UNIVERSIDADE FEDERAL DE ITAJUBÁ

Algoritmos e Estrutura de Dados I

COM 111

Árvore Binária de Pesquisa

Vanessa Cristina Oliveira de Souza

Introdução

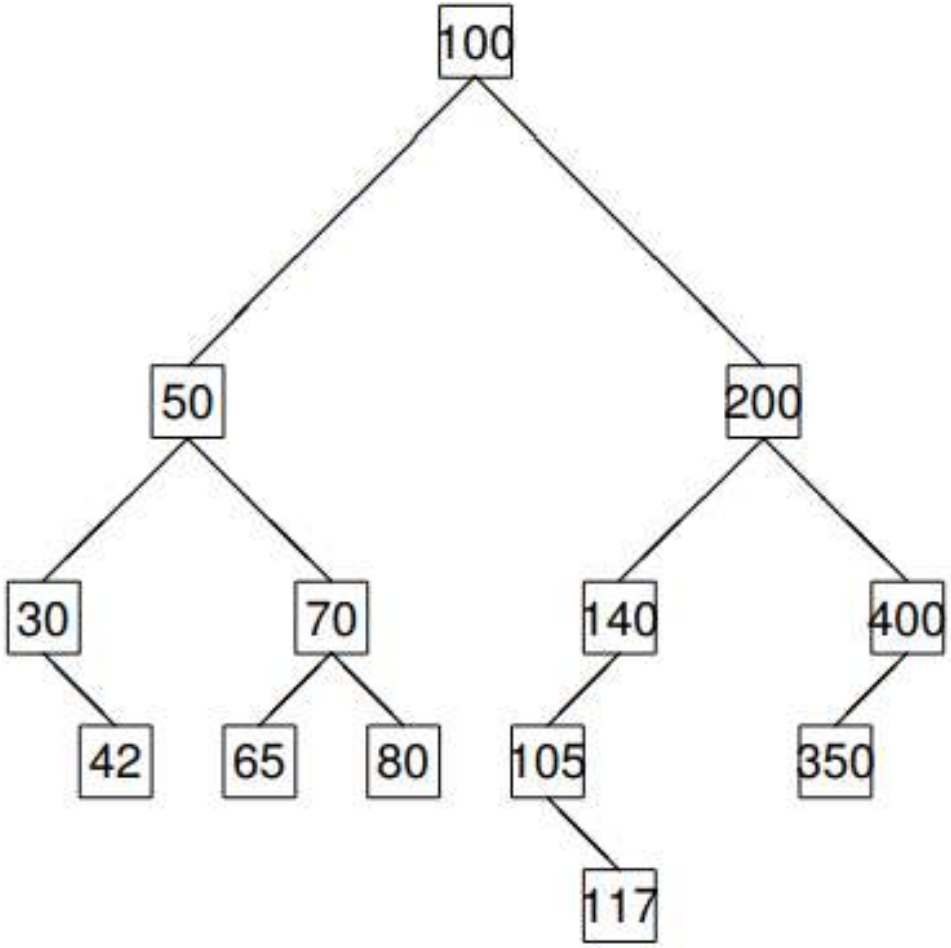
- A árvore de pesquisa é uma **estrutura de dados** muito eficiente para armazenar e recuperar informação.
  - Uma árvore binária de busca serve para o armazenamento de dados na memória principal do computador e a sua subsequente recuperação.

Introdução

- Uma árvore binária é definida como um conjunto finito de **nós** que ou está **vazio** ou consiste de um nó chamado **raiz** mais os elementos de duas árvores binárias distintas chamadas de **subárvores esquerda e direita** do nó raiz.

- Em uma árvore binária:
  - Cada nó possui uma chave
  - Cada nó tem no máximo duas subárvores.

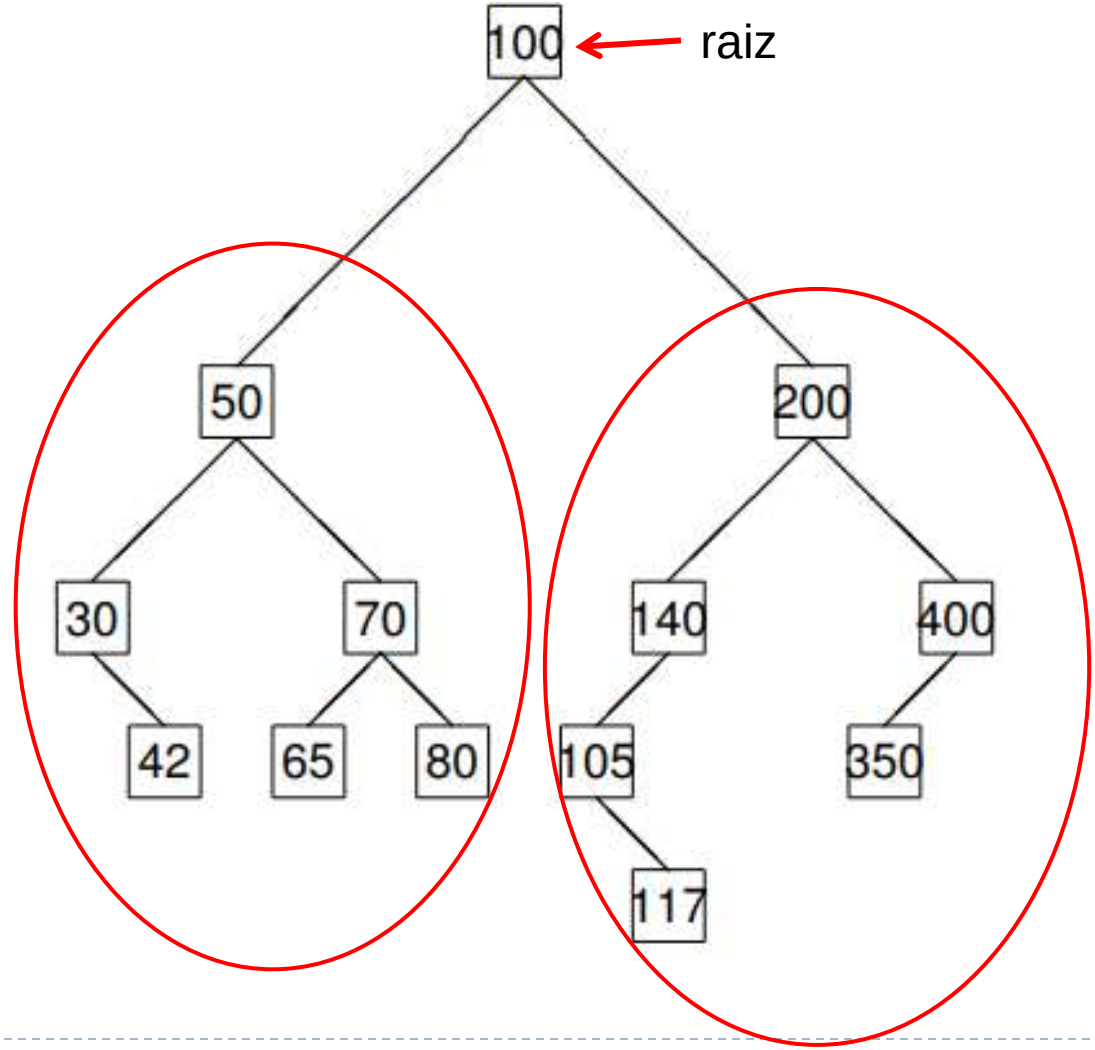
Introdução



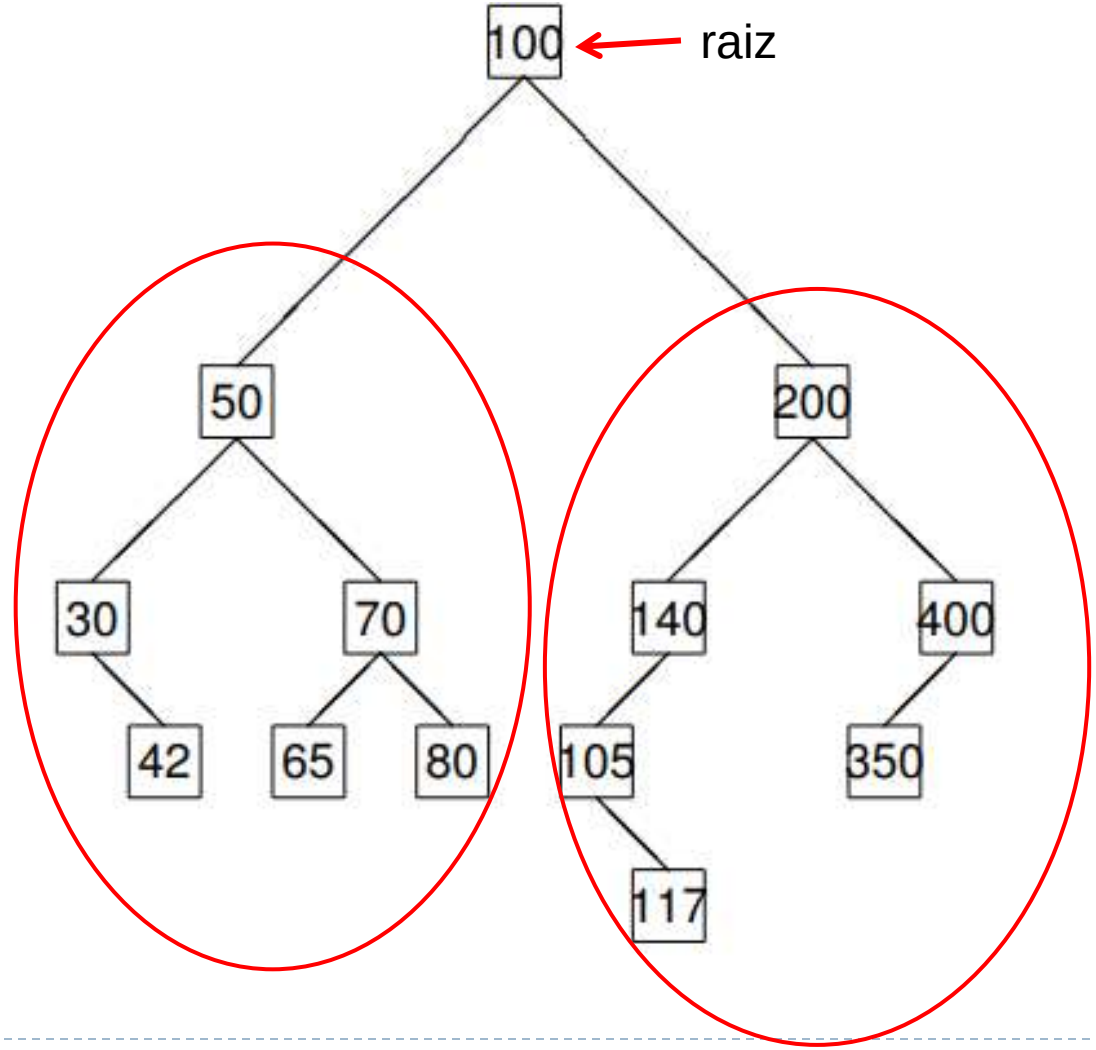
FONTE : https://www.ime.usp.br/~song/mac5710/slides/06bst.pdf

Propriedades da Árvore Binária

Propriedades



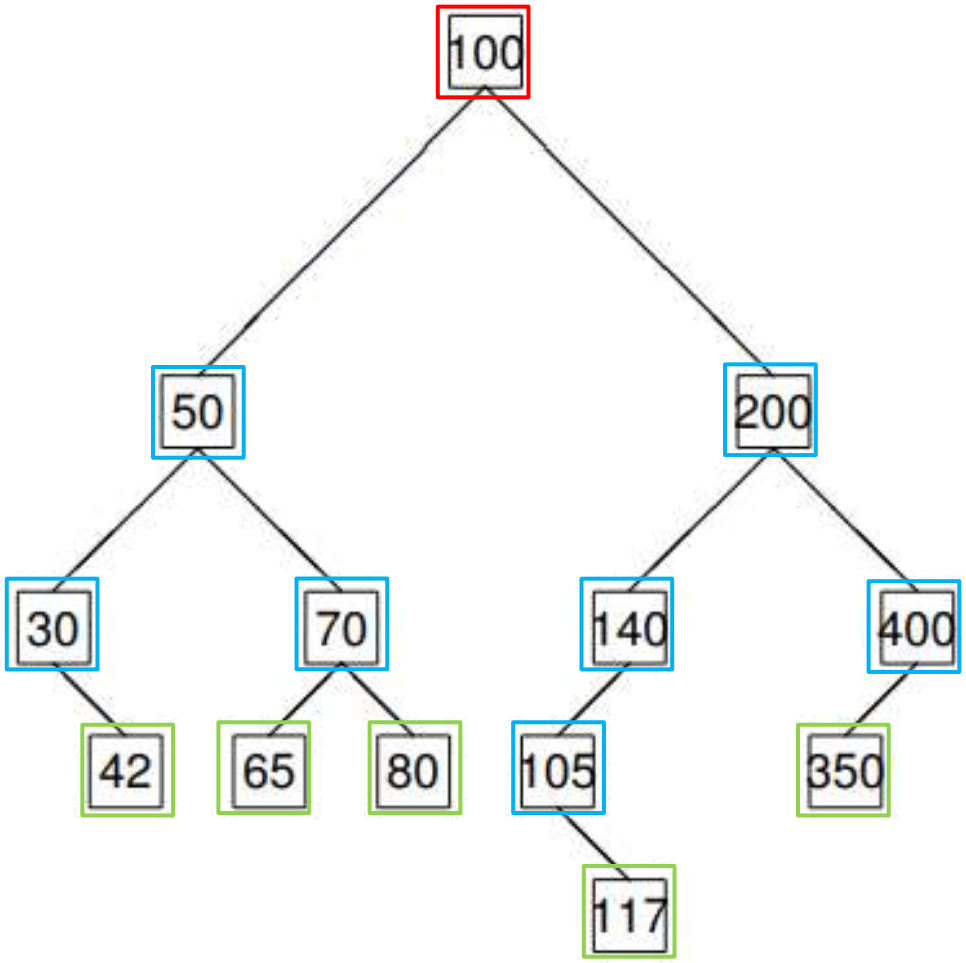
Propriedades



Propriedades

- O número de subárvores de um nó é chamado de grau daquele nó:
  - Nó de grau zero
    - Não tem filhos
    - É chamado de nó externo ou folha
  - Nó de grau 1
    - Possui apenas um filho (ou o da esquerda, ou o da direita)
  - Nó de grau 2
    - Possui os dois filhos (esquerda e direita)

Propriedades



Nó raiz

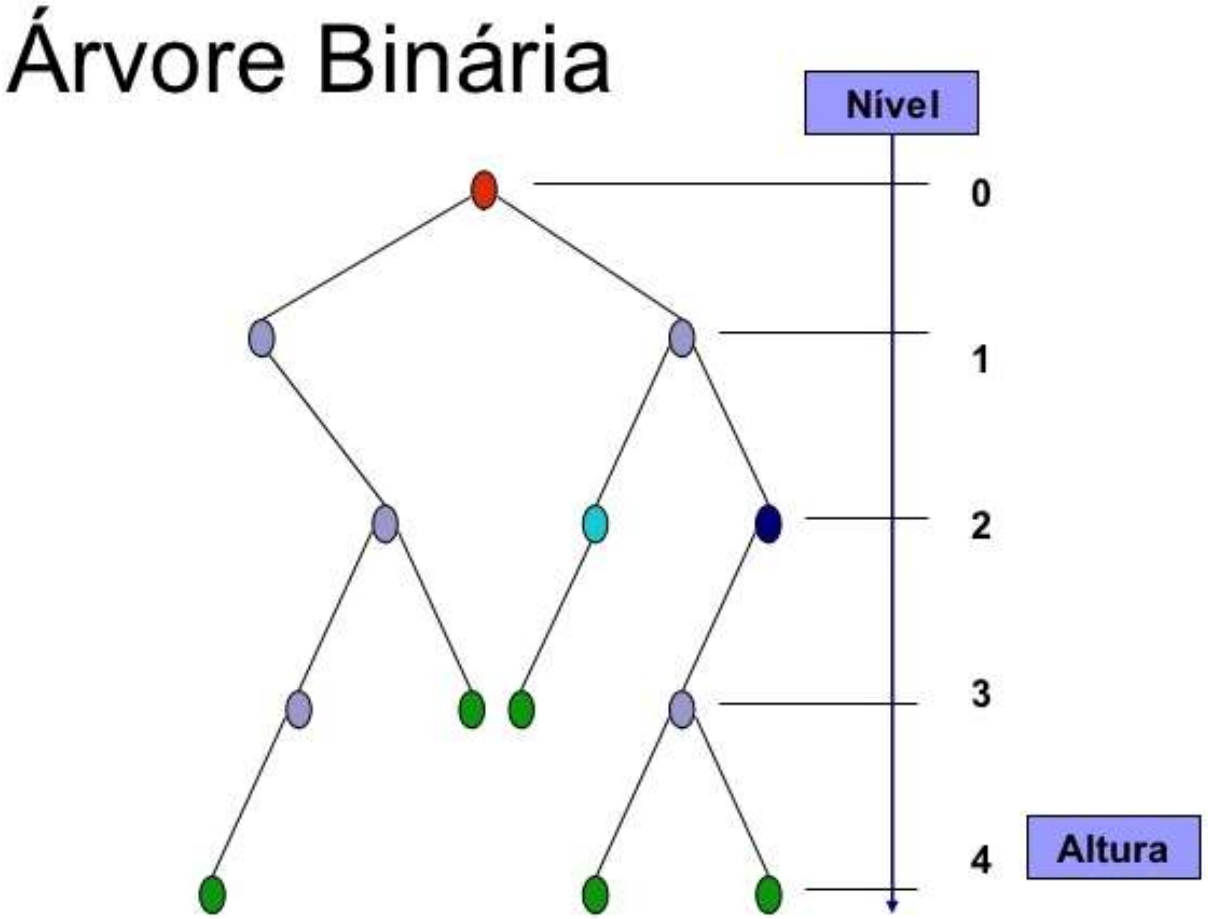
Nós Intermediários

Nós folha

Propriedades

- Altura da Árvore Binária
  - Comprimento do caminho mais longo da raiz até uma das folhas
  - A altura da raiz é **zero**
  - A altura de uma árvore vazia é -1
  - Determina o **esforço computacional** para encontrar um determinado nó

Propriedades



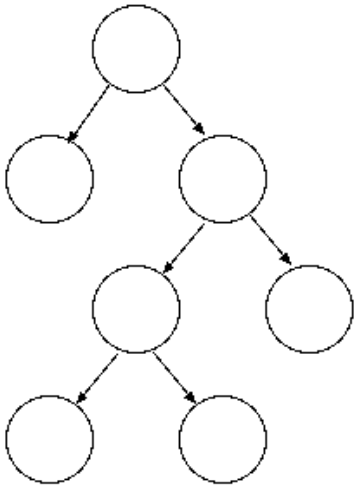
FONTE : https://pt.slideshare.net/trickrow/grafos-e-rvores

Classificação de Árvores Binárias

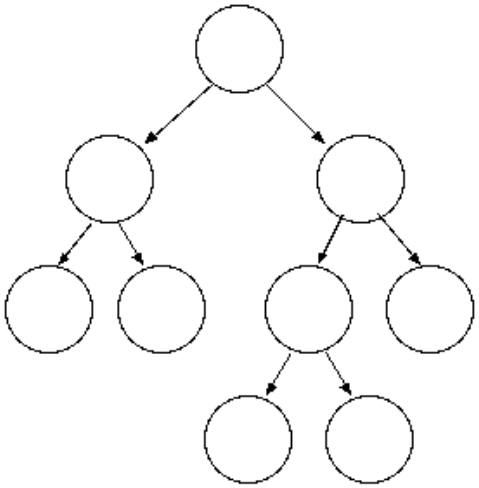
Classificação de Árvores Binárias

- Uma árvore binária pode ser:
  - Estritamente binária
    - Seus nós são de grau 0 ou grau 2
  - Completa
    - Se existem nós de grau 0, ou eles estão no penúltimo, ou no último nível
  - Cheia
    - Se existem nós de grau 0, eles estão no último nível da árvore

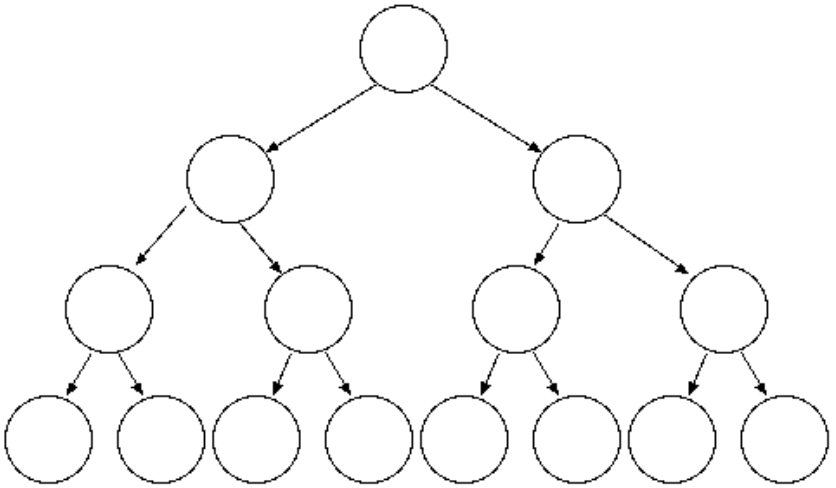
Classificação de Árvores Binárias



(a) Árvore estritamente binária



(a) Árvore binária completa

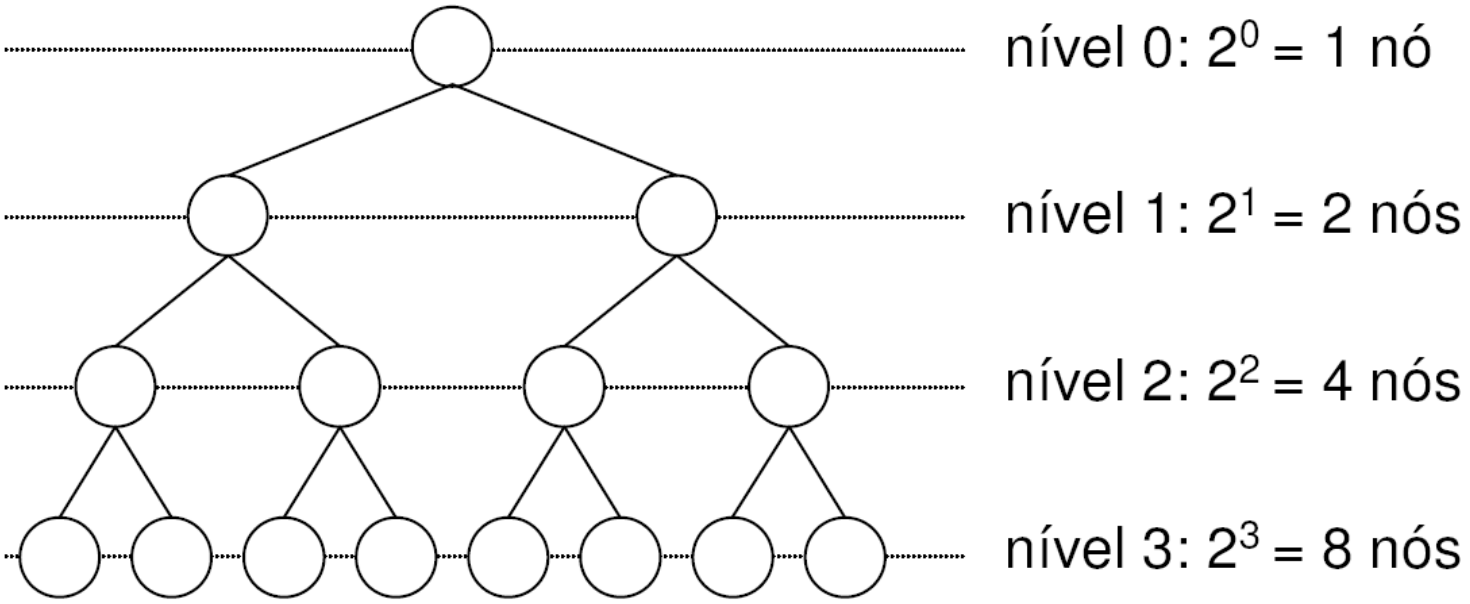


(a) Árvore binária cheia

Toda árvore cheia é completa e estritamente binária.

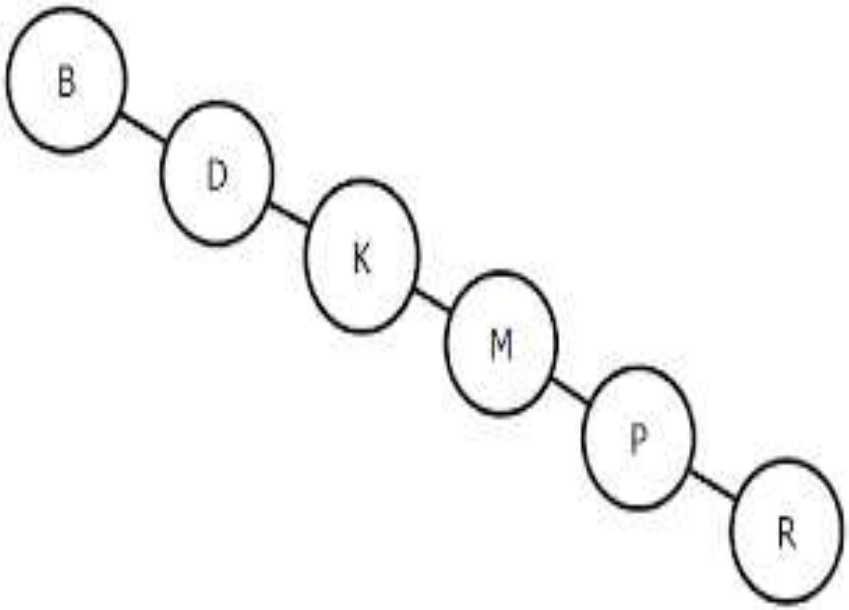
Classificação de Árvores Binárias

- **Árvore cheia**
  - É uma árvore completamente balanceada.
  - Complexidade $O(\log_2(n))$



Classificação de Árvores Binárias

- **Árvore Degenerada**
  - É uma árvore completamente **desbalanceada**, ou seja, torna-se uma **lista**.
  - A altura da árvore é n
  - Complexidade O(n)



Estrutura do TAD Árvore Binária de Pesquisa

TAD ABP

Estrutura

- O nó da árvore binária possui a chave e dois ponteiros : um para a subárvore esquerda e outro para a subárvore direita

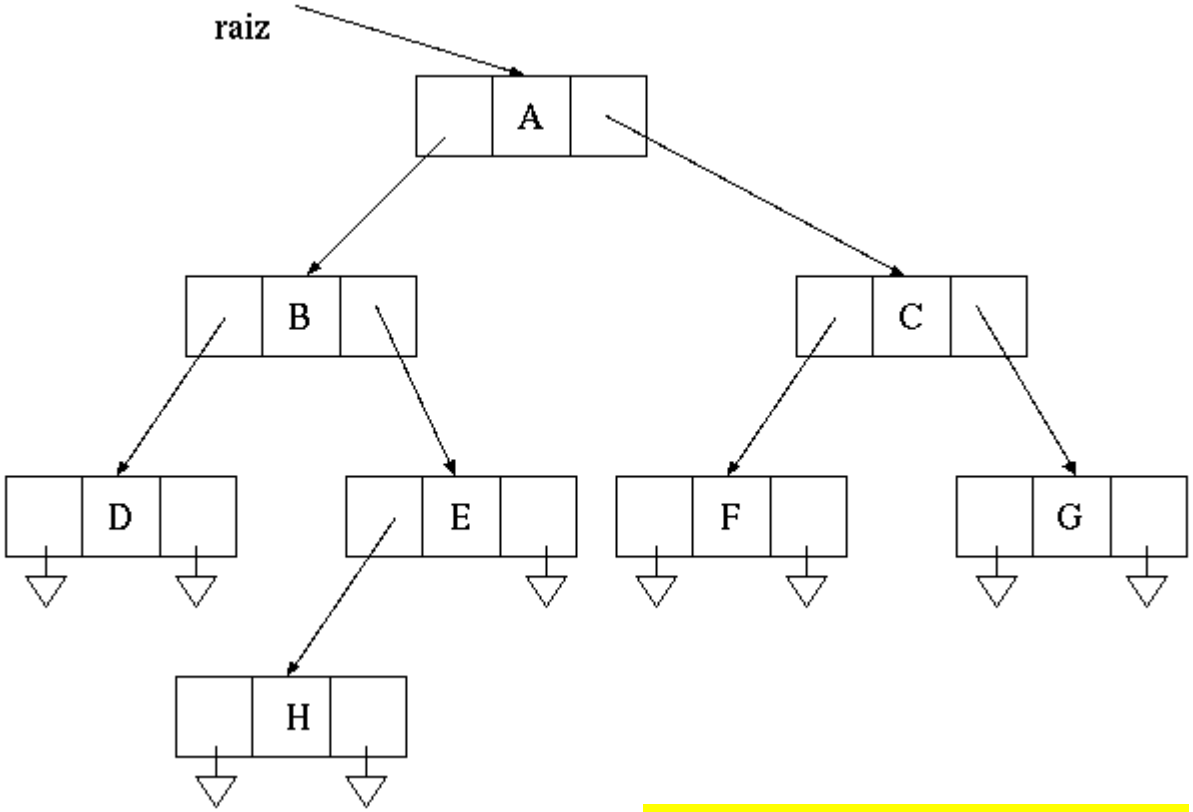
```
struct noArvore
{
    int chave;
    struct noArvore *esq;
    struct noArvore *dir;
}
```

Estrutura

- O tipo árvore é identificado pelo seu nó raiz
  - Se a árvore está vazia, a raiz é NULL

```
struct arvore
{
    struct noArvore *raiz;
}
```

Estrutura



A árvore está errada!!!!!

FONTE : http://equipe.nce.ufrj.br/adriano/c/apostila/arvore.htm

Operações em Árvores Binárias

TAD ABP

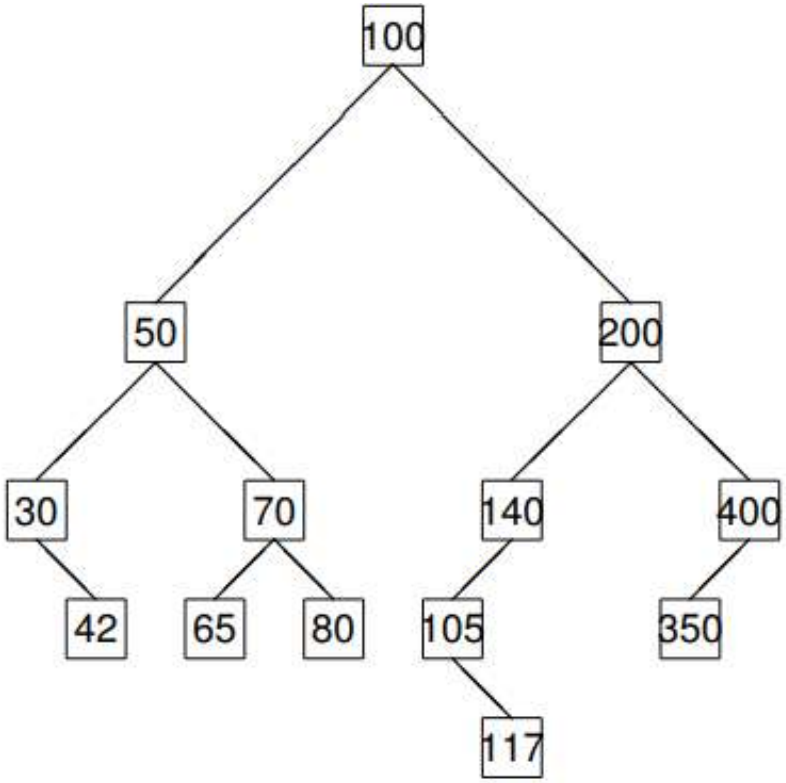
- As principais operações sobre as árvores binárias são:
  - Inserir um elemento
  - Excluir um elemento
  - Buscar um elemento
  - Encontrar o maior
  - Encontrar o menor
  - Buscar o elemento sucessor e o predecessor
  - Percorrimento
    - Em pré-ordem
    - Em ordem
    - Em pós-ordem

TAD ABP

- As principais operações sobre as árvores binárias são:
  - Inserir um elemento
  - Excluir um elemento
  - **Buscar um elemento**
  - Encontrar o maior
  - Encontrar o menor
  - Buscar o elemento sucessor e o predecessor
  - Percorrimento
    - Em pré-ordem
    - Em ordem
    - Em pós-ordem

Buscar um elemento

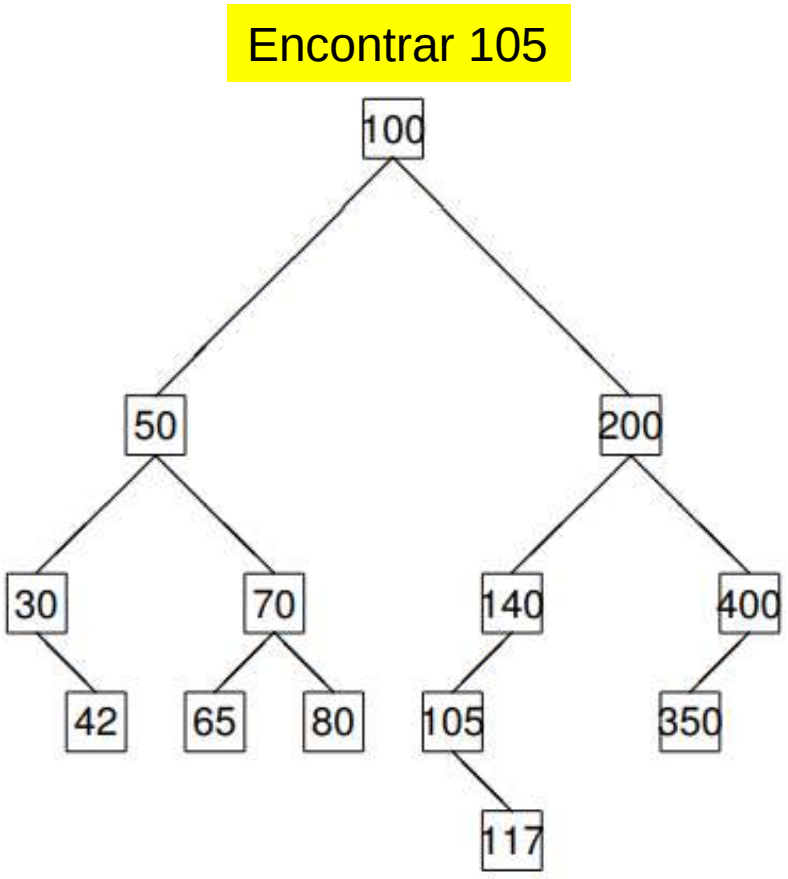
- A busca começa sempre na raiz
- Faz-se a comparação entre o elemento buscado e a chave
  - Se for igual:
    - Encontrou. Termina
  - Se o elemento é maior que a chave
    - Subárvore à direita
  - Se o elemento é menor que a chave
    - Subárvore à esquerda
- O processo para ao encontrar o elemento, ou encontrar NULL



FONTE : https://www.ime.usp.br/~song/mac5710/slides/06bst.pdf

Buscar um elemento

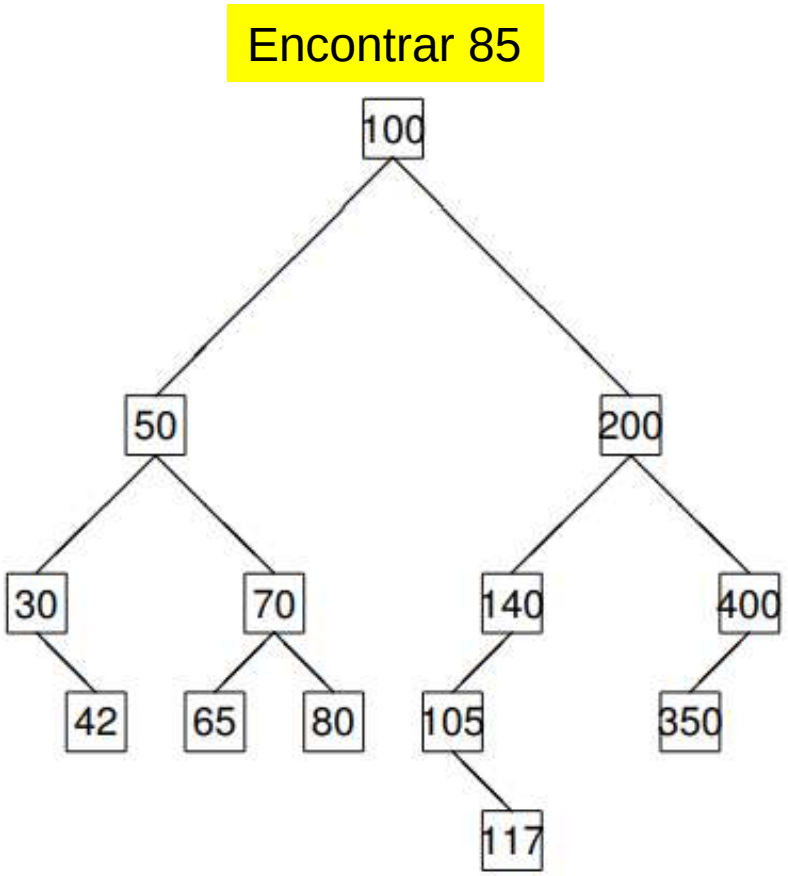
- A busca começa sempre na raiz
- Faz-se a comparação entre o elemento buscado e a chave
  - Se for igual:
    - Encontrou. Termina
  - Se o elemento é maior que a chave
    - Subárvore à direita
  - Se o elemento é menor que a chave
    - Subárvore à esquerda
- O processo para ao encontrar o elemento, ou encontrar NULL



FONTE : https://www.ime.usp.br/~song/mac5710/slides/06bst.pdf

Buscar um elemento

- A busca começa sempre na raiz
- Faz-se a comparação entre o elemento buscado e a chave
  - Se for igual:
    - Encontrou. Termina
  - Se o elemento é maior que a chave
    - Subárvore à direita
  - Se o elemento é menor que a chave
    - Subárvore à esquerda
- O processo para ao encontrar o elemento, ou encontrar NULL



FONTE : https://www.ime.usp.br/~song/mac5710/slides/06bst.pdf

TAD ABP

- As principais operações sobre as árvores binárias são:
  - Inserir um elemento
  - Excluir um elemento
  - Buscar um elemento
  - Encontrar o maior
  - Encontrar o menor
  - Buscar o elemento sucessor e o predecessor
  - **Percorrimento**
    - Em pré-ordem
    - Em ordem
    - Em pós-ordem

Percorrimento em ABP

- Uma árvore é uma estrutura não sequencial.
- Sendo assim, não existe ordem natural para percorrer árvores e portanto podemos escolher diferentes maneiras de percorrê-las.
- Nós iremos estudar três métodos para percorrer árvores.
  - Pré-ordem
  - Em ordem
  - Pós-ordem

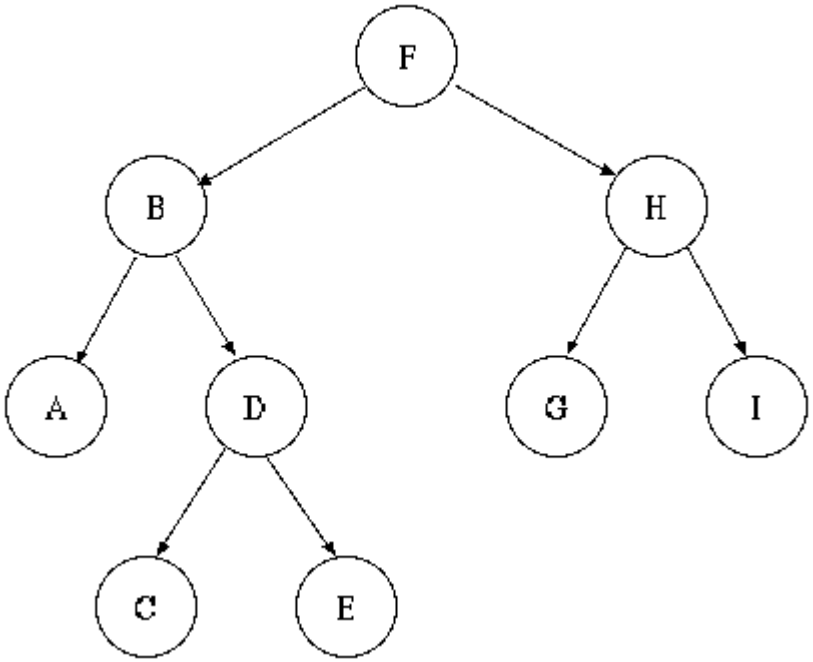
Percorrimento em ABP

- Pré-Ordem
  - Raiz
  - Esquerda
  - Direita
- Em ordem
  - Esquerda
  - Raiz
  - Direita
- Pós-Ordem
  - Direita
  - Esquerda
  - Raiz

Todos estes três métodos podem ser definidos recursivamente

Percorrimento em ABP

- Pré-Ordem
  - Raiz
  - Esquerda
  - Direita
- Em ordem
  - Esquerda
  - Raiz
  - Direita
- Pós-Ordem
  - Direita
  - Esquerda
  - Raiz



TAD ABP

- As principais operações sobre as árvores binárias são:
  - **Inserir um elemento**
  - Excluir um elemento
  - Buscar um elemento
  - Encontrar o maior
  - Encontrar o menor
  - Buscar o elemento sucessor e o predecessor
  - Percorrimento
    - Em pré-ordem
    - Em ordem
    - Em pós-ordem

Inserção

- A inserção começa na raiz.
  - Se a raiz é nula, o novo nó passa a ser a raiz da árvore.
  - Senão, encontra-se a **subárvore vazia** apropriada para inserção do novo nó
  - **Todo novo nó é uma folha!**

- **Exemplo:**
  - Desenhe a árvore binária de pesquisa que resulta a inserção sucessiva das chaves {90, 10, 170, 50, 80, 130, 100, 20, 30, 40, 70, 60 } em uma árvore inicialmente vazia.

Exercício

- **Exercício:**
  - Desenhe a árvore binária de pesquisa que resulta a inserção sucessiva das chaves {Q U E S T A O F A C I L} em uma árvore inicialmente vazia.
  - Mostre os percorrimentos em pré-ordem, ordem e pós-ordem
- **LISTA 17**

Atualizações no TAD ABP que facilitam o balanceamento

Sentinela

Pai

Árvore de pesquisa binária

- **Sentinela** é uma célula cabeça para simplificar as operações sobre algumas estruturas de dados, como lista, fila, pilha e árvore binária

- A sentinela guarda o endereço da estrutura durante toda a execução do programa.
  - Ou seja, o endereço da árvore não muda nunca!

- Um endereço único da estrutura tende a acabar com os casos especiais:
  - Exemplo : lista ordenada vazia -> insere o nó 20 -> insere o nó 15
  - No caso das árvores balanceadas, a raiz da árvore muda com as operações de balanceamento. A presença da sentinela faz essas alterações ter pouco impacto na estrutrua.

Árvore de pesquisa binária

- Guardar o **pai** do nó também é uma estratégia para simplificar as operações em árvore binária.

- Na estrutura árvore, o ponteiro para o pai tem a mesma conotação do ponteiro para o nó anterior na lista duplamente encadeada.
  - Ou seja, permite ‘voltar’ na estrutura sem ter que ir ao início da mesma

Estrutura

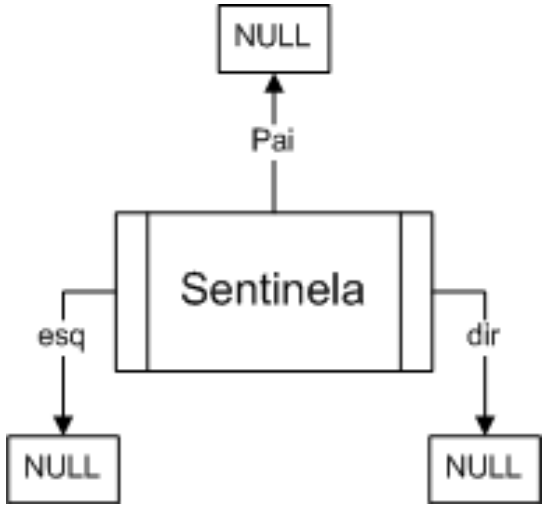
- O nó da árvore binária passa a ter mais um ponteiro para o pai.

```
struct noArvore
{
    int chave;
    struct noArvore *esq;
    struct noArvore *dir;
    struct noArvore *pai;
}
```

Árvore de pesquisa binária

- Operações
  - **Cria árvore**
    - Cria o sentinela e devolve seu endereço, que será único durante toda execução do programa.

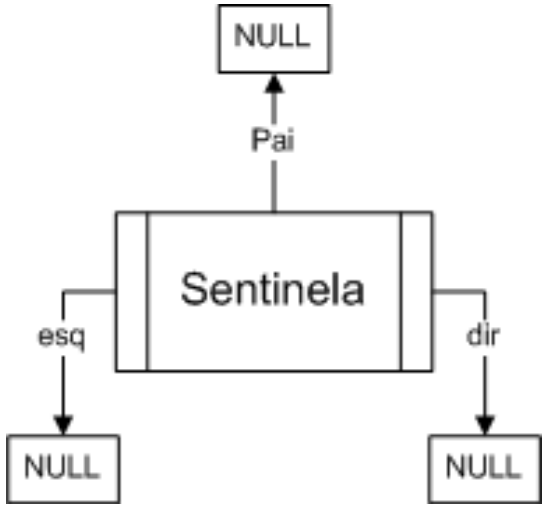
```
arvore * criaArvore();
```



Árvore de pesquisa binária

- Operações
  - **Cria árvore**
    - Cria o sentinela e devolve seu endereço, que será único durante toda execução do programa.

```
arvore * criaArvore();
```

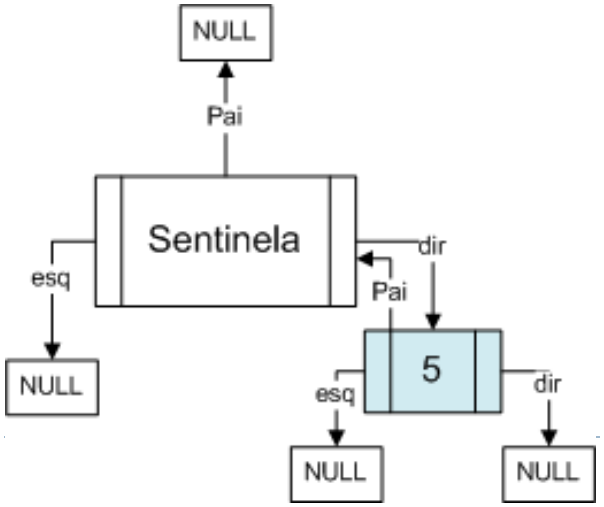


Árvore de pesquisa binária

- Operações
  - **Insere nó na árvore**
    - Insere um nó na árvore binária.
      - Se a árvore estiver vazia, cria a raiz
      - Se a árvore tiver elementos, adiciona seguindo a regra de menores a esquerda, maiores a direita.

```
void insereNo (arvore *A, int num); //e demais dados satélite
```

5 é raiz da árvore

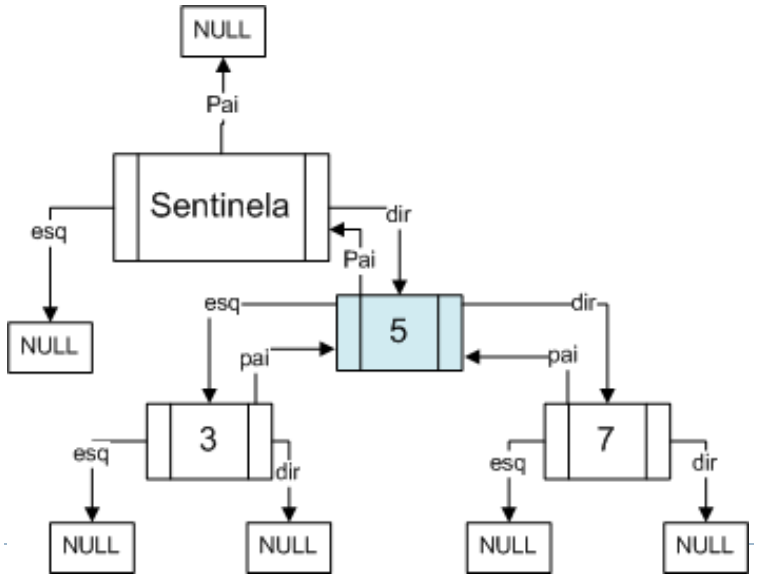


Árvore de pesquisa binária

- Operações
  - **Insere nó na árvore**
    - Insere um nó na árvore binária.
      - Se a árvore estiver vazia, cria a raiz
      - Se a árvore tiver elementos, adiciona seguindo a regra de menores a esquerda, maiores a direita.

```
void insereNo (arvore *A, int num);
```

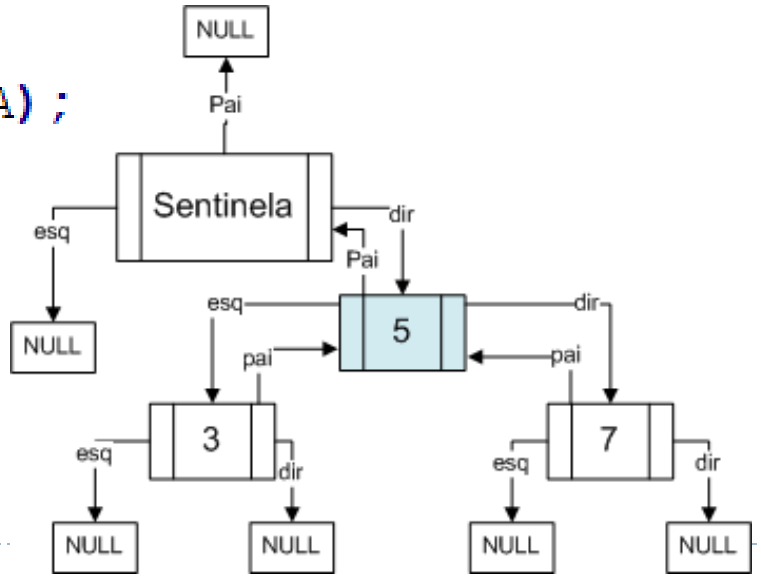
Inserir 7 e 3 a árvore



Árvore de pesquisa binária

- Operações
  - **Percorre os nós da ávore**
    - Pré-ordem: raiz, esq, dir – 5,3,7
    - Ordem : esq, raiz, dir – 3,5,7
    - Pós-ordem: esq, dir, raiz – 3,7,5

```
void percorreOrdem (arvore *A);
```



TAD ABP

- As principais operações sobre as árvores binárias são:
  - Cria árvore
  - Inserir um elemento
  - Excluir um elemento
  - Buscar um elemento
  - Percorrimento
    - Em pré-ordem
    - Em ordem
    - Em pós-ordem