Fatec
Rio Claro

# Estrutura de Dados

Prof. Orlando Saraiva Júnior
orlando.nascimento@fatec.sp.gov.br

# Árvores

# Árvores

Fatec
Rio Claro

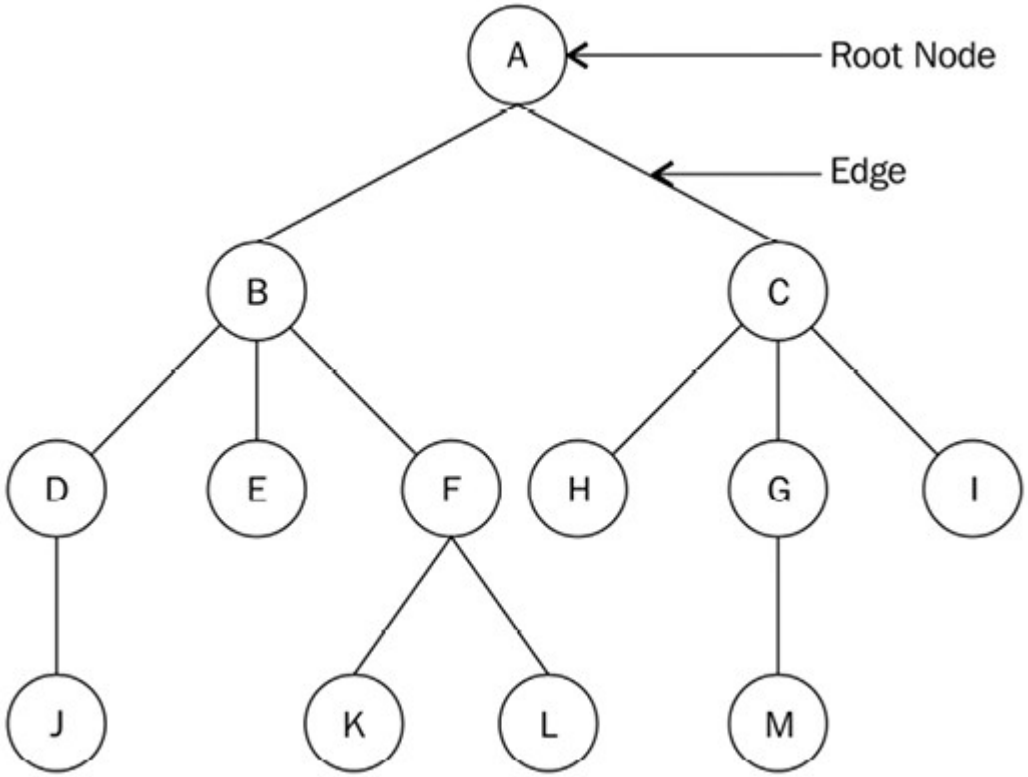
Uma árvore é uma forma hierárquica de estrutura de dados. Estruturas de dados como listas, filas e pilhas são lineares, pois os itens são armazenados sequencialmente. No entanto, uma árvore é uma estrutura de dados não linear, pois existe uma relação pai-filho entre os itens. O topo da estrutura de dados da árvore é conhecido como nó raiz. Este é o ancestral de todos os outros nós da árvore.

Estruturas de dados em árvore são muito importantes devido ao seu uso em diversas aplicações, como análise sintática de expressões, buscas eficientes e filas de prioridade. Certos tipos de documentos, como XML e HTML, também podem ser representados em uma árvore.

# Árvores

Fatec
Rio Claro

Uma árvore é uma estrutura de dados na qual os dados são organizados de forma hierárquica.



# Árvores

Fatec
Rio Claro

Aqui está uma lista de termos associados a uma árvore:

**Nó (Node):** Cada letra circulada no diagrama anterior representa um nó. Um nó é qualquer estrutura de dados que armazena dados.

**Nó raiz:** O nó raiz é o primeiro nó do qual todos os outros nós da árvore descendem. Em outras palavras, um nó raiz é um nó que não possui um nó pai. Em cada árvore, há sempre um único nó raiz. O nó raiz é o nó A na árvore do exemplo acima.

**Subárvore:** Uma subárvore é uma árvore cujos nós descendem de alguma outra árvore. Por exemplo, os nós F, K e L formam uma subárvore da árvore original.

**Grau:** O número total de filhos de um determinado nó é chamado de grau do nó. Uma árvore composta por apenas um nó tem grau 0. O grau do nó A no diagrama anterior é 2, o grau do nó B é 3, o grau do nó C é 3 e o grau do nó G é 1.

# Árvores

Fatec
Rio Claro

**Nó folha:** O nó folha não possui filhos e é o nó terminal da árvore fornecida. O grau do nó folha é sempre 0. No diagrama anterior, os nós J, E, K, L, H, M e I são todos nós folha.

**Aresta:** A conexão entre quaisquer dois nós na árvore é chamada de aresta. O número total de arestas em uma determinada árvore será, no máximo, um a menos que o total de nós na árvore. Um exemplo de aresta é mostrado na Figura 6.1.

**Pai:** Um nó que possui uma subárvore é o nó pai dessa subárvore. Por exemplo, o nó B é o pai dos nós D, E e F, e o nó F é o pai dos nós K e L.

**Filho:** Este é um nó descendente de um nó pai. Por exemplo, os nós B e C são filhos do nó pai A, enquanto os nós H, G e I são filhos do nó pai C.

**Irmão:** Todos os nós com o mesmo nó pai são irmãos. Por exemplo, o nó B é irmão do nó C e, da mesma forma, os nós D, E e F também são irmãos.

# Árvores

Fatec
Rio Claro

**Nível:** O nó raiz da árvore é considerado como estando no nível 0. Os filhos do nó raiz são considerados como estando no nível 1, e os filhos dos nós no nível 1 são considerados como estando no nível 2, e assim por diante. Por exemplo, na Figura 6.1, o nó raiz A está no nível 0, os nós B e C estão no nível 1, e os nós D, E, F, H, G e I estão no nível 2.

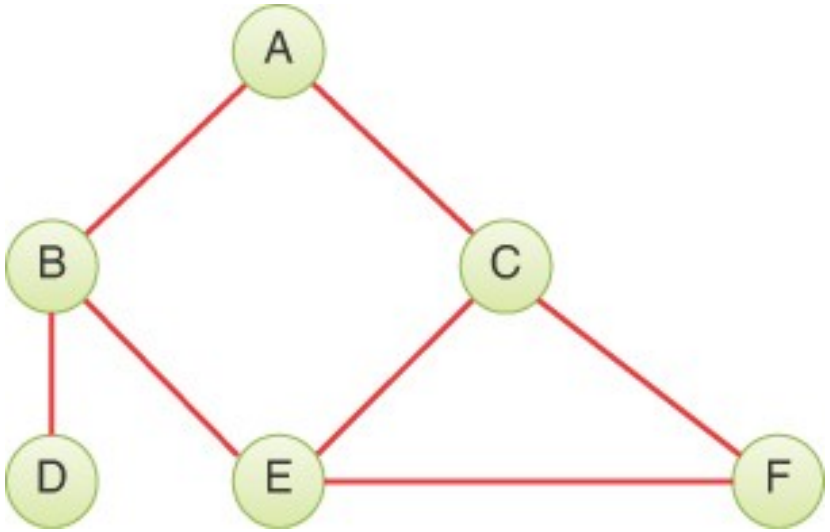
**Altura de uma árvore:** O número total de nós no caminho mais longo da árvore é a altura da árvore. Por exemplo, na Figura 6.1, a altura da árvore é 4, pois os caminhos mais longos, A-B-D-J, A-C-G-M e A-B-F-K, têm um número total de quatro nós cada.

**Profundidade:** A profundidade de um nó é o número de arestas da raiz da árvore até esse nó. No exemplo de árvore anterior, a profundidade do nó H é 2.

# Árvores

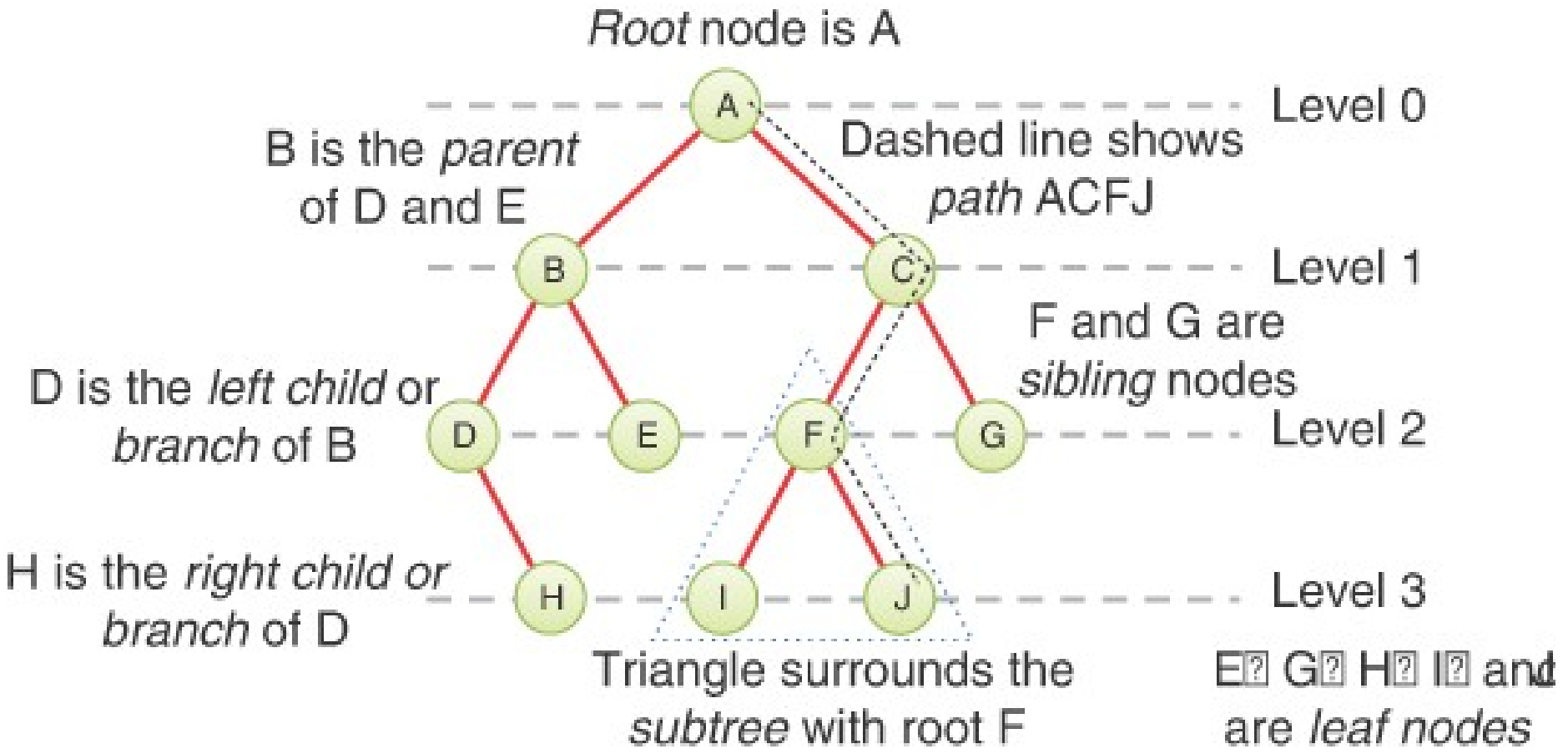
Fatec
Rio Claro

**Caminho:** Imagine alguém caminhando de um nó a outro ao longo das arestas que os conectam. A sequência resultante de nós é chamada de caminho. Para que uma coleção de nós e arestas seja definida como uma árvore, deve haver um (e apenas um!) caminho da raiz para qualquer outro nó. A figura mostra uma não-árvore. Você pode ver que ela viola essa regra porque há vários caminhos de A para os nós E e F. Este é um exemplo de um grafo que não é uma árvore.



# Árvores

Fatec
Rio Claro



# Árvores

Fatec
Rio Claro

Em estruturas de dados lineares, os itens de dados são armazenados em ordem sequencial, enquanto estruturas de dados não lineares armazenam itens de dados em uma ordem não linear, onde um item de dados pode ser conectado a mais de um outro item de dados.

Todos os itens de dados em estruturas de dados lineares, como matrizes, listas, pilhas e filas, podem ser percorridos em uma única passagem, enquanto isso não é possível no caso de estruturas de dados não lineares, como árvores; elas armazenam os dados de forma diferente de outras estruturas de dados lineares.

Em uma estrutura de dados em árvore, os nós são organizados em um relacionamento pai-filho. Não deve haver nenhum ciclo entre os nós em árvores. A estrutura em árvore possui nós para formar uma hierarquia, e uma árvore sem nós é chamada de árvore vazia.

# Uma analogia

Fatec
Rio Claro

Uma árvore comumente encontrada é o sistema de arquivos hierárquico em computadores desktop. Esse sistema foi modelado com base na tecnologia de armazenamento de documentos predominante nas empresas no século XX: arquivos contendo pastas que, por sua vez, continham subpastas, até documentos individuais.

Os sistemas operacionais de computador imitam isso, armazenando os arquivos em uma hierarquia. No topo da hierarquia está o diretório raiz.

Esse diretório contém "pastas", que são subdiretórios, e arquivos, que são como os documentos em papel. Cada subdiretório pode ter seus próprios subdiretórios e mais arquivos. Todos eles têm analogias na árvore: o diretório raiz é o nó raiz, os subdiretórios são nós com filhos e os arquivos são nós-folha.

# Uma analogia

Fatec
Rio Claro

Para especificar um arquivo específico em um sistema de arquivos, você usa o caminho completo do diretório raiz até o arquivo. Isso é o mesmo que o caminho para um nó de uma árvore.

Localizadores uniformes de recursos (URLs) usam uma construção semelhante para mostrar o caminho para um recurso na Internet.

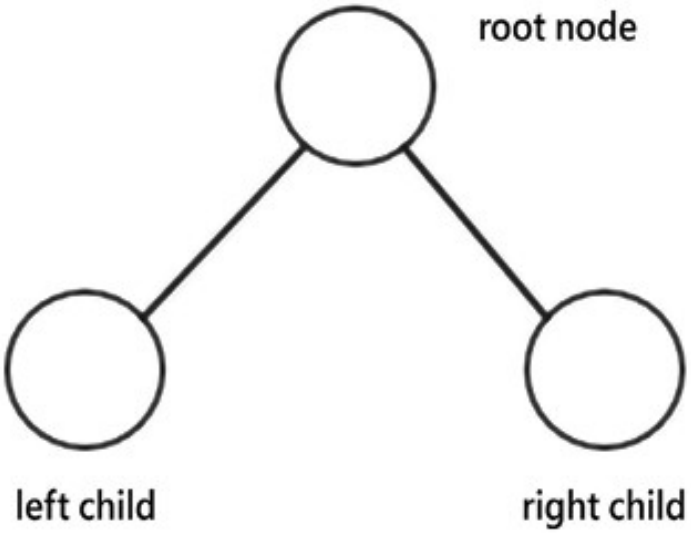
Tanto os nomes de caminho do sistema de arquivos quanto as URLs permitem vários níveis de subdiretórios. O sobrenome em um caminho de sistema de arquivos é um subdiretório ou um arquivo. Arquivos representam folhas; eles não têm filhos próprios.

# Árvores Binárias

# Árvore Binária

Fatec
Rio Claro

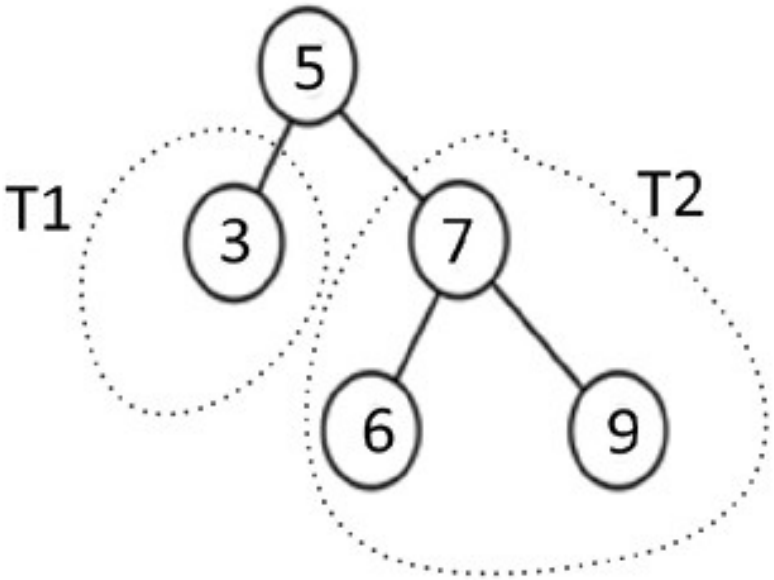
Uma árvore binária é uma coleção de nós, onde os nós na árvore podem ter zero, um ou dois nós filhos. Uma árvore binária simples tem no máximo dois filhos, ou seja, o filho da esquerda e o filho da direita.



# Árvore Binária

Fatec
Rio Claro

Os nós na árvore binária são organizados na forma de subárvore esquerda e subárvore direita. Por exemplo, uma árvore de cinco nós que possui um nó raiz, e duas subárvores, ou seja, a subárvore esquerda, T1, e a subárvore direita, T2:

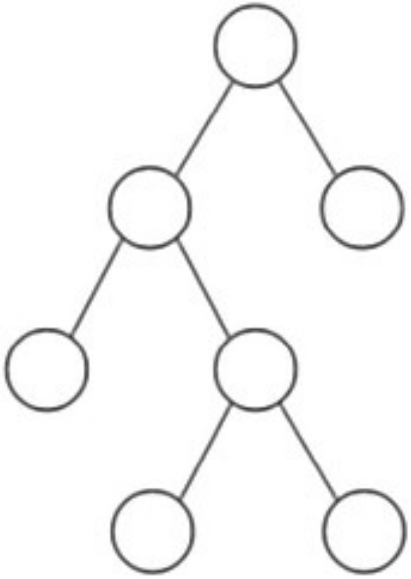


# Árvore Binária

Fatec
Rio Claro

Uma árvore binária regular não possui outras regras quanto à organização dos elementos na árvore. Ela deve satisfazer apenas a condição de que cada nó tenha no máximo dois filhos.

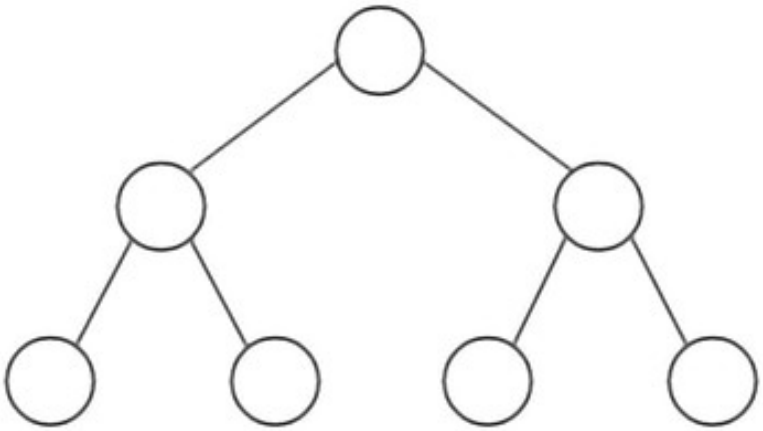
Uma árvore é chamada de *árvore binária completa* se todos os nós de uma árvore binária tiverem zero ou dois filhos, e se não houver nenhum nó com um filho.



# Árvore Binária

Fatec
Rio Claro

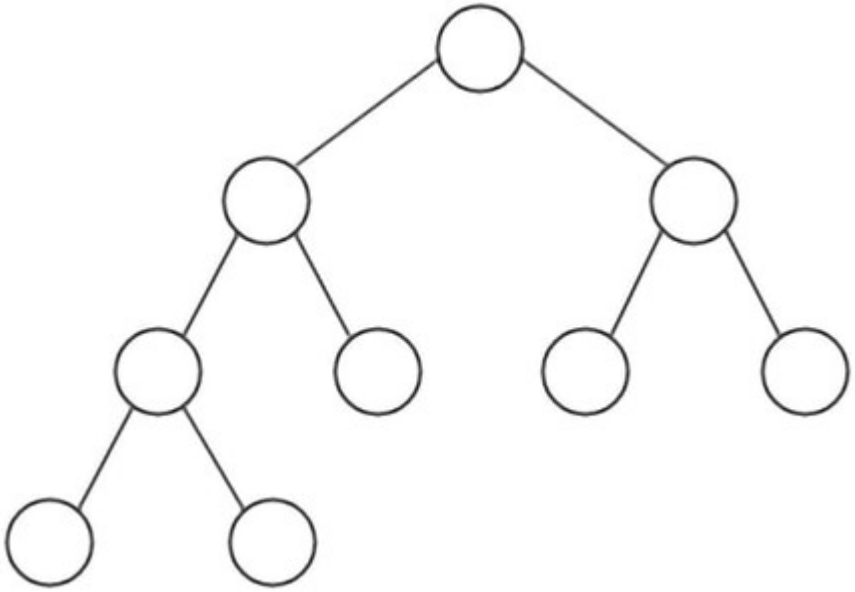
Uma *árvore binária perfeita* tem todos os nós preenchidos e não há espaço livre para novos nós; se adicionarmos novos nós, eles só poderão ser adicionados aumentando a altura da árvore.



# Árvore Binária

Fatec
Rio Claro

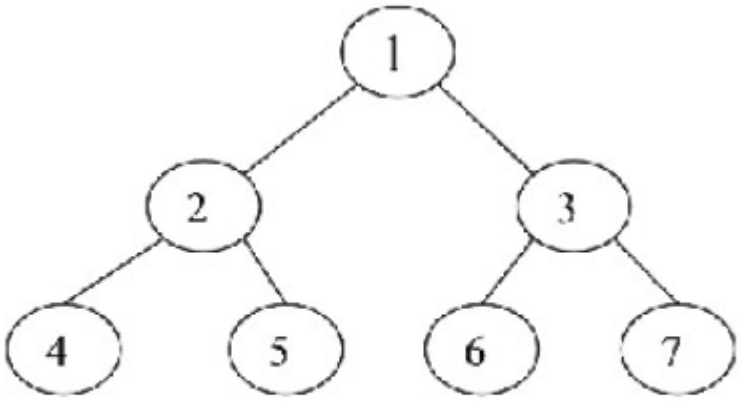
Uma árvore binária completa é preenchida com todos os nós possíveis, exceto com uma possível exceção no nível mais baixo da árvore. Todos os nós também são preenchidos no lado esquerdo



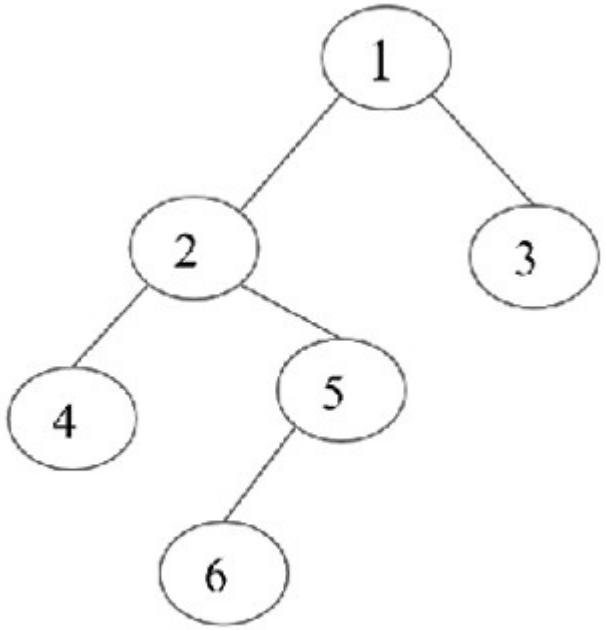
# Árvore Binária

Fatec
Rio Claro

Uma árvore binária pode ser balanceada ou desbalanceada. Em uma árvore binária balanceada, a diferença de altura entre as subárvores esquerda e direita de cada nó da árvore não é maior que 1




Balanced tree




Unbalanced tree

# Implementação

# Implementação Árvore Binária

Fatec Rio Claro

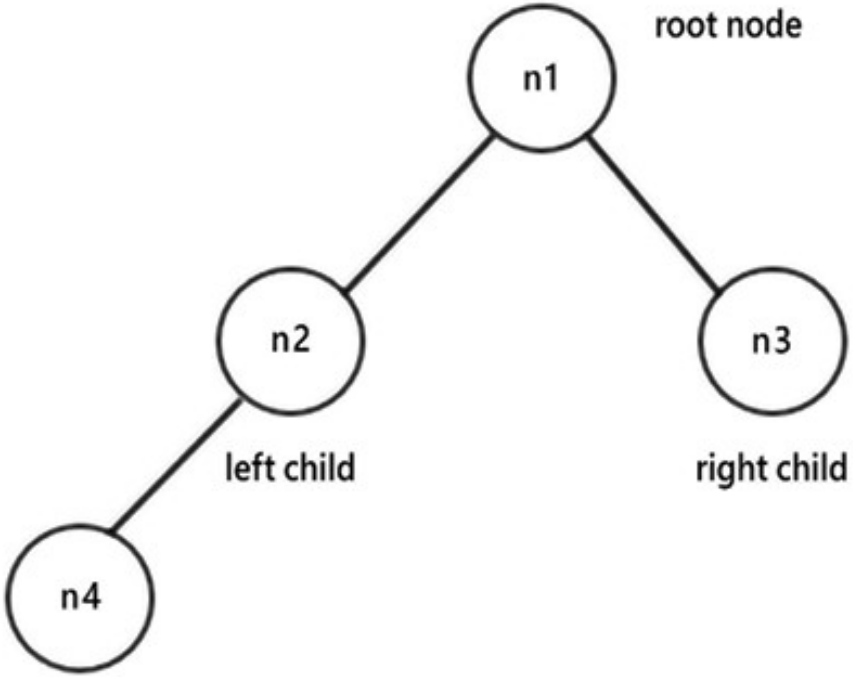
Em um nó de árvore binária, cada nó conterá itens de dados e duas referências que apontarão para seus filhos esquerdo e direito, respectivamente. Vejamos o seguinte código para construir uma classe Node de árvore binária em Python.

```
class Node:
    def __init__(self, data):
        self.data = data
        self.right_child = None
        self.left_child = None
```

# Implementação Árvore Binária

Fatec Rio Claro

Para entender melhor o funcionamento desta classe, vamos primeiro criar uma árvore binária de quatro nós — n1, n2, n3 e n4



# Implementação Árvore Binária

Fatec Rio Claro

Para isso, primeiro criamos quatro nós — n1, n2, n3 e n4:

```
n1 = Node("root node")
n2 = Node("left child node")
n3 = Node("right child node")
n4 = Node("left grandchild node")
```

Em seguida, conectamos os nós entre si de acordo com as propriedades de uma árvore binária discutidas anteriormente. n1 será o nó raiz, com n2 e n3 como seus filhos. Além disso, n4 será o filho esquerdo de n2.

```
n1.left_child = n2
n1.right_child = n3
n2.left_child = n4
```

# Implementação Árvore Binária

Fatec Rio Claro

O método para visitar todos os nós de uma árvore é chamado de **travessia em árvore**. No caso de uma estrutura de dados linear, a travessia dos elementos de dados é simples, pois todos os itens são armazenados sequencialmente, de modo que cada item de dados é visitado apenas uma vez.

No entanto, no caso de estruturas de dados não lineares, como árvores e grafos, os algoritmos de travessia são importantes. Para entender o percurso, vamos percorrer a subárvore esquerda da árvore binária que criamos na seção anterior.

Para isso, começamos pelo nó raiz, imprimimos o nó e descemos na árvore até o próximo nó à esquerda. Continuamos fazendo isso até chegarmos ao final da subárvore esquerda, assim:

```
current = n1
while current:
    print(current.data)
    current = current.left_child
```

# Implementação Árvore Binária

Fatec Rio Claro

Existem várias maneiras de processar e percorrer a árvore, dependendo da sequência de visitas ao nó raiz, à subárvore esquerda ou à subárvore direita.

Basicamente, existem dois tipos de abordagem: primeiro, uma em que partimos de um nó e percorremos todos os nós filhos disponíveis e, em seguida, continuamos a percorrer até o próximo irmão.

Há três variações possíveis desse método: **em ordem**, **pré-ordem** e **pós-ordem**. Outra abordagem para percorrer a árvore é começar pelo nó raiz e, em seguida, visitar todos os nós em cada nível e processar os nós nível por nível.

# Percurso em ordem

Fatec
Rio Claro

O percurso em ordem da árvore funciona da seguinte forma: começamos percorrendo a subárvore esquerda recursivamente e, uma vez que a subárvore esquerda é visitada, o nó raiz é visitado e, finalmente, a subárvore direita é visitada recursivamente. Ele tem as seguintes três etapas:

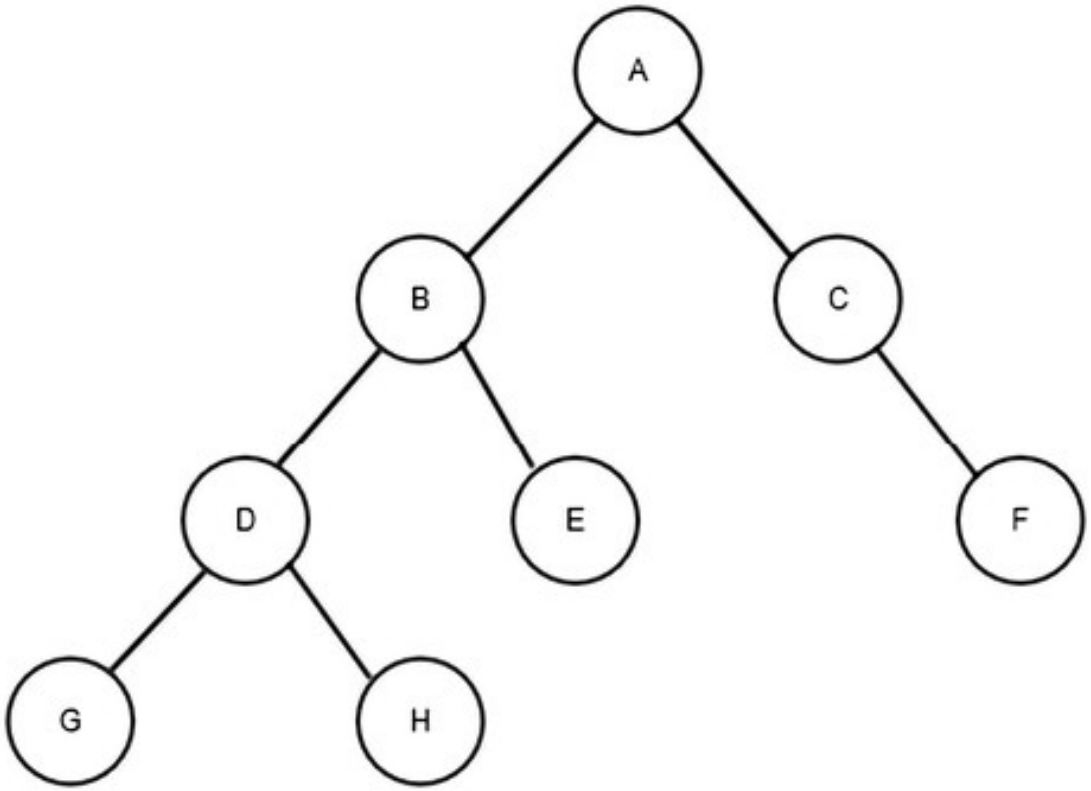
- Começamos percorrendo a subárvore esquerda e chamamos uma função de ordenação recursivamente
- Em seguida, visitamos o nó raiz
- Finalmente, percorremos a subárvore direita e chamamos uma função de ordenação recursivamente

Portanto, em resumo, para o percurso em ordem da árvore, visitamos os nós da árvore na ordem da subárvore esquerda, raiz e, em seguida, a subárvore direita.

# Percurso em ordem

Fatec
Rio Claro

G - D - H - B - E - A - C - F



# Percurso de pré-ordem

Fatec
Rio Claro

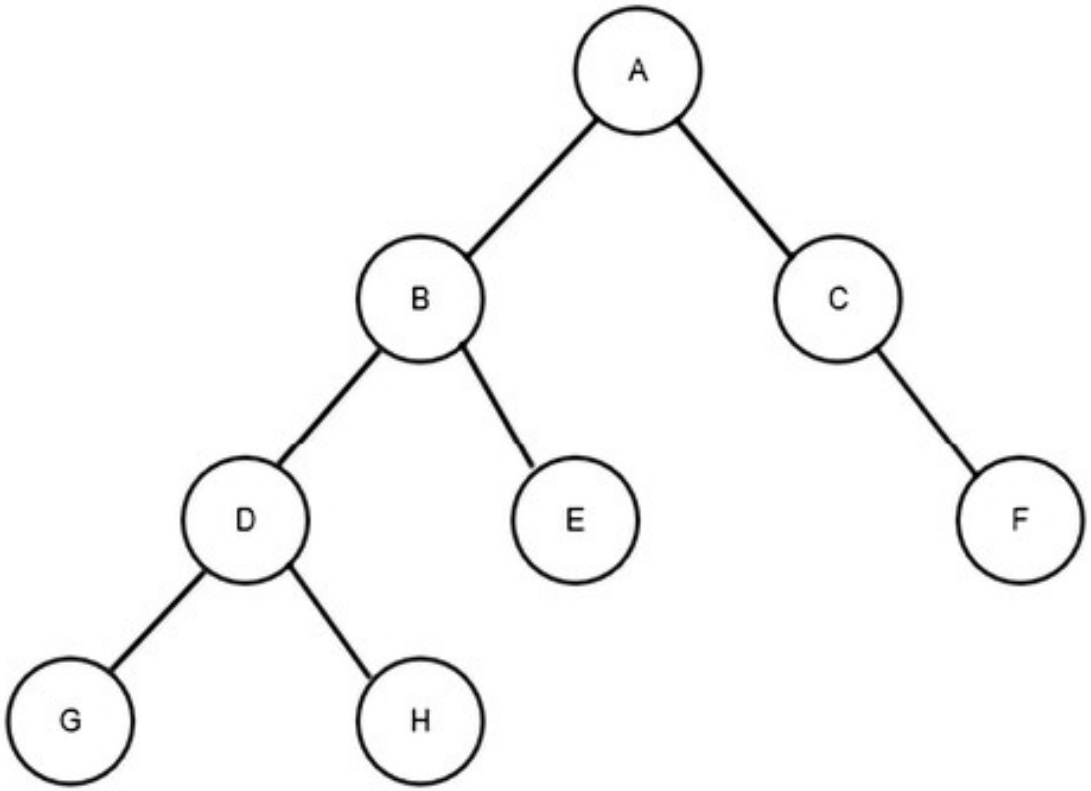
O percurso de pré-ordem da árvore percorre a árvore na ordem do nó raiz, da subárvore esquerda e, em seguida, da subárvore direita. Funciona da seguinte forma:

- Começamos o percurso pelo nó raiz.
- Em seguida, percorremos a subárvore esquerda e chamamos uma função de ordenação com a subárvore esquerda recursivamente.
- Em seguida, visitamos a subárvore direita e chamamos uma função de ordenação com a subárvore direita recursivamente.

# Percurso de pré-ordem

Fatec
Rio Claro

A – B – D – G – H – E – C - F



# Percurso pós-ordem

Fatec
Rio Claro

O percurso pós-ordem da árvore funciona da seguinte forma:

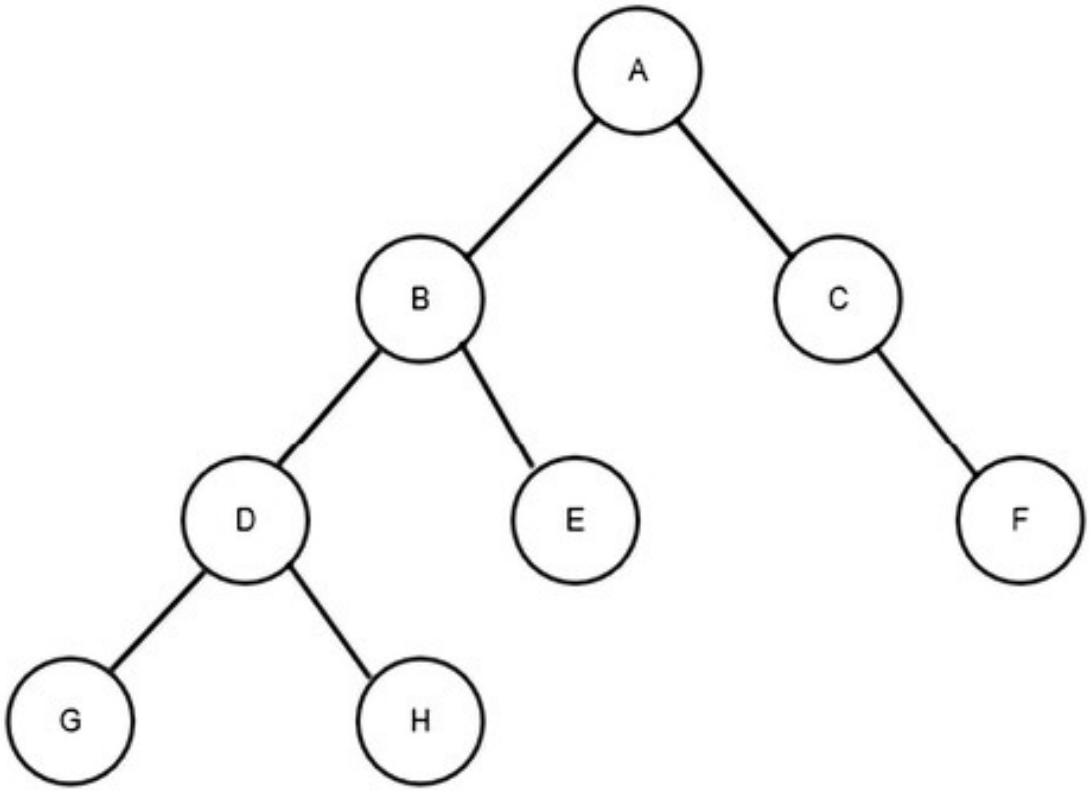
- Começamos percorrendo a subárvore esquerda e chamamos uma função de ordenação recursivamente.
- Em seguida, percorremos a subárvore direita e chamamos uma função de ordenação recursivamente.
- Finalmente, visitamos o nó raiz.

Portanto, em resumo, para o percurso pós-ordem da árvore, visitamos os nós da árvore na ordem da subárvore esquerda, da subárvore direita e, finalmente, do nó raiz.

# Percurso de pré-ordem

Fatec
Rio Claro

G – H – D – E – B – F – C - A



# Percurso em ordem de nível

Fatec
Rio Claro

Neste método de percurso, começamos visitando a raiz da árvore antes de visitar todos os nós do próximo nível da árvore.

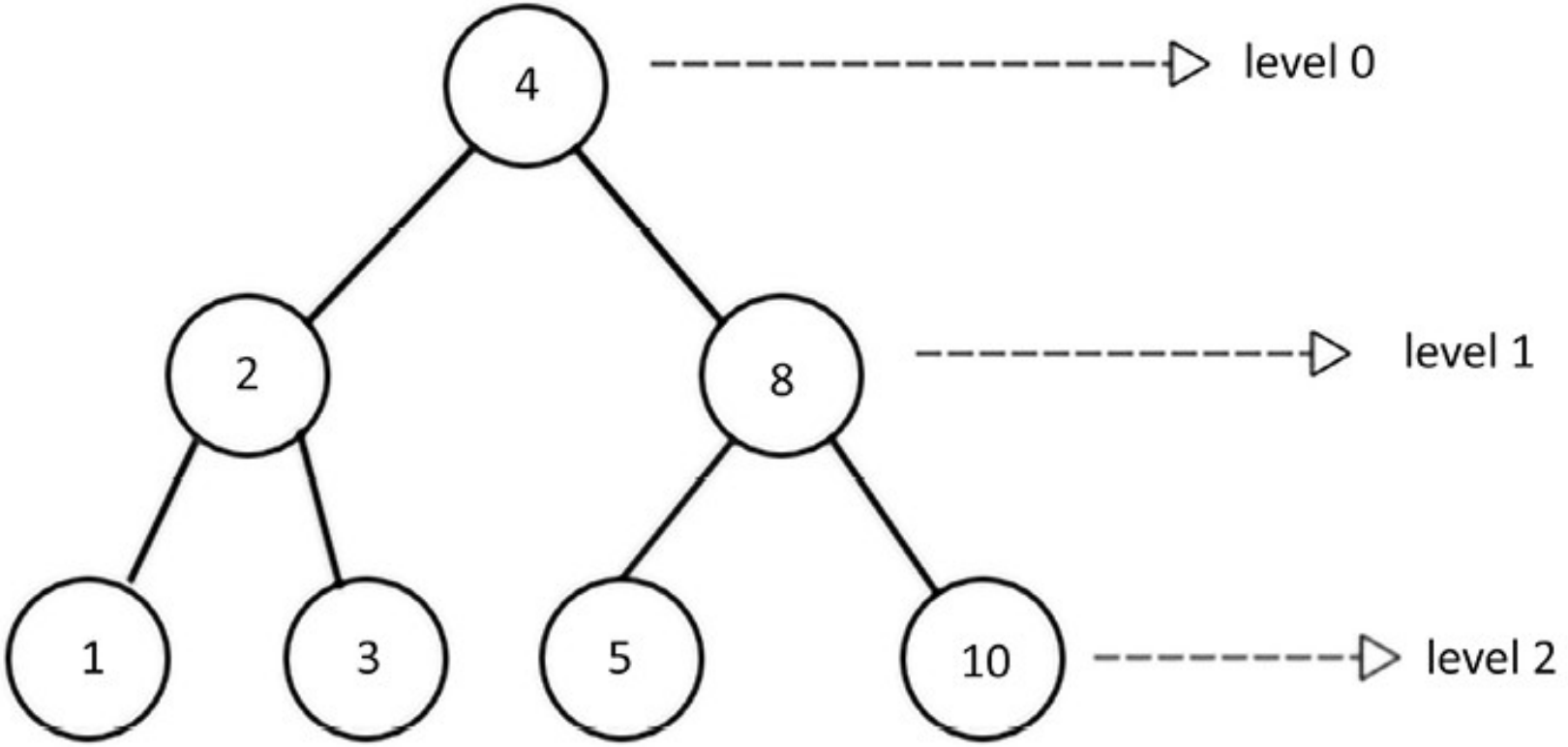
Em seguida, passamos para o próximo nível da árvore e assim por diante.

Esse tipo de percurso em árvore é como o percurso em largura funciona em um grafo, pois amplia a árvore percorrendo todos os nós de um nível antes de se aprofundar na árvore.

# Percurso em ordem de nível

Fatec
Rio Claro

4 - 2 - 8 - 1 - 3 - 5 - 10



# Percurso em ordem de nível

Fatec
Rio Claro

Neste método de percurso, começamos visitando a raiz da árvore antes de visitar todos os nós do próximo nível da árvore.

Em seguida, passamos para o próximo nível da árvore e assim por diante.

Esse tipo de percurso em árvore é como o percurso em largura funciona em um grafo, pois amplia a árvore percorrendo todos os nós de um nível antes de se aprofundar na árvore.

# Percurso em ordem de nível

Fatec
Rio Claro

Essa travessia em árvore em ordem de nível é implementada usando uma estrutura de dados de fila. Começamos visitando o nó raiz e o colocamos em uma fila. O nó na frente da fila é acessado (retirado da fila), podendo então ser impresso ou armazenado para uso posterior. Após adicionar o nó raiz, o nó filho da esquerda é adicionado à fila, seguido pelo nó da direita.

Assim, ao percorrer qualquer nível da árvore, todos os itens de dados desse nível são primeiramente inseridos na fila, da esquerda para a direita. Depois disso, todos os nós da fila são visitados, um por um. Esse processo é repetido para todos os níveis da árvore.

# Prática

Fatec
Rio Claro

Analisar o código ***level_order_traversal.py***

# Árvores binárias de busca

# Árvores Binárias de Buscas

Fatec Rio Claro

Uma árvore binária de busca (BST) é um tipo especial de árvore binária. É uma das estruturas de dados mais importantes e comumente utilizadas em aplicações de ciência da computação.

Uma árvore binária de busca é uma árvore que é estruturalmente uma árvore binária e armazena dados em seus nós de forma muito eficiente.

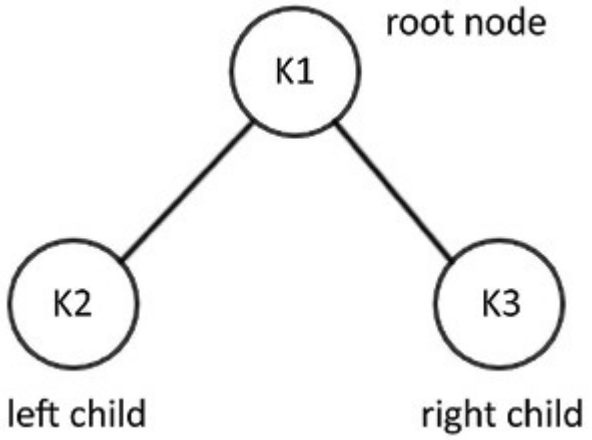
Ela fornece operações de busca, inserção e exclusão muito rápidas.

# Árvores Binárias de Buscas

Fatec Rio Claro

Uma árvore binária é chamada de árvore binária de busca se o valor em qualquer nó da árvore for maior que os valores em todos os nós de sua subárvore esquerda e menor que (ou igual a) os valores de todos os nós da subárvore direita.
Por exemplo, se K1, K2 e K3 são valores-chave em uma árvore de três nós , então ela deve satisfazer as seguintes condições:

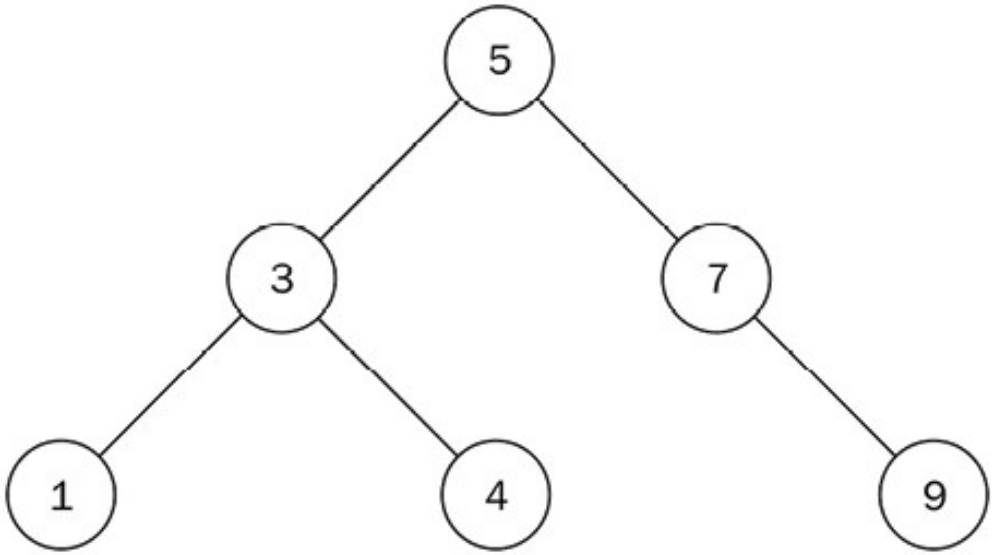
Os valores-chave K2 <= K1
Os valores-chave  K3 > K1



# Árvores Binárias de Buscas

Fatec Rio Claro

Vamos considerar outro exemplo para entender melhor as árvores de busca binárias. Nesta árvore, todos os nós na subárvore esquerda são menores que (ou iguais) o valor do nó pai. Todos os nós na subárvore direita deste nó também são maiores que o valor do nó pai.



# Operações em Árvore Binária de Busca

Fatec
Rio Claro

As operações que podem ser realizadas em uma árvore de busca binária são inserir, excluir, encontrar o mínimo, encontrar o máximo e pesquisar.

# Inserir nós

Fatec
Rio Claro

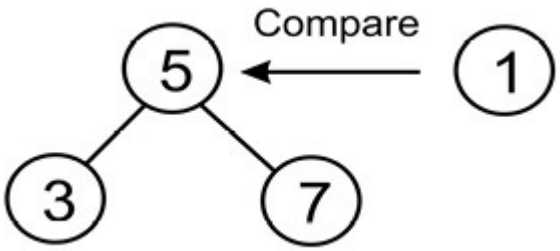
Uma das operações mais importantes a serem implementadas em uma árvore de busca binária é inserir itens de dados na árvore.

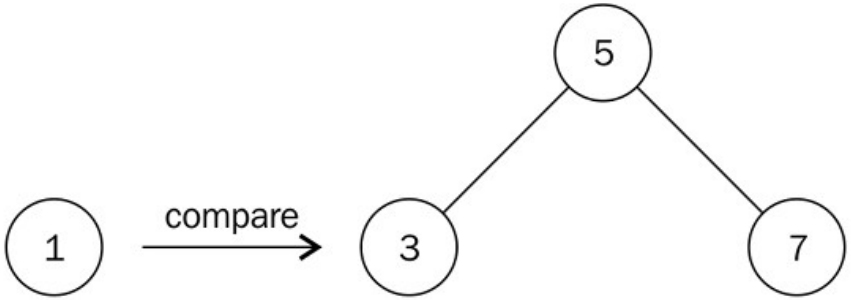
Para inserir um novo elemento em uma árvore de busca binária, precisamos garantir que as propriedades da árvore de busca binária não sejam violadas após a adição do novo elemento.

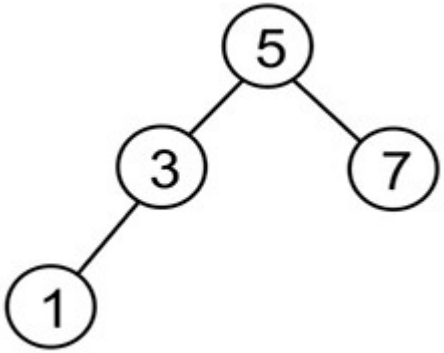
Para inserir um novo elemento, começamos comparando o valor do novo nó com o do nó raiz: se o valor for menor que o valor raiz, o novo elemento será inserido na subárvore da esquerda; caso contrário, será inserido na subárvore da direita. Dessa forma, vamos até o final da árvore para inserir o novo elemento.

# Inserir nós

Fatec
Rio Claro



# Pesquisando Nós

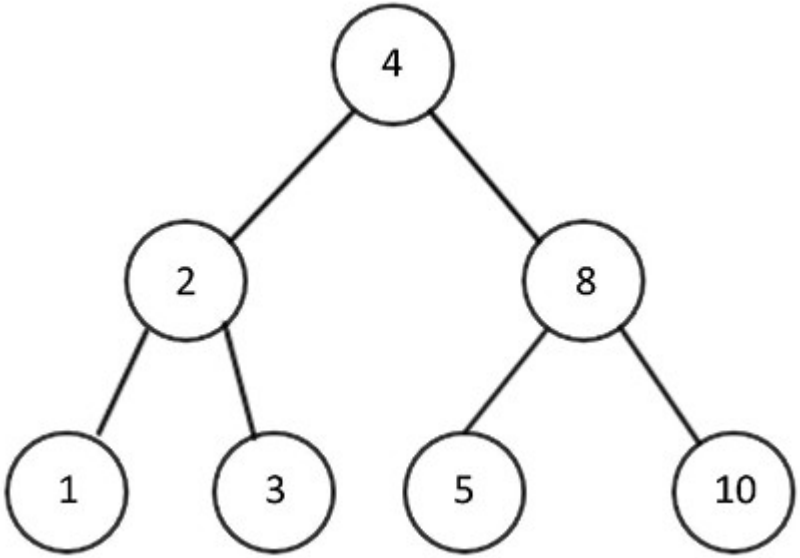
Fatec
Rio Claro

Uma árvore de busca binária é uma estrutura de dados em árvore na qual todos os nós na subárvore esquerda de um nó têm valores-chave mais baixos e a subárvore direita tem valores-chave mais altos.

Assim, pesquisar um elemento com um determinado valor-chave é bastante fácil. Vamos considerar um exemplo de árvore de busca binária com os nós 1, 2, 3, 4, 8, 5 e 10,

# Pesquisando Nós

Fatec
Rio Claro



# Pesquisando Nós

Fatec
Rio Claro

Na árvore anterior, se desejarmos procurar um nó com valor 5, por exemplo, partimos do nó raiz e comparamos a raiz com o valor desejado.

Como o nó 5 tem um valor maior que o valor 4 do nó raiz, passamos para a subárvore direita.

Na subárvore direita, temos o nó 8 como nó raiz, então comparamos o nó 5 com o nó 8. Como o nó a ser procurado tem um valor menor que o nó 8, o movemos para a subárvore esquerda.

Quando passamos para a subárvore esquerda, comparamos o nó 5 da subárvore esquerda com o valor 5 do nó necessário. Isso é uma correspondência, então retornamos "item found".

# Excluindo Nós

Fatec
Rio Claro

Outra operação importante em uma árvore de busca binária é a exclusão ou remoção de nós. Há três cenários possíveis que precisamos considerar durante esse processo. O nó que queremos remover pode ter o seguinte:

**Sem filhos:** Se não houver nó folha, remova-o diretamente.

**Um filho:** Neste caso, trocamos o valor desse nó com seu filho e, em seguida, excluímos o nó.

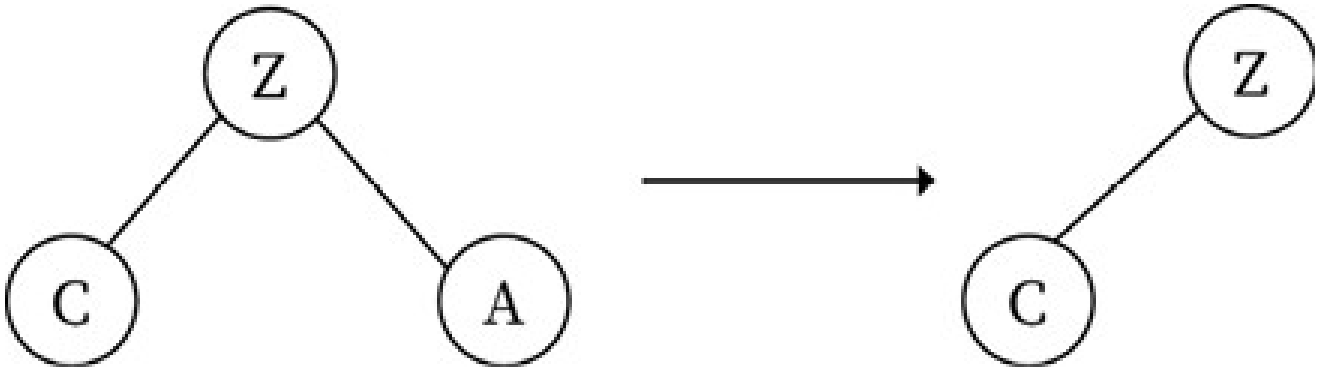
**Dois filhos:** Neste caso, primeiro encontramos o sucessor ou predecessor em ordem, trocamos seus valores e, em seguida, excluímos esse nó.

# Excluindo Nós

Fatec Rio Claro

O primeiro cenário é o mais fácil de lidar. Se o nó a ser removido não tiver filhos, podemos simplesmente removê-lo de seu pai.

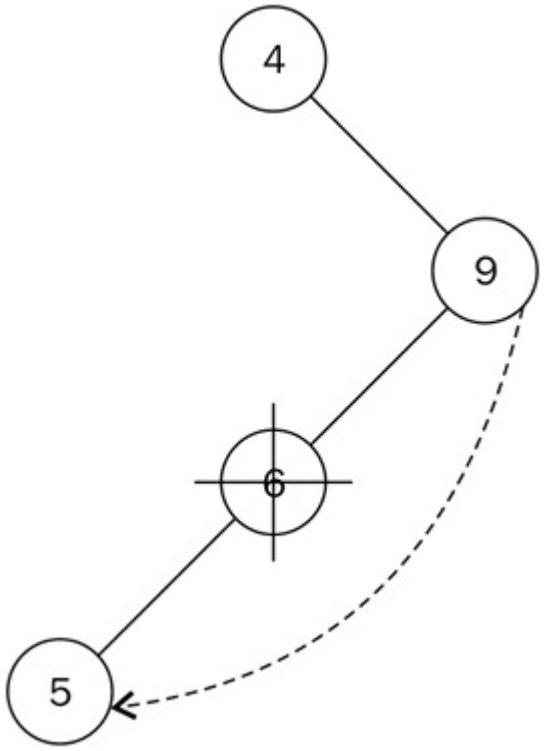
Suponha que queremos excluir o nó A, que não tem filhos. Nesse caso, podemos simplesmente excluí-lo de seu pai (nó Z):



# Excluindo Nós

Fatec Rio Claro

No segundo cenário, quando o nó que queremos remover tem um filho, o pai desse nó aponta para o filho daquele nó específico. Vejamos o diagrama a seguir:



# Excluindo Nós

Fatec
Rio Claro

No terceiro cenário, quando o nó que queremos excluir tem dois filhos, para excluí-lo, primeiro encontramos um nó sucessor e, em seguida, movemos o conteúdo do nó sucessor para o nó a ser excluído.

O nó sucessor é o nó que possui o menor valor na subárvore direita do nó a ser excluído; ele será o primeiro elemento quando aplicarmos a travessia em ordem na subárvore direita do nó a ser excluído.

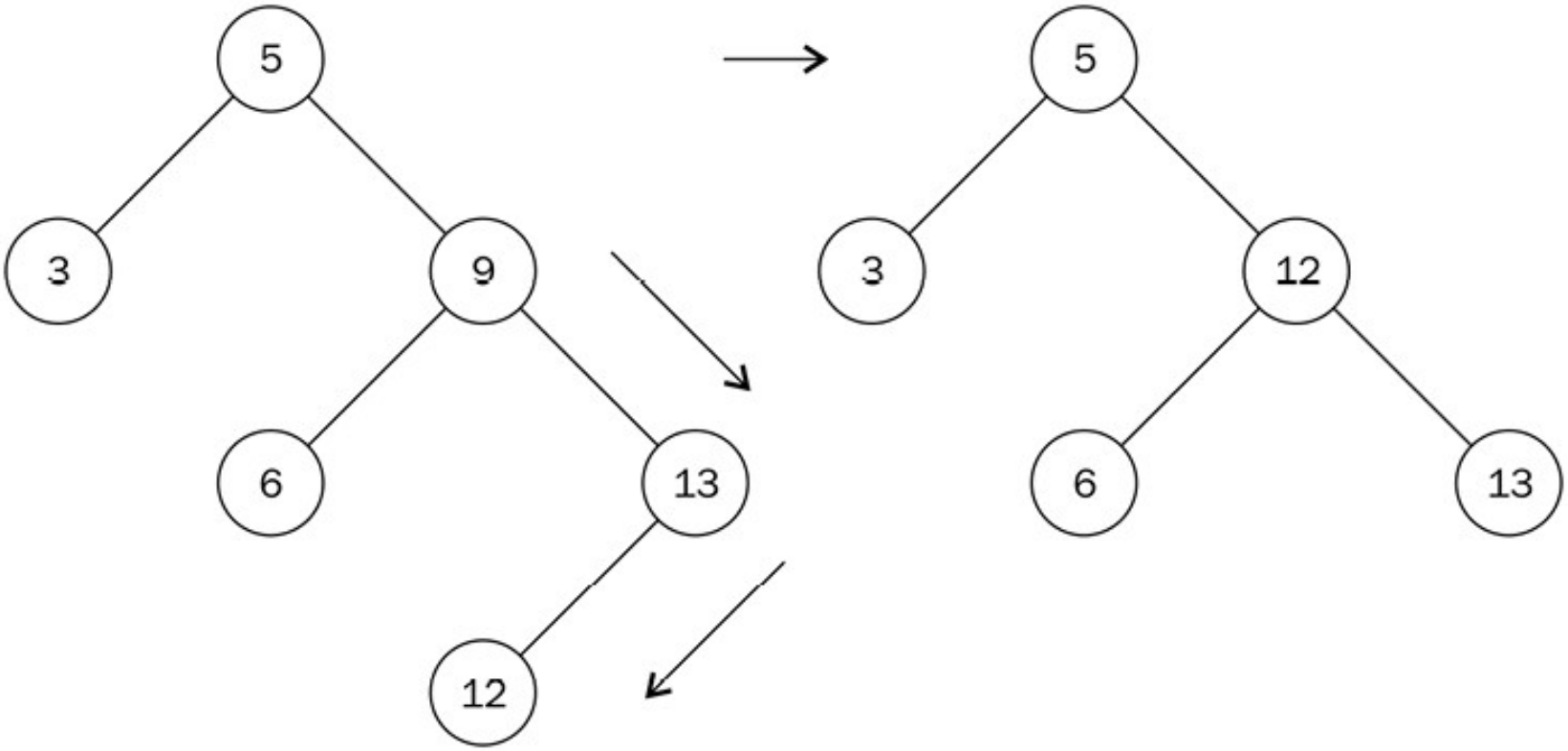
Vamos entender isso com a árvore de exemplo mostrada a seguir, onde queremos excluir o nó 9, que tem dois filhos:

Na árvore de exemplo, encontramos o menor elemento na subárvore direita do nó (ou seja, o primeiro elemento na travessia em ordem na subárvore direita), que é o nó 12.

Depois disso, substituímos o valor do nó 9 pelo valor 12 e removemos o nó 12. O nó 12 não tem filhos, então aplicamos a regra para remover nós sem filhos adequadamente.

# Excluindo Nós

Fatec
Rio Claro



# Encontrando Nós Mínimos e Máximos

Fatec Rio Claro

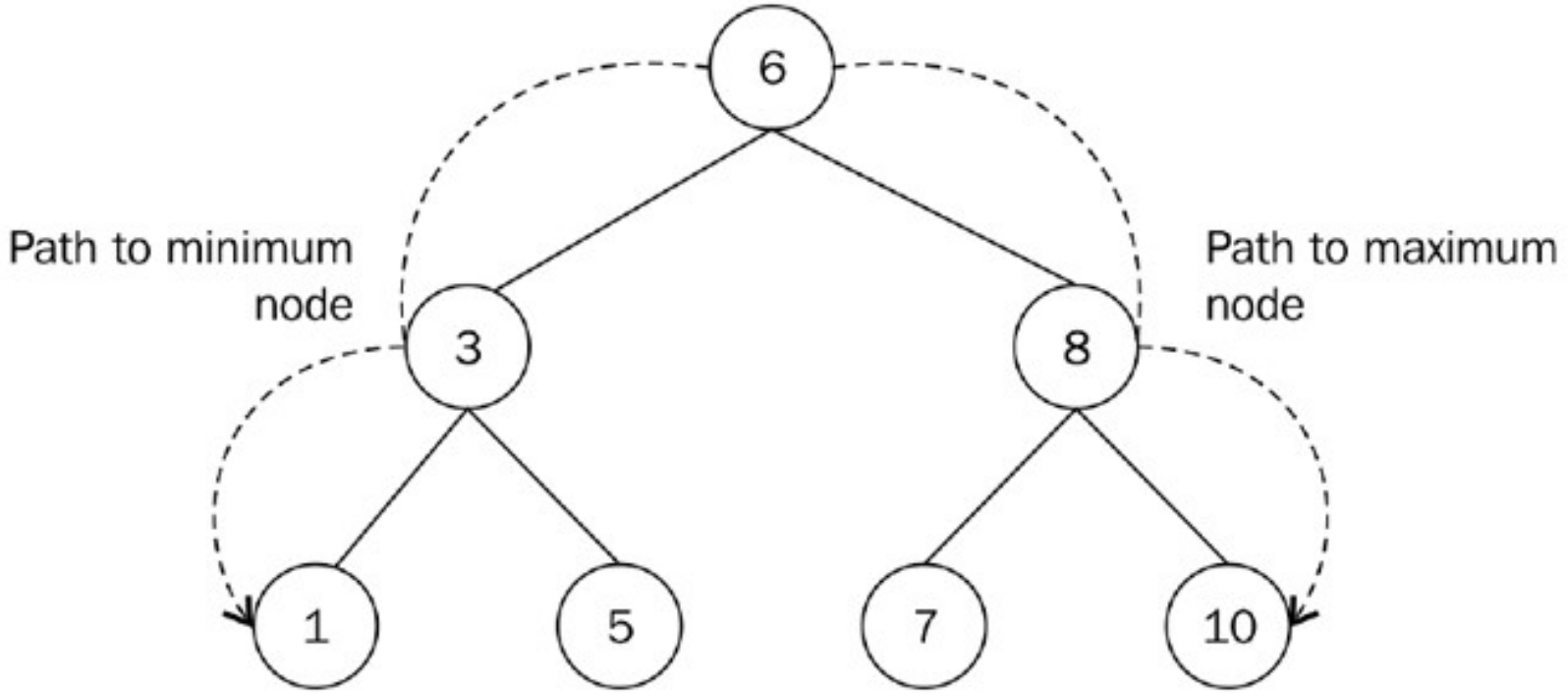
A estrutura da árvore de busca binária facilita muito a busca por um nó que tenha um valor máximo ou mínimo.

Para encontrar um nó com o menor valor na árvore, iniciamos o percurso a partir da raiz da árvore e visitamos o nó esquerdo a cada vez, até chegarmos ao final da árvore.

Da mesma forma, percorremos a subárvore direita recursivamente até chegarmos ao final para encontrar o nó com o maior valor na árvore.

# Encontrando Nós Mínimos e Máximos

Fatec
Rio Claro



# Benefícios de uma árvore de busca binária

# Benefícios de uma árvore de busca binária

Fatec
Rio Claro

Uma árvore de busca binária é, em geral, uma escolha melhor em comparação com matrizes e listas encadeadas quando estamos mais interessados em acessar os elementos com frequência em qualquer aplicação.

Uma árvore de busca binária é rápida para a maioria das operações, como busca, inserção e exclusão, enquanto matrizes proporcionam busca rápida, mas são comparativamente lentas em relação às operações de inserção e exclusão.

De forma semelhante, listas encadeadas são eficientes na execução de operações de inserção e exclusão, mas são mais lentas na execução da operação de busca.

A complexidade de tempo de execução no melhor caso para buscar um elemento em uma árvore de busca binária é **O(log n)**, e a complexidade de tempo no pior caso é **O(n)**, enquanto a complexidade de tempo tanto no melhor quanto no pior caso para buscar em listas é **O(n)**.

# Síntese

# Síntese

Fatec
Rio Claro

Estruturas de dados em árvore em geral fornecem melhor desempenho em comparação com estruturas de dados lineares em operações de busca, inserção e exclusão. Também discutimos como aplicar várias operações a estruturas de dados em árvore.

Estudamos árvores binárias, que podem ter no máximo dois filhos para cada nó. Além disso, aprendemos sobre árvores de busca binárias e discutimos como podemos aplicar diferentes operações a elas.

Árvores de busca binárias são muito úteis quando queremos desenvolver uma aplicação do mundo real na qual a recuperação ou busca de elementos de dados é uma operação importante. Precisamos garantir que a árvore esteja balanceada para o bom desempenho da árvore de busca binária.

Fatec
Rio Claro

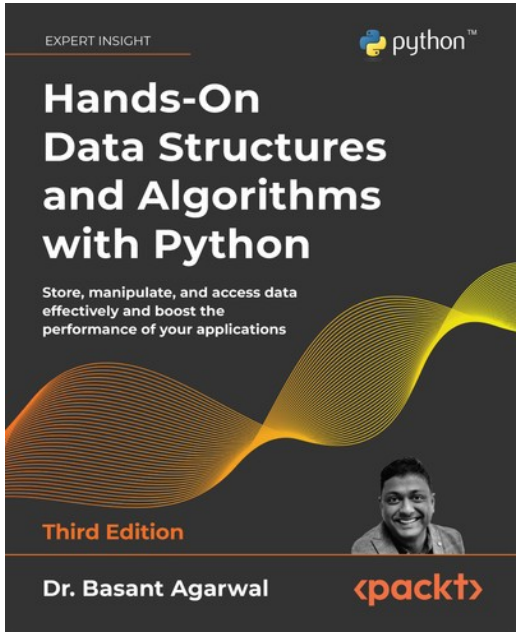
# Dúvidas

**Prof. Orlando Saraiva Júnior**
orlando.nascimento@fatec.sp.gov.br

# Prática

# Bibliografia

AGARWAL, Basant. Hands-On Data Structures and Algorithms with Python. 3. ed. Birmingham: Packt Publishing, 2022.



# Bibliografia

CANNING, John; BRODER, Alan; LAFORE, Robert. Data structures & algorithms in Python. Boston: Addison-Wesley Professional, 2019.