



Castor Informático

O Desafio Internacional de Pensamento Computacional

EDIÇÃO 2022

CATEGORIA: **JUNIORES** (9^o e 10^o 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 – Os Vizinhos da Maria	4
Resolução	5
2 – Porcas e Parafusos	7
Resolução	8
3 – Miss Infinito	10
Resolução	11
4 – Barragens dos Castores	13
Resolução	14
5 – Morangos	15
Resolução	16
6 – Jogo do Galo	18
Resolução	19
7 – Feiticeiro	20
Resolução	21
8 – Aldeias Entrelaçadas	22
Resolução	23
9 – Velas Coloridas	24
Resolução	25
10 – Labirinto	26
Resolução	27
11 – Listas	28
Resolução	29
12 – Base de Dados dos Castores	30
Resolução	31
13 – Caminho de um Drone	32
Resolução	33
14 – AI dos Castores	34
Resolução	35
15 – Um Jogo de Corte e Rato	36
Resolução	37



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.



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.





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



CC BY-NC-SA 4.0 - <https://creativecommons.org/licenses/by-nc-sa/4.0/>

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

Os 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:

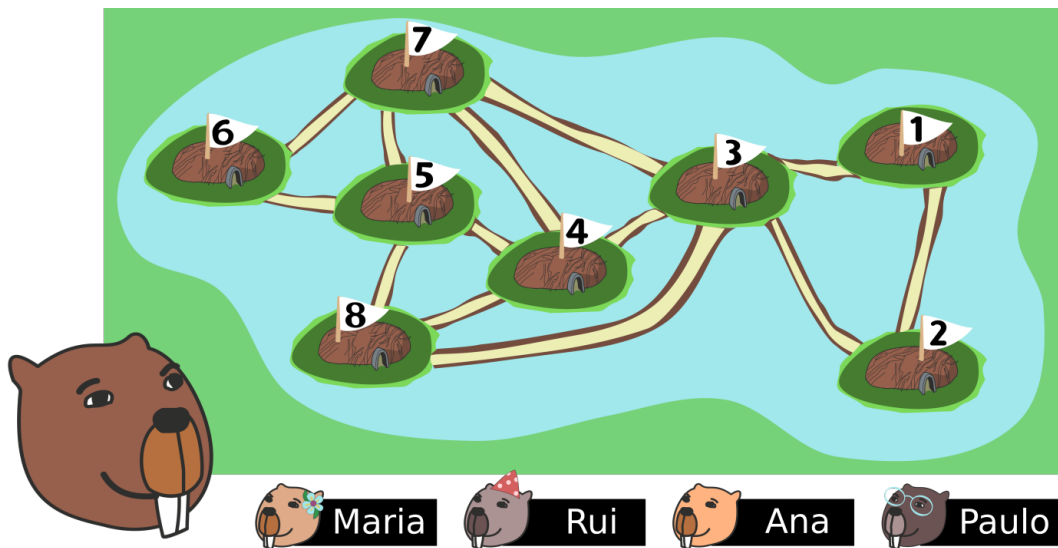
- Alemanha	- Austrália	- Áustria	- Brasil	- Canadá
- Chipre	- Eslováquia	- Filipinas	- Finlândia	- Hungria
- Irão	- Irlanda	- Itália	- Letónia	- Lituânia
- Macedónia	- Países Baixos	- Suíça	- Taiwan	- Uzbequistão
- Vietname				



1 – Os Vizinhos da Maria

O Bebras quer visitar a sua amiga Maria mas não sabe onde é que ela mora. Felizmente, ele tem um mapa e alguma informação. Dois castores são vizinhos se um caminho ligar as suas casas.

- Cada um dos três castores Maria, Rui e Paulo tem quatro vizinhos;
- O Rui e o Paulo são vizinhos da Ana;
- A Ana não tem mais nenhum vizinho.



Pergunta

Qual é o número da casa da Maria? (Escreve o número na folha de respostas)



1 – Os Vizinhos da Maria (Resolução)

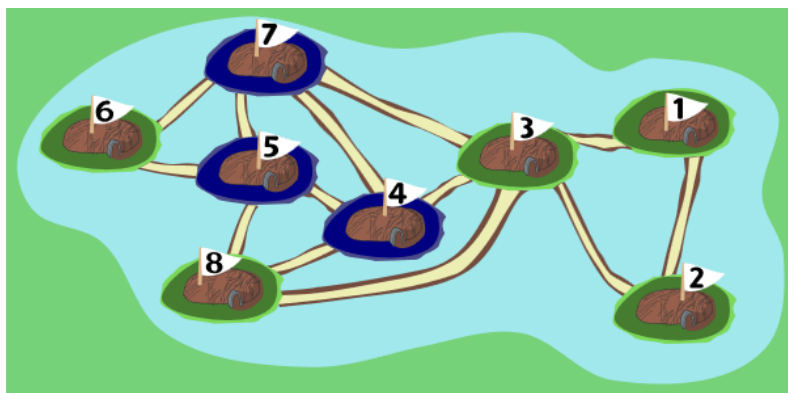
Solução

4

Resolução

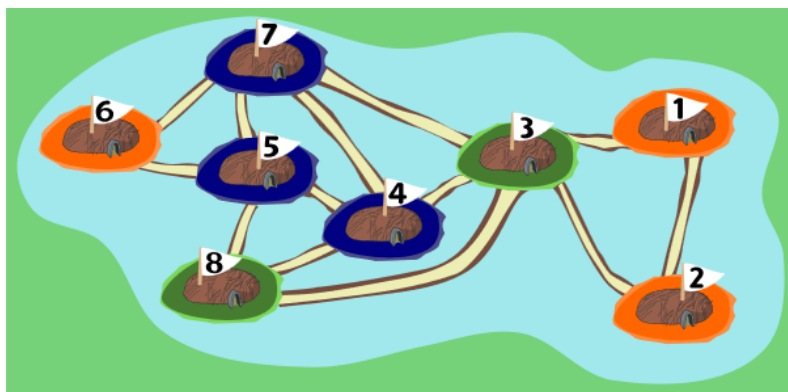
A resposta correta é o número 4.

Para resolver este problema é necessário focarmo-nos nos caminhos que ligam a cada casa. Precisamos primeiro de identificar as casas com quatro caminhos. Existem três dessas casas: 4, 5 e 7.



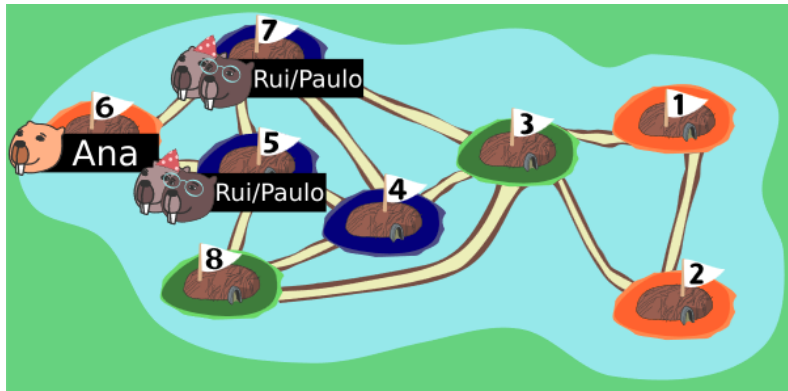
Assim, a Maria, o Rui e o Paulo vivem numa destas três casas, mas precisamos de descobrir onde vive exactamente a Maria.

As outras duas peças de informação descrevem a casa da Ana. Podemos deduzir que existem apenas dois caminhos a partir dela. Assim, a Ana vive numa das casas com os números 1, 2 ou 6.



Como a Ana é vizinha do Rui e do Paulo, podemos deduzir que:

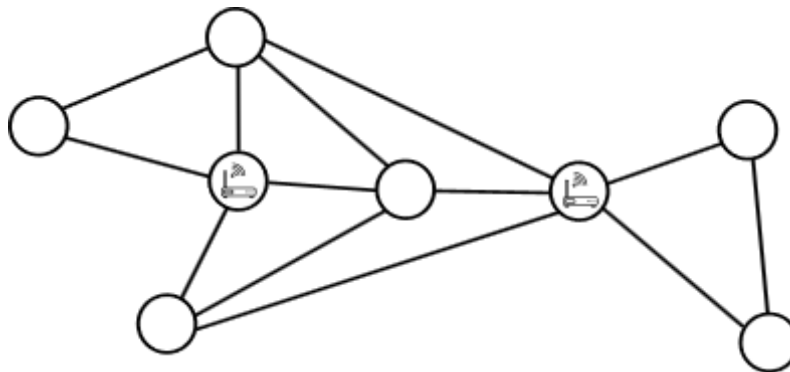
- A Ana mora na casa número 6;
- O Rui e o Paulo moram nos números 5 e 7 (ou ao contrário);



Assim, existe apenas uma casa com quatro caminhos que pode ser a casa da Maria e é a que tem o número 4!

Isto é Pensamento Computacional!

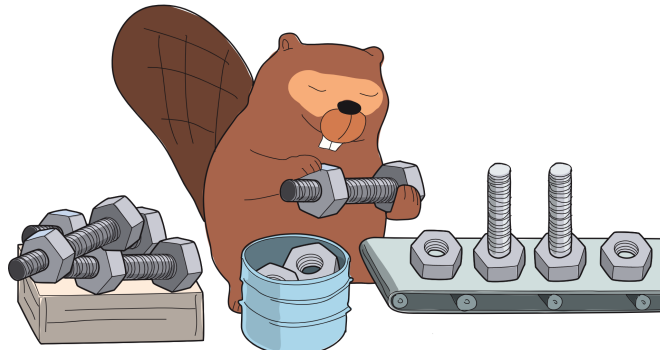
A teoria dos grafos é o estudo de redes utilizadas para modelar as relações em pares entre objectos. Um grafo pode ser visto como um conjunto de vértices (também chamados nós ou pontos) ligados com arestas (também chamados de ligações ou linhas). Nesta tarefa, as casas representam nós, e os caminhos representam arestas. Entre muitas outras aplicações, os grafos podem ser úteis ao descrever e resolver problemas de rede, tais como encontrar um bom ponto para um router num edifício ou certificar que cada uma das casas de um bairro tem um forte sinal Wi-Fi.





2 – 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.

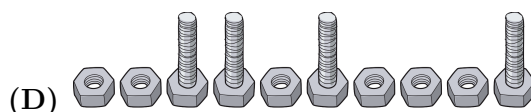
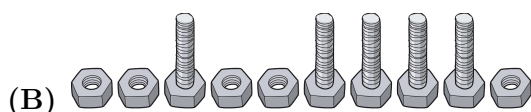
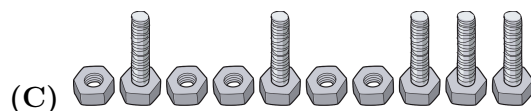
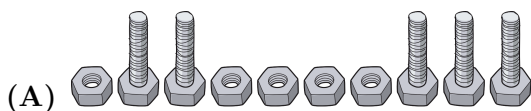
No entanto, as coisas podem correr mal para o Bruno de duas formas diferentes:

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?

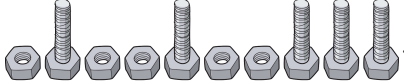
Respostas Possíveis





2 – Porcas e Parafusos (Resolução)

Solução


A resposta correta é a (C) .

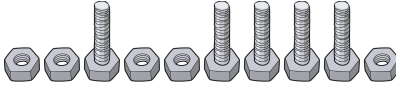
Resolução

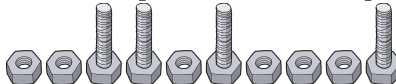
Podemos acompanhar o estado do balde e do tapete rolante da esquerda para a direita, com $N = \text{🍆}$ ("nuts") e $B = \text{🔩}$ ("bolts"). No caso da resposta (C), temos:

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 autômato 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 detecção 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.

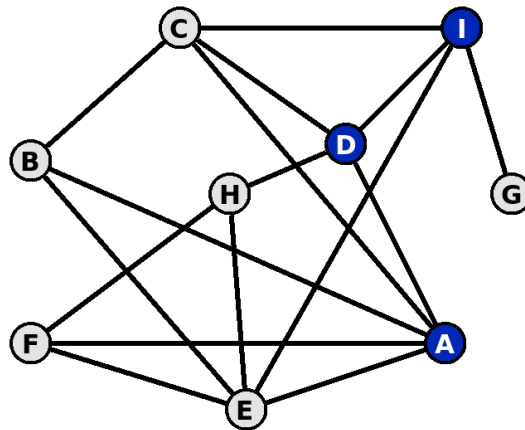


3 – Miss Infinito

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.



3 – Miss Infinito (Resolução)

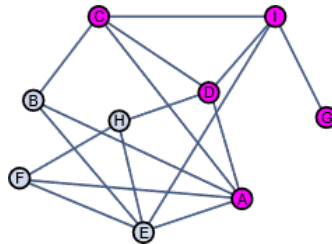
Solução

(D)

Resolução

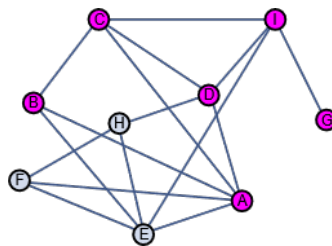
A resposta correta é a (D) Sexta-feira.

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

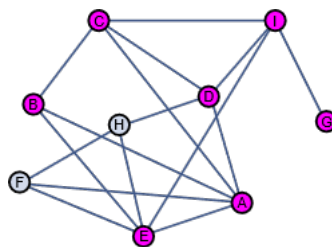


G fala apenas com um colega (I) e este começa a usar a alcunha na segunda-feira. C fala com 4 colegas e 3 deles usaram o nome na segunda-feira. 3 em 4 colegas é mais de metade. Assim, C também usa a alcunha na terça-feira.

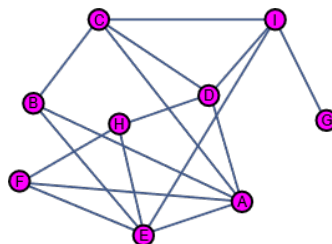
Na quarta-feira, B começa a usar a alcunha.



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!

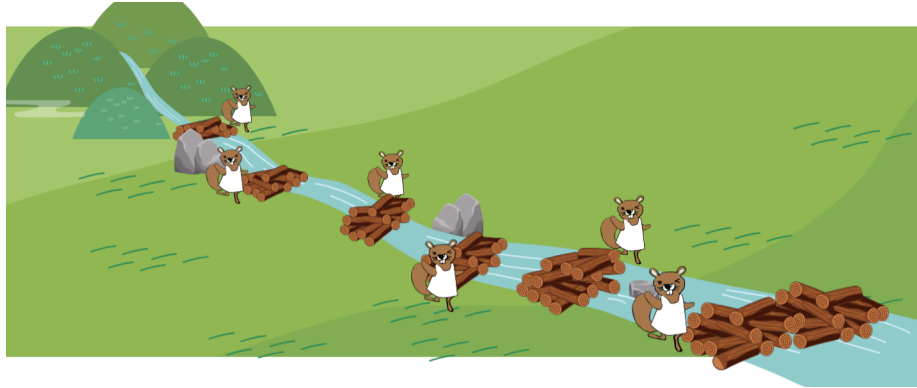
As redes sociais têm um papel fundamental na difusão de informação, infecções, ideias e influência entre os seus membros. Uma ideia ou inovação que aparece numa rede social correrá rapidamente em cascata através de interações sociais ou desaparecerá.

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.



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 uma 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)

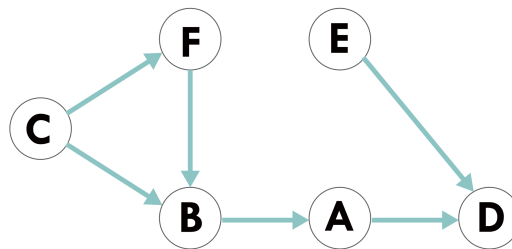
Solução

(D)

Resolução

A resposta correta é a (D) $E \rightarrow C \rightarrow F \rightarrow B \rightarrow A \rightarrow D$.

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 topológica 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*.



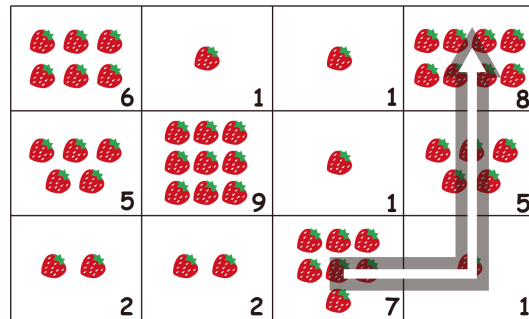
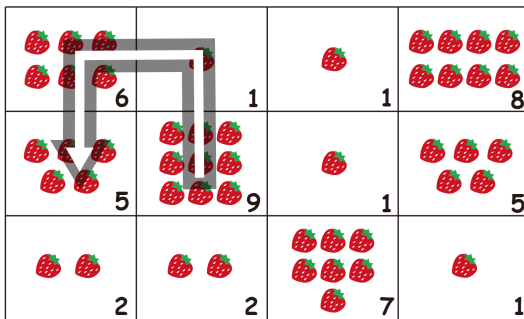
5 – Morangos

Os castores adoram jogar à “caça ao morango”!

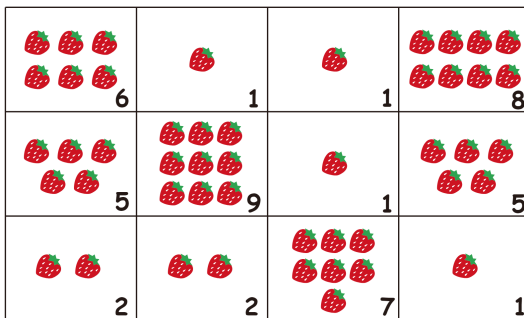
Neste jogo, um número diferente de morangos é colocado em cada célula de uma grelha. Um castor pode depois começar a partir de qualquer célula e dar 3 passos. Em cada passo, ele pode escolher ir para cima, para baixo, para a direita ou para a esquerda, para uma célula vizinha. No caminho escolhido, um castor visita quatro células e come todos os morangos nessas células.

Aqui estão dois exemplos para a mesma grelha:

- Escolhendo o caminho da esquerda, um castor comerá $9 + 1 + 6 + 5 = 21$ morangos.
- Escolhendo o caminho da direita, um castor comerá também $7 + 1 + 5 + 8 = 21$ morangos.



A pequena castora Sara está a jogar nesta grelha:



Pergunta

Quantos morangos é que a Sara pode comer no máximo? (Escreve um número na folha de respostas)



5 – Morangos (Resolução)

Solução

23

Resolução

A resposta correta é 23 morangos.

Embora existam 51 formas possíveis de terminar o jogo (ver tabela abaixo), não temos de verificar todas as 51 somas totais de morangos. Eis como.

Sabemos que a Sara consegue chegar a 4 células consecutivas e adjacentes. Essas 4 células apenas conseguem constituir as seguintes formas:

Forma	Quadrado	Linha	Forma em L	Forma em L horizontal	Forma em Z	Forma em Z vertical
Arranjo						
Número	6	3	12	16	8	6

Agora, verificamos estas formas incluindo qualquer uma das células com 9 ou 8 morangos (incluindo todas as formas com ambas as células). Uma vez que a verificação de combinações de células com 9 ou 8 morangos também incluirá as células com 5, 6 ou 7 morangos, todas as grelhas com maior número de morangos serão cobertas. Depois, podemos encontrar o maior número total de morangos:

	Número de Morangos que a Sara consegue comer num caminho		
	Incluindo a célula 9	Incluindo a célula 8	Maior dos dois
Quadrado	9+5+6+1=21	8+5+1+1=15	21
Linha	5+9+1+5=20	6+1+1+8=16	20
Forma em L 	7+2+9+1=19	8+5+1+7=21	21
Forma em L horizontal 	9+1+5+8=23	9+1+5+8=23	23
Forma em Z 	5+9+2+7=23	8+1+1+9=19	23
Forma em Z vertical 	6+5+9+2=22	8+5+1+7=21	22

Agora, note-se que sem as células com 8 e 9, a soma máxima de morangos em *qualquer* outra combinação de quatro células é $7 + 6 + 5 + 5 = 23$. Ou seja, também sem as células 8 e 9, o número 23 não pode ser ultrapassado. Assim, a Sara pode comer no máximo 23 morangos.

Isto é Pensamento Computacional!

Em ciência de computadores, muitos problemas têm como objetivo encontrar o valor máximo ou mínimo, tais como fazer o máximo de trabalho, gastar o mínimo de dinheiro, ou - como nesta tarefa - comer o número máximo de morangos. Nestes problemas, existem muitas opções para fazer o trabalho, gastar dinheiro ou comer morangos. Dentro destas opções, o desafio é encontrar aquelas com valores máximos ou mínimos; a essas opções chamamos solução ótima.

Normalmente, há muitas formas de encontrar uma solução ótima. Uma delas é listar todas as opções possíveis, calcular um a um os seus valores e escolher as que têm os melhores valores. Chamamos a este método pesquisa por força bruta ou pesquisa exaustiva. No entanto, este método pode demorar imenso tempo de computação se existirem muitas opções. Por este motivo, é útil analisar o problema e encontrar as restrições que podem reduzir o número de opções e encontrar a solução ótima mais rapidamente. Para muitos problemas, os cientistas de computadores conhecem métodos para encontrar a solução ótima de forma eficiente. Mas para outros, sabe-se que não podem ser resolvidos de forma eficiente.



6 – Jogo do Galo

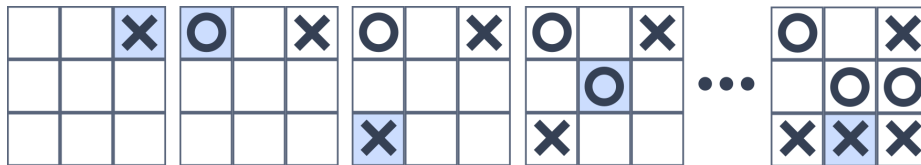
O Jogo do Galo é um jogo de papel e caneta para dois jogadores.

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):

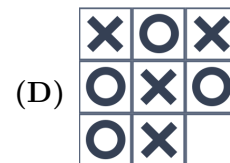
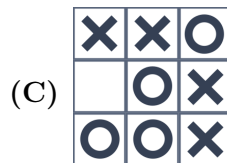
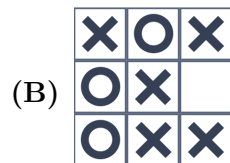
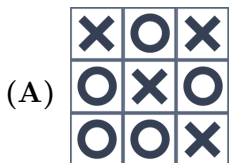


À imagem da direita chamamos folha de resultados de um jogo completo. Nem todas as folhas completadas aleatoriamente com **X** ou **O** são uma folha de resultados válida de acordo com as regras apresentadas anteriormente.

Pergunta

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

Respostas Possíveis





6 – 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.



7 – 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 transformações 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





7 – 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 com um 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 obtida 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 transformações 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.



8 – 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)



8 – 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.



9 – 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



9 – 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 incluem 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.

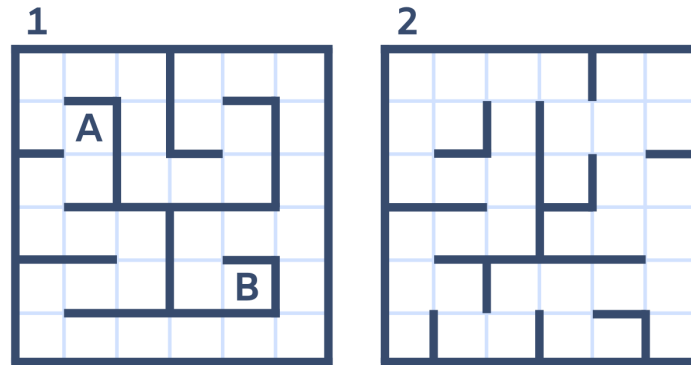
Representação: Nesta tarefa, os números são (não unicamente) representados por pares de cores (dizemos que os números são mapeados para pares de cores). Nesta tarefa devemos aplicar a representação continuamente a diferentes números para identificar os que são representados por um par de velas com a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



10 – 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



10 – Labirinto (Resolução)

Solução

(C)

Resolução

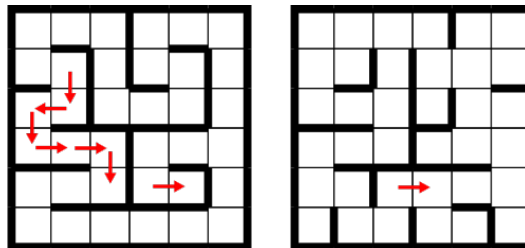
A resposta correta é a (C) 18.

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.

2	3	4	11	12	13
1	0	5	10	9	14
2	1	6	7	8	15
3	4	5	18	17	16
8	7	6	17	18	15
9	10	11	12	13	14

7	6	7	8	11	12
6	5	8	9	10	11
7	6	7	10	11	12
8	9	8	13	12	13
9	10	11	12	13	14
10	11	12	13	14	15

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!

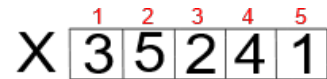
O problema do caminho mais curto é um problema bem estudado de teoria de grafos, e um dos métodos mais conhecidos para o resolver é o algoritmo de Dijkstra. As suas aplicações incluem por exemplo encontrar o caminho ótimo numa rede informática ou no mapa de uma cidade.

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.



11 – 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)



11 – 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.



12 – Base de Dados dos Castores

Uma dúzia de famílias vive na aldeia dos castores. O castor João criou a base de dados dos aldeões, registrando dados sobre cada castor na forma de sequências de 16-bits, desde o bit b_{15} (esquerda) até ao bit b_0 (direita), como se segue:

- b_{15} a b_{12} : quatro bits para o número da família;
- b_{11} : um bit para o género (0 = feminino, 1 = masculino);
- b_{10} a b_4 : sete bits para o peso (um número inteiro de quilogramas);
- b_3 e b_2 : dois bits para “trabalhador qualificado em” (00 = construção de alojamentos, 01 = construção de barragens, 10 = armazém de comida, 11 = educação de jovens castores);
- b_1 e b_0 : 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?

b_{11} e $\text{não}(b_{10})$ e b_9 e b_7 e $\text{não}(b_3$ e $b_2)$

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.



12 – Base de Dados dos Castores (Resolução)

Solução

(C)

Resolução

A resposta correta é a (C). Uma vez que $b_{11} = 1$ significa “macho”, $b_{10} = 0$ significa “pesar no máximo 63 kg”, $b_9 = 1$ e $b_7 = 1$ significa “pesar pelo menos (32 + 8) kg” e finalmente $b_3 = 0$ ou $b_2 = 0$ apenas exclui “trabalhador qualificado em educação de jovens castores” ($b_3 = 1$ e $b_2 = 1$).

Note-se que de acordo com as leis de De Morgan **não**(b_3 e b_2) é equivalente a **não**(b_3) **ou** **não**(b_2).

A resposta (A) está incorreta: $b_{11} = 1$ significa “macho” mas a resposta implica fêmeas. A expressão que denota o conjunto na resposta (A) é **não**(b_{11}) e b_8 e b_3 e **não**(b_2).

A resposta (B) está incorreta: $b_2 = 0$ também significa “trabalhador qualificado em armazém de comida”; além disso, $b_9 = 1$ e $b_7 = 1$ estabelece um limite máximo no peso. A expressão que denota o conjunto na resposta (B) é simplesmente b_{11} e b_{10} e **não**(b_3).

A resposta (D) está incorreta: $b_3 = 0$ também significa “trabalhador qualificado na construção de alojamentos”; além disso, $b_{10} = 0$ estabelece um limite superior no peso. A expressão que denota o conjunto na resposta (D) é b_{11} e **não**(b_{10}) e **não**(b_2) e b_2 .

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 operadores 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**.



13 – Caminho de um Drone

Um drone começa numa célula branca da grelha representada abaixo. Está virado para uma das quatro direções possíveis.

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

A seguir, visita exatamente 8 outras células brancas como se segue:

1. Move-se 2 células em frente.
2. Vira 90 graus para a esquerda (na sua célula atual).
3. Move-se 4 células em frente.
4. Vira 90 graus para a direita (na sua célula atual).
5. Move-se 2 células em frente.

Pergunta

Quantas células de começo possíveis existem? (Escreve um número na folha de respostas)



13 – Caminho de um Drone (Resolução)

Solução

8

Resolução

A resposta correta é 8. Para ver isto, notemos primeiro que existem 4 possibilidades para as células visitadas pelo drone:

	A	B	C	D	E	F	G	H	I
1	■		•	•	•	■			
2			•					■	
3		■	•		■				
4			•					■	
5	•	•	•		■				
6	■			■				■	
7									
8					■				
9			■			■			

	A	B	C	D	E	F	G	H	I
1	■				■				
2								■	
3		■			■				
4			•	•	•			■	
5			•		■				
6	■		•	■			■		
7			•						
8	•	•	•		■				
9			■			■			

	A	B	C	D	E	F	G	H	I
1	■				■				
2								■	
3		■			■	•	•	•	
4						•		■	
5					■	•			
6	■			■		•	■		
7				•	•	•			
8					■				
9			■			■			

	A	B	C	D	E	F	G	H	I
1	■				■				
2								■	
3		■			■				
4								■	
5			•		■				
6	■		•	■			■		
7			•	•	•	•	•		
8					■			•	
9			■			■		•	

Para encontrar estas possibilidades, é provavelmente mais fácil começar com uma pesquisa de cinco células brancas consecutivas e depois tentar estender os pontos finais em direções perpendiculares opostas. Para ter a certeza de que estás certo, deves fazer isto sistematicamente (e.g. da parte superior esquerda para a parte inferior direita da grelha).

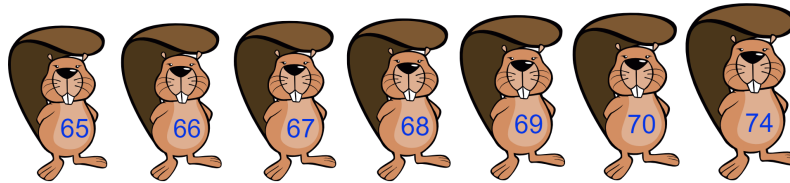
O caminho percorrido pelo drone é simétrico, pelo que o drone poderia começar a partir de qualquer um dos pontos finais de cada uma destas 4 possibilidades. Por conseguinte, existem 8 possíveis células de partida diferentes.

Isto é Pensamento Computacional!

A tarefa é um exemplo simples de correspondência invariante de imagem (ou modelo) de rotação, que é uma tarefa dentro do campo mais vasto da informática de processamento de imagem. Aqui o problema definiu o modelo (passos 1-5) a procurar, e a grelha de teste foi a imagem a partir da qual se pesquisou o modelo. A correspondência de modelos tem diversas aplicações e tem sido utilizada no reconhecimento facial e no processamento de imagens médicas. Em alguns dos métodos de correspondência do modelo, o padrão de pesquisa é muitas vezes descrito como um caminho.



14 – 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 \Rightarrow castor
- 11, 101, 110, 120, 130 \Rightarrow 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 \Rightarrow castor
- 86, 38 \Rightarrow não castor
- 40, 80 \Rightarrow 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



14 – 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>.

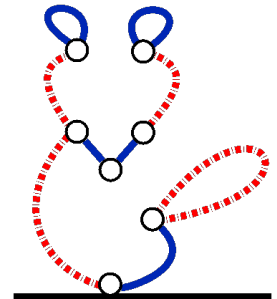


15 – Um Jogo de Corte e Rato

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.



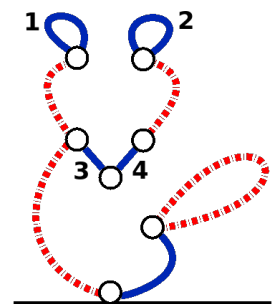
Uma possível sequência de jogadas é dada na tabela abaixo. São mostradas duas figuras por turno: a de cima marca o segmento que o jogador pretende cortar e a de baixo mostra o resultado desse corte.

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.



15 – Um Jogo de Corte e Rato (Resolução)

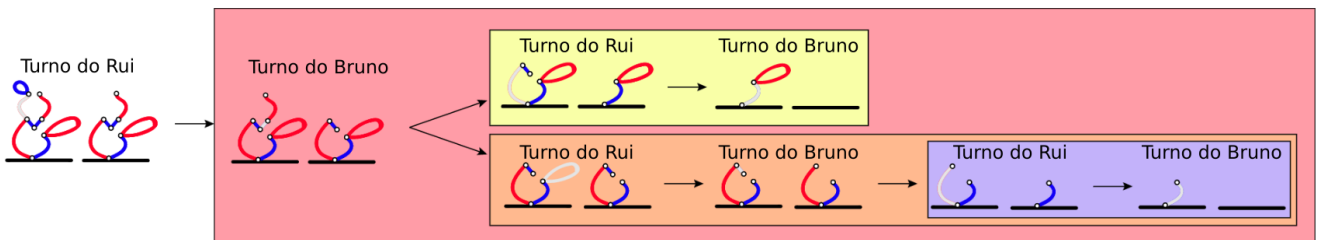
Solução

(B)

Resolução

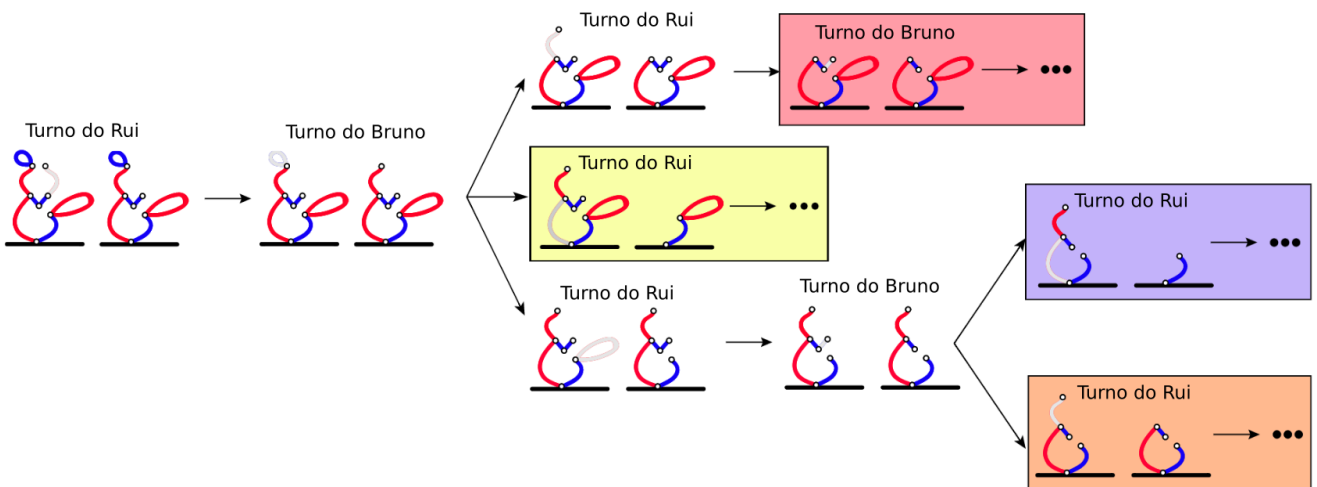
A resposta correta é a (B) Segmento 2.

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 a 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”.

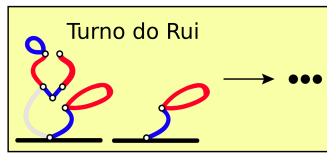


Para evitar jogar ao calhas, é vantajoso o Bruno *forçar* o Rui numa posição que faça parte da estratégia vencedora acima - uma vez que isto permite ao Bruno assegurar a vitória ao aplicar a mesma sequência de jogadas desta estratégia. Nos diagramas seguintes, estas posições são marcadas usando caixas 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.

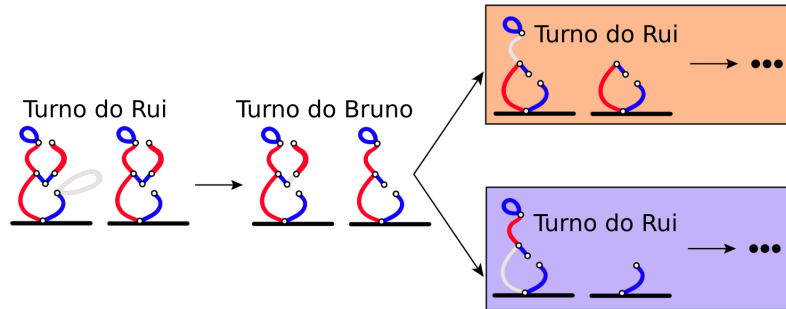
- O Rui corta a metade direita da “cara”



- O Rui corta a metade esquerda do “corpo”

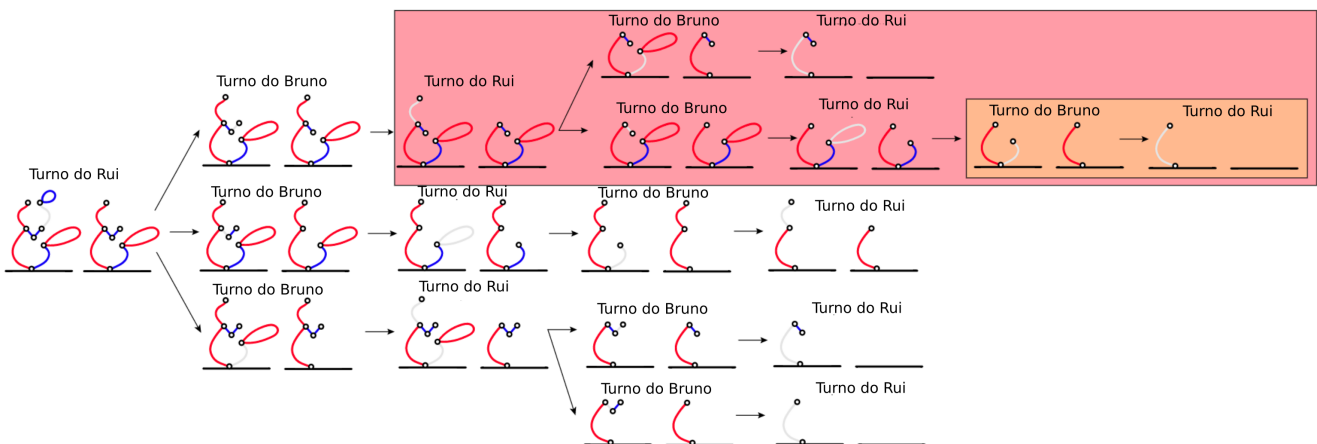


- O Rui corta a “cauda”

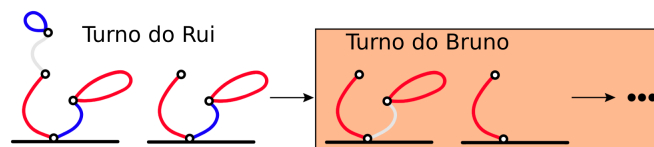


Para ficar mais completo, também podemos mostrar que o Bruno não tem hipótese de vencer se ele começar por cortar qualquer outra segmento e o Rui encontrar a forma perfeita de responder em cada turno.

- 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 influenciados 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.