Árvores

Árvores Binárias

Prof. Edson Alves - UnB/FGA

2018

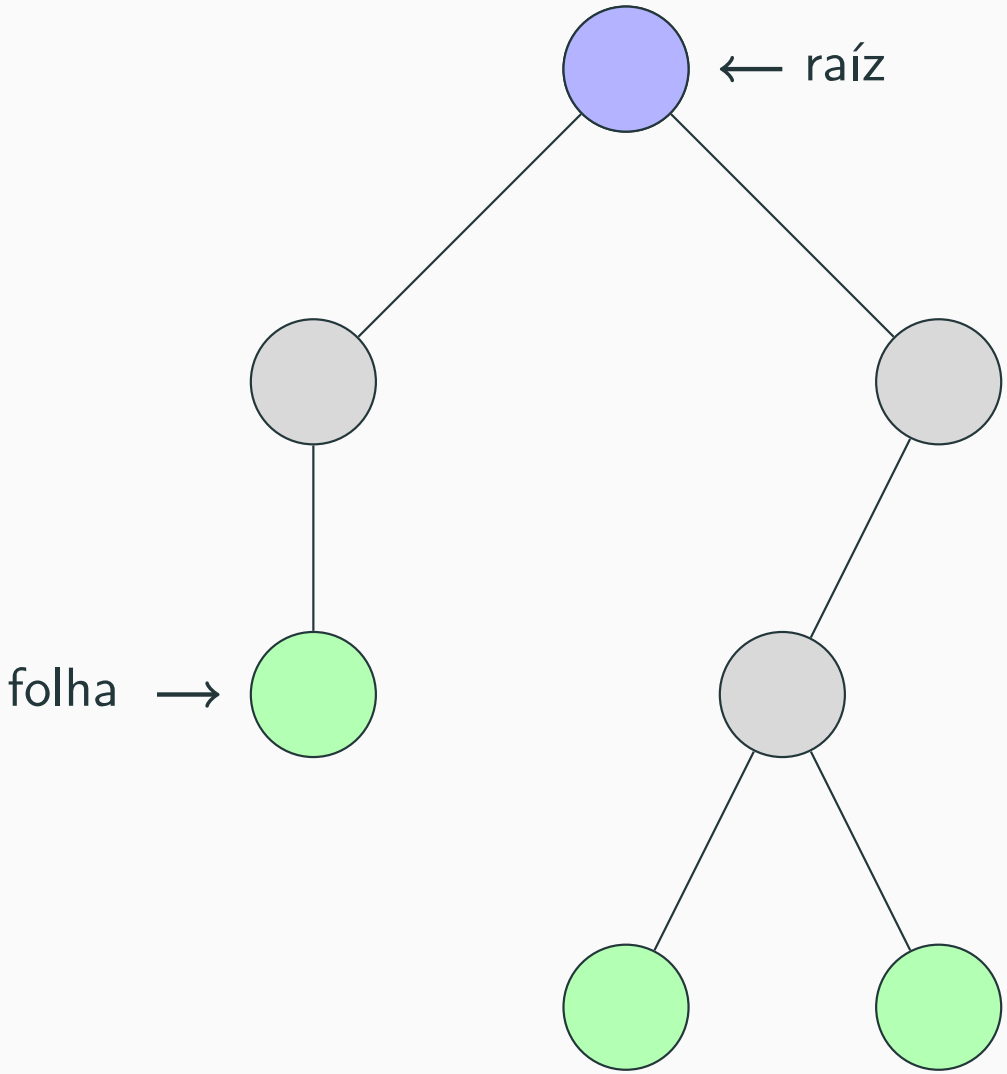
Sumário

Árvores

Características das árvores

- As árvores são estruturas compostas de nós e ramos (arestas)
- Ao contrário das árvores reais, a visualização de árvores em algoritmos é invertida, com a raiz no topo e as folhas na base
- A raiz é um nó que não tem pai
- Folhas são nós que não tem filhos
- Cada nó pode ser alcançado atráves de uma sequência única de ramos, denominada caminho
- O nível de um nó N corresponde ao número de nós do caminho de N até a raiz
- A altura de uma árvore é igual ao nível máximo dentre todos os nós da árvore
- Uma árvore vazia tem altura 0 (zero); uma árvore com um único nó tem altura 1

Visualização de uma árvore



Definição formal das árvores

As árvores são estruturas que podem ser definidas recursivamente da seguinte maneira:

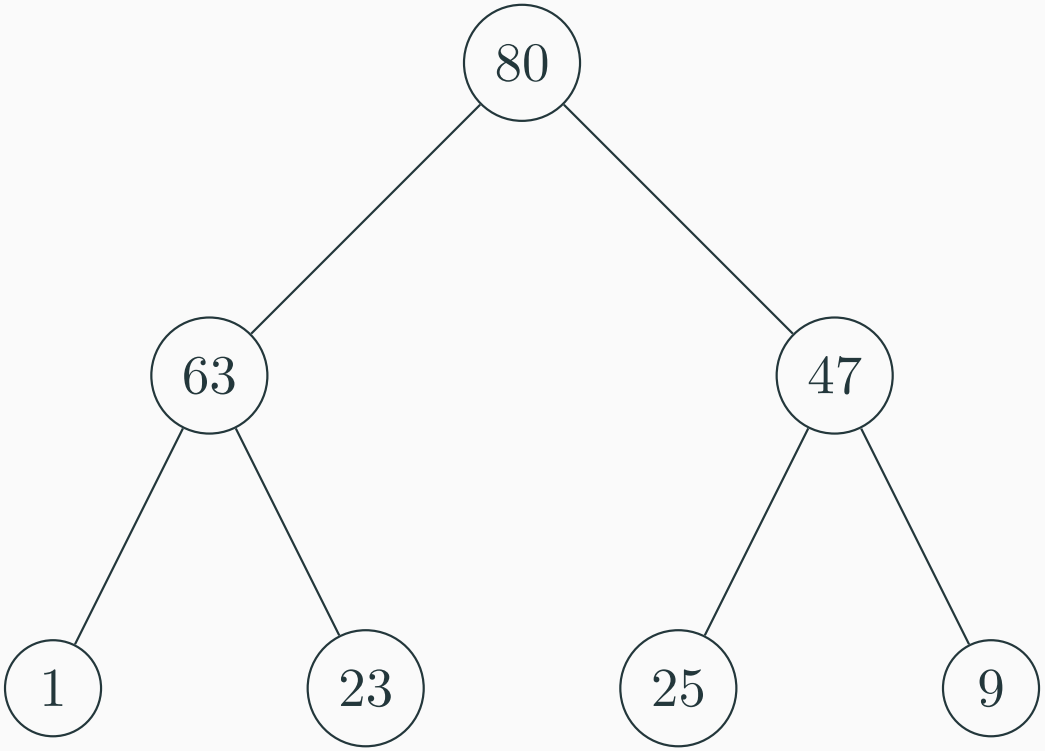
1. Uma estrutura vazia é uma árvore
2. Se $t_1, t_2, \ldots, t_k$ são árvores disjuntas, então a estrutura cuja raiz tem como filhos as raizes de $t_1, t_2, \ldots, t_k$ também é uma árvore
3. Apenas estruturas geradas pelas regras 1 e 2 são árvores

Árvores Binárias

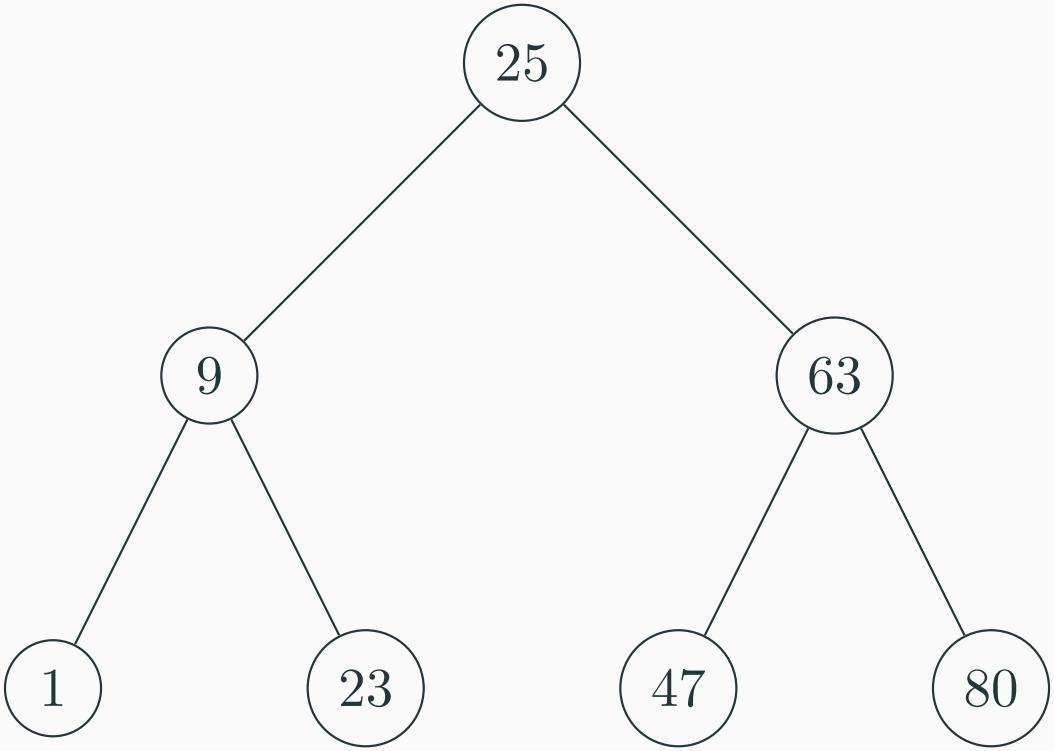
Árvores binárias

- A definição de árvores não impõem qualquer restrição no número de filhos que um nó pode ter
- Uma árvore é dita binária se cada nó tem, no máximo, dois filhos: o esquerdo e o direito
- O estabelecimento de uma ordem entre as informações armazenadas em um nó e seus filhos leva a especilizações úteis de uma árvore binária
- Por exemplo, uma *max heap* é uma árvore binária cuja informação contida no pai é maior ou igual a informação contida em seus filhos
- Já em uma árvore binária de busca, as informações contidas em qualquer nó da subárvore à esquerda de um nó N devem ser menores do que a informação contida em N
- Do mesmo modo, as informações contidas nos nós da subárvore à direita de N devem ser maiores do que a informação armazenada em N

Exemplo de max heap



Exemplo de árvore binária de busca



Implementação

Implementação de árvores binárias

- Uma árvore binária pode ser implementada, no mínimo, de duas maneiras: utilizando vetores ou utilizando listas encadeadas
- no caso dos vetores, a informação da raiz fica armazenada no índice 1 (o índice zero não é utilizado)
- Dado um nó pai armazenado no índice p, o nó à esquerda ocupa o índice $2p$ e o nó à direita ocupa o índice $2p + 1$
- Se um nó ocupa o índice $i \neq 1$, seu pai está armazenado no índice $i/2$
- No caso das listas duplamente encadeadas, cada nó da lista representa um nó da árvore binária

Implementação de uma árvore usando listas encadeadas

```
1  template<typename T>
2  class BinaryTree {
3  private:
4      struct Node {
5          T info;
6          Node *left, *right;
7      };
8
9      Node *root;
10
11 public:
12     BinaryTree() : root(nullptr) {}
13 };
```

Implementação de uma árvore usando vetores

```cpp
#include <vector>

template<typename T>
class BinaryTree {
private:
    std::vector<T> nodes;

    int left(int p) { return 2*p; }
    int right(int p) { return 2*p + 1; }
    int parent(int i) { return i/2; }

public:
    // O índice zero não é utilizado, mas deve estar alocado
    BinaryTree() : nodes(1) {}
};
```

Referências

1. **DROZDEK**, Adam. *Algoritmos e Estruturas de Dados em C++*, 2002.
2. **KERNIGHAN**, Bryan; **RITCHIE**, Dennis. *The C Programming Language*, 1978.
3. **STROUSTROUP**, Bjarne. *The C++ Programming Language*, 2013.
4. C++ Reference[1].

[1] https://en.cppreference.com/w/