Árvores em estrutura de dados

João Vítor Moreira Dos Santos Neris¹, Matheus Godoy Dos Reis²

¹Faculdade de Tecnologia – Faculdade de tecnologia de Itaquera (FATEC)  
Caixa Postal 08295 - 005 – São Paulo – SP – Brasil

[joao.neris@fatec.sp.gov.br](mailto:joao.neris@fatec.sp.gov.br), [matheus.reis19@fatec.sp.gob.br](mailto:matheus.reis19@fatec.sp.gob.br)

**Abstract.** This article talks about trees in data structures, we introduce the reader by showing that trees, most likely, are not a new subject for them, as we see trees in several different fields of study such as mathematics and biology, and in this article we will see trees in vision of computing, more specifically in data structure. Next we will cover the main concepts of the tree and analyze its properties. Having this basis, the reader will be able to more easily understand the concept of binary tree and binary search tree, which is the real objective of this article.

**Resumo.** Este artigo fala sobre árvore em estrutura de dados, introduzimos o leitor mostrando que árvores, muito provavelmente, não é um assunto novo para ele, já que vemos árvores em diversos campos de estudos diferentes como na matemática e biologia, e neste artigo veremos árvore na visão da computação, mais especificamente em estrutura de dados. A seguir abordaremos os principais conceitos da árvore e analisaremos suas propriedades. Possuindo esta base, o leitor poderá entender mais facilmente o conceito de árvore binária e árvore binária de busca que é o real objetivo deste artigo.

1. **Introdução**

Árvore é um conceito amplamente utilizado em diversos campos de estudos, podemos ver a aplicação de árvores em áreas como a matemática, por exemplo, temos a árvore de decisão, usado para estudar qual comportamento apropriado em determinada situação, você também pode ver aplicações dessa árvore na área de Machine Learning, também podemos falar de árvore no campo da biologia como a árvore genealógica, que vai fazer ligações entre você, seus pais, seus avôs e assim sucessivamente, também temos a famosa árvore filogenética que apresentam para a gente os ancestrais de nossa espécie e a de outros animais, também usamos árvores em diversas áreas da computação, como a árvore de roteamento que mostra a conexão entre roteadores, importante recurso da área de rede de computadores.

Dessa forma, entendemos que árvores não é um conceito exclusivo da computação, e sim algo que foi assimilado a um tipo de estrutura de dados que foi desenvolvida no campo da computação, já que a árvore em computação possui características das topologias em árvores.

1. **Árvore**

É um conjunto de nós interligados, consiste em um raiz onde abaixo dela estão suas subárvore. As árvores, diferentes das listas onde os dados se encontram em sequência, possuem seus dados armazenados de forma hierárquica. É também uma estrutura de dados dinâmica, extremamente eficiente para gerar mecanismos de busca, porém sua aplicação vai além de mecanismo de busca e servem para diversas áreas da computação.

* 1. **Nó**

O nó, também chamado de nodos, é um conceito computacional que possui diversas aplicações, no nosso caso o nó é uma struct composta por uma chave e dois ponteiros, a chave é onde receberemos o conteúdo do nosso nó e se caso o conteúdo for um valor inteiro essa chave pode simplesmente ser uma variável do tipo inteiro para armazenar o valor do dado, já os ponteiros são um recurso usado para fazer a ligação de um nó a outro, no caso da árvore binária existem dois ponteiros por nó, e isso é o que permite a criação da estrutura da árvore.

1. **Representado uma árvore**
   1. **Elementos da árvore**

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Figura 1. Elementos usados para representar a árvore visualmente.

* 1. **Estrutura de uma árvore**

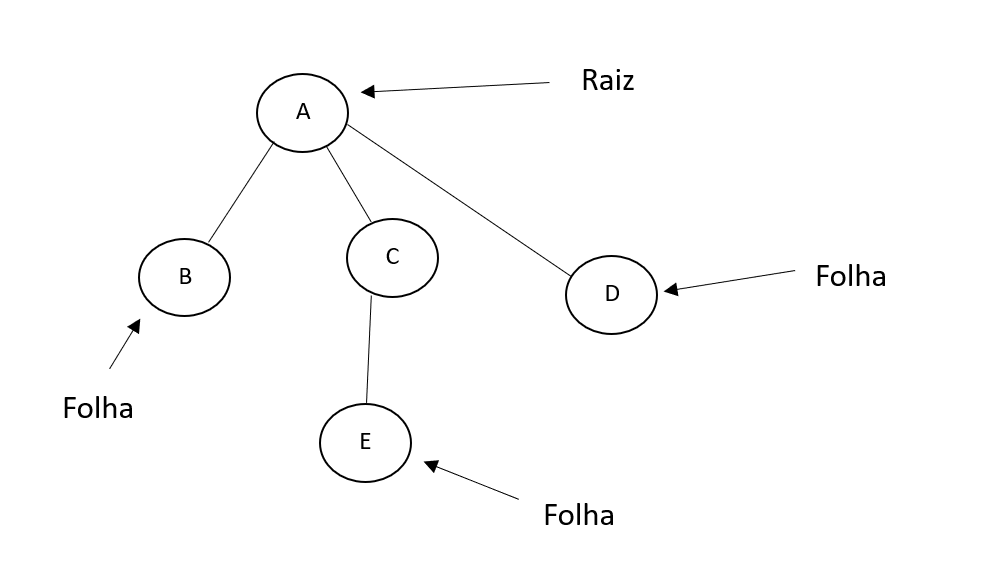


Figura 2. Estrutura simples de uma árvore.

A partir da árvore acima é possível fazer algumas premissas como os exemplos abaixo:

Premissa 1: A é pai de B, C, D;

Premissa 2: B, C, D são filhos de A;

Premissa 3: E é descendente de A;

Premissa 4: A é ancestral de E.

Premissa 5: C é um nó interno.

1. **Conceitos da árvore**
   1. **Grau do nó**

O grau de cada nó indica o número de sub-árvores que ele possui, o grau do nó é o mesmo número de filhos que ele possui.

* 1. **Grau de uma árvore**

O grau de uma árvore igual ao maior grau de um nó pertencente a esta arvore.

* 1. **Árvore unitária**

Árvore que possui um único nó que no caso é sua raiz, ou seja, um nó que não se liga a nenhuma outro nó (nó sem pai e sem filhos).

* 1. **Raíz**

Nó especial, representa o primeiro nó de uma árvore, a partir deste nó podemos chegar a qualquer outro nó da arvore, quando é dito que uma função recebe uma árvore, na verdade o que ela recebe é o endereço da raiz desta árvore.

* 1. **Folha**

Também conhecidos como nó externo, são nós que não possui nenhum outro nó se ligando abaixo dele, pode-se dizer que são nós que possuem filhos vazios, ou seja, são nós de grau zero.

* 1. **Subárvore**

Qualquer trecho da árvore que se encontra abaixo da raiz. Cada nó junto com seus descendentes que se encontra na árvore pode ser considerado uma subárvore, exceto a raiz, os nós que não possuem descendentes podem ser considerados subárvores com elementos nulos (NULL).

* 1. **Pai**

Nó que se liga acima de outro nó, todo nó possui um único pai e um nó pai pode ter no máximo dois filhos. Um outro tipo de interpretação que se pode ter a respeito é de que o nó pai já tem dois filhos, caso o ponteiro deste nó pai não aponte para nenhum outro nó, consideraríamos este filho como elemento vazio (NULL).

* 1. **Filho**

Nó que se liga abaixo de outro nó, todos nós possuem dois filhos quando estamos tratando de uma árvore binária.

* 1. **Descendente**

Nó que se liga a um nó acima através de outros nós, este conceito foi inspirado em conceitos das árvores filogenéticas.

* 1. **Ancestral**

Nó que se liga a um nó abaixo através de outros nós, um nó ancestral sempre será um nó interno.

* 1. **Floresta**

Um conjunto de árvores, tendo duas árvores já é possível considerar uma floresta, nós isolados (sem ligações) podem ser considerados uma floresta composta de árvores unitárias.

* 1. **Nó interno, nó interior ou intermediário**

Nó que não é folha, ou seja, nó que possui um grau diferente de zero.

* 1. **Nível da árvore**

Conforme os nós vão sendo adicionado na árvore, vai sendo gerado diferentes níveis. Para saber como traçar esses níveis é preciso enxergar a árvore como uma árvore genealógica e ir traçando suas gerações como pais, avôs, bisavôs e assim por diante, neste exemplo abaixo temos a raiz no nível 0, dois nós interno no nível 1 e um no nível 2 e 3 folhas no nível 2 e uma no nível 3.

Segue o exemplo abaixo:

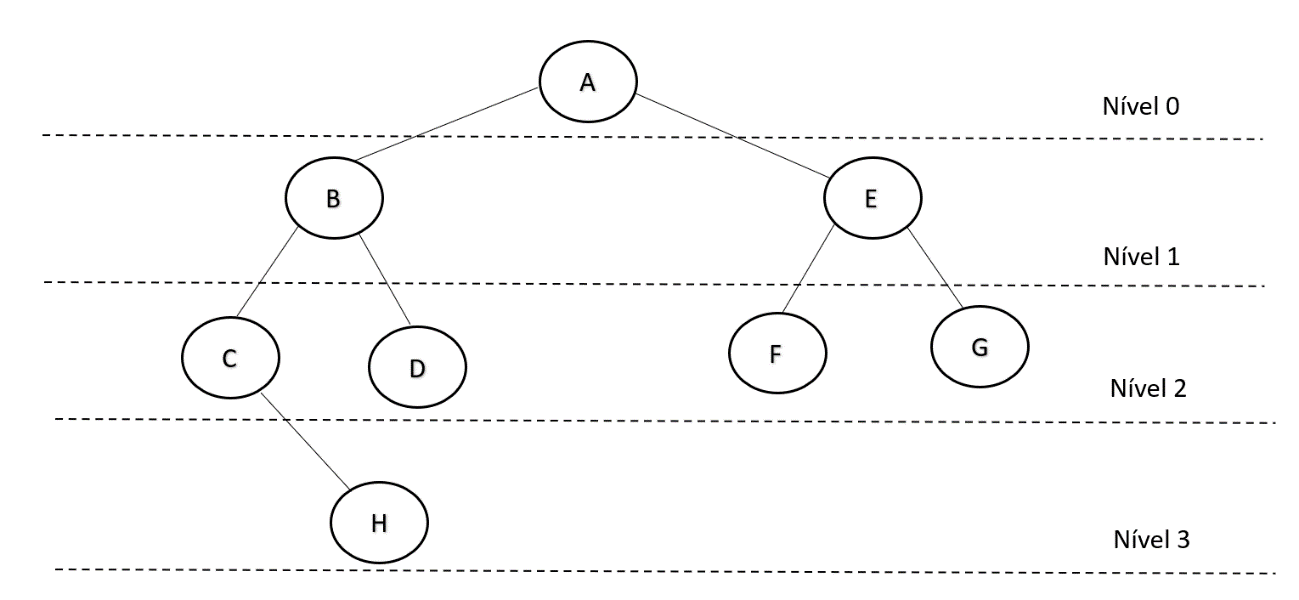


Figura 3. Estrutura de uma árvore com nós sendo separados por níveis.

* 1. **Caminho**

É uma sequência de nós diferentes, n1, n2, n3 ... nk, onde posso chegar a nk através de v1 e posso chegar a n1 através de vk.

* 1. **Altura de uma árvore**

A altura de uma árvore é o caminho mais longo entre a raiz e uma folha pertencentes a esta árvore. No exemplo acima a altura da árvore seria 3.

* 1. **Altura de um nó**

A altura de um nó é a maior distância entre o nó (raiz da Sub-árvore) e uma folha pertencente a está Sub-árvore. No exemplo acima a altura do nó B seria 2.

* 1. **Profundidade**

A profundidade de um nó é a sua distância até a raiz. No exemplo acima a profundidade de A é zero, a profundidade de D é 2 e a profundidade de H é 3.

1. **Árvore binária**
   1. **Lista sequencial**

Um nó ligado a um próximo nó, não possui uma relação hierárquica entre eles.

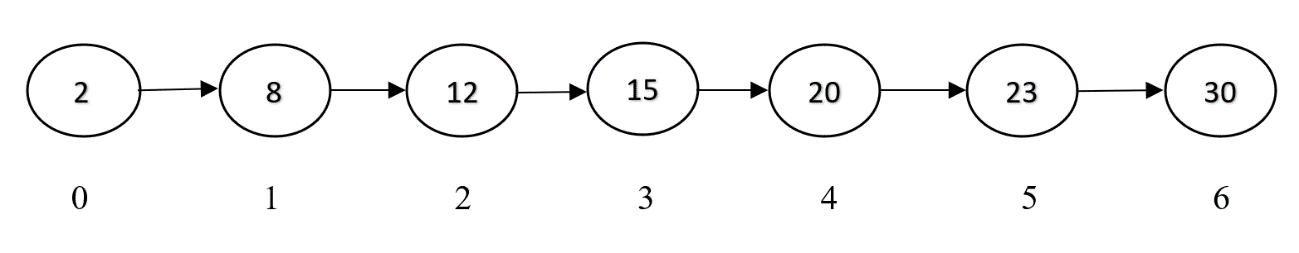


Figura 4. Lista sequencial com 7 elementos.

* 1. **Árvore encadeada**

Um nó ligando-se a outros nós de forma não sequencial e com hierarquia.

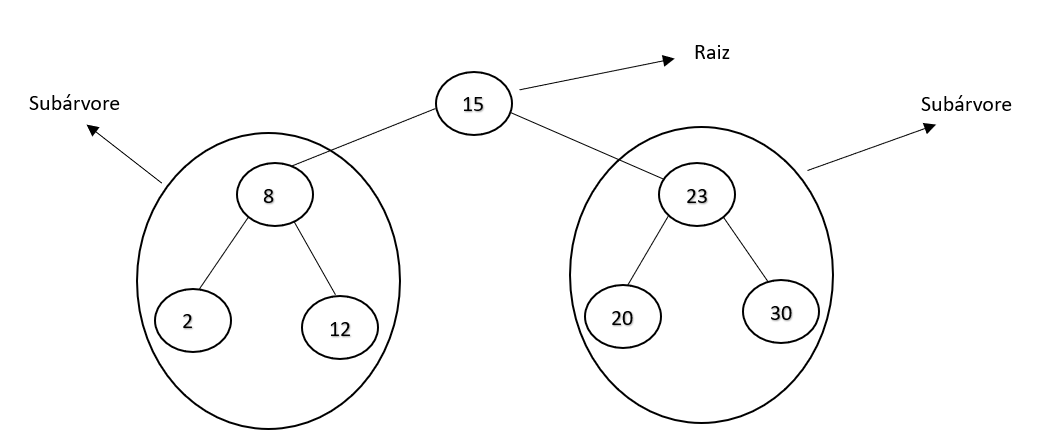


Figura 5. Árvore simples com marcações de suas subclasses.

Observe que a ordem dos nós na direção horizontal não mudou, continua o mesmo tanto na lista quanto na árvore, a real mudança que houve é na forma como esses nós estão se conectando.

Nesse sentido, é possível concluir que a lista seria um tipo particular de árvore onde seu primeiro elemento seria sua raiz e os outros elementos são nós que sempre se conectam de uma forma sequencial.

* 1. **Árvore binária**

A árvore binária é uma árvore construída computacionalmente onde é feita a conexão de nós, ou seja, a árvore binária é feita a partir de uma união de nós. Ela é um tipo específico de árvore onde você precisa ter 2 filhos por nó, cada subclasse da árvore binária é também uma árvore binária.

* 1. **Árvore binária de pesquisa ou Árvore de busca binária:**

É uma árvore binária onde todos os elementos inseridos na árvore que forem maiores do que o nó serão colocados na subárvore à direita, da mesma forma, todos os elementos inseridos na árvore que forem menores do que o nó serão colocados na subárvore à esquerda, dessa forma criando toda uma estrutura de árvore com seus elementos ordenados.

* 1. **Complexidade**

A complexidade das operações de uma árvore de busca binária vai depender do caminho mais longo entre a raiz da árvore e sua folha, ou seja, depende de sua altura. Uma árvore de busca binária possui altura O(log n).

1. **Implementação da árvore de busca binária em C**
   1. **Bibliotecas necessárias**

Utilizaremos funções de saída de dados, como o printf() e também utilizaremos funções de alocação de memória, como o malloc() e o free(), por isso há necessidade de ser declarada dessas bibliotecas serem declaradas.

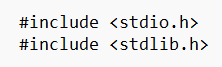


Figura 6. Trecho de código mostrando bibliotecas do programa.

* 1. **Struct do nó**

Isto é uma struct que define um nó, é uma parte fundamental do código, pois é a partir dela que toda a estrutura da árvore pode ser desenvolvida, ela é composta por uma variável do tipo inteiro e outros dois ponteiros, chamado esquerda e direita que vão fazer as ligações com os outros nós. Este int chave poderia ser qualquer outro tipo de dado de acordo com o seu objetivo com criação de uma árvore binária, no nosso caso aqui faremos um uso simples adicionado apenas números.

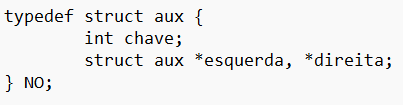


Figura 7. Trecho de código mostrando struct do programa.

* 1. **Typedef para nó da struct**

Este typedef serve para renomear a nossa struct, é como se ela ganhasse um apelido para facilitar o seu uso e entendimento, sendo assim é possível perceber que seu uso não é obrigatório.



Figura 8. Trecho de código o typedef do programa.

* 1. **Função Inicializa**

Esta função abaixo inicializa a árvore, ela retorna o valor NULL para uma variável, este valor é importante para que possa ser gerado a raiz em cima dele, seu uso não é obrigatório, esta parte poderia ser feita de forma manual sem prejuízos.

**Texto

Descrição gerada automaticamente**

Figura 9. Trecho de código mostrando a função inicializa do programa.

* 1. **Função criar novo nó**

Esta função é um criador de nó, ela vai usar o tipo de dado gerado pela struct para definir todos os nós da árvore, a struct define um nó para o nosso código, porém é necessário usar esta função para criar o nó e definir o valor de sua chave, esta função também atribui o valor NULL para os ponteiros direita e esquerda do nosso nó, o que faz todo sentido já que quando um novo nó é criado, ele, em teoria, não deve estar apontando para ninguém, para que ele aponte para outro nó, outro nó deve ser criado e seu endereço ser associado ao ponteiro esquerdo ou direito do último nó gerado, ele também alocará memória para o nó criado usando a função malloc() que vem da biblioteca stdlib.h, o malloc retorna o tamanho de bytes, com o auxílio da função sizeof() , de um nó gerado pela nossa struct, por fim retornará o novo nó criado.

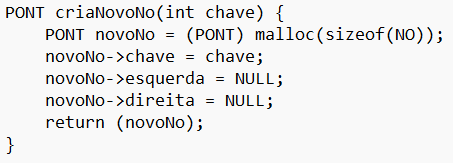


Figura 10. Trecho de código mostrando a função criar novo nó do programa.

* 1. **Função Adiciona**

Esta é outra função recursiva de suma importância para o nosso código, ela é que traz toda a organização dos elementos em nossa árvore, ela é quem garante que os menores elementos sempre estarão à esquerda da árvore e os maiores elementos sempre estarão a direta da árvore, isso é crucial para que o mecanismo de busca binária funcione de forma eficiente. Após a função receber o nó que deverá ser alocado na árvore e receber o endereço da raiz da árvore de onde o nó vai ser alocado a função verificará se o nó da árvore é nulo, caso for, o nó que foi solicitado pelo usuário a ser inserido é colocado no lugar desse nó null, caso contrário, isso significará que a função encontrou um nó com um número dentro, é neste momento que ela irá verificar se o valor do nó enviado pelo usuário é menor que o valor do nó que está na árvore, se sim, de forma recursiva será chamado a função adiciona enviando o ponteiro esquerdo do nó da arvore e o nó enviado pelo usuário, se não, de forma recursiva será chamado a função adiciona enviando o ponteiro direito do nó da arvore e o nó enviado pelo usuário.

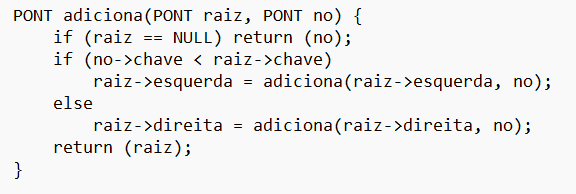


Figura 11. Trecho de código mostrando a função adiciona do programa.

* 1. **Função Busca**

Aqui temos um mecanismo de busca feito para uma arvore binária, é uma função que utiliza de recursividade, ela consegue desfrutar da forma como os dados estão organizados na árvore para fazer uma busca com pouquíssimas iterações utilizando-se de forma mínima os recursos de hardware do computador. Após receber através dos parâmetros da função um valor escolhido pelo usuário e o endereço de uma raiz, o funcionamento da função ‘busca’ baseia-se em verificar se o conteúdo do nó é igual ao valor solicitado pelo usuário, se verdadeiro, ele retorna o nó onde está localizado este valor, caso contrário ele faz uma verificação se o número que está neste nó é maior que o valor solicitado pelo usuário, se sim, ele retorna recursivamente a própria função mandando pelos parâmetros o valor solicitado pelo usuário e o ponteiro esquerdo do nó, se não, ele retorna recursivamente a própria função mandando pelos parâmetros o valor solicitado pelo usuário e o ponteiro direito do nó.

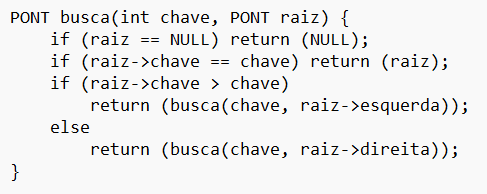


Figura 12. Trecho de código mostrando a função busca do programa.

**Referências**

[Wikipédia (2021)] [https://pt.wikipedia.org/wiki/%C3%81rvore\_bin%C3%A1ria\_ de\_busca](https://pt.wikipedia.org/wiki/%C3%81rvore_bin%C3%A1ria_%20de_busca)

[Wikipédia (2022)] <https://pt.wikipedia.org/wiki/%C3%81rvore_bin%C3%A1ria>

[USP] <http://wiki.icmc.usp.br/images/d/d6/SCC-601-Aula_13-Arvores.pdf>

[David Santiago] <https://algol.dev/arvores-estrutura-de-dados/>

[UFES (2012)] <http://www.inf.ufes.br/~pdcosta/ensino/2012-1-estruturas-de-dados/slides/Aula15%20%28arvores%29.pdf>

[WIKI (2013)] [https://pt.wikibooks.org/wiki/Algoritmos\_e\_Estruturas\_de\_Dados/%C 3%81rvore](https://pt.wikibooks.org/wiki/Algoritmos_e_Estruturas_de_Dados/%25C%203%81rvore)

[Prof. Marcelo Nogueira] <https://noginfo.com.br/arquivos/CC_LP_T13.pdf>

[Paulo Feofiloff (2018*)*] https://www.ime.usp.br/~pf/algoritmos/aulas/bint.html