Castor Informático

O Desafio Internacional de Pensamento Computacional

Edição 2021

Categoria: **Séniores** (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.

Conteúdo

Página

Preâmbulo 2
Organização 2
Estrutura da Prova 3
Sobre os Problemas 3
1 – Teias de Aranha 4
Resolução 5
2 – Balcões de Atendimento 6
Resolução 7
3 – Ordenando Sete Estudantes 9
Resolução 10
4 – Número Secreto 12
Resolução 13
5 – Peixes em Linha 14
Resolução 15
6 – Biblioteca 16
Resolução 17
7 – Encontro de Amigos 19
Resolução 20
8 – Quadro Roubado 24
Resolução 25
9 – Sequência Mais Longa 26
Resolução 27
10 – Queques 28
Resolução 29
11 – Turing 30
Resolução 32
12 – Representação Compacta 34
Resolução 35
13 – Robô Leitor de Símbolos 37
Resolução 38
14 – Segredo do Diário 41
Resolução 42
15 – Tabela de Verdade 43
Resolução 45

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.

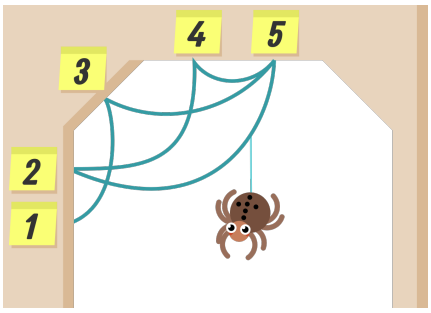
Sobre os Problemas

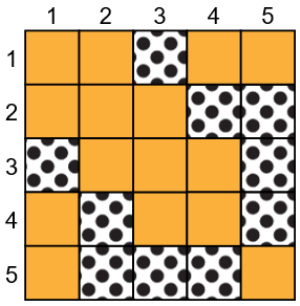
Os nomes dos autores dos problemas serão discriminados na versão final a divulgar no sítio oficial do Bebras - Castor Informático. Os problemas foram escolhidos, traduzidos e adaptados pela organização portuguesa. Para a edição portuguesa deste ano foram usados problemas com autores originários dos seguintes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

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

1 – 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:

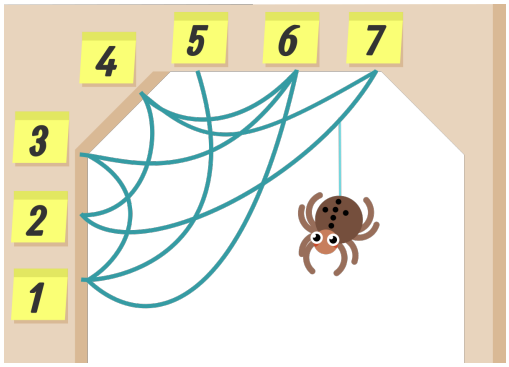




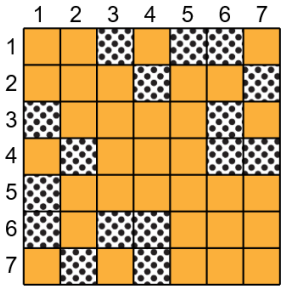
Por cada fio da teia de aranha, se ele se fixa nos números X e Y, ela coloca dois quadrados de tecido às bolinhas na sua grelha. Um pedaço quadrado de tecido às bolinhas é colocado onde a linha X se cruza com a coluna Y. Outro quadrado de tecido às bolinhas é colocado onde a coluna Y se cruza com a linha 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

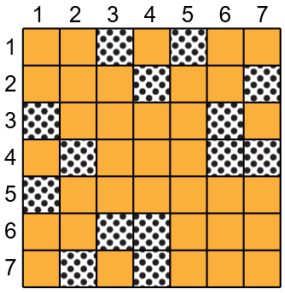
A Vanda viu agora a teia de aranha da imagem abaixo.
Como fica a manta que esta nova teia de aranha inspira?



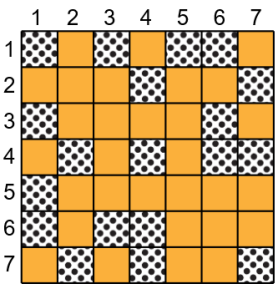
Respostas Possíveis



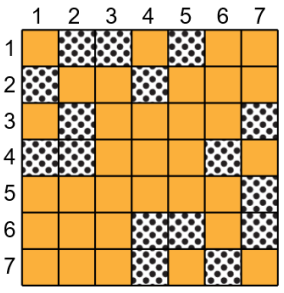
(A)



(B)



(C)



(D)

Dificuldade: **fácil** | Origem: 

1 – 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!

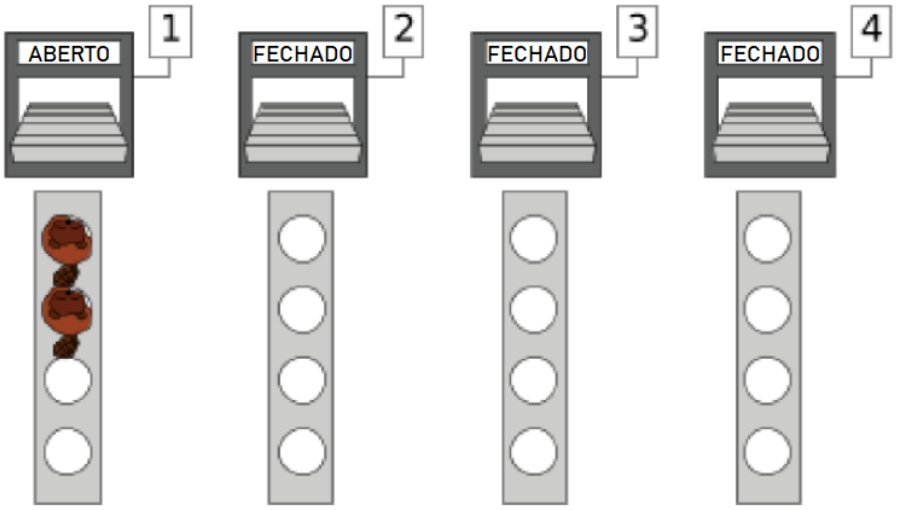
A teia de aranha pode ser considerada um grafo, um conceito frequentemente utilizado em ciência de computadores.

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?

2 – 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

2 – Balcões de Atendimento (Resolução)

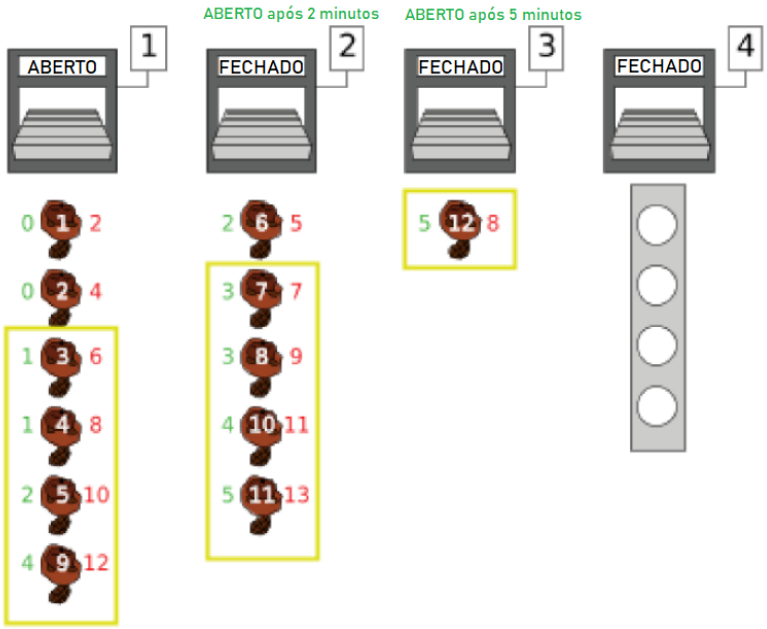
Solução

(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.

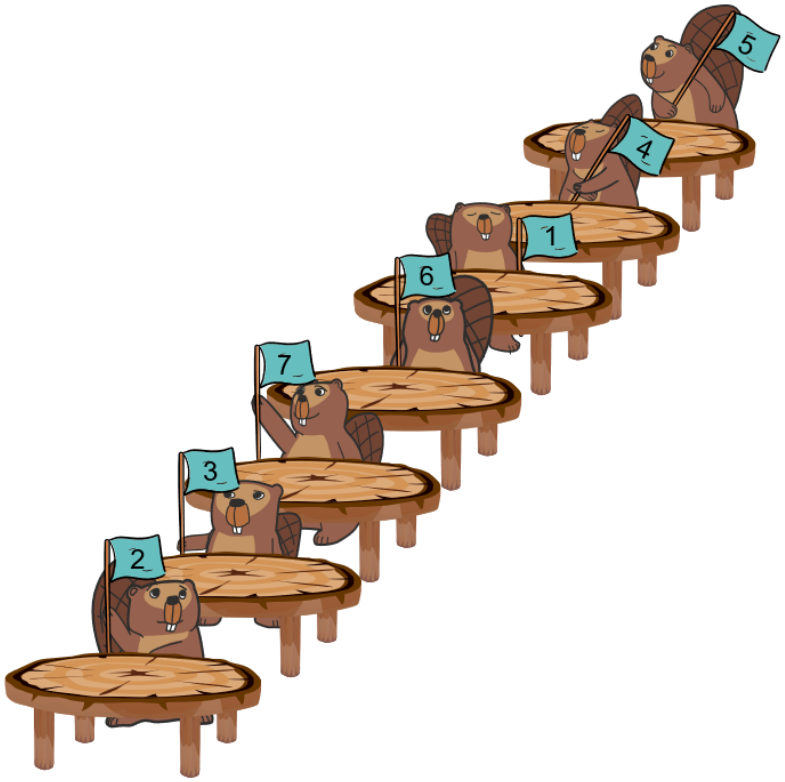
A resposta (A) corresponde ao tempo que o terceiro cliente termina na terceira fila.

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.

3 – 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

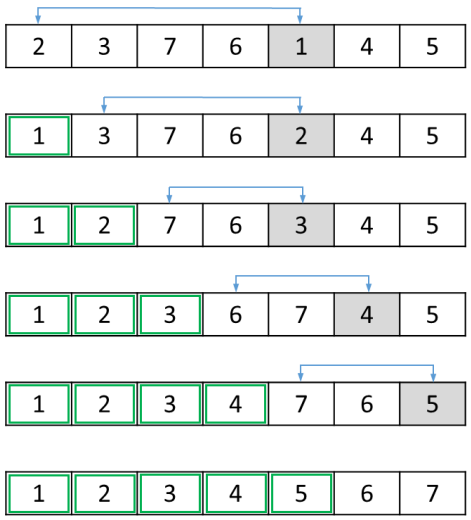
3 – Ordenando Sete Estudantes (Resolução)

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.

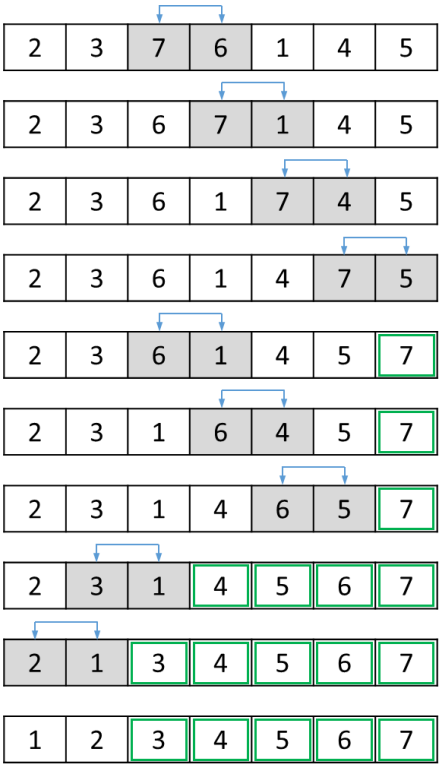
Com a primeira ordem (a ordem que podemos ver na imagem), selecionamos o número mais baixo (o número 1) e trocamo-lo com o número do primeiro lugar da lista (o número 2). No passo seguinte, a começar do segundo lugar da lista, selecionamos o número mais baixo (ou seja, o número 2) e trocamo-lo com o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

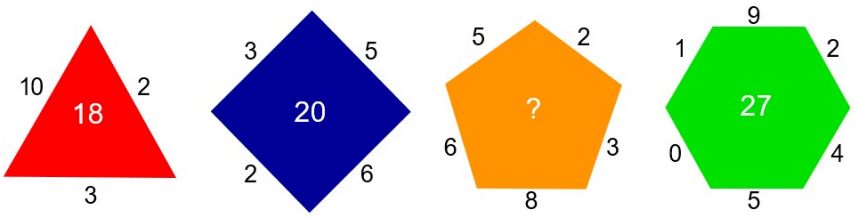
O algoritmo de ordenação por seleção melhora o algoritmo de ordenação por flutuação fazendo uma única troca por cada passagem pela lista. Para fazer isto, uma ordenação por seleção procura o número mais baixo enquanto faz uma passagem e, depois de completar a passagem, coloca-o na localização correta. 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:



4 – Número Secreto

No mundo do Bebras, o pagamento é feito com moedas especiais. Cada moeda tem o seu valor escrito no centro.



Pergunta

Qual é o valor que falta, indicado pelo ponto de interrogação?

Resposta

Escreve a tua resposta (um número inteiro).

4 – Número Secreto (Resolução)

Solução

29

Resolução

A resposta correta é 29. O valor de cada moeda é igual à soma dos números fora da figura adicionando o número de lados da figura. Neste exercício há quatro moedas diferentes:

- Triângulo vermelho: $10 + 2 + 3 + 3 = 18$
- Quadrado azul: $3 + 5 + 6 + 2 + 4 = 20$
- Pentágono laranja: $5 + 2 + 3 + 8 + 6 + 5 = 29$
- Hexágono verde: $1 + 9 + 2 + 4 + 5 + 0 + 6 = 27$

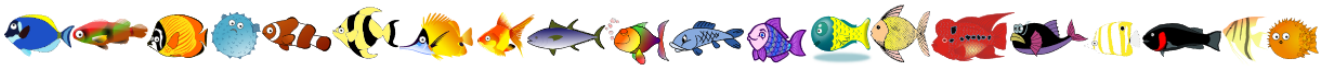
Isto é Pensamento Computacional!

O valor de cada moeda é formado por duas "partes". É importante descobrir que o valor da moeda (número branco no centro) é obtido a partir da soma dos números em cada lado da figura adicionando o número de lados da figura. Todos os cálculos seguem o mesmo padrão.

Padrões e o seu reconhecimento são utilizados em engenharia, computação, matemática e estão relacionados com objetos físicos ou abstratos. Para reconhecer um padrão, uma análise (por exemplo, por observação) tem de ser conduzida. Os padrões podem ser obtidos a partir de processos de segmentação, extração ou características e descrição onde cada objeto é representado por uma coleção de descritores. O propósito da análise é extrair dados que permitam reconhecer propriedades, marcar regularidades entre conjuntos de objetos. Em ciência de computadores, um padrão de projeto de software é uma solução-modelo comum que ajuda a acelerar o desenvolvimento de programas de computador.

5 – 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) 

5 – 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.

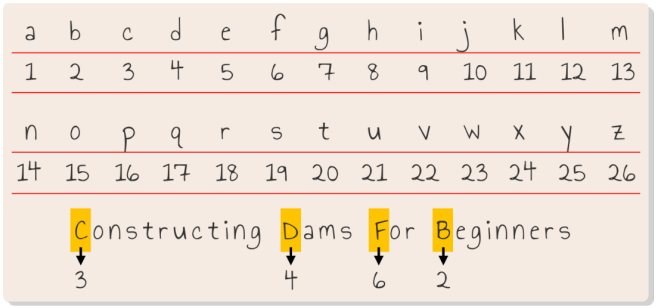
Isto é Pensamento Computacional!

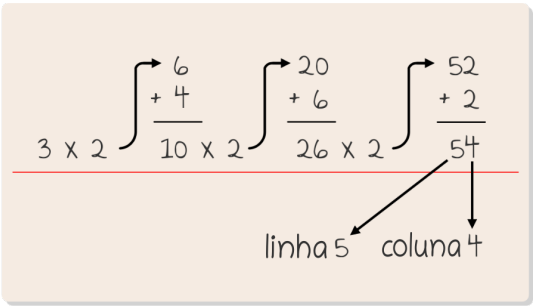
Qunado um programador de computadores trabalha com dados, ele precisa de determinar como representar esses dados. Dados relacionados são normalmente armazendos juntos numa coleção. Neste caso, uma segunda decisão importante é determinar como organizar a coleção na memória. Diferentes tipos de dados podem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

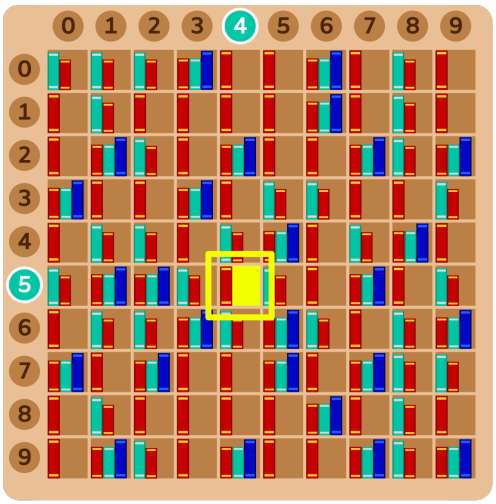
Tipos de dados são normalmente associados a operações comuns aplicadas nos dados. A operação chave neste exercício é a seleção de peixes entre duas dadas posições. Esta é das operações de sequenciamento mais importantes no geral. Quando a sequência é uma lista de letras ou outros caracteres, é tipicamente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

6 – 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

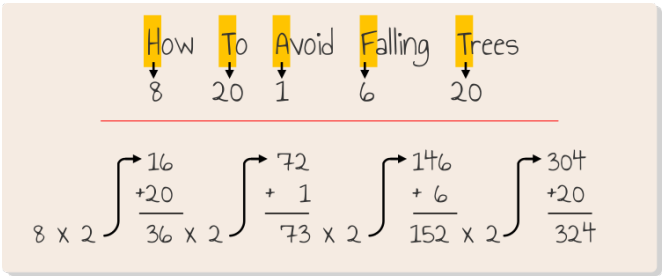
6 – Biblioteca (Resolução)

Solução

(B)

Resolução

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



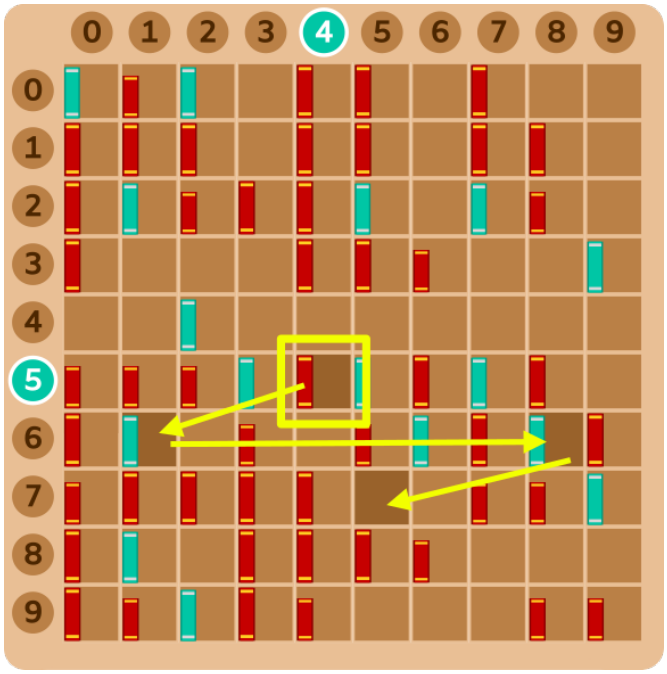
As outras respostas estão erradas. O livro (A) "Chewing on Trees Made Easy" está na linha 7 e coluna 9 (os números são 3, 15, 20, 13 e 5; os resultados intermédios são 21, 62, 137 e 279). O livro (C) "Tasty Trees to Gnaw On" está na linha 8 e coluna 9 (os números são 20, 20, 20, 7 e 15; os resultados 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.

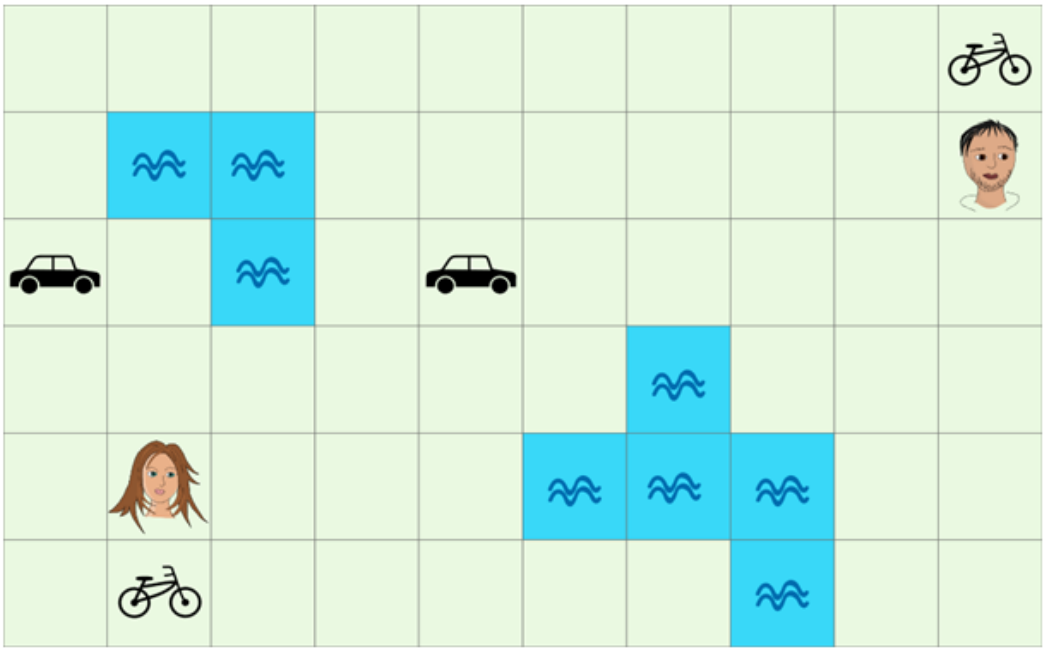
O problema não existe só para as bibliotecas. Grandes farmácias também precisam de ter um sistema para armazenar e encontrar os medicamentos. Nos últimos anos, cada vez mais farmácias estão a adotar sistemas de armazenamento automatizados. Para eles, uma ordenação sistemática, por exemplo, por tipo de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

7 – 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).

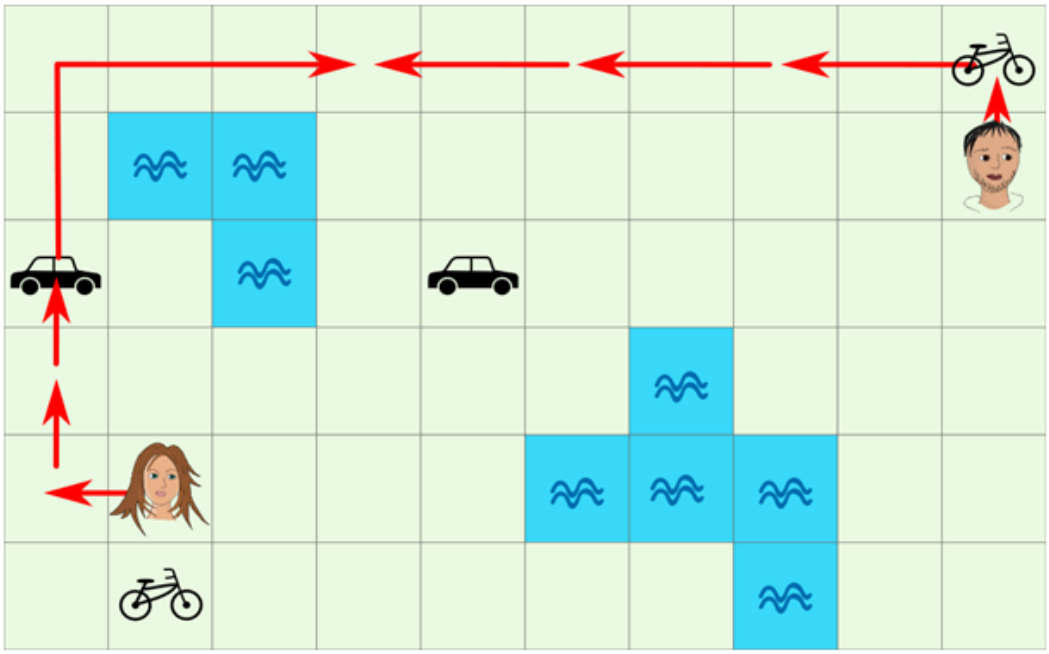
7 – Encontro de Amigos (Resolução)

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:

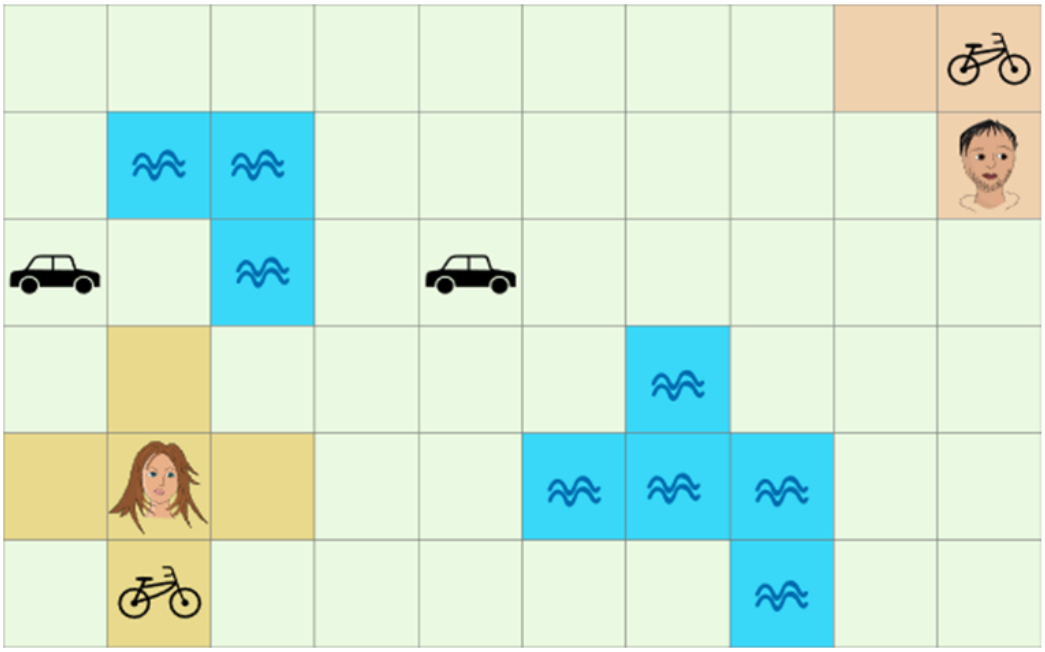
- Embora em 3 minutos seja possível chegar ao carro mais à esquerda, não há tempo suficiente para o conduzir a lado algum. E essa posição não pode ser atingida em três minutos pela outra pessoa. Portanto os carros não têm utilidade e poderíamos até removê-los do mapa.
- Os dois amigos estão a mais de 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!

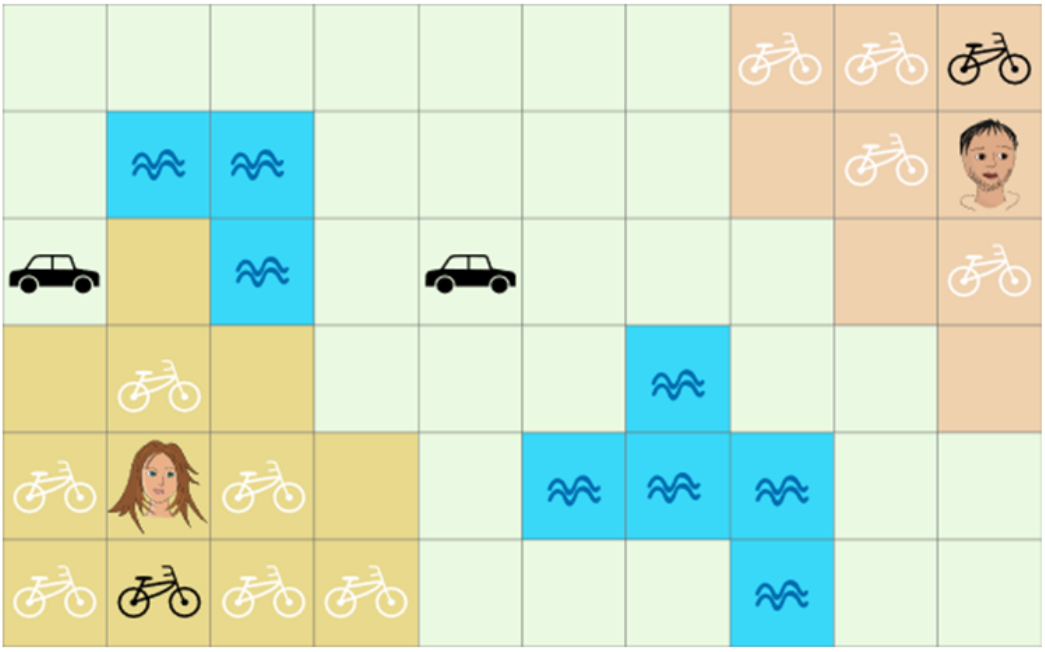
Como é que se procedeu para resolver este exercício? Encontrou-se uma rota curta por acidente e esperou que nenhuma outra mais curta pudesse ser encontrada, ou tentou dezenas de diferentes possibilidades,

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:

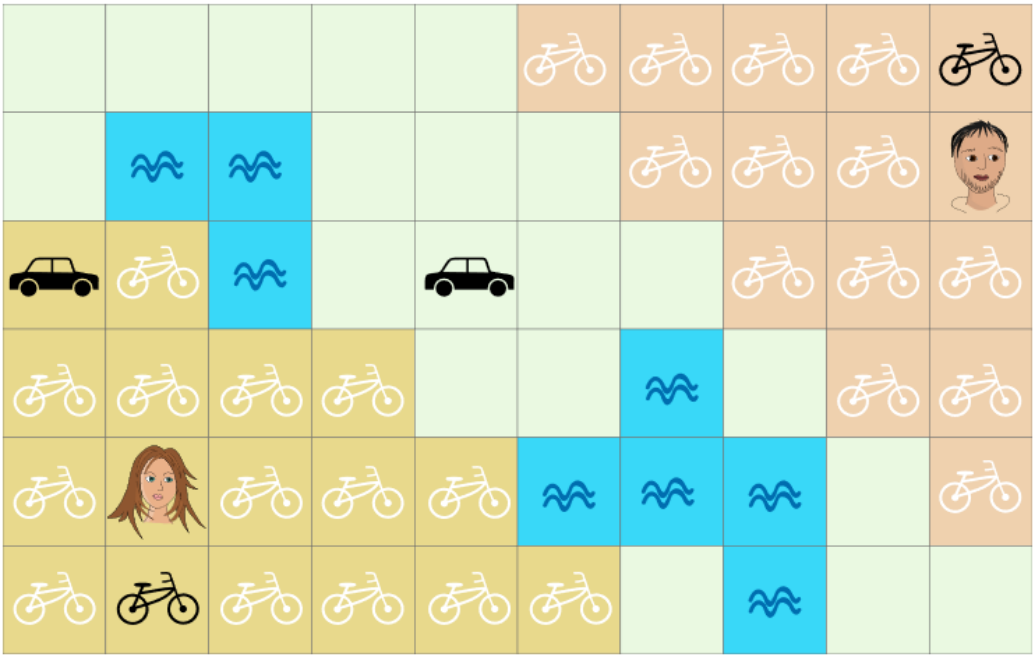
1. Marcar todos os quadrados no mapa que podem ser alcançados por cada amigo num minuto.



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.

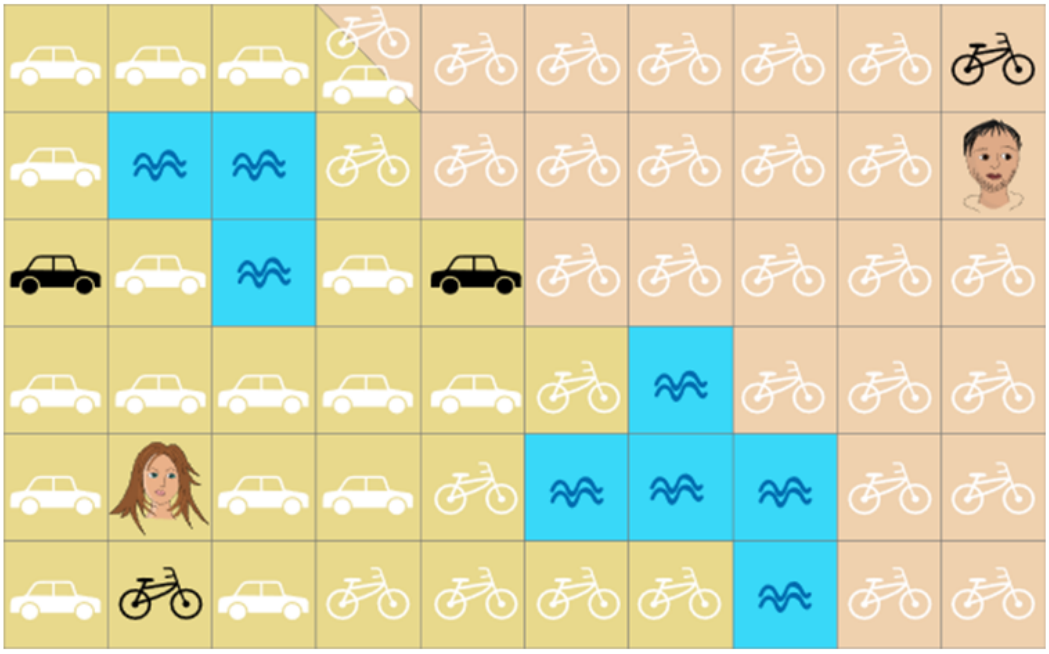


3. Marcar todos os quadrados que podem ser alcançados em um minuto a partir dos quadrados marcados em 2.



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.

8 – Quadro Roubado

A TransArt é uma empresa de logística especializada no transporte de quadros. Os quadros são levados até uma loja para inspeção e depois os transportadores levam-nos até ao seu destino final. Todos os novos quadros que chegam são colocados no topo de uma pilha de quadros. Todos os transportadores que levam quadros para transportá-los até ao seu destino final levam um quadro do topo da pilha.

Por motivos de segurança, a TransArt toma nota de todos os quadros que entram e saem.

Quadros trazidos para a loja

Hora	Quadro
11:40	Castores na relva
12:15	Castor feliz
12:55	Sol e Lua
13:30	Floresta encantada
14:18	Carvalho e bétula
15:10	Romance pantanoso

Quadros levados da loja

Hora	Transportador
12:25	A
13:35	C
14:35	A
14:40	B
15:20	C
15:35	D

Um dia, a TransArt foi avisada de que o quadro "Sol e Lua"nunca chegou ao museu que era suposto recebê-lo. O transportador que o levou da loja deve ter roubado o quadro!

Pergunta

Quem roubou o quadro "Sol e Lua"?

Respostas Possíveis

(A) Transportador A

(B) Transportador B

(C) Transportador C

(D) Transportador D

8 – Quadro Roubado (Resolução)

Solução

(B)

Resolução

Há dois tipos importantes de eventos: alguém coloca um quadro na pilha/monte ou alguém retira um quadro da pilha/monte. Pelas tabelas do enunciado, podemos criar uma nova tabela que mostra os eventos e o estado resultante da pilha, organizado por horas.

Hora	Evento	Quadros no monte
11:40	Chegada do “Castores na relva”	“Castores na relva”
12:15	Chegada do “Castor feliz”	“Castor feliz” “Castores na relva”
12:25	**A** leva “Castor feliz”	“Castores na relva”
12:55	Chegada do “Sol e Lua”	“Sol e Lua” “Castores na relva”
13:30	Chegada do “Floresta encantada”	Enchanted Forest “Sol e Lua” “Castores na relva”
13:35	**C** leva “Floresta encantada”	“Sol e Lua” “Castores na relva”
14:18	Chegada do “Carvalho e bétula”	“Carvalho e bétula” “Sol e Lua” “Castores na relva”
14:35	**A** leva “Carvalho e bétula”	“Sol e Lua” “Castores na relva”
14:40	**B** leva “Sol e Lua”	“Castores na relva”

...e aqui podemos parar para pedir esclarecimentos ao transportador B.

Isto é Pensamento Computacional!

Há três conceitos informáticos que figuram neste exercício.

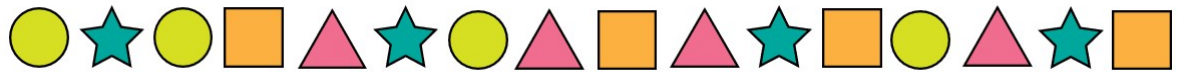
Um deles é o conceito de pilha. Uma pilha não é apenas uma pilha de quadros, mas também uma estrutura de dados, organizada de forma a que o último elemento colocado na pilha seja o primeiro a sair dela ("Último a Entrar, Primeiro a Sair"ou, no original em inglês, “Last In, First Out” ou LIFO).

O segundo é fusão de listas: para chegar à solução tivemos de pegar em duas listas ordenadas (eventos ordenados por hora) e fundi-las numa só lista ordenada - a que mostramos aqui na solução. Este passo é a base de alguns dos algoritmos mais rápidos para ordenar dados, o algoritmo de ordenação por mistura (em inglês, "merge sort").

Finalmente, podemos considerar toda a narrativa como a execução de um programa. O roubo do quadro é como um evento que faz com que um programa de computador pare (em inglês, "crash"). A isto chama-se uma exceção. Para encontrar a causa de uma exceção (o mau comportamento do transportador ou o mau comportamento das linhas num programa), precisamos de seguir a execução do programa até chegarmos ao ponto onde parou. Isto chama-se rastreamento. A seguir, um programador tentaria encontrar uma forma de tratar da exceção de modo a prevenir uma futura paragem do programa.

9 – 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

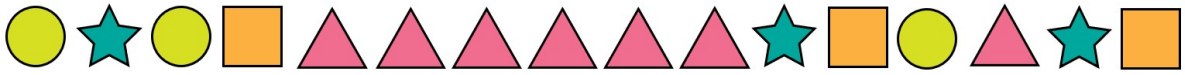
9 – Sequência Mais Longa (Resolução)

Solução

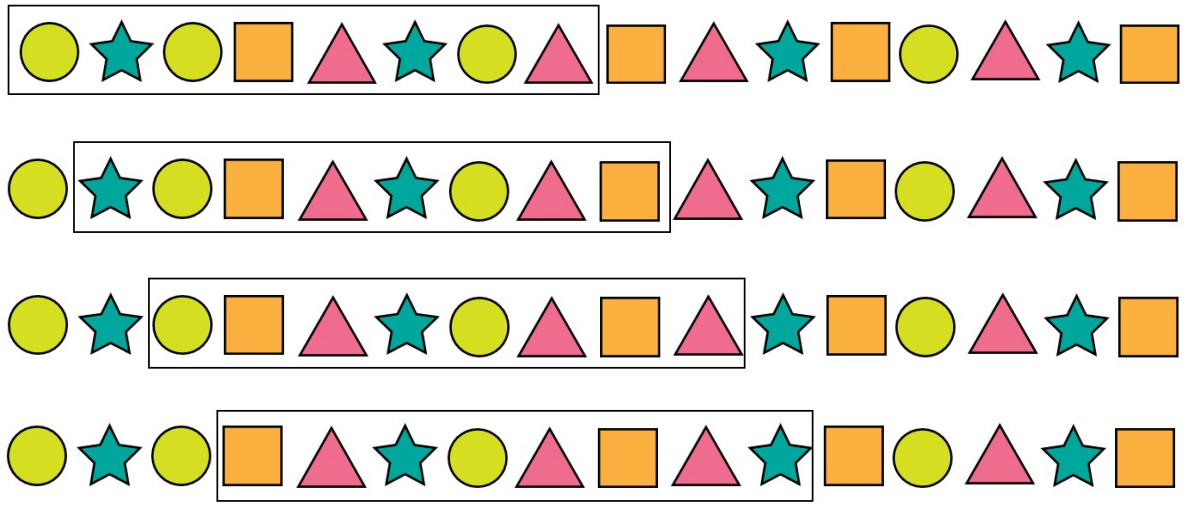
(C)

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.



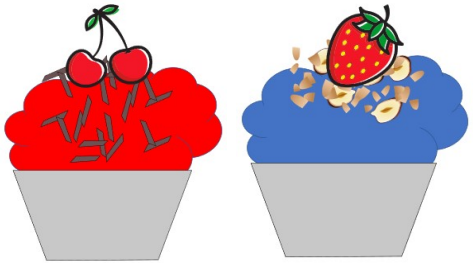
Assim, mostrámos que o comprimento da maior cadeia contínua de formas idênticas possível é 6.

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.

10 – 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:

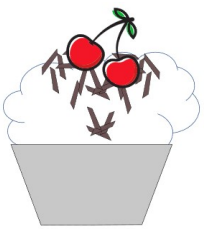
- a camada da cobertura muda de acordo com o seguinte padrão: verde → branco → vermelho → azul → repete novamente começando com o verde
- a camada dos pedaços muda de acordo com o seguinte padrão: granulado → pedaços de chocolate → pedaços de avelã → repete novamente começando com o granulado
- a camada 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)	(B)	(C)	(D)	(E)
				
vermelho chocolate cereja	branco avelã kiwi	azul avelã morango	branco chocolate cereja	azul chocolate mirtilo

10 – 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!

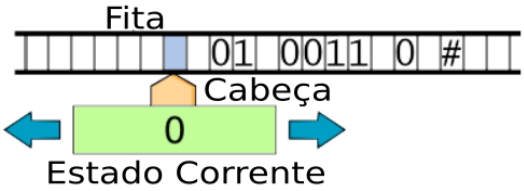
Este exercício ilustra os conceitos de pensamento computacional de algoritmos e reconhecimento de padrões, bem como o conceito de programação de computadores de uma lista ligada. Reconhecimento de padrões é o conceito de encontrar padrões no problema que vão permitir que sejam reutilizados na solução, 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.

11 – Turing

Uma máquina de Turing é um modelo computacional de um computador. Consiste num **estado** e numa **cabeça** de leitura/escrita que opera numa **fita** com símbolos. A cabeça pode mover-se para a esquerda (e) ou para a direita (d) um símbolo de cada vez. A máquina começa sempre no estado "0"(**estado inicial**).



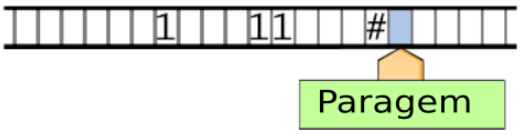
A nossa versão da máquina de Turing consegue correr programas e cada linha desse programa consiste em 5 elementos, representados no curto programa da imagem seguinte:

Estado Corrente	Símbolo Corrente	Novo Símbolo	Direção	Novo Estado
0	_	_	d	0
0	1	1	d	1
0	0	_	d	0
0	#	#	d	paragem
1	0	0	d	1
1	1	1	d	1
1	_	_	d	0
1	#	#	d	paragem

Quando o programa é executado, a primeira linha (a contar de cima) com o correspondente estado corrente e símbolo na corrente posição da cabeça é utilizada para determinar um **novo símbolo** para escrever por cima da fita na posição corrente, uma **direção** para mover a cabeça e um **novo estado** para a máquina.

"_"denota o caracter espaço. O estado "paragem"para o programa. Linhas vazias na tabela apenas são utilizadas para formatação da mesma.

O programa acima remove zeros à esquerda, chegando a este estado final:



A partir do estado inicial, a cabeça move-se para a direita e chega ao primeiro 0, permanecendo no estado 0. Aqui, dada a terceira regra, substituímos o primeiro 0 com um espaço, movendo para a direita e permanecendo no estado 0. Agora, a cabeça lê "1", portanto deixamos-lo na fita, movendo para a direita para o estado 1. Próximo símbolo é um espaço, por isso, de acordo com o programa, podemos deixar o espaço na fita e regressar ao estado 0. Este processo continua deste modo até chegar ao estado final acima.

Pergunta

O que é que o programa seguinte faz?

Estado Corrente	Símbolo Corrente	Novo Símbolo	Direção	Novo Estado
0	1	_	d	1
0	*	*	d	0
1	1	_	d	2
1	*	f	d	paragem
2	1	_	d	1
2	*	v	d	paragem

Nota:

- "*"na coluna 2 significa "qualquer caracter"que não tenha sido apanhado pelas regras anteriores do mesmo estado
- "*"na coluna 3 significa "o mesmo caracter que foi lido"

Respostas Possíveis

(A) O programa substitui "1"'s por "2"'s.

(B) O programa substitui "2"'s por "1"'s.

(C) Quando o programa encontra um grupo de "1"'s, imprime "v"(verdadeiro) se o número de "1"'s for par. Caso contrário, imprime "f"(falso)

(D) Quando o programa encontra um grupo de "1"'s, imprime "v"(verdadeiro) se o número de "1"'s for ímpar. Caso contrário, imprime "f"(falso)

11 – Turing (Resolução)

Solução

(C)

Resolução

A opção correta é (C). No estado inicial, a linha [0 * * r 0] significa que todos os símbolos exceto "1"s são saltados. A primeira linha muda o estado 1 se um símbolo 1 é detetado.

Estar no estado 1 significa que um número ímpar de "1"s foi detetado até agora. Se outro 1 é detetado, a máquina muda o estado para 2. Se alguma coisa que não um 1 é detetado, a máquina imprime "t"e para.

Estar no estado 2 significa que um número par de "1"s foi detetado até agora. Se outro 1 é detetado, a máquina muda para o estado 1. Se alguma coisa que não um 1 é detetado, a máquina imprime "f"e para.

Isto é exatamente o resultado sugerido pela opção C.

Na opção A, se o programa substituísse todos os "1"s por "2"s, seria de esperar uma linha do programa que tivesse 1 na segunda coluna e 2 na terceira coluna. Não é isto que acontece, e podemos ignorar esta opção.

Na opção B, tal como antes, para substituir "2"s por "1"s, seria de esperar uma linha do programa que tivesse 2 ou * na segunda coluna e 1 na terceira coluna. Mas essas linhas não existem, e esta opção está incorreta.

Na opção D, um número par de "1"s leva ao estado 1 (quer por estarmos no estado 0 e lermos um 1, ou por estarmos no estado 2 e lermos outro 1). Em qualquer um dos casos, do estado 1 com um número par de "1"s, podemos: (a) ler outro 1, levando-nos ao estado 2, mas tendo agora um número ímpar de "1"s, ou (b) ler um símbolo diferente, terminadno o programa com a saída "f"(falso). Então, a opção D é incorreta.

Isto é Pensamento Computacional!

Uma máquina de Turing é um modelo computacional para um computador, desenvolvido pelo matemático britânico Alan Turing em 1936.

Embora máquinas de Turing sejam um conceito simples, os investigadores concordam que qualquer algoritmo que pode ser corrido num computador clássico pode também ser corrido numa máquina de Turing, embora não muito eficientemente.

Por outro lado, se alguém demonstrar que um algoritmo não pode ser corrido numa máquina de Turing, esse algoritmo também não pode ser corrido num computador clássico. Portanto podemos pensar em máquinas de Turing como um computador que se reduz ao máximo e serve para tirar conclusões e encontrar propriedades gerais de computadores clássicos. Computadores quânticos têm de ser vistos como uma exceção, já que não fazem parte da mesma categoria que os computadores clássicos.

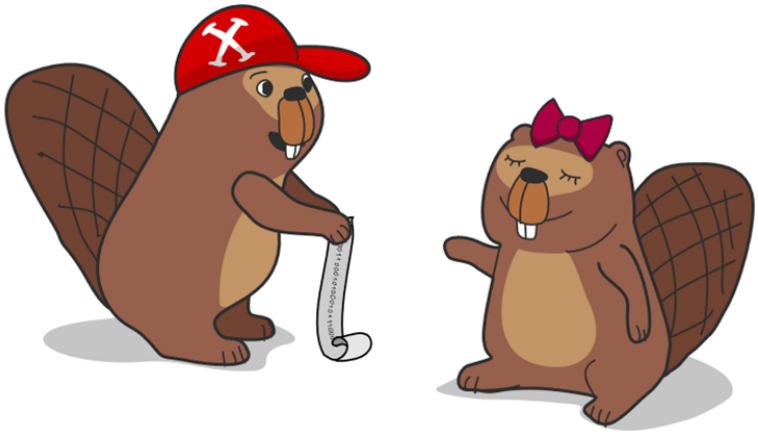
Em `http://morphett.info/turing` pode encontrar um simulador de uma máquina de Turing com programas incorporados que podem ser carregados e corridos. Adicionalmente, o seguinte programa multiplica um número binário por 2 (adicionando-o a ele próprio):

```
; --------------------------------------------------------------
; http://morphett.info/turing
; Multiply binary number by 2
; Input: one binary number eg '10011110' (158), 1111 (15)
; Output:                   eg '100111100' (316), 11110 (30)
;<current state> <current symbol> <new symbol> <direction> <new state>'
; go to end
0 1 1 r 0
0 0 0 r 0
0 _ _ l 1
; proceed no carry
1 0 0 l 1
1 1 0 l 2
; proceed carry
2 0 1 l 1
2 1 1 l 2
2 _ 1 * halt
```

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

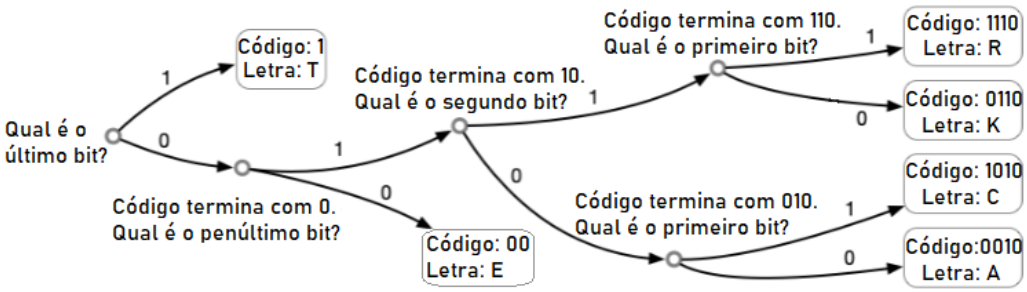
A resposta correta é: TAKECARE

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!

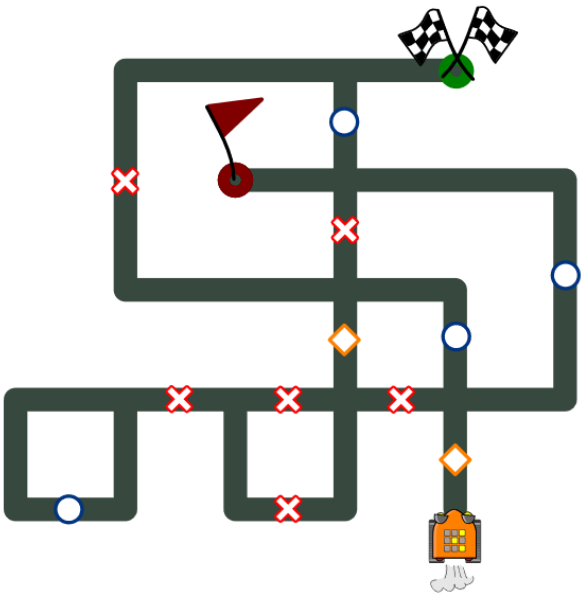
Todos os objetos com que um computador trabalha devem ser descritos como sequências de bits. Isto aplica-se também a textos. Espera-se que o objeto original possa ser reconstruído a partir da sua representação binária, mas isto só é possível se nunca acontecer dois ou mais objetos diferentes terem a 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.

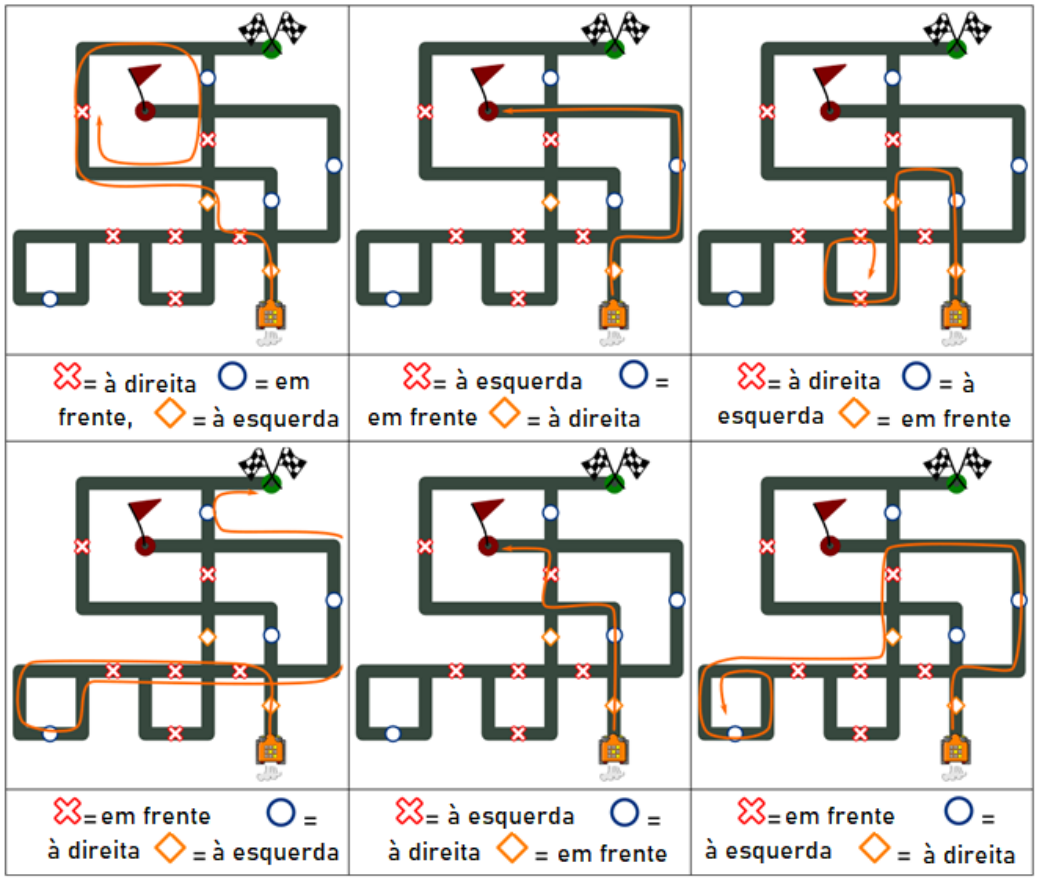
(F) ✕ = continuar em frente; ○ = virar à esquerda; ◇ = virar à direita.

13 – Robô Leitor de Símbolos (Resolução)

Solução

(D)

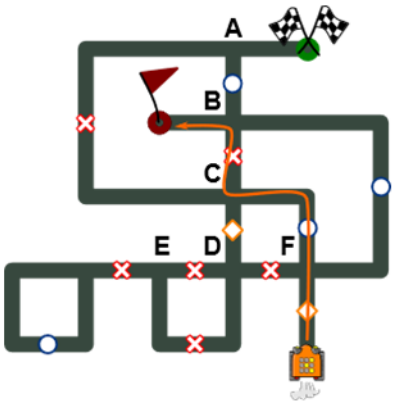
Resolução



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.

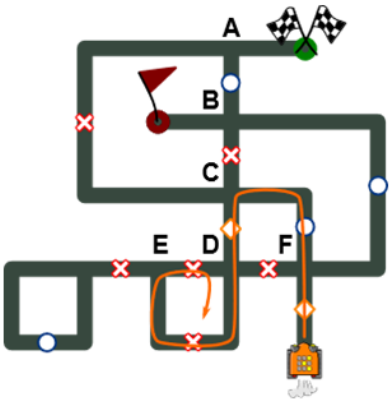


Tentativa 1.1

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.

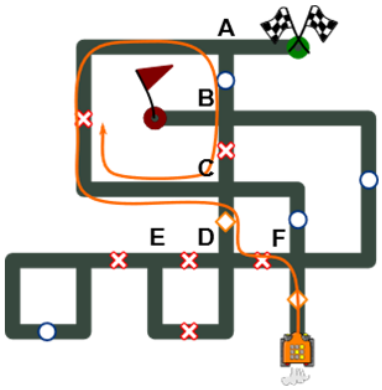


Tentativa 2

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

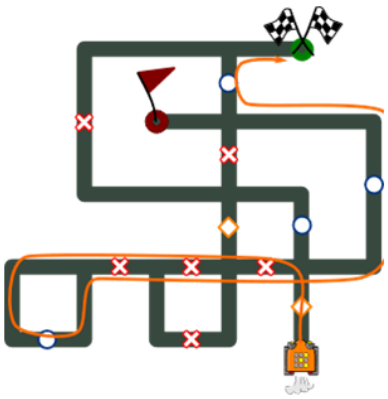
Tentativa 2.1

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.

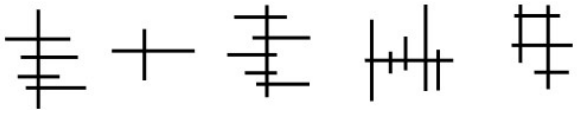
Puzzles como o Sudoku podem ser elegantemente resolvidos utilizando retraçamento.

14 – Segredo do Diário

A Petra e a Jana encontraram o diário secreto da sua amiga Lucie. Infelizmente para elas, a Lucie codificou o texto no seu diário utilizando linhas horizontais e verticais com a ajuda da seguinte tabela de letras:

A	B	C	D	E
F	G	H	I	J
K	L	M	N	O
P	R	S	T	U
V	W	X	Y	Z

As duas amigas repararam que no texto cifrado há muito mais do que 25 símbolos diferentes. Também conseguiram decifrar corretamente os símbolos abaixo, que codificam o nome do irmão da Lucie, PAVEL:



Pergunta

Decifra o nome do namorado da Lucie, escrito no diário com os seguintes símbolos:

Respostas Possíveis

(A) JOSEF

(B) PETER

(C) JESSE

(D) DENIS

14 – Localização Secreta (Resolução)

Solução

(A)

Resolução

O nome do namorado da Lucie é JOSEF. As linhas horizontais e verticais que se encontram nos símbolos codificados têm significado. O número de linhas horizontais corresponde ao número da linha na tabela de letras. De modo semelhante, o número de linhas verticais corresponde ao número da coluna. Por exemplo, o primeiro símbolo  tem 2 linhas horizontais e 5 linhas verticais. A letra que se encontra onde a linha 2 e a coluna 5 se encontram é J. Usando este processo para os restantes símbolos, pode desencriptar-se o nome JOSEF.

Isto é Pensamento Computacional!

Codificar e descodificar texto é uma parte muito importante da informática. É um requerimento comum para comunicar via internet de forma a estar seguro e para que a informação continue a estar protegida.

O método de codificação utilizado neste exercício não é utilizado na prática comum, mas o exercício demonstra algumas habilidades comuns de pensamento computacional. Em primeiro lugar, requer pensamento lógico para descobrir a relação entre os símbolos e as letras. Em segundo, requer abstração para identificar características importantes. Pode ter-se notado que a letra E está codificada duas vezes (uma vez em PAVEL e outra em JOSEF) mas os símbolos codificados têm aspectos diferentes. O comprimento das linhas ou onde se cruzam não é uma característica importante e podem ser ignorados. O importante é o número de linhas e se são verticais ou horizontais.

15 – Tabela de Verdade

Uma tabela de verdade descreve o resultado (○ ou ●) para uma ou mais variáveis dadas. O diagrama seguinte descreve, para todas as combinações possíveis de x, y e z, qual será a saída (ou *"output"*):

x	y	z	output
○	○	○	○
○	○	●	●
○	●	○	○
○	●	●	○
●	○	○	●
●	○	●	●
●	●	○	●
●	●	●	●

Poderias descrever esta tabela também com uma fórmula, listando exatamente para que valores dados o resultado seria ●:

(x=○ e y=○ e z=●) ou
(x=● e y=○ e z=○) ou
(x=● e y=○ e z=●) ou
(x=● e y=● e z=○) ou
(x=● e y=● e z=●)

Neste caso, utilizas exatamente 15 símbolos de entrada (x, y e z).

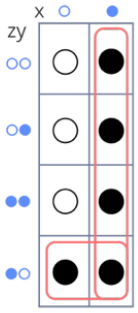
Se analisares a tabela com atenção, podes reparar que o resultado é sempre ● quando x=●, por isso podes encurtar esta fórmula para quatro símbolos:

(x=○ e y=○ e z=●) ou
(x=●)

Mas ainda consegues fazer melhor! Podes representar esta tabela com apenas três símbolos:

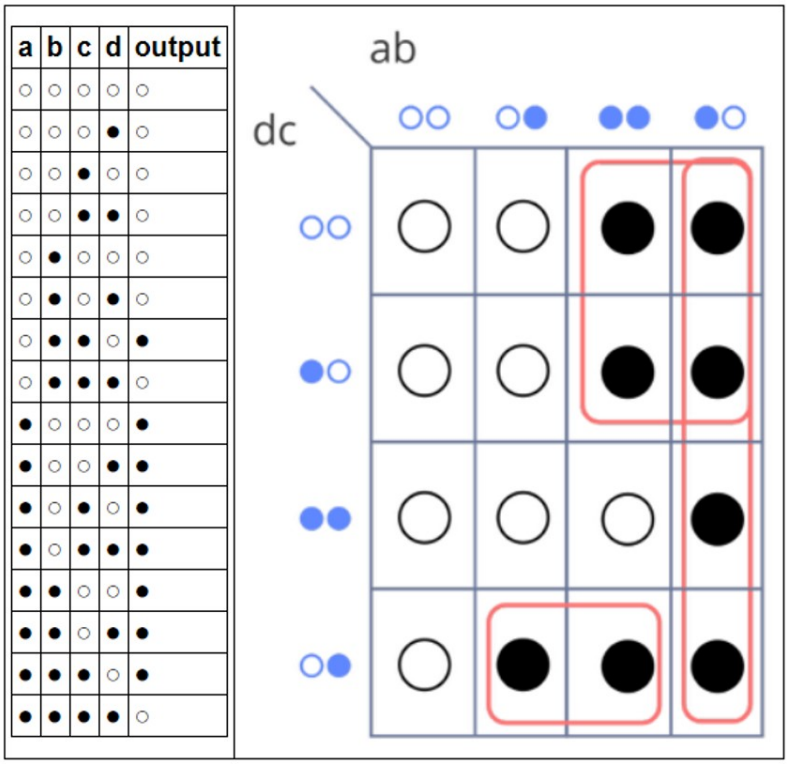
(y=○ e z=●) ou
(x=●)

Podes ver esses dois grupos desenhados no diagrama seguinte.



Pergunta

Aqui está uma tabela muito maior e também o chamado diagrama de Karnaugh correspondente, com a mesma informação (nota a ordem de 'c' e 'd'):



Qual é o número mínimo de símbolos que têm de ser utilizados para descrever esta tabela de verdade?

Resposta

Escreve a tua resposta (um número inteiro).

15 – Tabela de Verdade (Resolução)

Solução

7

Resolução

A resposta é 7.

Podemos chegar a este resultado olhando para o diagrama com os grupos circundados. Estes diagramas chamam-se diagramas de Karnaugh.

O tamanho dos símbolos resultantes rodeados indica quantos símbolos são necessários para descrever cada grupo.

Para descrever um único símbolo resultante precisamos de quatro símbolos de entrada. Para descrever dois símbolos resultantes adjacentes precisamos de três símbolos de entrada. Para descever quatro símbolos resultantes adjacentes (quer em 1×4 ou 2×2) precisamos de dois símbolos de entrada, e para descrever oito símbolos adjacentes (2×4) precisamos apenas de um símbolo de entrada.

Este diagrama mostra três grupos, dois grupos de 4 símbolos resultantes e 1 grupo de 2 símbolos resultantes. Portanto a resposta é $2 + 2 + 3 = 7$.

Deveria haver uma regra para grupo num diagrama de Karnaugh. O grupo 2×2, o 1×4 e o 2×1. A notação final dos valores de entrada seria dada pelas três regras seguintes com um total de 7 símbolos de entrada:

- (a=● and c=○) or
- (a=● and b=○) or
- (b=● and c=● and d=○)

Isto é Pensamento Computacional!

Diagramas de Karnaugh são utilizados para simplificar tabelas de lógica para que possam ser expressas usando o número mínimo de portas lógicas. Quando se constroem circuitos, o custo do projeto depende de quantos elementos diferentes são necessários no circuito. Qualquer forma de diminuir o número de valores de entrada a ser usados pode ser vista como positiva. Diagramas de Karnaugh ajudam-nos a encontrar este conjunto mínimo de valores de entrada.

Diagramas de Karnaugh são fixes porque foram desenhados de forma a que, para se mover de um quadrado para outro adjacente, nunca seja preciso alterar mais do que um bit. É de notar que os quadrados dão a volta! Do quadrado do topo numa coluna até ao quadrado do fundo na mesma coluna há apenas uma única mudança. Por causa desta propriedade especial podemos ver que se dois quadrados adjacentes tiverem ambos um resultado de "1", então o bit que vira para mover-se entre eles não é necessário na expressão final e podemos combinar ambas as linhas da tabela de verdade.

Diagramas de Karnaugh costumavam ser usados em tempos antigos para encontrar potenciais problemas em circuitos lógicos chamados de condições de corrida. Poderás querer aprofundar o teu conhecimento acerca de diagramas de Karnaugh e aprender mais sobre isto.

Diagramas de Karnaugh funcionam bem até 4 variáveis. Isto pode não parecer muito, mas muitos circuitos pequenos têm 4 ou menos valores de entrada ou podem ser separados em múltiplos circuitos com menos de 4 variáveis.