Castor Informático

O Desafio Internacional de Pensamento Computacional

Edição 2022

Categoria: **Seniores** (11º e 12º ano de escolaridade)

Tempo: **45 minutos**

Resolve tantos problemas quanto possível em 45 minutos.

Não é esperado que consigas resolver todos!

Responde apenas na folha de respostas.
É uma folha única, à parte, que deverás identificar com o teu nome.

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

Conteúdo

Página

Preâmbulo 2
Organização 2
Estrutura da Prova 3
Sobre os Problemas 3
1 – Porcas e Parafusos 4
Resolução 5
2 – Miss Infinito 7
Resolução 8
3 – Jogo do Galo 10
Resolução 11
4 – Barragens dos Castores 12
Resolução 13
5 – Velas Coloridas 14
Resolução 15
6 – Feiticeiro 16
Resolução 17
7 – Aldeias Entrelaçadas 18
Resolução 19
8 – Labirinto 20
Resolução 21
9 – Listas 22
Resolução 23
10 – Base de Dados dos Castores 24
Resolução 25
11 – AI dos Castores 26
Resolução 27
12 – Um Jogo de Corte e Rato 28
Resolução 29
13 – Empacotar 32
Resolução 33
14 – Recolhendo Pedras 34
Resolução 35
15 – Jóia Favorita 39
Resolução 40

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.

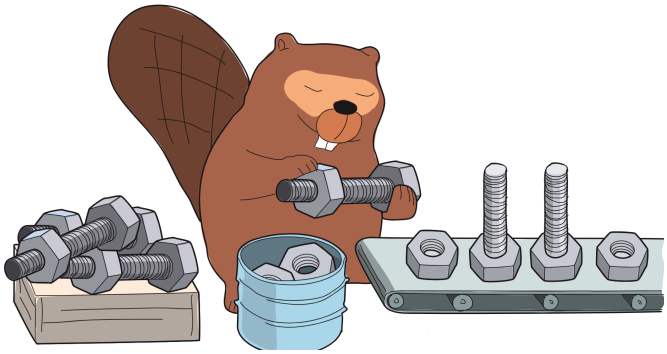
Sobre os Problemas

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
- Áustrália
- Áustria
- Brasil
- Canadá
- Chipre
- Eslováquia
- Filipinas
- Finlândia
- Hungria
- Irão
- Irlanda
- Itália
- Letónia
- Lituânia
- Macedónia
- Países Baixos
- Suiça
- Taiwan
- Uzbequistão
- Vietname

1 – Porcas e Parafusos

O castor Bruno trabalha na linha de montagem das porcas e parafusos.



A descrição do seu trabalho é a seguinte:

- O Bruno fica numa das pontas de um tapete rolante comprido, que contém uma linha de porcas e parafusos.
- O trabalho do Bruno é tirar cada elemento, uma porca ou um parafuso, do tapete rolante.
- Se o Bruno tirar uma porca do tapete rolante, ele coloca-a no balde ao seu lado.
- Se o Bruno tirar um parafuso do tapete rolante, ele pega numa porca do balde ao seu lado, enrosca-a no parafuso e coloca a peça montada numa caixa grande.

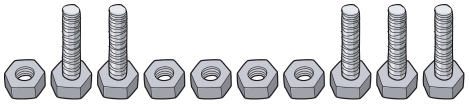
No entanto, as coisas podem correr mal para o Bruno de duas formas diferentes:

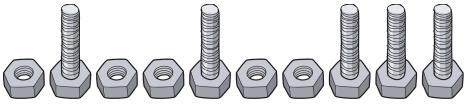
1. Se o Bruno tirar um parafuso do tapete rolante e não existir nenhuma porca no balde para enroscar.
2. Se não existirem mais porcas ou parafusos no tapete rolante e ainda estiverem porcas no balde.

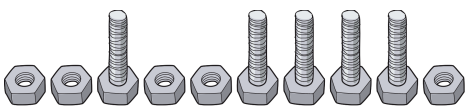
Pergunta

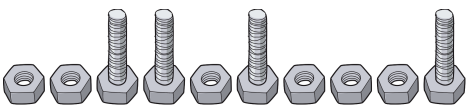
Que sequência de porcas  *e parafusos*  *, quando processada da esquerda para a direita,* ***não*** *causará problemas ao Bruno?*

Respostas Possíveis

(A) 

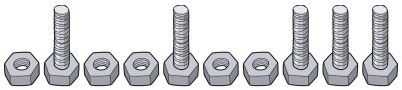
(C) 

(B) 

(D) 

1 – Porcas e Parafusos (Resolução)

Solução

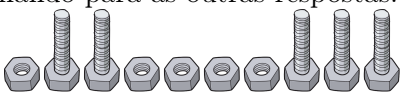
A resposta correta é a (C)  .

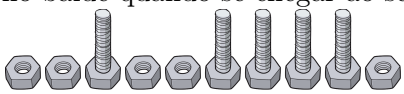
Resolução

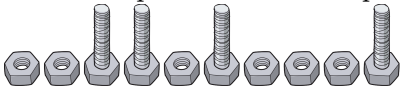
Podemos acompanhar o estado do balde e do tapete rolante da esquerda para a direita, com N =  (*"nuts"*) e B =  (*"bolts"*). No caso da resposta (C), temos:

Balde	Tapete Rolante
Vazio	N B N N B N N B B B
N	B N N B N N B B B
Vazio	N N B N N B B B
N	N B N N B B B
N N	B N N B B B
N	N N B B B
N N	N B B B
N N N	B B B
N N	B B
N	B
Vazio	Vazio

Olhando para as outras respostas:

A.  dará problemas depois de N B B, uma vez que não haverá nenhuma porca no balde quando se chegar ao segundo B.

B.  dará problemas depois de N N B N N B B B B, uma vez que não haverá nenhuma porca no balde depois do quinto B: note-se que há apenas 4 N's antes deste B.

D.  dará problemas depois de toda a sequência ser processada porque haverá duas porcas no balde, uma vez que há 6 N's e 4 B's.

Isto é Pensamento Computacional!

Esta tarefa destaca a utilização de um autómato com pilha (em inglês, *pushdown automaton* (PDA)). Um PDA é uma forma de descrever um algoritmo que se baseia no estado atual mas também tem uma quantidade ilimitada de memória sob a forma de uma pilha. Nesta tarefa, o estado é ter uma porca ou ter um parafuso no tapete rolante, e a pilha é o balde que contém as porcas.

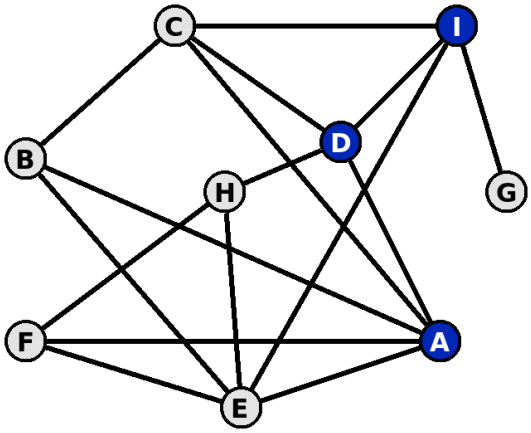
Um PDA pode ser usado para reconhecer ou analisar as chamadas linguagens sem contexto. Reconhecer ou analisar uma linguagem significa determinar se uma dada sequência de símbolos pertence à linguagem. Neste caso, podemos pensar nas porcas e parafusos como uma representação de parênteses equilibrados, onde N='('e B=')'. Ou seja, parênteses equilibrados são arranjos válidos de parênteses em expressões aritméticas. Exemplos de uma sequência de parênteses que não são equilibrados são (((() ou ()))(. A deteção de parênteses equilibrados é importante nos compiladores, uma vez que muitas linguagens de programação dependem de parênteses para indicar por exemplo o corpo de uma expressão condicional, bem como para representar expressões aritméticas.

2 – Miss Infinito

Os alunos numa sala de aula falam com os seus colegas conforme demonstra a figura. Por exemplo, o aluno H apenas fala com os alunos D, E e F durante o dia.

Na segunda-feira tiveram uma nova professora de Matemática. Por causa do seu cabelo, três estudantes (A, D e I) começaram imediatamente a chamar-lhe "Miss Infinito".

A alcunha espalhou-se entre os estudantes da seguinte forma: por cada estudante, se mais de metade dos colegas com quem fala usarem a alcunha, esse estudante irá usá-la no dia seguinte.



Pergunta

*Qual é o dia dessa mesma semana em que **todos** os alunos usam a alcunha "Miss Infinito" pela primeira vez?*

Respostas Possíveis

(A) Terça-feira

(B) Quarta-feira

(C) Quinta-feira

(D) Sexta-feira

(E) Nessa semana nunca chegam todos os estudantes a usar a alcunha.

2 – Miss Infinito (Resolução)

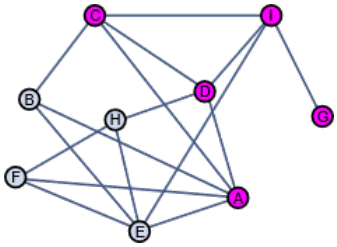
Solução

(D)

Resolução

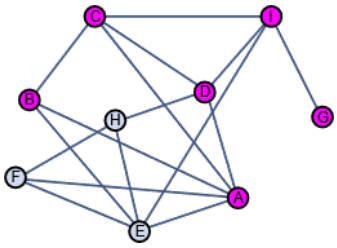
A resposta correta é a (D) Sexta-feira.

Na quinta-feira, os alunos G e C começam a usar a alcunha.

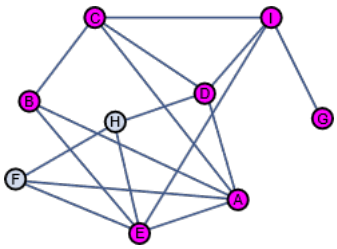


G fala apenas com um colega (I) e este começa a usar a alcunha na segunda-feira. C fala com 4 colegas e 3 deles usaram o nome na segunda-feira. 3 em 4 colegas é mais de metade. Assim, C também usa a alcunha na terça-feira.

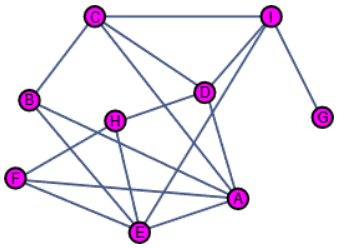
Na quarta-feira, B começa a usar a alcunha.



Na quinta-feira, E começa a usar a alcunha.



E na sexta-feira F e G começam a usar a alcunha.



Isto é Pensamento Computacional!

As redes sociais têm um papel fundamental na difusão de informação, infeções, ideias e influência entre os seus membros. Uma ideia ou inovação que aparece numa rede social correrá rapidamente em cascata através de interações sociais ou desaparecerá.

Maximizar a propagação da influência através das redes sociais é uma área de investigação muito ativa nas ciências informáticas, económicas e de gestão.

Esta tarefa considera um modelo especial de difusão chamado modelo de limite. Neste modelo, existe um limite para cada indivíduo, ou seja, a fracção das suas ligações que deve estar activa (usando a alcunha) para que o indivíduo se torne ativo.

3 – Jogo do Galo

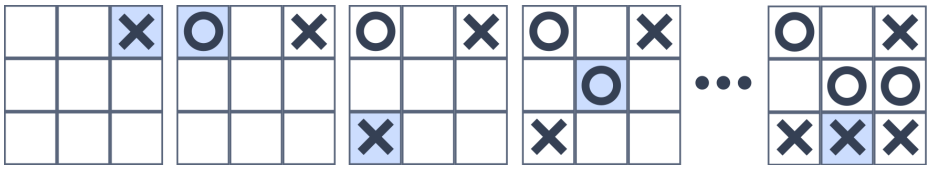
O Jogo do Galo é um jogo de papel e caneta para dois jogadores.

Regras:

Um jogador começa e depois ambos os jogadores marcam, à vez, os espaços numa grelha de três por três com um **X** ou um **O**. O primeiro jogador escolhe o **X** ou o **O**, e o outro jogador usa o outro símbolo. O jogador que conseguir colocar três das suas marcas numa fila horizontal, vertical ou diagonal é o vencedor. Se ninguém tiver sucesso e todas as nove caixas forem preenchidas, o jogo termina num empate.

Exemplo:

As imagens seguintes mostram as primeiras e a última jogada de uma partida (a última jogada de cada posição foi destacada):

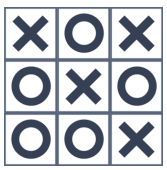


À imagem da direita chamamos folha de resultados de um jogo completo. Nem todas as folhas completadas aleatoriamente com **X** ou **O** são uma folha de resultados válida de acordo com as regras apresentadas anteriormente.

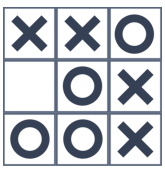
Pergunta

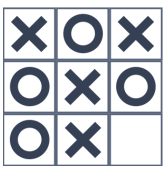
Qual das seguintes imagens é a única folha de resultados válida de um jogo completo de acordo com as regras acima?

Respostas Possíveis

(A) 

(B) 

(C) 

(D) 

3 – Jogo do Galo (Resolução)

Solução

(C)

Resolução

A resposta correta é a (C).

A resposta C está correta porque o jogador **O** venceu e depois o jogo terminou.

A resposta A não está correta. O jogador **X** ganhou o jogo mas o jogador **O** colocou mais marcas do que o jogador **X**, o que não é possível. Uma vez que o vencedor coloca sempre a última marca, ele apenas pode ter um número de marcas maior ou igual, mas não menor.

A resposta B não está correta porque há 5 marcas **X** e apenas 3 marcas **O**. Isso não é possível. A diferença entre estes números apenas pode ser 0 ou 1.

A resposta D não está correta porque o vencedor ainda não tinha sido determinado e os campos não estavam todos preenchidos.

Isto é Pensamento Computacional!

Resolvemos a tarefa verificando se as quatro imagens eram plausíveis ou não.

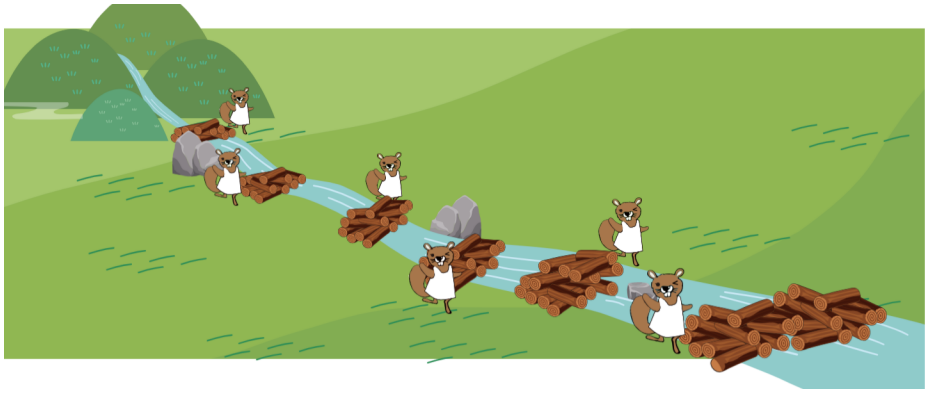
A partir das regras do jogo, podemos deduzir regras sobre a estrutura de uma folha de resultados, como por exemplo:

1) A diferença entre o número de marcas **X** e marcas **O** deve ser 0 ou 1.
2) Se nenhum jogador venceu, todos os campos devem estar preenchidos.
3) O vencedor deve ter um número igual ou mais uma marca do que o jogador derrotado.
4) Deve existir uma sequência vencedora numa folha de resultados.

Se uma imagem estiver em conflito com uma destas regras, não pode ser uma folha de resultados válida.

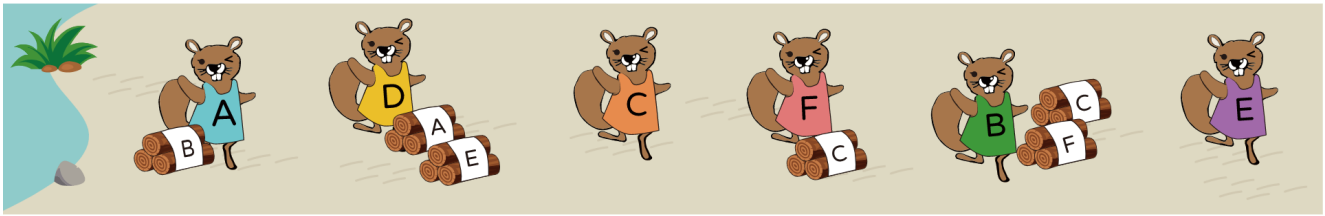
As regras são muito importantes nos sistemas computacionais de processamento de dados. Por exemplo, existem regras que definem o formato dos ficheiros de imagem, números de cartão de crédito ou mesmo números de telefone. Quando se tenta abrir um ficheiro com um editor de imagem, o software verifica primeiro se o conteúdo deste ficheiro é um dado válido de acordo com as regras de um formato de imagem.

4 – Barragens dos Castores



Seis castores (A, B, C, D, E, e F) construíram cada um a sua própria barragem ao longo da Ribeira dos Castores. Um dia veio umaa tempestade e alguns pedaços de madeira foram arrastados das barragens pela ribeira abaixo. Felizmente, todos os pedaços de lenha são marcados pelos construtores das barragens: por exemplo, os pedaços de madeira da barragem construída pelo Castor A têm um "A" marcado na madeira.

Após a tempestade, cada castor reúne-se para devolver os pedaços dos outros e recuperar os seus, como mostra a figura abaixo:



Pergunta

A julgar pelos pedaços que cada castor apanhou, qual é a ordem das barragens, de montante (lado mais próximo da nascente) a jusante (lado para onde a água desce)?

Respostas Possíveis

(A) A → B → C → D → E → F

(B) C → B → F → A → D → E

(C) C → F → B → D → A → E

(D) E → C → F → B → A → D

4 – Barragens dos Castores (Resolução)

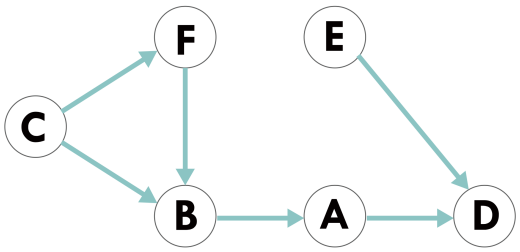
Solução

(D)

Resolução

A resposta correta é a (D) E → C → F → B → A → D.

Uma vez que os pedaços de madeira foram arrastados de montante para jusante, se o castor α encontrasse os pedaços de madeira do castor β, poderíamos assumir que o castor β estaria a montante do castor α. Ajuda a determinar a posição de cada castor se mapearmos a posição relativa entre dois castores com as setas representadas abaixo:



Isto é Pensamento Computacional!

Um grafo acíclico dirigido (*directed acyclic graph*, ou DAG) consiste em vértices e arestas direcionadas sem nenhum ciclo. Nesta tarefa podemos ver cada castor como um vértice. A direção de cada pedaço de madeira é uma aresta direcionada. Alinhar os castores na ordem correta de montante a jusante chama-se ordenação topológica; isto é, ordenar os vértices de acordo com a direção das arestas.

Podem existir várias ordens que encaixam na ordem topológica. Uma das formas de encontrar uma ordem tolopógica é utilizando o algoritmo de *Kahn*. Primeiro, seleciona-se um vértice V sem arestas de entrada. A seguir elimina-se o vértice V e todas as arestas que dele saem. Depois repete-se o processo anterior até o grafo estar vazio.

Em ciência de computadores, a ordenação topológica é utilizada em muitas aplicações, tais como o processamento de agendamentos, serialização de dados e resolução de dependências de símbolos em *linkers*.

5 – Velas Coloridas

O Simão tem velas com a forma dos algarismos 0 a 9. Há duas de cada algarismo. As velas vêm em três cores: laranja, vermelho, e azul. Todas as velas 0 são cor-de-laranja, todas as velas 1 são vermelhas, e assim por diante (ver tabela). Todos os anos, no seu aniversário, o Simão coloca velas no seu bolo para representar a sua nova idade.

Hoje é o 11.º aniversário do Simão e porque ambas as velas são da mesma cor, a sua família oferece-lhe um presente de aniversário extra. Ele deve esperar três anos até ter 14 anos para que ambas as suas velas voltem a ter a mesma cor. Depois haverá uma espera de três anos até aos 17 e mais cinco anos até completar 22.

Número	Cor
0	Laranja
1	Vermelho
2	Azul
3	Laranja
4	Vermelho
5	Azul
6	Laranja
7	Vermelho
8	Azul
9	Laranja

Pergunta

Se o Simão adotar este sistema a partir de hoje até ter 99 anos, qual será o número máximo de anos que ele terá de esperar entre dois aniversários em que duas velas da mesma cor são usadas para representar a sua idade?

Respostas Possíveis

(A) 5

(B) 6

(C) 7

(D) 8

5 – Velas Coloridas (Resolução)

Solução

(A)

Resolução

A resposta correta é (A) 5.

Seja XY a idade atual de uma pessoa em que X e Y são os dois algarismos de 0 a 9 que representam a idade. X e Y são da mesma cor de acordo com a tabela do enunciado.

Suponhamos que Y é 0, 1, 2, 3, 4, 5 ou 6. Nesse caso, a espera será de três anos até à idade X(Y+3) para ver duas velas da mesma cor.

A seguir, vamos olhar separadamente para os casos em que Y é 7, 8 ou 9:

- Se Y for 7, então X é também uma das velas vermelhas (1, 4, 7) e a espera será de cinco anos até à idade (X+1)2 para duas velas da mesma cor (azul);
- Se Y for 8, então X é uma das velas azuis (2, 5, 8) e a espera será de dois anos até à idade (X+1)0 para ver duas velas da mesma cor (laranja);
- Se Y for 9, então X é uma das velas laranja (0, 3, 6, 9) e a espera será de dois anos até à idade (X+1)1 para ver duas velas da mesma cor (vermelho).

Assim, o tempo de espera máximo será **5** anos. Nunca será necessário esperar 6 ou mais anos.

Isto é Pensamento Computacional!

Nesta tarefa, a sequência de números decimais de dois dígitos é mapeada para uma sequência de pares de cores. O elemento número n desta sequência abstrata são as cores das duas velas usadas para representar o número n. Frequentemente em ciência de computadores, dizem-nos as regras de como uma sequência é formada e devemos simular a sequência até que se verifiquem determinadas condições.

Os aspetos do pensamento computacional ilustrados com esta tarefa includem algoritmos, representação e reconhecimento de padrões.

Algoritmos: Esta tarefa é um exemplo de *análise de um algoritmo*. Nesta tarefa é-nos dado um algoritmo que gera uma sequência de pares de cores - as cores das velas utilizadas para representar cada número decimal de dois dígitos. Demoraria demasiado tempo a simular a sequência para encontrar a propriedade que procuramos, pelo que devemos antes analisar as propriedades do algoritmo, possivelmente incluindo a simulação de pequenos trechos da sequência.

Representação: Nesta tarefa, os números são (não unicamente) representados por pares de cores (dizemos que os números são mapeados para pares de cores). Nesta tarefa devemos aplicar a representação continuamente a diferentes números para identificar os que são representados por um par de velas com a mesma cor.

Reconhecimento de padrões: Em menor medida, esta tarefa é um exemplo de reconhecimento de padrões. Nesta tarefa, é-nos fornecido um padrão, que neste caso é uma sequência constituída por pares de velas coloridas, onde as cores são dadas por um algoritmo simples. Temos de prever repetidamente o que vem a seguir no padrão para identificar qual é o próximo elemento com uma determinada propriedade.

6 – Feiticeiro

Num castelo misterioso vive um único feiticeiro. Este feiticeiro consegue transformar-se numa fada ou criar uma fada ao seu lado (à direita). A fada, por sua vez, consegue transformar-se numa poção (à esquerda) e num dragão (à direita) ou transformar-se numa poção (à esquerda), num feiticeiro (ao centro) e num dragão (à direita).

A tabela seguinte mostra os conteúdos do castelo antes e depois de cada uma das quatro possíveis transformações.

Antes	Depois
	
	
	
	

Estas tranformações mágicas podem acontecer um qualquer número de vezes e em qualquer ordem. Dessa forma, qualquer feiticeiro e qualquer fada podem transformar-se em qualquer momento.

Pergunta

Começando com um único feiticeiro, qual dos estados do castelo ***não*** *é possível obter?*

Respostas Possíveis

(A) 

(B) 

(C) 

(D) 

6 – Feiticeiro (Resolução)

Solução

(B)

Resolução

A resposta é a opção (B).

Suponhamos que as transformações mágicas são numeradas de 1 a 4 como se segue:

Número	Antes	Depois
1		
2		
3		
4		

A opção A pode ser obtida começando comu m único feiticeiro e aplicando as transformações 1, 4, 2 e 3 por essa ordem.

A opção C pode ser obtida começando com um único feiticeiro e aplicando as transformações 2, 2, 3, 4 e 1 por essa ordem.

A opção D pode ser optida começando com um único feiticeiro e aplicando as transformações 2, 1, 3 e 3 por essa ordem.

Uma forma rápida de ver que a opção B não é possível é notar que as regras de transformação criam sempre uma poção e um dragão ao mesmo tempo. Assim, o número de poções no castelo será sempre igual ao número de dragões, o que não acontece na opção B.

Isto é Pensamento Computacional!

As tranformações mágicas podem ser vistas como um conjunto de regras utilizadas para gerar padrões de objetos no castelo.

Em ciência de computadores, uma *gramática livre de contexto* é uma ferramenta que pode ser usada para descrever regras que geram padrões. Gramáticas livres de contexto podem descrever linguagens (quer formais, quer naturais) e ao aplicar repetidamente as regras da gramática podemos gerar palavras (ou cadeias de caracteres) para formar uma linguagem.

Nesta tarefa, foi pedido que se determinasse quais das palavras não faziam parte da linguagem do castelo.

7 – Aldeias Entrelaçadas

À medida que os anos foram passando, as aldeias de *Repolholândia*, *Morangolândia* e *Cenourolândia* cresceram e começaram a sobrepor-se. Cada vez que uma nova casa é construída, os aldeões usam a seguinte regra para decidir a que aldeia a casa será atribuída:

A nova casa pertence à aldeia mais atribuída entre as X casas mais próximas. Empates resolvem-se atribuindo a nova casa à mesma aldeia da casa mais próxima.

Agora, duas novas casas foram construídas e atribuídas às aldeias usando o mesmo valor de X. A Casa 1 foi construída e atribuída antes da Casa 2.

Pergunta

Qual é o valor mais baixo possível de X para que a Casa 2 seja atribuída a Morangolândia? (Escreve um número na folha de respostas)

7 – Aldeias Entrelaçadas (Resolução)

Solução

5

Resolução

A resposta correta é X=5.

Se X=1, tanto a Casa 1 como a Casa 2 são atribuídas a Cenourolândia  .

Se X=2, a Casa 1 continua a ser atribuída a Cenourolândia  , uma vez que existe um empate que é resolvido atribuindo vendo a casa mais próxima é de Cenourolândia. A casa 2 também será atribuída a Cenourolândia, uma vez que os seus vizinhos mais próximos são ambos de Cenourolândia.

Se X=3, a Casa 3 será atribuída a Repolholândia  , uma vez que dois dos seus 3 vizinhos mais próximos são de Repolholândia. A Casa 2 é atribuída a Cenourolândia  , uma vez que todos os seus 3 vizinhos são diferentes mas o mais próximo é de Cenourolândia.

Se X=4, a Casa 1 é atribuída a Cenourolândia  , uma vez que tem dois vizinhos de Repolholândia  e dois de Cenourolândia e o mais próximo é de Cenourolândia. Consequentemente, a Casa 2 também é atribuída a Morangolândia, uma vez que tem dois vizinhos de Cenourolândia e dois de Morangolândia  mas o mais próximo é de Morangolândia.

Se X=5, a Casa 1 é atribuída a Cenourolândia  , de modo semelhante ao que acontece quando X=4, e a casa 2 é atribuída a Morangolândia  , uma vez que três dos seus cinco vizinhos mais próximos são de Morangolândia.

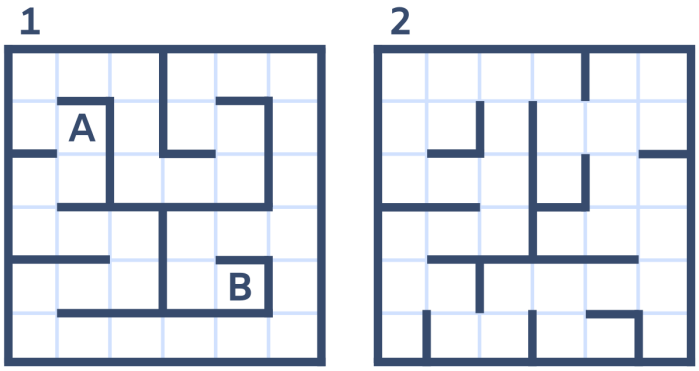
Quando X=6, a Casa 2 também é atribuída a Morangolândia, mas estes não é o valor mais baixo de X para o qual isto acontece.

Isto é Pensamento Computacional!

A regra para atribuir cada casa a uma aldeia é um exemplo de um algoritmo de classificação chamado *k-nearest neighbors* (kNN). O kNN classifica cada dado novo dado olhando para os k dados mais semelhantes que já foram classificados. Nesta tarefa, a variável X toma o papel de k. Em aprendizagem automática, o kNN é frequentemente utilizado por ser fácil de implementar e porque não requer a aplicação de um modelo complexo para os dados fornecidos.

8 – Labirinto

Um pequeno castor está num labirinto. O labirinto é constituído por dois andares, cada um com a sua própria grelha de obstáculos.



O castor pode mover-se entre duas células adjacentes dentro do mesmo piso se não existirem obstáculos entre as células; isto demora um segundo. O castor também pode usar a sua varinha mágica para se mover até à célula correspondente do outro piso; isto demora cinco segundos.

Por exemplo, se o castor estiver na célula A, há três movimentos possíveis:
1. Mover para a esquerda. Este movimento demora 1 segundo.
2. Mover para baixo. Este movimento demora 1 segundo.
3. Mover para a célula correspondente do outro piso. Este movimento demora 5 segundos.

O castor começa na célula A e quer chegar à célula B o mais rapidamente possível.

Pergunta

Qual é o tempo mais curto que o castor precisa para chegar à célula B a partir da célula A?

Respostas Possíveis

(A) 16

(B) 17

(C) 18

(D) 20

8 – Labirinto (Resolução)

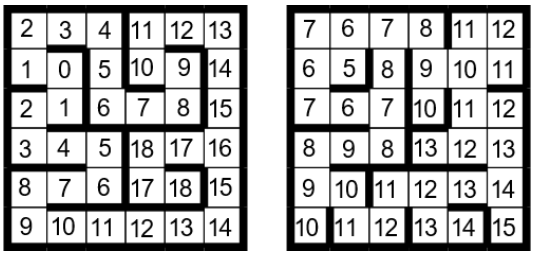
Solução

(C)

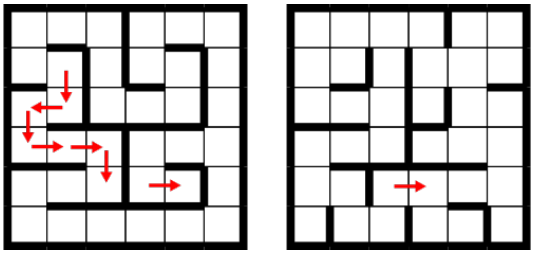
Resolução

A resposta correta é a (C) 18.

O problema dado é um problema do caminho mais curto entre duas localizações. Existem diferentes abordagens para obter a solução. A imagem abaixo mostra os comprimentos dos percursos ótimos para todas as células partindo de A.



É possível observar que o comprimento do caminho mais curto para B é 18. Um dos possíveis caminhos ótimos é o seguinte:



A Resposta (D)20 corresponde ao caminho ótimo quando há movimento em apenas um piso. A resposta A(16) corresponde ao limite de tempo mais baixo a chegar a B a partir de A se for através do outro piso se não existissem paredes.

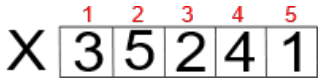
Isto é Pensamento Computacional!

O problema do caminho mais curto é um problema bem estudado de teoria de grafos, e um dos métodos mais conhecidos para o resolver é o algoritmo de Dijkstra. As suas aplicações incluem por exemplo encontrar o caminho ótimo numa rede informática ou no mapa de uma cidade.

Uma outra aplicação seria no desenho das ligações um circuito integrado de larga escala (*Very Large Scale Integration* ou VLSI). No problema dado, o movimento entre pisos demora mais tempo do que o movimento dentro de um piso (5 a 1). Isto corresponde ao facto de que em circuitos de várias camadas a cablagem cruzada é mais cara do que a cablagem intra-camada. Os circuitos VLSI comuns são colocados num cristal de silício de 10 camadas, enquanto que esta tarefa introduz apenas um problema de 2 camadas. Além disso, a concepção das VLSI requer o encaminhamento de milhões ou até biliões de fios para ligar transístores, entradas, saídas e células de memória para obter o circuito das VLSI para dispositivos digitais.

9 – Listas

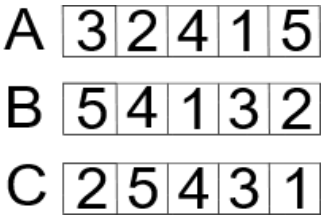
Podemos representar uma lista dos números 3, 5, 2, 4, 1 visualmente como demonstrado na figura abaixo (os números vermelhos mais pequenos em cima indicam as posições na lista).



Escrevemos (X 2) para descrever o número na posição 2. Portanto, (X 2) é 5.
De modo semelhante, (X 5) é 1.

As posições podem ser indicadas indiretamente. Por exemplo (X (X 3)) é 5 porque (X 3) é 2, portanto (X (X 3)) = (X 2) = 5.

Aqui estão 3 listas A, B e C.



Pergunta

Qual é o número descrito por (A (B (C 3)))? (Escreve o número na folha de respostas)

9 – Listas (Resolução)

Solução

4

Resolução

(C 3) = 4, portanto (B (C 3)) = (B 4) = 3, então (A (B (C 3)) = (A (B 4)) = (A 3) = 4.

Isto é Pensamento Computacional!

Estruturas de dados são essenciais para programação. As que conseguem conter uma lista de dados são especialmente úteis. As estruturas de dados, tal como muitas outras entidades em ciência de computadores podem ser interligadas, de modo a que um elemento de uma lista possa descrever uma posição noutra lista. Esta forma indireta de descrever posições é um conceito poderoso. Pode parecer confuso no início mas não é difícil calcular os valores passo a passo.

10 – Base de Dados dos Castores

Uma dúzia de famílias vive na aldeia dos castores. O castor João criou a base de dados dos aldeões, registando dados sobre cada castor na forma de sequências de 16-bits, desde o bit $b15$ (esquerda) até ao bit $b0$ (direita), como se segue:

- $b15$ a $b12$: quatro bits para o número da família;
- $b11$: um bit para o género (0 = feminino, 1 = masculino);
- $b10$ a $b4$: sete bits para o peso (um número inteiro de quilogramas);
- $b3$ e $b2$: dois bits para "trabalhador qualificado em" (00 = construção de alojamentos, 01 = construção de barragens, 10 = armazém de comida, 11 = educação de jovens castores);
- $b1$ e $b0$: dois bits para a comida preferida (00 = casca de árvore, 01 = plantas aquáticas, 10 = relvas, 11 = sedimentos).

Por exemplo, a sequência 0100 0 0100101 10 01 denota um castor que pertence à família 4, é do género feminino, pesa 37 kg, é uma trabalhadora qualificada num armazém de comida e gosta de plantas aquáticas.

Pergunta

O castor João consulta a base de dados formulando expressões Booleanas (em lógica positiva: 0 = falso, 1 = verdadeiro). Que conjunto de castores denota a seguinte expressão?

$b11$ **e não**($b10$) **e** $b9$ **e** $b7$ **e não**($b3$ **e** $b2$)

Respostas Possíveis

(A) Fêmeas que pesam pelo menos 16 kg, trabalhador experiente em armazém de comida.

(B) Machos que pesam pelo menos 64 kg, trabalhador experiente em construção de alojamentos ou barragens.

(C) Machos que pesam entre 40 a 63 kg, trabalhador experiente em construção de alguma coisa ou armazém de comida.

(D) Machos que pesam no máximo 39 kg, trabalhador experiente em construção de barragens.

10 – Base de Dados dos Castores (Resolução)

Solução

(C)

Resolução

A resposta correta é a (C). Uma vez que b11 = 1 significa "macho", b10 = 0 significa "pesar no máximo 63 kg", b9 = 1 e b7 = 1 significa "pesar pelo menos (32 + 8) kg" e finalmente b3 = 0 ou b2 = 0 apenas exclui "trabalhador qualificado em educação de jovens castores" (b3 = 1 e b2 = 1).

Note-se que de acordo com as leis de De Morgan **não**(b3 **e** b2) é equivalente a **não**(b3) **ou não**(b2).

A resposta (A) está incorreta: b11 = 1 significa "macho" mas a resposta implica fêmeas. A expressão que denota o conjunto na resposta (A) é **não**(b11) **e** b8 **e** b3 **e não**(b2).

A resposta (B) está incorreta: b2 = 0 também significa "trabalhador qualificado em armazém de comida"; além disso, b9 = 1 e b7 = 1 estabelece um limite máximo no peso. A expressão que denota o conjunto na resposta (B) é simplesmente b11 **e** b10 **e não**(b3).

A respota (D) está incorreta: b3 = 0 também significa "trabalhador qualificado na construção de alojamentos"; além disso, b10 = 0 estabelece um limite superior no peso. A expressão que denota o conjunto na resposta (D) é b11 **e não**(b10) **e não**(b2) **e** b2.

Note-se que o conjunto na resposta (A) é separado de cada um dos outros três, mas - em geral - isto não é verdade para os outros três pares de conjuntos ((B) e (C); (B) e (D); (C) e (D)); e nenhum dos conjuntos nas respostas (B), (C) e (D) é um subconjunto de outro.

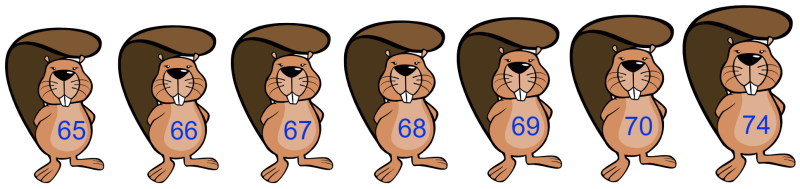
Isto é Pensamento Computacional!

No contexto acima descrito, uma expressão booleana (ou, melhor, uma fórmula de lógica proposicional) pode ser utilizada para consultar a base de dados simples, ou seja, pode ser vista como uma "chave de pesquisa". Em geral, esta fórmula identifica um subconjunto dos dados (e estas situações podem ser representadas graficamente por meio de diagramas de Venn). Exemplos do tipo aqui proposto encontram-se nos escritos de Zuse de 1943-44 sobre cálculo proposicional e as suas aplicações em circuitos de retransmissão. Konrad Zuse era um engenheiro civil alemão, mas a sua maior realização foi o primeiro computador eletromecânico controlado por programa, que se tornou operacional em 1941.

A maioria das linguagens de programação modernas têm o operador Booleano unário **não** e os operadores binários **e**, **ou**. Os operadores **não**, **e** são suficientes - assim como **não**, **ou** - pelas tais leis de De Morgan.

Os circuitos *booleanos*, constituídos pelas portas lógicas correspondentes, estão na base de muitos componentes digitais utilizados em computadores (multiplexadores, somadores, unidades lógicas aritméticas, etc.). Em 1913 (mas já descoberto por Charles S. Peirce em 1880) Henry M. Sheffer mostrou que todos os operados lógicos podem ser derivados a partir de um único, **nand** ("not-and" - "não-e"): A **nand** B é equivalente a **não** (A **e** B), portanto A **nand** A é equivalente a **não**(A). Tal como portas **nand**, portas **nor** ("not-or" - "não-ou": a **nor** B é equivalente a **não** (A **ou** B), portanto A **nor** A é equivalente a **não** (A)) são "portas universais"; eles podem ser combinados para formar qualquer outra porta lógica e, de facto, vários sistemas eletrónicos foram construídos exclusivamente a partir de portas **nand** e **nor**.

11 – AI dos Castores



Os castores construíram um sistema "admiravelmente inteligente"(AI); o sistema serve para medir o tamanho de um animal e, baseando-se somente nisso, decidir se o animal é um castor ou não. O sistema AI aprende a tomar as suas decisões a partir de exemplos.

Primeiro, o sistema AI aprende a partir de animais exemplo com os seguintes tamanhos:

- 65, 66, 67, 68, 69 ⇒ castor
- 11, 101, 110, 120, 130 ⇒ não castor

Depois de terminado de treinar o sistema AI, os castores avaliaram quão bem o sistema AI funciona na sua avaliação de novos exemplos. O resultado é o seguinte:

- 70, 74 ⇒ castor
- 86, 38 ⇒ não castor
- 40, 80 ⇒ castor

O sistema AI fez um erro pois os dois animais de tamanhos 40 e 80 não são, na verdade, castores!

Porque aconteceu isto? O sistema AI tinha observado que um animal com o tamanho 11 ou com o tamanho 101 **não é** um castor e que um animal com o tamanho 65 ou com o tamanho 69 **é** um castor. Olhando para as diferenças de tamanho, o sistema AI decidiu que apenas animais com tamanhos superiores a 38 e inferiores a 85 são castores.

Assim, de forma a melhorar o sistema AI, o castor deu-lhe um novo exemplo: um animal com o tamanho 42 **não é um castor**.

Pergunta

Depois do novo exemplo, como é que o sistema AI classifica dois animais de tamanho 48 e 84?

Respostas Possíveis

(A) castor, castor

(B) castor, não castor

(C) não castor, castor

(D) não castor, não castor

11 – AI dos Castores (Resolução)

Solução

(C)

Resolução

A resposta correta é a (C). Como é que o sistema AI inferiu o intervalo 38-85 utilizado para a identificação de castores? Na verdade não sabemos ao certo. Mas uma vez que entre os exemplos que o sistema viu 11 não era um castor e 65 era um castor, o sistema computou um limite para ser um castor ou não e colocou-o *algures* entre 11 e 65. Um ponto sensato para o limiar pode ser o ponto médio (11+65)/2 = 38 e um raciocínio semelhante pode ser aplicado no outro extremo: (69+101)/2 = 85. Usando o mesmo procedimento, depois de ter visto um novo exemplo (42 não é um castor) o limiar de 85 não deve mudar, mas o novo limiar esquerdo deve tornar-se (42+65)/2 = 53,5. Portanto, um animal de tamanho 48 seria identificado como "não castor", enquanto que um animal de tamanho 84 seria identificado como castor.

Isto é Pensamento Computacional!

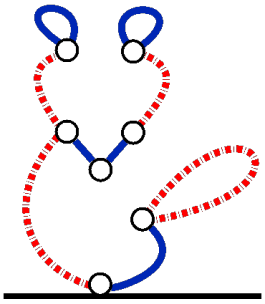
O sistema "Admiravelmente Inteligente" nesta tarefa da Bebras aplica um algoritmo de Aprendizagem Automática (*Machine Learning* ou ML). Na área de ML, os cientistas de computadores estudam algoritmos informáticos que podem melhorar automaticamente a sua tomada de decisão. Os algoritmos de ML constroem um modelo baseado em dados de exemplo, conhecidos como dados de treino, a fim de fazer previsões ou decisões sem serem explicitamente programados para o fazer. Assim, a sua qualidade pode depender fortemente da qualidade dos dados de formação utilizados, como se pode ver a partir desta tarefa.

Os algoritmos de ML são utilizados numa grande variedade de aplicações, tais como na medicina, filtragem de correio electrónico, reconhecimento da fala e visão por computador, onde é difícil ou inviável desenvolver algoritmos convencionais para realizar as tarefas necessárias.

ML é considerada como uma sub-área da Inteligência Artificial. Encontra-se uma simulação online da aprendizagem automática em https://machinelearningforkids.co.uk ou em https://code.org/oceans.

12 – Um Jogo de Corte e Rato

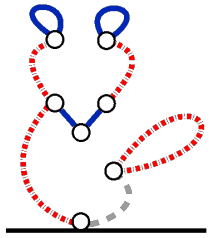
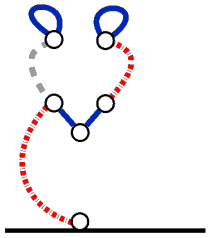
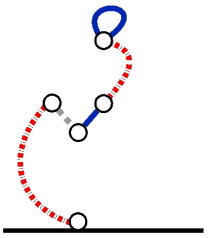
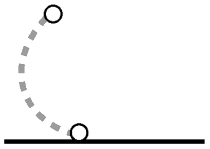
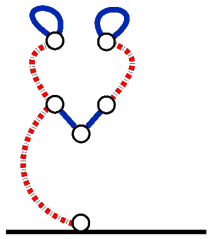
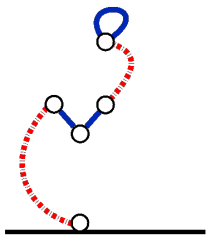
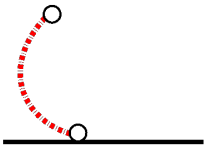
Dois amigos castores, o Bruno e o Rui, estão a jogar um jogo. Eles começam por traçar uma linha preta grossa no fundo de um papel, chamando-lhe "chão". Depois, desenham vários segmentos azuis (linhas contínuas) e vermelhas (linhas a tracejado), criando a seguinte figura em forma de rato da figura da direita.



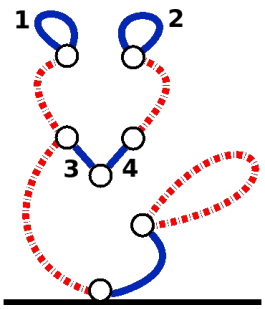
As regras do jogo são as seguintes:

- Eles fazem turnos para cortar qualquer segmento da sua escolha. No entanto, o Bruno apenas pode cortar segmentos azuis enquanto que o Rui apenas pode cortar segmentos vermelhos.
- Cortar um segmento remove esse segmento e todos os outros segmentos que já não estão ligados ao chão.
- O primeiro jogador que já não tem mais segmentos para cortar é considerado o derrotado.

Uma possível sequência de jogadas é dada na tabela abaixo. São mostradas duas figuras por turno: a de cima marca o segmento que o jogador pretendo cortar e a de baixo mostra o resultado desse corte.

Turno do Bruno	Turno do Rui	Turno do Bruno	Turno do Rui
			
			

Uma vez que o Bruno já não tem mais segmentos para cortar, ele perde o jogo e o Rui é declarado vencedor.



Pergunta

Se o Bruno for o primeiro a jogar e ele fizer sempre a melhor jogada possível em cada turno, qual dos segmentos é que ele deveria cortar primeiro para garantir a sua vitória - independentemente das jogadas do Rui? Utiliza a figura da direita como referência para a numeração dos segmentos.

Respostas Possíveis

(A) Segmento 1 **(B)** Segmento 2 **(C)** Segmento 3

(D) Segmento 4 **(E)** O Bruno não tem hipótese de ganhar.

12 – Um Jogo de Corte e Rato (Resolução)

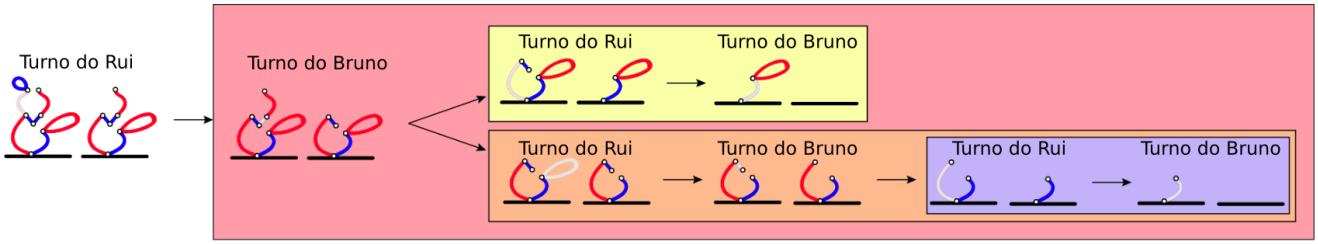
Solução

(B)

Resolução

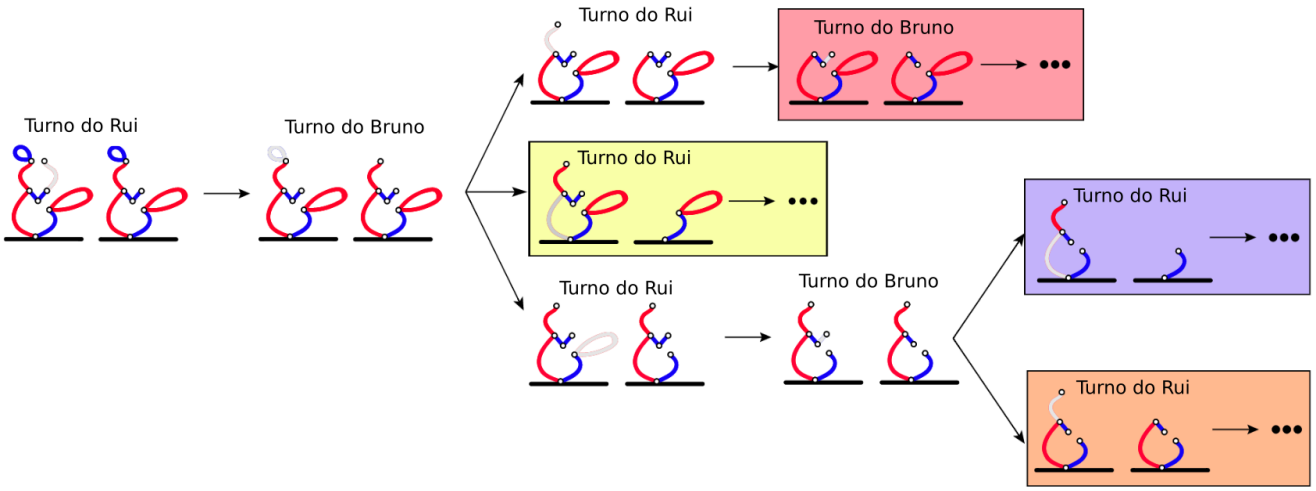
A resposta correta é a (B) Segmento 2.

Para provar que cortar o Segmento 2 garante a vitória do Bruno (assumindo que ele continua a fazer a melhor jogada em cada turno), começamos por selecionar uma das jogadas possíveis do Rui em resposta à sua jogada de abertura. A partir daí, jogamos sistematicamente através de jogos completos até encontrar uma sequência de jogadas vencedora (ou uma *estratégia*). O diagrama abaixo mostra uma estratégia vencedora se o Rui decidir cortar a metade esquerda da "cara".

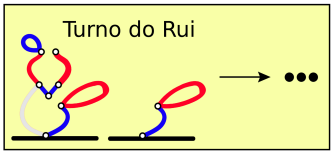


Para evitar jogar ao calhas, é vantajoso o Bruno *forçar* o Rui numa posição que faça parte da estratégia vencedora acima - uma vez que isto permite ao Bruno assegurar a vitória ao aplicar a mesma sequência de jogadas desta estratégia. Nos diagramas seguintes, estas posições são marcadas usando caixas com um código de cores, ilustrando como esta ideia pode ser aplicada para reagir às restantes respostas do Rui à jogada de abertura do Bruno.

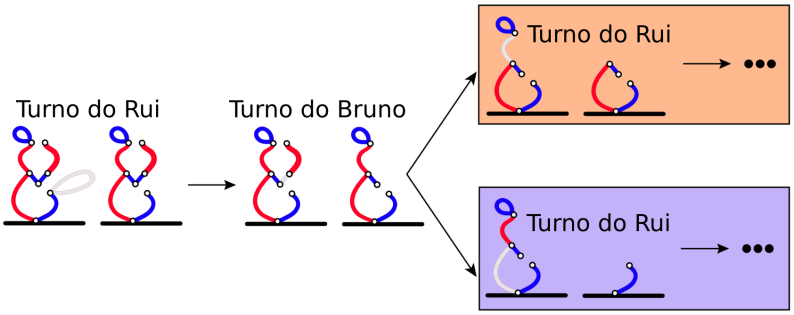
- O Rui corta a metade direita da "cara"



- O Rui corta a metade esquerda do "corpo"

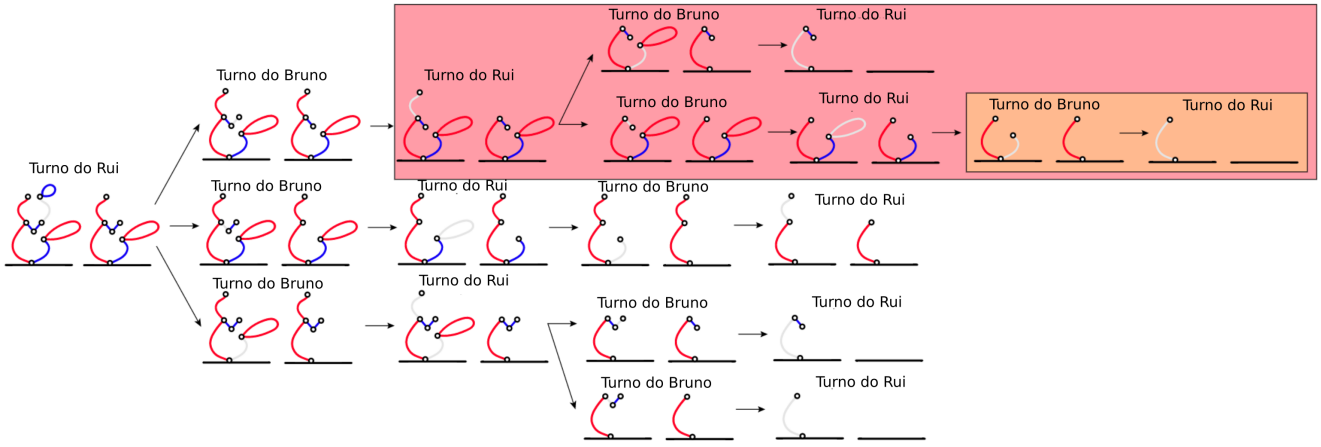


- O Rui corta a "cauda"

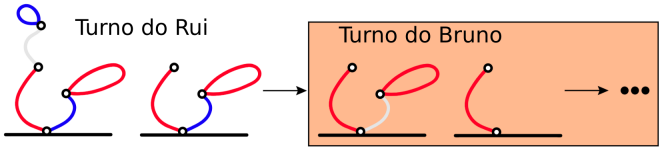


Para ficar mais completo, também podemos mostrar que o Bruno não tem hipótese de vencer se ele começar por cortar qualquer outra segmento e o Rui encontrar a forma perfeita de responder em cada turno.

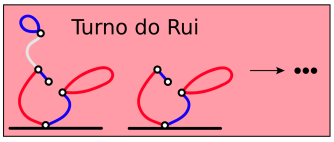
- O Bruno começa por cortar o Segmento 1



- O Bruno começa por cortar o Segmento 3



- O Bruno começa por cortar o Segmento 4



Isto é Pensamento Computacional!

O jogo nesta tarefa, conhecido como *Hackenbush*, foi introduzido pelo matemático inglês John H. Conway no início dos anos 80. Pertence a uma categoria de jogos chamada *jogos combinatórios*. Jogos combinatórios de dois participantes satisfazem três critérios: (1) são jogados *sequencialmente* (os jogadores têm turnos), (2) são determinísticos (não são infuenciados por dispositivos que introduzam aleatoriedade, tais como roletas ou dados), e (3) os jogadores têm sempre a informação perfeita (nenhuma informação é escondida ou ocultada). Exemplos populares incluem, xadrez, damas, jogo do galo, Nim e Hackenbush.

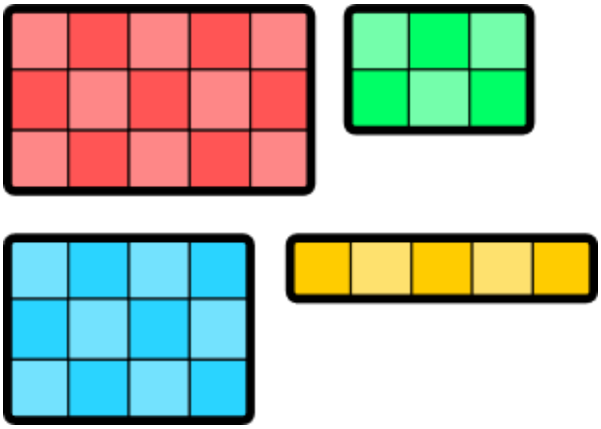
Os jogos combinatórios são de grande valor para a informática, especialmente para a ciência de computadores teórica e a inteligência artificial. Áreas de interesse incluem determinar o vencedor se ambas as partes jogarem sempre a melhor jogada possível (*jogada perfeita*), formulando estratégias para maximizar as chances de vitória e criando *heurísticas* (estimativas) para avaliar qual dos jogadores tem a vantagem numa dada posição.

A simplicidade do Hackenbush desmente conceitos profundos que nascem do seu estudo; sobretudo, a avaliação das suas posições está ligada a uma nova classe de números chamada *números surreais* (inventada por Conway e popularizada pelo cientista de computação Donald Knuth). Uma análise mais profunda mostra que a figura em forma de rato nesta tarefa favorece de facto Bob (azul) - independentemente de quem se mexe primeiro, Bob tem a garantia de vencer, assumindo um jogo perfeito. Hackenbush é também descrito como um *jogo frio*, ou seja, fazer uma jogada só pode piorar a posição de um jogador.

As *árvores de jogo* são construídas para enumerar as posições possíveis em cada jogada. No entanto, os jogos podem tornar-se rapidamente complexos, o que torna a geração de toda a árvore de jogo ineficiente ou inviável. Para este fim, foram concebidos algoritmos para evitar explorar as mesmas posições duas vezes (como fizemos na nossa solução) ou para "podar" (excluir) posições cujas avaliações ficam abaixo de um determinado limiar. Na inteligência artificial moderna, técnicas como a *aprendizagem de reforço*, onde um computador é treinado "aprendendo com os seus erros", estão a ser exploradas para melhorar a tomada de decisões para jogos complexos, como o xadrez ou Go.

13 – Empacotar

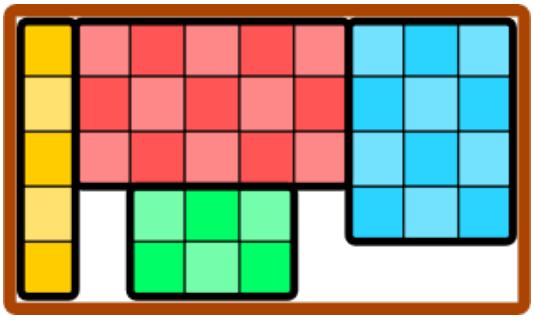
Uma fábrica produz 4 tipos de cerveja. Para cada tipo, eles usam uma caixa com um tamanho diferente. Podemos ver estas caixas pela vista de cima na imagem a seguir. Observa que a maior caixa contém 15 garrafas.



(A fotografia à direita mostra uma caixa para 12 garrafas a partir da vista de lado.)

Tens de preparar um presente especial para o presidente contendo uma caixa de cada sabor. Estas quatro caixas têm de ser colocadas num recipiente *retangular*. As caixas não podem ser empilhadas umas em cima das outras e queremos deixar o mínimo de lacunas possível (que depois podemos preencher uma garrafa).

Por exemplo, se usarmos o recipiente abaixo, temos de adicionar 7 garrafas para o recipiente ficar preenchido.



Pergunta

Num recipiente retangular que contém as quatro caixas-presente (uma de cada tipo) com o mínimo de lacunas possível, quantas garrafas teriam de ser adicionadas para este recipiente estar preenchido? (Escreve um número na folha de respostas)

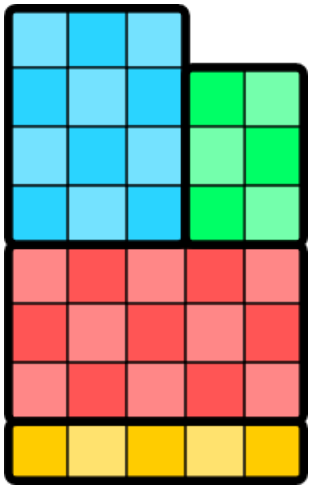
13 – Empacotar (Resolução)

Solução

2

Resolução

A resposta correta é 2. Aqui está representado um arranjo possível que atinge este valor para as caixas.



O número de garrafas em todas as 4 caixas juntas é 12 + 15 + 6 + 5 = 38. Um recipiente que contenha 38 garrafas com 0 lacunas deve ter as dimensões 1 x 38 ou 2 x 19. Nunca será possível encaixar a caixa 3 x 5 (ou a caixa 3 x 4) neste contentor.

Um contentor com 1 lacuna deve conter 39 garrafas. Há duas possibilidades, 1 x 39 (não é possível) ou 3 x 13. As duas maiores caixas ocupariam 9 filas dentro de um tal recipiente. As restantes 4 filas não são suficientes para colocar a caixa mais pequena de tamanho 1 x 5.

Assim, 2 lacunas é o mínimo que podemos ter em qualquer contentor, e podemos consegui-lo como mostrado acima.

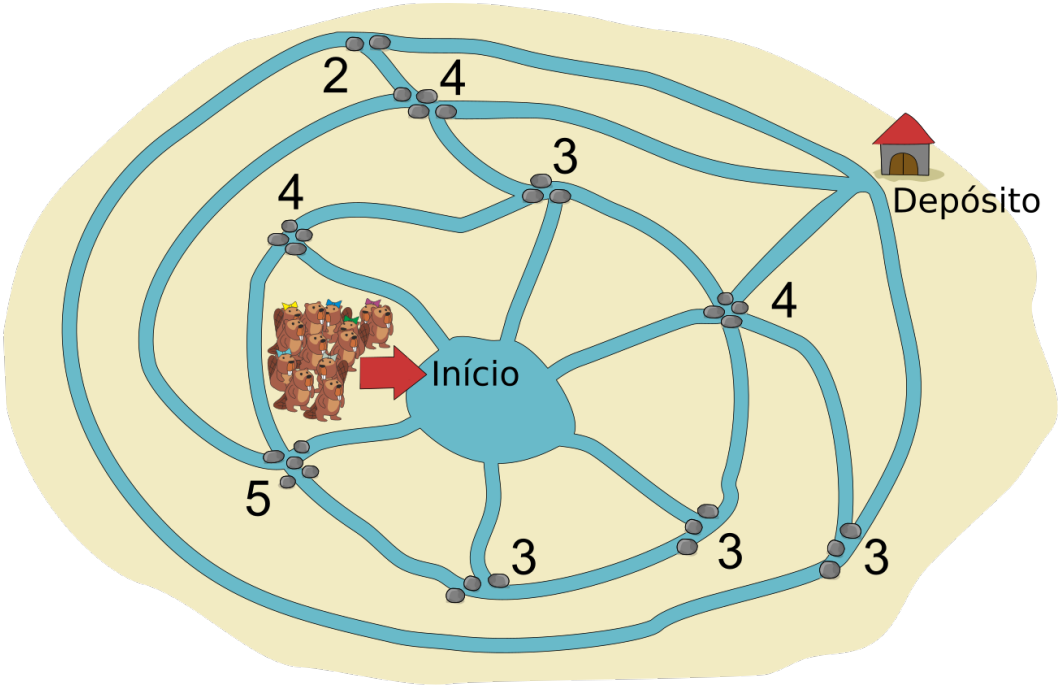
Isto é Pensamento Computacional!

Esta tarefa é um exemplo de *Problema do Empacotamento*. O empacotamento é um problema prático importante - por exemplo, carregar o maior número possível de caixas num camião, ou marcar lugares de estacionamento para permitir o estacionamento do maior número possível de carros.

Este é um exemplo de um *problema de optimização* onde queremos encontrar um algoritmo que rapidamente encontrará a melhor solução. Infelizmente, pode ser muito difícil encontrar a melhor solução para o empacotamento de contentores quando o número de caixas é grande, mesmo que utilizemos computadores muito potentes. Em vez disso, podemos utilizar algoritmos que fornecem apenas soluções aproximadas, mas que são suficientemente bons em situações práticas.

14 – Recolhendo Pedras

O alojamento familiar do castor é composto por 21 canais. 31 pedras precisam de ser removidas. Os castores vão dar um mergulho, recolhem as pedras e levam-nas para o depósito.



As pedras são pesadas. Um castor consegue carregar apenas uma ou duas pedras ao mesmo tempo, mas não mais. Para ir de uma interseção para a próxima, os castores nadam, surpreendentemente, sempre exatamente uma hora. Depois de começarem ao mesmo tempo os castores têm de recolher todas as pedras dentro de quatro horas.

Pergunta

Quantos castores são precisos no mínimo?

Respostas Possíveis

(A) 14 castores

(B) 18 castores

(C) 20 castores

(D) 24 castores

14 – Recolhendo Pedras (Resolução)

Solução

(A)

Resolução

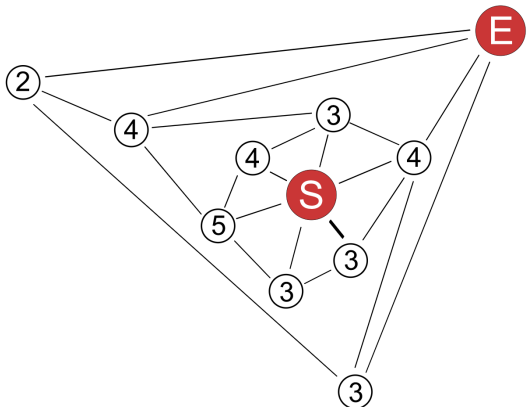
A resposta correta é a (A).

Versão curta:

Existe apenas um caminho que permite aos castores chegar ao depósito dentro de duas horas. Neste caminho existem 4 pedras. 2 castores podem carregar estas 4 pedras até ao depósito e podem continuar a carregar 4 pedras adicionais até ao depósito. Depois existem 23=31-8 pedras restantes e pelo menos mais 12 castores serão necessários, uma vez que as restantes pedras se encontram em caminhos de comprimento 3 ou 4 e dentro de 4 horas não há tempo restante para continuar a trazer mais depois de depositar as pedras. Podemos observar, ao inspecionar o alojamento dos castores, que as restantes 23 pedras podem ser levadas até ao depósito por 12 castores, 11 dos quais levam 2 pedras e 1 deles leva apenas uma.

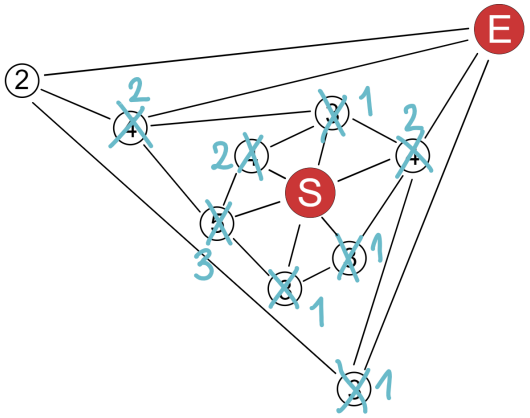
Segunda versão (mais pormenorizada):

No início, todas as pedras e o depósito dos castores estão em interseções do canal. Na imagem seguinte, representamos os canais por retas. Os números nas imagens representam o número de pedras na respetiva interseção.



Medimos o comprimento do caminho entre o início e o depósito contando o número de canais que os castores têm de atravessar a nado:

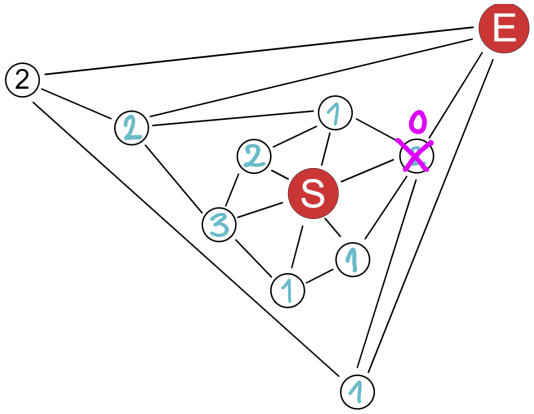
- Não há canais que conectam diretamente o início e o depósito (i.e., não há caminhos de 1 hora de comprimento).
- Há caminhos de comprimentos 2 horas, 3 horas e 4 horas.
- Os castores têm de recolher as pedras dentro de quatro horas no máximo. Os caminhos de mais de quatro horas não têm de ser considerados.



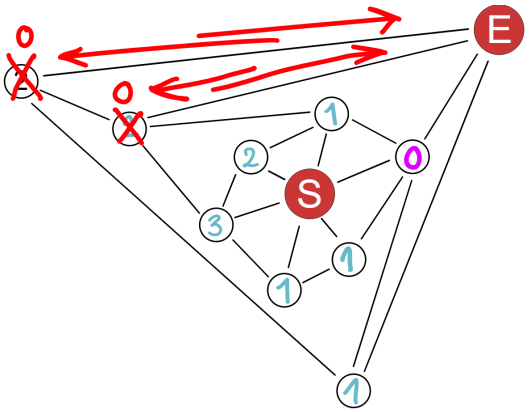
Primeiro, consideramos todas as interseções com pelo menos três pedras. Em cada interseção destas, um castor recolhe duas pedras e leva-as para o depósito. Para isto, precisamos de 8 castores. Eles colecionaram 16 pedras. 31-16=15 pedras restantes. Todos os castores exceto um estiveram pelo menos 3 horas a caminho. Eles não vão conseguir voltar aos canais para recolher mais pedras. Não podemos "reutilizá-los".

Um dos castores, no entanto, recolheu duas pedras dentro de duas horas. Ele teria tempo suficiente para voltar a uma interseção e recolher outras duas pedras e levá-las para o depósito. Assim, podemos "reutilizar"este castor mais tarde.

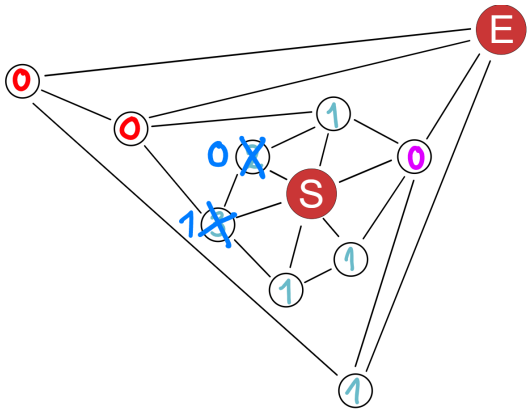
Um novo castor recolhe duas pedras dentro de duas horas. Também podemos "reutilizar"este castor. Sobram 15-2=13 pedras. Até agora, usámos 9 castores. Dois deles podem ser "reutilizados".



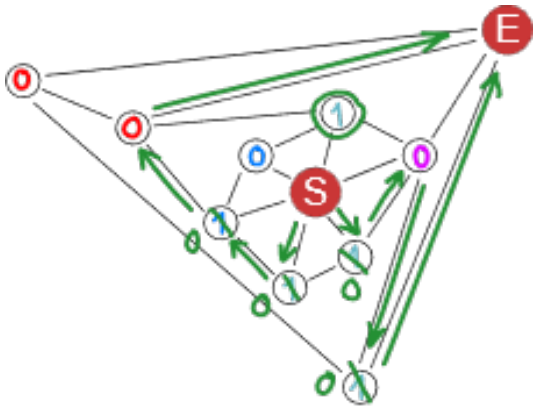
Agora, os dois castores podem ser "reutilizados". Cada um deles recolhe duas pedras. Ainda faltam 13-4=9 pedras. Até agora, usámos 9 castores. Nenhum deles pode ser "reutilizado" mais tarde.



Aplicamos a mesma estratégia e depois consideramos todas as interseções com pelo menos duas pedras. Precisamos de 2 castores. Eles recolhem 4 pedras. Há 9-4=5 pedras restantes. Até agora, usámos 9+2=11 castores.



O passo final consistem em encontrar uma solução (possivelmente por tentativa-erro) que utilize o mínimo número de castores possível. Dois castores recolhem duas pedras cada. Um terceiro e último castor recolhe a última pedra. A melhor solução é a (A) 11+3=14 castores.



Há 31 pedras, apenas 2 castores podem levar 2 pedras cada e depois voltar para recolher mais. Os outros castores podem levar no máximo 2. Portanto pelo menos 13.5 castores são necessários. A melhor solução requer 14 castores.

Isto é Pensamento Computacional!

Aos olhos de um leitor, a história é apelativa e rica de detalhes apelativos. No entanto, a primeira ação tomada por um informático consiste em identificar a informação de que realmente se precisa para resolver o problema e em remover tudo o resto. A informação significativa engloba:

- canais e interseções de canais (pedras, início e casa);
- limitações numéricas: um castor nada exatamente 1h de um interseção para a próxima; os castores precisam de estar em casa dentro de 4 horas; eles têm de recolher 31 pedras; eles podem carregar uma ou duas pedras;
- temos de recolher todas as 31 pedras usando o mínimo número de castores possível.

Por outro lado, não precisamos de considerar o percurso preciso dos canais e a sua posição exata no mapa, a aparência dos castores, etc. Esta informação pode ser omitida.

Os cientistas informáticos chamam a esta representação formal um grafo. Um grafo consiste em nós e arestas entre os nós. A distância entre dois nós pode ser expressa de várias maneiras, por exemplo, através da contagem do número de arestas que temos de atravessar para passar de um nó para outro. Os grafos são um dos principais instrumentos de resolução de problemas na vida de um cientista informático. Os sistemas de navegação são um exemplo bem conhecido de aplicações que fazem uso intensivo de grafos.

Na nossa tarefa, espera-se que os castores recolham todas as 31 pedras. Possivelmente, existe mais do que uma forma de alcançar este objetivo. Poder-se-ia imaginar utilizar 14, 18, 20 ou mesmo 31 castores diferentes. Contudo, a questão requer explicitamente uma solução que utilize o menor número possível de castores. Nesta perspectiva, devemos considerar cuidadosamente se existe uma forma de "reutilizar"um castor ou não.

Os problemas que procuram a melhor solução possível são muito comuns. Na terminologia da informática, são problemas de optimização. Espera-se que um sistema de navegação selecione a melhor (por exemplo, a forma óptima) para passar de A para B. É claro que o significado exato de "a melhor" solução precisa de ser definido a priori. No caso de um sistema de navegação, um critério poderia consistir em minimizar a distância em quilómetros, ou o tempo que o carro necessita para alcançar o alvo.

Em termos de pensamento computacional, os alunos provavelmente simplificarão ligeiramente a representação do problema dado - escreverão números em vez de desenhar pedras e representarão canais por meio de linhas. No entanto, a imagem já é uma forma informal mas precisa de um grafo. Os alunos são obrigados a decompor o problema em peças mais pequenas e geríveis, a resolver (possivelmente por tentativa-erro) cada uma delas, e a inferir se a sua solução satisfaz todos os constrangimentos e as exceções que potencialmente podem surgir.

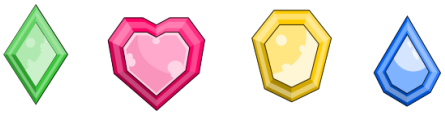
15 – Jóia Favorita

O Tiago tem uma coleção de jóias. Ele classifica as suas jóias desde as suas favoritas até às menos favoritas.

A Sara sabe que jóias estão na coleção do Tiago, mas não sabe como é que ele as classificou.

A Sara tem um plano para descobrir qual das jóias é a favorita do Tiago:

- A Sara escolhe quatro das jóias do Tiago e pergunta-lhe: "Deste grupo de quatro, qual é a tua jóia favorita?";
- A Sara escolhe um novo conjunto de quatro jóias e volta a fazer a mesma questão;
- Depois, escolhe um terceiro conjunto de quatro jóias e coloca a sua pergunta pela última vez.



Nota: quando a Sara escolhe o segundo e o terceiro conjunto de jóias, ela pode, por vezes, incluir jóias que já escolheu antes.

Pergunta

Se a Sara encontrar a jóia favorita do Tiago, qual é o maior número possível de jóias na coleção do Tiago?

Respostas Possíveis

(A) 8

(B) 10

(C) 11

(D) 12

15 – Jóia Favorita (Resolução)

Solução

(B)

Resolução

A resposta correta é a opção (B).

Com 10 jóias, a Sara pode perguntar ao Tiago sobre oito jóias diferentes com as duas primeiras perguntas. A resposta do Tiago a cada uma dessas perguntar é uma candidata à sua jóia favorita mas as restantes três não podem ser. Assim, com a terceira e última pergunta da Sara, ela pode incluir essas duas candidatas e duas outras jóias que ainda não fizeram parte de uma pergunta. A resposta do Tiago à terceira pergunta será a sua jóia favorita.

Mostrámos que existe uma estratégia da Sara que funciona se existirem 10 jóias (note-se que existem outras estratégias corretas que funcionam para 10 jóias).

Se o Tiago tiver pelo menos 11 jóias, consideramos as duas primeiras perguntas da Sara.

Se pelo menos uma jóia fizer parte de ambas as perguntas, então pelo menos quatro jóias serão excluídas das duas primeiras perguntas. Neste caso, a Sara deve perguntar sobre estas quatro jóias na terceira pergunta porque se ela não o fizer, corre o risco de uma das jóias ser a favorita do Tiago. Por outro lado, ela não terá nenhuma informação sobre a jóia preferida do Tiago entre as últimas quatro jóias em comparação com as outras 7 jóias. Neste caso, a estratégia dela não determina qual é a jóia favorita.

Se não existir nenhuma jóia que faz parte das duas primeiras perguntas da Sara, então as candidatas para jóia favorita do Tiago incluem as duas respostas a estas perguntas e as três restantes jóias rejeitadas. Isto dá um total de cinco possibilidades para jóia preferida do Tiago e a Sara não tem qualquer informação sobre como elas estão classificadas, por isso a sua estratégia não funciona.

Mostrámos que a estratégia da Sara não funciona se existirem mais do que 10 jóias.

Isto é Pensamento Computacional!

Para responder a esta tarefa, precisamos de encontrar a melhor estratégia que funcionará sempre para a Sara. A estratégia é um algoritmo que consiste numa sequência de passos que resolve o problema. Nesta tarefa avaliamos o algoritmo para encontrar o número máximo de jóias para o qual ele funciona. O algoritmo para encontrar o número máximo de jóias tem algumas restrições: 1. Encontrar a favorita entre quatro jóias. 2. Apenas três conjuntos de quatro podem ser avaliados.

Existem vários algoritmos que funcionam bem para algumas situações mas não para outras, por exemplo, um algoritmo de ordenação poderá ser capaz de ordenar números inteiros positivos mas não números inteiros negativos. Alguns algoritmos de procura de rotas podem encontrar a rota mais curta entre duas cidades, mas fazem-no demasiado lentamente quando aplicados a um mapa com milhares de cidades ligadas. Neste caso, são frequentemente escolhidos algoritmos mais rápidos que encontram uma das rotas mais curtas, mas não necessariamente a mais curta. Uma boa solução rápida é melhor do que não haver solução!

Em geral, há muitas formas de avaliar algoritmos programados para serem executados num computador. Por exemplo, geralmente, consideramos o tempo de execução de um algoritmo (quanto tempo esperamos que leve do início ao fim) ou a quantidade de espaço que um algoritmo utiliza, o que indica a quantidade de memória que o computador pode utilizar durante a sua execução.