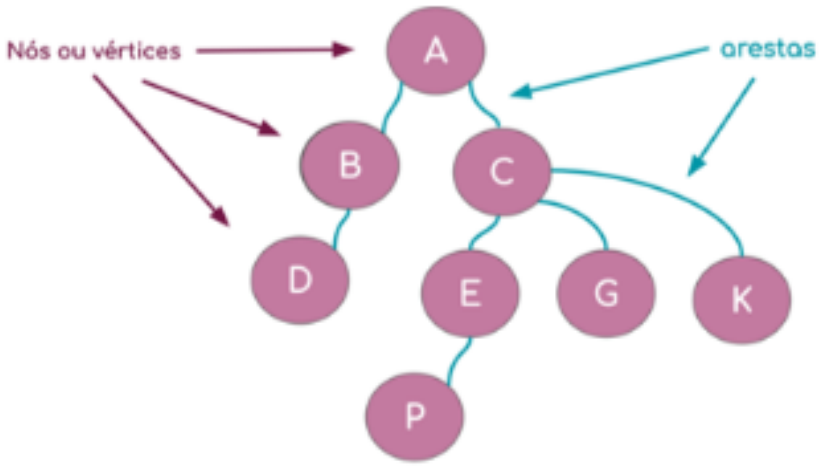
Conceito de Árvores

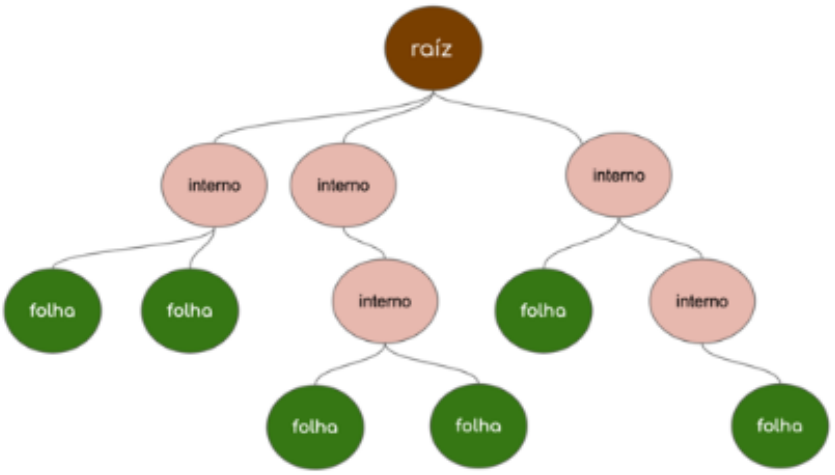
Na ciência da computação, uma árvore é uma estrutura de dados hierárquica e não linear, caracterizada por sua forma ramificada, análoga à de uma árvore biológica, da qual deriva sua nomenclatura. Essa estrutura é composta por entidades denominadas vértices ou nós, que representam os elementos de dados, e por conexões conhecidas como arestas, que estabelecem as relações entre os vértices. Um exemplo dessa estrutura pode ser observado na figura a seguir, onde se ilustra uma árvore que contém oito vértices interconectados por sete arestas (Szwarcfiter; Markenzon, 2021).



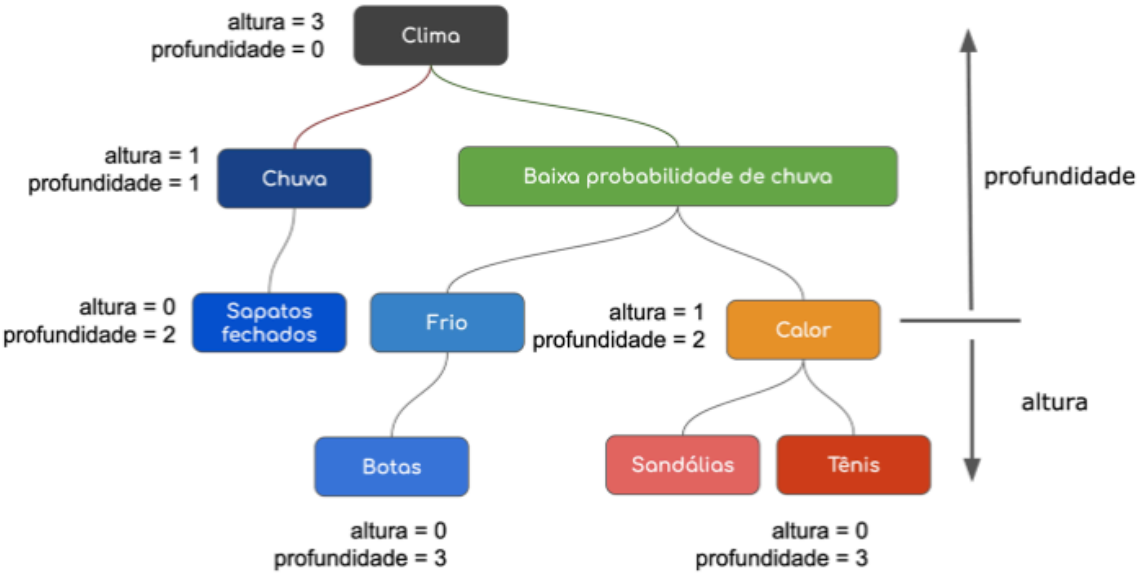
Em uma estrutura de dados em árvore, as conexões entre os vértices, conhecidas como arestas, seguem regras específicas para garantir a formação adequada da estrutura. Primeiramente, cada vértice na árvore pode ter um máximo de uma aresta que chega até ele, garantindo que não existam múltiplas entradas para o mesmo ponto. Em segundo lugar, o número de arestas que partem de um vértice específico pode variar entre zero e um valor máximo **N**, onde **N** representa o número máximo de arestas saindo desse vértice. Essas regras asseguram que, ao percorrer os vértices da árvore, não se formará um circuito fechado, mantendo a estrutura acíclica e hierárquica característica das árvores (Alves, 2021).

Em uma estrutura de dados em forma de árvore, cada elemento é chamado de vértice ou nó. O nó do qual emergem outras conexões é referido como "pai", enquanto os nós que recebem essas conexões são denominados "filhos". Nós que compartilham o mesmo pai são conhecidos como "irmãos". Por exemplo, se o nó **A** é o pai dos nós **B** e **C**, **B** é pai de **D**, e **C** é pai de **E**, **G** e **K**, esses relacionamentos estabelecem uma hierarquia clara dentro da estrutura da árvore.

A posição de um vértice na árvore determina sua nomenclatura específica. A "raiz" é o vértice inicial da árvore e é único, pois não possui um nó pai. Já os "nós folha" ou "terminais" são aqueles que não possuem filhos. Entre a raiz e os nós folha estão os nós "internos" ou "não terminais", que desempenham o papel de conectar diferentes partes da árvore.



A altura de um vértice em uma árvore é determinada pela contagem do número de arestas desde esse vértice até o mais distante em sua subárvore. Ou seja, a altura é medida de cima para baixo. Já a altura total da árvore é calculada a partir da raiz até o nó mais distante. Por outro lado, a profundidade de um vértice é calculada pelo número de arestas que se percorre desde esse vértice até a raiz, portanto, de baixo para cima. Importante notar que, embora os nós folha tenham a mesma altura, eles podem ter profundidades diferentes dependendo de sua localização na árvore.



Árvore Binária de busca

De forma mais técnica, uma árvore binária de busca (ABB), também conhecida como BST (*Binary Search Tree*) em inglês, é um subtipo de árvore binária. Nessa estrutura, todas as chaves (conteúdos dos nós) presentes na subárvore esquerda de um nó são menores do que as chaves do nó raiz. Por sua vez, na subárvore direita, todas as chaves são maiores (Alves, 2021). Esse padrão de organização é

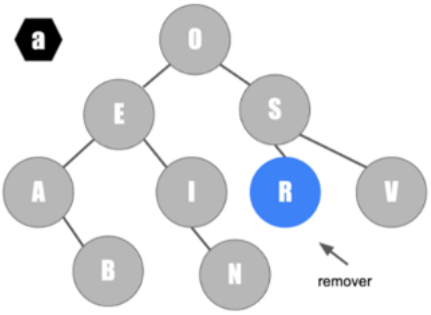
estabelecido durante o processo de inserção de dados na árvore, garantindo que a busca e recuperação de informações sejam realizadas de maneira eficiente.
Uma árvore de busca binária é uma estrutura de dados baseada em nós, onde cada nó possui no máximo dois filhos. Os elementos são organizados de forma que, para cada nó, todos os elementos na subárvore à esquerda são menores que o nó e todos os elementos na subárvore à direita são maiores. Vamos exemplificar como criar e inserir elementos nessa estrutura em Python, usando a seguinte sequência: 14, 4, 18, 0, 21, 17, 1, 8 e 13 (Takenaka, 2021).

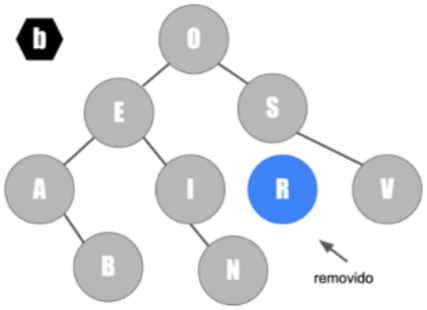
Ordenação em árvores

E como funciona a remoção de vértices?
A remoção de vértices em uma árvore binária de busca envolve três cenários distintos que requerem abordagens específicas:

Remoção de vértice-folha

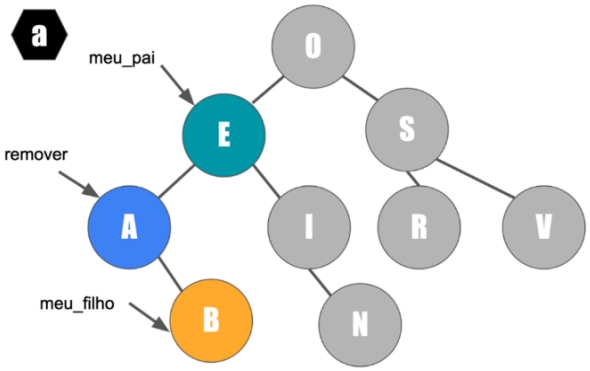
Este é o caso mais simples. Um vértice-folha é um nó que não possui filhos. Para removê-lo, basta desconectar esse nó da árvore, eliminando a referência a ele no nó pai.

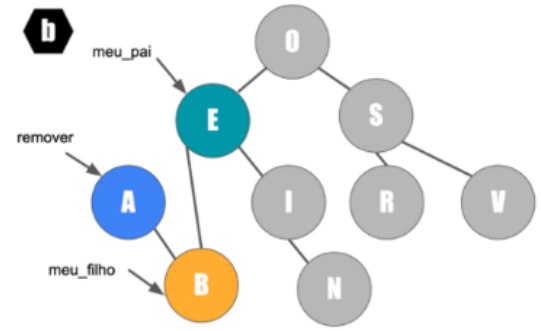


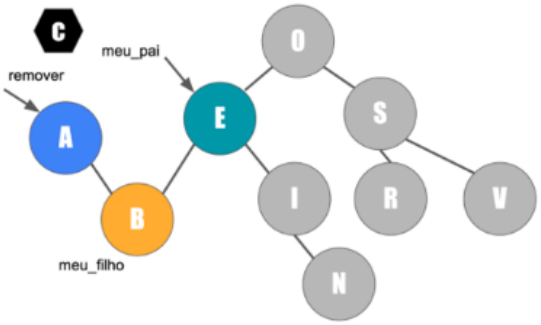


Remoção de vértice com um filho

Neste cenário, o vértice a ser removido tem apenas um filho (esquerdo ou direito). A remoção envolve substituir o vértice pelo seu filho único no nó pai. Isso significa que o nó pai passa a apontar diretamente para o filho do vértice removido, efetivamente removendo o vértice original da árvore.

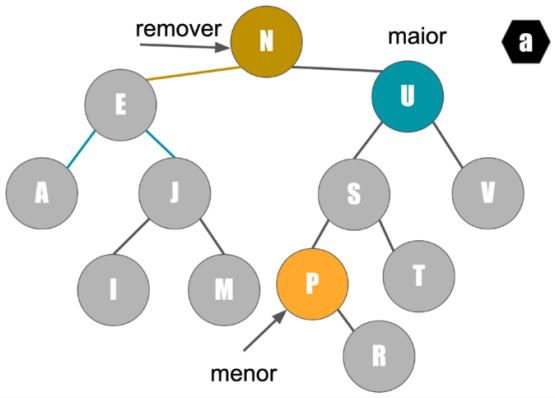


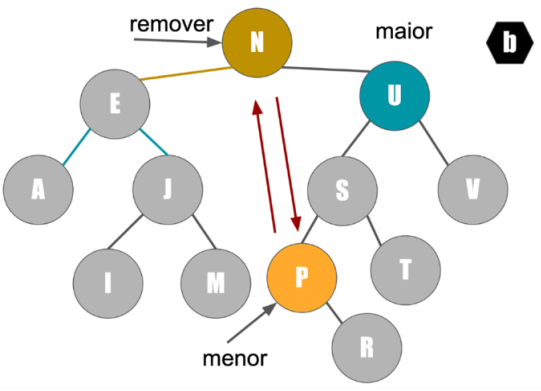




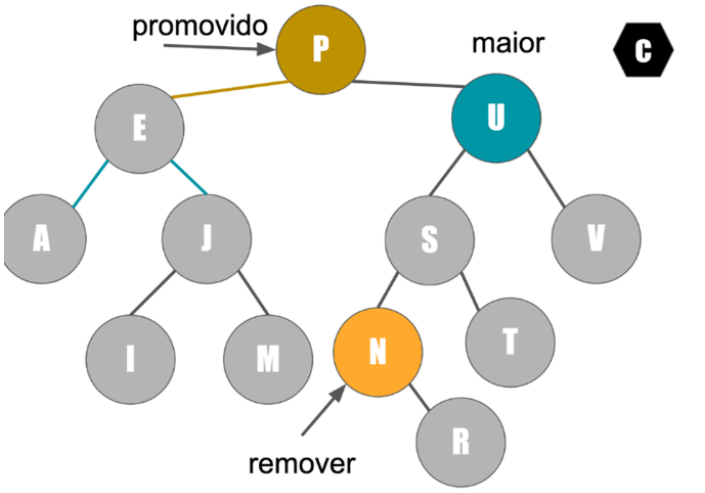
Remoção de vértice com dois filhos

Este é o caso mais complexo. Quando um vértice com dois filhos precisa ser removido, é necessário encontrar um substituto adequado que mantenha as propriedades da árvore binária de busca. Geralmente, escolhe-se o menor nó da subárvore direita do vértice a ser removido (conhecido como sucessor) ou o maior nó da subárvore esquerda (conhecido como antecessor). Após encontrar o substituto, este é movido para o local do vértice removido, e o processo de remoção é então aplicado ao substituto em sua posição original na árvore (Backes, 2023).





Note que como ele tomou o lugar do menor do lado direito, ou ele é um vértice-folha ou um vértice pai de um filho, que são casos já tratados anteriormente, então, deixamos a recursividade cuidar da remoção do vértice.

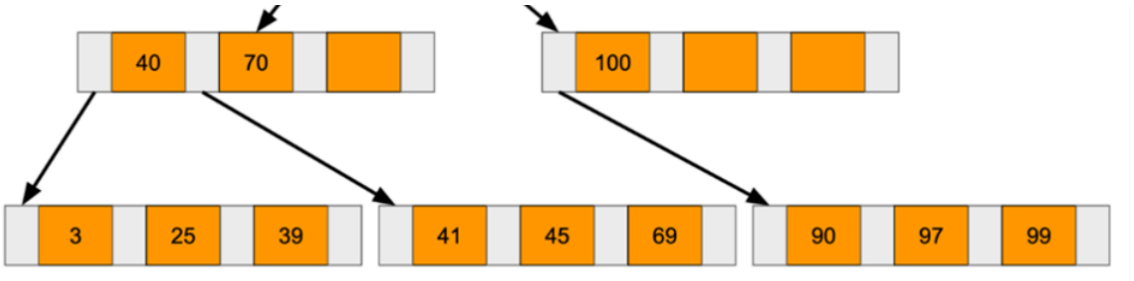


Árvore B

Árvores B são estruturas de dados hierárquicas e não lineares, destinadas ao armazenamento organizado de informações, facilitando operações como busca, inserção, remoção e navegação de forma eficaz. Essa estrutura é particularmente adequada para aplicações que manipulam extensos volumes de dados e exigem operações de leitura e escrita de grandes blocos de dados, tais como sistemas de gerenciamento de banco de dados e sistemas de arquivos. Em contraste com as árvores binárias de busca, que limitam cada nó a dois descendentes, os nós em uma árvore B podem ter um número substancialmente maior de filhos, definido pelo fator de ramificação da árvore.

Os dados em uma árvore B são mantidos balanceados, com cada nó contendo várias chaves ordenadas e ponteiros para outros nós. Essa organização permite que cada nó abrigue um número variável de chaves, respeitando um limite preestabelecido, facilitando que as operações mencionadas sejam executadas em tempo logarítmico. Isso é particularmente eficiente para grandes conjuntos de dados, garantindo acesso rápido e organizado.

A manutenção do equilíbrio na estrutura das árvores B é assegurada através da divisão e fusão de nós quando necessário, particularmente durante inserções e remoções. Essa capacidade de ajuste garante o balanceamento contínuo da árvore, otimizando o acesso aos dados armazenados.



A ordem de uma árvore B, definida como o número máximo de ponteiros que uma página pode ter, é estabelecida com base na capacidade máxima de chaves que uma página pode conter, garantindo que a ocupação de chaves em cada página nunca seja inferior à metade da sua capacidade total. Essa característica permite que as árvores B lidem com uma vasta gama de aplicações, oferecendo uma solução eficiente para a organização e recuperação de informações em sistemas complexos (Szwarcfiter; Markenzon, 2020).

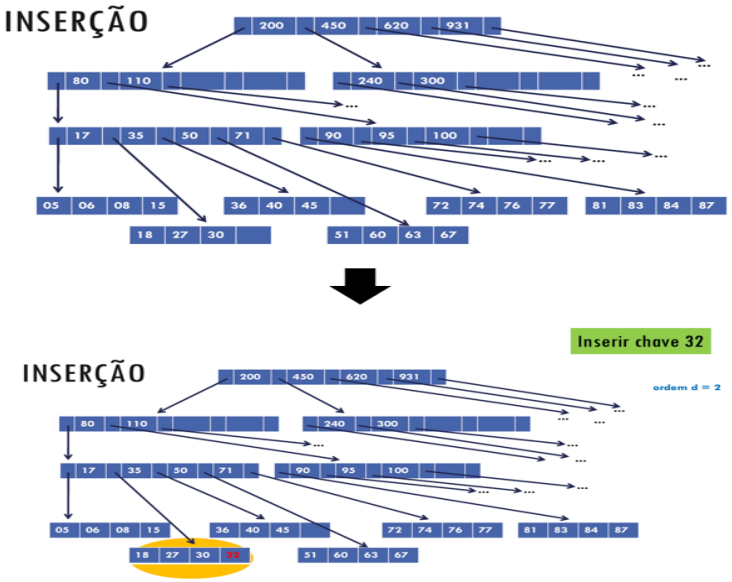
As árvores B representam uma categoria de estruturas de dados balanceadas que empregam procedimentos específicos de balanceamento durante as operações de inserção e remoção para preservar suas propriedades de organização. Diferentemente da árvore AVL (que iremos estudar posteriormente), onde o foco está em manter um balanceamento estrito através de rotações, as árvores B lidam com a complexidade adicional decorrente da presença de múltiplas chaves em cada vértice. Uma das características distintivas das árvores B é que suas folhas, ou páginas-folha, estão posicionadas no mesmo nível hierárquico, e o crescimento da estrutura ocorre de baixo para cima, das folhas em direção à raiz.

O processo de inserção em uma árvore B começa pela localização da página adequada para a nova chave, verificando-se a existência de espaço disponível. Se houver espaço, a chave é inserida de forma ordenada. Na eventualidade de a página estar cheia, procede-se à sua divisão, ou "split", criando uma nova página que alojará metade dos elementos da página original. Essa divisão exige que uma das chaves ascenda ao nível superior para manter a integridade estrutural da árvore, garantindo a correta referência às páginas divididas.

Localização da página apropriada: inicialmente, percorre-se a árvore a partir da raiz, seguindo os ponteiros adequados, para encontrar a página (nó) onde a nova chave deve ser inserida.

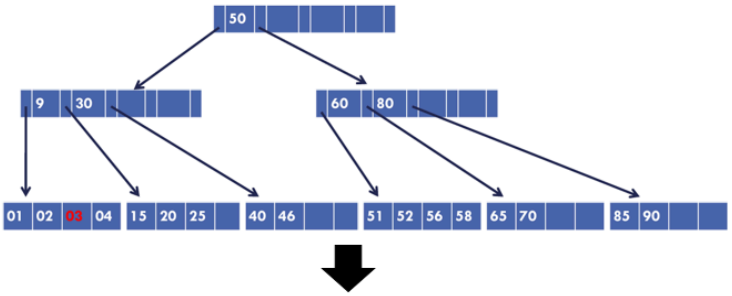
Inserção na página: se a página encontrada tem espaço para a nova chave, a chave é inserida mantendo a ordem das chaves na página.

Divisão da página (*split*): se a página estiver cheia, ela é dividida ao meio, criando uma nova página. Uma chave da página original é movida para o nó pai para servir como ponto de separação entre as duas novas páginas.

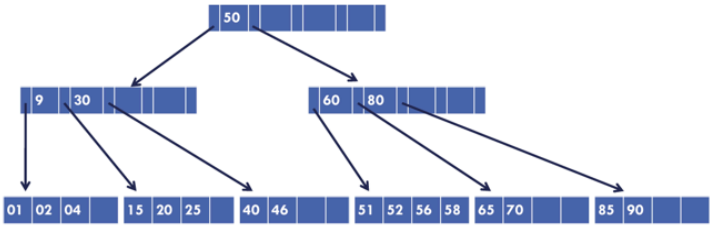


Remoção de uma chave da folha: se a chave a ser removida está em uma folha e a folha tem chaves suficientes, simplesmente remove-se a chave.
Remoção de uma chave de um nó interno: se o nó não é uma folha, há várias estratégias para lidar com a remoção, como substituir a chave por seu predecessor ou sucessor imediato, ou, se necessário, fundir nós.

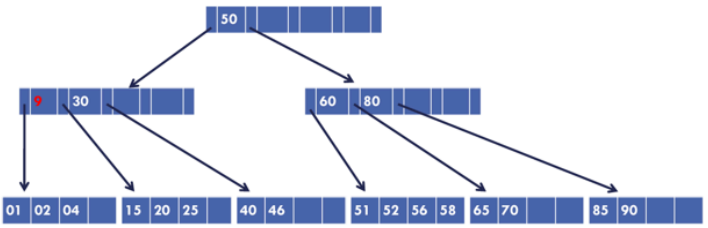
EXCLUSÃO DA CHAVE 03



EXCLUSÃO DA CHAVE 03

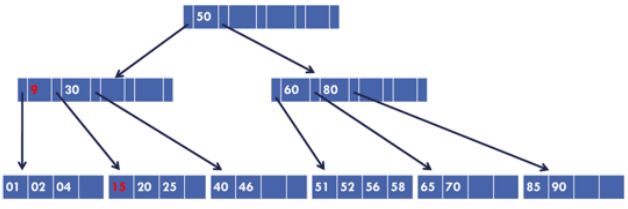


EXCLUSÃO DA CHAVE 9



Substituir pela chave imediatamente maior

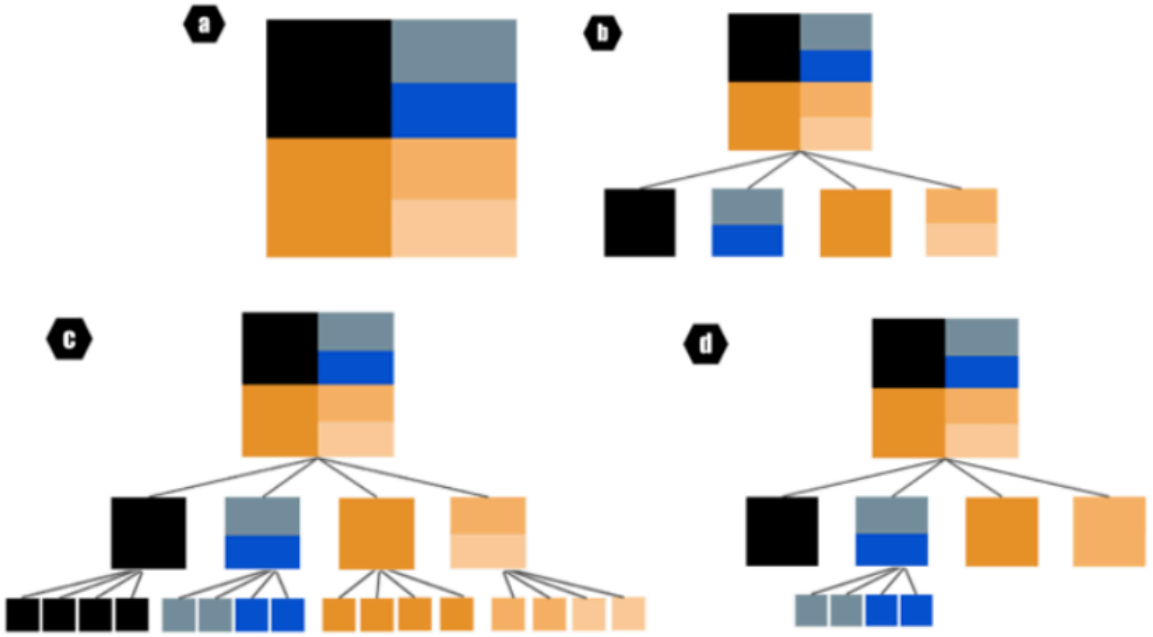
EXCLUSÃO DA CHAVE 9



Substituir pela chave imediatamente maior

Quadtree

A quadtree é uma estrutura de dados em árvore em que cada nó interno possui exatamente quatro filhos, representando uma maneira eficaz de particionar um espaço bidimensional. É amplamente aplicada no processamento de imagens, em aplicações de jogos digitais e na compactação de imagens, possibilitando a representação de espaços e objetos de maneira hierárquica e eficiente. Considerando uma imagem quadrada composta por diversos pixels, a quadtree permite sua decomposição em quatro quadrantes de dimensões iguais, cada um representado por um nó na estrutura da árvore, como ilustrado na Figura 5. Esse processo de decomposição é repetido recursivamente para cada quadrante até que se atinja um nível onde a divisão adicional não é mais necessária ou possível, tipicamente quando um quadrante se torna homogêneo em termos de cor ou quando atinge o tamanho de um pixel.



Nesse contexto, uma aplicação significativa das Quadtrees está na compactação de imagens. Quando um nó possui filhos que são suficientemente homogêneos em termos de cor ou padrão, estes podem ser substituídos por um único nó, reduzindo a complexidade da representação da imagem sem comprometer significativamente sua integridade visual. Por exemplo, se um nó tem quatro filhos todos de cor preta, este nó pode ser simplificado para um nó folha preto, reduzindo o número total de nós necessários para representar a imagem.

Principais conceitos de árvores AVL

Altura

A altura de uma árvore é determinada pela distância mais longa da raiz até um nó folha, contabilizando o número de arestas nesse caminho. Uma árvore é considerada balanceada se a sua altura é proporcional ao logaritmo do número de nós, o que indica eficiência nas operações de busca, inserção e remoção (Takenaka, 2021).

Diferentes tipos de árvores possuem distintas metodologias para alcançar e manter esse equilíbrio. As árvores AVL, por exemplo, utilizam a altura de cada nó como critério para o balanceamento, realizando ajustes conforme necessário para manter

a diferença de altura entre as subárvores esquerda e direita de qualquer nó a no máximo um. Por outro lado, as árvores rubro-negras empregam uma abordagem baseada na coloração dos nós, seguindo regras específicas para preservar a estrutura balanceada (Szwarcfiter; Markenzon, 2020).

Independentemente da técnica aplicada, a estratégia comum em árvores balanceadas envolve realizar rotações em nós que violam as regras de balanceamento. Isso significa que, para qualquer nó, a diferença entre as alturas de suas subárvores esquerda (E) e direita (D) não deve exceder uma unidade. Esse critério assegura que todas as operações essenciais na árvore, como busca, inserção e remoção, possam ser realizadas em tempo logarítmico, otimizando a eficiência do processo.

Durante as operações de inserção ou remoção em árvores binárias de busca (BST), as subárvores podem perder seu equilíbrio. O algoritmo das árvores AVL é projetado para assegurar que a árvore permaneça equilibrada, adotando medidas corretivas quando o fator de balanceamento de qualquer nó atinge 2 ou -2, indicando desequilíbrio.

Para realinhar a estrutura e preservar a eficiência das buscas, utilizamos rotações. Essas operações ajustam as ligações entre os nós sem comprometer a ordenação da árvore. Existem **quatro** situações específicas nas árvores AVL que necessitam de rotações para restaurar o balanceamento. A exploração desses cenários requer a compreensão dos conceitos de altura de um nó e do balanceamento.

Altura de um nó

Diferentemente da altura total da árvore, que é a altura do nó raiz, a altura individual de cada nó também é relevante para manter o equilíbrio da árvore como um todo. Portanto, a estrutura de um nó é adaptada para incluir a informação sobre sua própria altura, facilitando o cálculo do balanceamento e a aplicação das rotações necessárias para manter a árvore AVL equilibrada após inserções ou remoções (Takenaka, 2021).

Cálculo do balanceamento de um nó

Com a capacidade de determinar a altura tanto da subárvore esquerda quanto da subárvore direita de um nó, é possível calcular o seu balanceamento. O balanceamento de um nó é obtido pela diferença entre a altura de sua subárvore esquerda e a altura de sua subárvore direita.

$$fb = h_d(u) - h_e(u)$$

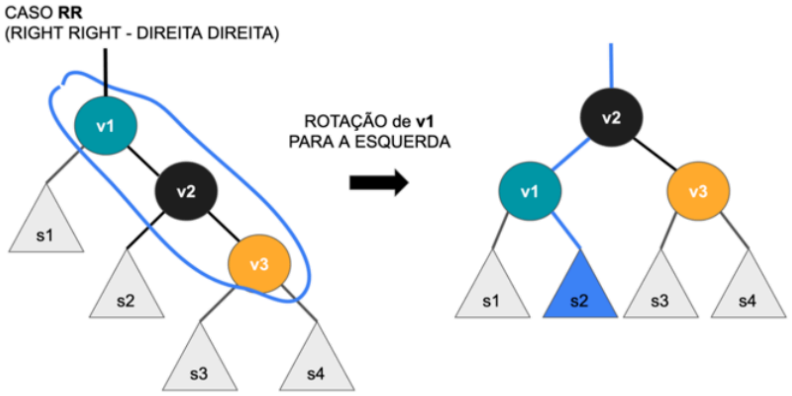
Inserção em Árvores AVL

Na inserção de dados em árvores AVL, o procedimento segue o mesmo método das árvores de busca binária (BST). Contudo, após a adição de um novo elemento, é necessário atualizar as alturas de todos os nós afetados e verificar se a árvore mantém as propriedades de uma AVL. Se a estrutura não estiver conforme os critérios de uma AVL, torna-se essencial realizar rotações específicas para restaurar o balanceamento (Takenaka, 2021).

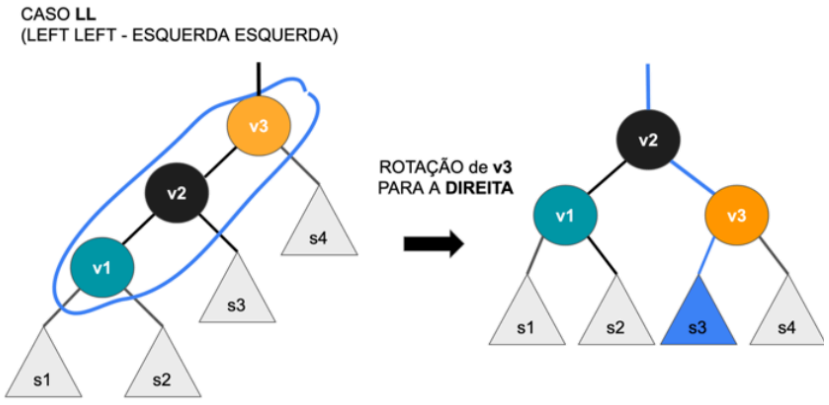
Vamos detalhar o processo de inserção utilizando a seguinte terminologia:

z: o nó recém-inserido.
y: o nó pai do elemento inserido.
x: o avô do nó inserido, ou seja, o pai de **y**.

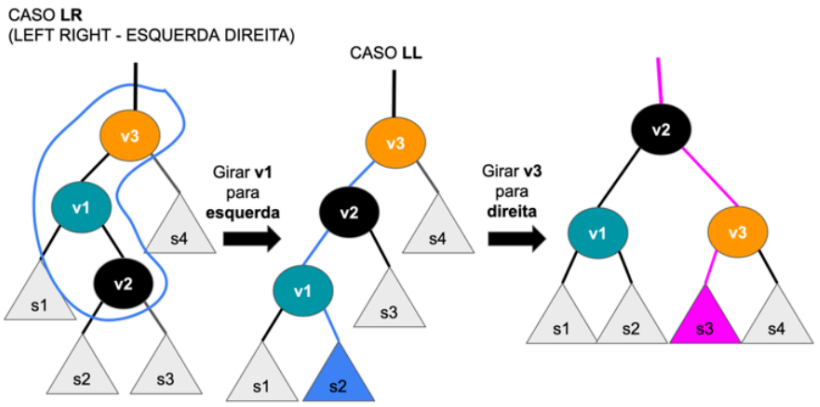
Caso 1. Rotação à direita (RR): esta situação ocorre quando o nó é inserido à esquerda de seu pai e este, por sua vez, também se encontra à esquerda de seu próprio pai, configurando uma estrutura que tende para a esquerda.
Na rotação à direita, quando temos uma configuração em que o nó e seu pai estão alinhados à esquerda, a rotação é aplicada para equilibrar a altura da subárvore. Por exemplo, numa configuração onde um nó **v1** possui um fator de balanceamento de +2 e seu filho **v2**, à direita, tem um fator de balanceamento de +1, realiza-se uma rotação simples à esquerda em **v1**, equilibrando assim a altura da subárvore, conforme ilustrado na figura referenciada (Takenaka, 2021).
Esse método assegura que as operações de inserção não comprometam a eficiência das buscas, inserções e remoções subsequentes, mantendo a estrutura de dados eficaz e balanceada.



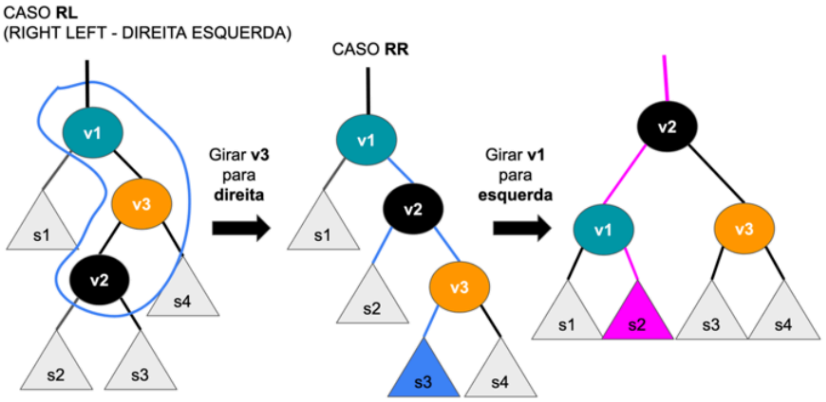
Caso 2. Rotação à esquerda (LL): quando um novo nó é adicionado à direita de seu pai e este último é posicionado à direita de seu próprio pai, indicando uma tendência à direita da árvore, é necessária uma rotação à esquerda para manter o equilíbrio.
No cenário de rotação LL, onde ambos os filhos de uma subárvore estão situados à esquerda, procede-se com uma rotação simples à direita do vértice superior dessa cadeia, ou seja, **v3**, com o objetivo de diminuir a discrepância de altura nessa subárvore. Esse processo é exemplificado na figura mencionada, onde o fator de balanceamento (fb) de **v3** é de -2 e o de **v2** é de -1 (Takenaka, 2021).
Esse método de rotação à esquerda é empregado para assegurar que a árvore mantenha sua estrutura balanceada e eficiente, facilitando operações futuras de busca, inserção e remoção ao minimizar as diferenças de altura entre as subárvores.



Caso 3. Rotação dupla à esquerda e à direita (LR): quando um nó é adicionado à direita do seu pai, que por sua vez é filho à esquerda de seu avô e a árvore inclina-se para um lado, é necessário realizar uma rotação dupla, começando à esquerda e seguindo à direita.
No cenário LR (Esquerda Direita), o filho esquerdo (**v1**) da raiz da subárvore (**v3**) possui um filho à direita (**v2**). Para reequilibrar, inicialmente realizamos uma rotação à esquerda em **v1**, reconfigurando a subárvore para um caso LL. Após essa etapa, procede-se com uma rotação à direita em **v3**, realinhando a subárvore para uma estrutura balanceada. O fator de balanceamento (fb) de **v3** é -2 e de **v1** é 1, indicando a necessidade dessa rotação composta para restabelecer o equilíbrio (Takenaka, 2021).



Caso 4. Rotação dupla à direita e à esquerda (RL): na situação inversa, quando o nó é inserido à esquerda do seu pai, que é filho à direita do seu avô, resultando numa inclinação desbalanceada da árvore, procedemos com uma rotação dupla, iniciando à direita e finalizando à esquerda.
No contexto RL (Direita Esquerda), o filho direito (**v3**) da raiz da subárvore (**v1**) tem um filho à esquerda (**v2**). Para corrigir o desequilíbrio, primeiramente realizamos uma rotação à direita em **v3**, ajustando a subárvore para um caso RR. Em seguida, aplica-se uma rotação à esquerda em **v1**, completando o reequilíbrio da estrutura. O fator de balanceamento de **v1** é 2 e de **v3** é -1, demonstrando a importância dessas rotações sequenciais para manter a estrutura da árvore otimizada para operações eficientes (Takenaka, 2021).



Remoção em Árvores AVL
A remoção de um nó em uma árvore AVL pode ser um dos três casos:

1. **Remoção de um vértice folha (_remover_folha):** se o nó a ser removido não tem filhos, ele é simplesmente removido, e a referência do pai a este nó é atualizada para **None**. O nó pai tem seu balanceamento e altura atualizados posteriormente.
2. **Remoção de um vértice com um filho (_remover_pai_de_um_filho):** se o nó a ser removido tem apenas um filho, o filho substitui a posição do nó removido na árvore, mantendo a estrutura de árvore. O pai do nó removido aponta agora para o filho do nó removido.
3. **Remoção de um vértice com dois filhos (_remover_pai_de_dois_filhos):** quando o nó a ser removido tem dois filhos, a estratégia é encontrar o sucessor do nó, que é o menor valor na subárvore direita. O valor desse sucessor é copiado para o nó a ser removido e, então, o sucessor é removido, o que reduz o problema a um dos dois casos anteriores, pois o sucessor sempre terá, no máximo, um filho.

Balanceamento

Após a inserção ou remoção de um nó, a árvore pode ficar desbalanceada. A AVL utiliza rotações para rebalancear a árvore:

- **Rotação para a esquerda (_rotacao_para_esquerda):** usada quando um nó tem um desbalanceamento à direita (RR). O nó filho direito do nó desbalanceado torna-se a nova raiz da subárvore, enquanto o nó desbalanceado se torna o filho esquerdo da nova raiz.

- **Rotação para a direita (_rotacao_para_direita):** usada quando um nó tem um desbalanceamento à esquerda (LL). O nó filho esquerdo do nó desbalanceado torna-se a nova raiz da subárvore, enquanto o nó desbalanceado se torna o filho direito da nova raiz.

ESTRUTURA DE DADOS

Árvores

FUNDAMENTOS DE ÁRVORES E ALGORITMOS

Estruturas hierárquicas fundamentais para organizar dados de maneira eficiente e otimizar algoritmos de busca, inserção e remoção.

ÁRVORES DE BUSCA BINÁRIA

Estrutura de dados que organiza elementos de forma hierárquica, facilitando operações de busca binária rápida.

ÁRVORES B, QUADTREES E AVL

Árvores B gerenciam grandes blocos de dados com eficiência, enquanto Quadtrees modelam espaços bidimensionais para otimização espacial.

AVL: Árvores auto-balanceadas que garantem operações eficientes ajustando alturas para manter o equilíbrio após cada inserção ou remoção.

Perguntas:

Questão 1

Analise as afirmações a seguir.

I. Uma árvore binária é uma árvore em que todos os vértices, exceto os folha, tem no máximo 2 filhos. Uma árvore binária de busca é uma árvore binária em que os vértices são organizados na árvore dependendo do valor da sua chave.

II. Cada vez que uma nova chave é inserida é comparada com o valor da chave do vértice atual. Se a chave é menor, ela será inserida em algum ponto na subárvore esquerda, se a chave é maior ela será inserida em algum ponto na subárvore direita.

III. A cada vértice visitado da subárvore, é feita a comparação novamente até que não existe mais vértices para serem visitados e aí é inserida a nova chave.

De acordo com as afirmativas acima, assinale a alternativa correta.

- I e II são falsas.
- I e III apenas são verdadeiras.
- I, II e III são verdadeiras.
- I, II e III são falsas.
- I apenas é verdadeira.

Correta: I, II, III são verdadeiras

Questão 2

As árvores binárias de busca têm particularidades em operações como inserção e remoção. A inserção em uma árvore binária de busca considera o valor da chave para definir sua localização na árvore. Uma árvore binária tem até 2 filhos. A árvore binária de busca a chave nova é comparada com a chave do vértice atual. Se a

chave nova é menor que a chave do vértice atual, sua inserção será na subárvore esquerda. Caso contrário, na subárvore direita. A procura pela sua posição é feita recursivamente até que encontre um vértice que não tenha filho direito ou esquerdo para comparar a chave nova. Então, significa que o vértice novo com a chave nova será o filho direito ou esquerdo. Na remoção de vértices de uma árvore, há 3 casos: quando é um vértice-folha, quando tem apenas 1 filho e quando tem 2 filhos. Para o caso 1, é só desvincular o vértice-folha. Para o caso 2, o filho do vértice a ser removido toma seu lugar. Para o caso 3, o vértice que tem a chave com o menor valor na subárvore esquerda troca de chave com o vértice que seria removido. Removemos o vértice que tinha a chave com o menor valor na subárvore esquerda. A sequência a seguir foi feita em uma árvore binária de busca:

- Inserções de: 5, 4, 6, 1, 3, 9, 0, 7, 8.

- Remoção de: 3, 5, 6.
- Inserção de 2.

Assinale a alternativa que apresenta os valores das chaves dos filhos do vértice 7.

4
8
9
4 e 8
4 e 9

Correta: 8

Questão 3:

A árvore é uma estrutura de dados não-linear, hierárquica, isto é, seu formato é ramificado, como uma árvore, por isso seu nome. Os vértices (ou nós) representam os dados e as arestas o relacionamento entre eles.De acordo com essa informação, analise as seguintes afirmativas:

I. As arestas ligam os vértices seguindo regras, número de arestas que chegam a um vértice: 0 a 1. Número de arestas que saem de um vértice: 0 a N, em que N é o número máximo de arestas.

II. Fazer percursos passando pelos vértices da árvore nunca formará um circuito fechado.
III. A representação hierárquica que é a mais comum. Representação de árvore usando o diagrama de inclusão ou diagrama de Venn que serve para representar conjuntos.
IV. Também existe a representação da árvore usando recuos, diagrama com barras, quanto mais espaços em branco no início da linha mais profundo é o vértice da árvore. A representação da árvore com numeração por níveis, assim como se usa em sumários. Apesar de cada item estar em uma linha, é possível deixar esta representação em linha, usando um separador, como ponto-e-vírgula entre os itens. Por fim a representação por parênteses aninhados.

Considerando as informações apresentadas, é correto o que se afirma em:

I, apenas.
II e III, apenas.
I e III, apenas.
I, II e IV apenas.
I, II, III e IV.

Correta: I, II, III, IV.

Questão 4

As árvores são estruturas de dados que podem ser usadas para representar dados de forma hierárquica, podem ser usadas para representar, por exemplo, um menu de programas. Os vértices ou nós da têm um relacionamento hierárquico entre eles: pais e filhos. A raiz que é o vértice inicial não tem pai, os seus filhos são os vértices

diretamente ligado a ele. Os vértices-folha não tem filhos. As árvores podem ser classificadas quanto ao número de máximo filhos de cada vértice.

Assinale a alternativa correta em relação à classificação das árvores em relação à quantidade de filhos.

Árvores n-árias são as árvores em que os vértices têm obrigatoriamente mais de 2 filhos.
Árvores binárias são as árvores em que os vértices têm no máximo 3 filhos.
Árvores n-árias são as árvores em que nenhum vértice tem 2 filhos.
Árvores binárias são as árvores em que os vértices têm pelo menos 2 filhos.
Árvores binárias são as árvores em que os vértices têm de 0 a 2 filhos.

Correta: Árvores binárias são as árvores em que os vértices têm de 0 a 2 filhos.

Questão 5

A árvore Quadtree é uma árvore que todos os vértices, exceto os folha, tem 4 filhos.

É usada para modelar espaços bidimensionais. Analise as seguintes asserções:

I. Uma das principais aplicações das árvores Quadtree é em processamento de imagens. Pode ser usada em jogos e em compactação de imagens quadradas.
II. Imagine uma imagem quadrada, e por acaso tem 4 cores: preta, laranja, cinza e azul, poderia ter mais ou menos cores, a quantidade de cores não importa. A cor laranja se apresenta em 3 tons. Decompomos a imagem em 4 quadrados de mesma dimensão. Cada um dos quadrados é representado por um vértice na árvore. Então, decompomos os quadrados em 4 novos quadrados e, assim sucessivamente, até ficar indivisível. Cada quadrado que

criamos ao decompor a imagem é representado por um vértice. Ao final, cada vértice folha representa um pixel da imagem original.

III. Aplicada em compactação de imagens, quando um vértice da árvore Quadtree tem todos os seus filhos de uma mesma cor ou de cores similares, podem deixar de existir, e o seu pai passa a ser o vértice folha. Ao invés de haver 4 filhos de cor preta, fica apenas o pai. O mesmo para a cor laranja. O quadrado composto por dois tons de laranja assume um único tom de laranja que é a mescla dos dois tons. Desta forma, a imagem pode ser representada de forma compactada. É isso que acontece quando se compacta as imagens com Quadtree.

Assinale a alternativa correta.

- I e II são falsas
- I e III apenas são verdadeiras
- **I, II e III são verdadeiras**
- I, II e III são falsas
- N.D.A

Correta: I, II, III são verdadeiras