

Aula 01

*Banco do Brasil (Escriturário - Agente de
Tecnologia) Desenvolvimento de
Software*

Autor:

Paolla Ramos, Felipe Mathias

15 de Junho de 2025

Índice

| | |
|--|-----|
| 1) Métodos de Ordenação - Conceitos Básicos - Teoria | 4 |
| 2) Métodos de Ordenação - Complexidade de Algoritmos - Teoria | 15 |
| 3) Métodos de Ordenação - Pesquisa de Dados - Teoria | 19 |
| 4) Métodos de Ordenação - Conceitos Básicos - Questões Comentadas | 22 |
| 5) Métodos de Ordenação - Complexidade de Algoritmos - Questões Comentadas | 34 |
| 6) Métodos de Ordenação - Pesquisa de Dados - Questões Comentadas | 36 |
| 7) Métodos de Ordenação - Conceitos Básicos - Lista de Questões | 39 |
| 8) Métodos de Ordenação - Complexidade de Algoritmos - Lista de Questões | 45 |
| 9) Métodos de Ordenação - Pesquisa de Dados - Lista de Questões | 47 |
| 10) Estruturas de Dados - Conceitos Básicos - Teoria | 50 |
| 11) Estruturas de Dados - Vetores e Matrizes - Teoria | 52 |
| 12) Estruturas de Dados - Lista Encadeada - Teoria | 54 |
| 13) Estruturas de Dados - Pilhas - Teoria | 59 |
| 14) Estruturas de Dados - Filas - Teoria | 60 |
| 15) Estruturas de Dados - Árvore - Teoria | 62 |
| 16) Estruturas de Dados - Grafos - Teoria | 80 |
| 17) Estruturas de Dados - Hashing - Teoria | 86 |
| 18) Estruturas de Dados - Bitmap - Teoria | 87 |
| 19) Estruturas de Dados - Estrutura de Arquivos - Teoria | 90 |
| 20) Estruturas de Dados - Conceitos Básicos - Questões Comentadas | 91 |
| 21) Estruturas de Dados - Vetores e Matrizes - Questões Comentadas | 95 |
| 22) Estruturas de Dados - Lista Encadeada - Questões Comentadas | 104 |
| 23) Estruturas de Dados - Pilhas - Questões Comentadas | 116 |
| 24) Estruturas de Dados - Filas - Questões Comentadas | 128 |
| 25) Estruturas de Dados - Árvore - Questões Comentadas | 139 |
| 26) Estruturas de Dados - Grafos - Questões Comentadas | 173 |
| 27) Estruturas de Dados - Hashing - Questões Comentadas | 187 |
| 28) Estruturas de Dados - Bitmap - Questões Comentadas | 189 |



Índice

| | |
|--|-----|
| 29) Estruturas de Dados - Listas, Pilhas, Filas - Questões Comentadas - CESGRANRIO | 194 |
| 30) Estruturas de Dados - Conceitos Básicos - Lista de Questões | 249 |
| 31) Estruturas de Dados - Lista Encadeada - Lista de Questões | 253 |
| 32) Estruturas de Dados - Pilhas - Lista de Questões | 260 |
| 33) Estruturas de Dados - Filas - Lista de Questões | 273 |
| 34) Estruturas de Dados - Árvore - Lista de Questões | 279 |
| 35) Estruturas de Dados - Grafos - Lista de Questões | 294 |
| 36) Estruturas de Dados - Hashing - Lista de Questões | 303 |
| 37) Estruturas de Dados - Bitmap - Lista de Questões | 306 |
| 38) Estruturas de Dados - Listas, Pilhas, Filas - Lista de Questões - CESGRANRIO | 311 |



MÉTODOS DE ORDENAÇÃO

ENTREVISTA



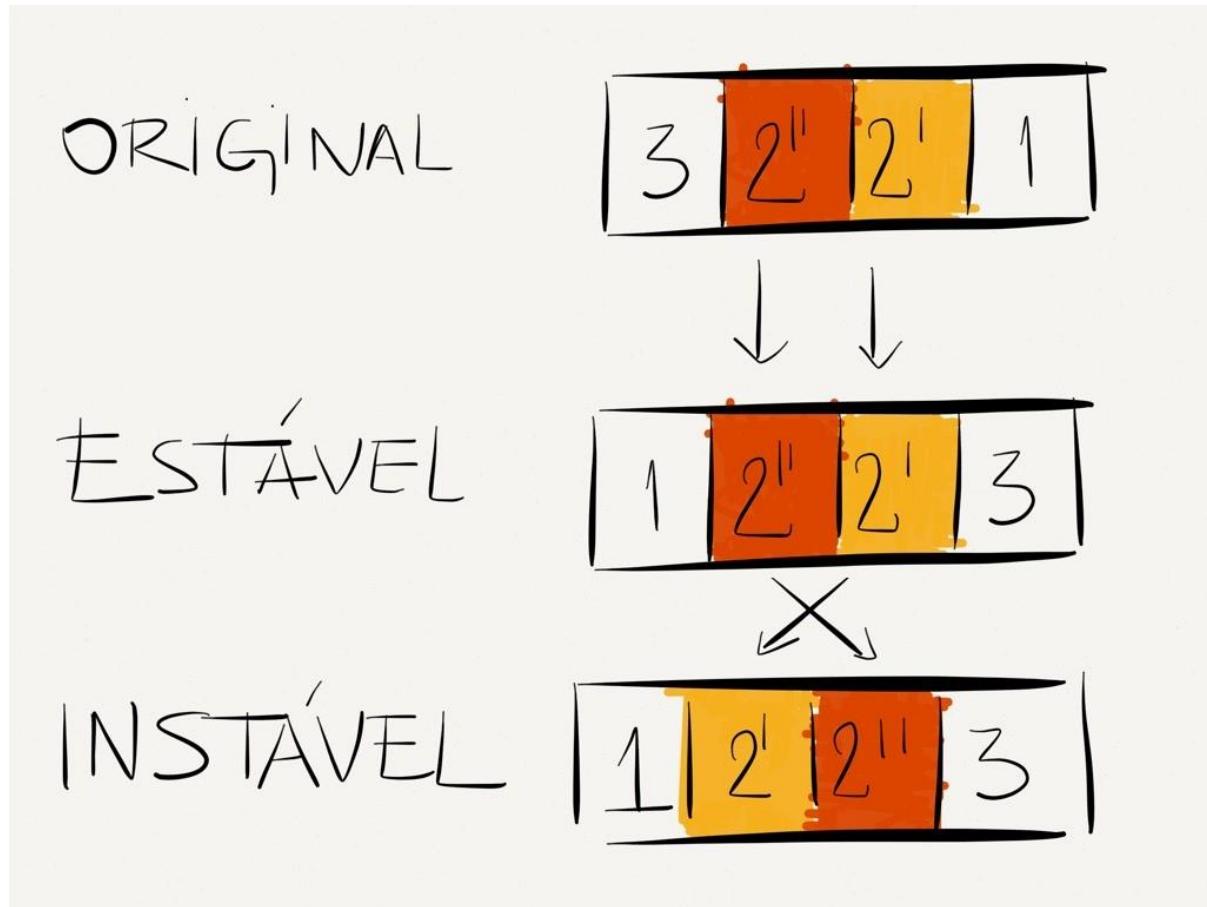
WWW.VIDADEHIPSTER.COM.BR

Letroakumi

Métodos de Ordenação são algoritmos que têm o objetivo principal de posicionar os elementos de uma estrutura de dados em uma **determinada ordem**. Para que, professor? Ora, isso possibilita o acesso mais rápido e eficiente aos dados. Existem dezenas de métodos, todavia nessa aula veremos apenas os mais importantes: BubbleSort, QuickSort, InsertionSort, SelectionSort, MergeSort, ShellSort e HeapSort.

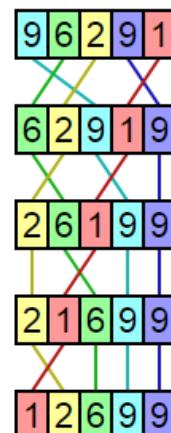
Antes de iniciar, vamos falar sobre o conceito de Estabilidade! Um método estável é aquele em que os itens com chaves iguais mantêm-se com a posição inalterada durante o processo de ordenação, i.e., preserva-se a ordem relativa dos itens com chaves duplicadas ou iguais. Métodos Estáveis: Bubble, Insertion e Merge; Métodos Instáveis: Selection, Quick, Heap e Shell. Vejamos um exemplo:





Na imagem acima, foi colocado um sinal de aspas simples e duplas apenas para diferenciá-los, mas trata-se do mesmo número. Um algoritmo estável ordena todo o restante e não perde tempo trocando as posições de elementos que possuam chaves idênticas. Já um algoritmo instável ordena todos os elementos, inclusive aqueles que possuem chaves idênticas (sob algum outro critério).

BubbleSort (Troca)



Esse algoritmo é o primeiro método de ordenação aprendido na faculdade, porque ele é bastante simples e intuitivo. Nesse método, os elementos da lista são movidos para as posições adequadas de forma contínua. Se um elemento está inicialmente em uma posição i e, para que a lista fique

ordenada, ele deve ocupar a posição j , então ele terá que passar por todas as posições entre i e j .



Em cada iteração do método, percorremos a lista a partir de seu início comparando cada elemento com seu sucessor, trocando-se de posição se houver necessidade. É possível mostrar que, se a lista tiver n elementos, após no máximo $(n-1)$ iterações, a lista estará em ordem (crescente ou decrescente). Observem abaixo o código para a Ordenação em Bolha:

```

ALGORITMO BOLHA
ENTRADA: UM VETOR L COM N POSIÇÕES
SAÍDA: O VETOR L EM ORDEM CRESCENTE

PARA i = 1 até n - 1
    PARA j = 0 até n - 1 - i
        SE L[j] > L[j+1]
            AUX = L[j]           // SWAP
            L[j] = L[j+1]
            L[j+1] = AUX
    FIM {BOLHA}

```

| Melhor Caso | Caso Médio | Pior Caso |
|-------------|------------|-----------|
| $O(n)$ | $O(n^2)$ | $O(n^2)$ |

InsertionSort (Inserção)

Esse algoritmo, também conhecido como Inserção Direta, é bastante simples e apresenta um desempenho significativamente melhor que o **BubbleSort**, em termos absolutos. Além disso, ele é extremamente eficiente para listas que já estejam substancialmente ordenadas e listas com pequeno número de elementos. Observem abaixo o código para a Ordenação de Inserção:

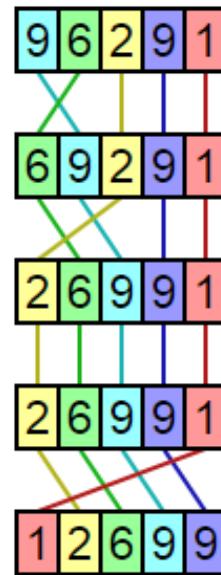
```

ALGORITMO INSERÇÃO
ENTRADA: UM VETOR L COM N POSIÇÕES
SAÍDA: O VETOR L EM ORDEM CRESCENTE

PARA i = 1 até n - 1
    PIVO = L[i]
    j = i - 1
    ENQUANTO j ≥ 0 e L[j] > PIVO
        L[j+1] = L[j]
        j = j - 1
    L[j+1] = PIVO
FIM {INSERÇÃO}

```





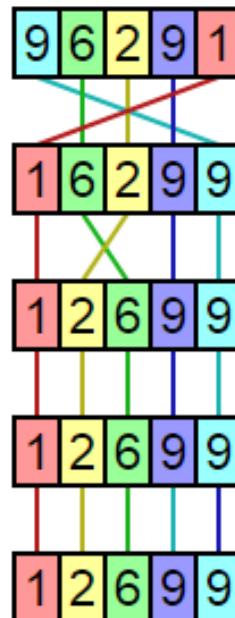
Nesse método, consideramos que a lista está dividida em parte esquerda, já ordenada, e parte direita, em possível desordem. Inicialmente, a parte esquerda contém apenas o primeiro elemento da lista. Cada iteração consiste em inserir o primeiro elemento da parte direta (pivô) na posição adequada da parte esquerda, de modo que a parte esquerda continue ordenada.

É fácil perceber que se a lista possui n elementos, após $(n-1)$ inserções, ela estará ordenada. Para inserir o pivô, percorremos a parte esquerda, da direita para a esquerda, deslocando os elementos estritamente maiores que o pivô uma posição para direita. O pivô deve ser colocado imediatamente à esquerda do último elemento movido.

| Melhor Caso | Caso Médio | Pior Caso |
|-------------|------------|-----------|
| $O(n)$ | $O(n^2)$ | $O(n^2)$ |



SelectionSort (Seleção)



Esse algoritmo consiste em selecionar o (swap) pelo item que estiver na primeira Essas duas operações são repetidas com Tem como ponto forte o fato de que realiza poucas operações de swap. Seu desempenho costuma ser superior ao BubbleSort e inferior ao InsertionSort.

menor1 elemento de um vetor e trocá-lo posição, i.e., inseri-lo no início do vetor. os itens restantes até o último elemento.

Assim como no InsertionSort, considera-se que a lista está dividida em parte esquerda, já ordenada, e parte direita, em possível desordem. Ademais, os elementos da parte esquerda são todos menores ou iguais aos elementos da parte direita. Cada iteração consiste em selecionar o menor elemento da parte direita (pivô) e trocá-lo com o primeiro elemento da parte direita.

Assim, a parte esquerda aumenta, visto que passa a incluir o pivô, e a parte direita diminui. Note que o pivô é maior que todos os demais elementos da parte esquerda, portanto a parte esquerda continua ordenada. Ademais, o pivô era o menor elemento da direita, logo todos os elementos da esquerda continuam sendo menores ou iguais aos elementos da direita. Inicialmente, a parte esquerda é vazia.

ALGORITMO SELEÇÃO
ENTRADA: UM VETOR L COM N POSIÇÕES
SAÍDA: O VETOR L EM ORDEM CRESCENTE

```
PARA i = 0 ate n - 2
    MIN = i
    PARA j = i + 1 ate n - 1
        SE L[j] < L[MIN]
            MIN= j
        TROCA(L[i], L[MIN])
    FIM {SELEÇÃO}
```

Melhor Caso

Caso Médio

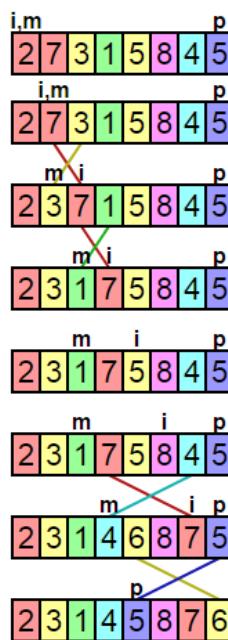
Pior Caso

¹ A definição formal afirma que é o maior valor; a maioria das implementações utiliza o menor valor. As questões de prova cobram algumas vezes o maior, outras vezes o menor.



$O(n^2)$ $O(n^2)$ $O(n^2)$

QuickSort (Troca)



Esse algoritmo divide um conjunto de itens em conjuntos menores, que são ordenados de forma independente, e depois os resultados são combinados para produzir a solução de ordenação do conjunto maior. Trata-se, portanto, de um algoritmo do tipo **Divisão-e-Conquista**, i.e., repartindo os dados em subgrupos, dependendo de um elemento chamado **pivô**.

Talvez seja o método de ordenação mais utilizado! Isso ocorre porque quase sempre ele é significativamente mais rápido do que todos os demais métodos de ordenação baseados em comparação. Ademais, suas características fazem com que ele, assim como o **MergeSort**, possa ser facilmente paralelizado. Ele também pode ser adaptado para realizar ordenação externa (QuickSort Externo).

Neste método, a lista é dividida em parte esquerda e parte direita, sendo que os elementos da parte esquerda são todos menores que os elementos da parte direita. Essa fase do processo é chamada de partição. Em seguida, as duas partes são ordenadas recursivamente (usando o próprio QuickSort). A lista está, portanto, ordenada corretamente!

Uma estratégia para fazer a partição é escolher um valor como pivô e então colocar na parte esquerda os elementos menores ou iguais ao pivô e na parte direita os elementos maiores que o pivô – galera, a escolha do pivô é crítica! Em geral, utiliza-se como pivô o primeiro elemento da lista, a despeito de existirem maneiras de escolher um “melhor” pivô.

Esse algoritmo é um dos métodos mais rápidos de ordenação, apesar de às vezes partições desequilibradas poderem conduzir a uma ordenação lenta. A eficácia do método depende da escolha do pivô mais adequado



ao conjunto de dados que se deseja ordenar. Alguns, por exemplo, utilizam a mediana de três elementos para otimizar o algoritmo.

Alguns autores consideram a divisão em três subconjuntos, sendo o terceiro contendo valores iguais ao pivô. O Melhor Caso ocorre quando o conjunto é dividido em subconjuntos de mesmo tamanho; o Pior Caso ocorre quando o pivô corresponde a um dos extremos (menor ou maior valor). Alguns o consideram um algoritmo frágil e não-estável, com baixa tolerância a erros.

```
PROCEDIMENTO PARTIÇÃO
    ENTRADA: UM VETOR L E AS POSIÇÕES INICIO E FIM
    SAÍDA:j, tal que L[INICIO]..L[j-1] ≤ L[j] e
        L[j+1]..L[FIM] > L[j]
        // j é a posição onde será colocado o pivô, como
        // ilustrado na figura abaixo
        PIVO = L[INICIO], i = INICIO + 1, j = FIM
        ENQUANTO i ≤ j
            ENQUANTO i ≤ j e L[i] ≤ PIVO
                i = i + 1
            ENQUANTO L[j] > PIVO
                j = j - 1
            SE i ≤ j
                TROCA(L[i],L[j])
                i = i + 1, j = j - 1
            TROCA(L[INICIO], L[j])
            DEVOLVA j
        FIM {PARTIÇÃO}
```

| Melhor Caso | Caso Médio | Pior Caso |
|---------------|---------------|-----------|
| $O(n \log n)$ | $O(n \log n)$ | $O(n^2)$ |

ShellSort (Inserção)

Esse algoritmo é uma extensão ou refinamento do algoritmo do InsertionSort, contornando o problema que ocorre quando o menor item de um vetor está na posição mais à direita. Ademais, difere desse último pelo fato de, em vez de considerar o vetor a ser ordenado como um único segmento, ele considera vários segmentos e aplica o **InsertionSort** em cada um deles.

É o algoritmo **mais eficiente** dentre os de ordem quadrática. Nesse método, as comparações e as trocas são feitas conforme determinada distância (gap) entre dois elementos, de modo que, se gap = 6, há comparação entre o 1º e 7º elementos ou entre o 2º e 8º elementos e assim sucessivamente, repetindo até que as últimas comparações e trocas tenham sido efetuadas e o gap tenha chegado a 1.



ALGORITMO SHELLSORT

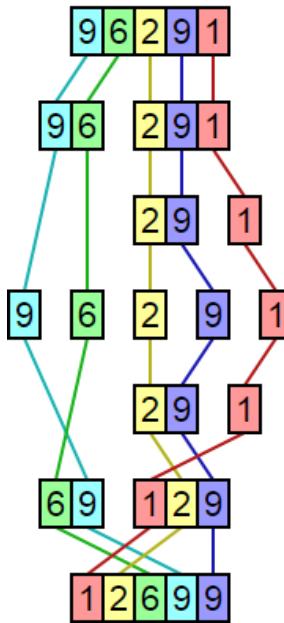
ENTRADA: UM VETOR L COM N POSIÇÕES
 SAÍDA: O VETOR L EM ORDEM CRESCENTE

```

H = 1
ENQUANTO H < n FAÇA H = 3 * H + 1
FAÇA
  H = H / 3           // divisão inteira
  PARA i = H até n - 1 // Inserção adaptado para h-listas
    PIVO = L[i]
    j = i - H
    ENQUANTO j ≥ 0 e L[j] > PIVO
      L[j + H] = L[j]
      j= j - H
    L[j + H] = PIVO
  ENQUANTO H > 1
FIM {SHELLSORT}
  
```

| Melhor Caso | Caso Médio | Pior Caso |
|---------------|----------------|-----------|
| $O(n \log n)$ | Depende do gap | $O(n^2)$ |

MerqeSort (Intercalação)



Esse algoritmo é baseado na estratégia de resolução de problemas conhecida como **divisão-e-conquista**. Essa técnica consiste basicamente em decompor a instância a ser resolvida em instâncias menores do mesmo tipo de problema, resolver tais instâncias (em geral, recursivamente) e por fim utilizar as soluções parciais para obter uma solução da instância original.

Naturalmente, nem todo problema pode ser resolvido através de divisão e conquista. Para que seja viável aplicar essa técnica a um problema, ele deve possuir duas propriedades estruturais. O problema deve ser



decomponível, i.e., deve ser possível decompor qualquer instância não trivial do problema em instâncias menores do mesmo tipo de problema.

Além disso, deve ser sempre possível utilizar as soluções obtidas com a resolução das instâncias menores para chegar a uma solução da instância original. No MergeSort, divide-se a lista em duas metades. Essas metades são ordenadas recursivamente (usando o próprio MergeSort) e depois são intercaladas. Abaixo segue uma possível solução:

ALGORITMO MERGESORT

ENTRADA: UM VETOR L E AS POSIÇÕES INICIO E FIM

SAÍDA: O VETOR L EM ORDEM CRESCENTE DA POSIÇÃO INICIO ATÉ A POSIÇÃO FIM

```

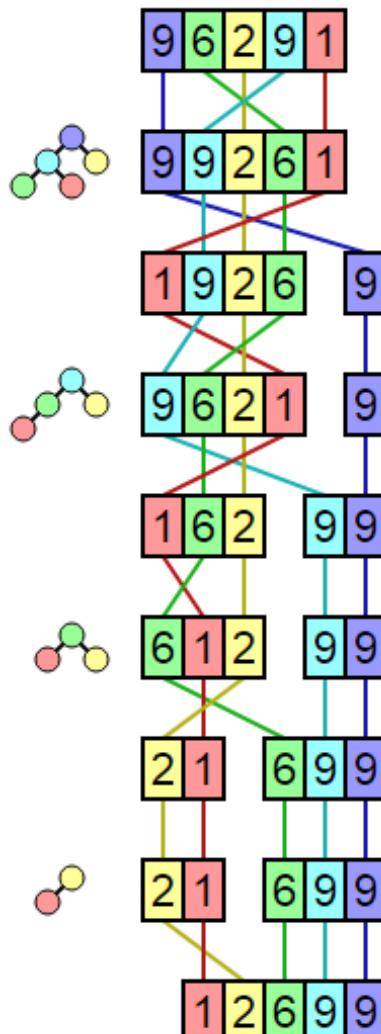
SE inicio < fim
    meio = (inicio + fim)/2           // divisão inteira
    SE inicio < meio
        MERGESORT(L, inicio, meio)
    SE meio + 1 < fim
        MERGESORT(L, meio + 1, fim)
    MERGE(L, inicio, meio, fim)
FIM {MERGESORT}

```

| Melhor Caso | Caso Médio | Pior Caso |
|---------------|---------------|---------------|
| $O(n \log n)$ | $O(n \log n)$ | $O(n \log n)$ |



HeapSort (Seleção)



Esse algoritmo utiliza uma estrutura para ordenar os elementos à medida que

de dados chamada **heap**, para os insere na estrutura.

Assim, ao final das inserções, os elementos podem ser sucessivamente removidos da raiz da heap, na ordem desejada.

Essa estrutura pode ser representada como uma árvore ou como um vetor. Entenderam? Inicialmente, insere-se os elementos da lista em um heap.

Em seguida, fazemos sucessivas remoções do menor elemento do heap, colocando os elementos removidos do heap de volta na lista – a lista estará então em ordem crescente.

O heapsort é um algoritmo de ordenação em que a sua estrutura auxiliar de armazenamento fora do arranjo de entrada é constante durante toda a sua execução.



ALGORITMO HEAP SORT

ENTRADA: UM VETOR L COM N POSIÇÕES
 SAÍDA: O VETOR L EM ORDEM CRESCENTE

```

inicialize um HBC H com n posições
PARA i = 0 até n - 1
    INSERE_HBC(H, L[i])
PARA i = 0 até n - 1
    L[i] = REMOVE_MENOR(H)
FIM {HEAP SORT}
  
```

| Melhor Caso | Caso Médio | Pior Caso |
|---------------|---------------|---------------|
| $O(n \log n)$ | $O(n \log n)$ | $O(n \log n)$ |

Resumão de Complexidades

| ALGORITMO | MELHOR CASO | CASO MÉDIO | PIOR CASO |
|---------------|---------------|----------------|---------------|
| BubbleSort | $O(n)$ | $O(n^2)$ | $O(n^2)$ |
| InsertionSort | $O(n)$ | $O(n^2)$ | $O(n^2)$ |
| SelectionSort | $O(n^2)$ | $O(n^2)$ | $O(n^2)$ |
| QuickSort | $O(n \log n)$ | $O(n \log n)$ | $O(n^2)$ |
| ShellSort | $O(n \log n)$ | Depende do gap | $O(n^2)$ |
| MergeSort | $O(n \log n)$ | $O(n \log n)$ | $O(n \log n)$ |
| HeapSort | $O(n \log n)$ | $O(n \log n)$ | $O(n \log n)$ |



COMPLEXIDADE DE ALGORITMOS

Galera, por que estudamos a complexidade de algoritmos? Para determinar o custo computacional (tempo, espaço, etc) para execução de algoritmos. Em outras palavras, ela classifica problemas computacionais de acordo com sua **dificuldade inerente**. É importante entender isso para posteriormente estudarmos a complexidade de métodos de ordenação e métodos de pesquisa.

Nosso estudo aqui será bastante superficial por duas razões: primeiro, concursos cobram pouco e, quando cobram, querem saber as complexidades dos métodos de ordenação mais conhecidos; segundo, essa é uma disciplina absurdamente complexa que envolve Análise Assintótica, Cálculo Diferencial, Análise Polinomial (Linear, Exponencial, etc). Logo, vamos nos ater ao que cai em concurso público!

Só uma pausa: passei dias sem dormir na minha graduação por conta dessa disciplina! Na UnB, ela era ministrada pelo Instituto de Matemática e era considerada a disciplina mais difícil do curso :-(. Continuando: lá em cima eu falei sobre custo computacional! Ora, para que eu escolha um algoritmo, eu preciso definir algum **parâmetro**.

Podemos começar com Tempo! Um algoritmo que realiza uma tarefa em 20 minutos é melhor do que um algoritmo que realiza uma tarefa em 20 dias? Não é uma boa estratégia, porque depende do computador que eu estou utilizando (e todo o hardware correspondente), depende das otimizações realizadas pelo compilador, entre outras variáveis.

Vamos analisar, então, Espaço! Um algoritmo que utiliza 20Mb de RAM é melhor do que um algoritmo que utiliza 20Gb? Seguem os mesmos argumentos utilizados para o Tempo, ou seja, não é uma boa opção! E agora, o que faremos? Galera, eu tenho uma sugestão: investigar a quantidade de vezes que operações são executadas na execução do algoritmo!

Essa estratégia independe do computador (e hardware associado), do compilador, da linguagem de programação, das condições de implementação, entre outros fatores – ela depende apenas da qualidade inerente do algoritmo¹ implementado. Utilizam-se algumas **simplificações matemáticas** para se ter uma ideia do comportamento do algoritmo. Prosseguindo...

Dada uma entrada de dados de tamanho N, podemos calcular o custo computacional de um algoritmo em seu pior caso, médio caso e melhor caso! Como assim, professor? Para entender isso, vamos utilizar a metáfora de um jogo de baralho! Imaginem que eu estou jogando contra vocês. Vocês embaralham e me entregam 5 cartas, eu embaralho novamente e lhes entrego 5 cartas.

Quem joga baralho sabe que uma boa alternativa para grande parte dos jogos é ordenar as cartas em ordem crescente de modo a encontrar mais facilmente a melhor carta para jogar. Agora observem... vocês receberam as seguintes cartas (nessa ordem): 4, 5, 6, 7, 8. Já eu recebi as seguintes cartas (também nessa ordem): 8, 7, 6, 5, 4 – nós queremos analisar a complexidade de ordenação dessas cartas.

¹ Pessoal, é claro que nossa visão sobre a complexidade dos algoritmos é teórica. Na prática, depende de diversos outros fatores, mas nosso foco é na visão analítica e, não, empírica.





Ora, convenhamos que vocês possuem o melhor caso, porque vocês deram a sorte de as cartas recebidas já estarem ordenadas. Já eu peguei o pior caso, porque as cartas estão ordenadas na ordem inversa. Por fim, o caso médio ocorre caso as cartas recebidas estejam em uma ordem aleatória. Com isso, espero que vocês tenham entendido o sentido de pior, médio e melhor casos.

Vamos partir agora para o estudo da **Notação Big-O** (ou Notação Assintótica)! Isso é simplesmente uma forma de representar o comportamento assintótico de uma função. No nosso contexto, ela busca expressar a quantidade de operações primitivas executadas como função do tamanho da entrada de dados. Vamos ver isso melhor!

A Notação Big-O é a representação relativa da complexidade de um algoritmo. É relativa porque só se pode comparar maçãs com maçãs, isto é, você não pode comparar um algoritmo de multiplicação aritmética com um algoritmo de ordenação de inteiros. É uma representação porque reduz a comparação entre algoritmos a uma simples variável por meio de observações e suposições.

E trata da complexidade porque se é necessário 1 segundo para ordenar 10.000 elementos, quanto tempo levará para ordenar 1.000.000? A complexidade, nesse exemplo particular, é a medida relativa para alguma coisa. Vamos ver isso por meio de um exemplo: soma de dois inteiros! A soma é uma operação ou um problema, e o método para resolver esse problema é chamado algoritmo!

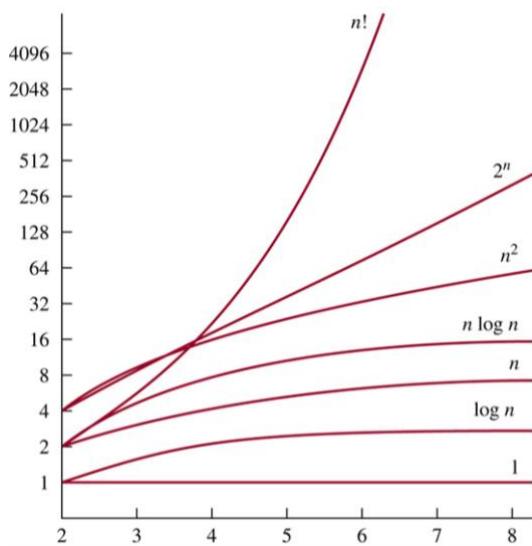
$$\begin{array}{r} \text{Transporte} \rightarrow & 1 & 1 & 1 & 1 & 0 \\ \text{Parcela 1} \rightarrow & & 1 & 1 & 0 & 1 \\ \text{Parcela 2} \rightarrow & + & 1 & 0 & 1 & 1 \\ \hline \text{Soma} \rightarrow & 1 & 1 & 0 & 0 & 0 \end{array}$$

Vamos supor que no algoritmo de somar (mostrado acima), a operação mais cara seja a adição. Observem que se somarmos dois números de 100 dígitos, teremos que fazer 100 adições. Se somarmos dois números de 10.000 dígitos, teremos que fazer 10.000 adições. Perceberam o padrão? A complexidade (aqui, número de operações) é diretamente proporcional ao número n de dígitos, i.e., $O(n)$.

Quando dizemos que um algoritmo é $O(n^2)$, estamos querendo dizer que esse algoritmo é da ordem de grandeza quadrática! Ele basicamente serve para te dizer quão rápido uma função cresce, por exemplo: um algoritmo $O(n)$ é melhor do que um algoritmo $O(n^2)$, porque ela cresce mais lentamente! Abaixo vemos uma lista das classes de funções mais comuns em ordem crescente de crescimento:



| Notação | Nome |
|-----------------|-----------------|
| $O(1)$ | Constante |
| $O(\log n)$ | Logarítmica |
| $O[(\log n)^c]$ | Polilogarítmica |
| $O(n)$ | Linear |
| $O(n \log n)$ | — |
| $O(n^2)$ | Quadrática |
| $O(n^3)$ | Cúbica |
| $O(n^c)$ | Polinomial |
| $O(c^n)$ | Exponencial |
| $O(n!)$ | Fatorial |



Quando dizemos que o Shellsort é um algoritmo $O(n^2)$, estamos querendo dizer que a complexidade (nesse caso, o número de operações) para ordenar um conjunto de n dados com o Algoritmo Shellsort é proporcional ao quadrado do número de elementos no conjunto! Grosso modo, para ordenar 20 números, é necessário realizar 400 operações (sem entrar em detalhes sobre a operação em si, nem sobre as simplificações matemáticas que são realizadas).

Entender como se chega a esses valores para cada método de ordenação e pesquisa é extremamente complexo! Galera, apesar de eu nunca ter visto isso em prova, é bom que vocês saibam que existem outras notações! Utiliza-se Notação Big-O (O) para pior caso; Notação Big-Ômega para melhor caso (Ω); e Notação Big-Theta (Θ) para caso médio.

Como na prática utiliza-se Big-O para tudo, o que eu recomendo (infelizmente, porque eu sei que vocês têm zilhões de coisas para decorar) é memorizar o pior caso dos principais métodos. Dessa forma, é possível responder a maioria das questões de prova sobre esse tema. Eventualmente, as questões pedem também caso médio e melhor caso, mas é menos comum. Bacana? :-)

Por último, uma pergunta muito frequente: Professor, já vi questões cobrando Logaritmo na Base 10, Logaritmo na Base 2, Logaritmo Neperiano, etc... isso não está errado? Galera, suponha que um algoritmo



levou um tempo $\Theta(\log_{10} n)$. Você também poderia dizer que levou um tempo $\Theta(\lg n)$ (ou seja, $\Theta(\log_2 n)$). Você pode utilizar qualquer base.

Sempre que a **base** do logaritmo é uma **constante**, não importa a base que usamos na notação assintótica. Por que não? Porque, para a notação assintótica, isso é completamente irrelevante. Beleza? Então, não se prendam a base do logaritmo, qualquer uma pode ser utilizada na representação de complexidade assintótica de algoritmos. Bacana? Exercícios...



PESQUISA DE DADOS

Uma das tarefas de maior importância na computação é a pesquisa de informações contidas em coleções de dados. Em geral, desejamos que essa tarefa seja executada sem que haja a necessidade de inspecionar toda a coleção de dados. Existem algumas maneiras de realizar pesquisas, tais como: Pesquisa Sequencial, Pesquisa Binária, Tabelas de Dispersão (Hashing), Árvores AVL, Árvores B e Árvores B+.

Busca Sequencial

Galera, imaginem que eu estou à procura de um valor X em um vetor L[]! Para tal, posso inspecionar as posições sequenciais de L[] a partir da primeira posição: se eu encontrar X, minha busca tem êxito; se eu alcanço a última posição e não encontro X, concluímos que esse valor não ocorre no vetor L[]. Como é chamada essa busca em que eu inspeciono uma estrutura posição por posição? **Sequencial ou Linear**.

Considerando que o vetor L[] contém N elementos, ordenados ou não, é fácil verificar que a busca sequencial requer tempo linearmente proporcional ao tamanho do vetor, i.e., da ordem O(n). Por conta disso, é comum dizer que a busca sequencial é uma Busca Linear. Entenderam? Quanto maior o vetor, maior o tempo em média para buscar um elemento! Quanto mais ao final, mais demorado.

A **Busca Sequencial** é muito lenta para grandes quantidades de dados, mas aceitável para listas pequenas e que mudam constantemente. Observa-se que no Melhor Caso, X está na primeira posição, logo necessita apenas de uma comparação; no Pior Caso, X está na última posição, logo necessita de N comparações; e no Caso Médio, X é encontrado após $(n+1)/2$ comparações.

A seguir, encontram-se dois algoritmos para realização de uma Busca Sequencial: o primeiro ocorre de forma simples e o segundo ocorre de forma recursiva. Galera, para vetores de médio ou grande porte, o tempo de busca sequencial é considerado completamente inaceitável, dado que existem técnicas mais eficientes! Professor, me dá um exemplo? Claro, veremos adiante: Busca Binária!

```
PROCEDIMENTO BUSCA_SEQUENCIAL ( L , X , POS )  
  
ENTRADA: UM VETOR L e UM VALOR X  
SAÍDA: POS = i, SE X OCORRE NA POSIÇÃO i DE L // SUCESSO  
       POS = 0, CASO CONTRÁRIO.  
  
POS = 0  
  
PARA i = 1 até N  
    SE L[i] = X  
        POS = i  
        ESCAPE  
    // fim para  
    DEVOLVA POS  
FIM {BUSCA_SEQUENCIAL}
```



```
PROCEDIMENTO BUSCA_SEQUENCIAL_REC ( L , N, X )
```

*ENTRADA: UM VETOR L DE TAMANHO N e UM VALOR X
SAÍDA: i, SE X OCORRE NA POSIÇÃO i DE L // SUCESSO
0, CASO CONTRÁRIO.*

```
SE N = 1
    SE L[1] = X
        DEVOLVA 1
    SENÃO
        DEVOLVA 0
    SENÃO
        SE L[N] = X
            DEVOLVA N
    SENÃO
        BUSCA_SEQUENCIAL_REC ( L , N-1, X )

FIM {BUSCA_SEQUENCIAL_REC}
```

Busca Binária

A Busca Binária é um **algoritmo de busca** em vetores que segue o paradigma de divisão-e-conquista. Parte-se do pressuposto de que o vetor está ordenado e realiza sucessivas divisões do espaço de busca, comparando o elemento chave com o elemento do meio do vetor. Possui complexidade da ordem de $O(\log_2 n)$, em que N é o tamanho do vetor de busca.

Quando o Vetor L[] estiver em ordem crescente, podemos determinar se X ocorre em L[] de forma mais rápida da seguinte forma: inspeciona-se a posição central do vetor! Se essa posição já contiver X, a busca para! Por que, professor? Porque nós já encontramos X! Se X for menor que esse elemento central, passamos a procurar X, recursivamente, no intervalo de L[] que se encontra à esquerda da posição central.

Se X for maior do que o elemento central, continuamos a procurar X, recursivamente, no intervalo de L que está à direita da posição central. Se o intervalo se tornar vazio, a busca para, tendo sido malsucedida. Esse procedimento é conhecido como Busca Binária e, facilmente, pode-se adaptar a busca em ordem decrescente. Segue abaixo um possível algoritmo:

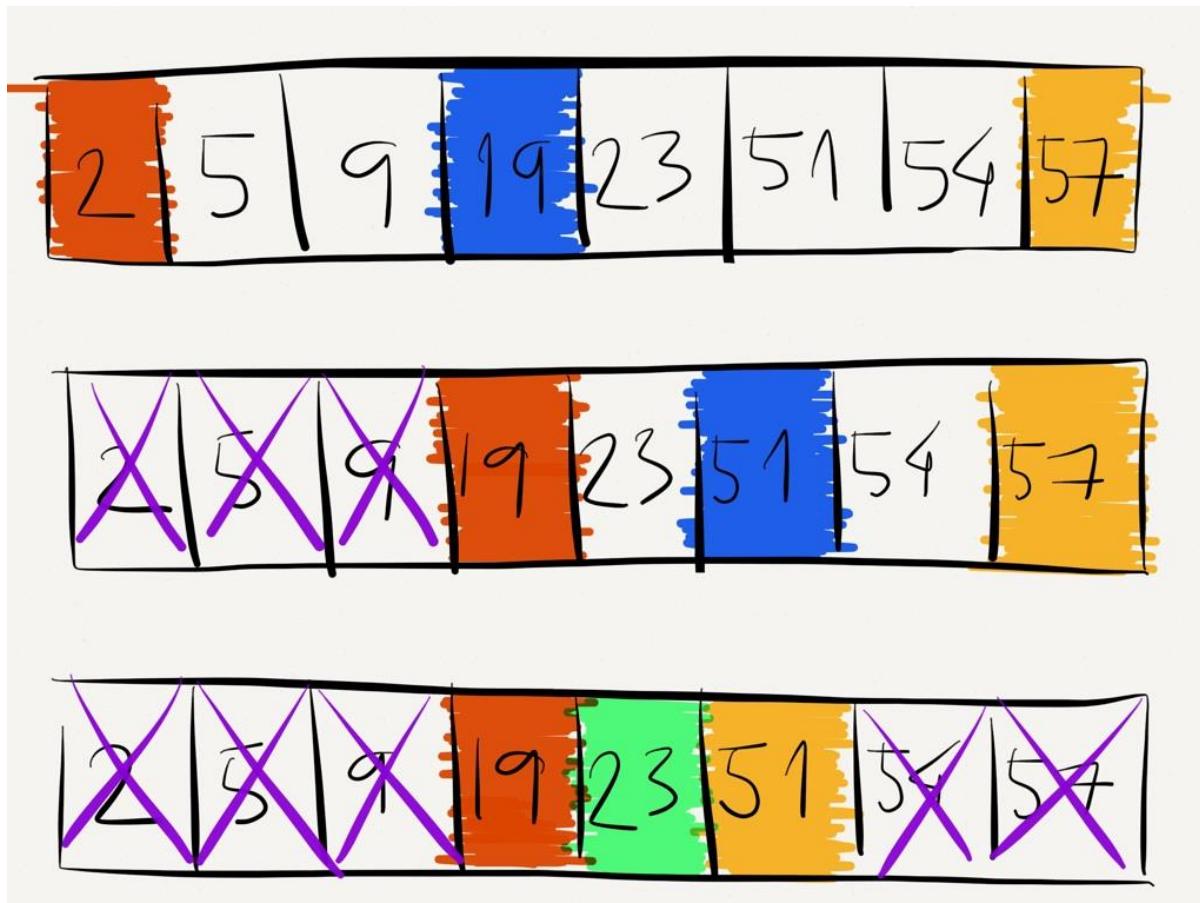
```
PROCEDIMENTO BUSCA_BINARIA
ENTRADA: UM VETOR L EM ORDEM CRESCENTE, UM VALOR X, E AS
         POSIÇÕES INICIO E FIM.
SAÍDA: SIM, SE X OCORRE ENTRE AS POSIÇÕES INICIO E FIM DE
         L; NÃO, CASO CONTRÁRIO.

SE INICIO > FIM
    DEVOLVA NÃO E PARE
MEIO = (INICIO+FIM) /2                                // DIVISÃO INTEIRA
SE X = L[MEIO]
    DEVOLVA SIM E PARE
SE X < L[MEIO]
    DEVOLVA BUSCA_BINARIA(L, X, INICIO, MEIO - 1)
SENAO
    DEVOLVA BUSCA_BINARIA(L, X, MEIO + 1, FIM)
FIM {BUSCA_BINARIA}
```



Na imagem abaixo, estamos à procura do valor 23! Em vermelho, encontra-se o elemento inicial $L[0] = 2$ e, em amarelo, encontra-se o elemento final $L[N-1] = 57$. Procuramos, então, o elemento central! Como? Ele é o elemento de índice $[0 + (N-1)]/2 = 7/2 = 3,5 = 3$ (Arredonda-se para baixo). Ora, $L[3] = 19$! Encontramos? Não, $23 > 19$! Sendo assim, $L[0] = 19$ e $L[4] = 57$.

Procuramos, então, o **elemento central!** Como? Ele é o elemento de índice $[0 + (N-1)]/2 = 4/2 = 2$. Ora, $L[2] = 51$! Encontramos? Não, $23 < 51$! Sendo assim, $L[0] = 19$ e $L[2] = 51$. Procuramos, então, o elemento central! Como? Ele é o elemento de índice $[0 + (N-1)]/2 = 2/2 = 1$. Ora, $L[1] = 23$! Encontramos? Sim! Então, nossa busca obteve êxito e encontramos o que buscávamos.



QUESTÕES COMENTADAS – MÉTODOS DE ORDENAÇÃO - MULTIBANCAS

1. (Instituto Cidades - 2012 - TCM-GO - Auditor de Controle Externo - Informática) São exemplos de algoritmos de ordenação, exceto:
 - a) Bubble Sort
 - b) Select Sort
 - c) Shell Sort
 - d) Busca Sequencial;
 - e) Quick Sort;

Comentários:

Conforme vimos em aula, a Busca Sequencial não é um algoritmo de ordenação! Na verdade, ele é um método de pesquisa sobre estruturas de dados. **Gabarito: D**

2. (FUMARC - 2012 - TJ-MG - Técnico Judiciário - Analista de Sistemas – I) Quicksort divide um conjunto de itens em conjuntos menores, que são ordenados de forma independe, e depois os resultados são combinados para produzir a solução de ordenação do conjunto maior.

Comentários:

Conforme vimos em aula, está perfeito! Sendo um algoritmo do tipo Dividir Para Conquistar, ele reparte o conjunto de dados em conjuntos menores, que são ordenados independentemente e depois combinados em uma solução maior. **Gabarito: C**

3. (CESPE - 2012 - MPE-PI - Analista Ministerial - Informática - Cargo 6) O heapsort é um algoritmo de ordenação em que a quantidade de elementos armazenada fora do arranjo de entrada é constante durante toda a sua execução.

Comentários:

Inicialmente, insere-se os elementos da lista em um heap. Em seguida, fazemos sucessivas remoções do menor elemento do heap, colocando os elementos removidos do heap de volta na lista – a lista estará então em ordem crescente. O heapsort é um algoritmo de ordenação em que a sua estrutura auxiliar de armazenamento fora do arranjo de entrada é constante durante toda a sua execução.

Essa questão é polêmica. O arranjo tem tamanho constante, mas a quantidade de elementos é variável. Diferente de outros algoritmos de ordenação que tem uma estrutura auxiliar de tamanho variável (assim como seus elementos), o Heap Sort tem uma estrutura auxiliar de tamanho fixo (porém a quantidade de elementos é variável). Como é dito por Neil Dale: "A heapsort is just as efficient in terms of space; only one



array is used to store the data. The heap sort requires only constant extra space". No entanto, a questão foi dada como correta! **Gabarito: C**

4. (CESPE - 2010 - ABIN - Oficial Técnico de Inteligência - Área de Suporte a Rede de Dados) A eficácia do método de ordenação rápida (quicksort) depende da escolha do pivô mais adequado ao conjunto de dados que se deseja ordenar. A situação ótima ocorre quando o pivô escolhido é igual ao valor máximo ou ao valor mínimo do conjunto de dados.

Comentários:

Alguns autores consideram a divisão em três subconjuntos, sendo o terceiro contendo valores iguais ao pivô. O Melhor Caso ocorre quando o conjunto é dividido em subconjuntos de mesmo tamanho; o Pior Caso ocorre quando o pivô corresponde a um dos extremos (menor ou maior valor). Alguns o consideram um algoritmo frágil e não-estável, com baixa tolerância a erros.

Conforme vimos em aula, a questão se refere ao pior caso! **Gabarito: E**

5. (CESPE - 2010 - ABIN - Oficial Técnico de Inteligência - Área de Suporte a Rede de Dados) A estabilidade de um método de ordenação é importante quando o conjunto de dados já está parcialmente ordenado.

Comentários:

Na imagem acima, foi colocado um sinal de aspas simples e duplas apenas para diferenciá-los, mas trata-se do mesmo número. Um algoritmo estável ordena todo o restante e não perde tempo trocando as posições de elementos que possuem chaves idênticas. Já um algoritmo instável ordena todos os elementos, inclusive aqueles que possuem chaves idênticas (sob algum outro critério).

Conforme vimos em aula, a estabilidade é irrelevante com dados parcialmente ordenados ou não! A estabilidade é importante quando se deseja ordenar um conjunto de dados por mais de um critério (Ex: primeiro pelas chaves e segundo por índices). Se esse não for o caso (e a questão não disse que era!), a estabilidade "não fede nem cheira". O fato de os dados estarem parcialmente ordenados não fará diferença em termos de ordenação – ambos serão ordenados da mesma maneira. **Gabarito: E**

6. (CESPE - 2010 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação - Administração de Dados) A classificação interna por inserção é um método que realiza a ordenação de um vetor por meio da inserção de cada elemento em sua posição correta dentro de um subvetor classificado.

Comentários:

Conforme vimos em aula, trata-se do InsertionSort! **Gabarito: C**

7. (FCC - 2009 - TRT - 15ª Região - Analista Judiciário - Tecnologia da Informação) São algoritmos de classificação por trocas apenas os métodos:

a) SelectionSort e InsertionSort.



- b) MergeSort e BubbleSort.
- c) QuickSort e SelectionSort.
- d) BubbleSort e QuickSort.
- e) InsertionSort e MergeSort.

Comentários:

Conforme vimos em aula, BubbleSort e QuickSort são métodos de troca; InsertionSort é um método de Inserção; SelectionSort e HeapSort são métodos de Seleção; e MergeSort é um método de Intercalação.

Gabarito: D

- 8. (CESGRANRIO - 2011 - PETROBRÁS – Analista de Sistemas – I) O tempo de pior caso do algoritmo QuickSort é de ordem menor que o tempo médio do algoritmo Bubblesort.**

Comentários:

| Algoritmo | Melhor Caso | Caso Médio | Pior Caso |
|---------------|---------------|-----------------------|---------------|
| BubbleSort | $O(n)$ | $O(n^2)$ | $O(n^2)$ |
| InsertionSort | $O(n)$ | $O(n^2)$ | $O(n^2)$ |
| SelectionSort | $O(n^2)$ | $O(n^2)$ | $O(n^2)$ |
| QuickSort | $O(n \log n)$ | $O(n \log n)$ | $O(n^2)$ |
| ShellSort | $O(n \log n)$ | Depende do <i>gap</i> | $O(n^2)$ |
| MergeSort | $O(n \log n)$ | $O(n \log n)$ | $O(n \log n)$ |
| HeapSort | $O(n \log n)$ | $O(n \log n)$ | $O(n \log n)$ |

Conforme vimos em aula, está incorreto! São iguais: $O(n^2)$. **Gabarito: E**

- 9. (CESGRANRIO - 2011 - PETROBRÁS – Analista de Sistemas – II) O tempo médio do QuickSort é $O(n \log 2n)$, pois ele usa como estrutura básica uma árvore de prioridades.**

Comentários:

| Algoritmo | Melhor Caso | Caso Médio | Pior Caso |
|------------|-------------|------------|-----------|
| BubbleSort | $O(n)$ | $O(n^2)$ | $O(n^2)$ |



| | | | |
|---------------|---------------|-----------------------|---------------|
| InsertionSort | $O(n)$ | $O(n^2)$ | $O(n^2)$ |
| SelectionSort | $O(n^2)$ | $O(n^2)$ | $O(n^2)$ |
| cQuickSort | $O(n \log n)$ | $O(n \log n)$ | $O(n^2)$ |
| ShellSort | $O(n \log n)$ | Depende do <i>gap</i> | $O(n^2)$ |
| MergeSort | $O(n \log n)$ | $O(n \log n)$ | $O(n \log n)$ |
| HeapSort | $O(n \log n)$ | $O(n \log n)$ | $O(n \log n)$ |

Conforme vimos em aula, de fato, ele tem tempo médio $O(n \log n)$, mas ele usa como estrutura básica uma lista ou um vetor! **Gabarito: E**

10. (CESGRANRIO - 2011 - PETROBRÁS – Analista de Sistemas – III) O tempo médio do QuickSort é de ordem igual ao tempo médio do MergeSort.

Comentários:

| Algoritmo | Melhor Caso | Caso Médio | Pior Caso |
|---------------|---------------|-----------------------|---------------|
| BubbleSort | $O(n)$ | $O(n^2)$ | $O(n^2)$ |
| InsertionSort | $O(n)$ | $O(n^2)$ | $O(n^2)$ |
| SelectionSort | $O(n^2)$ | $O(n^2)$ | $O(n^2)$ |
| QuickSort | $O(n \log n)$ | $O(n \log n)$ | $O(n^2)$ |
| ShellSort | $O(n \log n)$ | Depende do <i>gap</i> | $O(n^2)$ |
| MergeSort | $O(n \log n)$ | $O(n \log n)$ | $O(n \log n)$ |
| HeapSort | $O(n \log n)$ | $O(n \log n)$ | $O(n \log n)$ |

Conforme vimos em aula, ambos têm tempo $O(n \log n)$. **Gabarito: C**

11. (CESGRANRIO - 2012 - CMB – Analista de Sistemas – III) Em uma reunião de análise de desempenho de um sistema WEB, um programador apontou corretamente que a complexidade de tempo do algoritmo bubblesort, no pior caso, é:

- a) $O(1)$
- b) $O(\log n)$
- c) $O(n)$



- d) $O(n \log n)$
e) $O(n^2)$

Comentários:

| Algoritmo | Melhor Caso | Caso Médio | Pior Caso |
|---------------|---------------|-----------------------|---------------|
| BubbleSort | $O(n)$ | $O(n^2)$ | $O(n^2)$ |
| InsertionSort | $O(n)$ | $O(n^2)$ | $O(n^2)$ |
| SelectionSort | $O(n^2)$ | $O(n^2)$ | $O(n^2)$ |
| QuickSort | $O(n \log n)$ | $O(n \log n)$ | $O(n^2)$ |
| ShellSort | $O(n \log n)$ | Depende do <i>gap</i> | $O(n^2)$ |
| MergeSort | $O(n \log n)$ | $O(n \log n)$ | $O(n \log n)$ |
| HeapSort | $O(n \log n)$ | $O(n \log n)$ | $O(n \log n)$ |

Conforme vimos em aula, trata-se de $O(n^2)$! **Gabarito: E**

12. (CESPE - 2010 – INMETRO – Analista de Sistemas) Se f é uma função de complexidade para um algoritmo F , então $O(f)$ é considerada a complexidade assintótica ou o comportamento assintótico do algoritmo F . Assinale a opção que apresenta somente algoritmos que possuem complexidade assintótica quando $f(n) = O(n \log n)$.

- a) HeapSort e BubbleSort
b) QuickSort e InsertionSort
c) MergeSort e BubbleSort
d) InsertionSort
e) HeapSort, QuickSort e MergeSort

Comentários:

| Algoritmo | Melhor Caso | Caso Médio | Pior Caso |
|---------------|---------------|---------------|-----------|
| BubbleSort | $O(n)$ | $O(n^2)$ | $O(n^2)$ |
| InsertionSort | $O(n)$ | $O(n^2)$ | $O(n^2)$ |
| SelectionSort | $O(n^2)$ | $O(n^2)$ | $O(n^2)$ |
| QuickSort | $O(n \log n)$ | $O(n \log n)$ | $O(n^2)$ |



| | | | |
|-----------|---------------|-----------------------|---------------|
| ShellSort | $O(n \log n)$ | Depende do <i>gap</i> | $O(n^2)$ |
| MergeSort | $O(n \log n)$ | $O(n \log n)$ | $O(n \log n)$ |
| HeapSort | $O(n \log n)$ | $O(n \log n)$ | $O(n \log n)$ |

Conforme vimos em aula, trata-se da última opção. Vejam que ele não utilizou, nessa questão, o Pior Caso.

Gabarito: E

13. (FGV - 2013 – MPE/MS – Analista de Sistemas) Assinale a alternativa que indica o algoritmo de ordenação capaz de funcionar em tempo $O(n)$ para alguns conjuntos de entrada.

- a) Selectionsort (seleção)
- b) Insertionsort (inserção)
- c) Merge sort
- d) Quicksort
- e) Heapsort

Comentários:

| Algoritmo | Melhor Caso | Caso Médio | Pior Caso |
|---------------|---------------|-----------------------|---------------|
| BubbleSort | $O(n)$ | $O(n^2)$ | $O(n^2)$ |
| InsertionSort | $O(n)$ | $O(n^2)$ | $O(n^2)$ |
| SelectionSort | $O(n^2)$ | $O(n^2)$ | $O(n^2)$ |
| QuickSort | $O(n \log n)$ | $O(n \log n)$ | $O(n^2)$ |
| ShellSort | $O(n \log n)$ | Depende do <i>gap</i> | $O(n^2)$ |
| MergeSort | $O(n \log n)$ | $O(n \log n)$ | $O(n \log n)$ |
| HeapSort | $O(n \log n)$ | $O(n \log n)$ | $O(n \log n)$ |

Conforme vimos em aula, trata-se da segunda opção. **Gabarito: B**

14. (CESGRANRIO - 2010 – BACEN – Analista de Sistemas) Uma fábrica de software foi contratada para desenvolver um produto de análise de riscos. Em determinada funcionalidade desse software, é necessário realizar a ordenação de um conjunto formado por muitos números inteiros. Que algoritmo de ordenação oferece melhor complexidade de tempo (Big O notation) no pior caso?

- a) Merge sort



- b) Insertion sort
- c) Bubble sort
- d) Quick sort
- e) Selection sort

Comentários:

| Algoritmo | Melhor Caso | Caso Médio | Pior Caso |
|---------------|---------------|-----------------------|---------------|
| BubbleSort | $O(n)$ | $O(n^2)$ | $O(n^2)$ |
| InsertionSort | $O(n)$ | $O(n^2)$ | $O(n^2)$ |
| SelectionSort | $O(n^2)$ | $O(n^2)$ | $O(n^2)$ |
| QuickSort | $O(n \log n)$ | $O(n \log n)$ | $O(n^2)$ |
| ShellSort | $O(n \log n)$ | Depende do <i>gap</i> | $O(n^2)$ |
| MergeSort | $O(n \log n)$ | $O(n \log n)$ | $O(n \log n)$ |
| HeapSort | $O(n \log n)$ | $O(n \log n)$ | $O(n \log n)$ |

Conforme vimos em aula, trata-se da primeira opção! **Gabarito: A**

15. (CESPE - 2011 – FUB – Analista de Sistemas) Os métodos de ordenação podem ser classificados como estáveis ou não estáveis. O método é estável se preserva a ordem relativa de dois valores idênticos. Alguns métodos eficientes como shellsort ou quicksort não são estáveis, enquanto alguns métodos pouco eficientes, como o método da bolha, são estáveis.

Comentários:

Métodos Estáveis: BubbleSort, InsertionSort e MergeSort; Métodos Instáveis: SelectionSort, QuickSort, HeapSort e ShellSort. Portanto, item perfeito! **Gabarito: C**

16. (CESPE - 2012 – BASA – Analista de Sistemas) O método de classificação Shellsort iguala-se ao método Quicksort em termos de complexidade temporal, porém é mais eficiente para quantidades pequenas a moderadas de dados.

Comentários:

| Algoritmo | Melhor Caso | Caso Médio | Pior Caso |
|------------|-------------|------------|-----------|
| BubbleSort | $O(n)$ | $O(n^2)$ | $O(n^2)$ |



| | | | |
|---------------|---------------|-----------------------|---------------|
| InsertionSort | $O(n)$ | $O(n^2)$ | $O(n^2)$ |
| SelectionSort | $O(n^2)$ | $O(n^2)$ | $O(n^2)$ |
| QuickSort | $O(n \log n)$ | $O(n \log n)$ | $O(n^2)$ |
| ShellSort | $O(n \log n)$ | Depende do <i>gap</i> | $O(n^2)$ |
| MergeSort | $O(n \log n)$ | $O(n \log n)$ | $O(n \log n)$ |
| HeapSort | $O(n \log n)$ | $O(n \log n)$ | $O(n \log n)$ |

Não, a complexidade temporal é completamente diferente! A complexidade que nós estudamos em aula se trata do comportamento do algoritmo em termos de custo - ela fornece uma resposta digamos que "genérica". Por que? Porque são ignorados constantes, parâmetros, etc - perceba que nós dizemos, por exemplo, que o algoritmo x tem complexidade $O(n^2)$. O que isso significa? Significa apenas que esse algoritmo tem um custo polinomial, mas nós sabemos que funções polinomiais têm o formato $ax^2 + bx + c$. Onde está o a, bx, c? Nós ignoramos e ficamos apenas com o parâmetro mais importante. Além disso tudo, ainda temos que considerar que existe o pior caso, melhor caso e caso médio.

Para saber a complexidade temporal de um algoritmo de ordenação, nós temos que nos focar em diversos parâmetros. Além das constantes e parâmetros da função que representa a complexidade, nós temos que levar em consideração aspectos práticos (Por exemplo: o hardware em que será rodado o programa). Resumindo nosso papo: não há correlação direta entre complexidade de custo e complexidade temporal, logo eu não posso usar a função de um para calcular o outro. **Gabarito: E**

17. (CESPE - 2012 – BASA – Analista de Sistemas) O método de classificação Quicksort é estável e executado em tempo linearmente dependente da quantidade de dados que estão sendo classificados.

Comentários:

| Algoritmo | Melhor Caso | Caso Médio | Pior Caso |
|---------------|---------------|-----------------------|---------------|
| BubbleSort | $O(n)$ | $O(n^2)$ | $O(n^2)$ |
| InsertionSort | $O(n)$ | $O(n^2)$ | $O(n^2)$ |
| SelectionSort | $O(n^2)$ | $O(n^2)$ | $O(n^2)$ |
| QuickSort | $O(n \log n)$ | $O(n \log n)$ | $O(n^2)$ |
| ShellSort | $O(n \log n)$ | Depende do <i>gap</i> | $O(n^2)$ |
| MergeSort | $O(n \log n)$ | $O(n \log n)$ | $O(n \log n)$ |
| HeapSort | $O(n \log n)$ | $O(n \log n)$ | $O(n \log n)$ |

Conforme vimos em aula, QuickSort é instável e não possui complexidade temporal linear! **Gabarito: E**



18. (CESPE - 2012 – BASA – Analista de Sistemas) No método de ordenamento denominado shellsort, as comparações e as trocas são feitas conforme determinada distância entre dois elementos, de modo que, uma distância igual a 6 seria a comparação entre o primeiro elemento e o sétimo, ou entre o segundo elemento e o oitavo, e assim sucessivamente, repetindo-se esse processo até que as últimas comparações e trocas tenham sido efetuadas e a distância tenha diminuído até chegar a 1.

Comentários:

É o algoritmo mais eficiente dentre os de ordem quadrática. Nesse método, as comparações e as trocas são feitas conforme determinada distância (gap) entre dois elementos, de modo que, se gap = 6, há comparação entre o 1º e 7º elementos ou entre o 2º e 8º elementos e assim sucessivamente, repetindo até que as últimas comparações e trocas tenham sido efetuadas e o gap tenha chegado a 1.

Conforme vimos em aula, a questão descreveu perfeitamente o mecanismo de ordenação do Shellsort.

Gabarito: C

19. (FGV - 2008 – PETROBRÁS – Analista de Sistemas) Sobre o algoritmo de ordenação heapsort, assinale a afirmação correta.

- a) Utiliza ordenação por árvore de decisão, ao invés de ordenação por comparação.
- b) A estrutura de dados que utiliza, chamada heap, pode ser interpretada como uma árvore binária.
- c) Seu desempenho de pior caso é pior do que o do algoritmo quicksort.
- d) Seu desempenho de pior caso é o mesmo da ordenação por inserção.
- e) Seu desempenho de pior caso é menor do que o da ordenação por intercalação.

Comentários:

(a) Utiliza ordenação por seleção; (b) Perfeito; (c) Não, é melhor que o QuickSort; (d) Não, é melhor que o InsertionSort; (e) Não, é idêntico ao MergeSort. **Gabarito: B**

20. (CESGRANRIO – 2009 – BASA – Analista de Sistemas) Com relação aos algoritmos quicksort e mergesort, o tempo de execução para o:

- a) pior caso do quicksort é $(n \lg n)$.
- b) pior caso do mergesort é (n^2) .
- c) pior caso do mergesort é $(n \lg n)$.
- d) caso médio do mergesort é $O(\lg n)$.
- e) caso médio do quicksort é $O(n^2)$.

Comentários:



| Algoritmo | Melhor Caso | Caso Médio | Pior Caso |
|---------------|---------------|-----------------------|---------------|
| BubbleSort | $O(n)$ | $O(n^2)$ | $O(n^2)$ |
| InsertionSort | $O(n)$ | $O(n^2)$ | $O(n^2)$ |
| SelectionSort | $O(n^2)$ | $O(n^2)$ | $O(n^2)$ |
| QuickSort | $O(n \log n)$ | $O(n \log n)$ | $O(n^2)$ |
| ShellSort | $O(n \log n)$ | Depende do <i>gap</i> | $O(n^2)$ |
| MergeSort | $O(n \log n)$ | $O(n \log n)$ | $O(n \log n)$ |
| HeapSort | $O(n \log n)$ | $O(n \log n)$ | $O(n \log n)$ |

Conforme vimos em aula, trata-se da terceira opção. **Gabarito: C**

21. (CESPE – 2009 – UNIPAMPA – Analista de Sistemas) O algoritmo quicksort, que divide uma instrução em quatro blocos diferentes de busca, é um exemplo de estrutura de ordenação de dados.

Comentários:

Conforme vimos em aula, são dois blocos diferentes de busca. **Gabarito: E**

22. (CESPE - 2013 – CPRM – Analista de Sistemas) No algoritmo de ordenação denominado quicksort, escolhe-se um ponto de referência, denominado pivô, e separam-se os elementos em dois grupos: à esquerda, ficam os elementos menores que o pivô, e à direita ficam os maiores. Repete-se esse processo para os grupos de elementos formados (esquerda e direita) até que todos os elementos estejam ordenados.

Comentários:

Esse algoritmo divide um conjunto de itens em conjuntos menores, que são ordenados de forma independente, e depois os resultados são combinados para produzir a solução de ordenação do conjunto maior. Trata-se, portanto, de um algoritmo do tipo Divisão-e-Conquista, i.e., repartindo os dados em subgrupos, dependendo de um elemento chamado pivô.

Neste método, a lista é dividida em parte esquerda e parte direita, sendo que os elementos da parte esquerda são todos menores que os elementos da parte direita. Essa fase do processo é chamada de partição. Em seguida, as duas partes são ordenadas recursivamente (usando o próprio QuickSort). A lista está, portanto, ordenada corretamente!

Conforme vimos em aula, é exatamente assim! **Gabarito: C**



23. (CESPE - 2013 – MPU – Analista de Sistemas) Entre os algoritmos de ordenação e pesquisa bubble sort, quicksort e heapsort, o quicksort é considerado o mais eficiente, pois se caracteriza como um algoritmo de dividir-para-conquistar, utilizando operações de particionamento.

Comentários:

Conforme vimos em aula, o HeapSort é o mais eficiente no pior caso! **Gabarito: E**

24. (CESPE - 2013 – TRT/9 – Analista de Sistemas) No método Quicksort, o pivô é responsável pelo número de partições em que o vetor é dividido. Como o pivô não pode ser um elemento que esteja repetido no vetor, o Quicksort não funciona quando há elementos repetidos.

Comentários:

O pivô não é responsável pelo número de partições em que o vetor é dividido. Ademais, ele pode sim ser um elemento que esteja repetido no vetor! **Gabarito: E**

25. (FCC - 2011 - TRT - 14^a Região (RO e AC) - Analista Judiciário - Tecnologia da Informação) NÃO se trata de um método de ordenação (algoritmo):

- a) inserção direta.
- b) seleção direta.
- c) inserção por meio de incrementos decrescentes.
- d) direta em cadeias.
- e) particionamento.

Comentários:

(a) Trata-se do InsertionSort; (b) Trata-se do SelectionSort; (c) Trata-se do ShellSort; (d) Trata-se de um método de busca; (e) Trata-se do QuickSort. Portanto, é a quarta opção! **Gabarito: D**

26. (FCC - 2016 - TRT - 23^a REGIÃO (MT) - Analista Judiciário - Tecnologia da Informação) Considere o método de ordenação abaixo.

```
void ordena(int m,int x[]) {  
    int aux,j,i;  
    for(i=0;i<m-1;i++) {  
        for(j=0;j<m-i-1;j++)  
            if (x[j] > x[j+1]) {  
                aux=x[j];  
                x[j]=x[j+1];  
                x[j+1]=aux;  
            }  
    }  
}
```



Utilizando este algoritmo de ordenação, percorre-se a lista dada da esquerda para a direita, comparando pares de elementos consecutivos, trocando de lugar os que estão fora da ordem. Em cada troca, o maior elemento é deslocado uma posição para a direita. Trata-se de um algoritmo de ordenação:

- a) Select Sort.
- b) Insert Sort.
- c) Bubble Sort.
- d) Shell Sort.
- e) Quick Sort.

Comentários:

O algoritmo realiza trocas de dois em dois, percorrendo todos os elementos. Como vimos, esse algoritmo é o Bubble Sort. **Gabarito: C**



QUESTÕES COMENTADAS – COMPLEXIDADE DE ALGORITMOS - MULTIBANCAS

1. (VUNESP – 2012 – TJ/SP - Analista Judiciário - Tecnologia da Informação) Considerando o conceito de Complexidade de Algoritmos, representado por O(função), assinale a alternativa que apresenta, de forma crescente, as complexidades de algoritmos.
- a) $O(2n)$; $O(n3)$; $O(n2)$; $O(\log_2 n)$; $O(n \cdot \log_2 n)$.
 - b) $O(n2)$; $O(n3)$; $O(2n)$; $O(\log_2 n)$; $O(n \cdot \log_2 n)$.
 - c) $O(n3)$; $O(n2)$; $O(2n)$; $O(n \cdot \log_2 n)$; $O(\log_2 n)$.
 - d) $O(\log_2 n)$; $O(n \cdot \log_2 n)$; $O(n2)$; $O(n3)$; $O(2n)$.
 - e) $O(n \cdot \log_2 n)$; $O(\log_2 n)$; $O(2n)$; $O(n3)$; $O(n2)$.

Comentários:

| Notação | Nome |
|-----------------|-----------------|
| $O(1)$ | Constante |
| $O(\log n)$ | Logarítmica |
| $O[(\log n)^c]$ | Polilogarítmica |
| $O(n)$ | Linear |
| $O(n \log n)$ | – |
| $O(n^2)$ | Quadrática |
| $O(n^3)$ | Cúbica |
| $O(n^c)$ | Polinomial |
| $O(c^n)$ | Exponencial |
| $O(n!)$ | Fatorial |

Conforme vimos em aula, basta consultar a tabelinha! **Gabarito: D**

2. (FCC - 2010 - TRT - 8ª Região (PA e AP) - Analista Judiciário - Tecnologia da Informação) Numa competição de programação, ganhava mais pontos o time que apresentasse o algoritmo mais eficiente para resolver o pior caso de um determinado problema. A complexidade assintótica (notação Big O) dos algoritmos elaborados está ilustrada na tabela abaixo.



| Time | Complexidade |
|----------|---------------|
| Branco | $O(n^{20})$ |
| Amarelo | $O(n \log n)$ |
| Azul | $O(1)$ |
| Verde | $O(n!)$ |
| Vermelho | $O(2^n)$ |

O time que obteve a medalha de prata (2º algoritmo mais eficiente) é o:

- a) Branco.
- b) Amarelo.
- c) Azul.
- d) Verde.
- e) Vermelho.

Comentários:

| Notação | Nome |
|-----------------|-----------------|
| $O(1)$ | Constante |
| $O(\log n)$ | Logarítmica |
| $O[(\log n)^c]$ | Polilogarítmica |
| $O(n)$ | Linear |
| $O(n \log n)$ | – |
| $O(n^2)$ | Quadrática |
| $O(n^3)$ | Cúbica |
| $O(n^c)$ | Polinomial |
| $O(c^n)$ | Exponencial |
| $O(n!)$ | Fatorial |

Conforme vimos em aula, basta consultar a tabelinha! O primeiro é o time Azul ($O(1)$) e o segundo é o time Amarelo ($O(n \log n)$). **Gabarito: B**



QUESTÕES COMENTADAS – PESQUISA DE DADOS - MULTIBANCAS

1. (CESPE - 2013 – TRT/MS – Analista de Sistemas) Considerando que se deseja efetuar uma pesquisa de um valor sobre a chave primária de uma tabela de um banco de dados com uma chave primária com um tipo de campo que receba um valor inteiro e que se possa fazer essa pesquisa utilizando-se a busca sequencial ou a busca binária, assinale a opção correta.
- a) O método de busca binária requer, no máximo, $\ln(n)$ comparações para determinar o elemento pesquisado, em que n é o número de registros.
 - b) O método de busca binária será sempre mais rápido que o método de busca sequencial, independentemente de a tabela estar ordenada com base no elemento pesquisado.
 - c) O método de busca sequencial requererá, no máximo, n^2 comparações para determinar o elemento pesquisado, em que n será o número de registros.
 - d) O método de busca binária sempre efetuará menos comparações que o método de pesquisa sequencial.
 - e) O método de busca sequencial efetuará menos comparações para encontrar o elemento pesquisado quando a tabela estiver ordenada em comparação à situação quando a tabela estiver desordenada.

Comentários:

A Busca Binária é um algoritmo de busca em vetores que segue o paradigma de divisão-e-conquista. Parte-se do pressuposto de que o vetor está ordenado e realiza sucessivas divisões do espaço de busca, comparando o elemento chave com o elemento do meio do vetor. Possui complexidade da ordem de $O(\log_2 n)$, em que N é o tamanho do vetor de busca.

- a) Conforme vimos em aula, não é logaritmo neperiano (na verdade, é Base 2), mas a banca considerou correto mesmo assim.

A Busca Sequencial é muito lenta para grandes quantidades de dados, mas aceitável para listas pequenas e que mudam constantemente. Observa-se que no Melhor Caso, X está na primeira posição, logo necessita apenas de uma comparação; no Pior Caso, X está na última posição, logo necessita de N comparações; e no Caso Médio, X é encontrado após $(n+1)/2$ comparações.

- b) Conforme vimos em aula, isso não ocorre sempre! Se compararmos o Melhor Caso da Busca Sequencial com o Pior Caso da Busca Binária, a primeira será mais rápida.

Considerando que o vetor $L[]$ contém N elementos, ordenados ou não, é fácil verificar que a busca sequencial requer tempo linearmente proporcional ao tamanho do vetor, i.e., da ordem $O(n)$. Por conta disso, é comum dizer que a busca sequencial é uma Busca Linear. Entenderam? Quanto maior o vetor, maior o tempo em média para buscar um elemento! Quanto mais ao final, mais demorado.

- c) Conforme vimos em aula, ele possui complexidade da ordem de $O(n)$.

A Busca Sequencial é muito lenta para grandes quantidades de dados, mas aceitável para listas pequenas e que mudam constantemente. Observa-se que no Melhor Caso, X está na primeira posição, logo necessita apenas de uma comparação; no Pior Caso, X está na última posição, logo necessita de N comparações; e no Caso Médio, X é encontrado após $(n+1)/2$ comparações.



d) Conforme vimos em aula, isso não ocorre sempre! Se compararmos o Melhor Caso da Busca Sequencial com o Pior Caso da Busca Binária, a primeira será mais rápida.

Considerando que o vetor $L[]$ contém N elementos, ordenados ou não, é fácil verificar que a busca sequencial requer tempo linearmente proporcional ao tamanho do vetor, i.e., da ordem $O(n)$. Por conta disso, é comum dizer que a busca sequencial é uma Busca Linear. Entenderam? Quanto maior o vetor, maior o tempo em média para buscar um elemento! Quanto mais ao final, mais demorado.

e) Conforme vimos em aula, não é necessário que a lista esteja ordenada. Logo, isso não fará diferença.

Gabarito: A

2. (ESAF - 2001 – BACEN – Analista de Sistemas) Na pior hipótese, o número de comparações necessárias para pesquisar um elemento em um array de 2048 elementos pelo método de pesquisa binária será:

- a) 8
- b) 9
- c) 10
- d) 11
- e) 12

Comentários:

A Busca Binária é um algoritmo de busca em vetores que segue o paradigma de divisão-e-conquista. Parte-se do pressuposto de que o vetor está ordenado e realiza sucessivas divisões do espaço de busca, comparando o elemento chave com o elemento do meio do vetor. Possui complexidade da ordem de $O(\log_2 n)$, em que N é o tamanho do vetor de busca.

Conforme vimos em aula, a complexidade da Busca Binária é $O(\log_2 n)$, em que N é o tamanho do vetor de busca – no nosso caso, 2048! Quanto é $\log_2 2048$? 11! Por que, professor? Porque $2^{11} = 2048$! **Gabarito: D**

3. (CESPE - 2013 – TCE/RO – Analista de Sistemas) Considere uma tabela de um banco de dados com chave primária e tipo de campo que receba um valor inteiro. Ao se efetuar uma pesquisa de um valor sobre a chave primária dessa tabela, o método de busca binária requer, no máximo, $\lg(n)$ comparações para localizar o elemento pesquisado, em que n é o número de registros.

Comentários:

A Busca Binária é um algoritmo de busca em vetores que segue o paradigma de divisão-e-conquista. Parte-se do pressuposto de que o vetor está ordenado e realiza sucessivas divisões do espaço de busca, comparando o elemento chave com o elemento do meio do vetor. Possui complexidade da ordem de $O(\log_2 n)$, em que N é o tamanho do vetor de busca.

Conforme vimos em aula, de fato a complexidade da Busca Binária é $O(\log_2 n)$. **Gabarito: C**



4. (FCC – 2016 - TRT - 14ª Região (RO e AC) - Analista Judiciário - Tecnologia da Informação) Dada uma coleção de n elementos ordenados por ordem crescente, pretende-se saber se um determinado elemento x existe nessa coleção. Supondo que essa coleção está implementada como sendo um vetor $a[0...n-1]$ de n elementos inteiros, utilizando-se um algoritmo de pesquisa binária, o número de vezes que a comparação $x==a[i]$ será executada, no pior caso, é calculada por:

- a) $n/2$.
- b) $n-1$.
- c) \sqrt{n} .
- d) $\log_2(n)$.
- e) $n-2$.

Comentários:

A Busca Binária é um algoritmo de busca em vetores que segue o paradigma de divisão-e-conquista. Parte-se do pressuposto de que o vetor está ordenado e realiza sucessivas divisões do espaço de busca, comparando o elemento chave com o elemento do meio do vetor. Possui complexidade da ordem de $O(\log_2 n)$, em que N é o tamanho do vetor de busca. **Gabarito: D**



LISTA DE QUESTÕES – MÉTODOS DE ORDENAÇÃO - MULTIBANCAS

1. (Instituto Cidades - 2012 - TCM-GO - Auditor de Controle Externo - Informática) São exemplos de algoritmos de ordenação, exceto:
 - a) Bubble Sort
 - b) Select Sort
 - c) Shell Sort
 - d) Busca Sequencial;
 - e) Quick Sort;
2. (FUMARC - 2012 - TJ-MG - Técnico Judiciário - Analista de Sistemas – I) Quicksort divide um conjunto de itens em conjuntos menores, que são ordenados de forma independe, e depois os resultados são combinados para produzir a solução de ordenação do conjunto maior.
3. (CESPE - 2012 - MPE-PI - Analista Ministerial - Informática - Cargo 6) O heapsort é um algoritmo de ordenação em que a quantidade de elementos armazenada fora do arranjo de entrada é constante durante toda a sua execução.
4. (CESPE - 2010 - ABIN - Oficial Técnico de Inteligência - Área de Suporte a Rede de Dados) A eficácia do método de ordenação rápida (quicksort) depende da escolha do pivô mais adequado ao conjunto de dados que se deseja ordenar. A situação ótima ocorre quando o pivô escolhido é igual ao valor máximo ou ao valor mínimo do conjunto de dados.
5. (CESPE - 2010 - ABIN - Oficial Técnico de Inteligência - Área de Suporte a Rede de Dados) A estabilidade de um método de ordenação é importante quando o conjunto de dados já está parcialmente ordenado.
6. (CESPE - 2010 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação - Administração de Dados) A classificação interna por inserção é um método que realiza a ordenação de um vetor por meio da inserção de cada elemento em sua posição correta dentro de um subvetor classificado.
7. (FCC - 2009 - TRT - 15ª Região - Analista Judiciário - Tecnologia da Informação) São algoritmos de classificação por trocas apenas os métodos:
 - a) SelectionSort e InsertionSort.
 - b) MergeSort e BubbleSort.
 - c) QuickSort e SelectionSort.



- d) BubbleSort e QuickSort.
- e) InsertionSort e MergeSort.
8. (CESGRANRIO - 2011 - PETROBRÁS – Analista de Sistemas – I) O tempo de pior caso do algoritmo QuickSort é de ordem menor que o tempo médio do algoritmo Bubblesort.
9. (CESGRANRIO - 2011 - PETROBRÁS – Analista de Sistemas – II) O tempo médio do QuickSort é $O(n \log 2n)$, pois ele usa como estrutura básica uma árvore de prioridades.
10. (CESGRANRIO - 2011 - PETROBRÁS – Analista de Sistemas – III) O tempo médio do QuickSort é de ordem igual ao tempo médio do MergeSort.
11. (CESGRANRIO - 2012 - CMB – Analista de Sistemas – III) Em uma reunião de análise de desempenho de um sistema WEB, um programador apontou corretamente que a complexidade de tempo do algoritmo bubblesort, no pior caso, é:
- a) $O(1)$
 - b) $O(\log n)$
 - c) $O(n)$
 - d) $O(n \log n)$
 - e) $O(n^2)$
12. (CESPE - 2010 – INMETRO – Analista de Sistemas) Se f é uma função de complexidade para um algoritmo F , então $O(f)$ é considerada a complexidade assintótica ou o comportamento assintótico do algoritmo F . Assinale a opção que apresenta somente algoritmos que possuem complexidade assintótica quando $f(n) = O(n \log n)$.
- a) HeapSort e BubbleSort
 - b) QuickSort e InsertionSort
 - c) MergeSort e BubbleSort
 - d) InsertionSort
 - e) HeapSort, QuickSort e MergeSort
13. (FGV - 2013 – MPE/MS – Analista de Sistemas) Assinale a alternativa que indica o algoritmo de ordenação capaz de funcionar em tempo $O(n)$ para alguns conjuntos de entrada.
- a) Selectionsort (seleção)
 - b) Insertionsort (inserção)
 - c) Merge sort



- d) Quicksort
- e) Heapsort

14. (CESGRANRIO - 2010 – BACEN – Analista de Sistemas) Uma fábrica de software foi contratada para desenvolver um produto de análise de riscos. Em determinada funcionalidade desse software, é necessário realizar a ordenação de um conjunto formado por muitos números inteiros. Que algoritmo de ordenação oferece melhor complexidade de tempo (Big O notation) no pior caso?

- a) Merge sort
- b) Insertion sort
- c) Bubble sort
- d) Quick sort
- e) Selection sort

15. (CESPE - 2011 – FUB – Analista de Sistemas) Os métodos de ordenação podem ser classificados como estáveis ou não estáveis. O método é estável se preserva a ordem relativa de dois valores idênticos. Alguns métodos eficientes como shellsort ou quicksort não são estáveis, enquanto alguns métodos pouco eficientes, como o método da bolha, são estáveis.

16. (CESPE - 2012 – BASA – Analista de Sistemas) O método de classificação Shellsort iguala-se ao método Quicksort em termos de complexidade temporal, porém é mais eficiente para quantidades pequenas a moderadas de dados.

17. (CESPE - 2012 – BASA – Analista de Sistemas) O método de classificação Quicksort é estável e executado em tempo linearmente dependente da quantidade de dados que estão sendo classificados.

18. (CESPE - 2012 – BASA – Analista de Sistemas) No método de ordenamento denominado shellsort, as comparações e as trocas são feitas conforme determinada distância entre dois elementos, de modo que, uma distância igual a 6 seria a comparação entre o primeiro elemento e o sétimo, ou entre o segundo elemento e o oitavo, e assim sucessivamente, repetindo-se esse processo até que as últimas comparações e trocas tenham sido efetuadas e a distância tenha diminuído até chegar a 1.

19. (FGV - 2008 – PETROBRÁS – Analista de Sistemas) Sobre o algoritmo de ordenação heapsort, assinale a afirmação correta.

- a) Utiliza ordenação por árvore de decisão, ao invés de ordenação por comparação.
- b) A estrutura de dados que utiliza, chamada heap, pode ser interpretada como uma árvore binária.
- c) Seu desempenho de pior caso é pior do que o do algoritmo quicksort.
- d) Seu desempenho de pior caso é o mesmo da ordenação por inserção.
- e) Seu desempenho de pior caso é menor do que o da ordenação por intercalação.



20. (CESGRANRIO – 2009 – BASA – Analista de Sistemas) Com relação aos algoritmos quicksort e mergesort, o tempo de execução para o:

- a) pior caso do quicksort é $(n \lg n)$.
- b) pior caso do mergesort é (n^2) .
- c) pior caso do mergesort é $(n \lg n)$.
- d) caso médio do mergesort é $O(\lg n)$.
- e) caso médio do quicksort é $O(n^2)$.

21. (CESPE – 2009 – UNIPAMPA – Analista de Sistemas) O algoritmo quicksort, que divide uma instrução em quatro blocos diferentes de busca, é um exemplo de estrutura de ordenação de dados.

22. (CESPE - 2013 – CPRM – Analista de Sistemas) No algoritmo de ordenação denominado quicksort, escolhe-se um ponto de referência, denominado pivô, e separam-se os elementos em dois grupos: à esquerda, ficam os elementos menores que o pivô, e à direita ficam os maiores. Repete-se esse processo para os grupos de elementos formados (esquerda e direita) até que todos os elementos estejam ordenados.

23. (CESPE - 2013 – MPU – Analista de Sistemas) Entre os algoritmos de ordenação e pesquisa bubble sort, quicksort e heapsort, o quicksort é considerado o mais eficiente, pois se caracteriza como um algoritmo de dividir-para-conquistar, utilizando operações de particionamento.

24. (CESPE - 2013 – TRT/9 – Analista de Sistemas) No método Quicksort, o pivô é responsável pelo número de partições em que o vetor é dividido. Como o pivô não pode ser um elemento que esteja repetido no vetor, o Quicksort não funciona quando há elementos repetidos.

25. (FCC - 2011 - TRT - 14^a Região (RO e AC) - Analista Judiciário - Tecnologia da Informação) NÃO se trata de um método de ordenação (algoritmo):

- a) inserção direta.
- b) seleção direta.
- c) inserção por meio de incrementos decrescentes.
- d) direta em cadeias.
- e) particionamento.

26. (FCC - 2016 - TRT - 23^a REGIÃO (MT) - Analista Judiciário - Tecnologia da Informação)

Considere o método de ordenação abaixo.



```
void ordena(int m,int x[]) {  
    int aux,j,i;  
    for(i=0;i<m-1;i++) {  
        for(j=0;j<m-i-1;j++)  
            if (x[j] > x[j+1]) {  
                aux=x[j];  
                x[j]=x[j+1];  
                x[j+1]=aux;  
            }  
    }  
}
```

Utilizando este algoritmo de ordenação, percorre-se a lista dada da esquerda para a direita, comparando pares de elementos consecutivos, trocando de lugar os que estão fora da ordem. Em cada troca, o maior elemento é deslocado uma posição para a direita. Trata-se de um algoritmo de ordenação:

- a) Select Sort.
- b) Insert Sort.
- c) Bubble Sort.
- d) Shell Sort.
- e) Quick Sort.

GABARITO



1. D

2. C

3. C



- | | | |
|-------|-------|-------|
| 4. E | 12. E | 20. C |
| 5. E | 13. B | 21. E |
| 6. C | 14. A | 22. C |
| 7. D | 15. C | 23. E |
| 8. E | 16. E | 24. E |
| 9. E | 17. E | 25. D |
| 10. C | 18. C | 26. C |
| 11. E | 19. B | |



LISTA DE QUESTÕES – COMPLEXIDADE DE ALGORITMOS – MULTIBANCAS

1. (VUNESP – 2012 – TJ/SP - Analista Judiciário - Tecnologia da Informação) Considerando o conceito de Complexidade de Algoritmos, representado por O(função), assinale a alternativa que apresenta, de forma crescente, as complexidades de algoritmos.
 - a) $O(2n)$; $O(n3)$; $O(n2)$; $O(\log_2 n)$; $O(n \cdot \log_2 n)$.
 - b) $O(n2)$; $O(n3)$; $O(2n)$; $O(\log_2 n)$; $O(n \cdot \log_2 n)$.
 - c) $O(n3)$; $O(n2)$; $O(2n)$; $O(n \cdot \log_2 n)$; $O(\log_2 n)$.
 - d) $O(\log_2 n)$; $O(n \cdot \log_2 n)$; $O(n2)$; $O(n3)$; $O(2n)$.
 - e) $O(n \cdot \log_2 n)$; $O(\log_2 n)$; $O(2n)$; $O(n3)$; $O(n2)$.
2. (FCC - 2010 - TRT - 8ª Região (PA e AP) - Analista Judiciário - Tecnologia da Informação) Numa competição de programação, ganhava mais pontos o time que apresentasse o algoritmo mais eficiente para resolver o pior caso de um determinado problema. A complexidade assintótica (notação Big O) dos algoritmos elaborados está ilustrada na tabela abaixo.

| Time | Complexidade |
|----------|---------------|
| Branco | $O(n^{20})$ |
| Amarelo | $O(n \log n)$ |
| Azul | $O(1)$ |
| Verde | $O(n!)$ |
| Vermelho | $O(2^n)$ |

O time que obteve a medalha de prata (2º algoritmo mais eficiente) é o:

- a) Branco.
- b) Amarelo.
- c) Azul.
- d) Verde.
- e) Vermelho.



GABARITO

GABARITO



1. D
2. B



LISTA DE QUESTÕES – PESQUISA DE DADOS - MULTIBANCAS

1. (CESPE - 2013 – TRT/MS – Analista de Sistemas) Considerando que se deseja efetuar uma pesquisa de um valor sobre a chave primária de uma tabela de um banco de dados com uma chave primária com um tipo de campo que receba um valor inteiro e que se possa fazer essa pesquisa utilizando-se a busca sequencial ou a busca binária, assinale a opção correta.
 - a) O método de busca binária requer, no máximo, $\ln(n)$ comparações para determinar o elemento pesquisado, em que n é o número de registros.
 - b) O método de busca binária será sempre mais rápido que o método de busca sequencial, independentemente de a tabela estar ordenada com base no elemento pesquisado.
 - c) O método de busca sequencial requererá, no máximo, n^2 comparações para determinar o elemento pesquisado, em que n será o número de registros.
 - d) O método de busca binária sempre efetuará menos comparações que o método de pesquisa sequencial.
 - e) O método de busca sequencial efetuará menos comparações para encontrar o elemento pesquisado quando a tabela estiver ordenada em comparação à situação quando a tabela estiver desordenada.
2. (ESAF - 2001 – BACEN – Analista de Sistemas) Na pior hipótese, o número de comparações necessárias para pesquisar um elemento em um array de 2048 elementos pelo método de pesquisa binária será:
 - a) 8
 - b) 9
 - c) 10
 - d) 11
 - e) 12
3. (CESPE - 2013 – TCE/RO – Analista de Sistemas) Considere uma tabela de um banco de dados com chave primária e tipo de campo que receba um valor inteiro. Ao se efetuar uma pesquisa de um valor sobre a chave primária dessa tabela, o método de busca binária requer, no máximo, $\lg(n)$ comparações para localizar o elemento pesquisado, em que n é o número de registros.
4. (FCC – 2016 - TRT - 14ª Região (RO e AC) - Analista Judiciário - Tecnologia da Informação) Dada uma coleção de n elementos ordenados por ordem crescente, pretende-se saber se um determinado elemento x existe nessa coleção. Supondo que essa coleção está implementada como sendo um vetor $a[0...n-1]$ de n elementos inteiros, utilizando-se um algoritmo de pesquisa binária, o número de vezes que a comparação $x==a[i]$ será executada, no pior caso, é calculada por:
 - a) $n/2$.
 - b) $n-1$.



- c) \sqrt{n} .
- d) $\log_2(n)$.
- e) $n=2$.

GABARITO



GABARITO



- 1. A
- 2. D
- 3. C
- 4. D



ESTRUTURA DE DADOS

Pessoal, um programa pode ser visto como uma especificação formal da solução de um problema. Wirth expressa esse conceito por meio de uma **equação**:

$$\text{PROGRAMA} = \text{ALGORITMO} + \text{ESTRUTURA DE DADOS}$$

Nosso foco aqui é em Estruturas de Dados! Na evolução do mundo computacional, um fator extremamente importante trata da forma de **armazenar informações**. De nada adianta o enorme desenvolvimento de hardware e software se a forma de armazenamento e tratamento de dados não evoluir harmonicamente. E é por isso que as estruturas de dados são tão fundamentais.

As estruturas de dados, na maioria dos casos, baseiam-se nos tipos de armazenamento vistos dia a dia, i.e., nada mais são do que a transformação de uma forma de armazenamento já conhecida e utilizada no mundo real adaptada para o mundo computacional. Por isso, cada tipo de estrutura de dados possui **vantagens e desvantagens** e cada uma tem sua **área de atuação otimizada**.

Bem, não vou enrolar muito explicando o que é uma Estrutura de Dados! A melhor forma de saber é vendo exemplos. Antes disso, eu gostaria de falar sobre um conceito importante: **Dados Homogêneos e Heterogêneos**. Os primeiros são aqueles que possuem só um tipo básico de dados (Ex: Inteiros); os segundos são aqueles que possuem mais de um tipo básico de dados (Ex: Inteiros + Caracteres). Os tipos básicos de dados também são chamados de tipos primitivos.

Entenderam? Existem estruturas de dados que tratam de dados homogêneos, i.e., todos os dados são apenas de um tipo básico, tais como **Vetores**! Ora, em um vetor, todos os elementos são do mesmo tipo. Existem estruturas de dados que tratam de dados heterogêneos, i.e., os dados são de tipos básicos diferentes, tais como Listas! Ora, em uma lista, todos os elementos são, em geral, de tipos básicos diferentes.

Além dessa classificação, existe outra também importante: **Estruturas Lineares e Estruturas Não-Lineares**. As Estruturas Lineares são aquelas em que cada elemento pode ter um único predecessor (exceto o primeiro elemento) e um único sucessor (exceto o último elemento). Como exemplo, podemos citar Listas, Pilhas, Filas, Arranjos, entre outros.

Já as Estruturas Não-Lineares são aquelas em que cada elemento pode ter mais de um predecessor e/ou mais de um sucessor. Como exemplo, podemos citar Árvores, Grafos e Tabelas de Dispersão. Essa é uma classificação muito importante e muito simples de entender. Pessoal,



vocês perceberão que esse assunto é cobrado de maneira superficial na maioria das questões, mas algumas são nível doutorado!

Por fim, vamos falar sobre Tipos Abstratos de Dados (TAD). Podemos defini-lo como um modelo matemático (v, o), em que v é um conjunto de valores e o é um conjunto de operações que podem ser realizadas sobre valores. Eu sei, essa definição é horrível de entender! Para compreender esse conceito, basta lembrar de **abstração**, i.e., apresentar **interfaces** e esconder **detalhes**.

Os **Tipos Abstratos de Dados** são simplesmente um modelo para um certo tipo de estrutura de dados. Como assim, professor? Quando eu falo em pilha, eu estou falando de um tipo abstrato de dados que tem duas operações com comportamentos bem definidos e conhecidos: push (para inserir elementos na pilha); e pop (para retirar elementos da pilha).

E a implementação dessas operações? Isso não importa! Aliás, não importa implementação nem paradigma nem linguagem de programação. Não importa se a pilha é implementada com um paradigma orientado a objetos ou com um paradigma estruturado; não importa se a pilha é implementada em Java ou Pascal; não importa como é a implementação interna – isso serve para outras estruturas¹.

Podemos concluir, portanto, que um tipo abstrato de dados contém um modelo que contém valores e operações associadas, de forma que essas operações sejam precisamente independentes de uma implementação particular. Em geral, um TAD é especificado por meio de uma especificação algébrica que, em geral, contém três partes: **Especificação Sintática, Semântica e de Restrições**.

A Especificação Sintática define o nome do tipo, suas operações e o tipo dos argumentos das operações, definindo a assinatura do TAD. A Especificação Semântica descreve **propriedades e efeitos** das operações de forma independente de uma implementação específica. E a Especificação de Restrições estabelece as condições que devem ser satisfeitas antes e depois da aplicação das operações.

Em outras palavras, o nível semântico trata do comportamento de um tipo abstrato de dados; e o nível sintático trata da apresentação de um tipo abstrato de dados. Podemos dizer, então, que o TAD encapsula uma **estrutura de dados** com características **semelhantes** – podendo ser formado por outros TADs –, e **esconde** a efetiva implementação dessa **estrutura** de quem a manipula.

¹ Filas, Pilhas, Árvores, Deques, entre outros.

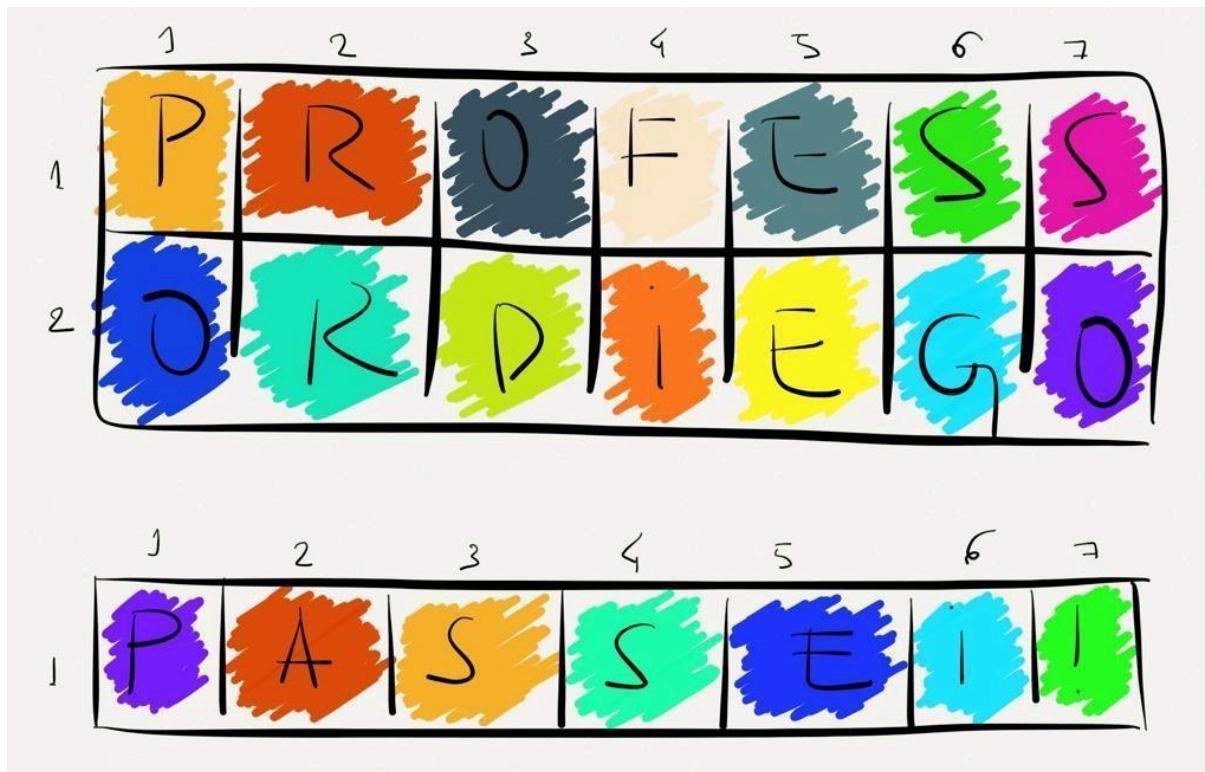


VETORES E MATRIZES

Um Vetor é uma estrutura de **dados linear** que necessita de somente **um índice** para que seus elementos sejam **indexados**. É uma estrutura homogênea, portanto armazena somente uma lista de valores do mesmo tipo. Ele pode ser estático ou dinâmico, com dados armazenados em posições contíguas de memória e permite o acesso direto ou aleatório a seus elementos.

Observem que, diferentemente das listas, filas e pilhas, ele vem praticamente **embutido** em qualquer **linguagem de programação**. E a *Matriz, professor?* Não muda muita coisa! Trata-se de um arranjo bidimensional ou multidimensional de alocação estática e sequencial. Ela necessita de um índice para referenciar a linha e outro para referenciar a coluna para que seus elementos sejam endereçados.

Da mesma forma que um vetor, uma matriz também pode ter tamanhos variados, todos os elementos são do mesmo tipo, cada célula contém somente um valor e os tamanhos dos valores são os mesmos. Os elementos ocupam posições contíguas na memória. A alocação dos elementos pode ser feita colocando os elementos **linha-por-linha ou coluna-por-coluna**.



Matriz 2x7 e Vetor (7 Posições)

(IBFC – 2022 – DETRAN-AM) Relacione as duas colunas quanto aos respectivos tipos de Estruturas de Dados:

| | |
|---------------|------------------|
| (A) Vetores | (1) Homogêneas |
| (B) Registros | (2) Heterogêneas |
| (C) Matrizes | |



- A2 - B1 - C2
- A1 - B1 - C2
- A2 - B2 - C1
- A1 - B2 - C1

Comentários:

Por definição, temos que:

Estruturas de dados homogêneas: seus elementos possuem o mesmo tipo de dado básico.

Estrutura de dados heterogênea: seus elementos possuem tipos de dados distintos.

Via de regra, um vetor e uma matriz possuem sempre o mesmo tipo de dados: um vetor de inteiro, um vetor de string, um vetor de booleanos, e assim por diante. Portanto, são homogêneos.

Já um registro é um agrupamento de várias variáveis, cada uma podendo ter um tipo de dados diferente.

Portanto, é heterogêneo. Assim, A1 – B2 – C1. **Gabarito:** Letra D

(UFRPE – 2022 – UFRPE) Sobre algoritmos e estrutura de dados, assinale a afirmativa correta.

- a) Listas encadeadas ou ligadas são estruturas de dados estáticas, o que significa que o número de nós não pode ser modificado durante a execução do programa.
- b) Pilhas são estruturas de dados do tipo FIFO (first-in first-out), em que o primeiro elemento a ser inserido será o primeiro a ser retirado.
- c) Árvores são estruturas de dados do tipo FIFO (first-in first-out), em que o primeiro elemento a ser inserido será o primeiro a ser retirado.
- d) Filas podem ser implementadas em listas encadeadas ou em vetores.
- e) Pilhas só podem ser implementadas em listas encadeadas.

Comentários:

Para começar, as listas encadeadas são estruturas de dados dinâmicas, não estáticas, permitindo que o número de nós seja modificado durante a execução do programa, contrariando a primeira afirmação. Portanto, a afirmação sobre listas encadeadas serem estáticas é incorreta. Além disso, a caracterização das pilhas como estruturas do tipo FIFO (first-in first-out) também está equivocada, pois na realidade elas operam segundo o princípio LIFO, em que o último elemento inserido é o primeiro a ser retirado. Quanto às árvores, a afirmação de que operam sob o princípio FIFO é igualmente falsa, pois as árvores são estruturas hierárquicas e não sequenciais, não se enquadando nessa descrição de operação. Já a afirmação de que pilhas só podem ser implementadas em listas encadeadas também não é verdadeira, visto que pilhas podem ser implementadas tanto em listas encadeadas quanto em vetores, oferecendo flexibilidade na escolha da implementação.

A única afirmação correta é a que menciona que filas podem ser implementadas em listas encadeadas ou em vetores. Esta é uma descrição acurada das possibilidades de implementação de filas, uma vez que ambas as estruturas permitem o funcionamento adequado do modelo FIFO, no qual o primeiro elemento inserido é o primeiro a ser retirado. Assim, nosso gabarito é a alternativa D. (**Gabarito:** Letra D).

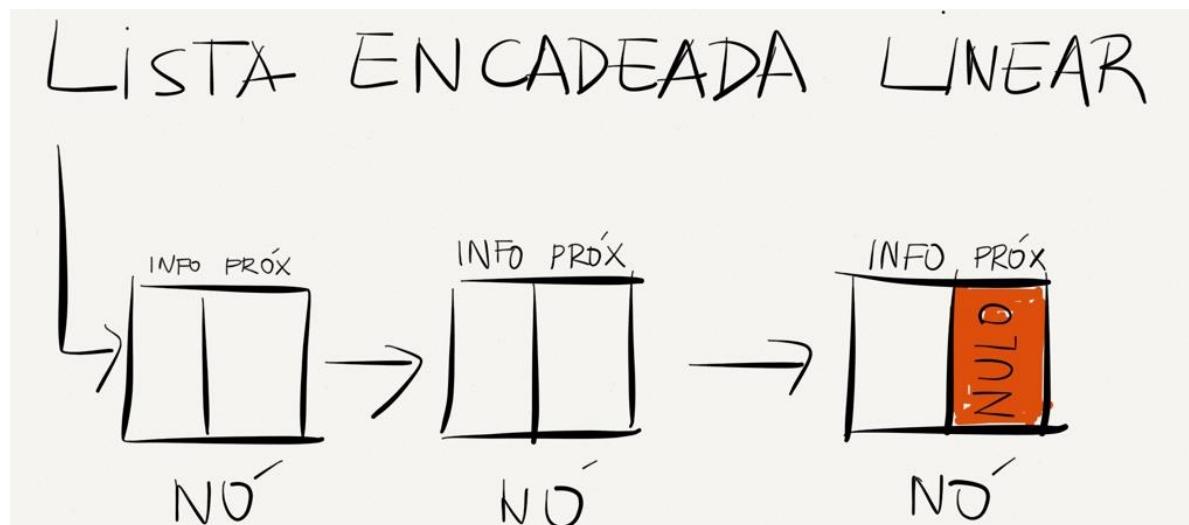


LISTA ENCADEADA

Também conhecida como Lista Encadeada Linear, Lista Ligada Linear ou Linked List, trata-se de uma estrutura de dados dinâmica formada por uma sequência encadeada de elementos chamados nós, que contêm dois campos: **campo de informação** e **campo de endereço**. O primeiro armazena o real elemento da lista e o segundo contém o endereço do próximo nó da lista.

Esse endereço, que é usado para acessar determinado nó, é conhecido como **ponteiro**. A lista ligada inteira é acessada a partir de um ponteiro externo que aponta para o primeiro nó na lista, i.e., contém o endereço do primeiro nó¹. Por ponteiro "externo", entendemos aquele que não está incluído dentro de um nó. Em vez disso, seu valor pode ser acessado diretamente, por referência a uma variável.

O campo do próximo endereço do último nó na lista contém um valor especial, conhecido como **NULL**, que não é um endereço válido. Esse ponteiro nulo é usado para indicar o final de uma lista. Uma lista é chamada Lista Vazia ou Lista Nula caso não tenha nós ou tenha apenas um nó sentinela. O valor do ponteiro externo para esta lista é o ponteiro nulo. Uma lista pode ser inicializada com uma lista vazia.



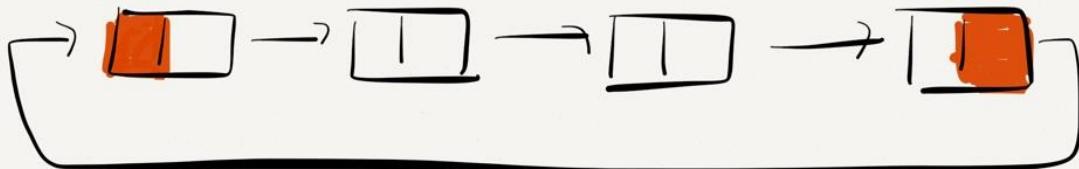
Suponha que seja feita uma mudança na estrutura de uma lista linear, de modo que o campo próximo no último nó contenha um ponteiro de volta para o primeiro nó, em vez de um ponteiro nulo. Esse tipo de lista é chamado **Lista Circular** (ou Fechada²), i.e., a partir de qualquer ponto, é possível atingir qualquer outro ponto da lista. Certinho, até agora?

¹ O endereço do primeiro nó pode ser encapsulado para facilitar possíveis futuras operações sobre a lista sem a necessidade de se conhecer sua estrutura interna. O primeiro elemento e o último nós são muitas vezes chamados de *Sentinela*.

² Se Listas Circulares são conhecidas como Listas Fechadas, as Listas Abertas são todas aquelas que são Não-Circulares. Por fim: da mesma forma que há Listas Circulares Simples, há também Listas Circulares Duplas. Nesse caso, o ponteiro anterior do primeiro elemento aponta para o último elemento e o ponteiro posterior do último elemento aponta para o primeiro elemento.

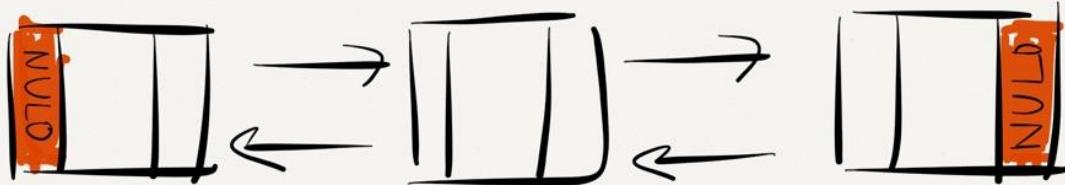


LISTAS CIRCULARES



Observe que uma Lista Circular não tem um primeiro ou último nó natural. Precisamos, portanto, estabelecer um **primeiro** e um **último nó** por convenção. Uma convenção útil é permitir que o ponteiro externo para a lista circular aponte para o último nó, e que o nó seguinte se torne o primeiro nó. Assim podemos incluir ou remover um elemento convenientemente a partir do início ou do final de uma lista.

LISTA DUPLAMENTE ENCADEADA



Embora uma lista circularmente ligada tenha vantagens sobre uma lista linear, ela ainda apresenta várias deficiências. Não se pode atravessar uma lista desse tipo no sentido contrário nem um nó pode ser eliminado de uma lista circularmente ligada sem se ter um ponteiro para o nó antecessor. Nos casos em que tais recursos são necessários, a estrutura de dados adequada é uma **lista duplamente ligada**.

Cada nó numa lista desse tipo contém **dois ponteiros**, um para seu **predecessor** e outro para seu **sucessor**. Na realidade, no contexto de listas duplamente ligadas, os termos predecessor e sucessor não fazem sentido porque a lista é totalmente simétrica. As listas duplamente ligadas podem ser lineares ou circulares e podem conter ou não um nó de cabeçalho.

Podemos considerar os nós numa lista duplamente ligada como consistindo em três campos: um campo info que contém as informações armazenadas no nó, e os campos left e right, que contêm ponteiros para os nós em ambos os lados. Dado um ponteiro para um elemento, pode-se acessar os elementos adjacentes e, dado um ponteiro para o último elemento, pode-se percorrer a lista em **ordem inversa**.



Existem **cinco** operações básicas sobre uma lista encadeada: Criação, em que se cria a lista na memória; Busca, em que se pesquisa nós na lista; Inclusão, em que se insere novos nós na lista em uma determinada posição; Remoção, em que se elimina um elemento da lista; e, por fim, Destruição, em que se destrói a lista junto com todos os seus nós.

IMPORTANTE

Pilhas e Filas são subespécies de Listas. No entanto, cuidado na hora de responder questões! De maneira genérica, Pilhas e Filas podem ser implementadas como Listas. No entanto, elas possuem características particulares de uma lista genérica. Ok?

Precisamos falar um pouco sobre Fragmentação! O que é isso, professor? Galera, falou em **fragmentação**, lembrem-se de **desperdício** de espaço disponível de **memória**. O fenômeno no qual existem vários blocos disponíveis pequenos e não-contíguos é chamado fragmentação externa porque o espaço disponível é desperdiçado fora dos blocos alocados.

Esse fenômeno é o oposto da fragmentação interna, no qual o espaço disponível é desperdiçado dentro dos blocos alocados, como apresenta a imagem abaixo. Sistemas Operacionais possuem uma estrutura de dados que armazena informações sobre áreas ou blocos livres (geralmente uma lista ou tabela). Uma lista encadeada elimina o problema da **fragmentação externa**. Por que?

Porque mantém os arquivos, cada um, como uma lista encadeada de blocos de disco. Dessa forma, uma parte de cada bloco é usada como ponteiro para o próximo bloco e o restante do bloco é usado para dados. Uma vantagem desse tipo de alocação é que o tamanho do arquivo não precisa ser conhecido antes de sua criação, já que cada bloco terá um ponteiro para o próximo bloco.

(CESPE/CEBRASPE – 2019 – MPC-PA) Assinale a opção que apresenta a denominação da estrutura de dados constituída por um conjunto de elementos individualizados, em que cada um dos elementos — com exceção dos elementos inicial e final — referencia sempre outros dois, um que o antecede e outro que o sucede.

- a) lista circular
- b) grafo
- c) lista duplamente encadeada
- d) árvore
- e) pilha

Comentários:

Vamos analisar item a item:

- a) lista circular

Uma lista em que cada elemento referencia o outro, porém com uma diferença: o último elemento referencia o primeiro também. A questão deixa claro que quer uma estrutura de dados em que os elementos inicial e final não estejam ligados. Falso.



b) Grafo

Estrutura de dados com um conjunto de vértices ligados entre si por arestas. Falso.

c) lista duplamente encadeada

Uma lista em que cada elemento referencia o outro em ambas as direções: um elemento referencia o anterior e o próximo, exceto os elementos inicial e final. É isso aí. Verdadeiro.

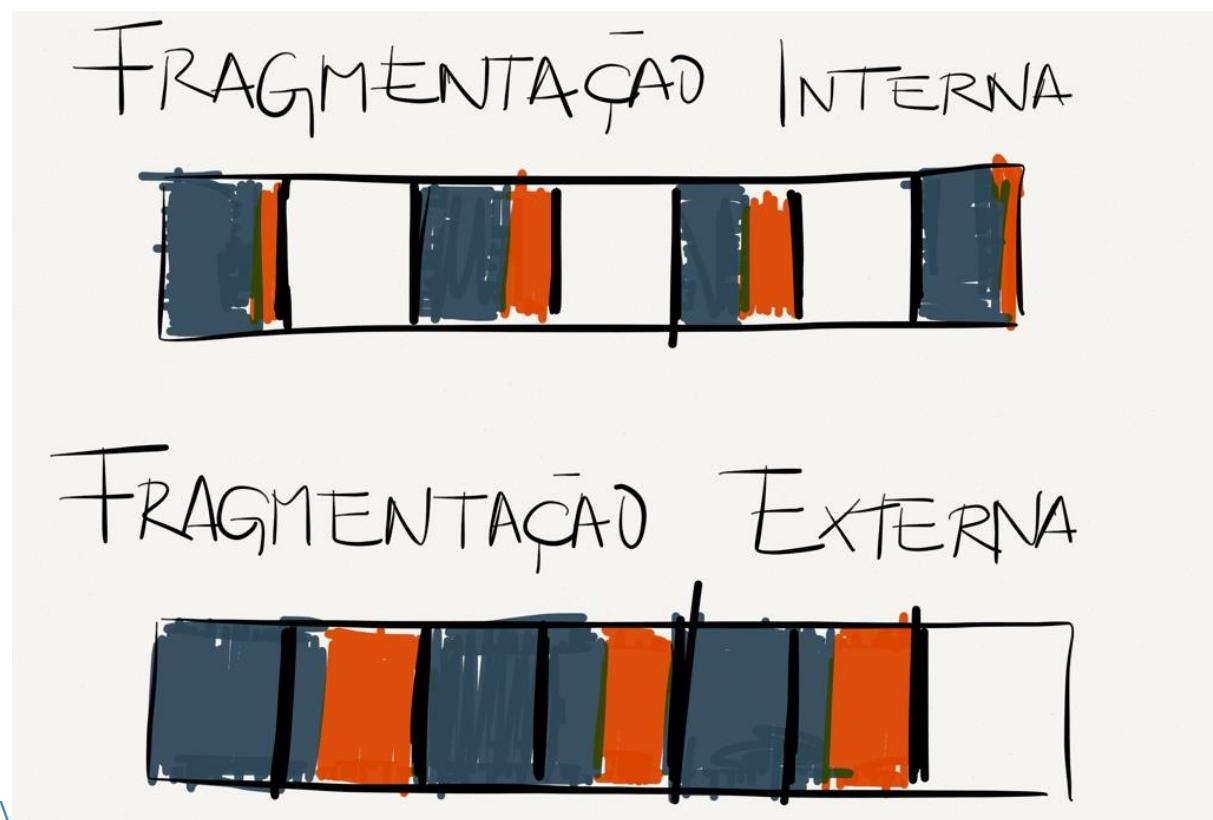
d) Árvore

Estrutura de dados em que um nó possui 1 ou mais filhos, mas cada filho possui somente um pai. Falso.

e) pilha

Estrutura de dados usando a regra LIFO, last-in first-out, ou seja, o último a entrar é o primeiro a sair. Falso.

(Gabarito: Letra C).



Galera... e o acesso a uma lista? A Lista é uma estrutura de **acesso sequencial**, i.e., é preciso percorrer nó por nó para acessar um **dado específico**. Logo, é proporcional ao número de elementos – Acesso O(n). E os Vetores? Eles são uma estrutura de acesso direto, i.e., pode-se acessar um elemento diretamente. Portanto, não precisa percorrer elemento por elemento (Acesso O(1))³.

(CESPE/CEBRASPE – 2017 – TRT-7) Considere uma estrutura de dados em que cada elemento armazenado apresenta ligações de apontamento com seu sucessor e com o seu predecessor, o que possibilita que ela seja percorrida em qualquer sentido. Trata-se de

³ No Acesso Sequencial: quanto mais ao fim, maior o tempo para acessar; no Acesso Direto: todos os elementos são acessados no mesmo tempo.



- a) uma fila.
- b) um grafo.
- c) uma lista duplamente encadeada.
- d) uma pilha.

Comentários:

Vamos analisar item a item:

- a) uma fila.

Estrutura de dados que segue a lógica FIFO (first-in, first-out), ou “primeiro a entrar é o primeiro a sair”. Os elementos são inseridos no fim da fila e removidos no início. Falso.

- b) um grafo.

Grafo é uma estrutura de dados com um conjunto de vértices e arestas que os ligam. Falso.

- c) uma lista duplamente encadeada.

É uma estrutura de dados que armazena uma sequência de elementos que possuem ligações de apontamento, ou ponteiros, entre si, em ambas as direções. Isso permite percorrer a lista em qualquer sentido. Verdadeiro.

- d) uma pilha.

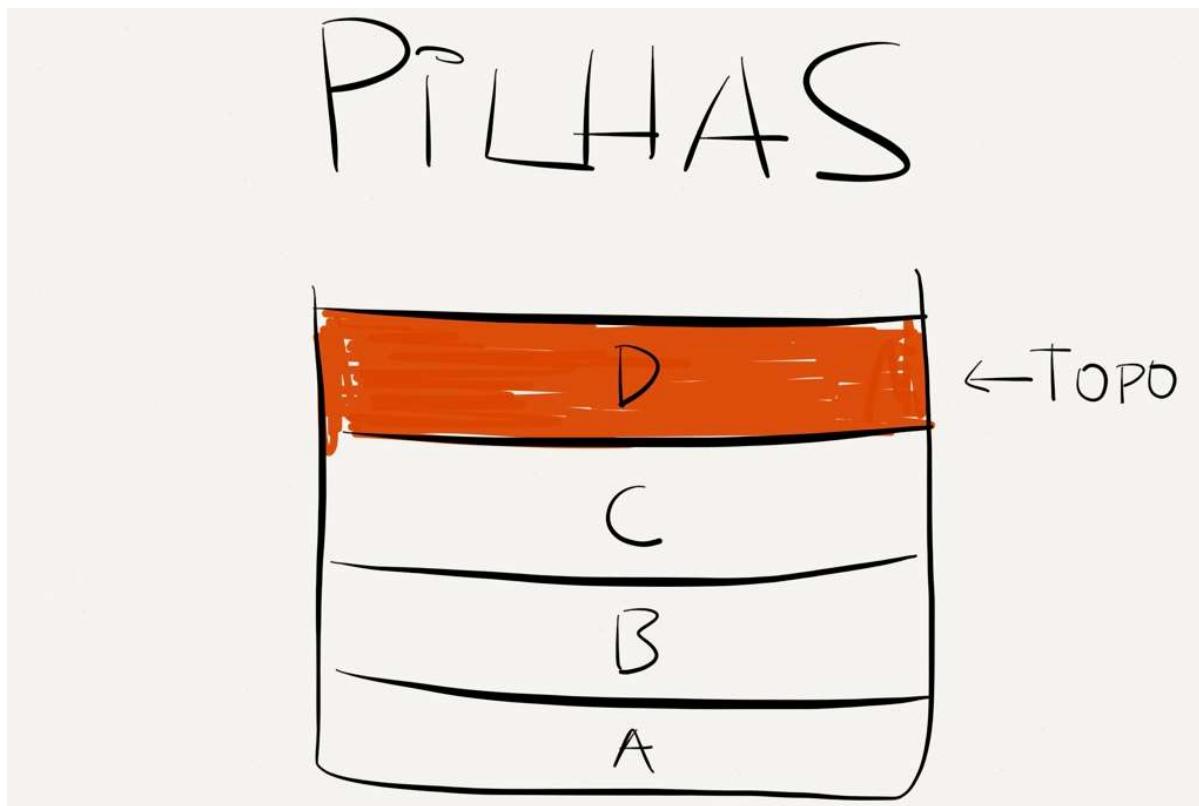
Estrutura de dados que segue a lógica LIFO, ou last-in first-out, que significa “o último a entrar é o primeiro a sair”. Falso. (Gabarito: Letra C).



PILHAS

A Pilha é um **conjunto ordenado** de itens no qual novos itens podem ser inseridos e eliminados em uma extremidade chamada **topo**. Novos itens podem ser colocados no topo da pilha (tornando-se o novo primeiro elemento) ou os itens que estiverem no topo da pilha poderão ser removidos (tornando-se o elemento mais abaixo o novo primeiro elemento).

Também conhecida como **Lista LIFO** (Last In First Out), basta lembrar de uma pilha de pratos esperando para serem lavados, i.e., o último a entrar é o primeiro a sair. A ordem em que os pratos são retirados da pilha é o oposto da ordem em que eles são colocados sobre a pilha e, como consequência, apenas o prato do topo da pilha está acessível.

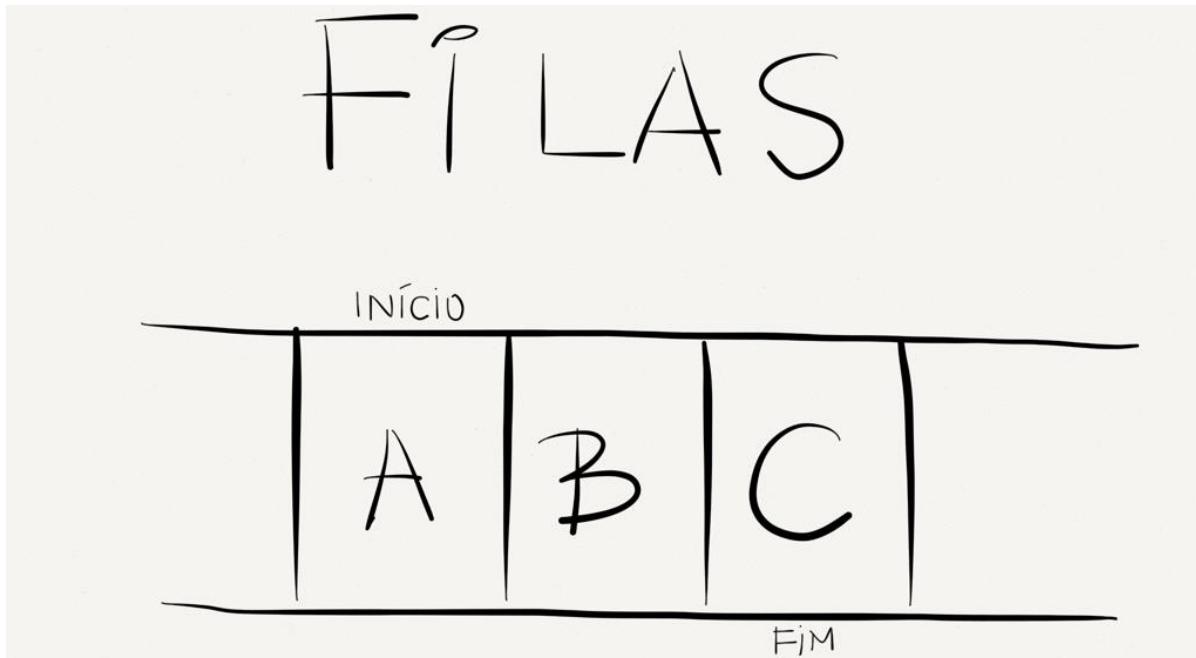


As Pilhas oferecem três operações básicas: push, que insere um novo elemento no topo da pilha; pop, que remove um elemento do topo da pilha; e top (também conhecida como check), que acessa e consulta o elemento do topo da pilha. Pilhas podem ser implementadas por meio de **Vetores** (Pilha Sequencial - Alocação Estática de Memória) ou **Listas** (Pilha Encadeada - Alocação Dinâmica de Memória).



FILAS

Uma fila é um conjunto **ordenado** de itens a partir do qual podem-se **eliminar** itens numa extremidade (chamada **início** da fila) e no qual podem-se **inserir** itens na outra extremidade (chamada **final** da fila). Também conhecida como Lista FIFO (First In First Out), basta lembrar de uma fila de pessoas esperando para serem atendidas em um banco, i.e., o primeiro a entrar é o primeiro a sair.



Quando um elemento é colocado na fila, ele ocupa seu lugar no fim da fila, como um aluno recém-chegado que ocupa o final da fileira. O elemento retirado da fila é sempre aquele que está no **início** da fila, como o aluno que se encontra no começo da fileira e que esperou mais tempo. As operações básicas são **Enqueue** (Enfileirar) e **Dequeue** (Desenfileirar). As Filas possuem **início** (ou cabeça) e **fim** (ou cauda).

É bom salientar outro conceito importante: Deque (Double Ended Queue)! É também conhecida como Filas Duplamente Encadeadas e permite a **eliminação e inserção de** itens em ambas as extremidades. Ademais, elas permitem algum tipo de priorização, visto que é possível inserir elementos de ambos os lados. Assim sendo, é comum em sistemas distribuídos!

Sistemas distribuídos sempre necessitam que algum tipo de processamento seja mais rápido, por ser mais prioritário naquele momento, deixando outros tipos mais lentos ou em fila de espera, por não requerem tanta pressa. Ele pode ser entendido como uma **extensão da estrutura** de dados Fila. Agora uma pergunta: Qual a diferença entre uma lista duplamente encadeada e um deque?

Pessoal, um deque gerencia elementos como um vetor, fornece **acesso aleatório** e tem quase a mesma interface que um **vetor**. Ele se diferencia de uma lista duplamente encadeada, entre outras coisas, por essa não fornecer acesso aleatório aos elementos, i.e., para acessar o quinto elemento, você deve navegar pelos quatro primeiros elementos – logo a lista é mais lenta nesse sentido. Bacana?



(CESPE/CEBRASPE – 2017 – TRE-TO) A estrutura de dados que consiste no armazenamento de cada elemento em um endereço calculado a partir da aplicação de uma função sobre a chave de busca denomina-se

- a) lista.
- b) tabela hashing.
- c) deque.
- d) fila.
- e) árvore binária balanceada.

Comentários:

Vamos analisar item a item:

- a) lista.

Estrutura de dados que armazena uma sequência de elementos, cada um apontando para o seu sucessor na lista. Falso.

- b) tabela hashing.

Estrutura de dados que permite armazenar e recuperar os elementos diretamente ou quase diretamente, por meio de uma chave de busca. Cada elemento é armazenado em um endereço que é calculado por uma função de hash, que é aplicada sobre a chave de busca. A função de hash gera um índice, que é um endereço do elemento na tabela. Dessa forma, o acesso é direto, ou próximo do direto, sem precisar percorrer toda a tabela. Verdadeiro.

- c) deque.

Estrutura de dados para inserir e remover tanto no início quanto no fim da estrutura. Falso.

- d) fila.

Estrutura de dados que segue a lógica FIFO (first-in, first-out), ou “primeiro a entrar é o primeiro a sair”. Os elementos são inseridos no fim da fila e removidos no início. Falso.

- e) árvore binária balanceada.

É uma estrutura de dados em formato de árvore. Cada nó possui, no máximo, dois filhos, e os níveis da árvore estão平衡ados, o que garante eficiência nas operações de inserção, remoção e busca. (Gabarito: Letra B)

(CESPE/CEBRASPE – 2017 – TRF-1) Acerca de estrutura de dados, julgue o próximo item.

A fila é uma lista de elementos em que os itens são sempre inseridos em uma das extremidades e excluídos da outra.

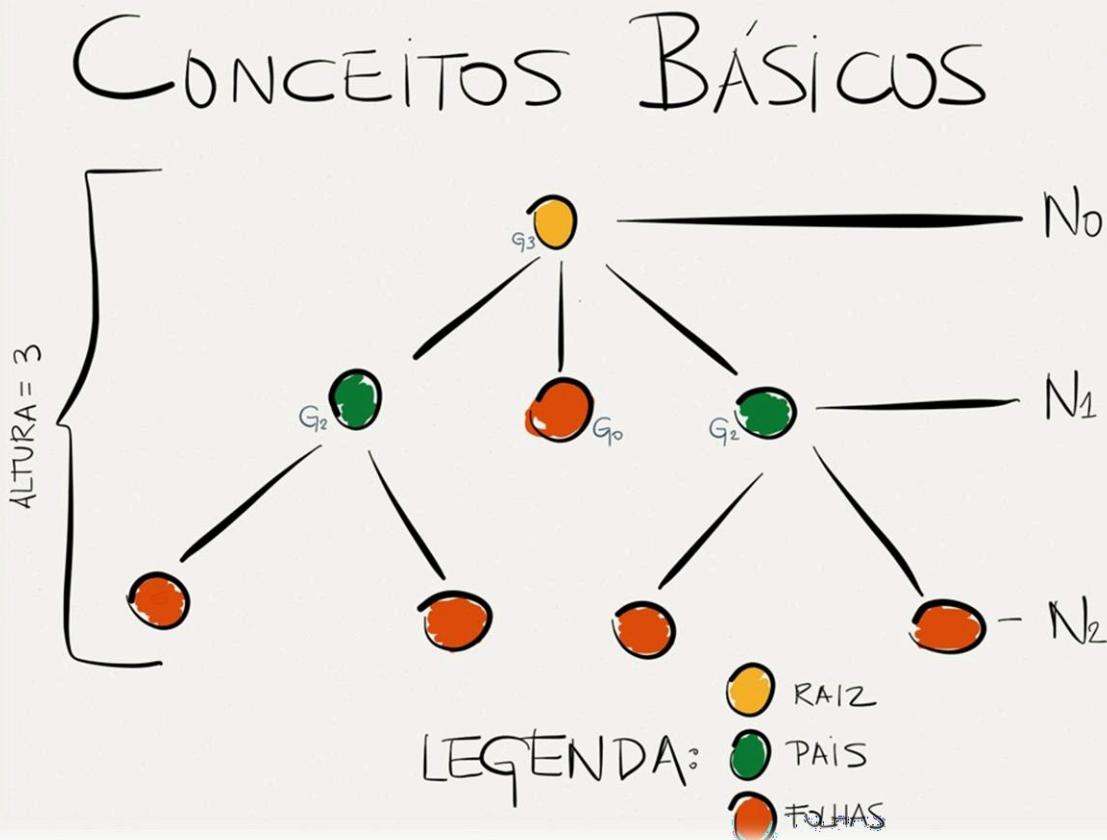
Comentários:

Fila é uma estrutura de dados que segue a lógica FIFO (first-in first-out) armazena elementos de forma sequencial, permitindo a inserção de novos elementos no final da estrutura e a remoção de elementos do início. Então, está certo. (Gabarito: Certo)



ESTRUTURAS DE DADOS: ÁRVORE

Uma árvore é uma estrutura de **dados hierárquica** (não-linear) composta por um **conjunto finito** de elementos com um único elemento raiz, com zero ou mais sub-árvore ligadas a esse elemento raiz. Como mostra a imagem abaixo, há uma única raiz, em amarelo. Há também nós folhas, em vermelho e seus pais, em verde. Observem ainda os conceitos de Altura, Grau e Nível de uma árvore.



O Grau informa a quantidade de filhos de um determinado nó! A Raiz tem Nível 0 (excepcionalmente, alguns autores consideram que tem Nível 1) e o nível de qualquer outro nó na árvore é um nível a mais que o nível de seu pai. Por fim, a Altura é a distância entre a raiz e seu descendente mais afastado. Dessas informações, podemos concluir que toda folha tem **Grau 0**.

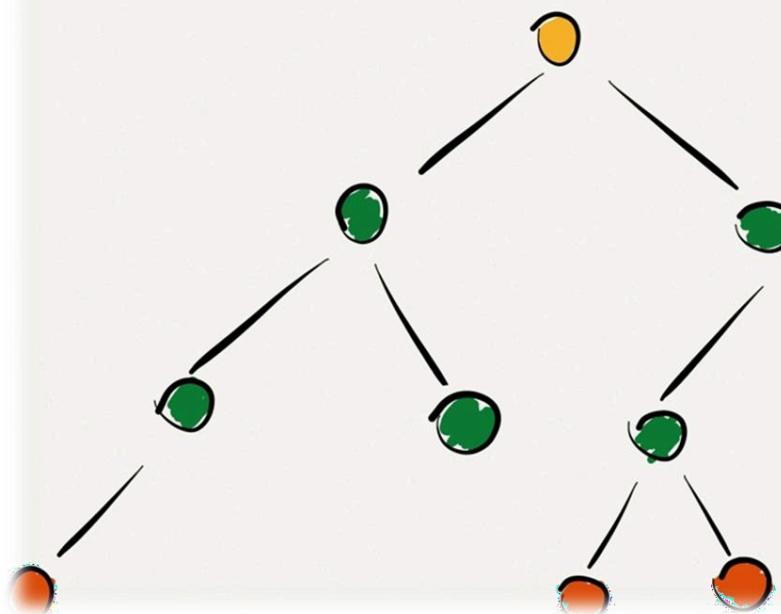
Existe um tipo particular de árvore chamado: **Árvore Binária!** O que é isso? É uma estrutura de dados hierárquica em que todos os nós têm grau 0, 1 ou 2. Já uma Árvore Estritamente Binária é aquela em que todos os nós têm grau **0 ou 2**. E uma Árvore Binária Completa é aquela em que todas as folhas estão no mesmo nível, como mostram as imagens abaixo.

(CESPE - 2012 - Banco da Amazônia - Técnico Científico - Administração de Dados) O tipo de dados árvore representa organizações hierárquicas entre dados.

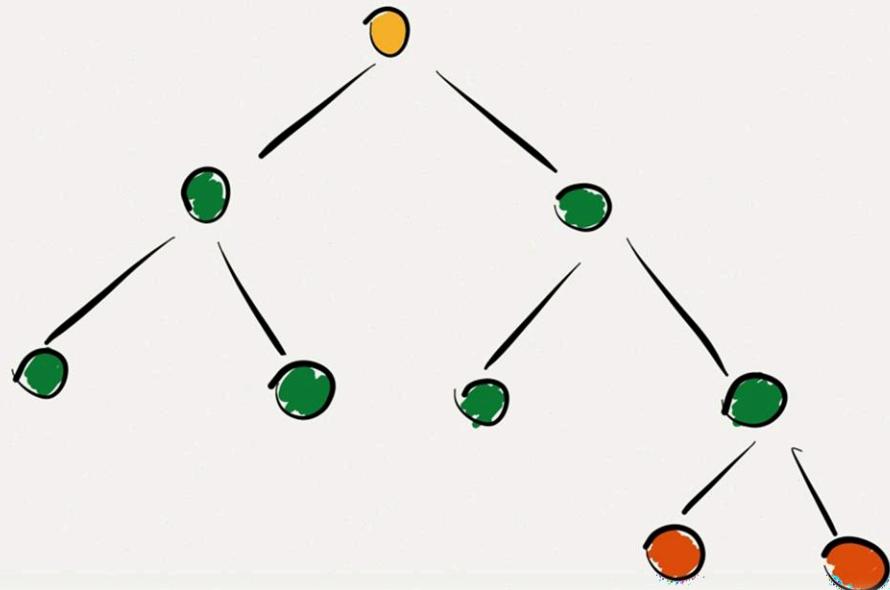


Comentários: Perfeito, observem que alguns autores tratam Tipos de Dados como sinônimo de Estruturas de Dados. **Gabarito: C**

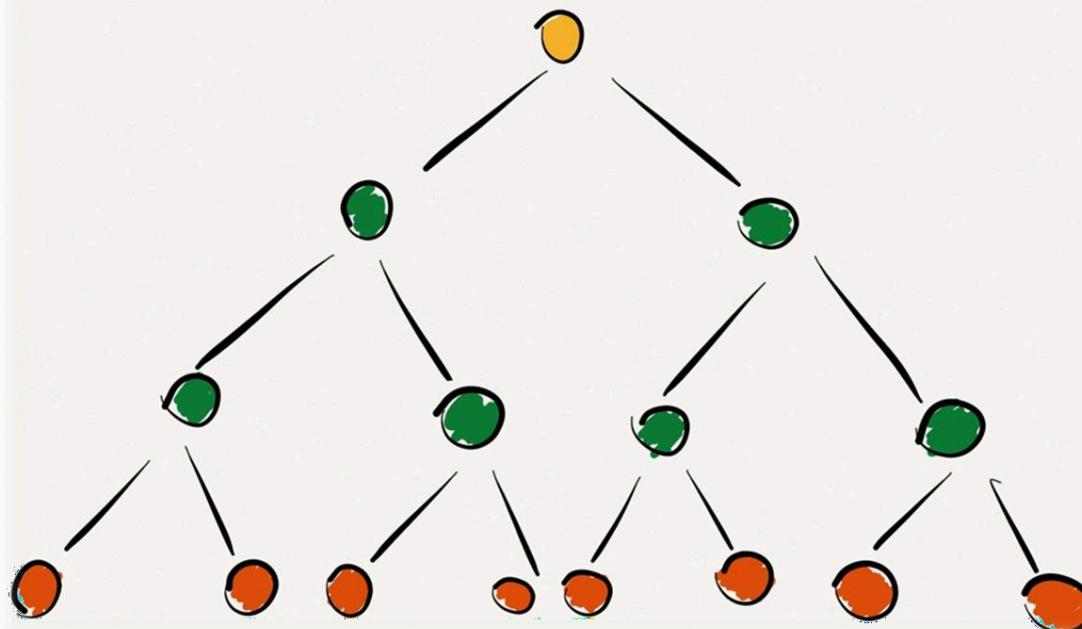
ÁRVORE BINÁRIA



ÁRVORE ESTRITAMENTE BINÁRIA



ÁRVORE BINÁRIA COMPLETA



Uma Árvore Binária Completa com x folhas conterá sempre $(2x - 1)$ nós. Observem a imagem acima e façam as contas: $2^3 \cdot 8 - 1 = 15$ nós! Uma árvore binária completa de **altura h e nível n** contém **$(2^h - 1)$** ou **$(2^{n+1} - 1)$** nós e usa-se **(2^n)** , para calcular a quantidade de nós em determinado nível. Na imagem acima, há uma árvore de $h = 4$ e $n = 3$; logo, existem $2^{3+1} - 1 = 15$ nós no total; e no Nível 3, existe $2^3 = 8$ nós.

Vamos falar um pouco agora sobre Árvore de Busca Binária! Trata-se de uma estrutura de dados vinculada, baseada em nós, onde cada nó contém uma chave e duas subárvores à esquerda e a direita. Para todos nós, a chave da subárvore esquerda deve ser menor que a chave desse nó, e a chave da subárvore direita deve ser maior. Beleza?

Todas estas subárvores devem qualificar-se como árvores binárias de busca. O pior caso de tempo de complexidade para a pesquisa em uma árvore binária de busca é a altura da árvore, isso pode ser tão pequeno como $O(\log n)$ para uma árvore com n elementos. Galera, abaixo nós vamos ver como árvores podem ser representadas e o conceito de árvore binária de busca ficará mais clara!

Como representamos árvores? Podemos representar uma árvore como um conjunto de **parênteses aninhados**. Nessa notação, $(P(F_1)(F_2))$ significa que P , F_1 , F_2 são nós e que F_1 , F_2 , são filhos do pai P . Ao transcrever isso para o desenho hierárquico de uma árvore, lemos da esquerda para a direita. Agora suponhamos que F_1 tem dois filhos N_1 e N_2 . Logo, reescrevemos a subárvore de F_1 como $(F_1(N_1)(N_2))$.

(CETAP - 2010 - AL-RR - Analista de Sistemas - A) Uma árvore binária é aquela que tem como conteúdo somente valores binários.

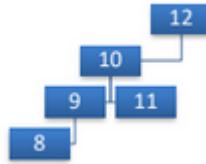
Comentários: Não! Uma árvore binária é aquela que tem, no máximo, grau 2! **Gabarito: E**

Podemos, então, substituir F_1 por $(F_1(N_1)(N_2))$. Ao final, ficará $(P(F_1(N_1)(N_2))(F_2))$. E assim por diante. Temos então que o primeiro elemento é a raiz e sempre que tivermos um parêntese, teremos uma nova **subárvore**. Vamos exemplificar: $(12(10(9(8))(11))(14(13)(15)))$. Como ficaria, professor? Sabemos que 12 será a raiz dessa árvore e, a partir daí, criamos a árvore da esquerda para direita.

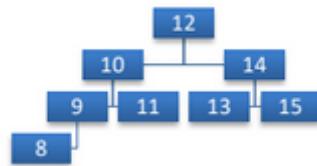
Observem o parêntese após o 12! Notem que ele só é fechado após o 11: $(12(10(9(8))(11))$). Isso significa que tudo que está em vermelho é subárvore da esquerda da raiz 12 – e o restante $(14(13)(15))$ é subárvore da direita da raiz 12. Observem o parêntese após o 10! Notem que ele só é fechado após o 8: $10(9(8))$. Isso significa que tudo que está em amarelo é subárvore da esquerda da raiz 10.

E o restante (11) é subárvore da direita da raiz 10. Observem o parêntese após o 9! Notem que ele só é fechado após o 8: $9(8)$. Isso significa que tudo que está em verde é subárvore da esquerda da raiz 9 – e o restante... que restante, professor? Pois é, não tem restante! Logo, 9 não tem subárvore da direita. Vejam abaixo como ficou e vamos analisar o outro lado.

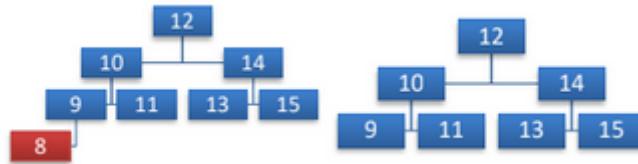




A subárvore da direita da raiz 12 tem raiz 14 e tem dois filhos: na esquerda, 13 e na direita 15. Fim! Galera, eu sei que parece complicado, mas leiam e pratiquem umas três vezes – de preferência no papel - que vocês internalizam tranquilamente esse conteúdo. É chatinho, mas não é difícil! Vejam abaixo como ficou o resultado final e vamos seguir em frente...

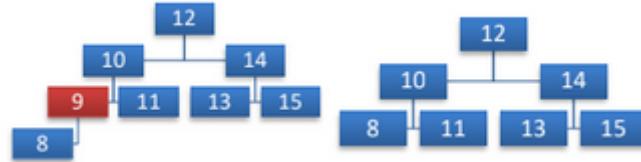


Bem, pessoal! Dito isso, vamos analisar agora como ficaria para remover um nó desta árvore. Existem três possibilidades para realizar essa operação: (1) remover um nó que não tem filhos; (2) remover um nó que tem apenas um filho; (3) e remover um nó que tenha dois filhos. O primeiro caso é muito simples: basta retirar o nó desejado e ponto final. Vejamos como fica:



No segundo caso, basta retirar o nó da árvore e conectar seu único filho (e sua subárvore, se houver) diretamente ao pai do nó removido. Vejamos:





Já o último caso, nós podemos utilizar duas estratégias. Você pode escolher qual deseja utilizar em uma situação específica. Vejamos:

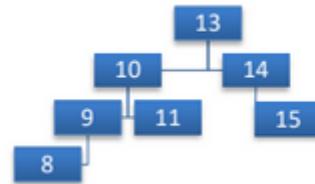
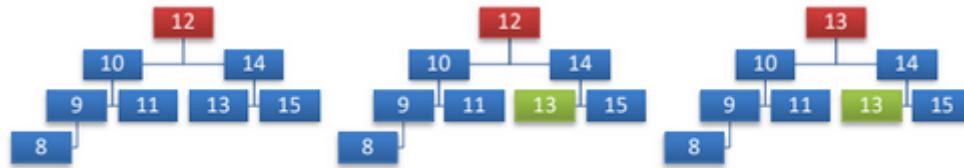
ESTRATÉGIA 1

PASSO 1: IDENTIFIQUE O ELEMENTO QUE VOCÊ DESEJA RETIRAR DA ÁRVORE (EM VERMELHO)

PASSO 2: IDENTIFIQUE O MENOR ELEMENTO DE TODA SUBÁRVORE À DIREITA DO NÓ IDENTIFICADO NO PASSO 1 (EM VERDE)

PASSO 3: COPIE O VALOR DO NÓ IDENTIFICADO NO PASSO 2 PARA O NÓ IDENTIFICADO NO PASSO 1

PASSO 4: REMOVA O ELEMENTO IDENTIFICADO NO PASSO 2.



ESTRATÉGIA 2

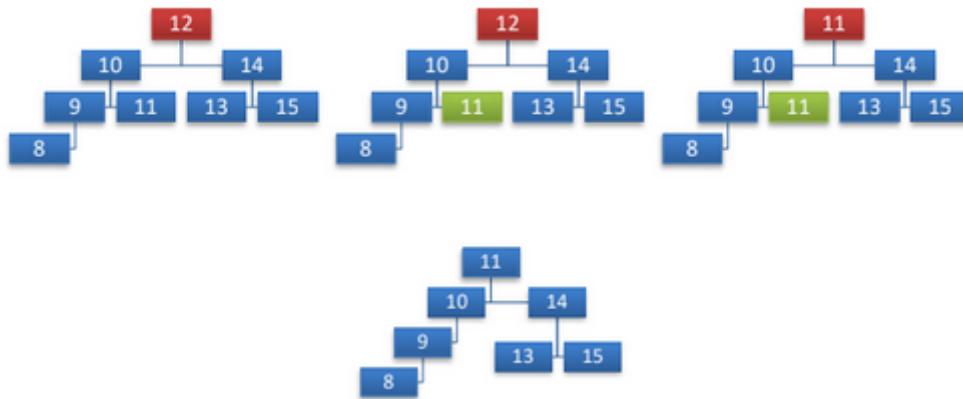
PASSO 1: IDENTIFIQUE O ELEMENTO QUE VOCÊ DESEJA RETIRAR DA ÁRVORE (EM VERMELHO)

PASSO 2: IDENTIFIQUE O MAIOR ELEMENTO DE TODA SUBÁRVORE À ESQUERDA DO NÓ IDENTIFICADO NO PASSO 1 (EM VERDE)



PASSO 3: COPIE O VALOR DO NÓ IDENTIFICADO NO PASSO 2 PARA O NÓ IDENTIFICADO NO PASSO 1

PASSO 4: REMOVA O ELEMENTO IDENTIFICADO NO PASSO 2.



Por fim, como seria a representação por parênteses aninhados? Na primeira estratégia, temos: $(13(10(9(8))(11))(14(15)))$; na segunda, temos $(11(10(9(8)))(14(13)(15)))$.

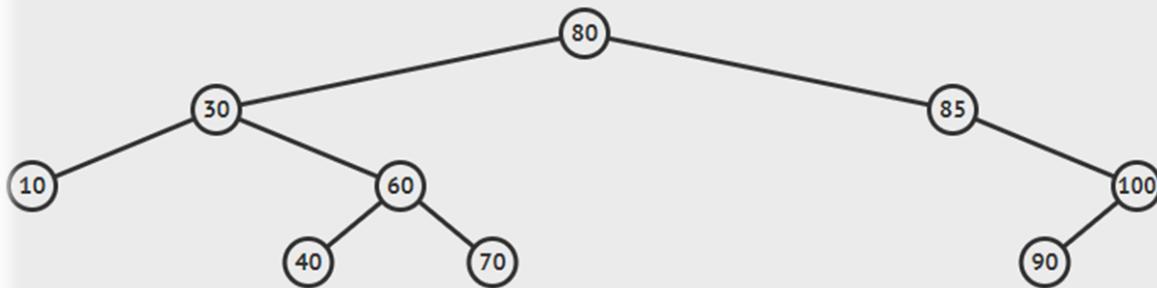
Galera, em uma Árvore de Busca Binária, podemos fazer três percursos: **pré-ordem, in-ordem e pós-ordem** (esses prefixos são em relação a raiz). É interessante notar que, quando se faz um percurso em ordem em uma árvore binária de busca, os valores dos nós aparecem em ordem crescente. A operação "Percorre" tem como objetivo percorrer a árvore numa dada ordem, enumerando os seus nós.

Quando um nó é enumerado, diz-se que ele foi "**visitado**". Vamos ver agora esses três percursos:

- **Pré-Ordem (ou Profundidade):** visita a raiz; percorre a subárvore esquerda em pré-ordem; percorre a subárvore direita em pré-ordem.
- **In-Ordem (ou Simétrica):** percorre a subárvore esquerda em in-ordem; visita a raiz; percorre a subárvore direita em in-ordem.
- **Pós-Ordem:** percorre a subárvore esquerda em pós-ordem; percorre a subárvore direita em pós-ordem; visita a raiz.

Vamos ver um exemplo:





Como ler essa árvore em Pré-Ordem? Vamos tentar...

//Percorrendo a Árvore em Pré-Ordem:

1. Visite a Raiz: {80};
2. Percorra a subárvore esquerda em pré-ordem:
 1. Visite a Raiz: {30};
 2. Percorra a subárvore esquerda em pré-ordem:
 1. Visite a Raiz: {10};
 2. Percorra a subárvore esquerda em pré-ordem: {Vazio}
 3. Percorra a subárvore direita em pré-ordem: {Vazio}
 3. Percorra a subárvore direita em pré-ordem:
 1. Visite a Raiz: {60};
 2. Percorra a subárvore esquerda em pré-ordem:
 1. Visite a Raiz: {40}
 2. Percorra a subárvore esquerda em pré-ordem: {Vazio}
 3. Percorra a subárvore direita em pré-ordem: {Vazio}
 3. Percorra a subárvore direita em pré-ordem:
 1. Visite a Raiz: {70}



Percorra a subárvore esquerda em pré-ordem: {Vazio}

Percorra a subárvore direita em pré-ordem: {Vazio}

Percorra a subárvore direita em pré-ordem:

Visite a Raiz: {85};

Percorra a subárvore esquerda em pré-ordem: {Vazio}

Percorra a subárvore direita em pré-ordem: {Vazio}

Visite a Raiz: {100};

Percorra a subárvore esquerda em pré-ordem:

Visite a Raiz: {90};

Percorra a subárvore esquerda em pré-ordem: {Vazio}

Percorra a subárvore direita

RESULTADO: 80, 30, 10, 60, 40, 70, 85, 100
90

Percorra a subárvore direita em pré-ordem:
{Vazio}

//Percorrendo a Árvore em In-Ordem:

Percorra a subárvore esquerda em in-ordem:

Percorra a subárvore esquerda em in-ordem:

Percorra a subárvore esquerda em in-ordem: {Vazio}

Visite a Raiz: {10}

Percorra a subárvore direita em in-ordem: {Vazio}

Visite a Raiz: {30}

Percorra a subárvore direita em in-ordem:



1. Percorra a subárvore esquerda em in-ordem:

1. Percorra a subárvore esquerda em in-ordem: {Vazio}

2. Visite a Raiz: {40}

3. Percorra a subárvore direita em in-ordem: {Vazio}

2. Visite a Raiz: {60}

3. Percorra a subárvore direita em in-ordem:

1. Percorra a subárvore esquerda em in-ordem: {Vazio}

2. Visite a Raiz: {70}

3. Percorra a subárvore direita em in-ordem: {Vazio}

2. Visite a Raiz: {80}

3. Percorra a subárvore direita em in-ordem:

1. Percorra a subárvore esquerda em in-ordem: {Vazio}

2. Visite a Raiz: {85}

3. Percorra a subárvore direita em in-ordem:

1. Percorra a subárvore esquerda em in-ordem:

1. Percorra a subárvore esquerda em in-ordem: {Vazio}

2. Visite a Raiz: {90}

3. Percorra a subárvore direita em in-ordem: {Vazio}

2. Visite a Raiz: {100}

3. Percorra a subárvore direita em in-ordem: {Vazio}

RESULTADO: 10, 30, 40, 60, 70, 80, 85, 90, 100.

//Percorrendo a Árvore em Pós-Ordem:



1. Percorra a subárvore esquerda em pós-ordem:

1. Percorra a subárvore esquerda em pós-ordem:

1. Percorra a subárvore esquerda em pós-ordem: {Vazio}
2. Percorra a subárvore direita em pós-ordem: {Vazio}
3. Visite a Raiz: {10}

2. Percorra a subárvore direita em pós-ordem:

1. Percorra a subárvore esquerda em pós-ordem: {Vazio}

1. Percorra a subárvore esquerda em pós-ordem: {Vazio}
2. Percorra a subárvore direita em pós-ordem: {Vazio}
3. Visite a Raiz: {40}

2. Percorra a subárvore direita em pós-ordem:

1. Percorra a subárvore esquerda em pós-ordem: {Vazio}
2. Percorra a subárvore direita em pós-ordem: {Vazio}
3. Visite a Raiz: {70}

3. Visite a Raiz: {60}

3. Visite a Raiz: {30}

2. Percorra a subárvore direita em pós-ordem:

1. Percorra a subárvore esquerda em pós-ordem: {Vazio}

2. Percorra a subárvore direita em pós-ordem:

1. Percorra a subárvore esquerda em pós-ordem:

1. Percorra a subárvore esquerda em pós-ordem: {Vazio}
2. Percorra a subárvore direita em pós-ordem: {Vazio}
3. Visite a Raiz: {90}



Percorra a subárvore direita em
pós-ordem: {Vazio}

Visite a Raiz: {100}

3. Visite a Raiz: {85}

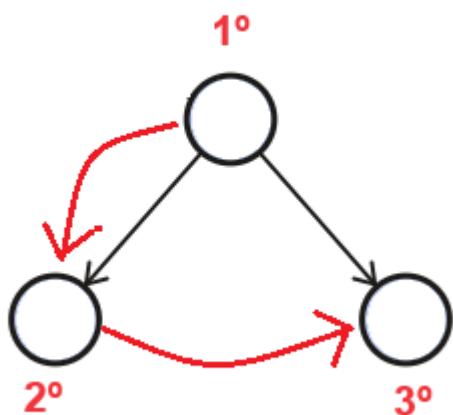
3. Visite a Raiz: {80}

RESULTADO: 10, 40, 70, 60, 30, 90, 100, 85,
80.

Portanto, há três modos para percorrer uma árvore binária:

Pré-ordem: No percurso pré-ordem, ou “pre order”, os nós são visitados na seguinte ordem: primeiro o nó pai, em seguida o filho esquerdo e, por último, o filho direito. Essa técnica é útil para criar uma cópia da árvore ou para realizar operações de pré-processamento antes de visitar os nós filhos.

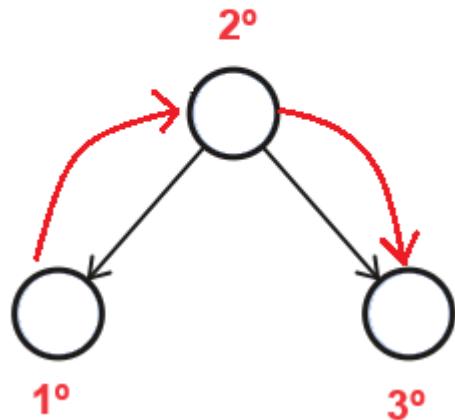
Pré-ordem: Pai-Esquerda-Direita



Em ordem (Simétrico): No percurso em ordem, também conhecido como “in order”, a árvore é percorrida de forma que os nós sejam visitados na seguinte ordem: primeiro o filho esquerdo, depois o nó pai e, por fim, o filho direito. Essa técnica é comumente utilizada para obter os elementos de uma árvore binária de busca em ordem crescente.

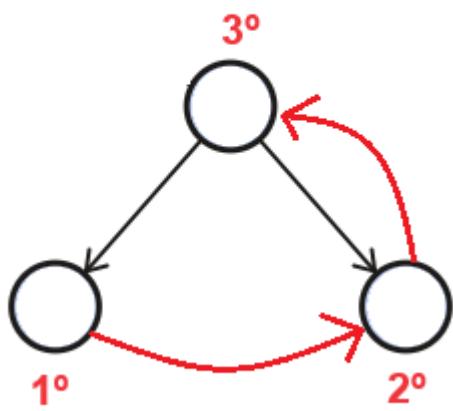


Em ordem:
Esquerda-Pai-Direita



Pós-ordem: O percurso pós-ordem, ou “post order”, envolve a visita aos nós na seguinte ordem: primeiro o filho esquerdo, depois o filho direito e, por último, o nó pai. Essa técnica é frequentemente empregada em cálculos que requerem informações dos nós filhos antes de processar o nó pai, como a avaliação de expressões aritméticas.

Pós-ordem:
Esquerda-Direita-Pai



A dica pra entender rapidamente esses métodos de percursos em árvores binárias é seguir essas duas regras:

- O nó filho da esquerda é sempre visitado antes que o da direita;
- O nome do método se refere ao momento em que o nó pai é visitado, ou seja:
 - Pré-ordem: o pai é visitado antes dos filhos, ou seja, Pai-Esquerda-Direita;
 - Em ordem: o pai é visitado entre os filhos, ou seja, Esquerda-Pai-Direita;
 - Pós-ordem: o pai é visitado após os filhos, ou seja, Esquerda-Direita-Pai;



Pré-ordem: **Pai** - Esquerda - Direita

Em ordem: Esquerda - **Pai** - Direita

Pós-ordem: Esquerda - Direita - **Pai**

Obs: O nó **esquerdo** é visitado **antes** do nó **direito** em *qualquer método de percurso em árvore binária.*

Algumas bancas possuem entendimentos específicos de percorrer a árvore, por exemplo, a CESGRANRIO, que no caso do percurso In-ordem inicia visitando 'DIREITA - PAI - ESQUERDA', em vez de 'ESQUERDA - PAI - DIREITA'.

Agora vamos falar um pouquinho sobre três tipos especiais e árvores: **Árvore B, Árvore B+ e Árvore AVL**. E aí, eu preciso bastante da atenção de vocês agora! Eu coloco esse assunto porque os editais pedem apenas Árvore e não especificam a profundidade do assunto. Com exceção da CESGRANRIO, é raro outras bancas cobrarem esse assunto na profundidade que veremos agora.

É um assunto mais difícil e que cai pouco em prova, portanto só recomendo seguir caso queiram realmente cercar todas as possibilidades. Galera, época de faculdade, segundo semestre, disciplina de Estrutura de Dados! O trabalho final da disciplina era construir um compactador! Isso mesmo! Uma espécie de WinZip, WinRAR, etc. E a estrutura usada para compactar arquivos de índices era uma Árvore B.

Uma árvore B é uma maneira de armazenar grandes quantidades de dados de tal forma que você pode procurá-los e recuperá-los muito rapidamente. As árvores B são a base da maioria dos bancos de dados **modernos**. Como eles funcionam é um bocado complicado, mas eu vou contar uma historinha que talvez os ajude a entender melhor! Vamo lá...

Imagine que você está procurando um par de novos fones de ouvido. Você tem algumas abordagens. Certo? Você poderia ir a todas as lojas do mundo até encontrar o produto que você procura. Como você pode imaginar, esta seria uma maneira horrível de fazer compras. Em vez disso, você poderia ir em uma FNAC, porque você sabe que eles vendem eletrônicos.

Uma vez que chegou à FNAC, você pode observar cada uma das descrições de corredor para ver onde os fones de ouvido são armazenados. Depois de encontrar o corredor correto, você pode escolher os fones de ouvido que você deseja. Ponto final! Observe como o objetivo desse processo é restringir o foco de uma pesquisa cada vez mais...

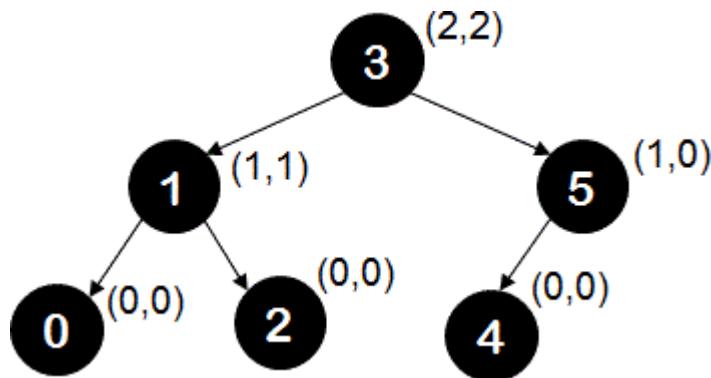
É assim que funcionam as Árvores B! Ao organizar os dados de forma específica, podemos aproveitá-las para que não desperdicemos nosso tempo buscando dados que tenham chance zero de armazenar os



E qual a diferença de uma Árvore B para uma Árvore B+? Galera, a principal diferença é que, em uma Árvore B, as chaves e os dados podem ser armazenados tanto nos nós internos da árvore quanto nas folhas da árvore, enquanto que em uma Árvore B+ as chaves podem ser armazenadas em qualquer nó, mas os dados só podem ser armazenados nas **folhas**.

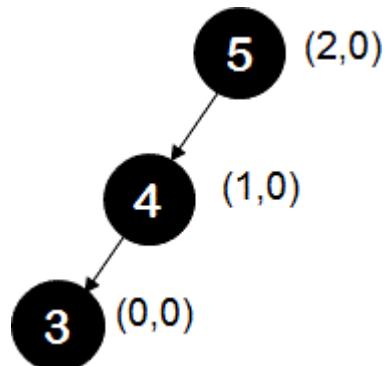


Por fim, vamos falar um pouco sobre Árvores AVL! Uma Árvore AVL (Adelson-Vesky e Landis) é uma Árvore Binária de Busca em que, para qualquer nó, a altura das subárvores da esquerda e da direita não podem ter uma diferença maior do que 1, portanto uma Árvore AVL é uma Árvore Binária de Busca autobalanceável. Calma que nós vamos ver isso em mais detalhes...



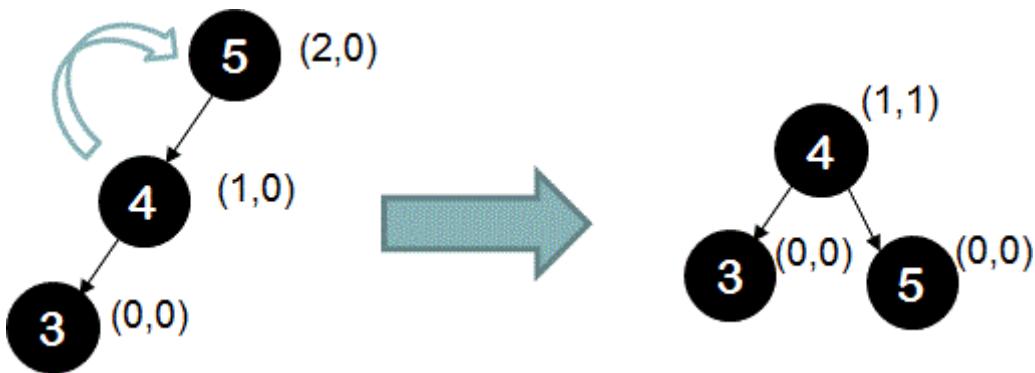
Observem o Nô 3: a altura da subárvore da esquerda é 2 e da subárvore da direita também é 2. Vejamos agora o Nô 1: a altura da subárvore da esquerda é 1 e da subárvore da direita também é 1. E o Nô 5: a altura da subárvore da esquerda é 1 e da subárvore da direita é 0 (visto que ela não existe). Vocês podem ver todos os nós e não encontrarão subárvore da esquerda e direita com diferença **maior que 1**.

Uma Árvore AVL mantém seu equilíbrio de altura executando operações de rotação se algum dos nós violar a propriedade da diferença maior que 1. Exemplo 1: para a seguinte árvore, a propriedade da árvore AVL é violada no Nô 5, porque a subárvore esquerda possui altura 2, mas a subárvore da direita tem altura 0, então eles diferem em 2. Entenderam?

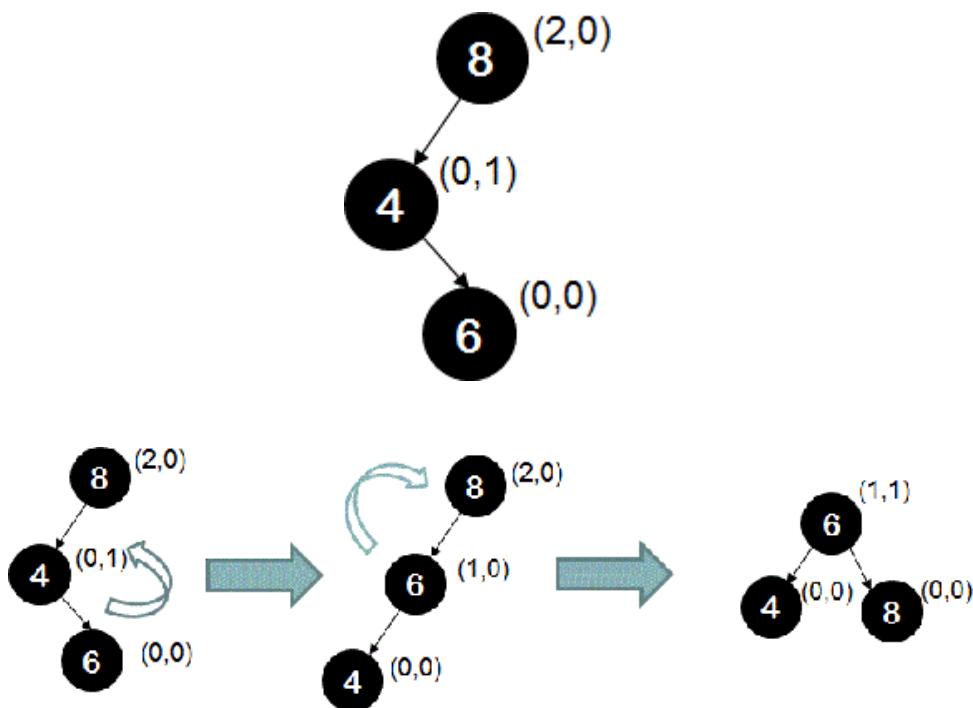


Essa diferença de 2 é construída nos dois ramos esquerdos - Ramo 5-4 e Ramo 4-3. Concordam? Se a diferença de 2 for construída por dois ramos esquerdos, chamamos esse caso de um Caso Esquerdo-Esquerdo (Genial, né?). Neste caso, realizamos uma rotação à direita no Ramo 5-4 como mostrado na imagem abaixo de forma a **rebalancear a árvore**.





Já na imagem abaixo, a Árvore AVL tem sua propriedade violada no Nô 8. A altura da subárvore esquerda é maior do que a altura da subárvore direita em 2. Essa diferença de 2 é construída por um ramo esquerdo (Ramo 8-4) e um ramo direito (Ramo 4-6), logo é um Caso Esquerdo-Direito. Para restaurar o equilíbrio de altura, executamos uma rotação esquerda seguida de uma rotação direita. Vejam...



Então, galera, caso a árvore não esteja balanceada, é necessário seu balanceamento através de rotações. No Caso Esquerda-Esquerda, basta fazer uma rotação simples para direita no nô desbalanceado. No caso Esquerda-Direita, temos que fazer uma rotação dupla, i.e., faz-se uma rotação para esquerda no nô filho e uma rotação para direita no nô desbalanceado.

No caso Direita-Direita, basta fazer uma rotação simples para a esquerda no nô desbalanceado. No caso Direita-Esquerda, temos que fazer uma rotação dupla, i.e., faz-se uma rotação para direita no nô filho, seguida de uma rotação para esquerda no nô desbalanceado. Bacana? Vocês entenderão isso melhor nos exercícios. Vamos ver agora a complexidade logarítmica dessas estruturas.

ÁRVORE BINÁRIA DE BUSCA

ALGORITMO

CASO MÉDIO

PIOR CASO

| | | |
|----------|-------------|--------|
| Espaço | $O(n)$ | $O(n)$ |
| Busca | $O(\log n)$ | $O(n)$ |
| Inserção | $O(\log n)$ | $O(n)$ |
| Remoção | $O(\log n)$ | $O(n)$ |

ÁRVORE B / ÁRVORE AVL

| ALGORITMO | CASO MÉDIO | PIOR CASO |
|-----------|-------------|-------------|
| Espaço | $O(n)$ | $O(n)$ |
| Busca | $O(\log n)$ | $O(\log n)$ |
| Inserção | $O(\log n)$ | $O(\log n)$ |
| Remoção | $O(\log n)$ | $O(\log n)$ |

Quem quiser brincar de Árvore Binária de Busca

<https://www.cs.usfca.edu/~galles/visualization/BST.html>

Quem quiser brincar de Árvore AVL

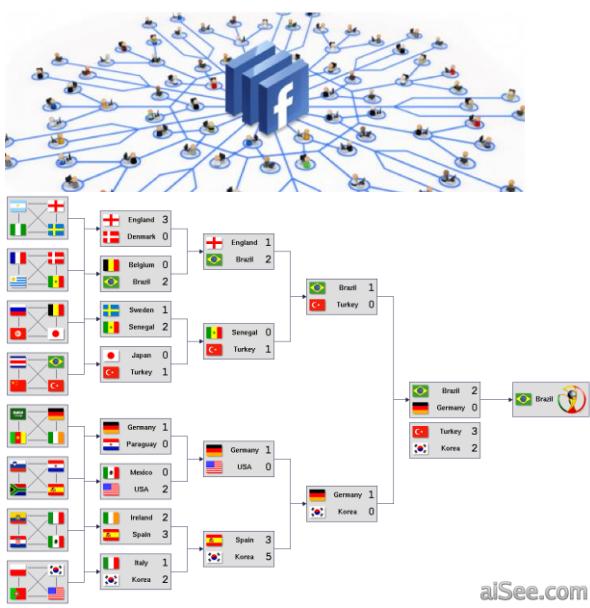
<https://www.cs.usfca.edu/~galles/visualization/AVLtree.html>



GRAFOS

Fiz uma disciplina na faculdade chamada Teoria dos Grafos! Aquilo era absurdamente complexo, mas para concursos a teoria é beeem mais tranquila e muito rara de cair. Portanto fiquem tranquilos, bacana? Uma definição de grafo afirma que ele é uma estrutura de dados que consiste em um **conjunto de nós** (ou vértices) e um **conjunto de arcos** (ou arestas).

Em outras palavras, podemos dizer que é simplesmente um conjunto de pontos e linhas que conectam vários pontos. Ou também que é uma representação abstrata de um conjunto de objetos e das relações existentes entre eles. Uma grande variedade de estruturas do mundo real podem ser representadas **abstratamente** através de **grafos**. Professor, pode me passar um exemplo? Claro!



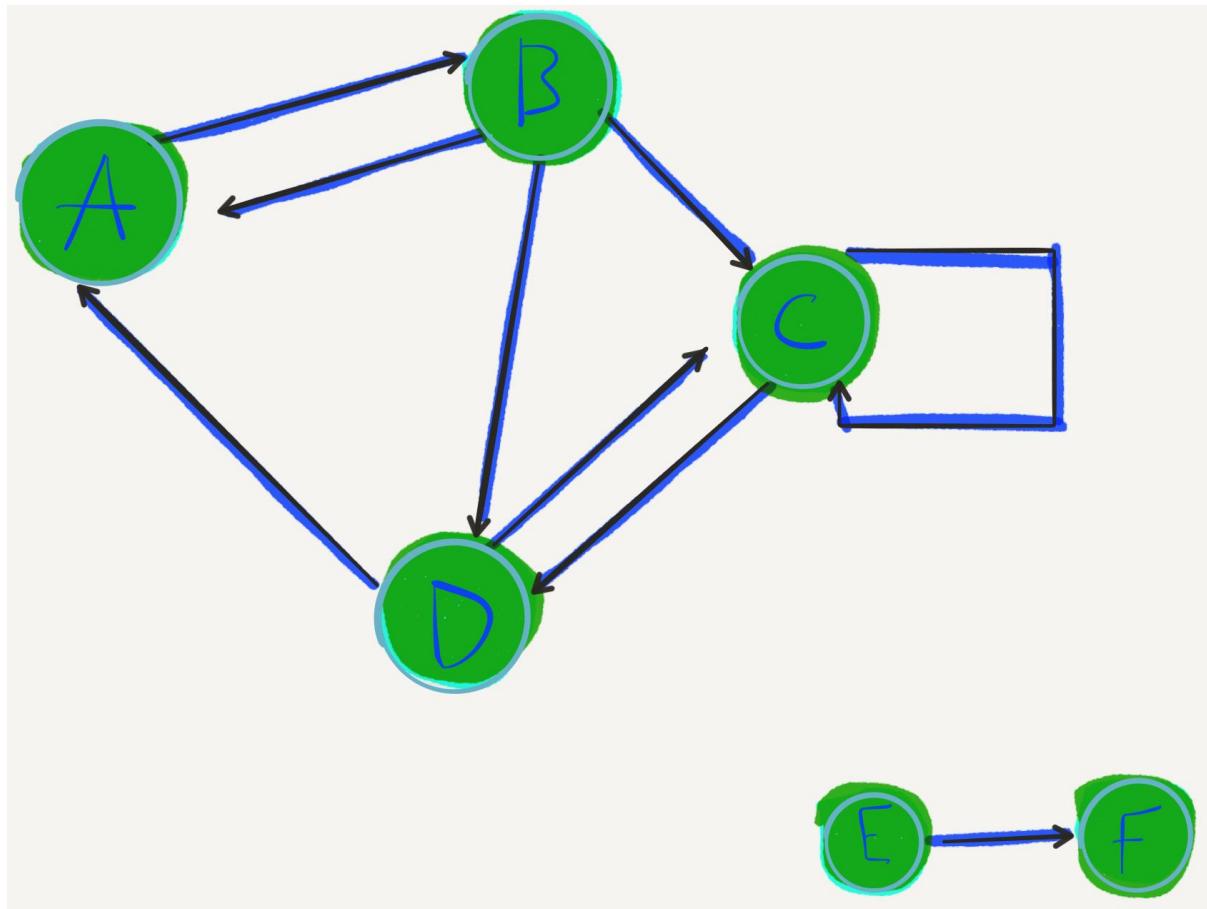
Vejam só como se trata de um conceito bastante abrangente! O Facebook pode ser considerado um grafo (pessoas se interligam)



através de amizades); um mapa rodoviário pode ser considerado um grafo (cidades se relacionam através de estradas); as eliminatórias de um campeonato de futebol também podem ser consideradas um grafo (times jogam entre si para ganhar o campeonato).

Abaixo temos um exemplo de um grafo que eu desenhei e que nos ajudará a entender melhor a terminologia utilizada. Seguem as premissas:

- $N = \{A, B, C, D, E, F\}$
 - $G = \{(A,B), (B,A), (B,C), (B,D), (C,C), (D,A), (D,C), (E,F)\}$



▪ **Nó, Nodo ou Vértice:**

Qualquer elemento de um conjunto N.

▪ **Aresta ou Arco:**

Qualquer elemento de um conjunto A.

▪ **Nós Adjacentes** (Relação de Adjacência):

São aqueles nós ligados por um arco (Ex: A é adjacente a D).

▪ **Arco Incidente** (Relação de Incidência):

São aqueles arcos que levam a um nó (Ex: (C,D) é incidente em D).



■ **Grau:**

É a quantidade de arcos que partem ou chegam em/de um nó.

■ **Caminho:**

Sequência de arcos que levam de um nó a outro (Ex: A → C <(A,B), (B,D), (D,C)>).

■ **Círculo ou Ciclo:**

É o caminho que leva ao mesmo nó do qual saiu (Ex: <(A,B), (B,D), (D,A)>).

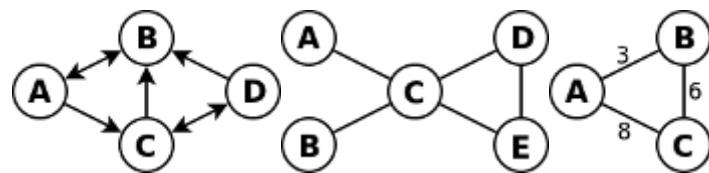
■ **Laço:**

É o círculo de um único arco (<(C,C)>).

■ **Ordem:**

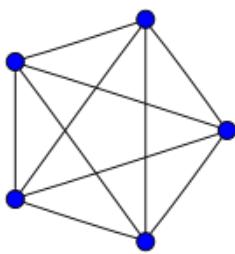
É a quantidade de vértices totais do grafo.

Vamos ver agora alguns conceitos importantes! Um grafo pode ser dirigido – também chamado direcionado, orientado ou dígrafo –, se as arestas tiverem uma direção (imagem da esquerda), ou não dirigido, se as arestas não tiverem direção (imagem central). Se as arestas tiverem associado um peso ou custo, o grafo passa a ser chamado **ponderado, valorado ou pesado** (imagem da direita).



Um grafo simples é aquele que não contém laços. Um grafo vazio é aquele que contém exclusivamente vértices (não contém arcos). Um grafo misto é aquele que possui arestas dirigidas e não-dirigidas. Um grafo trivial é aquele que possui somente um vértice. Um grafo é denso se contém muitos arcos em relação ao número de vértices e esparsos se contém poucos arcos. Como assim, professor?



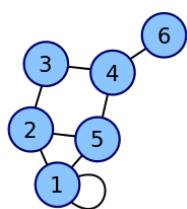


Um grafo é denso se o seu número de arcos é da mesma ordem que o quadrado do número de vértices; e é esparso se o número de arcos for da mesma ordem que o número de vértices. Um grafo é **regular** se todos os vértices têm o mesmo grau; e é **completo** se todo vértice é adjacente a todos os outros vértices (o grafo ao lado é regular e completo).

Um grafo é dito **conexo ou conectado** quando existir pelo menos um caminho entre cada par de vértices. Caso contrário, ele será dito desconexo, isto é, se há pelo menos um par de vértices que não esteja ligado a nenhuma cadeia (caminho). Vejam a imagem do grafo que eu desenhei lá em cima! Ele é conexo ou desconexo? Ele é desconexo, há um par de vértices (E,F) que não está ligado a nenhum caminho.

Se o grafo for **direcionado/orientado**, um grafo é dito fortemente conexo se existir um caminho de $A \rightarrow B$ e de $B \rightarrow A$, para cada par (A,B) de vértices de um grafo. Em outras palavras, o grafo é fortemente conexo se cada par de vértices participa de um circuito. Isto significa que cada vértice pode ser alcançável partindo-se de qualquer outro vértice do grafo.

Galera, existem outras maneiras de representar um grafo. Uma das maneiras mais comuns é por meio de uma **Matriz de Adjacências**. A definição precisa das entradas da matriz varia de acordo com as propriedades do grafo que se deseja representar, porém de forma geral o valor a_{ij} guarda informações sobre como os vértices v_i e v_j estão relacionados (isto é, informações sobre a adjacência de v_i e v_j).



Para representar um grafo não direcionado, simples e sem pesos, basta que as entradas a_{ij} da Matriz A contenham 1 se v_i e v_j são adjacentes e 0, caso contrário. Se as arestas do grafo tiverem pesos, a_{ij} pode conter, ao invés de 1 quando houver uma aresta entre v_i e v_j , o peso dessa mesma aresta.

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

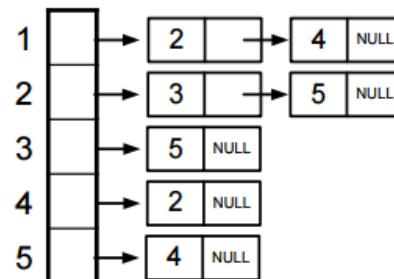
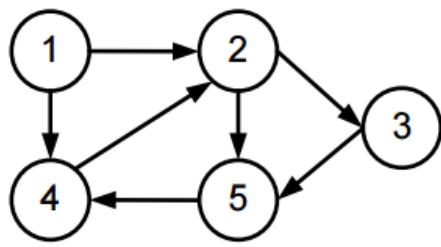
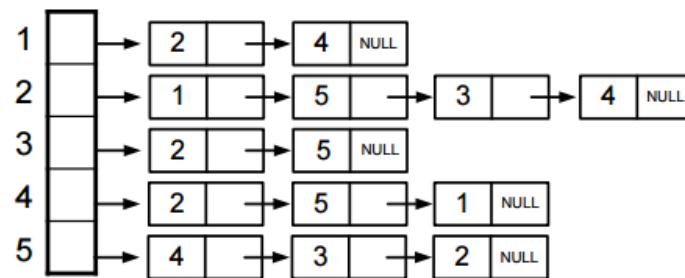
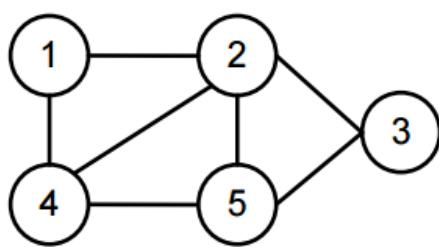
Vamos entender? O elemento $A_{11} = 1$, significando que o Nô 1 tem adjacência com o Nô 1 (ele mesmo, basta ver a figura); o elemento $A_{12} = 1$, significando que o Nô 1 tem adjacência com o Nô 2; elemento $A_{13} = 0$, significando que o Nô 1 não tem adjacência com o Nô 3; o elemento $A_{14} = 0$, significando que o Nô 1 não tem adjacência com o Nô 4.

O elemento $A_{15} = 1$, significando que o Nô 1 tem adjacência com o Nô 5; o elemento $A_{16} = 0$, significando que o Nô 1 não tem adjacência com o Nô 6. Galera, agora ficou Fácil de entender, concordam? Lembrando que, se fosse um grafo ponderado, bastaria colocar o peso, em vez de colocar 1 na Matriz de Adjacências. Existem só mais alguns detalhes.



Em grafos não direcionados, as matrizes de adjacência são simétricas ao longo da diagonal principal - isto é, a entrada a_{ij} é igual à entrada a_{ji} . **Matrizes de Adjacência de grafos direcionados**, no entanto, não são assim. Num dígrafo sem pesos, a entrada a_{ij} da matriz é 1 se há um arco de v_i para v_j e 0, caso contrário. Há outra maneira de representar também chamada Lista de Adjacências.

Se o grafo é não direcionado, cada entrada é um conjunto de dois nós contendo as duas extremidades da aresta correspondente; se ele for dirigido, cada entrada é uma tupla de dois nós, um indicando o nó de origem e o outro denotando o nó destino do arco correspondente. É bem simples, para cada nó, eu escrevo suas adjacências. No exemplo anterior: Nô 1 = 1, 2, 5; Nô 2 = 1, 3, 5; e assim por diante.



Professor, qual é melhor? Cara, cada representação tem suas **vantagens e desvantagens!** É fácil encontrar todos os vértices adjacentes a um vértice dado em uma representação lista de adjacência; você simplesmente lê a sua lista de adjacência. Com uma matriz de adjacência em vez disso se tem que pesquisar mais de uma linha inteira, gastando tempo $O(n)$.

Se, pelo contrário, deseja realizar um teste de vizinhança em dois vértices (isto é, determinar se eles têm uma aresta entre eles), uma matriz de adjacência proporciona isso na hora. No entanto, este teste de vizinhança em uma lista de adjacências requer **tempo proporcional ao número de arestas** associado com os dois **vértices**. Há diversos outros aspectos também a considerar, como tamanho.

Pensem comigo! Para um grafo com uma Matriz de Adjacência esparsa (não densa), uma representação de Lista de Adjacências do grafo ocupa **menos espaço**, porque ele não usa nenhum espaço para representar as arestas que não estão presentes. Lembram-se da lista? Nós só representamos os nós adjacentes, em contraste com a Matriz de Adjacência. No entanto, quanto mais denso, isso pode mudar.

(CESPE/CEBRASPE – 2017 – TRF-1) Acerca dos conceitos de árvores e grafos, julgue o item que se segue.

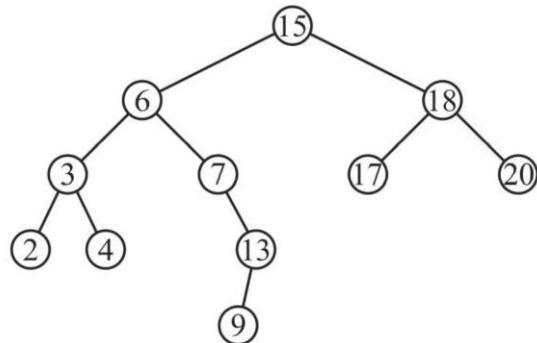


A soma dos graus de todos os vértices de um grafo é sempre par.

Comentários:

Um vértice é um nó do grafo. Já o grau de um vértice é o número de arestas do vértice. Cada aresta conta dois graus para a soma total, sendo um grau para cada vértice que ela conecta. Por isso, a soma será sempre par. (Gabarito: Correto).

(CESPE/CEBRASPE – 2020 – TJ-PA)



Thomas H. Cormen et al. **Algoritmos**: teoria e prática. Editora Campus, v. 2, 2002. p. 207.

De acordo com a figura anterior, o procedimento

CONSULTA (x)

```
1 while esquerda [x] ≠ NIL
2   do x ← esquerda [x]
3   return x
```

realiza, na árvore, a consulta de

- a) search.
- b) minimum.
- c) maximum.
- d) successor.
- e) predecessor.

Comentários:

Trata-se de uma árvore binária. Nessa estrutura, os nós filhos da esquerda sempre possuem um valor menor do que os da direita. Os menores valores da árvore estarão à esquerda.

Dito isso, o algoritmo percorre os nós da esquerda sempre. Ou seja, vai chegar no número 2.

Significa que está procurando o menor valor da árvore. Ou seja, resposta é *minimum*. (Gabarito: Letra B)



HASHING

As Tabelas de Dispersão, também conhecidas como Tabelas de Espelhamento ou Tabelas Hashing, armazenam uma coleção de valores, sendo que cada valor está associado a uma chave. Tais chaves têm que ser todas distintas e são usadas para mapear os valores na tabela. Esse mapeamento é feito por uma **função de hashing**, que chamaremos de **h**.

Por exemplo, podemos implementar uma tabela de dispersão usando um vetor com oito posições e utilizar $h(x) = x \text{ MOD } 8$ como função de hashing¹. Dizemos que $h(k)$ é a posição original da chave k e é nessa posição da tabela que a chave k deve ser inserida. A imagem abaixo ilustra a inserção de uma coleção de valores com suas respectivas chaves numa Tabela de Dispersão.

| | CHAVE | VALOR |
|---|-------|-------|
| 0 | | |
| 1 | 25 | LIA |
| 2 | 18 | ANA |
| 3 | | |
| 4 | 5 | RUI |
| 5 | | |
| 6 | | |
| 7 | 31 | GIL |

Observe que o valor Gil foi colocado na posição 7 da tabela, pois a chave associada a Gil é 31 e $h(31) = 7$. O que aconteceria se tratássemos de inserir nessa tabela o valor Ivo com chave 33? Observe que $h(33) = 1$, mas a posição 1 da tabela já está ocupada. Dizemos que as chaves 25 e 33 colidiram. Mais precisamente, duas chaves x e y colidem se $h(x) = h(y)$.

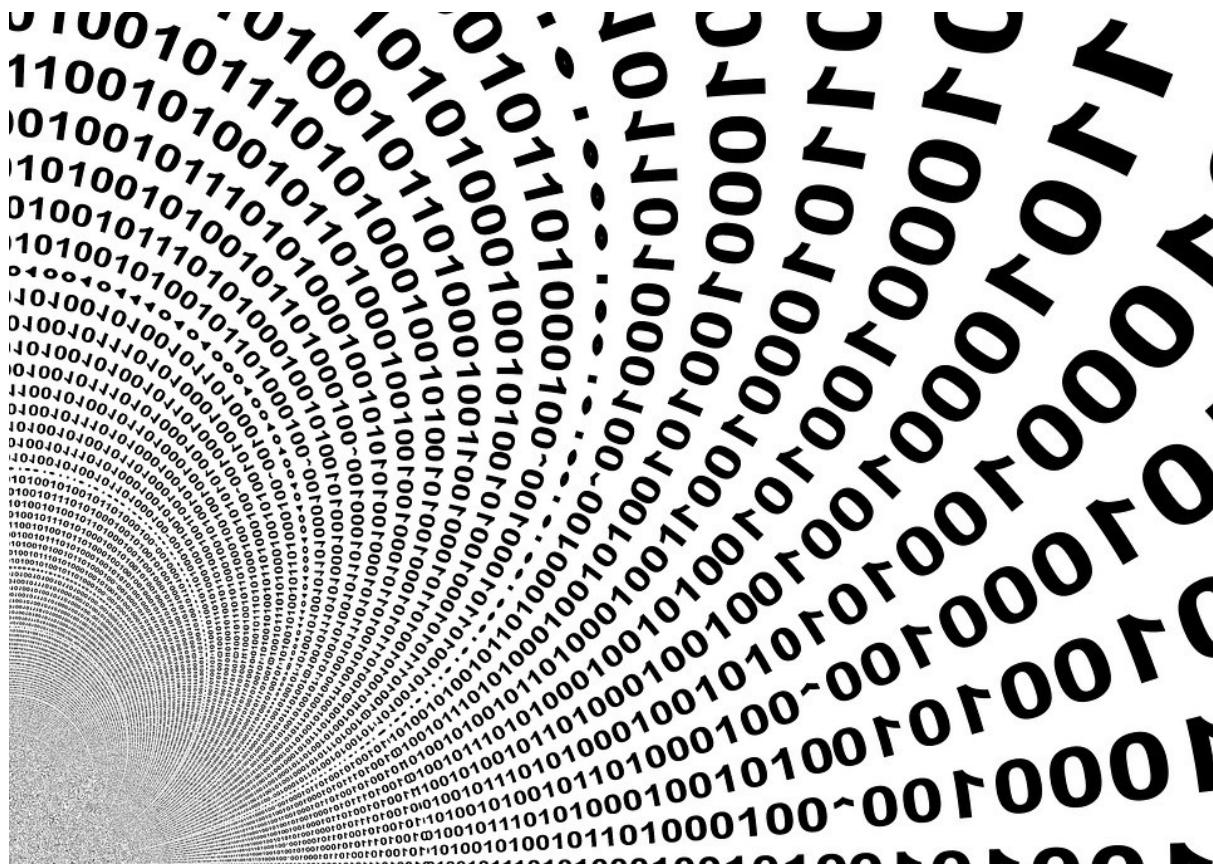
Note que o problema da colisão de chaves ocorre porque, na maioria dos casos, o domínio das chaves é maior que a quantidade de posições da tabela. Sendo assim, pelo princípio da casa de pombos, qualquer função do conjunto das chaves para o conjunto das posições da tabela **não é injetora**.

Não é aceitável recusar a inserção de uma chave que colida com outra já existente na tabela se ela ainda tiver posições livres. Precisamos de alguma estratégia para lidar com as colisões de chaves. Há diversas técnicas para lidar com as colisões, tais como Hashing Fechado ou Hashing Aberto. O Método de Hashing é recomendado para um **grande número de dados** que possuam faixas de **valores variáveis**.

¹ Galera... o MOD representa o resto de uma divisão. Ex: $10 \bmod 8 = 2$, porque $10/8$ possui quociente 1 e resto 2. Em outras palavras, $8*1 + 2 = 10$.



BITMAPS



Nesta aula vamos aprender um pouco mais sobre indexação utilizando **mapas de bits, os bitmaps**. Não estamos falando do formato de imagem que todos devem conhecer, bpm, muito utilizado antes de haver algoritmos de compressão, mas de uma estrutura de indexação que facilita o acesso a informações disponibilizadas em bancos de dados.

Antes de apresentar os bitmaps e como são utilizados para recuperar informações, temos que relembrar alguns conceitos de bancos de dados. Uma tabela é definida por suas colunas e por suas entradas (linhas) com os dados armazenados, chamadas de tuplas. Cada coluna apresenta uma característica da tabela, enquanto cada tupla apresenta uma observação da tabela. Para facilitar, vamos criar uma tabela de exemplo, representando “Banda”.

BANDA

| ROWID | NOME | PAÍS | ESTILO |
|-------|---------------|------------|------------|
| 1 | Radiohead | Inglaterra | Indie Rock |
| 2 | The Killers | EUA | Rock |
| 3 | Beach House | EUA | Indie Rock |
| 4 | The Cure | Inglaterra | Indie Rock |
| 5 | The Cardigans | Suécia | Rock |
| 6 | R.E.M. | EUA | Rock |
| 7 | The Hives | Suécia | Punk Rock |
| 8 | The Who | Inglaterra | Rock |



| | | | |
|----|-------------|------------|-----------|
| 9 | Green Day | EUA | Punk Rock |
| 10 | Sex Pistols | Inglaterra | Punk Rock |

Ainda relembrando um pouco de banco de dados, se quisermos buscar toda as bandas inglesas, podemos realizar a consulta abaixo:

```
SELECT * FROM BANDA WHERE PAIS = 'Inglaterra'
```

E se quiséssemos saber quais são dos EUA e da Inglaterra? Teríamos a consulta a seguir

```
SELECT * FROM BANDA WHERE PAIS = 'Inglaterra' OR PAIS = 'EUA'
```

As consultas funcionam normalmente, mas para realizá-las, o sistema gerenciador de banco de dados (SGDB) deve percorrer todas as tuplas (linhas/observações) e comparar 'Inglaterra' com o valor da coluna 'PAIS', na primeira consulta, e com 'Inglaterra' e 'EUA' na segunda. Essas "varreduras", chamadas de FULL TABLE SCAN, são muito custosas para o banco e, dependendo do tamanho da tabela, podem levar a um tempo excessivamente grande para serem executadas. Um modo de acessar de forma mais direta, sem necessidade de realizar pesquisas custosas é a criação de índices. Existem muitas técnicas para criar índices e uma delas é a utilização dos bitmaps, ou ainda, mapas de bits.

Um bitmap deve indicar quais tuplas possuem um certo valor para a coluna escolhida para se criar o índice. Se consultas como a que mostramos for muito comum, é válido criarmos um índice para a coluna 'PAIS' e facilitar o acesso.

O bitmap é uma tabela onde as linhas são os possíveis valores da coluna selecionada para criar o índice (nesse caso, todos os valores de 'PAIS') e as colunas são os números das tuplas (ROWID). Para cada linha do bitmap, serão preenchidos os valores 0 e 1 caso o valor de 'PAIS' da referida tupla da tabela 'Banda' ser ou não daquele país. Ou seja, para o caso específico, teremos três tuplas no bitmap, cada uma correspondente a um valor de 'PAIS': EUA, Inglaterra e Suécia. As colunas do bitmap são os números das tuplas da tabela 'Banda', ou seja: 1, 2, 3, 4, 5, 6, 7, 8, 9 e 10. Professor, não entendi nadinha! Para visualizar melhor como um bitmap é construído, vamos criar um representando a coluna 'PAIS':

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|------------|---|---|---|---|---|---|---|---|---|----|
| EUA | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| Inglaterra | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| Suécia | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |

Para entender o preenchimento temos que nos remeter à tabela original 'Banda'. Percebem que as bandas da Inglaterra são as de número 2, 3, 6, e 9. Desse modo, no bitmap, preenchemos com valor 1 as colunas correspondentes aos números (ROWID) das tuplas na tabela original. Da mesma forma se faz com EUA e Suécia. Percebem também que, uma vez uma linha preenchida com 1, as outras colunas serão obrigatoriamente 0, pois uma banda somente pode ser de um país e o índice deve refletir o mesmo.

Professor, e isso serve para... Se quisermos realizar as consultas anteriores utilizando os índices em bitmap temos que executar uma varredura completa, assim como antes de termos o índice, mas utilizando os bitmaos não haverá comparações de strings, mas



somente uma avaliação bit a bit, que é muito mais performática. Ademais, é possível compactar os bitmaps com algoritmos que reduzem bastante o espaço exigido dessas estruturas.

Os bitmaps são muito utilizados para criação de índices em SGDBs Oracle, PostgreSQL, Teradata, dentre outros. O seu uso é mais premente e performático em sistemas OLAP (Online Analytical Processing), pois nestes a necessidade de atualização dos dados é muito menor, algo que é relativamente custoso no uso de bitmaps, pois, qualquer alteração nas colunas onde há índices criados, deve-se atualizar os bits do mapa.

Somente como último exemplo, vamos criar um bitmap para outra coluna da tabela 'Banda', agora para a coluna 'Estilo'. Quantas linhas o bitmap dessa coluna terá? E quantas colunas?

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|-------------------|----------|----------|----------|----------|----------|----------|----------|----------|----------|-----------|
| Rock | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| Indie Rock | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Punk Rock | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |

Se fizermos uma pesquisa para saber quais bandas são de punk rock ou indie rock, pelo bitmap podemos rapidamente verificar que são as tuplas 1, 3, 4, 7, 9 e 10. Um computador pode ler vários bits de uma vez ao vez de realizar a comparação um a um, aumentando muito a performance de pesquisas que utilizam o índice em bitmap.

Mostramos a criação de bitmaps para somente uma coluna, mas é possível criarmos para mais de uma. Esses índices são interessantes quando consultas frequentes com mais de uma coluna. Um exemplo seria consultar bandas que são da Inglaterra que tocam o estilo indie rock. Para construir um índice com duas ou mais colunas, temos que representar todas as combinações possíveis entre os valores das colunas e preencher os bits com o valor 1 nas tuplas onde os dois valores ocorrem simultaneamente. Como exemplo, vamos criar um índice para PAISESTILO:

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|----------------------------|----------|----------|----------|----------|----------|----------|----------|----------|----------|-----------|
| EUARock | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| EUAIndieRock | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| EUAPunkRock | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| InglaterraRock | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| InglaterraIndieRock | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| InglaterraPunkRock | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| SuéciaRock | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| SuéciaIndieRock | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SuéciaPunkRock | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

A consulta das bandas inglesas que tocam indie rock, portanto, são as tuplas 1 e 4.

Bitmaps, portanto, são estruturas utilizadas como índices em bancos de dados para facilitar consultas, principalmente com colunas com pouca cardinalidade, ou seja, poucos valores possíveis.



ESTRUTURA DE ARQUIVOS

É senso comum que há a necessidade de **salvar** ou **guardar** nossos **dados** ou **programas**. E onde podemos fazer isso? Em geral, em dispositivos persistentes (Discos, PenDrive, Fitas, etc). O armazenamento de pequenos volumes de dados, via de regra, não encerra grandes problemas no que diz respeito à distribuição dos registros dentro de um arquivo.

À medida que cresce o volume de dados, a frequência ou a complexidade dos acessos, crescem também os problemas de eficiência do armazenamento dos arquivos e do acesso a seus registros, sendo a sofisticação das técnicas de armazenamento e recuperação de dados uma consequência da necessidade de **acesso rápido** a grandes arquivos ou arquivos muito solicitados.

A maneira intuitiva de **armazenar** um arquivo consiste na **distribuição** dos seus registros em uma ordem arbitrária, um após o outro, dentro da **área destinada**. Esta ordem pode ser, por exemplo, aquela na qual os registros são gerados, causando dificuldade na localização e perda de eficiência, mas esta técnica é bastante usada, principalmente durante as fases preliminares da geração de um arquivo.

Quando escrevemos um programa para gerar um arquivo de dados, costumamos agrupar os campos que fornecem informações sobre um mesmo indivíduo em um registro. Um arquivo é formado por uma coleção de registros lógicos, cada um deles representando um objeto ou entidade. E o que seria um registro lógico? É uma sequência de itens, sendo cada item chamado **campo ou atributo**.

Cada item corresponde a uma **característica ou propriedade** do objeto representado. Existem três formas de acessar arquivos: sequencial, direta ou indexado. No primeiro caso, os registros são lidos em sequência, um após o outro. A cada registro lido o indicador de posição de arquivo é avançado para que a próxima leitura ocorra iniciando naquele ponto.

No segundo caso, pode-se acessar qualquer posição do arquivo, para tanto é necessário preciso ajustar o indicador de posição do arquivo para onde deseja-se realizar uma operação de leitura/escrita. No terceiro caso, o arquivo é visto como um conjunto de registros indexados por uma chave.



QUESTÕES COMENTADAS – ESTRUTURA DE DADOS - MULTIBANCAS

1. (CESPE/CEBRASPE – 2020 – Ministério da Economia) A respeito de dados, informação, conhecimento e inteligência, julgue o próximo item.

Embora com características particulares, dados não estruturados podem ser classificados em sua totalidade, assim como os dados estruturados.

Comentários:

Dados não estruturados não possuem uma estrutura pré-definida e, portanto, é computacionalmente muito difícil que sejam classificados ou processados por meio de métodos automatizados. Já os dados estruturados possuem uma estrutura pré-definida, tornando mais fácil de serem classificados ou processados por métodos automatizados. Ou seja, os dois não são a mesma coisa para serem classificados em sua totalidade.

Gabarito: Errado

2. (FGV – 2015 – DPE/MT – Analista de Sistemas) No desenvolvimento de sistemas, a escolha de estruturas de dados em memória é especialmente relevante. Dentre outras classificações, é possível agrupar essas estruturas em lineares e não lineares, conforme a quantidade de sucessores e antecessores que os elementos da estrutura possam ter. Assinale a opção que apresenta, respectivamente, estruturas de dados lineares e não lineares.

- a) Tabela de dispersão e fila.
- b) Estrutura de seleção e pilha.
- c) Pilha e estrutura de seleção.
- d) Pilha e árvore binária de busca.
- e) Fila e pilha.

Comentários:

Além dessa classificação, existe outra também importante: Estruturas Lineares e Estruturas Não-Lineares. As Estruturas Lineares são aquelas em que cada elemento pode ter um único predecessor (exceto o primeiro elemento) e um único sucessor (exceto o último elemento). Como exemplo, podemos citar Listas, Pilhas, Filas, Arranjos, entre outros.

Já as Estruturas Não-Lineares são aquelas em que cada elemento pode ter mais de um predecessor e/ou mais de um sucessor. Como exemplo, podemos citar Árvores, Grafos e Tabelas de Dispersão. Essa é uma classificação muito importante e muito simples de entender. Pessoal, vocês perceberão que esse assunto é cobrado de maneira superficial na maioria das questões, mas algumas são nível doutorado!

Conforme vimos em aula, trata-se de pilha e árvore respectivamente.

Gabarito: Letra D



3. (CESPE – 2010 – DETRAN/ES – Analista de Sistemas) Um tipo abstrato de dados apresenta uma parte destinada à implementação e outra à especificação. Na primeira, são descritas, em forma sintática e semântica, as operações que podem ser realizadas; na segunda, os objetos e as operações são representados por meio de representação, operação e inicialização.

Comentários:

A Especificação Sintática define o nome do tipo, suas operações e o tipo dos argumentos das operações, definindo a assinatura do TAD. A Especificação Semântica descreve propriedades e efeitos das operações de forma independente de uma implementação específica. E a Especificação de Restrições estabelece as condições que devem ser satisfeitas antes e depois da aplicação das operações.

Conforme vimos em aula, temos duas partes: Especificação e Implementação. Sendo que a especificação se divide em Especificação Sintática e Semântica e Implementação se divide em Representação e Algoritmos. Logo, a questão está invertida: na segunda, são descritas, em forma sintática e semântica, as operações que podem ser realizadas; na primeira, os objetos e as operações são representados por meio de representação, operação e inicialização.

Gabarito: Errado

4. (CESPE – 2010 – TRT/RN – Analista de Sistemas) O tipo abstrato de dados consiste em um modelo matemático (v, o), em que v é um conjunto de valores e o é um conjunto de operações que podem ser realizadas sobre valores.

Comentários:

Por fim, vamos falar sobre Tipos Abstratos de Dados (TAD). Podemos defini-lo como um modelo matemático (v, o), em que v é um conjunto de valores e o é um conjunto de operações que podem ser realizadas sobre valores. Eu sei, essa definição é horrível de entender! Para compreender esse conceito, basta lembrar de abstração, i.e., apresentar interfaces e esconder detalhes.

Conforme vimos em aula, a questão está conforme nós descrevemos em nossa aula. Essa é a definição formal de Tipos Abstratos de Dados.

Gabarito: Correto

5. (CESPE – 2010 – BASA – Analista de Sistemas) A escolha de estruturas internas de dados utilizados por um programa pode ser organizada a partir de TADs que definem classes de objetos com características distintas.

Comentários:

Em outras palavras, o nível semântico trata do comportamento de um tipo abstrato de dados; e o nível sintático trata da apresentação de um tipo abstrato de dados. Podemos dizer, então, que o TAD encapsula uma estrutura de dados com características semelhantes – podendo ser formado por outros TADs –, e esconde a efetiva implementação dessa estrutura de quem a manipula.

Conforme vimos em aula, definem classes de objetos com características semelhantes.



6. (CESPE – 2010 – BASA – Analista de Sistemas) A descrição dos parâmetros das operações e os efeitos da ativação das operações representam, respectivamente, os níveis sintático e semântico em que ocorre a especificação dos TDAs.

Comentários:

A Especificação Sintática define o nome do tipo, suas operações e o tipo dos argumentos das operações, definindo a assinatura do TAD. A Especificação Semântica descreve propriedades e efeitos das operações de forma independente de uma implementação específica. E a Especificação de Restrições estabelece as condições que devem ser satisfeitas antes e depois da aplicação das operações.

Conforme vimos em aula, realmente se trata respectivamente do nível sintático e semântico.

7. (FCC – 2010 – TRE/AM – Analista de Sistemas) Em relação aos tipos abstratos de dados - TAD, é correto afirmar:

- a) O TAD não encapsula a estrutura de dados para permitir que os usuários possam ter acesso a todas as operações sobre esses dados.
- b) Na transferência de dados de uma pilha para outra, não é necessário saber como a pilha é efetivamente implementada.
- c) Alterações na implementação de um TAD implicam em alterações em seu uso.
- d) Um programador pode alterar os dados armazenados, mesmo que não tenha conhecimento de sua implementação.
- e) TAD é um tipo de dados que esconde a sua implementação de quem o manipula.

Comentários:

Em outras palavras, o nível semântico trata do comportamento de um tipo abstrato de dados; e o nível sintático trata da apresentação de um tipo abstrato de dados. Podemos dizer, então, que o TAD encapsula uma estrutura de dados com características semelhantes – podendo ser formado por outros TADs –, e esconde a efetiva implementação dessa estrutura de quem a manipula.

- (a) Errado, ele encapsula a estrutura de dados; (b) Correto, não é necessário saber a implementação, porém a FCC marcou o gabarito como errado; (c) Errado, a implementação pode mudar livremente; (d) Errado, sem conhecimento da implementação, ele não poderá alterar dados armazenados; (e) Correto, esse item também está correto (Lembrem-se: Na FCC, muitas vezes tempos que marcar a mais correta ou a menos errada – infelizmente!).



8. (FCC – 2009 – TRE/PI – Analista de Sistemas) Em relação a tipos abstratos de dados, é correto afirmar que:

- a) o TAD não encapsula a estrutura de dados para permitir que os usuários possam ter acesso a todas as operações disponibilizadas sobre esses dados.
- b) algumas pilhas admitem serem declaradas como tipos abstratos de dados.
- c) filas não permitem declaração como tipos abstratos de dados.
- d) os tipos abstratos de dados podem ser formados pela união de tipos de dados primitivos, mas não por outros tipos abstratos de dados.
- e) são tipos de dados que escondem a sua implementação de quem o manipula; de maneira geral as operações sobre estes dados são executadas sem que se saiba como isso é feito.

Comentários:

Em outras palavras, o nível semântico trata do comportamento de um tipo abstrato de dados; e o nível sintático trata da apresentação de um tipo abstrato de dados. Podemos dizer, então, que o TAD encapsula uma estrutura de dados com características semelhantes – podendo ser formado por outros TADs –, e esconde a efetiva implementação dessa estrutura de quem a manipula.

(a) Errado. Pelo contrário, ele encapsula a estrutura de dados de modo que o usuário não tem acesso a implementação das operações; (b) Correto. Todas elas admitem, porém a FCC marcou o gabarito como errado; (c) Errado. Elas permitem a declaração como tipos abstratos de dados; (d) Errado. Um TAD pode ser formado por outros TADs; (e) Correto. Escondem a implementação de quem os manipula.

Gabarito: Letra E



QUESTÕES COMENTADAS – VETORES E MATRIZES - MULTIBANCAS

1. (IBFC – 2022 – DETRAN-AM) Relacione as duas colunas quanto aos respectivos tipos de Estruturas de Dados:

| | |
|--|------------------------------------|
| (A) Vetores (B) Registros (C) Matrizes | (1) Homogêneas (2) Heterogêneas |
|--|------------------------------------|

- a) A2 - B1 - C2
- b) A1 - B1 - C2
- c) A2 - B2 - C1
- d) A1 - B2 - C1

Comentários:

Por definição, temos que:

- Estruturas de dados homogêneas: seus elementos possuem o mesmo tipo de dado básico.
- Estrutura de dados heterogênea: seus elementos possuem tipos de dados distintos.

Via de regra, um vetor e uma matriz possuem sempre o mesmo tipo de dados: um vetor de inteiro, um vetor de string, um vetor de booleanos, e assim por diante. Portanto, são homogêneos.

Já um registro é um agrupamento de várias variáveis, cada uma podendo ter um tipo de dados diferente. Portanto, é heterogêneo.

Assim, A1 – B2 – C1.

Gabarito: Letra D

2. (IADES – 2022 – ADASA) Com base nas definições referentes à estrutura de dados digitais, à vetorização e à digitalização, assinale a alternativa correta.

- a) A estrutura vetorial é composta por uma grade homogênea de linhas e colunas.
- b) A digitalização é o processo de mudança de documentos cartográficos do formato vetorial para o formato raster.
- c) A digitalização é o processo de transformação de documentos cartográficos do formato vetorial para o formato digital.
- d) A vetorização é o processo de conversão de um arquivo vetorial para o formato raster.
- e) A estrutura matricial é representada por uma matriz com “n” linhas e “m” colunas, na qual cada célula apresenta um valor “z” que pode indicar, por exemplo, uma cor ou um tom de cinza a ele atribuída.

Comentários:

Vamos analisar item a item:

- a) A estrutura vetorial é composta por uma grade homogênea de linhas e colunas.

A estrutura vetorial é linear e homogênea. O correto seria dizer que a estrutura matricial é composta por uma grade homogênea de linhas e colunas, e não vetorial. Falso.



- b) A digitalização é o processo de mudança de documentos cartográficos do formato vetorial para o formato raster.

A digitalização vai transformar documentos analógicos em formato digital. Falso.

- c) A digitalização é o processo de transformação de documentos cartográficos do formato vetorial para o formato digital.

A digitalização vai transformar documentos analógicos em formato digital. Falso.

- d) A vetorização é o processo de conversão de um arquivo vetorial para o formato raster.

A vetorização converte um arquivo raster em arquivo vetorial. Falso.

- e) A estrutura matricial é representada por uma matriz com "n" linhas e "m" colunas, na qual cada célula apresenta um valor "z" que pode indicar, por exemplo, uma cor ou um tom de cinza a ele atribuída.

Esta é uma boa definição de matriz. Correto!

Gabarito: Letra E

3. (UFPRE – 2022 – UFPRE) A estrutura de dados "vetor" (array) é um arranjo unidimensional que pode acomodar múltiplos dados. Sobre essas estruturas de dados, assinale a alternativa incorreta.

- a) Os dados de um vetor são mapeados numa área contígua da memória.

- b) Os dados de um vetor são do mesmo tipo.

- c) Cada um dos dados de um vetor pode ser acessado informando-se o identificador do vetor e o inteiro que indica a ordem do dado na sequência.

- d) Os dados de um vetor são armazenados na memória ordenadamente, em modo crescente.

- e) Pode-se atribuir um dado a um elemento de qualquer posição do vetor, independentemente do que foi atribuído aos demais elementos.

Comentários:

A questão quer a alternativa falsa. Vamos analisar item a item:

- a) Os dados de um vetor são mapeados numa área contígua da memória.

É isso mesmo, os vetores ficam numa área sequencial ou contígua da memória. Verdadeiro.

- b) Os dados de um vetor são do mesmo tipo.

De fato, o vetor é homogêneo, ou seja, todos os seus elementos têm o mesmo tipo básico. Verdadeiro.

- c) Cada um dos dados de um vetor pode ser acessado informando-se o identificador do vetor e o inteiro que indica a ordem do dado na sequência.

Isso mesmo, os dados do vetor podem ser acessados considerando um valor inteiro. Por exemplo, iniciando-se no 0, vetor[0] acessa o elemento na primeira posição, enquanto vetor[3] acessa o elemento na quarta posição. Verdadeiro.

- d) Os dados de um vetor são armazenados na memória ordenadamente, em modo crescente.

Não é regra que os dados em si do vetor estejam armazenados de modo crescente. Eles podem estar, mas não é regra. Falso.

- e) Pode-se atribuir um dado a um elemento de qualquer posição do vetor, independentemente do que foi atribuído aos demais elementos.

Sim, um dado pode ser atribuído a qualquer posição, e os valores das demais posições são irrelevantes. Verdadeiro.

Gabarito: Letra D



4. (FUNDATÉC – 2022 – Prefeitura de Restinga Sêca - RS) Assinale a estrutura de dados linear e estática, caracterizada por uma sequência de elementos de um mesmo tipo de dado e que são armazenados em posições consecutivas de memória.

- a) Matriz.
- b) Lista ligada.
- c) Registro.
- d) Árvore binária.
- e) Vetor.

Comentários:

Vamos de item a item:

- a) Matriz.

Matriz é uma estrutura de dados bidimensional, e não linear. Falso.

- b) Lista ligada.

Uma lista ligada não tem, necessariamente, os seus elementos em posições consecutivas de memória. Falso.

- c) Registro.

Um registro não necessariamente tem os elementos do mesmo tipo. Falso.

- d) Árvore binária.

Uma árvore binária é uma estrutura hierárquica, e não linear. Falso.

- e) Vetor.

A definição do enunciado é uma ótima definição de vetor. Verdadeiro.

Gabarito: Letra E

5. (FGV – 2021 – IMBEL) Considere um conjunto de 65.536 chaves ordenadas, distintas entre si, armazenadas num array.

Com relação ao processo de busca binária, assinale a opção que indica o número máximo de acessos ao array necessários para localizar uma determinada chave qualquer.

- a) 10
- b) 16
- c) 64
- d) 256
- e) 32.768

Comentários:

Para isso, devemos usar a fórmula $\log_2 n = 65.536$.

Na busca binária, o vetor é dividido ao meio sucessivamente, até achar o valor desejado. Vamos ver quantas vezes conseguimos dividir por 2 o valor:

$$65.536 / 2 = 32768 \text{ (1x)}$$

$$32768 / 2 = 16384 \text{ (2x)}$$

$$16384 / 2 = 8192 \text{ (3x)}$$

$$8192 / 2 = 4096 \text{ (4x)}$$

$$4096 / 2 = 2048 \text{ (5x)}$$

$$2048 / 2 = 1024 \text{ (6x)}$$



$$1024/2 = 512 \text{ (7x)}$$

$$512/2 = 256 \text{ (8x)}$$

$$256/2 = 128 \text{ (9x)}$$

$$128/2 = 64 \text{ (10x)}$$

$$64/2 = 32 \text{ (11x)}$$

$$32/2 = 16 \text{ (12x)}$$

$$16/2 = 8 \text{ (13x)}$$

$$8/2 = 4 \text{ (14x)}$$

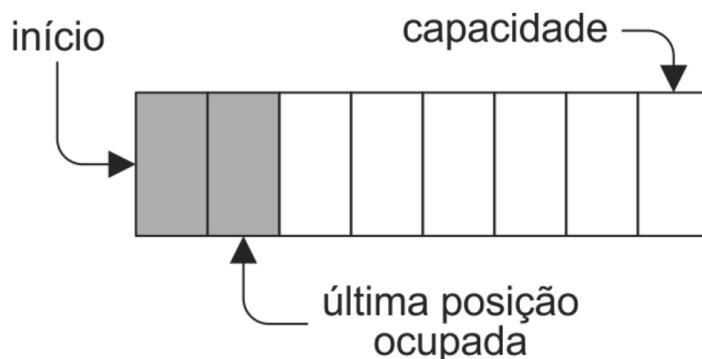
$$4/2 = 2 \text{ (15x)}$$

$$2/2 = 1 \text{ (16x)}$$

Portanto, são, no máximo, 16 vezes.

Gabarito: Letra B

6. (FCC – 2019 – SANASA Campinas) Um Analista de TI necessitou usar uma estrutura de dados simples que utilizasse pouca carga de memória de armazenamento. Tal estrutura é vista como um arranjo cuja capacidade pode variar dinamicamente, isto é, se o espaço reservado for totalmente ocupado e algum espaço adicional for necessário, este será alocado automaticamente não havendo a necessidade de se preocupar com a capacidade de armazenamento ou sua ocupação. Contudo, para que se possa utilizar essa coleção de dados de forma adequada, algumas informações necessárias devem ser mantidas internamente, tais como a quantidade total de elementos e a última posição ocupada na coleção, conforme exemplificado na figura abaixo.



Trata-se da estrutura linear unidimensional

- a) string.
- b) hashing.
- c) árvore.
- d) matriz.
- e) vetor.

Comentários:

Vamos analisar item a item:

- a) string.

Para ser uma string, deveria armazenar caracteres, o que não é informado no enunciado. Falso.

- b) hashing.

Para ser hashing, deveria ser mencionado um algoritmo de hashing, o que não é informado. Falso.

- c) árvore.



Para ser uma árvore, deveriam ser mencionadas estruturas como nós e filhos, o que não é o caso. Falso.

d) matriz.

Para ser uma matriz, a estrutura deveria ter linhas e colunas. Mas ela é uma estrutura unidirecional. Falso.

e) vetor.

As informações ditas no enunciado se referem a um vetor. Verdadeiro.

Gabarito: Letra E

7. (FCC – 2013 – MPE-AM) Considere o vetor vet a seguir:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| S | M | N | A | O | Z | A | A |

Após a execução dos seguintes comandos de atribuição:

aux ← vet[8]

vet[8] ← vet [1]

vet[4] ← vet[6]

vet[6] ← vet[3]

vet[3] ← vet[1] ← aux

A configuração do vetor (do índice 1 ao 8) será

a) ZONAAMAS

b) AMASZONA

c) SMAZONAS

d) AMASSONA

e) AMAZONAS

Comentários:

Vamos passo a passo:

aux ← vet[8]

É definido o valor de vet[8] para a variável aux. É a oitava posição do vetor, ou seja, A. Assim, aux = A.

vet[8] ← vet [1]

À posição 8, é atribuído o valor da posição 1. Assim:

SMNAOZAS

vet[4] ← vet[6]

À posição 4, é atribuído o valor da posição 6. Assim:

SMNZOZAS

vet[6] ← vet[3]

À posição 6, é atribuído o valor da posição 3. Assim:

SMNZONAS

vet[3] ← vet[1] ← aux

As posições 1 e 3, são atribuídos os valores de aux, que é A. Assim:

AMAZONAS

Gabarito: Letra E



8. (FCC – 2012 – TJ-RJ) O algoritmo conhecido como busca binária é um algoritmo de desempenho ótimo para encontrar a posição de um item em

- a) uma árvore B.
- b) uma lista ligada ordenada.
- c) uma árvore de busca binária.
- d) um heap binário.
- e) um vetor ordenado.

Comentários:

O algoritmo de busca binária realiza buscas em uma estrutura linear e ordenada, que permita acesso direto aos seus elementos por meio de um índice numérico, e dividindo a estrutura ao meio e buscando em suas metades.

Por isso, vamos analisar item a item:

- a) uma árvore B.

Uma árvore é uma estrutura hierarquizada, e não linear. Falso.

- b) uma lista ligada ordenada.

Uma lista ligada não permite o acesso direto aos seus elementos por meio de um índice numérico. Falso.

- c) uma árvore de busca binária.

Uma árvore é uma estrutura hierarquizada, e não linear. Falso.

- d) um heap binário.

Um heap binário não está necessariamente com os dados ordenados. Falso.

- e) um vetor ordenado.

Essa é a resposta correta, pois o vetor está ordenado, e permite acesso direto aos elementos por um índice numérico. Correto!

Gabarito: Letra E

9. (FCC - 2009 - TJ-PA - Analista Judiciário - Tecnologia da Informação) Considere uma estrutura de dados do tipo vetor. Com respeito a tal estrutura, é correto que seus componentes são, characteristicamente,

- a) heterogêneos e com acesso FIFO.

- b) heterogêneos e com acesso LIFO.

- c) heterogêneos e com acesso indexado-sequencial.

- d) homogêneos e acesso não indexado.

- e) homogêneos e de acesso aleatório por intermédio de índices.

Comentários:

Vetores possuem componentes homogêneos, i.e., todos os dados são de apenas um único tipo básico de dados. Ademais, seu acesso é aleatório por meio de índices! Bem, seria mais correto dizer que seu acesso é direto. **Gabarito: E**

Gabarito: Letra E



10. (CETAP - 2010 - AL-RR - Analista de Sistemas) Matrizes são estruturas de dados de n-dimensões. Por simplicidade, chamaremos de matrizes as matrizes bidimensionais numéricas (que armazenam números inteiros). Sendo assim, marque a alternativa INCORRETA.

- a) Uma matriz de m linhas e n colunas contêm ($m * n$) dados.
- b) Uma matriz pode ser representada utilizando listas ligadas.
- c) A soma dos elementos de uma matriz pode ser calculada fazendo dois laços aninhados, um sobre as linhas e o outro sobre as colunas.
- d) A soma de duas matrizes de m linhas e n colunas resulta em uma matriz de $2*m$ linhas e $2*n$ colunas.
- e) O produto de duas matrizes de n linhas e n colunas resulta em uma matriz de n linhas e n colunas.

Comentários:

(a) Perfeito, são $M \times N$ colunas; (b) Perfeito, podem ser utilizadas listas ligadas; (c) Perfeito, um laço para as linhas e outro para as colunas; (d) Não, a soma de uma Matriz 3×5 com outra Matriz 3×5 resulta em uma Matriz 3×5 ; (e) Perfeito, uma Matriz 2×2 multiplicada por outra Matriz 2×2 resulta em uma Matriz 2×2 .

Gabarito: Letra D

11. (CESPE - 2010 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação - Arquitetura de Tecnologia) Os dados armazenados em uma estrutura do tipo matriz não podem ser acessados de maneira aleatória. Portanto, usa-se normalmente uma matriz quando o volume de inserção e remoção de dados é maior que o volume de leitura dos elementos armazenados.

Comentários:

Podem, sim, ser acessados de maneira aleatória ou direta, por meio de seus índices. Ademais, usa-se normalmente uma matriz quando o volume de leitura de elementos armazenados é maior que o volume de inserção e remoção de dados. Ora, é possível fazer acesso direto, logo é eficiente mesmo com alto volume de leitura.

Gabarito: Errado

12. (CESPE - 2008 - TRT - 5ª Região (BA) - Técnico Judiciário - Tecnologia da Informação) Entre alguns tipos de estrutura de dados, podem ser citados os vetores, as pilhas e as filas.

Comentários:

Questão extremamente simples – realmente são exemplos de estruturas de dados.

Gabarito: Correto

13. (CESPE - 2011 - EBC - Analista - Engenharia de Software) Vetores são utilizados quando estruturas indexadas necessitam de mais que um índice para identificar um de seus elementos.

Comentários:



Não! Se são necessários mais de um índice, utilizam-se Matrizes! Vetores necessitam apenas de um índice.

Gabarito: Errado

14. (CESPE - 2010 - TRE-BA - Analista Judiciário - Análise de Sistemas) Vetores podem ser considerados como listas de informações armazenadas em posição contígua na memória.

Comentários:

Perfeito! Vetores podem ser considerados como listas de informações? Sim! As informações (dados) são armazenadas em posição contígua na memória? Sim, em geral são armazenados de forma contígua.

Gabarito: Correto

15. (CESPE - 2010 - TRE-BA - Analista Judiciário - Análise de Sistemas) Uma posição específica de um vetor pode ser acessada diretamente por meio de seu índice.

Comentários:

Perfeito! Observem que vetores são diferentes de listas, nesse sentido. Eu posso acessar qualquer elemento diretamente por meio de seu índice.

Gabarito: Correto

8. (FCC - 2016 - Copergás - PE - Analista Tecnologia da Informação) Considere o algoritmo a seguir, na forma de pseudocódigo:

Var n, i, j, k, x: inteiro
Var v: vetor[0..7] inteiro

Início

v[0] ← 12
v[1] ← 145
v[2] ← 1
v[3] ← 3
v[4] ← 67
v[5] ← 9
v[6] ← 45
n ← 8
k ← 3
x ← 0

Para j ← n-1 até k passo -1 faça

 v[j] ← v[j - 1];

 Fim_para

 v[k] ← x;

 Fim

Este pseudocódigo

a) exclui o valor contido na posição x do vetor v.



- b) insere o valor de x entre v[k-1] e v[k] no vetor v.
- c) exclui o valor contido na posição k do vetor v.
- d) tentará, em algum momento, acessar uma posição que não existe no vetor.
- e) insere o valor de k entre v[x] e v[x+1] no vetor v.

Comentários:

O algoritmo, na estrutura de repetição “Para $j \leftarrow n-1$ até k passo -1 faça” percorre o vetor de trás para frente, “empurando” os valores para o final do vetor. De fato, a estrutura vai da posição 7 até a 3 no vetor passando os valores da posição anterior para a posterior. Desse modo $v[6]$ vai para $v[7]$, $v[5]$ para $v[6]$, até que o valor de $v[3]$ fica igual ao valor de $v[4]$. Na última instrução $v[3]$ recebe outro valor. Ou seja, o algoritmo “afasta” os valores, a partir da posição três, para incluir um novo valor, entre $v[2]$ e $v[3]$ ($v[k-1]$ e $v[k]$).

Gabarito: Letra B



QUESTÕES COMENTADAS – LISTA ENCADEADA – MULTIBANCAS

1. (UFV – 2022 – UFV-MG) Considere as afirmativas a seguir sobre estrutura de dados:

- I. Uma estrutura de dados heterogênea envolve a utilização de mais de um tipo básico de dado.
- II. Uma lista encadeada pode ser definida como uma sequência de células em que cada célula contém um elemento e o endereço da célula seguinte.
- III. Uma pilha é uma estrutura de dados baseada no princípio “First In First Out” (FIFO).
- IV. Filas e pilhas são estruturas de dados lineares; o organograma de uma empresa pode ser representado por uma estrutura de árvore.

Está CORRETO o que se afirma, apenas, em:

- a) I e II.
- b) I e III.
- c) I, II e IV.
- d) II, III e IV.

Comentários:

Vamos analisar item a item:

I. Uma estrutura de dados heterogênea envolve a utilização de mais de um tipo básico de dado.

Isso mesmo. Significa que cada elemento da estrutura de dados pode ter mais de um tipo básico de dado. Por exemplo, um inteiro, uma string, um booleano, etc. Verdadeiro.

II. Uma lista encadeada pode ser definida como uma sequência de células em que cada célula contém um elemento e o endereço da célula seguinte.

Essa é a definição de lista encadeada simples. Certo.

III. Uma pilha é uma estrutura de dados baseada no princípio “First In First Out” (FIFO).

Na verdade, a pilha segue a estrutura de dados com a regra LIFO (last-in first-out), ou seja, o último a entrar é o primeiro a sair. Falso.

IV. Filas e pilhas são estruturas de dados lineares; o organograma de uma empresa pode ser representado por uma estrutura de árvore.

Filas e pilhas são estruturas lineares de fato, e uma árvore é uma estrutura hierárquica, e, por isso, um organograma de uma empresa pode ser representado por ela. Verdadeiro.

Corretas: I, II e IV.

Gabarito: Letra C

2. (Quadrix – 2022 – PRODAM-AM) Assinale a alternativa que apresenta o nome do tipo de estrutura em que cada elemento armazena um ou vários dados e um ponteiro para o próximo elemento, que permite o encadeamento e mantém a estrutura linear, sendo que, nesse tipo de estrutura, são abordadas as seguintes operações: inserir no início da lista; inserir no fim; consultar toda a lista; remover um elemento qualquer dela; e esvaziá-la.

- a) lista simplesmente encadeada e não ordenada
- b) lista simplesmente encadeada e ordenada
- c) lista duplamente encadeada e não ordenada



- d) lista duplamente encadeada e não ordenada
- e) lista triplamente encadeada

Comentários:

O enunciado fala em ponteiro para o próximo elemento, e não cita ponteiro para o elemento interior. Portanto, dá para inferir que se trata de uma lista simplesmente encadeada.

O enunciado também não fala em ordenação. Portanto, seria uma lista simplesmente encadeada e não ordenada.

Gabarito: Letra A

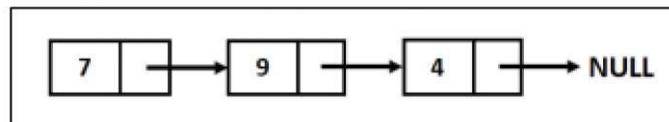
3. (FUNDATÉC – 2022 – IF-RS) Que tipo de estrutura de dados está representada na Figura 1 abaixo?

Figura 1 – Estrutura de dados

- a) Árvore binária.
- b) Fila.
- c) Pilha.
- d) Lista ligada.
- e) Vetor.

Comentários:

Note que cada elemento possui um ponteiro para o próximo, exceto o último, que aponta para “null”. Trata-se de uma lista ligada.

Como a questão não fala sobre regras de inserção e remoção (LIFO ou FIFO), não pode ser fila ou pilha. Como é uma estrutura sequencial, não pode ser árvore, que é uma estrutura hierárquica.

Para ser um vetor, deveria ter seus elementos em posições de memória sequenciais, e não utilizando ponteiros.

Portanto, é uma lista ligada.

Gabarito: Letra D

4. (IBADE – 2022 – SES-MG) Uma estrutura de dados onde existe uma coleção ordenada de entidades sendo a metodologia de busca com base no deslocamento relativo ao primeiro (cabeça) da coleção, chama-se:

- a) árvore.
- b) lista.
- c) pilha.
- d) fila.
- e) árvore binária.

Comentários:

Vamos analisar item a item:

- a) árvore.

Em uma árvore, o deslocamento é feito a partir do nó raiz, e não da cabeça. Falso.

- b) lista.

Em uma lista, temos a referência ao primeiro elemento, que é chamado de cabeça. Depois, cada elemento tem um apontamento para o próximo. Dessa forma, para buscar, temos que, de fato, ir da cabeça e seguir aos seguintes. Verdadeiro.

- c) pilha.



Em uma pilha, temos o topo da pilha. Neste caso, temos apenas 3 operações: top, para ver qual elemento está no topo da pilha; pop, para remover o elemento do topo da pilha; e push, para incluir um elemento na pilha. Não há operação para deslocar sobre ela. Falso.

d) fila.

Em uma fila, temos duas extremidades, e duas operações: uma para inserir um elemento ao final, e outra pra retirar do início. Não é o caso de haver uma “cabeça” inicial para percorrer as demais. Falso.

e) árvore binária.

Em uma árvore binária, percorre-se do nó raiz, e não da cabeça. Falso.

Gabarito: Letra B

5. (FGV – 2021 – TJ-RO) Considere a lista duplamente encadeada exibida a seguir.

- (1, 3, 0, “Verde”)
- (2, 4, 3, “Azul”)
- (3, 2, 1, “Amarelo”)
- (4, 0, 2, “Vermelho”)

Cada elemento pertencente à lista é representado por uma quádrupla, com o seguinte formato:

(<id>, <id do anterior>, <id do seguinte>, <conteúdo>).

A ordem do conteúdo dos componentes, segundo a instância da lista apresentada, é:

- a) Amarelo, Verde, Azul, Vermelho;
- b) Azul, Verde, Vermelho, Amarelo;
- c) Verde, Vermelho, Amarelo, Azul;
- d) Vermelho, Amarelo, Azul, Verde;
- e) Vermelho, Azul, Amarelo, Verde.

Comentários:

Considerando que o primeiro número é o ID, temos os seguintes IDs:

| ID | Cor |
|----|----------|
| 1 | Verde |
| 2 | Azul |
| 3 | Amarelo |
| 4 | Vermelho |

De linha a linha, vamos analisando:

(1, 3, 0, “Verde”)

O verde vem depois do 3 (Amarelo), e antes do 0. Como não existe 0, então sabemos que:

Lista: Amarelo - Verde

(2, 4, 3, “Azul”)

O azul vem depois do 4 (Vermelho), e antes do 3 (Amarelo), então sabemos que:

Lista: Vermelho - Azul – Amarelo – Verde

Apenas com essas duas linhas, já sabemos a ordem que é a letra D, certo? Mas... vamos conferir as demais.

(3, 2, 1, “Amarelo”)

Diz que o Amarelo vem depois do 2 (Azul) e antes do 1 (Verde). Está certo.

(4, 0, 2, “Vermelho”)

Diz que o Vermelho vem depois do 0 (não existe, portanto, é o primeiro), e depois do 2 (Azul). Certo também.

A lista é Vermelho - Azul – Amarelo – Verde.

Gabarito: Letra E

6. CESPE/CEBRASPE – 2019 – MPC-PA) Assinale a opção que apresenta a denominacão da estrutura de



que o sucede.

- a) lista circular
- b) grafo
- c) lista duplamente encadeada
- d) árvore
- e) pilha

Comentários:

Vamos analisar item a item:

- a) lista circular

Uma lista em que cada elemento referencia o outro, porém com uma diferença: o último elemento referencia o primeiro também. A questão deixa claro que quer uma estrutura de dados em que os elementos inicial e final não estejam ligados. Falso.

- b) Grafo

Estrutura de dados com um conjunto de vértices ligados entre si por arestas. Falso.

- c) lista duplamente encadeada

Uma lista em que cada elemento referencia o outro em ambas as direções: um elemento referencia o anterior e o próximo, exceto os elementos inicial e final. É isso aí. Verdadeiro.

- d) Árvore

Estrutura de dados em que um nó possui 1 ou mais filhos, mas cada filho possui somente um pai. Falso.

- e) pilha

Estrutura de dados usando a regra LIFO, last-in first-out, ou seja, o último a entrar é o primeiro a sair. Falso.

Gabarito: Letra C

7. (CESPE/CEBRASPE – 2017 – TRT-7) Considere uma estrutura de dados em que cada elemento armazenado apresenta ligações de apontamento com seu sucessor e com o seu predecessor, o que possibilita que ela seja percorrida em qualquer sentido. Trata-se de

- a) uma fila.
- b) um grafo.
- c) uma lista duplamente encadeada.
- d) uma pilha.

Comentários:

Vamos analisar item a item:

- a) uma fila.

Estrutura de dados que segue a lógica FIFO (first-in, first-out), ou “primeiro a entrar é o primeiro a sair”. Os elementos são inseridos no fim da fila e removidos no início. Falso.

- b) um grafo.

Grafo é uma estrutura de dados com um conjunto de vértices e arestas que os ligam. Falso.

- c) uma lista duplamente encadeada.

É uma estrutura de dados que armazena uma sequência de elementos que possuem ligações de apontamento, ou ponteiros, entre si, em ambas as direções. Isso permite percorrer a lista em qualquer sentido. Verdadeiro.

- d) uma pilha.

Estrutura de dados que segue a lógica LIFO, ou last-in first-out, que significa “o último a entrar é o primeiro a sair”. Falso.

Gabarito: Letra C



Comentários: Este é o conceito de lista duplamente encadeada, e não da lista encadeada. No caso, uma lista encadeada possui um endereço apenas para o nó posterior, e não para o anterior.

Gabarito: Errado

9. (FGV – 2018 – MPE-AL) Considere a representação de uma lista duplamente encadeada que armazena os times de futebol que participam de um torneio.

| Nó | Time | Anterior | Posterior |
|----|-------------|----------|-----------|
| 1 | Real Madrid | 4 | 2 |
| 2 | Roma | 1 | |
| 3 | Barcelona | | 5 |
| 4 | Bayern | 5 | 1 |
| 5 | Chelsea | 3 | 4 |

Assinale a ordem em que os times estão dispostos nessa lista.

- a) Barcelona, Chelsea, Bayern, Real Madrid, Roma.
- b) Chelsea, Bayern, Real Madrid, Roma, Barcelona.
- c) Real Madrid, Roma, Barcelona, Chelsea, Bayern.
- d) Barcelona, Bayern, Chelsea, Real Madrid, Roma
- e) Roma, Real Madrid, Bayern, Chelsea, Barcelona.

Comentários:

Vamos analisar linha a linha.

Real Madrid está depois do 4 (Bayern) e antes do 2 (Roma).

Lista: Bayern – Real Madrid – Roma

Roma está depois do 1 (Real Madrid), e não tem posterior, logo, é o último da lista.

Barcelona não tem anterior (logo, é o primeiro da lista), e está antes do 5 (Chelsea).

Lista: Barcelona – Chelsea – ??? – Bayern – Real Madrid – Roma

Bayern está depois do 5 (Chelsea) e antes do 1 (Real Madrid)

Lista: Barcelona – Chelsea – Bayern – Real Madrid – Roma

Já temos a resposta.

Gabarito: Letra A

10. (CESPE/CEBRASPE – 2018 – BNB)

Uma lista encadeada é basicamente uma estrutura de dados em lista em que cada nó possui três campos: um para os dados, um para o endereço do nó anterior, e outro para o endereço do nó posterior.

Comentários:

Este é o conceito de lista duplamente encadeada, e não da lista encadeada. No caso, uma lista encadeada possui um endereço apenas para o nó posterior, e não para o anterior.

Gabarito: Errado

11. (FCC – 2013 – DPE-SP) Em uma I, para cada novo elemento inserido na estrutura, alocamos um espaço de memória para armazená-lo. Desta forma, o espaço total de memória gasto pela estrutura é proporcional ao número de elementos nela armazenados. No entanto, não podemos garantir que os



direto aos elementos da lista. Para que seja possível percorrer todos os elementos da III , devemos explicitamente guardar o encadeamento dos elementos, o que é feito armazenando-se, junto com a informação de cada elemento, um IV para o próximo elemento da V .

As lacunas de I a V, são preenchidas, corretas e respectivamente, por:

- a) estrutura de pilha - tamanho - memória - array - pilha
- b) lista encadeada - memória - lista - ponteiro - lista
- c) estrutura de filas (FIFO) - disco - sequência - buffer - memória alocada
- d) arquitetura de memória primária - tamanho - fila - contador sequencial - conexão
- e) arquitetura TCP/IP - tamanho fixo - conexão - número de roteamento - tabela MTU

Comentários:

Podemos perceber que o enunciado está falando de listas encadeadas, já que, para cada novo elemento, alocamos um espaço na memória para armazená-lo, e não necessariamente estes espaços são contíguos, de modo que são ligados entre si por ponteiros ou referências. Portanto, é letra B.

A letra a) fala de pilha, portanto está errada já no primeiro item. A letra c) fala de filas, e, portanto, também está errada. Os demais itens d) e e) não têm nada a ver com o assunto.

Gabarito: Letra B

12. (FCC – FAURGS – SES-RS) Qual é a afirmativa correta sobre estruturas de dados?

- a) Uma pilha armazena os dados em uma estrutura de dados do tipo árvore binária.
- b) Listas encadeadas são estruturas que encadeiam os elementos através de um ponteiro no qual todos os elementos, exceto o último, apontam para o seguinte.
- c) Em uma pilha, o primeiro elemento a ser inserido será o primeiro a ser retirado, ou seja, adicionam-se itens no fim e removem-se do início.
- d) Uma fila armazena os dados em uma estrutura de dados do tipo grafo.
- e) Em uma fila, o primeiro elemento a ser inserido será o último a ser retirado, ou seja, adicionam-se e removem- se itens no início.

Comentários:

Vamos analisar item a item:

- a) Uma pilha armazena os dados em uma estrutura de dados do tipo árvore binária.

Não. Uma pilha armazena dados em uma estrutura de dados do tipo lista, seguindo a regra LIFO (last-in first-out), ou seja, o último a entrar é o primeiro a sair. Falso.

- b) Listas encadeadas são estruturas que encadeiam os elementos através de um ponteiro no qual todos os elementos, exceto o último, apontam para o seguinte.

Definição correta de listas encadeadas simples. Verdadeiro.

- c) Em uma pilha, o primeiro elemento a ser inserido será o primeiro a ser retirado, ou seja, adicionam-se itens no fim e removem-se do início.

Na verdade, uma pilha armazena dados em uma estrutura de dados do tipo lista, seguindo a regra LIFO (last-in first-out), ou seja, o último a entrar é o primeiro a sair. Falso.

- d) Uma fila armazena os dados em uma estrutura de dados do tipo grafo.

Não. Uma fila armazena dados em uma estrutura de dados do tipo lista, seguindo a regra FIFO (first-in last-out), ou seja, o primeiro a entrar é o primeiro a sair. Falso.

- e) Em uma fila, o primeiro elemento a ser inserido será o último a ser retirado, ou seja, adicionam-se e removem- se itens no início. Não. A regra da fila é a FIFO (first-in last-out), ou seja, o primeiro a entrar é o primeiro a sair. Falso.

Letra B

109

344



13. (FCC – 2013 – DPE-SP) Em uma I, para cada novo elemento inserido na estrutura, alocamos um espaço de memória para armazená-lo. Desta forma, o espaço total de memória gasto pela estrutura é proporcional ao número de elementos nela armazenados. No entanto, não podemos garantir que os elementos armazenados na lista ocuparão um espaço de II contíguo, portanto, não temos acesso direto aos elementos da lista. Para que seja possível percorrer todos os elementos da III, devemos explicitamente guardar o encadeamento dos elementos, o que é feito armazenando-se, junto com a informação de cada elemento, um IV para o próximo elemento da V.

As lacunas de I a V, são preenchidas, corretas e respectivamente, por:

- a) estrutura de pilha - tamanho - memória - array - pilha
- b) lista encadeada - memória - lista - ponteiro - lista
- c) estrutura de filas (FIFO) - disco - sequência - buffer - memória alocada
- d) arquitetura de memória primária - tamanho - fila - contador sequencial - conexão
- e) arquitetura TCP/IP - tamanho fixo - conexão - número de roteamento - tabela MTU

Comentários:

Podemos perceber que o enunciado está falando de listas encadeadas, já que, para cada novo elemento, alocamos um espaço na memória para armazená-lo, e não necessariamente estes espaços são contíguos, de modo que são ligados entre si por ponteiros ou referências. Portanto, é letra B.

A letra a) fala de pilha, portanto está errada já no primeiro item. A letra c) fala de filas, e, portanto, também está errada. Os demais itens d) e e) não têm nada a ver com o assunto.

Gabarito: Letra B

14. (CESPE - 2012 - Banco da Amazônia - Técnico Científico - Administração de Dados) O tempo de busca de um elemento em uma lista duplamente encadeada é igual à metade do tempo da busca de um elemento em uma lista simplesmente encadeada.

Comentários:

Não! Apesar de permitir que se percorra a lista em ambas as direções, em média ambas possuem o mesmo tempo de busca de um elemento.

Gabarito: Errado

15. (CESPE - 2012 - Banco da Amazônia - Técnico Científico - Administração de Dados) Em algumas implementações, uma lista vazia pode ter um único nó, chamado de sentinel, nó cabeça ou header. Entre suas possíveis funções, inclui-se simplificar a implementação de algumas operações realizadas sobre a lista, como inserir novos dados, recuperar o tamanho da lista, entre outras.

Comentários:

Perfeito! Ele simplifica a implementação de algumas operações porque se guarda o endereço do primeiro e do último elemento de uma estrutura de dados de modo que o programador não precisa conhecer a estrutura de implementação da lista para realizar suas operações. **Gabarito: C**

Gabarito: Correto



inicial.

Comentários:

Perfeito! Listas Encadeadas admitem alocação dinâmica, em contraste com as matrizes.

Gabarito: Correto

17. (CESPE - 2012 - Banco da Amazônia - Técnico Científico - Administração de Dados) As listas duplamente encadeadas diferenciam-se das listas simplesmente encadeadas pelo fato de, na primeira, os nós da lista formarem um anel com o último elemento ligado ao primeiro da lista.

Comentários:

Não, a diferença é que Listas Duplamente Encadeadas possuem dois ponteiros, que apontam para o nó sucessor e para o nó predecessor e Listas Simplesmente Encadeadas possuem apenas um ponteiro, que aponta para o nó sucessor.

Gabarito: Errado

18. (FCC - 2012 - TRE-SP - Analista Judiciário - Análise de Sistemas - E) Numa lista singularmente encadeada, para acessar o último nodo é necessário partir do primeiro e ir seguindo os campos de ligação até chegar ao final da lista.

Comentários:

Perfeito! Se é uma lista singularmente encadeada, é necessário percorrer cada elemento um-a-um até chegar ao final da lista.

Gabarito: Correto

19. (CESPE - 2011 - EBC - Analista - Engenharia de Software) Uma lista é uma coleção de elementos do mesmo tipo dispostos linearmente, que podem ou não seguir determinada organização. As listas podem ser dos seguintes tipos: de encadeamento simples, duplamente encadeadas e ordenadas.

Comentários:

Uma lista é, por natureza, heterogênea, i.e., seus elementos são compostos por tipos de dados primitivos diferentes. A questão afirmou que a lista é uma coleção de elementos do mesmo tipo. E ela está certa, veja só o pinguinha da questão:

Eu crio um tipo composto por dois tipos de dados diferentes: tipoEstrategia = {curso: caractere; duracao: inteiro}. Observe que o tipo tipoEstrategia é composto por tipos de dados primitivos diferentes (caractere e inteiro). Agora eu crio uma lista ListaEstrategia: tipoEstrategia. Veja que todos os elementos dessa lista terão o mesmo tipo (tipoEstrategia). Em outras palavras: a Lista Heterogênea é composta por elementos do mesmo tipo que, em geral, são compostos por mais de um tipo primitivo.

Além disso, as listas podem ser simplesmente encadeadas, duplamente encadeadas e ordenadas. E ainda podem ser circulares. Observem que alguns autores consideram Listas Ordenadas como um tipo de lista!



20. (CESPE - 2009 - ANAC - Técnico Administrativo - Informática) Em uma lista circular duplamente encadeada, cada nó aponta para dois outros nós da lista, um anterior e um posterior.

Comentários:

Perfeito! Há dois ponteiros: uma para o nó anterior e um para o nó posterior.

Gabarito: Correto

21. (CESPE - 2008 - TRT - 5ª Região (BA) - Técnico Judiciário - Tecnologia da Informação) A principal característica de uma lista encadeada é o fato de o último elemento da lista apontar para o elemento imediatamente anterior.

Comentários:

Não, o último elemento da lista não aponta para nenhum outro nó em uma lista não-circular.

Gabarito: Errado

22. (CESPE - 2009 - TCE-AC - Analista de Controle Externo - Processamentos de Dados) Uma lista encadeada é uma coleção de nodos que, juntos, formam uma ordem linear. Se é possível os nodos se deslocarem em ambas as direções na lista, diz-se que se trata de uma lista simplesmente encadeada.

Comentários:

Se é possível os nodos se deslocarem em ambas as direções na lista, diz-se que se trata de uma lista duplamente encadeada.

Gabarito: Errado



23. (CESPE - 2008 – HEMOBRÁS – Técnico de Informática) Uma estrutura do tipo lista, em que é desejável percorrer o seu conteúdo nas duas direções indiferentemente, é denominado lista duplamente encadeada.

Comentários:

Perfeito, é exatamente isso!

Gabarito: Correto

24. (CESPE - 2010 – TRE/MT – Analista de Sistemas – C) Uma lista duplamente encadeada é uma lista em que o seu último elemento referencia o primeiro.

Comentários:

Não, isso se trata de uma Lista Circular!

Gabarito: Errado

25. (CESPE - 2006 – SGA/AC – Analista de Sistemas) O principal problema da alocação por lista encadeada é a fragmentação.

Comentários:

*Não! Em geral, a alocação por lista encadeada elimina a fragmentação.

Gabarito: Errado

26. (CESPE - 2008 – MCT – Analista de Sistemas) O armazenamento de arquivos em disco pode ser feito por meio de uma lista encadeada, em que os blocos de disco são ligados por ponteiros. A utilização de lista encadeada elimina completamente o problema de fragmentação interna.

Comentários:

Não, ela elimina a fragmentação externa!

Gabarito: Errado

27. (CESPE - 2009 – FINEP – Analista de Sistemas) Uma lista encadeada é uma representação de objetos na memória do computador que consiste de uma sequência de células em que:

- a) cada célula contém apenas o endereço da célula seguinte.
- b) cada célula contém um objeto e o tipo de dados da célula seguinte.
- c) o último elemento da sequência aponta para o próximo objeto que normalmente possui o endereço físico como null.
- d) cada célula contém um objeto de algum tipo e o endereço da célula seguinte.
- e) a primeira célula contém o endereço da última célula.



Comentários:

Cada célula contém um objeto de algum tipo e o endereço da célula seguinte!

Gabarito: Letra D

28. (CESPE - 2010 – BASA – Analista de Sistemas) Em uma lista encadeada, o tempo de acesso a qualquer um de seus elementos é constante e independente do tamanho da estrutura de dados.

Comentários:

Claro que não! Em uma busca sequencial, o tempo de acesso é proporcional ao tamanho da estrutura de dados, i.e., quanto mais ao final da lista, maior o tempo de acesso! Por que, professor? Porque a lista é acessada sequencialmente (ou seja, é preciso percorrer elemento por elemento) e, não, diretamente (ou seja, pode-se acessar de modo direto). Um vetor tem acesso direto, portanto seu tempo de acesso é igual independentemente do tamanho da estrutura.

Gabarito: Errado

29. (CESPE - 2010 – INMETRO – Analista de Sistemas – C) Considere que Roberto tenha feito uso de uma lista encadeada simples para programar o armazenamento e o posterior acesso aos dados acerca dos equipamentos instalados em sua empresa. Considere, ainda, que, após realizar uma consulta acerca do equipamento X, Roberto precisou acessar outro equipamento Y que se encontrava, nessa lista, em posição anterior ao equipamento X. Nessa situação, pela forma como os ponteiros são implementados em uma lista encadeada simples, o algoritmo usado por Roberto realizou a consulta ao equipamento Y sem reiniciar a pesquisa do começo da lista.

Comentários:

Não! Infelizmente, ele teve que reiniciar a pesquisa a partir do primeiro elemento da lista, na medida em que ele não pode voltar. Por que, professor? Porque se trata de uma lista encadeada simples e, não, dupla. Portanto, a lista só é percorrida em uma única direção.

Gabarito: Errado

30. (FCC - 2003 – TRE/AM – Analista de Sistemas) Os dados contidos em uma lista encadeada estão:

- a) ordenados seqüencialmente.
- b) sem ordem lógica ou física alguma.
- c) em ordem física e não, necessariamente, em ordem lógica.
- d) em ordem lógica e, necessariamente, em ordem física.
- e) em ordem lógica e não, necessariamente, em ordem física.

Comentários:

A Ordem Física é sua disposição na memória do computador e a Ordem Lógica é como ela pode ser lida e entendida. Ora, a ordem em que ela se encontra na memória pouco importa, visto que cada sistema operacional e cada sistema de



31. (FCC - 2010 – DPE/SP – Analista de Sistemas) Uma estrutura de dados que possui três campos: dois ponteiros e campo de informação denomina-se:

- a) lista encadeada dupla.
- b) Lista encadeada simples.
- c) pilha.
- d) fila.
- e) vetor.

Comentários:

Trata-se da Lista Encadeada Dupla: dois ponteiros (Ant e Prox) e um campo de informação.

Gabarito: Letra A

32. (CESPE - 2010 – TRE/MT – Analista de Sistemas) O algoritmo para inclusão de elementos em uma pilha é usado sem nenhuma alteração para incluir elementos em uma lista.

Comentários:

Galera... uma pilha pode ser implementada por meio de uma lista! Ademais, o algoritmo para inclusão de elementos de ambas necessita do primeiro elemento (ou topo). Portanto, questão correta! **Gabarito: C**

Gabarito: Correto



QUESTÕES COMENTADAS - PILHAS - MULTIBANCAS

1. (FGV – 2022 – PC-AM) Assinale as operações características de uma estrutura de dados do tipo pilha (stack).

- a) IMPORT, EXPORT.
- b) INPUT, OUPUT.
- c) INSERT, REMOVE.
- d) PUSH, POP.
- e) READ, READLN.

Comentários:

Push e pop são duas operações de uma pilha:

- Push: inserir um elemento no topo da pilha;
- Pop: remover o último elemento inserido na pilha.

Quanto aos demais itens:

- IMPORT, EXPORT: operações de arquivos.
- INPUT, OUPUT: entrada e saída de dados. Não tem a ver com pilhas.
- INSERT, REMOVE: inserir e remover. Embora na pilha tenhamos inserção e remoção, estas operações, com este nome, estão associadas a listas.
- READ, READLN: operações para a entrada de dados.

Gabarito: Letra D

2. (FGV – 2022 – TJDFT) Júlio está desenvolvendo uma aplicação e precisa implementar um mecanismo de desfazer/refazer de um editor de texto utilizando o algoritmo LIFO (Last In, First Out).

Para implementar o algoritmo LIFO, Júlio deve usar a estrutura de dados:

- a) fila;
- b) pilha;
- c) árvore;
- d) nó folha;
- e) tabela hash.

Comentários:

Vamos analisar item a item:

- a) fila;

A fila segue a regra FIFO (first-in first-out), ou seja, o primeiro a entrar é o primeiro a sair. Falso.

- b) pilha;

A pilha segue a regra LIFO (last-in first-out), ou seja, o último a entrar é o primeiro a sair. A operação de inserir um valor é o push, e a operação de remover o último inserido é o pop. Verdadeiro.

- c) árvore;

Árvores não seguem necessariamente regras de inserção, além de não serem estruturas sequenciais. Falso.

- d) nó folha;

Nó folha é o nó de uma árvore que não possui filhos. Não tem nada a ver com o enunciado. Falso.



e) tabela hash.

Tabelas hash não seguem regra LIFO. Falso.

Gabarito: Letra B

3. (CESPE/CEBRASPE – 2022 – DPE-RO) Em um sistema operacional, a estrutura de dados utilizada para organizar chamadas de funções recursivas por meio da inserção ou remoção de elementos via operações como push e pop é denominada

- a) lista estática.
- b) fila.
- c) hash.
- d) pilha.
- e) lista dinâmica.

Comentários:

Push e pop são duas operações de uma pilha:

- Push: inserir um elemento no topo da pilha;
- Pop: remover o último elemento inserido na pilha.

A pilha leva em consideração o padrão LIFO, ou seja, last-in first-out, ou seja, o último a entrar é o primeiro a sair.

Por isso, suas duas operações são push e pop.

Quanto aos demais itens:

- lista estática: uma lista em que o número de elementos é fixo e pré-alocado.
- fila: segue o padrão FIFO, ou seja, first-in first-out, ou seja, o primeiro a entrar é o primeiro a sair.
- Hash: uma estrutura de dados para inserção, busca e remoção de forma rápida.
- lista dinâmica: uma lista em que o número de elementos é alterado dinamicamente enquanto o programa executa.

Gabarito: Letra D

4. (UFV – 2022 – UFV-MG) Considere as afirmativas a seguir sobre estrutura de dados:

- I. Uma estrutura de dados heterogênea envolve a utilização de mais de um tipo básico de dado.
- II. Uma lista encadeada pode ser definida como uma sequência de células em que cada célula contém um elemento e o endereço da célula seguinte.
- III. Uma pilha é uma estrutura de dados baseada no princípio “First In First Out” (FIFO).
- IV. Filas e pilhas são estruturas de dados lineares; o organograma de uma empresa pode ser representado por uma estrutura de árvore.

Está CORRETO o que se afirma, apenas, em:

- a) I e II.
- b) I e III.
- c) I, II e IV.
- d) II, III e IV.

Comentários:

Vamos analisar item a item:

- I. Uma estrutura de dados heterogênea envolve a utilização de mais de um tipo básico de dado.

Isso mesmo. Significa que cada elemento da estrutura de dados pode ter mais de um tipo básico de dado. Por exemplo, um inteiro, uma string, um booleano, etc. Verdadeiro.



- II. Uma lista encadeada pode ser definida como uma sequência de células em que cada célula contém um elemento e o endereço da célula seguinte.

Essa é a definição de lista encadeada simples. Certo.

- III. Uma pilha é uma estrutura de dados baseada no princípio “First In First Out” (FIFO).

Na verdade, a pilha segue a estrutura de dados com a regra LIFO (last-in first-out), ou seja, o último a entrar é o primeiro a sair. Falso.

- IV. Filas e pilhas são estruturas de dados lineares; o organograma de uma empresa pode ser representado por uma estrutura de árvore.

Filas e pilhas são estruturas lineares de fato, e uma árvore é uma estrutura hierárquica, e, por isso, um organograma de uma empresa pode ser representado por ela. Verdadeiro.

Corretas: I, II e IV.

Gabarito: Letra C

5. (IBFC – 2022 – DPE-MT) Assinale a alternativa que apresenta a relação entre as duas estruturas de dados da coluna da esquerda com as respectivas características técnicas da coluna da direita.

| | |
|-----------|---|
| | (A) O elemento inserido por primeiro é o primeiro elemento a sair da lista. |
| (1) PILHA | (B) O elemento inserido por último é o primeiro elemento a sair da lista. |
| (2) FILA | (C) Precisa-se de apenas um ponteiro para acessar a lista. |
| | (D) Precisa-se de dois ponteiros para acessar a lista. |

Assinale a alternativa correta.

- a) 1BC - 2AD
- b) 1AD - 2BC
- c) 1BD - 2AC
- d) 1AC - 2BD

Comentários:

Uma pilha segue a regra LIFO (last-in first-out), ou seja, o elemento inserido por último é o primeiro elemento a sair da lista. Portanto, 1-B.

A pilha, também precisa de apenas um ponteiro, que aponta para o topo da pilha. Portanto, 1-BC.

Já a fila segue a regra FIFO (first-in first-out), ou seja, o elemento inserido primeiro é o primeiro a sair da lista. Portanto, 2-A.

A fila também precisa de dois ponteiros para ser acessada, já que suas operações envolvem adicionar sempre os elementos no final, e remover do início. Portanto, 2-AD.

Gabarito: Letra A

6. (FUNDATEC – 2022 – IPE Saúde) Uma sequência de valores é armazenada em uma estrutura de dados, onde novos elementos são inseridos no final da lista e removidos também do final da mesma. Dessa forma, qualquer elemento só pode ser removido quando todos os elementos inseridos após ele também forem removidos. Essa descrição caracteriza uma estrutura de dados conhecida como:

- a) Lista duplamente encadeada.
- b) Lista simplesmente encadeada.
- c) Fila.
- d) Pilha.



e) Árvore binária.

Comentários:

Precisamos considerar dois pontos importantes:

- Os elementos são apenas inseridos ao final da lista.
- Os elementos também são removidos apenas no fim da lista.
- Os elementos só podem ser removidos se todos os inseridos após ele tiverem sido removidos. Ou seja, o último a ser inserido é o primeiro a ser removido, ou seja, regra LIFO.

A estrutura de dado que segue a regra LIFO é a pilha.

Na pilha, temos a operação push, que é inserir um elemento no fim da lista; e a operação pop, que é remover o elemento do fim da lista. E temos a regra LIFO.

Analisemos as demais estruturas de dados:

- Lista duplamente encadeada: cada nó tem uma referência ao nó anterior e ao próximo, exceto o primeiro e o último. A inserção e remoção é livre.
- Lista simplesmente encadeada: cada nó tem uma referência ao próximo nó, exceto o último. A inserção e remoção é livre.
- Fila: segue o padrão FIFO (first-in first-out), ou seja, o primeiro a entrar é o primeiro a sair. Além disso, no caso da fila, os elementos são sempre adicionados no fim, e removidos do início da lista.
- Árvore binária: uma árvore binária possui nós com no máximo dois filhos. Não seguem as regras do enunciado.

Gabarito: Letra D

7. (FGV – 2021 – FUNSAÚDE-CE) As operações POP e PUSH aplicáveis às estruturas de dados são conhecidas como

- a) árvores binárias.
- b) bitmaps.
- c) hashtables.
- d) listas encadeadas.
- e) pilhas.

Comentários:

Quando se fala em operações pop e push, só podemos estar falando de pilhas.

As pilhas segrem a regra LIFO (last-in first-out), ou seja, o último a entrar é o primeiro a sair. Isso faz com que tenha duas operações:

- PUSH: inclui um elemento na pilha.
- POP: retira o último elemento incluído da pilha.

Gabarito: Letra E

8. (FGV – 2021 – IMBEL) No contexto das estruturas de dados, considere uma pilha (stack) onde as seguintes operações foram executadas.

CLEAR
PUSH (12)
PUSH (14)
POP
PUSH (20)



PUSH (15)

POP

PUSH (19)

Assinale a opção que indica o número de elementos e o valor do elemento localizado no topo da pilha, ao final das operações.

- a) 3 / 12
- b) 3 / 15
- c) 3 / 19
- d) 5 / 12
- e) 5 / 19

Comentários:

Precisamos definir os comandos.

- CLEAR: limpa a pilha.
- PUSH: inclui um elemento na pilha.
- POP: retira o último elemento incluído da pilha.

Então, vamos seguir linha a linha:

CLEAR

Limpa a pilha.

PUSH (12)

Inclui o número 12 na pilha.

Pilha: $(\text{base}) [12]^{(\text{topo})}$

PUSH (14)

Inclui o número 14 na pilha.

Pilha: $(\text{base}) [12, 14]^{(\text{topo})}$

POP

Remove o número 14 da pilha.

Pilha: $(\text{base}) [12]^{(\text{topo})}$

PUSH (20)

Inclui o número 20 na pilha.

Pilha: $(\text{base}) [12, 20]^{(\text{topo})}$

PUSH (15)

Inclui o número 15 na pilha.

Pilha: $(\text{base}) [12, 20, 15]^{(\text{topo})}$

POP

Remove o número 15 da pilha.

Pilha: $(\text{base}) [12, 20]^{(\text{topo})}$

PUSH (19)

Inclui o número 19 na pilha.

Pilha: $(\text{base}) [12, 20, 19]^{(\text{topo})}$

Portanto, a pilha ficou com 3 elementos, sendo que, em seu topo, está o número 19.

Portanto, 3 / 19.

Gabarito: Letra C

9. (CESPE/CEBRASPE – 2018 – ABIN) Julgue o item subsequente, relativo à lógica de programação.

Pilha é uma estrutura de dados em que o último elemento a ser inserido será o primeiro a ser retirado.

Comentários:



A pilha é uma estrutura de dados que usa o princípio LIFO (Last In, First Out), que significa justamente "último a entrar, primeiro a sair".

Gabarito: Certo

10. (FGV – 2018 – AL-RO) Considere uma pilha de latas de sardinhas na prateleira de um supermercado.

Assinale a estrutura de dados que mais se assemelha ao modo como essas latas são manuseadas.

- a) Array.
- b) Binary tree.
- c) Hashing.
- d) Linked list.
- e) Stack.

Comentários:

Essa é praticamente uma questão de inglês... Precisamos saber o que cada uma dessas estruturas significa:

- a) Array

Significa vetor. É uma sequência de elementos de um determinado tipo, em posições sequenciais de memória. Falso.

- b) Binary tree.

Árvore binária. É uma árvore em que cada nó pode ter, no máximo, 3 filhos. Falso.

- c) Hashing.

Técnica para mapear conjuntos de dados em um conjunto de índices de um array, permitindo rápida busca dos dados. Falso.

- d) Linked list.

Lista encadeada. Uma estrutura de dados em que cada elemento possui uma referência ao próximo elemento da lista. Falso.

- e) Stack.

Também conhecida como pilha. Veja só que o enunciado fala justamente sobre **pilha** de sardinhas! É uma estrutura em que vale a regra LIFO (last-in first-out), ou seja, o último a entrar é o primeiro a sair. Verdadeiro.

Gabarito: Letra E

11. (FGV – 2018 – MPE-AL) Considere as seguintes operações sobre uma estrutura de dados, inicialmente vazia, organizada na forma de pilhas (ou stack),

PUSH (10)

PUSH (2)

POP ()

POP ()

PUSH (6)

Assinale a opção que apresenta a lista de elementos armazenados na estrutura, após a execução das operações acima.

- a) 10, 2, 6

- b) 10, 2

- c) 2, 6

- d) 6

- e) 2

Comentários:



Precisamos definir os comandos.

- PUSH: inclui um elemento na pilha.
- POP: retira o último elemento incluído da pilha.

Então, vamos seguir linha a linha:

PUSH (10)

Inclui o número 10 na pilha.

Pilha: (base) [10] (topo)

PUSH (2)

Inclui o número 2 na pilha.

Pilha: (base) [10, 2] (topo)

POP ()

Remove o número 2 na pilha.

Pilha: (base) [10] (topo)

POP ()

Remove o número 10 na pilha.

Pilha: (base) [] (topo)

PUSH (6)

Insere o número 6 na pilha.

Pilha: (base) [6] (topo)

Ao fim, temos apenas o elemento 6.

Gabarito: Letra D

12. (CESPE - 2011 - FUB - Analista de Tecnologia da Informação - Específicos) As pilhas são listas encadeadas cujos elementos são retirados e acrescentados sempre ao final, enquanto as filas são listas encadeadas cujos elementos são retirados e acrescentados sempre no início.

Comentários:

Bem... o que é o final de uma Pilha? Pois é, não se sabe! O que existe é o Topo da Pilha, de onde sempre são retirados e acrescentados elementos. Em Filas, elementos são retirados do início e acrescentados no final.

Gabarito: Errado

13. (CESPE - 2013 - INPI - Analista de Planejamento - Desenvolvimento e Manutenção de Sistemas) Na estrutura de dados do tipo lista, todo elemento novo que é introduzido na pilha torna-se o elemento do topo.

Comentários:

Que questão confusa! Vamos comigo: vocês sabem muito bem que filas e pilhas são considerados espécies de listas. A questão inicialmente fala de uma lista, mas depois ela menciona uma pilha – podemos inferir, então, que se trata de uma lista do tipo pilha. Em uma pilha, todo elemento novo que é introduzido torna-se o elemento do topo, logo... questão correta!

Bem, esse foi o Gabarito Preliminar, mas a banca mudou de opinião e, no Gabarito Definitivo, permaneceu como errada. E a justificativa dela foi: “A ausência de especificação do tipo de lista no item torna correta a



informação nele apresentada, razão pela qual se opta pela alteração de seu gabarito". Vejam que bizarro: se torna correta a informação apresentada, o gabarito definitivo deveria ser C e, não, E.
Além disso, a questão informa em sua segunda parte que se trata de uma pilha. Logo, não há que se falar em "ausência de especificação do tipo de lista". Enfim, questão péssima, horrível e mal redigida :(

Gabarito: Errado

14. (CESPE - 2012 - TJ-RO - Analista Judiciário - Analista de Sistemas Suporte – E) Visitas a sítios armazenadas em um navegador na ordem last-in-first-out é um exemplo de lista.

Comentários:

Não! Last-In-First-Out (LIFO) é um exemplo de Pilha! Cuidado, pilhas podem ser implementadas como listas, mas esse não é o foco da questão.

Gabarito: Errado

15. (ESAF - 2013 - DNIT - Analista Administrativo - Tecnologia da Informação) Assinale a opção correta relativa às operações básicas suportadas por pilhas.

- a) Push: insere um novo elemento no final da pilha.
- b) Pop: adiciona elementos ao topo da pilha.
- c) Pull: insere um novo elemento no interior da pilha.
- d) Top: transfere o último elemento para o topo da pilha.
- e) Top: acessa o elemento posicionado no topo da pilha.

Comentários:

(a) Não, é no topo; (b) Não, remove do topo; (c) Não, não existe; (d) Não, simplesmente acessa e consulta o elemento do topo; (e) Perfeito! **Gabarito: E**

Gabarito: Letra E

16. (FCC - 2012 – TST - Analista de Sistemas – C) As pilhas e as filas são estruturas de dados essenciais para os sistemas computacionais. É correto afirmar que a pilha é conhecida como lista FIFO - First In First Out.

Comentários:

Não! Pilha é LIFO e Fila é FIFO.

Gabarito: Errado

17. (FCC - 2012 – TRE/CE - Analista de Sistemas) Sobre pilhas é correto afirmar:

- a) Uma lista LIFO (Last-In/First-Out) é uma estrutura estática, ou seja, é uma coleção que não pode aumentar e diminuir durante sua existência.
- b) Os elementos na pilha são sempre removidos na mesma ordem em que foram inseridos.



- c) Uma pilha suporta apenas duas operações básicas, tradicionalmente denominadas push (insere um novo elemento no topo da pilha) e pop (remove um elemento do topo da pilha).
- d) Cada vez que um novo elemento deve ser inserido na pilha, ele é colocado no seu topo e, em qualquer momento, apenas aquele posicionado no topo da pilha pode ser removido.
- e) Sendo P uma pilha e x um elemento qualquer, a operação Push(P,x) diminui o tamanho da pilha P, removendo o elemento x do seu topo.

Comentários:

(a) Não, é uma estrutura dinâmica; (b) Não, é na ordem inversa (último a entrar é o primeiro a sair); (c) Não, há também Top ou Check, que acessar e consulta o elemento do topo; (e) Push é a operação de inserção de novos elementos na pilha, portanto aumenta seu tamanho adicionando o elemento x no topo.

E a letra D? Vamos lá! Sabemos que uma pilha é um tipo de lista - o que muda é apenas a perspectiva. Como assim? Eu vejo um monte de elementos em sequência. Ora, se eu coloco uma regra em que o primeiro elemento a entrar é o primeiro a sair, chamamos essa lista de fila; se eu coloco uma regra em que o primeiro elemento a entrar é o último a sair, chamamos essa lista de pilha.

Ok! Dito isso, o algoritmo é exatamente o mesmo, eu vou simplesmente mudar a perspectiva e a minha visão sobre a estrutura. Bacana?

Gabarito: Letra D

18. (CESPE - 2011 - EBC - Analista - Engenharia de Software) As pilhas, também conhecidas como listas LIFO ou PEPS, são listas lineares em que todas as operações de inserção e remoção de elementos são feitas por um único extremo da lista, denominado topo.

Comentários:

Não! LIFO é similar a UEPS (Último a Entrar, Primeiro a Sair). PEPS refere-se a Primeiro a Entrar, Primeiro a Sair, ou seja, FIFO.

Gabarito: Errado

19. (VUNESP - 2011 - TJM-SP - Analista de Sistemas - Judiciário) Lista do tipo LIFO (Last in, First Out) e lista do tipo FIFO (Firstin,First Out) são, respectivamente, características das estruturas de dados denominadas:

- a) Fila e Pilha.
- b) Pilha e Fila.
- c) Grafo e Árvore.
- d) Árvore e Grafo.
- e) Árvore Binária e Árvore Ternária.

Comentários:

E aí, já está automático para responder? Tem que ser automática: Pilha (LIFO) e Fila (FIFO).

Gabarito: Letra B



20. (CESPE - 2010 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação - Arquitetura de Tecnologia) A definição da estrutura pilha permite a inserção e a eliminação de itens, de modo que uma pilha é um objeto dinâmico, cujo tamanho pode variar constantemente.

Comentários:

Essa questão é polêmica, porque é inevitável pensar em Pilhas Sequenciais (implementadas por vetores estáticos)! No entanto, é comum que as bancas tratem por padrão Pilha como Pilha Encadeada (implementadas por listas dinâmicas). Dessa forma, a questão está perfeita!

Gabarito: Correto

21. (CESPE - 2010 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação - Administração de Dados) Na representação física de uma pilha sequencial, é necessário uso de uma variável ponteiro externa que indique a extremidade da lista linear onde ocorrem as operações de inserção e retirada de nós.

Comentários:

As Pilhas oferecem três operações básicas: push, que insere um novo elemento no topo da pilha; pop, que remove um elemento do topo da pilha; e top (também conhecida como check), que acessa e consulta o elemento do topo da pilha. Pilhas podem ser implementadas por meio de Vetores (Pilha Sequencial - Alocação Estática de Memória) ou Listas (Pilha Encadeada - Alocação Dinâmica de Memória).

Conforme vimos em aula, a questão trata de uma Pilha Sequencial (i.e., implementada por meio de Vetores). Dessa forma, não é necessário o uso de ponteiros – esse seria o caso de uma Pilha Encadeada. Eu posso realmente dizer que é suficiente, mas não posso afirmar que é necessária a utilização de um ponteiro externo. Eu até poderia dizer que é necessário o uso de um indicador, mas ele também não necessariamente será um ponteiro. Logo, discordo do gabarito!

Gabarito: Correto

22. (CESPE - 2009 - ANAC - Técnico Administrativo - Informática) As operações de inserir e retirar sempre afetam a base de uma pilha.

Comentários:

Não, sempre afetam o topo da pilha!

Gabarito: Errado

23. (FCC - 2009 - TRT - 16ª REGIÃO (MA) - Técnico Judiciário - Tecnologia da Informação) Pilha é uma estrutura de dados:

- a) cujo acesso aos seus elementos segue tanto a lógica LIFO quanto a FIFO.
- b) cujo acesso aos seus elementos ocorre de forma aleatória.
- c) que pode ser implementada somente por meio de vetores.
- d) que pode ser implementada somente por meio de listas.
- e) cujo acesso aos seus elementos segue a lógica LIFO, apenas.



Comentários:

(a) Não, somente LIFO; (b) Não, somente pelo Topo; (c) Não, pode ser por listas; (d) Não, pode ser por vetores; (e) Perfeito, é exatamente isso.

Gabarito: Errado

24. (CESPE - 2004 – STJ – Analista de Sistemas) Em geral, em uma pilha só se admite ter acesso ao elemento localizado em seu topo. Isso se adapta perfeitamente à característica das seqüências em que só o primeiro componente é diretamente acessível.

Comentários:

Perfeito, é exatamente isso – muda-se apenas a perspectiva!

Gabarito: Correto

25. (CESPE - 2004 – STJ – Analista de Sistemas) A seguir, está representada corretamente uma operação de desempilhamento em uma pilha de nome p.

se $p.\text{topo} = 0$ então
 nada {pilha vazia}
senão $p.\text{topo} \leftarrow p.\text{topo}-1$

Comentários:

Galera, a questão deu uma vaciladinho! O ideal seria dizer $p.\text{topo} = \text{null}$, mas vamos subentender que foi isso mesmo que ele quis dizer. Desse modo, se não tem topo, é porque a pilha está vazia. Se tiver topo, então o topo será o elemento anterior ao topo. O que ocorreu? Eu desempilhei a pilha!

Gabarito: Correto

26. (CESPE - 2010 – TRE/MT - Analista de Sistemas – A) O tipo nó é inadequado para implementar estruturas de dados do tipo pilha.

Comentários:

Não! Uma pilha pode ser implementada por meio de um vetor ou de uma lista. Nesse último caso, temos tipos nós!

Gabarito: Errado

27. (FGV – 2015 – DPE/MT – Analista de Sistemas) Assinale a opção que apresenta a estrutura de dados na qual o primeiro elemento inserido é o último a ser removido.

- a) Árvore
- b) Fila
- c) Pilha
- d) Grafo



Comentários:

Também conhecida como Lista LIFO (Last In First Out), basta lembrar de uma pilha de pratos esperando para serem lavados, i.e., o último a entrar é o primeiro a sair. A ordem em que os pratos são retirados da pilha é o oposto da ordem em que eles são colocados sobre a pilha e, como consequência, apenas o prato do topo da pilha está acessível.

Conforme vimos em aula, trata-se da Pilha. **Gabarito: C**

Gabarito: Correto

28. (FCC – 2012 – MPE/AP – Técnico Ministerial - Informática) Nas estruturas de dados,

- a) devido às características das operações da fila, o primeiro elemento a ser inserido será o último a ser retirado. Estruturas desse tipo são conhecidas como LIFO.
- b) as pilhas são utilizadas para controlar o acesso de arquivos que concorrem a uma única impressora.
- c) a fila é uma lista linear na qual as operações de inserção e retirada ocorrem apenas no início da lista.
- d) a pilha é uma lista linear na qual as operações de inserção e retirada são efetuadas apenas no seu topo.
- e) devido às características das operações da pilha, o último elemento a ser inserido será o último a ser retirado. Estruturas desse tipo são conhecidas como FIFO.

Comentários:

(a) as filas não são LIFO, mas sim FIFO, ou seja, o primeiro elemento da fila será, na verdade, o primeiro a ser retirado. Só pensarmos numa fila de banco, se alguém chega por último e é atendido primeiro, ficaria bem bravo, e vocês?? :D Item errado; (b) os trabalhos que chegam a uma impressora devem ser do tipo FIFO, ou seja, o primeiro trabalho enviado deve ser o primeiro a ser impresso. Item errado; (c) na fila os elementos são incluídos numa das extremidades e retirados da outra. Item errado; (d) na pilha as operações de inclusão na pilha quanto de retirada acontecem numa mesma extremidade. A extremidade escolhida é o topo da pilha. Item certo; (e) na verdade essas características são das filas. Item errado.

Gabarito: Letra D



QUESTÕES COMENTADAS - FILAS - MULTIBANCAS

1. (IBFC – 2022 – AFEAM) Assinale, das alternativas abaixo, a única que identifica respectivamente uma Estrutura de Dados do tipo FIFO (First In, First Out) e uma outra com a Estrutura de dados do tipo LIFO (Last In, First Out):

- a) lista – vetor
- b) pilha – fila
- c) vetor – lista
- d) fila - pilha

Comentários:

Uma estrutura que segue a regra de dados FIFO (first-in first-out), ou seja, o primeiro a entrar é o primeiro a sair, é a fila.

Já a estrutura que segue a regra de dados LIFO (last-in first-out), ou seja, o último a entrar é o primeiro a sair, é uma pilha.

Gabarito: Letra D

2. (IF-T0 – 2022 – IF-T0) Em estrutura de dados os conceitos de FILAS e PILHAS são usados para implementar diversos recursos computacionais que vão desde compiladores e interpretadores a mecanismos usados nas linguagens de programação para auxiliar os desenvolvedores no dia a dia.

Sobre essas estruturas, quais das definições abaixo são corretas?

- a) Nas FILAS é usado o princípio do primeiro a entrar é o último a sair, já as PILHAS obedecem a regra do primeiro a entrar é o último a sair.
- b) Nas FILAS é usado o princípio do primeiro a entrar é o primeiro a sair, já as PILHAS obedecem a regra do primeiro a entrar é o primeiro a sair.
- c) Nas FILAS é usado o princípio do segundo a entrar é o primeiro a sair, já as PILHAS obedecem a regra do último a entrar é o último a sair.
- d) Nas FILAS é usado o princípio do primeiro a entrar é o primeiro a sair, já as PILHAS obedecem a regra do primeiro a entrar é o último a sair.
- e) Nas FILAS é usado o princípio do primeiro a entrar é o segundo a sair, já as PILHAS obedecem a regra do segundo a entrar é o terceiro a sair.

Comentários:

As filas seguem o padrão FIFO (first-in first-out), ou seja, o primeiro a entrar é o primeiro a sair. Já as pilhas seguem o padrão LIFO (last-in first-out), ou seja, o último a entrar é o primeiro a sair.

Gabarito: Letra D

3. (UFRPE – 2022 – UFRPE) Sobre algoritmos e estrutura de dados, assinale a afirmativa correta.

- a) Listas encadeadas ou ligadas são estruturas de dados estáticas, o que significa que o número de nós não pode ser modificado durante a execução do programa.
- b) Pilhas são estruturas de dados do tipo FIFO (first-in first-out), em que o primeiro elemento a ser inserido será o primeiro a ser retirado.



- c) Árvores são estruturas de dados do tipo FIFO (first-in first-out), em que o primeiro elemento a ser inserido será o primeiro a ser retirado.
- d) Filas podem ser implementadas em listas encadeadas ou em vetores.
- e) Pilhas só podem ser implementadas em listas encadeadas.

Comentários:

Vamos analisar item a item:

- a) Listas encadeadas ou ligadas são estruturas de dados estáticas, o que significa que o número de nós não pode ser modificado durante a execução do programa.

Não são estáticas, mas, sim, dinâmicas: havendo memória livre, podem ser expandidas ou reduzidas conforme a necessidade. Falso.

- b) Pilhas são estruturas de dados do tipo FIFO (first-in first-out), em que o primeiro elemento a ser inserido será o primeiro a ser retirado.

Não, pilhas são LIFO (last-in first-out), ou seja, o último a entrar é o primeiro a sair. Falso.

- c) Árvores são estruturas de dados do tipo FIFO (first-in first-out), em que o primeiro elemento a ser inserido será o primeiro a ser retirado.

Árvores são estruturas hierárquicas, e não sequenciais, portanto, não podem seguir a regra FIFO. Falso.

- d) Filas podem ser implementadas em listas encadeadas ou em vetores.

De fato, filas podem ser implementadas usando listas encadeadas ou vetores. O importante é que siga a regra FIFO (first-in first-out), ou seja, o primeiro a entrar é o primeiro a sair. Verdadeiro.

- e) Pilhas só podem ser implementadas em listas encadeadas.

Não, é possível implementar pilhas com vetores também. Falso.

Gabarito: Letra D

4. (CESPE/CEBRASPE – 2021 – SEED-PR) Em determinada estrutura de dados, os valores seguem a regra segundo a qual o último a entrar é o primeiro a sair.

Essa estrutura é do tipo

- a) pilha.
- b) fila.
- c) lista encadeada.
- d) lista duplamente encadeada.
- e) matriz.

Comentários:

A estrutura de dados que segue a regra "último a entrar, primeiro a sair" é chamada de pilha. Uma pilha é uma estrutura de dados que armazena elementos em uma ordem específica, seguindo o método LIFO (Last In First Out). Isso significa que o último elemento a ser inserido na pilha será o primeiro a ser removido. Imagine uma pilha de pratos: o último prato a ser colocado na pilha será o primeiro a ser retirado.

Gabarito: Letra A

5. (CESPE/CEBRASPE – 2021 – SEED-PR) Na estrutura de dados denominada FILA,

- a) o último elemento a ser inserido será o primeiro a ser retirado.
- b) o primeiro elemento a ser inserido será o primeiro a ser retirado: adiciona-se item no fim e remove-se item do início.
- c) os elementos de um mesmo tipo de dado estão organizados de maneira sequencial e ordenada.



- d) os elementos não estão necessariamente armazenados sequencialmente na memória por ordem descrente de valores.
e) os elementos são formados de índices em duas dimensões: linhas e colunas.

Comentários:

Vamos analisar item a item:

- a) o último elemento a ser inserido será o primeiro a ser retirado.

Na fila, o padrão é o FIFO, ou seja, primeiro a ser inserido é o primeiro a ser retirado. Falso.

- b) o primeiro elemento a ser inserido será o primeiro a ser retirado: adiciona-se item no fim e remove-se item do início.

Isso, devido ao padrão FIFO, é o correto. Verdadeiro.

- c) os elementos de um mesmo tipo de dado estão organizados de maneira sequencial e ordenada.

Os elementos podem até ser organizados de maneira sequencial e ordenada, se assim for desejado, mas não é um requisito de uma fila. Falso.

- d) os elementos não estão necessariamente armazenados sequencialmente na memória por ordem descrente de valores.

De fato, os elementos não necessariamente estão armazenados sequencialmente na memória por ordem decrescente de valores. Este item, para mim, deveria estar certo, mas a banca considerou como errado. Penso que, nestes casos, o melhor é considerar o “mais certo”, e, sem dúvidas, a letra B é a definição certíssima de fila!

- e) os elementos são formados de índices em duas dimensões: linhas e colunas.

Isso seria uma matriz, e não uma fila. Falso.

Gabarito: Letra B

6. (CESPE/CEBRASPE – 2017 – TRT-7) A lógica FIFO (first-in first-out) é utilizada na estrutura de dados do tipo

- a) pointer ou ponteiros.
- b) queue ou filas.
- c) stack ou pilhas.
- d) array ou matrizes.

Comentários:

O FIFO, que significa first-in first-out, ou “primeiro a entrar, primeiro a sair”, é usado na estrutura de dados de queue, ou filas. Nela, os elementos são colocados no fim da fila, e removidos no início. O primeiro elemento que é inserido é o primeiro a ser removido.

Quanto aos demais itens, vamos analisar:

- Pointer ou ponteiros: variáveis que armazenam endereços de memória de outras variáveis.
- Stack ou pilhas: estrutura de dados que segue a lógica LIFO, ou last-in first-out, que significa “o último a entrar é o primeiro a sair”.
- Array ou matrizes: estruturas de dados que permitem armazenar e acessar vários valores de uma vez, sem seguir lógica FIFO ou LIFO.

Gabarito: Letra B

7. (CESPE/CEBRASPE – 2017 – TRF-1) Acerca de estrutura de dados, julgue o próximo item.

A fila é uma lista de elementos em que os itens são sempre inseridos em uma das extremidades e excluídos da outra.

Comentários:

Fila é uma estrutura de dados que segue a lógica FIFO (first-in first-out) armazena elementos de forma sequencial, permitindo a inserção de novos elementos no final da estrutura e a remoção de elementos do início. Então, está certo.

Gabarito: Certo

8. (CESPE/CEBRASPE – 2018 – TCE-MG)

Uma estrutura de dados em que o primeiro elemento inserido seja o primeiro elemento a ser retirado é denominada

- a) pilha.
- b) matriz.
- c) árvore binária.
- d) fila.
- e) lista.

Comentários:

A lógica em questão refere-se ao padrão FIFO, que é first-in first-out, ou seja, primeiro a entrar é o primeiro a sair. Vamos analisar item a item:

- a) pilha.

Uma pilha segue o padrão LIFO, que é last-in first-out, ou seja, último a entrar é o primeiro a sair. Falso.

- b) matriz.

É uma estrutura de dados com um conjunto de elementos de um determinado tipo e que estão organizados em linhas e colunas. Não há uma regra especial quanto à inserção ou remoção dos elementos. Falso.

- c) árvore binária.

Estrutura de dados do tipo árvore, mas com uma restrição: cada nó pode ter apenas dois filhos. Falso.

- d) fila.

Isso mesmo, uma fila segue justamente o padrão FIFO. Verdadeiro.

- e) lista.

Numa lista, é possível inserir ou remover de qualquer posição. Falso.

Gabarito: Letra D

9. (FCC – 2019 – TRF-4) O Round-Robin é um tipo de escalonamento preemptivo mais simples e consiste em repartir uniformemente o tempo da CPU entre todos os processos prontos para a execução. Os processos são organizados em uma estrutura de dados, alocando-se a cada um uma fatia de tempo da CPU, igual a um número de quanta. Caso um processo não termine dentro de sua fatia de tempo, retorna para o fim da estrutura e uma nova fatia de tempo é alocada para o processo que está no começo da estrutura e que dela sai para receber o tempo de CPU.

A estrutura de dados utilizada nesse tipo de escalonamento é:

- a) pilha.
- b) árvore B.
- c) fila circular.
- d) fila simples.
- e) árvore binária.

Comentários:



No Round-Robin, os processos são organizados em uma estrutura de dados. Fatias de tempo da CPU são locadas para cada um desses processos. No caso de um processo não terminar dentro da sua fatia de tempo, ele retorna ao fim da estrutura, e uma nova fatia de tempo é alocada para ele. Essa nova fatia está no começo da estrutura. Assim, a estrutura de dados precisa ser uma fila, já que permite que os processos sejam inseridos no final e removidos no início. Além disso, precisa ser circular, já que o processo sai do fim da fila e volta para o início.

Sobre as demais estruturas de dados:

- pilha: não serve, pois a inserção e a remoção dos elementos é sempre no final.
- árvore B: não é uma estrutura de dados linear.
- fila simples: não serve, pois o final não é ligado ao início.
- árvore binária: também, não é uma estrutura de dados linear.

Gabarito: Letra C

10. (FCC – 2013 – MPE-MA) Ana precisa utilizar uma estrutura de dados para gerenciar trabalhos de impressão em uma impressora compartilhada por vários computadores em uma rede. As regras dessa estrutura devem permitir que os trabalhos sejam impressos na ordem em que forem enviados, ou seja, o primeiro a enviar um pedido de impressão deve ser o primeiro a ter sua solicitação atendida. Não deve ser permitido inserir pedidos de impressão no meio dos pedidos já realizados.

A estrutura de dados mais adequada para Ana utilizar é

- a) pilha.
- b) lista encadeada ordenada.
- c) árvore binária.
- d) tabela hash.
- e) fila.

Comentários:

A estrutura de dados utilizada deve seguir a regra FIFO (first-in first-out), ou seja, o primeiro a entrar é o primeiro a sair também. Sair significa ter a solicitação atendida.

Portanto, vamos analisar item a item:

- a) pilha.

A pilha segue o padrão LIFO (last-in first-out), ou seja, o último a entrar é o primeiro a sair. Não serve. Falso.

- b) lista encadeada ordenada.

Uma lista encadeada não é o mais adequado, por não seguir padrões de inserção. Falso.

- c) árvore binária.

Árvore binária também não tem um padrão de inserção e remoção desejado, e também serve para dados estruturados hierarquicamente, o que não é o caso. Falso.

- d) tabela hash.

Tabelas hash servem para facilitar operações de inserção, remoção e busca. Não é o caso, não é necessário realizar busca. Falso.

- e) fila.

A fila é justamente a descrição do necessário: segue a regra FIFO, portanto, o primeiro a entrar é o primeiro a sair. Certo!

Gabarito: Letra E

11. (FCC – 2013 – TRE-SP) No que se refere a estruturas de dados é INCORRETO afirmar:



- a) Numa fila dupla, os elementos podem ser inseridos e removidos de qualquer um dos extremos da fila.
- b) Em qualquer situação é possível usar uma única fila dupla para representar duas filas simples.
- c) A implementação de uma fila dupla normalmente é mais eficiente com uma lista duplamente encadeada que com uma encadeada simples.
- d) Pela definição de fila, se os elementos são inseridos por um extremo da lista linear, eles só podem ser removidos pelo outro.
- e) Numa lista singularmente encadeada, para acessar o último nodo é necessário partir do primeiro e ir seguindo os campos de ligação até chegar ao final da lista.

Comentários:

O enunciado pede a incorreta. Vamos analisar item a item:

a) Numa fila dupla, os elementos podem ser inseridos e removidos de qualquer um dos extremos da fila. Isso mesmo. Como uma fila é dupla, ou seja, vai em ambas as direções, os elementos podem ser inseridos ou removidos de qualquer lado. Verdadeiro.

b) Em qualquer situação é possível usar uma única fila dupla para representar duas filas simples. Uma única fila dupla não pode representar duas filas simples. Isso porque uma fila precisa seguir o padrão FIFO (first-in first-out), ou seja, o primeiro a entrar é o primeiro a sair. Na prática, você tem um lado onde você insere elementos, e o outro lado de onde você remove. Uma fila dupla não pode representar duas filas simples porque, nela, você insere ou remove de qualquer lado. Seria necessário poder inserir de um lado e remover de outro, exclusivamente, e em quatro pontos. Errado.

- c) A implementação de uma fila dupla normalmente é mais eficiente com uma lista duplamente encadeada que com uma encadeada simples.

Está correto. Se já usa uma lista duplamente encadeada, em que cada nó faz referência ao próximo e ao anterior, fica mais fácil implementar uma fila dupla. Certo!

- d) Pela definição de fila, se os elementos são inseridos por um extremo da lista linear, eles só podem ser removidos pelo outro.

É isso, como já disse anteriormente. Correto.

- e) Numa lista singularmente encadeada, para acessar o último nodo é necessário partir do primeiro e ir seguindo os campos de ligação até chegar ao final da lista.

Isso. Numa lista singularmente encadeada, cada nó só possui referência ao próximo da lista, e você só sabe qual é o endereço do primeiro. Então, é preciso percorrer todos. Certo.

Gabarito: Letra B

12. (CESPE - 2010 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação - Análise de Sistemas) Em um programa existe a necessidade de guardar todas as alterações feitas em determinado dado para que seja possível desfazer alterações feitas ao longo de toda a sua existência. Nessa situação, a estrutura de dados mais adequada para o armazenamento de todas as alterações citadas seria uma fila.

Comentários:

Não! Pensem comigo: eu faço uma atividade, depois outra, depois mais uma e, por fim, mais outra. Se eu desejo desfazer a última atividade realizada para retornar a um estado anterior, eu preciso de uma pilha. Dessa forma, resgata-se o último estado válido e, não, o primeiro.

Gabarito: Errado



13. (CESPE - 2012 – TST – Analista de Sistemas – A) As pilhas e as filas são estruturas de dados essenciais para os sistemas computacionais. É correto afirmar que a fila é conhecida como lista LIFO - Last In First Out.

Comentários:

Não, Fila é FIFO!

Gabarito: Errado

14. (CESPE - 2012 - TRE-RJ - Técnico Judiciário - Programação de Sistemas) As filas são estruturas com base no princípio LIFO (last in, first out), no qual os dados que forem inseridos primeiro na fila serão os últimos a serem removidos. Existem duas funções que se aplicam a todas as filas: PUSH, que insere um dado no topo da fila, e POP, que remove o item no topo da fila.

Comentários:

Não, isso é uma Pilha (LIFO).

Gabarito: Errado

15. (FCC - 2012 - MPE-AP – Analista de Sistemas - A) Nas estruturas de dados, devido às características das operações da fila, o primeiro elemento a ser inserido será o último a ser retirado. Estruturas desse tipo são conhecidas como LIFO.

Comentários:

Não, será o primeiro a ser retirado – são do tipo FIFO!

Gabarito: Errado

16. (FCC - 2012 - MPE-AP – Analista de Sistemas - C) Nas estruturas de dados, a fila é uma lista linear na qual as operações de inserção e retirada ocorrem apenas no início da lista.

Comentários:

Não, isso é a definição de Pilha!

Gabarito: Errado

17. (FCC - 2012 - TRE-SP - Analista Judiciário - Análise de Sistemas – D) Pela definição de fila, se os elementos são inseridos por um extremo da lista linear, eles só podem ser removidos pelo outro.

Comentários:

Exato! Essa é a definição de fila: insere-se por um extremo e remove-se por outro.

Gabarito: Correto



18. (FCC - 2011 - TRT - 19ª Região (AL) - Analista Judiciário - Tecnologia da Informação) FIFO refere-se a estruturas de dados do tipo:

- a) fila.
- b) árvore binária.
- c) pilha.
- d) matriz quadrada.
- e) cubo.

Comentários:

Trata-se da Fila!

Gabarito: Letra A

19. (ESAF - 2010 - CVM - Analista de Sistemas - prova 2) Uma fila é um tipo de lista linear em que:

- a) as inserções são realizadas em um extremo e as remoções no outro extremo.
- b) as inserções e remoções são realizadas em um mesmo extremo.
- c) podem ser realizadas apenas inserções.
- d) a inserção de um elemento requer a remoção de outro elemento.
- e) a ordem de saída não corresponde à ordem de entrada dos elementos.

Comentários:

As inserções são realizadas em um extremo e as remoções são realizadas no outro extremo, por isso é FIFO!

Gabarito: Letra A

20. (CESPE - 2010 - DETRAN-ES - Analista de Sistemas) No armazenamento de dados pelo método FIFO (first in - first out), a estrutura de dados é representada por uma fila, em cuja posição final ocorrem inserções e, na inicial, retiradas.

Comentários:

Perfeito! Basta lembrar de uma fila: o primeiro a entrar é o primeiro a sair.

Gabarito: Correto

21. (CESPE - 2008 - TRT - 5ª Região (BA) - Técnico Judiciário - Tecnologia da Informação) Entre alguns tipos de estrutura de dados, podem ser citados os vetores, as pilhas e as filas.

Comentários:

Perfeito, são todos exemplos de estruturas de dados!

Gabarito: Correto



22. (CESPE - 2004 – SES/PA – Analista de Sistemas) Uma estrutura mais geral que as pilhas e filas é o deque, em que as inserções, retiradas e acessos são permitidos em ambas as extremidades.

Comentários:

Perfeito, dequeus permitem todas essas operações!

Gabarito: Correto

23. (CESPE - 2009 – TCE/AC – Analista de Sistemas – D) Um deque (double ended queue) requer inserção e remoção no topo de uma lista e permite a implementação de filas com algum tipo de prioridade. A implementação de um deque, geralmente é realizada com a utilização de uma lista simplesmente encadeada.

Comentários:

Não, pode ser do início ou fim da lista! De fato, permite a implementação de filas com algum tipo de prioridade, mas geralmente é realizada com a utilização de filas duplamente encadeadas.

Gabarito: Errado

24. (FCC - 2007 – TRT/23 – Analista de Sistemas) Uma estrutura de dados com vocação de FIFO de duplo fim e que admite a rápida inserção e remoção em ambos os extremos é:

- a) uma pilha.
- b) uma splay tree.
- c) um deque.
- d) uma lista linear.
- e) uma árvore AVL.

Comentários:

Trata-se de um Deque – fila duplamente encadeada!

Gabarito: Letra C

25. (CESPE - 2004 – PBV/RR - Analista de Sistemas) As filas com prioridade são listas lineares nas quais os elementos são pares da forma (q_i, p_i) , em que q é o elemento do tipo base e p é uma prioridade. Elas possuem uma política de fila do tipo FIFO (first in first out) entre os elementos de mesma prioridade.

Comentários:

Perfeito! É assim que funciona a prioridade em conjunto com filas. **Gabarito: C**

Gabarito: Correto



26. (CESPE - 2004 – STJ – Analista de Sistemas) A seguir, está representada corretamente uma operação de retirada em uma fila de nome f.

```
se f.começo = nil então
    erro {fila vazia}
senão j ← f.começo.info
```

Comentários:

Não, o correto seria:

```
se f.começo = nil então
    erro {fila vazia}
senão f.começo ← f.começo.anterior
```

Por que, professor? As duas primeiras linhas estão apenas dizendo que se o primeiro elemento da fila for Null (ou Nil), vai dar erro porque a fila está vazia, logo não há como retirar elementos de uma fila vazia. Agora vejam a última linha: ele atribui a uma variável j o valor f.começo.info. Na verdade, ele simplesmente está colocando em j os dados do primeiro elemento da fila, mas a questão pede o código para retirar um elemento da lista e não para capturar os dados do primeiro elemento. Na resposta, eu coloco que f.começo, i.e., o primeiro elemento da lista vai ser f.começo.anterior, ou seja, temos um novo primeiro elemento e eu retirei aquele elemento anterior.

Gabarito: Errado

27. (FCC - 2016 - Prefeitura de Teresina - PI - Analista Tecnológico - Analista de Suporte Técnico)

- Considerando uma estrutura de dados do tipo fila, e a seguinte sequência de comandos sobre essa fila (sendo que o comando Push representa uma inserção de elemento e o comando Pop representa uma exclusão de elemento) e considerando também que a fila estava inicialmente vazia:

Push 3, Push 5, Pop 3, Push 7, Pop 5, Push 9, Push 8

Após a execução dessa sequência de comandos, o conjunto de elementos que resulta na fila é:

- a) 3 – 5 – 7 – 9 – 8.
- b) 7 – 9 – 8 – 3 – 5.
- c) 3 – 3 – 5 – 5 – 7 – 9 – 8.
- d) 7 – 9 – 8.
- e) 3 – 5 – 3 – 7 – 5 – 9 – 8.

Comentários:

Como a questão nem pediu a ordem, ficou bem fácil. Push inclui e pop retira. Se há dois pop's, os elementos 3 e 5 são removidos da fila, sobrando 7, 9 e 8.

Gabarito: Letra D

28. (FCC - 2016 – TRT - 23ª REGIÃO (MT) - Técnico Judiciário - Tecnologia da Informação) Estruturas de dados básicas, como as pilhas e filas, são usadas em uma gama variada de aplicações. As filas, por exemplo, suportam alguns métodos essenciais, como o:

- a) enqueue(x), que insere o elemento x no fim da fila, sobrepondo o último elemento.



- b) dequeue(), que remove e retorna o elemento do começo da fila; um erro ocorrerá se a fila estiver vazia.
- c) push(x), que insere o elemento x no topo da fila, sem sobrepor nenhum elemento.
- d) pop(), que remove o elemento do início da fila e o retorna, ou seja, devolve o último elemento inserido.
- e) top(), que retorna o elemento do fim da fila sem removê-lo; um erro ocorrerá se a fila estiver vazia.

Comentários:

Queue = fila! Stack = pilha! Sendo assim, o que seria “enqueue”? Para quem tem idade (assim como eu!! 😊) deve se lembrar do famoso Winamp (ah, minha época de ouvir mp3!). Quanto clicávamos no botão direito de uma música, sempre tinha a opção “enqueue in Winamp”, ou seja, incluir na fila de reprodução. Enqueue, portanto, inclui na fila, enquanto dequeue remove!

Gabarito: Letra B

29. (FCC - 2017 – TRE/BA - Analista de Sistemas) A estrutura que, além de ser similar à fila, é apropriada para ampliar as características desta, permitindo inserir e retirar elementos tanto do início quanto do fim da fila, é o(a):

- a) árvore.
- b) lista duplamente encadeada.
- c) deque.
- d) fila circular.
- e) pilha

Comentários:

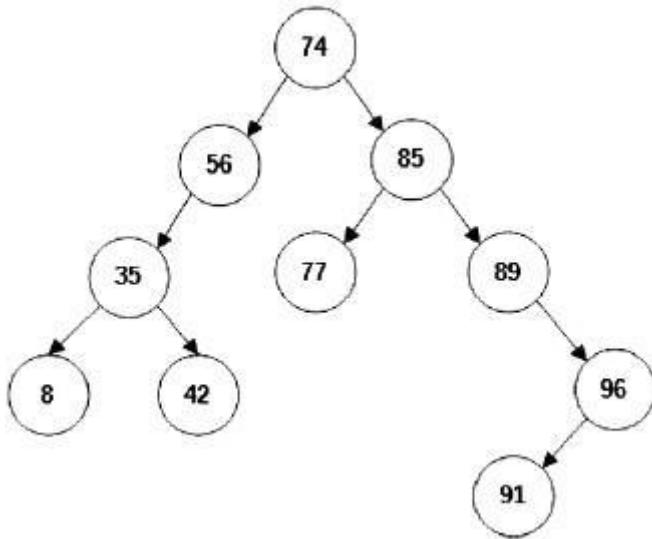
Trata-se de um Deque (Double Ended Queue). Deque é uma estrutura de dados similar à fila e que permite que elementos possam ser adicionados ou removidos da frente (cabeça) ou de trás (cauda). Qual a diferença entre uma lista duplamente encadeada e um deque? Um deque gerencia elementos como um vetor, fornece acesso aleatório e tem quase a mesma interface que um vetor.

Uma lista duplamente encadeada se difere de um deque por não fornecer acesso aleatório aos elementos, i.e., para acessar o quinto elemento, você deve necessariamente navegar pelos quatro primeiros elementos – logo a lista é mais lenta nesse sentido. Existem outras diferenças, mas essa é a diferença fundamental entre essas duas estruturas. Bacana?

Gabarito: Letra C



1. (IFTM – IFTM– 2023) Dada a árvore de busca binária da imagem a seguir, insira os elementos 23, 12, 64, 93, 71 e 86 (nessa ordem).



Após a inserção, qual é a saída do percurso pós-ordem?

- a) 8, 12, 23, 35, 42, 56, 64, 71, 74, 77, 85, 86, 89, 91, 93, 96.
- b) 74, 56, 35, 8, 23, 12, 42, 64, 71, 85, 77, 89, 86, 96, 91, 93.
- c) 12, 23, 8, 42, 35, 71, 64, 56, 77, 86, 93, 91, 96, 89, 85, 74.
- d) 12, 23, 8, 42, 35, 64, 56, 71, 77, 86, 93, 91, 96, 89, 85, 74.
- e) Nenhuma das alternativas.

Comentários:

Vamos relembrar o conceito de árvore binária? Uma árvore binária é uma estrutura de dados hierárquica que consiste em nós e arestas. Cada nó possui no máximo dois filhos, chamados de filho esquerdo e filho direito. A raiz da árvore é o nó superior, que não possui pais. As árvores binárias podem ser usadas para representar diversos tipos de dados, como números, strings, objetos, etc. Elas são frequentemente usadas em algoritmos de busca, ordenação e processamento de dados. Em uma árvore binária balanceada, cada nó da árvore tem a mesma altura ou a altura do nó esquerdo é um a menos que a altura do nó direito. Isso garante que a árvore possa ser percorrida de forma eficiente, usando algoritmos como a busca binária. Existem diversos algoritmos de平衡amento para árvores binárias. Um dos algoritmos mais conhecidos é o algoritmo AVL, que usa uma propriedade chamada fator de平衡amento para determinar se um nó está desbalanceado. A inserção pós-ordem é um tipo de inserção de elementos numa árvore binária. Nessa inserção, os elementos são inseridos na árvore na seguinte ordem: 1. Inserir todos os elementos filhos do nó atual. 2. Inserir o próprio nó. A inserção pós-ordem é frequentemente usada em algoritmos de processamento de dados que precisam visitar todos os elementos da árvore uma única vez.

Vamos juntos, passo a passo para entender como é feita a inserção nesse caso.

Inserção do elemento 23:



2. Comparamos 23 com a raiz do lado esquerdo, que é 56. Como 23 é menor que 56, seguimos para o lado esquerdo da árvore.
3. Comparamos 23 com a raiz do lado esquerdo, que é 35. Como 23 é menor que 35, seguimos para o lado esquerdo da árvore.
4. Comparamos 23 com a raiz do lado esquerdo, que é 8. Como 23 é maior que 8, inserimos 23 como o novo filho direito de 8.

Inserção do elemento 12:

1. Comparamos 12 com a raiz, que é 74. Como 12 é menor que 74, seguimos para o lado esquerdo da árvore.
2. Comparamos 12 com a raiz do lado esquerdo, que é 56. Como 12 é menor que 56, seguimos para o lado esquerdo da árvore.
3. Comparamos 12 com a raiz do lado esquerdo, que é 35. Como 12 é menor que 35, inserimos 12 como o novo filho esquerdo de 35.

Inserção do elemento 64:

1. Comparamos 64 com a raiz, que é 74. Como 64 é maior que 74, seguimos para o lado direito da árvore.
2. A raiz do lado direito é vazia, então inserimos 64 como a nova raiz do lado direito.

Inserção do elemento 93:

1. Comparamos 93 com a raiz, que é 74. Como 93 é maior que 74, seguimos para o lado direito da árvore.
2. Comparamos 93 com a raiz do lado direito, que é 64. Como 93 é maior que 64, inserimos 93 como o novo filho direito de 64.

Inserção do elemento 71:

1. Comparamos 71 com a raiz, que é 74. Como 71 é menor que 74, seguimos para o lado esquerdo da árvore.
2. A raiz do lado esquerdo é 56. Comparamos 71 com 56. Como 71 é maior que 56, inserimos 71 como o novo filho direito de 56.

Inserção do elemento 86:

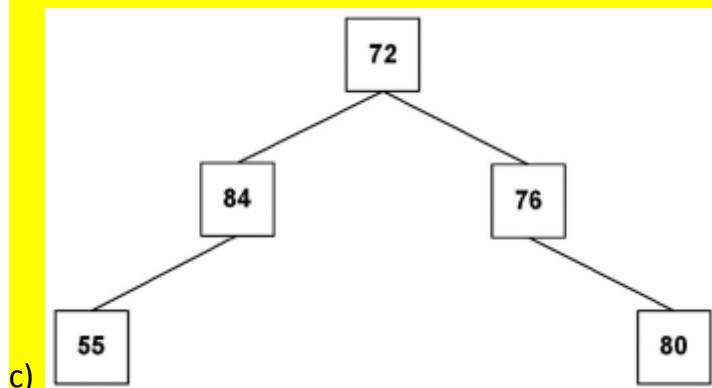
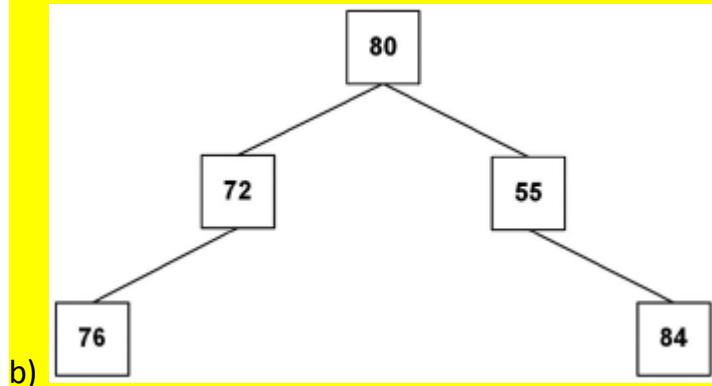
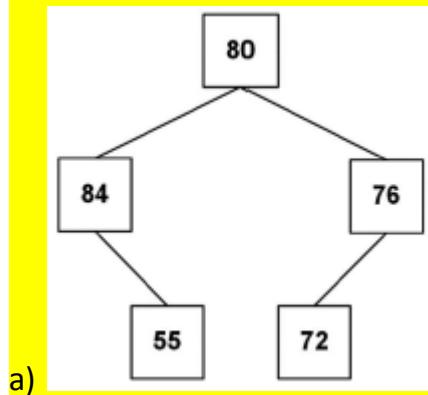
1. Comparamos 86 com a raiz, que é 74. Como 86 é menor que 74, seguimos para o lado esquerdo da árvore.
2. Comparamos 86 com a raiz do lado esquerdo, que é 56. Como 86 é maior que 56, seguimos para o lado esquerdo da árvore.
3. Comparamos 86 com a raiz do lado esquerdo, que é 35. Como 86 é maior que 35, seguimos para o lado esquerdo da árvore.

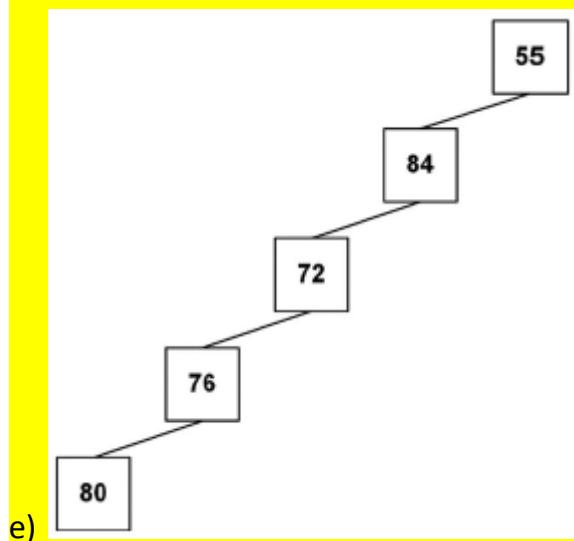
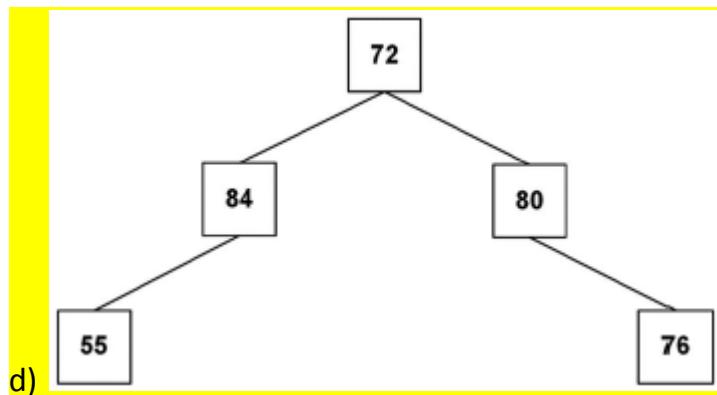


2. (CESGRANRIO – Banco do Brasil – 2023). Um estudante de computação decidiu escrever um método Java para exibir, no console, em pré-ordem, os valores dos nós de uma árvore binária recebida como parâmetro. Ao executar esse método, os seguintes valores foram exibidos no console:

80 84 55 76 72

Considerando os valores exibidos, qual árvore foi recebida como parâmetro?





Comentários:

O gabarito da questão é a Alternativa (A). A questão menciona que estamos lidando com uma árvore binária, o que significa que cada nó pode ter no máximo dois filhos. Existem três tipos de percursos em uma árvore binária: Pré-Ordem, Em Ordem e Pós-Ordem. No caso específico, estamos lidando com um percurso em pré-ordem, em que visitamos a raiz primeiro, em seguida os nós à esquerda e depois os nós à direita. A ordem de exibição em pré-ordem de uma árvore binária é a seguinte:

- 1 O valor do nó raiz.
- 2 Os valores dos nós da subárvore esquerda, em pré-ordem.
- 3 Os valores dos nós da subárvore direita, em pré-ordem.

Considerando os valores exibidos, a ordem de exibição é a seguinte:

O primeiro valor, 80, representa a raiz da árvore.

O segundo valor, 84, representa a raiz da sub-árvore à direita, pois estamos em pré-ordem.

O 55 é o filho à esquerda da raiz da sub-árvore à direita (84).

O 76 é o filho à direita da raiz da sub-árvore à direita (80).

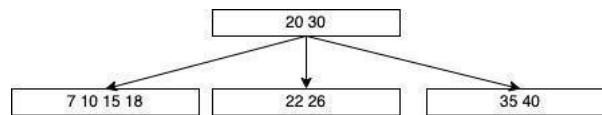
O 72 é o filho à esquerda da raiz da sub-árvore à direita (76).

A ordem desses valores indica a ordem de visita durante o percurso em pré-ordem.

Gabarito: Letra A



no máximo 4 filhas, contendo as chaves 7, 10, 15, 18, 20, 22, 26, 30, 35, 40.



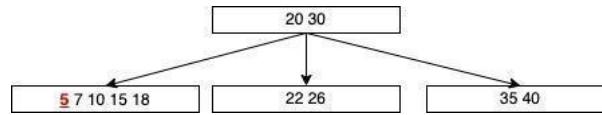
Após a inserção da chave 5, a configuração das chaves do nó raiz da árvore seria

- a) 5, 20, 30
- b) 10, 20, 30
- c) 5, 30
- d) 10, 30
- e) 20, 30

3.1 Comentários:

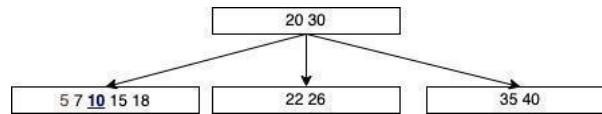
Em uma árvore B, os valores estão ordenados: ordenados dentro do mesmo nível, e os filhos da esquerda possuem valores menores do que os filhos da direita.

Dessa forma, se formos inserir o número 5, ele ficaria neste ponto:

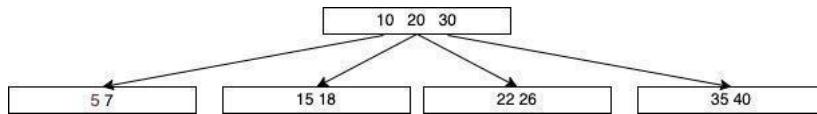


Porém, o enunciado diz que a árvore B pode ter apenas 4 filhos, e, se colocarmos o valor 5 neste ponto, teremos 5 filhos. Dessa forma, temos que rebalancear a árvore.

Para rebalanceá-la, temos que pegar o filho do meio:



O filho do meio é o 10. Devemos jogar para o nível de cima.



O nó raiz é justamente o primeiro da árvore. Ou seja, a resposta é 10 20 30.

Gabarito: Letra B

3. (FGV – 2022 – SEMSA Manaus) Numa estrutura de dados do tipo Árvore B, onde cada nó não raiz pode conter entre 2 e 4 chaves, a complexidade do algoritmo de busca é da ordem



- a) log de N na base 2.
- b) log de N na base d.
- c) N vezes log de N na base 2.
- d) N.
- e) N^2 .

3.2 Comentários:

O limite superior da profundidade da árvore é $d \leq 1 + \log_{[m/2]}((N+1)/2)$, sendo que:

- $m =$ a ordem da árvore, ou seja, o número máximo de chaves numa página não folha. Neste caso, $m = 2d$.
- $N =$ a quantidade de registros.

Portanto:

$$d \leq 1 + \log_{[m/2]}((N+1)/2)$$

$$d \leq 1 + \log_{[2d/2]}((N+1)/2)$$

$$d \leq 1 + \log_{[d]}((N+1)/2)$$

Isso é da ordem de log de N na base d, ou $O(\log_d N)$.

Gabarito: Letra B

4. (UFPRE – 2022 – UFPRE) Acerca de estruturas de dados, assinale a alternativa correta.

- a) A estrutura denominada Pilha é considerada do tipo FIFO (first in, first out); o primeiro elemento inserido será o primeiro elemento a ser removido.
- b) A estrutura denominada Fila é considerada do tipo FILO (first in, last out); o primeiro elemento a ser inserido será o último elemento a ser removido.
- c) A estrutura denominada lista simplesmente encadeada não ordenada armazena um ou vários dados em cada elemento, e tem um ponteiro apontado para o último elemento que permite o encadeamento e a estrutura linear.
- d) A estrutura denominada árvore é um conjunto finito de elementos, onde cada elemento é denominado nó, e o primeiro nó é conhecido como raiz da árvore.
- e) A estrutura denominada árvore AVL é uma árvore binária não balanceada, em que cada nó representa uma diferença de altura entre as subárvore direita e esquerda de 1, 2 ou 3 nós.

3.3 Comentários:

Vamos analisar item a item:

- a) A estrutura denominada Pilha é considerada do tipo FIFO (first in, first out); o primeiro elemento inserido será o primeiro elemento a ser removido.

Na verdade, uma pilha é LIFO (last-in first-out), ou seja, o último a entrar é o primeiro a sair. Falso.

- b) A estrutura denominada Fila é considerada do tipo FILO (first in, last out); o primeiro elemento a ser inserido será o último elemento a ser removido.

Na verdade, a fila é FIFO (first-in first-out), ou seja, o primeiro a entrar é o primeiro a sair. Falso.

- c) A estrutura denominada lista simplesmente encadeada não ordenada armazena um ou vários dados em cada elemento, e tem um ponteiro apontado para o último elemento que permite o encadeamento e a estrutura linear.

O ponteiro está apontado para o primeiro elemento, e não para o último. Falso.

- d) A estrutura denominada árvore é um conjunto finito de elementos, onde cada elemento é denominado nó, e o primeiro nó é conhecido como raiz da árvore.

Alternativa sem erro sobre árvore. Verdadeiro.

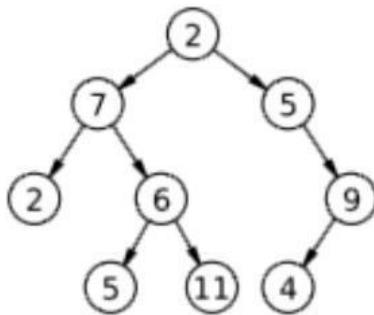
- e) A estrutura denominada árvore AVL é uma árvore binária não balanceada, em que cada nó representa uma diferença de altura entre as subárvore direita e esquerda de 1, 2 ou 3 nós.



A árvore AVL é uma árvore binária balanceada, e a diferença entre as subárvore da direita e da esquerda é de no máximo 1 nó. Falso.

Gabarito: Letra D

5. (MetroCapital Soluções – 2022 – Prefeitura de Nova Odessa – SP) A Estrutura de dados (ED) é um modo particular de armazenamento e organização de dados em um computador de modo que possam ser usados eficientemente. Analise a imagem a seguir:



Qual estrutura de dados representa a imagem:

- a) Pilha.
- b) Fila.
- c) Grafo.
- d) Vetor de vetores.
- e) Árvore binária.

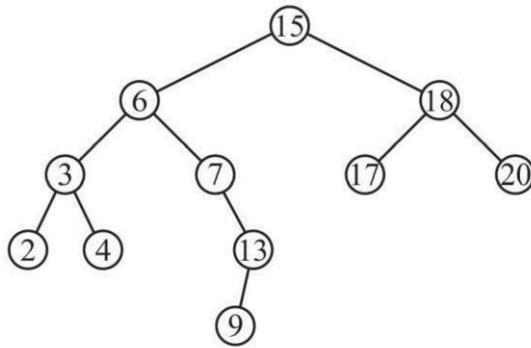
3.4 Comentários:

Se observar, temos um nó raiz, que liga a mais dois nós filhos. Cada nó filho também liga a pelo menos dois nós filhos.

Isso é justamente uma estrutura de árvore binária, que é uma árvore com a restrição de 2 nós filhos por nó.

Gabarito: Letra E

6. (CESPE/CEBRASPE – 2020 – TJ-PA)



Thomas H. Cormen et al. *Algoritmos: teoria e prática*. Editora Campus, v. 2, 2002. p. 207.

De acordo com a figura anterior, o procedimento



CONSULTA (x)

1 while esquerda [x] ≠ NIL

2 do x ← esquerda [x]

3 return x

realiza, na árvore, a consulta de

- a) search.
- b) minimum.
- c) maximum.
- d) successor.
- e) predecessor.

3.5 Comentários:

Trata-se de uma árvore binária. Nessa estrutura, os nós filhos da esquerda sempre possuem um valor menor do que os da direita. Os menores valores da árvore estarão à esquerda.

Dito isso, o algoritmo percorre os nós da esquerda sempre. Ou seja, vai chegar no número 2.

Significa que está procurando o menor valor da árvore. Ou seja, resposta é *minimum*.

Gabarito: Letra B

7. (CESPE/CEBRASPE – 2022 – Petrobrás) Uma árvore de decisão representa um determinado número de caminhos possíveis de decisão e os resultados de cada um deles, apresentando muitos pontos positivos, ou seja, são fáceis de entender e interpretar. Elas têm processo de previsão completamente transparente e lidam facilmente com diversos atributos numéricos, assim como atributos categóricos, podendo até mesmo classificar dados sem atributos definidos.

De acordo com os aspectos construtivos de uma árvore de decisão, julgue o item a seguir.

A entropia de uma árvore de decisão aborda o aspecto da quantidade de informações que está associada às respostas que podem ser obtidas às perguntas formuladas, representando o grau de incerteza associado aos dados.

3.6 Comentários:

A entropia é uma medida de mistura ou impureza de um conjunto de dados. No caso de uma construção de uma árvore de decisão, mede o grau de incerteza dos dados em cada nó da árvore.

Gabarito: Correto

8. (CESPE/CEBRASPE – 2022 – Petrobrás) Uma árvore de decisão representa um determinado número de caminhos possíveis de decisão e os resultados de cada um deles, apresentando muitos pontos positivos, ou seja, são fáceis de entender e interpretar. Elas têm processo de previsão completamente transparente e lidam facilmente com diversos atributos numéricos, assim como atributos categóricos, podendo até mesmo classificar dados sem atributos definidos.

De acordo com os aspectos construtivos de uma árvore de decisão, julgue o item a seguir.

Se o processo adotado para a construção de árvores de decisão for determinístico, uma forma de obtenção de árvores aleatórias, que compõem as florestas aleatórias, pode ser realizada por meio do bootstrap dos dados, em que cada árvore é treinada com base no resultado de bootstrap_sample (inputs).



3.7 Comentários:

As florestas aleatórias correspondem a uma técnica de aprendizado de máquina em que se um conjunto de árvores de decisão com base em um conjunto de dados.

É por meio do processo bootstrap que essas árvores são treinadas usando amostras aleatórias dos dados, com reposição. Dessa forma, cada árvore da floresta é treinada em um conjunto de dados diferente, incrementando a variabilidade das árvores e a robustez da floresta aleatória. Feito o treinamento, as árvores são então utilizadas para previsões ou classificações.

Gabarito: Correto

9. (FGV – 2022 – MPE-GO) Árvores B são muito usadas na implementação de índices em bancos de dados.

Uma árvore desse tipo é dita balanceada quando

- a) a complexidade do algoritmo de busca é logarítmica.
- b) as chaves são armazenadas em ordem de classificação, crescente ou decrescente.
- c) é possível localizar registros referenciados por um intervalo de chaves.
- d) o número de ponteiros em cada nó intermediário é constante.
- e) toda página folha tem o mesmo número de páginas intermediárias até a raiz.

3.8 Comentários:

Vamos analisar item a item:

- a) a complexidade do algoritmo de busca é logarítmica.

A complexidade de busca da Árvore B é logarítmica, mas não é uma condição para que ela seja balanceada. Falso.

b) as chaves são armazenadas em ordem de classificação, crescente ou decrescente. Também não é o motivo pela qual ela seria balanceada. Falso.

c) é possível localizar registros referenciados por um intervalo de chaves. Até é possível, mas não é o que torna ela balanceada. Falso.

d) o número de ponteiros em cada nó intermediário é constante. Não é o que a torna balanceada. Falso.

- e) toda página folha tem o mesmo número de páginas intermediárias até a raiz.

Isso. Se cada página folha tiver o mesmo número de páginas intermediárias até a raiz, teremos uma árvore com altura fixa, e, portanto, balanceada. Verdadeiro.

Gabarito: Letra E

10. (FGV – 2018 – Câmara de Salvador - BA) Gerenciadores de bancos de dados frequentemente empregam índices implementados na forma de árvores B. Nesse tipo de organização, considerando-se uma árvore na qual o número máximo de chaves numa página não folha é 19 (ou seja, $d=20$), o número máximo de acessos necessários para localizar uma chave, num universo de 10 milhões de chaves distintas, é:

- a) 4;
- b) 7;
- c) 19;
- d) 100;
- e) 316.

3.9 Comentários:

O limite superior da profundidade da árvore é $d \leq 1 + \log_{d/2}((N+1)/2)$, sendo que:



- $d =$ a ordem da árvore, ou seja, o número máximo de chaves numa página não folha. Neste caso, $d = 20$.

- $N =$ a quantidade de registros. Neste caso, $N = 10.000.000$. Portanto, calculemos:

$$d \leq 1 + \log_{[d/2]}((N+1)/2)$$

$$d \leq 1 + \log_{[20/2]}((10.000.000+1)/2)$$

$$d \leq 1 + \log_{[10]}((10.000.001/2))$$

$$d \leq 1 + \log_{[10]}(5.000.000,5)$$

Para calcularmos na mão este logaritmo, vamos fazer multiplicando 10 por 10 até chegarmos no valor, para termos alguma ideia. Portanto:

$$10 * 10 = 100 \text{ (2x)}$$

$$100 * 10 = 1.000 \text{ (3x)}$$

$$1.000 * 10 = 10.000 \text{ (4x)}$$

$$10.000 * 10 = 100.000 \text{ (5x)}$$

$$100.000 * 10 = 1.000.000 \text{ (6x)}$$

$$1.000.000 * 10 = 10.000.000 \text{ (7x)}$$

Note do 6 para o 7, o valor ultrapassou 5.000.000,5. Portanto $\log_{[10]}(5.000.000,5)$ é mais ou menos 6,5, com um arredondamento bem estimado.

Então:

$$d \leq 1 + \log_{[10]}(5.000.000,5)$$

$$d \leq 1 + 6,5$$

$$d \leq 7,5$$

Portanto, a quantidade máxima de acessos é $d \leq 7,5$. Como a quantidade máxima de acessos é um número inteiro, então é 7.

Gabarito: Letra B

11. (FGV – 2018 – MPE-AL) Em uma árvore B de ordem d, onde cada nó que não o raiz possui entre d e 2d chaves, estão armazenadas 30.000 chaves.

Sabendo-se que $d=8$, assinale a opção que indica o número máximo de nós visitados para a localização de uma chave.

- 3
- 5
- 7
- 15
- 15.000

3.10 Comentários:

O limite superior da profundidade da árvore é $d \leq 1 + \log_{[m/2]}((N+1)/2)$, sendo que:

- $m =$ a ordem da árvore, ou seja, o número máximo de chaves numa página não folha. Neste caso, $m = 16$, porque a quantidade máxima de chaves numa página é $2d = 2*8 = 16$.

- $N =$ a quantidade de registros. Neste caso, $N = 30.000$. Assim:

$$d \leq 1 + \log_{[m/2]}((N+1)/2)$$

$$d \leq 1 + \log_{[16/2]}((30000+1)/2)$$

$$d \leq 1 + \log_{[8]}((30001)/2)$$

$$d \leq 1 + \log_{[8]}(15000,5)$$

Vamos calcular $\log_{[8]}(15000,5)$ multiplicando 8 por 8 até o valor ultrapassar 15.000,5:

$$8 * 8 = 64 \text{ (2x)}$$



$$64 \cdot 8 = 512 \text{ (3x)}$$

$$512 \cdot 8 = 4.096 \text{ (4x)}$$

$$4.096 \cdot 8 = 32.768 \text{ (5x)}$$

Note que o valor ultrapassa 15.000 do 4 para o 5. Portanto, $\log_8(15000,5)$ está entre 4 e 5. Vamos arredondar e dizer que é 4,5.

$$d \leq 1 + 4,5$$

$$d \leq 5,5$$

Como a quantidade máxima de tentativas é um inteiro, então é 5.

Gabarito: Letra B

12. (FGV – 2018 – Banestes) Sobre as características de índices estruturados na forma de Btrees e Hash tables, analise as afirmativas a seguir.

- I. Hash tables aplicam-se somente em buscas que referenciam a chave por inteiro (operador =).
- II. B-trees favorecem consultas que buscam chaves num determinado intervalo (operadores \geq e \leq).
- III. B-trees são usualmente mais lentas para buscas pela chave (operador =).
- IV. Hash tables favorecem buscas, com o operador ‘LIKE’ do SQL, que não contenham caracteres curingas na primeira posição.
- V. B-trees não se aplicam em buscas que se referem a uma substring à esquerda da chave.

Está correto o que se afirma em:

- a) nenhuma;
- b) somente I, II e III;
- c) somente I, IV e V;
- d) somente II, III, IV;
- e) I, II, III, IV e V.

3.11 Comentários:

Vamos analisar item a item:

- I. Hash tables aplicam-se somente em buscas que referenciam a chave por inteiro (operador =).

De fato, as hash tables precisam referenciar a chave inteiramente para que se possa encontrar os valores. Certo.

- II. B-trees favorecem consultas que buscam chaves num determinado intervalo (operadores \geq e \leq). Como, na árvore B, os valores menores estão nos filhos da esquerda, enquanto os valores maiores estão nos filhos da direita, então, torna-se mais fácil a busca por intervalo. Certo.

- III. B-trees são usualmente mais lentas para buscas pela chave (operador =).

São mais lentas do que hash tables, porque, no caso das hash tables, se eu sei a chave exatamente, então o acesso é direto, ou próximo do direto. Já em B-trees, seria $O(\log n)$. Certo.

- IV. Hash tables favorecem buscas, com o operador ‘LIKE’ do SQL, que não contenham caracteres curingas na primeira posição.

Em hash tables, o operador like não é favorecido. O operador like busca pelo valor parcial da chave, mas, nas hash tables, a busca é facilitada apenas quando temos o valor inteiro da chave. Errado.

- V. B-trees não se aplicam em buscas que se referem a uma substring à esquerda da chave.

Pelo contrário, quando temos substrings, as B-trees podem ser aplicadas e são mais eficientes do que hash tables. Errado.

Gabarito: Letra B

13. (FCC – 2017 – TRE-SP) Considere, hipoteticamente, que um Técnico do TRE-SP tem, em seu computador, a seguinte organização de um diretório



Principal: Dados

Dentro de Dados: Técnicos Práticos

Dentro de Técnicos: Árvores Hash

Recursão Filas Pilhas

Dentro de Práticos: Programas

A Fazer Prontos

Dentro de Prontos: Eleições

Urnas

Dentro de Programas: Corretos

ComErro

Dentro de ComErro: Urgentes

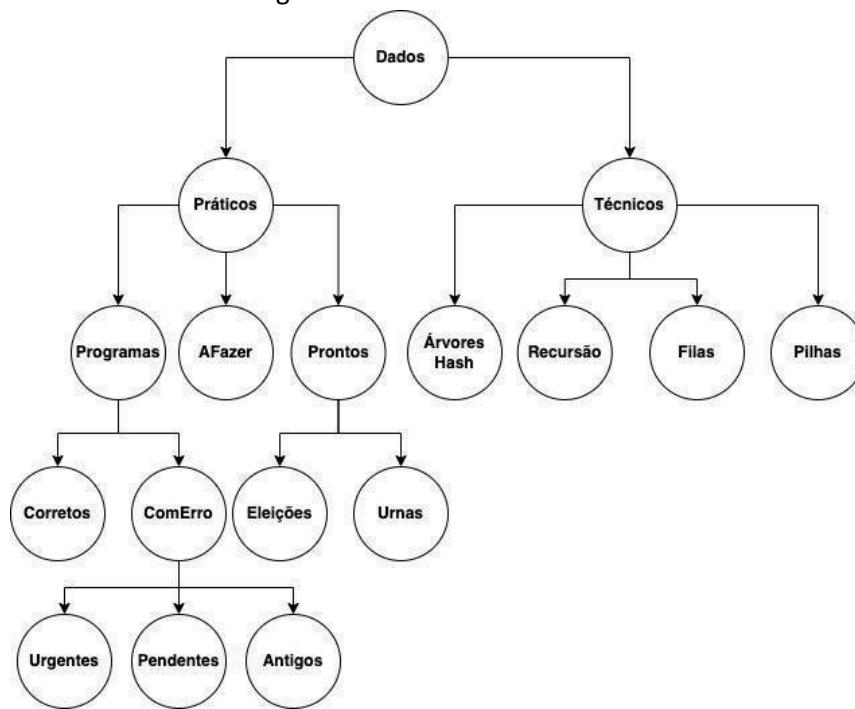
Pendentes Antigos

A estrutura de dados

- fila é a mais adequada para representar este diretório.
- pilha é a mais adequada para representar este diretório.
- árvore binária, ao armazenar este diretório, terá Dados na raiz e nós com grau 2, 3, 5 e folhas.
- árvore, que consegue armazenar este diretório, é de ordem 5.
- hashing, ao armazenar este diretório, não terá colisões na tabela de dispersão

3.12 Comentários:

Pela descrição, a estrutura de dados seria a seguinte:



Isso é uma estrutura de árvore.

Portanto, excluímos as alternativas a), b) e e). Sobram as alternativas c) e d).

A alternativa c) fala de árvore binária. Não se trata de uma árvore binária, pois, em uma árvore binária, cada nó possui apenas dois filhos, o que não é o caso dessa estrutura.

Resta a alternativa d). De fato, é uma árvore, e, de fato, a sua ordem é 5. A ordem de uma árvore corresponde à quantidade máxima de níveis que ela possui. Note que ela possui 5 níveis.

Gabarito: Letra D

14. (FCC – 2019 – TRF-4) Determinada estrutura de dados foi projetada para minimizar o número de acessos à memória secundária. Como o número de acessos à memória secundária depende

diretamente da altura da estrutura, esta foi concebida para ter uma altura inferior às estruturas hierarquizadas similares, para um dado número de registros. Para manter o número de registros armazenados e, ao mesmo tempo, diminuir a altura, uma solução é aumentar o grau de ramificação da estrutura (o número máximo de filhos que um nó pode ter). Assim, esta estrutura possui um grau de ramificação geralmente muito maior que 2. Além disso, a cada nó são associados mais de um registro de dados: se o grau de ramificação de um nó for g , este pode armazenar até $g-1$ registros.

Esta estrutura de dados é utilizada em banco de dados e sistema de arquivos, sendo denominada

- árvore digital ou trie.
- árvore B.
- lista linear duplamente encadeada circular.
- árvore rubro-negra.
- árvore binária de busca não balanceada.

3.13 Comentários:

A questão se refere à Árvore B. É uma modificação da árvore binária de busca balanceada que armazena os dados de forma organizada, permitindo a eles acesso rápido, reduzindo a quantidade de vezes em que é necessário acessar a memória secundária.

Este número de acessos à memória secundária depende diretamente da altura da estrutura, então ela foi criada para ter uma altura menor do que a de outras estruturas hierarquizadas semelhantes.

Quanto às demais estruturas de dados, sabemos que:

- digital ou trie: estrutura de dados para armazenar uma coleção de strings de forma organizada, para ser usado de forma eficiente em um dicionário.
- lista linear duplamente encadeada circular: uma lista em que cada nó tem ligação com o anterior e o próximo. Além disso, o primeiro nó tem ligação com o último, e vice-versa.
- árvore rubro-negra: uma variante de uma árvore de busca balanceada. Os nós da árvore podem ser coloridos de vermelho ou preto. Há algumas regras:
 - os nós vermelhos sempre têm dois filhos pretos
 - a raiz e as folhas devem ser nós pretos.
- árvore binária de busca não balanceada: os elementos são dispostos na árvore de modo que as operações são realizadas em tempo linear. É o contrário da Árvore B, que é balanceada.

Gabarito: Letra B

15. (FCC – 2015 – DPE-SP) Atenção: Para responder à próxima questão, considere as declarações em pseudocódigo abaixo.

Considere que * indica ponteiro ou apontador.

Tipo tipoNo = registro

```
    info: inteiro
    *prox: tipoNo
```

fim registro

Var *inicio, *ant, *aux, *novo, *fim: tipoNo

Função Fila1 (info: inteiro)

Início

```
    novo ← aloca
    (*tipoNo) novo->info
    ← info
    novo->prox ← NULO
```



```

se (inicio = NULO)
  então
    inicio ←
    novo fim ←
    novo
    inicio ← novo
    fim ← novo

  senão
    fim ← novo
    aux ← inicio
    enquanto (aux NULO) faça
      ant ← aux
      aux ← aux->prox
    fim enquanto
    ant->prox ← novo
  fim se
Fim

Função Fila2(info: inteiro)
Início
  se (inicio = NULO)
    então
      imprima ("Fila vazia")

    senão
      aux ← inicio
      inicio ← inicio->prox
    se (inicio = NULO)
      fim ← NULO;
    fim se

    libera (aux)
  fim se
Fim

```

As funções Fila1 e Fila2 implementam operações em filas. Além das filas, há diversas outras estruturas muito úteis na solução de problemas, dentre as quais encontram-se as

- pilhas, também conhecidas como listas FIFO (First In, First Out).
- deques, que são pilhas que permitem inserir e remover dados em ambas as extremidades.
- árvores n-árias, estruturas de dados lineares que não são adequadas para representar dados que devem ser dispostos de maneira hierárquica, como diretórios criados em um computador.
- árvores binárias de busca, cujas funções que realizam percursos são naturalmente implementadas usando-se recursividade.
- árvores binárias平衡adas, nas quais, para cada nó, as alturas de suas subárvores diferem de no máximo, 2. Nelas, o custo das operações depende da altura da árvore, por isso elas devem ter a maior altura possível.

3.14 Comentários:

Vamos analisar item a item:

- pilhas, também conhecidas como listas FIFO (First In, First Out). Pilhas seguem a regra LIFO (last-in first-out), ou seja, o último a entrar é o primeiro a sair. Falso.
- deques, que são pilhas que permitem inserir e remover dados em ambas as extremidades. Deques são listas, e não pilhas, que permitem inserir e remover dados em ambas as extremidades. Falso.



- c) árvores n-árias, estruturas de dados lineares que não são adequadas para representar dados que devem ser dispostos de maneira hierárquica, como diretórios criados em um computador.

Árvores são estruturas de dados hierárquicas, e não lineares. Portanto, são adequadas para representar dados de maneira hierárquica. Falso.

- d) árvores binárias de busca, cujas funções que realizam percursos são naturalmente implementadas usando-se recursividade.

Normalmente, a forma de percorrer as árvores, especialmente as binárias – em que sabemos que um nó necessariamente possui apenas dois nós filhos – é por meio de recursão. Certo.

- e) árvores binárias balanceadas, nas quais, para cada nó, as alturas de suas subárvores diferem de, no máximo,

2. Nelas, o custo das operações depende da altura da árvore, por isso elas devem ter a maior altura possível.

As árvores não podem ter a maior altura possível, e, sim, a menor altura possível. Errado.

Gabarito: Letra D

16. (FGV – 2022 – SEFAZ-AM) A estrutura de dados usada em índices multiníveis dinâmicos em banco de dados relacionais, que garantem que tais estruturas sempre estejam balanceadas e que o espaço desperdiçado pela exclusão de itens de dados, se houver, nunca se torne excessivo, é denominada

- a) fila.
- b) hash.
- c) bitmap.
- d) árvore B.
- e) árvore binária.

3.15 Comentários:

Vamos analisar item a item:

- a) fila

Fila é uma estrutura de dados linear que seja a regra FIFO, first-in first-out, ou seja, primeiro a entrar é o primeiro a sair. Não envolve balanceamento e não tem multiníveis. Falso.

- b) hash.

Estrutura de dados que usa uma função hash para transformar entradas variáveis em saídas fixas de um vetor. Não tem multiníveis dinâmicos ou balanceamento. Falso.

- c) bitmap.

Armazena bits em um vetor. Não envolve balanceamento. Falso.

- d) árvore B.

São um tipo de árvore binária de busca balanceada. As árvores B são justamente as estruturas usadas para implementar os índices multiníveis dinâmicos em bancos de dados relacionais. Verdadeiro.

- e) árvore binária.

Uma árvore binária não necessariamente é balanceada, portanto não é a resposta correta. Falso.

Gabarito: Letra D

17. (FGV – 2021 – IMBEL) Considere uma árvore B+ com as seguintes características.

- I. A raiz é uma folha ou um nó que contém, no mínimo, dois filhos.
- II. Cada nó diferente do nó raiz e das folhas possui no mínimo d filhos.
- III. Cada nó tem no máximo 2d filhos. Cada nó possui entre d-1 e 2d-1 chaves, exceto o raiz que possui entre 1 e 2d-1 chaves.
- IV. Somente os nós folhas contêm dados associados às chaves.



Assinale o número máximo de acessos necessários para localizar uma chave, com d=10, num universo de 10 milhões de chaves.

- a) 5
- b) 7
- c) 10
- d) 100
- e) 1.000

3.16 Comentários:

O número máximo de acessos é calculado pela fórmula $\log_d n$. Para d = 10 e n = 10.000.000, temos $\log_{10} 10000000 = 7$.

Gabarito: Letra B

18. (CESPE - 2012 - Banco da Amazônia - Técnico Científico - Administração de Dados) As operações de busca em uma árvore binária não a alteram, enquanto operações de inserção e remoção de nós provocam mudanças sistemáticas na árvore.

Comentários:

Perfeito! Operações de Busca não alteram nenhuma estrutura de dados. Já Operações de Inserção e Remoção podem provocar diversas mudanças estruturais.

Gabarito: Correto

19. (CETAP - 2010 - AL-RR - Analista de Sistemas - A) Uma árvore binária é aquela que tem como conteúdo somente valores binários.

Comentários:

Não! Uma árvore binária é aquela que tem, no máximo, grau 2!

Gabarito: Errado

20. (CESPE - 2012 - Banco da Amazônia - Técnico Científico - Administração de Dados) O tipo de dados árvore representa organizações hierárquicas entre dados.

Comentários:

Perfeito, observem que alguns autores tratam Tipos de Dados como sinônimo de Estruturas de Dados.

Gabarito: Correto

21. (CETAP - 2010 - AL-RR - Analista de Sistemas - B) Uma árvore é composta por duas raízes, sendo uma principal e a outra secundária.



Comentários:

Não, uma árvore possui somente um nó raiz!

Gabarito: Errado

22. (CESPE - 2010 - DETRAN-ES - Analista de Sistemas) Denomina-se árvore binária a que possui apenas dois nós.

Comentários:

Não, árvore binária é aquela em que cada nó tem, no máximo, dois filhos!

Gabarito: Errado

23. (FUNCAB - 2010 - SEJUS-RO - Analista de Sistemas - II) Árvores são estruturas de dados estáticas com sua raiz representada no nível um.

Comentários:

Não! Árvores são estruturas dinâmicas e sua raiz, em geral, é representada no nível 0 (mas depende de autor para autor).

Gabarito: Errado

24. (CESPE - 2009 - ANAC - Especialista em Regulação - Economia) Considerando-se uma árvore binária completa até o nível 5, então a quantidade de folhas nesse nível será 24.

Comentários:

Não! A quantidade de folhas em um determinado nível – considerando a raiz como nível 0 –, é dada pela fórmula 2^d , portanto $2^5 = 32$. ou “é dada pela fórmula 2^d , portanto $2^5 = 32$..”

Gabarito: Errado

25. (CESPE - 2009 – ANAC - Analista de Sistemas) Uma árvore binária completa até o nível 10 tem 2.047 nós.

Comentários:

Se possui 10 níveis, possui $2^{d+1}-1$: 2047 nós!” ou “Se possui 10 níveis, possui $2^{(d+1)} - 1$: 2047 nós!!”

Gabarito: Correto

26. (FGV – 2015 – DPE/MT – Analista de Sistemas) No desenvolvimento de sistemas, a escolha de estruturas de dados em memória é especialmente relevante. Dentre outras classificações, é possível agrupar essas estruturas em lineares e não lineares, conforme a quantidade de sucessores e



antecessores que os elementos da estrutura possam ter. Assinale a opção que apresenta, respectivamente, estruturas de dados lineares e não lineares.

- a) Tabela de dispersão e fila.
- b) Estrutura de seleção e pilha.
- c) Pilha e estrutura de seleção.
- d) Pilha e árvore binária de busca.
- e) Fila e pilha.

4 Comentários:

Além dessa classificação, existe outra também importante: Estruturas Lineares e Estruturas Não-Lineares. As Estruturas Lineares são aquelas em que cada elemento pode ter um único predecessor (exceto o primeiro elemento) e um único sucessor (exceto o último elemento). Como exemplo, podemos citar Listas, Pilhas, Filas, Arranjos, entre outros.

Já as Estruturas Não-Lineares são aquelas em que cada elemento pode ter mais de um predecessor e/ou mais de um sucessor. Como exemplo, podemos citar Árvores, Grafos e Tabelas de Dispersão. Essa é uma classificação muito importante e muito simples de entender. Pessoal, vocês perceberão que esse assunto é cobrado de maneira superficial na maioria das questões, mas algumas são nível doutorado!

Conforme vimos em aula, trata-se de pilha e árvore respectivamente.

Gabarito: Letra D

27. (CESPE - 2010 – TRE/MT – Analista de Sistemas – B) As listas, pilhas, filas e árvores são estruturas de dados que têm como principal característica a sequencialidade dos seus elementos.

Comentários:

Não! Árvores têm como principal característica a sequencialidade dos seus elementos.

Gabarito: Errado

28. (FCC - 2012 - MPE-AP - Analista Ministerial - Tecnologia da Informação – A) A árvore é uma estrutura linear que permite representar uma relação de hierarquia. Ela possui um nó raiz e subárvores não vazias.

Comentários:

Árvore é uma estrutura linear? Não, hierárquica!

Gabarito: Errado

29. (CESPE - 2010 – TRE/MT - Analista de Sistemas – E) O uso de recursividade é totalmente inadequado na implementação de operações para manipular elementos de uma estrutura de dados do tipo árvore.



Comentários:

Pelo contrário, é fundamental para implementação de operações.

Gabarito: Errado

30. (FCC - 2011 - TRT - 19ª Região (AL) - Técnico Judiciário - Tecnologia da Informação) Em uma árvore binária, todos os nós têm grau:

- a) 2.
- b) 0, 1 ou 2.
- c) divisível por 2.
- d) maior ou igual a 2.
- e) 0 ou 1.

5 Comentários:

Olha a pegadinha! Todos os nós têm grau 0 (Folha), 1 (Único filho) ou 2 (Dois filhos).

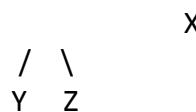
Gabarito: Letra B

31. (CESPE - 2011 – STM – Analista de Sistemas) Enquanto uma lista encadeada somente pode ser percorrida de um único modo, uma árvore binária pode ser percorrida de muitas maneiras diferentes.

Comentários:

Galera, pense em uma árvore bem simples com um pai (raiz) e dois filhos. O Modo Pré-fixado vai ler primeiro a raiz, depois a sub-árvore da esquerda e depois a sub-árvore da direita. O Modo In-fixado vai ler primeiro a sub-árvore da esquerda, depois a raiz e depois a sub-árvore da direita. O Modo Pós-fixado vai ler primeiro a sub-árvore da esquerda, depois a sub-árvore da direita e depois a raiz.

Vamos resumir: o modo de percorrimento de uma árvore pode obedecer três regras de acordo com a posição da raiz (pai): pré-fixado (raiz, esquerda, direita); in-fixado (esquerda, raiz, direita); e pós-fixado (esquerda, direita, raiz). Vamos agora ver uma árvore de exemplo:



- Percorrimento Pré-fixado: X Y Z
- Percorrimento In-fixado: Y X Z
- Percorrimento Pós-fixado: Y Z X

Logo, uma árvore pode ser percorrida de modo pré-fixado, in-fixado e pós-fixado.

Gabarito: Correto



32. (FCC - 2016 - Prefeitura de Teresina - PI - Analista Tecnológico - Analista de Suporte Técnico)

Considerando a estrutura de dados denominada árvore,

- a) a sua altura é definida como a profundidade média de todos os seus vértices.
- b) um vértice com um ou dois filhos é denominado folha.
- c) cada nó tem no mínimo dois filhos em uma árvore binária.
- d) as folhas de uma árvore binária completa podem ter profundidades distintas entre si.
- e) a profundidade de um vértice em uma árvore é definida como o comprimento da raiz da árvore até esse vértice.

6 Comentários:

(a) Errado! Altura é definida pela folha mais profunda; (b) Errado! Folhas não possuem filhos, do contrário não seriam folhas (as arves... somo nozes... péssimo, professor :P); (c) Errado! Os nós de uma árvore binária podem ter NO MÁXIMO dois filhos, e não no mínimo; (d) Errado! Uma árvore binária completa é aquele em que todos os nós internos possuem seus dois filhos (máximo). Desse modo, todas as folhas devem ter a mesma profundidade; (e) Certo!

Gabarito: Letra E

33. (CESPE – 2017 – TRE/BA - Analista de Sistemas) No estabelecimento de uma estrutura hierárquica, foi definida a seguinte árvore binária S:

$$S = (12(10(9(8))(11))(14(13)(15)))$$

7 Considerando o resultado da operação de exclusão do nó 12, assinale a opção que corresponde a nova estrutura da árvore S.

- a) $(10(9(8))(11(14(13)(15)))$
- b) $(11(9(8)(10))(14(13)(15)))$
- c) $(11(10(9(8)))(14(13)(15)))$
- d) $(13(10(9)(11))(14(15)))$
- e) $(13(11(9)(10))(14(15)))$

8 Comentários:

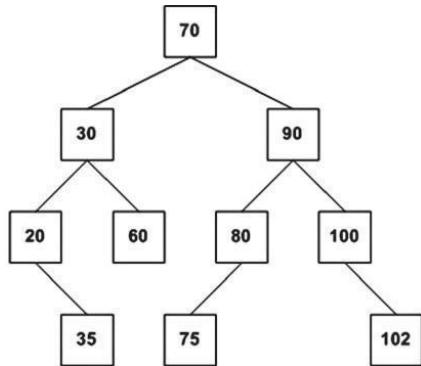
Conforme vimos em aula, a questão já inicia com um problema grave: ela não especifica qual tipo de árvore. Ainda assim, ela está errada e é fácil descobrir que não pode ser a Letra C (Gabarito Preliminar) porque a quantidade de parênteses abertos e fechados são diferentes – logo jamais poderia ser essa a resposta. Concluímos, então, que essa questão não possui resposta alguma, visto que falta fechar um parêntese. A Letra C $(11(10(9(8)))(14(13)(15)))$ está quase certa, mas faltou um parêntese: $(11(10(9(8)))(14(13)(15)))$.

Gabarito: Anulada

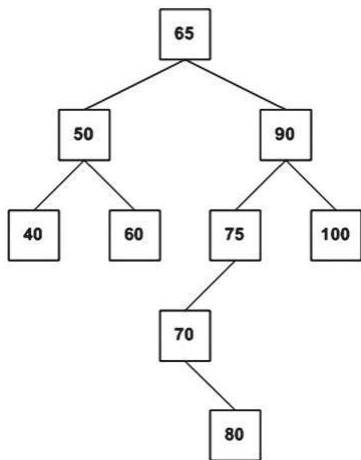


34. (CESGRANRIO – 2012 – PETROBRÁS - Analista de Sistemas) Qual figura representa uma árvore AVL?

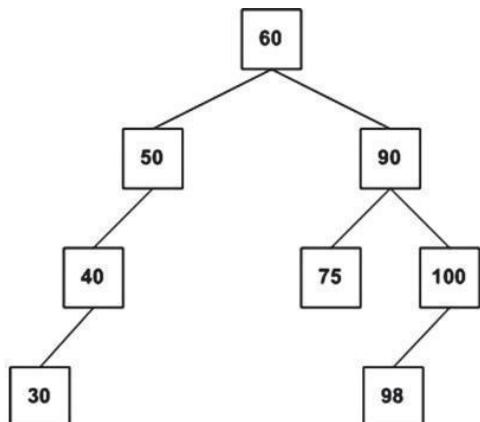
a)



b)

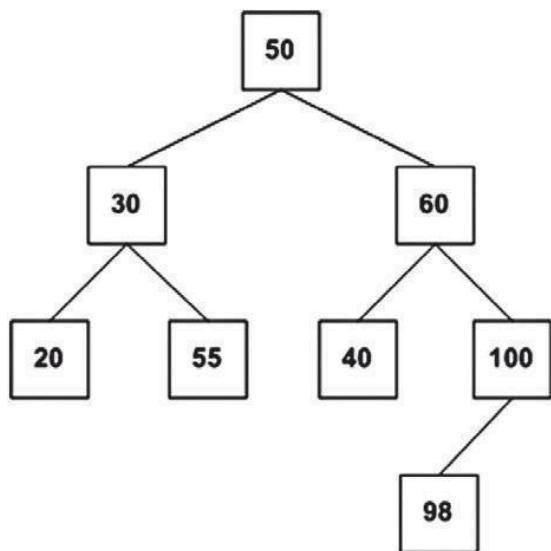


c)

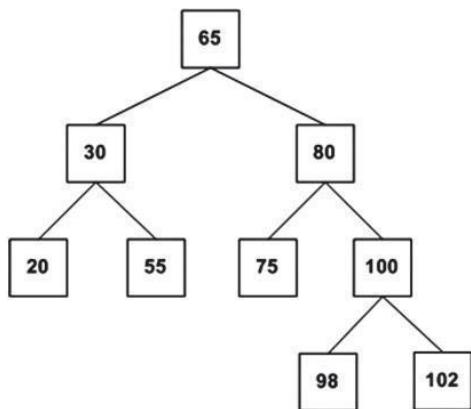


d)





e)



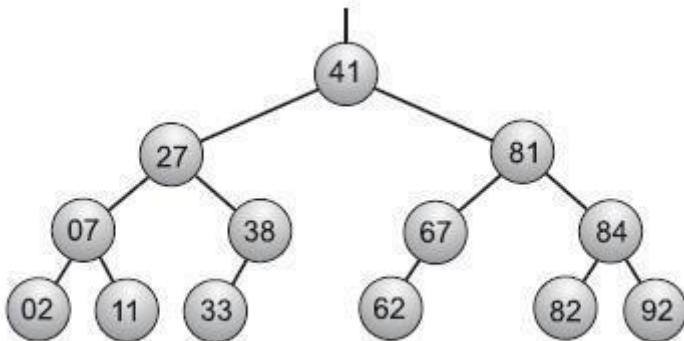
9 Comentários:

Vamos relembrar dois conceitos: (1) Uma Árvore AVL é uma árvore binária de busca em que, para todos os seus nós, a diferença da altura da subárvore da esquerda para a altura da subárvore da direita deve ser no máximo 1. (2) Uma Árvore Binária de Busca é aquela em que, para todos os nós, a subárvore da esquerda possui um valor menor que a subárvore da direita. Dito isso, vamos analisar agora as opções:

- (a) Errado. Não é uma árvore binária de busca, visto que o 35 é maior que 30 e está na subárvore da esquerda;
- (b) Errado. Não é uma árvore binária de busca, visto que o 80 é maior que 75 e está na subárvore da esquerda;
- (c) Errado. Observem que se trata realmente de uma árvore binária de busca, porém está desbalanceada. A diferença de altura da subárvore da esquerda de 50 e a subárvore da direita de 50 é 2 (que é maior do 1), portanto não é uma Árvore AVL.
- (d) Errado. Não é uma árvore binária de busca, visto que o 55 é maior que 50 e está na subárvore da esquerda;



35. (CESGRANRIO – 2006 – DECEA - Analista de Sistemas) Suponha a seguinte árvore AVL.



A inserção do elemento 30 nessa árvore:

- a) aumenta a profundidade da árvore após uma rotação.
- b) provoca uma rotação à direita.
- c) deixa os nós 02 e 07 no mesmo nível.
- d) altera a raiz da árvore (nó 41).
- e) torna o nó 33 pai do nó 27.

10 Comentários:

Vamos lá! A questão quer inserir 30. Onde ele entraria? À esquerda do 33! No entanto, isso tornaria a árvore desbalanceada. Por que? Porque a árvore à esquerda do nó 38 teria altura 2 e o nó à direita do nó 38 teria altura 0 – a diferença seria 2, que é maior do que 1. Podemos notar que se trata de um Caso Esquerda- Esquerda, logo temos que fazer uma rotação simples para direita.

Rotacionamos o Ramo 38-33 para direita. Dessa forma, teríamos o 33 no lugar do 38 e o 38 à direita do 33. Pronto! Agora vamos analisar as opções: (a) Errado. A profundidade permanece a mesma; (b) Correto. Foi isso que nós fizemos; (c) Errado. Isso não faz nenhum sentido; (d) Errado. Isso também não faz nenhum sentido; (e) Errado. Também nenhum sentido.

36. (CESPE – 2012 – TJ/RO - Analista de Sistemas) Assinale a opção em que é apresentado exemplo de estrutura de informação do tipo abstrata, balanceada, não linear e com relacionamento hierárquico.

- a) lista duplamente encadeada
- b) árvore binária
- c) pilha
- d) árvore AVL



e) deque

11 Comentários:

Tipo abstrato? Todos são! Não Linear? Árvore Binária e Árvore AVL! Com relacionamento hierárquico? Árvore Binária e Árvore AVL! Balanceada? Somente a Árvore AVL.

Gabarito: Letra D

37. (FCC – 2008 – TRT/18 - Analista de Sistemas) Árvore AVL balanceada em altura significa que, para cada nó da árvore, a diferença entre as alturas das suas sub- árvores (direita e esquerda) sempre será:

- a) menor ou igual a 2.
- b) igual a 0 ou -1.
- c) maior que 1.
- d) igual a 1.
- e) igual a -1, 0 ou 1.

12 Comentários:

Por fim, vamos falar um pouco sobre Árvores AVL! Uma Árvore AVL (Adelson-Vesky e Landis) é uma Árvore Binária de Busca em que, para qualquer nó, a altura das subárvore da esquerda e da direita não podem ter uma diferença maior do que 1, portanto uma Árvore AVL é uma Árvore Binária de Busca autobalanceável. Calma que nós vamos ver isso em mais detalhes...

Conforme vimos em aula, trata-se da última opção. Lembrando que, na aula, nós falamos que era a diferença não podia ser maior que 1. Uma outra forma de escrever isso é dizer que o módulo da diferença entre as alturas não pode ser maior do que 1. E outra forma de escrever isso é dizer que um nó só poder ter uma diferença de altura de suas subárvore de 1, 0 ou -1. Certinho?

Gabarito: Letra E

38. (CESGRANRIO – 2010 – PETROBRÁS - Analista de Sistemas) Uma árvore AVL é uma estrutura de dados muito usada para armazenar dados em memória. Ela possui algumas propriedades que fazem com que sua altura tenha uma relação muito específica com o número de elementos nela armazenados. Para uma folha, cuja altura é igual a um, tem-se uma árvore AVL com 6 nós.

Qual é a altura máxima que esta árvore pode ter?

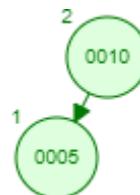
- a) 6
- b) 5
- c) 4
- d) 3
- e) 2

13 Comentários:

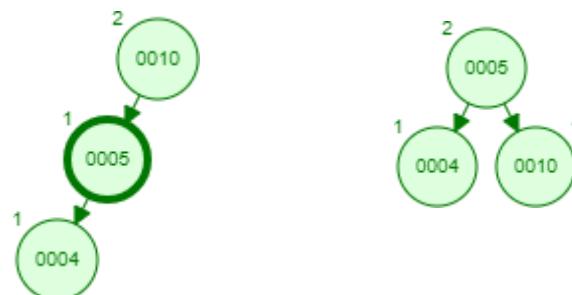


Galera, vocês precisam saber uma coisa: uma minoria dos autores considera que as folhas de uma árvore possuem altura 1 (sendo que a maioria considera que a altura de uma folha é 0) – a questão afirma que, para uma folha, a altura é igual a 1. Dito isso, vamos analisar a questão! Ela te deu uma regra: você tem que construir uma Árvore AVL com 6.

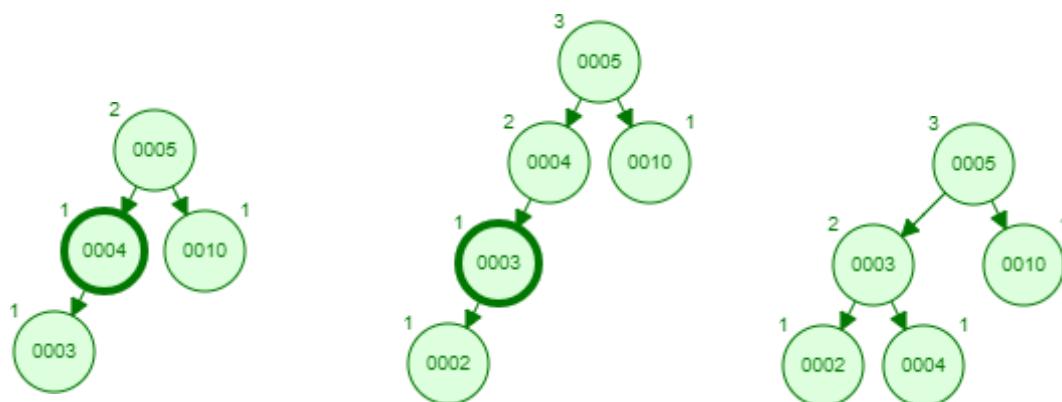
Em outras palavras, você tem que tentar construir a árvore com a maior altura possível, mas ela tem que estar balanceada. Então, eu começo com dois nós:



Como quero atingir a altura máxima, coloco mais um nó abaixo de 0005 e a árvore ficará com altura 3, mas ficará desbalanceada. Após balancear, fica como na direita:

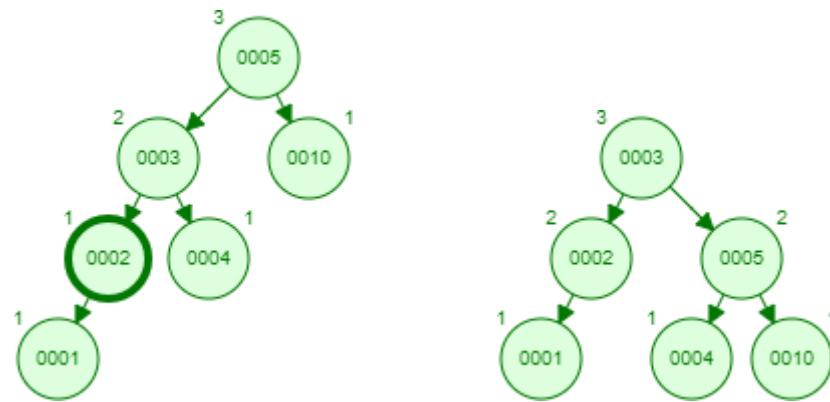


Preciso encontrar a altura máxima, então coloco mais um nó (0003) e me sobrará mais duas tentativas. Se eu colocar mais um abaixo de 0003, ficará desbalanceado:



Vejam na imagem acima! Coloquei 0003 e não desbalanceou; coloquei 0002, desbalanceou e na última eu tive que rebalancear. Vamos inserir o último:





Pronto! Com seis nós, a altura máxima que atingiremos será 3. Portanto, a resposta para nossa questão é Letra D.

Gabarito: Letra D

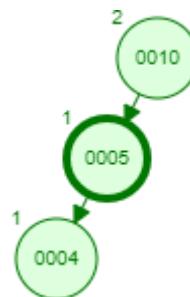
39. (CESGRANRIO – 2011 – PETROBRÁS - Analista de Sistemas) Uma árvore AVL é uma árvore binária de busca autobalanceada que respeita algumas propriedades fundamentais. Como todas as árvores, ela tem uma propriedade chamada altura, que é igual ao valor da altura de sua raiz.

Sabendo que a altura de uma folha é igual a um e que a altura de um nó pai é igual ao máximo das alturas de seus filhos mais um, qual estrutura NÃO pode representar uma árvore AVL?

- a) Uma árvore vazia
- b) Uma árvore com dois nós
- c) Uma árvore com três nós e altura igual a dois
- d) Uma árvore com três nós e altura igual a três
- e) Uma árvore com seis nós e altura igual a três

14 Comentários:

(a) Errado, uma árvore vazia é uma Árvore AVL; (b) Errado, é impossível uma árvore com dois nós não ser uma Árvore AVL; (c) Errado, uma árvore com três nós e altura igual a dois é perfeitamente balanceada; (d) Correto, uma árvore com três nós e altura igual a três não pode estar balanceada (vide imagem abaixo); (e) Errado, uma árvore com seis nós e altura igual a três também está平衡ada. Gabarito: D.



Gabarito: Letra D



40. (CESGRANRIO – 2011 – PETROBRÁS - Analista de Sistemas) Após a inserção de um nó, é necessário verificar cada um dos nós ancestrais desse nó inserido, relativamente à consistência com as regras estruturais de uma árvore AVL.

PORQUE

15 O fator de balanceamento de cada nó, em uma árvore AVL, deve pertencer ao conjunto formado por {-2, -1, 0, +1, +2}.

16 Analisando-se as afirmações acima, conclui-se que:

- a) as duas afirmações são verdadeiras, e a segunda justifica a primeira.
- b) as duas afirmações são verdadeiras, e a segunda não justifica a primeira.
- c) a primeira afirmação é verdadeira, e a segunda é falsa.
- d) a primeira afirmação é falsa, e a segunda é verdadeira.
- e) as duas afirmações são falsas.

17 Comentários:

Galera, a primeira frase está perfeita! Após inserir um novo nó, você tem que verificar os nós ancestrais para se certificar de que a Árvore AVL continua balanceada. No entanto, o fator de balanceamento de cada nó deve pertencer ao conjunto formado por {-1, 0, 1}. Lembrem-se: o módulo da diferença jamais pode ser maior do que 1, portanto a primeira está verdadeira e a segunda falsa.

Gabarito: Letra C

41. (CESGRANRIO – 2010 – EPE - Analista de Sistemas) Um programador decidiu utilizar, em determinado sistema de análise estatística, uma árvore AVL como estrutura de dados. Considerando-se n a quantidade de elementos dessa árvore, o melhor algoritmo de pesquisa, com base em comparações, possui complexidade de tempo, no pior caso, igual a:

- a) $O(1)$
- b) $O(\log n)$.
- c) $\Omega(n)$
- d) $\Omega(n \log n)$
- e) $\Omega(n^2)$

18 Comentários:

| ÁRVORE B / ÁRVORE AVL | | |
|-----------------------|-------------|-------------|
| ALGORITMO | CASO MÉDIO | PIOR CASO |
| Espaço | $O(n)$ | $O(n)$ |
| Busca | $O(\log n)$ | $O(\log n)$ |
| Inserção | $O(\log n)$ | $O(\log n)$ |
| Remoção | $O(\log n)$ | $O(\log n)$ |



Questão tranquila! Trata-se do $O(\log n)$.

Gabarito: Letra B

42. (CESGRANRIO – 2012 – PETROBRÁS - Analista de Sistemas) Todos os N nomes de uma lista de assinantes de uma companhia telefônica foram inseridos, em ordem alfabética, em três estruturas de dados: uma árvore binária de busca, uma árvore AVL e uma árvore B.

As alturas resultantes das três árvores são, respectivamente,

- a) $O(\log(N))$, $O(\log(N))$, $O(1)$
- b) $O(\log(N))$, $O(N)$, $O(\log(N))$
- c) $O(N)$, $O(\log(N))$, $O(1)$
- d) $O(N)$, $O(\log(N))$, $O(\log(N))$
- e) $O(N)$, $O(N)$, $O(\log(N))$

19 Comentários:

| ÁRVORE BINÁRIA DE BUSCA | | |
|-------------------------|-------------|-----------|
| ALGORITMO | CASO MÉDIO | PIOR CASO |
| Espaço | $O(n)$ | $O(n)$ |
| Busca | $O(\log n)$ | $O(n)$ |
| Inserção | $O(\log n)$ | $O(n)$ |
| Remoção | $O(\log n)$ | $O(n)$ |

| ÁRVORE B / ÁRVORE AVL | | |
|-----------------------|-------------|-------------|
| ALGORITMO | CASO MÉDIO | PIOR CASO |
| Espaço | $O(n)$ | $O(n)$ |
| Busca | $O(\log n)$ | $O(\log n)$ |
| Inserção | $O(\log n)$ | $O(\log n)$ |
| Remoção | $O(\log n)$ | $O(\log n)$ |

Questão tranquila! Trata-se do $O(n)$, $O(\log n)$ e $O(\log n)$ respectivamente.

Gabarito: Letra D

43. (IBFC – 2014 – TRE/AM - Analista de Sistemas) Quanto ao Algoritmo e estrutura de dados no caso de árvore AVL (ou árvore balanceada pela altura), analise as afirmativas abaixo, dê valores Verdadeiro (V) ou Falso (F) e assinale a alternativa que apresenta a sequência correta de cima para baixo:

() Uma árvore AVL é dita balanceada quando, para cada nó da árvore, a diferença entre as alturas das suas sub-árvore (direita e esquerda) não é maior do que um.

20 () Caso a árvore não esteja balanceada é necessário seu balanceamento através da rotação simples ou rotação dupla.



Assinale a alternativa correta:

- a) F-F
- b) F-V
- c) V-F
- d) V-V

21 Comentários:

A primeira alternativa está impecável, assim como a segunda. Vimos exaustivamente em aula!

Gabarito: Letra D

44. (CESGRANRIO – 2010 – PETROBRÁS - Analista de Sistemas) Uma sequência desordenada de números armazenada em um vetor é inserida em uma árvore AVL. Após a inserção nesta árvore, é feito um percurso em ordem simétrica (em ordem) e o valor de cada nó visitado é inserido em uma pilha. Depois de todos os nós serem visitados, todos os números são retirados da pilha e apresentados na tela. A lista de números apresentada na tela está:

- a) ordenada ascendentemente de acordo com os números.
- b) ordenada descendente mente de acordo com os números.
- c) na mesma ordem do vetor original.
- d) na ordem inversa do vetor original.
- e) ordenada ascendentemente de acordo com sua altura na árvore.

22 Comentários:

Essa questão é legal! Olha só...

Eu tenho um vetor com um bocado de valor desordenado. Esses valores são colocados em uma Árvore AVL. Ora, uma árvore AVL é uma árvore binária de busca, portanto segue suas propriedades. Logo, não importa se estava desordenado. À medida que são inseridos os valores desordenados na árvore, ela vai se balanceando e ordenando os dados.

Após isso, vamos retirar os dados da Árvore AVL. Como? Da esquerda para a direita! E vamos colocá-los em uma pilha. Se estamos lendo da esquerda para a direita, estamos retirando do menor valor para o maior valor, logo o maior valor da pilha será o maior valor da Árvore AVL. Por fim, ao retirar os elementos da pilha, retiramos do topo (maior) para a base (menor), logo em ordem descendente.

Gabarito: Letra B

45. (FGV – 2009 – MEC - Analista de Sistemas) Acerca das estruturas de dados Árvores, analise as afirmativas a seguir.

- I. A árvore AVL é uma árvore binária com uma condição de balanço, porém não completamente balanceada.



- II. Árvores admitem tratamento computacional eficiente quando comparadas às estruturas mais genéricas como os grafos.
- III. Em uma Árvore Binária de Busca, todas as chaves da subárvore esquerda são maiores que a chave da raiz.

23 Assinale:

- a) se somente a afirmativa I estiver correta.
- b) se somente as afirmativas I e II estiverem corretas.
- c) se somente as afirmativas I e III estiverem corretas.
- d) se somente as afirmativas II e III estiverem corretas.
- e) se todas as afirmativas estiverem corretas.

24 Comentários:

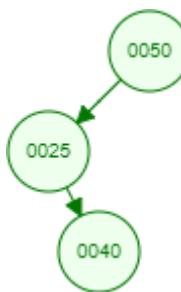
(I) Correto, ela é não é completamente balanceada; (II) Correto, isso é verdade! (III) Errado, são menores que a chave da raiz.

Gabarito: Letra B

46. (CESPE – 2014 – TJ/SE - Analista de Sistemas) Em uma árvore AVL (Adelson-Velsky e Landis), caso a diferença de altura entre as sub-árvores de um nó seja igual a 2 e a diferença de altura entre o nó filho do nó desbalanceado seja igual a -1, deve-se realizar uma rotação dupla com o filho para a direita e o pai para a esquerda a fim de que a árvore volte a ser balanceada.

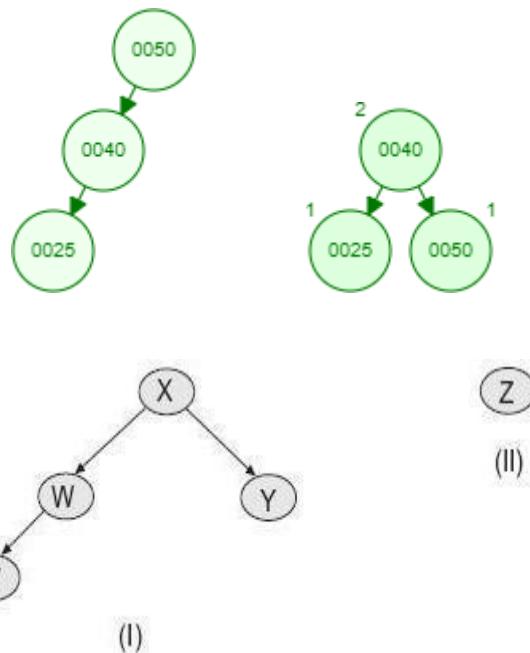
Comentários:

Vamos entender a questão! Nós temos um nó (0050) cujo fator de平衡amento é 2. Isso significa que a altura da subárvore à esquerda desse nó (2) menos a altura da subárvore à direita (0) é igual a 2 [2-0 = 2]. O nó filho (0025) desse nó pai (0050) está com balanceamento igual a -1, visto que a altura da subárvore à esquerda desse nó (0) menos a altura da subárvore à direita (1) é igual a -1 [0-1 = -1].



Logo, temos um Caso Esquerda-Direita. E sabemos que para balancear essa árvore, devemos fazer uma rotação dupla: primeiro à esquerda no Ramo 0025-0040 (primeira imagem) e depois à direita no Ramo 0025- 0050 (segunda imagem), portanto é exatamente o oposto do que a questão diz. No entanto, o Gabarito Definitivo foi Correto! Sinceramente, não faço ideia de onde o CESPE tirou isso...





Gabarito: Letra C

47. (CESPE – 2010 – PETROBRÁS - Analista de Sistemas) As árvores usadas como estruturas de pesquisa têm características especiais que garantem sua utilidade e propriedades como facilidade de acesso aos elementos procurados em cada instante. A esse respeito, considere as afirmações abaixo.

- I - A árvore representada na figura (I) acima não é uma árvore AVL, pois as folhas não estão no mesmo nível.
- II - A sequência 20, 30, 35, 34, 32, 33 representa um percurso sintaticamente correto de busca do elemento 33 em uma árvore binária de busca.
- III - A árvore representada na figura (II) acima é uma árvore binária, apesar da raiz não ter filhos.

25 É (São) correta(s) APENAS a(s) afirmativa(s):

- a) I.
- b) II.
- c) III.
- d) I e II.
- e) II e III.

26 Comentários:

(I) Errado. Trata-se, sim, de uma Árvore AVL; (II) Correto. Temos a sequência 20, 30, 35, 34, 32, 33 e estamos procurando o 33. 33>20, logo descemos à direita; 33>30, logo descemos à direita de novo; 33<35, logo descemos à esquerda; 33<34, logo descemos à esquerda de novo; 33>32, logo descemos à direita. Pronto, encontramos o 33; (III) Correto. Trata-se de um Árvore Binária sem filhos.

Gabarito: Letra E



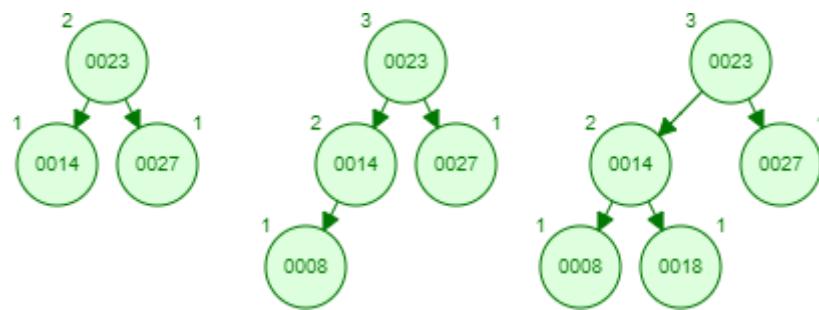
48. (CESPE – 2010 – PETROBRÁS - Analista de Sistemas) No sistema de dados do Departamento de Recursos Humanos de uma grande empresa multinacional, os registros de funcionários são armazenados em uma estrutura de dados do tipo árvore binária AVL, onde cada registro é identificado por uma chave numérica inteira. Partindo de uma árvore vazia, os registros cujas chaves são 23, 14, 27, 8, 18, 15, 30, 25 e 32 serão, nessa ordem, adicionados à árvore.

Dessa forma, o algoritmo de inserção na árvore AVL deverá realizar a primeira operação de rotação na árvore na ocasião da inserção do elemento:

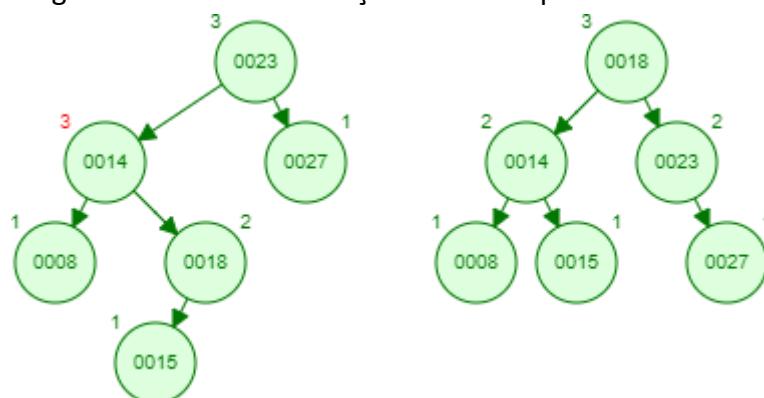
- a) 30
- b) 25
- c) 18
- d) 15
- e) 8

27 Comentários:

Vamos simular! Vamos inserir logo os três primeiros: 23, 14 e 27 (primeira imagem); depois inserimos o número 8 (segunda imagem); e inserimos o número 18 (terceira imagem). Vejamos como ficou:



Agora vamos inserir o 15! Esse nó iria para a esquerda de 18 como mostra a primeira imagem abaixo. Nesse caso, o nó 23 ficaria desbalanceado. O que fazer? Temos um Caso Esquerda-Direita, portanto temos que fazer uma rotação dupla: primeiro à esquerda no Ramo 14-18 e depois à direita no Ramo 18-23. O resultado é mostrado a imagem abaixo e é na inserção do nó 15 que devemos fazer a primeira rotação.



Gabarito: Letra D



49. (CESPE – 2014 – TJ/SE - Analista de Sistemas) Existem dois vetores, chamados A e B, que estão ordenados e contêm N elementos cada, respeitando a propriedade $A[N-1] < B[0]$, onde os índices de ambos os vetores vão de 0 a N-1. Retiram-se primeiro todos os elementos de A na ordem em que se apresentam e inserem-se esses elementos em uma árvore binária de busca, fazendo o mesmo depois com os elementos de B, que são inseridos na mesma árvore de busca que os de A. Depois, retiram-se os elementos da árvore em um percurso pós ordem, inserindo-os em uma pilha. Em seguida retiram-se os elementos da pilha, que são inseridos de volta nos vetores, começando pelo elemento 0 do vetor A e aumentando o índice em 1 a cada inserção, até preencher todas as N posições, inserindo, então, os N elementos restantes no vetor B da mesma maneira.

Ao final do processo, tem-se que os vetores:

- a) estão ordenados e $A[i] < B[i]$, para todo $i=0,.., N-1$.
- b) estão ordenados e $A[i] > B[i]$, para todo $i=0,.., N-1$.
- c) estão ordenados e não existe mais uma propriedade que relate A[i] e B[i].
- d) não estão ordenados e $A[i] < B[i]$, para todo $i=0,.., N-1$.
- e) não estão ordenados e $A[i] > B[i]$, para todo $i=0,.., N-1$.

28 Comentários:

Ele diz que nós temos dois vetores em que $A[N-1] < B[0]$. Então, vamos imaginá-los aqui:

- Vetor A = [1, 3, 5]
- Vetor B = [7, 9, 11]

Depois ele diz que são retirados todos os elementos de A na ordem que se apresentam e são inseridos em uma árvore binária de busca (lembre-se que uma árvore binária de busca é aquela em que todos os nós da subárvore esquerda possuem um valor numérico menor que o da raiz e os nós da subárvore direita possuem um valor numérico maior que o da raiz). Se você desenhar essa árvore, vai perceber que ela vai ficar em ordem toda para a direita - sem nenhum elemento para esquerda. Dito isso, ficou:

- 1, 3, 5, 7, 9, 11.

Depois ele disse que os elementos foram retirados da árvore em pós-ordem, ou seja, subárvore à esquerda, depois subárvore à direita e só depois raiz. Portanto, não tem elemento na esquerda, você retira o elemento da direita e depois a raiz. E são colocados em uma pilha, logo ficaria:

- 11, 9, 7, 5, 3, 1

Lembrando que numa pilha você insere sempre elementos no topo, logo 1 seria o topo da pilha. Depois ele diz que você retira os elementos da pilha (também sempre pelo topo) e coloca de volta nos vetores. Logo, ficaria:



- Vetor A = [1, 3, 5]

- Vetor B = [7, 9, 11]

Pronto! Note que fica exatamente a mesma coisa e os vetores ficariam ordenados e $A[i] < B[i]$, para todo $i=0, \dots, N-1$. Bacana?

Gabarito: Letra A



QUESTÕES COMENTADAS - GRAFOS - MULTIBANCAS

1. (CESPE/CEBRASPE – 2017 – TRF-1) Acerca dos conceitos de árvores e grafos, julgue o item que se segue.

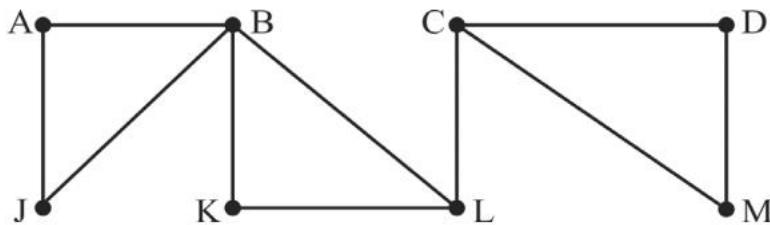
A soma dos graus de todos os vértices de um grafo é sempre par.

Comentários:

Um vértice é um nó do grafo. Já o grau de um vértice é o número de arestas do vértice. Cada aresta conta dois graus para a soma total, sendo um grau para cada vértice que ela conecta. Por isso, a soma será sempre par.

Gabarito: Certo

2. (CESPE/CEBRASPE – 2018 – PF)



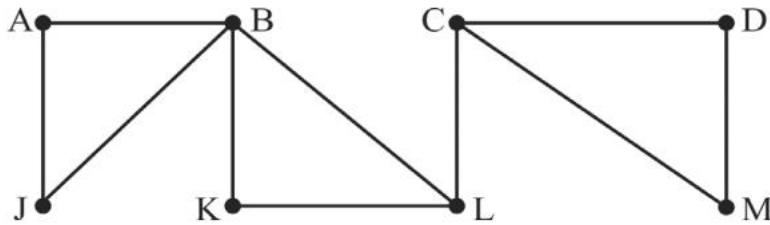
Considerando a terminologia e os conceitos básicos de grafos, julgue o item a seguir, relativo ao grafo precedente. No grafo em apreço, existem três ciclos com comprimento quatro: AJBA, BKLB e CDMC.

Comentários:

Se observar, os ciclos que ligam os pontos AJBA, BKLB e CDMC possuem 3 arestas cada. Significa que o comprimento deles é “três”, e não “quatro”.

Gabarito: Errado

3. (CESPE/CEBRASPE – 2018 – PF)



Considerando a terminologia e os conceitos básicos de grafos, julgue o item a seguir, relativo ao grafo precedente. O grafo em questão tem diâmetro igual a quatro.

Comentários:

O diâmetro de um grafo corresponde à maior distância mínima entre um par de vértices. Para calculá-lo, é necessário achar os menores caminhos entre cada par de vértices. O maior desses caminhos será o diâmetro. No caso, você pode notar que, neste grafo, os vértices que estão mais distantes entre si são os vértices A ou J, para os vértices D ou M. E existem caminhos mínimos entre eles, por exemplo:

- ABLCD: tamanho 4
- JBLCD: tamanho 4
- ABLCM: tamanho 4
- JBLCM: tamanho 4

Existem caminhos maiores, se eu quiser fazer, por exemplo:

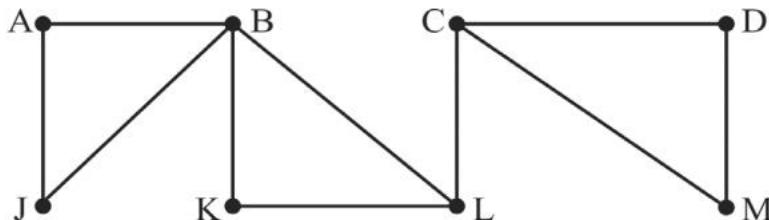


- ABKLCD: tamanho 5
- JBLLCD: tamanho 5
- ABKLCDM: tamanho 6
- JBLLCDM: tamanho 6

Porém o diâmetro leva em consideração a maior distância mínima entre os vértices. E as distâncias mínimas são de tamanho 4 – a maior que conseguimos encontrar. Portanto, o diâmetro do grafo é 4.

Gabarito: Certo

4. (CESPE/CEBRASPE – 2018 – PF)



Considerando a terminologia e os conceitos básicos de grafos, julgue o item a seguir, relativo ao grafo precedente. Os vértices A, B, C, D, J, K, L, M têm graus iguais, respectivamente, a 2, 4, 3, 2, 2, 2, 3, 2.

Comentários:

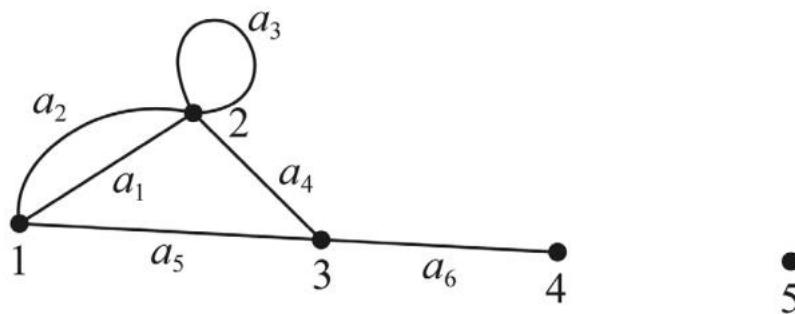
O grau de um vértice é a quantidade de arestas que estão ligadas a ele.

Portanto:

- A tem 2 arestas, grau 2.
- B tem 4 arestas, grau 4.
- C tem 3 arestas, grau 3.
- D tem 2 arestas, grau 2.
- J tem 2 arestas, grau 2.
- K tem 2 arestas, grau 2.
- L tem 3 arestas, grau 3.
- M tem 2 arestas grau 2.

Gabarito: Certo

5. (CESPE/CEBRASPE – 2018 – IFF)



Considerando o grafo precedente, assinale a opção correta.

- Os nós 1 e 4 são adjacentes.
- O nó 5 é adjacente a si mesmo.
- Os arcos a1 e a2 são arcos irmãos.
- Os nós 2 e 3 têm grau 3.
- O grafo não pode ser classificado como conexo.



Comentários:

Vamos analisar item a item:

- a) Os nós 1 e 4 são adjacentes.

Os nós 1 e 4 não são adjacentes. Para isso, eles deveriam estar sendo ligados por um arco diretamente, o que não é o caso. Falso.

- b) O nó 5 é adjacente a si mesmo.

Para que ele fosse adjacente a si mesmo, era necessário ter um arco realizando uma auto-ligação. Não é o caso. Falso.

- c) Os arcos a_1 e a_2 são arcos irmãos.

Nesta situação, em que há múltiplos arcos ligando um mesmo par de nós, temos um multigrafo, e não arcos irmãos. Falso.

- d) Os nós 2 e 3 têm grau 3.

Um grau de um nó corresponde a quantidade de arcos fazendo ligação nele. Portanto, o nó 2 possui grau 4, e o nó 3 grau 3. Falso.

- e) O grafo não pode ser classificado como conexo.

Um grafo conexo possui pelo menos um caminho entre todos os pares de nós. Neste grafo, não dá para chegar no nó 5. Então o grafo não é conexo. Verdadeiro.

Gabarito: Certo

6. (CESPE/CEBRASPE – 2017 – TRE-TO) A estrutura de dados formada por conjuntos de pontos (nós ou vértices) em um conjunto de linhas (arestas e arcos) que conectam vários pontos é denominada

- a) lista encadeada.
- b) fila circular.
- c) grafo.
- d) árvore.
- e) pilha.

Comentários:

Vamos de item a item:

- a) lista encadeada.

Sequência de elementos que apontam para o seu sucessor. Falso.

- b) fila circular.

Estrutura que se comporta como uma fila, mas que permite a inserção de elementos no final, e a remoção de elementos no início, de forma circular. Falso.

- c) grafo.

Grafo é uma estrutura de dados com um conjunto de vértices ou nós e arestas ou arcos que os ligam. Verdadeiro.

- d) árvore.

Estrutura em que cada nó possui um determinado número de filhos. Falso.

- e) pilha.

Estrutura de dados que segue a lógica LIFO, ou last-in first-out, que significa “o último a entrar é o primeiro a sair”. Falso.

Gabarito: Letra C

7. (CESPE/CEBRASPE – 2019 – TJ-AM) A respeito de lógica, estrutura e linguagem de programação, julgue o item seguinte.

Na estrutura do tipo grafo, cada elemento indica o próximo elemento, seja aquele que o antecede ou aquele que é seu sucessor, e cada elemento está associado a somente um antecessor e a vários sucessores.

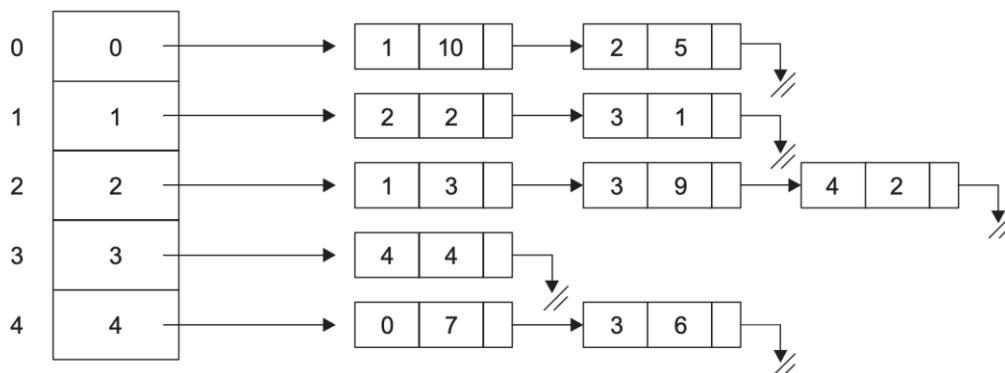
Comentários:



Em um grafo, cada vértice pode estar ligado a vários outros vértices por meio de arestas. Porém, os vértices em si não possuem relação de antecessor e sucessor. As arestas podem ou não ter direção – tornando o grafo direcionado ou não. Dessa forma, dizer que o elemento está associado somente a um antecessor e vários sucessores não é correta.

Gabarito: Errado

8. (FCC – 2017 – ARTESP) Considere a estrutura de dados abaixo.



Esta estrutura representa cinco localidades indicadas por 0, 1, 2, 3, 4 com as rotas e as respectivas distâncias entre elas.

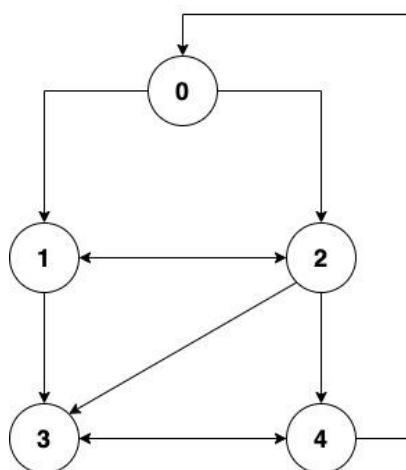
Por exemplo, da localidade 0 há rota para a localidade 1 (distância 10) e para a localidade 2 (distância 5). Um Especialista em

Tecnologia da Informação da ARTESP afirma, corretamente, que

- partindo de qualquer uma das localidades é possível ir para todas as outras e voltar para a localidade de origem.
- a distância da rota direta partindo de uma localidade x para uma localidade y não é a mesma da rota de retorno de y para x.
- a rota direta mais longa entre duas localidades é 9.
- a rota mais curta partindo da localidade 3 e chegando na localidade 2 é 9.
- é possível ir e voltar de todas as localidades adjacentes.

Comentários:

É possível desenhar as rotas da seguinte maneira:



Vamos analisar item a item:

- partindo de qualquer uma das localidades é possível ir para todas as outras e voltar para a localidade de origem.



É possível ir a qualquer uma das localidades para todas as outras, e voltar para a localidade de origem. A banca definiu este item como errado. Acredito que tenha considerado que é possível, mas não diretamente, e, também, não pelo mesmo caminho. Falso.

- b) a distância da rota direta partindo de uma localidade x para uma localidade y não é a mesma da rota de retorno de y para x.

As distâncias são diferentes, de fato. Por exemplo, de 0 para 1, a distância é 10. Mas, de 1 para 0, é possível fazer o caminho 1-3-4-0, cuja distância é $1+4+7 = 12$, ou o caminho 1-2-4-0, cuja distância é $2+2+7 = 11$. Verdadeiro.

- c) a rota direta mais longa entre duas localidades é 9.

A rota direta mais longa entre duas localidades é entre 0 e 1, cuja distância é 10. Falso.

- d) a rota mais curta partindo da localidade 3 e chegando na localidade 2 é 9.

Temos duas rotas possíveis:

- 3-4-0-2: distância $4+7+5 = 16$
- 3-4-0-1-2: distância $4+7+10+2 = 23$

Falso.

- e) é possível ir e voltar de todas as localidades adjacentes.

É possível ir e voltar, porém por caminhos diferentes. Falso.

Gabarito: Letra B

9. (FCC – 2017 – ARTESP) Nas rodovias paulistas os veículos pagam pedágio em função do número de eixos e da sua categoria. Há 15 categorias de veículos. Para realizar o cálculo do pedágio, existe uma tarifa mínima que é multiplicada por um valor relativo ao número de eixos. Considere a estrutura abaixo que indica a categoria do veículo pelo número da coluna; na primeira linha armazena a quantidade de eixos; na segunda linha armazena o valor pelo qual a tarifa mínima deve ser multiplicada.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|----|----|----|----|-----|
| 0 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 4 | 3 |
| 1 | 0 | 1 | 1 | 2 | 2 | 3 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 | 1,5 |

Exemplos: o veículo 0 é motocicleta/motoneta/bicicleta a motor que tem 2 eixos, mas é isento; o veículo 2 é caminhonete/furgão que tem 2 eixos e paga 1 tarifa; o veículo 13 é um caminhonete/automóvel com reboque que tem 4 eixos e paga 2 tarifas.

Considerando que n é a categoria do veículo, que tm é a tarifa mínima e que a estrutura é denominada mpedagio, o trecho em pseudocódigo que calcula vp, o valor pedágio, corretamente, é:

- a) $vp \leftarrow mpedagio[n,0] * mpedagio[n,1] * tm$
- b) $vp \leftarrow mpedagio[1,n] * tm$
- c) $vp \leftarrow vp + (mpedagio[0,n] + mpedagio[1,n]) * tm$
- d) $vp \leftarrow (mpedagio[n,0] / mpedagio[n,1]) * tm$
- e) se ($n = 0$) então $vp \leftarrow 0$ senão $vp \leftarrow (mpedagio[0,n] / 2) * tm$ fim se

Comentários:

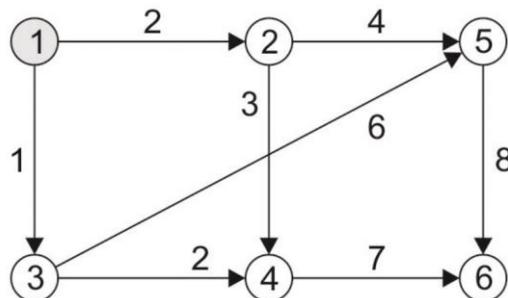
A primeira linha do mpedagio armazena a quantidade de eixos do veículo; já a segunda, armazena o valor pelo qual a tarifa deve ser multiplicada. Então, o valor do pedágio será justamente o valor na linha 1, na coluna n ($mpedagio[1,n]$), multiplicado por tm. Ou seja, $vp \leftarrow mpedagio[1,n] * tm$.

As demais alternativas estão incorretas.

Gabarito: Letra B



10. (FCC – 2017 – ARTESP) Considere a estrutura abaixo que representa um problema de rotas em pequena escala.



Considere, por hipótese, que solicitou-se a um Agente de Fiscalização à Regulação de Transporte da ARTESP utilizar alguma estratégia lógica para, partindo do ponto 1, chegar ao ponto 6 usando a menor rota. De um mesmo ponto pode haver mais de uma rota, com distâncias diferentes. A lógica correta utilizada pelo Agente, em função dos pontos a serem percorridos, foi

- a) $\{1\} \{2,3\} \{2,4\} \{5,6\}\{6\}$, caminho mais curto 1-2-5-6.
- b) $\{1\} \{2\} \{4\} \{6\}$, caminho mais curto 1-2-4-6.
- c) $\{1\} \{3,2\} \{4,5\} \{6\}$, caminho mais curto 1-3-4-6.
- d) $\{6\} \{5,4\} \{3,1\} \{1\}$, caminho mais curto 6-4-3-1, que é igual a 1-3-4-6.
- e) $\{6\} \{4\} \{5,3\} \{2,1\} \{1\}$, caminho mais curto 6-4-3-5-2-1, que é igual a 1-2-5-3-4-6.

Comentários:

Cada aresta tem um peso de distância. Vamos de item a item, calculando o tamanho dos caminhos, somando as distâncias das arestas, e ao final vamos ver qual é o menor.

- a) $\{1\} \{2,3\} \{2,4\} \{5,6\}\{6\}$, caminho mais curto 1-2-5-6.

Este caminho, 1-2-5-6, tem peso $2+4+8 = 14$.

- b) $\{1\} \{2\} \{4\} \{6\}$, caminho mais curto 1-2-4-6.

Este caminho, 1-2-4-6, tem peso $2+3+7 = 12$.

- c) $\{1\} \{3,2\} \{4,5\} \{6\}$, caminho mais curto 1-3-4-6.

Este caminho 1-3-4-6 tem peso $1+2+7 = 10$.

- d) $\{6\} \{5,4\} \{3,1\} \{1\}$, caminho mais curto 6-4-3-1, que é igual a 1-3-4-6.

O caminho 1-3-4-6 não é igual ao caminho 6-4-3-1, pois o caminho 6-4-3-1 é impossível, visto que se trata de um grafo direcional, e não existe caminho inverso. Alternativa inválida.

- e) $\{6\} \{4\} \{5,3\} \{2,1\} \{1\}$, caminho mais curto 6-4-3-5-2-1, que é igual a 1-2-5-3-4-6.

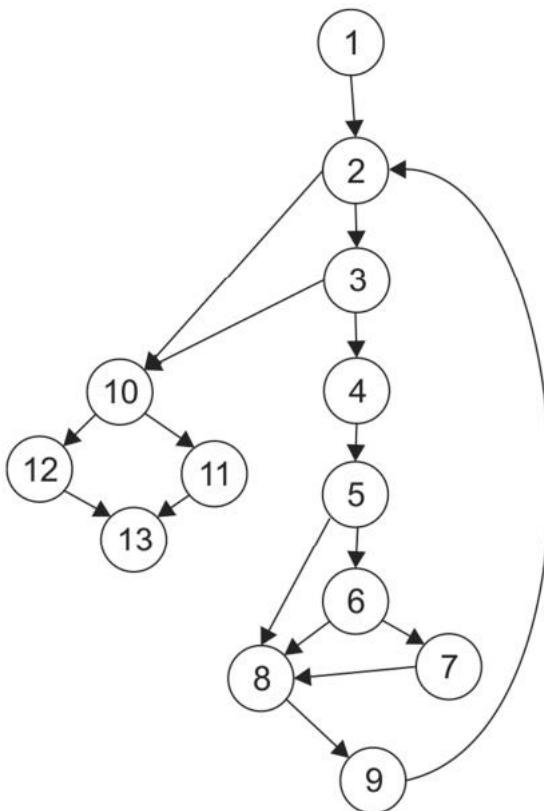
A mesma coisa do item anterior. Alternativa inválida.

Portanto, a alternativa com o menor caminho é a letra C.

Gabarito: Letra C

11. (FCC – 2018 – DPE-AM) Considere o grafo abaixo.





A complexidade ciclomática é uma métrica que mede a complexidade de um determinado módulo (uma classe, um método, uma função etc.), a partir da contagem do número de caminhos independentes que ele pode executar até o seu fim. Um caminho independente é aquele que apresenta pelo menos uma nova condição (possibilidade de desvio de fluxo) ou um novo conjunto de comandos a serem executados. O resultado da complexidade ciclomática indica quantos testes, pelo menos, precisam ser executados para que se verifiquem todos os fluxos possíveis que o código pode tomar, a fim de garantir uma completa cobertura de testes.

(Adaptado de: <https://www.treinaweb.com.br/blog/complexidade-cicloomatica-analise-estatica-e-refatoracao/>)

Considerando que no grafo acima há 17 arestas e 13 nós, o cálculo da complexidade ciclomática resulta em

- a) 6
- b) 4
- c) 7
- d) 20
- e) 18

Comentários:

Para responder a esta questão, deve-se usar a fórmula da complexidade ciclomática:

$$C_C = \text{Quantidade}_{\text{Arestas}} - \text{Quantidade}_{\text{Nós}} + 2$$

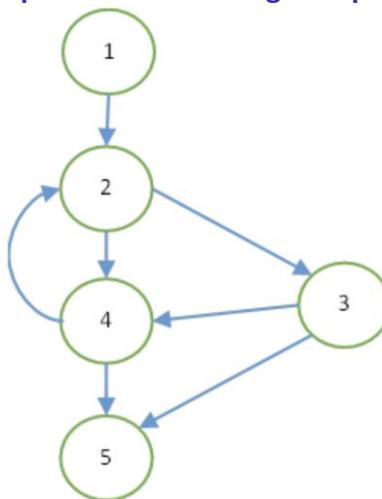
Ou seja, $C_C = 17 - 13 + 2 = 6$.

Gabarito: Letra A

12. (FGV – 2019 – DPE-RJ) Para que um sistema seja testado adequadamente, é preciso realizar uma quantidade mínima de testes. Para apoiar essa definição, foi criada a Complexidade Ciclomática de McCabe, com fundamentação na teoria dos grafos. Essa técnica define uma métrica de software que fornece uma medida quantitativa da complexidade lógica de um programa, apresentando um limite



superior para a quantidade de casos de testes de software que devem ser conduzidos. A Complexidade Ciclomática pode ser calculada tanto pelo número de regiões quanto pelo número de arestas e nós.



Com base no grafo de fluxo acima, correspondente a um trecho de código a ser testado, a quantidade mínima de testes que devem ser realizados para garantir que cada caminho do código tenha sido percorrido em ao menos um teste é:

- a) 11 (onze);
- b) 6 (seis);
- c) 5 (cinco);
- d) 4 (quatro);
- e) 3 (três).

Comentários:

Para responder a esta questão, deve-se usar a fórmula da complexidade ciclomática:

$$C_C = \text{Quantidade}_{\text{Arestas}} - \text{Quantidade}_{\text{Nós}} + 2$$

Os nós são cada um dos círculos no grafo, enquanto as arestas são cada uma das ligações entre os nós.

Portanto, observando o grafo, percebemos que há 5 nós e 7 arestas.

Ou seja, $C_C = 7 - 5 + 2 = 4$.

Gabarito: Letra D

13. (CESPE - 2010 – TJ/ES – Analista de Suporte) Considerando-se a implementação de um grafo denso, direcionado e ponderado, se o número de vértices ao quadrado tem valor próximo ao número de arcos, o uso de uma matriz de adjacência simétrica apresenta vantagens em relação ao uso de uma lista de adjacência.

Comentários:

Em grafos não direcionados, as matrizes de adjacência são simétricas ao longo da diagonal principal - isto é, a entrada a_{ij} é igual à entrada a_{ji} . Matrizes de Adjacência de grafos direcionados, no entanto, não são assim. Num dígrafo sem pesos, a entrada a_{ij} da matriz é 1 se há um arco de v_i para v_j e 0, caso contrário. Há outra maneira de representar também chamada Lista de Adjacências.

Vamos por partes! Ele afirma que o grafo é denso, direcionado e ponderado. Em seguida, ele afirma que o número de vértices ao quadrado tem valor próximo ao número de arcos. Precisava dizer isso? Não está errado, mas ele já havia afirmado que era um grafo denso. A Matriz de Adjacência simétrica é uma representação utilizada em grafos não-direcionados, logo a questão já está errada. **Gabarito: E**



14. (FCC - 2013 – MPE/SE – Analista do Ministério Público) Considere:

I. Estrutura de dados que possui uma sequência de células, na qual cada célula contém um objeto de algum tipo e o endereço da célula seguinte.

II. Podem ser orientados, regulares, completos e bipartidos e possuir ordem, adjacência e grau.

III. Possuem o método de varredura esquerda-raiz-direita (e-r-d).

Os itens de I a III descrevem, respectivamente,

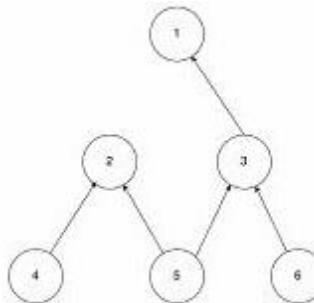
- a) árvores binárias, listas ligadas e arrays.
- b) arrays, árvores binárias e listas ligadas.
- c) grafos, árvores binárias e arrays.
- d) listas ligadas, grafos e árvores binárias.
- e) grafos, listas ligadas e árvores binárias.

Comentários:

(a) Trata-se das Listas Ligadas, visto que falou de sequência, objeto e endereço da célula seguinte; (b) Trata-se dos Grafos, visto que falou dos tipos regular, bipartido, completo e orientado; (c) Trata-se das Árvores Binárias, visto que falou de método de varredura e raiz.

Gabarito: Letra D

15. (CESPE - 2013 – TCE/ES – Analista de Sistemas) Considerando o grafo ilustrado acima, assinale a opção em que é apresentada a descrição em vértices (V) e arestas (A).



- a) $V = \{1, 2, 3, 4, 5, 6\}$
 $A = \{(2, 4), (2, 3), (2, 5), (3, 6), (1, 5)\}$

- b) $V = \{2, 4, 1, 3, 6, 5\}$
 $A = \{(4, 2), (1, 3), (5, 2), (6, 3), (5, 3)\}$

- c) $V = \{1, 2, 3, 4, 5, 6\}$
 $A = \{(4, 2), (3, 4), (5, 2), (6, 3), (5, 3)\}$

- d) $V = \{1, 2, 3, 4, 5, 6\}$
 $A = \{(4, 2), (3, 1), (5, 1), (6, 2), (5, 3)\}$

- e) $V = \{2, 4, 1, 3, 6, 5\}$
 $A = \{(4, 2), (3, 1), (5, 2), (6, 3), (5, 3)\}$



Comentários:

Galera, os vértices não precisam estar ordenados, logo todos os itens estão corretos. No entanto, as arestas precisam corresponder ao grafo. Vamos por eliminação: (a) A aresta (2,3) não existe; (b) Quase tudo certo, mas a ordem (1,3) está errada – seria (3,1); (c) A aresta (3,4) não existe; (d) A aresta (5,1) não existe; (e) Tudo perfeito!

Gabarito: Letra E

- 16. (CESPE - 2012 – TJ/RO – Analista de Suporte – ITEM B)** Grafo corresponde a uma estrutura abstrata de dados que representa um relacionamento entre pares de objetos e que pode armazenar dados em suas arestas e vértices, ou em ambos.

Comentários:

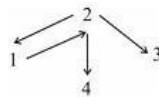
Fiz uma disciplina na faculdade chamada Teoria dos Grafos! Aquilo era absurdamente complexo, mas para concursos a teoria é beeem mais tranquila e muito rara de cair. Portanto fiquem tranquilos, bacana? Uma definição de grafo afirma que ele é uma estrutura de dados que consiste em um conjunto de nós (ou vértices) e um conjunto de arcos (ou arestas).

Em outras palavras, podemos dizer que é simplesmente um conjunto de pontos e linhas que conectam vários pontos. Ou também que é uma representação abstrata de um conjunto de objetos e das relações existentes entre eles. Uma grande variedade de estruturas do mundo real podem ser representadas abstratamente através de grafos. Professor, pode me passar um exemplo? Claro!

Conforme vimos em aula, a definição está perfeita!

Gabarito: Correto

- 17. (CESPE - 2012 – PEFOCE – Perito Criminal)** Considere que um grafo G seja constituído por um conjunto (N) e por uma relação binária (A), tal que $G = (N, A)$, em que os elementos de N são denominados nós (ou vértices) e os elementos de A são denominados arcos (ou arestas). Em face dessas informações e do grafo abaixo, é correto afirmar que esses conjuntos são $N = \{1,2,3,4\}$ e $A = \{(1,2),(2,1),(2,4),(2,3)\}$.



Comentários:

Conforme vimos em aula, a questão está perfeita! Temos quatro nós: $N = \{1,2,3,4\}$; e quatro arestas: $A = \{(1,2),(2,1),(2,4),(2,3)\}$ – observem que a ordenação do grafo ordenado está perfeita.

Gabarito: Correto

- 18. (CESPE - 2012 – BASA – Analista de Sistemas)** É misto o grafo com arestas não dirigidas que representam ruas de dois sentidos e com arestas dirigidas que correspondem a trechos de um único sentido, modelado para representar o mapa de uma cidade cujos vértices sejam os cruzamentos ou finais de ruas e cujas arestas sejam os trechos de ruas sem cruzamentos.



Comentários:

Um grafo simples é aquele que não contém laços. Um grafo vazio é aquele que contém exclusivamente vértices (não contém arcos). Um grafo misto é aquele que possui arestas dirigidas e não-dirigidas. Um grafo trivial é aquele que possui somente um vértice. Um grafo é denso se contém muitos arcos em relação ao número de vértices e esparso se contém poucos arcos. Como assim, professor?

Conforme vimos em aula, um grafo misto é aquele que possui arestas dirigidas e não-dirigidas. O caso citado na questão é um exemplo perfeito e foi retirado integralmente do livro Projeto de algoritmos: Fundamentos, análise e exemplos da internet de Michael T. Goodrich e Roberto Tamassia.

Gabarito: Correto

19. (CESPE - 2012 – BASA – Analista de Sistemas) Para modelar a rede que conecta todos os computadores em uma sala de escritório com a menor metragem possível de cabos, é adequado utilizar um grafo G cujos vértices representem os possíveis pares (u, v) de computadores e cujas arestas representem o comprimento dos cabos necessários para ligar os computadores u e v , determinando-se o caminho mínimo, que contenha todos os vértices de G , a partir de um dado vértice v .

Comentários:

Galera, leiam devagar a questão! Vejam essa parte: “(...) cujos vértices representem os possíveis pares (u,v) ”. Vocês, é claro, se lembram do conceito de vértices e arestas. Ora, vértice representa um par de computadores? Não, vértices são os computadores! Quem representa pares são as arestas!

Gabarito: Errado

20. (CESPE - 2012 – BASA – Analista de Sistemas) Um grafo completo contém pelo menos um subgrafo ponderado.

Comentários:

Essa questão não faz o menor sentido! Podemos dizer que um grafo completo contém pelo menos um subgrafo. No entanto, os conceitos de completude e ponderação são completamente independentes. Eu posso ter um grafo completo ponderado ou não; e posso ter um grafo ponderado completo ou não.

Gabarito: Errado

21. (CESPE - 2012 – BASA – Analista de Sistemas) Um grafo não direcionado é dito conectado quando há pelo menos um caminho entre dois vértices quaisquer do grafo.

Comentários:

Um grafo é dito conexo ou conectado quando existir pelo menos um caminho entre cada par de vértices. Caso contrário, ele será dito desconexo, isto é, se há pelo menos um par de vértices que não esteja ligado a



nenhuma cadeia (caminho). Vejam a imagem do grafo que eu desenhei lá em cima! Ele é conexo ou desconexo? Ele é desconexo, há um par de vértices (E,F) que não está ligado a nenhum caminho.

Conforme vimos em aula, a questão comete um minúsculo deslize! Seria mais correto dizer que um grafo não direcionado é dito conectado quando sempre há pelo menos um caminho entre dois vértices quaisquer do grafo. Por que, professor? Vejam o grafo que eu desenhei na teoria dessa aula.

Pela definição da questão, ele é conectado quando há pelo menos um caminho entre dois vértices quaisquer do grafo. Vamos pegar dois vértices quaisquer? Sim, vamos pegar o par A-C! Há pelo menos um caminho entre eles? Sim, logo podemos dizer – por essa definição – que ele é conectado.

No entanto, o par D-E não possui um caminho entre eles, mas a definição da questão não disse para verificar todos os pares de vértices. Entenderam a sutileza? Portanto, seria ideal dizer que ele é conectado quando há pelo menos um caminho entre cada par de vértices do grafo.

Dessa forma, cobrimos todos os pares. É tão sutil que nem eu havia percebido – quem me alertou foi um aluno.

Gabarito: Correto

22. (CESPE - 2012 – TJ/AC – Analista de Sistemas) Define-se um grafo como fortemente conexo se todos os nós puderem ser atingidos a partir de qualquer outro nó.

Comentários:

Se o grafo for direcionado/orientado, um grafo é dito fortemente conexo se existir um caminho de $A \rightarrow B$ e de $B \rightarrow A$, para cada par (A,B) de vértices de um grafo. Em outras palavras, o grafo é fortemente conexo se cada par de vértices participa de um circuito. Isto significa que cada vértice pode ser alcançável partindo-se de qualquer outro vértice do grafo.

Conforme vimos em aula, um grafo é fortemente conexo se todos os nós puderem ser atingidos a partir de qualquer outro nó. Como vimos em na descrição, cada vértice pode ser alcançável partindo-se de qualquer outro vértice do grafo.

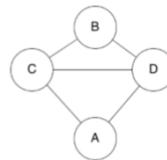
Gabarito: Correto

23. (CESPE - 2013 – CRPM – Analista de Sistemas) Considere que o grafo não orientado representado na figura abaixo possua as seguintes características:

$$G_1 = (V_1, A_1)$$

$$V_1 = \{A, B, C, D\}$$

$$A_1 = \{(A, C), (A, D), (B, C), (B, D), (A, B)\}.$$



Nesse caso, é correto afirmar que o grafo G_1 possui quatro vértices, nomeados de A, B, C e D, e cinco arcos, que conectam pares de vértices, conforme especificado em A_1 .

Comentários:

A questão está quase perfeita, mas ela possui um deslize: não existe a aresta (A, B) – seria (C, D) .



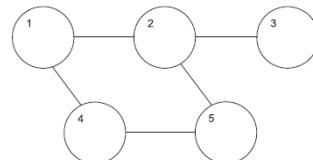
24. (CESPE - 2012 – BASA – Analista de Sistemas) A implementação de um grafo do tipo ponderado e direcionado na forma de uma matriz de adjacência utiliza menor quantidade de memória que a implementação desse mesmo grafo na forma de uma lista encadeada.

Comentários:

Pensem comigo! Para um grafo com uma Matriz de Adjacência esparsa (não densa), uma representação de Lista de Adjacências do grafo ocupa menos espaço, porque ele não usa nenhum espaço para representar as arestas que não estão presentes. Lembram-se da lista? Nós só representamos os nós adjacentes, em contraste com a Matriz de Adjacência. No entanto, quanto mais denso, isso pode mudar.

Conforme vimos em aula, para grafos esparsos (não densos), a Lista de Adjacência (que é uma lista encadeada) ocupa menos espaço em memória, na medida em que não necessita representar vértices não adjacentes – diferente da Matriz de Adjacência.

25. (CESGRANRIO - 2014 – BASA – Analista de Sistemas) O grafo acima pode ser representado pela seguinte matriz:



a)
$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

b)
$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

c)
$$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

d)
$$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$



e)

$$\begin{array}{|cccccc} \hline & 0 & 1 & 1 & 0 & 0 \\ \hline 1 & 0 & 1 & 0 & 1 & \\ 1 & 1 & 0 & 1 & 0 & \\ 1 & 1 & 1 & 0 & 1 & \\ 1 & 1 & 0 & 0 & 0 & \\ \hline \end{array}$$

Comentários:

Vamos ver se vocês se lembram! O vértice A11 tem ser 0, porque não há uma aresta do Nô 1 para o Nô 1. O vértice A12 tem ser 1, porque há uma aresta do Nô 1 para o Nô 2. O vértice A13 tem ser 0, porque não há uma aresta do Nô 1 para o Nô 3. O vértice A14 tem ser 1, porque há uma aresta do Nô 1 para o Nô 4. O vértice A15 tem ser 0, porque não há uma aresta do Nô 1 para o Nô 5. Logo, a primeira linha deve ser: 0, 1, 0, 1, 0. Eliminamos todos os itens, exceto os dois primeiros. Vamos pegar um vértice específico agora para descobrir qual está certo. Observem o vértice A23 e percebam que ele deve ser 1, porque existe uma aresta do Nô 2 para o Nô 3. Descobrimos a resposta!

Gabarito: Letra A

26. (CESPE - 2012 – TJ/SE – Analista de Sistemas) Um grafo é formado por um par de conjuntos de vértices e arestas, não podendo o conjunto de vértices ser particionado em subconjuntos.

Comentários:

Podem, sim. O nome disso é: Grafo Bipartido! Um grafo é dito ser bipartido quando seu conjunto de vértices V puder ser particionado em dois subconjuntos V1 e V2. Portanto não há óbice quanto ao particionamento de um conjunto de vértices em subconjuntos.

Gabarito: Errado

27. (CESPE - 2012 – TRT/AM – Analista de Sistemas) Um grafo é uma estrutura de dados consistida em um conjunto de nós (ou vértices) e um conjunto de arcos (ou arestas). O grafo em que os arcos possuem um número ou peso associados a eles, é chamado de grafo:

- a) predecessor.
- b) adjacente.
- c) incidente.
- d) ponderado.
- e) orientado.

Comentários:

Vamos ver agora alguns conceitos importantes! Um grafo pode ser dirigido – também chamado direcionado, orientado ou dígrafo –, se as arestas tiverem uma direção (imagem da esquerda), ou não dirigido, se as arestas não tiverem direção (imagem central). Se as arestas tiverem associado um peso ou custo, o grafo passa a ser chamado ponderado, valorado ou pesado (imagem da direita).

Conforme vimos em aula, um grafo com arcos numerados ou com peso são chamados Grafos Ponderados ou Grafos Valorados ou Grafos Pesados.

Gabarito: Letra D



QUESTÕES COMENTADAS - HASHING - MULTIBANCAS

1. (CESPE - 2010 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação - Administração de Dados) A pesquisa sequencial é aplicável em estruturas não ordenadas.

Comentários:

Perfeito! Para fazer uma pesquisa sequencial, não é necessário que os dados estejam ordenados – diferentemente da pesquisa binária. **Gabarito: C**

2. (CESPE - 2012 - Banco da Amazônia - Técnico Científico - Administração de Dados) As colisões ocorrem na utilização de tabela hash porque várias chaves podem resultar na mesma posição.

Comentários:

Perfeito! É raro, mas acontece... **Gabarito: C**

3. (CESPE - 2010 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação - Administração de Dados) Ocorre o hashing quando não há o armazenamento de cada entrada de uma tabela em um específico endereço calculado a partir da aplicação de uma função chave da entrada.

Comentários:

Pelo contrário, ocorre o hashing quando há o armazenamento de cada entrada de uma tabela em um específico endereço calculado a partir da aplicação de uma função chave da entrada. **Gabarito: E**

4. (FCC - 2008 - METRÔ-SP - Analista Treinee - Análise de Sistemas) O objetivo de fazer uma busca rápida a partir de uma chave de pesquisa simples e obter o valor desejado é alcançado pela estrutura de dados especial denominada:

- a) array.
- b) lista.
- c) vetor.
- d) árvore binária.
- e) tabela de hashing.



Comentários:

Trata-se da Tabela de Hashing! **Gabarito: E**

5. (CESPE - 2012 - Banco da Amazônia - Técnico Científico - Administração de Dados) A busca que utiliza uma tabela hash realiza comparação das chaves para encontrar a posição do elemento que está sendo buscado.

Comentários:

Não, eles utilizam a chave para gerar resultados que, esses sim, são comparados.

Gabarito: E

6. (CESPE - 2010 - DETRAN-ES - Analista de Sistemas) No método de hashing, por meio de acesso sequencial, são utilizados tabelas e mapas para recuperar informações de endereço de arquivos de forma rápida e eficiente.

Comentários:

Não, Método de Hashing não faz Acesso Sequencial. **Gabarito: E**

7. (FCC - 2015 - DPE-SP - Programador) Um Programador da Defensoria Pública do Estado de São Paulo foi solicitado a propor uma solução para o problema: Há uma quantidade grande de dados classificáveis por chave e estes dados devem ser divididos em subconjuntos com base em alguma característica das chaves. Um método eficiente deve ser capaz de localizar em qual subconjunto deve-se colocar cada chave e depois estes subconjuntos bem menores devem ser gerenciados por algum método simples de busca para que se localize uma chave rapidamente. O Programador propôs como solução, corretamente, a implementação de:

- a) Deques.
- b) Tabela e função hash.
- c) Pilhas.
- d) Fila duplamente encadeada.
- e) Árvore Binária de Busca.

Comentários:

Como se fala em subdivisão em conjuntos com base nas chaves, podemos afirmar que se trata-se da Tabela de Hashing! As árvores de busca binária não dividem os elementos em subconjuntos, somente ordena a estrutura não-linear de forma a facilitar uma busca binária. **Gabarito: B**



QUESTÕES COMENTADAS - BITMAP - MULTIBANCAS

1. (FGV - 2015 - TJ-RO - Analista Judiciário - Análise de Sistemas)

| ID | Nome | Curso |
|------|------|------------|
| 1210 | A | Física |
| 356 | B | Química |
| 23 | C | Matemática |
| 57 | D | Física |
| 45 | E | Física |
| 6 | F | Matemática |
| 210 | G | Matemática |

Se fosse construído um índice de banco de dados do tipo "bitmap" para essa tabela, tendo o campo Curso como chave, o conteúdo desse índice seria:

a)

| | |
|------|---|
| 6 | 3 |
| 23 | 3 |
| 45 | 1 |
| 57 | 1 |
| 210 | 3 |
| 356 | 2 |
| 1210 | 1 |

b)

| | |
|------------|---------|
| Física | 1001100 |
| Química | 0100000 |
| Matemática | 0010011 |

c)

| | |
|----|-----|
| 6 | 001 |
| 23 | 001 |
| 45 | 100 |



57 100
 210 001
 356 010
 1210 100

d)

0011001100
 0100100000
 0010010011

e)

| | |
|-------------|------------|
| 00000000110 | Matemática |
| 00000010111 | Matemática |
| 00000101101 | Física |
| 00000111011 | Física |
| 00011010010 | Matemática |
| 00101100100 | Química |
| 10010111010 | Química |

Comentários:

A questão solicita a criação de um índice usando bitmap da coluna curso. Lembrando da aula teórica que a tabela bitmap terá como linhas os valores da coluna fonte do índice, ou seja, Física, Química e Matemática. As colunas do bitmap são os números das tuplas (ROWID). O conteúdo do bitmap é o preenchimento de acordo com o valor de cada tupla original relativo à coluna fonte do índice. Construindo o bitmap para curso temos:

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------------|---|---|---|---|---|---|---|
| Física | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| Química | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| Matemática | 0 | 0 | 1 | 0 | 0 | 1 | 1 |

A resposta, portanto, é a letra B. **Gabarito: B**



2. (FGV - 2014 - DPE-RJ - Técnico Superior Especializado - Administração de Dados)

| Candidato | |
|------------------|----------------------|
| inscrição | candidatoNome |
| 101 | João |
| 102 | Maria |
| 105 | Gabriela |
| 106 | João |

| Prova | |
|------------------|-------------|
| provaNome | data |
| Português | 12/01/2014 |
| Matemática | 12/01/2014 |
| Prática | 19/01/2014 |

| Avaliação | | |
|------------------|------------------|-------------|
| inscrição | provaNome | nota |
| 101 | Português | 10 |
| 102 | Português | 8 |
| 105 | Português | 3 |
| 106 | Português | 5 |
| 101 | Matemática | 7 |
| 102 | Matemática | 4 |
| 105 | Matemática | 8 |
| 101 | Prática | 5 |
| 102 | Prática | 5 |
| 105 | Prática | NULL |
| 106 | Prática | 4 |

Índices do tipo Bitmap podem ser usados em tabelas de bancos de dados. O quadro abaixo representa o mapa de bits na indexação da tabela Avaliação quando a chave considerada é a concatenação dos atributos Inscrição e provaNome.

| | |
|----------------------|---------------------|
| 101Matemática | 00001000000 |
| 101Português | 100000000000 |
| 101Prática | 00000001000 |
| 102Matemática | 00000100000 |
| 102Português | 010000000000 |
| 102Prática | 00000000100 |
| 105Matemática | 00000010000 |



| | |
|---------------------|--------------------|
| 105Português | 00100000000 |
| 105Prática | 00000000010 |
| 106Português | 00010000000 |
| 106Prática | 00000000001 |

Num mapa de bits para a mesma tabela, usando apenas o atributo Inscrição, o mapa de bits seria

a)

101 10001001000
102 01000100100
105 00100010010
106 00010000001

b)

101 1000
102 0100
105 0010
106 0001

c)

101 10000000000
102 01000100000
105 00100000000
106 00010000000

d)

101 00000001000
102 00000000100
105 00000000010
106 00000000001

e)

101 111



102 111

105 111

106 011

Conforme vimos na parte teórica da aula, os índices podem ser criados a partir de uma coluna só, mas também para concatenação de colunas. A questão pede o índice mais simples, somente para a coluna Inscrição

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|------------|----------|----------|----------|----------|----------|----------|----------|----------|----------|-----------|-----------|
| 101 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 102 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 105 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 106 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

A resposta, portanto, é letra A. **Gabarito: A**



QUESTÕES COMENTADAS

Cesgranrio

1. (CESGRANRIO-TRANSPETRO- 2023) Estruturas de dados referem-se aos diferentes mecanismos de organização de dados para atender a diferentes requisitos de processamento. Dentre as estruturas de dados, é eficiente para inserção e remoção de elementos em qualquer posição, incluindo início, meio e fim, além de oferecer acesso aos elementos em posições intermediárias, a seguinte estrutura de dados:

- a) pilha
- b) fila
- c) lista encadeada
- d) array estático
- e) vetor dinâmico

Comentários:

A estrutura de dados mais eficiente para inserção e remoção de elementos em qualquer posição (incluindo início, meio e fim) e que oferece acesso aos elementos em posições intermediárias é a lista encadeada.

Pilha (a): Uma pilha é uma estrutura de dados que segue o princípio Last In, First Out (LIFO), onde o último elemento a ser adicionado é o primeiro a ser removido. A inserção e remoção são eficientes, mas apenas no topo da pilha, não em posições intermediárias ou no fim.

Fila (b): Uma fila segue o princípio First In, First Out (FIFO), onde o primeiro elemento a ser adicionado é o primeiro a ser removido. Semelhante à pilha, a inserção e remoção são eficientes no início e no fim da fila, respectivamente, mas não em posições intermediárias.

Lista encadeada (c): A lista encadeada permite a inserção e remoção de elementos em qualquer posição com eficiência. Cada elemento (nó) na lista encadeada contém o dado e a referência para o próximo nó, facilitando a inserção e remoção em qualquer ponto da lista sem a necessidade de deslocar outros elementos, como seria necessário em um array estático ou um vetor dinâmico.

Array estático (d): Em um array estático, o tamanho é definido na criação e não pode ser alterado. A inserção e remoção de elementos em qualquer lugar que não seja o final do array exigem o deslocamento de elementos subsequentes, o que não é eficiente.

Vetor dinâmico (e): Semelhante ao array estático, porém seu tamanho pode ser ajustado dinamicamente. Apesar disso, inserções e remoções no início ou no meio também requerem deslocamento de elementos, o que reduz a eficiência para essas operações, apesar de serem mais flexíveis que arrays estáticos em termos de gestão de capacidade.

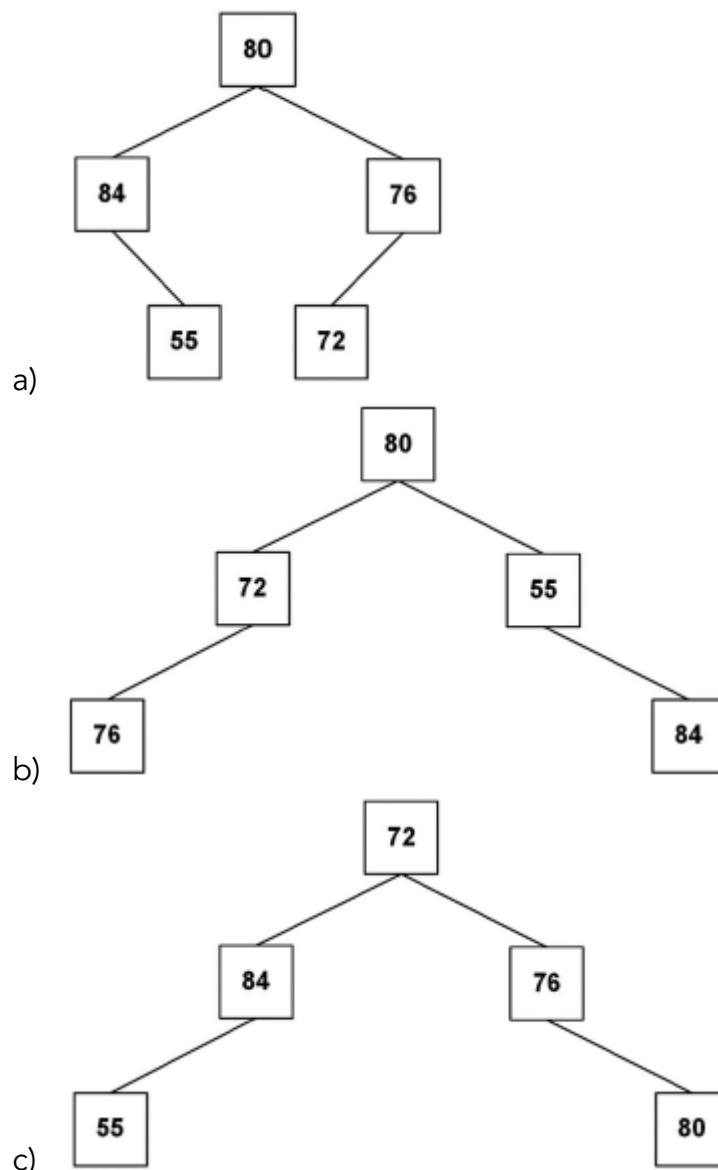
Portanto, a letra C (lista encadeada) é a resposta correta, pois essa estrutura de dados atende a todos os requisitos mencionados de maneira eficiente.

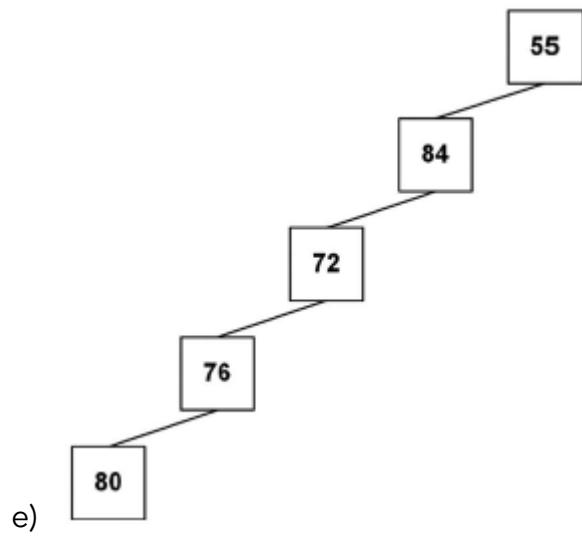
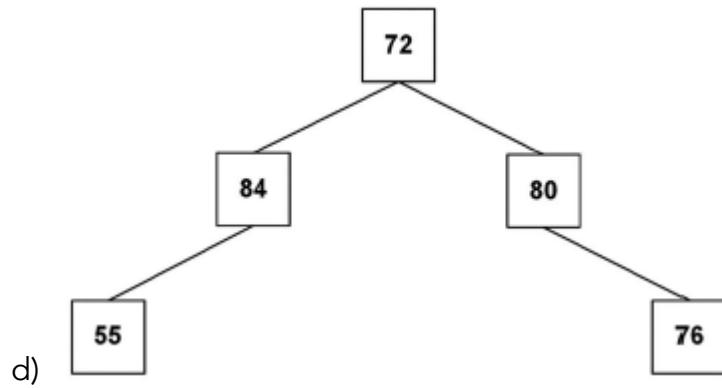


2. (CESGRANRIO– Esc BB– 2023) Um estudante de computação decidiu escrever um método Java para exibir, no console, em pré-ordem, os valores dos nós de uma árvore binária recebida como parâmetro. Ao executar esse método, os seguintes valores foram exibidos no console:

80 84 55 76 72

Considerando os valores exibidos, qual árvore foi recebida como parâmetro?





Comentários:

A primeira coisa a entender é que a árvore em questão é uma árvore binária, o que significa que cada nó da árvore pode ter no máximo dois filhos. Existem três tipos de percursos em árvores binárias:

Pré-ordem: Visita a raiz, depois a subárvore esquerda e por fim a subárvore direita.

Em ordem: Visita a subárvore esquerda, a raiz e por fim a subárvore direita.

Pós-ordem: Visita a subárvore esquerda, a subárvore direita e por fim a raiz.

No caso da questão, o texto informa que a árvore foi percorrida em pré-ordem e os valores exibidos foram: 80, 84, 55, 76, 72.

Raiz: O primeiro valor (80) é a raiz da árvore.

Subárvore esquerda: O segundo valor (84) é o filho esquerdo da raiz.

Subárvore direita: O terceiro valor (55) é o filho direito da raiz.

Subárvore esquerda do filho esquerdo: O quarto valor (76) é o filho esquerdo do filho esquerdo da raiz.

Subárvore direita do filho esquerdo: O quinto valor (72) é o filho direito do filho esquerdo da raiz.

Gabarito: Letra A



3. (CESGRANRIO–AgeRIO– 2023) As classes Java a seguir são usadas na implementação de árvores binárias.

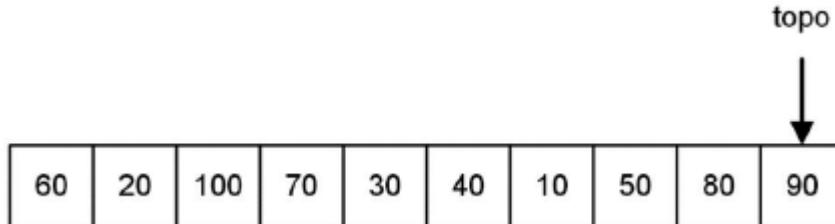
```
public class ArvNo {  
    int info;  
    ArvNo esq=null,dir=null;  
}  
  
public class Arv {  
    private ArvNo raiz;  
  
    public Arv(){  
    }  
  
    private void percorre(ArvNo r) {  
        if(r==null)  
            return;  
  
        percorre(r.dir);  
        percorre(r.esq);  
        System.out.printf("%d ", r.info);  
    }  
  
    public void exibeArvore() {  
        percorre(raiz);  
    }  
}
```

A classe Main abaixo faz uso da classe Arv.



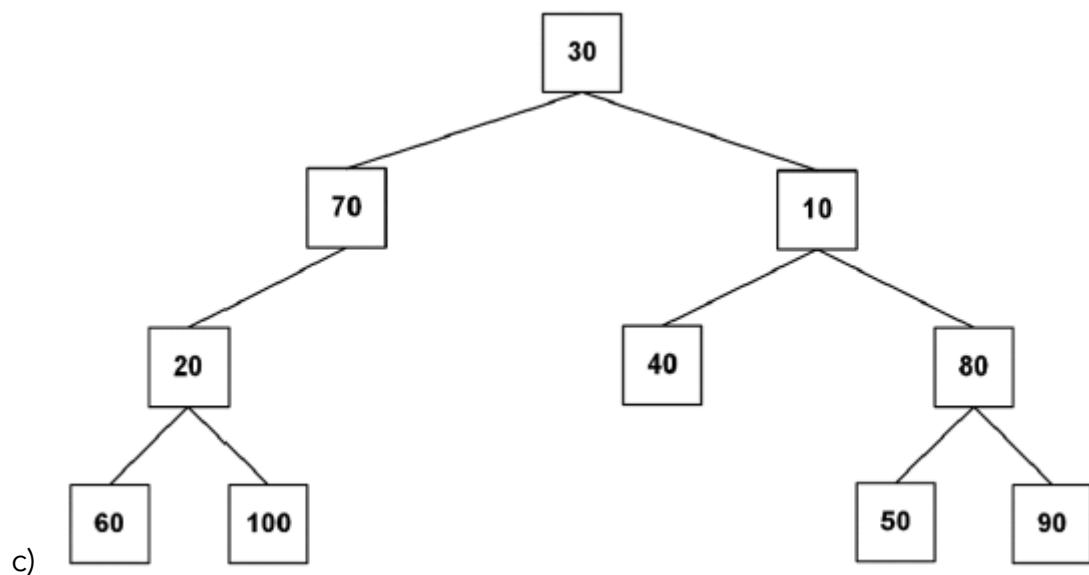
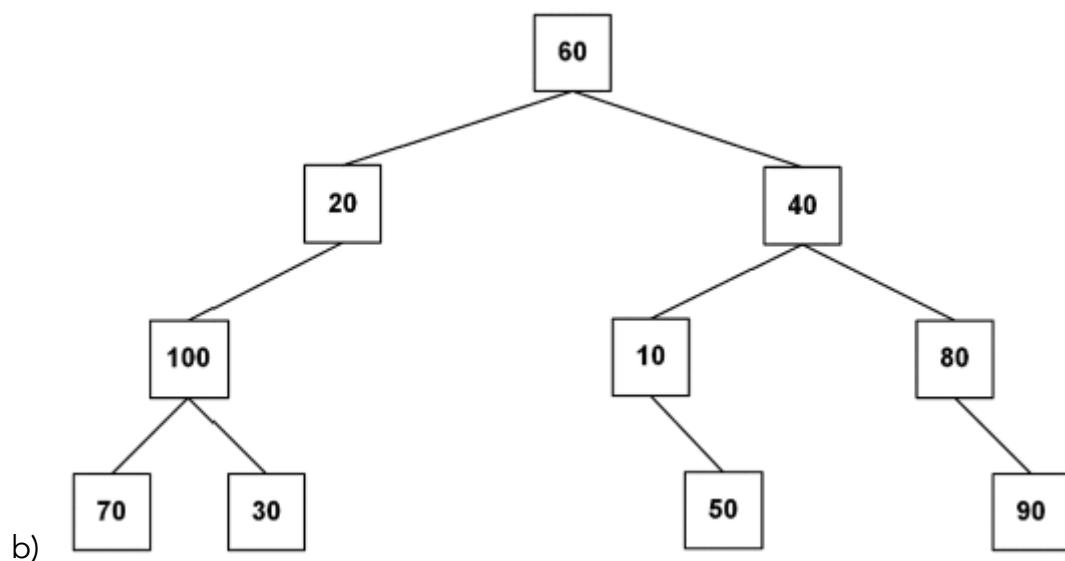
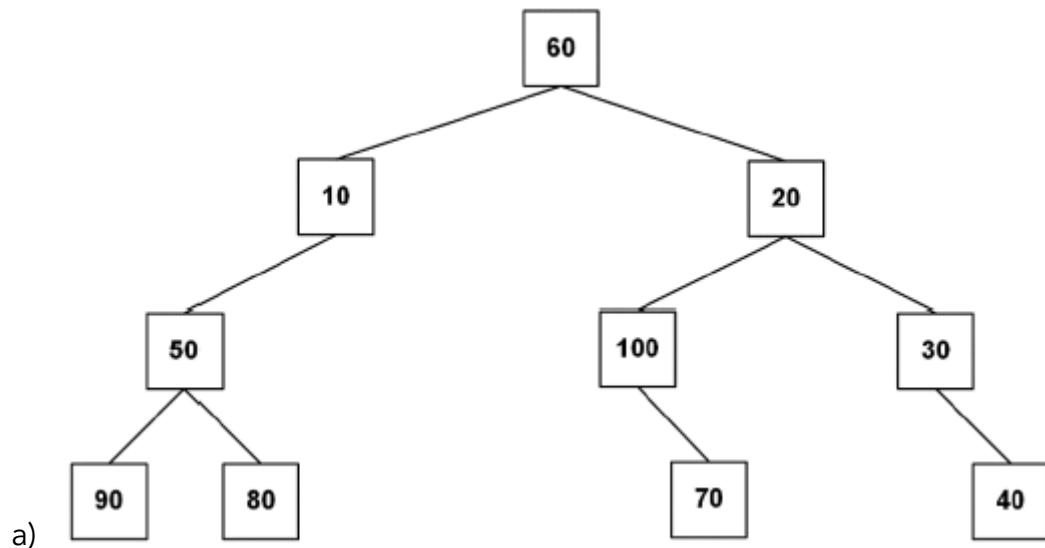
```
public class Main {  
  
    public static void main(String[] args) {  
        Stack<Integer> p;  
        Arv a;  
  
        // Comandos relativos à criação de uma pilha  
        // e de uma árvore binária.  
        //  
        // Esses comandos são irrelevantes para a  
        // resolução da questão.  
  
        percorrePilha(p);  
        System.out.println();  
        a.exibeArvore();  
    }  
  
    public static void percorrePilha(Stack<Integer> p) {  
        while( !p.isEmpty())  
            System.out.printf("%d ", p.pop());  
    }  
}
```

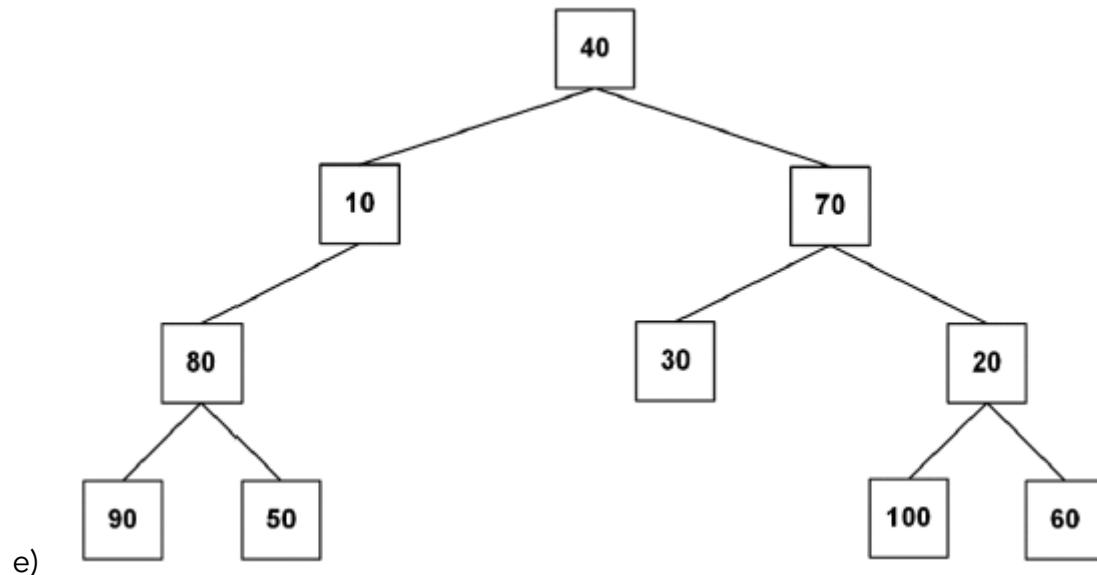
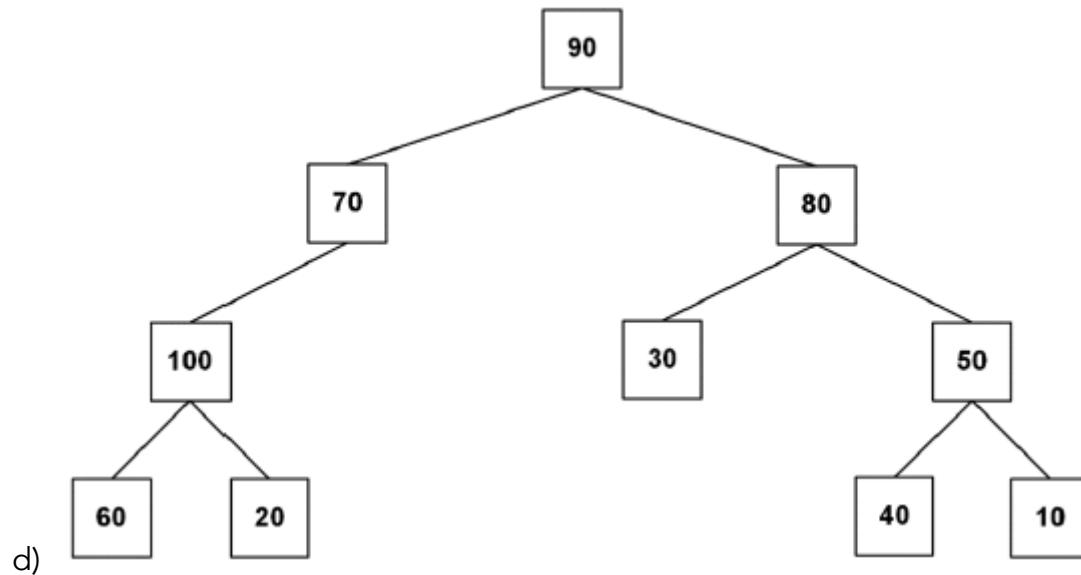
Admita que o método main acima vá ser executado, e que uma pilha como a mostrada na Figura a seguir vá ser passada como parâmetro para o método percorrePilha.



Qual árvore binária fará com que o comando a.exibeArvore() exiba no console os mesmos números inteiros, e na mesma ordem, que o método percorrePilha?







Comentários:

Vamos analisar o problema passo a passo para entendermos como encontrar a árvore binária que atende aos requisitos. O método percorrePilha remove elementos de uma pilha e os imprime na seguinte ordem:

1. Remove o elemento do topo da pilha e o imprime.
2. Se o elemento removido tiver um filho direito, empilha o filho direito.
3. Se o elemento removido tiver um filho esquerdo, empilha o filho esquerdo.

Com a ordem dos números inteiros em mente, podemos construir a árvore binária de forma que a ordem de percurso em pós-ordem seja a mesma da ordem da pilha.

O percurso em pós-ordem visita os nós da árvore binária na seguinte ordem:

- Subárvore esquerda
- Subárvore direita



- Raiz

A árvore possui os seguintes elementos:

- O elemento 60 é a raiz.
- O elemento 20 é o filho esquerdo da raiz.
- O elemento 40 é o filho direito da raiz.
- O elemento 100 é o filho esquerdo do elemento 20.
- O elemento 10 é o filho direito do elemento 40.
- O elemento 80 é o filho direito do elemento 40.
- O elemento 70 é o filho esquerdo do elemento 100.
- O elemento 30 é o filho direito do elemento 100.
- O elemento 50 é o filho esquerdo do elemento 80.
- O elemento 90 é o filho direito do elemento 80.

Ao percorrer essa árvore em pós-ordem, obtemos a seguinte ordem:

1. 90
2. 80
3. 50
4. 10
5. 70
6. 30
7. 100
8. 40
9. 20
10. 60

a resposta está correta. A árvore binária da alternativa B é a que faz com que o comando a.exibeArvore() exiba os mesmos números inteiros e na mesma ordem que o método percorrePilha.

Gabarito: Letra B

4. (CESGRANRIO–TRANSPETRO– 2023) Considere uma árvore AVL que possui 12 nós. A altura dessa árvore é
- a) 3
 - b) 4
 - c) 5
 - d) 6
 - e) 7.

Comentários:

Uma árvore AVL garante que a diferença de altura entre as subárvore de qualquer nó seja no máximo 1. Isso significa que a altura da árvore pode ser estimada usando o logaritmo de base 2 do número de nós. Para calcular a altura mínima da árvore, podemos usar a seguinte fórmula:

Altura mínima = $\log_2(n + 1) + 1$. Onde n é o número de nós na árvore.



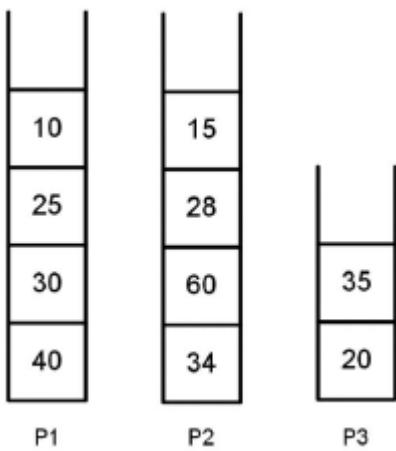
No caso de 12 nós: Altura mínima = $\log_2(12 + 1) + 1 = \log_2(13) + 1 \approx 3.60$

Como a altura precisa ser um número inteiro, arredondamos 3.60 para cima, resultando em uma altura mínima de 4. A altura máxima da árvore AVL com n nós é $n - 1$. No caso de 12 nós, a altura máxima é $12 - 1 = 11$.

A altura de uma árvore AVL com 12 nós pode ser 4 (altura mínima) ou um valor entre 4 e 11 (alturas intermediárias). No entanto, a questão pede a altura no pior caso, o que significa que a resposta correta é 4.

Gabarito: Letra B

5. (CESGRANRIO-Esc BB- 2023) A Figura a seguir exibe o conteúdo de três pilhas: P1, P2 e P3.



Admita que um método Java, chamado exibePilha, receba essas três pilhas como parâmetros e execute os seguintes passos:

1. Cria duas pilhas auxiliares, A1 e A2, inicialmente vazias;
2. Remove um elemento de P1 e o insere em A1. Em seguida, remove um elemento de P2 e o insere em A1. Repete esses dois procedimentos até que P1 e P2 fiquem, ambas, vazias;
3. Remove um elemento de P3 e o insere em A1. Repete esse procedimento até que P3 fique vazia;
4. Remove um elemento de A1 e o insere em A2. Repete esse procedimento até que A1 fique vazia;
5. Remove um elemento de A2 e o exibe no console. Repete esse procedimento 4 vezes.

O que será exibido no console, quando o método exibePilha for executado, tendo P1, P2 e P3 sido passadas como parâmetros?

- a) 10 15 25 28
- b) 10 25 30 40
- c) 15 10 28 25
- d) 20 35 34 40
- e) 40 34 30 60



Comentários:

O método exibePilha concatena os elementos das pilhas P1, P2 e P3 na ordem P1, P2, P3. A ordem de exibição dentro de cada pilha é da última para a primeira (LIFO).

O algoritmo apresentado descreve um procedimento de reordenação e mescla de elementos provenientes de várias pilhas (P1, P2 e P3) utilizando duas pilhas auxiliares (A1 e A2), aproveitando a característica fundamental das pilhas que é o princípio LIFO (Last In, First Out - Último a Entrar, Primeiro a Sair).

No início, elementos são removidos das pilhas P1 e P2 e inseridos em A1, o que já inicia um processo de inversão na ordem em que esses elementos aparecem. Essa etapa é seguida pela inclusão dos elementos da pilha P3 em A1, continuando a inversão da ordem dos elementos. A estratégia aqui é agrupar todos os elementos em uma única pilha auxiliar (A1), mas devido ao processo de remoção e inserção em pilhas, a ordem dos elementos é invertida quando comparada à sua ordem original nas pilhas P1, P2 e P3.

No passo seguinte, o algoritmo faz uso de uma segunda inversão de ordem ao transferir os elementos de A1 para A2. Esta etapa reverte a ordem dos elementos para a sequência desejada, uma vez que os elementos são removidos de A1 e inseridos em A2, aplicando novamente o princípio LIFO.

Finalmente, os elementos são removidos de A2 e exibidos, resultando na ordem final de apresentação 10, 15, 25, 28.

Gabarito: Letra A

6. (CESGRANRIO-PNS– 2022) Seja uma função que realiza uma busca binária sobre um array de números inteiros ordenados. Não se sabe, em princípio, se os números estão ordenados ascendente ou descendente. O cabeçalho dessa função é o seguinte:

```
int busca (int [ ] vet, int elem)
```

Isto é, a função busca recebe um array de números inteiros (vet) e um número inteiro (elem) como parâmetros, e retorna um número inteiro. Caso exista em vet um inteiro igual a elem, a função retornará o índice desse inteiro no array; caso contrário, a função retornará -1.

O algoritmo de busca binária produz um índice (ind) a cada iteração sobre o array, tendo em vista comparar o elemento que se deseja procurar (elem) com o elemento vet [ind]. Isto é:

```
if ( vet [ ind ] == elem )  
return ind;
```

No comando acima, diz-se que houve uma visita ao elemento vet [ind].

Admita que a função busca foi chamada por meio do comando a seguir:



```
int resp = busca (vet, 50);
```

Sabendo-se que os elementos visitados foram 54, 17, 33 e 50, nesta ordem, qual array foi passado como parâmetro para a função busca?

- a) [95, 90, 87, 54, 52, 50, 33, 17, 11, 10]
- b) [5, 10, 11, 17, 33, 50, 54, 87, 90, 95]
- c) [121, 111, 93, 87, 60, 54, 50, 33, 17, 5]
- d) [5, 17, 33, 50, 54, 60, 87, 93, 111, 121]
- e) [130, 121, 111, 90, 70, 60, 54, 50, 33, 17]

Comentários:

O algoritmo de busca binária é uma técnica eficiente para encontrar um elemento em um array ordenado, funcionando através do princípio de divisão e conquista. Esse método reduz significativamente o número de comparações necessárias para encontrar um elemento, comparado à busca linear, especialmente em arrays grandes. A eficácia do algoritmo depende da ordenação prévia do array, e ele pode ser adaptado para lidar tanto com arrays ordenados de forma ascendente quanto descendente.

O algoritmo começa definindo dois índices que delimitam o subconjunto do array que está sendo examinado: um índice inicial (geralmente chamado de esquerda ou low) e um índice final (chamado de direita ou high). Inicialmente, esquerda é definido como 0 (o início do array) e direita é definido como o tamanho do array menos 1 (o final do array). Esses limites são ajustados ao longo do algoritmo para se concentrarem em subconjuntos progressivamente menores do array.

O algoritmo calcula o índice central do subconjunto atual como a média dos índices esquerda e direita. O elemento no índice central é comparado com o elemento buscado (elem). Se o elemento no índice central for igual ao elemento buscado, o algoritmo retorna o índice central, indicando que o elemento foi encontrado.

Se o elemento buscado não for igual ao elemento central, o algoritmo decide se continua a busca na metade esquerda ou direita do subconjunto atual, com base na comparação dos valores.

Se o elemento buscado for menor do que o elemento central (considerando um array ordenado de forma ascendente), o algoritmo atualiza o índice direita para ser o índice central menos 1, descartando a metade direita do subconjunto.

Se o elemento buscado for maior, o índice esquerda é atualizado para ser o índice central mais 1, descartando a metade esquerda. O processo de divisão e conquista continua até que o elemento seja encontrado ou até que o subconjunto se torne vazio (quando esquerda se torna maior que direita), caso em que o algoritmo retorna -1, indicando que o elemento não está presente no array.

A sequência de elementos visitados durante a busca binária é: 54, 17, 33, 50. O primeiro elemento visitado foi 54. O segundo elemento visitado foi 17, o que indica que o elemento



buscado está entre 17 e 54. O terceiro elemento visitado foi 33, confirmando que o valor de elem está entre 33 e 54. O quarto elemento visitado foi 50, que é o valor de elem.

Com base na sequência de elementos visitados e na lógica da busca binária, podemos concluir que o array original está ordenado de forma crescente. Analisando cada alternativa:

- a) O array está ordenado de forma decrescente, o que não é compatível com a sequência de elementos visitados.
- b) O array está ordenado de forma crescente, mas o elemento 52 não foi visitado.
- c) O array está ordenado de forma decrescente, o que não é compatível com a sequência de elementos visitados.
- d) O array está ordenado de forma crescente e os elementos visitados estão presentes na sequência correta.
- e) O array está ordenado de forma decrescente, o que não é compatível com a sequência de elementos visitados.

Gabarito: Letra D

7. (CESGRANRIO-BASA- 2022) Uma função, chamada converte, tem por objetivo converter um número inteiro na base decimal (d), recebido como parâmetro, em um número inteiro na base binária (b), isto é, um número que seja formado apenas pelos algarismos 0 e 1, como nos exemplos abaixo.

Exemplos:

converte(7) = 111
converte(12) = 1100
converte(16) = 10000

Admita que o inteiro (d), recebido como parâmetro, é tal que $d \geq 0$ e $d \leq 1024$.

Qual função executa essa conversão corretamente?

- a) static long converte(int dec) {
 if(dec == 0)
 return dec;
 long r=converte(dec/2);
 return dec % 2 + r * 10;
}
- b) static long converte(int dec) {
 long bin=0;
 while(dec > 0) {
 int r= dec % 2;
 dec=dec / 2;
 bin+=r << 2;
 }
 return bin;
}
- c) static long converte(int dec) {



```

if(dec > 0)
    return dec;
long r=converte(dec%2);
    return dec/ 2 + r * 10;
}

d) static long converte(int dec) {
    long bin=0,fat=1;
    do {
        int r=dec % 2;
        dec=dec/2;
        bin+=r * fat;
        fat*=10;
    } while(dec >= 0);
    return bin;
}

e) static long converte(int dec) {
    long bin=0;
    for(long fat=1; dec > 0; fat*=10) {
        int r=dec % 2;
        dec=dec / 2;
        bin+=r * fat;

    }
    return bin;
}

```

Comentários:

A alternativa A é correta porque implementa corretamente o processo de conversão de um número na base decimal para a base binária usando recursão. O método divide o número decimal por 2 sucessivamente até que o resultado dessa divisão seja 0, que é a condição de parada da recursão. Em cada chamada recursiva, o método calcula o resto da divisão do número atual por 2 (`dec % 2`), que será 0 ou 1, representando o bit atual na posição binária. Esse bit é então somado ao resultado da chamada recursiva multiplicado por 10, para posicionar corretamente o dígito binário no retorno. Esse processo constrói o número binário de forma correta, de trás para frente, começando pelos bits menos significativos.

Vejamos por que as demais alternativas estão incorretas:

b) O uso de `bin+=r << 2;` é incorreto porque o operador `<<` desloca os bits para a esquerda no valor binário. A intenção era provavelmente deslocar o valor binário uma casa decimal para cada novo bit encontrado (`<< 1` seria o deslocamento correto para bits), mas mesmo assim, a implementação deveria usar `fat` para garantir a multiplicação correta por 10 na base decimal, e não um deslocamento de bits.

c) Esta função retorna imediatamente o valor de `dec` se `dec > 0`, o que significa que ela não realiza nenhuma conversão. Além disso, a lógica de conversão está mal-formulada e não segue o processo correto de divisão por 2 e acumulação dos restos.



d) A condição no loop do-while está incorreta (`do {...} while(dec >= 0);`), resultando em um loop infinito para qualquer valor de dec diferente de -1, porque dec alcançará 0 mas nunca será menor que 0, a condição para sair do loop nunca será verdadeira. A intenção era provavelmente usar `do {...} while(dec > 0);`.

e) Apesar de essa alternativa implementar a lógica corretamente, similar ao que é feito na alternativa A, a afirmação de que o gabarito é a letra A está incorreta. Na verdade, a alternativa E também executa a conversão corretamente, empregando um loop for para dividir o número por 2 sucessivamente e acumular o resto multiplicado por um fator que representa a posição binária correta (potência de 10). Portanto, a justificativa do gabarito parece estar baseada em um erro de interpretação ou em uma discrepância nos critérios de avaliação fornecidos. Na realidade, tanto a alternativa A quanto a E realizam a conversão de forma adequada, com A utilizando recursão e E um loop for, ambas construindo o número binário corretamente.

Gabarito: Letra A

8. (CESGRANRIO–BASA– 2022) Em linguagens de programação como Java, onde existem estruturas de repetição, a recursão pode ser muitas vezes substituída pela repetição, com ganhos de desempenho.

Considere a seguinte função recursiva segredo, em Java:

```
public static int segredo(int a) {  
    if (a<2) {  
        return 0;  
    } else {  
        return segredo(a-2)+1;  
    }  
}
```

Que fragmento de código, em Java, contendo uma estrutura de repetição, é adequado para substituí-la?

- a) public static
int alternativaA(int a) { int s = 0;
for (int i=a;i>2;i--) {
 s++;
}
return s;
}
b) public static int alternativaB(int a) {
 int s = 0;
 for (int i=a;i<2 && i>0;i--) {
 s++;
 }
 return s;
}



c) public static int alternativaC(int a) {
 int s = 0;

```
    while (a>=2) {
        a-=2;
        s++;
    }
    return s;
}
```

d) public static int alternativaD(int a) {

```
    int s = 0;
    do {
        a-=2;
        s++;
    } while (a>0 && a<=2);
    return s;
}
```

e) public static int alternativaE(int a) {

```
    int s = 0;
    do {
        a-=2;
        s++;
    } while (a>0 && a<2);
    return s;
}
```

Comentários:

A função segredo recursiva decremente o valor de a por 2 a cada chamada e incrementa o retorno em 1, até que a seja menor que 2. Isso efetivamente conta quantas vezes o número pode ser dividido por 2 antes de se tornar menor que 2.

A alternativa C implementa corretamente a lógica da função recursiva segredo de forma iterativa. Ela utiliza um loop while que continua a subtrair 2 de a enquanto a for maior ou igual a 2, incrementando s a cada iteração. Isso replica exatamente o comportamento da função recursiva original, realizando a mesma quantidade de decrementos de 2 em a e incrementos em s até que a seja menor que 2.

Letra A: O loop for decremente i de a até i ser maior que 2, incrementando s a cada passo. No entanto, este loop nunca executa quando i é exatamente 2, porque a condição de parada é $i > 2$, o que ignora uma subtração final que seria contada.

Letra B: O loop for está configurado para uma condição que nunca é verdadeira ($i < 2 \&\& i > 0$). Além disso, essa lógica não corresponde à da função segredo, pois verifica uma condição de incremento que não replica o comportamento original de subtrair 2 até que a seja menor que 2.

Letra D: O loop do-while executa pelo menos uma vez independente do valor de a, mas a condição while ($a > 0 \&\& a \leq 2$) não replica corretamente o comportamento da função segredo.



Ela para o loop de maneira prematura para certos valores de a e permite uma iteração adicional indevidamente para a inicialmente menor que 2.

Letra E: Similarmente ao D, o loop do-while não replica corretamente a lógica da função segredo. A condição while ($a > 0 \&\& a < 2$) é ainda mais restritiva e menos alinhada com a função original, terminando o loop prematuramente e não executando a subtração e incremento corretamente para todos os casos.

Gabarito: Letra C

9. (CESGRANRIO–PNS– 2022) Seja um array composto por 7 números inteiros.

[5, 15, 77, 21, 5, 25, 2]

Esse array foi usado por um profissional de teste de software para testar uma função que ordena, de forma ascendente, um array de números inteiros. Essa função implementa o algoritmo de ordenação por seleção.

Para avaliar a evolução do array sendo ordenado, o profissional de teste solicitou ao programador que criou a função de ordenação que fizesse uma modificação no código, de modo que o somatório dos elementos do array com índices 2, 3 e 4 seja exibido no console imediatamente antes do incremento da variável (i) que controla a execução do comando de repetição mais externo.

Feitas as modificações solicitadas, o código da função passou a ter a seguinte forma geral:

```
ordena (int vetor[ ]) {  
  
    int i = 0;  
  
    int tam = length (vetor); // comentário: a função length retorna a quantidade de elementos de  
    // um array  
  
    while ( i < tam ) {  
  
        while () {  
  
            // comentário: isso é apenas a forma geral do algoritmo de ordenação  
  
            // não é o código completo  
  
        }  
  
        print ( vetor[2] + vetor[3] + vetor[4] );  
  
        i = i + 1;  
    }  
}
```



O que será exibido no console pelo comando print na 3^a iteração do comando de repetição mais externo?

- a) 32
- b) 38
- c) 41
- d) 103
- e) 113

Comentários:

Para entender a questão, é necessário acompanhar o processo de ordenação por seleção nas primeiras iterações e verificar o que acontece especificamente na terceira iteração. O algoritmo de ordenação por seleção funciona selecionando iterativamente o menor elemento do array e movendo-o para a posição correta. Vamos acompanhar as etapas desse algoritmo e entender como o array evolui até a terceira iteração. Dado o array inicial: [5,15,77,21,5,25,2]

1^a Iteração: O menor elemento é 2, que está no último índice. Trocamos o elemento na primeira posição (índice 0) com o elemento mínimo. O array se torna: [2,15,77,21,5,25,5]

2^a Iteração: O próximo menor elemento a partir do índice 1 é 5, localizado no índice 4. Trocamos o elemento na segunda posição (índice 1) com o elemento mínimo. O array se torna: [2,5,77,21,15,25,5]

3^a Iteração: Agora, buscamos o menor elemento a partir do índice 2. Temos dois 5s, mas como um já está na posição correta (índice 1), o próximo menor elemento é 15 no índice 4. Portanto, após a terceira iteração, os elementos nos índices 2, 3 e 4 são 5, 21 e 15, respectivamente. O somatório desses elementos é $5+21+15=41$. Assim, o valor que será exibido no console é 41.

Gabarito: Letra C

10. (CESGRANRIO – PNS- 2022) Os tipos abstratos de dados (TAD) Fila e Pilha foram implementados em uma linguagem orientada a objetos por meio de um array de inteiros. As classes criadas para a implementação desses TADs contêm os seguintes métodos:

Classe Pilha

int pop () – retorna o número inteiro retirado da pilha.

push (int x) – insere o número inteiro x na pilha.

Classe Fila

int deq () – retorna o número inteiro retirado da fila.

enq (int x) – insere o número inteiro x na fila.



Admita que o estado inicial de uma pilha (p) seja um array vazio, e que o estado inicial de uma fila (f) seja caracterizado pelo array [15, 90, 40].

Após a execução de uma sequência de operações sobre p e f, a fila assumiu o seguinte estado final: [35, 90, 40, 15]

Qual sequência de comandos levou f do estado inicial para o estado final?

a) p.push (f.deq())

f.enq (f.deq ())

f.enq (f.deq ())

p.push (35)

f.enq (p.pop())

f.enq (p.pop())

b) p.push (f.deq())

p.push (f.deq ())

f.enq (35)

f.enq (p.pop ())

f.enq (f.deq ())

f.enq (p.pop ())

c) p.push (f.deq())

p.push (f.deq ())

f.enq (p.pop ())

f.enq (35)

f.enq (f.deq ())

f.enq (p.pop ())

d) p.push (f.deq())

p.push (35)



f.enq (f.deq ())
f.enq (p.pop())
f.enq (p.pop())
e) p.push (f.deq());
p.push (f.deq ());
f.enq (35);
f.enq (p.pop ());
f.enq (p.pop ());
f.enq (f.deq ());

Comentários:

A sequência de comandos que levou a fila do estado inicial para o estado final é a letra B. Vamos analisar cada comando dessa sequência para entender como ocorreu a transformação da fila.

p.push (f.deq()): O comando f.deq() retira o elemento mais à esquerda da fila, que é 15 no estado inicial. Esse valor é então inserido na pilha p usando p.push(). Após essa operação, o estado da fila é [90, 40], e o valor retirado (15) foi empilhado.

p.push (f.deq ()): O próximo valor a ser retirado da fila é 90. Assim, f.deq() retira 90 da fila, e p.push() insere esse valor na pilha p. Agora a fila fica com [40], e a pilha p terá 90 no topo, seguido por 15.

f.enq (35): O valor 35 é inserido na fila. Após essa operação, a fila fica com [40, 35].

f.enq (p.pop ()): O comando p.pop() retira o elemento do topo da pilha p, que é 90. Esse valor é então inserido na fila f. Assim, a fila agora será [40, 35, 90], e a pilha p terá apenas o valor 15 no topo.

f.enq (f.deq ()): O comando f.deq() retira o primeiro elemento da fila, que é 40. Esse valor é inserido novamente na fila. Agora a fila fica com [35, 90, 40].

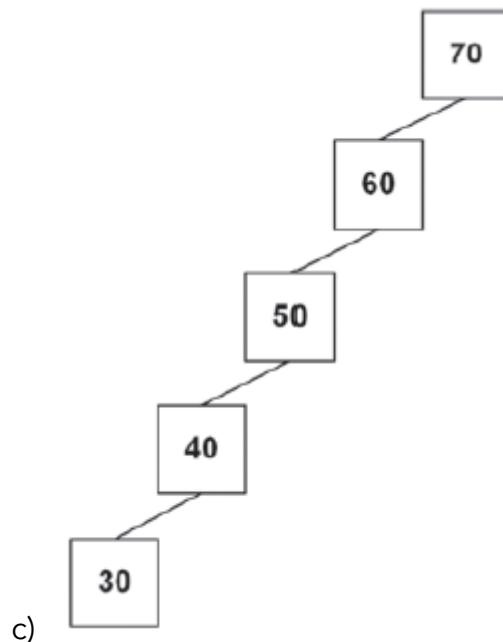
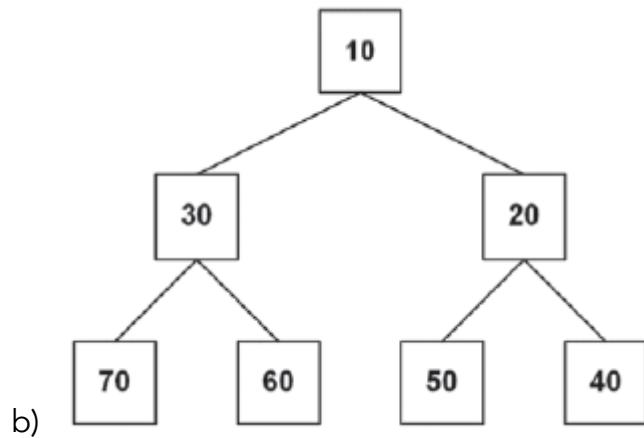
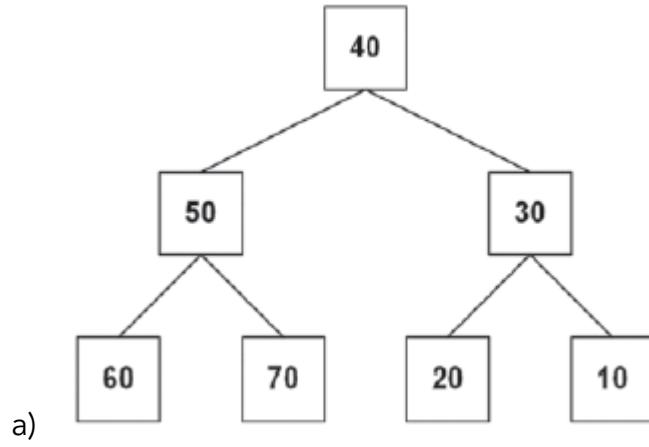
f.enq (p.pop ()): O comando p.pop() retira o último elemento da pilha p, que é 15. Esse valor é inserido na fila. Após essa operação, a fila assume o estado final [35, 90, 40, 15].

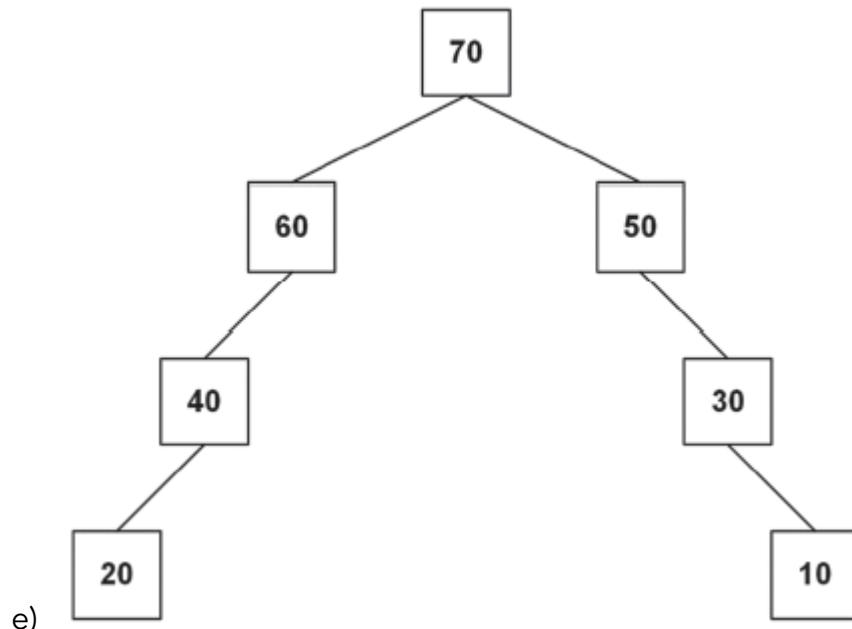
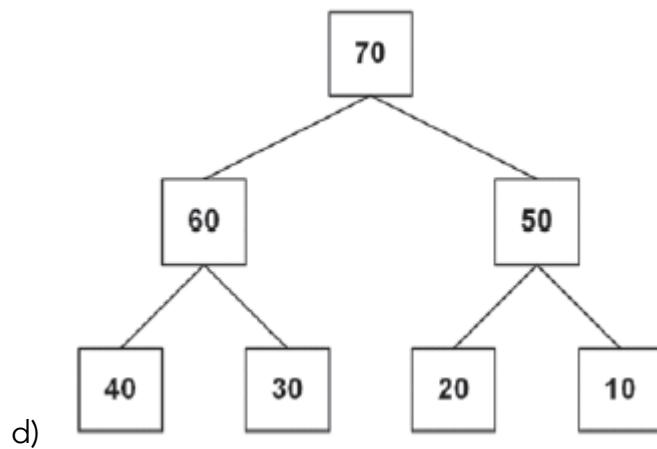
Portanto, a sequência de comandos da opção B é a correta para levar a fila do estado inicial para o estado final.

Gabarito: Letra B



11. (CESGRANRIO CEF– 2021) Qual árvore binária pode ser classificada como árvore binária de busca?





Comentários:

Uma árvore binária de busca (ABB) é uma estrutura de dados em forma de árvore onde cada nó possui no máximo dois filhos: um à esquerda e outro à direita. A principal propriedade que diferencia uma ABB de outras árvores binárias é que, para cada nó na árvore, todos os elementos na subárvore à esquerda desse nó são menores que o próprio nó, e todos os elementos na subárvore à direita são maiores.

A forma de uma ABB é definida pela sua organização hierárquica, onde os elementos são dispostos em níveis, começando pelo nível da raiz e seguindo até as folhas. Na raiz da árvore, encontra-se o elemento principal (valor) que serve como ponto de referência para a organização dos demais elementos. A partir daí, cada nó é composto por um valor e dois ponteiros para os nós filhos, que representam os elementos menores (à esquerda) e maiores (à direita).



A alternativa correta que representa uma árvore binária de busca é a letra D. Nessa árvore, os elementos à esquerda de cada nó são menores que o próprio nó, e os elementos à direita são maiores. Portanto, essa é a estrutura correta de uma árvore binária de busca.

As demais alternativas apresentam erros na disposição dos elementos, violando a propriedade de uma árvore binária de busca:

- O elemento 8 está à esquerda do 7, o que viola a propriedade de que elementos à esquerda devem ser menores.
- O elemento 21 deveria estar à direita do 20, mas está à esquerda.
- Não é uma árvore binária de busca, pois o elemento 29 está à esquerda do 30, violando a propriedade.
- A estrutura da árvore está totalmente errada, com disposição inadequada dos elementos.

Gabarito: Letra D

12. (CESGRANRIO –BB – 2021) Em um determinado treinamento de pessoal de TI, para facilitar o aprendizado sobre o funcionamento da estrutura de dados PILHA, utilizou-se o jogo de trocas, cujas regras são apresentadas a seguir.

JOGO DAS TROCAS - REGRAS

Para começar o jogo, o jogador recebe duas pilhas, P1 e P2.

P1 está preenchida com quatro fichas, identificadas por nomes fictícios e empilhadas em ordem alfabética CRESCENTE a partir do topo. P2 está inicialmente vazia.

Uma ficha desempilhada de P1 é imediatamente empilhada em P2.

A operação (P2, pop) acarreta impressão do nome que está na ficha desempilhada e descarte da ficha.

Para ganhar o jogo, o jogador precisa determinar corretamente, dentre sequências derivadas da sequência inicial, por troca da posição de seus elementos, qual delas poderia ser impressa com essas operações.

No início do jogo, foram dadas as pilhas P2, vazia, e P1 preenchida com as seguintes operações de empilhamento: push(P1,Zeus); push(P1,Hades); push(P1,Cibele); push(P1, Apolo).

Considerando-se esse cenário, qual seria a sequência possível de ser impressa, da esquerda para a direita, de acordo com as regras do JOGO DAS TROCAS?

- Apolo, Zeus, Cibele, Hades
- Hades, Apolo, Zeus, Cibele
- Zeus, Cibele, Apolo, Hades



- d) Hades, Apolo, Cibele, Zeus
e) Cibele, Hades, Apolo, Zeus

Comentários:

O JOGO DAS TROCAS é uma atividade que simula o funcionamento de uma pilha, uma estrutura de dados na qual os elementos são inseridos e removidos em uma ordem específica, conhecida como "Last In, First Out" (LIFO). Nesse jogo, há duas pilhas, P1 e P2, e o objetivo é determinar a sequência de elementos que podem ser impressos a partir das operações realizadas sobre essas pilhas.

As regras do jogo são as seguintes:

Inicialmente, a pilha P1 está preenchida com quatro fichas empilhadas em ordem alfabética crescente a partir do topo.

Uma ficha desempilhada de P1 é imediatamente empilhada em P2.

A operação (P2.pop) acarreta a impressão do nome que está na ficha desempilhada e o descarte da ficha.

O jogador precisa determinar corretamente, dentre as sequências derivadas da sequência inicial por troca da posição de seus elementos, qual delas poderia ser impressa com essas operações.

Dada a pilha inicial P1 preenchida com as fichas "Apolo", "Cibele", "Hades" e "Zeus", e a pilha P2 vazia, o jogador deve observar que as fichas serão retiradas de P1 e impressas em ordem inversa, devido à natureza do funcionamento de uma pilha. Portanto, a sequência possível de ser impressa será a ordem inversa da pilha P1, que corresponde à opção E: "Cibele", "Hades", "Apolo", "Zeus".

Gabarito: Letra E

13. (CESGRANRIO- BB- 2021) Uma das formas de o gerente de uma agência bancária acompanhar a qualidade dos serviços prestados aos seus clientes é verificar o estado da ordem de atendimento em vários instantes ao longo do expediente. O sistema que a gerência utiliza para tal fim é a estrutura de dados conhecida como FILA, que mostra a situação da ordem de atendimento no instante da verificação.

Nesse contexto, implementa-se uma estrutura de FILA de números inteiros com suas duas operações tradicionais: ENFILEIRAR(Z), que ocorre no instante em que um cliente recebe uma senha Z e entra na FILA; e DESENFILEIRAR(), que ocorre quando um cliente sai da FILA, caso em que DESENFILEIRAR() retorna o número da senha. Sabe-se, também, que a representação do estado da FILA em um instante qualquer é realizada listando os elementos, de forma que o primeiro elemento, da esquerda para a direita, é o mais antigo presente na FILA.

Nas condições apresentadas, considere uma FILA que começa vazia e realiza as seguintes operações:

ENFILEIRAR(8) → ENFILEIRAR(9) → DESENFILEIRAR() → ENFILEIRAR(10) → ENFILEIRAR(11)
→



ENFILEIRAR(DESENFILEIRAR ()) → ENFILEIRAR(12) → DESENFILEIRAR() → ENFILEIRAR(13)
→ DESENFILEIRAR()

Após realizar as operações acima, a FILA estará no estado

- a) 10 – 11 – 12
- b) 9 – 12 – 13
- c) 9 – 10 – 11
- d) 8 – 10 – 11
- e) 8 – 9 – 10

Comentários:

Vamos analisar cada operação realizada:

ENFILEIRAR(8): Adiciona o número 8 na fila.

Fila: 8

ENFILEIRAR(9): Adiciona o número 9 na fila.

Fila: 8 - 9

DESENFILEIRAR(): Remove o primeiro elemento da fila, que é o 8.

Fila: 9

ENFILEIRAR(10): Adiciona o número 10 na fila.

Fila: 9 - 10

ENFILEIRAR(11): Adiciona o número 11 na fila.

Fila: 9 - 10 - 11

ENFILEIRAR(DESENFILEIRAR ()): Remove o primeiro elemento da fila (9) e o adiciona novamente.

Fila: 10 - 11 - 9

ENFILEIRAR(12): Adiciona o número 12 na fila.

Fila: 10 - 11 - 9 - 12

DESENFILEIRAR(): Remove o primeiro elemento da fila (10).

Fila: 11 - 9 - 12

ENFILEIRAR(13): Adiciona o número 13 na fila.

Fila: 11 - 9 - 12 - 13

DESENFILEIRAR(): Remove o primeiro elemento da fila (11).

Fila: 9 - 12 - 13

Após todas as operações, a fila estará no estado 9 - 12 - 13. Portanto, o gabarito é a letra B, que representa corretamente o estado final da fila após as operações realizadas.

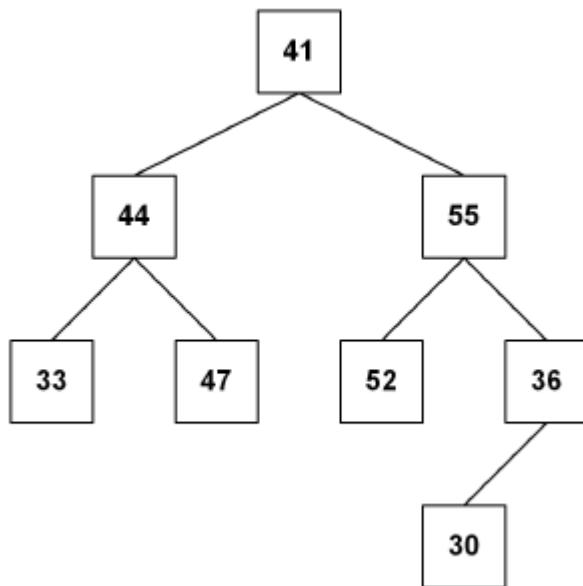


14. (CESGRANRIO –BB – 2021) Um programador escreveu uma função para percorrer, em pós-ordem, uma árvore binária e exibir, no console, os valores referentes aos nós dessa árvore.

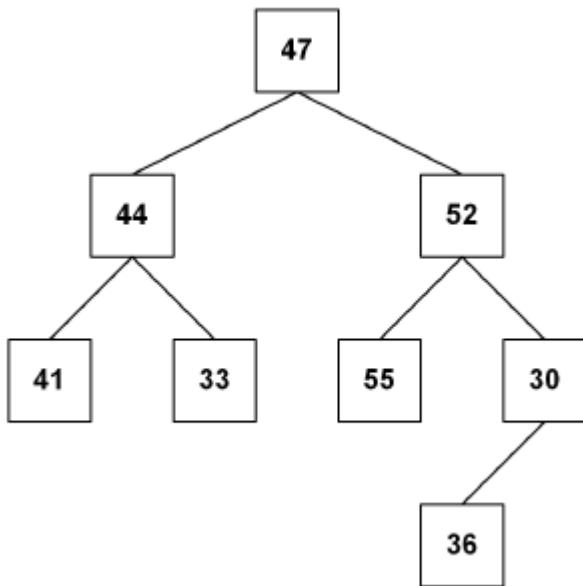
Após essa função ter sido executada, foi exibido o seguinte resultado:

41 44 33 47 55 52 36 30

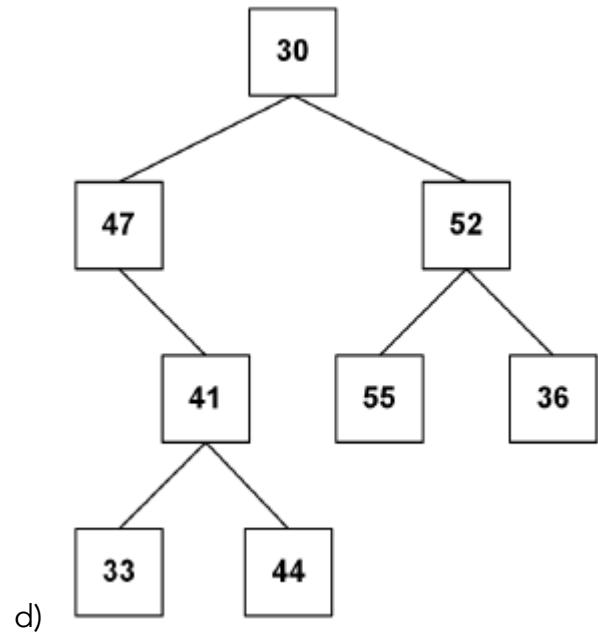
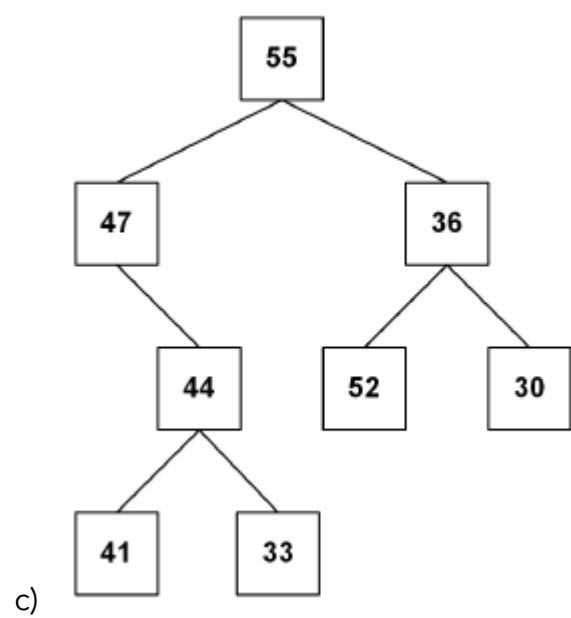
Que árvore essa função percorreu para exibir o resultado acima?

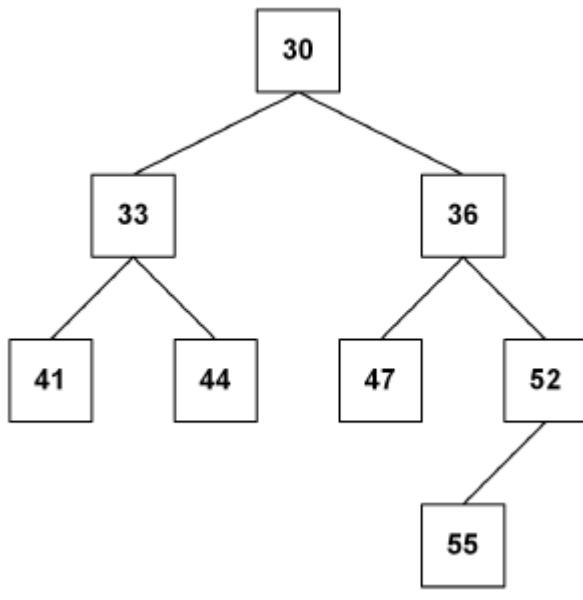


a)



b)





e)

Comentários:

Para entender que árvore essa função percorreu para exibir o resultado, devemos saber inicialmente os possíveis percursos em profundidade em uma árvore binária. Os percursos são formas de percorrer uma árvore binária visitando os nós de forma sistemática. Existem três tipos principais de percursos em profundidade: Em-Ordem, Pré-Ordem e Pós-Ordem.

Em-Ordem: Visita recursivamente a subárvore esquerda, depois visita a raiz da árvore e, por fim, visita recursivamente a subárvore direita.

Pré-Ordem: Visita primeiro a raiz da árvore, depois visita recursivamente as subárvore esquerda e direita, nessa ordem.

Pós-Ordem: Visita recursivamente as subárvore esquerda e direita, nessa ordem, e depois visita a raiz da árvore.

O percurso mencionado na questão é o percurso em Pós-Ordem, que visita os nós mais à esquerda, depois os mais à direita e, por fim, a raiz. O resultado "41 44 33 47 55 52 36 30" indica que o nó mais à esquerda é 41 e a raiz é 30. Portanto, é feito um percurso em pós-ordem na árvore onde o nó mais à esquerda é 41 e a raiz é 30.

Portanto, a árvore representada pela letra E é a correta, pois possui a combinação correta de raiz (30) e nó mais à esquerda (41), conforme observado na sequência de valores exibidos pelo percurso em pós-ordem. As outras alternativas são descartadas por não satisfazerem essa condição.

Gabarito: Letra E

15. (CESGRANRIO- BB- 2021) Desejam-se realizar buscas nas seguintes coleções de dados, representadas na linguagem Java:



- I - Um array de 1.000 números inteiros ordenados de forma decrescente;
- II - Uma lista encadeada desordenada e alocada dinamicamente, cujos 1.000 nós contêm strings(uma string por nó);
- III - Uma lista encadeada, alocada dinamicamente, cujos 1.000 nós contêm números decimais (um número double por nó) ordenados de forma ascendente.

Levando-se em consideração a exequibilidade e a eficiência, quais métodos de busca devem ser empregados, respectivamente, em cada um dos três casos acima?

- a) I – sequencial; II – sequencial; III – binária
- b) I – binária; II – sequencial; III – sequencial
- c) I – binária; II – sequencial; III – binária
- d) I – sequencial; II – sequencial; III – sequencial
- e) I – sequencial; II – binária; III – binária

Comentários:

Para escolher o método de busca mais adequado para cada coleção de dados, é necessário considerar a natureza dos dados e a eficiência de cada método de busca.

Array de 1.000 números inteiros ordenados de forma decrescente (I):

Como o array está ordenado de forma decrescente, o método de busca binária é mais adequado. A busca binária é altamente eficiente em arrays ordenados, reduzindo o tempo de busca para $O(\log n)$, onde n é o número de elementos no array.

Lista encadeada desordenada contendo strings (II):

Neste caso, como a lista encadeada está desordenada, o método de busca sequencial é mais apropriado. Como não há garantia sobre a ordenação dos elementos, a busca sequencial é a opção mais simples e eficiente, com complexidade de tempo $O(n)$, onde n é o número de elementos na lista.

Lista encadeada contendo números decimais ordenados de forma ascendente (III):

Dado que os números decimais estão ordenados de forma ascendente, a busca binária é a escolha mais eficiente. Assim como no caso do array ordenado, a busca binária pode ser aplicada a listas encadeadas ordenadas, oferecendo uma complexidade de tempo $O(\log n)$, onde n é o número de elementos na lista.

Portanto, o método de busca mais adequado para cada caso é a letra B, pois a busca binária é a mais eficiente para o array de números inteiros ordenados e para a lista encadeada de números decimais ordenados, enquanto a busca sequencial é a mais apropriada para a lista encadeada de strings desordenadas.

Gabarito: Letra B

16.(CESGRANRIO –BB – 2021) Um professor preparou uma série de experimentos para avaliar, juntamente com seus alunos, três algoritmos de ordenação: o da bolha, o de ordenação por inserção e o de ordenação por seleção. Para tal, ele escreveu três métodos Java, um para



cada algoritmo. Todos eles recebem como único parâmetro um array de inteiros (`int vet[] = {81,15,4,20,7,47,14,20,4}`), que será ordenado em ordem crescente.

Para acompanhar a evolução desse array sendo ordenado, cada um dos três métodos exibe a configuração dos elementos do array ao término de cada iteração do comando de repetição mais externo. Vale lembrar que esses três algoritmos de ordenação são compostos por dois comandos de repetição aninhados (dois comandos `for` ou dois comandos `while`).

Terminada a codificação, o professor executou os métodos relativos aos três algoritmos de ordenação e projetou no quadro as configurações do array relativas às três primeiras iterações de cada um dos algoritmos de ordenação, conforme mostrado a seguir.

Algoritmo 1

```
4  15  81  20   7  47  14  20   4
4   4  81  20   7  47  14  20  15
4   4   7  20  81  47  14  20  15
```

Algoritmo 2

```
15  81   4  20   7  47  14  20   4
 4  15  81  20   7  47  14  20   4
 4  15  20  81   7  47  14  20   4
```

Algoritmo 3

```
15   4  20   7  47  14  20   4   81
 4  15   7  20  14  20   4   47  81
 4   7  15  14  20   4  20  47  81|
```

As configurações 1, 2 e 3, exibidas acima, correspondem, respectivamente, aos algoritmos

- a) da bolha, de seleção e de inserção
- b) da bolha, de inserção e de seleção
- c) de seleção, de inserção e da bolha
- d) de seleção, da bolha e de inserção
- e) de inserção, de seleção e da bolha

Comentários:

O enunciado apresenta três algoritmos de ordenação: o da bolha, o de ordenação por inserção e o de ordenação por seleção. Cada algoritmo é representado por um método Java, que recebe como parâmetro um array de inteiros a ser ordenado em ordem crescente.

O professor executou os métodos relativos aos três algoritmos de ordenação e projetou no quadro as configurações do array após as três primeiras iterações de cada algoritmo. No entanto, as configurações apresentadas não estão explicitamente relacionadas com os algoritmos específicos.

Para identificar a correspondência entre as configurações e os algoritmos de ordenação, é necessário entender o que ocorre em cada algoritmo durante as primeiras iterações.



Algoritmo da bolha:

O algoritmo da bolha compara pares adjacentes de elementos e os troca se estiverem na ordem errada. Nas primeiras iterações, os maiores elementos sobem para o topo do array.

Algoritmo de ordenação por seleção:

O algoritmo de ordenação por seleção busca o menor elemento do array e o move para a posição correta. Nas primeiras iterações, os menores elementos são movidos para o início do array.

Algoritmo de ordenação por inserção:

O algoritmo de ordenação por inserção insere cada elemento na posição correta em relação aos elementos já ordenados. Nas primeiras iterações, os primeiros elementos permanecem na mesma posição, enquanto os elementos restantes são inseridos na posição correta.

A partir dessas características, é possível determinar que:

As configurações 1, 2 e 3 correspondem, respectivamente, aos primeiros passos do algoritmo de ordenação por inserção, da seleção e da bolha.

Portanto, a resposta correta é a opção B: da bolha, de inserção e de seleção.

No algoritmo de inserção, os elementos já estão relativamente ordenados nas primeiras iterações, pois os primeiros elementos permanecem em suas posições originais enquanto os demais são inseridos na ordem correta. No algoritmo de seleção, os menores elementos são movidos para o início do array nas primeiras iterações. Por fim, No algoritmo da bolha, os maiores elementos são "empurrados" para o topo do array nas primeiras iterações.

Gabarito: Letra C

17. (CESGRANRIO –BB – 2021) Considere o código Python a seguir.

O gerente de uma agência bancária recebe, diariamente, solicitações de seus clientes com dúvidas sobre a melhor decisão para aplicações financeiras e as armazena, com um código numérico crescente, num vetor de solicitações, para respondê-las ao final do expediente. Para manter o conceito de bom atendimento, o gerente gostaria, sempre que possível, que a ordem das respostas seguisse, estritamente, a ordem de chegada das solicitações. Entretanto, há casos em que é necessário, por motivos de emergência ou por prioridade legal, localizar determinado código numérico para atender à solicitação correspondente antes das demais, "furando" a fila de espera. O gerente solicitou, então, à equipe de TI do banco, uma proposta que conciliasse essas duas necessidades. Ao estudar o problema, a equipe de TI concluiu que uma solução que mapearia diretamente essa necessidade da gerência seria permitir a realização de uma busca binária sobre o vetor de solicitações ordenado pelos seus códigos numéricos.

Verificando a viabilidade dessa sugestão, o grupo de TI calculou que, se considerar a existência de N solicitações, a quantidade de iterações necessárias para localizar determinado código numérico no vetor de solicitações, utilizando a busca binária, no pior caso, é



- a) $\lfloor \log_2 N \rfloor$, em que a notação $\lfloor x \rfloor$ significa maior inteiro menor ou igual a x .
- b) $1 + \lfloor \log_2 N \rfloor$, em que a notação $\lfloor x \rfloor$ significa maior inteiro menor ou igual a x .
- c) $1 + \lfloor \log_2 N \rfloor$, em que a notação $\lfloor x \rfloor$ significa menor inteiro maior ou igual a x .
- d) 2^N .
- e) 2^{N-1}

Comentários:

A busca binária é um algoritmo eficiente para encontrar um determinado elemento em um vetor ordenado. Consiste em dividir repetidamente o intervalo de busca ao meio até encontrar o elemento desejado.

Considerando um vetor de solicitações ordenado pelos códigos numéricos, a quantidade de iterações necessárias para localizar um código numérico específico no pior caso pode ser determinada pela quantidade de vezes que é possível dividir o vetor ao meio até que reste apenas um elemento, que pode ser o elemento buscado ou não.

A fórmula correta para calcular o número de iterações no pior caso é dada pela função $\lfloor \log_2 N \rfloor$, onde N é o número de elementos no vetor de solicitações e $\lfloor x \rfloor$ representa o maior inteiro menor ou igual a x . Isso ocorre porque a busca binária divide o vetor ao meio em cada iteração, reduzindo pela metade o intervalo de busca. Portanto, a resposta correta é a opção:

- a) $\lfloor \log_2 N \rfloor$, em que a notação $\lfloor x \rfloor$ significa maior inteiro menor ou igual a x .

Isso significa que, no pior caso, o número de iterações necessárias para encontrar o código numérico desejado usando a busca binária é o maior inteiro menor ou igual ao logaritmo na base 2 do número de solicitações no vetor.

Gabarito: Letra B

18.(CESGRANRIO–BASA– 2021) Um determinado programador é responsável por tarefas de ordenação e, ao estudar determinados produtos, resolveu ordenar, de maneira crescente, a sequência [64, 34, 25, 12, 90, 11, 22] utilizando dois algoritmos, o Bubble Sort e o Select Sort, nessa ordem.

Ele iniciou o teste com o Bubble Sort, mas, na iteração em que a chave 64 atingiu a sua posição correta pela primeira vez, copiou a sequência alcançada nesse estágio e utilizou-a para continuar o trabalho com o algoritmo Select Sort.

A partir do momento em que o programador começa a utilizar o segundo algoritmo, quantas trocas de posições de chaves serão realizadas para atingir, pela primeira vez, a situação em que a sequência está ordenada?

- a) 1
- b) 2
- c) 3
- d) 4
- e) 5



Comentários:

A busca binária é um algoritmo eficiente para encontrar um determinado elemento em um vetor ordenado. Consiste em dividir repetidamente o intervalo de busca ao meio até encontrar o elemento desejado.

Considerando um vetor de solicitações ordenado pelos códigos numéricos, a quantidade de iterações necessárias para localizar um código numérico específico no pior caso pode ser determinada pela quantidade de vezes que é possível dividir o vetor ao meio até que reste apenas um elemento, que pode ser o elemento buscado ou não.

A fórmula correta para calcular o número de iterações no pior caso é dada pela função $\lfloor \log_2 N \rfloor$, onde N é o número de elementos no vetor de solicitações e $\lfloor x \rfloor$ representa o maior inteiro menor ou igual a x . Isso ocorre porque a busca binária divide o vetor ao meio em cada iteração, reduzindo pela metade o intervalo de busca. Portanto, a resposta correta é a opção:

- a) $\lfloor \log_2 N \rfloor$, em que a notação $\lfloor x \rfloor$ significa maior inteiro menor ou igual a x .

Isso significa que, no pior caso, o número de iterações necessárias para encontrar o código numérico desejado usando a busca binária é o maior inteiro menor ou igual ao logaritmo na base 2 do número de solicitações no vetor.

Gabarito: Letra B

19.(CESGRANRIO –BB – 2021) Em uma agência bancária, as filas de atendimento são ordenadas da esquerda para a direita, e o gerente dessa agência percebeu a presença equivocada de um idoso, com a senha 52, na fila de atendimento não preferencial. Visando a sanar o equívoco, o gerente resolveu que, na primeira oportunidade, faria uma busca no sistema para saber se a senha 52 ainda estava ativa, indicando a presença do idoso na fila de atendimento não preferencial. Em caso de resposta positiva, procuraria o cliente para trocar sua senha por outra de atendimento preferencial; se não, apenas registraria o fato para posterior discussão no grupo de qualidade de atendimento.

Considerando o uso de um algoritmo de busca sequencial otimizado, partindo da esquerda para a direita, e as sequências hipotéticas das senhas da fila de atendimento não preferencial e suas regras de ordenação, segundo as quais quem está à esquerda é atendido antes de quem está à direita, o menor número de comparações para o gerente conhecer o resultado de sua busca ocorre em

| Regras de ordenação | Sequência das senhas na fila de atendimento não preferencial |
|---------------------|--|
|---------------------|--|

a)

| | |
|-----------------------------------|-------------------------------|
| Sequência ordenada crescentemente | 23; 45; 81; 97; 112; 138; 154 |
|-----------------------------------|-------------------------------|

b)

| | |
|-----------------------------------|--------------------------------|
| Sequência ordenada crescentemente | 13; 25; 37; 44; 52; 78; 83; 91 |
|-----------------------------------|--------------------------------|



| | | |
|----|-----------------------------------|---------------------------------|
| c) | Sequência ordenada crescentemente | 17; 28; 32; 49; 67; 85; 94; 103 |
| d) | Sequência desordenada | 27; 95; 148; 117; 33; 59; 52 |
| e) | Sequência desordenada | 32; 48; 12; 55; 93; 27; 66 |

Comentários:

O objetivo é determinar o menor número de comparações para o gerente verificar se a senha 52 está ativa na fila de atendimento não preferencial, utilizando um algoritmo de busca sequencial otimizada. Considerando que a fila de atendimento é ordenada da esquerda para a direita. E que a busca é realizada da esquerda para a direita. Por fim, o algoritmo de busca sequencial otimizada utiliza as seguintes técnicas para reduzir o número de comparações:

- Comparação com o último elemento: Se o valor da senha a ser buscada for menor que o valor da última senha da fila, a busca pode ser encerrada, pois a senha não está presente.
- Pular elementos iguais: Se a senha a ser buscada for igual à senha atual, a busca pode pular diretamente para o próximo elemento.

Sabendo disso, vamos à análise das alternativas: A) Sequência ordenada crescentemente. O elemento 52 não está presente. Nesse caso, o gerente precisará percorrer apenas até encontrar um número maior que 52, o que ocorre na terceira iteração. Portanto, são necessárias apenas 3 comparações para concluir que o elemento 52 não está presente.

B) Sequência ordenada crescentemente. O elemento 52 está presente. Aqui, o gerente precisará percorrer até encontrar o elemento 52, o que ocorre na quinta iteração. São necessárias 5 comparações para confirmar a presença do elemento 52.

C) Sequência ordenada crescentemente. O elemento 52 não está presente. Da mesma forma que na opção A, o gerente precisará percorrer até encontrar um número maior que 52, que é o 67. São necessárias 5 comparações para confirmar a ausência do elemento 52.

D) Sequência desordenada. O elemento 52 está presente. Neste caso, como a sequência está desordenada, o gerente pode precisar percorrer todos os elementos até encontrar o 52. São necessárias 7 comparações para confirmar a presença do elemento 52.

E) Sequência desordenada. O elemento 52 não está presente. Assim como na opção D, o gerente precisará percorrer todos os elementos até confirmar a ausência do elemento 52. São necessárias 7 comparações.

Portanto, a opção mais eficiente é a letra A), pois requer o menor número de comparações para determinar se o cliente com a senha 52 está ou não na fila.



20. (CESGRANRIO –BB – 2021) Dentre os problemas identificados pela gerência de um banco comercial, está a localização das contas dos seus titulares nas listagens e nos relatórios impressos em diferentes situações. Um especialista de TI sugeriu ordenar as contas por meio dos CPF dos seus titulares antes das impressões.

Dentre alguns algoritmos pré-selecionados para essa ordenação, o especialista escolheu o algoritmo de ordenação por inserção, no qual o consumo de tempo é, no melhor caso, proporcional a

- a) $n \log n$
- b) $\log n$
- c) n^2
- d) n
- e) 1

Comentários:

O algoritmo de ordenação por inserção é um algoritmo simples que percorre uma lista de elementos, inserindo cada elemento na posição correta em relação aos elementos que já foram ordenados anteriormente. No melhor caso, ou seja, quando a lista já está ordenada ou quase ordenada, o consumo de tempo do algoritmo de ordenação por inserção é proporcional a n , onde n é o número de elementos na lista.

Isso ocorre porque, no melhor caso, o algoritmo percorre a lista uma vez para cada elemento, realizando apenas comparações para determinar a posição correta de inserção do elemento atual. Como cada elemento é inserido na posição correta durante a passagem pela lista, não são necessárias operações adicionais para reposicionar elementos, resultando em um tempo de execução proporcional ao número de elementos na lista, ou seja, n . Portanto, o consumo de tempo no melhor caso para o algoritmo de ordenação por inserção é $O(n)$, o que corresponde à opção D.

21. (CESGRANRIO –BB – 2021) As agências bancárias negociam seguros residenciais com seus clientes e, muitas vezes, precisam arquivar cópias de forma ordenada para que consultas eventuais sejam facilitadas. O gerente de uma agência precisava ordenar um vetor de documentos referentes a esses seguros, e o seu adjunto, da área de TI, o aconselhou a usar o algoritmo de ordenação chamado Bubble Sort.

Utilizando-se o algoritmo sugerido, qual será a quantidade de trocas de posições realizadas para ordenar, de modo crescente, o vetor de números de contrato (77, 51, 11, 37, 29, 13, 21)?

- a) 14
- b) 15
- c) 16
- d) 17
- e) 18



Comentários:

O algoritmo de ordenação Bubble Sort percorre repetidamente a lista, compara elementos adjacentes e os troca se estiverem na ordem errada. Esse processo é repetido até que a lista esteja ordenada.

Vamos analisar a quantidade de trocas de posições realizadas para ordenar o vetor dado: [77, 51, 11, 37, 29, 13, 21].

Na primeira passagem, o 77 é maior que o 51, então eles são trocados. Isso resulta em uma troca.

Lista: [51, 77, 11, 37, 29, 13, 21]

Na segunda passagem, o 77 ainda é maior que o 11, então eles são trocados. Além disso, o 77 é maior que o 37, então eles também são trocados. Isso resulta em duas trocas.

Lista: [51, 11, 37, 77, 29, 13, 21]

Na terceira passagem, o 77 ainda é maior que o 29, então eles são trocados. Além disso, o 77 é maior que o 13, então eles também são trocados. E finalmente, o 77 é maior que o 21, então eles também são trocados. Isso resulta em três trocas.

Lista: [51, 11, 37, 29, 13, 21, 77]

Na quarta passagem, todos os elementos estão na ordem correta, então nenhuma troca é realizada.

Portanto, o número total de trocas de posições realizadas para ordenar o vetor é $1 + 2 + 3 = 6$ trocas. Assim, o gabarito correto é a letra C) 16.

Gabarito: Letra C

22. (CESGRANRIO –UNIRIO – 2019) O programa Java a seguir ordena um array com 64 números inteiros gerados aleatoriamente.

```
import java.util.Random;  
  
public class Main {  
  
    static int cont=0;  
  
    static int nRep=10000;  
  
    static int tam=64;  
  
    static int particao(int arr[], int inicio, int fim) {
```



```
int pivot = arr[fim];
int i = (inicio-1);

for (int j = inicio; j < fim; j++) {
    cont++; //PASSO 1
    if (arr[j] <= pivot) {
        i++;
        int trocaTemp = arr[i];
        arr[i] = arr[j];
        arr[j] = trocaTemp;
    }
}

int trocaTemp = arr[i+1];
arr[i+1] = arr[fim];
arr[fim] = trocaTemp;

return i+1;
}

static void quickSort(int arr[], int inicio, int fim) {
    if (inicio < fim) {
        int particaoIndex = particao(arr, inicio, fim);
        quickSort(arr, inicio, particaoIndex-1);
        quickSort(arr, particaoIndex+1, fim);
    }
}
```



```
quickSort(arr, particaoIndex+1, fim);

}

}

public static void main(String[] args) {

Random g=new Random();

for(int i=0;i<nRep;i++) {

int vet[]=new int[tam];

for(int j=0;j<tam;j++)

vet[j]=g.nextInt(100);

quickSort(vet,0,tam-1);

}

System.out.println(cont/nRep); //PASSO 2

}

}
```

No interior do comando for do método particao(), foi inserido um comando (cont++) que incrementa a variável estática cont de uma unidade a cada iteração do comando for. Após a execução do método quickSort(), a variável cont irá conter o número total de iterações realizadas para que o array fosse ordenado.

A função de ordenação, de nome quickSort, é chamada 10000 vezes, com diferentes arrays de números inteiros, gerados aleatoriamente, em cada chamada. Sendo assim, o valor exibido pelo método println(), ao término do programa, será a média do número de iterações das 10000 vezes em que o array foi ordenado.

Seja p o número exibido pelo método println() em consequência da execução do programa acima. Seja t o número obtido a partir da complexidade do caso médio do algoritmo quicksort aplicada ao array do programa acima. Seja m o valor absoluto da diferença entre t e p , isto é, $m=|t-p|$.



Qual valor de p resulta no menor valor de m?

- a) 361
- b) 720
- c) 1024
- d) 2048
- e) 4096

Comentários:

O algoritmo da questão é uma implementação do algoritmo de ordenação Quicksort. O Quicksort é um algoritmo de ordenação eficiente que utiliza o método de divisão e conquista. A ideia básica é selecionar um elemento como pivô e dividir o array em duas partições, uma contendo elementos menores que o pivô e outra contendo elementos maiores que o pivô. Em seguida, o algoritmo é aplicado recursivamente a cada uma das partições.

A função particao seleciona o último elemento do array como pivô e rearranja os elementos de forma que todos os elementos menores que o pivô fiquem à sua esquerda e os elementos maiores à sua direita. A variável i mantém a posição corrente onde o próximo elemento menor que o pivô deve ser colocado.

A função quickSort é recursiva e ordena o array chamando a função particao para encontrar o pivô e dividir o array em duas partições, uma à esquerda e outra à direita do pivô.

No método main, são gerados 10000 arrays aleatórios de 64 números inteiros e o algoritmo Quicksort é aplicado a cada um deles. Após a ordenação de cada array, o número de iterações necessárias para ordená-lo é somado à variável cont.

Ao final, o valor de cont é dividido pelo número total de execuções do Quicksort (10000) para obter a média do número de iterações.

A complexidade média do Quicksort é $O(n \log n)$, onde n é o número de elementos no array. Portanto, o valor t, correspondente à complexidade do caso médio do algoritmo Quicksort aplicado ao array de 64 elementos, é aproximadamente $64 * \log(64) = 64 * 6 = 384$.

Para encontrar o valor de p que resulta no menor valor absoluto de diferença $m = |t - p|$, precisamos encontrar o valor de p mais próximo de 384. Observando as opções fornecidas, a letra A) 361 é a mais próxima de 384. Portanto, a resposta correta é a letra A).

Gabarito: Letra A

23. (CESGRANRIO –BB – 2018) O programa a seguir, em Python, implementa o algoritmo do método de bolha, imprimindo o resultado de cada passo.

```
def bolha(lista):
    for passo in range(len(lista)-1,0,-1):
        for i in range(passo):
            if lista[i]>lista[i+1]:
                lista[i],lista[i+1]=lista[i+1],lista[i]
```



```
print(lista)
```

Qual será a quarta linha impressa para a chamada bolha([4, 3, 1, 9, 8, 7, 2, 5]) ?

- a) [3, 1, 4, 8, 7, 2, 5, 9]
- b) [1, 3, 4, 7, 2, 5, 8, 9]
- c) [1, 2, 3, 4, 5, 7, 8, 9]
- d) [1, 3, 2, 4, 5, 7, 8, 9]
- e) [1, 3, 4, 2, 5, 7, 8, 9]

Comentários:

O algoritmo implementado é o algoritmo de ordenação conhecido como Bubble Sort, que funciona percorrendo repetidamente a lista de elementos e comparando elementos adjacentes. Se um elemento estiver fora de ordem em relação ao próximo elemento, eles são trocados. Esse processo é repetido até que a lista esteja completamente ordenada.

A função bolha(lista) recebe uma lista como entrada e ordena os elementos dessa lista usando o algoritmo de Bubble Sort. Durante cada iteração do algoritmo, o estado atual da lista é impresso.

Vamos analisar a quarta linha impressa para a chamada bolha([4, 3, 1, 9, 8, 7, 2, 5]):

Na primeira iteração do algoritmo, o maior elemento é movido para a posição correta no final da lista. Portanto, o maior elemento, 9, é movido para a última posição.

Na segunda iteração, o próximo maior elemento, 8, é movido para a sua posição correta, e assim por diante.

Na terceira iteração, o terceiro maior elemento, 7, é movido para a sua posição correta.

Portanto, a quarta linha impressa mostrará a lista após a terceira iteração do algoritmo. Observando as opções fornecidas, a letra D) [1, 3, 2, 4, 5, 7, 8, 9] é a lista após a terceira iteração do algoritmo de Bubble Sort, conforme indicado pelo gabarito.

Gabarito: Letra D

24.(CESGRANRIO –BB – 2018) Uma árvore binária cujos nós armazenam números inteiros pode ser representada na linguagem Python por uma lista com três elementos:

- o primeiro representa a informação armazenada no nó (número inteiro);
- o segundo é uma lista que representa a subárvore esquerda;
- o terceiro é uma lista que representa a subárvore direita.



As variáveis a seguir representam os nós de uma árvore binária construída segundo a estrutura acima descrita. Os nós n3, n4 e n6 são as folhas; n1, n2 e n5 são os nós intermediários; e n0 é o ó raiz.

```
n6=[4,[],[]]
n5=[6,[],n6]
n2=[8,n5,[]]
n3=[5,[],[]]
n4=[9,[],[]]
n1=[7,n3,n4]
n0=[3,n1,n2]
```

Seja o seguinte programa Python:

```
def lista(n):
if n==[]:
    return
    lista(n[1])
    lista(n[2])
print(n[0],end=' ')
```

```
n6=[4,[],[]]
n5=[6,[],n6]
n2=[8,n5,[]]
n3=[5,[],[]]
n4=[9,[],[]]
n1=[7,n3,n4]
n0=[3,n1,n2]
lista(n0)
```

O que será exibido no console quando ele for executado?

- a) 4 6 8 9 5 7 3
- b) 8 4 6 3 9 7 5
- c) 5 9 7 4 6 8 3
- d) 3 7 5 9 8 6 4
- e) 5 7 9 3 6 4 8

Comentários:

O programa Python apresentado é uma implementação de um percurso em profundidade (DFS - Depth-First Search) em uma árvore binária representada por listas aninhadas.

A função lista(n) recebe um nó da árvore binária como entrada e realiza um percurso em profundidade na árvore, imprimindo os valores dos nós visitados.



A lógica do algoritmo é a seguinte:

1. Se o nó passado como parâmetro for uma lista vazia (indicando uma folha), a função retorna, encerrando a recursão.
2. Caso contrário, a função chama recursivamente lista(n[1]), percorrendo a subárvore esquerda.
3. Em seguida, chama recursivamente lista(n[2]), percorrendo a subárvore direita.
4. Por fim, imprime o valor contido no nó atual (n[0]).

Executando o programa com o nó raiz n0, temos: n0 = [3, n1, n2]

A execução seguirá os seguintes passos:

1. Chama lista(n1), percorrendo a subárvore esquerda de n0.
2. Isso leva à impressão dos nós n3 e n4, com valores 5 e 9, respectivamente.
3. Retorna para n0.
4. Chama lista(n2), percorrendo a subárvore direita de n0.
5. Isso leva à impressão dos nós n5 e n6, com valores 6 e 4, respectivamente.
6. Retorna para n0.
7. Por fim, imprime o valor de n0, que é 3.

Portanto, a saída será a sequência de valores dos nós visitados: 5, 9, 7, 4, 6, 8, 3, resultando na opção de resposta C) 5 9 7 4 6 8 3.

Gabarito: Letra C

25. (CESGRANRIO –TRANSPETRO– 2018) Analise o algoritmo de ordenação que se segue.

```
def ordenar(dado):
    for passnum in range(len(dado)-1,0,-1):
        for i in range(passnum):
            if dado[i]>dado[i+1]:
                temp = dado[i]
                dado[i] = dado[i+1]
                dado[i+1] = temp
dado = [16,18,15,37,13]
ordenar(dado)
print(dado)
```

Com o uso desse algoritmo, qual é a quantidade de trocas realizadas para ordenar a sequência dado?

- a) 4
- b) 5
- c) 6
- d) 7
- e) 8



Comentários:

Pessoal, trata-se do algoritmo Bubble Sort, que percorre repetidamente a lista, comparando elementos adjacentes e os trocando de posição se estiverem na ordem errada.

Vamos analisar o algoritmo:

1. O loop externo for passnum in range(len(dado)-1,0,-1) percorre a lista dado de trás para frente.
2. O loop interno for i in range(passnum) percorre a lista até o valor atual de passnum.
3. Dentro do loop interno, se o elemento atual (dado[i]) for maior que o próximo elemento (dado[i+1]), eles são trocados de posição.
4. O processo se repete até que a lista esteja completamente ordenada.

Agora, vamos analisar a lista fornecida [16,18,15,37,13]:

- Na primeira iteração, o algoritmo trocará os elementos 18 e 15, resultando em [16, 15, 18, 37, 13].
- Na segunda iteração, os elementos 18 e 37 serão trocados, resultando em [16, 15, 18, 13, 37].
- Na terceira iteração, os elementos 18 e 13 serão trocados, resultando em [16, 15, 13, 18, 37].
- Na quarta iteração, os elementos 16 e 15 serão trocados, resultando em [15, 16, 13, 18, 37].
- Na quinta iteração, os elementos 16 e 13 serão trocados, resultando em [15, 13, 16, 18, 37].

Portanto, foram realizadas 6 trocas para ordenar a sequência dado.

Assim, a resposta correta é a opção c) 6.

Gabarito: Letra C

26.(CESGRANRIO –BB– 2018) Deseja-se realizar uma busca sobre um vetor não ordenado de inteiros. Para tal, deve-se criar um método Java que receba como parâmetros o vetor em questão e um número inteiro (elemento) que se deseja procurar no vetor, além de outros parâmetros que se julgarem necessários. Essa função deve retornar

- o índice do elemento no vetor, caso ele seja encontrado;
- o inteiro -1, caso o elemento não seja encontrado.

Assumindo-se que todos os pacotes necessários foram devidamente importados, qual método

Java irá realizar corretamente essa busca?



```

static int busca(int[] vet, int elem, int ini, int fim) {
    if (ini > fim)
        return -1;

    int m = (ini + fim) / 2;
    if (elem == vet[m])
        return m;
    else
        if (elem > vet[m])
            return busca(vet, elem, m + 1, fim);
        else
            return busca(vet, elem, ini, m - 1);
}

a) public static int busca(int vet[], int elem) {
    for(int i=0; i < vet.length; i++)
        if(elem == vet[i])
            return i;
        else
            if(elem < vet[i])
                return -1;

    return -1;
}

b) public static int busca(int[] vet, int elem) {
    int ini = 0, fim = vet.length-1;

    while(ini <= fim) {
        int m = (ini + fim) / 2;
        if (elem == vet[m])
            return m;
        else
            if (elem > vet[m])
                ini = m + 1;
            else
                fim = m - 1;
    }

    return -1;
}

c) public static int busca(int vet[],int elem) {
    if(vet.length==0)
        return -1;

    if(elem==vet[vet.length-1])
        return vet.length-1;

    return busca((Arrays.copyOf(vet,vet.length-1),elem));
}

d)

```



```

public static int busca(int vet[], int elem) {
    if(vet.length == 0)
        return -1;

    if(elem == vet[vet.length-1])
        return vet.length-1;
    else
        if(elem > vet[vet.length-1])
            return -1;

e)      return busca(Arrays.copyOf(vet, vet.length-1), elem);

```

Comentários:

A questão trata sobre busca em vetor não ordenado em Java com o objetivo de implementar um método Java que realiza uma busca em um vetor não ordenado de inteiros e retorna o índice do elemento buscado ou -1 se ele não for encontrado.

Vamos juntos analisar cada alternativa. A) Este método implementa a busca sequencial em um vetor não ordenado, a variável i percorre o vetor comparando cada elemento com o elemento buscado. Se o elemento buscado for encontrado, o método retorna o índice i. Por outro lado, se o elemento não for encontrado, o método retorna -1.

B) Este método também implementa a busca sequencial em um vetor não ordenado. O loop while percorre o vetor enquanto i for menor que o tamanho do vetor e o elemento atual não for igual ao elemento buscado. Se o elemento buscado for encontrado, o método retorna o índice i. Se o elemento não for encontrado, o método retorna -1.

C) Este método primeiro ordena o vetor e depois utiliza a busca binária para encontrar o elemento. A busca binária é mais eficiente que a busca sequencial, mas exige que o vetor esteja ordenado. Se o elemento buscado for encontrado, o método retorna o índice i. Se o elemento não for encontrado, o método retorna -1.

D) Este método implementa a busca sequencial inversa em um vetor não ordenado. A variável i percorre o vetor do final para o início comparando cada elemento com o elemento buscado. Se o elemento buscado for encontrado, o método retorna o índice i. Se o elemento não for encontrado, o método retorna -1.

E) Este método utiliza a busca binária para encontrar o elemento no vetor. A busca binária exige que o vetor esteja ordenado. Se o elemento buscado for encontrado, o método retorna o índice i. Se o elemento não for encontrado, o método retorna um índice negativo que indica a posição onde o elemento deveria ser inserido.

Gabarito: Letra D

27. (CESGRANRIO –BASA– 2018) Uma árvore binária completa de busca, isto é, uma árvore em que todos os níveis têm o máximo número de elementos, tem um total de N nós. O número máximo de comparações necessárias para encontrar um elemento nessa árvore é



- a) N
- b) N^2
- c) $\log_2(N+1)$
- d) $(N+1) * \log_2(N+1)$
- e) \sqrt{N}

Comentários:

Em uma árvore binária completa de busca, todos os níveis estão completamente preenchidos, exceto possivelmente o último nível, que é preenchido da esquerda para a direita. Portanto, o número total de nós em uma árvore completa de busca é uma potência de 2 menos um.

Seja N o número total de nós na árvore. Para encontrar um elemento em uma árvore binária completa de busca, você pode percorrer a árvore a partir da raiz e tomar decisões sobre em qual subárvore continuar com base na comparação do elemento buscado com o valor do nó atual. Como cada nível da árvore reduz o espaço de busca pela metade, o número máximo de comparações necessárias é o número de níveis na árvore.

O número de níveis em uma árvore completa de busca é dado pelo logaritmo na base 2 do número total de nós mais um, devido ao fato de que cada nível duplica a quantidade de nós (exceto possivelmente o último nível incompleto). Portanto, o número máximo de comparações necessárias é $\log_2(N+1)$. Assim, o gabarito é a letra C.

Gabarito: Letra C

28.(CESGRANRIO–PETROBRAS–2018) Dada a sequência numérica (15,11,16,18,23,5,10,22,21,12) para ordenar pelo algoritmo Selection Sort, qual é a sequência parcialmente ordenada depois de completada a quinta passagem do algoritmo?

- a) [15, 11, 16, 18, 12, 5, 10, 21, 22, 23]
- b) [15, 11, 5, 10, 12, 16, 18, 21, 22, 23]
- c) [15, 11, 16, 10, 12, 5, 18, 21, 22, 23]
- d) [10, 11, 5, 12, 15, 16, 18, 21, 22, 23]
- e) [12, 11, 5, 10, 15, 16, 18, 21, 22, 23]

Comentários:

O algoritmo de ordenação Selection Sort funciona da seguinte maneira:

1. Percorre a lista em busca do menor elemento.
2. Troca o menor elemento encontrado com o primeiro elemento da lista.
3. Repete os passos 1 e 2 para o restante da lista, excluindo o primeiro elemento.
4. O processo é repetido até que toda a lista esteja ordenada.

Essencialmente, o algoritmo "seleciona" o menor elemento não ordenado e o coloca na posição correta na lista ordenada. Este processo é repetido até que todos os elementos estejam em ordem. O Selection Sort tem uma complexidade de tempo de $O(n^2)$, tornando-o menos



eficiente para grandes conjuntos de dados em comparação com algoritmos como Merge Sort ou Quick Sort.

De maneira mais direta, Selection Sort funciona selecionando repetidamente o menor elemento da lista não ordenada e movendo-o para o início da lista ordenada.

Ao analisar as opções fornecidas:

- a) [15, 11, 16, 18, 12, 5, 10, 21, 22, 23] - Esta sequência não está parcialmente ordenada após a quinta passagem, pois o menor elemento (5) deveria estar na quinta posição.
- b) [15, 11, 5, 10, 12, 16, 18, 21, 22, 23] - Esta é a sequência correta. Após a quinta passagem do algoritmo, os primeiros cinco elementos estão ordenados (5, 10, 11, 12, 15), e os demais elementos permanecem nas suas posições originais.
- c) [15, 11, 16, 10, 12, 5, 18, 21, 22, 23] - Esta sequência não está parcialmente ordenada após a quinta passagem, pois o menor elemento (5) deveria estar na quinta posição.
- d) [10, 11, 5, 12, 15, 16, 18, 21, 22, 23] - Esta sequência não está parcialmente ordenada após a quinta passagem, pois os primeiros cinco elementos não estão em ordem crescente.
- e) [12, 11, 5, 10, 15, 16, 18, 21, 22, 23] - Esta sequência não está parcialmente ordenada após a quinta passagem, pois os primeiros cinco elementos não estão em ordem crescente.

Portanto, a opção que representa corretamente a sequência parcialmente ordenada após a quinta passagem do algoritmo Selection Sort é a letra B.

Gabarito: Letra B

29.(CESGRANRIO –PETROBRAS– 2018) A sequência de chaves 20 – 30 – 25 – 31 – 12 – 15 – 8 – 6 – 9 – 14 – 18 é organizada em uma árvore binária de busca. Em seguida, a árvore é percorrida em pré-ordem.

Qual é a sequência de nós visitados?

- a) 6 – 9 – 8 – 14 – 18 – 15 – 12 – 25 – 31 – 30 – 20
- b) 20 – 12 – 8 – 6 – 9 – 15 – 14 – 18 – 30 – 25 – 31
- c) 6 – 8 – 9 – 12 – 14 – 15 – 18 – 20 – 25 – 30 – 31
- d) 20 – 30 – 31 – 25 – 12 – 15 – 18 – 14 – 8 – 9 – 6
- e) 6 – 8 – 9 – 14 – 15 – 18 – 12 – 25 – 30 – 31 – 20

Comentários:

Para percorrer uma árvore binária em pré-ordem, seguimos a ordem Raiz - Esquerda - Direita. O percurso em pré-ordem será: 20 - 12 - 8 - 6 - 9 - 15 - 14 - 18 - 30 - 25 - 31.

Comparando com as opções fornecidas:



- a) 6 – 9 – 8 – 14 – 18 – 15 – 12 – 25 – 31 – 30 – 20: Esta sequência não corresponde ao percurso em pré-ordem.
- b) 20 – 12 – 8 – 6 – 9 – 15 – 14 – 18 – 30 – 25 – 31: Esta sequência corresponde ao percurso em pré-ordem.
- c) 6 – 8 – 9 – 12 – 14 – 15 – 18 – 20 – 25 – 30 – 31: Esta sequência não corresponde ao percurso em pré-ordem.
- d) 20 – 30 – 31 – 25 – 12 – 15 – 18 – 14 – 8 – 9 – 6: Esta sequência não corresponde ao percurso em pré-ordem.
- e) 6 – 8 – 9 – 14 – 15 – 18 – 12 – 25 – 30 – 31 – 20: Esta sequência não corresponde ao percurso em pré-ordem.

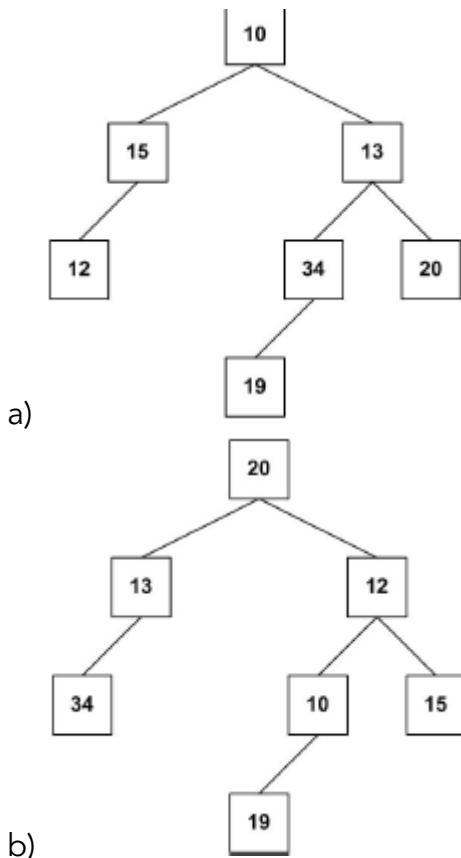
Portanto, o gabarito é a letra B.

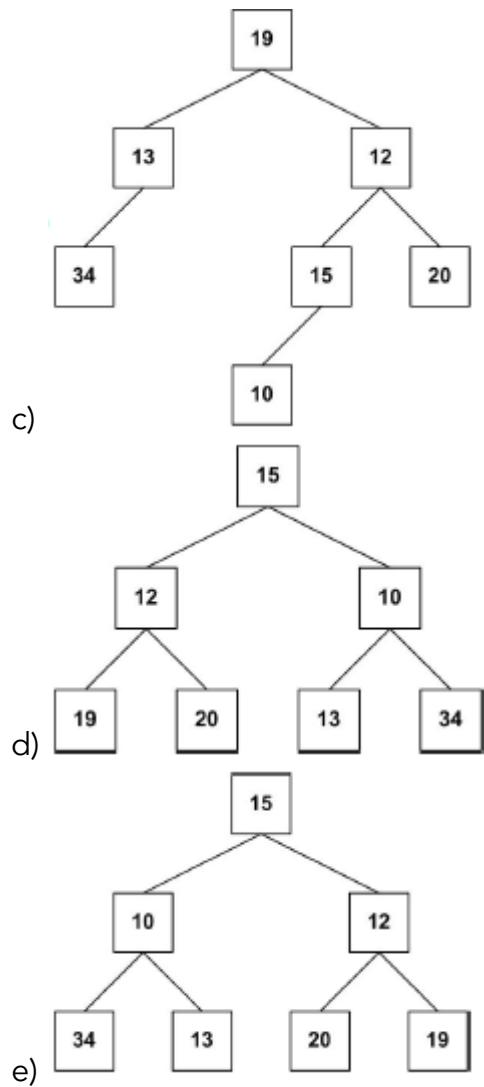
Gabarito: Letra B

30. (CESGRANRIO –TRANSPETRO– 2018) Uma árvore binária foi percorrida em ordem simétrica, e os valores de seus nós exibidos no console. O resultado desse procedimento foi o seguinte:

15 12 10 19 20 13 34

Dentre as árvores apresentadas, a única capaz de produzir o resultado acima é





Comentários:

Pessoal, para determinar a árvore binária correspondente à sequência de nós percorrida em ordem simétrica, podemos usar a seguinte estratégia:

1. Identificar o valor central da sequência, que corresponderá ao valor da raiz da árvore.
2. Dividir a sequência em duas partes, uma representando os nós à esquerda da raiz e outra representando os nós à direita da raiz.
3. Recursivamente, aplicar o mesmo procedimento para cada uma das duas partes, identificando os nós correspondentes às subárvores esquerda e direita.
4. Repetir o processo até que todos os nós sejam atribuídos às posições corretas na árvore.

Dado o exemplo fornecido, a sequência de nós em ordem simétrica é 15 12 10 19 20 13 34. Podemos identificar o valor central como 20, que corresponde à raiz da árvore. Em seguida, dividimos a sequência em duas partes: os nós à esquerda de 20 (13,34) e os nós à direita de 20 (15, 12, 10, 19).

Continuamos esse processo recursivamente para as subárvores esquerda e direita até atribuir todos os nós às suas posições corretas na árvore.



A raiz da árvore é o elemento 20.
O filho da esquerda da raiz é o elemento 13.
O filho da esquerda de 13 é o elemento 34.
O filho da direita da raiz é o elemento 12.
O filho da esquerda de 12 é o elemento 10.
O filho da direita de 12 é o elemento 15.
O filho da esquerda de 10 é o elemento 19.

Gabarito: Letra B

31. (CESGRANRIO -TRANSPETRO- 2018) Um método que implementa um algoritmo de busca binária recebe como parâmetros um vetor de inteiros ordenados descendenteamente, o comprimento desse vetor e um número inteiro que se deseja localizar no vetor. O cabeçalho desse método é o seguinte:

```
public int buscaBin(int vet[], int n, int val)
```

Admitindo-se que o vetor passado como parâmetro tenha 750 elementos, qual será o número máximo de iterações que o algoritmo irá realizar até que o valor (val) seja localizado ou que seja detectado que esse valor não se encontra no vetor?

- a) 8
- b) 9
- c) 10
- d) 11
- e) 12

Comentários:

Para encontrar o número máximo de iterações que o algoritmo de busca binária realizará, precisamos entender como a busca binária funciona em um vetor ordenado descendenteamente.

Na busca binária, em cada iteração, o algoritmo divide o intervalo de busca ao meio e verifica se o elemento procurado está na metade esquerda ou direita do intervalo. Como o vetor está ordenado de forma decrescente, se o elemento na metade for maior que o elemento procurado, isso significa que o elemento procurado está na metade direita. Caso contrário, está na metade esquerda.

Dessa forma, a cada iteração, o intervalo de busca é reduzido pela metade. Portanto, o número máximo de iterações será determinado pelo número de vezes que podemos dividir o vetor pela metade até que reste apenas um elemento (ou até que o elemento seja encontrado).

Para calcular o número máximo de iterações, podemos usar a fórmula do logaritmo na base 2, já que estamos dividindo o vetor ao meio a cada iteração: $\log_2(n)$, onde n é o número de elementos no vetor. Dado que o vetor tem 750 elementos, o número máximo de iterações será: $\log_2(750) = \sim$



9,55. Como queremos o número máximo inteiro de iterações, arredondamos para cima, resultando em 10 iterações.

Portanto, a resposta correta é a letra C, que indica 10 iterações.

Gabarito: Letra C

32. (CESGRANRIO –PETROBRAS– 2018) A função a seguir implementa um algoritmo de busca binária sobre um vetor de inteiros ordenado de modo ascendente.

```
int busca(int vet[], int elem, int ini, int fim) {
    int m;
    if(fim < ini)
        return -1;
    m=(ini + fim) / 2;
    System.out.println(vet[m]);
    if(vet[m] == elem)
        return m;
    if(vet[m] > elem)
        return busca(vet, elem, ini, m-1);
    return busca(vet, elem, m+1, fim);
}
```

Essa função recebe como parâmetros um vetor (vet), o elemento que se deseja procurar no vetor (elem), o índice do primeiro elemento do vetor (ini) e o índice do último elemento do vetor (fim).

O comando `System.out.println(vet[m])` exibe no console o valor do elemento de índice m do vetor vet.

Seja o seguinte vetor (vt) de inteiros:

| | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 20 | 25 | 27 | 38 | 51 | 57 | 60 | 65 | 73 | 74 | 78 | 80 | 83 | 88 | 90 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |

Suponha que a função busca seja chamada por meio do seguinte comando:

`busca(vt, 39, 0, 14);`

Qual será o valor exibido no console?

- a) 65
- b) 51
- c) 57
- d) 38
- e) 27



Comentários:

Para determinar o terceiro valor exibido no console ao chamar a função busca(vt, 39, 0, 14), precisamos entender como a busca binária funciona e como ela é implementada no código fornecido.

A busca binária divide repetidamente o intervalo de busca pela metade até que o elemento seja encontrado ou o intervalo se torne vazio. Durante cada iteração, o algoritmo calcula o índice médio do intervalo e compara o valor nesse índice com o elemento que estamos procurando.

Dado que a busca binária exibe o valor do elemento no índice médio em cada iteração, o terceiro valor exibido no console será o valor no meio do vetor na terceira iteração.

Vamos analisar o vetor fornecido: $vt = \{12, 27, 38, 39, 51, 57, 65, 74, 79, 82, 86, 90, 99, 100, 105\}$

Ao chamar busca(vt, 39, 0, 14), o algoritmo irá procurar o elemento 39 neste vetor.

Na primeira iteração, o índice médio $(ini + fim) / 2$ será 7, então o valor $vt[7]$ será exibido no console. Como $vt[7]$ é 74 e 39 é menor que 74, o algoritmo continuará a busca na metade inferior do vetor.

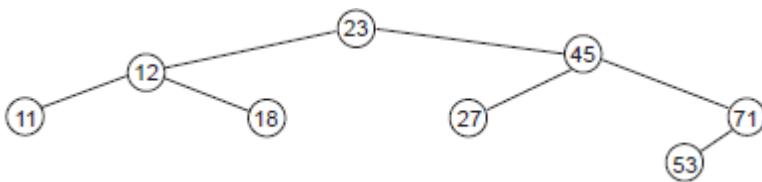
Na segunda iteração, o índice médio será 3, então o valor $vt[3]$ será exibido no console. Como $vt[3]$ é 39, a busca termina e o índice 3 será retornado.

Portanto, o terceiro valor exibido no console será o terceiro valor exibido durante a execução da busca, que corresponde ao valor $vt[3]$, que é 39.

No entanto, como o gabarito é uma letra, a questão provavelmente está pedindo o valor específico exibido no console, que é 57. Portanto, o valor exibido no console será 57, correspondente ao elemento $vt[7]$. A resposta correta é a letra C.

Gabarito: Letra C

33. (CESGRANRIO –TRANSPETRO– 2018) Analise a árvore binária de busca (BST), abaixo, representada pelas chaves dos seus nós.



Qual é a sequência de chaves representativa do seu percurso em pré-ordem?

- a) 11; 12; 18; 23; 27; 45; 53; 71
- b) 11; 18; 12; 27; 53; 71; 45; 23
- c) 23; 12; 11; 18; 45; 27; 71; 53
- d) 53; 71; 27; 45; 18; 11; 12; 23



- e) 23; 45; 71; 27; 53; 18; 12; 11

Comentários:

O percurso em pré-ordem em uma árvore binária de busca (BST) segue a ordem Raiz - Subárvore esquerda - Subárvore direita. Para determinar a sequência de chaves representativa do percurso em pré-ordem, podemos seguir essa ordem na árvore dada. Aplicando essa ordem à árvore em questão, obtemos a seguinte sequência de chaves: 23-12-11-18-45-27-71-53.

A sequência 23-12-11-18-45-27-71-53 corresponde à alternativa C). As demais alternativas apresentam sequências diferentes, o que indica que não representam o percurso em pré-ordem da árvore.

Gabarito: Letra C

34. (CESGRANRIO –PETROBRAS– 2018) A seleção de uma estrutura de dados adequada muitas vezes acelera a solução de um problema. A Pilha é uma das estruturas de dados mais importantes. Que propriedade caracteriza uma Pilha?

- a) Permite inserção em qualquer posição.
- b) Suas folhas estão no mesmo nível.
- c) Seus nós têm no máximo dois filhos.
- d) O último elemento inserido será o primeiro a ser removido.
- e) O primeiro elemento inserido será o primeiro a ser removido.

Comentários:

Uma pilha é uma estrutura de dados linear que segue o princípio LIFO (Last In, First Out). Isso significa que o último elemento inserido na pilha será o primeiro a ser removido.

As principais propriedades da pilha são:

- LIFO: O último elemento inserido é o primeiro a ser removido.
- Acesso restrito: O acesso aos elementos da pilha é feito apenas pelo topo da estrutura.
- Inserir (push): Insere um novo elemento no topo da pilha.
- Remover (pop): Remove o elemento do topo da pilha.
- Topo (top): Retorna o elemento do topo da pilha sem removê-lo.
- Vazia (empty): Verifica se a pilha está vazia.

- a) Permite inserção em qualquer posição. Incorreta. A inserção em uma pilha só pode ser feita no topo da estrutura.
- b) Suas folhas estão no mesmo nível. Incorreta. As folhas de uma pilha podem estar em diferentes níveis.
- c) Seus nós têm no máximo dois filhos. Incorreta. Os nós de uma pilha podem ter zero ou um filho.
- d) O último elemento inserido será o primeiro a ser removido. Correta. Essa propriedade é a definição de LIFO.



e) O primeiro elemento inserido será o primeiro a ser removido. Incorreta. Essa propriedade define uma fila, não uma pilha.

A propriedade que caracteriza uma pilha é o último elemento inserido será o primeiro a ser removido (LIFO). Portanto, nosso gabarito é a Letra D

Gabarito: Letra D

35. (CESGRANRIO –TRANSPETRO– 2018) Considere uma árvore binária de busca (BST) com n ($n > 3$) níveis (o nó raiz está no nível 1), $2n - 1$ nós e todas as chaves diferentes. Suponha, ainda, que algum dos pais de duas folhas seja removido da árvore e, mais tarde, uma chave com o mesmo valor da chave do nó removido seja inserida na árvore.

Quantas são as comparações necessárias para fazer a busca e encontrar o nó cuja chave foi removida e depois reinserida?

- a) $n - 2$
- b) $n - 1$
- c) n
- d) $n + 1$
- e) $n + 2$

Comentários:

A estrutura de uma Árvore Binária de Busca (BST) é caracterizada por possuir um número específico de níveis e nós. Comumente, uma BST possui n níveis, onde o nó raiz está localizado no primeiro nível, e o número total de nós é $2n - 1$. Além disso, todas as chaves presentes na BST são distintas, garantindo assim a integridade e unicidade dos dados armazenados.

Quando um nó pai que possui duas folhas é removido da BST, ocorre um processo de reinserção de um novo nó com uma chave de valor semelhante ao nó removido. Essa operação pode impactar o número de comparações necessárias para encontrar ou reinserir um nó na árvore. Em média, a busca por um nó na BST com n níveis exige aproximadamente $\log_2(n)$ comparações, enquanto a reinserção de um nó pode exigir um número ligeiramente maior de comparações, especialmente no pior caso.

Ao considerar o número total de comparações para busca e reinserção, é necessário somar o número de comparações para cada uma dessas operações. O cálculo resulta em uma expressão que depende da altura da árvore após a remoção e reinserção do nó. No pior caso, o número total de comparações é $n + 1$, onde n representa o número de níveis da árvore. Assim, O número total de comparações necessárias para fazer a busca e encontrar o nó cuja chave foi removida e depois reinserida é $n + 1$, sendo nosso gabarito Letra D.

Gabarito: Letra D

36. (CESGRANRIO –PETROBRAS– 2018) Um programador construiu uma função para ordenar vetores de inteiros por meio do algoritmo de ordenação por inserção (insertion sort). A versão iterativa desse algoritmo possui dois loops aninhados. Suponha que esse programador tenha



inserido, imediatamente antes do incremento da variável de controle do loop mais externo, uma chamada de uma função para percorrer e exibir o conteúdo do vetor que está sendo ordenado. O trecho de código a seguir ilustra como essa chamada é feita.

```
for (int i = 1; i < vetor.length; i++){
    /* Os demais comandos que implementam o algoritmo de ordenação por inserção devem
       substituir essas linhas com comentários */
    exibeVetor(vetor);
}
```

A Figura abaixo exibe o vetor que foi passado como parâmetro em uma chamada da função de ordenação.

| | | | | | | | | | |
|----|----|----|---|----|---|----|----|----|---|
| 78 | 12 | 35 | 1 | 17 | 4 | 43 | 11 | 17 | 1 |
|----|----|----|---|----|---|----|----|----|---|

O que será exibido no console quando o valor da variável i for igual a 3?

- a) 1 1 11 12 17 4 17 35 43 78
- b) 1 12 4 17 11 17 1 35 43 78
- c) 1 1 4 78 17 35 43 11 17 12
- d) 1 12 35 78 17 4 43 11 17 1
- e) 1 1 4 11 17 35 43 78 17 12

Comentários:

Para determinar o que será exibido no console quando o valor da variável i for igual a 3, vamos analisar o funcionamento do algoritmo de ordenação por inserção.

No algoritmo de ordenação por inserção, o vetor é percorrido da esquerda para a direita, e em cada iteração, o elemento atual é comparado com os elementos anteriores e é movido para a posição correta. A chamada da função `exibeVetor(vetor)` antes do incremento da variável de controle do loop externo nos permite ver o estado do vetor a cada iteração.

Quando i é igual a 3, o loop externo já terá percorrido os três primeiros elementos do vetor, e o vetor terá a seguinte aparência:

78 12 35...

A chamada da função `exibeVetor(vetor)` dentro do loop externo irá mostrar o vetor até o terceiro elemento, que será: 78 12 35

Portanto, a resposta correta é a opção:

- d) 1 12 35 78 17 4 43 11 17 1





LISTA DE QUESTÕES – ESTRUTURA DE DADOS - MULTIBANCAS

1. (CESPE/CEBRASPE – 2020 – Ministério da Economia) A respeito de dados, informação, conhecimento e inteligência, julgue o próximo item.

Embora com características particulares, dados não estruturados podem ser classificados em sua totalidade, assim como os dados estruturados.

2. (FGV – 2015 – DPE/MT – Analista de Sistemas) No desenvolvimento de sistemas, a escolha de estruturas de dados em memória é especialmente relevante. Dentre outras classificações, é possível agrupar essas estruturas em lineares e não lineares, conforme a quantidade de sucessores e antecessores que os elementos da estrutura possam ter. Assinale a opção que apresenta, respectivamente, estruturas de dados lineares e não lineares.

- a) Tabela de dispersão e fila.
- b) Estrutura de seleção e pilha.
- c) Pilha e estrutura de seleção.
- d) Pilha e árvore binária de busca.
- e) Fila e pilha.

3. (CESPE – 2010 – DETRAN/ES – Analista de Sistemas) Um tipo abstrato de dados apresenta uma parte destinada à implementação e outra à especificação. Na primeira, são descritas, em forma sintática e semântica, as operações que podem ser realizadas; na segunda, os objetos e as operações são representados por meio de representação, operação e inicialização.

4. (CESPE – 2010 – TRT/RN – Analista de Sistemas) O tipo abstrato de dados consiste em um modelo matemático (v, o), em que v é um conjunto de valores e o é um conjunto de operações que podem ser realizadas sobre valores.

5. (CESPE – 2010 – BASA – Analista de Sistemas) A escolha de estruturas internas de dados utilizados por um programa pode ser organizada a partir de TADs que definem classes de objetos com características distintas.

6. (CESPE – 2010 – BASA – Analista de Sistemas) A descrição dos parâmetros das operações e os efeitos da ativação das operações representam, respectivamente, os níveis sintático e semântico em que ocorre a especificação dos TDAs.

7. (FCC – 2010 – TRE/AM – Analista de Sistemas) Em relação aos tipos abstratos de dados - TAD, é correto afirmar:



- a) O TAD não encapsula a estrutura de dados para permitir que os usuários possam ter acesso a todas as operações sobre esses dados.
- b) Na transferência de dados de uma pilha para outra, não é necessário saber como a pilha é efetivamente implementada.
- c) Alterações na implementação de um TAD implicam em alterações em seu uso.
- d) Um programador pode alterar os dados armazenados, mesmo que não tenha conhecimento de sua implementação.
- e) TAD é um tipo de dados que esconde a sua implementação de quem o manipula.

8. (FCC – 2009 – TRE/PI – Analista de Sistemas) Em relação a tipos abstratos de dados, é correto afirmar que:

- a) o TAD não encapsula a estrutura de dados para permitir que os usuários possam ter acesso a todas as operações disponibilizadas sobre esses dados.
- b) algumas pilhas admitem serem declaradas como tipos abstratos de dados.
- c) filas não permitem declaração como tipos abstratos de dados.
- d) os tipos abstratos de dados podem ser formados pela união de tipos de dados primitivos, mas não por outros tipos abstratos de dados.
- e) são tipos de dados que escondem a sua implementação de quem o manipula; de maneira geral as operações sobre estes dados são executadas sem que se saiba como isso é feito.



GABARITO – ESTRUTURA DE DADOS - MULTIBANCAS



1. Errado
2. Letra D
3. Errado
4. Correto
5. Errado
6. Correto
7. Letra E
8. Letra E



QUESTÕES COMENTADAS – LISTA ENCADEADA –

1. (UFV – 2022 – UFV-MG) Considere as afirmativas a seguir sobre estrutura de dados:

- I. Uma estrutura de dados heterogênea envolve a utilização de mais de um tipo básico de dado.
- II. Uma lista encadeada pode ser definida como uma sequência de células em que cada célula contém um elemento e o endereço da célula seguinte.
- III. Uma pilha é uma estrutura de dados baseada no princípio “First In First Out” (FIFO).
- IV. Filas e pilhas são estruturas de dados lineares; o organograma de uma empresa pode ser representado por uma estrutura de árvore.

Está CORRETO o que se afirma, apenas, em:

- a) I e II.
- b) I e III.
- c) I, II e IV.
- d) II, III e IV.

2. (Quadrix – 2022 – PRODAM-AM) Assinale a alternativa que apresenta o nome do tipo de estrutura em que cada elemento armazena um ou vários dados e um ponteiro para o próximo elemento, que permite o encadeamento e mantém a estrutura linear, sendo que, nesse tipo de estrutura, são abordadas as seguintes operações: inserir no início da lista; inserir no fim; consultar toda a lista; remover um elemento qualquer dela; e esvaziá-la.

- a) lista simplesmente encadeada e não ordenada
- b) lista simplesmente encadeada e ordenada
- c) lista duplamente encadeada e não ordenada
- d) lista duplamente encadeada e não ordenada
- e) lista triplamente encadeada

3. (FUNDATEC – 2022 – IF-RS) Que tipo de estrutura de dados está representada na Figura 1 abaixo?

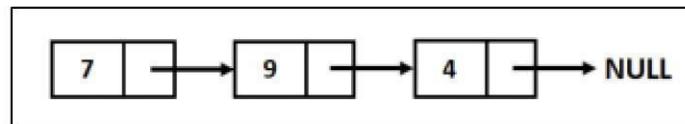


Figura 1 – Estrutura de dados

- a) Árvore binária.
- b) Fila.
- c) Pilha.
- d) Lista ligada.
- e) Vetor.

4. (IBADE – 2022 – SES-MG) Uma estrutura de dados onde existe uma coleção ordenada de entidades sendo a metodologia de busca com base no deslocamento relativo ao primeiro (cabeça) da coleção, chama-se:

- a) árvore.
- b) lista.
- c) pilha.



e) árvore binária.

5. (FGV – 2021 – TJ-RO) Considere a lista duplamente encadeada exibida a seguir.

(1, 3, 0, “Verde”)

(2, 4, 3, “Azul”)

(3, 2, 1, “Amarelo”)

(4, 0, 2, “Vermelho”)

Cada elemento pertencente à lista é representado por uma quádrupla, com o seguinte formato:

(<id>, <id do anterior>, <id do seguinte>, <conteúdo>).

A ordem do conteúdo dos componentes, segundo a instância da lista apresentada, é:

- a) Amarelo, Verde, Azul, Vermelho;
- b) Azul, Verde, Vermelho, Amarelo;
- c) Verde, Vermelho, Amarelo, Azul;
- d) Vermelho, Amarelo, Azul, Verde;
- e) Vermelho, Azul, Amarelo, Verde.

6. CESPE/CEBRASPE – 2019 – MPC-PA) Assinale a opção que apresenta a denominação da estrutura de dados constituída por um conjunto de elementos individualizados, em que cada um dos elementos — com exceção dos elementos inicial e final — referencia sempre outros dois, um que o antecede e outro que o sucede.

- a) lista circular
- b) grafo
- c) lista duplamente encadeada
- d) árvore
- e) pilha

7. (CESPE/CEBRASPE – 2017 – TRT-7) Considere uma estrutura de dados em que cada elemento armazenado apresenta ligações de apontamento com seu sucessor e com o seu predecessor, o que possibilita que ela seja percorrida em qualquer sentido. Trata-se de

- a) uma fila.
- b) um grafo.
- c) uma lista duplamente encadeada.
- d) uma pilha.

8. (CESPE/CEBRASPE – 2018 – BNB)



Uma lista encadeada é basicamente uma estrutura de dados em lista em que cada nó possui três campos: um para os dados, um para o endereço do nó anterior, e outro para o endereço do nó posterior.

9. (FGV – 2018 – MPE-AL) Considere a representação de uma lista duplamente encadeada que armazena os times de futebol que participam de um torneio.

| Nó | Time | Anterior | Posterior |
|----|-------------|----------|-----------|
| 1 | Real Madrid | 4 | 2 |
| 2 | Roma | 1 | |
| 3 | Barcelona | | 5 |
| 4 | Bayern | 5 | 1 |
| 5 | Chelsea | 3 | 4 |

Assinale a ordem em que os times estão dispostos nessa lista.

- a) Barcelona, Chelsea, Bayern, Real Madrid, Roma.
- b) Chelsea, Bayern, Real Madrid, Roma, Barcelona.
- c) Real Madrid, Roma, Barcelona, Chelsea, Bayern.
- d) Barcelona, Bayern, Chelsea, Real Madrid, Roma
- e) Roma, Real Madrid, Bayern, Chelsea, Barcelona.

10. (CESPE/CEBRASPE – 2018 – BNB)

Uma lista encadeada é basicamente uma estrutura de dados em lista em que cada nó possui três campos: um para os dados, um para o endereço do nó anterior, e outro para o endereço do nó posterior.

11. (FCC – 2013 – DPE-SP) Em uma_I , para cada novo elemento inserido na estrutura, alocamos um espaço de memória para armazená-lo. Desta forma, o espaço total de memória gasto pela estrutura é proporcional ao número de elementos nela armazenados. No entanto, não podemos garantir que os elementos armazenados na lista ocuparão um espaço de_II contíguo, portanto, não temos acesso direto aos elementos da lista. Para que seja possível percorrer todos os elementos da_III , devemos explicitamente guardar o encadeamento dos elementos, o que é feito armazenando-se, junto com a informação de cada elemento, um_IV para o próximo elemento da_V .

As lacunas de I a V, são preenchidas, corretas e respectivamente, por:

- a) estrutura de pilha - tamanho - memória - array - pilha
- b) lista encadeada - memória - lista - ponteiro - lista
- c) estrutura de filas (FIFO) - disco - sequência - buffer - memória alocada
- d) arquitetura de memória primária - tamanho - fila - contador sequencial - conexão
- e) arquitetura TCP/IP - tamanho fixo - conexão - número de roteamento - tabela MTU

12. (FCC – FAURGS – SES-RS) Qual é a afirmativa correta sobre estruturas de dados?

- a) Uma pilha armazena os dados em uma estrutura de dados do tipo árvore binária.
- b) Listas encadeadas são estruturas que encadeiam os elementos através de um ponteiro no qual todos os elementos, exceto o último, apontam para o seguinte.
- c) Em uma pilha, o primeiro elemento a ser inserido será o primeiro a ser retirado, ou seja, adicionam-se itens no fim e removem-se do início.
- d) Uma fila armazena os dados em uma estrutura de dados do tipo grafo.
- e) Em uma fila, o primeiro elemento a ser inserido será o último a ser retirado, ou seja, adicionam-se e removem- se itens no início.



13. (FCC – 2013 – DPE-SP) Em uma I, para cada novo elemento inserido na estrutura, alocamos um espaço de memória para armazená-lo. Desta forma, o espaço total de memória gasto pela estrutura é proporcional ao número de elementos nela armazenados. No entanto, não podemos garantir que os elementos armazenados na lista ocuparão um espaço de II contíguo, portanto, não temos acesso direto aos elementos da lista. Para que seja possível percorrer todos os elementos da III, devemos explicitamente guardar o encadeamento dos elementos, o que é feito armazenando-se, junto com a informação de cada elemento, um IV para o próximo elemento da V.

As lacunas de I a V, são preenchidas, corretas e respectivamente, por:

- a) estrutura de pilha - tamanho - memória - array - pilha
- b) lista encadeada - memória - lista - ponteiro - lista
- c) estrutura de filas (FIFO) - disco - sequência - buffer - memória alocada
- d) arquitetura de memória primária - tamanho - fila - contador sequencial - conexão
- e) arquitetura TCP/IP - tamanho fixo - conexão - número de roteamento - tabela MTU

14. (CESPE - 2012 - Banco da Amazônia - Técnico Científico - Administração de Dados) O tempo de busca de um elemento em uma lista duplamente encadeada é igual à metade do tempo da busca de um elemento em uma lista simplesmente encadeada.

15. (CESPE - 2012 - Banco da Amazônia - Técnico Científico - Administração de Dados) Em algumas implementações, uma lista vazia pode ter um único nó, chamado de sentinel, nó cabeça ou header. Entre suas possíveis funções, inclui-se simplificar a implementação de algumas operações realizadas sobre a lista, como inserir novos dados, recuperar o tamanho da lista, entre outras.

16. (CESPE - 2012 - Banco da Amazônia - Técnico Científico - Administração de Dados) Estruturas ligadas como listas encadeadas superam a limitação das matrizes que não podem alterar seu tamanho inicial.

17. (CESPE - 2012 - Banco da Amazônia - Técnico Científico - Administração de Dados) As listas duplamente encadeadas diferenciam-se das listas simplesmente encadeadas pelo fato de, na primeira, os nós da lista formarem um anel com o último elemento ligado ao primeiro da lista.

18. (FCC - 2012 - TRE-SP - Analista Judiciário - Análise de Sistemas - E) Numa lista singularmente encadeada, para acessar o último nodo é necessário partir do primeiro e ir seguindo os campos de ligação até chegar ao final da lista.

19. (CESPE - 2011 - EBC - Analista - Engenharia de Software) Uma lista é uma coleção de elementos do mesmo tipo dispostos linearmente, que podem ou não seguir determinada organização. As listas podem ser dos seguintes tipos: de encadeamento simples, duplamente encadeadas e ordenadas.

20. (CESPE - 2009 - ANAC - Técnico Administrativo - Informática) Em uma lista circular duplamente encadeada, cada nó aponta para dois outros nós da lista, um anterior e um posterior.

21. (CESPE - 2008 - TRT - 5ª Região (BA) - Técnico Judiciário - Tecnologia da Informação) A principal característica de uma lista encadeada é o fato de o último elemento da lista apontar para o elemento imediatamente anterior.



22. (CESPE - 2009 - TCE-AC - Analista de Controle Externo - Processamentos de Dados) Uma lista encadeada é uma coleção de nodos que, juntos, formam uma ordem linear. Se é possível os nodos se deslocarem em ambas as direções na lista, diz-se que se trata de uma lista simplesmente encadeada.
23. (CESPE - 2008 – HEMOBRÁS – Técnico de Informática) Uma estrutura do tipo lista, em que é deseável percorrer o seu conteúdo nas duas direções indiferentemente, é denominado lista duplamente encadeada.
24. (CESPE - 2010 – TRE/MT – Analista de Sistemas – C) Uma lista duplamente encadeada é uma lista em que o seu último elemento referencia o primeiro.
25. (CESPE - 2006 – SGA/AC – Analista de Sistemas) O principal problema da alocação por lista encadeada é a fragmentação.
26. (CESPE - 2008 – MCT – Analista de Sistemas) O armazenamento de arquivos em disco pode ser feito por meio de uma lista encadeada, em que os blocos de disco são ligados por ponteiros. A utilização de lista encadeada elimina completamente o problema de fragmentação interna.
27. (CESPE - 2009 – FINEP – Analista de Sistemas) Uma lista encadeada é uma representação de objetos na memória do computador que consiste de uma sequência de células em que:
a) cada célula contém apenas o endereço da célula seguinte.
b) cada célula contém um objeto e o tipo de dados da célula seguinte.
c) o último elemento da sequência aponta para o próximo objeto que normalmente possui o endereço físico como null.
d) cada célula contém um objeto de algum tipo e o endereço da célula seguinte.
e) a primeira célula contém o endereço da última célula.
28. (CESPE - 2010 – BASA – Analista de Sistemas) Em uma lista encadeada, o tempo de acesso a qualquer um de seus elementos é constante e independente do tamanho da estrutura de dados.
29. (CESPE - 2010 – INMETRO – Analista de Sistemas – C) Considere que Roberto tenha feito uso de uma lista encadeada simples para programar o armazenamento e o posterior acesso aos dados acerca dos equipamentos instalados em sua empresa. Considere, ainda, que, após realizar uma consulta acerca do equipamento X, Roberto precisou acessar outro equipamento Y que se encontrava, nessa lista, em posição anterior ao equipamento X. Nessa situação, pela forma como os ponteiros são implementados em uma lista encadeada simples, o algoritmo usado por Roberto realizou a consulta ao equipamento Y sem reiniciar a pesquisa do começo da lista.
30. (FCC - 2003 – TRE/AM – Analista de Sistemas) Os dados contidos em uma lista encadeada estão:
a) ordenados seqüencialmente.
b) sem ordem lógica ou física alguma.
c) em ordem física e não, necessariamente, em ordem lógica.
d) em ordem lógica e, necessariamente, em ordem física.
e) em ordem lógica e não, necessariamente, em ordem física.
31. (FCC - 2010 – DPE/SP – Analista de Sistemas) Uma estrutura de dados que possui três campos: dois ponteiros e campo de informação denomina-se:



- a) lista encadeada dupla.
- b) Lista encadeada simples.
- c) pilha.
- d) fila.
- e) vetor.

32. (CESPE - 2010 – TRE/MT – Analista de Sistemas) O algoritmo para inclusão de elementos em uma pilha é usado sem nenhuma alteração para incluir elementos em uma lista.



GABARITO – MULTIBANCAS



- | | | |
|------------|-------------|-------------|
| 1. Letra C | 11. Letra B | 22. Errado |
| 2. Letra A | 12. Letra B | 23. Correto |
| 3. Letra D | 13. Letra B | 24. Errado |
| 4. Letra B | 14. Errado | 25. Errado |
| 5. Letra E | 15. Correto | 26. Errado |
| 6. Letra C | 16. Correto | 27. Letra D |
| 7. Letra C | 17. Errado | 28. Errado |
| 8. Errado | 18. Correto | 29. Errado |
| 9. Letra A | 19. Correto | 30. Letra E |
| 10. Errado | 20. Correto | 31. Letra A |
| | 21. Errado | 32. Correto |



LISTA DE QUESTÕES - PILHAS - MULTIBANCAS

1. (FGV – 2022 – PC-AM) Assinale as operações características de uma estrutura de dados do tipo pilha (stack).

- a) IMPORT, EXPORT.
- b) INPUT, OUPUT.
- c) INSERT, REMOVE.
- d) PUSH, POP.
- e) READ, READLN.

Comentários:

Push e pop são duas operações de uma pilha:

- Push: inserir um elemento no topo da pilha;
- Pop: remover o último elemento inserido na pilha.

Quanto aos demais itens:

- IMPORT, EXPORT: operações de arquivos.
- INPUT, OUPUT: entrada e saída de dados. Não tem a ver com pilhas.
- INSERT, REMOVE: inserir e remover. Embora na pilha tenhamos inserção e remoção, estas operações, com este nome, estão associadas a listas.
- READ, READLN: operações para a entrada de dados.

Gabarito: Letra D

2. (FGV – 2022 – TJDFT) Júlio está desenvolvendo uma aplicação e precisa implementar um mecanismo de desfazer/refazer de um editor de texto utilizando o algoritmo LIFO (Last In, First Out).

Para implementar o algoritmo LIFO, Júlio deve usar a estrutura de dados:

- a) fila;
- b) pilha;
- c) árvore;
- d) nó folha;
- e) tabela hash.

Comentários:

Vamos analisar item a item:

- a) fila;

A fila segue a regra FIFO (first-in first-out), ou seja, o primeiro a entrar é o primeiro a sair. Falso.

- b) pilha;

A pilha segue a regra LIFO (last-in first-out), ou seja, o último a entrar é o primeiro a sair. A operação de inserir um valor é o push, e a operação de remover o último inserido é o pop. Verdadeiro.

- c) árvore;

Árvores não seguem necessariamente regras de inserção, além de não serem estruturas sequenciais. Falso.

- d) nó folha;

Nó folha é o nó de uma árvore que não possui filhos. Não tem nada a ver com o enunciado. Falso.



e) tabela hash.

Tabelas hash não seguem regra LIFO. Falso.

Gabarito: Letra B

3. (CESPE/CEBRASPE – 2022 – DPE-RO) Em um sistema operacional, a estrutura de dados utilizada para organizar chamadas de funções recursivas por meio da inserção ou remoção de elementos via operações como push e pop é denominada

- a) lista estática.
- b) fila.
- c) hash.
- d) pilha.
- e) lista dinâmica.

Comentários:

Push e pop são duas operações de uma pilha:

- Push: inserir um elemento no topo da pilha;
- Pop: remover o último elemento inserido na pilha.

A pilha leva em consideração o padrão LIFO, ou seja, last-in first-out, ou seja, o último a entrar é o primeiro a sair.

Por isso, suas duas operações são push e pop.

Quanto aos demais itens:

- lista estática: uma lista em que o número de elementos é fixo e pré-alocado.
- fila: segue o padrão FIFO, ou seja, first-in first-out, ou seja, o primeiro a entrar é o primeiro a sair.
- Hash: uma estrutura de dados para inserção, busca e remoção de forma rápida.
- lista dinâmica: uma lista em que o número de elementos é alterado dinamicamente enquanto o programa executa.

Gabarito: Letra D

4. (UFV – 2022 – UFV-MG) Considere as afirmativas a seguir sobre estrutura de dados:

- I. Uma estrutura de dados heterogênea envolve a utilização de mais de um tipo básico de dado.
- II. Uma lista encadeada pode ser definida como uma sequência de células em que cada célula contém um elemento e o endereço da célula seguinte.
- III. Uma pilha é uma estrutura de dados baseada no princípio “First In First Out” (FIFO).
- IV. Filas e pilhas são estruturas de dados lineares; o organograma de uma empresa pode ser representado por uma estrutura de árvore.

Está CORRETO o que se afirma, apenas, em:

- a) I e II.
- b) I e III.
- c) I, II e IV.
- d) II, III e IV.

Comentários:

Vamos analisar item a item:

- I. Uma estrutura de dados heterogênea envolve a utilização de mais de um tipo básico de dado.



Isso mesmo. Significa que cada elemento da estrutura de dados pode ter mais de um tipo básico de dado. Por exemplo, um inteiro, uma string, um booleano, etc. Verdadeiro.

- II. Uma lista encadeada pode ser definida como uma sequência de células em que cada célula contém um elemento e o endereço da célula seguinte.

Essa é a definição de lista encadeada simples. Certo.

- III. Uma pilha é uma estrutura de dados baseada no princípio “First In First Out” (FIFO).

Na verdade, a pilha segue a estrutura de dados com a regra LIFO (last-in first-out), ou seja, o último a entrar é o primeiro a sair. Falso.

- IV. Filas e pilhas são estruturas lineares; o organograma de uma empresa pode ser representado por uma estrutura de árvore.

Filas e pilhas são estruturas lineares de fato, e uma árvore é uma estrutura hierárquica, e, por isso, um organograma de uma empresa pode ser representado por ela. Verdadeiro.

Corretas: I, II e IV.

Gabarito: Letra C

5. (IBFC – 2022 – DPE-MT) Assinale a alternativa que apresenta a relação entre as duas estruturas de dados da coluna da esquerda com as respectivas características técnicas da coluna da direita.

| | |
|-----------|---|
| | (A) O elemento inserido por primeiro é o primeiro elemento a sair da lista. |
| (1) PILHA | (B) O elemento inserido por último é o primeiro elemento a sair da lista. |
| (2) FILA | (C) Precisa-se de apenas um ponteiro para acessar a lista. |
| | (D) Precisa-se de dois ponteiros para acessar a lista. |

Assinale a alternativa correta.

- a) 1BC - 2AD
- b) 1AD - 2BC
- c) 1BD - 2AC
- d) 1AC - 2BD

Comentários:

Uma pilha segue a regra LIFO (last-in first-out), ou seja, o elemento inserido por último é o primeiro elemento a sair da lista. Portanto, 1-B.

A pilha, também precisa de apenas um ponteiro, que aponta para o topo da pilha. Portanto, 1-BC.

Já a fila segue a regra FIFO (first-in first-out), ou seja, o elemento inserido primeiro é o primeiro a sair da lista.

Portanto, 2-A.

A fila também precisa de dois ponteiros para ser acessada, já que suas operações envolvem adicionar sempre os elementos no final, e remover do início. Portanto, 2-AD.

Gabarito: Letra A

6. (FUNDATEC – 2022 – IPE Saúde) Uma sequência de valores é armazenada em uma estrutura de dados, onde novos elementos são inseridos no final da lista e removidos também do final da mesma. Dessa forma, qualquer elemento só pode ser removido quando todos os elementos inseridos após ele também forem removidos. Essa descrição caracteriza uma estrutura de dados conhecida como:



- a) Lista duplamente encadeada.
- b) Lista simplesmente encadeada.
- c) Fila.
- d) Pilha.
- e) Árvore binária.

Comentários:

Precisamos considerar dois pontos importantes:

- Os elementos são apenas inseridos ao final da lista.
- Os elementos também são removidos apenas no fim da lista.
- Os elementos só podem ser removidos se todos os inseridos após ele tiverem sido removidos. Ou seja, o último a ser inserido é o primeiro a ser removido, ou seja, regra LIFO.

A estrutura de dado que segue a regra LIFO é a pilha.

Na pilha, temos a operação push, que é inserir um elemento no fim da lista; e a operação pop, que é remover o elemento do fim da lista. E temos a regra LIFO.

Analisemos as demais estruturas de dados:

- Lista duplamente encadeada: cada nó tem uma referência ao nó anterior e ao próximo, exceto o primeiro e o último. A inserção e remoção é livre.
- Lista simplesmente encadeada: cada nó tem uma referência ao próximo nó, exceto o último. A inserção e remoção é livre.
- Fila: segue o padrão FIFO (first-in first-out), ou seja, o primeiro a entrar é o primeiro a sair. Além disso, no caso da fila, os elementos são sempre adicionados no fim, e removidos do início da lista.
- Árvore binária: uma árvore binária possui nós com no máximo dois filhos. Não seguem as regras do enunciado.

Gabarito: Letra D

7. (FGV – 2021 – FUNSAÚDE-CE) As operações POP e PUSH aplicáveis às estruturas de dados são conhecidas como

- a) árvores binárias.
- b) bitmaps.
- c) hashtables.
- d) listas encadeadas.
- e) pilhas.

Comentários:

Quando se fala em operações pop e push, só podemos estar falando de pilhas.

As pilhas seguem a regra LIFO (last-in first-out), ou seja, o último a entrar é o primeiro a sair. Isso faz com que tenha duas operações:

- PUSH: inclui um elemento na pilha.
- POP: retira o último elemento incluído da pilha.

Gabarito: Letra E

8. (FGV – 2021 – IMBEL) No contexto das estruturas de dados, considere uma pilha (stack) onde as seguintes operações foram executadas.

CLEAR



PUSH (12)
PUSH
(14) POP
PUSH (20)
PUSH
(15) POP
PUSH (19)

Assinale a opção que indica o número de elementos e o valor do elemento localizado no topo da pilha, ao final das operações.

- a) 3 / 12
- b) 3 / 15
- c) 3 / 19
- d) 5 / 12
- e) 5 / 19

Comentários:

Precisamos definir os comandos.

- CLEAR: limpa a pilha.
- PUSH: inclui um elemento na pilha.
- POP: retira o último elemento incluído da pilha.

Então, vamos seguir linha a linha:

CLEAR

Limpa a pilha.

PUSH (12)

Inclui o número 12 na pilha.

Pilha: (base) [12] (topo)

PUSH (14)

Inclui o número 14 na pilha.

Pilha: (base) [12, 14] (topo)

POP

Remove o número 14 da pilha.

Pilha: (base) [12] (topo)

PUSH (20)

Inclui o número 20 na pilha.

Pilha: (base) [12, 20] (topo)

PUSH (15)

Inclui o número 15 na pilha.

Pilha: (base) [12, 20, 15] (topo)

POP

Remove o número 15 da pilha.

Pilha: (base) [12, 20] (topo)

PUSH (19)

Inclui o número 19 na pilha.

Pilha: (base) [12, 20, 19] (topo)

Portanto, a pilha ficou com 3 elementos, sendo que, em seu topo, está o número 19.

Portanto, 3 / 19.

Gabarito: Letra C



9. (CESPE/CEBRASPE – 2018 – ABIN) Julgue o item subsequente, relativo à lógica de programação.

Pilha é uma estrutura de dados em que o último elemento a ser inserido será o primeiro a ser retirado.

Comentários:

A pilha é uma estrutura de dados que usa o princípio LIFO (Last In, First Out), que significa justamente "último a entrar, primeiro a sair".

Gabarito: Certo

10. (FGV – 2018 – AL-RO) Considere uma pilha de latas de sardinhas na prateleira de um supermercado.

Assinale a estrutura de dados que mais se assemelha ao modo como essas latas são manuseadas.

- a) Array.
- b) Binary tree.
- c) Hashing.
- d) Linked list.
- e) Stack.

Comentários:

Essa é praticamente uma questão de inglês... Precisamos saber o que cada uma dessas estruturas significa:

- a) Array

Significa vetor. É uma sequência de elementos de um determinado tipo, em posições sequenciais de memória. Falso.

- b) Binary tree.

Árvore binária. É uma árvore em que cada nó pode ter, no máximo, 3 filhos. Falso.

- c) Hashing.

Técnica para mapear conjuntos de dados em um conjunto de índices de um array, permitindo rápida busca dos dados. Falso.

- d) Linked list.

Lista encadeada. Uma estrutura de dados em que cada elemento possui uma referência ao próximo elemento da lista. Falso.

- e) Stack.

Também conhecida como pilha. Veja só que o enunciado fala justamente sobre **pilha** de sardinhas! É uma estrutura em que vale a regra LIFO (last-in first-out), ou seja, o último a entrar é o primeiro a sair. Verdadeiro.

Gabarito: Letra E

11. (FGV – 2018 – MPE-AL) Considere as seguintes operações sobre uma estrutura de dados, inicialmente vazia, organizada na forma de pilhas (ou stack),

PUSH (10)

PUSH

(2) POP

()

POP ()

PUSH

(6)

Assinale a opção que apresenta a lista de elementos armazenados na estrutura, após a execução das operações acima.

- a) 10, 2, 6

- b) 10, 2



e) 2

Comentários:

Precisamos definir os comandos.

- PUSH: inclui um elemento na pilha.
- POP: retira o último elemento incluído da pilha.

Então, vamos seguir linha a linha:

PUSH (10)

Inclui o número 10 na pilha.

Pilha: (base) [10] (topo)

PUSH (2)

Inclui o número 2 na pilha.

Pilha: (base) [10, 2] (topo)

POP ()

Remove o número 2 na pilha.

Pilha: (base) [10] (topo)

POP ()

Remove o número 10 na pilha.

Pilha: (base) [] (topo)

PUSH (6)

Insere o número 6 na pilha.

Pilha: (base) [6] (topo)

Ao fim, temos apenas o elemento 6.

Gabarito: Letra D

12. (CESPE - 2011 - FUB - Analista de Tecnologia da Informação - Específicos) As pilhas são listas encadeadas cujos elementos são retirados e acrescentados sempre ao final, enquanto as filas são listas encadeadas cujos elementos são retirados e acrescentados sempre no início.

Comentários:

Bem... o que é o final de uma Pilha? Pois é, não se sabe! O que existe é o Topo da Pilha, de onde sempre são retirados e acrescentados elementos. Em Filas, elementos são retirados do início e acrescentados no final.

Gabarito: Errado

13. (CESPE - 2013 - INPI - Analista de Planejamento - Desenvolvimento e Manutenção de Sistemas) Na estrutura de dados do tipo lista, todo elemento novo que é introduzido na pilha torna-se o elemento do topo.

Comentários:

Que questão confusa! Vamos comigo: vocês sabem muito bem que filas e pilhas são considerados espécies de listas. A questão inicialmente fala de uma lista, mas depois ela menciona uma pilha – podemos inferir, então, que se trata de uma lista do tipo pilha. Em uma pilha, todo elemento novo que é introduzido torna-se o elemento do topo, logo... questão correta!



Bem, esse foi o Gabarito Preliminar, mas a banca mudou de opinião e, no Gabarito Definitivo, permaneceu como errada. E a justificativa dela foi: “A ausência de especificação do tipo de lista no item torna correta a informação nele apresentada, razão pela qual se opta pela alteração de seu gabarito”. Vejam que bizarro: se torna correta a informação apresentada, o gabarito definitivo deveria ser C e, não, E. Além disso, a questão informa em sua segunda parte que se trata de uma pilha. Logo, não há que se falar em “ausência de especificação do tipo de lista”. Enfim, questão péssima, horrível e mal redigida :(

Gabarito: Errado

14. (CESPE - 2012 - TJ-RO - Analista Judiciário - Analista de Sistemas Suporte – E) Visitas a sítios armazenadas em um navegador na ordem last-in-first-out é um exemplo de lista.

Comentários:

Não! Last-In-First-Out (LIFO) é um exemplo de Pilha! Cuidado, pilhas podem ser implementadas como listas, mas esse não é o foco da questão.

Gabarito: Errado

15. (ESAF - 2013 - DNIT - Analista Administrativo - Tecnologia da Informação) Assinale a opção correta relativa às operações básicas suportadas por pilhas.

- a) Push: insere um novo elemento no final da pilha.
- b) Pop: adiciona elementos ao topo da pilha.
- c) Pull: insere um novo elemento no interior da pilha.
- d) Top: transfere o último elemento para o topo da pilha.
- e) Top: acessa o elemento posicionado no topo da pilha.

Comentários:

(a) Não, é no topo; (b) Não, remove do topo; (c) Não, não existe; (d) Não, simplesmente acessa e consulta o elemento do topo; (e) Perfeito! **Gabarito: E**

Gabarito: Errado

16. (FCC - 2012 – TST - Analista de Sistemas – C) As pilhas e as filas são estruturas de dados essenciais para os sistemas computacionais. É correto afirmar que a pilha é conhecida como lista FIFO - First In First Out.

Comentários:

Não! Pilha é LIFO e Fila é FIFO.

Gabarito: Errado

17. (FCC - 2012 – TRE/CE - Analista de Sistemas) Sobre pilhas é correto afirmar:

- a) Uma lista LIFO (Last-In/First-Out) é uma estrutura estática, ou seja, é uma coleção que não pode aumentar e diminuir durante sua existência.



- b) Os elementos na pilha são sempre removidos na mesma ordem em que foram inseridos.
- c) Uma pilha suporta apenas duas operações básicas, tradicionalmente denominadas push (insere um novo elemento no topo da pilha) e pop (remove um elemento do topo da pilha).
- d) Cada vez que um novo elemento deve ser inserido na pilha, ele é colocado no seu topo e, em qualquer momento, apenas aquele posicionado no topo da pilha pode ser removido.
- e) Sendo P uma pilha e x um elemento qualquer, a operação Push(P,x) diminui o tamanho da pilha P, removendo o elemento x do seu topo.

Comentários:

(a) Não, é uma estrutura dinâmica; (b) Não, é na ordem inversa (último a entrar é o primeiro a sair); (c) Não, há também Top ou Check, que acessar e consulta o elemento do topo; (e) Push é a operação de inserção de novos elementos na pilha, portanto aumenta seu tamanho adicionando o elemento x no topo.

E a letra D? Vamos lá! Sabemos que uma pilha é um tipo de lista - o que muda é apenas a perspectiva. Como assim? Eu vejo um monte de elementos em sequência. Ora, se eu coloco uma regra em que o primeiro elemento a entrar é o primeiro a sair, chamamos essa lista de fila; se eu coloco uma regra em que o primeiro elemento a entrar é o último a sair, chamamos essa lista de pilha.
Ok! Dito isso, o algoritmo é exatamente o mesmo, eu vou simplesmente mudar a perspectiva e a minha visão sobre a estrutura. Bacana?

Gabarito: Letra D

18. (CESPE - 2011 - EBC - Analista - Engenharia de Software) As pilhas, também conhecidas como listas LIFO ou PEPS, são listas lineares em que todas as operações de inserção e remoção de elementos são feitas por um único extremo da lista, denominado topo.

Comentários:

Não! LIFO é similar a UEPS (Último a Entrar, Primeiro a Sair). PEPS refere-se a Primeiro a Entrar, Primeiro a Sair, ou seja, FIFO.

Gabarito: Errado

19. (VUNESP - 2011 - TJM-SP - Analista de Sistemas - Judiciário) Lista do tipo LIFO (Last in, First Out) e lista do tipo FIFO (Firstin,First Out) são, respectivamente, características das estruturas de dados denominadas:

- a) Fila e Pilha.
- b) Pilha e Fila.
- c) Grafo e Árvore.
- d) Árvore e Grafo.
- e) Árvore Binária e Árvore Ternária.

Comentários:

E aí, já está automático para responder? Tem que ser automática: Pilha (LIFO) e Fila (FIFO).



20. (CESPE - 2010 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação - Arquitetura de Tecnologia) A definição da estrutura pilha permite a inserção e a eliminação de itens, de modo que uma pilha é um objeto dinâmico, cujo tamanho pode variar constantemente.

Comentários:

Essa questão é polêmica, porque é inevitável pensar em Pilhas Sequenciais (implementadas por vetores estáticos)! No entanto, é comum que as bancas tratem por padrão Pilha como Pilha Encadeada (implementadas por listas dinâmicas). Dessa forma, a questão está perfeita!

Gabarito: Correto

21. (CESPE - 2010 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação - Administração de Dados) Na representação física de uma pilha sequencial, é necessário uso de uma variável ponteiro externa que indique a extremidade da lista linear onde ocorrem as operações de inserção e retirada de nós.

Comentários:

As Pilhas oferecem três operações básicas: push, que insere um novo elemento no topo da pilha; pop, que remove um elemento do topo da pilha; e top (também conhecida como check), que acessa e consulta o elemento do topo da pilha. Pilhas podem ser implementadas por meio de Vetores (Pilha Sequencial - Alocação Estática de Memória) ou Listas (Pilha Encadeada - Alocação Dinâmica de Memória).

Conforme vimos em aula, a questão trata de uma Pilha Sequencial (i.e., implementada por meio de Vetores). Dessa forma, não é necessário o uso de ponteiros – esse seria o caso de uma Pilha Encadeada. Eu posso realmente dizer que é suficiente, mas não posso afirmar que é necessária a utilização de um ponteiro externo. Eu até poderia dizer que é necessário o uso de um indicador, mas ele também não necessariamente será um ponteiro. Logo, discordo do gabarito!

Gabarito: Correto

22. (CESPE - 2009 - ANAC - Técnico Administrativo - Informática) As operações de inserir e retirar sempre afetam a base de uma pilha.

Comentários:

Não, sempre afetam o topo da pilha!

Gabarito: Errado

23. (FCC - 2009 - TRT - 16ª REGIÃO (MA) - Técnico Judiciário - Tecnologia da Informação) Pilha é uma estrutura de dados:

- a) cujo acesso aos seus elementos segue tanto a lógica LIFO quanto a FIFO.
- b) cujo acesso aos seus elementos ocorre de forma aleatória.
- c) que pode ser implementada somente por meio de vetores.
- d) que pode ser implementada somente por meio de listas.



e) cujo acesso aos seus elementos segue a lógica LIFO, apenas.

Comentários:

(a) Não, somente LIFO; (b) Não, somente pelo Topo; (c) Não, pode ser por listas; (d) Não, pode ser por vetores; (e) Perfeito, é exatamente isso.

Gabarito: Errado

24. (CESPE - 2004 – STJ – Analista de Sistemas) Em geral, em uma pilha só se admite ter acesso ao elemento localizado em seu topo. Isso se adapta perfeitamente à característica das seqüências em que só o primeiro componente é diretamente acessível.

Comentários:

Perfeito, é exatamente isso – muda-se apenas a perspectiva!

Gabarito: Correto

25. (CESPE - 2004 – STJ – Analista de Sistemas) A seguir, está representada corretamente uma operação de desempilhamento em uma pilha de nome p.

```
se p.topo = 0 então
    nada {pilha vazia}
senão p.topo ← p.topo-1
```

Comentários:

Galera, a questão deu uma vaciladinho! O ideal seria dizer p.topo = null, mas vamos subentender que foi isso mesmo que ele quis dizer. Desse modo, se não tem topo, é porque a pilha está vazia. Se tiver topo, então o topo será o elemento anterior ao topo. O que ocorreu? Eu desempilhei a pilha!

Gabarito: Correto

26. (CESPE - 2010 – TRE/MT - Analista de Sistemas – A) O tipo nó é inadequado para implementar estruturas de dados do tipo pilha.

Comentários:

Não! Uma pilha pode ser implementada por meio de um vetor ou de uma lista. Nesse último caso, temos tipos nós!

Gabarito: Errado

27. (FGV – 2015 – DPE/MT – Analista de Sistemas) Assinale a opção que apresenta a estrutura de dados na qual o primeiro elemento inserido é o último a ser removido.

- a) Árvore
- b) Fila
- c) Pilha



e) Tabela de dispersão

Comentários:

Também conhecida como Lista LIFO (Last In First Out), basta lembrar de uma pilha de pratos esperando para serem lavados, i.e., o último a entrar é o primeiro a sair. A ordem em que os pratos são retirados da pilha é o oposto da ordem em que eles são colocados sobre a pilha e, como consequência, apenas o prato do topo da pilha está acessível.

Conforme vimos em aula, trata-se da Pilha. **Gabarito: C**

Gabarito: Correto

28. (FCC – 2012 – MPE/AP – Técnico Ministerial - Informática) Nas estruturas de dados,

- a) devido às características das operações da fila, o primeiro elemento a ser inserido será o último a ser retirado. Estruturas desse tipo são conhecidas como LIFO.
- b) as pilhas são utilizadas para controlar o acesso de arquivos que concorrem a uma única impressora.
- c) a fila é uma lista linear na qual as operações de inserção e retirada ocorrem apenas no início da lista.
- d) a pilha é uma lista linear na qual as operações de inserção e retirada são efetuadas apenas no seu topo.
- e) devido às características das operações da pilha, o último elemento a ser inserido será o último a ser retirado. Estruturas desse tipo são conhecidas como FIFO.

Comentários:

(a) as filas não são LIFO, mas sim FIFO, ou seja, o primeiro elemento da fila será, na verdade, o primeiro a ser retirado. Só pensarmos numa fila de banco, se alguém chega por último e é atendido primeiro, ficaria bem bravo, e vocês?? :D Item errado; (b) os trabalhos que chegam a uma impressora devem ser do tipo FIFO, ou seja, o primeiro trabalho enviado deve ser o primeiro a ser impresso. Item errado; (c) na fila os elementos são incluídos numa das extremidades e retirados da outra. Item errado; (d) na pilha as operações de inclusão na pilha quanto de retirada acontecem numa mesma extremidade. A extremidade escolhida é o topo da pilha. Item certo; (e) na verdade essas características são das filas. Item errado.

Gabarito: Letra D



GABARITO



- | | | |
|-----|-------------|-------|
| 1. | 11. | 21. C |
| 2. | 12. E | 22. E |
| 3. | 13. E | 23. E |
| 4. | 14. E | 24. C |
| 5. | 15. Letra E | 25. C |
| 6. | 16. E | 26. E |
| 7. | 17. D | 27. C |
| 8. | 18. E | 28. D |
| 9. | 19. B | |
| 10. | 20. C | |



LISTA DE QUESTÕES - FILAS - MULTIBANCAS

1. (IBFC – 2022 – AFEAM) Assinale, das alternativas abaixo, a única que identifica respectivamente uma Estrutura de Dados do tipo FIFO (First In, First Out) e uma outra com a Estrutura de dados do tipo LIFO (Last In, First Out):
 - a) lista – vetor
 - b) pilha – fila
 - c) vetor – lista
 - d) fila - pilha
2. (IF-T0 – 2022 – IF-T0) Em estrutura de dados os conceitos de FILAS e PILHAS são usados para implementar diversos recursos computacionais que vão desde compiladores e interpretadores a mecanismos usados nas linguagens de programação para auxiliar os desenvolvedores no dia a dia. Sobre essas estruturas, quais das definições abaixo são corretas?
 - a) Nas FILAS é usado o princípio do primeiro a entrar é o último a sair, já as PILHAS obedecem a regra do primeiro a entrar é o último a sair.
 - b) Nas FILAS é usado o princípio do primeiro a entrar é o primeiro a sair, já as PILHAS obedecem a regra do primeiro a entrar é o primeiro a sair.
 - c) Nas FILAS é usado o princípio do segundo a entrar é o primeiro a sair, já as PILHAS obedecem a regra do último a entrar é o último a sair.
 - d) Nas FILAS é usado o princípio do primeiro a entrar é o primeiro a sair, já as PILHAS obedecem a regra do primeiro a entrar é o último a sair.
 - e) Nas FILAS é usado o princípio do primeiro a entrar é o segundo a sair, já as PILHAS obedecem a regra do segundo a entrar é o terceiro a sair.
3. (UFRPE – 2022 – UFRPE) Sobre algoritmos e estrutura de dados, assinale a afirmativa correta.
 - a) Listas encadeadas ou ligadas são estruturas de dados estáticas, o que significa que o número de nós não pode ser modificado durante a execução do programa.
 - b) Pilhas são estruturas de dados do tipo FIFO (first-in first-out), em que o primeiro elemento a ser inserido será o primeiro a ser retirado.
 - c) Árvores são estruturas de dados do tipo FIFO (first-in first-out), em que o primeiro elemento a ser inserido será o primeiro a ser retirado.
 - d) Filas podem ser implementadas em listas encadeadas ou em vetores.
 - e) Pilhas só podem ser implementadas em listas encadeadas.
4. (CESPE/CEBRASPE – 2021 – SEED-PR) Em determinada estrutura de dados, os valores seguem a regra segundo a qual o último a entrar é o primeiro a sair.
Essa estrutura é do tipo
 - a) pilha.
 - b) fila.
 - c) lista encadeada.
 - d) lista duplamente encadeada.
 - e) matriz.
5. (CESPE/CEBRASPE – 2021 – SEED-PR) Na estrutura de dados denominada FILA,



- a) o último elemento a ser inserido será o primeiro a ser retirado.
- b) o primeiro elemento a ser inserido será o primeiro a ser retirado: adiciona-se item no fim e remove-se item do início.
- c) os elementos de um mesmo tipo de dado estão organizados de maneira sequencial e ordenada.
- d) os elementos não estão necessariamente armazenados sequencialmente na memória por ordem descrente de valores.
- e) os elementos são formados de índices em duas dimensões: linhas e colunas.
- 6. (CESPE/CEBRASPE – 2017 – TRT-7) A lógica FIFO (first-in first-out) é utilizada na estrutura de dados do tipo**
- a) pointer ou ponteiros.
- b) queue ou filas.
- c) stack ou pilhas.
- d) array ou matrizes.
- 7. (CESPE/CEBRASPE – 2017 – TRF-1) Acerca de estrutura de dados, julgue o próximo item.**
- A fila é uma lista de elementos em que os itens são sempre inseridos em uma das extremidades e excluídos da outra.
- 8. (CESPE/CEBRASPE – 2018 – TCE-MG)**
- Uma estrutura de dados em que o primeiro elemento inserido seja o primeiro elemento a ser retirado é denominada
- a) pilha.
- b) matriz.
- c) árvore binária.
- d) fila.
- e) lista.
- 9. (FCC – 2019 – TRF-4) O Round-Robin é um tipo de escalonamento preemptivo mais simples e consiste em repartir uniformemente o tempo da CPU entre todos os processos prontos para a execução. Os processos são organizados em uma estrutura de dados, alocando-se a cada um uma fatia de tempo da CPU, igual a um número de quanta. Caso um processo não termine dentro de sua fatia de tempo, retorna para o fim da estrutura e uma nova fatia de tempo é alocada para o processo que está no começo da estrutura e que dela sai para receber o tempo de CPU.**

A estrutura de dados utilizada nesse tipo de escalonamento é:

- a) pilha.
- b) árvore B.
- c) fila circular.
- d) fila simples.
- e) árvore binária.

- 10. (FCC – 2013 – MPE-MA) Ana precisa utilizar uma estrutura de dados para gerenciar trabalhos de impressão em uma impressora compartilhada por vários computadores em uma rede. As regras dessa estrutura devem permitir que os trabalhos sejam impressos na ordem em que forem enviados, ou seja, o primeiro a enviar um pedido de impressão deve ser o primeiro a ter sua solicitação atendida. Não deve ser permitido inserir pedidos de impressão no meio dos pedidos já realizados.**

A estrutura de dados mais adequada para Ana utilizar é

- a) pilha.



- b) lista encadeada ordenada.
- c) árvore binária.
- d) tabela hash.
- e) fila.

11. (FCC – 2013 – TRE-SP) No que se refere a estruturas de dados é INCORRETO afirmar:

- a) Numa fila dupla, os elementos podem ser inseridos e removidos de qualquer um dos extremos da fila.
- b) Em qualquer situação é possível usar uma única fila dupla para representar duas filas simples.
- c) A implementação de uma fila dupla normalmente é mais eficiente com uma lista duplamente encadeada que com uma encadeada simples.
- d) Pela definição de fila, se os elementos são inseridos por um extremo da lista linear, eles só podem ser removidos pelo outro.
- e) Numa lista singularmente encadeada, para acessar o último nodo é necessário partir do primeiro e ir seguindo os campos de ligação até chegar ao final da lista.

12. (CESPE - 2010 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação - Análise de Sistemas) Em um programa existe a necessidade de guardar todas as alterações feitas em determinado dado para que seja possível desfazer alterações feitas ao longo de toda a sua existência. Nessa situação, a estrutura de dados mais adequada para o armazenamento de todas as alterações citadas seria uma fila.

13. (CESPE - 2012 – TST – Analista de Sistemas – A) As pilhas e as filas são estruturas de dados essenciais para os sistemas computacionais. É correto afirmar que a fila é conhecida como lista LIFO - Last In First Out.

14. (CESPE - 2012 - TRE-RJ - Técnico Judiciário - Programação de Sistemas) As filas são estruturas com base no princípio LIFO (last in, first out), no qual os dados que forem inseridos primeiro na fila serão os últimos a serem removidos. Existem duas funções que se aplicam a todas as filas: PUSH, que insere um dado no topo da fila, e POP, que remove o item no topo da fila.

15. (FCC - 2012 - MPE-AP – Analista de Sistemas - A) Nas estruturas de dados, devido às características das operações da fila, o primeiro elemento a ser inserido será o último a ser retirado. Estruturas desse tipo são conhecidas como LIFO.

16. (FCC - 2012 - MPE-AP – Analista de Sistemas - C) Nas estruturas de dados, a fila é uma lista linear na qual as operações de inserção e retirada ocorrem apenas no início da lista.

17. (FCC - 2012 - TRE-SP - Analista Judiciário - Análise de Sistemas – D) Pela definição de fila, se os elementos são inseridos por um extremo da lista linear, eles só podem ser removidos pelo outro.

18. (FCC - 2011 - TRT - 19ª Região (AL) - Analista Judiciário - Tecnologia da Informação) FIFO refere-se a estruturas de dados do tipo:

- a) fila.
- b) árvore binária.
- c) pilha.
- d) matriz quadrada.
- e) cubo.



19. (ESAF - 2010 - CVM - Analista de Sistemas - prova 2) Uma fila é um tipo de lista linear em que:
- a) as inserções são realizadas em um extremo e as remoções no outro extremo.
 - b) as inserções e remoções são realizadas em um mesmo extremo.
 - c) podem ser realizadas apenas inserções.
 - d) a inserção de um elemento requer a remoção de outro elemento.
 - e) a ordem de saída não corresponde à ordem de entrada dos elementos.
20. (CESPE - 2010 - DETRAN-ES - Analista de Sistemas) No armazenamento de dados pelo método FIFO (first in - first out), a estrutura de dados é representada por uma fila, em cuja posição final ocorrem inserções e, na inicial, retiradas.
21. (CESPE - 2008 - TRT - 5ª Região (BA) - Técnico Judiciário - Tecnologia da Informação) Entre alguns tipos de estrutura de dados, podem ser citados os vetores, as pilhas e as filas.
22. (CESPE - 2004 – SES/PA – Analista de Sistemas) Uma estrutura mais geral que as pilhas e filas é o deque, em que as inserções, retiradas e acessos são permitidos em ambas as extremidades.
23. (CESPE - 2009 – TCE/AC – Analista de Sistemas – D) Um deque (double ended queue) requer inserção e remoção no topo de uma lista e permite a implementação de filas com algum tipo de prioridade. A implementação de um deque, geralmente é realizada com a utilização de uma lista simplesmente encadeada.
24. (FCC - 2007 – TRT/23 – Analista de Sistemas) Uma estrutura de dados com vocação de FIFO de duplo fim e que admite a rápida inserção e remoção em ambos os extremos é:
- a) uma pilha.
 - b) uma splay tree.
 - c) um deque.
 - d) uma lista linear.
 - e) uma árvore AVL.
25. (CESPE - 2004 – PBV/RR - Analista de Sistemas) As filas com prioridade são listas lineares nas quais os elementos são pares da forma (q_i, p_i) , em que q é o elemento do tipo base e p é uma prioridade. Elas possuem uma política de fila do tipo FIFO (first in first out) entre os elementos de mesma prioridade.
26. (CESPE - 2004 – STJ – Analista de Sistemas) A seguir, está representada corretamente uma operação de retirada em uma fila de nome f .
- ```
se f.começo = nil então
 erro {fila vazia}
senão j ← f.começo.info
```
27. (FCC - 2016 - Prefeitura de Teresina - PI - Analista Tecnológico - Analista de Suporte Técnico) Considerando uma estrutura de dados do tipo fila, e a seguinte sequência de comandos sobre essa fila (sendo que o comando Push representa uma inserção de elemento e o comando Pop representa uma exclusão de elemento) e considerando também que a fila estava inicialmente vazia:



**Push 3, Push 5, Pop 3, Push 7, Pop 5, Push 9, Push 8**

**Após a execução dessa sequência de comandos, o conjunto de elementos que resulta na fila é:**

- a) 3 – 5 – 7 – 9 – 8.
- b) 7 – 9 – 8 – 3 – 5.
- c) 3 – 3 – 5 – 5 – 7 – 9 – 8.
- d) 7 – 9 – 8.
- e) 3 – 5 – 3 – 7 – 5 – 9 – 8.

**28. (FCC - 2016 – TRT - 23ª REGIÃO (MT) - Técnico Judiciário - Tecnologia da Informação) Estruturas de dados básicas, como as pilhas e filas, são usadas em uma gama variada de aplicações. As filas, por exemplo, suportam alguns métodos essenciais, como o:**

- a) enqueue(x), que insere o elemento x no fim da fila, sobrepondo o último elemento.
- b) dequeue(), que remove e retorna o elemento do começo da fila; um erro ocorrerá se a fila estiver vazia.
- c) push(x), que insere o elemento x no topo da fila, sem sobrepor nenhum elemento.
- d) pop(), que remove o elemento do início da fila e o retorna, ou seja, devolve o último elemento inserido.
- e) top(), que retorna o elemento do fim da fila sem removê-lo; um erro ocorrerá se a fila estiver vazia.

**29. (FCC - 2017 – TRE/BA - Analista de Sistemas) A estrutura que, além de ser similar à fila, é apropriada para ampliar as características desta, permitindo inserir e retirar elementos tanto do início quanto do fim da fila, é o(a):**

- a) árvore.
- b) lista duplamente encadeada.
- c) deque.
- d) fila circular.
- e) pilha



## GABARITO

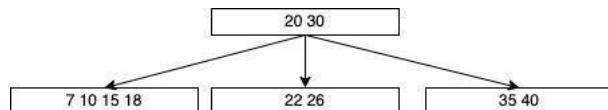


- |             |             |             |
|-------------|-------------|-------------|
| 1. Letra D  | 11. Letra B | 21. Correto |
| 2. Letra D  | 12. Errado  | 22. Correto |
| 3. Letra D  | 13. Errado  | 23. Errado  |
| 4. Letra A  | 14. Errado  | 24. Letra C |
| 5. Letra B  | 15. Errado  | 25. Correto |
| 6. Letra B  | 16. Errado  | 26. Errado  |
| 7. Correto  | 17. Correto | 27. Letra D |
| 8. Letra D  | 18. Letra A | 28. Letra B |
| 9. Letra C  | 19. Letra A | 29. Letra C |
| 10. Letra E | 20. Correto |             |



## LISTA DE QUESTÕES - ESTRUTURAS DE DADOS - ÁRVORE 42 -

1. (FGV – 2022 – SEMSA Manaus) Observe a configuração de uma árvore B, onde uma página pode ter no máximo 4 filhas, contendo as chaves 7, 10, 15, 18, 20, 22, 26, 30, 35, 40.



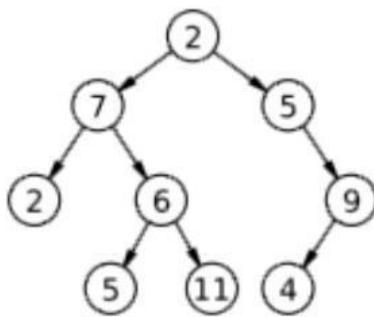
Após a inserção da chave 5, a configuração das chaves do nó raiz da árvore seria

- a) 5, 20, 30
- b) 10, 20, 30
- c) 5, 30
- d) 10, 30
- e) 20, 30

2. (FGV – 2022 – SEMSA Manaus) Numa estrutura de dados do tipo Árvore B, onde cada nó não raiz pode conter entre d e 2.d chaves, a complexidade do algoritmo de busca é da ordem
- a) log de N na base 2.
  - b) log de N na base d.
  - c) N vezes log de N na base 2.
  - d) N.
  - e)  $N^2$ .

3. (UFPRE – 2022 – UFPRE) Acerca de estruturas de dados, assinale a alternativa correta.
- a) A estrutura denominada Pilha é considerada do tipo FIFO (first in, first out); o primeiro elemento inserido será o primeiro elemento a ser removido.
  - b) A estrutura denominada Fila é considerada do tipo FILO (first in, last out); o primeiro elemento a ser inserido será o último elemento a ser removido.
  - c) A estrutura denominada lista simplesmente encadeada não ordenada armazena um ou vários dados em cada elemento, e tem um ponteiro apontado para o último elemento que permite o encadeamento e a estrutura linear.
  - d) A estrutura denominada árvore é um conjunto finito de elementos, onde cada elemento é denominado nó, e o primeiro nó é conhecido como raiz da árvore.
  - e) A estrutura denominada árvore AVL é uma árvore binária não balanceada, em que cada nó representa uma diferença de altura entre as subárvore direita e esquerda de 1, 2 ou 3 nós.
4. (MetroCapital Soluções – 2022 – Prefeitura de Nova Odessa – SP) A Estrutura de dados (ED) é um modo particular de armazenamento e organização de dados em um computador de modo que possam ser usados eficientemente. Analise a imagem a seguir:

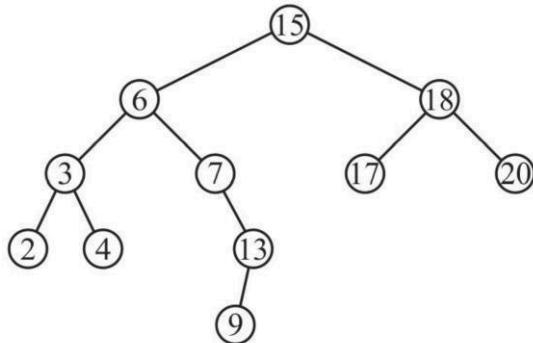




Qual estrutura de dados representa a imagem:

- a) Pilha.
- b) Fila.
- c) Grafo.
- d) Vetor de vetores.
- e) Árvore binária.

5. (CESPE/CEBRASPE – 2020 – TJ-PA)



Thomas H. Cormen et al. *Algoritmos: teoria e prática*. Editora Campus, v. 2, 2002. p. 207.

De acordo com a figura anterior, o procedimento CONSULTA (x)

1 while esquerda [x] ≠ NIL

2        do x ← esquerda [x]

3            return x

realiza, na árvore, a consulta de

- a) search.
- b) minimum.
- c) maximum.
- d) successor.
- e) predecessor.

6. (CESPE/CEBRASPE – 2022 – Petrobrás) Uma árvore de decisão representa um determinado número de caminhos possíveis de decisão e os resultados de cada um deles, apresentando muitos pontos positivos, ou seja, são fáceis de entender e interpretar. Elas têm processo de previsão completamente transparente e lidam facilmente com diversos atributos numéricos, assim como atributos categóricos, podendo até mesmo classificar dados sem atributos definidos.



De acordo com os aspectos construtivos de uma árvore de decisão, julgue o item a seguir.

A entropia de uma árvore de decisão aborda o aspecto da quantidade de informações que está associada às respostas que podem ser obtidas às perguntas formuladas, representando o grau de incerteza associado aos dados.

7. (CESPE/CEBRASPE – 2022 – Petrobrás) Uma árvore de decisão representa um determinado número de caminhos possíveis de decisão e os resultados de cada um deles, apresentando muitos pontos positivos, ou seja, são fáceis de entender e interpretar. Elas têm processo de previsão completamente transparente e lidam facilmente com diversos atributos numéricos, assim como atributos categóricos, podendo até mesmo classificar dados sem atributos definidos.

De acordo com os aspectos construtivos de uma árvore de decisão, julgue o item a seguir.

Se o processo adotado para a construção de árvores de decisão for determinístico, uma forma de obtenção de árvores aleatórias, que compõem as florestas aleatórias, pode ser realizada por meio do bootstrap dos dados, em que cada árvore é treinada com base no resultado de bootstrap\_sample (inputs).

8. (FGV – 2022 – MPE-GO) Árvores B são muito usadas na implementação de índices em bancos de dados.

Uma árvore desse tipo é dita balanceada quando

- a) a complexidade do algoritmo de busca é logarítmica.
- b) as chaves são armazenadas em ordem de classificação, crescente ou decrescente.
- c) é possível localizar registros referenciados por um intervalo de chaves.
- d) o número de ponteiros em cada nó intermediário é constante.
- e) toda página folha tem o mesmo número de páginas intermediárias até a raiz.

9. (FGV – 2018 – Câmara de Salvador - BA) Gerenciadores de bancos de dados frequentemente empregam índices implementados na forma de árvores B. Nesse tipo de organização, considerando-se uma árvore na qual o número máximo de chaves numa página não folha é 19 (ou seja,  $d=20$ ), o número máximo de acessos necessários para localizar uma chave, num universo de 10 milhões de chaves distintas, é:

- a) 4;
- b) 7;
- c) 19;
- d) 100;
- e) 316.

10. (FGV – 2018 – MPE-AL) Em uma árvore B de ordem  $d$ , onde cada nó que não o raiz possui entre  $d$  e  $2d$  chaves, estão armazenadas 30.000 chaves.

Sabendo-se que  $d=8$ , assinale a opção que indica o número máximo de nós visitados para a localização de uma chave.

- a) 3
- b) 5
- c) 7
- d) 15
- e) 15.000

11. (FGV – 2018 – Banestes) Sobre as características de índices estruturados na forma de Btrees e Hash tables, analise as afirmativas a seguir.

- I. Hash tables aplicam-se somente em buscas que referenciam a chave por inteiro (operador =).
- II. B-trees favorecem consultas que buscam chaves num determinado intervalo (operadores  $\geq$  e  $\leq$ ).



IV. Hash tables favorecem buscas, com o operador ‘LIKE’ do SQL, que não contenham caracteres curingas na primeira posição.

V. B-trees não se aplicam em buscas que se referem a uma substring à esquerda da chave.

Está correto o que se afirma em:

- a) nenhuma;
- b) somente I, II e III;
- c) somente I, IV e V;
- d) somente II, III, IV;
- e) I, II, III, IV e V.

**12. (FCC – 2017 – TRE-SP) Considere, hipoteticamente, que um Técnico do TRE-SP tem, em seu computador, a seguinte organização de um diretório**

Principal: Dados

Dentro de Dados: Técnicos Práticos

Dentro de Técnicos: Árvores Hash      Recursão      Filas      Pilhas

Dentro de Práticos: Programas      AFazer Prontos

Dentro de Prontos: Eleições      Urnas

Dentro de Programas: Corretos

ComErro

Dentro de ComErro: Urgentes      Pendentes      Antigos

A estrutura de dados

- a) fila é a mais adequada para representar este diretório.
- b) pilha é a mais adequada para representar este diretório.
- c) árvore binária, ao armazenar este diretório, terá Dados na raiz e nós com grau 2, 3, 5 e folhas.
- d) árvore, que consegue armazenar este diretório, é de ordem 5.
- e) hashing, ao armazenar este diretório, não terá colisões na tabela de dispersão

**13. (FCC – 2019 – TRF-4) Determinada estrutura de dados foi projetada para minimizar o número de acessos à memória secundária. Como o número de acessos à memória secundária depende diretamente da altura da estrutura, esta foi concebida para ter uma altura inferior às estruturas hierarquizadas similares, para um dado número de registros. Para manter o número de registros armazenados e, ao mesmo tempo, diminuir a altura, uma solução é aumentar o grau de ramificação da estrutura (o número máximo de filhos que um nó pode ter). Assim, esta estrutura possui um grau de ramificação geralmente muito maior que 2. Além disso, a cada nó são associados mais de um registro de dados: se o grau de ramificação de um nó for g, este pode armazenar até g-1 registros.**

Esta estrutura de dados é utilizada em banco de dados e sistema de arquivos, sendo denominada

- a) árvore digital ou trie.
- b) árvore B.
- c) lista linear duplamente encadeada circular.
- d) árvore rubro-negra.
- e) árvore binária de busca não balanceada.

**14. (FCC – 2015 – DPE-SP) Atenção: Para responder à próxima questão, considere as declarações em pseudocódigo abaixo.**

Considere que \* indica ponteiro ou apontador.



```

 info: inteiro
 *prox: tipoNo
 fim registro
Var *inicio, *ant, *aux, *novo, *fim: tipoNo
Função Fila1 (info: inteiro)
Início
 novo ← aloca (*tipoNo)
 novo->info ← info
 novo->prox ← NULO
 se (inicio = NULO)
 então
 inicio ← novo
 fim ← novo
 senão
 fim ← novo
 aux ← inicio
 enquanto (aux ≠ NULO)
 ant ← aux
 aux ← aux->prox
 fim enquanto
 ant->prox ← novo
 fim se
Fim

Função Fila2(info: inteiro)
Início
 se (inicio = NULO)
 então
 imprima ("Fila vazia")
 senão
 aux ← inicio
 inicio ← inicio->prox
 se (inicio = NULO)
 fim ← NULO;
 fim se
 fim se
 libera (aux)
fim se
Fim

```

As funções Fila1 e Fila2 implementam operações em filas. Além das filas, há diversas outras estruturas muito úteis na solução de problemas, dentre as quais encontram-se as

- pilhas, também conhecidas como listas FIFO (First In, First Out).
- deques, que são pilhas que permitem inserir e remover dados em ambas as extremidades.
- árvores n-árias, estruturas de dados lineares que não são adequadas para representar dados que devem ser dispostos de maneira hierárquica, como diretórios criados em um computador.
- árvores binárias de busca, cujas funções que realizam percursos são naturalmente implementadas usando-se recursividade.



- e) árvores binárias平衡adas, nas quais, para cada nó, as alturas de suas subárvores diferem de, no máximo, 2. Nelas, o custo das operações depende da altura da árvore, por isso elas devem ter a maior altura possível.

**15. (FGV – 2022 – SEFAZ-AM)** A estrutura de dados usada em índices multiníveis dinâmicos em banco de dados relacionais, que garantem que tais estruturas sempre estejam balanceadas e que o espaço desperdiçado pela exclusão de itens de dados, se houver, nunca se torne excessivo, é denominada

- a) fila.
- b) hash.
- c) bitmap.
- d) árvore B.
- e) árvore binária.

**16. (FGV – 2021 – IMBEL)** Considere uma árvore B+ com as seguintes características.

- I. A raiz é uma folha ou um nó que contém, no mínimo, dois filhos.
- II. Cada nó diferente do nó raiz e das folhas possui no mínimo d filhos.
- III. Cada nó tem no máximo 2d filhos. Cada nó possui entre d-1 e 2d-1 chaves, exceto o raiz que possui entre 1 e 2d-1 chaves.
- IV. Somente os nós folhas contêm dados associados às chaves.

Assinale o número máximo de acessos necessários para localizar uma chave, com d=10, num universo de 10 milhões de chaves.

- a) 5
- b) 7
- c) 10
- d) 100
- e) 1.000

**17. (CESPE - 2012 - Banco da Amazônia - Técnico Científico - Administração de Dados)** As operações de busca em uma árvore binária não a alteram, enquanto operações de inserção e remoção de nós provocam mudanças sistemáticas na árvore.

**18. (CETAP - 2010 - AL-RR - Analista de Sistemas - A)** Uma árvore binária é aquela que tem como conteúdo somente valores binários.

**19. (CESPE - 2012 - Banco da Amazônia - Técnico Científico - Administração de Dados)** O tipo de dados árvore representa organizações hierárquicas entre dados.

**20. (CETAP - 2010 - AL-RR - Analista de Sistemas - B)** Uma árvore é composta por duas raízes, sendo uma principal e a outra secundária.

**21. (CESPE - 2010 - DETRAN-ES - Analista de Sistemas)** Denomina-se árvore binária a que possui apenas dois nós.

**22. (FUNCAB - 2010 - SEJUS-RO - Analista de Sistemas - II)** Árvores são estruturas de dados estáticas com sua raiz representada no nível um.



**23. (CESPE - 2009 - ANAC - Especialista em Regulação - Economia) Considerando-se uma árvore binária completa até o nível 5, então a quantidade de folhas nesse nível será 24.**

**24. (CESPE - 2009 – ANAC - Analista de Sistemas) Uma árvore binária completa até o nível 10 tem 2.047 nós.**

**25. (FGV – 2015 – DPE/MT – Analista de Sistemas) No desenvolvimento de sistemas, a escolha de estruturas de dados em memória é especialmente relevante. Dentre outras classificações, é possível agrupar essas estruturas em lineares e não lineares, conforme a quantidade de sucessores e antecessores que os elementos da estrutura possam ter. Assinale a opção que apresenta, respectivamente, estruturas de dados lineares e não lineares.**

- a) Tabela de dispersão e fila.
- b) Estrutura de seleção e pilha.
- c) Pilha e estrutura de seleção.
- d) Pilha e árvore binária de busca.
- e) Fila e pilha.

**26. (CESPE - 2010 – TRE/MT – Analista de Sistemas – B) As listas, pilhas, filas e árvores são estruturas de dados que têm como principal característica a sequencialidade dos seus elementos.**

**27. (FCC - 2012 - MPE-AP - Analista Ministerial - Tecnologia da Informação – A) A árvore é uma estrutura linear que permite representar uma relação de hierarquia. Ela possui um nó raiz e subárvore não vazias.**

**28. (CESPE - 2010 – TRE/MT - Analista de Sistemas – E) O uso de recursividade é totalmente inadequado na implementação de operações para manipular elementos de uma estrutura de dados do tipo árvore.**

**29. (FCC - 2011 - TRT - 19ª Região (AL) - Técnico Judiciário - Tecnologia da Informação) Em uma árvore binária, todos os nós têm grau:**

- a) 2.
- b) 0, 1 ou 2.
- c) divisível por 2.
- d) maior ou igual a 2.
- e) 0 ou 1.

**30. (CESPE - 2011 – STM – Analista de Sistemas) Enquanto uma lista encadeada somente pode ser percorrida de um único modo, uma árvore binária pode ser percorrida de muitas maneiras diferentes.**

**31. (FCC - 2016 - Prefeitura de Teresina - PI - Analista Tecnológico - Analista de Suporte Técnico) Considerando a estrutura de dados denominada árvore,**



- a) a sua altura é definida como a profundidade média de todos os seus vértices.
- b) um vértice com um ou dois filhos é denominado folha.
- c) cada nó tem no mínimo dois filhos em uma árvore binária.
- d) as folhas de uma árvore binária completa podem ter profundidades distintas entre si.
- e) a profundidade de um vértice em uma árvore é definida como o comprimento da raiz da árvore até esse vértice.

**32. (CESPE – 2017 – TRE/BA - Analista de Sistemas)** No estabelecimento de uma estrutura hierárquica, foi definida a seguinte árvore binária S:

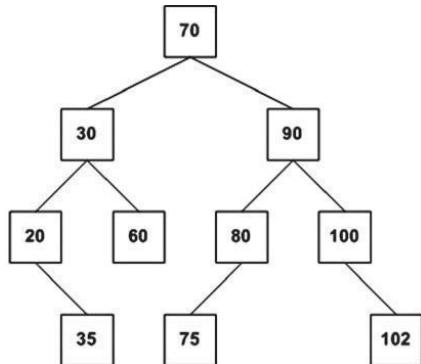
$$S = (12(10(9(8))(11))(14(13)(15)))$$

Considerando o resultado da operação de exclusão do nó 12, assinale a opção que corresponde a nova estrutura da árvore S.

- a)  $(10(9(8))(11(14(13)(15)))$
- b)  $(11(9(8)(10))(14(13)(15)))$
- c)  $(11(10(9(8)))(14(13)(15)))$
- d)  $(13(10(9)(11))(14(15)))$
- e)  $(13(11(9)(10))(14(15)))$

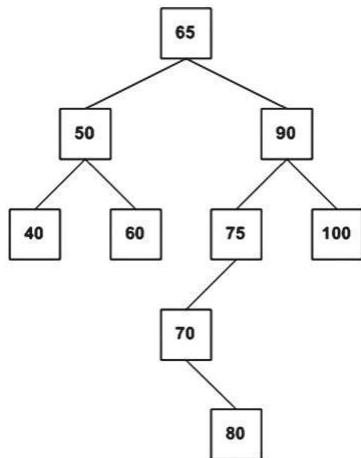
**33. (CESGRANRIO – 2012 – PETROBRÁS - Analista de Sistemas)** Qual figura representa uma árvore AVL?

a)

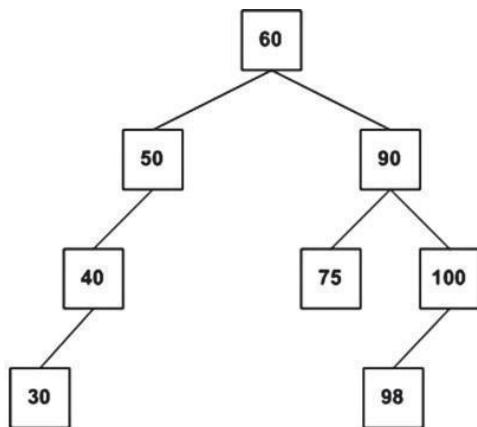


b)

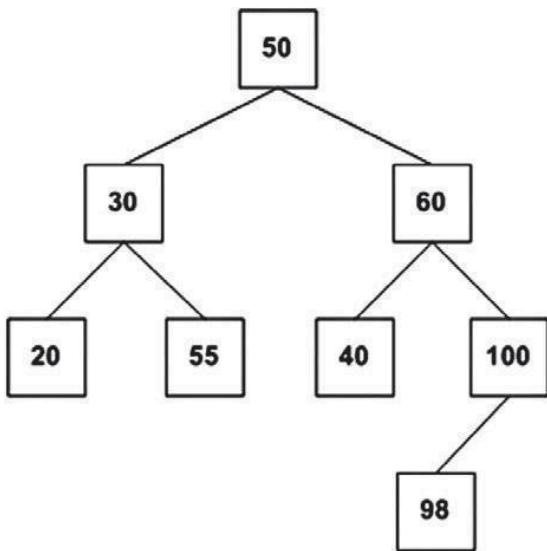




c)

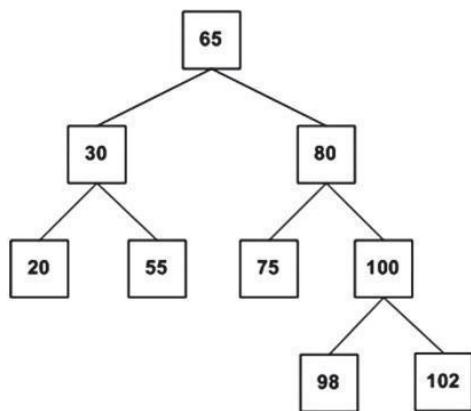


d)

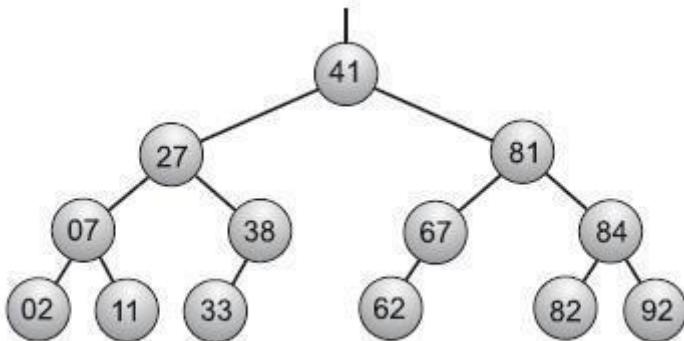


e)





34. (CESGRANRIO – 2006 – DECEA – Analista de Sistemas) Com base na seguinte árvore AVL.



A inserção do elemento 30 nessa árvore:

- a) aumenta a profundidade da árvore após uma rotação.
- b) provoca uma rotação à direita.
- c) deixa os nós 02 e 07 no mesmo nível.
- d) altera a raiz da árvore (nó 41).
- e) torna o nó 33 pai do nó 27.

35. (CESPE – 2012 – TJ/RO – Analista de Sistemas) Assinale a opção em que é apresentado exemplo de estrutura de informação do tipo abstrata, balanceada, não linear e com relacionamento hierárquico.

- a) lista duplamente encadeada
- b) árvore binária
- c) pilha
- d) árvore AVL
- e) deque

36. (FCC – 2008 – TRT/18 - Analista de Sistemas) Árvore AVL balanceada em altura significa que, para cada nó da árvore, a diferença entre as alturas das suas sub-árvore (direita e esquerda) sempre será:



- a) menor ou igual a 2.
- b) igual a 0 ou -1.
- c) maior que 1.
- d) igual a 1.
- e) igual a -1, 0 ou 1.

**37. (CESGRANRIO – 2010 – PETROBRÁS - Analista de Sistemas)** Uma árvore AVL é uma estrutura de dados muito usada para armazenar dados em memória. Ela possui algumas propriedades que fazem com que sua altura tenha uma relação muito específica com o número de elementos nela armazenados. Para uma folha, cuja altura é igual a um, tem-se uma árvore AVL com 6 nós.

Qual é a altura máxima que esta árvore pode ter?

- a) 6
- b) 5
- c) 4
- d) 3
- e) 2

**38. (CESGRANRIO – 2011 – PETROBRÁS - Analista de Sistemas)** Uma árvore AVL é uma árvore binária de busca autobalanceada que respeita algumas propriedades fundamentais. Como todas as árvores, ela tem uma propriedade chamada altura, que é igual ao valor da altura de sua raiz.

Sabendo que a altura de uma folha é igual a um e que a altura de um nó pai é igual ao máximo das alturas de seus filhos mais um, qual estrutura NÃO pode representar uma árvore AVL?

- a) Uma árvore vazia
- b) Uma árvore com dois nós
- c) Uma árvore com três nós e altura igual a dois
- d) Uma árvore com três nós e altura igual a três
- e) Uma árvore com seis nós e altura igual a três

**39. (CESGRANRIO – 2011 – PETROBRÁS - Analista de Sistemas)** Após a inserção de um nó, é necessário verificar cada um dos nós ancestrais desse nó inserido, relativamente à consistência com as regras estruturais de uma árvore AVL.

#### PORQUE

O fator de balanceamento de cada nó, em uma árvore AVL, deve pertencer ao conjunto formado por  $\{-2, -1, 0, +1, +2\}$ .

Analisando-se as afirmações acima, conclui-se que:



- a) as duas afirmações são verdadeiras, e a segunda justifica a primeira.
- b) as duas afirmações são verdadeiras, e a segunda não justifica a primeira.
- c) a primeira afirmação é verdadeira, e a segunda é falsa.
- d) a primeira afirmação é falsa, e a segunda é verdadeira.
- e) as duas afirmações são falsas.

**40. (CESGRANRIO – 2010 – EPE - Analista de Sistemas)** Um programador decidiu utilizar, em determinado sistema de análise estatística, uma árvore AVL como estrutura de dados. Considerando-se  $n$  a quantidade de elementos dessa árvore, o melhor algoritmo de pesquisa, com base em comparações, possui complexidade de tempo, no pior caso, igual a:

- a)  $O(1)$
- b)  $O(\log n)$ .
- c)  $\Omega(n)$
- d)  $\Omega(n \log n)$
- e)  $\Omega(n^2)$

**41. (CESGRANRIO – 2012 – PETROBRÁS - Analista de Sistemas)** Todos os  $N$  nomes de uma lista de assinantes de uma companhia telefônica foram inseridos, em ordem alfabética, em três estruturas de dados: uma árvore binária de busca, uma árvore AVL e uma árvore B.

As alturas resultantes das três árvores são, respectivamente,

- a)  $O(\log(N))$ ,  $O(\log(N))$ ,  $O(1)$
- b)  $O(\log(N))$ ,  $O(N)$ ,  $O(\log(N))$
- c)  $O(N)$ ,  $O(\log(N))$ ,  $O(1)$
- d)  $O(N)$ ,  $O(\log(N))$ ,  $O(\log(N))$
- e)  $O(N)$ ,  $O(N)$ ,  $O(\log(N))$

**42. (IBFC – 2014 – TRE/AM - Analista de Sistemas)** Quanto ao Algoritmo e estrutura de dados no caso de árvore AVL (ou árvore balanceada pela altura), analise as afirmativas abaixo, dê valores Verdadeiro (V) ou Falso (F) e assinale a alternativa que apresenta a sequência correta de cima para baixo:

- ( ) Uma árvore AVL é dita balanceada quando, para cada nó da árvore, a diferença entre as alturas das suas sub-árvores (direita e esquerda) não é maior do que um.
- ( ) Caso a árvore não esteja balanceada é necessário seu balanceamento através da rotação simples ou rotação dupla.

Assinale a alternativa correta:

- a) F-F
- b) F-V
- c) V-F
- d) V-V



43. (CESGRANRIO – 2010 – PETROBRÁS - Analista de Sistemas) Uma sequência desordenada de números armazenada em um vetor é inserida em uma árvore AVL. Após a inserção nesta árvore, é feito um percurso em ordem simétrica (em ordem) e o valor de cada nó visitado é inserido em uma pilha. Depois de todos os nós serem visitados, todos os números são retirados da pilha e apresentados na tela. A lista de números apresentada na tela está:

- a) ordenada ascendentemente de acordo com os números.
- b) ordenada descendenteamente de acordo com os números.
- c) na mesma ordem do vetor original.
- d) na ordem inversa do vetor original.
- e) ordenada ascendentemente de acordo com sua altura na árvore.

44. (FGV – 2009 – MEC - Analista de Sistemas) Acerca das estruturas de dados Árvores, analise as afirmativas a seguir.

- I. A árvore AVL é uma árvore binária com uma condição de balanço, porém não completamente balanceada.
- II. Árvores admitem tratamento computacional eficiente quando comparadas às estruturas mais genéricas como os grafos.
- III. Em uma Árvore Binária de Busca, todas as chaves da subárvore esquerda são maiores que a chave da raiz.

45. (CESPE – 2014 – TJ/SE - Analista de Sistemas) Em uma árvore AVL (Adelson-Velsky e Landis), caso a diferença de altura entre as sub-árvores de um nó seja igual a 2 e a diferença de altura entre o nó filho do nó desbalanceado seja igual a -1, deve-se realizar uma rotação dupla com o filho para a direita e o pai para a esquerda a fim de que a árvore volte a ser balanceada.

46. (CESPE – 2010 – PETROBRÁS - Analista de Sistemas) As árvores usadas como estruturas de pesquisa têm características especiais que garantem sua utilidade e propriedades como facilidade de acesso aos elementos procurados em cada instante. A esse respeito, considere as afirmações abaixo.

- I - A árvore representada na figura (I) acima não é uma árvore AVL, pois as folhas não estão no mesmo nível.
- II - A sequência 20, 30, 35, 34, 32, 33 representa um percurso sintaticamente correto de busca do elemento 33 em uma árvore binária de busca.
- III - A árvore representada na figura (II) acima é uma árvore binária, apesar da raiz não ter filhos.

É (São) correta(s) APENAS a(s) afirmativa(s):



- a) I.
- b) II.
- c) III.
- d) I e II.
- e) II e III.

**47. (CESPE – 2010 – PETROBRÁS - Analista de Sistemas)** No sistema de dados do Departamento de Recursos Humanos de uma grande empresa multinacional, os registros de funcionários são armazenados em uma estrutura de dados do tipo árvore binária AVL, onde cada registro é identificado por uma chave numérica inteira. Partindo de uma árvore vazia, os registros cujas chaves são 23, 14, 27, 8, 18, 15, 30, 25 e 32 serão, nessa ordem, adicionados à árvore.

Dessa forma, o algoritmo de inserção na árvore AVL deverá realizar a primeira operação de rotação na árvore na ocasião da inserção do elemento:

- a) 30
- b) 25
- c) 18
- d) 15
- e) 8

**48. (CESPE – 2014 – TJ/SE - Analista de Sistemas)** Existem dois vetores, chamados A e B, que estão ordenados e contêm N elementos cada, respeitando a propriedade  $A[N-1] < B[0]$ , onde os índices de ambos os vetores vão de 0 a  $N-1$ . Retiram-se primeiro todos os elementos de A na ordem em que se apresentam e inserem-se esses elementos em uma árvore binária de busca, fazendo o mesmo depois com os elementos de B, que são inseridos na mesma árvore de busca que os de A. Depois, retiram-se os elementos da árvore em um percurso pós ordem, inserindo-os em uma pilha. Em seguida retiram-se os elementos da pilha, que são inseridos de volta nos vetores, começando pelo elemento 0 do vetor A e aumentando o índice em 1 a cada inserção, até preencher todas as N posições, inserindo, então, os N elementos restantes no vetor B da mesma maneira.

Ao final do processo, tem-se que os vetores:

- a) estão ordenados e  $A[i] < B[i]$ , para todo  $i=0, \dots, N-1$ .
- b) estão ordenados e  $A[i] > B[i]$ , para todo  $i=0, \dots, N-1$ .
- c) estão ordenados e não existe mais uma propriedade que relate  $A[i]$  e  $B[i]$ .
- d) não estão ordenados e  $A[i] < B[i]$ , para todo  $i=0, \dots, N-1$ .
- e) não estão ordenados e  $A[i] > B[i]$ , para todo  $i=0, \dots, N-1$ .



## GABARITO

1. Letra B
2. Letra B
3. Letra D
4. Letra E
5. Letra B
6. Correto
7. Correto
8. Letra E
9. Letra B
10. Letra B
11. Letra B
12. Letra D
13. Letra B
14. Letra D
15. Letra D
16. Letra B
17. Correto
18. Errado
19. Correto
20. Errado
21. Errado
22. Errado
23. Errado
24. Correto
25. Letra D
26. Errado
27. Errado
28. Errado
29. Errado
30. Letra B
31. Letra E
32. Anulada
33. Letra E
34. Letra B
35. Letra D
36. Letra E
37. Letra D
38. Letra D
39. Letra C
40. Letra B
41. Letra D
42. Letra D
43. Letra B
44. Letra B
45. Letra C
46. Letra E
47. Letra D
48. Letra A

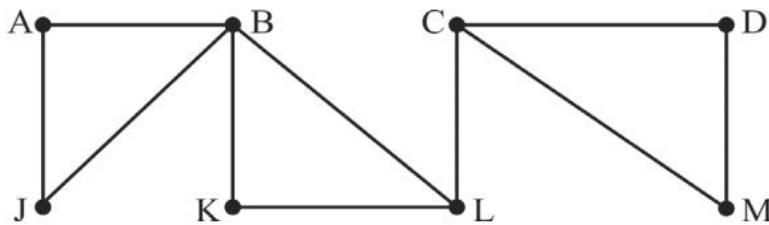


## LISTA DE QUESTÕES - GRAFOS - MULTIBANCAS

1. (CESPE/CEBRASPE – 2017 – TRF-1) Acerca dos conceitos de árvores e grafos, julgue o item que se segue.

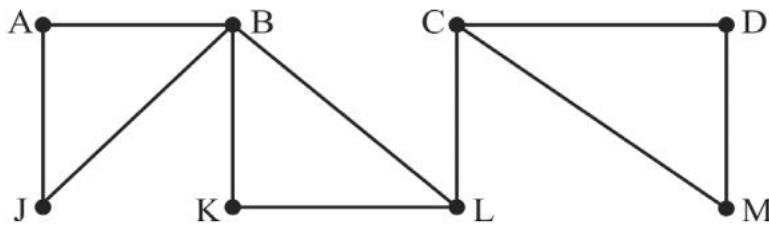
A soma dos graus de todos os vértices de um grafo é sempre par.

2. (CESPE/CEBRASPE – 2018 – PF)



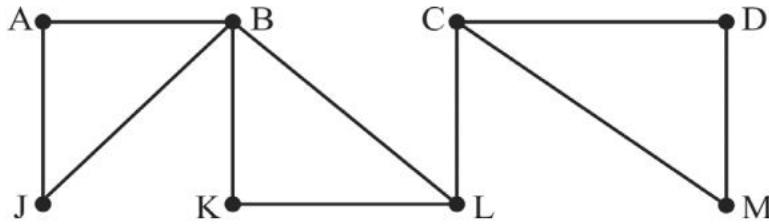
Considerando a terminologia e os conceitos básicos de grafos, julgue o item a seguir, relativo ao grafo precedente. No grafo em apreço, existem três ciclos com comprimento quatro: AJBA, BKLB e CDMC.

3. (CESPE/CEBRASPE – 2018 – PF)



Considerando a terminologia e os conceitos básicos de grafos, julgue o item a seguir, relativo ao grafo precedente. O grafo em questão tem diâmetro igual a quatro.

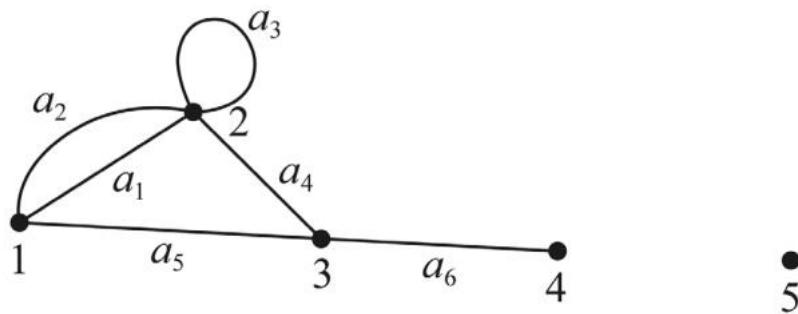
4. (CESPE/CEBRASPE – 2018 – PF)



Considerando a terminologia e os conceitos básicos de grafos, julgue o item a seguir, relativo ao grafo precedente. Os vértices A, B, C, D, J, K, L, M têm graus iguais, respectivamente, a 2, 4, 3, 2, 2, 2, 3, 2.

5. (CESPE/CEBRASPE – 2018 – IFF)



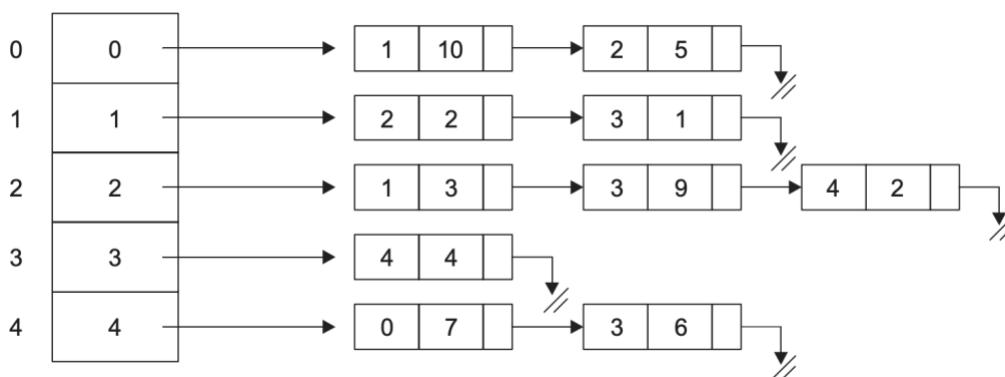


Considerando o grafo precedente, assinale a opção correta.

- a) Os nós 1 e 4 são adjacentes.
  - b) O nó 5 é adjacente a si mesmo.
  - c) Os arcos a1 e a2 são arcos irmãos.
  - d) Os nós 2 e 3 têm grau 3.
  - e) O grafo não pode ser classificado como conexo.
6. (CESPE/CEBRASPE – 2017 – TRE-TO) A estrutura de dados formada por conjuntos de pontos (nós ou vértices) em um conjunto de linhas (arestas e arcos) que conectam vários pontos é denominada
- a) lista encadeada.
  - b) fila circular.
  - c) grafo.
  - d) árvore.
  - e) pilha.
7. (CESPE/CEBRASPE – 2019 – TJ-AM) A respeito de lógica, estrutura e linguagem de programação, julgue o item seguinte.

Na estrutura do tipo grafo, cada elemento indica o próximo elemento, seja aquele que o antecede ou aquele que é seu sucessor, e cada elemento está associado a somente um antecessor e a vários sucessores.

8. (FCC – 2017 – ARTESP) Considere a estrutura de dados abaixo.



Esta estrutura representa cinco localidades indicadas por 0, 1, 2, 3, 4 com as rotas e as respectivas distâncias entre elas.

Por exemplo, da localidade 0 há rota para a localidade 1 (distância 10) e para a localidade 2 (distância 5). Um Especialista em

Tecnologia da Informação da ARTESP afirma, corretamente, que

- a) partindo de qualquer uma das localidades é possível ir para todas as outras e voltar para a localidade de origem.
- b) a distância da rota direta partindo de uma localidade x para uma localidade y não é a mesma da rota de retorno de y para x.



- c) a rota direta mais longa entre duas localidades é 9.  
 d) a rota mais curta partindo da localidade 3 e chegando na localidade 2 é 9.  
 e) é possível ir e voltar de todas as localidades adjacentes.

9. (FCC – 2017 – ARTESP) Nas rodovias paulistas os veículos pagam pedágio em função do número de eixos e da sua categoria. Há 15 categorias de veículos. Para realizar o cálculo do pedágio, existe uma tarifa mínima que é multiplicada por um valor relativo ao número de eixos. Considere a estrutura abaixo que indica a categoria do veículo pelo número da coluna; na primeira linha armazena a quantidade de eixos; na segunda linha armazena o valor pelo qual a tarifa mínima deve ser multiplicada.

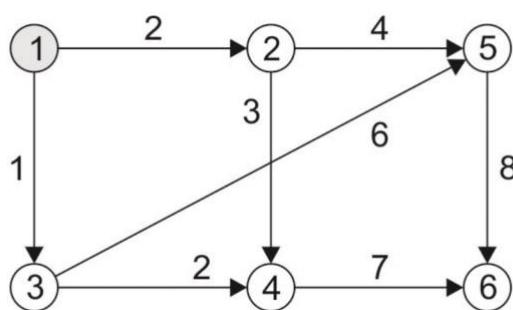
|   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14  |
|---|---|---|---|---|---|---|---|---|---|---|----|----|----|----|-----|
| 0 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 4 | 5 | 6 | 7  | 8  | 9  | 4  | 3   |
| 1 | 0 | 1 | 1 | 2 | 2 | 3 | 3 | 4 | 5 | 6 | 7  | 8  | 9  | 2  | 1,5 |

Exemplos: o veículo 0 é motocicleta/motoneta/bicicleta a motor que tem 2 eixos, mas é isento; o veículo 2 é caminhonete/furgão que tem 2 eixos e paga 1 tarifa; o veículo 13 é um caminhonete/automóvel com reboque que tem 4 eixos e paga 2 tarifas.

Considerando que  $n$  é a categoria do veículo, que  $tm$  é a tarifa mínima e que a estrutura é denominada `mpedagio`, o trecho em pseudocódigo que calcula  $vp$ , o valor pedágio, corretamente, é:

- a)  $vp \leftarrow mpedagio[n,0] * mpedagio[n,1] * tm$
- b)  $vp \leftarrow mpedagio[1,n] * tm$
- c)  $vp \leftarrow vp + (mpedagio[0,n] + mpedagio[1,n]) * tm$
- d)  $vp \leftarrow (mpedagio[n,0]/ mpedagio[n,1]) * tm$
- e) se ( $n = 0$ ) então  $vp \leftarrow 0$  senão  $vp \leftarrow (mpedagio[0,n]/ 2) * tm$  fim se

10. (FCC – 2017 – ARTESP) Considere a estrutura abaixo que representa um problema de rotas em pequena escala.

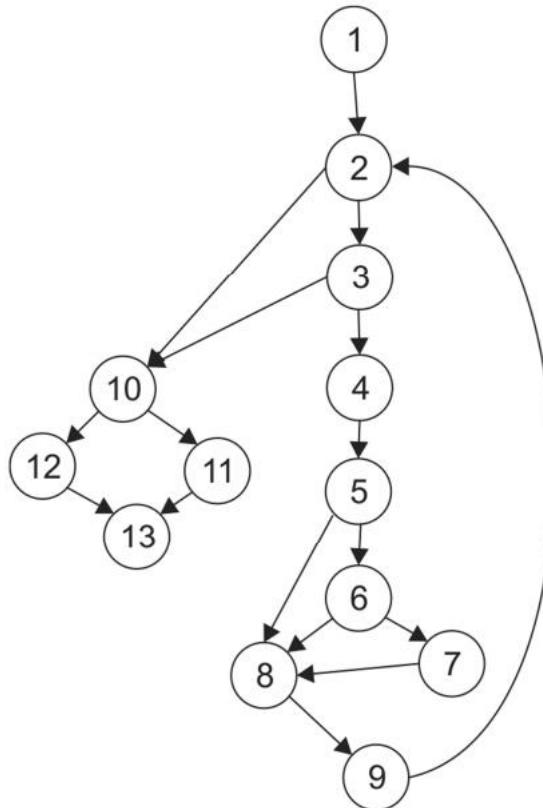


Considere, por hipótese, que solicitou-se a um Agente de Fiscalização à Regulação de Transporte da ARTESP utilizar alguma estratégia lógica para, partindo do ponto 1, chegar ao ponto 6 usando a menor rota. De um mesmo ponto pode haver mais de uma rota, com distâncias diferentes. A lógica correta utilizada pelo Agente, em função dos pontos a serem percorridos, foi

- a) {1} {2,3} {2,4} {5,6}{6}, caminho mais curto 1-2-5-6.
- b) {1} {2} {4} {6}, caminho mais curto 1-2-4-6.
- c) {1} {3,2} {4,5} {6}, caminho mais curto 1-3-4-6.
- d) {6} {5,4} {3,1} {1}, caminho mais curto 6-4-3-1, que é igual a 1-3-4-6.
- e) {6} {4} {5,3} {2,1} {1}, caminho mais curto 6-4-3-5-2-1, que é igual a 1-2-5-3-4-6.



11. (FCC – 2018 – DPE-AM) Considere o grafo abaixo.



A complexidade ciclomática é uma métrica que mede a complexidade de um determinado módulo (uma classe, um método, uma função etc.), a partir da contagem do número de caminhos independentes que ele pode executar até o seu fim. Um caminho independente é aquele que apresenta pelo menos uma nova condição (possibilidade de desvio de fluxo) ou um novo conjunto de comandos a serem executados. O resultado da complexidade ciclomática indica quantos testes, pelo menos, precisam ser executados para que se verifiquem todos os fluxos possíveis que o código pode tomar, a fim de garantir uma completa cobertura de testes.

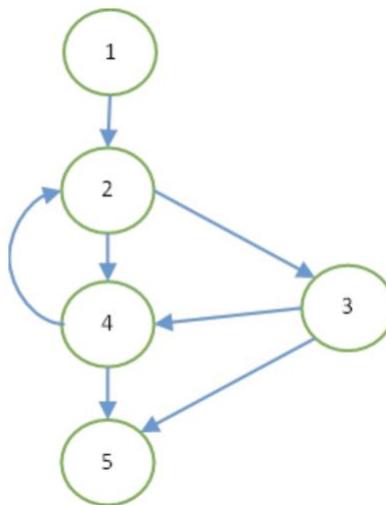
(Adaptado de: <https://www.treinaweb.com.br/blog/complexidade-cicломática-análise-estática-e-refatoração/>)

Considerando que no grafo acima há 17 arestas e 13 nós, o cálculo da complexidade ciclomática resulta em

- a) 6
- b) 4
- c) 7
- d) 20
- e) 18

12. (FGV – 2019 – DPE-RJ) Para que um sistema seja testado adequadamente, é preciso realizar uma quantidade mínima de testes. Para apoiar essa definição, foi criada a Complexidade Ciclomática de McCabe, com fundamentação na teoria dos grafos. Essa técnica define uma métrica de software que fornece uma medida quantitativa da complexidade lógica de um programa, apresentando um limite superior para a quantidade de casos de testes de software que devem ser conduzidos. A Complexidade Ciclomática pode ser calculada tanto pelo número de regiões quanto pelo número de arestas e nós.





Com base no grafo de fluxo acima, correspondente a um trecho de código a ser testado, a quantidade mínima de testes que devem ser realizados para garantir que cada caminho do código tenha sido percorrido em ao menos um teste é:

- a) 11 (onze);
- b) 6 (seis);
- c) 5 (cinco);
- d) 4 (quatro);
- e) 3 (três).

13. (CESPE - 2010 – TJ/ES – Analista de Suporte) Considerando-se a implementação de um grafo denso, direcionado e ponderado, se o número de vértices ao quadrado tem valor próximo ao número de arcos, o uso de uma matriz de adjacência simétrica apresenta vantagens em relação ao uso de uma lista de adjacência.

14. (FCC - 2013 – MPE/SE – Analista do Ministério Público) Considere:

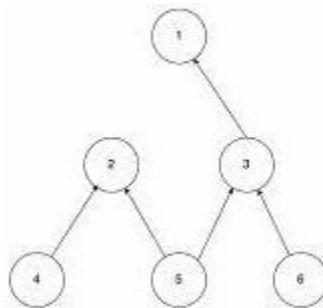
- I. Estrutura de dados que possui uma sequência de células, na qual cada célula contém um objeto de algum tipo e o endereço da célula seguinte.
- II. Podem ser orientados, regulares, completos e bipartidos e possuir ordem, adjacência e grau.
- III. Possuem o método de varredura esquerda-raiz-direita (e-r-d).

Os itens de I a III descrevem, respectivamente,

- a) árvores binárias, listas ligadas e arrays.
- b) arrays, árvores binárias e listas ligadas.
- c) grafos, árvores binárias e arrays.
- d) listas ligadas, grafos e árvores binárias.
- e) grafos, listas ligadas e árvores binárias.

15. (CESPE - 2013 – TCE/ES – Analista de Sistemas) Considerando o grafo ilustrado acima, assinale a opção em que é apresentada a descrição em vértices (V) e arestas (A).





- a)  $V = \{1, 2, 3, 4, 5, 6\}$   
 $A = \{(2, 4), (2, 3), (2, 5), (3, 6), (1, 5)\}$

- b)  $V = \{2, 4, 1, 3, 6, 5\}$   
 $A = \{(4, 2), (1, 3), (5, 2), (6, 3), (5, 3)\}$

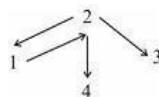
- c)  $V = \{1, 2, 3, 4, 5, 6\}$   
 $A = \{(4, 2), (3, 4), (5, 2), (6, 3), (5, 3)\}$

- d)  $V = \{1, 2, 3, 4, 5, 6\}$   
 $A = \{(4, 2), (3, 1), (5, 1), (6, 2), (5, 3)\}$

- e)  $V = \{2, 4, 1, 3, 6, 5\}$   
 $A = \{(4, 2), (3, 1), (5, 2), (6, 3), (5, 3)\}$

16. (CESPE - 2012 – TJ/RO – Analista de Suporte – ITEM B) Grafo corresponde a uma estrutura abstrata de dados que representa um relacionamento entre pares de objetos e que pode armazenar dados em suas arestas e vértices, ou em ambos.

17. (CESPE - 2012 – PEFOCE – Perito Criminal) Considere que um grafo  $G$  seja constituído por um conjunto ( $N$ ) e por uma relação binária ( $A$ ), tal que  $G = (N, A)$ , em que os elementos de  $N$  são denominados nós (ou vértices) e os elementos de  $A$  são denominados arcos (ou arestas). Em face dessas informações e do grafo abaixo, é correto afirmar que esses conjuntos são  $N = \{1, 2, 3, 4\}$  e  $A = \{(1, 2), (2, 1), (2, 4), (2, 3)\}$ .



18. (CESPE - 2012 – BASA – Analista de Sistemas) É misto o grafo com arestas não dirigidas que representam ruas de dois sentidos e com arestas dirigidas que correspondem a trechos de um único sentido, modelado para representar o mapa de uma cidade cujos vértices sejam os cruzamentos ou finais de ruas e cujas arestas sejam os trechos de ruas sem cruzamentos.

19. (CESPE - 2012 – BASA – Analista de Sistemas) Para modelar a rede que conecta todos os computadores em uma sala de escritório com a menor metragem possível de cabos, é adequado utilizar um grafo  $G$  cujos vértices representem os possíveis pares  $(u, v)$  de computadores e cujas arestas representem o comprimento dos cabos necessários para ligar os computadores  $u$  e  $v$ ,



determinando-se o caminho mínimo, que contenha todos os vértices de  $G$ , a partir de um dado vértice  $v$ .

20. (CESPE - 2012 – BASA – Analista de Sistemas) Um grafo completo contém pelo menos um subgrafo ponderado.

21. (CESPE - 2012 – BASA – Analista de Sistemas) Um grafo não direcionado é dito conectado quando há pelo menos um caminho entre dois vértices quaisquer do grafo.

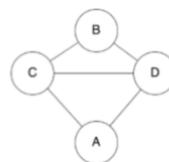
22. (CESPE - 2012 – TJ/AC – Analista de Sistemas) Define-se um grafo como fortemente conexo se todos os nós puderem ser atingidos a partir de qualquer outro nó.

23. (CESPE - 2013 – CRPM – Analista de Sistemas) Considere que o grafo não orientado representado na figura abaixo possua as seguintes características:

$$G_1 = (V_1, A_1)$$

$$V_1 = \{A, B, C, D\}$$

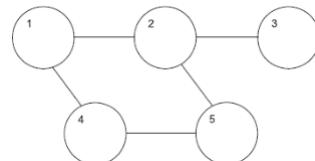
$$A_1 = \{(A, C), (A, D), (B, C), (B, D), (A, B)\}.$$



Nesse caso, é correto afirmar que o grafo  $G_1$  possui quatro vértices, nomeados de  $A$ ,  $B$ ,  $C$  e  $D$ , e cinco arcos, que conectam pares de vértices, conforme especificado em  $A_1$ .

24. (CESPE - 2012 – BASA – Analista de Sistemas) A implementação de um grafo do tipo ponderado e direcionado na forma de uma matriz de adjacência utiliza menor quantidade de memória que a implementação desse mesmo grafo na forma de uma lista encadeada.

25. (CESGRANRIO - 2014 – BASA – Analista de Sistemas) O grafo acima pode ser representado pela seguinte matriz:



a)

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

b)

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$



c)

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

d)

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

e)

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

26. (CESPE - 2012 – TJ/SE – Analista de Sistemas) Um grafo é formado por um par de conjuntos de vértices e arestas, não podendo o conjunto de vértices ser particionado em subconjuntos.

27. (CESPE - 2012 – TRT/AM – Analista de Sistemas) Um grafo é uma estrutura de dados consistida em um conjunto de nós (ou vértices) e um conjunto de arcos (ou arestas). O grafo em que os arcos possuem um número ou peso associados a eles, é chamado de grafo:

- a) predecessor.
- b) adjacente.
- c) incidente.
- d) ponderado.
- e) orientado.



## GABARITO



- |            |             |             |
|------------|-------------|-------------|
| 1. Correto | 10. Letra C | 19. Errado  |
| 2. Errado  | 11. Letra A | 20. Errado  |
| 3. Correto | 12. Letra D | 21. Correto |
| 4. Correto | 13. Errado  | 22. Correto |
| 5. Letra E | 14. Letra D | 23. Errado  |
| 6. Letra C | 15. Letra E | 24. Errado  |
| 7. Errado  | 16. Correto | 25. Letra A |
| 8. Letra B | 17. Correto | 26. Errado  |
| 9. Letra B | 18. Correto | 27. Letra D |



## LISTA DE QUESTÕES - HASHING - MULTIBANCAS

1. (CESPE - 2010 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação - Administração de Dados) A pesquisa sequencial é aplicável em estruturas não ordenadas.
2. (CESPE - 2012 - Banco da Amazônia - Técnico Científico - Administração de Dados) As colisões ocorrem na utilização de tabela hash porque várias chaves podem resultar na mesma posição.
3. (CESPE - 2010 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação - Administração de Dados) Ocorre o hashing quando não há o armazenamento de cada entrada de uma tabela em um específico endereço calculado a partir da aplicação de uma função chave da entrada.
4. (FCC - 2008 - METRÔ-SP - Analista Treinee - Análise de Sistemas) O objetivo de fazer uma busca rápida a partir de uma chave de pesquisa simples e obter o valor desejado é alcançado pela estrutura de dados especial denominada:
  - a) array.
  - b) lista.
  - c) vetor.
  - d) árvore binária.
  - e) tabela de hashing.
5. (CESPE - 2012 - Banco da Amazônia - Técnico Científico - Administração de Dados) A busca que utiliza uma tabela hash realiza comparação das chaves para encontrar a posição do elemento que está sendo buscado.
6. (CESPE - 2010 - DETRAN-ES - Analista de Sistemas) No método de hashing, por meio de acesso sequencial, são utilizados tabelas e mapas para recuperar informações de endereço de arquivos de forma rápida e eficiente.
7. (FCC - 2015 - DPE-SP - Programador) Um Programador da Defensoria Pública do Estado de São Paulo foi solicitado a propor uma solução para o problema: Há uma quantidade grande de dados classificáveis por chave e estes dados devem ser divididos em subconjuntos com base em alguma característica das chaves. Um método eficiente deve ser capaz de localizar em qual subconjunto deve-se colocar cada chave e depois estes subconjuntos bem menores devem ser gerenciados por algum método simples de busca para que se localize uma chave



rapidamente. O Programador propôs como solução, corretamente, a implementação de:

- a) Deques.
- b) Tabela e função hash.
- c) Pilhas.
- d) Fila duplamente encadeada.
- e) Árvore Binária de Busca.



## GABARITO



- 1. C
- 2. C
- 3. E

- 4. E
- 5. E
- 6. E

- 7. B



## LISTA DE QUESTÕES - BITMAP - MULTIBANCAS

1. (FGV - 2015 - TJ-RO - Analista Judiciário - Análise de Sistemas)

| ID   | Nome | Curso      |
|------|------|------------|
| 1210 | A    | Física     |
| 356  | B    | Química    |
| 23   | C    | Matemática |
| 57   | D    | Física     |
| 45   | E    | Física     |
| 6    | F    | Matemática |
| 210  | G    | Matemática |

Se fosse construído um índice de banco de dados do tipo "bitmap" para essa tabela, tendo o campo Curso como chave, o conteúdo desse índice seria:

a)

6           3  
23          3  
45          1  
57          1  
210         3  
356         2  
1210       1

b)

Física       1001100  
Química    0100000  
Matemática  0010011

c)

6  001  
23 001  
45 100  
57 100

|      |     |
|------|-----|
| 210  | 001 |
| 356  | 010 |
| 1210 | 100 |

d)

|            |
|------------|
| 0011001100 |
| 0100100000 |
| 0010010011 |

e)

|             |            |
|-------------|------------|
| 00000000110 | Matemática |
| 00000010111 | Matemática |
| 00000101101 | Física     |
| 00000111011 | Física     |
| 00011010010 | Matemática |
| 00101100100 | Química    |
| 10010111010 | Química    |

2. (FGV - 2014 - DPE-RJ - Técnico Superior Especializado - Administração de Dados)



| Candidato |               |
|-----------|---------------|
| inscrição | candidatoNome |
| 101       | João          |
| 102       | Maria         |
| 105       | Gabriela      |
| 106       | João          |

| Prova      |            |
|------------|------------|
| provaNome  | data       |
| Português  | 12/01/2014 |
| Matemática | 12/01/2014 |
| Prática    | 19/01/2014 |

| Avaliação |            |      |
|-----------|------------|------|
| inscrição | provaNome  | nota |
| 101       | Português  | 10   |
| 102       | Português  | 8    |
| 105       | Português  | 3    |
| 106       | Português  | 5    |
| 101       | Matemática | 7    |
| 102       | Matemática | 4    |
| 105       | Matemática | 8    |
| 101       | Prática    | 5    |
| 102       | Prática    | 5    |
| 105       | Prática    | NULL |
| 106       | Prática    | 4    |

Índices do tipo Bitmap podem ser usados em tabelas de bancos de dados. O quadro abaixo representa o mapa de bits na indexação da tabela Avaliação quando a chave considerada é a concatenação dos atributos Inscrição e provaNome.

|               |              |
|---------------|--------------|
| 101Matemática | 00001000000  |
| 101Português  | 100000000000 |
| 101Prática    | 00000001000  |
| 102Matemática | 00000100000  |
| 102Português  | 010000000000 |
| 102Prática    | 00000000100  |
| 105Matemática | 00000010000  |
| 105Português  | 001000000000 |
| 105Prática    | 00000000010  |
| 106Português  | 000100000000 |
| 106Prática    | 00000000001  |



Num mapa de bits para a mesma tabela, usando apenas o atributo Inscrição, o mapa de bits seria

a)

101 10001001000  
102 01000100100  
105 001000100010  
106 00010000001

b)

101 1000  
102 0100  
105 0010  
106 0001

c)

101 10000000000  
102 01000100000  
105 00100000000  
106 00010000000

d)

101 00000001000  
102 00000000100  
105 00000000010  
106 00000000001

e)

101 111  
102 111  
105 111  
106 011



# GABARITO



1. B
2. A



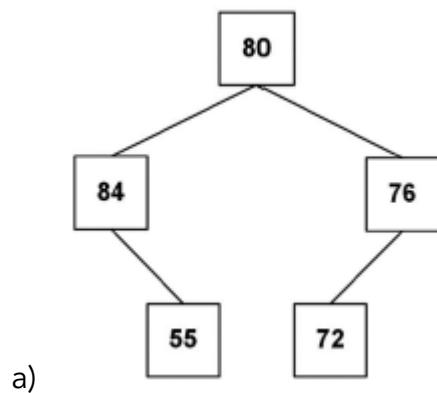
## LISTA DE QUESTÕES

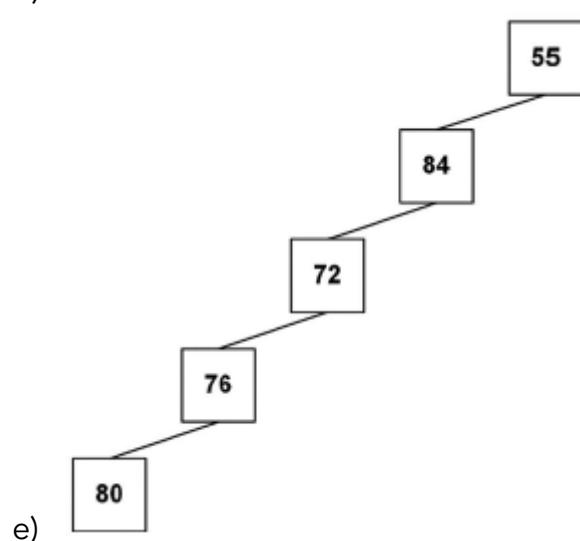
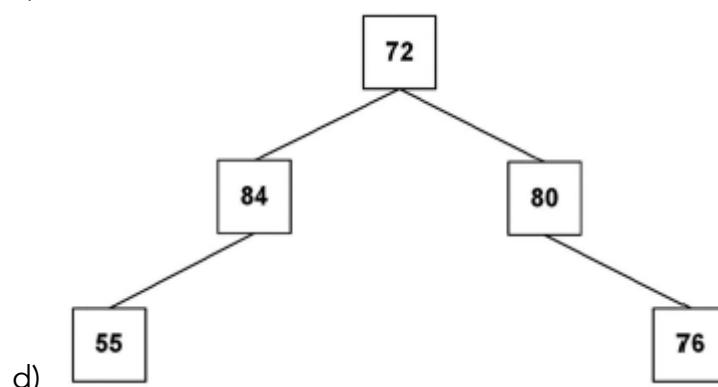
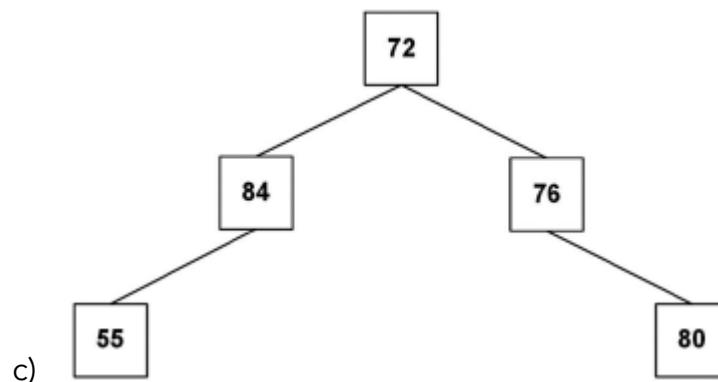
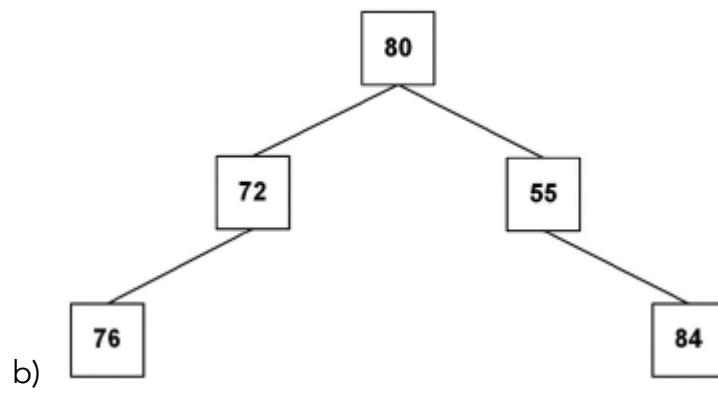
### Cesgranrio

1. (CESGRANRIO–TRANSPETRO– 2023) Estruturas de dados referem-se aos diferentes mecanismos de organização de dados para atender a diferentes requisitos de processamento. Dentre as estruturas de dados, é eficiente para inserção e remoção de elementos em qualquer posição, incluindo início, meio e fim, além de oferecer acesso aos elementos em posições intermediárias, a seguinte estrutura de dados:
  - a) pilha
  - b) fila
  - c) lista encadeada
  - d) array estático
  - e) vetor dinâmico
2. (CESGRANRIO– Esc BB– 2023) Um estudante de computação decidiu escrever um método Java para exibir, no console, em pré-ordem, os valores dos nós de uma árvore binária recebida como parâmetro. Ao executar esse método, os seguintes valores foram exibidos no console:

80 84 55 76 72

Considerando os valores exibidos, qual árvore foi recebida como parâmetro?





3. (CESGRANRIO–AgeRIO– 2023) As classes Java a seguir são usadas na implementação de árvores binárias.

```
public class ArvNo {
 int info;
 ArvNo esq=null,dir=null;
}

public class Arv {
 private ArvNo raiz;

 public Arv(){
 }

 private void percorre(ArvNo r) {
 if(r==null)
 return;

 percorre(r.dir);
 percorre(r.esq);
 System.out.printf("%d ", r.info);
 }

 public void exibeArvore() {
 percorre(raiz);
 }
}
```

A classe Main abaixo faz uso da classe Arv.



```
public class Main {

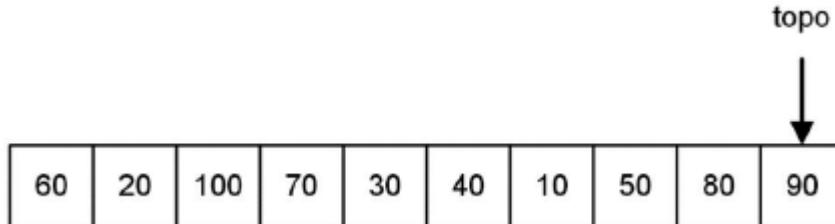
 public static void main(String[] args) {
 Stack<Integer> p;
 Arv a;

 // Comandos relativos à criação de uma pilha
 // e de uma árvore binária.
 //
 // Esses comandos são irrelevantes para a
 // resolução da questão.

 percorrePilha(p);
 System.out.println();
 a.exibeArvore();
 }

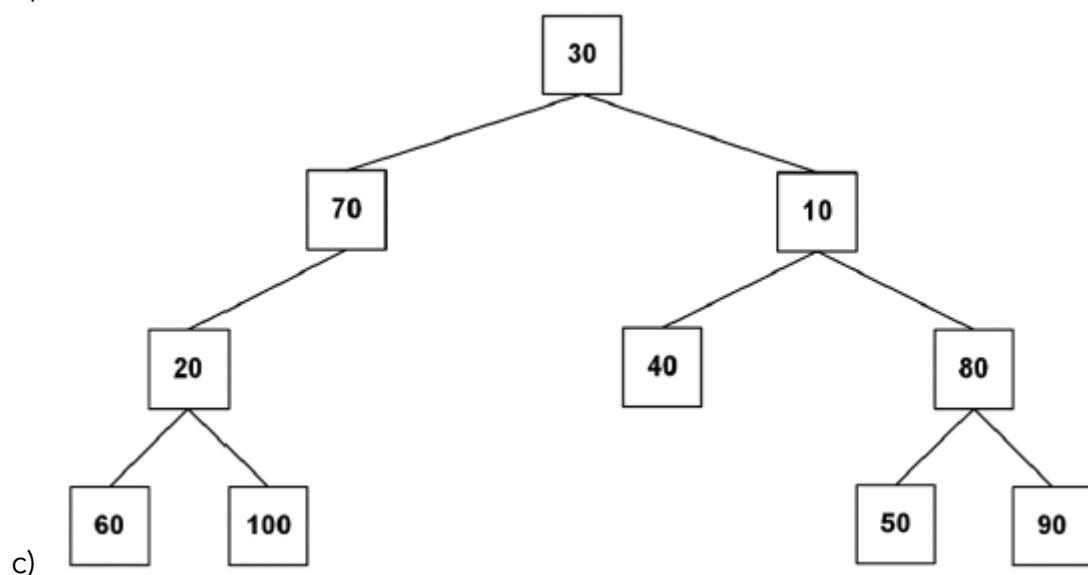
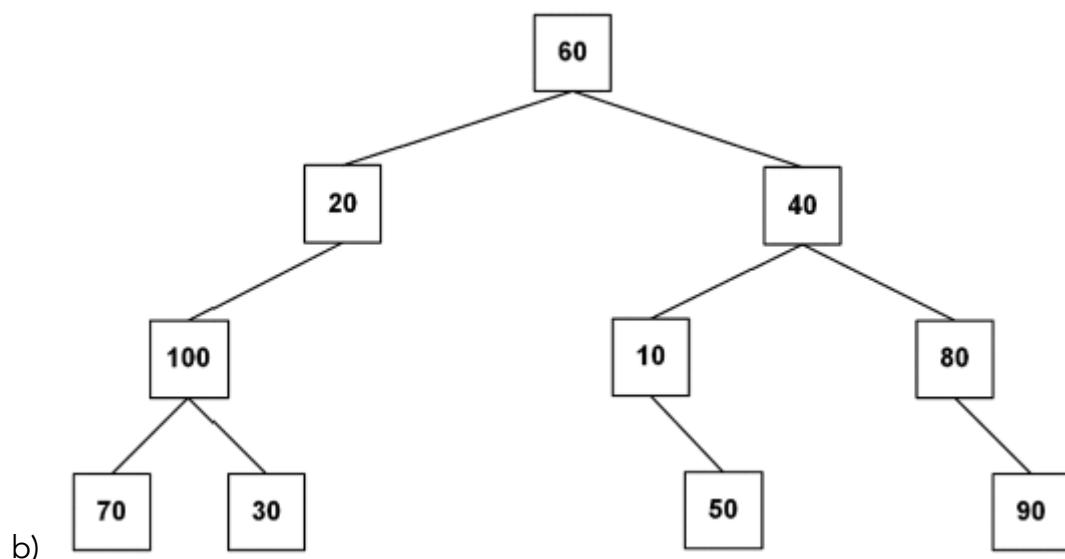
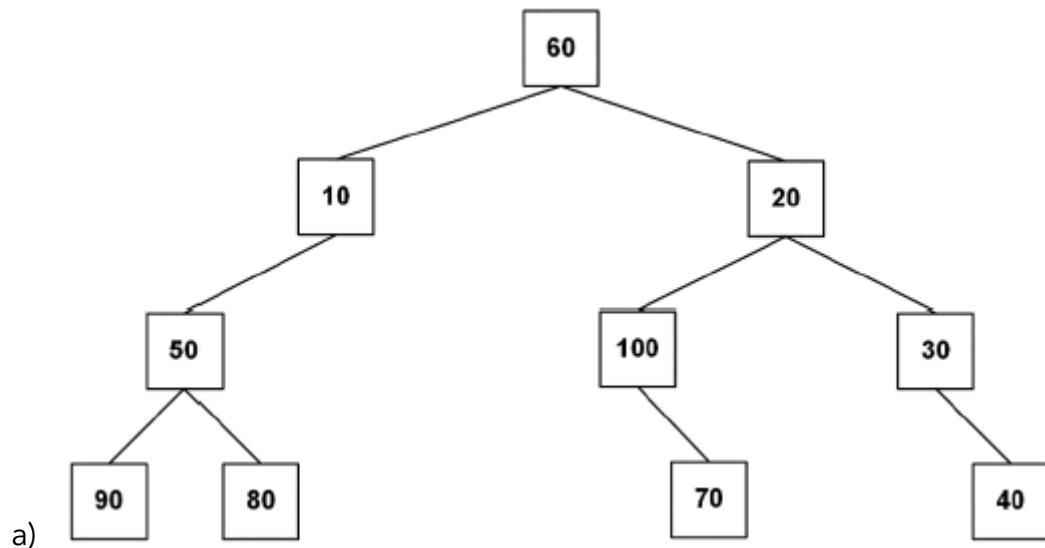
 public static void percorrePilha(Stack<Integer> p) {
 while(!p.isEmpty())
 System.out.printf("%d ", p.pop());
 }
}
```

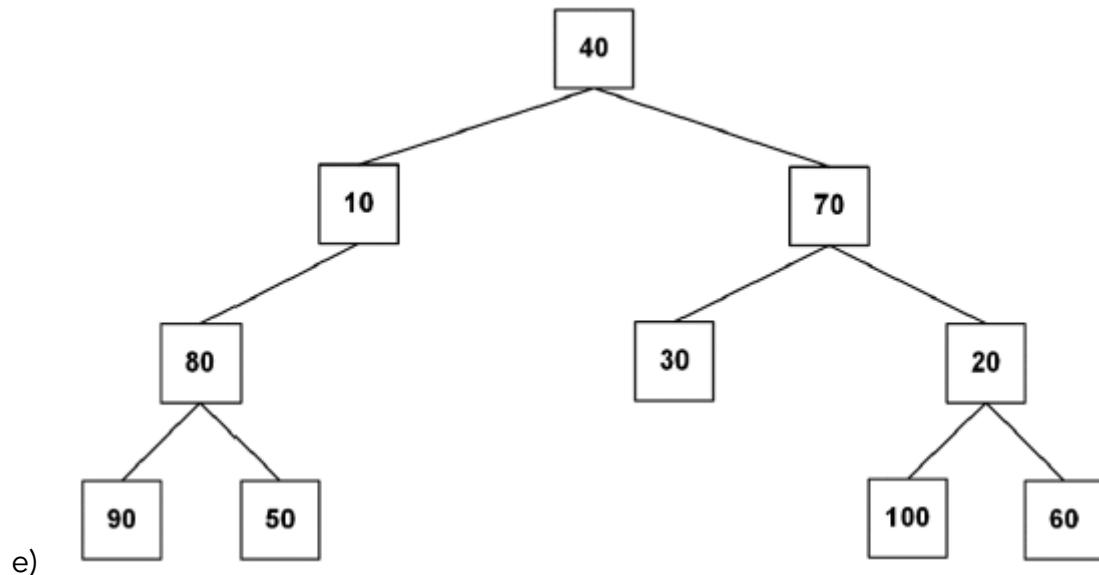
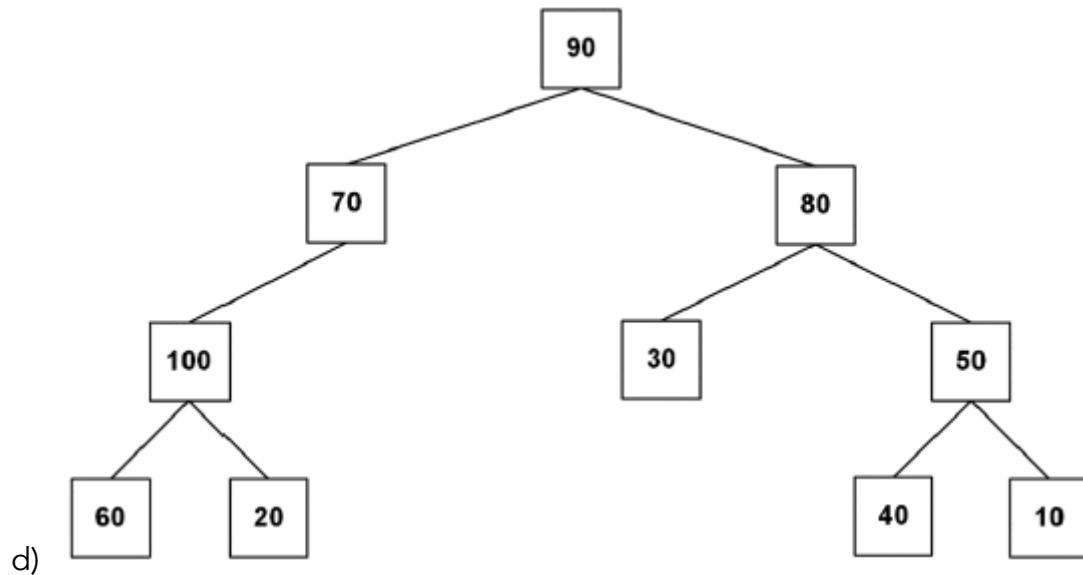
Admita que o método main acima vá ser executado, e que uma pilha como a mostrada na Figura a seguir vá ser passada como parâmetro para o método percorrePilha.



Qual árvore binária fará com que o comando a.exibeArvore() exiba no console os mesmos números inteiros, e na mesma ordem, que o método percorrePilha?





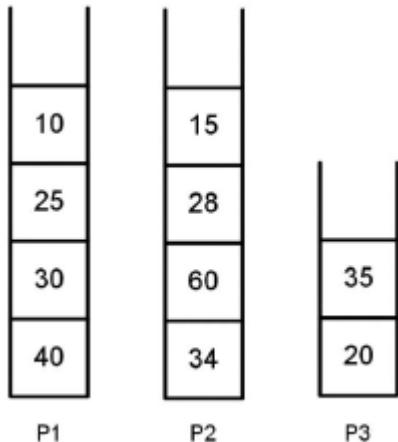


4. (CESGRANRIO–TRANSPETRO– 2023) Considere uma árvore AVL que possui 12 nós. A altura dessa árvore é

- a) 3
- b) 4
- c) 5
- d) 6
- e) 7.

5. (CESGRANRIO–Esc BB– 2023) A Figura a seguir exibe o conteúdo de três pilhas: P1, P2 e P3.





Admita que um método Java, chamado exibePilha, receba essas três pilhas como parâmetros e execute os seguintes passos:

1. Cria duas pilhas auxiliares, A1 e A2, inicialmente vazias;
2. Remove um elemento de P1 e o insere em A1. Em seguida, remove um elemento de P2 e o insere em A1. Repete esses dois procedimentos até que P1 e P2 fiquem, ambas, vazias;
3. Remove um elemento de P3 e o insere em A1. Repete esse procedimento até que P3 fique vazia;
4. Remove um elemento de A1 e o insere em A2. Repete esse procedimento até que A1 fique vazia;
5. Remove um elemento de A2 e o exibe no console. Repete esse procedimento 4 vezes.

O que será exibido no console, quando o método exibePilha for executado, tendo P1, P2 e P3 sido passadas como parâmetros?

- a) 10 15 25 28
- b) 10 25 30 40
- c) 15 10 28 25
- d) 20 35 34 40
- e) 40 34 30 60

6. (CESGRANRIO-PNS– 2022) Seja uma função que realiza uma busca binária sobre um array de números inteiros ordenados. Não se sabe, em princípio, se os números estão ordenados ascendente ou descendente. O cabeçalho dessa função é o seguinte:

```
int busca (int [] vet, int elem)
```

Isto é, a função busca recebe um array de números inteiros (vet) e um número inteiro (elem) como parâmetros, e retorna um número inteiro. Caso exista em vet um inteiro igual a elem, a função retornará o índice desse inteiro no array; caso contrário, a função retornará -1.

O algoritmo de busca binária produz um índice (ind) a cada iteração sobre o array, tendo em vista comparar o elemento que se deseja procurar (elem) com o elemento vet [ ind ]. Isto é:

```
if (vet [ind] == elem)
```



```
return ind;
```

No comando acima, diz-se que houve uma visita ao elemento vet [ ind ].

Admita que a função busca foi chamada por meio do comando a seguir:

```
int resp = busca (vet, 50);
```

Sabendo-se que os elementos visitados foram 54, 17, 33 e 50, nesta ordem, qual array foi passado como parâmetro para a função busca?

- a) [ 95, 90, 87, 54, 52, 50, 33, 17, 11, 10 ]
- b) [ 5, 10, 11, 17, 33, 50, 54, 87, 90, 95 ]
- c) [ 121, 111, 93, 87, 60, 54, 50, 33, 17, 5 ]
- d) [ 5, 17, 33, 50, 54, 60, 87, 93, 111, 121 ]
- e) [ 130, 121, 111, 90, 70, 60, 54, 50, 33, 17 ]

7. (CESGRANRIO–BASA– 2022) Uma função, chamada converte, tem por objetivo converter um número inteiro na base decimal (d), recebido como parâmetro, em um número inteiro na base binária (b), isto é, um número que seja formado apenas pelos algarismos 0 e 1, como nos exemplos abaixo.

Exemplos:

```
converte(7) = 111
converte(12) = 1100
converte(16) = 10000
```

Admita que o inteiro (d), recebido como parâmetro, é tal que  $d \geq 0$  e  $d \leq 1024$ .

Qual função executa essa conversão corretamente?

- a) static long converte(int dec) {  
    if(dec == 0)  
        return dec;  
    long r=converte(dec/2);  
    return dec % 2 + r \* 10;  
}
- b) static long converte(int dec) {  
    long bin=0;  
    while(dec > 0) {  
        int r= dec % 2;  
        dec=dec / 2;  
        bin+=r << 2;  
    }  
    return bin;  
}
- c) static long converte(int dec) {  
    if(dec > 0)



```

 return dec;
 long r=converte(dec%2);
 return dec/ 2 + r * 10;
}
d) static long converte(int dec) {
 long bin=0,fat=1;
 do {
 int r=dec % 2;
 dec=dec/2;
 bin+=r * fat;
 fat*=10;
 } while(dec >= 0);
 return bin;
}
e) static long converte(int dec) {
 long bin=0;
 for(long fat=1; dec > 0; fat*=10) {
 int r=dec % 2;
 dec=dec / 2;
 bin+=r * fat;

 }
 return bin;
}

```

8. (CESGRANRIO–BASA– 2022) Em linguagens de programação como Java, onde existem estruturas de repetição, a recursão pode ser muitas vezes substituída pela repetição, com ganhos de desempenho.

Considere a seguinte função recursiva segredo, em Java:

```

public static int segredo(int a) {
 if (a<2) {
 return 0;
 } else {
 return segredo(a-2)+1;
 }
}

```

Que fragmento de código, em Java, contendo uma estrutura de repetição, é adequado para substituí-la?

- a) public static
- ```

int alternativaA(int a) { int s = 0;
for (int i=a;i>2;i--) {
    s++;
}
return s;
}

```



```

}

b) public static int alternativaB(int a) {
    int s = 0;
    for (int i=a;i<2 && i>0;i--) {
        s++;
    }
    return s;
}

c) public static int alternativaC(int a) {
    int s = 0;

    while (a>=2) {
        a-=2;
        s++;
    }
    return s;
}

d) public static int alternativaD(int a) {
    int s = 0;
    do {
        a-=2;
        s++;
    } while (a>0 && a<=2);
    return s;
}

e) public static int alternativaE(int a) {
    int s = 0;
    do {
        a-=2;
        s++;
    } while (a>0 && a<2);
    return s;
}

```

9. (CESGRANRIO–PNS– 2022) Seja um array composto por 7 números inteiros.

[5, 15, 77, 21, 5, 25, 2]

Esse array foi usado por um profissional de teste de software para testar uma função que ordena, de forma ascendente, um array de números inteiros. Essa função implementa o algoritmo de ordenação por seleção.

Para avaliar a evolução do array sendo ordenado, o profissional de teste solicitou ao programador que criou a função de ordenação que fizesse uma modificação no código, de modo que o somatório dos elementos do array com índices 2, 3 e 4 seja exibido no console imediatamente antes do incremento da variável (i) que controla a execução do comando de repetição mais externo.

Feitas as modificações solicitadas, o código da função passou a ter a seguinte forma geral:



```
ordena (int vetor[ ]) {  
  
    int i = 0;  
  
    int tam = length (vetor); // comentário: a função length retorna a quantidade de elementos de  
    // um array  
  
    while ( i < tam ) {  
  
        while () {  
  
            // comentário: isso é apenas a forma geral do algoritmo de ordenação  
  
            // não é o código completo  
  
        }  
  
        print ( vetor[2] + vetor[3] + vetor[4] );  
  
        i = i + 1;  
    }  
}
```

O que será exibido no console pelo comando print na 3ª iteração do comando de repetição mais externo?

- a) 32
- b) 38
- c) 41
- d) 103
- e) 113

10. (CESGRANRIO – PNS- 2022) Os tipos abstratos de dados (TAD) Fila e Pilha foram implementados em uma linguagem orientada a objetos por meio de um array de inteiros. As classes criadas para a implementação desses TADs contêm os seguintes métodos:

Classe Pilha

int pop () – retorna o número inteiro retirado da pilha.

push (int x) – insere o número inteiro x na pilha.

Classe Fila

int deq () – retorna o número inteiro retirado da fila.

enq (int x) – insere o número inteiro x na fila.



Admita que o estado inicial de uma pilha (p) seja um array vazio, e que o estado inicial de uma fila (f) seja caracterizado pelo array [15, 90, 40].

Após a execução de uma sequência de operações sobre p e f, a fila assumiu o seguinte estado final: [35, 90, 40, 15]

Qual sequência de comandos levou f do estado inicial para o estado final?

a) p.push (f.deq())

f.enq (f.deq ())

f.enq (f.deq ())

p.push (35)

f.enq (p.pop())

f.enq (p.pop())

b) p.push (f.deq())

p.push (f.deq ())

f.enq (35)

f.enq (p.pop ())

f.enq (f.deq ())

f.enq (p.pop ())

c) p.push (f.deq())

p.push (f.deq ())

f.enq (p.pop ())

f.enq (35)

f.enq (f.deq ())

f.enq (p.pop ())

d) p.push (f.deq())

p.push (35)

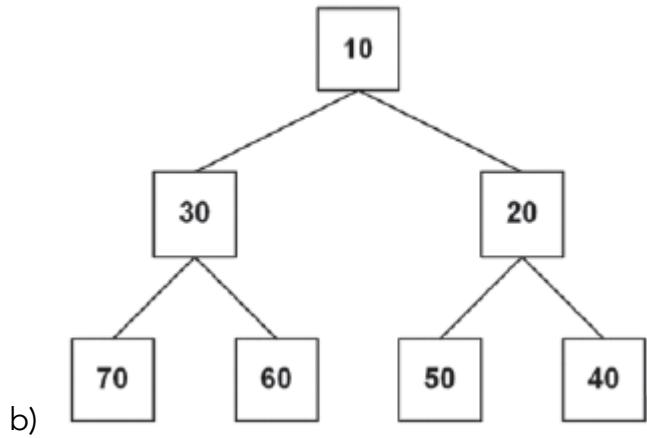
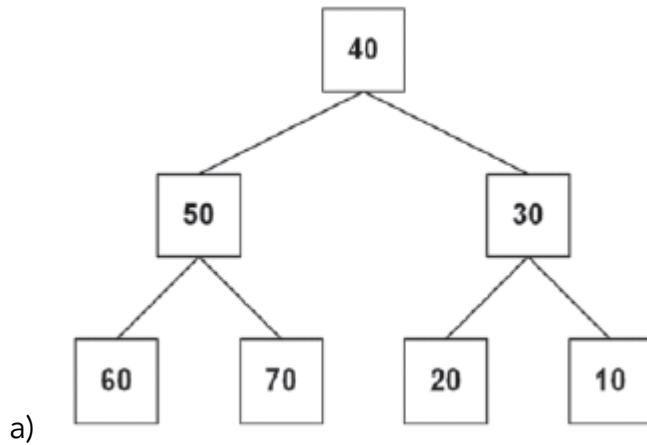


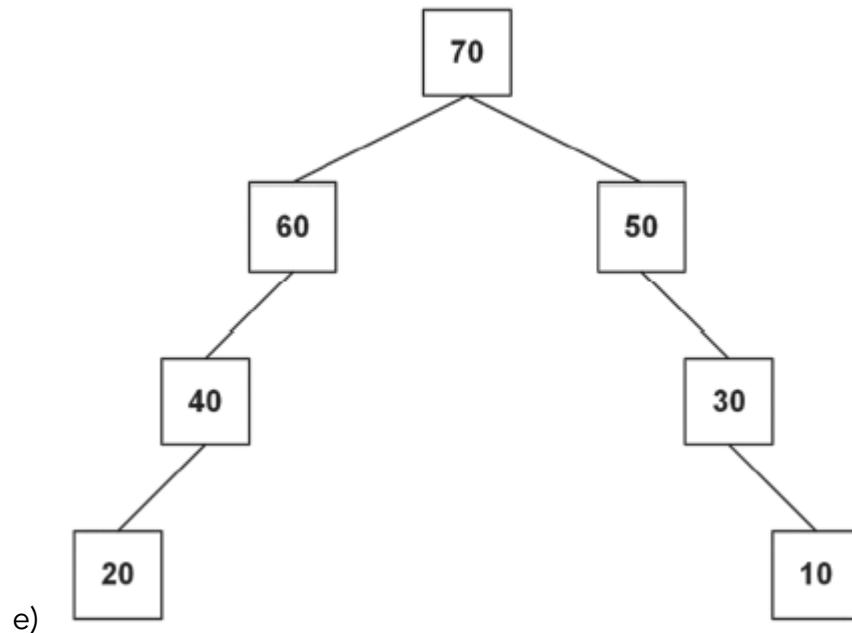
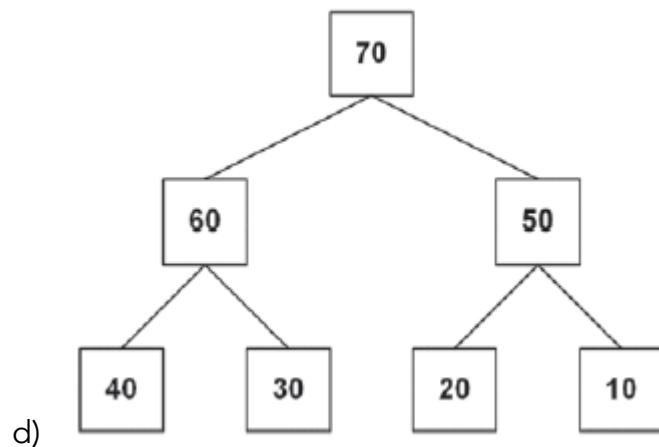
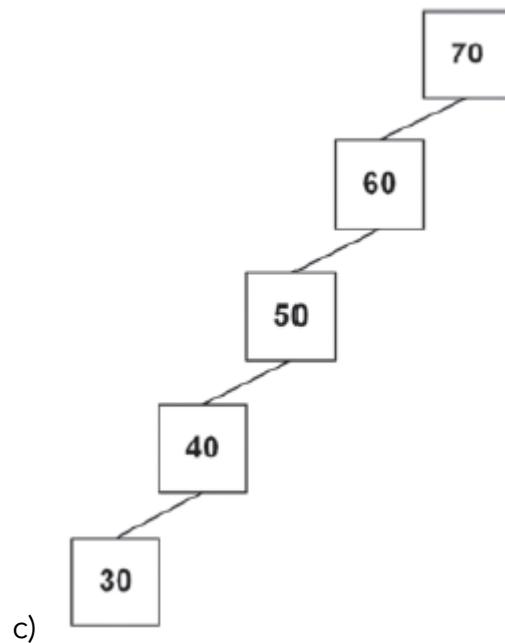
```

f.enq (f.deq ( ))
f.enq ( p.pop() )
f.enq ( p.pop() )
e) p.push ( f.deq() );
p.push (f.deq ( ));
f.enq ( 35 );
f.enq ( p.pop () );
f.enq (p.pop () );
f.enq (f.deq ( ));

```

11. (CESGRANRIO CEF– 2021) Qual árvore binária pode ser classificada como árvore binária de busca?





12. (CESGRANRIO –BB – 2021) Em um determinado treinamento de pessoal de TI, para facilitar o aprendizado sobre o funcionamento da estrutura de dados PILHA, utilizou-se o jogo de trocas, cujas regras são apresentadas a seguir.

JOGO DAS TROCAS - REGRAS

Para começar o jogo, o jogador recebe duas pilhas, P1 e P2.

P1 está preenchida com quatro fichas, identificadas por nomes fictícios e empilhadas em ordem alfabética CRESCENTE a partir do topo. P2 está inicialmente vazia.

Uma ficha desempilhada de P1 é imediatamente empilhada em P2.

A operação (P2.pop) acarreta impressão do nome que está na ficha desempilhada e descarte da ficha.

Para ganhar o jogo, o jogador precisa determinar corretamente, dentre sequências derivadas da sequência inicial, por troca da posição de seus elementos, qual delas poderia ser impressa com essas operações.

No início do jogo, foram dadas as pilhas P2, vazia, e P1 preenchida com as seguintes operações de empilhamento: push(P1,Zeus); push(P1,Hades); push(P1,Cibele); push(P1, Apolo).

Considerando-se esse cenário, qual seria a sequência possível de ser impressa, da esquerda para a direita, de acordo com as regras do JOGO DAS TROCAS?

- a) Apolo, Zeus, Cibele, Hades
- b) Hades, Apolo, Zeus, Cibele
- c) Zeus, Cibele, Apolo, Hades
- d) Hades, Apolo, Cibele, Zeus
- e) Cibele, Hades, Apolo, Zeus

13. (CESGRANRIO– BB- 2021) Uma das formas de o gerente de uma agência bancária acompanhar a qualidade dos serviços prestados aos seus clientes é verificar o estado da ordem de atendimento em vários instantes ao longo do expediente. O sistema que a gerência utiliza para tal fim é a estrutura de dados conhecida como FILA, que mostra a situação da ordem de atendimento no instante da verificação.

Nesse contexto, implementa-se uma estrutura de FILA de números inteiros com suas duas operações tradicionais: ENFILEIRAR(Z), que ocorre no instante em que um cliente recebe uma senha Z e entra na FILA; e DESENFILEIRAR(), que ocorre quando um cliente sai da FILA, caso em que DESENFILEIRAR() retorna o número da senha. Sabe-se, também, que a representação do estado da FILA em um instante qualquer é realizada listando os elementos, de forma que o primeiro elemento, da esquerda para a direita, é o mais antigo presente na FILA.



Nas condições apresentadas, considere uma FILA que começa vazia e realiza as seguintes operações:

ENFILEIRAR(8) → ENFILEIRAR(9) → DESENFILEIRAR() → ENFILEIRAR(10) → ENFILEIRAR(11)
→

ENFILEIRAR(DESENFILEIRAR ()) → ENFILEIRAR(12) → DESENFILEIRAR() → ENFILEIRAR(13)
→ DESENFILEIRAR()

Após realizar as operações acima, a FILA estará no estado

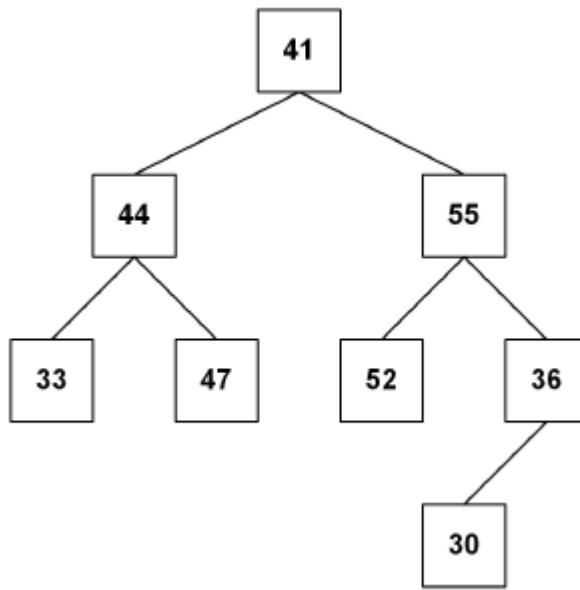
- a) 10 – 11 – 12
- b) 9 – 12 – 13
- c) 9 – 10 – 11
- d) 8 – 10 – 11
- e) 8 – 9 – 10

14. (CESGRANRIO –BB – 2021) Um programador escreveu uma função para percorrer, em pós-ordem, uma árvore binária e exibir, no console, os valores referentes aos nós dessa árvore.

Após essa função ter sido executada, foi exibido o seguinte resultado:

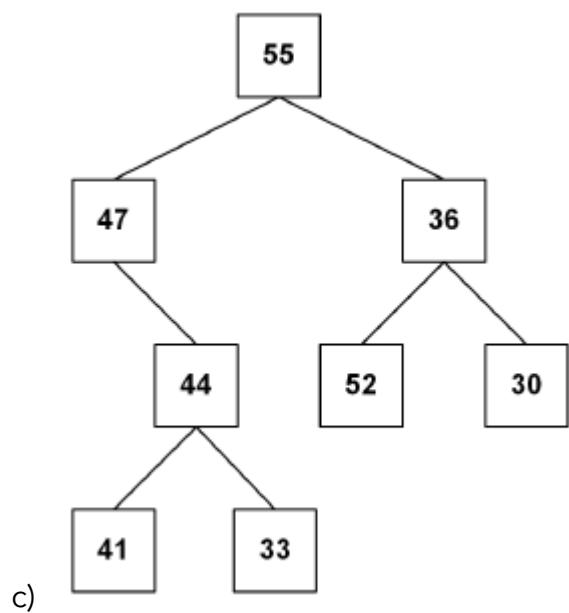
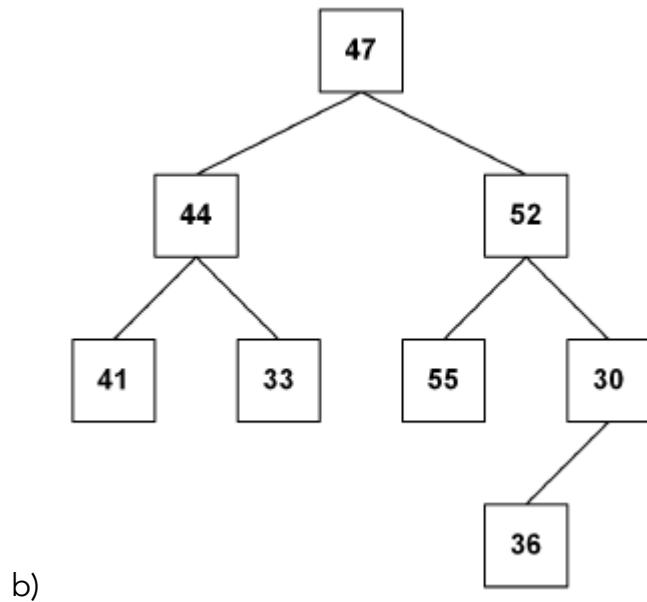
41 44 33 47 55 52 36 30

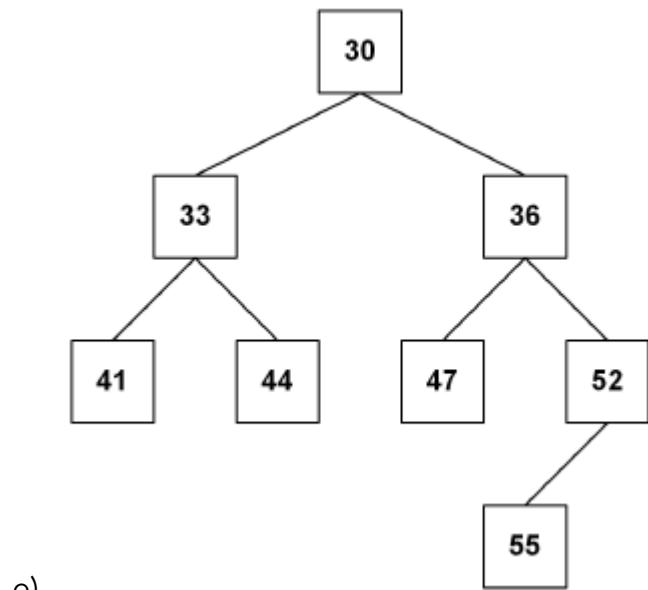
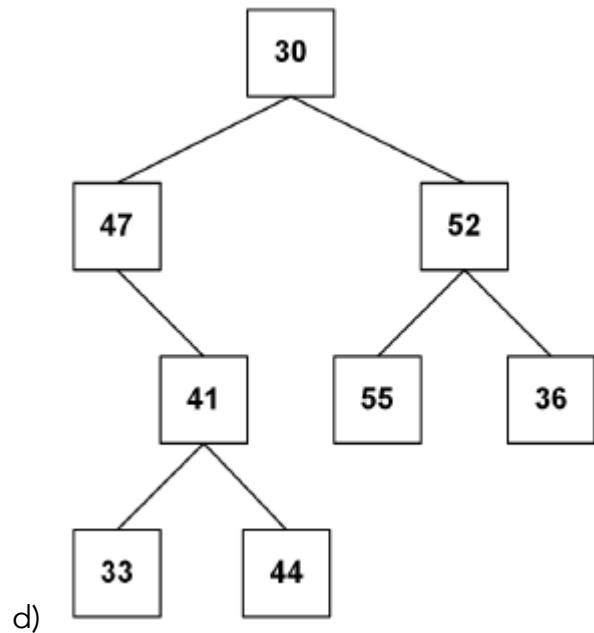
Que árvore essa função percorreu para exibir o resultado acima?



a)







15. (CESGRANRIO– BB- 2021) Desejam-se realizar buscas nas seguintes coleções de dados, representadas na linguagem Java:

- I - Um array de 1.000 números inteiros ordenados de forma decrescente;
- II - Uma lista encadeada desordenada e alocada dinamicamente, cujos 1.000 nós contêm strings(uma string por nó);
- III - Uma lista encadeada, alocada dinamicamente, cujos 1.000 nós contêm números decimais (um número double por nó) ordenados de forma ascendente.

Levando-se em consideração a exequibilidade e a eficiência, quais métodos de busca devem ser empregados, respectivamente, em cada um dos três casos acima?

- a) I – sequencial; II – sequencial; III – binária



- b) I – binária; II – sequencial; III – sequencial
- c) I – binária; II – sequencial; III – binária
- d) I – sequencial; II – sequencial; III – sequencial
- e) I – sequencial; II – binária; III – binária

16.(CESGRANRIO –BB – 2021) Um professor preparou uma série de experimentos para avaliar, juntamente com seus alunos, três algoritmos de ordenação: o da bolha, o de ordenação por inserção e o de ordenação por seleção. Para tal, ele escreveu três métodos Java, um para cada algoritmo. Todos eles recebem como único parâmetro um array de inteiros (`int vet[] = {81,15,4,20,7,47,14,20,4}`), que será ordenado em ordem crescente.

Para acompanhar a evolução desse array sendo ordenado, cada um dos três métodos exibe a configuração dos elementos do array ao término de cada iteração do comando de repetição mais externo. Vale lembrar que esses três algoritmos de ordenação são compostos por dois comandos de repetição aninhados (dois comandos `for` ou dois comandos `while`).

Terminada a codificação, o professor executou os métodos relativos aos três algoritmos de ordenação e projetou no quadro as configurações do array relativas às três primeiras iterações de cada um dos algoritmos de ordenação, conforme mostrado a seguir.

Algoritmo 1

```
4 15 81 20 7 47 14 20 4
4 4 81 20 7 47 14 20 15
4 4 7 20 81 47 14 20 15
```

Algoritmo 2

```
15 81 4 20 7 47 14 20 4
4 15 81 20 7 47 14 20 4
4 15 20 81 7 47 14 20 4
```

Algoritmo 3

```
15 4 20 7 47 14 20 4 81
4 15 7 20 14 20 4 47 81
4 7 15 14 20 4 20 47 81|
```

As configurações 1, 2 e 3, exibidas acima, correspondem, respectivamente, aos algoritmos

- a) da bolha, de seleção e de inserção
- b) da bolha, de inserção e de seleção
- c) de seleção, de inserção e da bolha
- d) de seleção, da bolha e de inserção
- e) de inserção, de seleção e da bolha

17.(CESGRANRIO –BB – 2021) Considere o código Python a seguir.

O gerente de uma agência bancária recebe, diariamente, solicitações de seus clientes com dúvidas sobre a melhor decisão para aplicações financeiras e as armazena, com um código numérico crescente, num vetor de solicitações, para respondê-las ao final do expediente. Para



manter o conceito de bom atendimento, o gerente gostaria, sempre que possível, que a ordem das respostas seguisse, estritamente, a ordem de chegada das solicitações. Entretanto, há casos em que é necessário, por motivos de emergência ou por prioridade legal, localizar determinado código numérico para atender à solicitação correspondente antes das demais, “furando” a fila de espera. O gerente solicitou, então, à equipe de TI do banco, uma proposta que conciliasse essas duas necessidades. Ao estudar o problema, a equipe de TI concluiu que uma solução que mapearia diretamente essa necessidade da gerência seria permitir a realização de uma busca binária sobre o vetor de solicitações ordenado pelos seus códigos numéricos.

Verificando a viabilidade dessa sugestão, o grupo de TI calculou que, se considerar a existência de N solicitações, a quantidade de iterações necessárias para localizar determinado código numérico no vetor de solicitações, utilizando a busca binária, no pior caso, é

- a) $\lfloor \log_2 N \rfloor$, em que a notação $\lfloor x \rfloor$ significa maior inteiro menor ou igual a x .
- b) $1 + \lfloor \log_2 N \rfloor$, em que a notação $\lfloor x \rfloor$ significa maior inteiro menor ou igual a x .
- c) $1 + \lfloor \log_2 N \rfloor$, em que a notação $\lfloor x \rfloor$ significa menor inteiro maior ou igual a x .
- d) 2^N .
- e) 2^{N-1}

18.(CESGRANRIO–BASA– 2021) Um determinado programador é responsável por tarefas de ordenação e, ao estudar determinados produtos, resolveu ordenar, de maneira crescente, a sequência [64, 34, 25, 12, 90, 11, 22] utilizando dois algoritmos, o Bubble Sort e o Select Sort, nessa ordem.

Ele iniciou o teste com o Bubble Sort, mas, na iteração em que a chave 64 atingiu a sua posição correta pela primeira vez, copiou a sequência alcançada nesse estágio e utilizou-a para continuar o trabalho com o algoritmo Select Sort.

A partir do momento em que o programador começa a utilizar o segundo algoritmo, quantas trocas de posições de chaves serão realizadas para atingir, pela primeira vez, a situação em que a sequência está ordenada?

- a) 1
- b) 2
- c) 3
- d) 4
- e) 5

19.(CESGRANRIO –BB – 2021) Em uma agência bancária, as filas de atendimento são ordenadas da esquerda para a direita, e o gerente dessa agência percebeu a presença equivocada de um idoso, com a senha 52, na fila de atendimento não preferencial. Visando a sanar o equívoco, o gerente resolveu que, na primeira oportunidade, faria uma busca no sistema para saber se a senha 52 ainda estava ativa, indicando a presença do idoso na fila de atendimento não preferencial. Em caso de resposta positiva, procuraria o cliente para trocar sua senha por outra de atendimento preferencial; se não, apenas registraria o fato para posterior discussão no grupo de qualidade de atendimento.



Considerando o uso de um algoritmo de busca sequencial otimizado, partindo da esquerda para a direita, e as sequências hipotéticas das senhas da fila de atendimento não preferencial e suas regras de ordenação, segundo as quais quem está à esquerda é atendido antes de quem está à direita, o menor número de comparações para o gerente conhecer o resultado de sua busca ocorre em

| Regras de ordenação | Sequência das senhas na fila de atendimento não preferencial |
|---------------------|--|
| a) | Sequência ordenada crescentemente 23; 45; 81; 97; 112; 138; 154 |
| b) | Sequência ordenada crescentemente 13; 25; 37; 44; 52; 78; 83; 91 |
| c) | Sequência ordenada crescentemente 17; 28; 32; 49; 67; 85; 94; 103 |
| d) | Sequência desordenada 27; 95; 148; 117; 33; 59; 52 |
| e) | Sequência desordenada 32; 48; 12; 55; 93; 27; 66 |

20. (CESGRANRIO –BB – 2021) Dentre os problemas identificados pela gerência de um banco comercial, está a localização das contas dos seus titulares nas listagens e nos relatórios impressos em diferentes situações. Um especialista de TI sugeriu ordenar as contas por meio dos CPF dos seus titulares antes das impressões.

Dentre alguns algoritmos pré-selecionados para essa ordenação, o especialista escolheu o algoritmo de ordenação por inserção, no qual o consumo de tempo é, no melhor caso, proporcional a

- a) $n \log n$
- b) $\log n$
- c) n^2
- d) n
- e) 1

21. (CESGRANRIO –BB – 2021) As agências bancárias negociam seguros residenciais com seus clientes e, muitas vezes, precisam arquivar cópias de forma ordenada para que consultas eventuais sejam facilitadas. O gerente de uma agência precisava ordenar um vetor de documentos referentes a esses seguros, e o seu adjunto, da área de TI, o aconselhou a usar o algoritmo de ordenação chamado Bubble Sort.



Utilizando-se o algoritmo sugerido, qual será a quantidade de trocas de posições realizadas para ordenar, de modo crescente, o vetor de números de contrato (77, 51, 11, 37, 29, 13, 21)?

- a) 14
- b) 15
- c) 16
- d) 17
- e) 18

22. (CESGRANRIO –UNIRIO – 2019) O programa Java a seguir ordena um array com 64 números inteiros gerados aleatoriamente.

```
import java.util.Random;

public class Main {

    static int cont=0;
    static int nRep=10000;
    static int tam=64;

    static int particao(int arr[], int inicio, int fim) {
        int pivot = arr[fim];
        int i = (inicio-1);

        for (int j = inicio; j < fim; j++) {
            cont++; //PASSO 1
            if (arr[j] <= pivot) {
                i++;
                int trocaTemp = arr[i];
                arr[i] = arr[j];
                arr[j] = trocaTemp;
            }
        }
    }
}
```



```
}
```

```
int trocaTemp = arr[i+1];
```

```
arr[i+1] = arr[fim];
```

```
arr[fim] = trocaTemp;
```



```
return i+1;
```

```
}
```



```
static void quickSort(int arr[], int inicio, int fim) {
```

```
if (inicio < fim) {
```

```
int particaoIndex = particao(arr, inicio, fim);
```

```
quickSort(arr, inicio, particaoIndex-1);
```

```
quickSort(arr, particaoIndex+1, fim);
```

```
}
```

```
}
```

```
public static void main(String[] args) {
```

```
Random g=new Random();
```



```
for(int i=0;i<nRep;i++) {
```

```
int vet[]=new int[tam];
```

```
for(int j=0;j<tam;j++)
```

```
vet[j]=g.nextInt(100);
```

```
quickSort(vet,0,tam-1);
```



```
}
```

```
System.out.println(cont/nRep); //PASSO 2
```

```
}
```

```
}
```

No interior do comando for do método particao(), foi inserido um comando (cont++) que incrementa a variável estática cont de uma unidade a cada iteração do comando for. Após a execução do método quickSort(), a variável cont irá conter o número total de iterações realizadas para que o array fosse ordenado.

A função de ordenação, de nome quickSort, é chamada 10000 vezes, com diferentes arrays de números inteiros, gerados aleatoriamente, em cada chamada. Sendo assim, o valor exibido pelo método println(), ao término do programa, será a média do número de iterações das 10000 vezes em que o array foi ordenado.

Seja p o número exibido pelo método println() em consequência da execução do programa acima. Seja t o número obtido a partir da complexidade do caso médio do algoritmo quicksort aplicada ao array do programa acima. Seja m o valor absoluto da diferença entre t e p, isto é, $m=|t-p|$.

Qual valor de p resulta no menor valor de m?

- a) 361
- b) 720
- c) 1024
- d) 2048
- e) 4096

23. (CESGRANRIO –BB – 2018) O programa a seguir, em Python, implementa o algoritmo do método de bolha, imprimindo o resultado de cada passo.

```
def bolha(lista):  
    for passo in range(len(lista)-1,0,-1):  
        for i in range(passo):  
            if lista[i]>lista[i+1]:  
                lista[i],lista[i+1]=lista[i+1],lista[i]  
    print(lista)
```

Qual será a quarta linha impressa para a chamada bolha([4, 3, 1, 9, 8, 7, 2, 5]) ?

- a) [3, 1, 4, 8, 7, 2, 5, 9]
- b) [1, 3, 4, 7, 2, 5, 8, 9]



- c) [1, 2, 3, 4, 5, 7, 8, 9]
- d) [1, 3, 2, 4, 5, 7, 8, 9]
- e) [1, 3, 4, 2, 5, 7, 8, 9]

24.(CESGRANRIO –BB – 2018) Uma árvore binária cujos nós armazenam números inteiros pode ser representada na linguagem Python por uma lista com três elementos:

- o primeiro representa a informação armazenada no nó (número inteiro);
- o segundo é uma lista que representa a subárvore esquerda;
- o terceiro é uma lista que representa a subárvore direita.

As variáveis a seguir representam os nós de uma árvore binária construída segundo a estrutura acima descrita. Os nós n3, n4 e n6 são as folhas; n1, n2 e n5 são os nós intermediários; e n0 é o ó raiz.

```
n6=[4,[],[]]  
n5=[6,[],n6]  
n2=[8,n5,[]]  
n3=[5,[],[]]  
n4=[9,[],[]]  
n1=[7,n3,n4]  
n0=[3,n1,n2]
```

Seja o seguinte programa Python:

```
def lista(n):  
if n==[]:  
    return  
    lista(n[1])  
    lista(n[2])  
    print(n[0],end=' ')
```

```
n6=[4,[],[]]  
n5=[6,[],n6]  
n2=[8,n5,[]]  
n3=[5,[],[]]  
n4=[9,[],[]]  
n1=[7,n3,n4]  
n0=[3,n1,n2]  
lista(n0)
```

O que será exibido no console quando ele for executado?



- a) 4 6 8 9 5 7 3
- b) 8 4 6 3 9 7 5
- c) 5 9 7 4 6 8 3
- d) 3 7 5 9 8 6 4
- e) 5 7 9 3 6 4 8

25. (CESGRANRIO –TRANSPETRO– 2018) Analise o algoritmo de ordenação que se segue.

```
def ordenar(dado):
    for passnum in range(len(dado)-1,0,-1):
        for i in range(passnum):
            if dado[i]>dado[i+1]:
                temp = dado[i]
                dado[i] = dado[i+1]
                dado[i+1] = temp
dado = [16,18,15,37,13]
ordenar(dado)
print(dado)
```

Com o uso desse algoritmo, qual é a quantidade de trocas realizadas para ordenar a sequência dado?

- a) 4
- b) 5
- c) 6
- d) 7
- e) 8

26.(CESGRANRIO –BB– 2018) Deseja-se realizar uma busca sobre um vetor não ordenado de inteiros. Para tal, deve-se criar um método Java que receba como parâmetros o vetor em questão e um número inteiro (elemento) que se deseja procurar no vetor, além de outros parâmetros que se julgarem necessários. Essa função deve retornar

- o índice do elemento no vetor, caso ele seja encontrado;
- o inteiro -1, caso o elemento não seja encontrado.

Assumindo-se que todos os pacotes necessários foram devidamente importados, qual método

Java irá realizar corretamente essa busca?



a)

```
static int busca(int[] vet, int elem, int ini, int fim) {
    if (ini > fim)
        return -1;

    int m = (ini + fim) / 2;
    if (elem == vet[m])
        return m;
    else
        if (elem > vet[m])
            return busca(vet, elem, m + 1, fim);
        else
            return busca(vet, elem, ini, m - 1);
```

b)

```
public static int busca(int vet[], int elem) {
    for(int i=0; i < vet.length; i++)
        if(elem == vet[i])
            return i;
        else
            if(elem < vet[i])
                return -1;

    return -1;
```

c)

```
public static int busca(int[] vet, int elem) {
    int ini = 0, fim = vet.length-1;

    while(ini <= fim) {
        int m = (ini + fim) / 2;
        if (elem == vet[m])
            return m;
        else
            if (elem > vet[m])
                ini = m + 1;
            else
                fim = m - 1;
    }

    return -1;
```

d)

```
public static int busca(int vet[], int elem) {
    if(vet.length==0)
        return -1;

    if(elem==vet[vet.length-1])
        return vet.length-1;

    return busca((Arrays.copyOf(vet,vet.length-1),elem));
```



```

public static int busca(int vet[], int elem) {
    if(vet.length == 0)
        return -1;

    if(elem == vet[vet.length-1])
        return vet.length-1;
    else
        if(elem > vet[vet.length-1])
            return -1;

    e)   return busca(Arrays.copyOf(vet, vet.length-1), elem);
}

```

27. (CESGRANRIO –BASA– 2018) Uma árvore binária completa de busca, isto é, uma árvore em que todos os níveis têm o máximo número de elementos, tem um total de N nós. O número máximo de comparações necessárias para encontrar um elemento nessa árvore é

- a) N
- b) N^2
- c) $\log_2(N+1)$
- d) $(N+1)*\log_2(N+1)$
- e) \sqrt{N}

28.(CESGRANRIO–PETROBRAS–2018) Dada a sequência numérica (15,11,16,18,23,5,10,22,21,12) para ordenar pelo algoritmo Selection Sort, qual é a sequência parcialmente ordenada depois de completada a quinta passagem do algoritmo?

- a) [15, 11, 16, 18, 12, 5, 10, 21, 22, 23]
- b) [15, 11, 5, 10, 12, 16, 18, 21, 22, 23]
- c) [15, 11, 16, 10, 12, 5, 18, 21, 22, 23]
- d) [10, 11, 5, 12, 15, 16, 18, 21, 22, 23]
- e) [12, 11, 5, 10, 15, 16, 18, 21, 22, 23]

29.(CESGRANRIO –PETROBRAS– 2018) A sequência de chaves 20 – 30 – 25 – 31 – 12 – 15 – 8 – 6 – 9 – 14 – 18 é organizada em uma árvore binária de busca. Em seguida, a árvore é percorrida em pré-ordem.

Qual é a sequência de nós visitados?

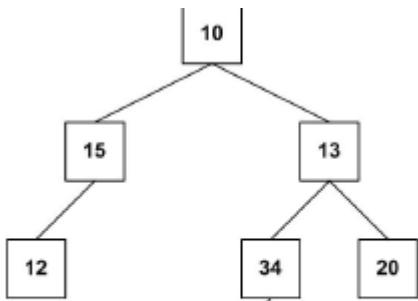
- a) 6 – 9 – 8 – 14 – 18 – 15 – 12 – 25 – 31 – 30 – 20
- b) 20 – 12 – 8 – 6 – 9 – 15 – 14 – 18 – 30 – 25 – 31
- c) 6 – 8 – 9 – 12 – 14 – 15 – 18 – 20 – 25 – 30 – 31
- d) 20 – 30 – 31 – 25 – 12 – 15 – 18 – 14 – 8 – 9 – 6
- e) 6 – 8 – 9 – 14 – 15 – 18 – 12 – 25 – 30 – 31 – 20

30. (CESGRANRIO –TRANSPETRO– 2018) Uma árvore binária foi percorrida em ordem simétrica, e os valores de seus nós exibidos no console. O resultado desse procedimento foi o seguinte:

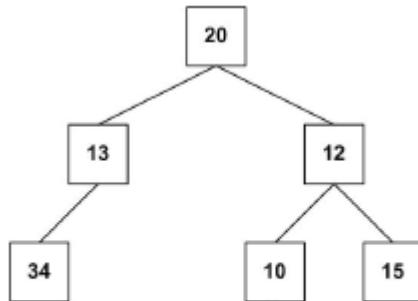
15 12 10 19 20 13 34



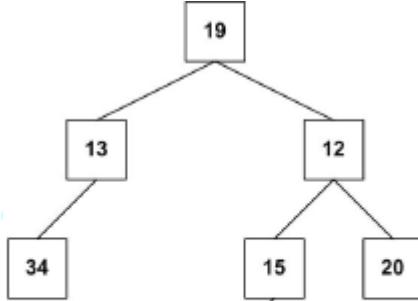
Dentre as árvores apresentadas, a única capaz de produzir o resultado acima é



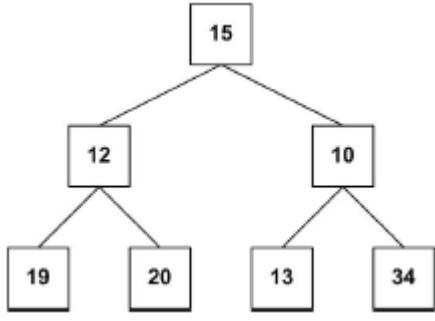
a)



b)

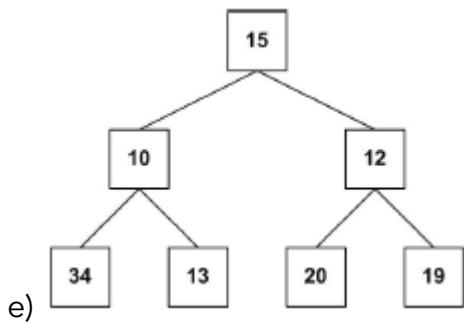


c)



d)





31. (CESGRANRIO –TRANSPETRO– 2018) Um método que implementa um algoritmo de busca binária recebe como parâmetros um vetor de inteiros ordenados descendenteamente, o comprimento desse vetor e um número inteiro que se deseja localizar no vetor. O cabeçalho desse método é o seguinte:

```
public int buscaBin(int vet[], int n, int val)
```

Admitindo-se que o vetor passado como parâmetro tenha 750 elementos, qual será o número máximo de iterações que o algoritmo irá realizar até que o valor (val) seja localizado ou que seja detectado que esse valor não se encontra no vetor?

- a) 8
- b) 9
- c) 10
- d) 11
- e) 12

32. (CESGRANRIO –PETROBRAS– 2018) A função a seguir implementa um algoritmo de busca binária sobre um vetor de inteiros ordenado de modo ascendente.

```
int busca(int vet[], int elem, int ini, int fim) {
    int m;
    if(fim < ini)
        return -1;
    m=(ini + fim) / 2;
    System.out.println(vet[m]);
    if(vet[m] == elem)
        return m;
    if(vet[m] > elem)
        return busca(vet, elem, ini, m-1);
    return busca(vet, elem, m+1, fim);
}
```

Essa função recebe como parâmetros um vetor (vet), o elemento que se deseja procurar no vetor (elem), o índice do primeiro elemento do vetor (ini) e o índice do último elemento do vetor (fim).



O comando `System.out.println(vet[m])` exibe no console o valor do elemento de índice m do vetor vet.

Seja o seguinte vetor (vt) de inteiros:

| | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 20 | 25 | 27 | 38 | 51 | 57 | 60 | 65 | 73 | 74 | 78 | 80 | 83 | 88 | 90 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |

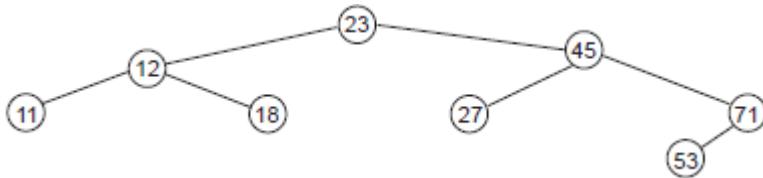
Suponha que a função busca seja chamada por meio do seguinte comando:

`busca(vt, 39, 0, 14);`

Qual será o 3º valor exibido no console?

- a) 65
- b) 51
- c) 57
- d) 38
- e) 27

33. (CESGRANRIO –TRANSPETRO– 2018) Analise a árvore binária de busca (BST), abaixo, representada pelas chaves dos seus nós.



Qual é a sequência de chaves representativa do seu percurso em pré-ordem?

- a) 11; 12; 18; 23; 27; 45; 53; 71
- b) 11; 18; 12; 27; 53; 71; 45; 23
- c) 23; 12; 11; 18; 45; 27; 71; 53
- d) 53; 71; 27; 45; 18; 11; 12; 23
- e) 23; 45; 71; 27; 53; 18; 12; 11

34. (CESGRANRIO –PETROBRAS– 2018) A seleção de uma estrutura de dados adequada muitas vezes acelera a solução de um problema. A Pilha é uma das estruturas de dados mais importantes. Que propriedade caracteriza uma Pilha?

- a) Permite inserção em qualquer posição.
- b) Suas folhas estão no mesmo nível.
- c) Seus nós têm no máximo dois filhos.
- d) O último elemento inserido será o primeiro a ser removido.
- e) O primeiro elemento inserido será o primeiro a ser removido.



35. (CESGRANRIO –TRANSPETRO– 2018) Considere uma árvore binária de busca (BST) com n ($n > 3$) níveis (o nó raiz está no nível 1), $2n - 1$ nós e todas as chaves diferentes. Suponha, ainda, que algum dos pais de duas folhas seja removido da árvore e, mais tarde, uma chave com o mesmo valor da chave do nó removido seja inserida na árvore.

Quantas são as comparações necessárias para fazer a busca e encontrar o nó cuja chave foi removida e depois reinserida?

- a) $n - 2$
- b) $n - 1$
- c) n
- d) $n + 1$
- e) $n + 2$

36. (CESGRANRIO –PETROBRAS– 2018) Um programador construiu uma função para ordenar vetores de inteiros por meio do algoritmo de ordenação por inserção (insertion sort). A versão iterativa desse algoritmo possui dois loops aninhados. Suponha que esse programador tenha inserido, imediatamente antes do incremento da variável de controle do loop mais externo, uma chamada de uma função para percorrer e exibir o conteúdo do vetor que está sendo ordenado. O trecho de código a seguir ilustra como essa chamada é feita.

```
for (int i = 1; i < vetor.length; i++){  
    /* Os demais comandos que implementam o algoritmo de ordenação por inserção devem substituir essas linhas com comentários */  
    exibeVetor(vetor);  
}
```

A Figura abaixo exibe o vetor que foi passado como parâmetro em uma chamada da função de ordenação.

| | | | | | | | | | |
|----|----|----|---|----|---|----|----|----|---|
| 78 | 12 | 35 | 1 | 17 | 4 | 43 | 11 | 17 | 1 |
|----|----|----|---|----|---|----|----|----|---|

O que será exibido no console quando o valor da variável i for igual a 3?

- a) 1 1 11 12 17 4 17 35 43 78
- b) 1 12 4 17 11 17 1 35 43 78
- c) 1 1 4 78 17 35 43 11 17 12
- d) 1 12 35 78 17 4 43 11 17 1
- e) 1 1 4 11 17 35 43 78 17 12



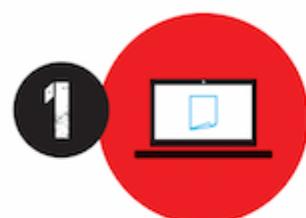
GABARITO

- | | | |
|-------------|-------------|-------------|
| 1. LETRA C | 13. LETRA B | 25. LETRA C |
| 2. LETRA A | 14. LETRA E | 26. LETRA D |
| 3. LETRA B | 15. LETRA B | 27. LETRA C |
| 4. LETRA B | 16. LETRA C | 28. LETRA B |
| 5. LETRA A | 17. LETRA B | 29. LETRA B |
| 6. LETRA D | 18. LETRA B | 30. LETRA B |
| 7. LETRA A | 19. LETRA A | 31. LETRA C |
| 8. LETRA C | 20. LETRA D | 32. LETRA C |
| 9. LETRA C | 21. LETRA C | 33. LETRA C |
| 10. LETRA B | 22. LETRA A | 34. LETRA D |
| 11. LETRA D | 23. LETRA D | 35. LETRA D |
| 12. LETRA E | 24. LETRA C | 36. LETRA D |



ESSA LEI TODO MUNDO CONHECE: PIRATARIA É CRIME.

Mas é sempre bom revisar o porquê e como você pode ser prejudicado com essa prática.



1

Professor investe seu tempo para elaborar os cursos e o site os coloca à venda.



2

Pirata divulga ilicitamente (grupos de rateio), utilizando-se do anonimato, nomes falsos ou laranjas (geralmente o pirata se anuncia como formador de "grupos solidários" de rateio que não visam lucro).



3

Pirata cria alunos fake praticando falsidade ideológica, comprando cursos do site em nome de pessoas aleatórias (usando nome, CPF, endereço e telefone de terceiros sem autorização).



4

Pirata compra, muitas vezes, clonando cartões de crédito (por vezes o sistema anti-fraude não consegue identificar o golpe a tempo).



5

Pirata fere os Termos de Uso, adultera as aulas e retira a identificação dos arquivos PDF (justamente porque a atividade é ilegal e ele não quer que seus fakes sejam identificados).



6

Pirata revende as aulas protegidas por direitos autorais, praticando concorrência desleal e em flagrante desrespeito à Lei de Direitos Autorais (Lei 9.610/98).



7

Concursado(a) desinformado participa de rateio, achando que nada disso está acontecendo e esperando se tornar servidor público para exigir o cumprimento das leis.



8

O professor que elaborou o curso não ganha nada, o site não recebe nada, e a pessoa que praticou todos os ilícitos anteriores (pirata) fica com o lucro.



Deixando de lado esse mar de sujeira, aproveitamos para agradecer a todos que adquirem os cursos honestamente e permitem que o site continue existindo.