Universidade Federal da Bahia
Departamento de Ciência da Computação

Árvore

Ricardo Araújo Rios

TAD Árvore

Introdução

Introdução

- A utilização de estruturas de dados organizadas através de encadeamento fornece uma flexibilidade maior do que vetores.
- Entretanto, Listas, Pilhas e Filas são estruturas lineares.
- Como representar elementos de maneira hierárquica?
- Uma forma não-linear de organizar elementos de maneira hierárquica é utilizando “árvore”.

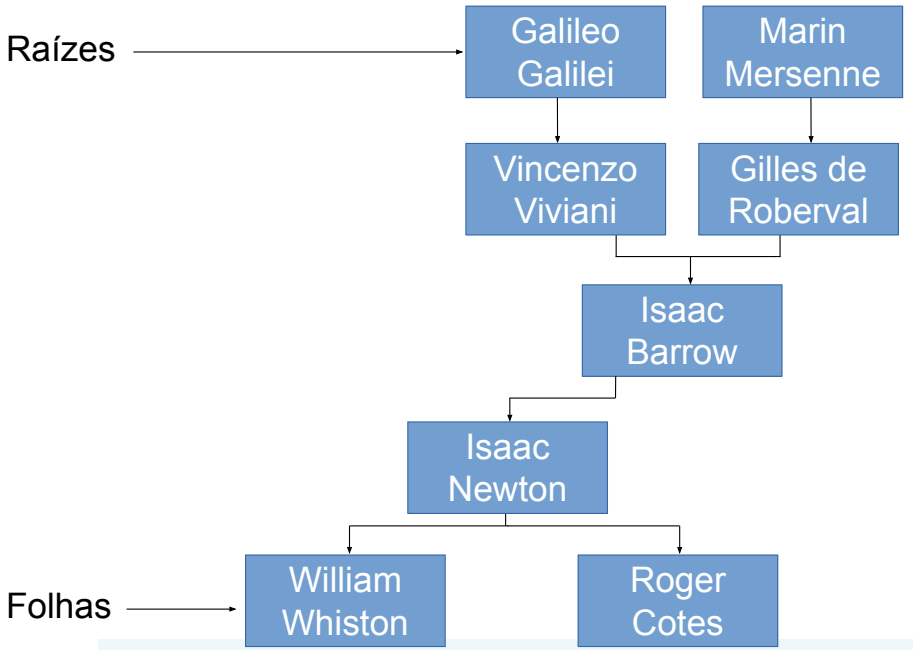
Introdução

- Estrutura de dados árvore é composta por nós e arcos.
- Ao contrário das árvores naturais, essa estrutura é representada de maneira invertida.
- A raiz está localizada no topo, enquanto que as folhas na parte mais baixa dessa representação.

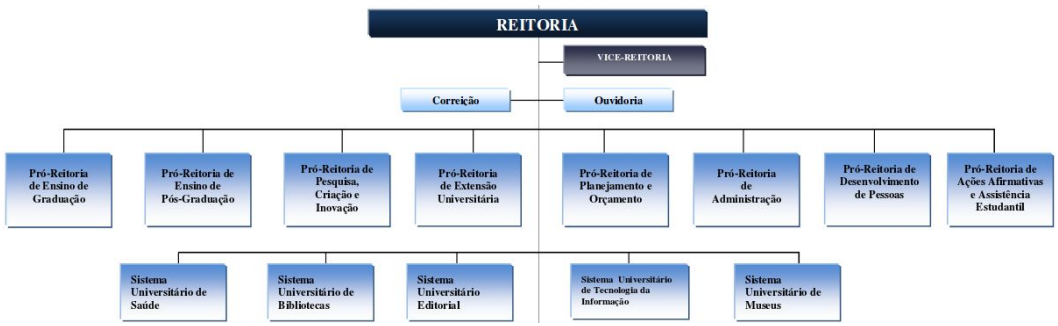
Introdução

- O que é a raiz em um TAD Árvore?
  - É um nó que não possui pais. Esse nó possui apenas filhos.
- E o que são folhas?
  - Nós que não possuem filhos.

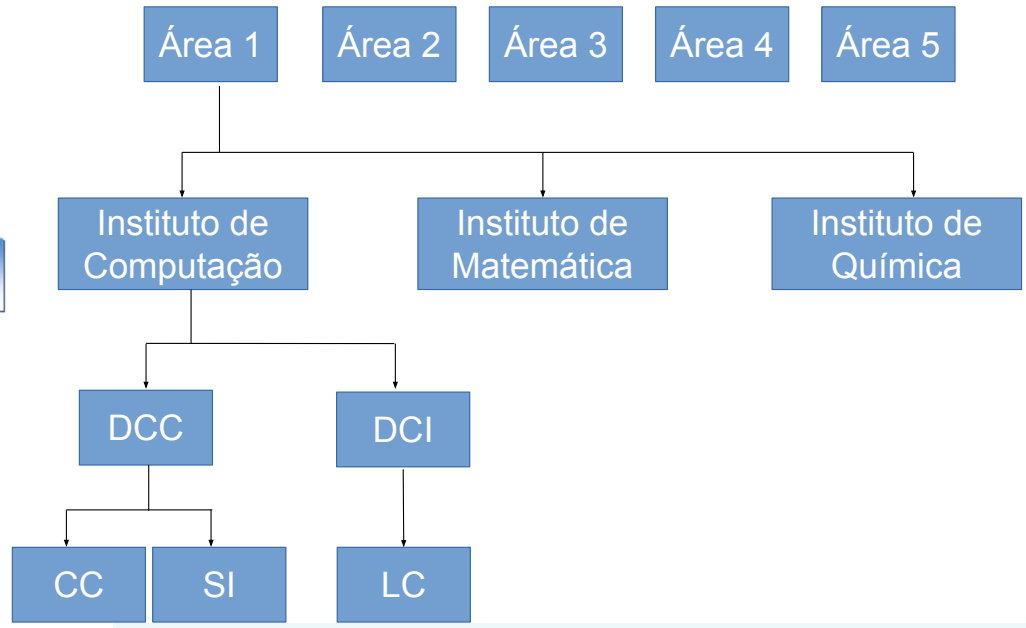
Mathematics Genealogy Tree



Estrutura UFBA



Unidades Universitárias



Introdução

- Vantagens:
  - Representatividade no relacionamento entre os dados;
  - Facilidade na manipulação computacional dos dados;
  - Exemplo:
    - Qual a quantidade de professores no DCC?
    - Qual o total de alunos do IC?

Introdução

- Observe que para recuperar informações de uma árvore, não é necessário percorrer todos os elementos:
  - Consulta seletiva em regiões específicas da árvore;
  - Para chegar a um nó, deve-se encontrar um caminho percorrendo os relacionamentos entre os elementos;

TAD Árvore

Representação

Representação

- Uma árvore ***T*** é um conjunto finito de elementos, chamados de nós ou vértices, tal que:
  - ***T = Ø,*** árvore vazia;
  - ***T = {r} U {T_1} U {T_2} U {T_3} U {T_4} U ... U {T_n}***;
    - ***{r}*** é um nó especial da árvore chamado raiz;
    - ***{T_1}, {T_2}, {T_3}, {T_4}, ... , {T_n}*** são subárvores de ***{r}***;

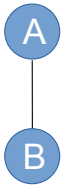
Representação

- Logo, uma árvore pode ser recursivamente representada por:
  - $T = \{r, T_1, T_2, T_3, T_4, ..., T_n\}$;
- Uma sequência de chaves pode ser utilizada para representar o relacionamento entre os nós de uma árvore.
- Exemplo:

Representação

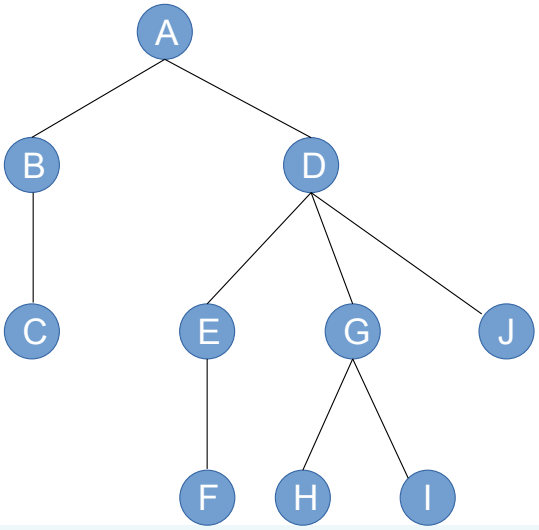
- T = {A}

- T = {A, {B}}



Representação

- T = {A, {B, {C}}, {D, {E, {F}}, {G, {H},{I}}, {J}}}

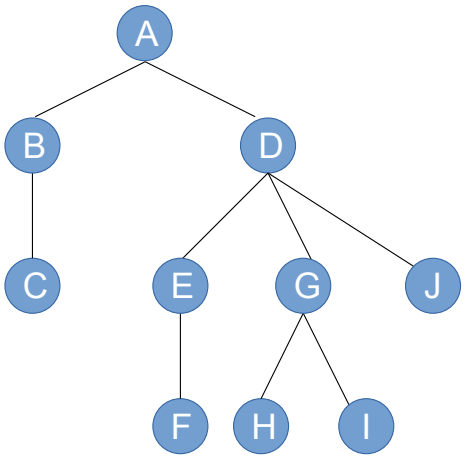


Representação

- Exercício: Represente graficamente as seguintes árvores:
  - T = {2, {1}, {3}}
  - T = {4, {2, {1}, {3}}, {6, {5}, {7}}}
  - Considerando a árvore genealógica apresentada no exemplo anterior, qual seria a representação para os nós: Cotes, Whiston, Newton, Barrow, Vincenzo e Galileo?

Representação

- Grau de um nó
  - Representa o número de subárvores de um nó.



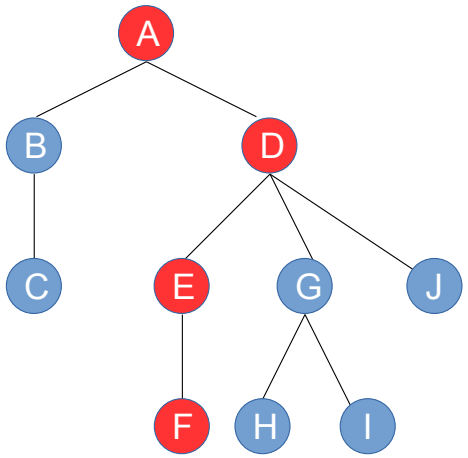
Nó	Grau
A	2
B	1
D	3
G	2
H	0

Representação

- O grau de uma árvore é o maior grau dentre todos os nós da árvore.
  - Qual o grau da árvore anterior?
    - Resposta: 3
- Subárvores de grau 0 são chamadas de folhas.
- Para representar a hierarquia em árvores, nós podem ser chamados de:
  - Pai
  - Filho
  - Irmão

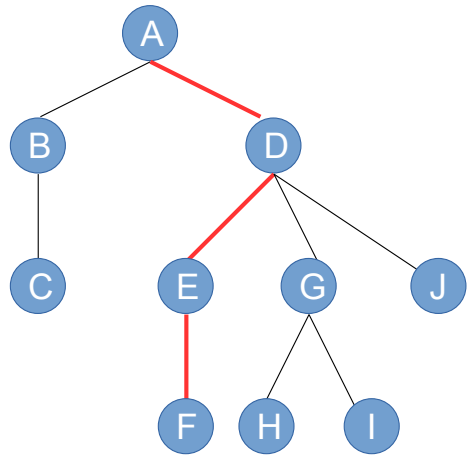
Representação

- Um caminho de uma árvore é uma sequência não vazia de nós.
  - Exemplo:
    - P(A,F) = {A, D, E, F}



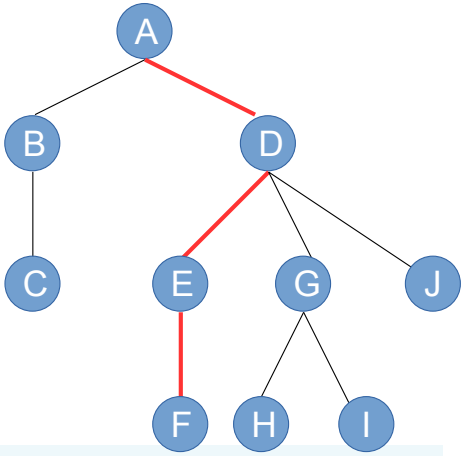
Representação

- O comprimento de um caminho é o número de arcos entre o nó inicial e final do caminho, ou seja, o número de nós -1.
  - Exemplo:
    - L(P) = 3



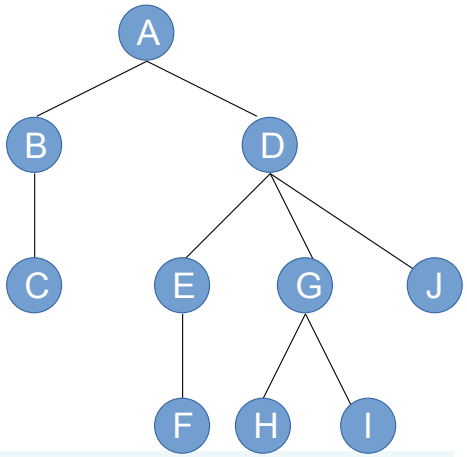
Representação

- A altura de um nó é o comprimento do maior caminho entre o nó e uma folha.
- A altura de uma árvore é a altura de sua raiz.
  - Exemplo:
    - altura(A) = altura(T) = 3



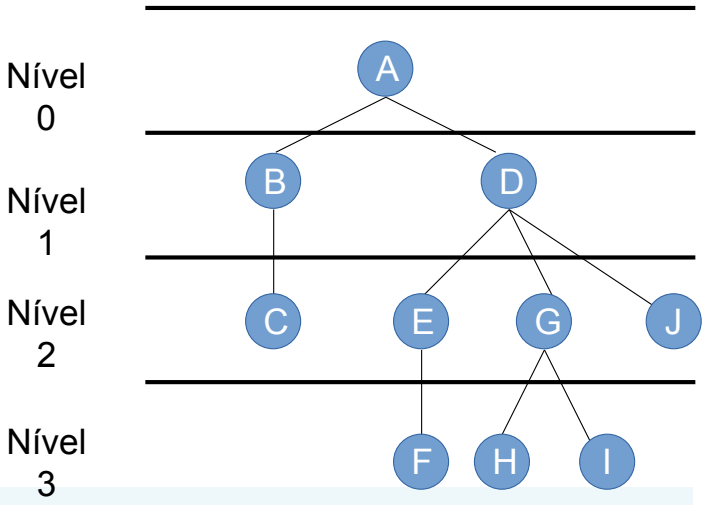
Representação

- A profundidade de um nó r_n é o comprimento do caminho entre o nó raíz e r_n.
  - Exemplo:
    - altura(E) = 1
    - profundidade(E) = 2



Representação

- O conjunto de nós de mesma profundidade é chamado de nível.
  - Exemplo:



Exercício

- Considere a seguinte árvore:
  - T = {A, {B, {C, {D}}, {E, {F }, {G}}}, {H, {I}}}
- Responda:
  - Qual a representação gráfica dessa árvore?
  - Encontre o grau, altura e profundidade de cada nó.
  - Separe os nós dessa árvore por níveis.
  - Existe um caminho entre os nós B e D? Qual o seu comprimento?
  - Existe um caminho entre os nós B e I? Qual o seu comprimento?

Exercício

- Considerando um TAD Árvore, responda:
  - Como você implementaria uma estrutura árvore?
  - Suponha que você deseja implementar uma função para transformar uma lista em uma árvore. De acordo com sua implementação e considerando que cada nó pode ter no máximo 3 filhos, qual seria a estrutura de árvore para a seguinte lista?
    - L = {1, 2, 3, 4, 5, 6, 7, 8, 9}

Árvore Binária

- Introdução
- Implementação
- Percorrendo Árvore Binária

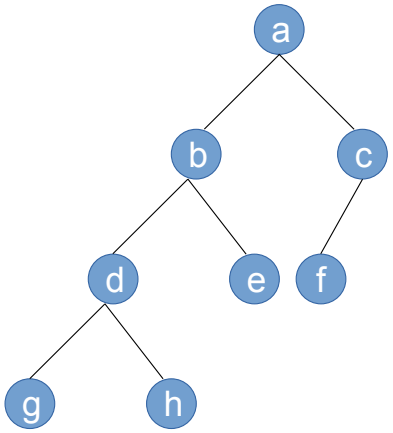
Árvore Binária

Introdução

Introdução

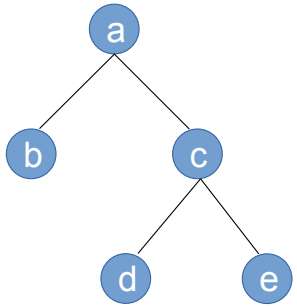
- Árvore Genérica (n-árias)
  - Exemplo: Estrutura de Arquivos em Sistemas Operacionais
- Uma Árvore Binária T é um conjunto finito de elementos (nós ou vértices) tal que:
  - Se **T = 0**, a árvore é vazia;
  - **T** contém um nó especial **r** chamado raiz e todos os demais nós podem ser subdivididos em duas subárvores:
    - $\mathbf{T_E}$: Subárvore esquerda
    - $\mathbf{T_D}$: Subárvore direita

Introdução



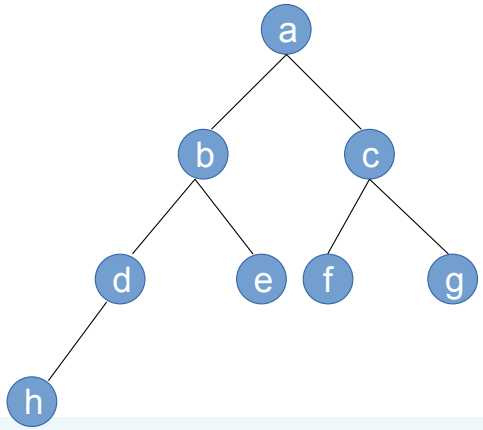
Introdução

- Árvore Estritamente Binária:
  - Os nós possuem 0 (nenhum) ou 2 filhos;
  - Os nós não-folhas possuem obrigatoriamente 2 filhos;



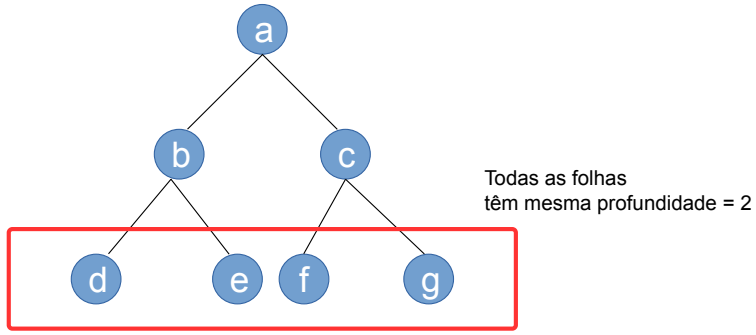
Introdução

- Árvore Binária Completa:
  - Se a profundidade da árvore é **d**, então todos os nós folhas estão no nível **d-1** ou **d**;
  - O nível **d-1** está completamente preenchido;
  - Os nós folhas do nível **d** estão mais à esquerda;



Introdução

- Árvore Binária Cheia:
  - Toda árvore cheia é completa e estritamente binária;
  - Todos os nós folhas estão no mesmo nível;

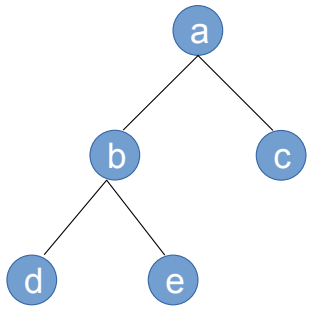


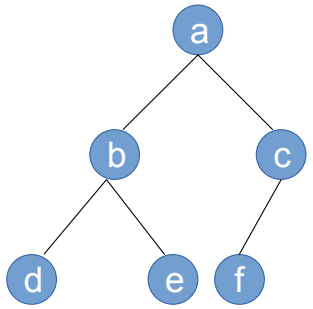
Introdução

- Árvore Binária Cheia:
  - Relação Nós x Profundidade:
    - d = 0 → 1 nó
    - d = 1 → 3 nós
    - d = 2 → 7 nós
    - …
    - A quantidade de nós pode ser calculada pela profundidade da árvore utilizando a seguinte fórmula:
      - $2^{(d+1)} - 1$
    - De maneira semelhante temos:
      - $d = \log_2(n+1) - 1$

Introdução

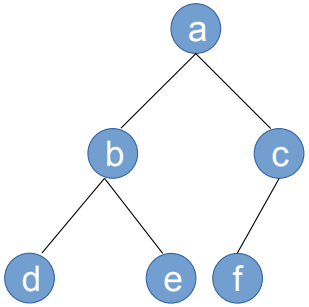
- Árvore Binária Balanceada:
  - Para cada nó, as alturas de suas subárvores diferem em, no máximo, 1;





Introdução

- Árvore Binária Perfeitamente Balanceada:
  - Para cada nó, o número de nós de suas subárvores esquerda e direita diferem em, no máximo, 1;
  - Toda Árvore Binária Perfeitamente Balanceada é Balanceada, mas o inverso não é necessariamente verdade;

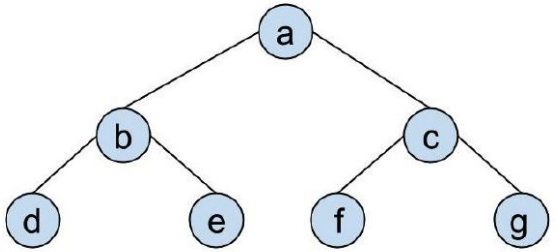


Árvore Binária

Implementação

Implementação

- Árvore Binária Estática:
  - Armazena os nós por níveis em um array:



0	1	2	3	4	5	6		n
a	b	c	d	e	f	g		...

Implementação

- Árvore Binária Estática:
  - Para um vetor indexado a partir da posição 0, se um nó está na posição i, então seus filhos estão nas posições:
    - 2i + 1 → filho da esquerda
    - 2i + 2 → filho da direita
  - Vantagem:
    - Não é necessário armazenar os ponteiros para os filhos;
  - Desvantagem:
    - Possibilidade de espaços vazios no vetor

Implementação

- Árvore Binária Dinâmica:
  - Ver código

Árvore Binária

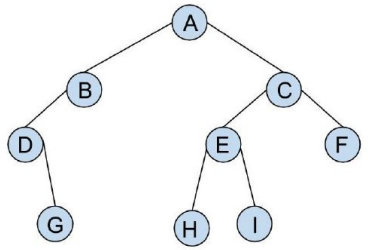
Percorrendo Árvore Binária

Percurso

- Árvore Binária:
  - Percorrer uma árvore binária significa visitar seus nós apenas uma vez para, por exemplo, imprimir ou modificar seu valor.
  - Um percurso gera uma sequência linear de nós visitados;
  - Exemplo de métodos para percorrer uma árvore:
    - Pré-ordem (pre-order)
    - Em-ordem (in-order)
    - Pós-ordem (post-order)

Percurso

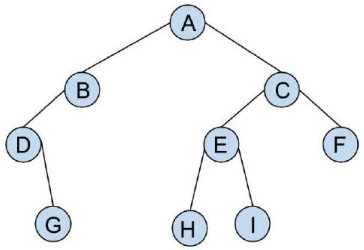
- Pré-ordem (pré-order):
  - Visita a raiz;
  - Percorre a árvore da esquerda em pré-ordem;
  - Percorre a árvore da direita em pré-ordem;



- Resultado: ABDGCEHIF

Percurso

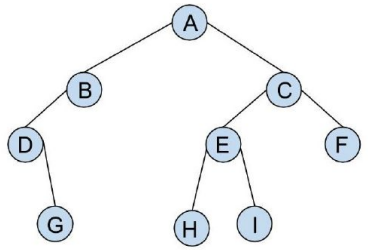
- Em-ordem (in-order):
  - Percorrer a árvore da esquerda em in-order;
  - Visita a raiz;
  - Percorrer a árvore da direita em in-order;



- Resultado: DGBAHEICF

Percurso

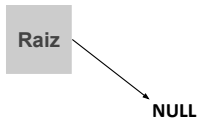
- pós-ordem (post-order):
  - Percorrer a árvore da esquerda em pós-ordem;
  - Percorrer a árvore da direita em pós-ordem;
  - Visita a raiz;



- Resultado: GDBHIEFCA

Insere

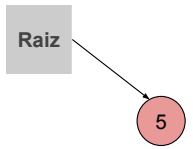
- Algoritmo



Inserir: 5

Insere

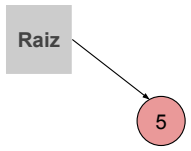
- Algoritmo



Inserir: 5

Insere

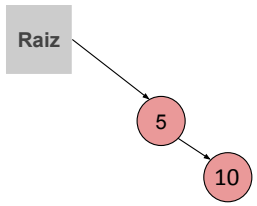
- Algoritmo



Inserir: 10

Insere

- Algoritmo

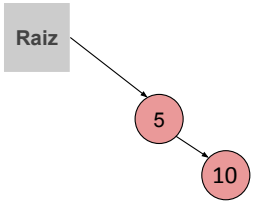


Inserir: 10

Insere

- Algoritmo

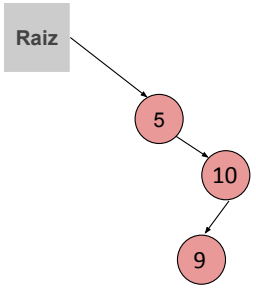
Inserir: 9



Insere

- Algoritmo

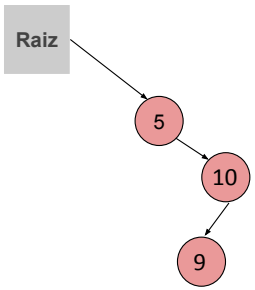
Inserir: 9



Insere

- Algoritmo

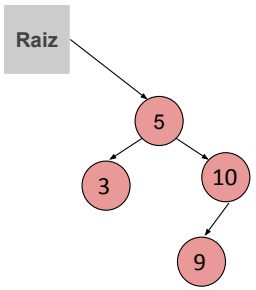
Inserir: 3



Insere

- Algoritmo

Inserir: 3

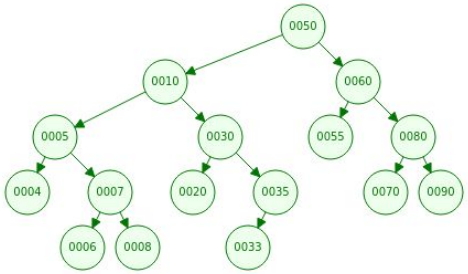


Remove

- Algoritmo: remover um nó *z*
  - Casos:
    1. Se *z* não tem filhos, pai substitui *z* por *NULL*
    2. Se *z* tem apenas um filho, o filho ocupa o lugar de *z*
    3. Se *z* tem dois filhos, a partir da subárvore à esquerda, procure o antecessor de *z* e o utilize no lugar de *z*. Se *z* tiver um filho *y* à esquerda, a subárvore à direita do pai de *z* deve apontar para *y* .

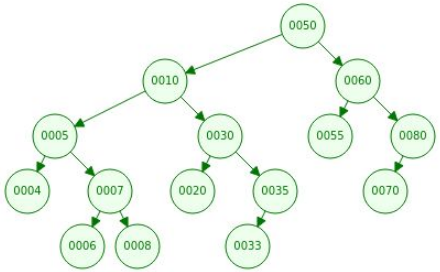
Remove

- Algoritmo: remover um nó *z*
  - Casos:
    1. Se *z* não tem filhos, pai substitui *z* por *NULL*
       - `Exemplo: remover 90`



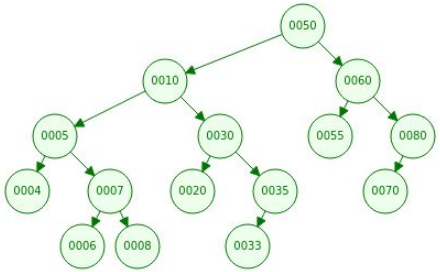
Remove

- Algoritmo: remover um nó *z*
  - Casos:
    1. Se *z* não tem filhos, pai substitui *z* por *NULL*
       - `Exemplo: remover 90`



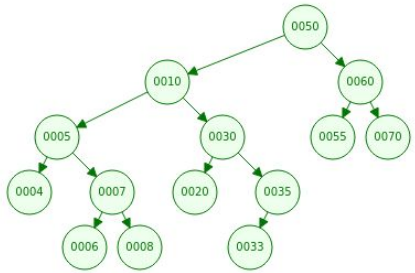
Remove

- Algoritmo: remover um nó *z*
  - Casos:
    2. Se *z* tem apenas um filho, o filho ocupa o lugar de *z*
       - `Exemplo: remover 80`



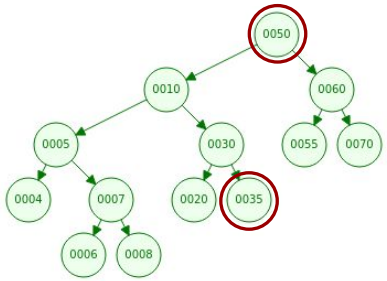
Remove

- Algoritmo: remover um nó z
  - Casos:
    2. Se z tem apenas um filho, o filho ocupa o lugar de z
       - `Exemplo: remover 80`



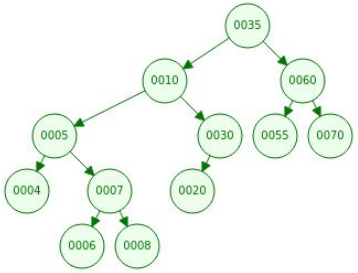
Remove

- Algoritmo: remover um nó z
  - Casos:
    3. Se z tem dois filhos, a partir da subárvore à esquerda, procure o antecessor w de z e o utilize no lugar de z. Se w tiver um filho y à esquerda, a subárvore à direita do pai de w deve apontar para y.
       - `Exemplo: remover 50`



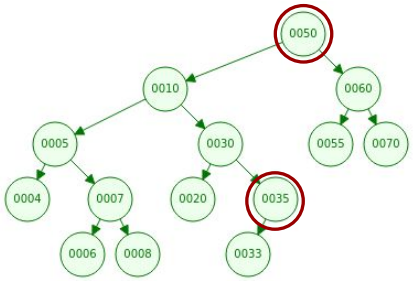
Remove

- Algoritmo: remover um nó z
  - Casos:
    3. Se z tem dois filhos, a partir da subárvore à esquerda, procure o antecessor w de z e o utilize no lugar de z. Se w tiver um filho y à esquerda, a subárvore à direita do pai de w deve apontar para y.
       - `Exemplo: remover 50`



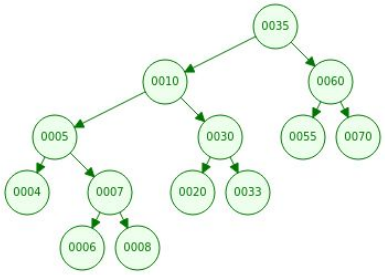
Remove

- Algoritmo: remover um nó z
  - Casos:
    3. Se z tem dois filhos, a partir da subárvore à esquerda, procure o antecessor w de z e o utilize no lugar de z. Se w tiver um filho y à esquerda, a subárvore à direita do pai de w deve apontar para y.
       - `Exemplo: remover 50`



Remove

- Algoritmo: remover um nó z
  - Casos:
    3. Se z tem dois filhos, a partir da subárvore à esquerda, procure o antecessor w de z e o utilize no lugar de z. Se w tiver um filho y à esquerda, a subárvore à direita do pai de w deve apontar para y .
       - `Exemplo: remover 50`



Referências

[1] Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C., Algoritmos – Teoria e Prática, 2ª Edição, Elsevier, 2002;

[2] Kleinberg, J., Tardos, E., Algorithm Design, Pearson, 2006;

[3] Goodrich, M. T., Tamassia, R., Estrutura de Dados e Algoritmos em Java, 4ª Edição, Bookman, 2007;

[4] Ziviani, N., Projeto de algoritmos com implementações em Java e C++, Thomson, 2007