Árvore de busca binária

Programação II – Engenharia de Telecomunicações

Prof. Emerson Ribeiro de Mello

mello@ifsc.edu.br

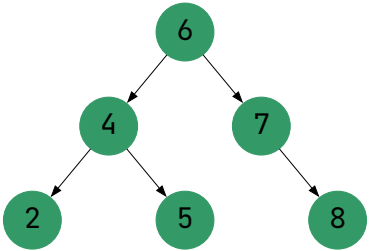
INSTITUTO FEDERAL
Santa Catarina
Campus São José

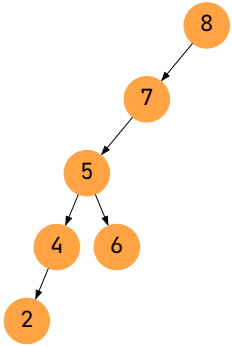
Licenciamento

Sumário

Percursos em árvores binárias

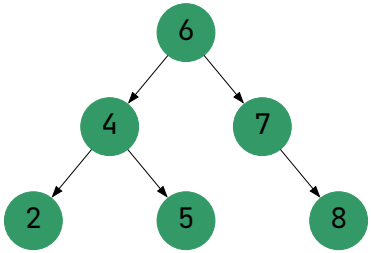
Árvores de busca binária

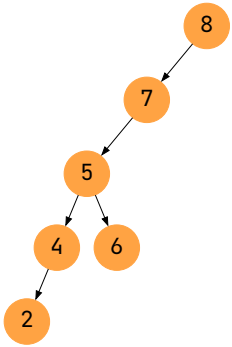




- Árvores de busca binária diferentes podem representar o mesmo conjunto de chaves
- O tempo de execução do pior caso para a maioria das operações é proporcional à altura da árvore. Árvore com altura 4 é menos eficiente que uma árvore com altura 2.

Árvores de busca binária





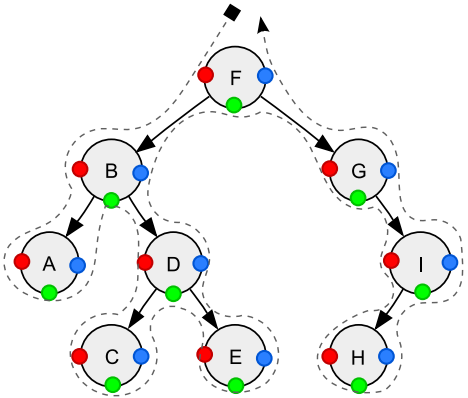
Segundo Cormen et al. (2012), a altura esperada de uma árvore de busca binária construída de forma aleatória é $O(\log n)$

Travessia ou percurso em profundidade (*depth-first*)

Percorrer todos os nós da árvore uma única vez

- **pré-ordem**: raiz, esquerda, direita
  - `F, B, A, D, C, E, G, I, H`
- **in-ordem**: esquerda, raiz, direita
  - `A, B, C, D, E, F, G, H, I`
- **pós-ordem**: esquerda, direita, raiz
  - `A, C, E, D, B, H, I, G, F`

Percorrer uma árvore de busca binária tem complexidade $\Theta(n)$



Travessia em profundidade
Fonte: Wikipedia

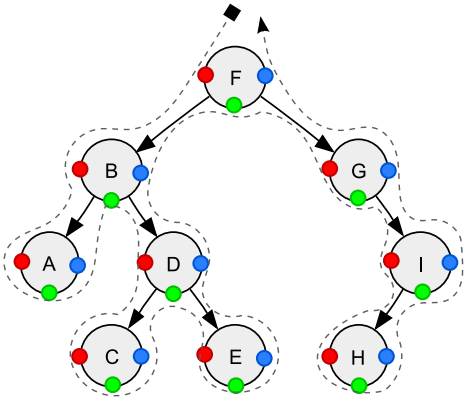
Travessia ou percurso em profundidade (*depth-first*)

Percorrer todos os nós da árvore uma única vez

- **pré-ordem**: raiz, esquerda, direita
  - `F, B, A, D, C, E, G, I, H`
- **in-ordem**: esquerda, raiz, direita
  - `A, B, C, D, E, F, G, H, I`
- **pós-ordem**: esquerda, direita, raiz
  - `A, C, E, D, B, H, I, G, F`

Percorrer uma árvore de busca binária tem complexidade $\Theta(n)$

- É possível generalizar a travessia para árvores m-árias, basta percorrer cada uma das subárvores de um nó



Travessia em profundidade
Fonte: Wikipedia

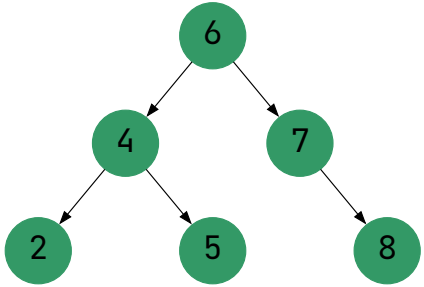
Travessia in-ordem em profundidade

Exemplo

Algoritmo 1: Travessia in-order

```
1 função travessia_inorder(x):
2     se x ≠ nulo então
3         travessia_inorder(x.esquerda);
4         imprima(x.valor);
5         travessia_inorder(x.direita);
6     fim
7 fim
```

Qual a saída para a árvore da figura ao lado?



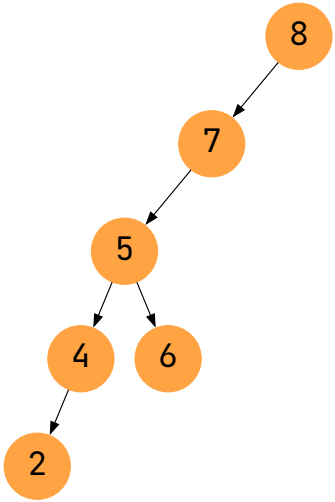
Travessia in-ordem em profundidade

Exemplo

Algoritmo 1: Travessia in-order

```
1 função travessia_inorder(x):
2     se x ≠ nulo então
3         travessia_inorder(x.esquerda);
4         imprima(x.valor);
5         travessia_inorder(x.direita);
6     fim
7 fim
```

Qual a saída para a árvore da figura ao lado?



Algortimo para travessia in-ordem iterativo

Exemplo usando pilha

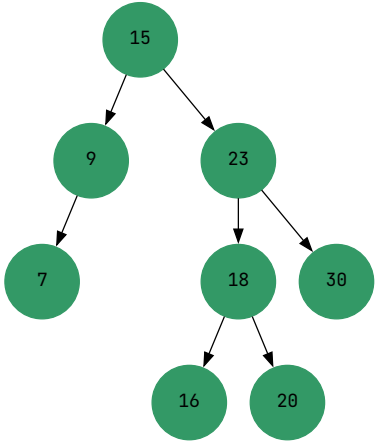
Algoritmo 2: Travessia in-order iterativo

```
1  função travessia_inorder(x):
2      enquanto x ≠ nulo OU pilha ≠ vazia faça
3          enquanto x ≠ nulo faça
4              pilha.empilhe(x);
5              x ← x.esquerda;
6          fim
7          x ← pilha.desempilhe();
8          imprima(x.valor);
9          x ← x.direita;
10     fim
11 fim
```

Travessia ou percurso em largura (*breadth-first*)

Percorrer todos os nós da árvore

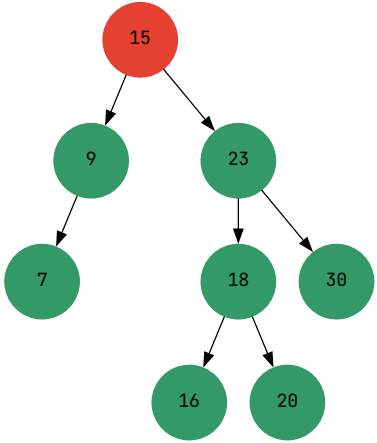
- Percorre todos os nós da árvore nível a nível, da esquerda para a direita
  - `15, 9, 23, 7, 18, 30, 16, 20`
- Não consiste de um algoritmo recursivo e sim iterativo
- É feito uso de fila para armazenar os nós a serem visitados



Travessia ou percurso em largura (*breadth-first*)

Percorrer todos os nós da árvore

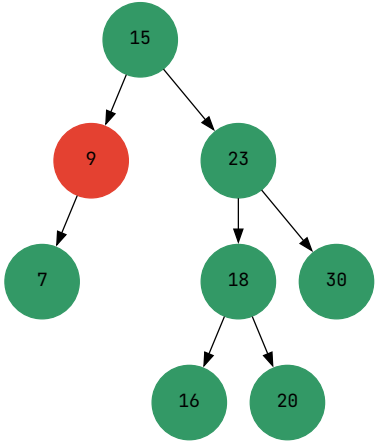
- Percorre todos os nós da árvore nível a nível, da esquerda para a direita
  - `15, 9, 23, 7, 18, 30, 16, 20`
- Não consiste de um algoritmo recursivo e sim iterativo
- É feito uso de fila para armazenar os nós a serem visitados



Travessia ou percurso em largura (*breadth-first*)

Percorrer todos os nós da árvore

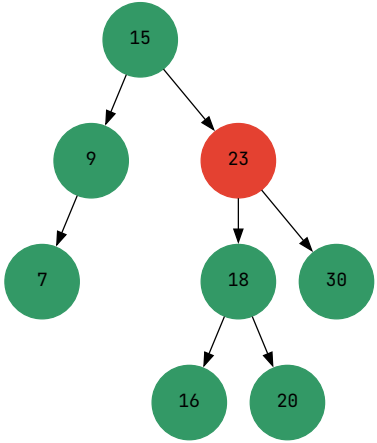
- Percorre todos os nós da árvore nível a nível, da esquerda para a direita
  - `15, 9, 23, 7, 18, 30, 16, 20`
- Não consiste de um algoritmo recursivo e sim iterativo
- É feito uso de fila para armazenar os nós a serem visitados



Travessia ou percurso em largura (*breadth-first*)

Percorrer todos os nós da árvore

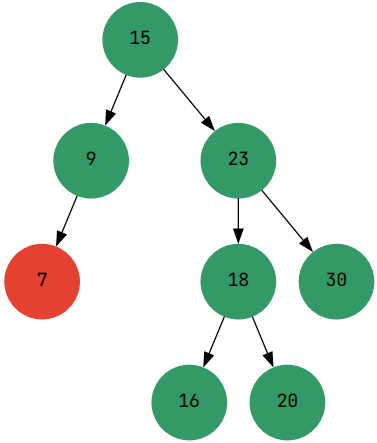
- Percorre todos os nós da árvore nível a nível, da esquerda para a direita
  - `15, 9, 23, 7, 18, 30, 16, 20`
- Não consiste de um algoritmo recursivo e sim iterativo
- É feito uso de fila para armazenar os nós a serem visitados



Travessia ou percurso em largura (*breadth-first*)

Percorrer todos os nós da árvore

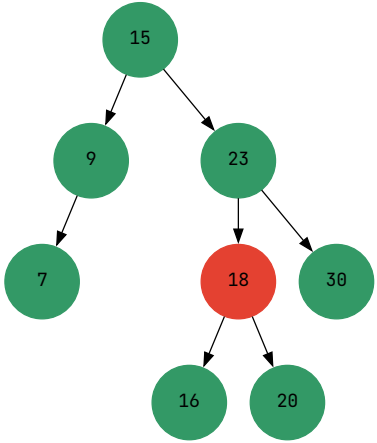
- Percorre todos os nós da árvore nível a nível, da esquerda para a direita
  - `15, 9, 23, 7, 18, 30, 16, 20`
- Não consiste de um algoritmo recursivo e sim iterativo
- É feito uso de fila para armazenar os nós a serem visitados



Travessia ou percurso em largura (*breadth-first*)

Percorrer todos os nós da árvore

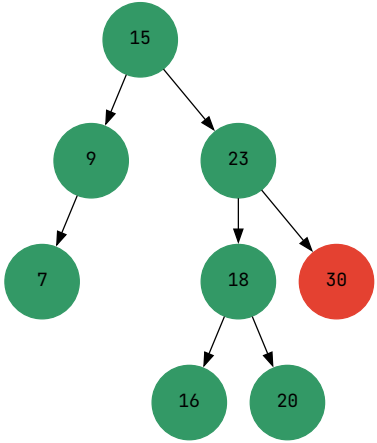
- Percorre todos os nós da árvore nível a nível, da esquerda para a direita
  - `15, 9, 23, 7, 18, 30, 16, 20`
- Não consiste de um algoritmo recursivo e sim iterativo
- É feito uso de fila para armazenar os nós a serem visitados



Travessia ou percurso em largura (*breadth-first*)

Percorrer todos os nós da árvore

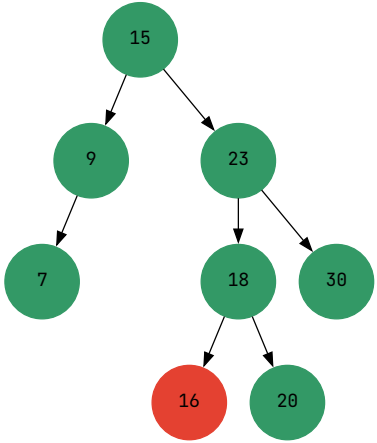
- Percorre todos os nós da árvore nível a nível, da esquerda para a direita
  - `15, 9, 23, 7, 18, 30, 16, 20`
- Não consiste de um algoritmo recursivo e sim iterativo
- É feito uso de fila para armazenar os nós a serem visitados



Travessia ou percurso em largura (*breadth-first*)

Percorrer todos os nós da árvore

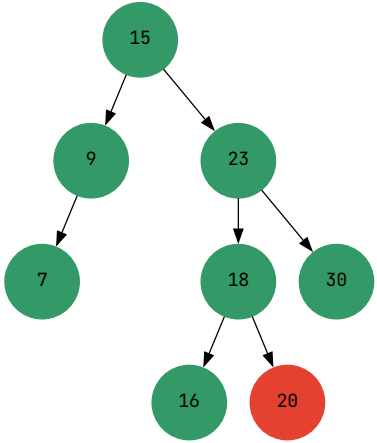
- Percorre todos os nós da árvore nível a nível, da esquerda para a direita
  - `15, 9, 23, 7, 18, 30, 16, 20`
- Não consiste de um algoritmo recursivo e sim iterativo
- É feito uso de fila para armazenar os nós a serem visitados



Travessia ou percurso em largura (*breadth-first*)

Percorrer todos os nós da árvore

- Percorre todos os nós da árvore nível a nível, da esquerda para a direita
  - `15, 9, 23, 7, 18, 30, 16, 20`
- Não consiste de um algoritmo recursivo e sim iterativo
- É feito uso de fila para armazenar os nós a serem visitados

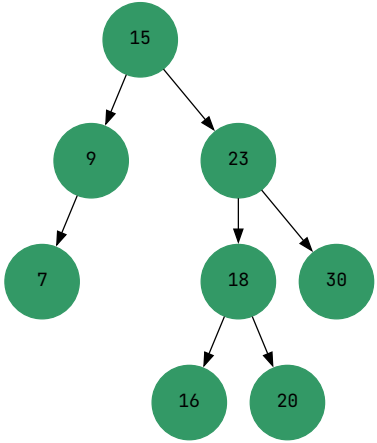


Travessia ou percurso em largura (*breadth-first*)

Percorrer todos os nós da árvore

Algoritmo 3: Percurso em largura

```
1  função percurso_largura(x):
2      enquanto x ≠ nulo faça
3          imprima(x.valor);
4          se x.esquerda ≠ nulo então
5              fila.enfileire(x.esquerda);
6          fim
7          se x.direita ≠ nulo então
8              fila.enfileire(x.direita);
9          fim
10         x ← fila.desenfileire();
11     fim
12 fim
```



Busca em largura ou profundidade

- **Busca em profundidade** (*Depth-First Search*, DFS) é usada para encontrar um nó em particular
  - Geralmente utilizada como uma estratégia de exploração em algoritmos de *backtracking*[1]
  - Usada para encontrar soluções para labirintos ou quebra-cabeças
- **Buscar em largura** (*Breadth-First Search*, BFS) é usada para encontrar o caminho mais curto entre dois nós
  - Usado para encontrar o caminho mais curto em um labirinto ou quebra-cabeça
  - Encontrar amigos de amigos em uma rede social (grau de separação)

[1] https://pt.wikipedia.org/wiki/Backtracking

Exercícios

Exercício 1

Aproveite as implementações das aulas anteriores e implemente funções para

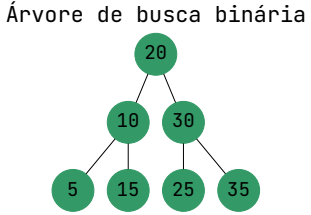
1. Percorrer uma árvore binária em pré-ordem, in-ordem e pós-ordem
   - Imprimir os valores dos nós
2. Percorrer uma árvore binária em largura
   - Imprimir os valores dos nós

Exercício 2

- Imprimir a árvore como um grafo usando a *dot language*[2]
  - **Dica:** Lógica parecida com a usada para imprimir a árvore em pré-ordem
- Para visualizar, use o site https://dreampuf.github.io/GraphvizOnline/ ou a extensão *Graphviz Preview* do VSCode (é necessário instalar o Graphviz)

```
strict graph{
  label="Árvore de busca binária";
  node [shape="circle", color="#339966", style="filled",
     fixedsize=true];

  20 -- 10;
  20 -- 30;
  10 -- 5;
  10 -- 15;
  30 -- 25;
  30 -- 35;
}
```



[2] https://graphviz.org/doc/info/lang.html

Exercício 3

Segundo Cormen et al. (2012) a altura esperada de uma árvore de busca binária construída de forma aleatória é $O(\log n)$

- Usando o código que fez na última aula, crie uma árvore de busca binária com 1000 nós e comprove a afirmação acima
- É importante que o intervalo de valores sorteados seja grande o suficiente para que a árvore não tenha muitos nós repetidos

Referências

Aula baseada em

- CORMEN, Thomas H. et al. **Algoritmos: teoria e prática**. LTC, 2012. Disponível em: <https://app.minhabiblioteca.com.br/reader/books/9788595158092>.
- LAGO PEREIRA, Silvio do. **Estruturas de Dados em C - Uma Abordagem Didática**. Saraiva, 2016. ISBN 9788536517254. Disponível em: <https://app.minhabiblioteca.com.br/#/books/9788536517254>. Acesso em: 1 nov. 2023.
- SZWARCFITER, Jayme Luiz; MARKENZON, Lilian. **Estruturas de dados e seus algoritmos**. LTC, 2010. Disponível em: <https://app.minhabiblioteca.com.br/reader/books/978-85-216-2995-5>.