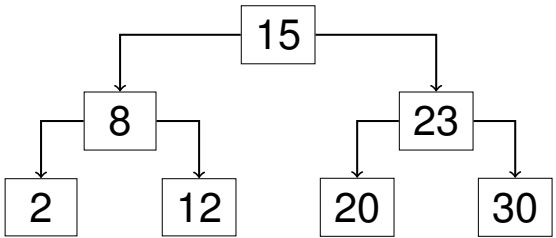
AULA 16
ESTRUTURA DE DADOS

Árvores Binárias de Pesquisa

Norton T. Roman & Luciano A. Digiampietri

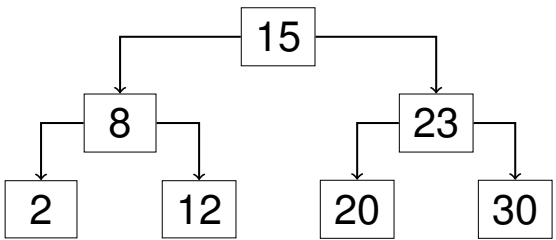
Árvores Binárias de Pesquisa

Voltemos à nossa árvore binária.



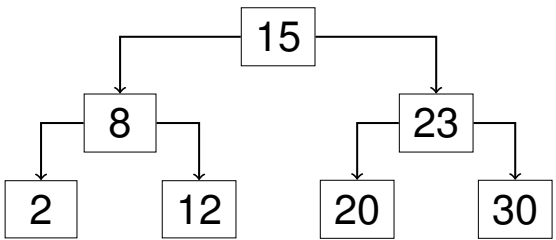
Árvores Binárias de Pesquisa

Voltemos à nossa árvore binária.
Na verdade, ela é mais que uma árvore binária.



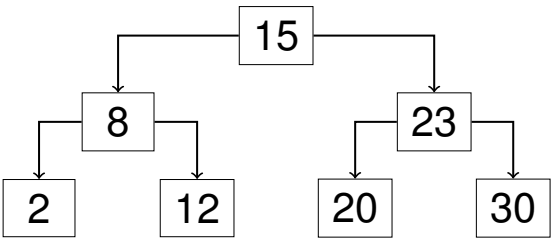
Árvores Binárias de Pesquisa

Voltemos à nossa árvore binária.
Na verdade, ela é mais que uma árvore binária.
Trata-se de uma árvore binária de pesquisa.



Árvores Binárias de Pesquisa

Uma árvore binária de pesquisa é uma árvore binária em que, a cada nó, todos os registros com chaves menores que a deste nó estão na subárvore da esquerda, enquanto que os registros com chaves maiores estão na subárvore da direita



Árvores Binárias de Pesquisa

Para nossa árvore binária de pesquisa, veremos as seguintes funções:

Inicialização da árvore

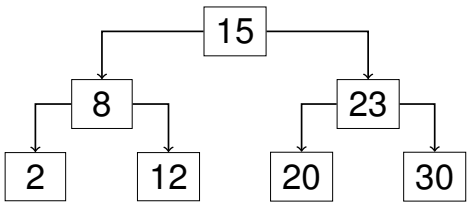
Inserção de um elemento

Busca de um elemento

Contagem do número de elementos

Leitura da árvore

Remoção de um elemento



Árvores Binárias de Pesquisa

Para nossa árvore binária de pesquisa, veremos as seguintes funções:

Inicialização da árvore

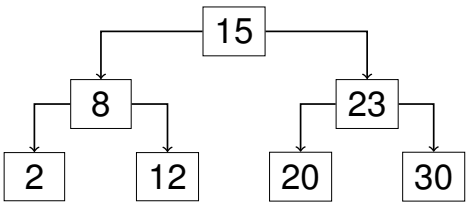
Inserção de um elemento

Busca de um elemento

Contagem do número de elementos

Leitura da árvore

Remoção de um elemento



ABP – Inicialização da Árvore

Considere o código que define um nó na árvore:

```
#include <stdio.h>
#include <stdlib.h>
#define true 1
#define false 0

typedef int bool;
typedef int TIPOCHAVE;
```

```
typedef struct aux {
  TIPOCHAVE chave;
  /* Dados armazenados vão aqui */
  struct aux *esq, *dir;
} NO;

typedef NO* PONT;
```

ABP – Inicialização da Árvore

Para representarmos nossa árvore, precisaremos tão somente do endereço do nó raiz.

E para inicializarmos a árvore basta tornarmos esse endereço `null`

```
PONT inicializa() {
  return(NULL);
}

int main() {
  PONT r = inicializa();
}
```

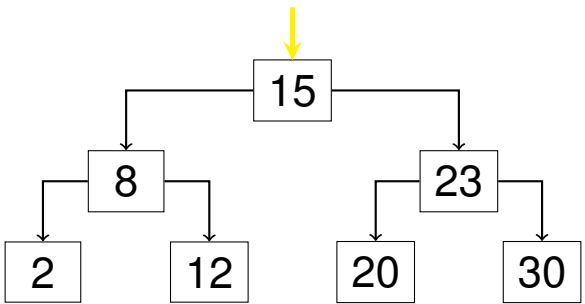
ABP – Inserção

Vamos supor que não permitimos duplicação de chaves.
  Caso permitamos, basta definir uma política.
  Por exemplo: chaves $\leq$ à de um determinado nó ficam na subárvore à esquerda deste.

Como então inserimos um elemento (um nó) na árvore?

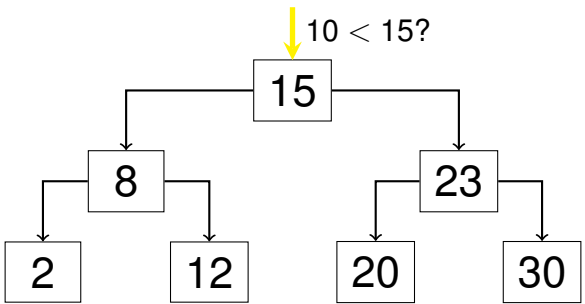
ABP – Inserção (ideia)

Ex: Inserir 10.



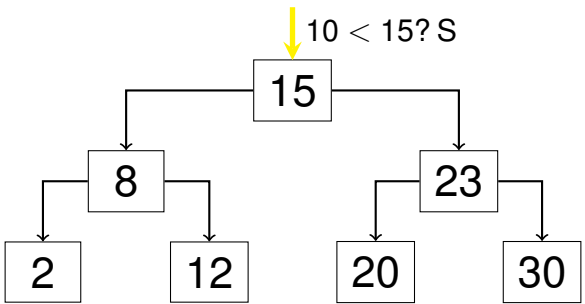
ABP – Inserção (ideia)

Ex: Inserir 10.



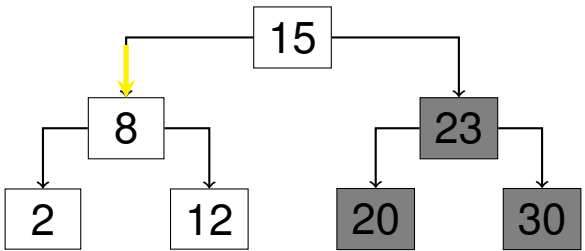
ABP – Inserção (ideia)

Ex: Inserir 10.



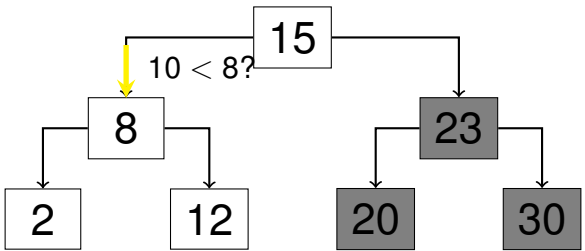
ABP – Inserção (ideia)

Ex: Inserir 10.



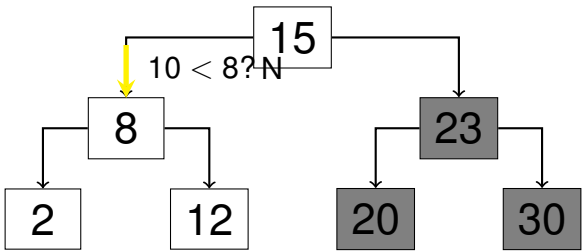
ABP – Inserção (ideia)

Ex: Inserir 10.



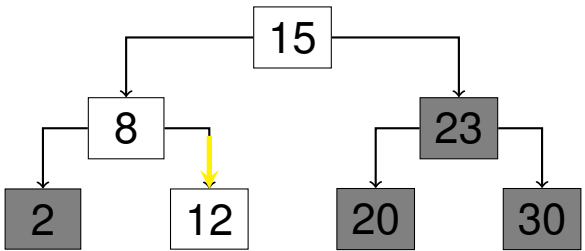
ABP – Inserção (ideia)

Ex: Inserir 10.



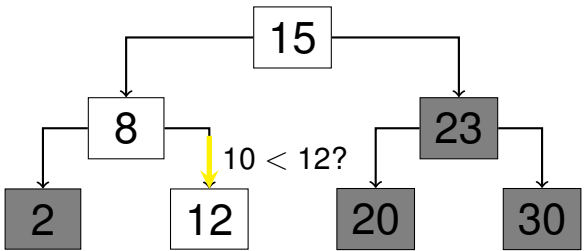
ABP – Inserção (ideia)

Ex: Inserir 10.



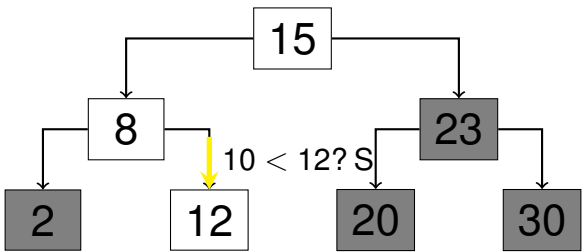
ABP – Inserção (ideia)

Ex: Inserir 10.



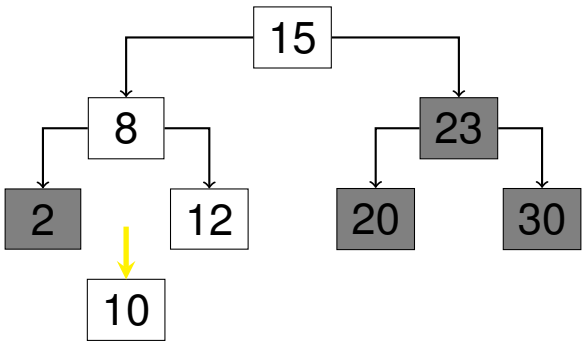
ABP – Inserção (ideia)

Ex: Inserir 10.



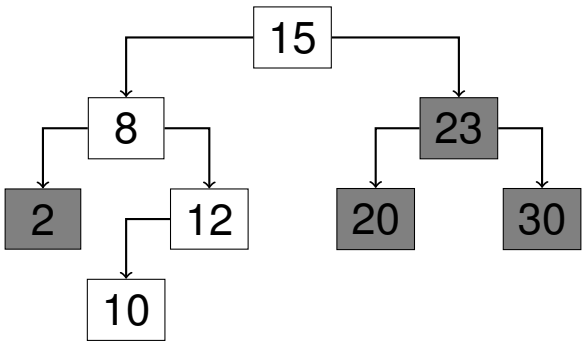
ABP – Inserção (ideia)

Ex: Inserir 10.



ABP – Inserção (ideia)

Ex: Inserir 10.



ABP – Algoritmo (Inserção)

Se a raiz for `null`, inserimos lá

Senão:

- Se a chave do elemento a ser inserido for menor que a da raiz:
  - Insere na subárvore da esquerda
- Senão:
  - Insere na subárvore da direita

ABP – Inserção

Note que essa versão é recursiva

Se a raiz for null, inserimos lá

Senão:

- Se a chave do elemento a ser inserido for menor que a da raiz:
  - Insere na subárvore da esquerda
- Senão:
  - Insere na subárvore da direita

```
PONT adiciona(PONT raiz, PONT no){
  if (raiz == NULL) return(no);
  if (no->chave < raiz->chave)
    raiz->esq =
      adiciona(raiz->esq, no);
  else
    raiz->dir =
      adiciona(raiz->dir, no);
  /*ignoro chave igual*/
  return(raiz);
}
```

ABP – Inserção

Note que essa versão é recursiva

Se a raiz for null, inserimos lá

Senão:

- Se a chave do elemento a ser inserido for menor que a da raiz:
  - Insere na subárvore da esquerda
- Senão:
  - Insere na subárvore da direita

```
PONT adiciona(PONT raiz, PONT no){
  if (raiz == NULL) return(no);
  if (no->chave < raiz->chave)
    raiz->esq =
      adiciona(raiz->esq, no);
  else
    raiz->dir =
      adiciona(raiz->dir, no);
  /*ignoro chave igual*/
  return(raiz);
}
```

ABP – Inserção

Note que essa versão é recursiva

Se a raiz for null, inserimos lá

Senão:

Se a chave do elemento a ser inserido for menor que a da raiz:

Insere na subárvore da esquerda

Senão:

Insere na subárvore da direita

```
PONT adiciona(PONT raiz, PONT no){
  if (raiz == NULL) return(no);
  if (no->chave < raiz->chave)
    raiz->esq =
      adiciona(raiz->esq, no);
  else
    raiz->dir =
      adiciona(raiz->dir, no);
  /*ignoro chave igual*/
  return(raiz);
}
```

ABP – Inserção

Note que essa versão é recursiva

Se a raiz for null, inserimos lá

Senão:

- Se a chave do elemento a ser inserido for menor que a da raiz:
  - Insere na subárvore da esquerda
- Senão:
  - Insere na subárvore da direita

```
PONT adiciona(PONT raiz, PONT no){
  if (raiz == NULL) return(no);
  if (no->chave < raiz->chave)
    raiz->esq =
      adiciona(raiz->esq, no);
  else
    raiz->dir =
      adiciona(raiz->dir, no);
  /*ignoro chave igual*/
  return(raiz);
}
```

ABP – Inserção

Note que essa versão é recursiva

Se a raiz for null, inserimos lá

Senão:

- Se a chave do elemento a ser inserido for menor que a da raiz:
  - Insere na subárvore da esquerda
- Senão:
  - Insere na subárvore da direita

```
PONT adiciona(PONT raiz, PONT no){
  if (raiz == NULL) return(no);
  if (no->chave < raiz->chave)
    raiz->esq =
      adiciona(raiz->esq, no);
  else
    raiz->dir =
      adiciona(raiz->dir, no);
  /*ignoro chave igual*/
  return(raiz);
}
```

ABP – Inserção

E como usamos isso no código?

```
PONT adiciona(PONT raiz, PONT no){
  ...
}

int main() {
  PONT r = inicializa();


  r = adiciona(r,no);
}
```

ABP – Inserção

E como usamos isso no código?

Note que o nó já está alocado em memória

```
PONT adiciona(PONT raiz, PONT no){
  ...
}

int main() {
  PONT r = inicializa();


  r = adiciona(r,no);
}
```

ABP – Inserção

E como usamos isso no código?

Note que o nó já está alocado em memória

- Precisamos antes de uma função para alocar o nó

```
PONT adiciona(PONT raiz, PONT no){
  ...
}

int main() {
  PONT r = inicializa();

  PONT no = criaNovoNo(23);
  r = adiciona(r,no);
}
```

ABP – Inserção

```
PONT criaNovoNo(TIPOCHAVE ch) {
  PONT novoNo = (PONT)malloc(sizeof(NO));
  novoNo->esq = NULL;
  novoNo->dir = NULL;
  novoNo->chave = ch;
  return(novoNo);
}
```

ABP – Inserção

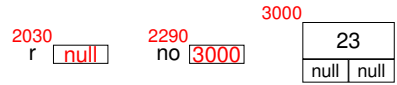
Ex: Adicionando 23 e 12

```
PONT adiciona(PONT raiz, PONT no){
  if (raiz == NULL) return(no);
  if (no->chave < raiz->chave)
    raiz->esq = adiciona(raiz->esq, no);
  else
    raiz->dir = adiciona(raiz->dir, no);
  return(raiz);
}

int main() {
  PONT r = inicializa();
  PONT no = criaNovoNo(23);
  r = adiciona(r,no);
  ...
}
```

ABP – Inserção

Ex: Adicionando 23 e 12

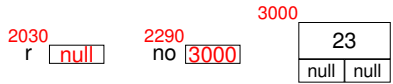
```
PONT adiciona(PONT raiz, PONT no){
  if (raiz == NULL) return(no);
  if (no->chave < raiz->chave)
    raiz->esq = adiciona(raiz->esq, no);
  else
    raiz->dir = adiciona(raiz->dir, no);
  return(raiz);
}

int main() {
  PONT r = inicializa();
  PONT no = criaNovoNo(23);
  r = adiciona(r,no);
  ...
}
```



ABP – Inserção

Ex: Adicionando 23 e 12

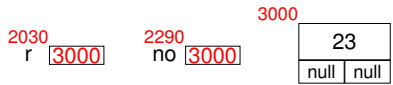
```
PONT adiciona(PONT raiz, PONT no){
  if (raiz == NULL) return(no);
  if (no->chave < raiz->chave)
    raiz->esq = adiciona(raiz->esq, no);
  else
    raiz->dir = adiciona(raiz->dir, no);
  return(raiz);
}

int main() {
  PONT r = inicializa();
  PONT no = criaNovoNo(23);
  r = adiciona(r,no);
  ...
}
```

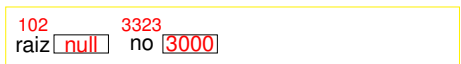


ABP – Inserção

Ex: Adicionando 23 e 12

```
PONT adiciona(PONT raiz, PONT no){
  if (raiz == NULL) return(no);
  if (no->chave < raiz->chave)
    raiz->esq = adiciona(raiz->esq, no);
  else
    raiz->dir = adiciona(raiz->dir, no);
  return(raiz);
}

int main() {
  PONT r = inicializa();
  PONT no = criaNovoNo(23);
  r = adiciona(r,no);
  ...
}
```

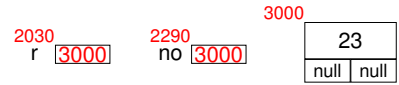




ABP – Inserção

Ex: Adicionando 23 e 12

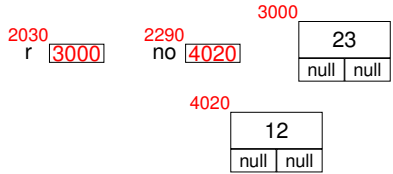
```
PONT adiciona(PONT raiz, PONT no){
  if (raiz == NULL) return(no);
  if (no->chave < raiz->chave)
    raiz->esq = adiciona(raiz->esq, no);
  else
    raiz->dir = adiciona(raiz->dir, no);
  return(raiz);
}

int main() {
  ...
  no = criaNovoNo(12);
  r = adiciona(r,no);
  ...
}
```



ABP – Inserção

Ex: Adicionando 23 e 12

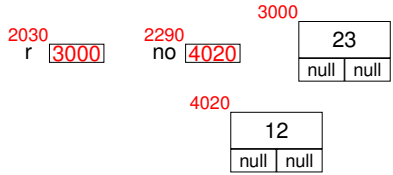
```
PONT adiciona(PONT raiz, PONT no){
  if (raiz == NULL) return(no);
  if (no->chave < raiz->chave)
    raiz->esq = adiciona(raiz->esq, no);
  else
    raiz->dir = adiciona(raiz->dir, no);
  return(raiz);
}

int main() {
  ...
  no = criaNovoNo(12);
  r = adiciona(r,no);
  ...
}
```



ABP – Inserção

Ex: Adicionando 23 e 12

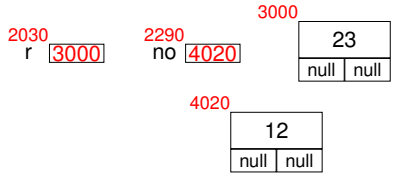
```
PONT adiciona(PONT raiz, PONT no){
  if (raiz == NULL) return(no);
  if (no->chave < raiz->chave)
    raiz->esq = adiciona(raiz->esq, no);
  else
    raiz->dir = adiciona(raiz->dir, no);
  return(raiz);
}

int main() {
  ...
  no = criaNovoNo(12);
  r = adiciona(r,no);
  ...
}
```



ABP – Inserção

Ex: Adicionando 23 e 12

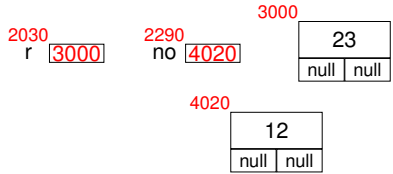
```
PONT adiciona(PONT raiz, PONT no){
  if (raiz == NULL) return(no);
  if (no->chave < raiz->chave)
    raiz->esq = adiciona(raiz->esq, no);
  else
    raiz->dir = adiciona(raiz->dir, no);
  return(raiz);
}

int main() {
  ...
  no = criaNovoNo(12);
  r = adiciona(r,no);
  ...
}
```



ABP – Inserção

Ex: Adicionando 23 e 12

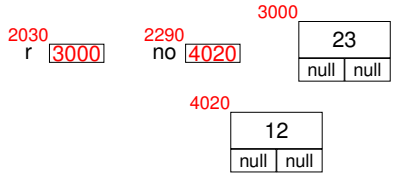
```
PONT adiciona(PONT raiz, PONT no){
  if (raiz == NULL) return(no);
  if (no->chave < raiz->chave)
    raiz->esq = adiciona(raiz->esq, no);
  else
    raiz->dir = adiciona(raiz->dir, no);
  return(raiz);
}

int main() {
  ...
  no = criaNovoNo(12);
  r = adiciona(r,no);
  ...
}
```

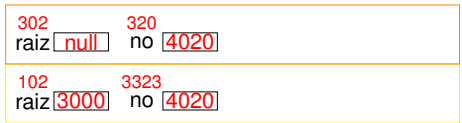


ABP – Inserção

Ex: Adicionando 23 e 12

```
PONT adiciona(PONT raiz, PONT no){
  if (raiz == NULL) return(no);
  if (no->chave < raiz->chave)
    raiz->esq = adiciona(raiz->esq, no);
  else
    raiz->dir = adiciona(raiz->dir, no);
  return(raiz);
}

int main() {
  ...
  no = criaNovoNo(12);
  r = adiciona(r,no);
  ...
}
```

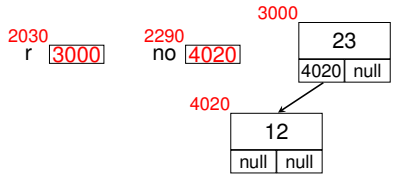


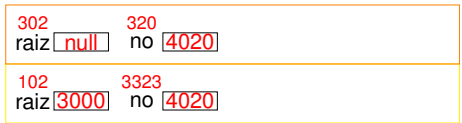


ABP – Inserção

Ex: Adicionando 23 e 12

```
PONT adiciona(PONT raiz, PONT no){
  if (raiz == NULL) return(no);
  if (no->chave < raiz->chave)
    raiz->esq = adiciona(raiz->esq, no);
  else
    raiz->dir = adiciona(raiz->dir, no);
  return(raiz);
}

int main() {
  ...
  no = criaNovoNo(12);
  r = adiciona(r,no);
  ...
}
```

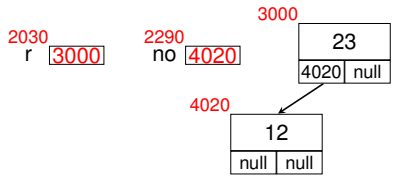




ABP – Inserção

Ex: Adicionando 23 e 12

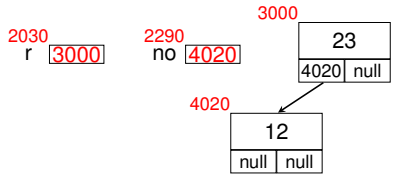
```
PONT adiciona(PONT raiz, PONT no){
  if (raiz == NULL) return(no);
  if (no->chave < raiz->chave)
    raiz->esq = adiciona(raiz->esq, no);
  else
    raiz->dir = adiciona(raiz->dir, no);
  return(raiz);
}

int main() {
  ...
  no = criaNovoNo(12);
  r = adiciona(r,no);
  ...
}
```



ABP – Inserção

Ex: Adicionando 23 e 12

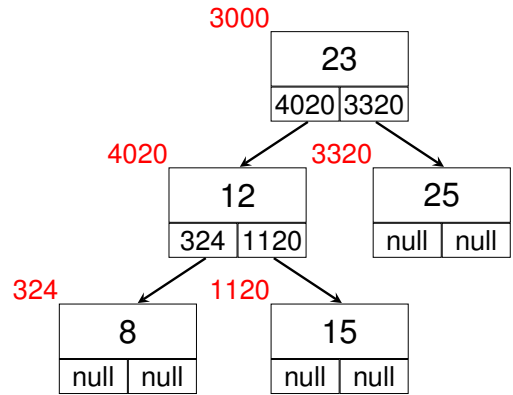
```
PONT adiciona(PONT raiz, PONT no){
  if (raiz == NULL) return(no);
  if (no->chave < raiz->chave)
    raiz->esq = adiciona(raiz->esq, no);
  else
    raiz->dir = adiciona(raiz->dir, no);
  return(raiz);
}

int main() {
  ...
  no = criaNovoNo(12);
  r = adiciona(r,no);
  ...
}
```



ABP – Inserção

```
PONT adiciona(PONT raiz, PONT no){
  if (raiz == NULL) return(no);
  if (no->chave < raiz->chave)
    raiz->esq = adiciona(raiz->esq, no);
  else
    raiz->dir = adiciona(raiz->dir, no);
  return(raiz);
}
```

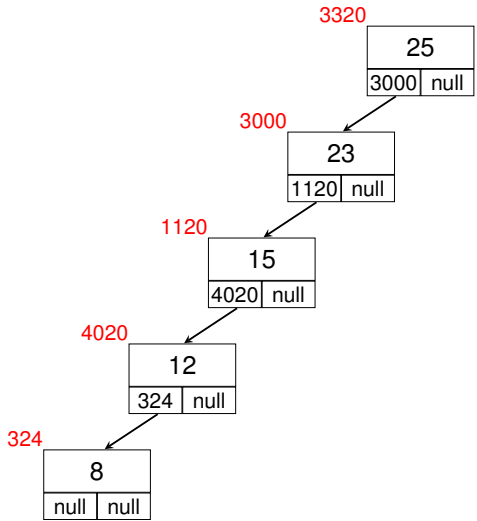
Adicionando, por exemplo, 15, 25, 8, nessa ordem, teremos a árvore:



ABP – Inserção

A ordem de inserção determina a forma da árvore.

Ex: 25, 23, 15, 12, 8, nessa ordem, geram a árvore:



AULA 16
ESTRUTURA DE DADOS

Árvores Binárias de Pesquisa

Norton T. Roman & Luciano A. Digiampietri