Operações em Árvores

João Paulo Dias de Almeida
jp.dias.almeida@gmail.com

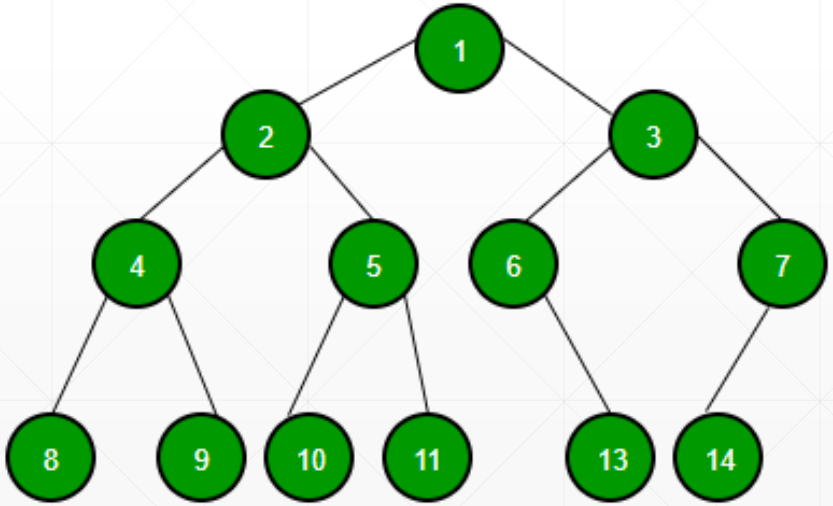
Universidade Federal de Sergipe

O que vamos aprender hoje?

- Entender os diferentes percursos em uma árvore
- Implementar árvores binárias
- Diferenciar árvore binária de árvore binária de busca

Árvore binária

- Uma árvore que possui **no máximo** dois filhos é chamada de árvore binária
  - Filho esquerdo e filho direito



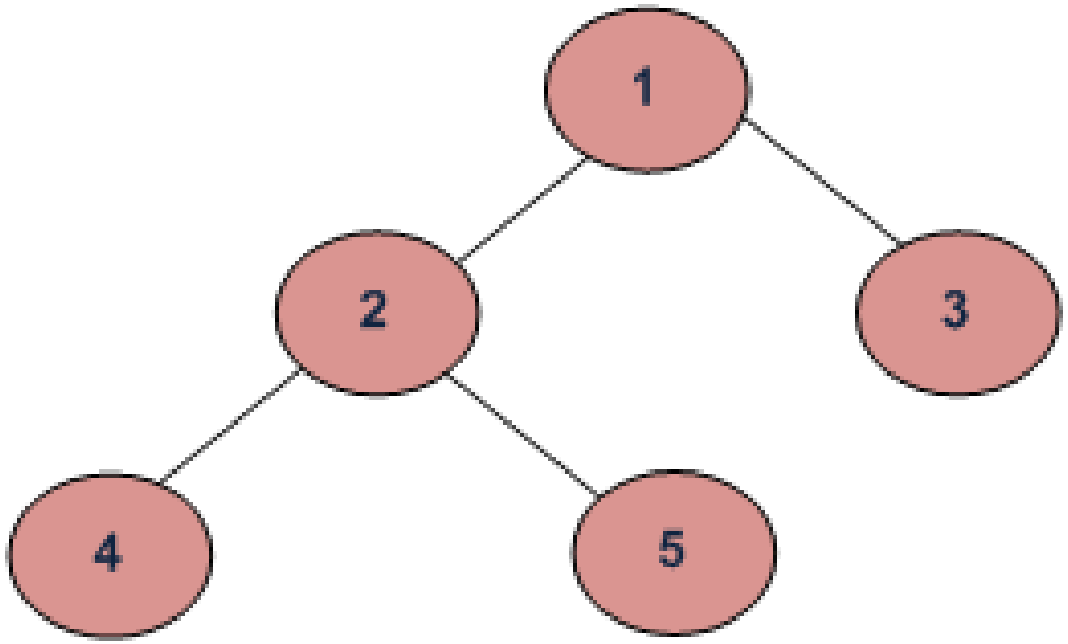
Ordem de Percurso

Percurso em árvore binária

- Uma das operações possíveis em uma árvore é a de percorrer a árvore
  - É o ato de caminhar sobre a árvore enumerando cada um dos seus nós uma vez
    - Sempre que um nó é enumerado, dizemos que ele foi visitado
- Não existe uma única forma de percorrer uma árvore
  - Pré-ordem ou profundidade
  - Em ordem ou ordem simétrica
  - Pós-ordem

Percurso em árvore binária

- Em algumas situações será necessário visitar todos os nós da árvore
- Então podemos utilizar um dos percursos:
  - **Pré-ordem**
    - Raiz → subárvore esquerda → subárvore direita
  - **Em ordem**
    - Subárvore esquerda → raiz → subárvore direita
  - **Pós-ordem**
    - Subárvore esquerda → subárvore direita → raiz



Percurso

- Pré-ordem:

  1 – 2 - 4 – 5 - 3

- Em ordem:

  4 – 2 – 5 – 1 – 3

- Pós-ordem:

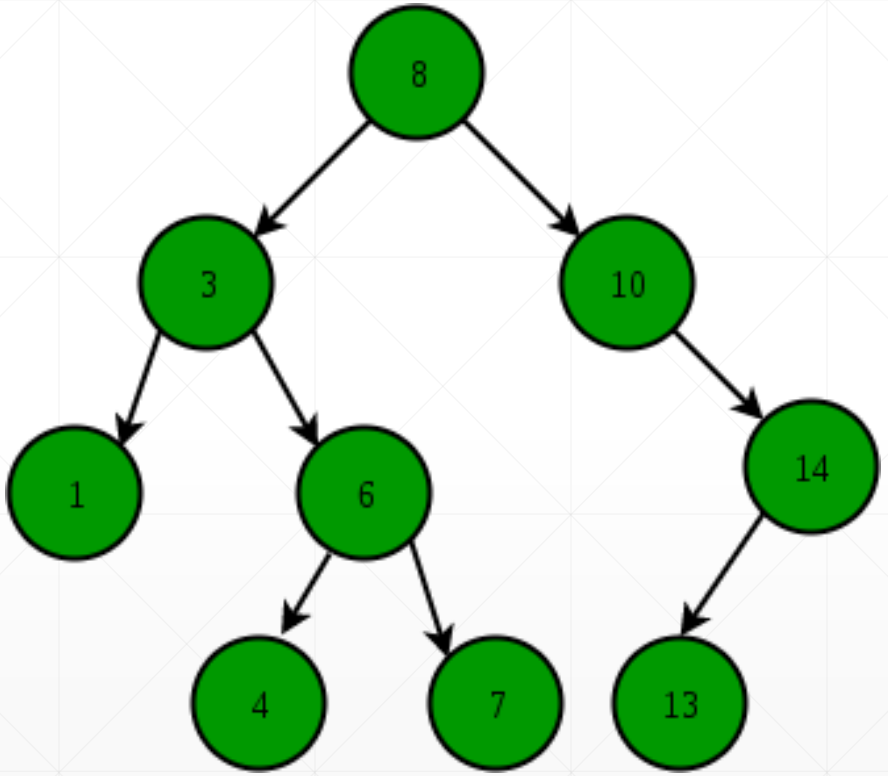
  4 – 5 – 2 – 3 – 1

- **A complexidade de todos algoritmos de percurso é O(n)**

Árvore binária de busca

- É a árvore binária cujos elementos estão ordenados
  - A subárvore a **esquerda** de todo nó possui apenas nós com valores **inferiores**
  - A subárvore a **direita** de todo nó possui apenas nós com valores **superiores**
- Além disso:
  - Não devem haver nós duplicados
  - Todas as subárvores também devem ser árvores binárias de busca

Árvore Binária de Busca



Inserção, remoção, e busca de um nó são mais rápidos

Árvore Binária de Busca

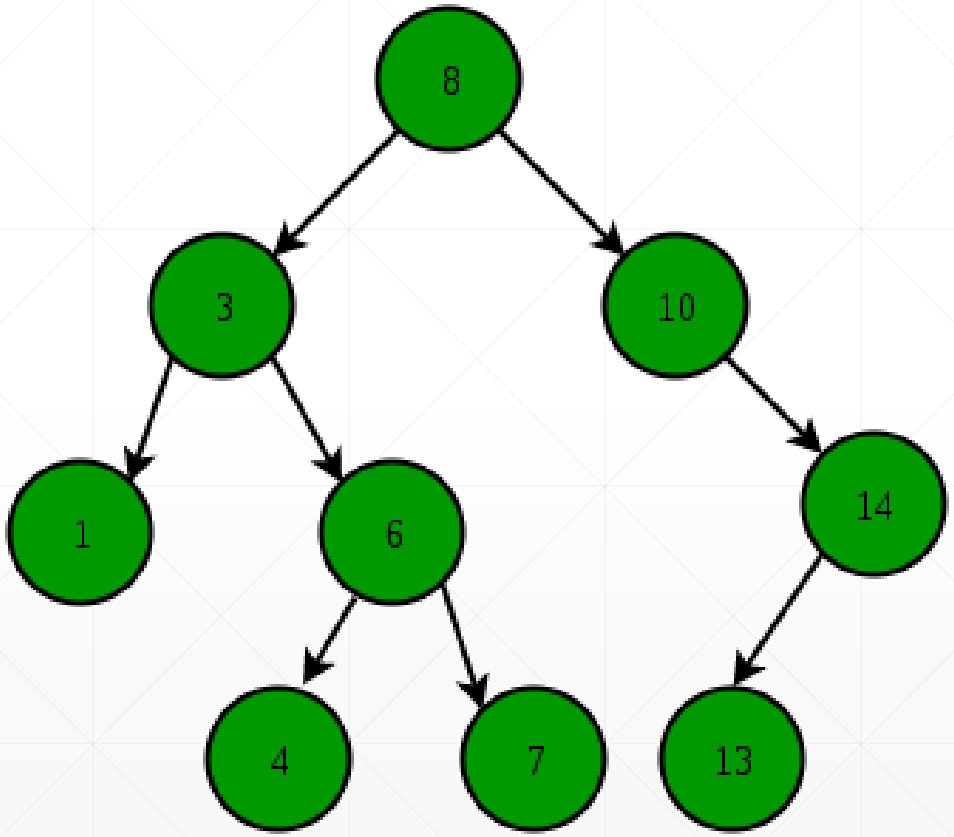
- Características:
  - Busca **e** atualização rápidas
- Listas duplamente encadeadas (desordenadas)
  - Inserção e remoção → O(1)
  - Busca → O(n)
- Arrays ordenados
  - Busca (binária) → O (log n)
  - Atualização → O(n)

Operações

Busca

- Algoritmo:
  1. Comece da raiz
  2. Compare o elemento a ser buscado com a raiz, se menor do que a raiz, percorra recursivamente a subárvore esquerda, caso contrário, percorra a subárvore direita
  3. Se o elemento for encontrado, retorne true, caso contrário false

Busca: exemplo

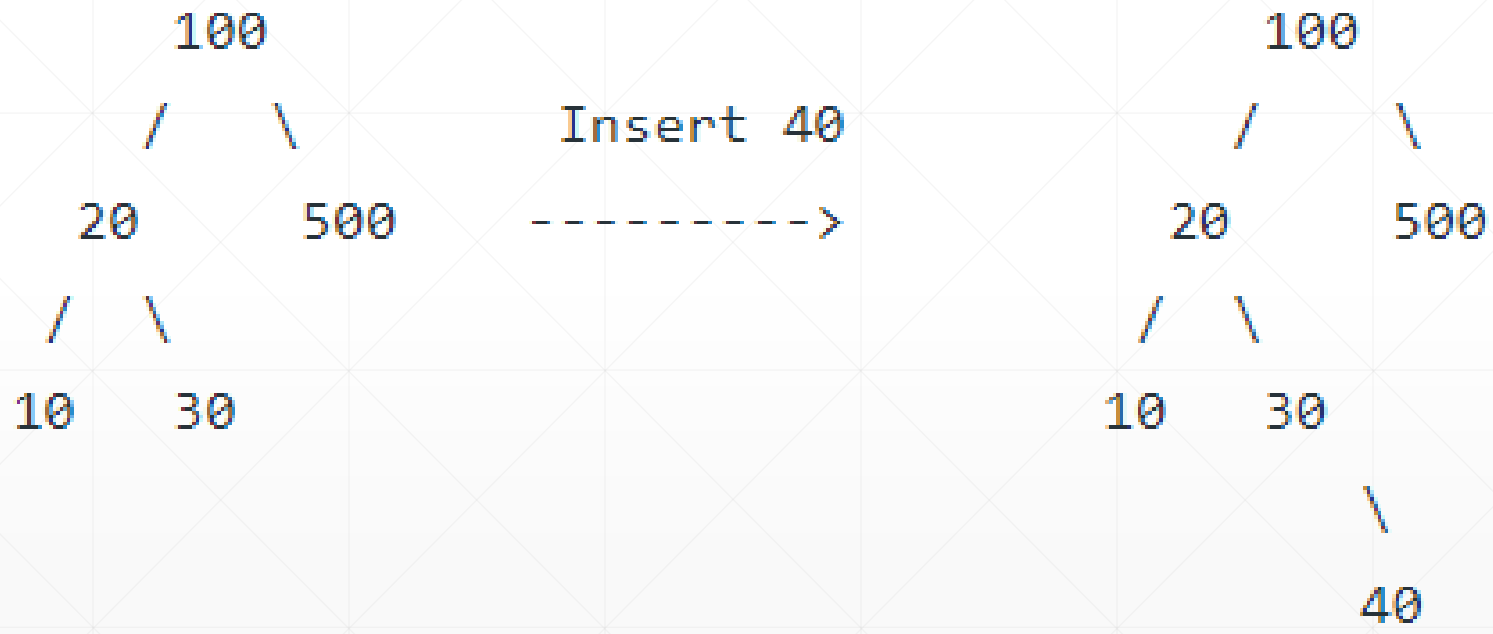


Exemplo - Busca

Inserção

- Um novo nó sempre é inserido na folha
  - Começamos a percorrer a árvore da raiz até chegar a folha
  - Ao chegar na folha, o novo nó é acrescentado como filho desta folha

Inserção



Exemplo – Inserção

Remoção

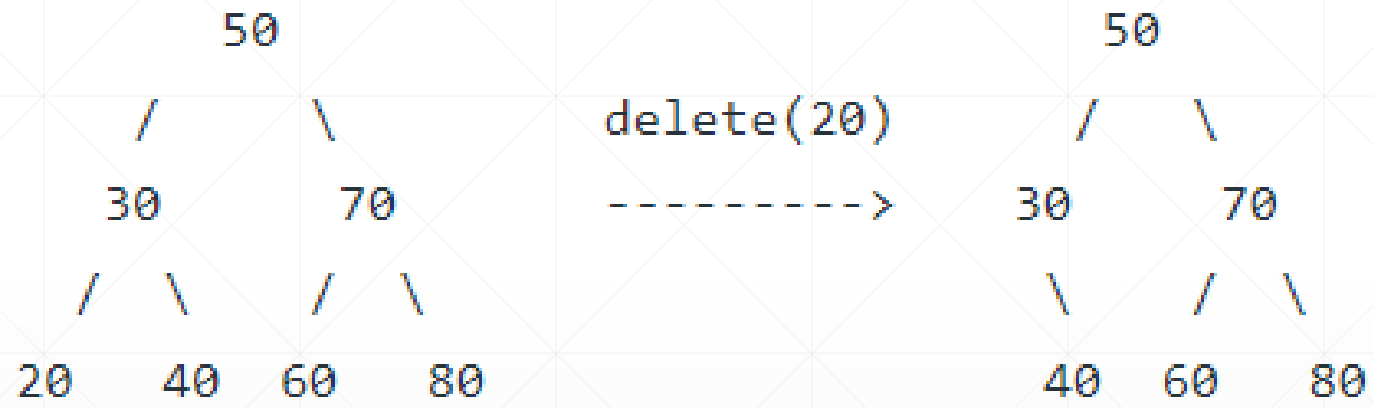
- Quando removemos um nó, três cenários podem ocorrer:
  1. Nó a ser removido é uma folha

     Apenas remova da árvore
  2. Nó a ser removido possui apenas um filho:

     Copie o filho, atualize as referências, e remova o nó
  3. Nó a ser removido possui dois filhos:

     1) Encontre o sucessor em ordem do nó. 2) Copie o conteúdo do sucessor. 3) atualize as referências e remova o nó desejado

Nó a ser removido é uma **folha**



Remoção

- Quando removemos um nó, três cenários podem ocorrer:
  1. Nó a ser removido é uma folha

     Apenas remova da árvore
  2. Nó a ser removido possui apenas um filho:

     Copie o filho, atualize as referências removendo o nó
  3. Nó a ser removido possui dois filhos:

     1) Encontre o sucessor em ordem do nó. 2) Copie o conteúdo do sucessor 3) atualize as referências e remova o nó desejado

Nó a ser removido possui apenas **um filho**

Remoção

- Quando removemos um nó, três cenários podem ocorrer:
  1. Nó a ser removido é uma folha

     Apenas remova da árvore
  2. Nó a ser removido possui apenas um filho:

     Copie o filho, atualize as referências, e remova o nó
  3. Nó a ser removido possui dois filhos:

     1) Encontre o sucessor em ordem do nó. 2) Copie o conteúdo do sucessor. 3) atualize as referências e remova o nó desejado

Nó a ser removido possui **dois filhos**

Exemplo – Remoção

Operações: características

- O pior caso das operações de busca, inserção e remoção é O(h), onde h é a altura da árvore
  - A altura de uma árvore **desbalanceada** pode se tornar n, resultando em um custo O(n)
- O percurso em ordem sempre produz uma saída ordenada

Referências

- GeeksforGeeks. Tree Traversals. Disponível em: https://www.geeksforgeeks.org/tree-traversals-inorder-preorder-and-postorder/?ref=gcse. Acessado em: 03/02/2022
- GeeksforGeeks. Binary Search Tree. Disponível em: https://www.geeksforgeeks.org/binary-search-tree-set-1-search-and-insertion/.Acessado em: 03/02/2022
- SKIENA, Steven. The Algorithm Design Manual. 6ª edição. Springer, 2020.

Referências

- CORMEN, Thomas. Desmistificando Algoritmos. Editora Campus, 2012.
- GOMES, G. Árvores. Notas de aula disponíveis em: https://bit.ly/3oowxBT. Acessado em: 28/01/2022

Dúvidas?