Bebras PT

Edição 2023

Categoria
Seniores (11º e 12º 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

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. 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"*

1. 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!

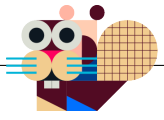
Este exercício envolve um exemplo de um modelo artificial para a modelação da linguagem e a produção de texto. Trata-se de um m*odelo probabilístico*, o que significa que cada saída do modelo se baseia em probabilidades. Há muitos tipos de modelos. Neste caso, o modelo está a produzir uma palavra de 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.

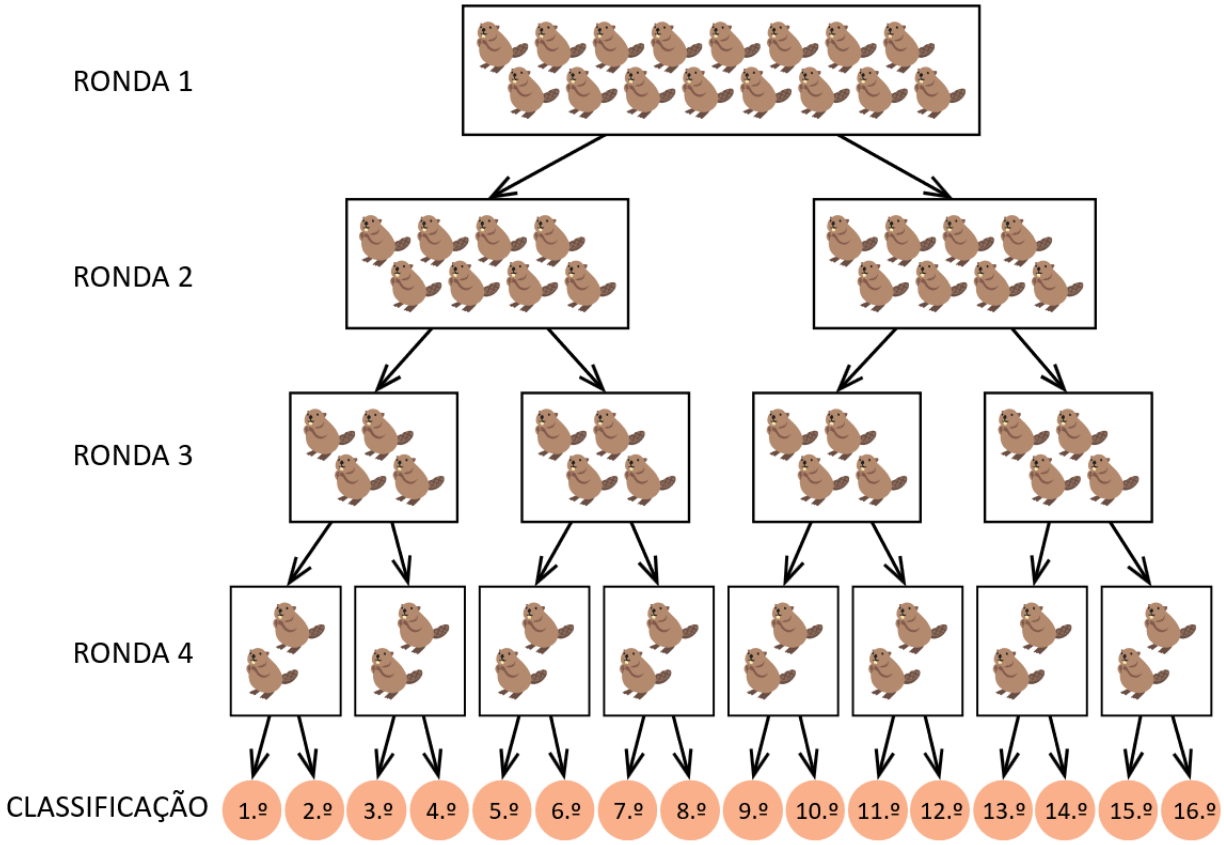
O algoritmo de inferência é o que utilizamos após o treino para produzir o resultado. Por exemplo, quando escrevemos a pergunta *"Olá, ChatGPT. O que é a aprendizagem automática?"*, o modelo pega nessa sequência de palavras e utiliza algo equivalente a uma tabela de probabilidades muito grande para produzir 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).



2. 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

2. 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.

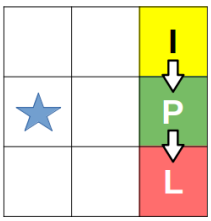
Se alguma vez tentou identificar o número secreto de alguém dando um palpite e fazendo com que a pessoa respondesse com "maior"ou "menor", também utilizou uma árvore de decisão. As árvores de decisão são utilizadas em Ciência de Computadores na Inteligência Artificial, e mais especificamente na Aprendizagem 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.

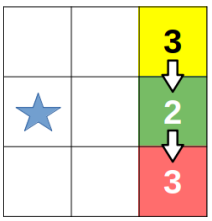
3. Mais Perto ou Mais Longe

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

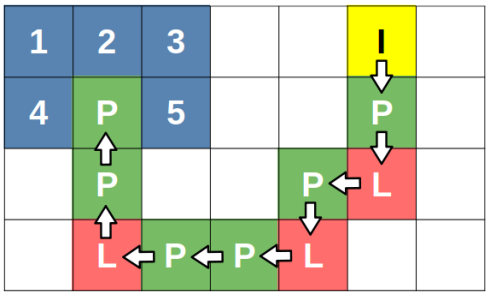
O Daniel começa num quadrado **I**nicial (**I**) e pode mover-se um passo de cada vez apenas no sentido horizontal ou vertical para os quadrados vizinhos. Após cada passo, o Daniel recebe um sinal que indica se está mais **P**erto (**P**) ou mais **L**onge (**L**) do tesouro, sendo que a distância ao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

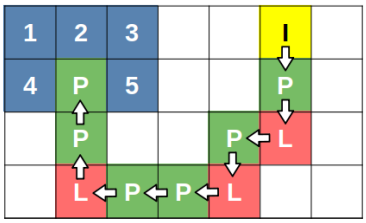
3. Mais Perto ou Mais Longe (Resolução)

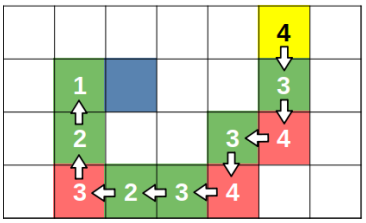
Solução

(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.

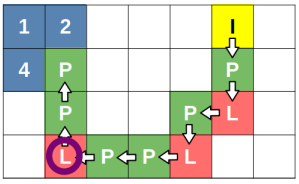




Para as outras possíveis posições do tesouro, encontramos sempre pelo menos uma letra incorrecta.

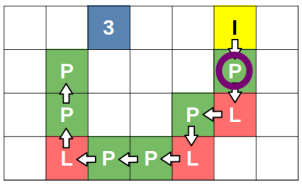
Para o tesouro nos quadrados 1, 2 ou 4:

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.

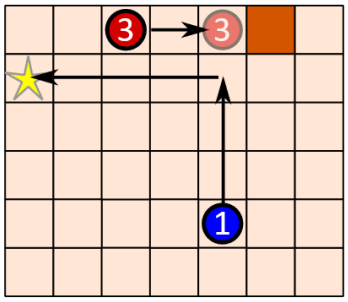
É importante notar que a distância entre os quadrados neste cenário é medida pela *distância de Manhattan* (também designada por métrica do táxi), sendo que o jogador apenas se pode deslocar na horizontal ou na vertical.

4. 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:

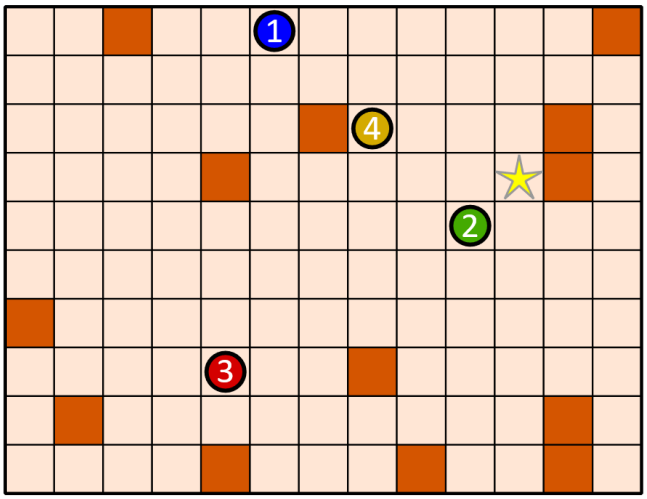
1. Quadrados castanhos no tabuleiro são obstáculos.
2. Outros robôs também são obstáculos.
3. O robô pode ser programado para mover em 4 direções apenas. As direções são cima (), baixo (), direita () e esquerda ().
4. Quando o robô começa a mover-se numa direção, ele não para até atingir um obstáculo 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

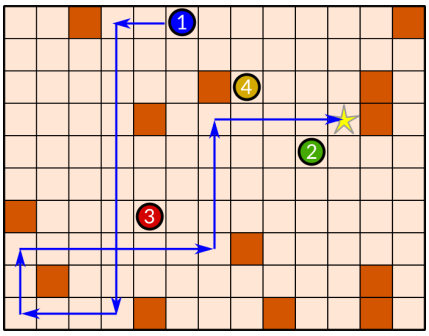
4. Robôs de Choque (Resolução)

Solução

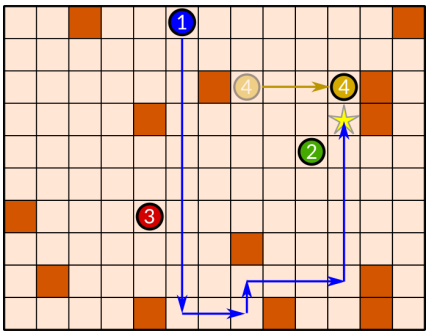
(A)

Resolução

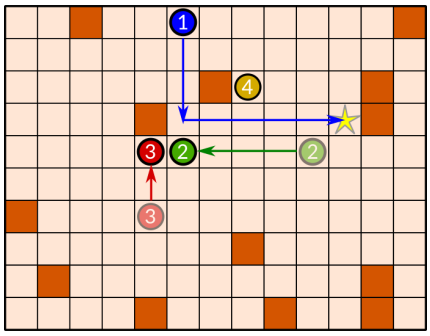
A resposta correta é a (A) com 4 movimentos. Nas imagens explicativas que se seguem, cada movimento dos robôs está assinalado com uma seta. A colaboração dos robôs é importante para resolver este problema e mostramos como podemos melhorar a solução ao colocarmos os robôs a colaborar.



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.

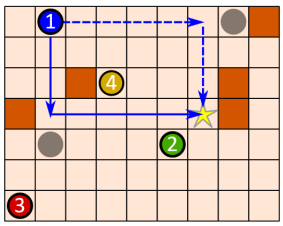


Ao deslocar primeiro o robô 3, este pode ser um obstáculo para o robô 2, que por sua vez pode ficar a ser um obstáculo para o robô 1. Depois, o robô 1 só precisa de fazer mais 2 movimentos.

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!

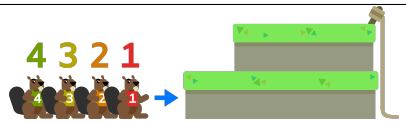
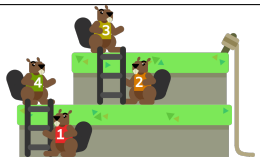
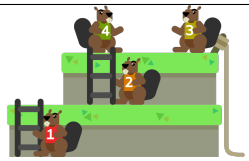
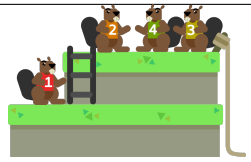
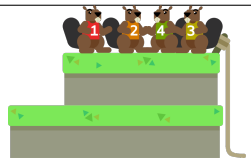
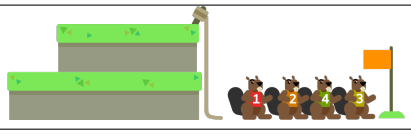
A colaboração é um aspeto essencial em Ciência de Computadores. Sem uma colaboração adequada, os robôs podem correr o risco de se danificarem a si próprios ou de causarem danos às pessoas. Ao desenvolver sistemas de informação que são utilizados por várias pessoas, os programadores devem ter em conta 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.

5. Subindo as Colinas

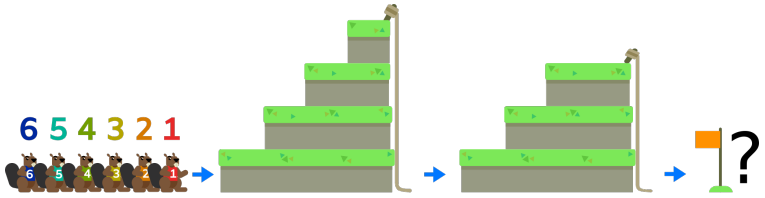
Os castores fazem sempre as suas caminhadas em fila indiana e todos transportam uma escada. Quando chegam a uma degrau da colina, o castor que está à frente segura a sua escada para os outros subirem. Todos os castores que estão a segurar a escada permanecem no seu lugar até que todos os os que não estão a segurar a escada tenham subido até ao topo da colina. Depois, cada castor que estava a segurar uma escada sobe para o nível seguinte usando a sua própria escada. Continuam a subir desta forma até que todos cheguem ao topo. Finalmente, descem com uma corda mantendo a nova ordem.

Por exemplo, se 4 castores chegarem a uma colina com dois degraus na ordem inicial **1234**:

	4 castores chegam à colina na ordem **1234**
	O castor 1 segura a escada para os outros castores
	O castor 2 segura a escada para os outros castores
	Os castores 3 e 4 chegam ao topo da colina
	Os castores 2 e 1 usam as suas escadas para subir 1 nível
	O castor 1 usa a sua escada e chega ao topo da colina
	Os castores descem pela corda mantendo a sua nova ordem **3421**

Pergunta

Seis castores fazem uma caminhada no terreno representado na figura seguinte na ordem inicial **123456**. *Por que ordem é que os castores chegam à bandeira final?*



Resposta

(escreve uma ordem na forma de um número de 6 dígitos na folha de respostas)

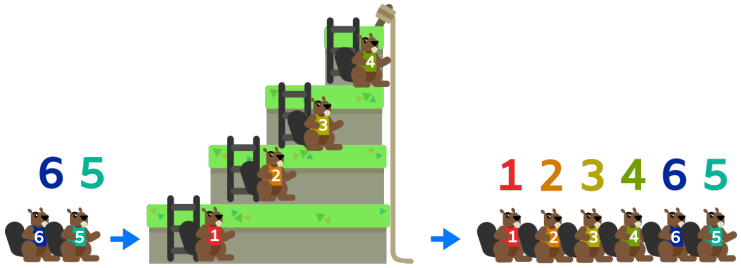
5. Subindo as Colinas (Resolução)

Solução

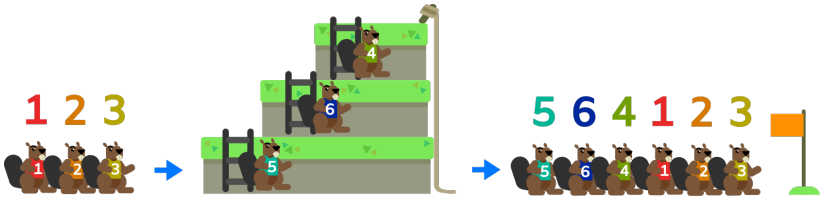
321465

Resolução

Os castores começaram na ordem **123456** e chegam à primeira colina. Os castores 1, 2, 3 e 4 seguram as escadas nos respetivos níveis enquanto os castores 5 e 6 sobem até ao topo por essa ordem. Depois os castores 4, 3, 2 e 1 sobem por esta ordem até estarem todos no topo. A ordem após esta subida é **564321**.



Os castores chegam à segunda colina por esta ordem. Como primeiro castor da fila, o castor 5 sobe a escada no nível 1, o castor 6 no nível 2 e o castor 4 no nível 3, permitindo que os castores 3, 2 e 1 subam ao topo da segunda colina por esta ordem. Depois, 3, 2 e 1 sobem ao topo e a ordem fica **321465**, que é a ordem com que chegam ao final.



Isto é Pensamento Computacional!

A forma como os castores sobem as colinas neste problema reflecte a estrutura de uma *pilha* e de uma *fila* em Ciência de Computadores. Os castores criam escadas empilhando e sobem as colinas numa fila.

A pilha e a fila são dois tipos de estruturas de dados. Armazenar dados numa pilha é exatamente como empilhar objectos na vida real: adicionamos um objeto à pilha colocando-o no topo da pilha, e o único objeto acessível é o que está no topo da pilha. Assim, se retirarmos dados/objectos de uma pilha um a um, o último dado ou objeto colocado na pilha será o primeiro a ser removido. Descrevemos a ordem como último a entrar, primeiro a sair (*LIFO - Last In, First Out*).

Por outro lado, armazenar dados numa fila é exatamente como as pessoas fazem fila para comprar bilhetes: a primeira pessoa na fila será a primeira a obter um bilhete. Por outras palavras, se retirarmos os dados/objectos de uma fila um a um, o primeiro dado ou objeto colocado na fila será o primeiro a sair. Descrevemos a ordem como primeiro a entrar, primeiro a sair (*FIFO - First In, First Out*).

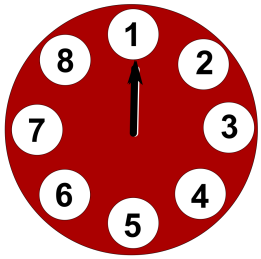
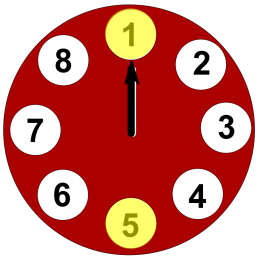
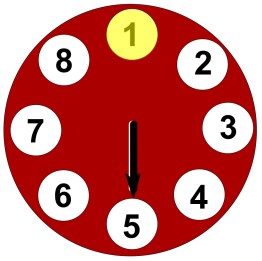
6. Abrir o Cofre

Um castor tem de abrir um cofre acendendo a combinação de números correcta.

Em cada movimento, o castor pode rodar a seta no sentido dos ponteiros do relógio exatamente por 3 ou 4 passos (cada passo avança um número).

A seta modifica a luz do número em que pousa: se a luz do número estava apagada, a luz acende-se; se a luz do número estava acesa, a luz apaga-se.

Por exemplo, isto é o que acontece se o castor fizer 3 movimentos, cada um a rodar 4 passos:

Posição Inicial	Depois do 1º movimento (4 passos)	Depois do 2º movimento (4 passos)	Depois do 3º movimento (4 passos)
			

Pergunta

Para abrir o cofre, o castor precisa de acender **apenas o 7 e o 8** *(nenhum outro número deve ficar aceso).*

Qual é o **número mínimo de movimentos** *que o castor precisa de fazer para acender apenas o 7 e o 8 a partir da posição inicial mostrada acima?*

Respostas possíveis

(A) 3 **(B)** 4 **(C)** 5 **(D)** 6 **(E)** 7

6. Abrir o Cofre (Resolução)

Solução

(B)

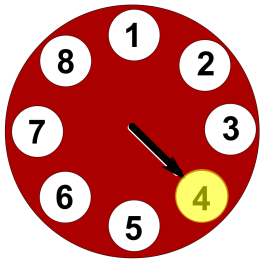
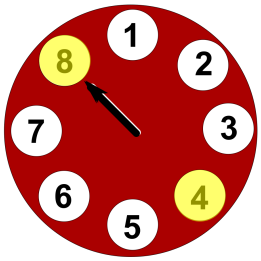
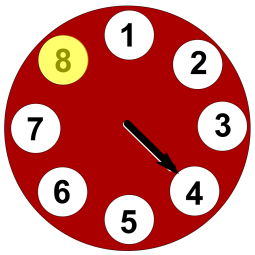
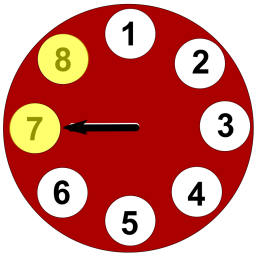
Resolução

A resposta correta é a (B) 4 movimentos.

Uma forma de começar a resolver este problema é perceber que os números 8 e 7 podem ser alcançados fazendo duas jogadas a partir da posição inicial. Para chegar ao 7, deslocamo-lo duas vezes em 3 passos. Para chegar ao 8, movemo-lo uma vez por 4 passos e depois por 3.

Além disso, se estivermos no 8, podemos chegar ao 7 em dois passos, mas não o contrário: chegar ao 8 a partir do 7 exigiria mais passos e complicaria a solução. Isto é uma indicação de que tentar chegar primeiro ao 8 e depois ao 7 parece uma opção promissora, que poderia talvez ser completada em apenas 4 passos. Mas é claro que, com 4 passos, também modificamos o estado dos números intermédios em que aterramos. Assim, para mantermos apenas o 7 e o 8 acesos e nenhum outro número, temos de garantir que aterramos no mesmo número intermediário tanto quando vamos para o 8, como quando depois chegamos ao 7.

Isto funciona se nos movermos da seguinte forma: primeiro, 3 passos (acendemos o 4), depois 4 passos (acendemos o 8), depois 4 passos novamente (apagamos o 4) e, finalmente, mais 3 passos para acender o 7.

Depois do 1º movimento (3 passos)	Depois do 2º movimento (4 passos)	Depois do 3º movimento (4 passos)	Depois do 4º movimento (3 passos)
			

Seria possível fazer uma lista sistemática de todos os possíveis conjuntos de movimentos para perceber que é impossível fazer menos do que 4 movimentos, mas como vimos atrás, precisamos de pelo menos 2 movimentos para chegar ao 7 ou ao 8 e em nenhum deles com apenas mais um movimento conseguimos chegar ao outro número, pelo que podemos deduzir que 4 movimentos é o mínimo possível.

Isto é Pensamento Computacional!

Neste problema, são-nos dadas regras (instruções) que têm de ser repetidas várias vezes. No entanto, não sabemos quantas vezes cada regra instrução tem de ser usada para atingir o objetivo. Uma abordagem é escrever um programa que gera todas as sequências possíveis de movimentos (um total de 4 neste problema) e verificar o resultado para cada caso. Os cientistas de computadores referem-se a isto como uma *pesquisa de força bruta*. No entanto, este método fica impraticável para casos maiore, pois o número de possíveis combinações de movimentos cresce exponencialmente.

Neste caso particular, a abordagem usada foi semelhante a *"tentativa e erro"*, em que tentamos encontrar a solução seguindo iterativamente as regras. É importante notar que num caso geral a primeira solução encontrada por tentativa e erro nem sempre é a solução óptima, mas neste caso conseguimos também mostrar que era impossível fazer melhor.

7. Matrículas de Carros

Em muitos países, os carros têm uma variedade de desenhos e formatos de série para as matrículas. Geralmente, as matrículas utilizam letras do alfabeto inglês (composto por 26 letras) e algarismos.

A castora Isabel está interessado em saber qual dos países fornecidos pode registar o maior número de carros, assumindo que as letras (A-Z) e os algarismos (0-9) estão sempre dispostos da mesma forma que no modelo fornecido.

São permitidas todas as combinações possíveis de letras e dígitos. Especificamente, se um carácter no exemplo for uma letra, pode ser qualquer letra nesse local, e a mesma regra aplica-se se o carácter na matrícula de exemplo for um dígito.

Os códigos de país na parte azul da matrícula não mudam e não são contabilizados.

Pergunta

Em qual dos seguintes países existem mais possíveis matrículas diferentes?

Respostas possíveis

(A) FIN BZM-184 Finlândia

(B) SK BL 976AA Eslováquia

(C) RO BN 08 CTL Roménia

(D) SG 197052 Suíça

(E) UA AK 9265 AK Ucrânia

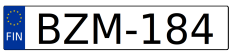
7. Matrículas de Carros (Resolução)

Solução

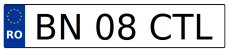
(E)

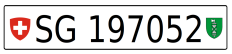
Resolução

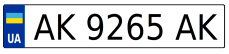
A resposta correta é a (E) Ucrânia
Para encontrar a solução, é necessário contar o número de letras e algarismos, independentemente da sua ordem.
Imaginemos que o alfabeto tinha apenas 3 letras: {A,B,C}. Neste caso uma sequência de duas letras deste alfabeto tinha $3^2 = 9$ possibilidades: AA, AB, AC, BA, BB, BC, CA, CB, CC. De uma forma mais geral, uma sequência de k elementos deste alfabeto teria 3^k possibilidades.
Podemos aplicar a mesma lógica para as matrículas, sabendo que na realidade existe 26 possibilidades para uma letra (de A a Z) e 10 possíveis algarismos (de 0 a 9). Assim sendo, uma matrícula com k letras e n algarismos terá $26^k \times 10^n$ possibilidades, independentemente da ordem em que as letras e algarismos aparecem.

- (A) 3 letras e 3 algarismos → $26^3 \times 10^3$; 

- (B) 4 letras e 3 algarismos → $26^4 \times 10^3$; 

- (C) 5 letras e 2 algarismos → $26^5 \times 10^2$; 

- (D) 2 letras e 6 algarismos → $26^2 \times 10^6$; 

- (E) 4 letras e 4 algarismos → $26^4 \times 10^4$. 

Usando uma calculadora seria possível calcular os valores exatos de cada uma expressões anteriores. É no entanto possível fazer o problema sem calculadora, pois podemos, por exemplo, dividir números e tomar decisões comparando o resultado com 1. Podemos selecionar dois países com as potências mais semelhantes para comparar.

- (A) vs (B) → (B) é maior porque (B)/(A) = 26 > 1 (a opção (A) é eliminada)
- (B) vs (C) → (C) é maior porque (C)/(B) = 26/10 > 1 (a opção (B) é eliminada)
- (C) vs (E) → (E) é maior porque (E)/(C) = 100/26 > 1 (opção (C) é removida)
- (D) vs (E) → (E) é maior porque E/D = 26^2/100 > 1 (opção (D) é removida)

Isto é Pensamento Computacional!

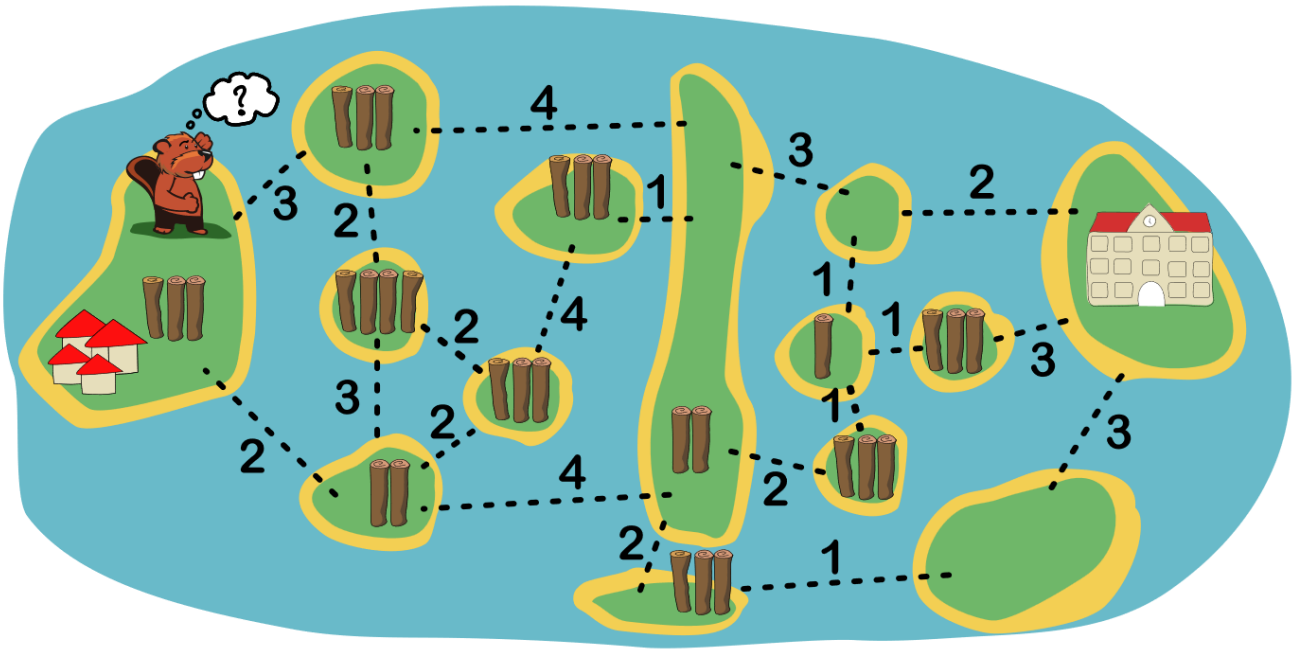
Este problema requer alguns princípios básicos de contagem, que encaixam num ramo da Matemática conhecido como *Análise Combinatória*. Neste caso, contar quantas combinações de matrículas existem.

A mMtemática surge muitas vezes "de mão dada"com a Ciência de Computadores e é frequente as contagens serem importantes em algoritmos e programas. Um exemplo disso é saber até que ponto uma palavra-passe é forte, analisando quantas possíveis combinações de caracteres válidos existem. Se pensarmos que k letras de um alfabeto com n possíveis caracteres induz n^k possibilidades, é por isso que muitas das vezes recebes pedidos como a palavra passe ter pelo menos k caracteres (o que aumenta o expoente) e coisas como ter pelo menos uma letra, um dígito e um caracter especial (o que aumenta n, o tamanho do "alfabeto"possível, e consequentemente a base do expoente).

8. Construção de Pontes

O castor Paulo é um construtor de pontes. A sua próxima tarefa é construir pontes para que os alunos da sua aldeia possam chegar à nova escola.

- Infelizmente, o Paulo não sabe nadar, por isso, primeiro tem de construir uma ponte que possa ser usada para atravessar a água tantas vezes quantas as necessárias;
- Para construir uma ponte, o Paulo precisa de troncos suficientes. Na sua aldeia, ele só pode apanhar 3 troncos para começar. Ele também pode viajar entre quaisquer ilhas ligadas por pontes para apanhar mais troncos. O mapa abaixo mostra quantos troncos são necessários para construir uma ponte em cada possível travessia de água e quantos troncos estão disponíveis em cada ilha;
- Cada tronco só pode ser usado uma vez, mas nem todos os troncos disponíveis têm de ser usados.



Pergunta

Qual é o menor número de troncos que o Paulo precisa de usar em pontes para criar um caminho que os alunos possam usar para ir da aldeia para a escola por terra e por pontes?

Respostas possíveis

(A) 11 **(B)** 12 **(C)** 13 **(D)** 14 **(E)** 17

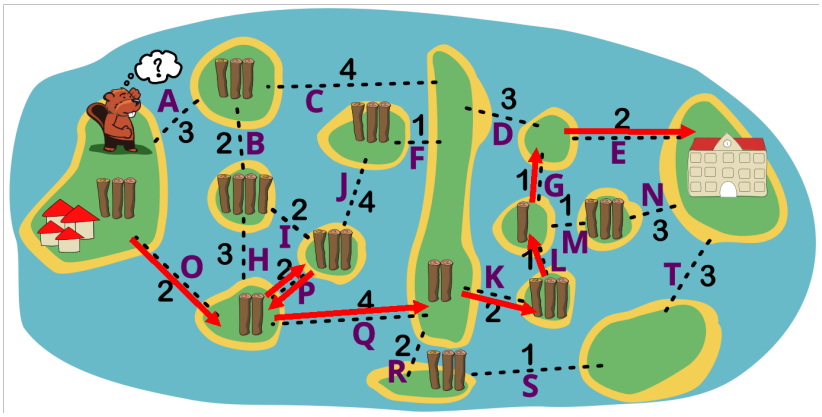
8. Construção de Pontes (Resolução)

Solução

(D)

Resolução

A resposta correta é a (D) com 14 troncos, usando a solução ilustrada na seguinte figura:



Podemos mostrar que o caminho correto é viável (cada etapa tem troncos recolhidos >= troncos usados) registando novamente o número acumulado de troncos (recolhidos, usados) em cada etapa:

$$(3,2) \rightarrow (5,4) \rightarrow (8,4) \rightarrow (8,8) \rightarrow (10,10) \rightarrow (13,11) \rightarrow (14,12) \rightarrow (14,14)$$

Podemos provar que isto é mínimo encontrando todos os caminhos que ligam a aldeia à escola que têm ≤ 14 troncos usados. Note-se que todos os caminhos têm de passar pela grande ilha no meio, pelo que podemos dividir o problema em duas partes. Em segundo lugar, o caminho da aldeia para o centro precisa de pelo menos 6 troncos e o caminho do centro para a escola precisa de pelo menos 5.

Primeiro resolvemos a parte esquerda: procuramos todos os caminhos viáveis que utilizem ≤ 9 troncos (porque qualquer solução para a parte esquerda ≥ 10 utilizaria pelo menos 15 troncos no total). Temos os seguintes caminhos viáveis que se ligam ao centro e usam ≤ 9 troncos, mostrados com troncos (recolhidos,usados):

$$\{A,B,C\} = (10,9), \{O,H,Q\} = (9,9), \{O,P,Q\} = (8,8), \{O,P,J,F\} = (11,9)$$

Resolvemos então o lado direito utilizando um limite inferior de 8 troncos utilizados para o lado esquerdo, pelo que procuramos todos os caminhos que utilizam ≤ 6 troncos:

$$\{D,E\} = (2,5), \{D,G,E\} = (3,6), \{K,L,G,E\} = (6,6), \{R,S,T\} = (5,6)$$

Combinamos agora as soluções para as partes esquerda e direita para encontrar quaisquer pares viáveis que tenham ≤ 14 troncos usados no total e encontramos apenas $\{O,P,Q\} = (8,8) + \{K,L,G,E\} = (6,6) \rightarrow (14,14)$, uma vez que todos os outros pares têm troncos recolhidos $<$ troncos usados.

Isto é Pensamento Computacional!

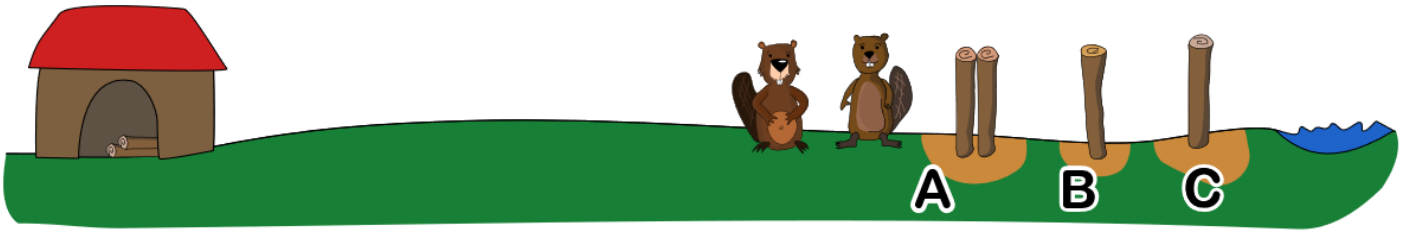
As ilhas e as pontes potenciais podem ser representadas como nós e arestas de um gráfo. O número de troncos recolhidos em cada ilha e utilizados para construir cada ponte pode ser atribuído aos respetivos nós e arestas. O caminho pode ser representado como uma lista de arestas na ordem em que são construídas. A resolução deste problema em geral pode ser feita utilizando algoritmos de descoberta de caminhos. Uma pesquisa exaustiva de todos os caminhos válidos funcionaria, mas levaria um tempo exponencial em relação ao tamanho do grafo.

Em alternativa, o problema pode ser enquadrado como uma pesquisa guiada de dividir e conquistar, como acontece com a solução de exemplo. Esta abordagem utiliza técnicas de otimização de restrições, como a *consistência de caminhos* e *ramificar e limitar*. Estas técnicas constituem a base de possíveis soluções para alguns dos problemas de alocação de recursos mais difíceis do mundo real, puzzles (como o Sudoku!) e inteligências artificiais que jogam.

9. Troncos para o Armazém

Os castores Tiago e Vasco têm de transportar troncos - um de cada vez! - da margem do rio para o armazém à esquerda. Para tornar o seu trabalho mais agradável, decidem fazer turnos no trabalho. Quem está de serviço seleciona um tronco e carrega-o para a esquerda, até encontrar outro tronco ou chegar ao armazém. Quem for obrigado a carregar o último tronco até ao armazém perde o jogo.

Dizemos que um castor tem uma estratégia vencedora se houver uma forma de ele jogar e ganhar sempre, independentemente da forma como o adversário joga.



Hoje, têm de transportar quatro troncos, dispostos da forma indicada, e o Tiago é o primeiro a chegar.

Pergunta

Apenas uma das seguintes afirmações é verdadeira: qual é?

Respostas possíveis

(A) O Tiago tem uma estratégia vencedora e deve começar por transportar um dos dois troncos de A para o armazém.

(B) O Tiago tem uma estratégia vencedora e deve começar por transportar o tronco em B para o lado dos dois troncos em A.

(C) O Tiago tem uma estratégia vencedora e deve começar por transportar o tronco em C para o lado do tronco em B.

(D) Nenhuma das estratégias do Tiago das outras opções é vencedora.

9. Troncos para o Armazém (Resolução)

Solução

(B)

Resolução

Chamemos de *posição* à sequência de números que representam, da esquerda para a direita, quantos troncos estão em cada grupo fora do armazém, do mais próximo para o mais afastado do armazém. O jogo passa de uma posição para outra, não sendo possível voltar a uma posição anterior. No nosso caso, a posição inicial pode representada pela sequência **211** (omitimos um separador para simplificar).

Se o Tiago levar o tronco de B para o lado dos dois troncos de A, então a nova posição é **31** e o jogo pode continuar de duas formas:

- O Vasco transporta um tronco de A para o armazém (chega à posição **21**); o Tiago transporta o tronco de C para A (chega à posição **3**); o Vasco não tem escolha a não ser transportar um dos três troncos para o armazém, o Tiago transporta um dos dois restantes e o último tronco depende do Vasco!
- O Vasco transporta o tronco em C para o lado dos três troncos em A (para alcançar a posição 4), o Tiago transporta um tronco para o armazém (posição **3**) e a continuação é como acima.

A correção desta resposta (2) implica que a última resposta (4) está errada.

Se o Tiago começar por transportar um dos dois troncos de A para o armazém, resposta (A), então a nova posição é **111**, a partir da qual o Vasco transporta o tronco em C para ao lado do tronco em B atingindo a posição **12** agora o jogo pode continuar de duas maneiras:

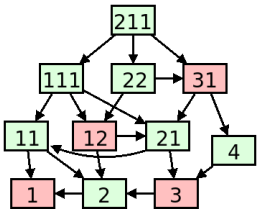
- O Tiago transporta um tronco de B para A (chega à posição **21**) e o jogo continua como atrás descrito, mas com os papéis dos dois jogadores invertidos, pelo que o Tiago perde.
- O Tiago transporta um tronco de A para o armazém (chega à posição **2**) e o Vasco ganha ao transporte um desses restantes, deixando o último para o Tiago.

Se o Tiago começar por transportar o tronco de C para B, resposta (C), então a nova posição é **22**, a partir da qual o Vasco transporta um tronco de A para o armazém atingindo a posição **12**, como acima.

Isto é Pensamento Computacional!

Este problema é uma instância de um jogo combinatório imparcial (uma vez que os dois jogadores têm o mesmo conjunto de jogadas legais a partir de qualquer posição); além disso, haverá sempre um vencedor, uma vez que o empate final é impossível neste caso.

Para representar todas as possíveis jogadas, podemos construir um grafo como o da figura à direita, onde um nó é uma posição e um arco representa um jogada.



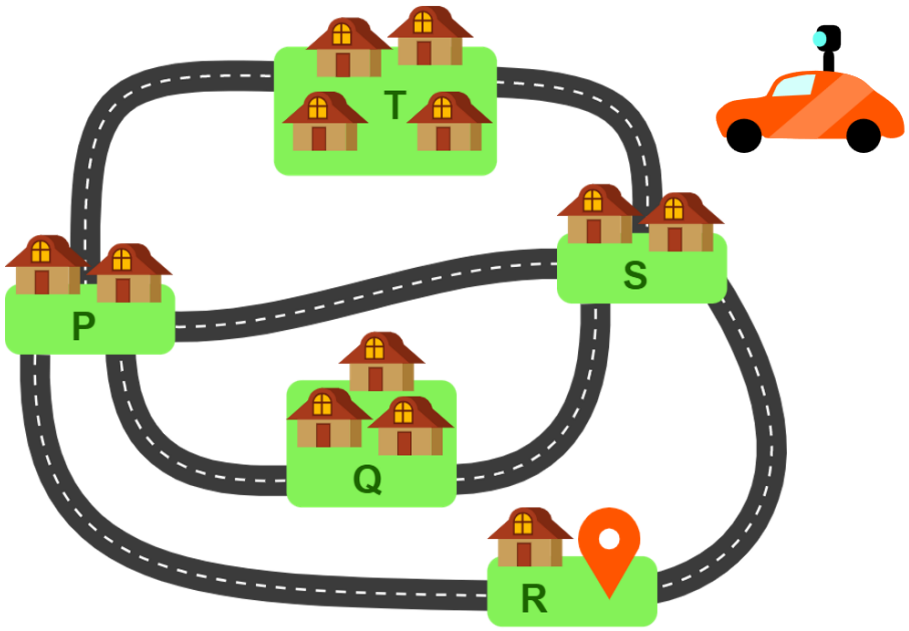
Podemos pintar de vermelho os nós que garantidamente sáo "perdedores"(quem jogar aí perde) e de verde os que são "vencedores"(quem jogar aí ganha). Para colorir o grafo podemos começar por colocar a vermelho a posição **1** (último tronco) e depois a partir daí pintar de verde todas as posições que têm pelo menos uma possibilidade de levar um nó vermelho (pois significa que se jogarmos isso o próximo jogador perde.

É possível ver assim que a posição inicial **211** é vencedora (ou seja, o Tiago pode garantir que ganha) e a jogada que deve fazer deve levar à posição **31** (pois isso implica que o Vasco perde).

A *Teoria de Jogos* é um ramo da Matemática (e de Ciência de Computadores) que estuda precisamente cenários como este. Foi o matemático Charles L. Bouton que iniciou os estudos teóricos sobre esta categoria de jogos em particular, em 1901, com a análise de um jogo que se tornou famoso e paradigmático, o Nim; resultados fundamentais e mais gerais foram depois obtidos, independentemente, por Roland P. Sprague em 1936 e por Patrick M. Grundy em 1939, dando origem à função de Sprague-Grundy para classificar posições de um jogo, que contém muito mais informações do que apenas saber se a posição é vencedora ou perdedora.

10. CastorMaps

A *CastorMaps* está a recolher imagens de **todas** as estradas que ligam as aldeias apresentadas no mapa abaixo. Devido a limitações de tempo, **dispõem de apenas 7 horas** para captar estas imagens. Podem escolher qualquer rota que lhes permita realizar esta tarefa, tendo em conta que é preciso exatamente 1 hora para captar imagens de cada estrada que liga duas aldeias.



Pergunta

Partindo da aldeia R, **quantos percursos diferentes** *poderia o carro utilizar para completar as imagens do mapa em exatamente 7 horas?*

Respostas possíveis

(A) 3 **(B)** 4 **(C)** 6 **(D)** 8 **(E)** 10 **(F)** 12

10. CastorMaps (Resolução)

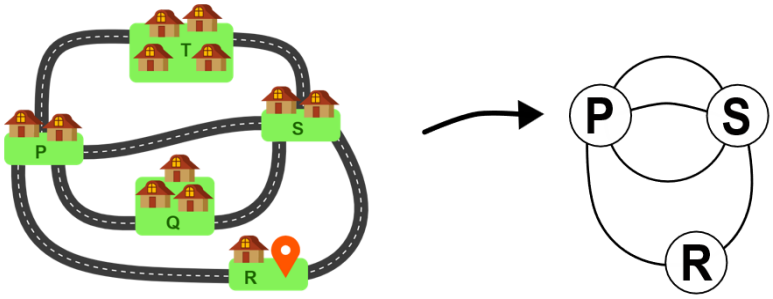
Solução

(F)

Resolução

A resposta correcta é a (F) pois existem 12 percursos diferentes.

O carro tem de atravessar cada caminho exatamente uma vez para recolher todas as fotografias em sete horas. Consequentemente, os únicos sítios onde existe a possibilidade de escolher o caminho a seguir são a aldeia inicial (R) e as aldeias (P e S) ligadas a mais do que dois caminhos. Podemos simplificar o enunciado do problema construindo um modelo abstrato que ignora as aldeias Q e T, como exemplificado na figura abaixo:



Dizem-nos que temos de começar em R e, como temos de percorrer cada caminho exatamente uma vez, isto significa que também temos de terminar em R.

Consideremos primeiro o movimento para a direita a partir de R. As únicas opções de percursos são quantas maneiras diferentes de ir de S a P que visitam cada um dos três caminhos entre S e P. A resposta é seis. Podemos começar por qualquer um dos três caminhos, mas depois de escolhermos um só podemos escolher um dos outros dois e, depois de fazermos essa escolha, não há escolha para o terceiro caminho. Assim, o número de escolhas é $3 \times 2 \times 1 = 6$, como se mostra aqui:

(cima, meio, baixo) (cima, baixo, meio) (meio, cima, baixo) (meio, baixo, cima) (baixo, cima, meio) (baixo, meio, cima)

Partindo de R e deslocando-se para a direita, há portanto seis maneiras de regressar a R percorrendo cada caminho exatamente uma vez. E se partirmos de R e nos deslocarmos para a esquerda? A resposta também é seis e, de facto, cada um dos percursos possíveis é apenas o inverso dos primeiros seis percursos. Isto dá um total de doze percursos possíveis que visitam cada uma das sete arestas exatamente uma vez:

R→S→T→P→S→Q→P→R R→P→Q→S→P→T→S→R R→S→T→P→Q→S→P→R R→P→S→Q→P→T→S→R
R→S→P→T→S→Q→P→R R→P→Q→S→T→P→S→R R→S→P→Q→S→T→P→R R→P→T→S→Q→P→S→R
R→S→Q→P→S→T→P→R R→P→T→S→P→Q→S→R R→S→Q→P→T→S→P→R R→P→S→T→P→Q→S→E

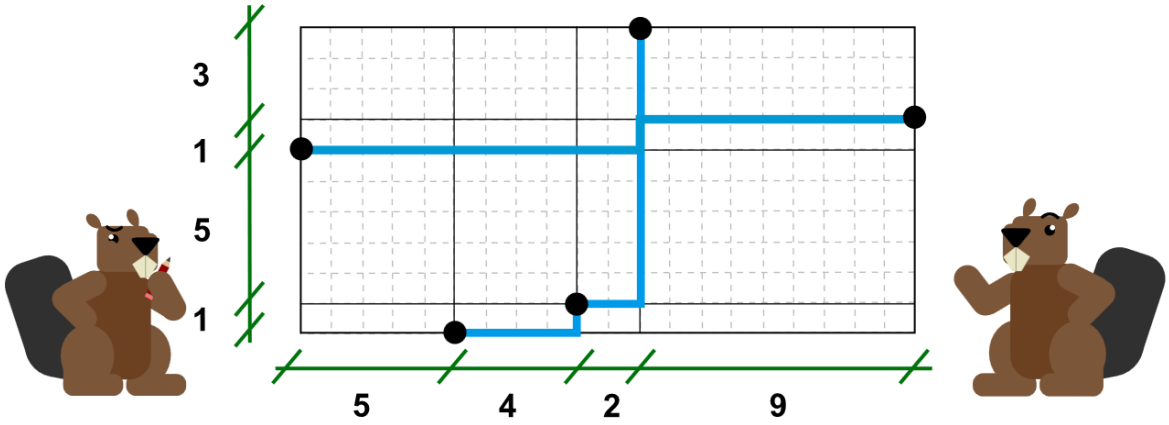
Isto é Pensamento Computacional!

Um *grafo* é constituído por um conjunto de nós e um conjunto de arestas, cada uma ligando dois nós. Uma sequência de arestas ligadas num grafo é designada por caminho. Neste problema, as aldeias são os vértices e as estradas que ligam as aldeias são as arestas.

O grafo neste problema é especial, uma vez que está ligado (existe um caminho de qualquer vértice para qualquer outro vértice seguindo as arestas do grafo) e nenhum vértice tem um número ímpar de arestas ligadas a ele. Para os grafos que têm estas duas propriedades especiais, existe uma forma de seguir todas as arestas do grafo exatamente uma vez e regressar ao vértice inicial, o que se chama um *circuito euleriano*. Os circuitos eulerianos foram discutidos pela primeira vez por Leonhard Euler enquanto resolvia o famoso problema das Sete Pontes de Königsberg em 1736. Podemos encontrar circuitos eulerianos aplicando o algoritmo de Fleury ou o algoritmo de Hierholzer. Um motorista de entregas pode gostar que o seu percurso seja um circuito euleriano para não ter de refazer os passos anteriores.

11. Plano do Canal

O arquiteto Mauro, o castor, apresenta o seu plano para a construção de um canal de água para ligar as cinco habitações da aldeia, indicadas no mapa com círculos pretos. O Mauro considera agradável seguir as linhas horizontais e verticais que atravessam cada uma das habitações; no seu desenho de 10 × 20, indica as distâncias entre essas linhas e, a azul mais carregado, um possível traçado do canal.



O castor Jaime observa que este canal tem um comprimento total de 36 unidades... no entanto podia ser mais curto!

Pergunta

Qual é o comprimento do canal mais curto que pode ser feito unindo segmentos horizontais e verticais nessas linhas que atravessam as cinco habitações?

Respostas possíveis

(A) 30 **(B)** 31 **(C)** 32 **(D)** 33 **(E)** 34 **(F)** 35

11. Plano do Canal (Resolução)

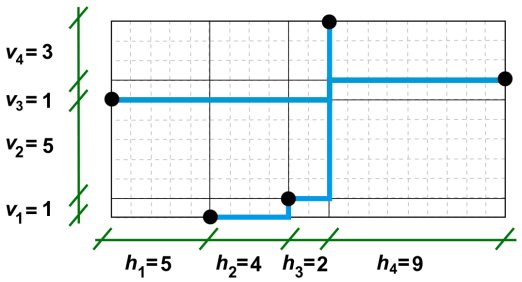
Solução

(E)

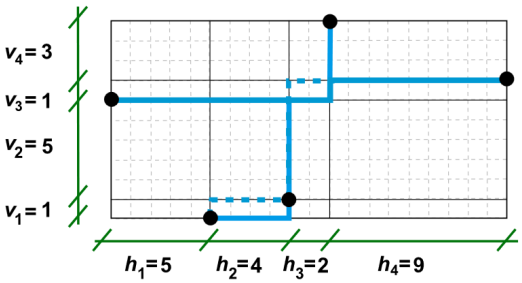
Resolução

A resposta correcta é 34.

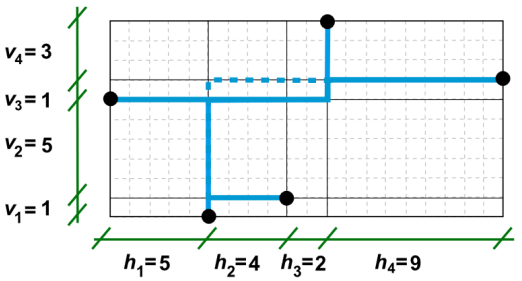
Consideremos os comprimentos das secções horizontais ($h_1 = 5$, $h_2 = 4$, $h_3 = 2$, $h_4 = 9$) e verticais ($v_1 = 1$, $v_2 = 5$, $v_3 = 1$, $v_4 = 3$), cada uma das quais deve ser "coberta"por pelo menos um troço de canal; assim, o comprimento mínimo não pode ser inferior a 30. (Note-se que este objetivo só poderia ser atingido se as habitações estivessem dispostas, no máximo, numa linha horizontal e numa linha vertical).



No projeto do Mauro (ver figura ao lado), as secções h_2 e h_3 são cobertas duas vezes e, de facto, o comprimento total do canal é $30 + 4 + 2 = 36$.



Para obter uma solução óptima, basta deslocar o troço do canal que cobre a secção v_2, como mostra a figura ao lado (onde os segmentos a tracejado mostram alternativas equivalentes para atravessar dois lados de um retângulo), de modo a que apenas a secção h_2 seja coberta duas vezes e o comprimento total seja $30 + 4 = 34$.



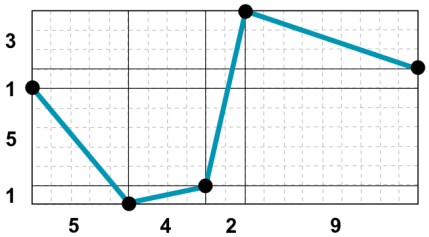
Deslocando o troço do canal que cobre a secção v_2 mais para a esquerda, obtêm-se duas outras soluções óptimas.

Claramente, cobrir a secção v_2 duas vezes não é conveniente, uma vez que $v_2 > h_2$. As restantes localizações possíveis para o único troço (vertical) que cobre a secção v_2 são as extremidades esquerda e direita do mapa; nestes casos, pelo menos a secção h_1 ou a secção h_4 (ambas maiores que h_2) seriam cobertas duas vezes, respetivamente. Isto mostra que o comprimento total mínimo é 34, com h_2 coberto duas vezes.

Isto é Pensamento Computacional!

Este problema pode ser enquadrado em *Geometria Computacional*, um ramo da Ciência de Computadores dedicado ao estudo de algoritmos que podem ser enunciados em termos de geometria

Se os castores tivessem querido simplesmente ligar as habitações da sua aldeia, sem introduzir outros pontos de intersecção, teriam tido de encontrar uma *árvore de suporte de custo mínimo*; no caso presente, o resultado seria o indicado na figura ao lado, com um comprimento total de cerca de $30,64$ unidades e, portanto, com uma economia de cerca de



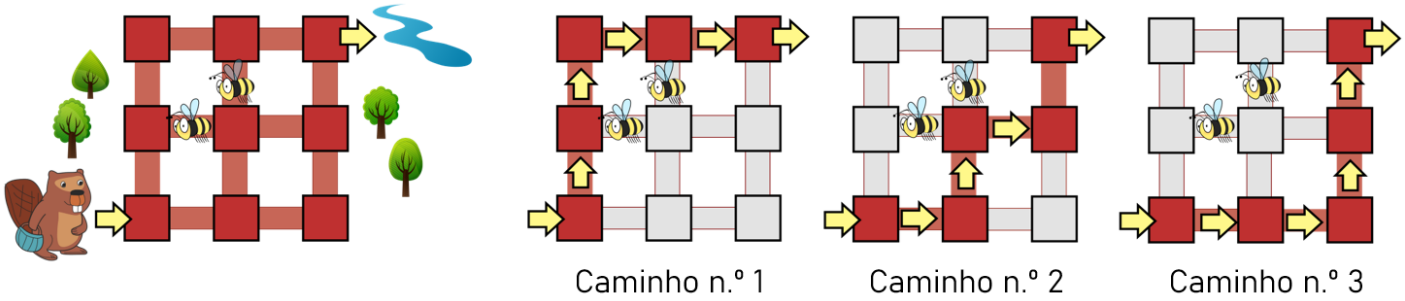
Existem muitas outras variações possíveis para o problema de encontrar a melhor maneira de ligar um conjunto de pontos. Este é um problema muito estudado e com muitas aplicações, como por exemplo saber como fazer as ligações em circuitos electrónicos ou como ligar um conjunto de casas à rede de eletricidade ou de água.

12. Caminhando na Floresta

O castor Pedro adora passear numa floresta. Ele começa sempre a sua caminhada no **canto inferior esquerdo** do mapa e vai até ao **canto superior direito**, onde se encontra um belo rio.

Além disso, **evita sempre as abelhas** e vai sempre **para cima ou para a direita** quando caminha .

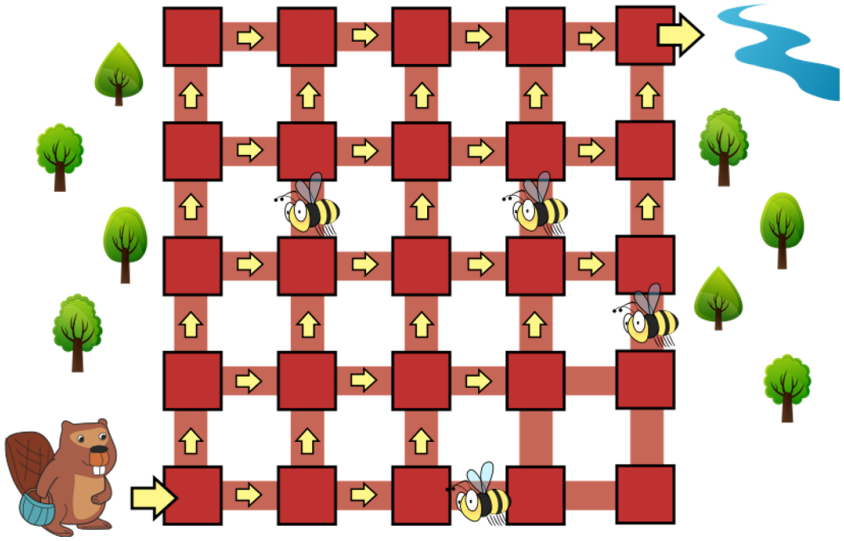
O Pedro aborrece-se facilmente e prefere sempre evitar percorrer exatamente o mesmo caminho, por isso rapidamente descobriu que a floresta tem 3 caminhos diferentes que pode utilizar:



O Pedro visita regularmente o seu amigo João, que vive numa floresta maior, e quer descobrir quantos caminhos diferentes existem lá. Podes ajudá-lo?

Pergunta

Quantos caminhos diferentes *pode o Pedro usar na seguinte floresta, assumindo que vai sempre para cima ou para a direita e que quer evitar as abelhas?*



Resposta
(escreve um número inteiro na folha de respostas)

12. Caminhando na Floresta (Resolução)

Solução

32

Resolução

Há 32 caminhos diferentes que evitam as abelhas e vão sempre para cima e para a direita.

Poder-se-ia tentar enumerar todos os caminhos possíveis, mas isso levaria muito tempo e seria muito suscetível a erros. Há, no entanto, uma estratégia que nos permite calcular muito facilmente a resposta desejada de uma forma que se adapta melhor à dimensão da floresta.

Comecemos com a floresta original 3×3 mais pequena. Poderíamos representá-la mais facilmente com uma matriz de 3×3, em que cada célula é uma interseção e as abelhas são representadas por segmentos vermelhos que indicam que não as podemos atravessar. Nesta grelha, queremos ir da célula $(1,1)$ para a célula $(3,3)$:

(1,3)	(2,3)	(3,3)
(1,2)	(2,2)	(3,2)
(1,1)	(2,1)	(3,1)

Assim, poder-se-ia escrever em cada célula o número de caminhos que, começando nessa célula, podem chegar à célula $(3,3)$. Inicialmente, introduz-se um 1 (para 1 caminho possível) no canto superior direito, e também um 1 em todas as células do topo (1.ª linha) e da direita (3.ª coluna) da grelha.

Sem considerar as abelhas, o número de caminhos possíveis a partir de (x,y), representado por $nr\_caminhos(x,y)$, é definido como $nr\_caminhos(x,y) = nr\_caminhos(x+1,y) + nr\_caminhos(x,y+1)$, ou seja, os caminhos a partir de uma célula são todos os caminhos que começam na célula acima mais os da célula à esquerda (isto funciona para todas as células exceto a linha superior ou a coluna da direita, razão pela qual as inicializámos primeiro). Se houver uma abelha a separar-nos de uma célula vizinha, simplesmente não somamos o número correspondente.

Podemos agora preencher as células com o número de caminhos de forma a que, quando precisarmos de um valor, este já lá esteja (por exemplo, preencher de cima para baixo e em cada linha da direita para a esquerda). No final, o nosso resultado está na célula $(1,1)$, que mostra os 3 caminhos existentes.

1	1	1
1	1	1
3	2	1

Para resolver a grelha 5×5 pedida, podemos agora simplesmente preencher de forma similar as respectivas células, com o resultado final na célula $(1,1)$:

1	1	1	1	1
5	4	3	2	1
9	4	4	1	1
18	9	5	1	0
32	14	5	1	0

Isto é Pensamento Computacional!

Para resolver este problema, foi necessário pensar num algoritmo adequado, ou seja, numa sequência de passos que permitisse efetuar o cálculo desejado. Normalmente, existem vários algoritmos possíveis para um determinado problema, que demorariam diferentes quantidades de tempo a ser aplicados.

Tentar resolver enumerando todos os caminhos possíveis é uma estratégia conhecida como *pesquisa exaustiva* (ou *força bruta*), que não seria viável para florestas maiores, pois o número de caminhos cresce exponencialmente com o tamanho da floresta.

O algoritmo proposto aqui segue uma técnica conhecida como *programação dinâmica*. Na sua essência, primeiro dividimos o problema em instâncias menores do mesmo problema (o que é conhecido por *dividir para conquistar*), ou seja, caracterizamos a solução de um problema (o número numa célula) como sendo uma composição (neste caso uma soma) de dois problemas menores (as células vizinhas). Além disso, o número de células é muito menor do que o número real de caminhos, uma vez que, de certa forma, podemos reutilizar os resultados e nunca temos de recalcular o número de caminhos a partir de uma determinada célula.

A programação dinâmica é uma estratégia muito geral e pode ser aplicada a muitas classes diferentes de problemas. Os seus ganhos de eficiência devem-se precisamente ao facto de os resultados dos sub-problemas poderem ser reutilizados.