



# XIII Olimpíada Cearense de Informática

2ª FASE - 31 de Outubro de 2025

## MODALIDADE PROGRAMAÇÃO

### **Leia atentamente as instruções:**

- Não serão permitidos empréstimos de materiais, consultas e comunicação entre os candidatos, tampouco o uso de livros e apontamentos. Relógios e aparelhos eletrônicos em geral deverão ser desligados. O não cumprimento destas exigências ocasionará a exclusão do candidato deste Exame;
- Aguarde o Aplicador da Prova autorizar a abertura do Caderno de Prova. Após a autorização, confira todas as questões antes de iniciar o Exame;
- Este Caderno de Prova contém 20 (vinte) questões objetivas, cada qual com apenas 1 (uma) alternativa correta;
- Não serão permitidas perguntas ao Aplicador da Prova sobre as questões da Prova;
- A duração desta prova será de 2 (duas) horas;
- O tempo mínimo para ausentar-se definitivamente da sala é de 1 (uma) hora;
- Ao concluir a prova, permaneça em seu lugar e comunique ao Aplicador de Prova, sinalizando com uma de suas mãos;
- Aguarde autorização para devolver o Caderno de Prova.
- Em questões de pseudocódigos envolvendo estruturas de dados como vetores, caso não seja especificado na questão, assuma indexação iniciada em 0.

## 2 | XIII Olimpíada Cearense de Informática

**Questão 1.** Uma sequência de números [5, 8, 10, 3, 2] é processada da seguinte forma:

1. Os números são inseridos, nesta ordem, em uma Fila (Queue) chamada F.
2. Duas Pilhas (Stacks) vazias, Pilha A e Pilha B, são criadas.
3. Enquanto a Fila F não estiver vazia, um número X é removido dela:
  - Se X for par, ele é inserido na Pilha A.
  - Se X for ímpar, ele é inserido na Pilha B.
4. Finalmente, todos os números são removidos da Pilha B e adicionados à sequência de saída.
5. Imediatamente após, todos os números são removidos da Pilha A e adicionados ao final da mesma sequência de saída.

Qual é a sequência de números resultante?

- a) [5, 8, 10, 3, 2]
- b) [2, 10, 8, 3, 5]
- c) [5, 3, 8, 10, 2]
- d) [2, 3, 5, 8, 10]
- e) [3, 5, 2, 10, 8]

---

**Questão 2.** Um garoto muito curioso chamado Leleudo tinha N copos de água completamente cheios, C baldes secos com capacidade máxima M. Ele tentou criar uma função que, dado um vetor com N entradas que representa o volume de cada copo, retorna para Leleudo se é possível despejar todos os N copos, em ordem, usando no máximo os C baldes, obedecendo às regras:

- Cada balde só pode receber um conjunto de copos consecutivos do vetor.
- A soma dos volumes em um balde não pode ultrapassar M.
- Só é possível despejar um copo dentro de um balde se, quando ele for colocado no balde, este não derrame.

```
func EncherBaldes(n, c, m, copos):  
    se n == 0:  
        retorne verdadeiro  
    baldes_usados = 1  
    soma_atual = 0  
    para i de 0 até N-1:  
        se(soma_atual + copos[i] <= M):  
            soma_atual = soma_atual + copos[i]  
        senao:  
            baldes_usados = baldes_usados + 1  
            soma_atual = copos[i]  
    se baldes_usados <= C:  
        retorne verdadeiro  
    senao:  
        retorne falso
```

### 3 | XIII Olimpíada Cearense de Informática

Leleudo sabe que o código não funciona, mas não sabe o motivo, por isso, perguntou a você qual o problema deste código. Qual resposta deve ser dada a Leleudo?

- a) A falha está na inicialização, *baldes\_usados* deveria começar em 0. Como começa em 1, o código sempre retorna falso para casos que usam exatamente C baldes, pois conta um a mais.
- b) O erro está na linha *soma\_atual = copos[i]*. O código assume que um novo copo sempre cabe em um balde novo, mas ele nunca verifica se *copos[i]* (sozinho) é maior que M, o que faz com que ele inicie posteriormente a variável *novo\_balde* com um valor incorreto.
- c) A estratégia gulosa não é correta, visto que é possível que a estratégia correta pule um copo, o que não é o que acontece.
- d) O algoritmo falha se C for maior que N, já que é necessário que todos os baldes sejam usados.
- e) Na realidade, Leleudo não percebeu que a falha dele é de eficiência, não de lógica, visto que se *baldes\_usados* ultrapassar o valor de C, o código ainda rodará por todos os outros copos, o que pode fazer com que demore muito para o computador entregar a resposta e, por isso, ele pensou que não teria resposta e seu código tivesse errado.

---

**Questão 3.** Um grupo de 5 marinheiros, que possuem cargos diferentes ( $A > B > C > D > E$ ), recebeu uma recompensa de 20 moedas. Eles precisam repartir essa recompensa entre eles a partir de uma votação que acontecerá do seguinte modo:

- O atual marinheiro com maior cargo deve propor uma repartição das moedas entre os marinheiros que ainda estão participando, por exemplo, o A pode propor (10,5,3,1,1).
- Depois da proposta, ocorre uma votação na qual cada marinheiro restante pode votar SIM ou NÃO.
- Caso tenha tido pelo menos 50% dos votos ( pode ser exatamente 50%) SIM então essa proposta é válida e é assim que a repartição é feita.
- Caso contrário, o atual líder fica com má fama e é retirado da discussão, de modo que ele não participará de nenhuma das próximas votações e não receberá nenhuma moeda.

Supondo que todos os marinheiros votem da forma mais lógica possível e que ele fique com o máximo de moedas ao final de todas as votações, qual deve ser a proposta de A que garante que fique com mais pontos possível?

- a) (20,0,0,0,0)
- b) (19,0,1,0,0)
- c) (18,0,0,1,1)
- d) (18,1,0,1,0)
- e) (18,0,1,0,1)

## 4 | XIII Olimpíada Cearense de Informática

**Questão 4.** Analise o seguinte pseudocódigo e, com base nele, marque a alternativa correta.

```
func misteriosa(vetor, n, s):  
    se s==0:  
        retorne verdadeiro  
    se n==0:  
        retorne falso  
    se vetor[n-1] > s:  
        retorne misteriosa(vetor, n-1, s)  
    retorne misteriosa(vetor, n-1, s) OU misteriosa(vetor, n-1, s-vetor[n-1])
```

- a) Essa função apenas retorna verdadeiro se o valor  $s$  pertence ao vetor  $v$  e a complexidade associada é  $o(n)$ .
  - b) A função retorna verdadeiro se o valor  $s$  pode ser obtido como a soma de alguns elementos do vetor  $v$ . A complexidade associada é  $o(2^n)$ .
  - c) Como em cada chamada da função o valor de  $n$  diminui em 1, a complexidade é  $o(n)$  e ela busca se o último elemento tem o valor  $s$ .
  - d) Essa função busca se algum elemento é igual a 0 e a complexidade é de  $o(2^n)$ .
  - e) A função retorna verdadeiro se o valor  $s$  pode ser obtido como a soma de alguns elementos do vetor  $v$ . A complexidade associada é de  $o(n)$ .
- 

**Questão 5.** Milena e seus amigos adoram brincar de jogos de cartas utilizando um baralho comum. Duas brincadeiras específicas lhe chamaram atenção por serem situações interessantes de se programar em um algoritmo simples.

No primeiro jogo, 9 cartas são distribuídas para cada jogador, enquanto na mesa resta um bolo de cartas e um local de descarte. Na sua vez, cada jogador pode escolher entre duas ações: puxar a primeira carta do bolo de cartas cegamente e descartar uma carta de sua escolha da sua mão de 10 cartas, ou puxar a última carta descartada no local de descarte (a carta descartada pelo jogador anterior). Note que a segunda ação só é possível a partir da segunda jogada. O objetivo do jogo é formar exatamente três trios de cartas iguais antes dos demais jogadores.

No segundo jogo, primeiramente, Milena embaralha as 52 cartas do baralho e as posiciona na mesa. Cada jogador então repete a seguinte operação até que reste apenas uma carta no baralho: descarta-se a primeira carta, e a segunda carta é colocada na última posição do bolo de cartas. Deseja-se apenas ver qual a última carta restante após essa sequência de operações. Após cada participante jogar uma vez, o jogador com a maior carta vence.

Tendo em vista os dois cenários e o seu interesse em tentar implementar em código cada um, assinale a alternativa que descreve corretamente as estruturas de dados mais adequadas para cada situação.

- a) A melhor opção é fazer uso apenas de pilhas, estruturas de dados em que o último elemento a ser inserido é também o primeiro elemento a ser retirado.
  - b) A melhor opção é fazer uso apenas de vetores comuns, estruturas de dados genéricas que armazenam uma sequência de dados
- .

## 5 | XIII Olimpíada Cearense de Informática

- c) A melhor opção é fazer uso apenas de filas, estruturas de dados em que o último elemento a ser inserido é também o último elemento a ser retirado, enquanto o primeiro a ser retirado é o primeiro a ser inserido.
- d) Dada a natureza diferente dos dois jogos, em um deles o uso de pilhas é adequado, enquanto o uso de filas é o mais adequado para o outro.
- e) É necessário uma estrutura de dados mais avançada, como uma árvore binária de busca.

---

**Questão 6.** Um laboratório de engenharia registra a taxa de desgaste em micrômetros em 6 pontos de teste distintos em um novo material. Esses dados, cruciais para a análise de durabilidade, são armazenados em um vetor  $D$  (indexado de 1 a  $n$ , em que  $n$  é o tamanho do vetor), onde cada posição corresponde a um ponto de teste. O vetor  $D$  é então processado por uma função recursiva  $Y$ , mas a finalidade exata da métrica retornada pelo algoritmo está sob análise e precisa ser identificada.

```
func Y(A[], p, r):  
    m = ⌊(p + r)/2⌋  
    se p = r então:  
        Retorna p  
    u = Y (A, p, m)  
    v = Y (A, m + 1, r)  
    se A[u] < A[v] então:  
        Retorna u  
    senão:  
        Retorna v
```

Considerando a execução do Algoritmo  $Y$  com as taxas de desgaste  $D = [15, 8, 20, 4, 12, 8]$  com parâmetros  $p=1$  e  $r=6$ . Encontre sucintamente a funcionalidade do Algoritmo  $Y$  e seu retorno.

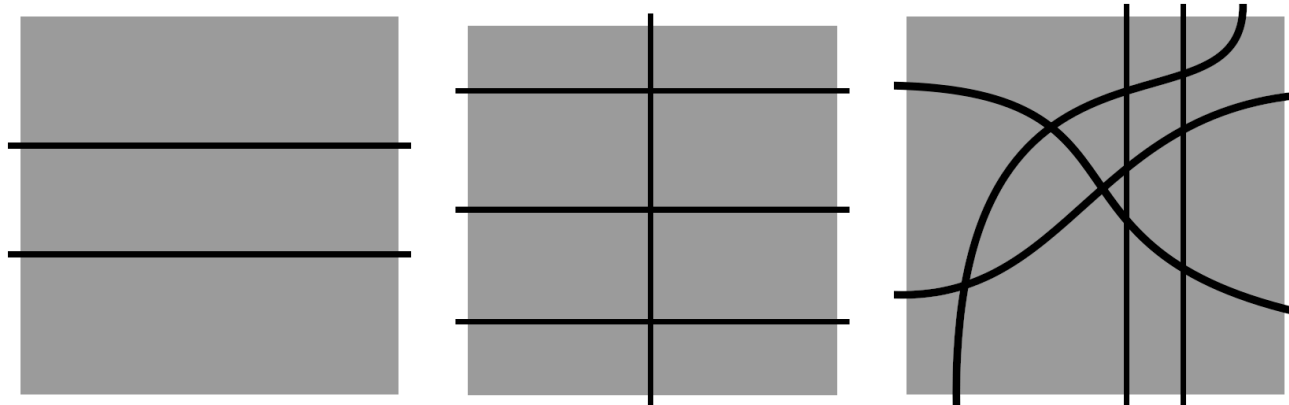
- a) Encontrar o elemento que está no meio exato da sequência, funcionando para vetores pares ao assumir o menor valor entre o encontro das duas metades e o retorno é 4.
- b) Encontrar o índice do elemento máximo do vetor, pois a comparação final ( $A[u] < A[v]$ ) está invertida e ele seleciona o maior índice de uma sub-árvore de máximo, sendo o retorno 3.
- c) Encontrar o índice do elemento mínimo em um vetor (ou subvetor), onde o índice retornado corresponde à posição do menor valor. O retorno será 4.
- d) Encontrar o índice da primeira ocorrência do primeiro elemento do vetor, ignorando os passos recursivos complexos e retornando o último índice verificado no caso base da última partição à direita, sendo o índice retornado 6.
- e) Encontrar o valor do elemento mínimo do vetor, e não seu índice, pois o retorno de  $p$  ou  $r$  no caso base é posteriormente substituído pelo valor da posição  $A[u]$  ou  $A[v]$  na comparação final. O índice retornado é 4.

## 6 | XIII Olimpíada Cearense de Informática

**Questão 7.** Um garoto divide um quadrado cinza em  $N$  pedaços utilizando linhas pretas não necessariamente retas e seguindo as regras abaixo:

- Toda linha parte de um dos quatro lados que formam o quadrado e termina no lado oposto.
- Cada linha passa por exatamente duas arestas.
- Cada linha passa por uma aresta uma única vez.
- Não pode haver um ponto onde se encontram uma aresta do quadrado e duas linhas desenhadas pelo menino simultaneamente.
- Jamais uma linha intersecciona um dos vértices do quadrado.
- Por um ponto dentro do quadrado pode passar no máximo duas das linhas desenhadas pelo menino.
- Duas linhas podem se interseccionar dentro do quadrado no máximo uma vez.

Por exemplo, ele pode dividir o quadrado das seguintes maneiras:



Sendo  $L$  o número de linhas desenhadas pelo menino e  $P$  o número de pontos em que há intersecção entre duas linhas, que fórmula pode representar o número de pedaços em que o quadrado foi dividido?

- a)  $N = L * P$
- b)  $N = L + P$
- c)  $N = L + P - 1$
- d)  $N = L + P + 1$
- e)  $N = 2*L + P$

## 7 | XIII Olimpíada Cearense de Informática

**Questão 8.** Imagine que você é o secretário de uma clínica odontológica. Certo dia, você não conseguiu ir ao trabalho e um colega secretário assumiu seu posto durante aquele expediente. Curiosamente, naquele dia, inúmeros pacientes marcaram uma consulta com a doutora K, de modo que a agenda da doutora ficou lotada pelos próximos 30 dias. Entretanto, você notou que o secretário substituto não levou em conta a possibilidade de uma consulta possuir um horário conflitante com outra na hora de marcá-las, isso resultou em várias consultas conflitantes em todos os 30 dias cobertos pelo secretário. Tendo em vista a natureza exaustiva do trabalho de consertar o deslize do seu colega, você decidiu fazer um algoritmo para resolver o seguinte problema: dada uma lista de consultas possivelmente conflitantes em um dia, qual o maior número de consultas que podem ser atribuídas à doutora K de modo que não haja conflitos de horários? Considere que cada consulta consiste essencialmente em um horário de início e um horário de término.

Segue uma proposta de abordagem para esse problema: busca-se o maior número possível de consultas em um único dia, portanto, deve-se olhar para a lista de consultas e procurar a consulta com o menor horário de término, inseri-la na solução, retirá-la da lista original e repetir o processo. A partir da segunda iteração do algoritmo, faz-se necessário checar se a consulta com o menor horário de término encontrada não conflita com alguma consulta já inserida na solução.

Sobre essa proposta de algoritmo, assinale a alternativa correta:

- a) Esse algoritmo funciona corretamente para alcançar o número máximo de consultas em um único dia, aplicando conceitos básicos de programação dinâmica.
- b) Esse algoritmo funciona corretamente para alcançar o número máximo de consultas em um único dia, empregando uma estratégia de divisão e conquista.
- c) Esse algoritmo funciona corretamente para alcançar o número máximo de consultas em um único dia, representando um bom exemplo de algoritmo guloso.
- d) Esse algoritmo funciona apenas para resolver o problema das consultas conflitantes, entretanto, ele não garante a quantidade máxima de consultas em um dia.
- e) Esse algoritmo não funciona corretamente para descobrir a quantidade máxima de consultas em um único dia, tampouco soluciona o problema das consultas conflitantes.

---

**Questão 9.** Carlos é um jovem que tem como um de seus passatempos favoritos buscar padrões matemáticos em operações diversas desmontando e remontando os objetos com os quais trabalha. O último desafio que esteve lhe encucando foi o seguinte: Dada uma figura de  $n \geq 2$  dimensões cujos lados medem  $(x + k_1)$ ,  $(x + k_2)$ , ...,  $(x + k_n)$  com a soma de  $k_1, k_2, \dots, k_n$  sendo igual a 0, o que pode ser feito para que o produto de seus lados seja maximizado, sabendo que  $k_1, k_2, \dots, k_n$  são números inteiros e que  $n$  e  $x$  são números inteiros positivos?

- a) A soma dos valores absolutos de  $k_1, k_2, \dots, k_n$  deve ser a maior possível.
- b) No caso de  $n$  ser par,  $k_1, k_2, \dots, k_n$  devem ser todos iguais a 1 ou -1. No caso de  $n$  ser ímpar,  $k_1, k_2, \dots, k_n$  deverão assumir somente os valores 1, -1 e 0.
- c) No caso de  $n$  ser par, os valores absolutos de  $k_1, k_2, \dots, k_n$  devem ser todos iguais a  $n/2$ .
- d) A soma dos valores absolutos de  $k_1, k_2, \dots, k_n$  deve ser igual a 0.
- e) Não há um valor máximo para esse produto.

## 8 | XIII Olimpíada Cearense de Informática

**Questão 10.** O “Merge” é um algoritmo de tempo linear ( $O(n)$ ) usado de forma auxiliar no famoso algoritmo de ordenação MergeSort. Sua função é intercalar dois vetores de tamanho  $n/2$  pré-ordenados em um só vetor de modo a manter esse novo vetor de tamanho  $n$  ordenado com apenas  $n$  operações.

Considere agora o problema de intercalar  $k$  vetores ordenados de tamanho  $n$  cada em um só vetor ordenado de tamanho  $kn$ .

Uma ideia é usar o Merge para intercalar o primeiro vetor com o segundo, em seguida, usar o Merge para intercalar o resultado do primeiro Merge com o terceiro vetor, e assim sucessivamente até o  $k$ -ésimo vetor.

Uma solução alternativa aplicando a técnica de Divisão e Conquista consiste em usar o Merge no primeiro e no segundo vetor, depois usar o Merge no terceiro e quarto vetor, e assim sucessivamente, até intercalar o  $k-1$ -ésimo vetor com o  $k$ -ésimo vetor. Após isso, teremos uma nova lista de vetores intercalados, agora cada um com tamanho  $2n$ , repete-se essa operação até que tenhamos apenas um vetor de tamanho  $kn$ .

Analise ambos os algoritmos e responda quanto a ordem de eficiência de cada um em função do tamanho inicial de cada vetor individual ( $n$ ) e da quantidade de vetores ( $k$ ).

- a) O primeiro algoritmo é mais eficiente, com complexidade  $O(nk)$ .
- b) O primeiro algoritmo é mais eficiente, com complexidade  $O(\log nk)$ .
- c) O segundo algoritmo é mais eficiente, com complexidade  $O(\log nk)$ .
- d) O segundo algoritmo é mais eficiente, com complexidade  $O(nk \log k)$ .
- e) Ambos os algoritmos realizam o mesmo número de operações, portanto têm a mesma complexidade.

---

**Questão 11.** Em álgebra linear, o produto escalar é uma operação realizada entre dois vetores que consiste em somarmos o produto entre as  $i$ -ésimas dimensões dos vetores. Por exemplo, para os vetores  $A=(2, 1, -2)$  e  $B=(4, 4, 2)$ , o produto escalar será o seguinte:  $(2*4)+(1*4)+(-2*2) = 8+4-4 = 8$ .

Por sua vez, obter a norma de um determinado vetor consiste em, dado um vetor  $A$ , obter a raiz quadrada do produto escalar de  $A$  com ele mesmo. Considerando o vetor  $A = (2, 1, -2)$  do exemplo anterior, a sua norma é igual a  $\sqrt{(2^2+1^2+(-2)^2)} = \sqrt{9} = 3$ .

Por fim, normalizar um vetor consiste em pegar cada valor de um vetor e dividir pela sua norma. Para o exemplo anterior, o vetor  $A$  normalizado seria  $A' = (\frac{2}{3}, \frac{1}{3}, -\frac{2}{3})$ .

Desejando-se construir um algoritmo para normalizar qualquer vetor, qual das seguintes alternativas realiza esta operação corretamente?

**Obs.:** A função  $\text{sqrt}(x)$  calcula a raiz quadrada de um número qualquer. Por exemplo,  $\text{sqrt}(9) = 3$

**Obs2.:** Os itens da questão 11 estão na página 9.



## 9 | XIII Olimpíada Cearense de Informática

**Entrada:** Vetor, tamanho\_vetor (considere um vetor com indexação de 1 a *tamanho\_vetor*)

**Saída:** Vetor normalizado

a) produto\_escalar = 0

para i = 1 até tamanho\_vetor:

produto\_escalar = produto\_escalar + (vetor[i] \* vetor[i])

norma = sqrt(produto\_escalar)

para i = 1 até tamanho\_vetor:

vetor[i] = vetor[i] / norma

retorna vetor

b) produto\_escalar = 0

para i = 1 até tamanho\_vetor:

produto\_escalar = produto\_escalar + (vetor[i] \* vetor[i])

para i = 1 até tamanho\_vetor:

vetor[i] = vetor[i] / produto\_escalar

retorna vetor

c) produto\_escalar = 0

para i = 1 até tamanho\_vetor:

produto\_escalar = produto\_escalar \* (vetor[i] + vetor[i])

norma = produto\_escalar \* produto\_escalar

para i = 1 até tamanho\_vetor:

vetor[i] = vetor[i] / norma

retorna vetor

d) produto\_escalar = 1

para i = 1 até tamanho\_vetor:

produto\_escalar = produto\_escalar + (vetor[i] \* vetor[i])

para i = 1 até tamanho\_vetor:

vetor[i] = vetor[i] / produto\_escalar

retorna vetor

e) produto\_escalar = 0

para i = 1 até tamanho\_vetor:

produto\_escalar = produto\_escalar + (vetor[i] \* vetor[i])

norma = sqrt(produto\_escalar)

para i = 1 até norma:

vetor[i] = vetor[i] / produto\_escalar

retorna vetor

## 10 | XIII Olimpíada Cearense de Informática

**Questão 12.** Considere o seguinte problema: dado um vetor  $V$  de números inteiros ordenados de forma crescente e um inteiro  $X$ , queremos descobrir se existem ou não dois inteiros  $x$  e  $y$  pertencentes a  $V$  que, somados, resultam em  $X$ . Analise os seguintes algoritmos propostos para resolver esse problema.

*OBS:  $n$  representa o tamanho do vetor  $V$  em ambas as funções, considere vetores indexados de 0 a  $n$ .*

```
func Algoritmo1(V, n, X):  
    para i = 0 até n-1:  
        inicio = i+1  
        fim = n-1  
        Enquanto inicio < fim:  
            meio = (inicio + fim)/2  
            se V[i] + V[meio] == X:  
                retorne "Existem dois números cuja soma é a desejada"  
            se V[i] + V[meio] > X:  
                fim = meio - 1  
            senão:  
                inicio = meio + 1  
    retorne "Não existem dois números cuja soma é a desejada"
```

```
func Algoritmo2(V, n, X):  
    i = 0  
    j = n-1  
    Enquanto i < j:  
        se V[i] + V[j] == X:  
            retorne "Existem dois números cuja soma é a desejada"  
        se V[i] + V[j] > X:  
            j = j - 1  
        senão:  
            i = i + 1  
    retorne "Não existem dois números cuja soma é a desejada"
```

Assinale a alternativa correta.

- a) Ambos os algoritmos funcionam perfeitamente para resolver o problema, possuindo um custo computacional próximo.
- b) Ambos os algoritmos funcionam perfeitamente para resolver o problema, sendo o Algoritmo1 o mais eficiente entre os dois.
- c) Ambos os algoritmos funcionam perfeitamente para resolver o problema, sendo o Algoritmo2 o mais eficiente entre os dois.
- d) Apenas um dos algoritmos resolve o problema corretamente.
- e) Nenhum dos algoritmos resolve o problema corretamente.

## 11 | XIII Olimpíada Cearense de Informática

**Questão 13.** Imagine que você está em um programa de auditório e o apresentador, que sabe o que há atrás de cada porta, lhe mostra três portas fechadas (Porta 1, Porta 2 e Porta 3). Atrás de uma das portas há um carro de luxo (o prêmio). Atrás das outras duas portas há mini-cabras (o prêmio de consolação). O jogo funciona da seguinte forma:

1. Você deve escolher uma porta. Você escolhe a Porta 1.
2. O apresentador, que sabe onde o carro está, abre uma das *outras* portas. Ele abre a Porta 3 e revela que há uma mini-cabra atrás dela.
3. Agora, restam duas portas fechadas: a sua escolha original (Porta 1) e a outra porta (Porta 2).
4. O apresentador lhe oferece a chance de mudar sua escolha: "Você quer manter sua escolha na Porta 1 ou quer trocar para a Porta 2?".

Para maximizar suas chances de ganhar o carro, qual é a decisão correta?

- a) Manter a escolha na Porta 1.
- b) Mudar a escolha para a Porta 2.
- c) Tanto faz. Após o apresentador abrir a Porta 3, a chance para a Porta 1 e a Porta 2 é a mesma (50/50).
- d) A probabilidade é impossível de calcular, pois depende da sorte.
- e) Nesse caso, a escolha é irrelevante dado que o apresentador sabe o que há atrás de cada porta. Caso o apresentador não soubesse, seria mais vantajoso mudar a escolha para a Porta 2.

---

**Questão 14.** Existem 100 lâmpadas em um corredor, numeradas de 1 a 100, todas inicialmente apagadas, 100 pessoas passam pelo corredor:

- A Pessoa 1 aperta o interruptor de todas as lâmpadas (1, 2, 3, ...).
- A Pessoa 2 aperta o interruptor de todas as lâmpadas pares (2, 4, 6, ...).
- A Pessoa 3 aperta o interruptor de todas as lâmpadas múltiplas de 3 (3, 6, 9, ...).
- ...
- A Pessoa 100 aperta o interruptor da lâmpada 100.

Ao final disso tudo, quais lâmpadas estarão acesas?

- a) Todas as lâmpadas com número primo.
- b) Todas as lâmpadas com número ímpar.
- c) Todas as lâmpadas com número par.
- d) Todas as lâmpadas cujo número é um quadrado perfeito.
- e) Nenhuma das alternativas anteriores.

---

**Questão 15.** Numa cidade, 85% dos táxis são da companhia Verde e 15% são da companhia Azul. Ocorreu um atropelamento à noite, e uma testemunha identificou o táxi como Azul. Testes mostraram que a testemunha consegue identificar a cor correta em 80% das vezes e erra 20% das vezes. Qual é a probabilidade de o táxi ser *realmente* Azul?

- a) 80%
- b) 15%
- c) 50%
- d) 41%
- e) É impossível ter certeza da probabilidade apenas com essas informações.

## 12 | XIII Olimpíada Cearense de Informática

**Questão 16.** Considere o seguinte problema: dada uma pilha de tapiocas numeradas de acordo com o seu tamanho, queremos ordenar essa pilha de modo a posicionar as maiores tapiocas na base da pilha (considere aqui que a pilha de tapiocas é representada por um vetor de inteiros, indexado de 1 a  $n$ , onde a base da pilha é o fim do vetor). Dentro desse problema, apenas uma operação é permitida, a função `VirarTapiocas`, que consiste em inserir uma espátula em baixo de uma tapioca do meio da pilha e inverter a ordem das tapiocas na subpilha acima da espátula.

```
func VirarTapiocas(V, k):  
    inicio = 1  
    fim = k  
    enquanto inicio < fim:  
        temp = V[inicio]  
        V[inicio] = V[fim]  
        V[fim] = temp  
        inicio = inicio + 1  
        fim = fim - 1
```

Usando essa função, é possível criar o `TapiocaSort`, um algoritmo de ordenação baseado na operação de virar tapiocas. A função auxiliar `AcharIndiceMax(V, n)` retorna o índice do maior elemento do vetor `V` até a posição  $n$ .

```
func TapiocaSort(V, n):  
    para tamanho_atual = n até 2 passo -1:  
        indice_max = AcharIndiceMax(V, tamanho_atual)  
        se indice_max != tamanho_atual:  
            VirarTapiocas(V, tamanho_atual)
```

De acordo com as definições dadas e os algoritmos fornecidos, assinale a alternativa correta sobre essa implementação do `TapiocaSort`.

- a) O `TapiocaSort` funciona corretamente com ordem de complexidade linear  $O(n)$ .
- b) O `TapiocaSort` funciona corretamente com ordem de complexidade quadrática  $O(n^2)$ .
- c) O `TapiocaSort` funciona corretamente com ordem de complexidade logarítmica  $O(\log n)$ .
- d) O `TapiocaSort` funciona corretamente com ordem de complexidade cúbica  $O(n^3)$ .
- e) O `TapiocaSort` implementado não funciona corretamente para ordenar as tapiocas na pilha.

### 13 | XIII Olimpíada Cearense de Informática

**Questão 17.** Dois jogadores, Alice e Beto, disputam um jogo com uma pilha que, inicialmente, contém  $N$  moedas. As regras são as seguintes:

1. Alice começa jogando.
2. Os jogadores se revezam, e a cada turno, um jogador pode remover 1 ou 2 moedas da pilha.
3. O jogador que retirar a última moeda da pilha vence o jogo.

Exemplos:

- Se  $N = 1$  ou  $N = 2$ , Alice pode remover todas as moedas em seu primeiro turno e ganhar imediatamente.
- Se  $N = 3$ , a situação muda. Se Alice remover 1 moeda, restarão 2 para Beto. Se Alice remover 2, restará 1. Em ambos os cenários, Beto pode retirar todas as moedas restantes em seu turno e vencer. Portanto, para  $N = 3$ , Beto possui uma estratégia vencedora garantida.

Considerando as regras do jogo, para qual dos seguintes valores de  $N$ , Beto tem a estratégia vencedora, ou seja, pode garantir a vitória independentemente das jogadas de Alice?

- a) 11
- b) 57
- c) 16
- d) 32
- e) 47

---

**Questão 18.** A UFC deseja fazer a manutenção do cabeamento de rede que interliga os campi e que interliga cada departamento da universidade. Cada departamento é descrito como um ponto e uma ligação direta entre os departamentos é designada como uma linha. Só podem haver linhas entre dois pontos, portanto, não podem haver linhas que comecem em pontos e terminam em outra linha ou linhas que só se ligam a outras linhas. A universidade deseja saber a quantidade de configurações existentes para montar seu sistema de comunicação. Dizemos que a configuração  $A$  é diferente da configuração  $B$  quando  $A$  tiver pelo menos uma linha que interliga dois departamentos que não existe em  $B$  ou  $A$  não tiver pelo menos uma linha que interliga dois departamentos que existe em  $B$ . Dessa forma, determine a quantidade de configurações existentes sabendo que a quantidade de departamentos da UFC é  $n$ , que não é obrigatório que todos os departamentos recebam ligações de rede e que cada par de departamentos não pode receber 2 ou mais ligações diretas:

- a)  $2^{\frac{n(n-1)}{2}}$
- b)  $\frac{n(n-1)}{2}$
- c)  $2^{n(n-1)}$
- d)  $2^{\frac{n!}{2}}$
- e)  $\infty$

## 14 | XIII Olimpíada Cearense de Informática

**Questão 19.** Um algoritmo guloso consiste em um algoritmo que visa procurar escolhas parciais que parecem ótimas sem se importar se essas escolhas são de fato as escolhas corretas para solucionar o problema inteiro. Um clássico exemplo de um problema que pode ser abordado de forma gulosa é o problema do troco: dado um conjunto de valores de moedas disponíveis e um inteiro  $X$ , ache o número mínimo de moedas que, somadas, resultam em  $X$ .

A ideia gulosa para resolver esse problema consiste em: escolha sempre a maior moeda que seja menor ou igual a  $X$ , retire o valor dessa moeda de  $X$  e acrescente 1 em um contador de moedas, repita a operação até que  $X$  seja 0.

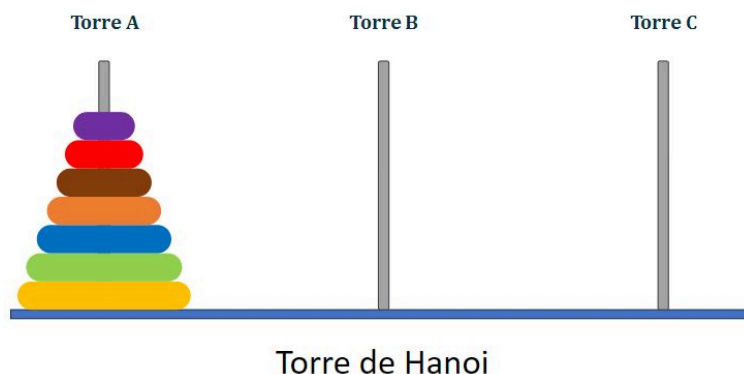
Por exemplo, seja o conjunto de moedas  $\{1, 5, 10, 50, 100\}$ , queremos um troco para 126 unidades monetárias. Primeiro, retiramos 100 do número inteiro, restando 26 e adicionando uma moeda na resposta. Em seguida, retiramos 10 do 26, restando 16 e adicionando mais uma moeda. Novamente, retiramos 10 do 16, restando 6. Após isso, retiramos 5, e, em seguida, retiramos 1, retornando uma contagem final de 5 moedas. Essa contagem é supostamente mínima, isto é, não é possível achar uma combinação de moedas dentro desse conjunto que some 126 e possua menos que 5 moedas.

Sobre o algoritmo guloso nesse problema, assinale a alternativa correta.

- a) O algoritmo guloso é uma resolução ótima para o Problema do Troco, encontrando sempre a solução mínima em qualquer conjunto de valores de moedas para qualquer valor inteiro  $X$ .
- b) O algoritmo guloso é uma resolução ótima para o Problema do Troco, entretanto, a depender do valor  $X$  em certas situações, o algoritmo falha.
- c) O algoritmo guloso é uma resolução correta para o Problema do Troco, entretanto, sua eficiência torna-o inviável em termos de custo computacional.
- d) O algoritmo guloso seria uma solução ótima, entretanto, em determinados conjuntos de valores de moedas, o algoritmo pode falhar ao retornar o número mínimo de moedas para somar um inteiro  $X$  qualquer.
- e) O uso de programação dinâmica é preferível à implementação do algoritmo guloso nesse caso, encontrando a solução correta e usando menos memória no processo.

## 15 | XIII Olimpíada Cearense de Informática

**Questão 20.** A Torre de Hanoi é um quebra-cabeça muito conhecido que consiste em 3 torres (pinos) e  $n$  discos com tamanhos decrescentes.



O objetivo do quebra-cabeça é transferir todos os discos da Torre A (início) até a torre B (fim) com algumas restrições: Um disco maior não pode ficar em cima de um disco menor, e apenas um disco pode ser movido por vez.

Esse quebra-cabeça é muito usado na introdução à computação por ser um ótimo exemplo de problema resolvível de forma recursiva, apresentando um certo nível de complexidade quanto à recursão usada, embora seja um quebra-cabeça simples. A lógica recursiva se baseia na seguinte ideia:

1. **Caso Base (o mais simples):** Se houver apenas 1 disco ( $n=1$ ), mova-o diretamente do pino de origem para o pino de destino.
2. **Passo Recursivo (para  $n$  discos):**
  - **Passo 1:** Mova recursivamente  $n-1$  discos do pino de **origem** para o pino **auxiliar** (usando o pino de destino como temporário).
  - **Passo 2:** Mova o  $n$ -ésimo disco (o maior, que sobrou na base) do pino de **origem** para o pino de **destino**.
  - **Passo 3:** Mova recursivamente os  $n-1$  discos que estão no pino **auxiliar** para o pino de **destino** (usando, agora, o pino de origem como auxiliar temporário).

Considere a seguinte implementação dessa ideia:

```
func TorreDeHanoi(n, pino_origem, pino_destino, pino_auxiliar):  
    se n == 1:  
        escreva("Mover disco 1 de", pino_origem, "para", pino_destino)  
    senao:  
        TorreDeHanoi(n - 1, pino_origem, pino_auxiliar, pino_destino)  
        escreva("Mover disco", n, "de", pino_origem, "para", pino_destino)  
        TorreDeHanoi(n - 1, pino_auxiliar, pino_destino, pino_origem)
```

## 16 | XIII Olimpíada Cearense de Informática

Imagine que esse procedimento foi chamado com a origem na Torre A, destino na Torre C, e auxiliar na Torre B, com apenas 4 discos e assinale a alternativa correta.

- a) O procedimento funciona corretamente, resolvendo o quebra-cabeça, neste caso, com 15 operações (movimentos de discos de um pino para outro).
- b) O procedimento não funciona corretamente, resultando nos discos na orientação correta, entretanto, no pino errado (os discos foram posicionados na Torre B ao invés da Torre C).
- c) O procedimento funciona corretamente, resolvendo o quebra-cabeça, nesse caso, com 7 operações.
- d) O procedimento não funciona corretamente pois ele, em algum momento da execução, viola as regras do quebra-cabeça.
- e) O procedimento não funciona corretamente, visto que ele entra em um loop infinito, movendo discos de forma irrelevante para a resolução do quebra-cabeça e nunca chegando ao final.