Castor Informático

O Desafio Internacional de Pensamento Computacional

Edição 2021

Categoria: **Juniores** (9º e 10º 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.

Conteúdo

Preâmbulo

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

Tem já uma longa história e foi iniciado em 2004 pela Prof. Valentina Dagienė, da Universidade de Vilnius, na Lituânia. 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 e envolve vários níveis de abstração e as capacidades mentais que são necessárias para não só desenhar programas e aplicações, mas também saber explicar e interpretar um mundo como um sistema complexo de processos de informação.

A expressão "pensamento computacional" tornou-se conhecida em 2006 e pode ser vista como a nova literacia do século XXI. O desafio do Bebras promove precisamente este tipo de habilidades e conceitos informáticos 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

O ***Bebras - Castor Informático*** é organizado pelo Departamento de Ciência de Computadores (DCC/FCUP) da Faculdade de Ciências da Universidade do Porto (FCUP), juntamente com o TreeTree2.

[dcc] DEPARTAMENTO DE CIÊNCIA DE COMPUTADORES
FACULDADE DE CIÊNCIAS DA UNIVERSIDADE DO PORTO

U.PORTO FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

TREETREE2

O **Departamento de Ciência de Computadores da Faculdade de Ciências da Universidade do Porto** é o ponto de contacto português junto da organização internacional. Para além de ser uma instituição de referência no ensino e na investigação, o DCC/FCUP apoia este tipo de iniciativas desde há muitos anos, sendo também um dos principais organizadores das Olimpíadas Nacionais de Informática.

O **TreeTree2** é uma organização sem fins lucrativos que pretende cumprir o potencial criativo e intelectual dos jovens. Desenvolve vários programas de divulgação e ensino da ciência e engenharia. Noutras iniciativas, e na promoção e desenvolvimento do pensamento computacional em particular, conta com o apoio do Instituto Superior Técnico e financiamento da Fundação Calouste Gulbenkian.

TÉCNICO LISBOA

FUNDAÇÃO CALOUSTE GULBENKIAN

Estrutura da Prova

- Existe apenas uma fase, a qual é constituída por uma prova escrita com questões de escolha múltipla ou de resposta aberta. Existem perguntas de três níveis de dificuldade diferentes, cuja pontuação é da seguinte forma:

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

- A prova é individual e tem a duração de 45 minutos.
- Os alunos respondem unicamente na folha de respostas, independente do enunciado da prova, a qual será fornecida conjuntamente com a prova. As respostas deverão ser depois preenchidas numa folha de cálculo que será fornecida ao professor responsável, que a deverá posteriormente enviar para a organização.
- **Os enunciados da prova devem ser recolhidos no final do concurso**. Os alunos poderão consultar mais tarde novamente os enunciados quando estes foram divulgados publicamente.
- **As possíveis folhas de rascunho entregues aos alunos também devem ser recolhidas no final do concurso.**
- A gestão de situações de fraude ou de comportamento impróprio durante a realização do concurso ficará a cargo da Escola que deverá gerir a situação de acordo com as suas regras internas.

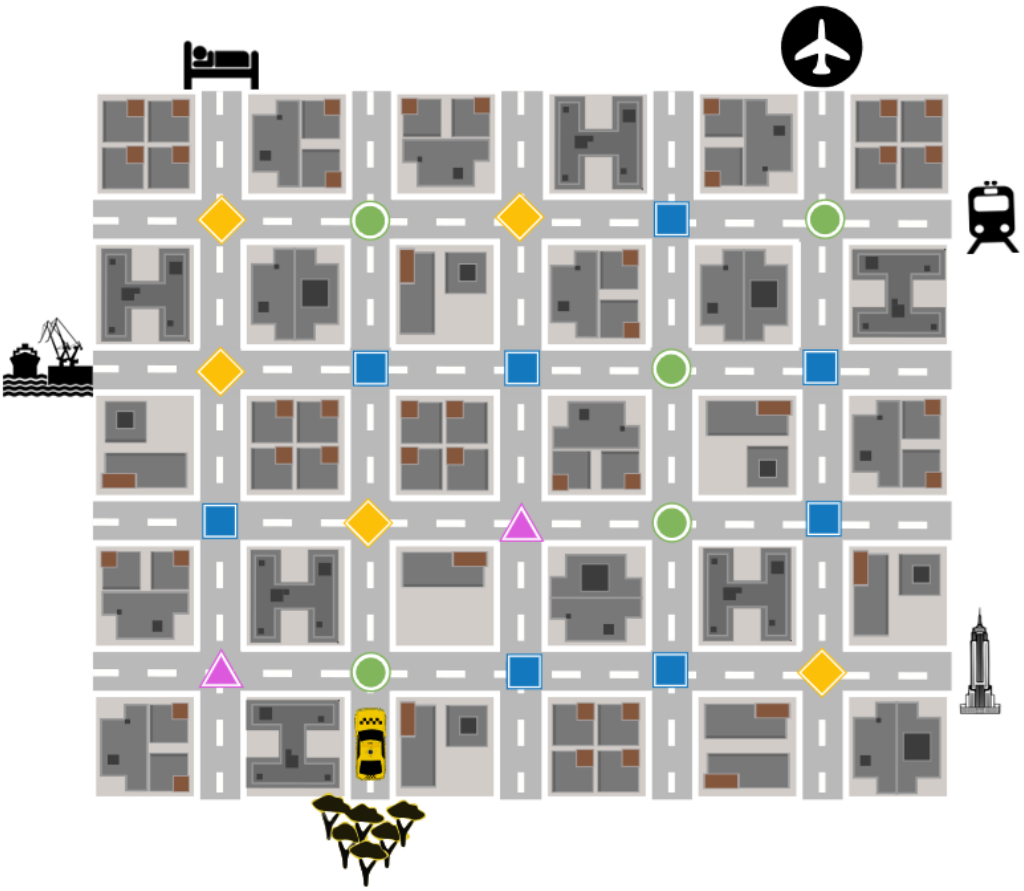
Sobre os Problemas

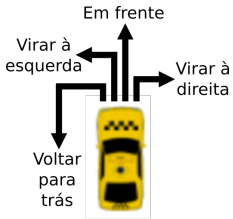
Os nomes dos autores dos problemas serão discriminados na versão final a divulgar no sítio oficial do Bebras - Castor Informático. Os problemas foram escolhidos, traduzidos e adaptados pela organização portuguesa. Para a edição portuguesa deste ano foram usados problemas com autores originários dos seguintes países:

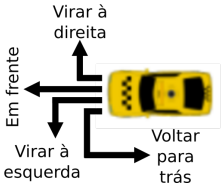
- Alemanha
- Áustria
- Bélgica
- Canadá
- Coreia do Sul
- Eslováquia
- Eslovénia
- Espanha
- EUA
- Irlanda
- Islândia
- Lituânia
- Paquistão
- Portugal
- R. Checa
- Suiça
- Ucrânia
- Uruguai
- Uzbequistão

1 – Sinais de Trânsito

Na cidade inteligente de Bebrasópolis, os sinais de trânsito sabem para onde é que os táxis autónomos se devem dirigir e dão-lhes direções usando os seguintes símbolos:







Pergunta

Os sinais de trânsito desta imagem direcionam o táxi do parque  *até ao aeroporto*  . *Qual é o significado de cada sinal de trânsito?*

Respostas Possíveis

(A)
- Em frente
- Virar à direita
- Virar à esquerda
- Voltar para trás

(B)
- Em frente
- Virar à esquerda
- Virar à direita
- Voltar para trás

(C)
- Virar à direita
- Virar à esquerda
- Em frente
- Voltar para trás

(D)
- Virar à esquerda
- Virar à direita
- Em frente
- Voltar para trás

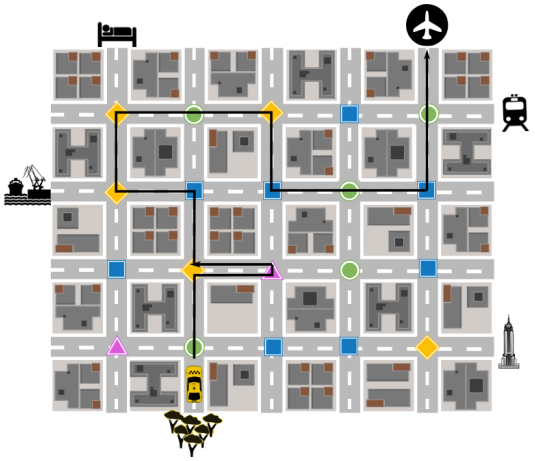
1 – Sinais de Trânsito (Resolução)

Solução

(A)

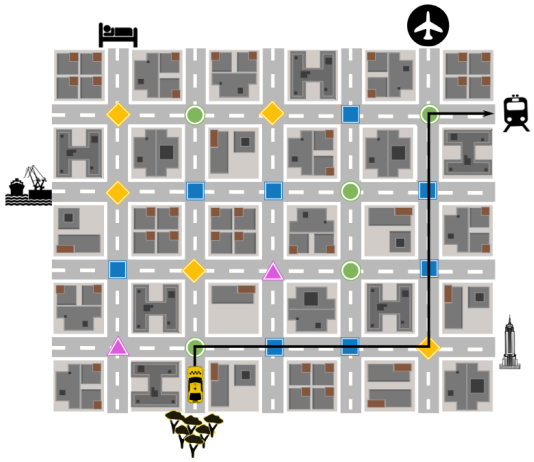
Resolução

A resposta correta é (A). Explicação (o caminho desde o ponto de início até ao destino é o representado):

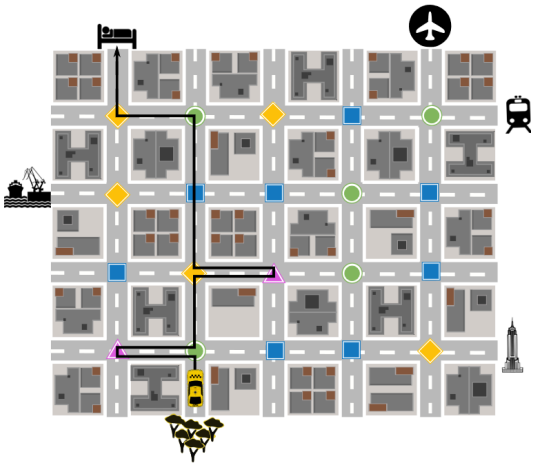


A resposta (B) está incorreta:

A resposta (C) está incorreta:



A resposta (D) está incorreta:



Isto é Pensamento Computacional!

O conceito de pensamento computacional ilustrado neste exercício são algoritmos. Um programa de computador muito simples é escrito utilizando quatro tipos diferentes de instruções. De acordo com o output do programa, descobre-se que símbolo corresponde a cada instrução.

Carros autónomos e outros veículos autónomos são exemplos de inteligência artificial que aos poucos se torna parte do quotidiano. O táxi neste exercício precisaria de estar equipado com uma vasta gama de sensores (como câmaras, radares, ultrassons) para perceber o seu ambiente. Software de visão computacional utilizaria estes sensores para manter o táxi na faixa, seguir sinais e evitar peões.

Enquanto que o táxi neste exercício é automático, não é completamente autónomo porque segue um sinal após o outro para chegar ao seu destino. Um veículo autónomo utilizaria inteligência artificial para decidir o seu próprio percurso com base no ambiente, dados de GPS e mapas, relatórios de trânsito e até mesmo informação de outros veículos autónomos!

2 – Mensagem com Troncos

Os castores da margem norte do Rio Madeira pensaram num número entre 0 e 15. Para indicarem aos seus amigos da margem sul em que número pensaram, eles deixam uma mensagem à beira rio utilizando um código que inventaram.

Eles usam quatro troncos que são posicionados na vertical ou na horizontal. Cada tronco vale um número diferente. A começar pelo tronco mais à esquerda, o primeiro vale 8, o segundo vale 4, o terceiro vale 2 e o quarto (o mais à direita) vale 1. Quando um tronco é posicionado na vertical significa que o castor deve adicionar o valor correspondente. Quando é posicionado na horizontal, o valor desse tronco deve ser ignorado.

A imagem abaixo mostra um código indicando o número 7, porque o tronco mais à esquerda está na horizontal e os restantes três à direita estão na vertical ($0 + 4 + 2 + 1 = 7$).

Pergunta

Que código deveria ser usado para indicar o número 11?

Respostas Possíveis

(A) 

(B) 

(C) 

(D) 

2 – Mensagem com Troncos (Resolução)

Solução

(B)

Resolução

A resposta correta é (B).

A contar da esquerda, o primeiro tronco vale 8, o segundo 4, o terceiro 2 e o quarto 1. A opção B tem o primeiro, terceiro e quarto troncos posicionados verticalmente (a contar da esquerda), o que significa que o Castor deveria adicionar o valor desses 3 troncos. Ou seja, $8 + 0 + 2 + 1 = 11$.

A opção A está errada porque o valor é $0 + 0 + 2 + 1 = 3$.

A opção C está errada porque o valor é $0 + 4 + 0 + 1 = 5$.

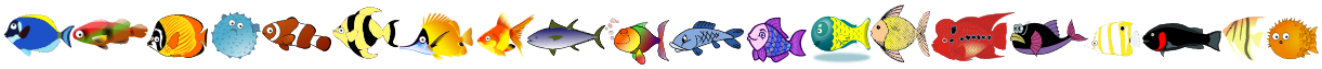
A opção D está errada porque o valor é $8 + 4 + 0 + 1 = 13$.

Isto é Pensamento Computacional!

Este exercício é sobre o sistema de representação de números binários. Os computadores armazenam fisicamente apenas dois tipos de valores: zeros e uns. Isto é extremamente conveniente em termos do desenho do equipamento. Portas lógicas e chips são muito mais fáceis de desenhar se apenas tiverem de processar dois valores (como uma voltagem alta: 1, ou baixa: 0). Num sistema de representação binário, cada dígito representa uma potência de base 2 e os zeros e uns indicam se cada potência de 2 deve ser adicionada. Por exemplo, o número binário $100101 = 1 \times 32 + 0 \times 16 + 0 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 37$ é representado no sistema decimal por 37.

3 – Peixes em Linha

Os seguintes peixes nadam em linha, como mostra a figura abaixo:



Ocasionalmente, alguém diz a posição de dois peixes. Se as posições forem A e B, tais que A<B, então:

- todos os peixes à esquerda do peixe na posição A fogem e
- todos os peixes à direita do peixe na posição B fogem.

Por exemplo, depois de alguém dizer as posições 2 e 17, haveria 16 peixes restantes na fila (agora nas posições 1, 2, ..., 16) como se segue:

As posições estão numeradas desde 1, à esquerda, e são renumeradas depois de algum peixe fugir. Começando com a linha original de 20 peixes,

- alguém diz as posições 4 e 18, depois
- alguém diz as posições 6 e 12 e depois
- alguém diz as posições 2 e 5.

Pergunta

Depois disto, qual das seguintes é a nova fila de peixes?

Respostas Possíveis

(A) 

(B) 

(C) 

(D) 

3 – Peixes em Linha (Resolução)

Solução

(D)

Resolução

A resposta correta é a D.

Uma forma de determinar que peixes ficam é registar a fila inteira restante de peixes depois de cada vez que alguém diz duas posições. Contudo, podemos ser mais inteligentes e tomar nota apenas dos peixes que restam mais à esquerda. Isto porque quando os peixes fogem, apenas os peixes que estavam originalmente em posições adjacentes permanecem em linha.

Depois das posições 4 e 18 serem chamadas, os peixes  ,  e  fogem, tal como alguns à direita da posição 18, portanto  vai ser o peixe restante mais à esquerda. Além disso, $18-4+1=15$ peixes vão continuar em linha. (No geral, depois de chamar as posições $A+B$, $B-A+1$ peixes vão permanecer em linha. Consegues perceber porquê?)

A seguir, as posições 6 e 12 são chamadas,os peixes  ,  ,  ,  e  fogem, assim como os peixes à direita do da posição 12, pelo que  vai ser o peixe restante mais à esquerda. Além disso, $12-6+1=7$ peixes vão permanecer em linha.

Finalmente, depois de as posições 2 e 5 serem chamadas, o peixe  foge, tal como alguns peixes à direita do da posição 5, portanto,  vai ser o peixe restante mais à esquerda. Além disso, $5-2+1=4$ peixes permanecem em linha.

Isto significa que os quatro peixes a começar com  constituem a nova linha final de peixes.

Isto é Pensamento Computacional!

Qunado um programador de computadores trabalha com dados, ele precisa de determinar como representar esses dados. Dados relacionados são normalmente armazendos juntos numa coleção. Neste caso, uma segunda decisão importante é determinar como organizar a coleção na memória. Diferentes tipos de dados podem ser utilizados para isto e um dos tipos de dados mais comum é a sequência. Neste exercício, os peixes estão organizados numa sequência.

Tipos de dados são normalmente associados a operações comuns aplicadas nos dados. A operação chave neste exercício é a seleção de peixes entre duas dadas posições. Esta é das operações de sequenciamento mais importantes no geral. Quando a sequência é uma lista de letras ou outros caracteres, é tipicamente chamada de cadeia de caracteres e esta operação comum é frequentemente chamada de sub-cadeia de caracteres, ou fatia, ou algo semelhante.

4 – Observar a Floresta

Os guardas florestais têm de observar os tipos de animais que passeiam nos caminhos. Eles observam os caminhos a partir de torres de observação muito altas. Em cada torre de observação só há espaço para um guarda florestal.

Quando um guarda florestal está numa torre, ele consegue observar apenas os caminhos adjacentes a essa torre, ou seja, ele consegue observar apenas os caminhos que partem (ou chegam) a essa torre.

Pergunta

Qual é o número mínimo de torres que têm de ter um guarda florestal para ser possível observar todos os caminhos?

Resposta

Escreve a tua resposta (um número inteiro entre 1 e 7).

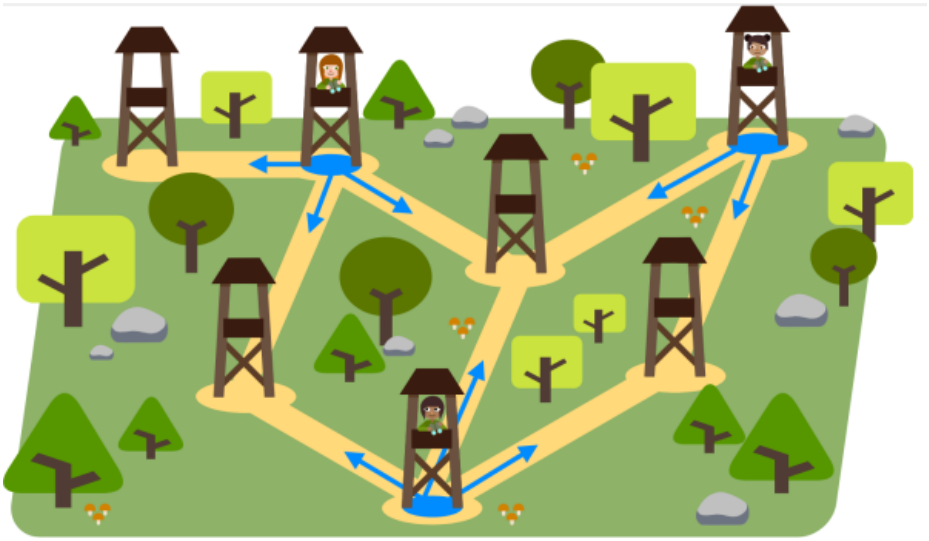
4 – Observar a Floresta (Resolução)

Solução

3

Resolução

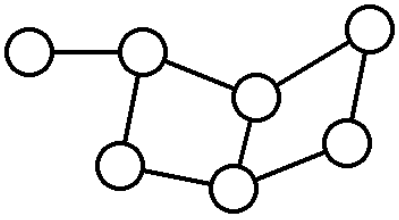
A resposta correta é 3. As três torres de observação ocupadas e os respetivos caminhos observados estão representados na figura abaixo.



Há 8 caminhos. Se houvesse apenas duas torres de observação ocupadas, uma delas teria de observar pelo menos 4 caminhos. Isto não é possível porque nenhuma das torres está ao lado de 4 caminhos.

Isto é Pensamento Computacional!

Em informática, muitas coisas podem ser representadas por grafos. Grafos consistem em nós (= círculos) e arestas (= linhas) que ligam os nós. Para o nosso exemplo, um grafo tem este aspecto:



Podemos perguntar-nos "Quais os nós (= torres de observação) que temos de escolher de forma a que todas a que todas as arestas (= caminhos da floresta) estão ao lado de um nó (= torre de observação) escolhido?". Esta pergunta é também conhecida como cobertura de vértices mínima. Pode ser aplicada, por exemplo, quando se instalam candeeiros nos passeios, que devem iluminar todas as ruas. Outro exemplo são câmaras, que devem cobrir todos os corredores.

5 – Pulseiras

A Maria gosta muito de fazer pulseiras com missangas de várias formas e gostava de partilhar facilmente os seus desenhos com os seus amigos usando uma representação compacta. Cada forma é descrita com uma única letra (E para estrela, T para triângulo, R para retângulo e L para linha). Em vez de escrever a sequência de missangas na pulseira, ela usa as seguintes regras:

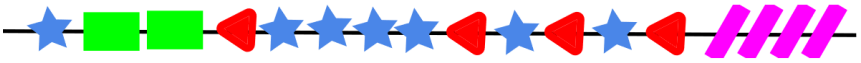
- Se houver várias missangas iguais seguidas umas das outras, ela pode simplesmente escrever o número de missangas e depois a letra correspondente;
- Se houver um padrão repetido de missangas, ela pode escrever o número de repetições e depois a sequência repetida entre parênteses;
- De outra forma, pode simplesmente escrever a letra da missanga.

Por exemplo, para a pulseira da imagem abaixo:

Uma descrição possível seria: ETRTRTREEEL com 11 símbolos.
Uma outra descrição possível seria: E3(TR)3EL com um comprimento de 9 símbolos.

Pergunta

Quantos símbolos há na representação mais curta para a pulseira da imagem seguinte?
(Nota: um símbolo é um dígito, uma letra ou um parêntese)



Respostas Possíveis

(A) 12

(B) 13

(C) 14

(D) 15

5 – Pulseiras (Resolução)

Solução

(B)

Resolução

A resposta correta é a (B), 13 símbolos. Há várias representações possíveis para esta pulseira, dependendo se se usa o padrão repetido estrela-triângulo ou triângulo-estrela. Todos eles dão origem a uma representação, sendo que os dois resultados abaixo são os mais curtos:

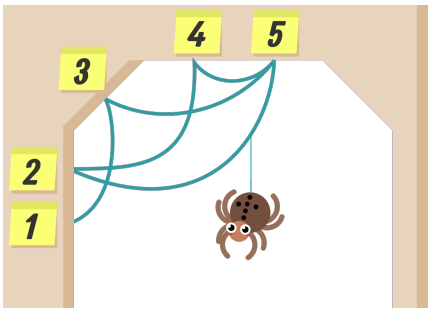
- E2RT3E3(ET)4L com 13 símbolos;
- ERRT4STETET4L com 13 símbolos;
- E2RT4E2(TE)T4L com 14 símbolos.

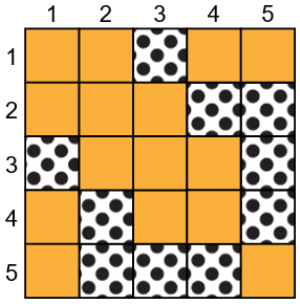
Isto é Pensamento Computacional!

Este exercício está relacionado com o ramo da informática dedicado à compressão de dados. A ideia deste ramo é desenhar técnicas que permitam representar e guardar dados ocupando a menor quantidade de memória, sendo capaz de recuperar os dados originais a qualquer momento. Todas estas técnicas utilizam a mesma observação, que é identificar redundâncias para poder representá-las de forma mais compacta. Isto é exatamente o que acontece neste exercício quando a repetição das missangas é representada numa forma muito mais curta do que explicitamente enumerando as missangas repetidas ou sequência repetida de missangas.

6 – Teias de Aranha

Quando a Vanda vê uma teia de aranha interessante, ela usa-a como inspiração para fazer uma manta. Ela numera os pontos onde a teia se fixa de 1 a N e depois organiza o tecido numa grelha de N por N:

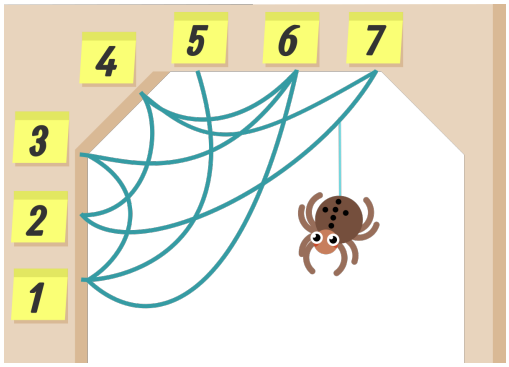




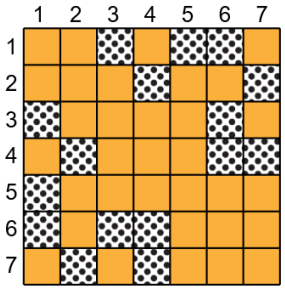
Por cada fio da teia de aranha, se ele se fixa nos números X e Y, ela coloca dois quadrados de tecido às bolinhas na sua grelha. Um pedaço quadrado de tecido às bolinhas é colocado onde a linha X se cruza com a coluna Y. Outro quadrado de tecido às bolinhas é colocado onde a coluna Y se cruza com a linha X. O resto da grelha é preenchido utilizando quadrados de tecido de padrão liso. A figura acima indica uma teia de aranha e a manta que ela inspirou.

Pergunta

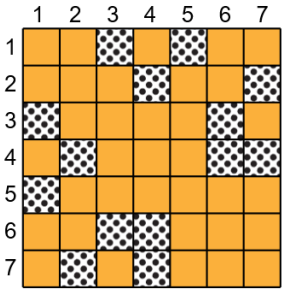
A Vanda viu agora a teia de aranha da imagem abaixo.
Como fica a manta que esta nova teia de aranha inspira?



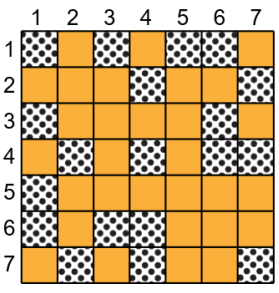
Respostas Possíveis



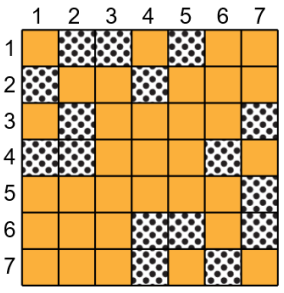
(A)



(B)



(C)



(D)

6 – Teias de Aranha (Resolução)

Solução

(A)

Resolução

A resposta correta é (A).

A teia tem um fio desde o ponto 1 até ao 3, 5 e 6. Assim, a primeira linha da manta terá quadrados de tecido às bolinhas nas colunas 3, 5 e 6.

A teia tem um fio desde o ponto 2 até ao 4 e 7. Assim, a segunda linha da manta terá quadrados de tecido às bolinhas nas colunas 4 e 7.

A teia tem um fio desde o ponto 3 até ao 1 e 6. Assim, a terceira linha da manta terá quadrados de tecido às bolinhas nas colunas 1 e 6.

A teia tem um fio desde o ponto 4 até ao 2, 6 e 7. Assim, a quarta linha da manta terá quadrados de tecido às bolinhas nas colunas 2, 6 e 7.

A teia tem um fio desde o ponto 5 até ao 1. Assim, a quinta linha da manta terá quadrados de tecido às bolinhas na coluna 1.

A teia tem um fio desde o ponto 6 até ao 1, 3 e 4. Assim, a sexta linha da manta terá quadrados de tecido às bolinhas nas colunas 1, 3 e 4.

A teia tem um fio desde o ponto 7 até ao 2 e 4. Assim, a sétima linha da manta terá quadrados de tecido às bolinhas nas colunas 2 e 4.

A opção B está incorreta porque falta tecido às bolinhas na linha 1, coluna 6 e na linha 6, coluna 1.

A opção C está incorreta porque há tecido às bolinhas colocado incorretamente na linha 1, coluna 1, na linha 4, coluna 4 e na linha 7, coluna 7.

A opção D está incorreta porque todo o padrão da manta está rodado 90 graus.

Isto é Pensamento Computacional!

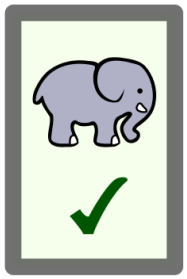
A teia de aranha pode ser considerada um grafo, um conceito frequentemente utilizado em ciência de computadores.

Um grafo é constituído por vértices (os pontos onde se ancoram a teia) e arestas (as peças de seda entre dois pontos de ancoragem). Os grafos são utilizados para representar objetos e as relações entre objetos. Por exemplo, um grafo pode mostrar pessoas que são amigas nas redes sociais ou voos entre países.

Neste exercício, a manta da Vanda demonstra uma forma alternativa de representar um grafo, conhecida como uma matriz de adjacências. Matrizes de adjacências são representações úteis, uma vez que fornecem uma forma eficiente de responder a questões sobre a estrutura de um grafo. Por exemplo, existe um vértice num sítio em particular? E quantas arestas se ligam a um dado vértice?

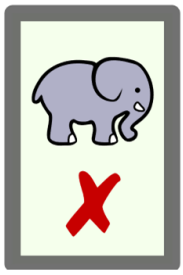
7 – Elefantes no Frigorífico

O pai da Joana usa cartas de dupla face para explicar o que aconteceu no frigorífico: um dos lados indica se algum elefante visitou o frigorífico e o outro indica se existem pegadas na manteiga. Estas são as 4 faces possíveis de um lado de uma carta:



Um elefante visitou o frigorífico

Existem pegadas na manteiga.



Nenhum elefante visitou o frigorífico.

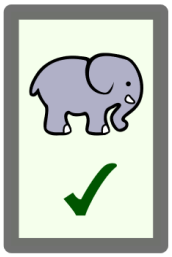
Não existem pegadas na manteiga.

O pai da Joana disse: "Se um elefante visitou o frigorífico, então existem pegadas na manteiga."
A Joana duvida que isto seja verdade.

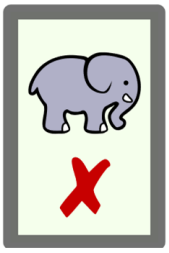
Pergunta

Assumindo que estás a ver apenas uma das faces da carta, seleciona todas as cartas com as quais a Joana poderia eventualmente provar que o pai está errado se visse também depois a imagem que está atrás. (nota que podes selecionar 1, 2, 3 ou 4 cartas)

Respostas Possíveis

(A) 

(B) 

(C) 

(D) 

7 – Elefantes no Frigorífico (Resolução)

Solução

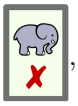
(A) e (D)

Resolução

A resposta correta é  e 

Se a Joana virar  e o outro lado disser que não há pegadas na manteiga, então ela prova com sucesso que o pai está errado. Claro que o outro lado pode dizer que há pegadas na manteiga, mas pelo menos virar esta carta dá à Joana uma oportunidade de provar ao pai que a sua afirmação está errada.

Se a Joana virar  , não interessa o que é que o outro lado diz. Se o outro lado disser que um elefante visitou o frigorífico, reforça a afirmação do pai. Se o outro lado disser que nenhum elefante visitou o frigorífico, então não contradiz a afirmação do pai, uma vez que ele não disse nada sobre o que aconteceria se nenhum elefante visitasse o frigorífico.

Se a Joana virar  , não interessa o que diz o outro lado, uma vez que o pai da Joana não disse nada sobre o que aconteceria se nenhum elefante visitasse o frigorífico.

Se a Joana virar  e o outro lado disser que um elefante visitou o frigorífico, então ela provou ao pai que a sua afirmação estava incorreta. Claro que o outro lado pode dizer nenhum elefante visitou o frigorífico, mas pelo menos virar esta carta dá à Joana uma oportunidade de provar ao pai que a sua afirmação está errada.

Isto é Pensamento Computacional!

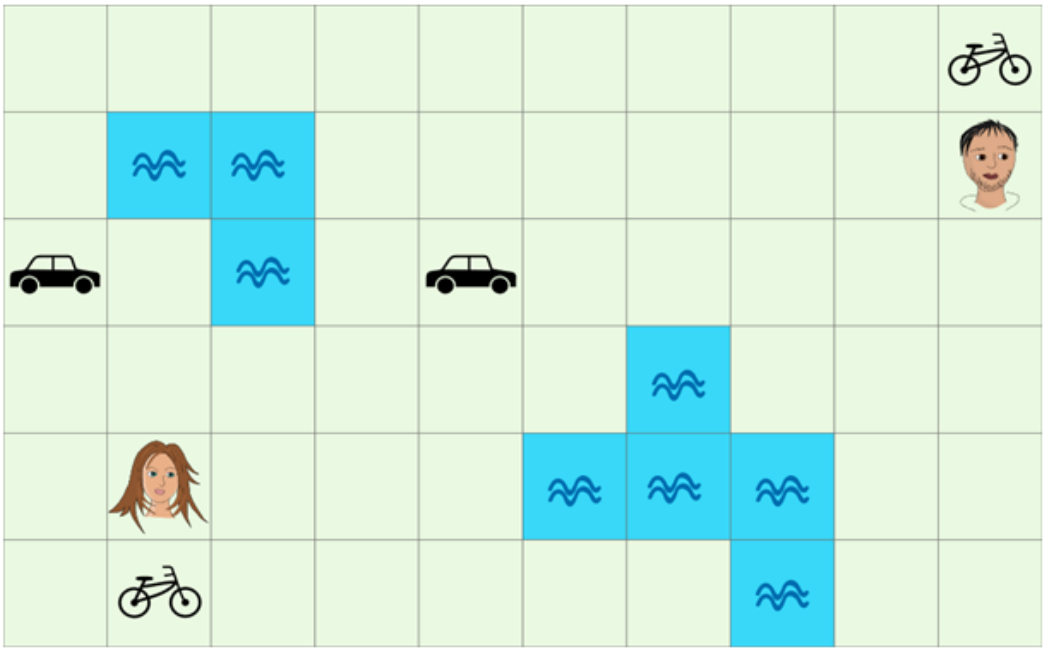
A lógica desempenha um papel crucial na teoria e aplicação da ciência de computadores, desde a simples lógica Booleana até às formas de lógica utilizadas nos sistemas de Inteligência Artificial modernos. A demonstração utilizada neste exercício chama-se Implicação ("Se A, então B", representado frequentemente como "$A \Rightarrow B$") . É frequentemente utilizada em Sistemas Periciais (sistemas que tentam emular a capacidade decisão de especialistas ou peritos) como o Prolog. Estes sistemas foram alguns dos primeiros sistemas de Inteligência Artificial bem-sucedidos. Os computadores conseguem representar demonstrações como estas internamente e combiná-las com outras demonstrações fornecidas - essencialmente executando matemática sobre informação - para calcular novos ou surpreendentes resultados. A Implicação pode ser representada como uma operação binária e, como tal, pode ser representada numa tabela, assim como AND (em português, "e"), NOT ("não") e OR ("ou"):

A	B	A ⇒ B
Falso	Falso	Verdadeiro
Falso	Verdadeiro	Verdadeiro
Verdadeiro	Falso	Falso
Verdadeiro	Verdadeiro	Verdadeiro

Esta tabela de verdade é algo surpreendente para várias pessoas porque usam "se A, então B" com o (diferente) significado "se e só se A, então B" que significa "exatamente quando A é verdade, então e só então B também é verdade". Cientistas de computadores às vezes escrevem "iff" em vez de "if and only if" (em português, "se e só se") se pretenderem utilizar este significado. Mas a afirmação original "se A, então B" é sempre verdadeira, excepto quando A é verdadeira e B é falsa.

8 – Encontro de Amigos

Dois amigos precisam de se encontrar urgentemente - vê o mapa abaixo. Para se deslocar eles podem ir do quadrado onde estão para um quadrado adjacente, na horizontal ou vertical, demorando exatamente um minuto. Se eles chegarem a uma bicicleta ou a um carro, eles podem utilizá-los para viajar mais depressa: 2 quadrados num minuto de bicicleta, 5 quadrados num minuto de carro. Eles não podem viajar por cima da água.



Pergunta

Qual é o número mínimo de minutos que os amigos precisam para se encontrarem no mesmo quadrado?

Resposta

Escreve a tua resposta (um número inteiro).

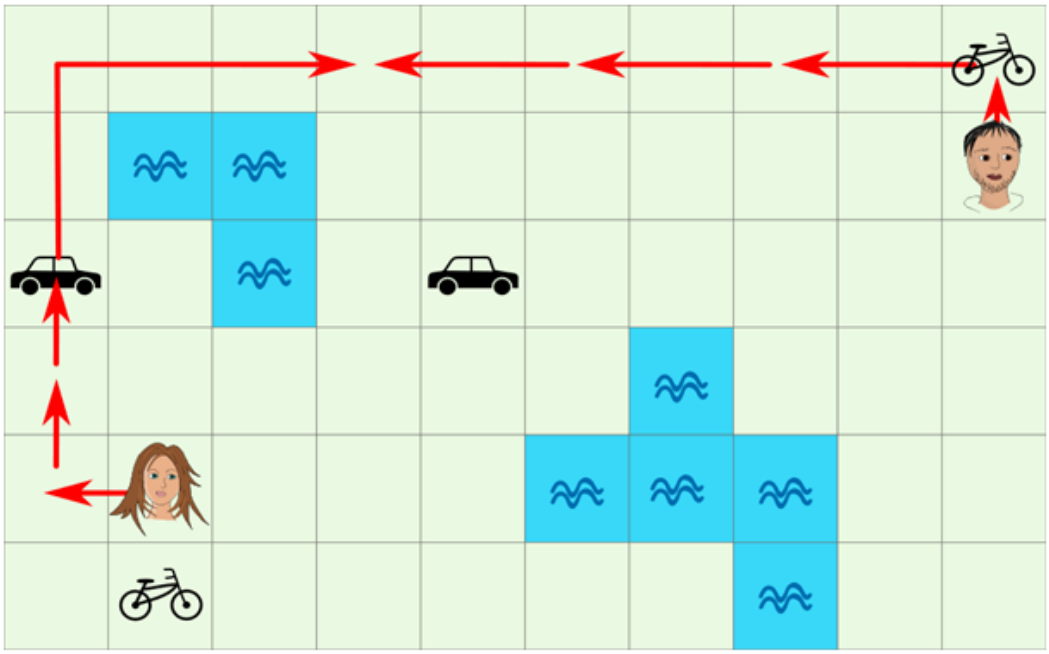
8 – Encontro de Amigos (Resolução)

Solução

A resposta correta é 4 minutos.

Resolução

A resposta correta é 4 minutos. Este valor pode ser obtido através da rota ilustrada abaixo:



Outra opção é apanhar a bicicleta mais à esquerda até ao carro mais à esquerda e depois continuar como na figura anterior.

Para perceber porque é que 3 minutos não são suficientes, seguem-se os seguintes pontos:

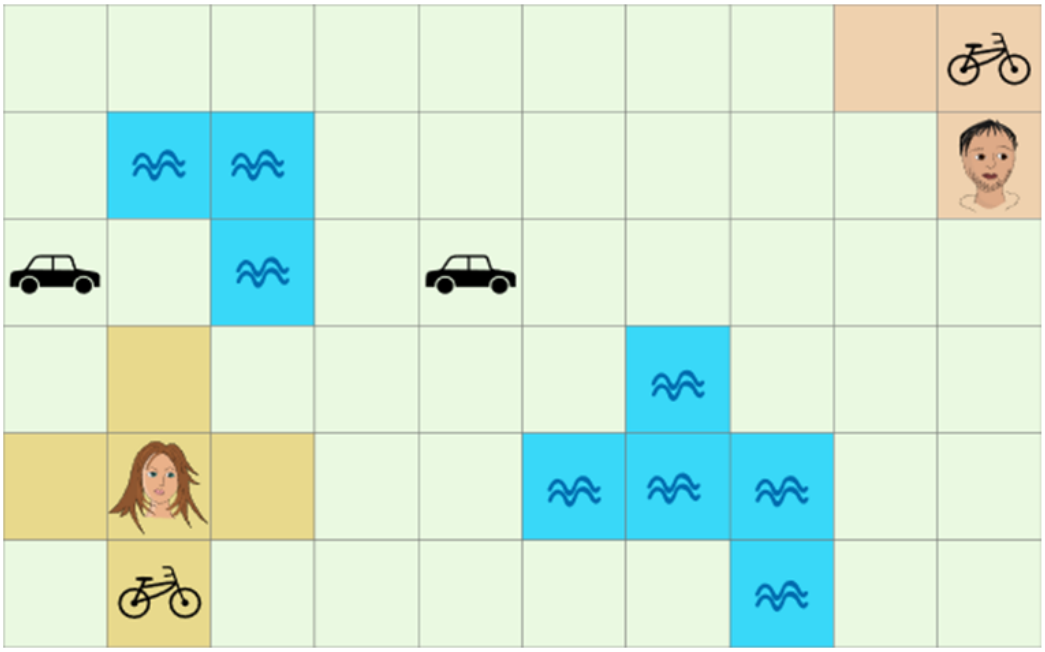
- Embora em 3 minutos seja possível chegar ao carro mais à esquerda, não há tempo suficiente para o conduzir a lado algum. E essa posição não pode ser atingida em três minutos pela outra pessoa. Portanto os carros não têm utilidade e poderíamos até removê-los do mapa.
- Os dois amigos estão a mais de 5 minutos de distância a pé, portanto precisam de uma bicicleta. Aliás, ambos precisam de uma bicicleta porque estão separados por mais de 9 posições. Mas chegar a essa bicicleta custa um minuto e com apenas dois minutos restantes não conseguem chegar um ao outro, mesmo de bicicleta.

Isto é Pensamento Computacional!

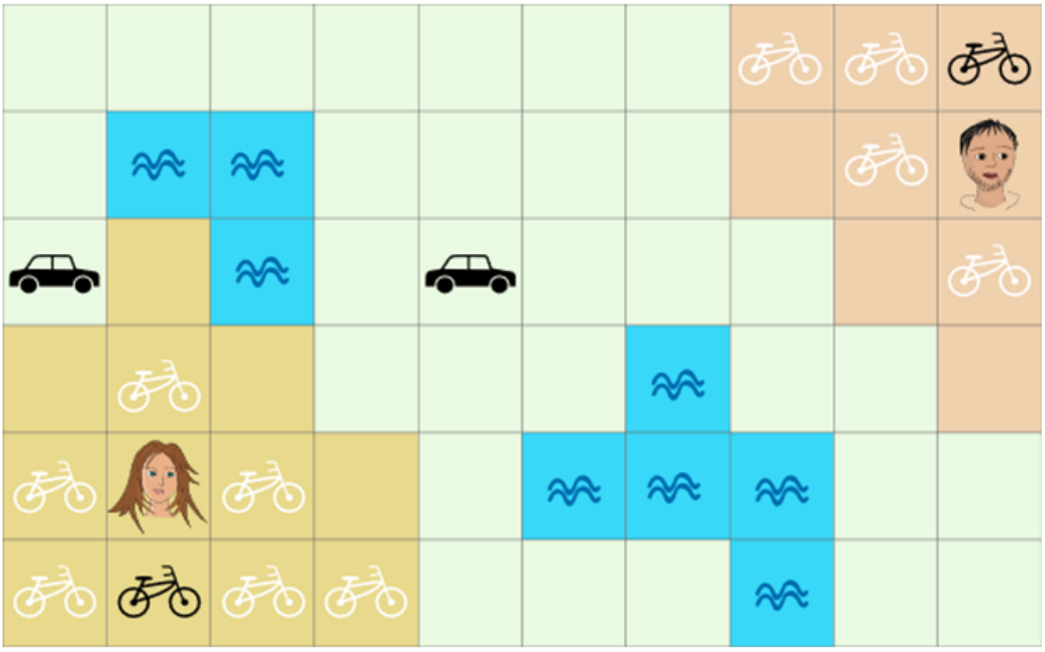
Como é que se procedeu para resolver este exercício? Encontrou-se uma rota curta por acidente e esperou que nenhuma outra mais curta pudesse ser encontrada, ou tentou dezenas de diferentes possibilidades,

tendo em mente o tempo mais curto? Um programa de computador desenhado para este tipo de tarefas utilizaria uma abordagem sistemática, mais provavelmente utilizando um algoritmo que se chama procura em largura (no original, em inglês, "breadth-first search"). Para este exercício, isto seria feito como se segue:

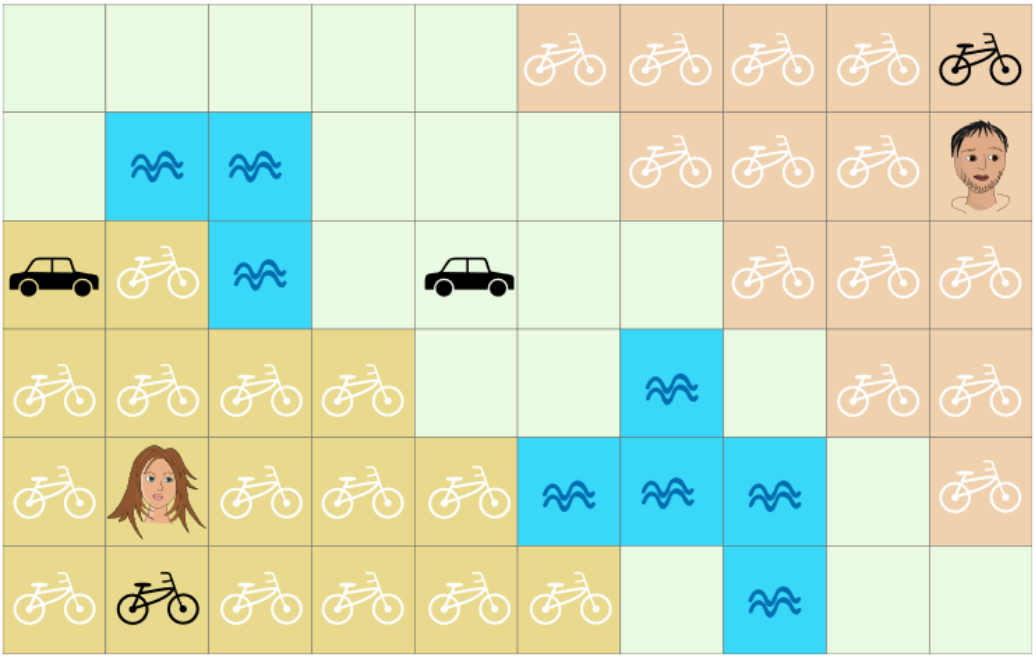
1. Marcar todos os quadrados no mapa que podem ser alcançados por cada amigo num minuto.



2. Marcar todos os quadrados que podem ser alcançados (no máximo) num minuto a partir das posições marcadas no passo 1 e registar que meio de transporte estavam a utilizar.

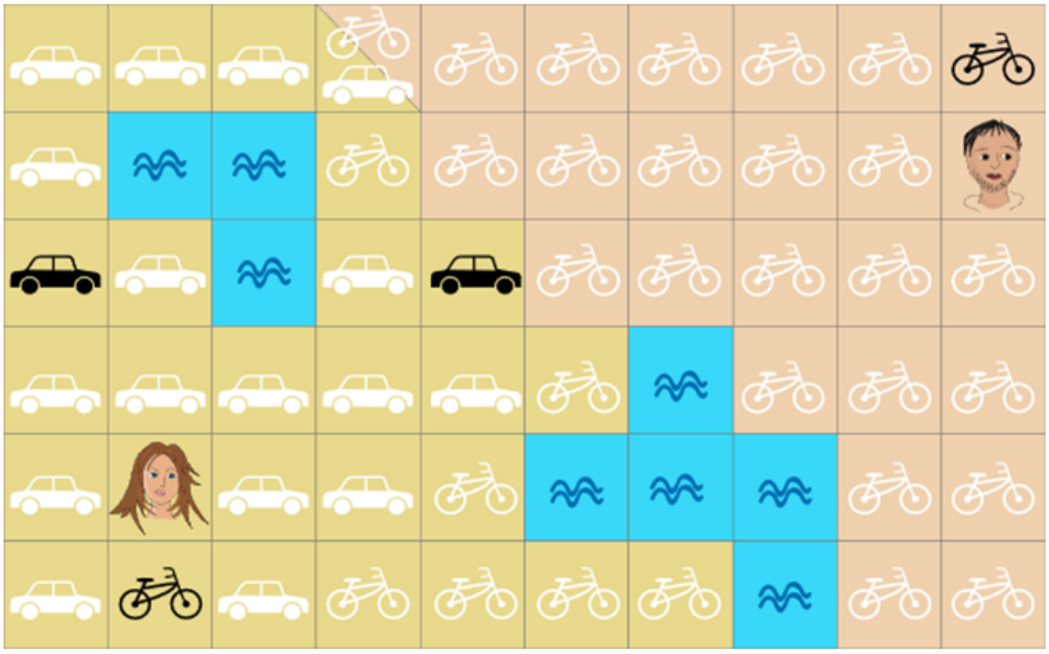


3. Marcar todos os quadrados que podem ser alcançados em um minuto a partir dos quadrados marcados em 2.



Como nenhumas das áreas marcadas se sobrepõem, podemos observar que os amigos não conseguem encontrar-se em 3 minutos.

4. Fazer mais um passo: marcar todos os quadrados que podem ser alcançados num minuto a partir dos quadrados marcados em 3.



Agora há duas regiões marcadas que se sobrepõem (num único quadrado), indicando que depois de 4 minutos os dois amigos conseguem encontrar-se. Provavelmente está familiarizado com o software que encontra a rota mais rápida entre dois locais num mapa, tendo o cuidado de apenas seguir estradas e não atravessar montanhas e rios. Este exercício é muito semelhante, mas agora duas pessoas movem-se uma na direção da outra em vez de uma pessoa se mover em direção a uma posição fixa.

Por causa da forma sistemática em que a procura por uma solução é efetuada, um computador frequentemente encontrará soluções que à partida não são óbvias - por vezes um desvio com menos semáforos pode ser uma melhor opção do que uma estrada direta ou um caminho de transportes públicos com várias trocas rápidas pode acabar por se revelar mais rápido do que um autocarro direto.

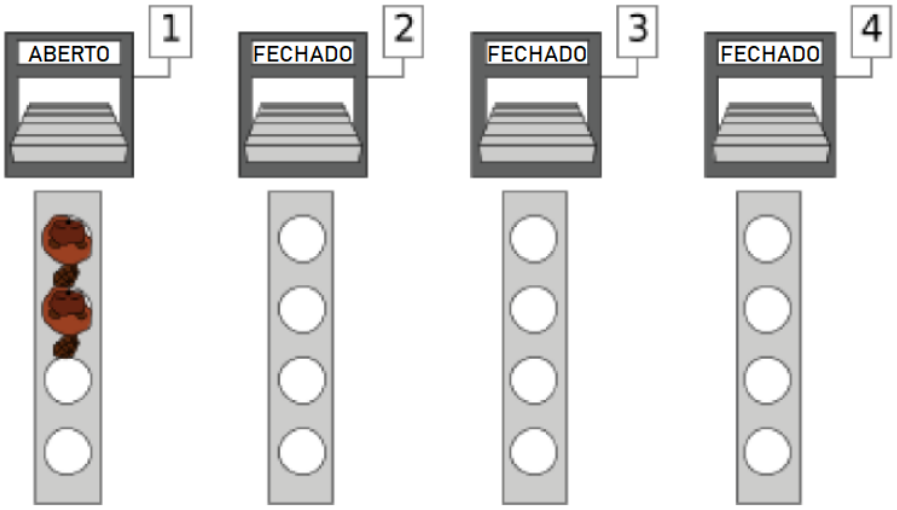
Em ciência de computadores, vários métodos são conhecidos por encontrarem a melhor solução para um problema como este. Tirando o método da busca em largura descrito acima, também há a técnica de ramificar e limitar, que é bastante semelhante mas utiliza atalhos práticos para acelerar a busca: se já encontrámos uma solução boa, então podemos descartar opções em que não conseguimos produzir uma solução melhor que a melhor encontrada até ao momento.

Quando um problema se torna demasiado complexo, percorrer todas as soluções possíveis para encontrar a melhor levará demasiado tempo, até para um computador rápido. E, na prática, frequentemente é suficiente encontrar uma muito boa resposta mesmo que não seja a melhor possível. (Se conseguir chegar ao destino em 78 minutos, provavelmente não se importará muito se outra rota o pudesse levar ao mesmo sítio em 77.)

Uma técnica utilizada nesse caso, é o mais intuitivo algoritmo ganancioso, que em cada passo escolhe o que parece ser a melhor opção no momento e não olha em frente para ver o que poderia acontecer em passos seguintes. Neste exercício isso significaria que os dois amigos tomariam sempre um passo que os aproximasse, o que nesta circunstância não representa uma boa estratégia porque então seguiriam a pé a maior parte do caminho. Existem, contudo, outros tipos de problema em que esta estratégia gananciosa produz resultados relativamente bons, e é muito mais rápida que outros métodos.

9 – Balcões de Atendimento

Uma loja tem quatro balcões de atendimento numerados 1, 2, 3 e 4. Cada balcão pode ter uma fila de no máximo 4 clientes, incluindo o cliente que está a ser atendido. Cada balcão consegue atender apenas um cliente de cada vez e demora 2 minutos a atender um cliente. Inicialmente, apenas o balcão 1 está aberto.



Quando um cliente quer pagar a conta, ele junta-se ao final da fila do primeiro balcão que não está cheio. Primeiro tenta o balcão 1, depois o 2, etc.

Se não houver espaço disponível em nenhum dos balcões, abre um novo balcão e o cliente espera nessa fila. Contudo, demora 1 minuto a abrir um novo balcão e portanto demora 3 minutos a atender o primeiro cliente de um balcão que acabou de abrir. Cada cliente seguinte será atendido em 2 minutos, como habitualmente.

Num dado momento, se houver clientes a deixar uma fila depois de serem atendidos e um novo cliente a querer juntar-se à fila, podes assumir que o cliente atendido sai primeiro e deixa um espaço livre na fila que novos clientes podem ocupar.

Pergunta

12 clientes chegam aos balcões, dois a cada minuto (dois clientes chegam inicialmente, outros dois após 1 minuto, etc.). Quanto tempo demora a atender todos os clientes?

Respostas Possíveis

(A) 8 minutos

(B) 11 minutos

(C) 12 minutos

(D) 13 minutos

9 – Balcões de Atendimento (Resolução)

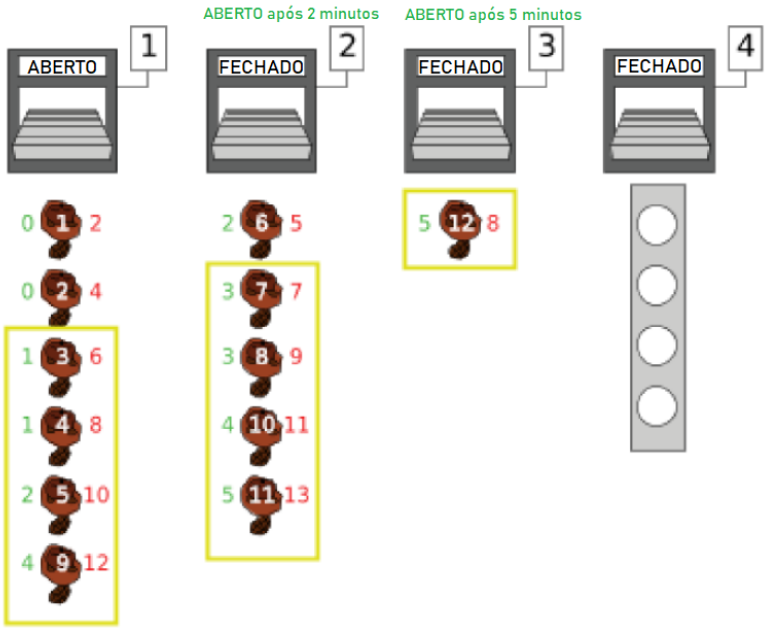
Solução

(D)

Resolução

A resposta correta é (D), 13 minutos.

Na imagem seguinte, podemos encontrar o tempo que em que cada cliente chega (a verde) e quando é que sai (a vermelho):



Como se pode observar, os tempos em que os clientes saem de cada fila são previsíveis, uma vez que cada cliente sai 2 minutos depois do último. Para resolver o problema, apenas precisamos de monitorizar estes tempos (representados a vermelho), que podem ser computados para cada cliente quando ele chega.

Após 2 minutos, quando 2 novos clientes chegam, podemos remover o primeiro castor do balcão 1 (pois este foi atendido) e um novo cliente junta-se à fila do balcão 1 e será servido 2 minutos depois do último cliente na fila ($8+2=10$), enquanto outros novos clientes vão para um novo balcão aberto (balcão 2) e serão servidos 3 minutos depois do tempo corrente ($2+3=5$).

Após 5 minutos, as filas serão as que estão representadas nos retângulos amarelos. Contudo, teremos de esperar que o último cliente seja atendido, que será o último cliente da segunda fila, aos 13 minutos.

Assim, a resposta correta é (D) 13 minutos.

A resposta (C) corresponde ao tempo em que o último cliente termina na primeira fila.

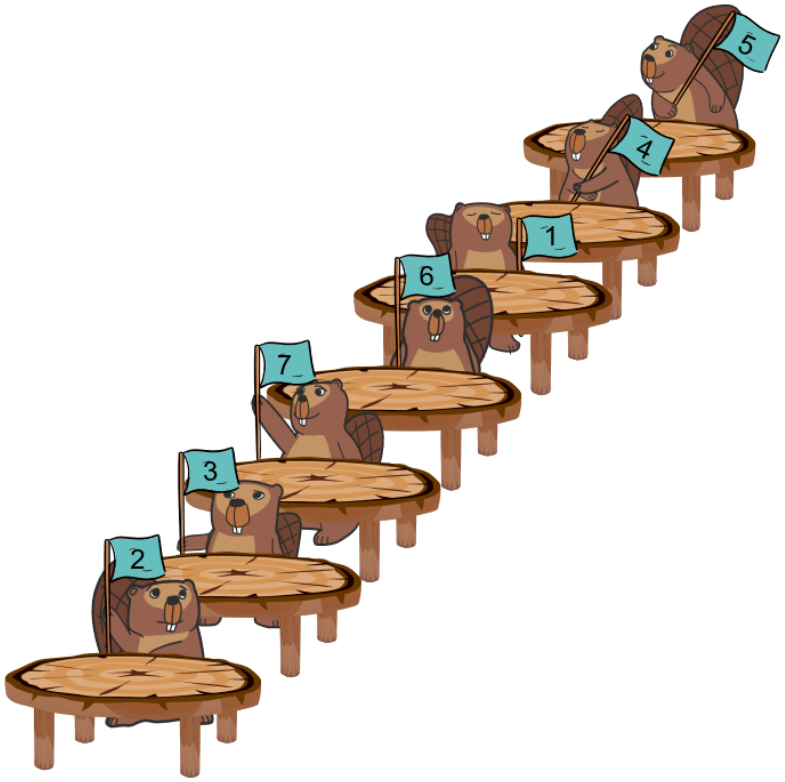
A resposta (A) corresponde ao tempo que o terceiro cliente termina na terceira fila.

Isto é Pensamento Computacional!

Serviços em nuvem como o Google Could, Amazon Web Services, Microsoft Azure, etc., escalam a disponibilidade dos recursos de computação dinamicamente, baseando-se nos requerimentos de utilização dos clientes. Desta forma, os clientes apenas pagam pelos recursos de que necessitam. Este provisionamento flexível é chamado de escalonamento dinâmico. Este exercício ilustra um simples exemplo de escalonamento dinâmico. Aqui os recursos são os balcões de atendimento. Manter um balcão de atendimento aberto requer pagar a mais um membro da equipa, por exemplo, portanto os balcões são abertos conforme a procura, à medida que o número de clientes aumenta. Isto mantém o custo baixo quando a procura é baixa. Na prática, o escalonamento dinâmico também envolve a redução da escala quando a procura baixa. No contexto deste exercício, isto implicaria uma estratégia que envolve fechar balcões quando o número de clientes diminui.

10 – Ordenando Sete Estudantes

Uma turma da Escola dos Castores tem apenas sete castores. A cada um foi dada uma bandeira com um número. Eles estão sentados numa fila, uns atrás dos outros. No início, eles estão sentados desordenadamente, como mostra a figura.



O professor da turma quer ordenar os castores desde o 1 à frente até ao 7 atrás. Eles apenas podem ser ordenados usando operações de troca. Em cada operação, apenas dois castores se podem mover, trocando de lugar um com o outro. Por exemplo: quando o castor 3 e o 1 trocam, quer dizer que o castor 3 vai para o lugar do 1 e o castor 1 vai para o lugar do 3. Utilizando um número finito de trocas, os castores estarão ordenados por ordem crescente da secretária da frente até à última.

Pergunta

Qual é o número mínimo de trocas necessárias para que os castores fiquem na ordem desejada?

Respostas Possíveis

(A) 3 trocas

(B) 4 trocas

(C) 5 trocas

(D) 6 trocas

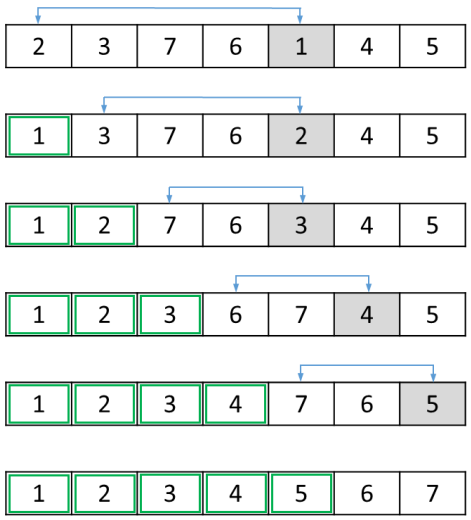
10 – Ordenando Sete Estudantes (Resolução)

Solução

(C)

Resolução

A resposta correta é (C), 5 trocas, uma vez que temos de trocar cinco vezes dois castores diferentes, como podemos ver na imagem seguinte.



Vamos aplicar o algoritmo de ordenação por seleção. O algoritmo divide a lista de entradas em duas partes: a sublista de itens já ordenados, que é construída da esquerda para a direita na frente (esquerda) da lista e a sublista de itens restantes que ainda não foram ordenados ocupam o resto da lista. Inicialmente, a sublista ordenada está vazia e a sublista não ordenada é a entrada inteira da lista. O algoritmo continua encontrando o elemento mais pequeno (ou o maior, dependendo da ordem pretendida) na sublista não ordenada, trocando-o com o elemento não ordenado mais à esquerda (colocando-o em ordem) e move os limites da sublista ordenada um elemento para a direita.

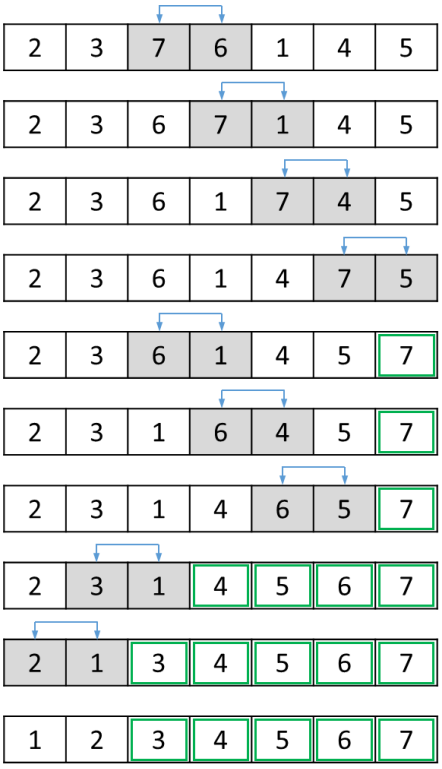
Com a primeira ordem (a ordem que podemos ver na imagem), selecionamos o número mais baixo (o número 1) e trocamo-lo com o número do primeiro lugar da lista (o número 2). No passo seguinte, a começar do segundo lugar da lista, selecionamos o número mais baixo (ou seja, o número 2) e trocamo-lo com o número do segundo lugar da lista (número 3). No final, precisamos de 5 passos.

Isto é Pensamento Computacional!

Algoritmos de ordenação permitem-nos, como indica o nome, ordenar informação de uma forma especial, com base num dado critério de ordenação. Em ciência de computadores, a ordenação de dados tem um papel importante, quer como um fim em si mesmo quer para outras operações mais complexas. Muitas técnicas foram desenvolvidas nesta área, cada uma com determinadas características e com as suas vantagens e desvantagens.

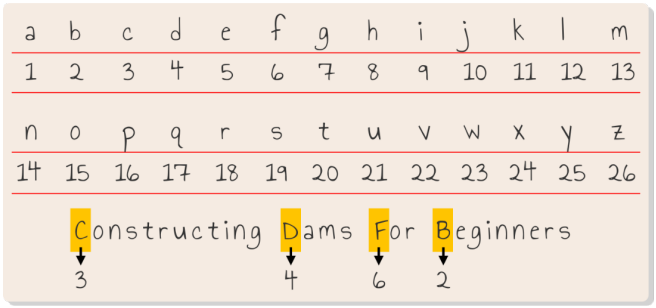
O algoritmo de ordenação por seleção melhora o algoritmo de ordenação por flutuação fazendo uma única troca por cada passagem pela lista. Para fazer isto, uma ordenação por seleção procura o número mais baixo enquanto faz uma passagem e, depois de completar a passagem, coloca-o na localização correta. Tal como acontece com a classificação por flutuação, após a primeira passagem, o item mais baixo está no local correto. Depois da segunda passagem, o seguinte número mais baixo está no local correto. Este processo continua e requer, em geral, $n - 1$ passagens para ordenar n itens, pois o item final deve estar no lugar após a $(n - 1)$-ésima passagem.

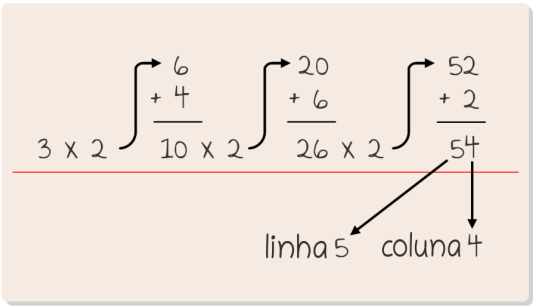
Dada a redução no número de trocas, a ordenação por seleção costuma correr mais rápido do que a ordenação por flutuação, embora, para outros problemas de ordenação, possa não ser o algoritmo mais rápido (há muitos algoritmos para ordenar "coisas"). Se compararmos a ordenação por seleção com a ordenação por flutuação, podemos ver que a primeira é mais rápida:

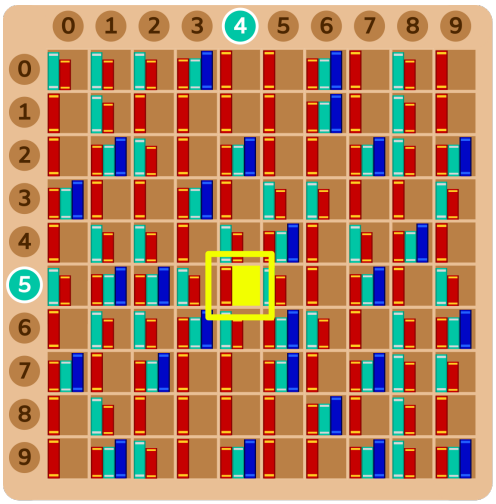


11 – Biblioteca

O Tiago acompanha a sua irmã mais velha Susana até à Biblioteca Bebras. A biblioteca tem apenas uma estante gigante. Eles querem requisitar o livro "Constructing Dams for Beginners". Assim que chegaram a Susana dirigiu-se à estante e tirou de lá o livro correto. "Como é que sabias onde estava o livro?", perguntou o Tiago. A Susana sorriu e mostrou-lhe um pedaço de papel:







"Tirei a primeira letra de todas as palavras no título e converti-as num número usando a tabela. A seguir, multipliquei o número da primeira letra por 2 e adicionei ao número da segunda letra. Depois, multipliquei esse resultado por 2 e adicionei ao número da terceira letra. Finalmente, multipliquei o resultado por 2 mais uma vez e adicionei ao número da última letra. Olhei para a linha do penúltimo dígito e para a coluna do último dígito para encontrar o livro. Foi muito fácil encontrar o livro certo a partir dos três que lá estavam", explicou a Susana. "Mas e para números maiores que 99?", perguntou o Tiago. A Susana respondeu "Eu ignoro todos os dígitos excepto os últimos dois."

Pergunta

Qual dos seguintes livros pode ser encontrado na linha 2 e coluna 4?

Respostas Possíveis

(A) Chewing on Trees Made Easy

(B) How to Avoid Falling Trees

(C) Tasty Trees to Gnaw On

(D) The Bark Gourmet Guide

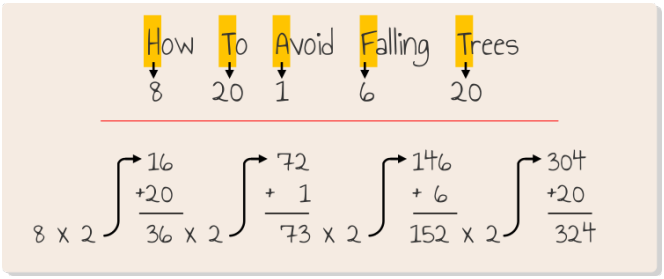
11 – Biblioteca (Resolução)

Solução

(B)

Resolução

A resposta correta é B: "How to Avoid Falling Trees", e a sua localização pode ser calculada do seguinte modo:



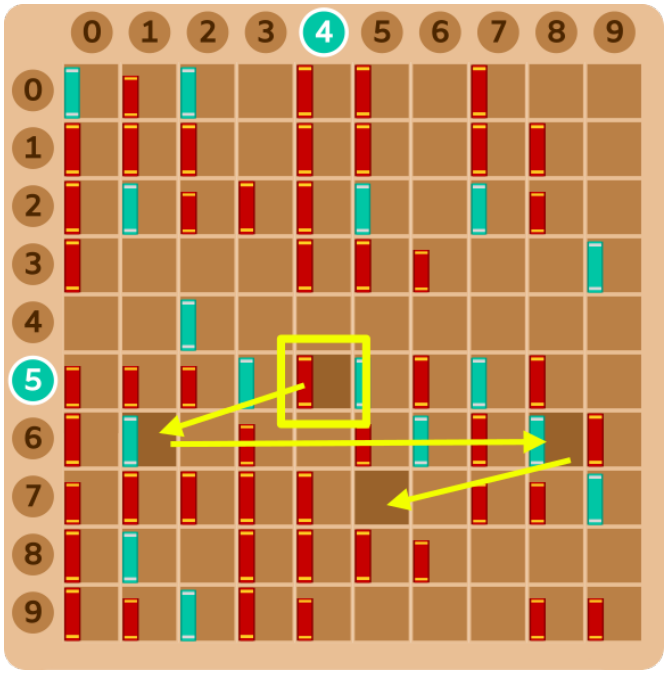
As outras respostas estão erradas. O livro (A) "Chewing on Trees Made Easy" está na linha 7 e coluna 9 (os números são 3, 15, 20, 13 e 5; os resultados intermédios são 21, 62, 137 e 279). O livro (C) "Tasty Trees to Gnaw On" está na linha 8 e coluna 9 (os números são 20, 20, 20, 7 e 15; os resultados intermédios são 60, 140, 287 e 589). Por fim, o livro (D) "Tree Bark Gourmet Guide" está também na linha 8 e coluna 9 (os números são 20, 2, 7 e 7; os resultados intermédios são 42, 91 e 189).

Isto é Pensamento Computacional!

Por detrás do algoritmo da Biblioteca do Bebras está um conceito denominado "hashing". Se nenhum sistema estiver por detrás da estante, seria necessária uma busca (linear) de cada livro para encontrar o livro desejado. Em média, teriam de ser verificados 50% de todos os livros para encontrar o desejado. Imagine-se fazer esta procura na Biblioteca de Alexandria ($\approx$ 100000 livros), na Biblioteca do Congresso dos Estados Unidos da América ($\approx$ 38000000 de livros) ou até simplesmente numa biblioteca local ou na biblioteca da escola.

O problema não existe só para as bibliotecas. Grandes farmácias também precisam de ter um sistema para armazenar e encontrar os medicamentos. Nos últimos anos, cada vez mais farmácias estão a adotar sistemas de armazenamento automatizados. Para eles, uma ordenação sistemática, por exemplo, por tipo de medicamento, não interessa. Em vez disso, procuram uma distribuição equilibrada nas estantes.

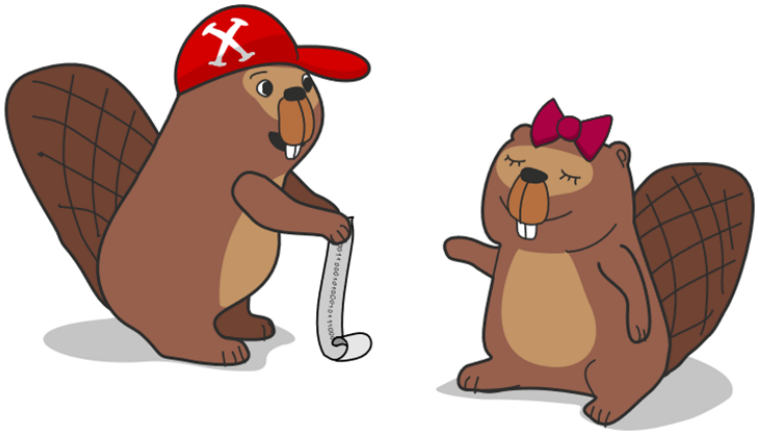
Aqui, entra o conceito de "hashing". Um valor hash é um valor calculado através de uma função hash a partir de propriedades de um item. No caso deste exercício, o título de um livro é transformado em dois dígitos que representam a linha e coluna de um cubículo na estante. Claro que diferentes livros podem acabar por ter o mesmo valor, tal como acontece com os livros "Tree Bark Gourmet Guide" e "Tasty Trees to Gnaw On". Há diferentes formas de enfrentar conflitos como este. Uma delas é simplesmente colocar vários itens no mesmo local, como num cubículo de uma estante. Se tal não for possível, o próximo espaço vazio é escolhido ou um espaço vazio n espaços afastado para ter uma distribuição mais equilibrada. Ao procurar por esse item, apenas o mesmo número de espaços são verificados até que se encontre um espaço vazio. Para eventualmente preencher todos os espaços, os vários valores de n são escolhidos de forma a que o único divisor que partilhem seja 1, ou seja, para que sejam primos entre si (como no exemplo abaixo).


0 1 2 3 4 5 6 7 8 9
0
1
2
3
4
5
6
7
8
9

12 – Representação Compacta

O castor Xavier quer representar algumas letras com dígitos binários 0 e 1. Ele repara que as letras T e E são as mais frequentes. Assim, ele decide atribuir-lhes uma representação mais curta e codificar as letras T, E, A, K, C e R como se segue:

Letra	T	E	A	K	C	R
Código	1	00	0010	0110	1010	1110



O Xavier enviou a seguinte mensagem codificada à Ivone:

1 0 0 1 0 0 1 1 0 0 0 1 0 1 0 0 0 1 0 1 1 1 0 0 0

A Ivone já descobriu que a mensagem termina com a letra E.

Pergunta

Em letras (sem espaços a separar), qual é a mensagem completa que o Xavier escreveu?

Resposta

Escreve a tua resposta (uma mensagem).

12 – Representação Compacta (Resolução)

Solução

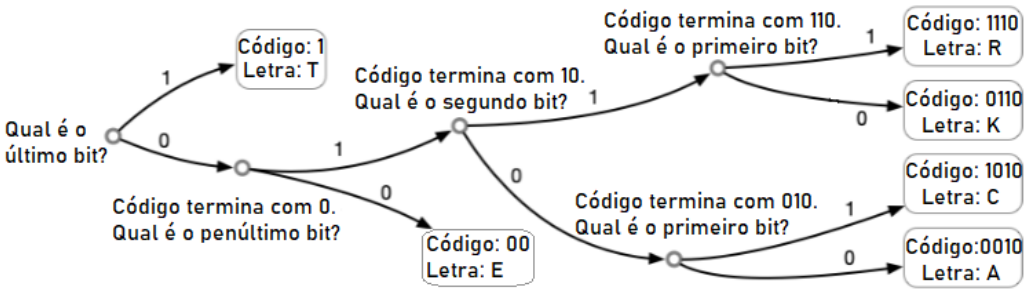
A resposta correta é: TAKECARE

Resolução

Aqui está está a correspondência entre caracteres e a sua representação binária na mensagem do Xavier:

Letra	T	E	A	K	C	R
Código	1	00	0010	0110	1010	1110

Para reconstruir a mensagem, tem de se encontrar uma forma de segmentar toda a mensagem numa sequência de palavras codificadas. Se começarmos pela esquerda, não é tão fácil fazê-lo: vamos deparar-nos com possíveis ambiguidades. Experimentemos: conseguimos identificar com alguma facilidade que a primeira letra é T e corresponde a "1", mas a segunda letra é um problema: tanto pode ser E, representado por "00"ou A, representado por "0010". Nesta fase, não conseguimos ter a certeza. No entando, a mensagem é inequívoca: se fizermos a escolha errada na segunda posição e escolhermos E, vamos ficar sem hipóteses mais à frente e perceber que a única possibilidade seria A. No nosso caso, conseguimos perceber que se começarmos pelo fim nunca teremos de tentar adivinhar ao descodificar uma letra. Isto acontece porque o código é de sufixo livre em inglês, "suffix-free"): não há nenhuma palavra codificada que termine numa sequência de uns e zeros que seria em si mesma uma outra palavra codificada. Assim, podemos facilmente reconstruir o texto inequivocamente ao ler o código binário da direita para a esquerda. Quando o código de uma letra é encontrado podemos trocar o código pela letra. O diagrama abaixo ilustra como a mensagem binária pode ser lida da direita para a esquerda, gerando letras inequivocamente:



Se quisessemos descodificar esta mensagem inequivocamente em todos os passos da esquerda para a direita, precisaríamos de um código de prefixo livre (em inglês, "prefix-free"), ou seja, um código em que nenhuma palavra codificada comece com uma sequência de zeros e uns que sejam por si só código para outra palavra. O código do Xavier não é de prefixo livre, pois o código 0010 para a letra A começa com 00, que é, em si mesmo, código para a letra E.

Isto é Pensamento Computacional!

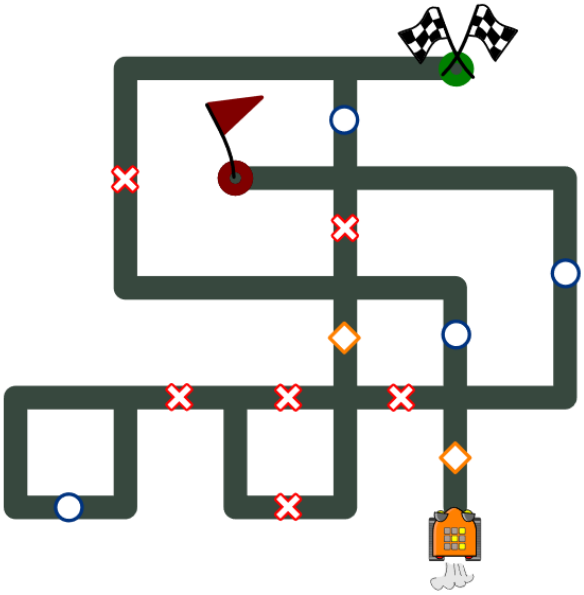
Todos os objetos com que um computador trabalha devem ser descritos como sequências de bits. Isto aplica-se também a textos. Espera-se que o objeto original possa ser reconstruído a partir da sua representação binária, mas isto só é possível se nunca acontecer dois ou mais objetos diferentes terem a mesma representação binária. Cientistas de computadores são solicitados a desenvolver sistemas de código que possam eficientemente recontruir o objeto original (por exemplo um texto) a partir da sua representação binária.

Se quisermos comprimir um texto (para obter uma representação binária do texto que seja o mais curta possível), então uma boa estratégia será atribuir códigos binários mais curtos para as letras mais frequentes e utilizar códigos mais compridos para as letras raras. Tem que se ter cuidado neste caso para escolher códigos que garantam uma descodificação (reconstrução do texto original) inequívoca eficiente. Muito boas escolhas para este propósito são códigos de prefixos e sufixos livres, cujos princípios estão descritos na explicação da resposta acima.

13 – Robô Leitor de Símbolos

Um robô começa na posição representada e move-se ao longo das linhas. Existem três símbolos ◇, ○ e ✕ nas linhas que decidem a direção que o robô deve tomar na próxima interseção. O robô não deve chegar ao símbolo ⚑. Cada símbolo tem um significado diferente e pode significar:

- **Virar à esquerda** na próxima interseção;
- **Virar à direita** na próxima interseção;
- **Continuar em frente** na próxima interseção.



Infelizmente, não sabemos que símbolo significa o quê. O significado do símbolo permanece o mesmo, independentemente da direção em que o robô se está a mover. As setas na figura indicam como o robô viraria, vindo de qualquer uma das direções, se o símbolo do triângulo significasse virar à esquerda na próxima intersecção.

Pergunta

Ajuda o robô a chegar à meta  *atribuindo os significados certos aos símbolos.*

Respostas Possíveis

(A) ✕ = virar à direita; ○ = continuar em frente; ◇ = virar à esquerda.

(B) ✕ = virar à esquerda; ○ = continuar em frente; ◇ = virar à direita.

(C) ✕ = virar à direita; ○ = virar à esquerda; ◇ = continuar em frente.

(D) ✕ = continuar em frente; ○ = virar à direita; ◇ = virar à esquerda.

(E) ✕ = virar à esquerda; ○ = virar à direita; ◇ = continuar em frente.

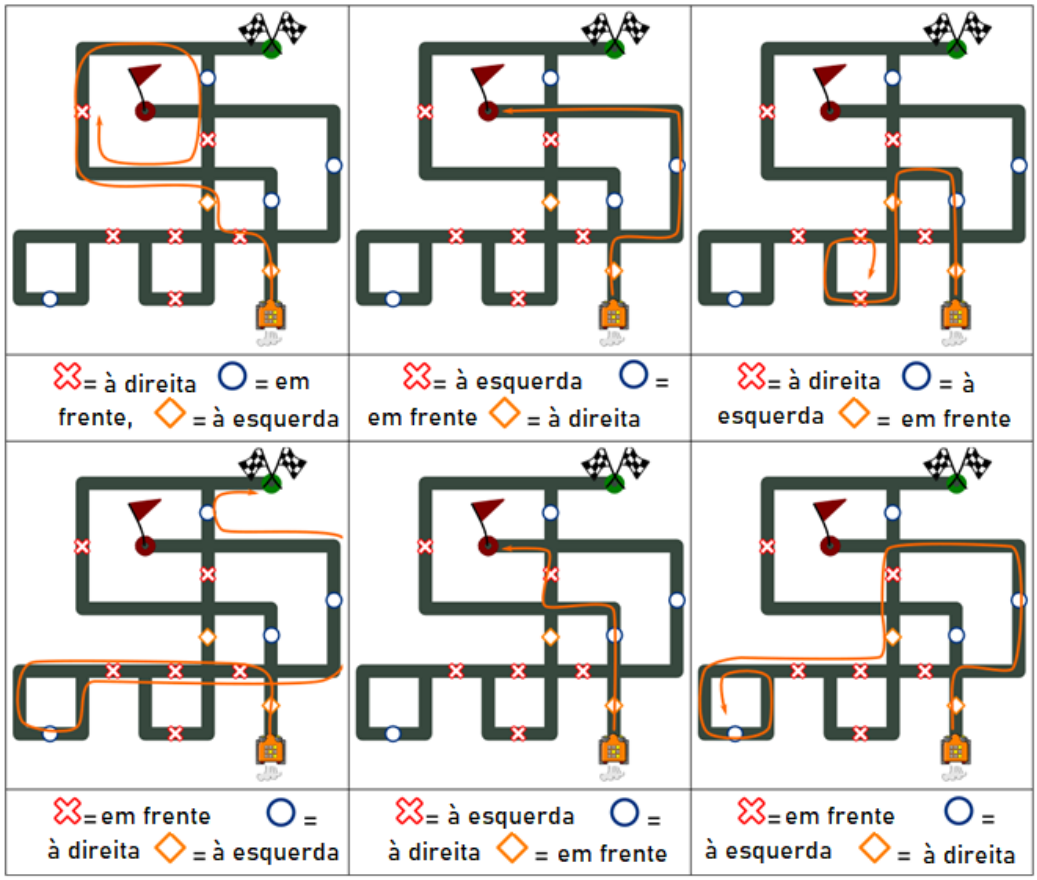
(F) ✕ = continuar em frente; ○ = virar à esquerda; ◇ = virar à direita.

13 – Robô Leitor de Símbolos (Resolução)

Solução

(D)

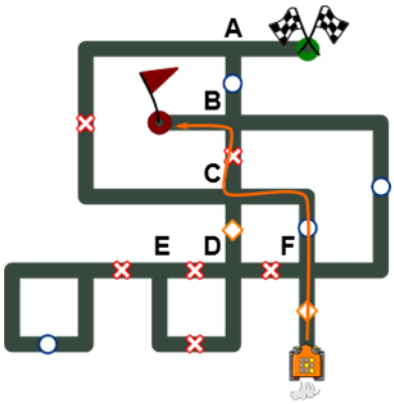
Resolução



Outra estratégia é traçar o caminho que o robô poderia tomar e atribuir significado aos símbolos à medida que acompanhamos o caminho. Se o robô chegar à bandeira vermelha ou se entrar num ciclo ou os significados não puderem ser aplicados aos símbolos de forma consistente, voltamos a traçar o passo anterior (neste caso interseção) e tentamos um novo caminho. Quando o robô chega à interceção F, tem três opções: virar à esquerda, continuar em frente ou virar à direita.

Tentativa 1

Vamos assumir que ele segue em frente. Isto implicaria que ◇ significaria "Continuar em frente". O robô continuaria até à interceção C. Uma vez que "Continuar em frente" já está atribuído a ◇ e que o robô acabou de ver ○, ele poderia virar à esquerda ou virar à direita.

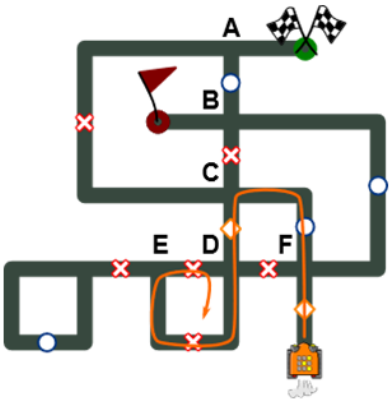


Tentativa 1.1

Vamos assumir que o robô viraria à direita na interseção C, implicando que ○ significa "Virar à direita" e que, portanto, ⊠ significaria "Virar à esquerda". O robô viraria agora à esquerda na interceção B, chegando a ⚑. Isto não é o que queremos, portanto regressamos à interseção anterior C.

Tentativa 1.2

Vamos assumir que o robô viraria à esquerda na interseção C, implicando que ○ significaria "Virar à esquerda" e, portanto, ⊠ significaria "Virar à direita". Na interseção D, o robô seguiria em frente, pois ◇ significa "Continuar em frente" e chegaria à interseção E. Viraria à direita em E, pois viu ⊠. Na interseção D, viraria à direita outra vez e então entraria num ciclo. Esta não é a solução que queremos, portanto voltaríamos para a interseção C, uma vez que fizemos escolhas que fixaram significados a símbolos para verificar se seria possível algum outro caminho. Dado que não há outra forma de interpretar, concluiríamos que as nossas atribuições anteriores estariam também incorretas e voltaríamos à interceção F para tentar um caminho alternativo.

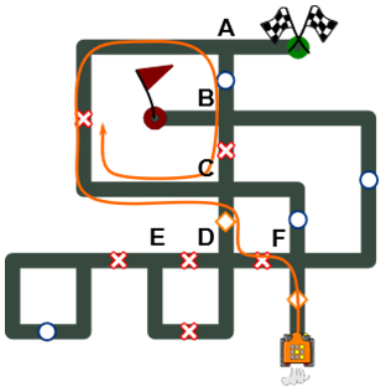


Tentativa 2

Assumamos que o robô vira à esquerda na interseção F, implicando que ◇ significa "Virar à esquerda".

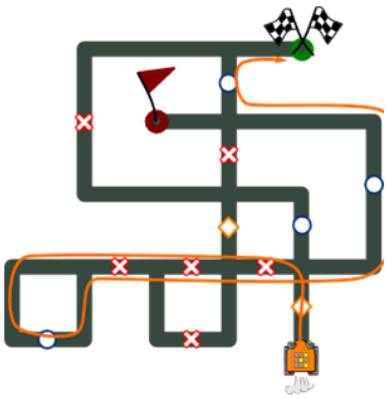
Tentativa 2.1

Quando chega à interseção D, existem novamente 3 possibilidades. Não pode virar à esquerda porque isso significaria que ⊠ e ◇ teriam o mesmo significado. O robô poderia virar à direita em D, o que implicaria que ⊠ significa "Virar à direita" e, então, ○ significaria "Continuar em frente". A partir de D o robô continua até C. Em C vira à esquerda pois viu ◇. Na interseção A viraria à direita porque viu ⊠. Na interseção B continuaria em frente até à interseção C pois teria visto ○. Na interseção C viraria à direita novamente depois de passar por outro símbolo ⊠, entrando num ciclo. Isto não é o que pretendemos, portanto voltaríamos até à interseção D, onde decidimos virar à direita, e tentaríamos outro caminho.



Tentativa 2.2

Assumamos agora que o robô continua em frente (lembre-se que não pode virar à esquerda), implicando que ⊠ significaria "Continuar em frente" e, portanto, ○ significaria "Virar à direita". Agora podemos seguir o caminho e perceber que o robô chegaria a ⚑. Assim, a interpretação correta dos sinais seria: ◇ significa "Virar à esquerda", ⊠ "Continuar em frente" e ○ "Virar à direita".



Isto é Pensamento Computacional!

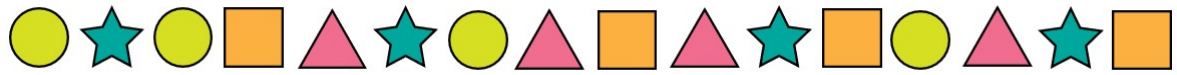
A primeira estratégia proposta na explicação é designada por força bruta. Isto significa considerar e verificar todas as possibilidades até encontrar o resultado desejado. Às vezes pode haver muitas possibilidades e considerá-las todas poderia levar demasiado tempo e, portanto, seria necessário encontrar outras estratégias. A segunda estratégia chama-se retraçamento (em inglês, "backtracking"). Nesta técnica constroem-se, de modo incremental, candidatos para as soluções e abandona-se um candidato assim que se determina que o candidato não constitui uma solução válida. A vantagem do retraçamento face à força bruta é que não é necessário reconsiderar as novas soluções candidatas desde o início, uma vez que apenas é necessário regressar ao passo onde foi feita a última escolha e continuar a partir daí.

Para este exercício a força bruta pode funcionar melhor, visto que temos poucas variáveis (símbolos) e que há poucos caminhos que o robô pode tomar. No entanto, no geral, para problemas maiores e mais complexos, quando o número de variáveis aumenta e há mais caminhos a explorar, o retraçamento (ou *backtracking*) constitui uma solução melhor e mais elegante.

Puzzles como o Sudoku podem ser elegantemente resolvidos utilizando retraçamento.

14 – Sequência Mais Longa

Aqui está uma sequência de comprimento 16, utilizando quatro formas diferentes:



Podes mudar exatamente três das formas na sequência, transformando-as em qualquer outra das formas.

Pergunta

Qual é o maior possível comprimento de uma cadeia (subsequência contígua, ou seja, sem cortes) composta por formas iguais?

Respostas Possíveis

(A) 4

(B) 5

(C) 6

(D) 7

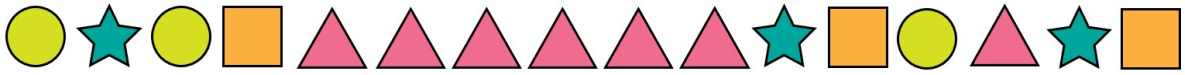
14 – Sequência Mais Longa (Resolução)

Solução

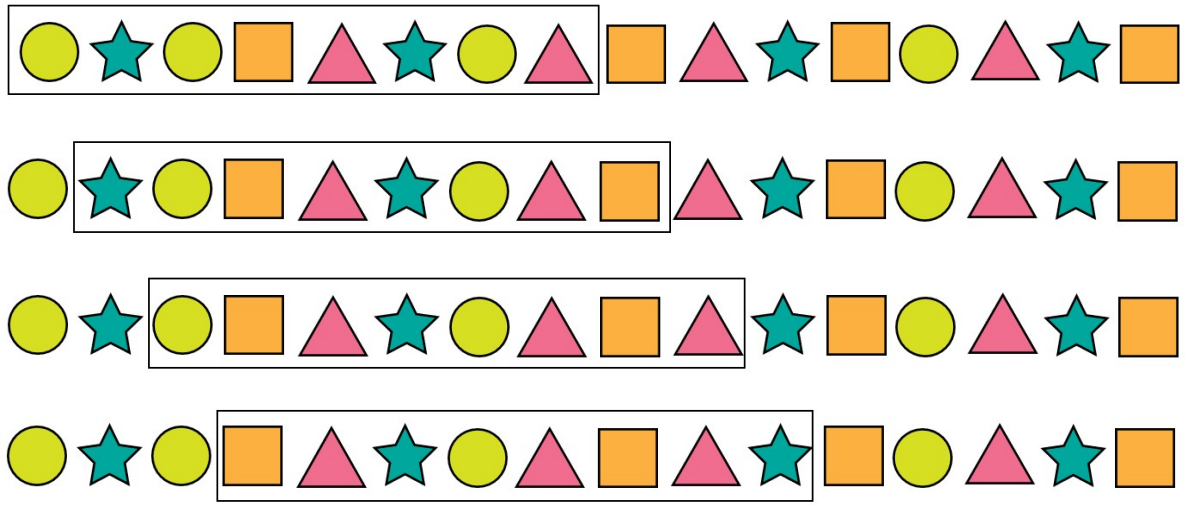
(C)

Resolução

A resposta correta é (C), 6. Para o demonstrar, precisamos de provar duas coisas: (1) que uma cadeia contínua de 6 é possível e (2) que uma cadeia contínua de comprimento superior a 6 não é possível. A primeira parte é fácil de provar. Aqui está como uma cadeia contínua de 6 triângulos pode ser feita:



Para provar que uma cadeia contínua de comprimento superior a 6 não é possível, consideremos uma qualquer cadeia de comprimento 7. Uma vez que apenas é permitido alterar três formas, qualquer cadeia de 7 na sequência original deve ter já quatro formas idênticas. Há dez cadeias de comprimento 7 na cadeia original, algumas das quais estão representadas abaixo. Em nenhuma delas se encontram quatro formas idênticas. É fácil verificar o mesmo para as restantes 6 cadeias que não foram evidenciadas.



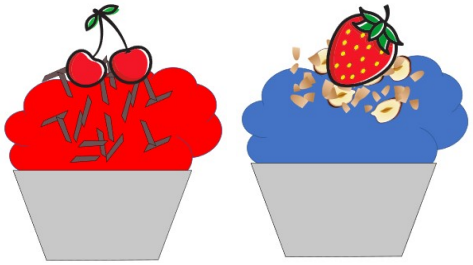
Assim, mostrámos que o comprimento da maior cadeia contínua de formas idênticas possível é 6.

Isto é Pensamento Computacional!

Este exercício é sobre o sistema de representação numérico binário. Fisicamente, os computadores apenas guardam dois tipos de valores: zeros e uns. Isto é extremamente conveniente em termos de design de hardware. Portas lógicas e circuitos integrados são muito mais fáceis de desenhar se elas apenas precisarem de processar dois valores (como um valor alto e baixo de tensão). Num sistema de representação binário, cada dígito representa uma potência de 2, e os zeros e uns indicam se cada potência de 2 deve ser adicionada ou não. Por exemplo, o número binário $100101 = 1\times32+0\times16+0\times8+1\times4+0\times2+1\times1 = 37$ é representado no sistema decimal por 37.

15 – Queques

A pastelaria do Bebras produz queques para os castores trabalhadores da cidade. Cada queque é decorado com três doces camadas. Primeiro, cada queque leva uma camada de cobertura, depois uma camada com pedaços e por fim uma camada de fruta. O primeiro exemplo abaixo tem uma camada de cobertura vermelha, uma camada de pedaços de chocolate e uma camada de cerejas. O segundo exemplo tem cobertura azul, pedaços de avelã e morangos.



Na linha de montagem, cada camada é alterada de um queque para o próximo tal como se segue:

- a camada da cobertura muda de acordo com o seguinte padrão: verde → branco → vermelho → azul → repete novamente começando com o verde
- a camada dos pedaços muda de acordo com o seguinte padrão: granulado → pedaços de chocolate → pedaços de avelã → repete novamente começando com o granulado
- a camada da fruta muda de acordo com o seguinte padrão: cereja → kiwi → morango → laranja → mirtilo → repete novamente começando com a cereja

O castor Benjamim pregou uma partida na pastelaria. Ele mudou o padrão de duas das camadas:

- O Benjamim alterou o padrão da fruta para que de cada vez passe à frente os próximos dois frutos no padrão. Por exemplo, se um pedaço de laranja for colocado no queque, então o próximo queque teria um kiwi no topo.
- O Benjamin inverteu o padrão dos pedaços.

Pergunta

Se o primeiro queque tiver cobertura verde, granulado e uma cereja no topo, qual será o aspecto do sexto queque?

(A)

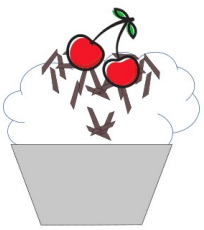
vermelho
chocolate
cereja

(B)

branco
avelã
kiwi

(C)

azul
avelã
morango



(D)

branco
chocolate
cereja

(E)

azul
chocolate
mirtilo

15 – Queques (Resolução)

Solução

(D)

Resolução

A resposta correta é (D), branco, chocolate, cereja.

O primeiro queque tem cobertura verde, granulado e uma cereja no topo. Temos de encontrar o sexto queque. Começando pela camada da cobertura, a ordem é: verde, branco, vermelho, azul, verde, branco - assim, o sexto queque terá cobertura branca. Passando para a camada dos pedaços, sabemos que o Benjamim reverteu a ordem aqui. O primeiro queque tem granulado. Assim, a ordem é: granulado, avelãs, chocolate, granulado, avelã, chocolate - assim, o sexto queque tem pedaços de chocolate. Passemos então para a camada da fruta. Sabemos que o benjamin mudou a fruta de modo a que passe à frente os dois pedaços de fruta seguintes depois de ser colocado um. A ordem, então, é: cereja, laranja, kiwi, mirtilo, morango, cereja - logo, o sexto queque tem uma camada com cereja.

Isto é Pensamento Computacional!

Este exercício ilustra os conceitos de pensamento computacional de algoritmos e reconhecimento de padrões, bem como o conceito de programação de computadores de uma lista ligada. Reconhecimento de padrões é o conceito de encontrar padrões no problema que vão permitir que sejam reutilizados na solução, quer sob a forma de ciclos na solução, quer reutilizando partes da solução de problemas anteriormente resolvidos.

Neste exercício, a sequência de opções para cada camada forma um padrão, mas também há um padrão de forma a que a aplicação de cada camada (cobertura, pedaços, fruta) siga o mesmo algoritmo fundamental. Um algoritmo é uma lista de instruções. Seguir uma lista de instruções é um conceito muito importante em ciência de computadores. É assim que um computador funciona - dizemos-lhe o que fazer e ele segue esses passos. Para algumas linguagens de programação, a ordem das instruções é também muito importante. Ao alterar a ordem, podemos alterar as saídas (em inglês, "outputs") do programa. A sequência de ingredientes neste exercício é muito importante para cada camada. Quando escrevemos programas de computador, precisamos de armazenar dados dentro do nosso programa. Utilizamos estruturas de dados para armazená-los de forma eficiente e organizada. Há muitos tipos de estruturas de dados, tal como a estrutura de dados mais relevante neste exercício: uma lista ligada. Uma lista ligada é uma coleção linear de elementos de dados onde a ordem não é dada pela sua posição física em memória. Em vez disso, cada elemento indica o elemento seguinte e isto permite-nos aceder a um dado elemento na lista ao percorrer a lista desde o início. Listas ligadas podem aumentar de tamanho dinamicamente e, ao contrário do que acontece em arranjos, é fácil inserir e eliminar em qualquer posição dentro de uma lista ligada.

Por exemplo, para eliminar um elemente em qualquer posição na lista, apenas precisamos de mudar para onde o elemento anterior da lista o indica.