Bebras PT

Edição 2023

Categoria
Cadetes (7º e 8º ano de escolaridade)

Tempo
45 minutos

Resolve tantos problemas quanto possível em 45 minutos.

Não é esperado que consigas resolver todos!

Responde apenas na folha de respostas.

É uma folha única, à parte, que deverás identificar com o teu nome.

Os enunciados e folhas de rascunho devem ser obrigatoriamente recolhidos no final da prova.

TREETREE2 FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO Deloitte. FUNDAÇÃO CALOUSTE GULBENKIAN

O **Bebras** é uma iniciativa internacional destinada a promover o pensamento computacional e a Informática (Ciência de Computadores). Foi desenhado para motivar alunos de todas as idades mesmo que não tenham experiência prévia.

Esta iniciativa começou em 2004 na Lituânia e todos os anos participam mais de 3 milhões de aluno de todo o mundo. O seu nome original vem dessa origem - "bebras" significa "castor" em lituano. A comunidade internacional adotou esse nome, porque os castores buscam a perfeição no seu dia-a-dia e são conhecidos por serem muito trabalhadores e inteligentes.

O que é o Pensamento Computacional?

O pensamento computacional é um conjunto de técnicas de resolução de problemas que envolve a maneira de expressar um problema e a sua solução de modo a que um computador (seja um humano ou máquina) a possa executar. É muito mais do que simplesmente saber programar. O desafio do Bebras promove precisamente este tipo de habilidades e conceitos como a capacidade de partir um problema complexo em problemas mais simples, o desenho de algoritmos, o reconhecimento de padrões ou a capacidade de generalizar e abstrair.

Organização Portuguesa

O Bebras começou em **Portugal** em 2019 e ano passado contou com a participação de mais de 70 mil estudantes de cerca de 500 escolas de todo o país.

É organizado por uma equipa de pessoas ligadas à Educação e à Ciência de Computadores da **TreeTree2** e do Departamento de Ciência de Computadores da Faculdade de Ciências da Universidade do Porto (**DCC/FCUP**)

Estrutura da Prova

Existe apenas uma fase a nível nacional, a qual é constituída por uma prova individual com 12 questões de três níveis de dificuldade diferentes, cuja pontuação é da seguinte forma:

Dificuldade	Correto	Incorreto	Não respondido
fácil	+6 pontos	-2 pontos	0 pontos
média	+9 pontos	-3 pontos	0 pontos
difícil	+12 pontos	-4 pontos	0 pontos

Sobre os Problemas

CC BY-NC-SA 4.0

Os problemas aqui colocados foram criados pela comunidade internacional da iniciativa Bebras e estão protegidos por uma licença da Creative Commons Atribuição-NãoComercial-CompartilhaIgual 4.0 Internacional.

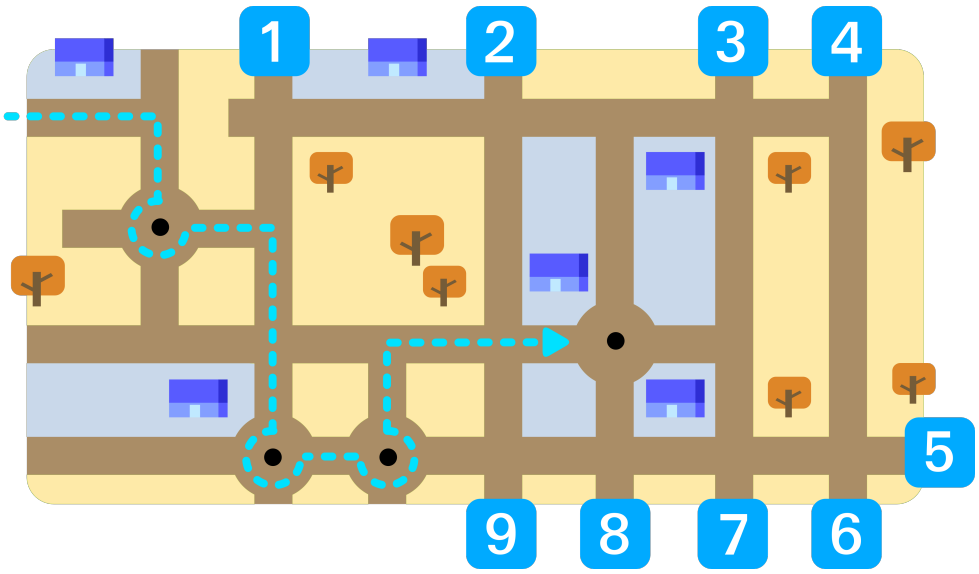
Os problemas da edição portuguesa foram escolhidos, traduzidos e adaptados pela organização portuguesa. Para a deste ano foram usados problemas com autores originários dos seguintes países:

- Arábia Saudita
- Canadá
- Chéquia
- China
- Eslováquia
- Estados Unidos
- Filipinas
- Hungria
- Índia
- Irlanda
- Itália
- Japão
- Lituânia
- Nova Zelândia
- Paquistão
- Perú
- Portugal
- Suiça
- Taiwan
- Turquia
- Uruguai
- Vietname

1. Carro Autónomo

O Filipe está a viajar no seu carro autónomo que anda de forma automática.

Sempre que o carro chega ao mesmo tipo de intersecção segue sempre o mesmo comportamento e usa sempre a mesma saída. Por exemplo, sempre que chega a uma rotunda ele sai na terceira saída possível.



Pergunta

A qual número num quadrado chega o carro do Filipe no final da sua viagem?

Respostas possíveis

(A) 1 **(B)** 2 **(C)** 3 **(D)** 4 **(E)** 5 **(F)** 6 **(G)** 7 **(H)** 8 **(I)** 9

1. Carro Autónomo (Resolução)

Solução

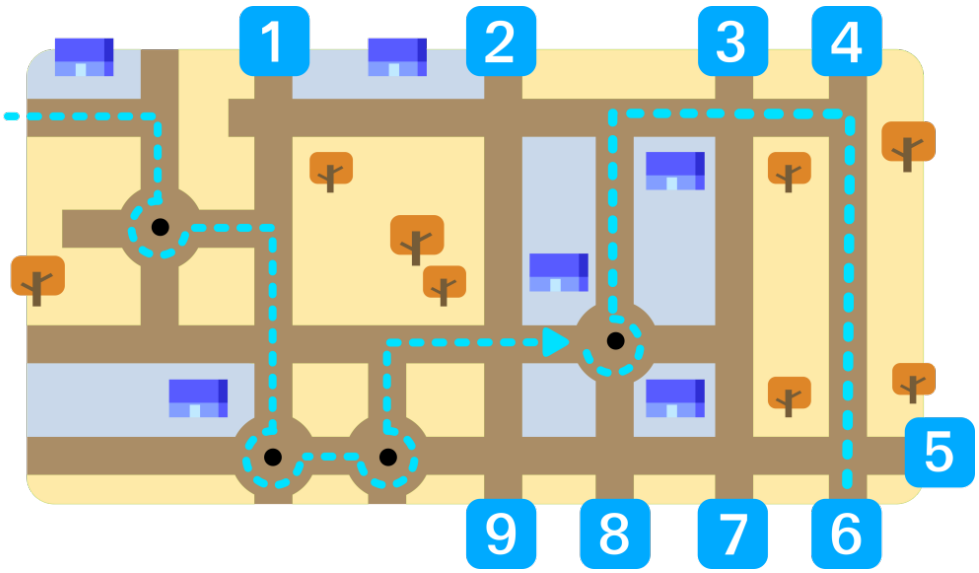
(F)

Resolução

A partir da imagem, pode concluir-se que o sistema de navegação segue as seguintes regras para cada tipo de cruzamento:

- Num cruzamento em T, o carro vira sempre à direita.
- Numa rotunda, o carro sai sempre na terceira saída para direita
- Num cruzamento +, o carro segue sempre em frente.

O cruzamento seguinte é uma rotunda, pelo que o carro vai pela terceira saída, indo para cima. O cruzamento seguinte é um cruzamento em T, pelo que o carro vai para a direita. Segue em frente no cruzamento + e vira à direita no cruzamento cruzamento em T na parte superior do mapa. Finalmente, o carro passa em frente pelo cruzamento + e pára em 6. O mapa abaixo mostra este trajeto:



Isto é Pensamento Computacional!

É uma situação típica em Ciência de Computadores a observação de um *sistema dinâmico* que é controlado por um *algoritmo* que segue determinadas regras. O comportamento futuro do sistema só pode ser previsto através de uma análise cuidadosa do algoritmo e da determinação das regras exactas que este segue. O sistema que é estudado pode ser um sistema da vida real ou um sistema de *software*, por exemplo, para efeitos de perceber possíveis erros de um programa.

2. Árvore Mágica

O castor David tem uma árvore mágica a crescer perto da sua casa.

Sempre que um pássaro pousa nela (), a árvore faz nascer 2 maçãs.

Sempre que um esquilo a trepa (), a árvore deixa cair 1 maçã (se tiver alguma).

Sempre que uma cobra visita a árvore (), todas as maçãs desaparecem instantaneamente!

Uma manhã, o David verifica que a árvore mágica contém **25** maçãs. O David passa então o resto do dia a fazer desenhos de todos os animais que vêm à árvore. Os desenhos, por ordem, são:

Início do dia

Final do dia

Pergunta

Quantas maçãs tem a árvore no final do dia?

Respostas possíveis

(A) 3

(B) 7

(C) 17

(D) 31

2. Árvore Mágica (Resolução)

Solução

(B)

Resolução

A resposta é a opção (B). Há 7 maçãs na árvore ao fim do dia.

Como todas as maçãs desaparecem instantaneamente sempre que uma cobra visita a árvore, podemos ignorar tudo o que acontece antes da chegada de uma cobra ( ). Depois da última cobra, quatro pássaros ( ) pousam na árvore, o que significa que dela brotarão 4 × 2 = 8 maçãs. Depois, um esquilo ( ) sobe à árvore, o que faz cair uma maçã, deixando 8 - 1 = 7 maçãs.

Isto é Pensamento Computacional!

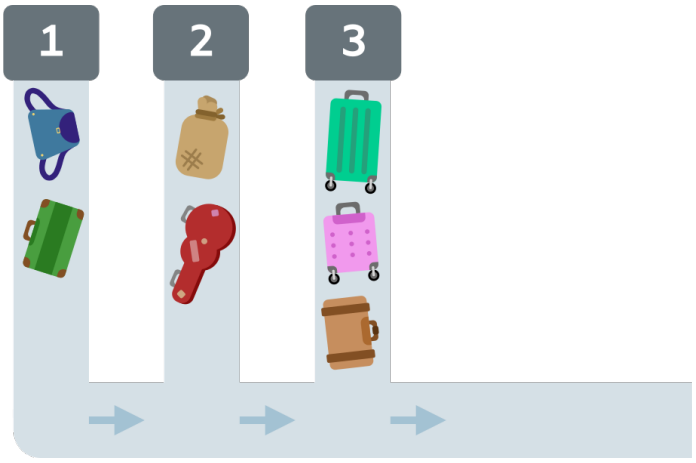
Este problema introduz as ideias subjacentes a dois conceitos fundamentais de programação.

O primeiro é a ideia de uma *variável*. Uma variável é usada para armazenar informações que um programa de computador precisa. O valor de uma variável pode mudar consoante o resto das instruções do programa. Neste problema, o número de maçãs na árvore é uma variável e o seu valor pode aumentar ( ), diminuir ( ) ou reiniciar ( ).

Para decidir como alterar o valor de uma variável, um programa de computador precisa de ter a capacidade de tomar decisões. Este é o segundo conceito fundamental de programação e é designado por *seleção*. A tomada de decisões faz-se utilizando instruções especiais chamadas instruções condicionais que permitem selecionar entre diferentes resultados possíveis. Normalmente, estas instruções assumem a forma "se isto, então aquilo". Neste problema, uma expressão condicional seria "se um pássaro pousar na árvore, então aumenta o número de maçãs em 2". Consegues encontrar outras expressões condicionais neste problema?

3. Check-in no Aeroporto

No aeroporto de Castorlândia, os passageiros podem deixar as suas malas em qualquer um dos 3 balcões de check-in mostrados abaixo.



Os operadores dos balcões colocam as malas, uma de cada vez, na passadeira rolante vertical. À medida que cada nova mala é colocada, o operador carrega num botão para fazer avançar a mala. Quando a mala atinge a passadeira horizontal, é automaticamente transferida para a mesma.

As malas que já estão nas passadeiras verticais são mostradas na imagem acima.

Pergunta

Qual das opções mostra uma possível ordem das malas na passadeira horizontal?

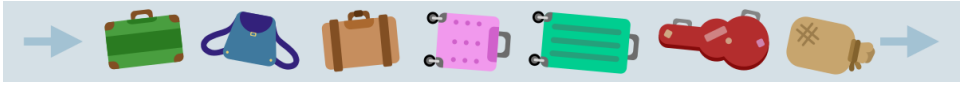
Respostas possíveis

(A)

(B)

(C)

(D)



3. Check-in no Aeroporto (Resolução)

Solução

(A)

Resolução

O que sabemos é que a ordem relativa das malas na passadeira horizontal tem de respeitar a ordem relativa em que estavam no mesmo balcão, ou sejam, mantêm a ordem entre si que tinham quando estava na mesma passadeira vertical.

A resposta correta é (A). A primeira mala é  do balcão 2, a segunda mala é  do balcão 1, a terceira mala é  do balcão 1, a quarta mala é  do balcão 3, a quinta mala é  do balcão 3, a sexta mala é  do balcão 2 e a sétima mala é  do balcão 3.

A seguir exemplificamos uma inconsistência em cada uma das hipóteses incorretas.

(B) está errada porque  deveria vir antes de  para respeitar a ordem em que estavam no balcão 1.

(C) está errada porque  deveria vir antes de  para respeitar a ordem em que estavam no balcão 2.

(D) está errada porque  deveria vir antes de  para respeitar a ordem em que estavam no balcão 3.

Isto é Pensamento Computacional!

Um *fila* é uma estrutura de dados utilizada para representar informações por ordem sequencial, sendo que primeiro elemento a entrar numa fila é sempre o primeiro a sair dela.

Neste problema, temos diferentes filas iniciais (representadas pelas passadeiras verticais) e, em todas elas, as malas são os seus elementos. Assim, para todos os sacos das filas verticais, essa propriedade da primeira mala a entrar ser a primeira a sair deve ser mantida.

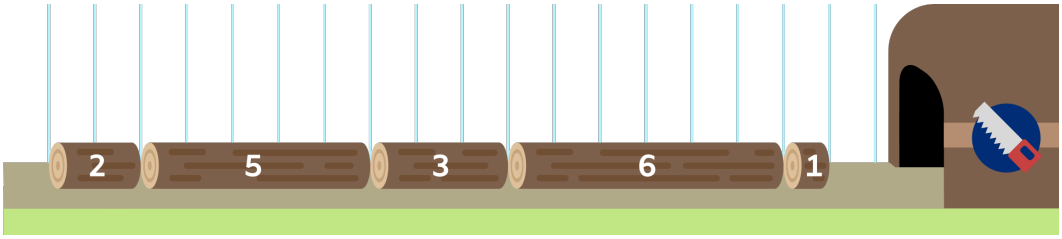
A passadeira horizontal pode ser entendida como um programador de um sistema operativo, que gere os pedidos de vários utilizadores (as passadeiras verticais). Consequentemente, há muitas possibilidades de retirar elementos das filas e, por isso, há muitas sequências possíveis de malas na passadeira horizontal.

4. Armazenamento de Troncos

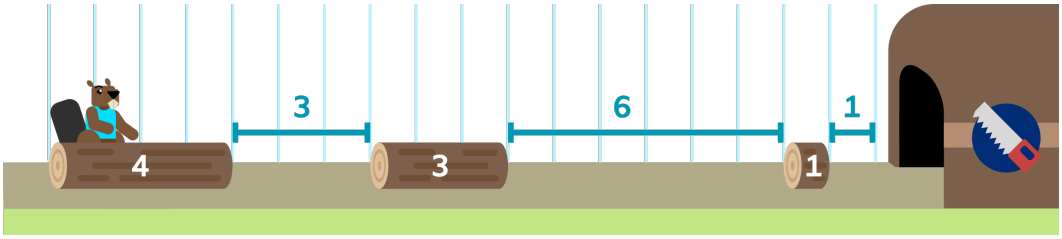
A castora Joana corta troncos de madeira de vários comprimentos e depois vende-os. Sempre que acaba de cortar um tronco, coloca-o no chão ao longo da estrada estreita de 18 metros, um a seguir ao outro, porque os troncos não cabem lado a lado.

Quando a Joana coloca um tronco no chão, coloca-o no primeiro lugar disponível a partir da esquerda onde o tronco cabe. Quando vende um determinado tronco, este é simplesmente retirado do lugar que ocupava anteriormente.

A Joana preparou, por esta ordem, troncos com comprimentos 2, 5, 3, 6 e 1 metros. Isto leva a esta disposição ao longo da estrada:



A seguir, ela vende os troncos com tamanhos de 6, 2 e 5 metros. De seguida, corta um novo tronco de 4 metros. Como o coloca o mais à esquerda possível, a estrada fica com este aspeto:



Pergunta

A Joana tem de cortar troncos com comprimentos de 1, 2, 3 e 4 metros. Qual das seguintes ordens lhe permitirá armazená-los todos na estrada, desde que utilize a mesma regra?

Respostas possíveis

(A) 

(B) 

(C) 

(D) 

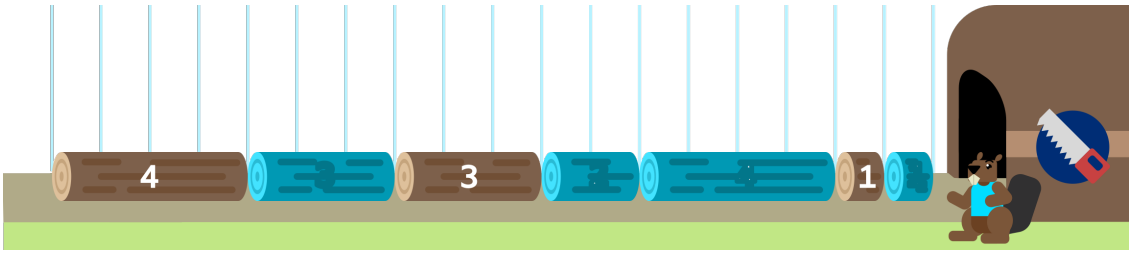
(E) 

4. Armazenamento de Troncos (Resolução)

Solução

(C)

Resolução



A resposta correcta é (C). Todas as outras respostas levam a que a Joana não consiga colocar o último tronco.

O estado da estrada antes de a Joana cortar os novos troncos, como se mostra na pergunta, tem três lugares livres: o lugar mais à esquerda tem 3 metros de comprimento, o lugar do meio tem 6 metros de comprimento e o último, à direita, tem 1 metro de comprimento.

A resposta (C) preenche totalmente o lugar mais à esquerda com o primeiro tronco, depois preenche totalmente o lugar do meio com os dois troncos seguintes e o último tronco entra diretamente no lugar mais à direita: funciona.

A resposta (A) preenche totalmente o lugar mais à esquerda com os dois primeiros troncos, mas depois coloca o tronco de 3 metros de comprimento no lugar do meio. Agora, o lugar do meio ficou reduzido a apenas 3 metros, e o último tronco, que tem 4 metros de comprimento, não cabe nem aí nem no lugar mais à direita.

A resposta (B) coloca o tronco de 1 metro de comprimento no lugar mais à esquerda e, a seguir, o tronco de 4 metros no lugar do meio. O lugar mais à esquerda ainda pode ser completamente preenchido com o terceiro tronco, que tem 2 metros de comprimento, mas não há espaço contíguo suficiente para colocar o último tronco.

A resposta (D) tem um problema semelhante, mas ainda mais cedo: o terceiro tronco de 4 metros já não pode ser colocado. De qualquer modo, os lugares mais à esquerda e mais à direita são demasiado pequenos e o lugar do meio está meio ocupado pelo segundo tronco de 3 metros.

Na resposta (E) é o último tronco que não consegue ser colocado. Consegues ver porquê?

Isto é Pensamento Computacional!

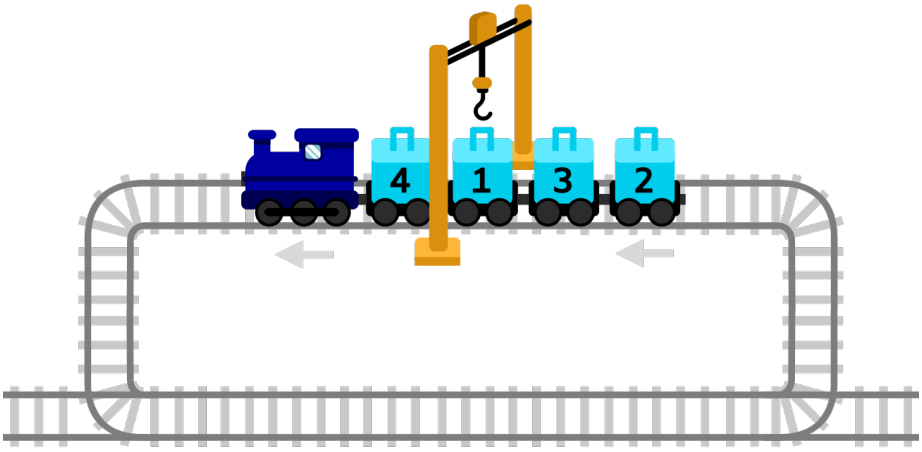
Podemos ver a estrada como uma memória dentro de um computador, e os registos a serem guardados e, mais tarde, vendidos, como processos informáticos que requerem uma certa quantidade de memória para funcionar. Neste exemplo, embora haja espaço suficiente para colocar teoricamente todos os registos, a ordem (e a estratégia) pela qual eles entram e saem torna por vezes impossível encaixá-los todos. O mesmo pode acontecer nos computadores: dependendo da ordem pela qual os pedaços de memória são atribuídos e libertados, pode ser impossível atribuir um novo pedaço de memória de um determinado tamanho, mesmo quando o total de memória livre é superior a esse tamanho.

Este problema é conhecido como *fragmentação de memória*. As estratégias de alocação mais inteligentes do que a da Joana ou os modelos de memória que permitem, por exemplo, a deslocação de blocos, ajudam a atenuar o problema da fragmentação. A fragmentação não é apenas um problema com a memória: a atribuição de espaço num disco para escrever ficheiros de vários tamanhos leva a problemas muito semelhantes.

5. Descargas

Um comboio de mercadorias tem várias carruagens, cada um com uma caixa numerada. Uma única grua é utilizada para descarregar. A grua está numa posição fixa. Para descarregar uma caixa, esta tem de ser posicionada diretamente por baixo da grua.

As caixas têm de ser descarregadas por ordem crescente a partir da caixa 1. O comboio só pode deslocar-se para a frente. Está numa via circular, pelo que pode dar a volta à via e regressar para que mais caixas possam ser descarregadas pela grua.



No exemplo acima, as caixas têm de ser descarregadas na sequência 1, 2, 3, 4.

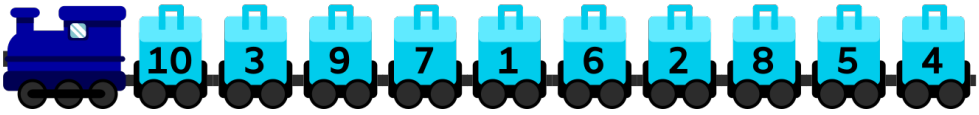
Na primeira volta de descarga, o comboio salta a caixa 4, descarrega a caixa 1, salta a caixa 3 e descarrega a caixa 2.

Na segunda volta, salta a caixa 4 e descarrega a caixa 3.

O comboio tem de voltar para uma terceira volta e descarrega a última caixa, a 4.

Pergunta

Quantas voltas serão necessárias para descarregar todas as caixas do comboio seguinte?



Respostas possíveis

(A) 4 **(B)** 5 **(C)** 6 **(D)** 7 **(E)** 8 **(F)** 9 **(G)** 10

5. Descargas (Resolução)

Solução

(D)

Resolução

A ordem necessária para descarregar é 1, 2, 3, 4, 5, 6, 7, 8, 9, 10. Se seguirmos o procedimento descrito acima, durante a primeira volta as carruagens 1 e 2 serão descarregadas em conjunto, depois 3 e 4 em conjunto, depois 5, depois 6, depois 7 e 8 em conjunto, depois 9 e finalmente 10. Isto corresponde a 7 voltas.

Em alternativa, pode observar-se o princípio geral de que, para cada número desta sequência 1, 2, ..., se o número seguinte surgir à sua esquerda no comboio, é necessária uma volta adicional. No caso da posição, se o 3 aparecer à esquerda do 2, então o 3 será ignorado para descarregar o 2, pelo que é necessária uma volta extra para colocar o 3 debaixo da grua. No desafio dado, o número de pares que estão fora de ordem é (2,3), (4,5), (5,6), (6,7), (8,9) e (9, 10), pelo que são necessárias 6 voltas extra, num total de 7 voltas.

Isto é Pensamento Computacional!

Para qualquer número da carruagem, se o número maior seguinte estiver à sua esquerda no comboio, chamamos a isto uma *"inversão"*. Para cada inversão, é necessária uma volta extra. Se contarmos o número de inversões, obtemos a resposta.

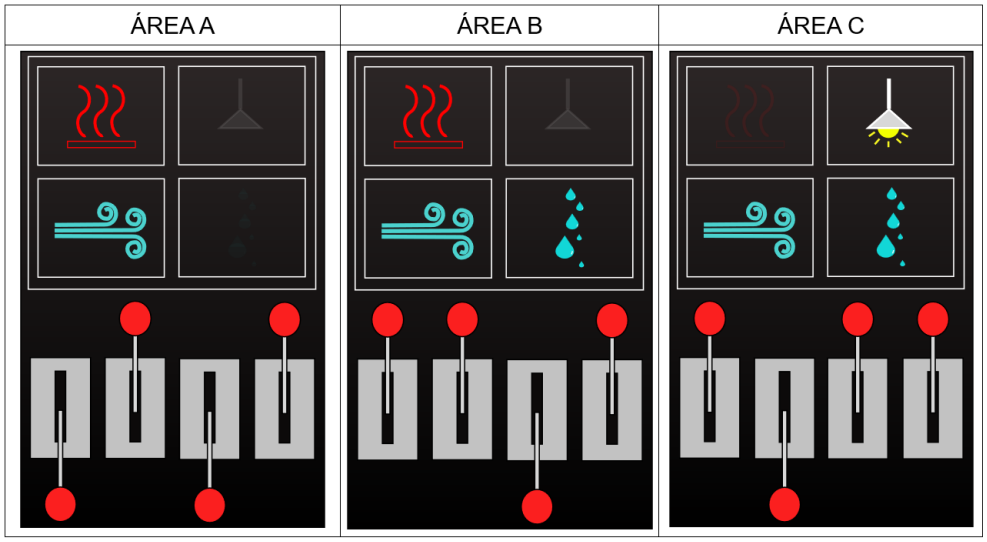
A contagem das inversões em relação a uma sequência desejada tem muitas aplicações. Para alguns algoritmos de ordenação, como o *bubble sort*, o número de inversões indica-nos quantas trocas são necessárias para ordenar uma dada sequência. Se dois clientes classificarem o mesmo conjunto de itens por ordem de preferência, o número de inversões nas suas classificações diz-nos até que ponto os seus gostos estão alinhados. Isto pode ser utilizado por exemplo por lojas *online* para identificar clientes "semelhantes"e fazer recomendações de produtos.

6. Alavancas

A Estação Espacial tem 3 diferentes áreas, todas com o mesmo painel de controlo com 4 alavancas para controlar os sistemas de calor, ventilação, luz e humidade. Todas as alavancas de cada painel funcionam da mesma forma e estão na mesma ordem.

Infelizmente alguém se esqueceu de colocar etiquetas para saber qual alavanca controla qual sistema!

Felizmente, observando as posições atuais das alavancas e o estado de cada sistema, é possível descobrir qual alavanca controla qual sistema:

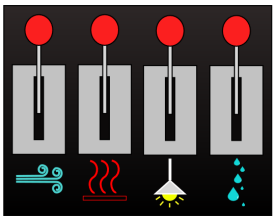


Pergunta

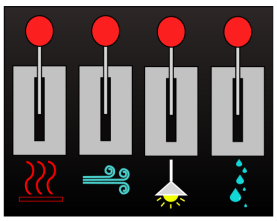
Qual é o sistema (aquecimento, ventilação, luz ou humidade) que está a ser controlado por cada alavanca?

Respostas possíveis

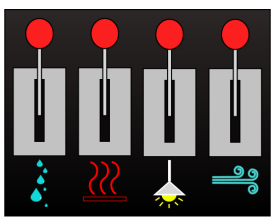
(A)



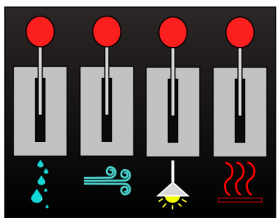
(C)



(B)



(D)



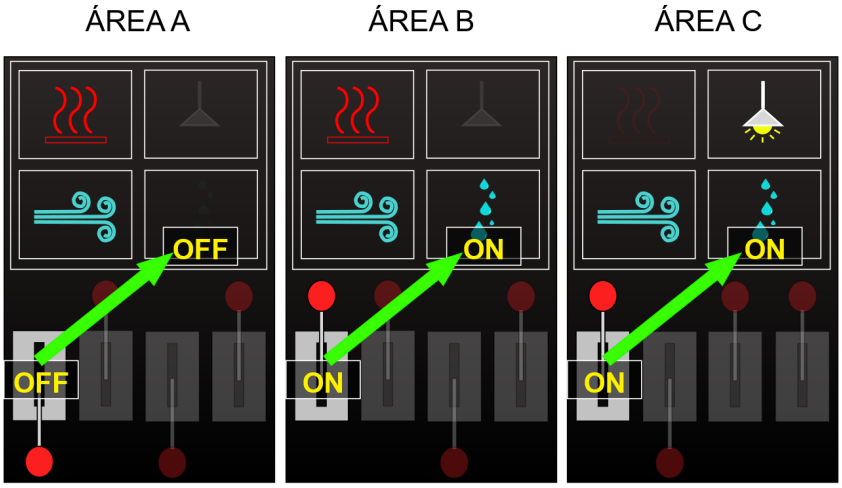
6. Alavancas (Resolução)

Solução

(B)

Resolução

A resposta correcta é a (B) (HUMIDADE / AQUECIMENTO / LUZ / VENTILAÇÃO). A forma mais fácil de detetar a função de cada alavanca é verificar o padrão de alteração de cada alavanca em cada posição e compará-lo com a alteração de cada sistema em todos os painéis. Em primeiro lugar, olhamos para as áreas A e B. As únicas diferenças entre elas são a humidade e a posição da 1ª alavanca. A partir daí, podemos deduzir que a alavanca posicionada para cima liga o sistema correspondente e que a 1ª alavanca controla a humidade.



As áreas A e B têm 2ª alavanca na posição LIGADO, mas está na posição DESLIGADO na área C. O sistema nos estados LIGADO-LIGADO-DESLIGADO é AQUECIMENTO. Portanto, a 2ª alavanca controla o AQUECIMENTO. Como LUZ está LIGADO apenas na àrea C, encontramos a alavanca que está na posição LIGADO apenas na Área C. Deduzimos que a 3ª alavanca controla a LUZ. Portanto, a 4ª alavanca controla a VENTILAÇÃO.

É muito importante avaliar sempre a posição das alavancas em todos os 3 painéis. Se avaliarmos apenas 2 painéis, as respostas erradas podem parecer as correctas por não termos informação suficiente.

A expressão *"funcionam da mesma forma"* ajuda a compreender que as posições LIGADO/DESLIGADO são as mesmas para todas as alavancas. Caso contrário, não é possível determinar o AQUECIMENTO e a LUZ.

Isto é Pensamento Computacional!

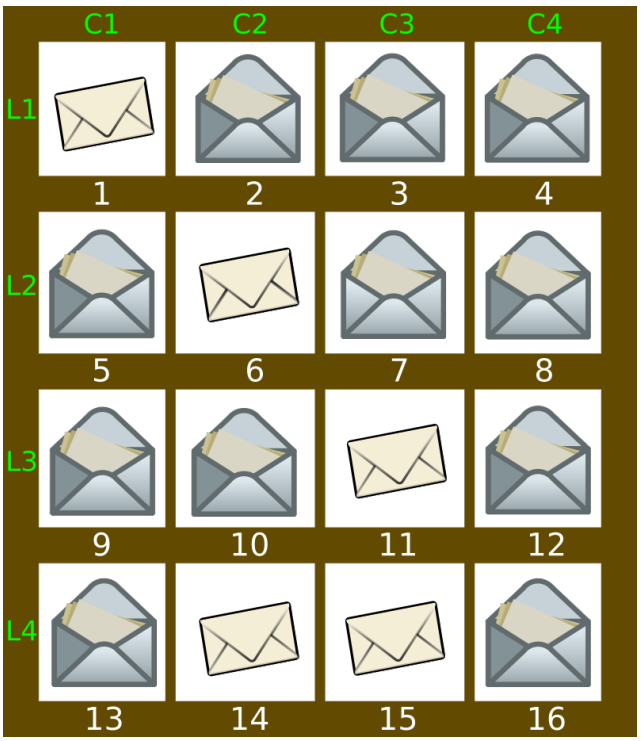
Este problema pode ajudar-nos a ter uma ideia geral dos conceitos de *representação de dados* e *números binários*. Aqui, as posições das alavancas são basicamente binárias, no sentido em que cada alavanca só pode ter 2 posições. Podemos chamar a essas posições CIMA BAIXO, LIGADO e DESLIGADO, ou podemos representar esses estados com números, como, por exemplo, 1 e 0. O estado e os dados recolhidos em dispositivos digitais são representados através de números. Carrega-se num botão e pode aparecer um 0 na memória do dispositivo. Roda-se uma alavanca e um 1 ou um 0 é armazenado algures na memória do dispositivo.

Mesmo quando os dados estão a ser transferidos entre dispositivos, todos estes 0s e 1s continuam presentes sob a forma de impulsos eléctricos. É assim que o seu o seu telemóvel envia comandos para os seus altifalantes sem fiospara colocar a sua música favorita. São todos 1's e 0's sob a forma de electrões ou ondas de rádio, mas que representam todo o tipo de informação rica... como as posições das alavancas numa Estação Espacial!

7. Cartas Fechadas

A República dos Castores mantém um armário cheio de cartas secretas. Entre as 16 cartas desse armário, numeradas de 1 a 16, 10 tinham sido abertas, enquanto as outras 6 ainda estavam seladas dentro dos seus envelopes.

Uma noite, um espião inimigo entrou sorrateiramente e abriu **uma** das cartas seladas. No entanto, esqueceu-se de a selar novamente. Na manhã seguinte, a República dos Castores inicia uma investigação depois de constatar que existem agora 11 cartas abertas, como se pode ver abaixo:



O guarda não se lembra de todos os pormenores, mas tem a certeza de que, **antes** de o espião se ter infiltrado:

- O número de cartas abertas nas colunas C2 e C4 combinadas era par.
- O número de cartas abertas nas colunas C3 e C4 combinadas era par.
- O número de cartas abertas nas linhas L2 e L4 combinadas era par.
- O número de cartas abertas nas linhas L3 e L4 combinadas era par.

Pergunta

Qual das cartas foi aberta pelo espião inimigo?

Respostas possíveis

(A) 5 **(B)** 9 **(C)** 10 **(D)** 13

(E) Não existem informações suficientes para identificar a carta aberta

7. Cartas Fechadas (Resolução)

Solução

(D)

Resolução

A resposta correcta é 13 cartas e podemos seguir esta linha de raciocínio:

1. Há um número par de cartas abertas nas colunas C2 e C4 combinadas, o que corresponde à recordação do guarda. Como só há uma carta aberta pelo espião, isso implica que a carta aberta pelo espião deve estar na coluna C1 ou C3.
2. Há um número par de cartas abertas nas colunas C3 e C4 combinadas, o que corresponde à recordação do guarda. Dada a afirmação anterior, isto implica que a carta aberta pelo espião deve estar na coluna C1.
3. Há um número ímpar de cartas abertas nas linhas L2 e L4 combinadas, o que não corresponde à recordação do guarda. Isto implica que a carta aberta pelo espião deve estar na linha L2 ou L4.
4. Há um número ímpar de cartas abertas nas linhas L3 e L4 combinadas, o que não corresponde à recordação do guarda. Dada a afirmação anterior, isto implica que a carta aberta pelo espião deve estar na linha L4.

Portanto, a carta lida pelo espião está na coluna C1 e na linha L4, o que aponta para (D) 13.

Isto é Pensamento Computacional!

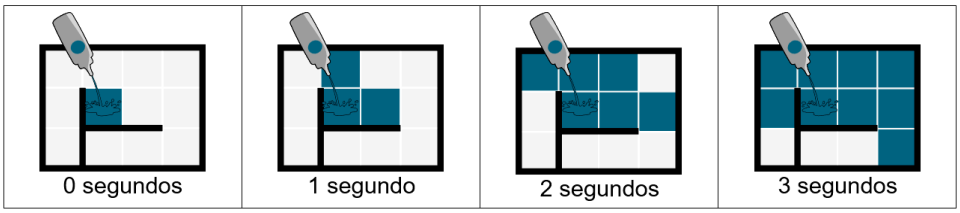
Este desafio introduz o conceito de *códigos de correção de erros*. Os dados digitais são essencialmente uma sequência de *bits*: 1s e 0s. Quando passam por uma rede, podem ser corrompidos devido a ruído ou à ação de entidades maliciosas. Os dados armazenados em DVD também podem ser corrompidos se estes dispositivos de armazenamento forem riscados. Assim, é importante dispor de um esquema capaz não só de detetar a ocorrência de corrupção (*deteção de erros*), mas também de corrigir os *bits* corrompidos (*correção de erros*).

O primeiro código moderno de correção de erros foi introduzido por Richard Hamming em 1950. Este desafio inspira-se especificamente na ideia do *código de Hamming [15,11]*, em que as cartas fechadas e abertas correspondem a 0s e 1s, respetivamente. Este esquema de codificação pega num dado de 11 *bits* e insere 4 *bits* de paridade. Estes *bits* de paridade ocupam os lugares 2, 3, 5 e 9, enquanto os *bits* dos dados originais ocupam os restantes lugares (exceto o lugar 1). Os *bits* de paridade são definidos de modo a que haja um número par de 1s na 2ª e 4ª colunas, 3ª e 4ª colunas, 2ª e 4ª linhas, e 3ª e 4ª linhas. Finalmente, o *bit* no lugar 1 é definido de modo a que haja um número par de 1s no total.

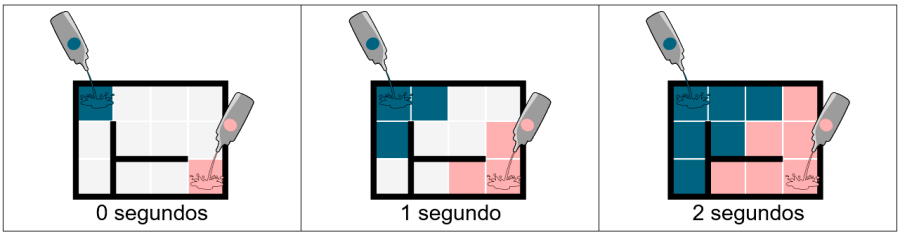
Como se vê na solução apresentada, este esquema permite que o *bit* corrompido seja identificado e posteriormente corrigido, desde que haja no máximo 1 *bit* corrompido. Embora sejam necessários esquemas de correção de erros mais sofisticados para corrigir 2 ou mais *bits* corrompidos, a eficiência do código de Hamming contribui para ainda hoje em dia seja utilizado em sítios como memórias de computador.

8. Aguarela

Quando os castores põem aguarela num labirinto, a cor espalha-se para os quadrados vizinhos a cada segundo. A cor não se espalha através das paredes, como podes ver abaixo.

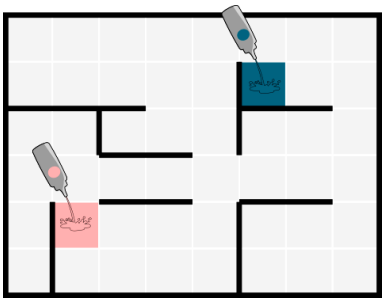


Se os castores puserem mais do que uma aguarela no labirinto, a primeira cor que chegar ao quadrado vai preenchê-lo na totalidade. Quando as cores chegam a um quadrado ao mesmo tempo, o quadrado fica da cor mais escura.



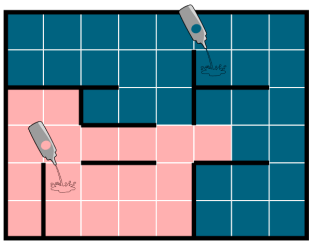
Pergunta

Os castores puseram duas cores no labirinto mostrado na imagem à direita. Qual será o aspeto do labirinto quando todos os quadrados estiverem preenchidos com uma cor?

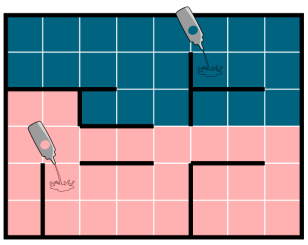


Respostas possíveis

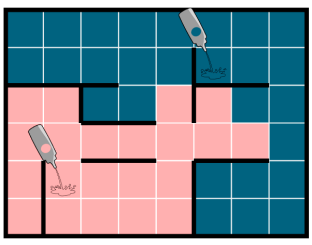
(A)



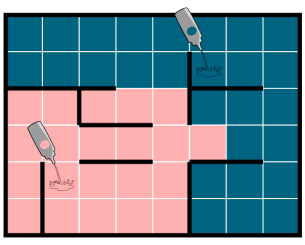
(C)



(B)



(D)



8. Aguarela (Resolução)

Solução

(A)

Resolução

A resposta correta é a (A). A imagem abaixo mostra o estado do labirinto a cada segundo:

Isto é Pensamento Computacional!

A configuração do problema assemelha-se a uma matriz bidimensional, o que basicamente significa uma tabela com linhas e colunas. Esta forma de representar os dados é bastante funcional para simular o estado do labirinto segundo a segundo e resolver o problema.

No entanto, também é possível representá-lo como um *grafo* em que cada quadrado está ligado aos seus vizinhos. Este grafo permite-nos identificar rapidamente que cor chegará a um quadrado sem simular todo o cenário segundo a segundo. Poderia usa-se uma *pesquisa em largura*, que começa numa qualquer nó do grafo e visita todos os nós no nível de profundidade atual antes de passar para os nós no nível de profundidade seguinte. A pesquisa em largura pode ser utilizada para resolver muitos problemas em teoria de grafos.

9. BebrasGPT

O BebrasGPT é um *chatbot* recentemente desenvolvido para produzir frases de três palavras, prevendo a palavra seguinte com base na sequência de palavras anterior. Cada palavra é escolhida uma a uma, sendo que a palavra seguinte é escolhida com base nas probabilidades.

As tabelas abaixo mostram algumas dessas probabilidades.

Probabilidades para a segunda palavra:

	"adoram"	*"odeiam"*
"Gatos"	0,7	0,3
"Castores"	0,6	0,4

Probabilidades para a terceira palavra:

	"nadar"	*"correr"*
"Gatos adoram"	0,2	0,8
"Gatos odeiam"	0,9	0,1
"Castores adoram"	0,7	0,3
"Castores odeiam"	0,1	0,9

Por exemplo, se a frase começa com a palavra *"Gatos"*, a probabilidade de a frase de 3 palavras ser *"Gatos adoram correr"* é de 0,56 porque:

- a probabilidade da segunda palavra ser *"adoram"* se a palavra anterior for *"Gatos"* é de 0,7;
- a probabilidade da palavra seguinte ser *"correr"* se a sequência anterior for *"Gatos adoram"* é de 0,8;
- portanto, como o modelo prevê as palavras uma a uma, a probabilidade é $0{,}7 \times 0{,}8 = 0{,}56$.

Pergunta

Se uma frase começa com a palavra "Castores", qual é o resultado mais provável do BebrasGPT?

Respostas possíveis

(A) *"Castores odeiam nadar"*

(B) *"Castores odeiam correr"*

(C) *"Castores adoram nadar"*

(D) *"Castores adoram correr"*

9. BebrasGPT (Resolução)

Solução

(C)

Resolução

A resposta correta é a (C): *"Castores adoram nadar"*. As probabilidades de cada frase são as seguintes:

- (A) *"Castores odeiam nadar"*: $0,4 \times 0,1 = 0,04$
- (B) *"Castores odeiam correr"*: $0,4 \times 0,9 = 0,36$
- (C) *"Castores adoram nadar"*: $0,6 \times 0,7 = 0,42$
- (D) *"Castores adoram correr"*: $0,6 \times 0,3 = 0,18$

Isto é Pensamento Computacional!

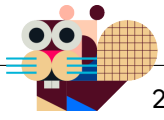
Este exercício envolve um exemplo de um modelo artificial para a modelação da linguagem e a produção de texto. Trata-se de um m*odelo probabilístico*, o que significa que cada saída do modelo se baseia em probabilidades. Há muitos tipos de modelos. Neste caso, o modelo está a produzir uma palavra de cada vez, com base em toda a sequência de palavras anteriores, o que é semelhante a modelos como o ChatGPT. É claro que estes modelos são muito, muito maiores do que o modelo apresentado no problema.

Um aspeto importante relacionado com a *inteligência artificial* e a *aprendizagem automática* é a diferença entre dois tipos de *algoritmos*:

- o algoritmo de aprendizagem;
- e o algoritmo de inferência.

O algoritmo de aprendizagem é utilizado para "treinar"o modelo. Isto corresponde, no nosso exemplo, a encontrar os valores das probabilidades que devem constar da tabela acima. No nosso caso, isto já foi feito. O modelo já está treinado. Normalmente, esta é a parte mais difícil em termos de teoria. Por exemplo, o ChatGPT esteve a treinar durante muitos meses em várias GPUs diferentes, que custaram milhões de euros.

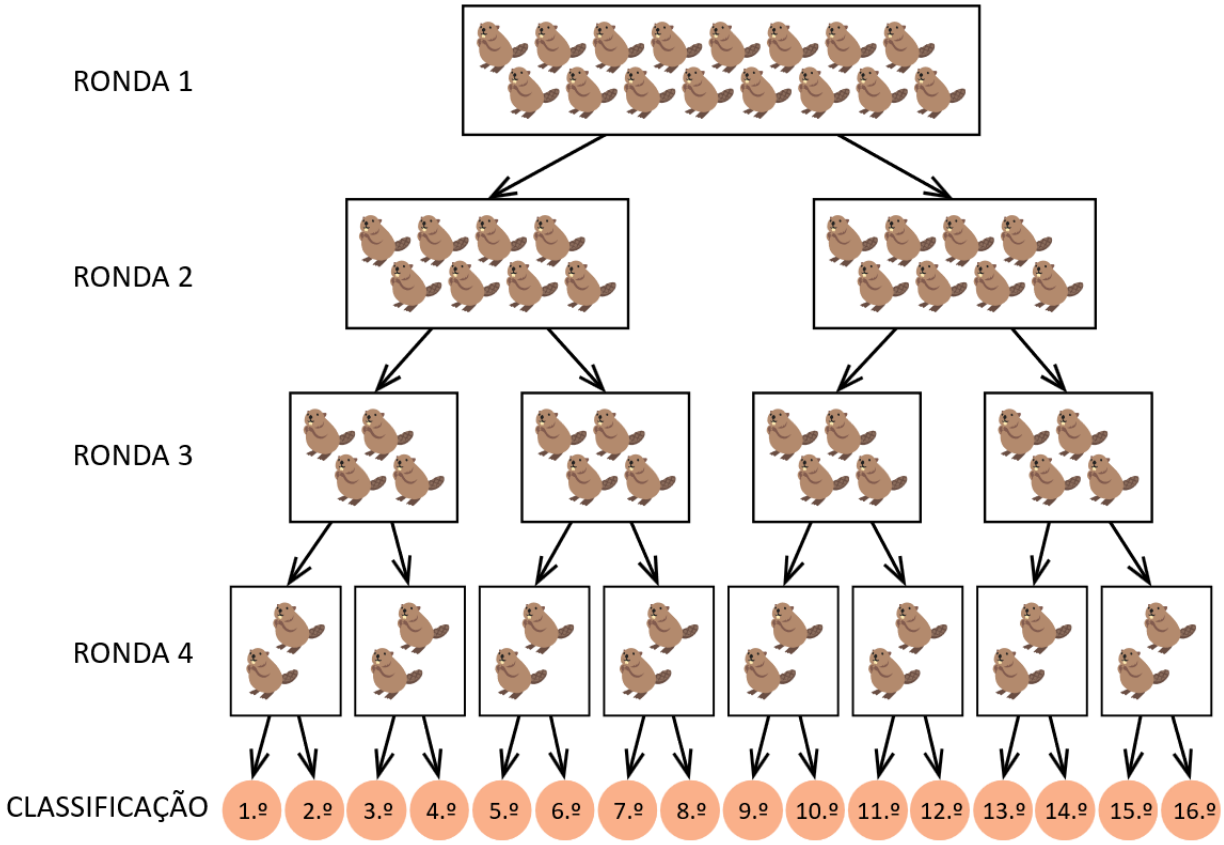
O algoritmo de inferência é o que utilizamos após o treino para produzir o resultado. Por exemplo, quando escrevemos a pergunta *"Olá, ChatGPT. O que é a aprendizagem automática?"*, o modelo pega nessa sequência de palavras e utiliza algo equivalente a uma tabela de probabilidades muito grande para produzir a resposta à pergunta. Apesar de o custo de executar o algoritmo de inferência uma vez ser muito inferior ao do treino, se o modelo for utilizado por milhões de pessoas todos os dias (como o ChatGPT), o custo da inferência é superior ao do treino (que foi um custo único).



10. Bebrasbol

Hoje é o torneio anual de Bebrasbol. Dezasseis jogadores chegaram para competir em quatro rondas, a fim de determinar a sua classificação geral do 1º ao 16º lugar.

Todos os dezasseis jogadores competem juntos na ronda 1, mas depois de cada ronda os jogadores separam-se. Os jogadores vencedores seguem a seta da esquerda para a ronda seguinte da competição (ou classificação final). Os jogadores derrotados seguem a seta da direita para a ronda seguinte da competição (ou para a classificação final).



Por exemplo, um jogador que ganhe durante as rondas 1 e 2, mas perca durante as rondas 3 e 4, fica com uma classificação de 4º lugar.

Pergunta

O Nuno era um jogador do torneio Bebrasbol. Se o Nuno perdeu exatamente uma única ronda (e venceu três), em qual dos seguintes lugares ele **não** *pode ter ficado?*

Resposta

(A) 2º **(B)** 3º **(C)** 5º **(D)** 7º **(E)** 9º

10. Bebrasbol (Resolução)

Solução

(D)

Resolução

Como há quatro rondas e Nuno perdeu exatamente numa ronda, há quatro cenários possíveis:

1. Perdeu na ronda 1 e ganhou em todas as outras. Assim, o Nuno seguiria as setas "direita, esquerda, esquerda, esquerda", o que o levaria ao 9º lugar.
2. Perdeu na ronda 2 e ganhou em todas as outras. Assim, o Nuno seguiria as setas "esquerda, direita, esquerda, esquerda", o que o levaria ao 5º lugar.
3. Perdeu na ronda 3 e ganhou em todas as outras. Assim, o Nuno seguiria as setas "esquerda, esquerda, direita, esquerda", o que o levaria ao 3º lugar.
4. Perdeu na ronda 4 e ganhou em todas as outras. Assim, o Nuno seguiria as setas "esquerda, esquerda, esquerda, direita", o que o levaria ao 2º lugar.

Os cenários descritos anteriormente corespondem respetivamente às hipóteses (E), (C), (B) e (A), pelo que a única hipótese que sobra é a (D), já que o Nuno nunca consegue ficar em 7º lugar (para isso teria de perder em duas rondas).

Isto é Pensamento Computacional!

O diagrama neste problema, que modela o torneio, é um tipo de estrutura chamada *árvore de decisão*. O topo da árvore (ou raíz) é onde o processo de decisão começa. A partir daí, selecionamos um ramo a seguir, dependendo da resposta a uma pergunta de decisão. Neste problema, a pergunta de decisão é "ganhei ou perdi durante a ronda?". A resposta "ganhei"significa que seguimos o ramo esquerdo e a resposta "perdi"significa que seguimos o ramo direito. Quando se esgotam os ramos, dizemos que chegámos às folhas da árvore de decisão. As folhas representam os resultados finais, que neste problema são classificações.

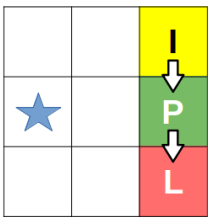
Se alguma vez tentou identificar o número secreto de alguém dando um palpite e fazendo com que a pessoa respondesse com "maior"ou "menor", também utilizou uma árvore de decisão. As árvores de decisão são utilizadas em Ciência de Computadores na Inteligência Artificial, e mais especificamente na Aprendizagem Automática. Por exemplo, quando um programa está a ser concebido para distinguir entre várias imagens, como "cão", "peixe"ou "semáforo", são utilizadas várias características como "forma retangular", "dois olhos"e "barbatanas"como questões de decisão. Com treino repetido, o programa desenvolve características distintivas suficientes para classificar corretamente uma imagem.

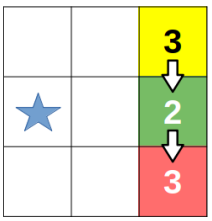
11. Mais Perto ou Mais Longe

O Daniel está a jogar um jogo para descobrir onde está enterrado o tesouro numa grelha de quadrados.

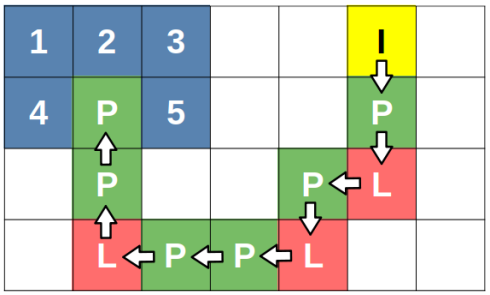
O Daniel começa num quadrado **I**nicial (**I**) e pode mover-se um passo de cada vez apenas no sentido horizontal ou vertical para os quadrados vizinhos. Após cada passo, o Daniel recebe um sinal que indica se está mais **P**erto (**P**) ou mais **L**onge (**L**) do tesouro, sendo que a distância ao tesouro é o número mínimo de passos para lá chegar.

Por exemplo, na grelha 3×3 mostrada abaixo, o tesouro está enterrado debaixo do quadrado marcado com ★. O Daniel dá dois passos em frente, seguindo as setas. As distâncias entre os dois quadrados e o tesouro são mostradas abaixo, à direita. O Daniel recebe os sinais **P** e **L**, respetivamente, após cada passo.





Agora, é dada ao Daniel outra grelha 4×7, onde o seu caminho segue as setas e os sinais obtidos também são comunicados. De seguida, o Daniel recebe uma pista que indica que o tesouro está enterrado num dos cinco quadrados numerados.



Pergunta

Em qual quadrado numerado foi enterrado o tesouro?

Respostas possíveis

(A) 1 **(B)** 2 **(C)** 3 **(D)** 4 **(E)** 5

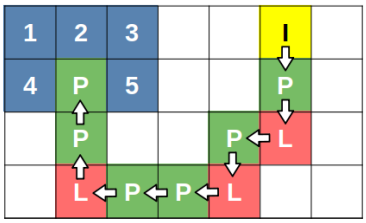
11. Mais Perto ou Mais Longe (Resolução)

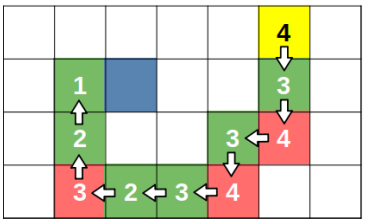
Solução

(E)

Resolução

A resposta correcta é o quadrado 5. Verificamos a exatidão das marcações "P"e "L"no tabuleiro de jogo, escrevendo em cada quadrado a distância ao quadrado 5. Podemos ver que, se a distância ao quadrado 5 aumentou quando nos movemos na direção das setas, a letra "L"é anotada, e se a sua distância ao quadrado 5 diminuiu, então "P"é anotado no quadrado. A sequência de sinais obtida é consistente com o resultado relatado. Assim, sabemos que o tesouro foi colocado no quadrado 5.

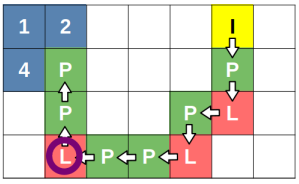




Para as outras possíveis posições do tesouro, encontramos sempre pelo menos uma letra incorrecta.

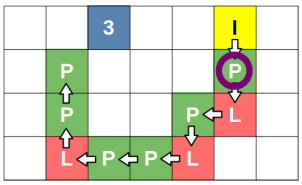
Para o tesouro nos quadrados 1, 2 ou 4:

Se o tesouro estava enterrado num dos quadrados 1, 2 ou 4, então a letra "L"assinalada teria de ser um "P", porque esse quadrado está mais próximo do tesouro (em todas estas três localizações) do que o quadrado de onde veio.



Para o tesouro no quadrado 3:

Se o tesouro estivesse enterrado no quadrado 3, então o quadrado assinalado "L"está incorreto porque esse quadrado está mais longe do tesouro do que o quadrado em que o jogador começou.



Isto é Pensamento Computacional!

A *aprendizagem por reforço* é uma técnica de *aprendizagem automática* que se preocupa com a forma como os agentes inteligentes devem realizar acções num ambiente para maximizar a recompensa cumulativa. Os componentes fundamentais de um sistema de aprendizagem por reforço incluem um agente ou aprendente, o *ambiente* com o qual interage, a política que segue para tomar decisões e o sinal de *recompensa* que recebe depois de realizar as acções. Para avaliar o sinal de recompensa, a função de valor mede a "qualidade"de um determinado estado.

Neste problema específico, a nova localização no tabuleiro de jogo após uma jogada representa o *ambiente*, ao passo que o sinal obtido a partir da distância detectada serve como *sinal de recompensa* (com "P"a indicar uma recompensa positiva e "L"a indicar uma recompensa negativa). O Daniel, que recolhe os sinais de recompensa tomando decisões óptimas para a etapa seguinte, serve de agente. Ele deve considerar a possibilidade de um quadrado conter o tesouro, o que é semelhante à função de valor.

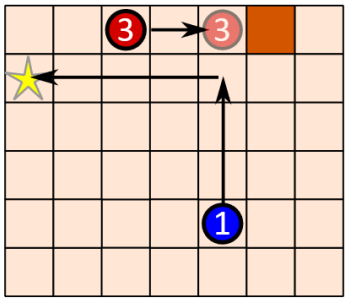
É importante notar que a distância entre os quadrados neste cenário é medida pela *distância de Manhattan* (também designada por métrica do táxi), sendo que o jogador apenas se pode deslocar na horizontal ou na vertical.

12. Robôs de Choque

O objetivo do jogo Robôs de Choque é dar instruções a um robô para que ele se mova da sua posição atual para a estrela. Um obstáculo é definido como um objeto que o robô não consegue atravessar. O jogo tem as seguintes regras:

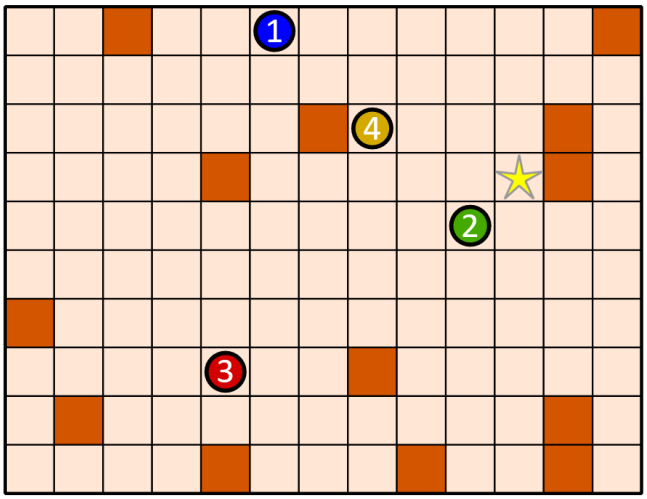
1. Quadrados castanhos no tabuleiro são obstáculos.
2. Outros robôs também são obstáculos.
3. O robô pode ser programado para mover em 4 direções apenas. As direções são cima ( ), baixo ( ), direita ( ) e esquerda ( ).
4. Quando o robô começa a mover-se numa direção, ele não para até atingir um obstáculo ou a borda do tabuleiro.
5. Quando um robô inicia o seu movimento, os restantes robôs esperam até que ele pare.

No exemplo a seguir, o jogador pode mover o robô 1 até à estrela combinando 3 movimentos. O primeiro move o robô 3 para a direita. Isto fará com que ele se mova e pare junto ao obstáculo indicado. De seguida, desloca o robô 1 para cima e aproveita o robô 3 parado. Por fim, desloca o robô 1 para a esquerda.



Pergunta

Dada a situação no tabuleiro representado a seguir, qual é o número mínimo de movimentos necessários para o robô 1 parar na estrela ?



Respostas possíveis

(A) 4 **(B)** 6 **(C)** 7 **(D)** 9

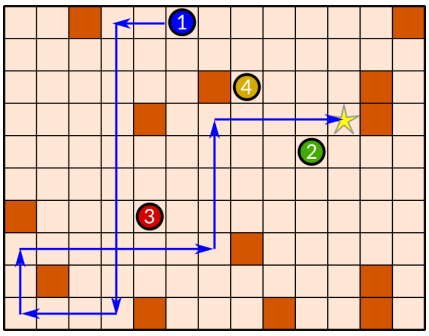
12. Robôs de Choque (Resolução)

Solução

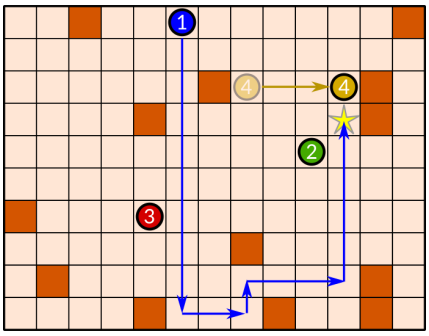
(A)

Resolução

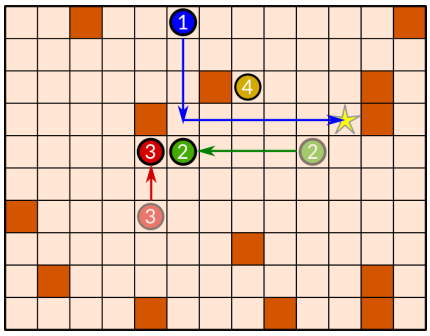
A resposta correta é a (A) com 4 movimentos. Nas imagens explicativas que se seguem, cada movimento dos robôs está assinalado com uma seta. A colaboração dos robôs é importante para resolver este problema e mostramos como podemos melhorar a solução ao colocarmos os robôs a colaborar.



Utilizando apenas o robô 1, a deslocação para a estrela a partir do ponto de partida leva a 7 movimentos no mínimo.



O robô 4 pode ser um obstáculo para o robô 1 após o seu movimento e esta tática conduz a apenas 6 movimentos.

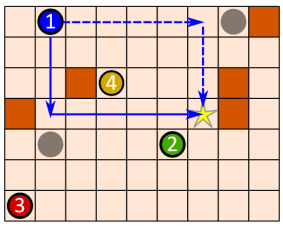


Ao deslocar primeiro o robô 3, este pode ser um obstáculo para o robô 2, que por sua vez pode ficar a ser um obstáculo para o robô 1. Depois, o robô 1 só precisa de fazer mais 2 movimentos.

Como seria possível obter a solução correta?

Se começarmos a pensar nos movimentos possíveis do robô 1, vemos que há demasiadas combinações. Por isso, podemos começar pelo fim, para explorar o último movimento do robô 1. Uma solução possível é que ele venha da esquerda para a estrela (como poderia vir de uma das outras três direções). Para vir da esquerda, é necessário um obstáculo por baixo. Esse quadrado pode potencialmente servir como local adequado para os outros robots pararem. Este processo repete-se nas etapas seguintes até chegarmos à solução.

Porque é que 3 movimentos não são suficientes?



Não há maneira de mover o robô 1 para o quadrado alvo com 1 movimento.

Há duas maneiras a partir de 2 movimentos (ver figura ao lado). Para realizar esse movimento, precisamos de colocar um robô num dos quadrados marcados a cinzento primeiro, para que o robô 1 possa parar o seu movimento, chocando contra ele. Mas não podemos fazer com que nenhum dos outros robôs se mova para um destes dois quadrados cinzentos num só movimento. Portanto, 4 movimentos é o limite inferior da solução.

Isto é Pensamento Computacional!

A colaboração é um aspeto essencial em Ciência de Computadores. Sem uma colaboração adequada, os robôs podem correr o risco de se danificarem a si próprios ou de causarem danos às pessoas. Ao desenvolver sistemas de informação que são utilizados por várias pessoas, os programadores devem ter em conta os vários processos que precisam de funcionar em harmonia. Estabelecer a ordem correta, definir funções e implementar uma estrutura hierárquica para utilizadores e componentes do sistema são factores críticos para promover uma colaboração eficiente dentro do sistema.

No entanto, a colaboração de vários robôs pode ser uma tarefa de otimização complexa, sendo que existem várias estratégias possíveis para encontrar soluções de alta qualidade.