CENTRO DE CIÊNCIA E TECNOLOGIA
LABORATÓRIO DE CIÊNCIAS MATEMÁTICAS
UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE

Árvores de Busca Binária

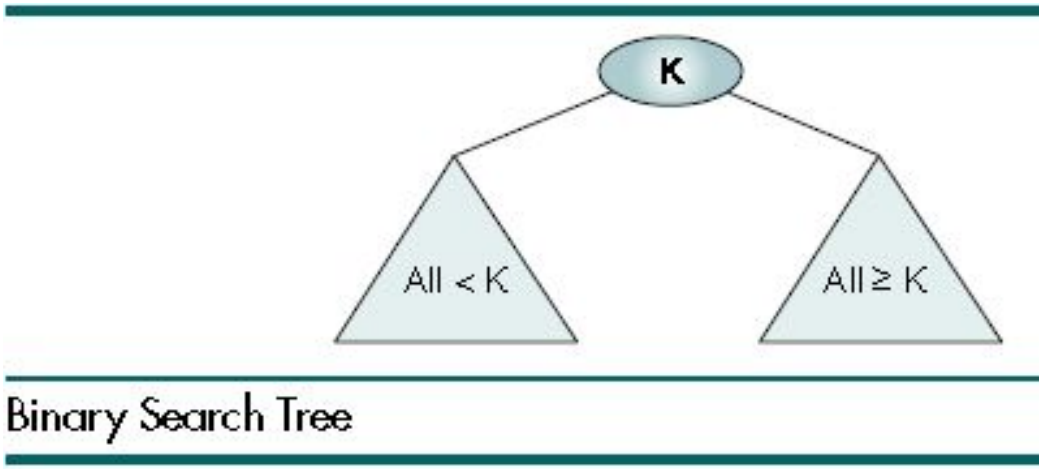
Disciplina: Estrutura de Dados I

Prof. Fermín Alfredo Tang Montané

Curso: Ciência da Computação

Árvore de Busca Binária
Definição

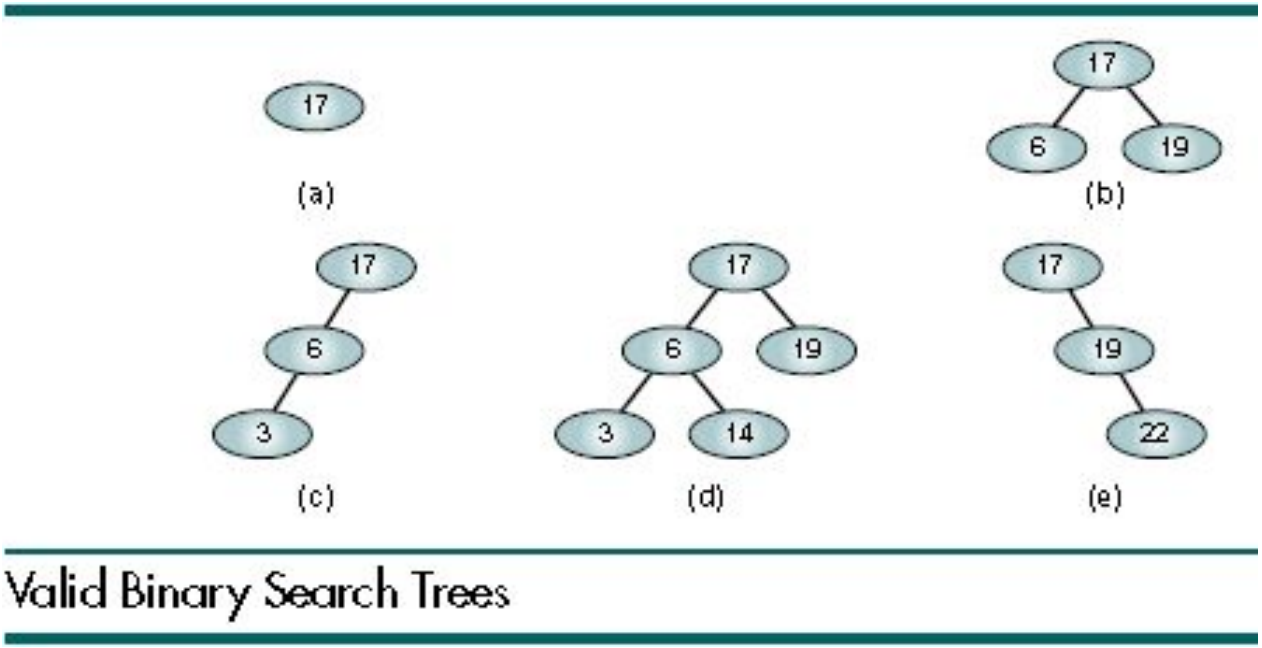
- Uma Árvore de Busca Binária (*Binary Search Tree BST*) é uma árvore binária que possui as seguintes propriedades:
  - Todos os elementos na subárvore esquerda são menores que a raiz;
  - Todos os elementos na subárvore direita são maiores ou iguais que a raiz;
  - Cada subárvore é uma árvore de busca binária.



- Em geral, a informação representada em cada nó de uma árvore de busca binária é um registro ao invés de um único elemento. Neste caso, a definição da árvore que determina o sequenciamento dos nós se aplica ao campo chave do registro.

Árvore de Busca Binária
Exemplos

- A figura mostra 5 exemplos de árvores de busca binárias.

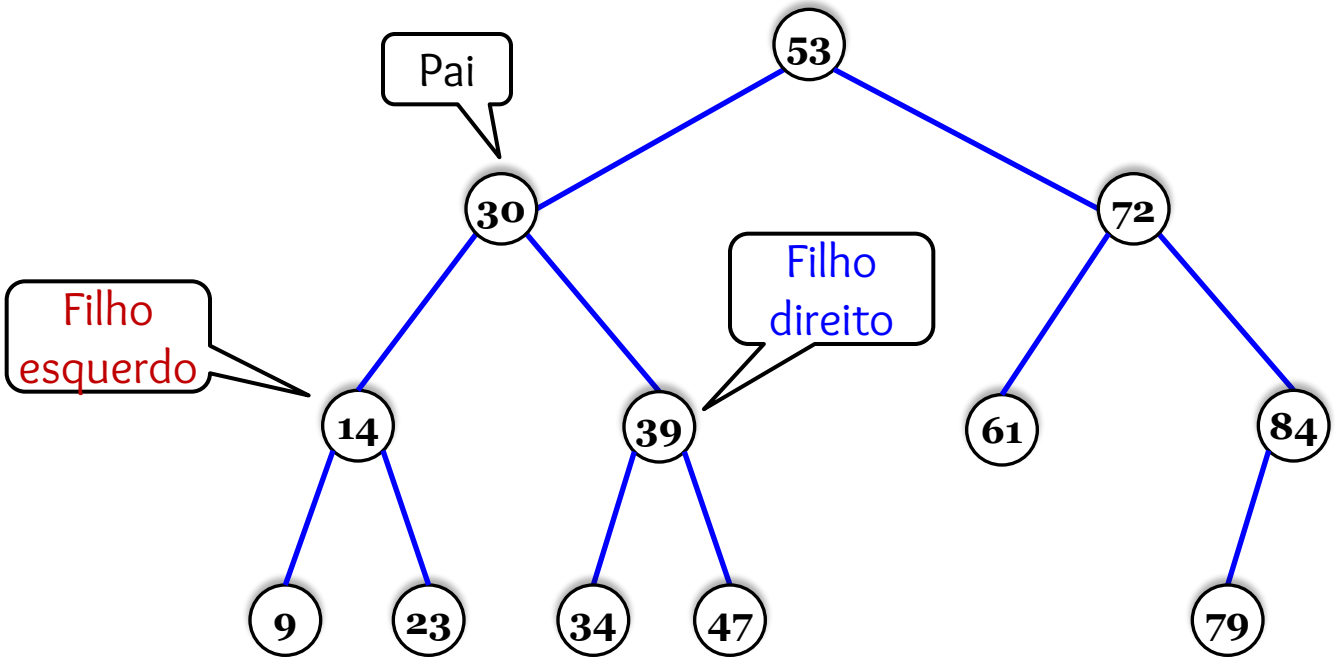


Valid Binary Search Trees

- Observe que as árvores (a) e (b) são árvores completas e balanceadas.
- A árvore (d) é quase completa e balanceada;
- Já as árvores (c) e (e) não são nem completas nem estão balanceadas.

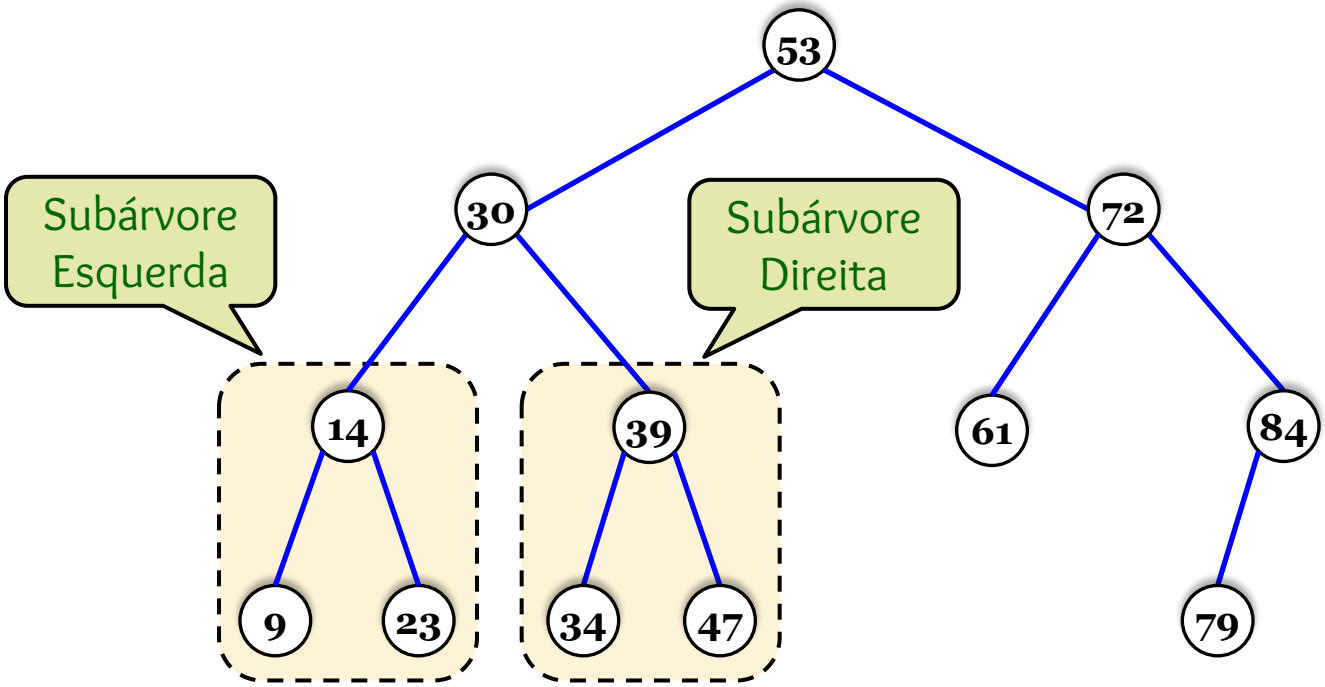
Árvore de Busca Binária
Exemplos

- Outro exemplo de árvore de busca binária.



Árvore de Busca Binária
Exemplos

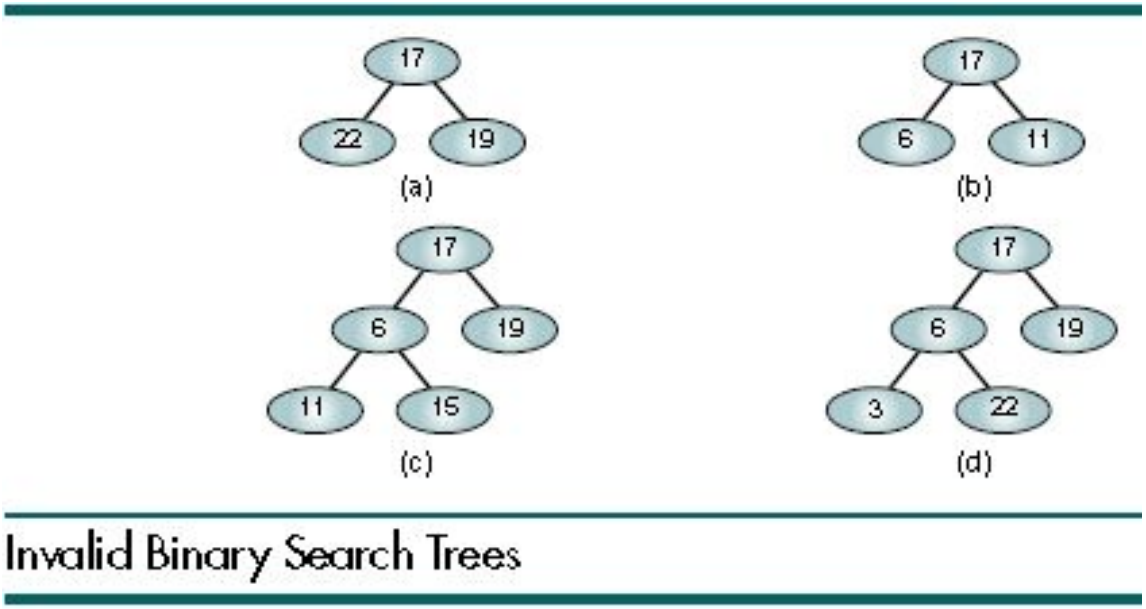
- Árvore de Busca Binária.- Como consequência da definição, temos:
- i) A chave de qualquer nó na sub-árvore esquerda do nó n, é estritamente menor que a chave do nó n;
- ii) A chave de qualquer nó na sub-árvore direita do nó n, é maior ou igual que a chave do nó n.



Árvore de Busca Binária

Contra-exemplos

- A figura mostra 4 exemplos de árvores binárias que não atendem as propriedades de árvores de busca binária.



Invalid Binary Search Trees

- Alguma ou mais de uma propriedade é violada.

Árvore de Busca Binária

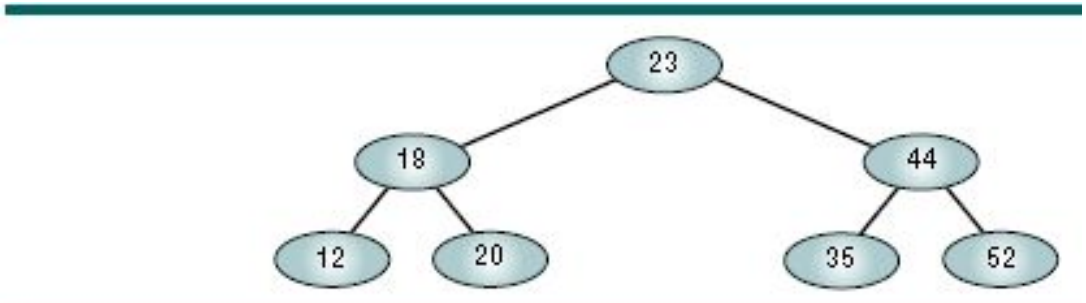
Operações

- Estudaremos várias operações usadas em árvores de busca binária.
- Entre elas temos:
  - Algoritmos de percurso;
  - Algoritmos de busca;
  - Algoritmos para construir uma árvore de busca binária.

Árvore de Busca Binária

Percursos

- As operações de percurso de arvores de busca binária são idênticas as operações de percurso de árvores binária estudadas previamente:
  - Pré-ordem, Em-ordem e Pós-ordem.
- No entanto, no caso de árvores de busca binária temos particular interesse em mostrar os resultados do percurso.
- Considere a seguinte árvore de busca binária:

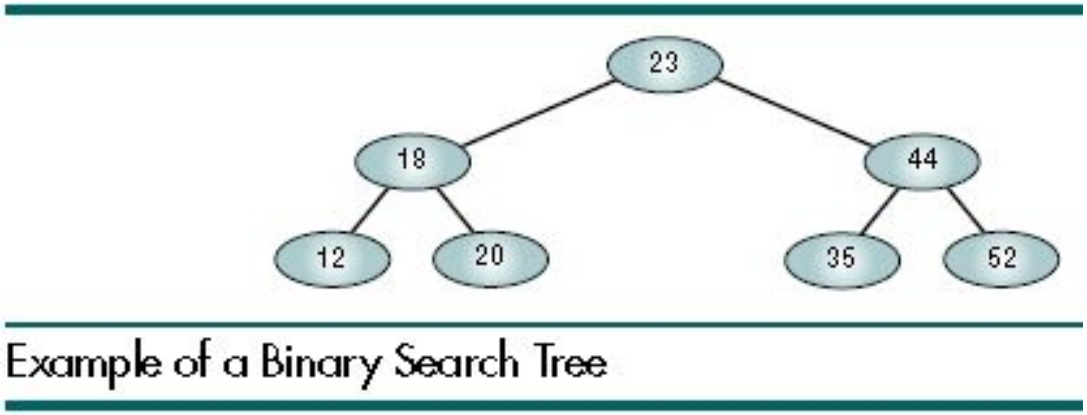


Example of a Binary Search Tree

- Realizemos os três tipos de percurso.

Árvore de Busca Binária

Percursos



Example of a Binary Search Tree

- O percurso Pré-ordem produz a seguinte sequência:

23	18	12	20	44	35	52

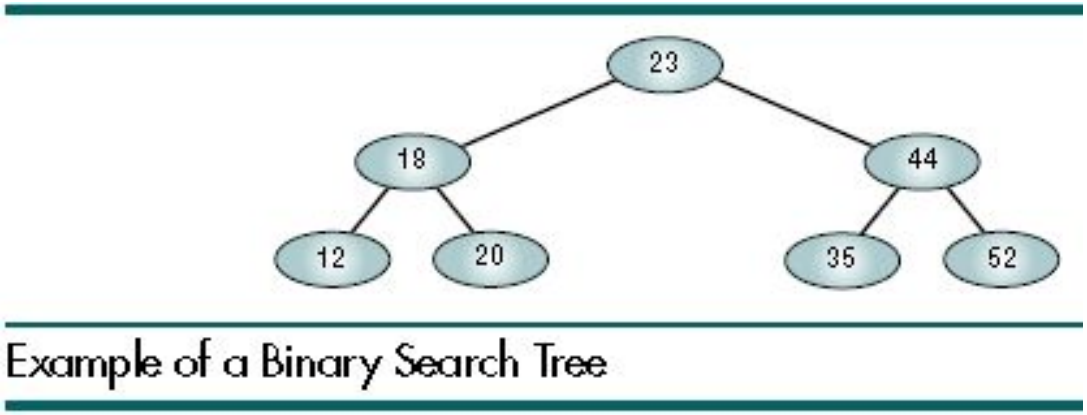
- O percurso Pós-ordem produz a seguinte sequência:

12	20	18	35	52	44	23

- Observe que em ambos casos, embora o percurso seja válido ele não tem muita utilidade.

Árvore de Busca Binária

Percursos



Example of a Binary Search Tree

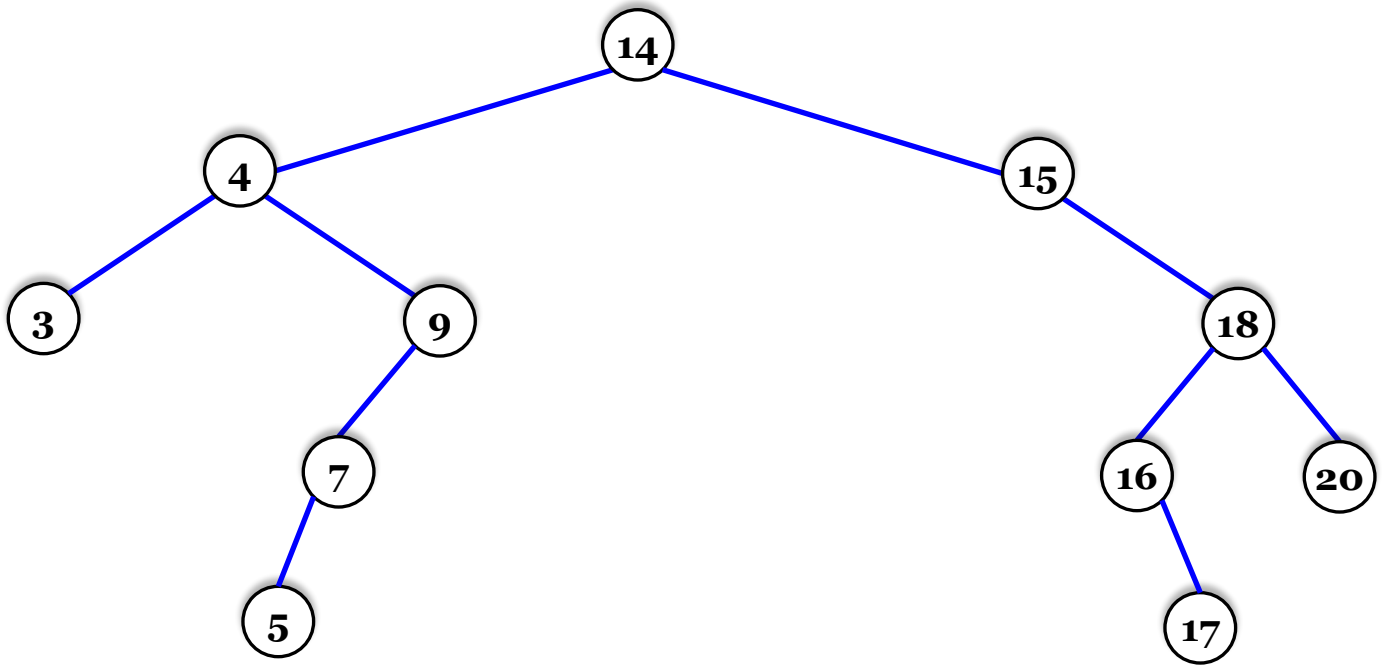
- Finalmente, considere o percurso Em-ordem:

12	18	20	23	35	44	52

- Observa-se que o percurso em-ordem de uma árvore de busca binária é muito útil. Este percurso produz uma **sequência ordenada**.
- A sequência está em **ordem crescente**.
- Lembre que o percurso em-ordem, processa primeiro a subárvore esquerda, depois a raiz e depois subárvore direita (LNR). Um percurso alternativo, seria inverter a ordem das subárvores esquerda e direita (RNL). Neste caso, o percurso produziria uma sequência em **ordem decrescente**.

Árvore de Busca Binária

Percursos



- Finalmente, considere o percurso Em-ordem:

1	2	3	4	5	6	7	8	9	10	11
3	4	5	7	9	14	15	16	17	18	20

Árvore de Busca Binária

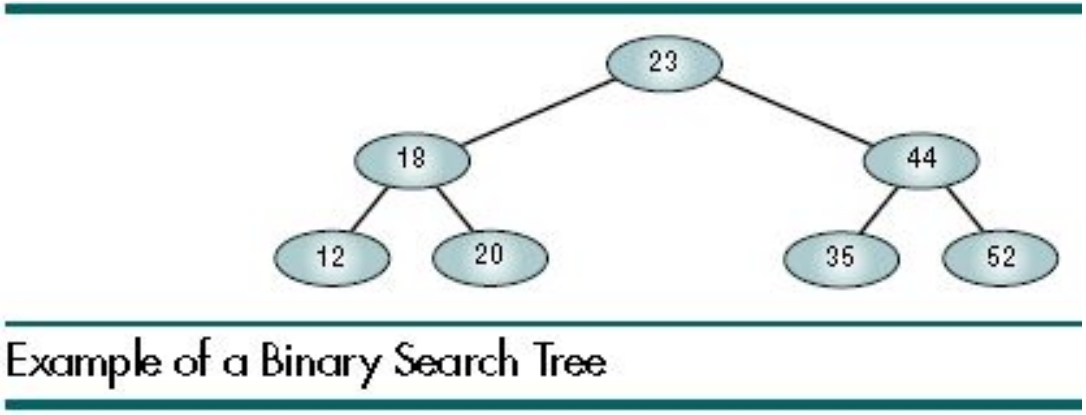
Buscas

- Nesta seção estudamos três algoritmos de busca:
  - Encontrar o menor elemento;
  - Encontrar o maior elemento;
  - Encontrar um elemento específico (Busca em BST)
- Lembre que o percurso em-ordem, processa primeiro a subárvore esquerda, depois a raiz e depois subárvore direita (LNR). Um percurso alternativo, seria inverter a ordem das subárvores esquerda e direita (RNL). Neste caso, o percurso produziria uma sequência em **ordem decrescente**.

Árvore de Busca Binária

Buscas – Menor Elemento

- Na árvore de exemplo, o menor elemento é aquele com valor 12, que corresponde à folha mais a esquerda da árvore.



Example of a Binary Search Tree

- A operação de encontrar o menor elemento então consiste em: começando na raiz, **seguir repetidamente o ramo esquerdo até chegar a um nó folha**.

Árvore de Busca Binária

Buscas – Menor Elemento

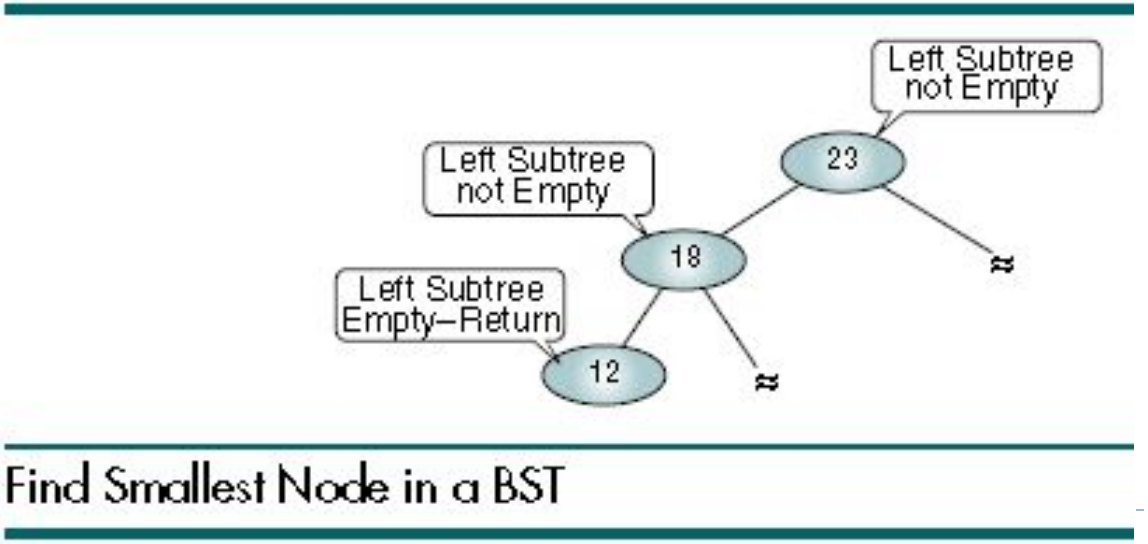
- A operação de encontrar o menor elemento então consiste em: começando na raiz, **seguir repetidamente o ramo esquerdo até chegar a um nó folha**. O algoritmo para encontrar o menor elemento é o seguinte:

```
Algorithm findSmallestBST (root)
This algorithm finds the smallest node in a BST.
   Pre    root is a pointer to a nonempty BST or subtree
   Return address of smallest node
1  if (left subtree empty)
   1   return (root)
2  end if
3  return findSmallestBST (left subtree)
end findSmallestBST
```

- Observe que trata-se de um algoritmo recursivo.

Árvore de Busca Binária
Buscas – Menor Elemento

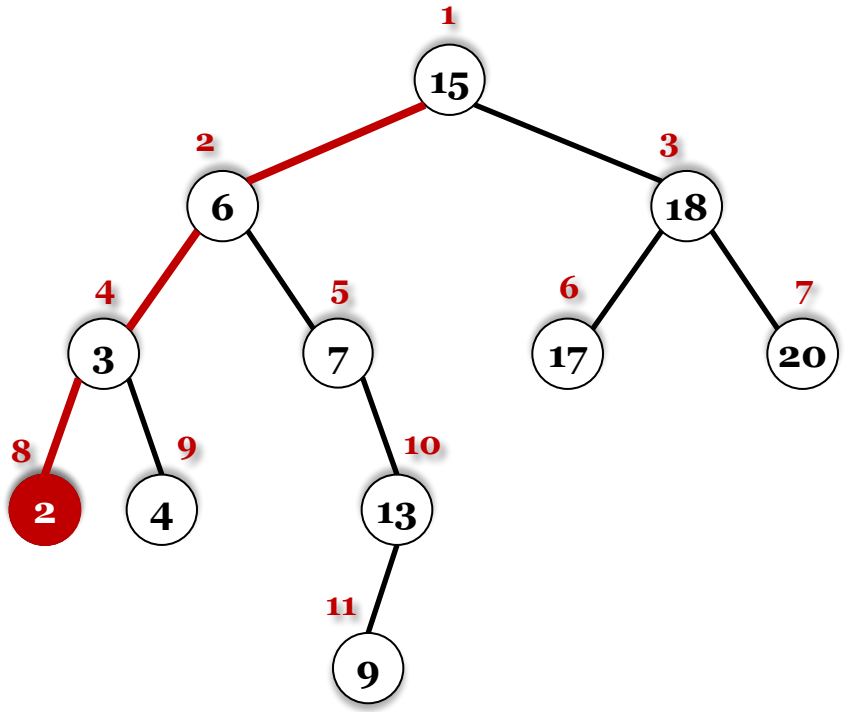
- Observe que trata-se de um algoritmo recursivo.
- A figura ilustra o processo de busca seguido pelo algoritmo.
- A primeira chamada começa com a raiz da árvore. Depois, segue-se um caminho descendente pelas subárvores à esquerda. Se a sub-árvore à esquerda não é nula, o processo continua fazendo uma chamada recursiva **findSmallestBST()** com a subárvore esquerda. O processo termina (caso base) quando uma subárvore esquerda está vazia. Inicia-se um processo de retorno das chamadas recursivas, repassando o menor elemento até voltar á primeira chamada (caminho ascendente).



Árvore de Busca Binária

Buscas – Menor Elemento - Exemplo

- Outro exemplo sobre a busca do menor elemento.



Exemplo:

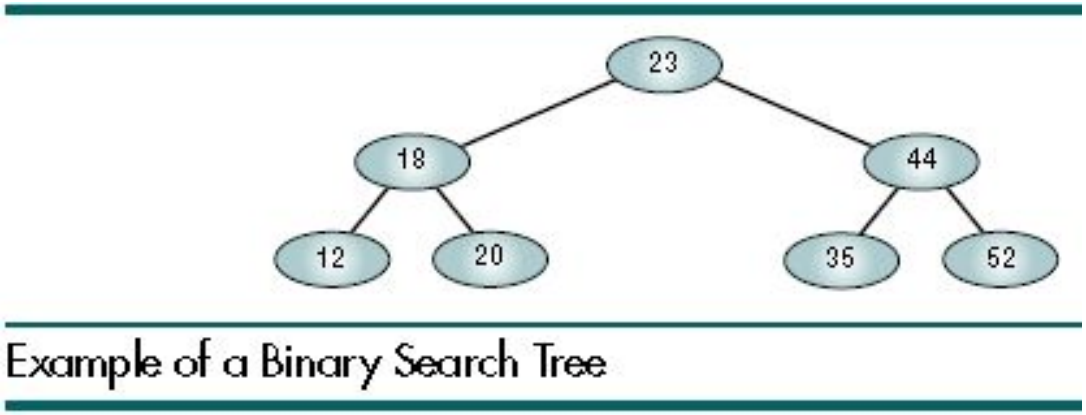
- O mínimo da árvore com raiz em 15 é achado seguindo o caminho 15-6-3-2 a partir da raiz.
- O mínimo é 2.

Complexidade:

- A complexidade da busca do mínimo no pior caso é de O(h), onde h é a altura da árvore.

Árvore de Busca Binária
Buscas – Maior Elemento

- A operação de encontrar o maior elemento é semelhante ao caso anterior. Trocando a busca nos ramos esquerdos pelos ramos direitos.
- Na árvore de exemplo, o maior elemento é aquele com valor 52, que corresponde à folha mais à direita da árvore.



Example of a Binary Search Tree

- A operação de encontrar o maior elemento então consiste em: começando na raiz, **seguir repetidamente o ramo direito até chegar a um nó folha**.

Árvore de Busca Binária
Buscas – Maior Elemento

- A operação de encontrar o maior elemento então consiste em: começando na raiz, **seguir repetidamente o ramo direito até chegar a um nó folha**. O algoritmo para encontrar o maior elemento é o seguinte:

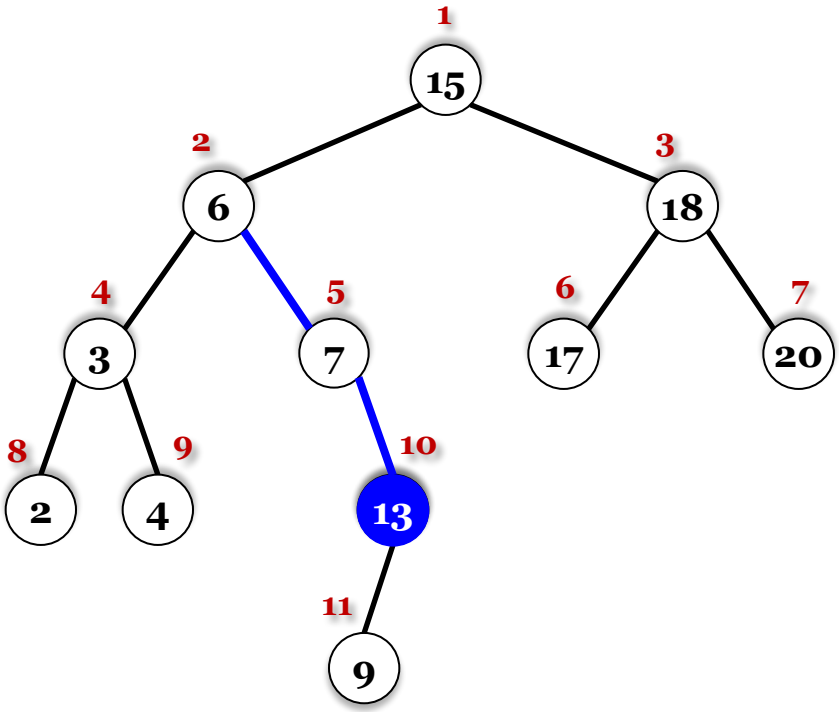
```
Algorithm findLargestBST (root)
This algorithm finds the largest node in a BST.
   Pre    root is a pointer to a nonempty BST or subtree
   Return address of largest node returned
1  if (right subtree empty)
   1   return (root)
2  end if
3  return findLargestBST (right subtree)
end findLargestBST
```

- Novamente, observe que trata-se de um algoritmo recursivo.

Árvore de Busca Binária

Buscas – Maior Elemento - Exemplo

- Outro exemplo sobre a busca do maior elemento.



Exemplo:

- O máximo da árvore com raiz em 6 é achado seguindo o caminho 6-6-7-13 a partir da raiz.
- O máximo é 13.

Complexidade:

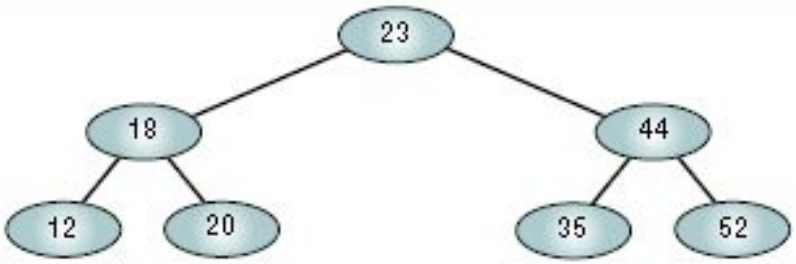
- A complexidade da busca do mínimo no pior caso é de O(h), onde h é a altura da árvore.

Árvore de Busca Binária

Busca BST – Busca Elemento Específico

- A operação de encontrar um elemento específico em uma árvore de busca binária é a mais importante caraterística que estas árvores possuem.

- Considere a tarefa de encontrar um elemento específico, por exemplo, o nó com valor 20, na árvore de exemplo.
- Começamos comparando o valor 20 (argumento de busca) com o valor na raiz da árvore. Como 20 é menor que o valor na raiz, 23, descemos pelo ramo esquerdo. Sabe-se que a sub-árvore esquerda contém os nós com valores menores que a raiz.
- Em seguida, comparamos o valor 20, como valor na raiz da sub-árvore esquerda. Como 20 é maior que o valor da raiz, 18, descemos pelo ramo direito.
- Finalmente, comparamos o valor 20, com a raiz da nova sub-árvore e achamos o valor desejado.



Example of a Binary Search Tree

Árvore de Busca Binária

Busca BST – Busca Elemento Específico

- O algoritmo que correspondente a operação de busca de um elemento específico, pode ser implementado de maneira recursiva como segue:

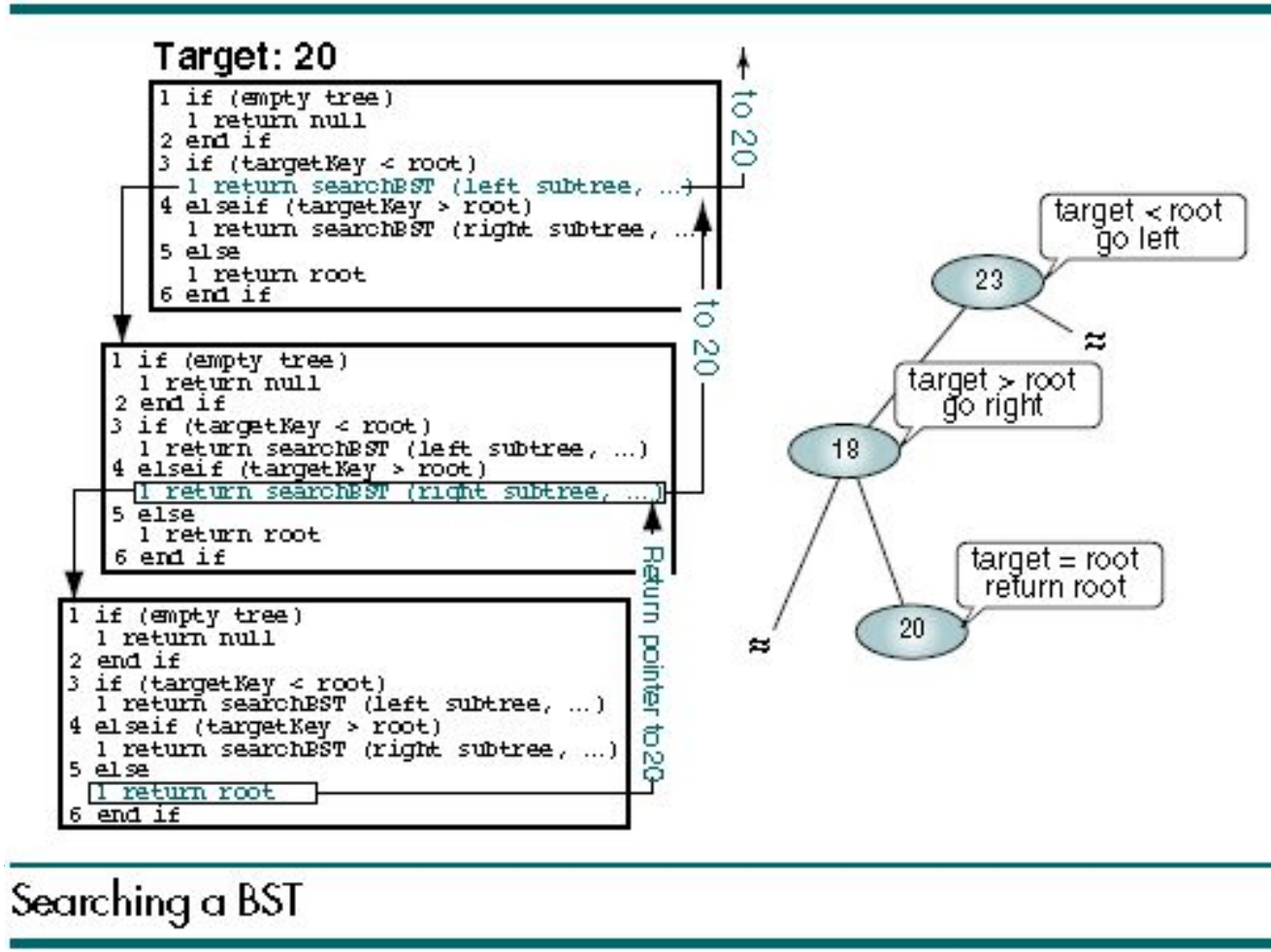
```
Algorithm searchBST (root, targetKey)
Search a binary search tree for a given value.
   Pre    root is the root to a binary tree or subtree
          targetKey is the key value requested
   Return the node address if the value is found
          null if the node is not in the tree
1  if (empty tree)
      Not found
   1  return null
2  end if
3  if (targetKey < root)
   1  return searchBST (left subtree, targetKey)
4  else if (targetKey > root)
   1  return searchBST (right subtree, targetKey)
5  else
      Found target key
   1  return root
6  end if
end searchBST
```

Árvore de Busca Binária

Busca BST – Busca Elemento Específico

- A figura ilustra o processo seguido pelo algoritmo de busca específico.
- Existem dois casos base:

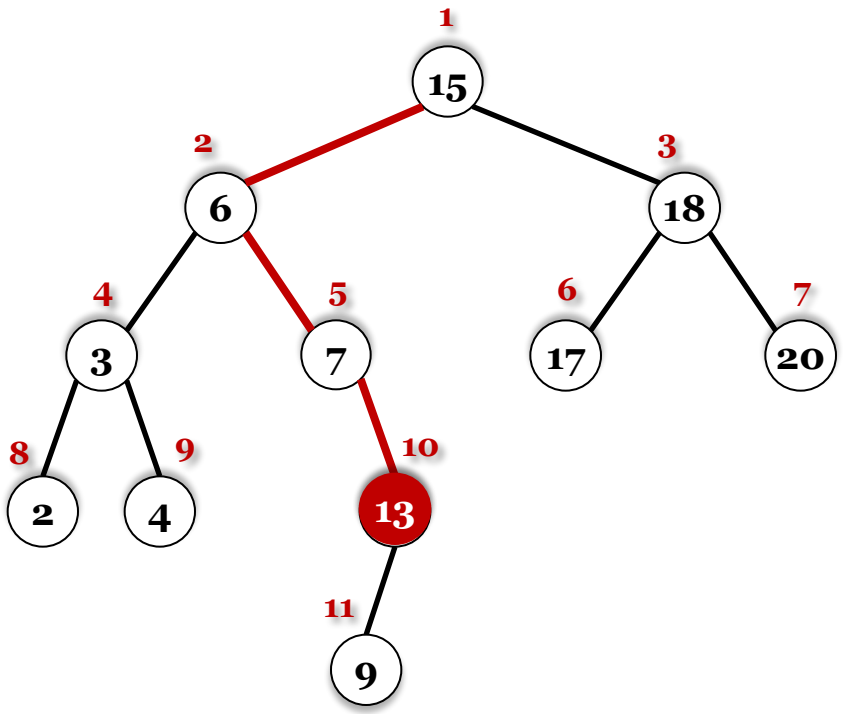
- O argumento de busca é encontrado na árvore e neste caso retorna-se o endereço do nó correspondente, ou;
- O argumento não existe e neste caso, retorna-se um ponteiro nulo.



Árvore de Busca Binária

Busca BST – Busca Elemento Específico

- A operação mais comum executada sobre uma árvore de pesquisa binária, é procurar por uma chave armazenada na árvore.



Exemplo:

- Para procurar pela chave 13 na árvore:

  O caminho 15-6-7-13 é seguido a partir da raiz.

Complexidade:

- A complexidade da busca no pior caso é de O(h), onde h é a altura da árvore.

Árvore de Busca Binária

Inserção

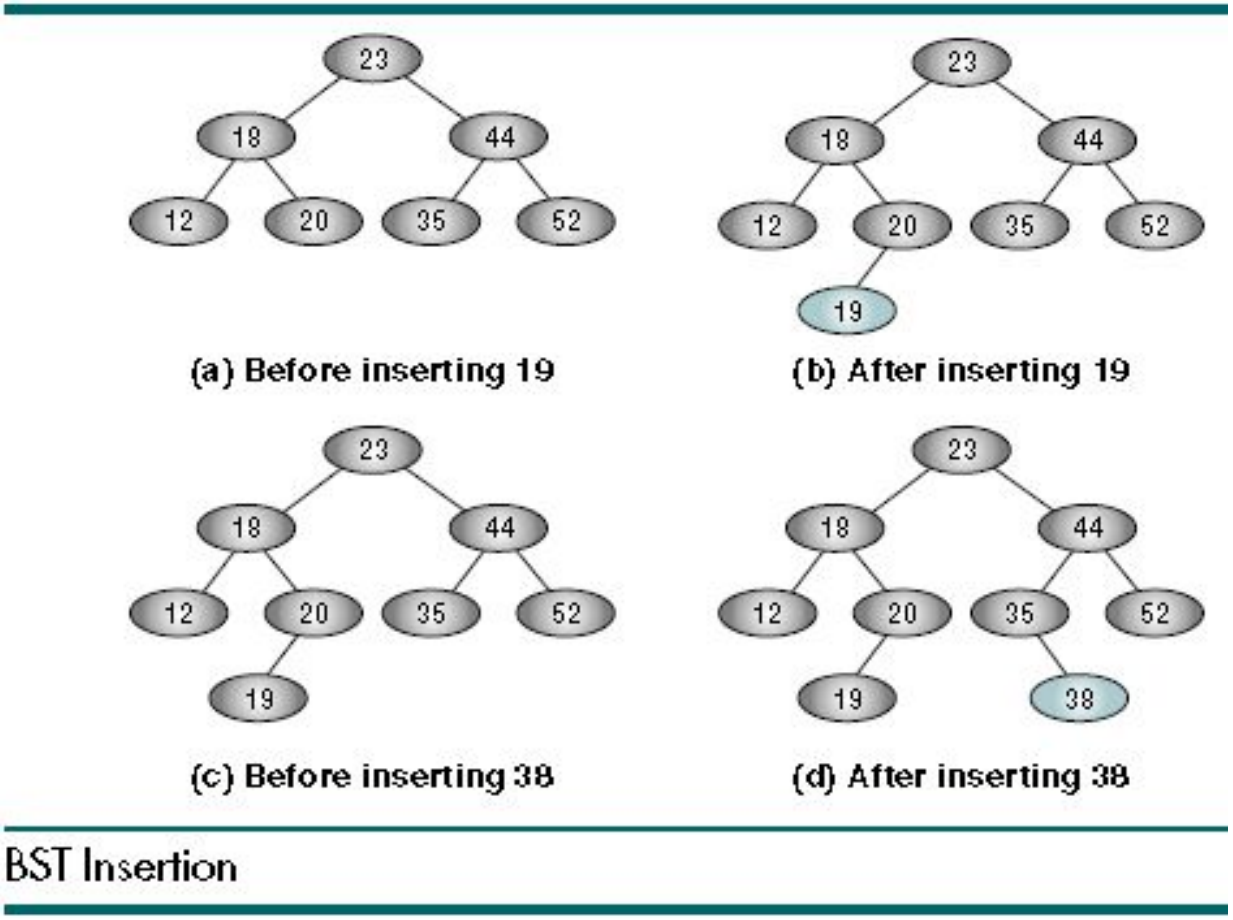
- A operação de inserção adiciona um novo nó (dado ou registro) à árvore de busca binária (BST). Para realizar a **inserção de um novo nó**, é preciso determinar qual será a posição que corresponderá ao novo nó na árvore.
- Este processo é bastante simples, basta comparar o valor do novo nó, com o valor de cada nó da árvore, começando pela raiz, descendo pelo ramo adequado conforme o novo nó seja menor (descer pelo ramo esquerdo) ou maior ou igual (descer pelo ramo direito). **O processo continua até achar um nó que não possui filho no ramo escolhido.**
- **A inserção sempre acontece em um nó sem filho no ramo escolhido, que pode ser um nó folha ou um nó com apenas um filho.**

Árvore de Busca Binária

Inserção - Exemplo

- A figura ilustra a inserção de dois nós. Primeiro o nó 19 e depois o nó 38.

- Para inserir o nó 19, primeiro segue-se o caminho 23-18-20 e encontra-se um **ramo esquerdo vazio**.
- O nó 19, é inserido como filho esquerdo do nó 20.
- Para inserir o nó 38, primeiro segue-se o caminho 23-44-35 e encontra-se um **ramo direito vazio**.
- O nó 38, é inserido como filho direito do nó 35.
- **Em ambos casos, a inserção aconteceu em um nó folha.**

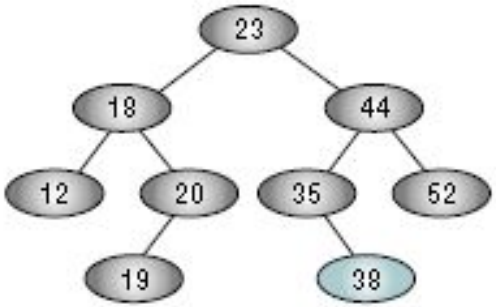


Árvore de Busca Binária

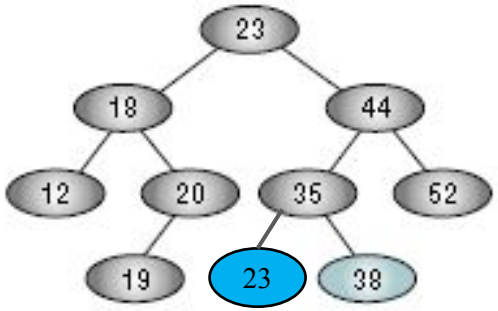
Inserção - Exemplo

- Considere a inserção do nó 23 duplicado.

- Para inserir o nó 23, primeiro segue-se o caminho 23-44-35 e encontra-se um ramo esquerdo vazio.
- O nó 23, é inserido como filho esquerdo do nó 35.
- **Neste caso, a inserção acontece em um nó com um único filho.**



(e) Antes de inserir 23

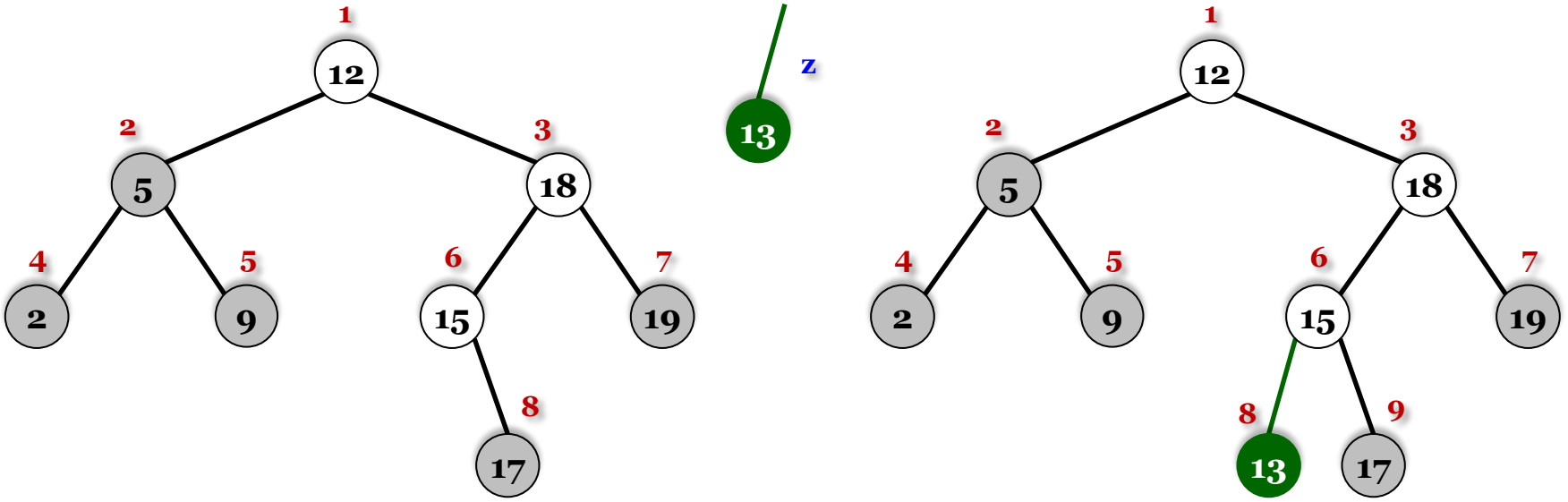


(f) Depois de inserir 23

Árvore de Busca Binária

Inserção - Exemplo

- Outro exemplo de Inserção.



Árvore de Busca Binária

Inserção – Algoritmo

- O algoritmo que correspondente a operação de inserção pode ser implementado de maneira recursiva como segue:

- Se a árvore ou subárvore está vazia, inserimos o novo dado como raiz.
- Se a árvore ou subárvore não esta vazia, determinamos qual ramo seguir por comparação e fazemos uma chamada recursiva addBST() indicando a sub-árvore esquerda ou direita de acordo com o caso.

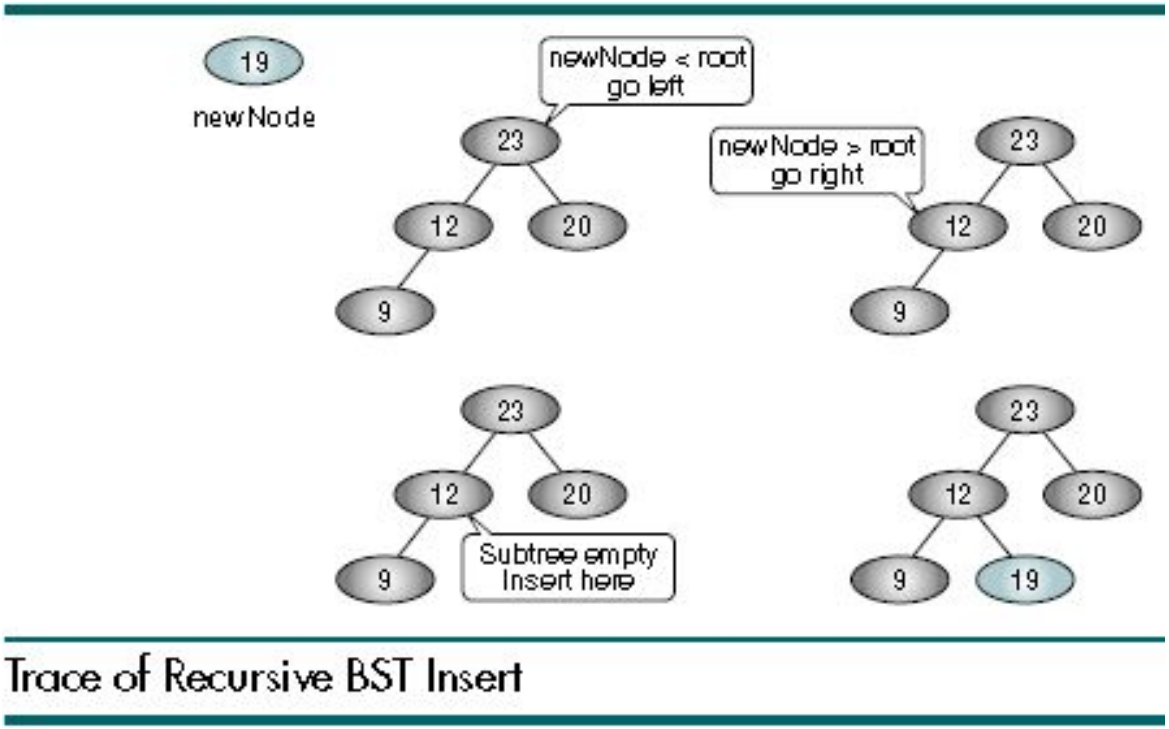
```
Algorithm addBST (root, newNode)
Insert node containing new data into BST using recursion.
   Pre    root is address of current node in a BST
          newNode is address of node containing data
   Post   newNode inserted into the tree
   Return address of potential new tree root
1  if (empty tree)
   1   set root to newNode
   2   return newNode
2  end if
   Locate null subtree for insertion
3  if (newNode < root)
   1   return addBST (left subtree, newNode)
4  else
   1   return addBST (right subtree, newNode)
5  end if
end addBST
```

- Esta implementação é bastante elegante.

Árvore de Busca Binária
Inserção – Algoritmo

- A figura ilustra o processo de inserção do nó 19 na árvore com raiz 23.

- Para inserir o nó 19 na árvore começamos pela raiz.
- Como o novo nó (19), é menor que a raiz (23), realizamos uma chamada recursiva addBST(), usando a sub-árvore esquerda com raiz em 12.
- Agora o novo nó (19), é maior que a raiz (12), fazemos uma chamada recursiva addBST(), usando a sub-árvore direita, que é nula.
- Como a sub-árvore é nula, inserimos o novo nó (19) como raiz. Filho direito de (12).



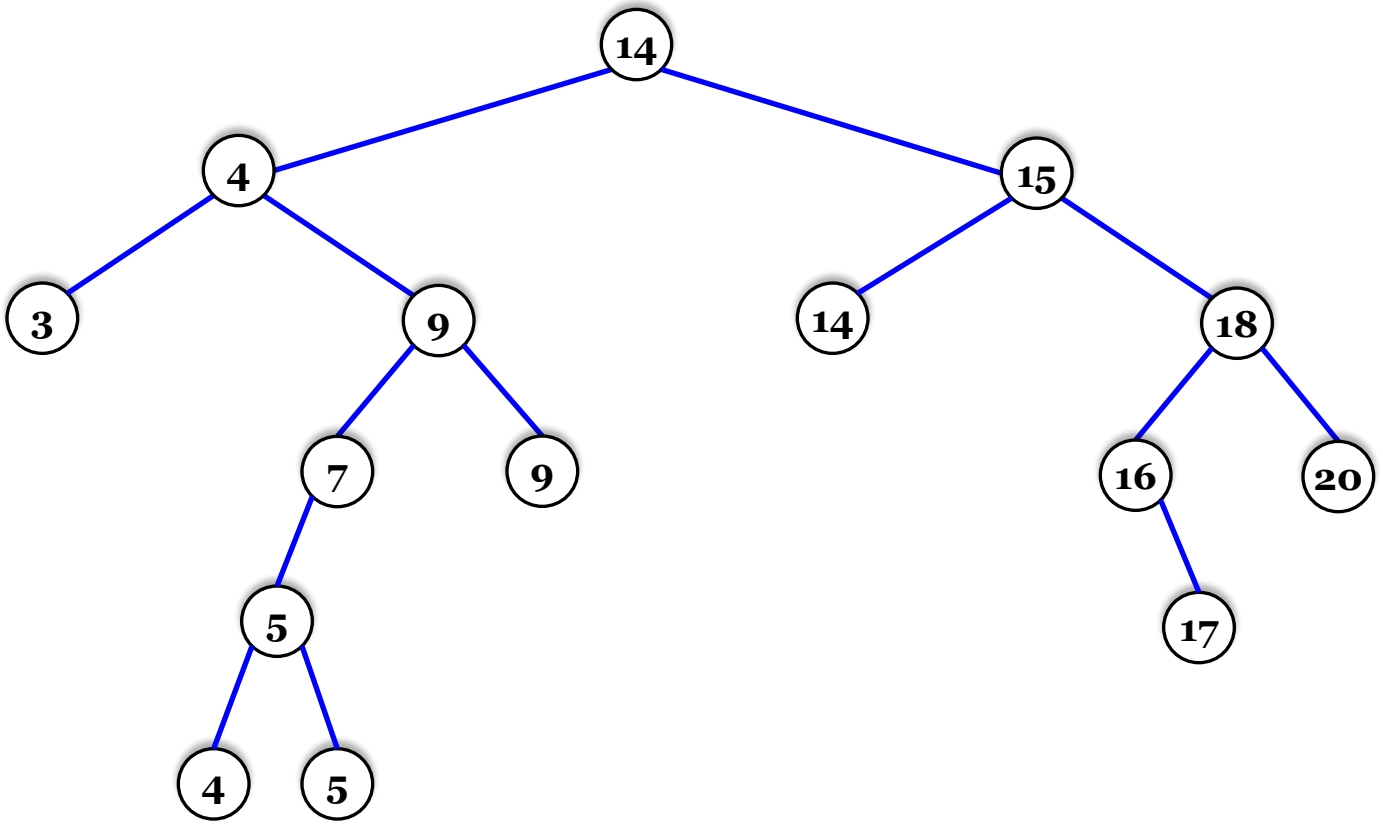
Trace of Recursive BST Insert

Árvore de Busca Binária

Inserção – Exemplo

- Considere seguinte sequência de inserções em uma árvore de busca binária.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
14	15	4	9	7	18	3	5	16	4	20	17	9	14	5



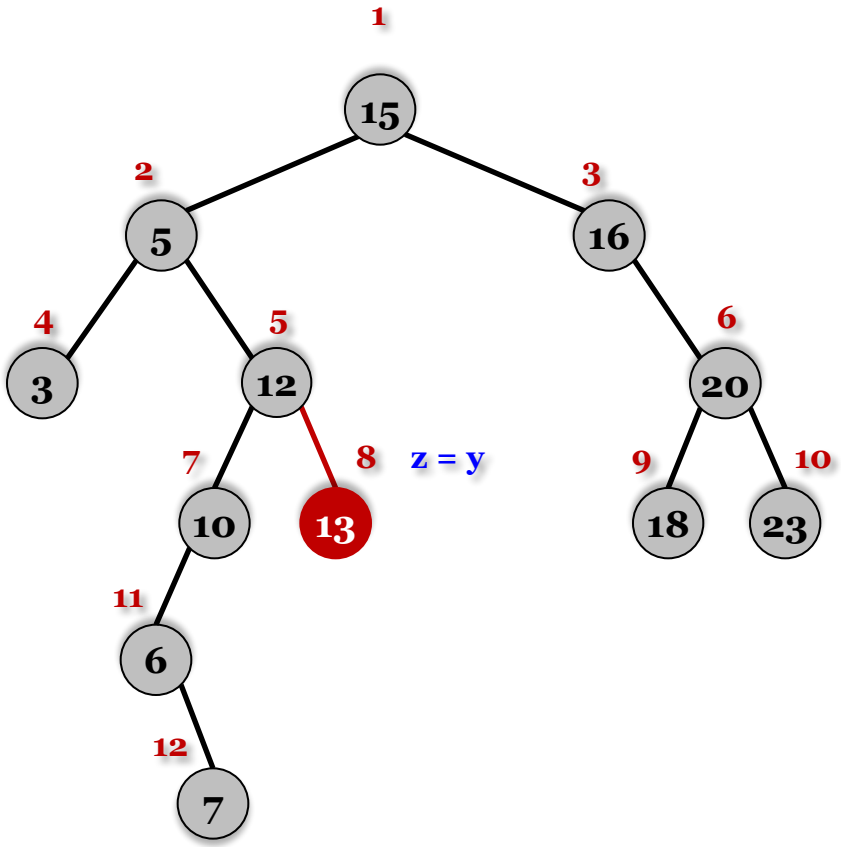
Árvore de Busca Binária
Remoção

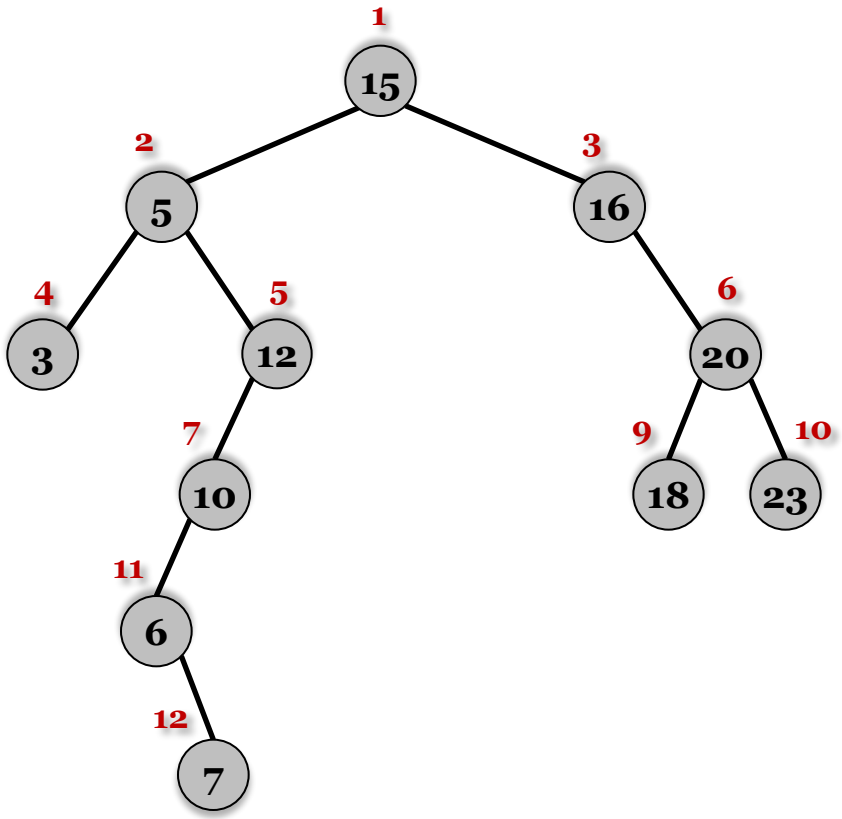
- Para realizar a **remoção de um nó** de uma árvore de busca binária, primeiro devemos localizar esse nó. Existem quatro casos possíveis:
  - 1.- O nó a ser removido não possui filhos. Neste caso, apenas eliminamos o nó.
  - 2.- O nó a ser removido somente possui um filho direito (sub-árvore direita). Neste caso, eliminamos o nó e reconectamos a sub-árvore direita ao nó pai do nó que foi removido, preservando o mesmo tipo de relação.
  - 3.- O nó a ser removido somente possui um filho esquerdo (sub-árvore esquerda). Neste caso, eliminamos o nó e reconectamos a sub-árvore esquerda ao nó pai do nó que foi removido, preservando o mesmo tipo de relação.
  - 4.- O nó a ser removido possui dois filhos (duas sub-árvores). Neste caso, é possível eliminar um nó no meio da árvore, porém o resultado tende a criar uma árvore muito desbalanceada. Em vez disso, procura-se manter a estrutura o mais inalterada possível, encontrando dados (nas folhas ou quase folhas) que possam ocupar o lugar do dado removido no meio. Podemos fazer isto de duas maneiras:
    - i) Encontrar o maior nó, na sub-árvore esquerda do nó removido e copiar (sobre-escrever) seu dado no dado do nó removido.
    - ii) Encontrar o menor nó, na sub-árvore direita do nó removido e copiar (sobre-escrever) seu dado no dado do nó removido.
  - Em ambos casos, o nó encontrado (maior o menor) será copiado e removido.

Árvore de Busca Binária

Remoção

- Exemplo do Caso1.- O nó a ser removido não possui filhos.

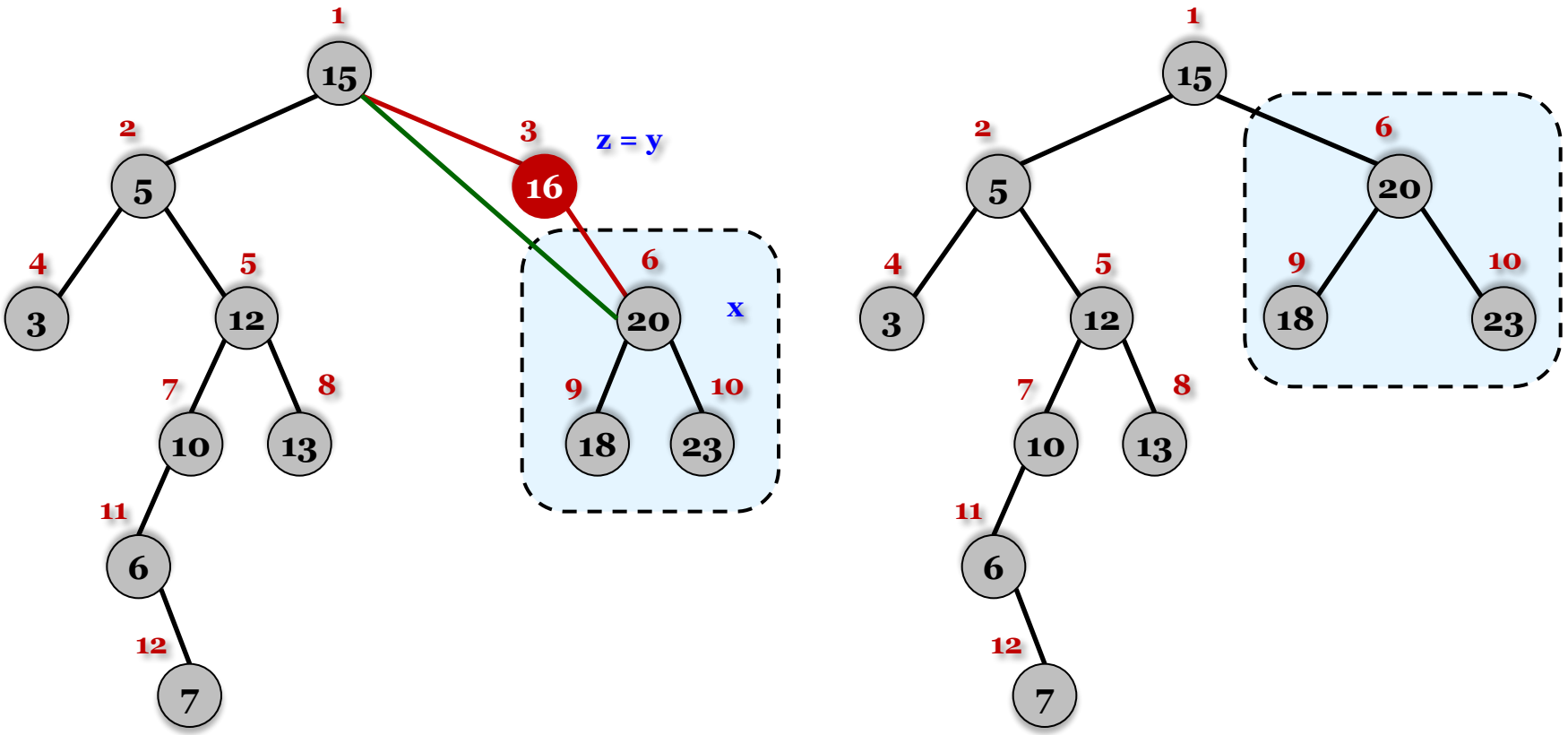




Árvore de Busca Binária

Remoção

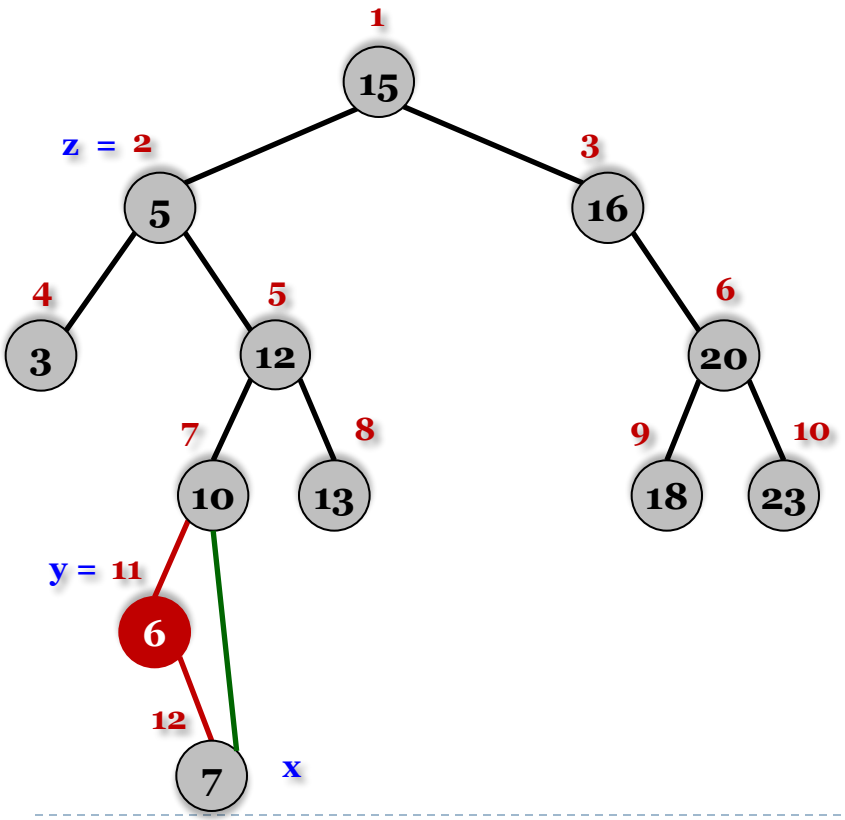
- Exemplo do Caso 2.- O nó a ser removido possui 1 filho direito.

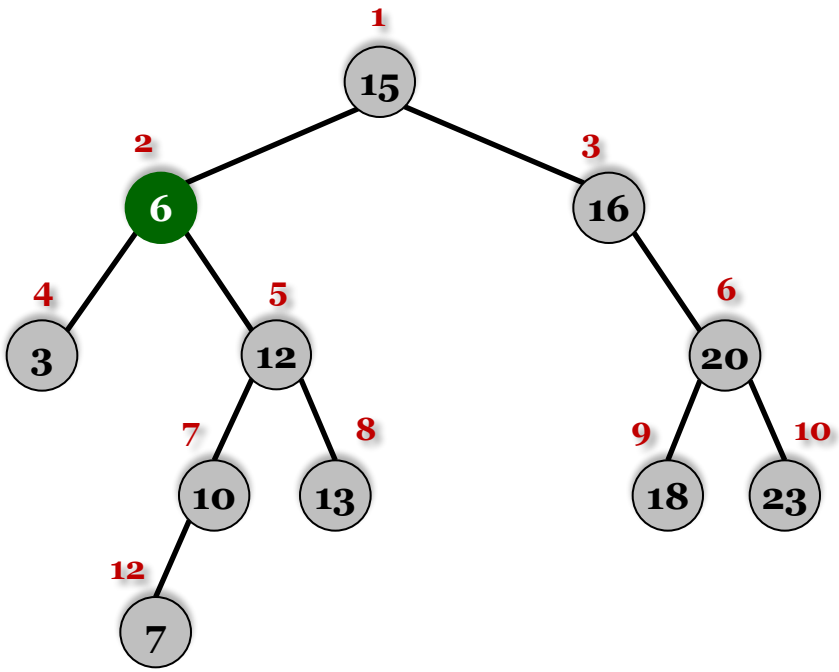


Árvore de Busca Binária

Remoção

- Exemplo do Caso 4.- O nó a ser removido possui 2 filhos.





Árvore de Busca Binária
Remoção-Algoritmo

- O algoritmo que correspondente a operação de remoção pode ser implementado de maneira recursiva.

- Exitem dois casos base:
- i) Se o nó não é encontrado.
- ii) Se o nó foi encontrado e não possui uma das subárvores.
- Já no caso em que o nó possui as das subárvores aplica-se o caso 4.

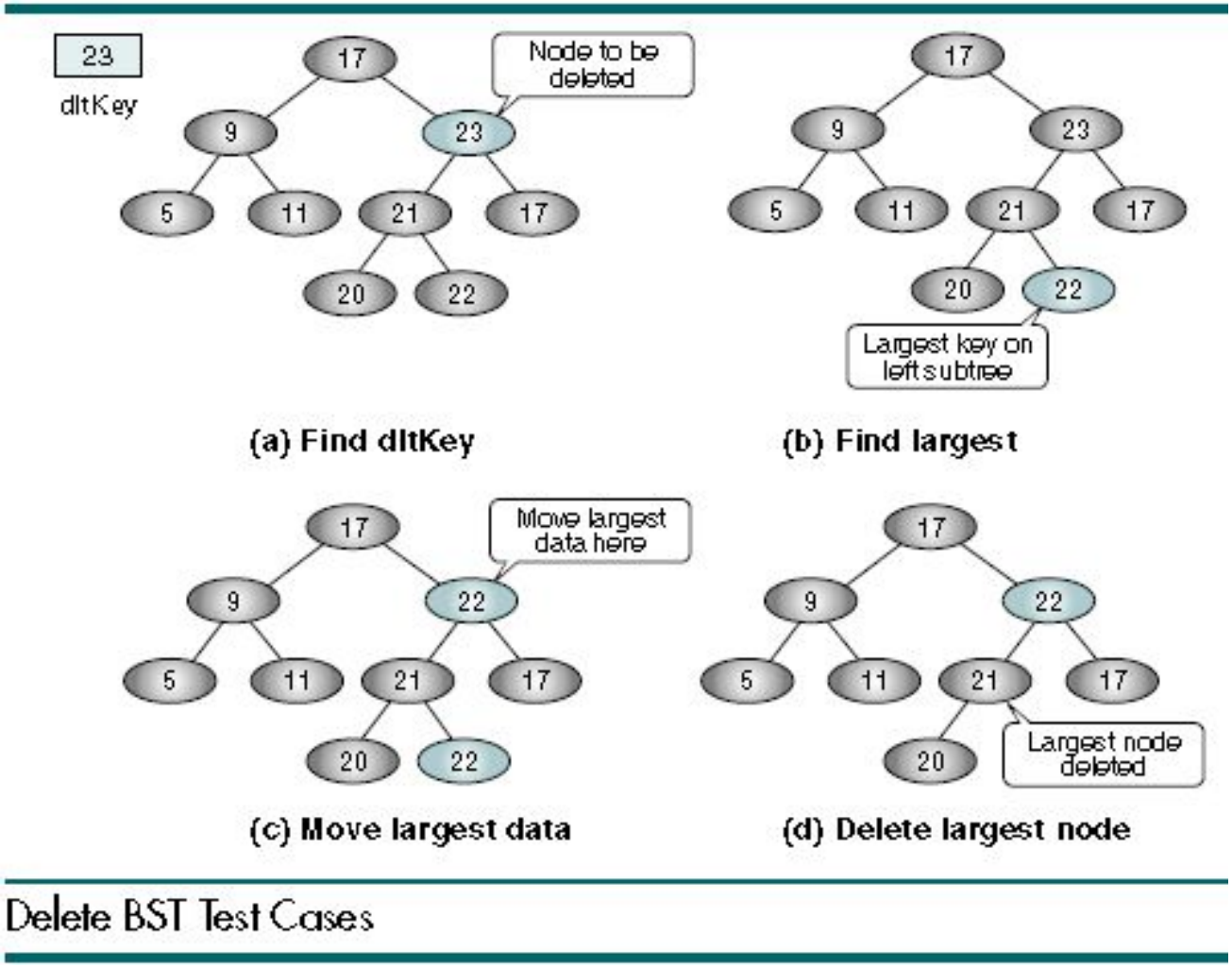
- **Observe a chamada recursiva que tenta eliminar o maior nó na sub-árvore esquerda.**

```
Algorithm deleteBST (root, dltKey)
This algorithm deletes a node from a BST.
   Pre    root is reference to node to be deleted
          dltKey is key of node to be deleted
   Post   node deleted
          if dltKey not found, root unchanged
   Return true if node deleted, false if not found
1  if (empty tree)
   1  return false
2  end if
3  if (dltKey < root)
   1  return deleteBST (left subtree, dltKey)
4  else if (dltKey > root)
   1  return deleteBST (right subtree, dltKey)
5  else
      Delete node found--test for leaf node
   1  If (no left subtree)
      1  make right subtree the root
      2  return true
   2  else if (no right subtree)
      1  make left subtree the root
      2  return true
   3  else
         Node to be deleted not a leaf. Find largest node on
         left subtree.
      1  save root in deleteNode
      2  set largest to largestBST (left subtree)
      3  move data in largest to deleteNode
      4  return deleteBST (left subtree of deleteNode,
                           key of largest
   4  end if
6  end if
end deleteBST
```

Árvore de Busca Binária

Remoção – Caso 4

- A figura ilustra o caso mais complexo, o caso 4, quando o nó a ser removido possui dois filhos.



Referências

- Gilberg, R.F. e Forouzan, B.A. Data Structures_A Pseudocode Approach with C. Capítulo 7. Binary Search Trees. Segunda Edição. Editora Cengage, Thomson Learning, 2005.