

N_EST DAD_A7 – Texto de Apoio

Site: [EAD Mackenzie](#)

Tema: ESTRUTURA DE DADOS {TURMA 03A} 2023/1

Livro: N_EST DAD_A7 – Texto de Apoio

Impresso por: FELIPE BALDIM GUERRA .

Data: segunda, 1 mai 2023, 02:13

Índice

TAD ÁRVORE

Formas de Representação de Árvores

Árvores Binárias

Percursos em Árvores Binárias

Árvores Binárias – Representação Gráfica

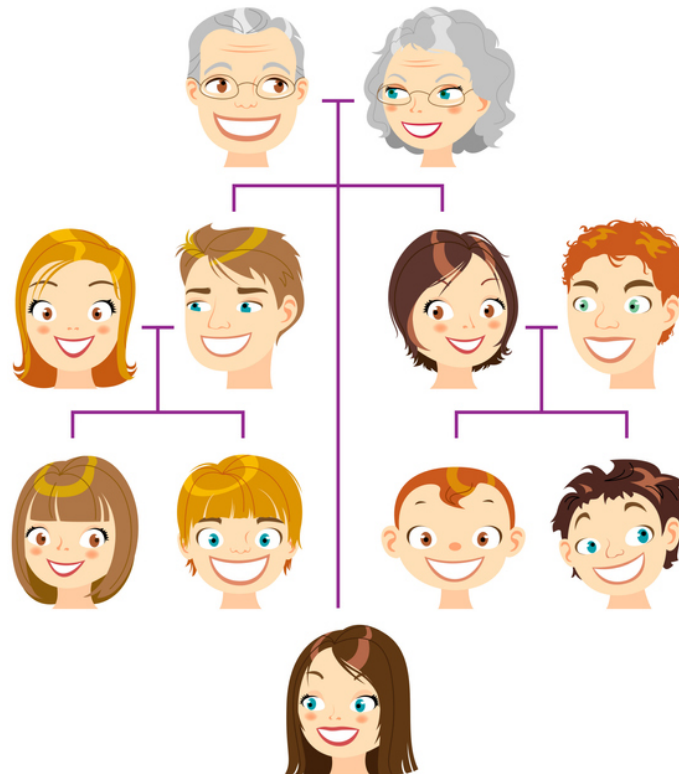
Árvores Binárias de Busca

TAD ÁRVORE

Conceituação e Terminologia

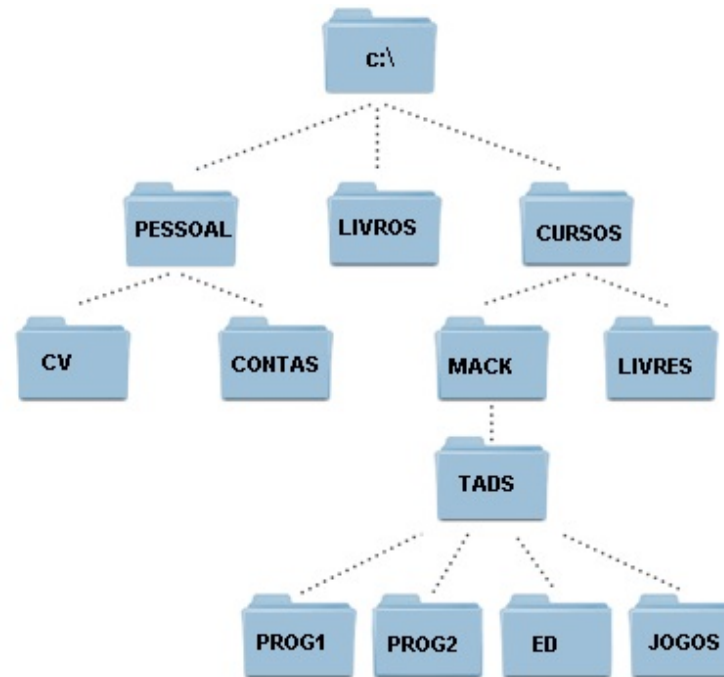
A árvore é uma estrutura de dados não linear e, por esse motivo, é possível fazer uma movimentação e recuperação dos dados de forma **hierárquica**, com elementos posicionados “acima” e outros, “abaixo”. Toda a terminologia que utilizaremos são originadas das conhecidas árvores genealógicas: pai, filho, ancestral, descendente etc.

Figura 1 – Árvore genealógica



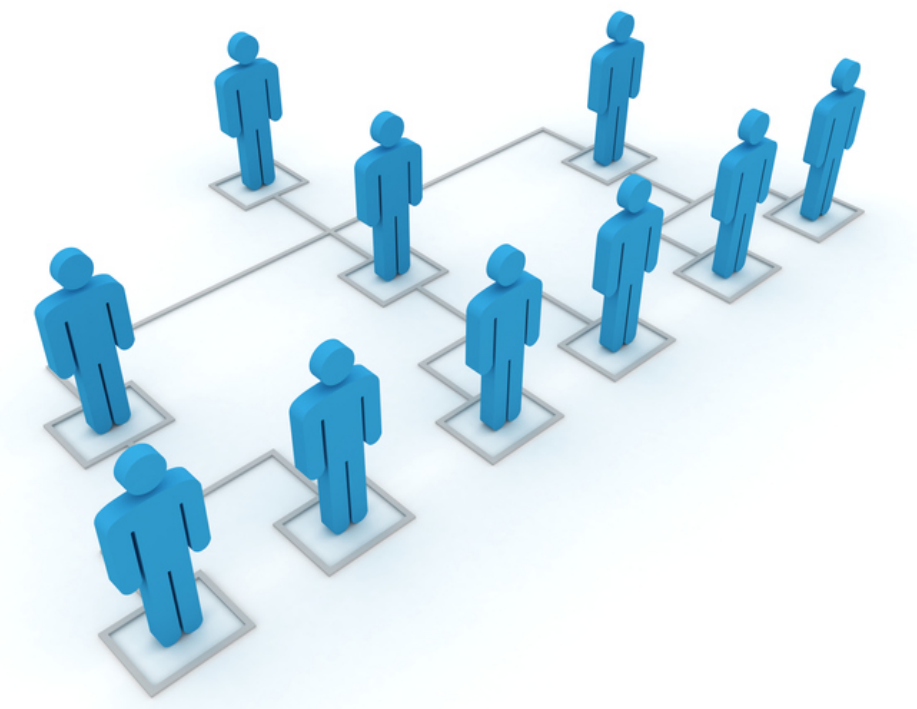
Repare que, na Figura 1, os elementos de uma família são representados em uma espécie de árvore: ou seja, as famílias são como os galhos de uma árvore. Com certeza, você já conhece outras estruturas hierárquicas e sabe como compreendê-las. Veja alguns exemplos:

Figura 2 – Árvore de diretórios



Fonte: Elaborada pela autora.

Figura 3 – Organograma



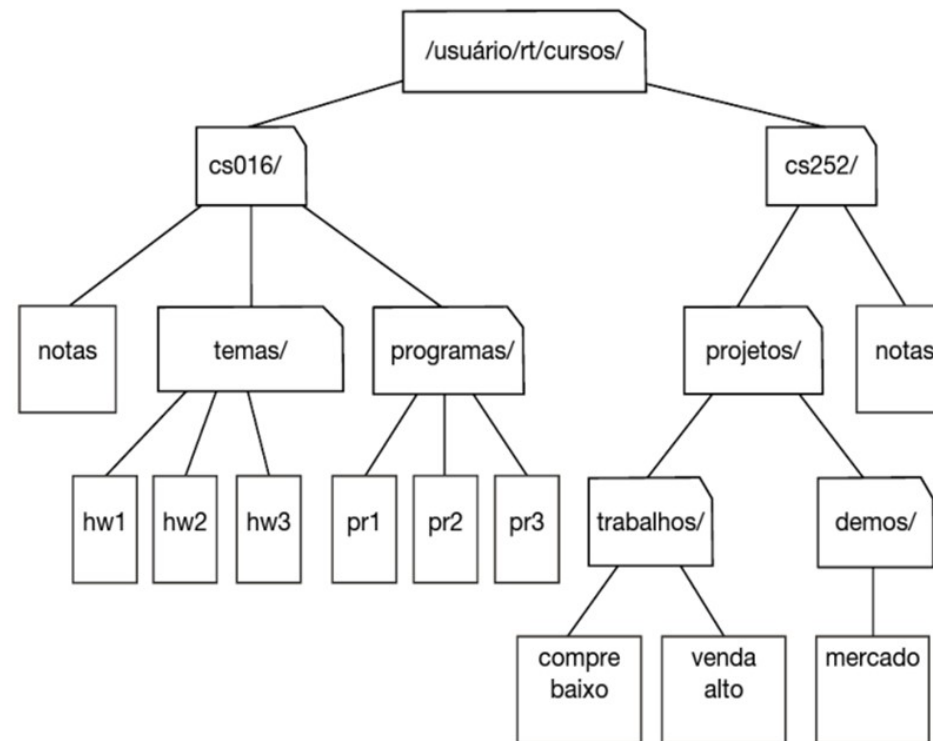
TERMINOLOGIA

- Cada elemento da árvore recebe o nome de **NÓ**.
- Se a árvore não estiver vazia, ela tem um nó especial chamado **RAIZ**, que não tem pai.
- Com exceção da **RAIZ**, cada **NÓ** da árvore tem um elemento **PAI** e nenhum ou mais elementos **FILHOS**.
- O nó **RAIZ** de uma árvore é o primeiro elemento a ser representado, e os demais **NÓS** ficam conectados abaixo (o contrário de uma árvore real).
- Pais e filhos são ligados por **ARESTAS**.
- Dois nós que são filhos do mesmo pai são chamados de **IRMÃOS**. Um nó é chamado de **EXTERNO** ou **FOLHA** se não tem filhos, e é chamado de **INTERNO** se tem um ou mais filhos.
- O nó **RAIZ** é um nó especial e não é considerado **FOLHA** nem **INTERNO**.
- A **PROFUNDIDADE** (nível) de um nó **V** qualquer em uma árvore é determinada pelo número de arestas no caminho da raiz até o nó **V**. A profundidade (nível) da raiz é definida como 0.
- A **ALTURA** de uma árvore é a maior profundidade encontrada entre seus nós.

- Uma **FLORESTA** é o conjunto de árvores disjuntas.

EXEMPLO:

Figura 4 – Exemplo de árvore



Fonte: GOODRICH; TAMASSIA (2013, p. 286).

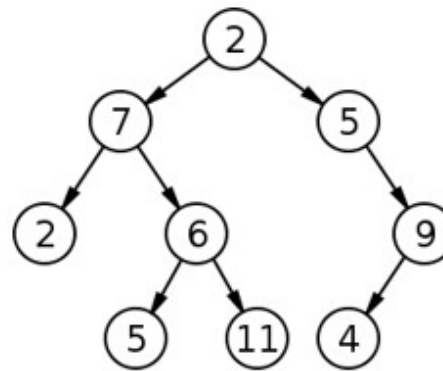
- RAIZ: /usuário/rt/cursos
- NÓS FOLHA/EXTERNO: notas – hw1 – hw2 – hw3 – pr1 – pr2 – pr3 – compre baixo – venda alto – mercado – notas
- NÓS INTERNOS: cs016/ - temas/ - programas/ -cs252/ - projetos/ - trabalhos/ - demos/
- PAI DE MERCADO: demos/

- PROFUNDIDADE (NÍVEL) DO NÓ /CS252 E MERCADO: 1 e 4, respectivamente.
- ALTURA DA ÁRVORE: 4

Formas de Representação de Árvores

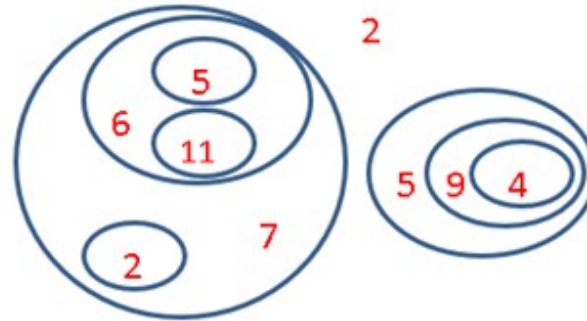
Existem, basicamente, três formas diferentes de se representar o conteúdo de uma árvore. Em nosso curso, adotaremos a representação hierárquica. Veja, abaixo, uma mesma árvore sendo representada de três formas distintas: hierárquica, diagramas de inclusão e parêntesis aninhados.

Figura 5 – Representação hierárquica de uma árvore



Fonte: Elaborada pela autora.

Figura 6 – Representação de uma árvore por Diagrama de Inclusão



Fonte: Elaborada pela autora.

Figura 7 – Representação de uma árvore por parêntesis aninhados

(2 (5 (9 (4))) (7 (2) (6 (5) (11))))

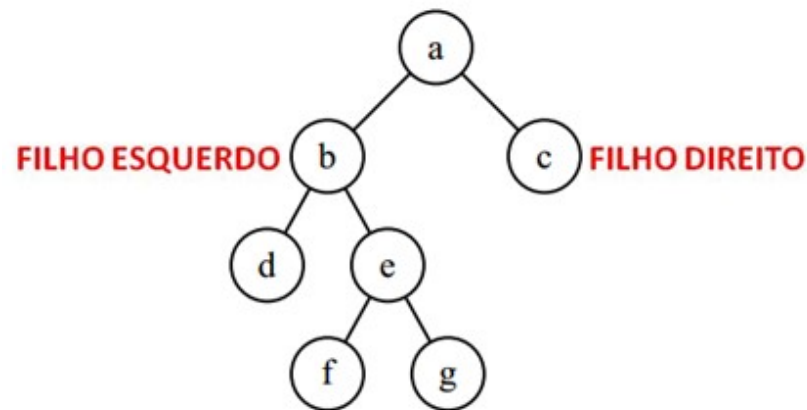
Fonte: Elaborada pela autora.

Árvores Binárias

Os exemplos anteriores mostraram árvores conhecidas como “gerais”, que não têm regras de construção. Conheceremos agora a árvore binária (*binary tree*), que é um tipo especial de árvore, na qual cada nó tem, no máximo, dois filhos que são chamados de:

- **FILHO ESQUERDO;** e
- **FILHO DIREITO.**

Figura 8 – Representação de uma árvore binária



Fonte: Elaborada pela autora.

A Figura 8 apresenta uma árvore binária, na qual cada nó está respeitando a regra de formação: ter no máximo dois filhos (nenhum filho, um filho ou dois filhos). Analise essa árvore e responda: qual é a altura dessa árvore? Quais nós são internos e externos?

As árvores binárias são muito utilizadas como estruturas de armazenamento e concentraremos nosso foco nelas.

Muitas vezes, precisamos ter a dimensão do tamanho de uma árvore binária mesmo antes de construí-la. Para isso, dada a altura de uma árvore, conseguimos determinar: a quantidade máxima de nós do tipo folha e a quantidade máxima de nós dessa árvore.

Número máximo de Nós do tipo folha $\rightarrow 2^h$ (h = altura)

Número máximo de Nós $\rightarrow 2^{h+1} - 1$ (h = altura)

Exemplo:

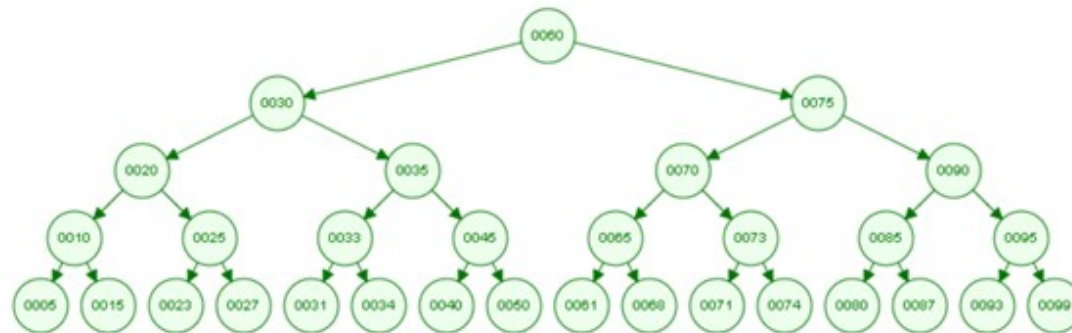
Sabendo-se que uma árvore terá uma altura = 4, determine a quantidade máxima de nós do tipo folha e a quantidade máxima de nós que essa árvore poderá ter.

Número máximo de nós do tipo folha = $2^h = 2^4 = 16$ nós do tipo folha

Número máximo de nós da árvore = $2^{h+1} - 1 = 2^{4+1} - 1 = 2^5 - 1 = 32 - 1 = 31$ nós

Comprovando:

Figura 9 – Representação da quantidade máxima de nós de uma árvore binária com $h = 4$



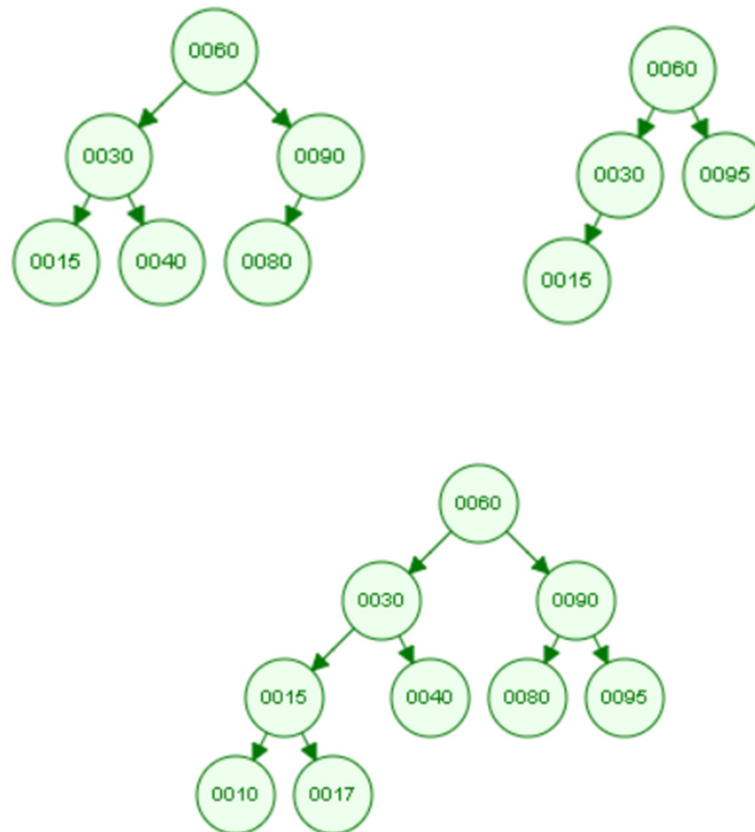
Fonte: Elaborada pela autora.

Veja que podemos categorizar ainda mais as árvores binárias: elas podem ser completas ou cheias.

ÁRVORES COMPLETAS:

Uma árvore binária é chamada de completa quando todos os seus níveis, exceto, possivelmente, o último, estão completamente cheios, e o último nível tem o preenchimento dos nós obrigatoriamente da esquerda para a direita. Veja alguns exemplos:

Figura 10 – Exemplos de árvores binárias completas

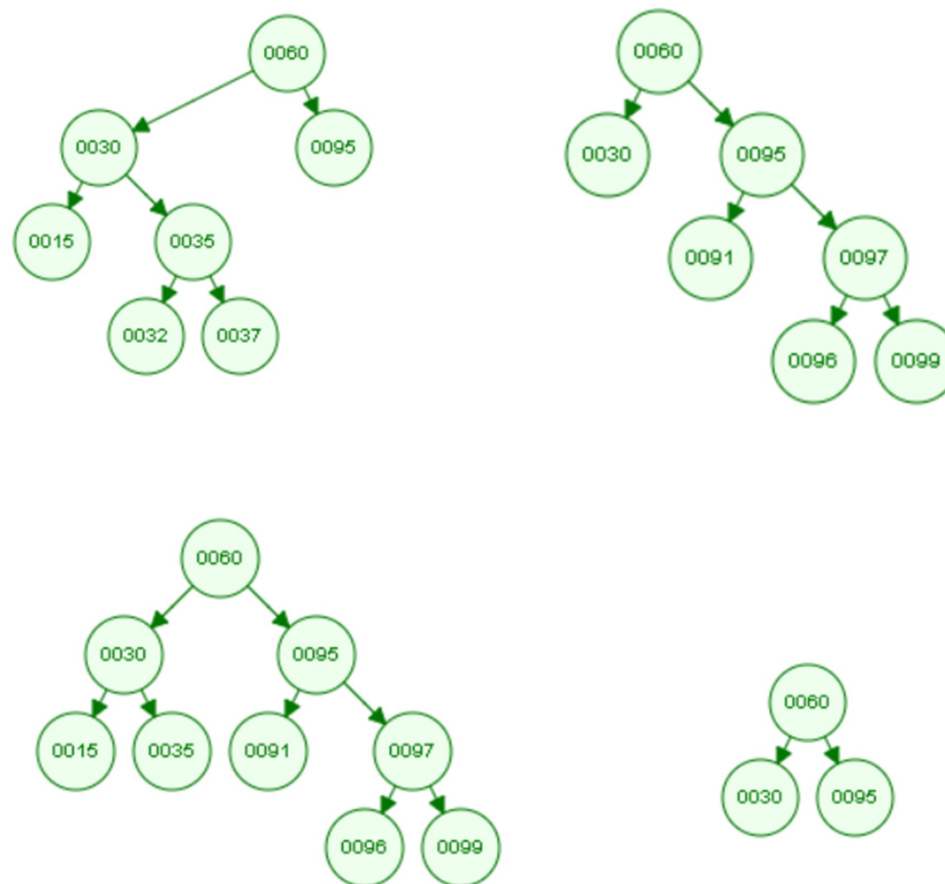


Fonte: Elaborada pela autora.

ÁRVORES CHEIAS:

Uma árvore binária é chamada de cheia quando cada um dos nós ou é uma folha, ou tem dois filhos. Veja alguns exemplos:

Figura 11 – Exemplos de árvores binárias cheias



Fonte: Elaborada pela autora.

Percursos em Árvores Binárias

Se você precisasse percorrer todos os nós de uma árvore binária, por onde você começaria? Pela raiz? E depois? Iria para a direita ou para a esquerda? Que tipo de percurso você utilizaria?

O percurso é um processo que visita todos os nós de uma árvore binária, resultando em uma sequência de nós que foram acessados.

Existem duas categorias básicas de percurso em árvores: descendente (top-down) ou ascendente (bottom-up). No percurso descendente, partimos da raiz e chegamos até uma folha; no percurso ascendente, partimos de uma folha e chegamos até a raiz.

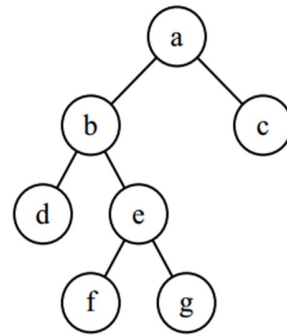
No **percurso descendente**, existem três maneiras básicas de percorrer a árvore:

- em pré-ordem (preorder) ou prefixado;
- em pós-ordem (posorder) ou posfixado; e
- em-ordem (inorder) ou infixado.

PERCURSO PRÉ-ORDEM

- R-E-D ➡ Raiz, esquerda, direita.
- Lista o nó raiz, seguido de suas subárvores (da esquerda para a direita), cada uma em pré-ordem.

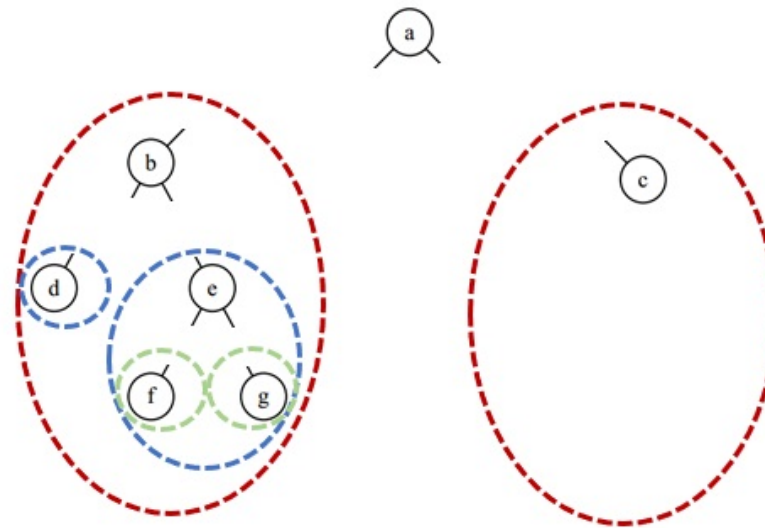
Figura 12 – Exemplo de percurso pré-ordem



RESULTADO DO PERCURSO: abdefgc

Fonte: Elaborada pela autora.

Para poder compreender um percurso, é importante notar que existe uma subárvore à direita e à esquerda do nó raiz e de cada um dos outros nós. Para fazer o percurso, basta aplicar o método RED em cada uma das subárvores.



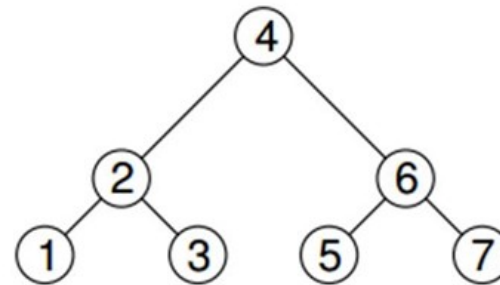
Repare que o nó raiz tem duas subárvores (à esquerda e à direita), sinalizadas em vermelho. O nó c não tem filhos, portanto, não tem subárvore. Por sua vez, o nó b tem dois filhos, e suas subárvores podem ser observadas na marcação em azul. O mesmo ocorre com o nó e, que tem suas subárvores sinalizadas em verde.

Aplicando o método RED, começamos pelo nó raiz **(a)** e vamos para a subárvore da esquerda, cuja raiz é **(b)**, seguimos para a subárvore à esquerda de b e chegamos em **(d)**. Como (d) não tem filhos, partimos para a subárvore à direita de (b) e chegamos na raiz **(e)**. Repetindo o método, devemos ir para o nó à esquerda **(f)** e depois para a direita **(g)**. Com isso, encerramos a subárvore da esquerda de (a) e partimos para a subárvore da direita, que só tem o nó **(c)**.

PERCURSO PÓS-ORDEM

- E-D-R ➡ Esquerda, direita, raiz.
- Percorre a subárvore esquerda (pós-ordem).
- Percorre a subárvore direita (pós-ordem).
- Visita o nó raiz.

Figura 13 – Exemplo de percurso pós-ordem



RESULTADO DO PERCURSO: 1325764

Fonte: Elaborada pela autora.

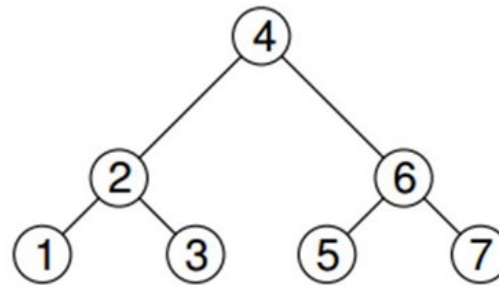
Neste percurso, começamos pelo nó mais à esquerda da subárvore mais à esquerda **(1)** e seguimos, na sequência, aplicando o método EDR, indo para o nó à direita **(3)**, seguindo para a raiz da subárvore **(2)**. Depois, vamos para a subárvore da direita e reaplicamos o método EDR: esquerda

(5), direita (7) e raiz da subárvore (6). Finalmente, chegamos na raiz da árvore (4).

PERCURSO EM-ORDEM

- E-R-D ➔ Esquerda, raiz, direita.
- Percorre a subárvore esquerda (em-ordem).
- Visita o nó raiz.
- Percorre a subárvore direita (em-ordem).

Figura 14 – Exemplo de percurso em-ordem



RESULTADO DO PERCURSO: 1234567

Fonte: Elaborada pela autora.

Nesse percurso, começamos pelo nó mais à esquerda da subárvore mais à esquerda (1) e seguimos na sequência, aplicando o método ERD, indo para o nó raiz (2), seguindo para a direita (3). Seguimos para a raiz da árvore (4) e depois, vamos para a subárvore da direita e reaplicamos o método ERD: esquerda (5), raiz da subárvore (6) e direita (7).

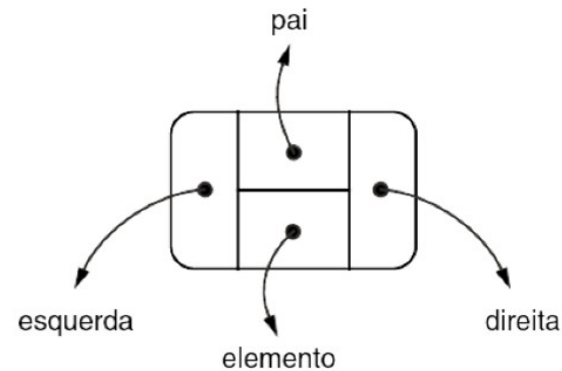
COMPLEXIDADE NO PERCURSO DE ÁRVORES BINÁRIAS:

- Cada percurso requer trabalho constante em cada nó.
- O tipo do percurso é indiferente.
- A iteração sobre todos os n elementos em uma árvore binária requer um tempo $O(n)$.

Árvores Binárias – Representação Gráfica

Como foi dito, uma árvore binária é composta por nós. Veja agora uma representação gráfica de um nó. Usaremos esse esquema na implementação deste TAD.

Figura 15 – Representação de um nó

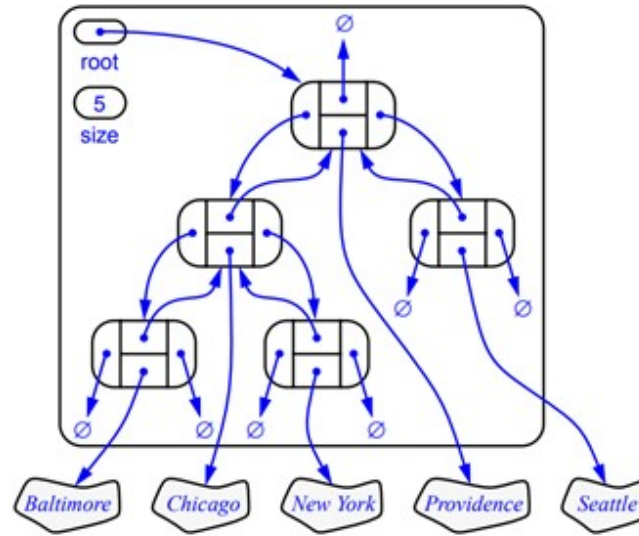


Fonte: GOODRICH; TAMASSIA (2013, p. 305).

Veja que um nó contém, além de seu conteúdo (elemento/tipo), apontadores para o nó pai e seus possíveis descendentes (à direita e à esquerda). Caso o nó não tenha pai (no caso da raiz), ou então, não tenha um dos filhos, as referências esquerda e/ou direita são apontadas para null.

Como fica, então, a representação de uma árvore binária, a partir do modelo anterior? Veja uma árvore binária que armazena nomes de cidades americanas.

Figura 16 – Representação de uma árvore binária



Fonte: GOODRICH; TAMASSIA (2013, p. 305).

Repare que existe o registro de um nó especial: a **RAIZ** e, também, uma variável para contar quantos elementos têm na árvore (**SIZE**).

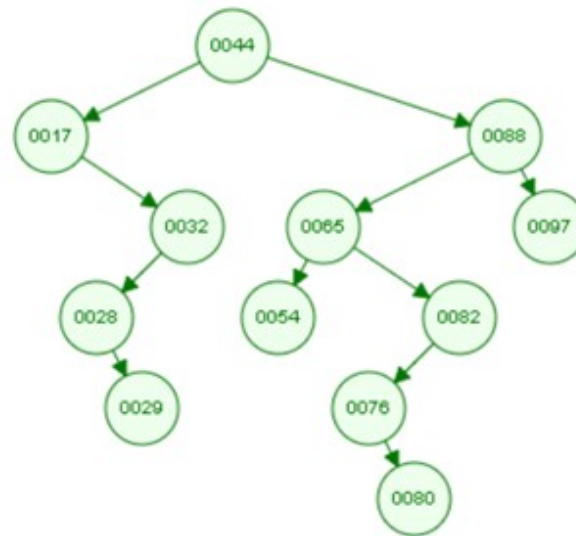
Árvores Binárias de Busca

Uma árvore binária de busca (ABB) é uma estrutura do tipo árvore binária que satisfaz as seguintes condições:

- Todo nó tem um valor que é considerado como campo-chave (não há repetição de conteúdo).
- Para cada nó na árvore, sua chave é maior do que a chave de seu filho à esquerda e menor que a chave de seu filho à direita.

A figura 17 mostra um exemplo de árvore binária de busca T, onde representamos as chaves (números inteiros) dos nós:

Figura 17 – Representação de uma árvore binária de busca



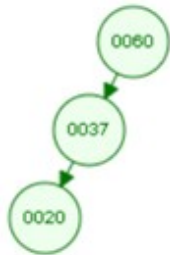
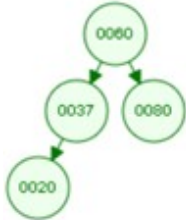


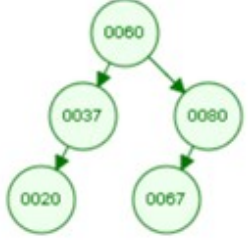
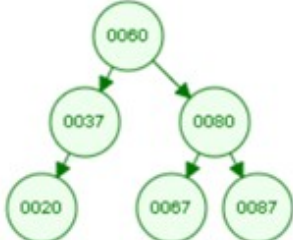
Fonte: Elaborada pela autora.

Repare que, se pegarmos qualquer nó dessa árvore, por exemplo, o nó com conteúdo 65: o valor que está à direita é maior que 65 e o valor que está à esquerda é menor. Essa regra é aplicada para todos os nós.

Essa distribuição vai ocorrendo ao longo da construção da árvore. Construiremos uma ABB com os seguintes valores chave: 60 – 37 – 20 – 80 – 67 – 87.

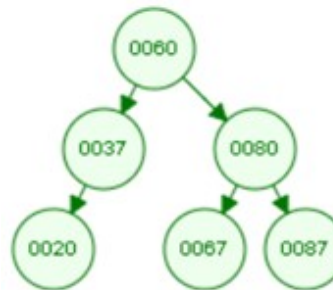
Acompanhe a sequência abaixo:

Chave	Observação	ABB
60	O primeiro nó é sempre a raiz da árvore.	
37	Este valor é menor que 60 e, portanto, deve ficar à esquerda de 60.	
20	Este valor é menor que 60 (deve ficar à esquerda) e é menor que 37. Ficará à esquerda de 37.	
80	É maior que 60. Ficará à direita.	

67	É maior que 60 e deverá ficar à direita. É menor que 80 e ficará à esquerda de 80.	 <pre> graph TD 0060((0060)) --> 0037((0037)) 0060 --> 0080((0080)) 0037 --> 0020((0020)) 0080 --> 0067((0067)) </pre>
87	É maior que 60 e deverá ficar à direita. É maior que 80 e ficará à sua direita.	 <pre> graph TD 0060((0060)) --> 0037((0037)) 0060 --> 0080((0080)) 0037 --> 0020((0020)) 0080 --> 0067((0067)) 0080 --> 0087((0087)) </pre>

BUSCA DE ELEMENTOS EM UMA ABB

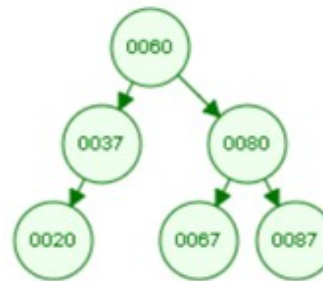
Dado um valor k a ser procurado em uma ABB e o nó raiz, parte-se para a busca deste elemento, sempre realizando comparações do valor procurado (k) com o valor encontrado. Veja:



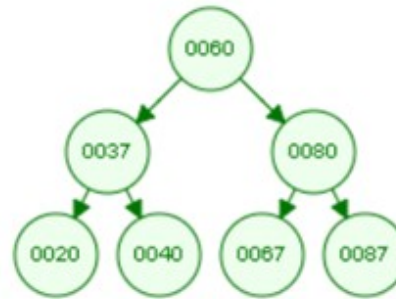
- Valor procurado: $k = 67$.
- A busca inicia no nó raiz. A comparação é realizada (67 e 60) e, como k é maior que 60, as buscas se direcionam para o lado direito da árvore.
- Nova comparação é realizada (67 e 80): como k é menor que 80, a busca é direcionada para o lado esquerdo.
- Nova comparação é realizada (67 e 67). Como são iguais, a busca se encerra de forma positiva: o elemento existe na árvore.
- Caso não tivesse chegado a uma igualdade e não tivéssemos mais filhos para pesquisar (null), a pesquisa se encerraria de forma negativa: não existe o elemento na árvore.

INSERÇÃO DE ELEMENTOS EM UMA ABB

Para inserir um elemento, deve-se primeiro fazer uma busca, de forma a localizar a posição correta de inserção e, na sequência, inserir o nó.



- Valor a ser inserido: $k = 40$.
- A busca inicia no nó raiz. A comparação é realizada (40 e 60) e, como k é menor que 60, as buscas se direcionam para o lado esquerdo da árvore.
- Nova comparação é realizada (40 e 37): como k é maior que 37, a busca é direcionada para o lado direito.
- Como não há elementos à direita de 37, o nó é inserido nessa posição.



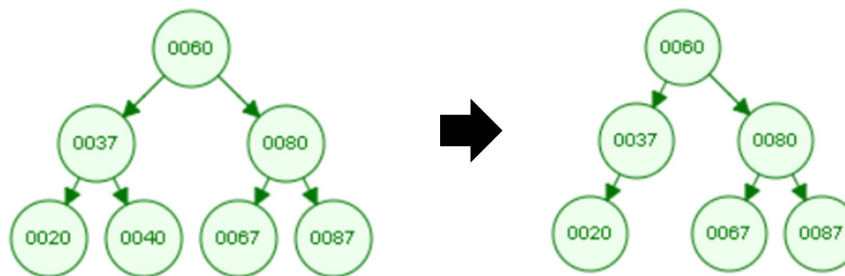
Repare que, ao fazer o percurso em-ordem (ERD) em uma ABB, você gera os valores em ordem crescente:

20 - 37 - 40 - 60 - 67 - 80 - 87

EXCLUSÃO DE ELEMENTOS EM UMA ABB

A exclusão também envolve uma busca, uma vez que só é possível excluir um nó que exista na árvore. Existem três casos.

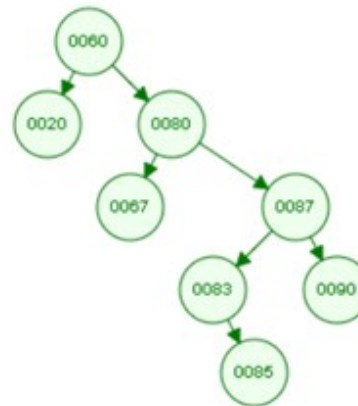
CASO 1 ➔ **Quando o nó a ser removido não tem filhos:** basta fazer o apontador do nó ficar igual a NULL. Vamos excluir o nó 40. Realizamos a busca do valor 40 e fazemos com que o filho direito de 37 passe a apontar para null.



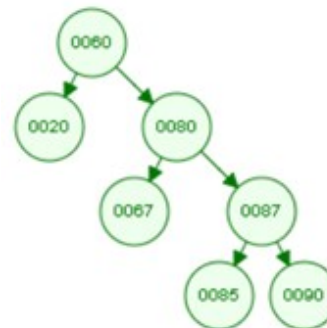
CASO 2 ➔ **Quando o nó a ser removido não tem um dos filhos:** o outro filho assume o local do nó excluído. Vamos excluir o nó 37. Como ele só tem um filho, o nó 20 ocupará seu lugar.



CASO 3 ➔ **Quando o nó a ser removido tem os dois filhos:** vamos excluir o nó 80 da árvore abaixo.



- Passo 1 – identificar, entre todos os descendentes de 80, quem é seu primeiro sucessor (obviamente, à direita). No caso, é 83.
- Passo 2 – excluímos o sucessor encontrado da árvore, aplicando o caso 1 ou o caso 2 (neste exemplo, o caso 2).



- Passo 3 – excluímos o nó desejado (80) e colocamos o sucessor encontrado em seu lugar (83). Resultado final:

