de la secuencia del tubo.

Con secuencias mucho más largas, enumerar todas las subsecuencias crecientes llevaría demasiado tiempo. Por suerte, existen algoritmos eficientes que la encuentran en un tiempo aproximadamente proporcional a n·log n: para un millón de elementos, del orden de veinte millones de pasos, nada para un ordenador moderno. Muchos problemas de subsecuencias se resuelven con la técnica de **programación dinámica**.

Esto es pensamiento computacional

Enumerar todas las soluciones posibles llevaría mucho tiempo, y muchas ni siquiera cumplirían los requisitos. Por eso la clave es encontrar una forma eficiente de identificar las soluciones posibles y descartar las imposibles: **pensamiento algorítmico**. Además, al decidir si se ensarta una cuenta hay que valorar cómo limita las cuentas que se podrán usar después; por ejemplo, ensartar la 5 impide usar de la 1 a la 4. Esa **simulación** de las consecuencias de cada decisión es otro aspecto del pensamiento computacional.

Palabras clave y enlaces: Subsecuencia creciente más larga, programación dinámica · en.wikipedia.org/wiki/Longest_increasing_subsequence · es.wikipedia.org/wiki/Programación_dinámica

Créditos y licencia

Las tareas proceden del taller internacional Bebras 2024 (Serbia). Cada tarea indica su código y el país que la propuso. Las imágenes son de los autores y autoras de las tareas originales.

Traducción y adaptación al castellano: Bebras España (bebras.ehu.es).

Copyright © Bebras – International Challenge on Informatics and Computational Thinking. Esta obra está bajo una licencia Creative Commons Reconocimiento-CompartirIgual 4.0 Internacional (CC BY-SA 4.0): creativecommons.org/licenses/by-sa/4.0.