Unidade VI:
Árvores Alvinegras

Instituto de Ciências Exatas e Informática
Departamento de Ciência da Computação

Introdução

- Estrutura de dados mais eficiente de representar as árvores-2.3.4, evitando o desperdício de memória

- Substitui a representação múltipla de nós por uma representação única contendo os atributos: elemento, apontadores esq e dir e um bit de cor

Exercício Resolvido (1)

- Implemente a classe Nó da árvore alvinegra

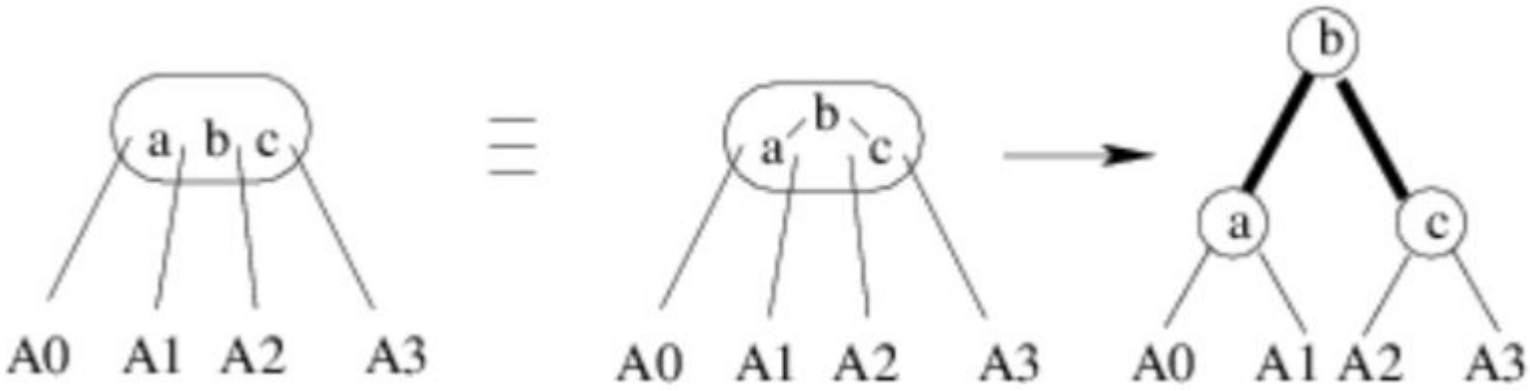
Exercício Resolvido (1)

- Implemente a classe Nó da árvore alvinegra

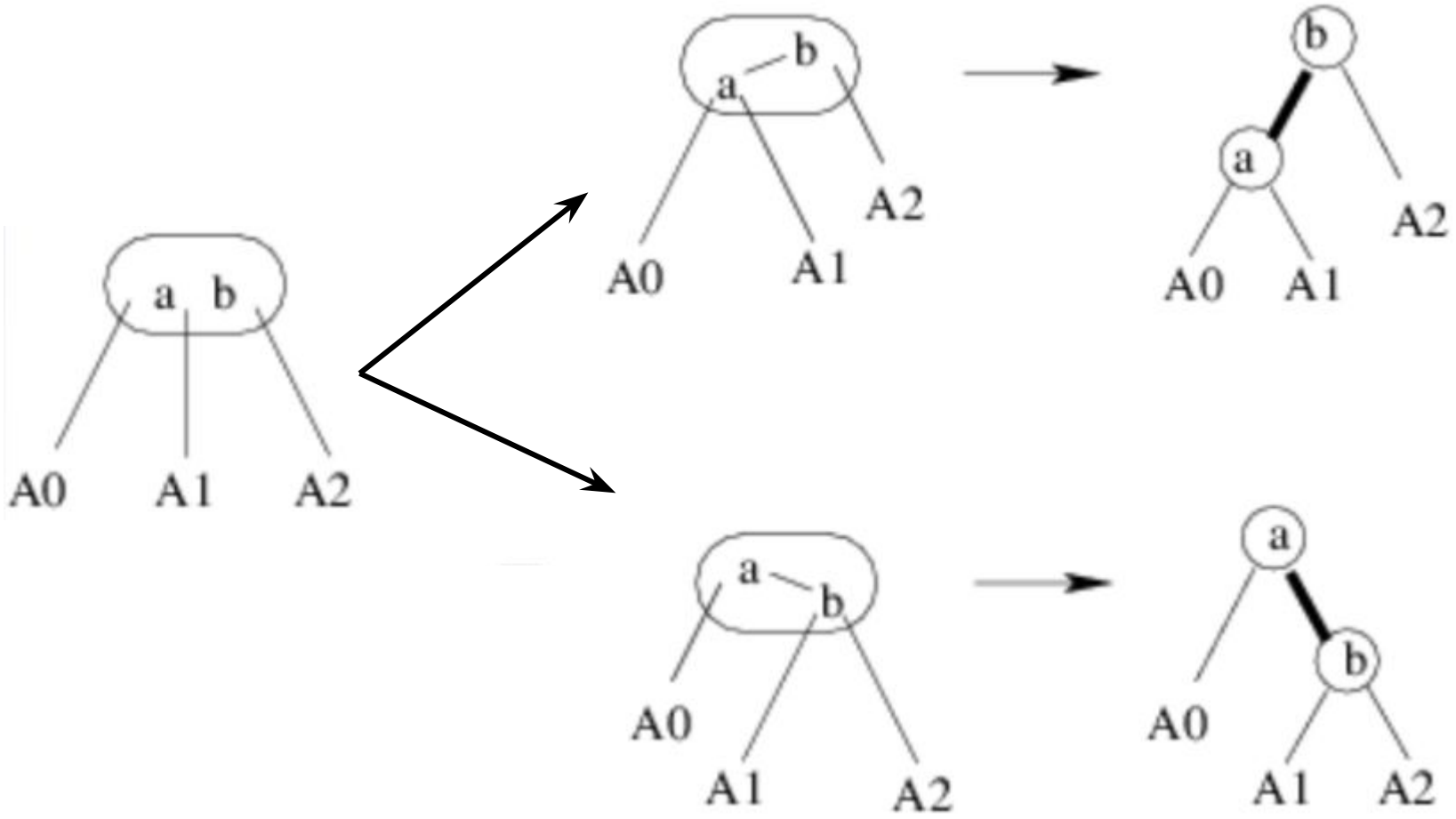
```
class NoAN{
  public boolean cor;
  public int elemento;
  public NoAN esq, dir;
  public NoAN (){
      this(-1);
  }
  public NoAN (int elemento){
      this(elemento, false, null, null);
  }
  public NoAN (int elemento, boolean cor){
      this(elemento, cor, null, null);
  }
  public NoAN (int elemento, boolean cor, NoAN esq, NoAN dir){
    this.cor = cor;
    this.elemento = elemento;
    this.esq = esq;
    this.dir = dir;
  }
}
```

Colorindo as Arestas

- Simula a hierarquia 2.3.4, colorindo as arestas entre os nós de duas formas:
  - **Aresta preta (traço grosso)**: Entre gêmeos (mesmo nó na 2.3.4)
  - **Aresta branca (traço fino)**: Entre não gêmeos na 2.3.4

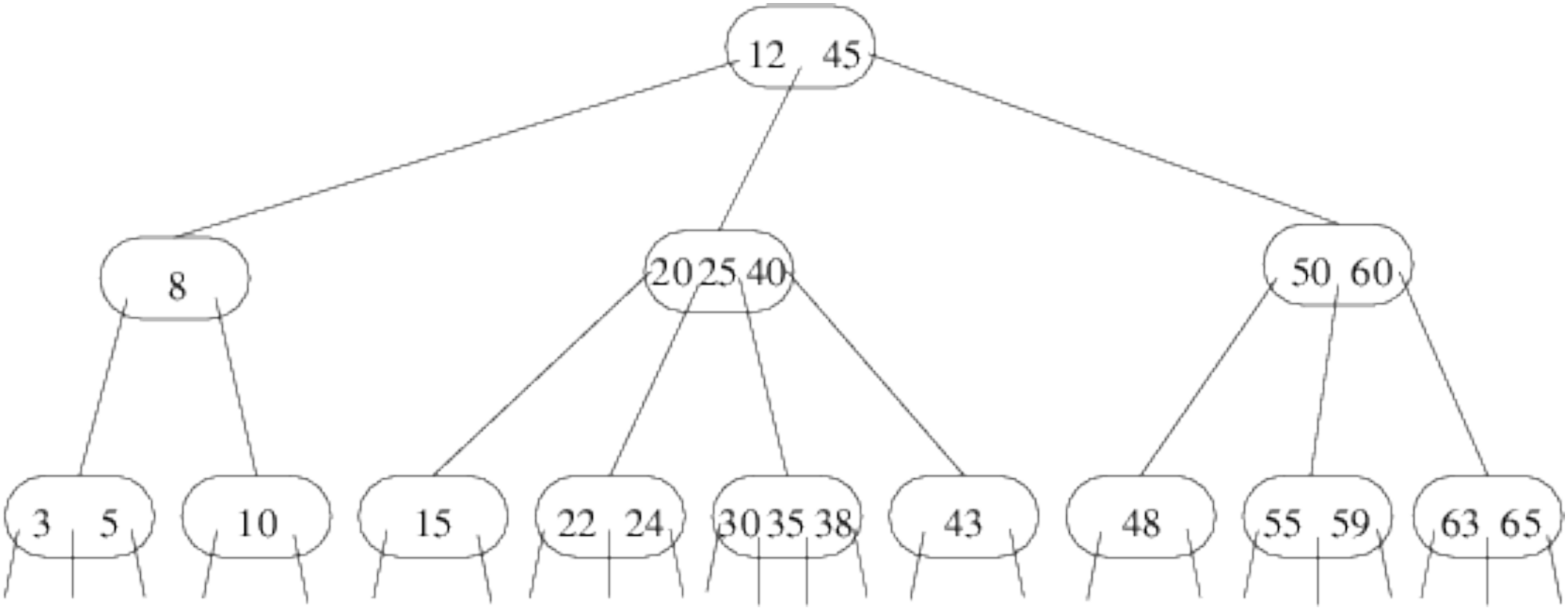


Colorindo as Arestas para Nós do Tipo 3-Nó



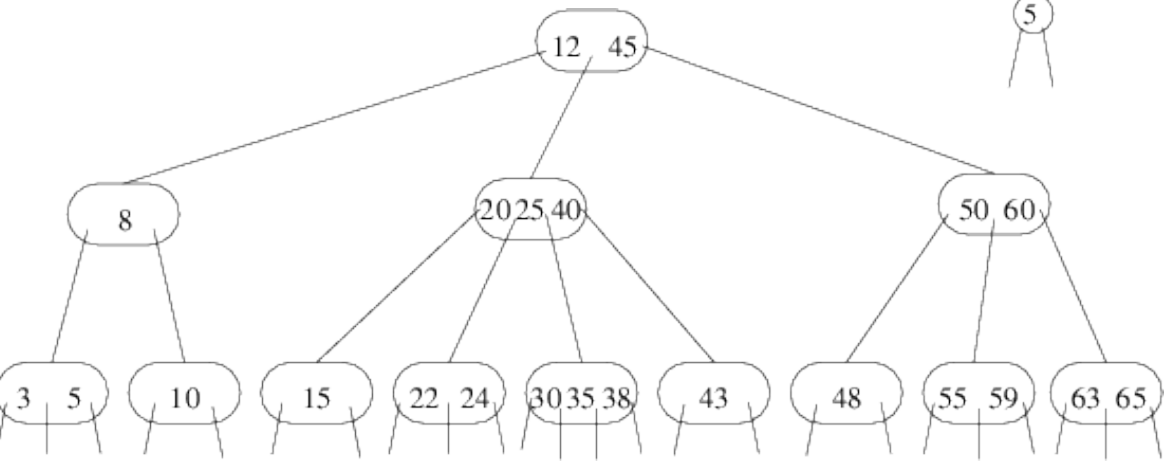
Exercício Resolvido (2)

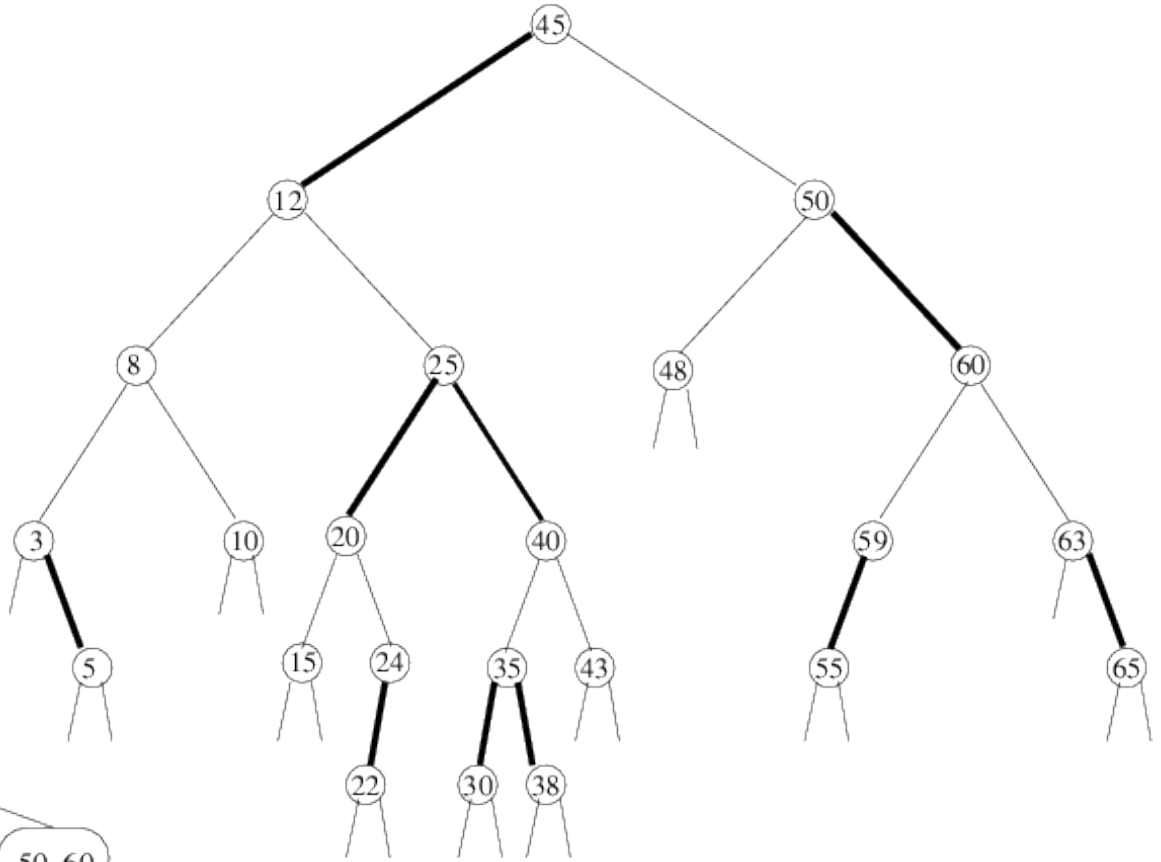
- Encontre uma árvore alvinegra equivalente à 2.3.4 abaixo:



Exercício Resolvido (2)

- Encontre uma árvore alvinegra equivalente à 2.3.4 abaixo:





Exercício Resolvido (3)

- Dada uma 2.3.4 com altura *h*, quais são a menor e maior alturas possíveis da alvinegra correspondente? Quando temos cada uma dessas alturas?

Exercício Resolvido (3)

- Dada uma 2.3.4 com altura *h*, quais são a menor e maior alturas possíveis da alvinegra correspondente? Quando temos cada uma dessas alturas?

Resposta: A menor é *h* e a maior, 2h. Elas acontecem, por exemplo, quando todos os nós são do tipo 2-nó e 4-nó, respectivamente

Exercício Resolvido (4)

- Qual é a relação entre as arestas da 2.3.4 com as brancas da alvinegra?

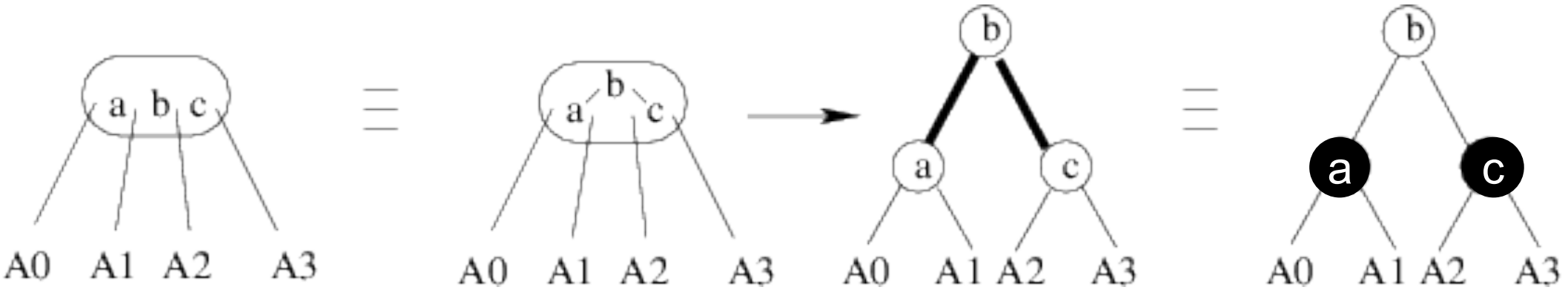
Exercício Resolvido (4)

- Qual é a relação entre as arestas da 2.3.4 com as brancas da alvinegra?

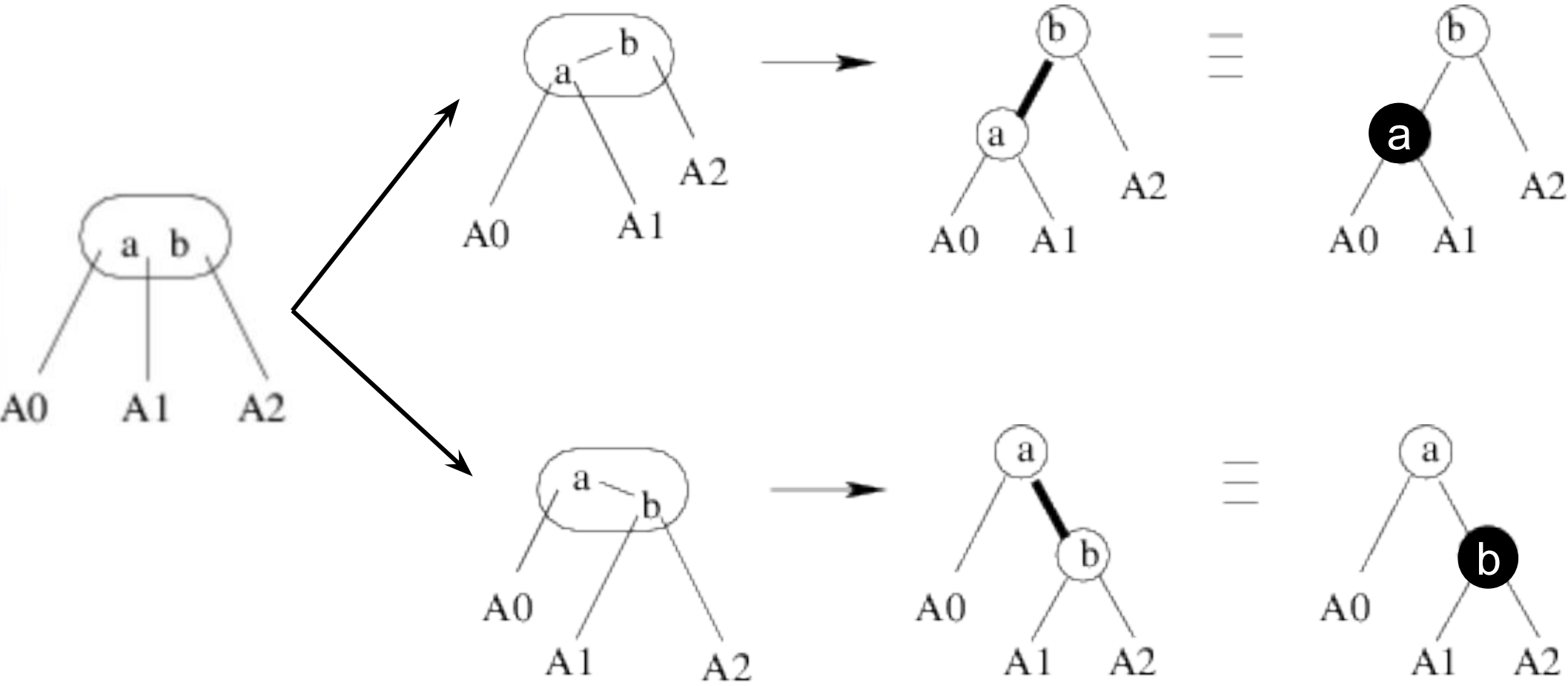
Resposta: As brancas são exatamente as arestas da 2.3.4

Colorindo os Nós

- Nossa atribuição de cores, na verdade, será nos nós em vez das arestas:
  - **Nó preto**: Nó gêmeo do pai na 2.3.4
  - **Nó branco**: Nó **NÃO** gêmeo do pai na 2.3.4

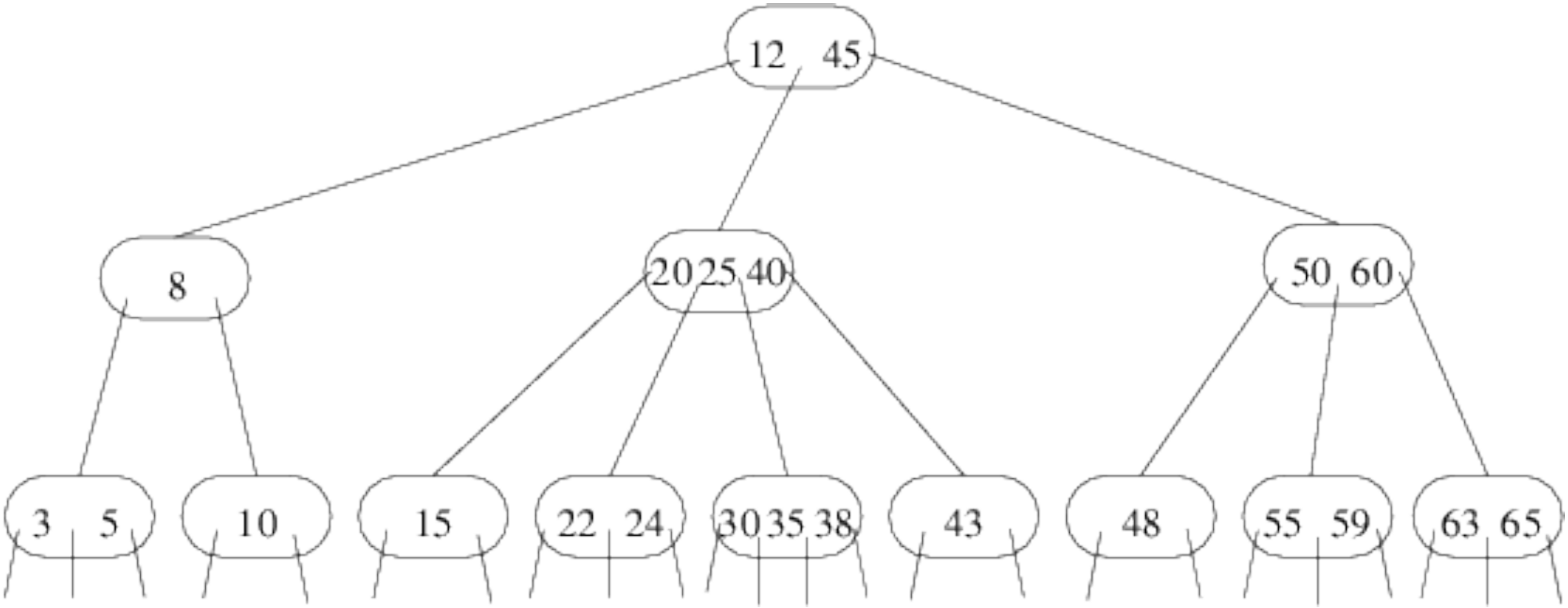


Colorindo os Nós do Tipo 3-Nó



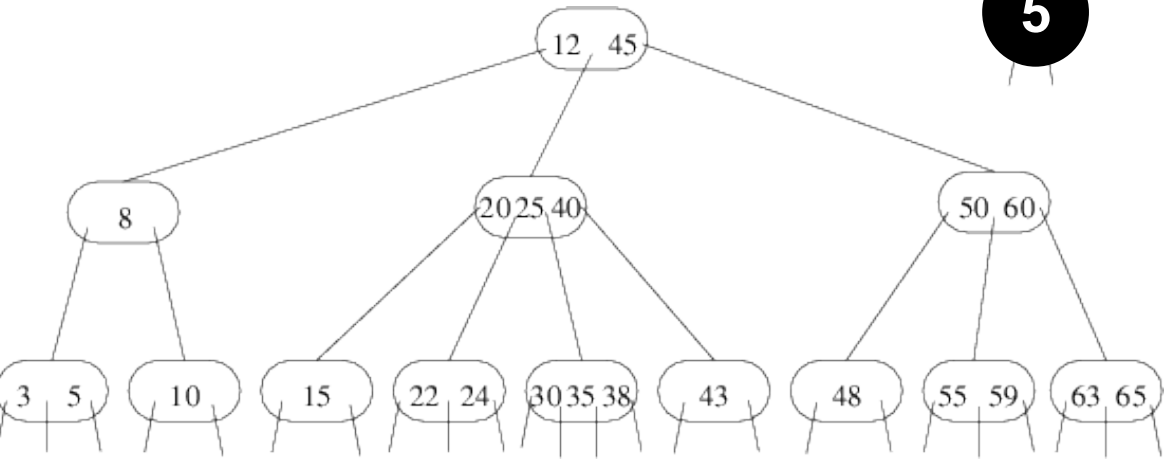
Exercício Resolvido (5)

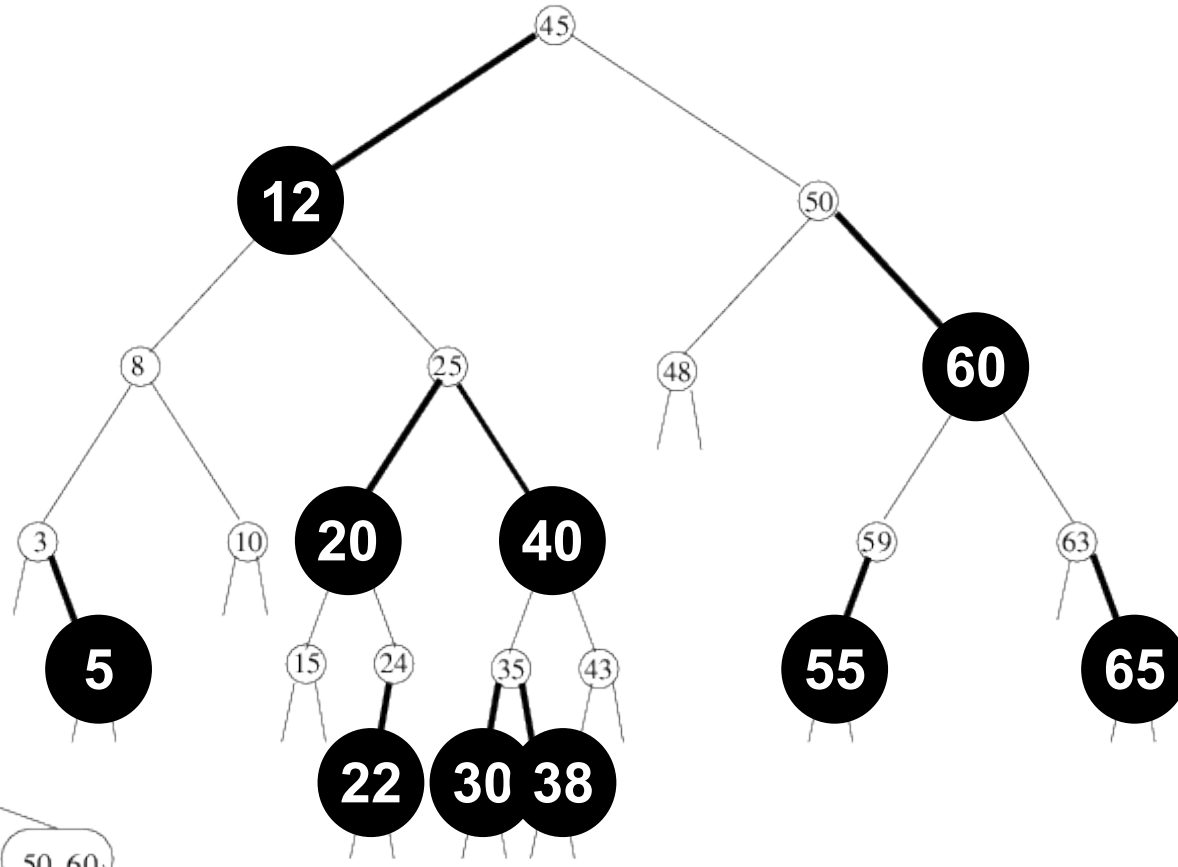
- Encontre uma árvore alvinegra equivalente à 2.3.4 abaixo**, colorindo os nós:**



Exercício Resolvido (5)

- Encontre uma árvore alvinegra equivalente à 2.3.4 abaixo**, colorindo os nós:**





Pesquisa e Caminhamento

- As duas operações desconsideram as cores e utilizam o mesmo procedimento das árvores binárias

- O pior caso (e o médio também) da pesquisa faz $\Theta(\lg(n))$ comparações

- O caminhamento faz $\Theta(n)$ visitas

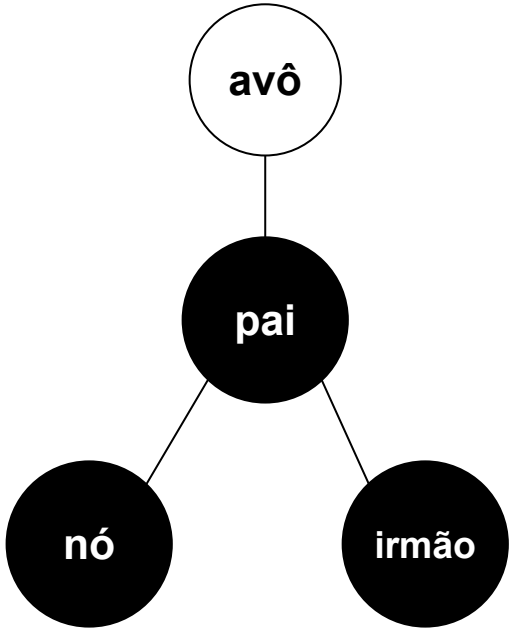
Exercício Resolvido (6)

- Qual é o número máximo de nós pretos consecutivos?

Exercício Resolvido (6)

- Qual é o número máximo de nós pretos consecutivos?

Resposta: Nunca teremos nós pretos consecutivos. Isso significaria mais de três elementos em um nó 2.3.4. Nesse caso, o nó e seu irmão seriam gêmeos do pai e esse, gêmeo do avô



Inserção de Elementos em Folhas Pretas

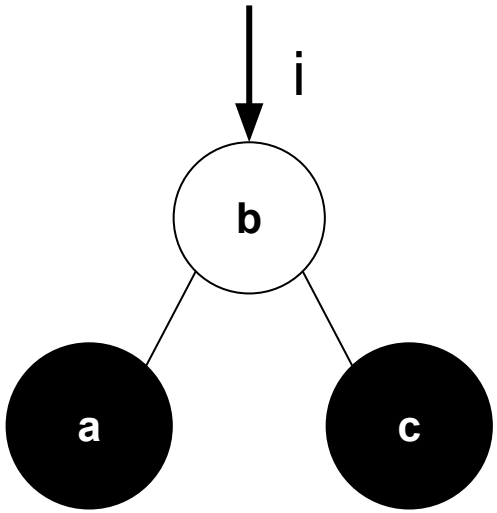
- Nossa inserção nas árvores binárias e 2.3.4 acontece nas folhas

- Inserindo um elemento em um 2-nó ou 3-nó da 2.3.4, tal elemento será gêmeo dos existentes naquele nó. Em um 4-nó, temos uma fragmentação e o novo elemento será gêmeo em um dos nós fragmentados

- Na alvinegra, toda folha é preta, pois seu elemento é gêmeo do elemento pai na 2.3.4 correspondente

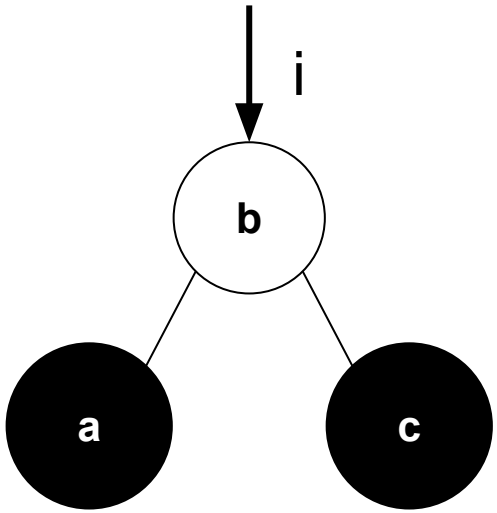
Exercício Resolvido (7)

- Faça o método ***boolean isNoTipoQuatro(NoAN i)*** que recebe um ponteiro para um nó da alvinegra e retorna true se esse elemento e seus dois filhos são gêmeos na 2.3.4 correspondente



Exercício Resolvido (7)

- Faça o método ***boolean isNoTipoQuatro(NoAN i)*** que recebe um ponteiro para um nó da alvinegra e retorna true se esse elemento e seus dois filhos são gêmeos na 2.3.4 correspondente

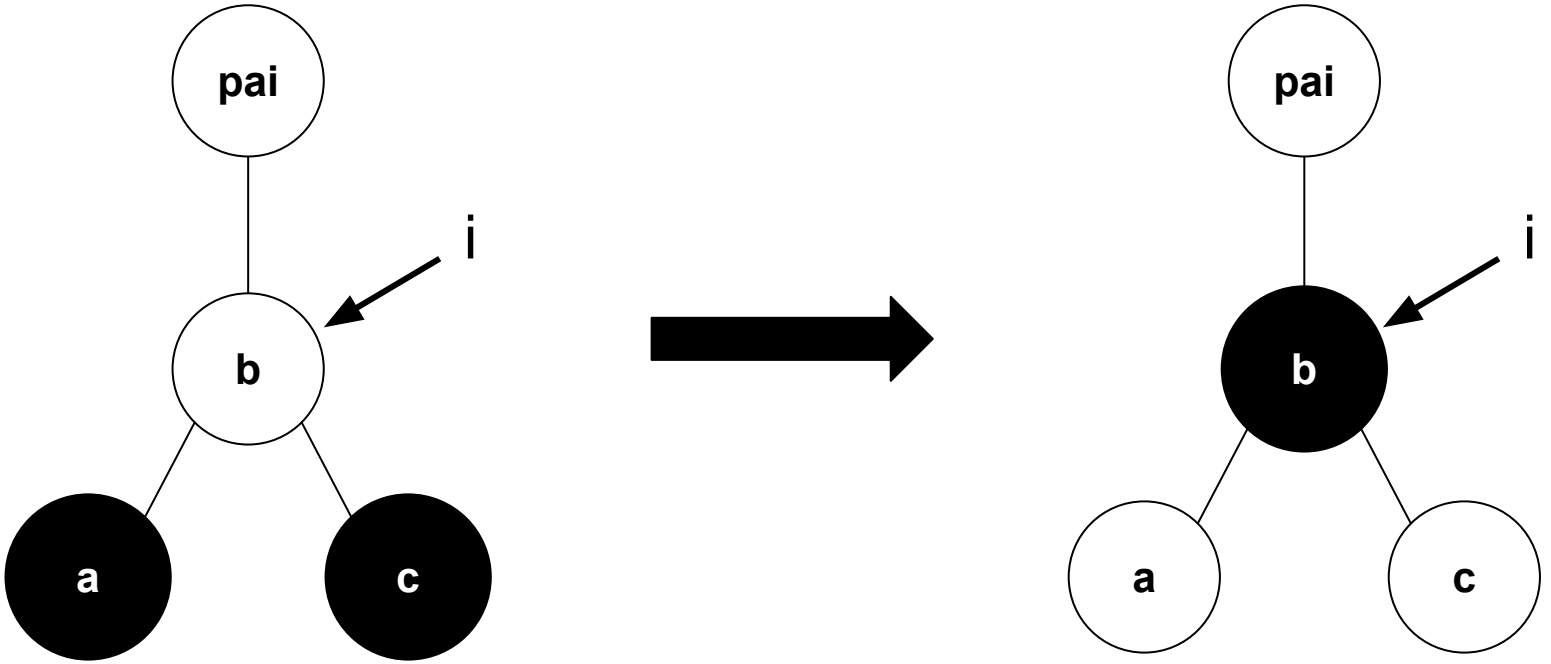



```
boolean isNoTipoQuatro(NoAN i){
    return (i.esq != null && i.dir != null && i.esq.cor == true && i.dir.cor == true);
}
```

Inserção

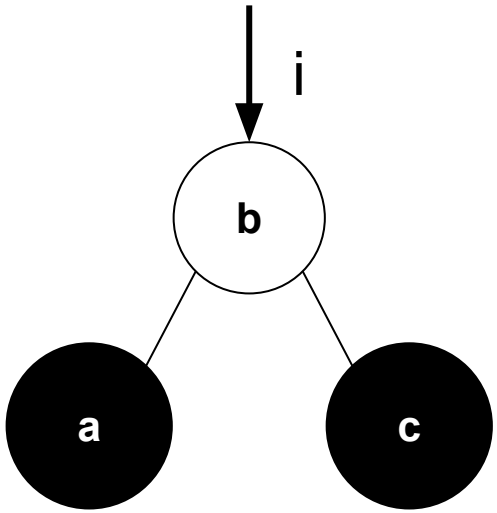
- Pode ser feita simulando a inserção com fragmentação na descida na 2.3.4

- A fragmentação de um 4-nó corresponde a inversão de cores dos elementos gêmeos



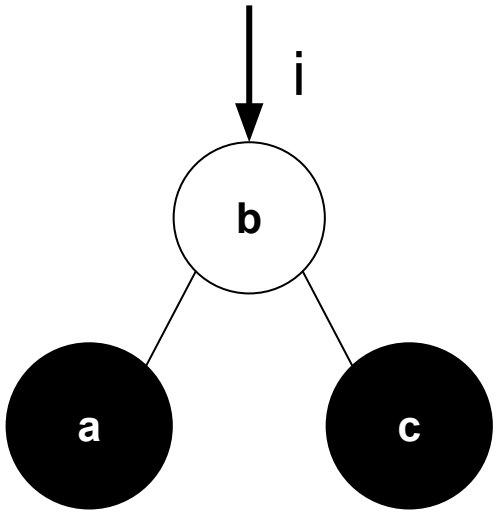
Exercício Resolvido (8)

- Faça o método ***void fragmentarNoTipoQuatro(NoAN i)*** que recebe um ponteiro para um nó da alvinegra e fragmenta esse nó (e seus filhos)



Exercício Resolvido (8)

- Faça o método ***void fragmentarNoTipoQuatro(NoAN i)*** que recebe um ponteiro para um nó da alvinegra e fragmenta esse nó (e seus filhos)

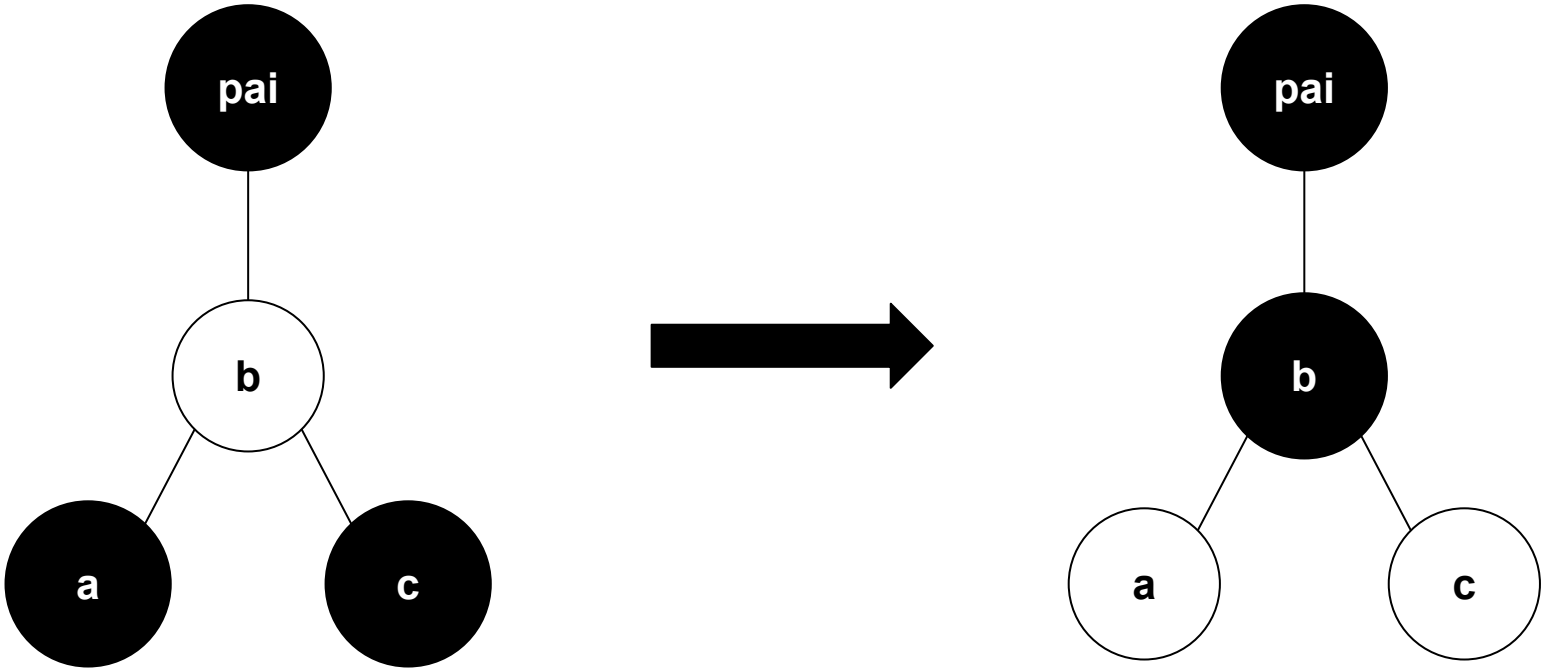



```
//supondo que seja um nó do tipo quatro
void fragmentarNoTipoQuatro(NoAN i){
      i.cor = !i.cor;
      i.esq.cor = !i.esq.cor;
      i.dir.cor =!i.dir.cor;
}
```

```
//supondo que seja um nó do tipo quatro
void fragmentarNoTipoQuatro(NoAN i){
      i.cor = true;
      i.esq.cor = false;
      i.dir.cor = false;
}
```

Efeito Colateral da Inversão de Cores

- Acontece quando o pai do nó(b) é preto



E agora José?

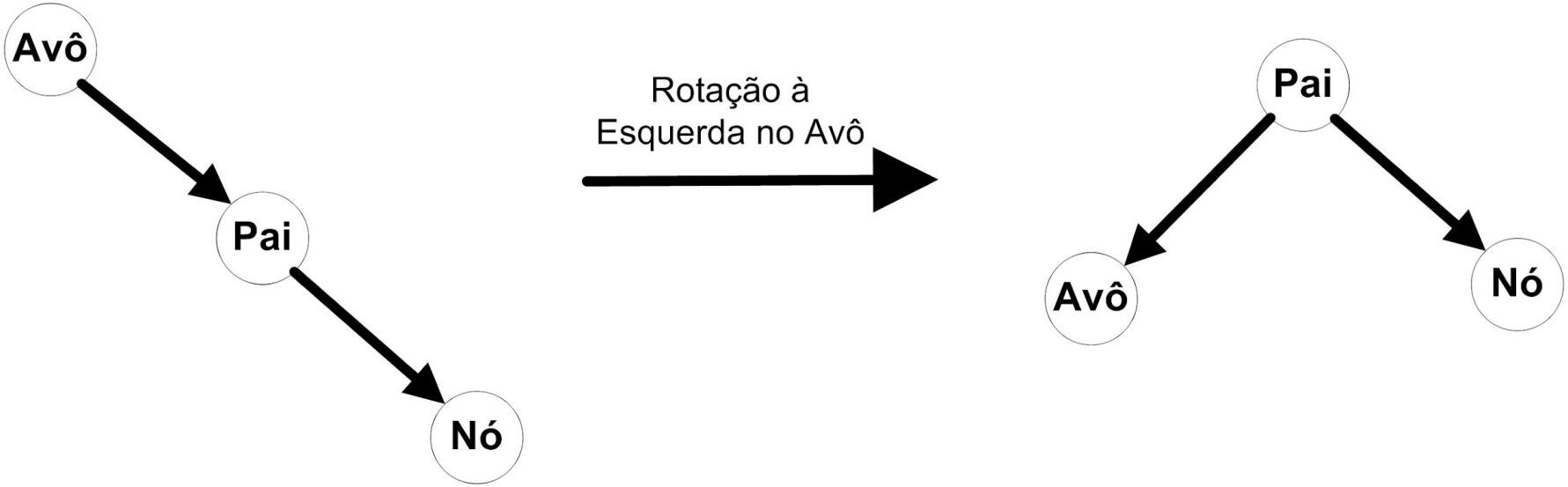
Balanceamento

- A cor preta indica uma tendência de inserção e, consequentemente, dois nós pretos consecutivos indica necessidade de balanceamento

- Após a inversão de cores ou a inserção em uma folha, se o pai do nó também for preto, rotacionamos seu avô considerando o **alinhamento entre avô, pai e nó**

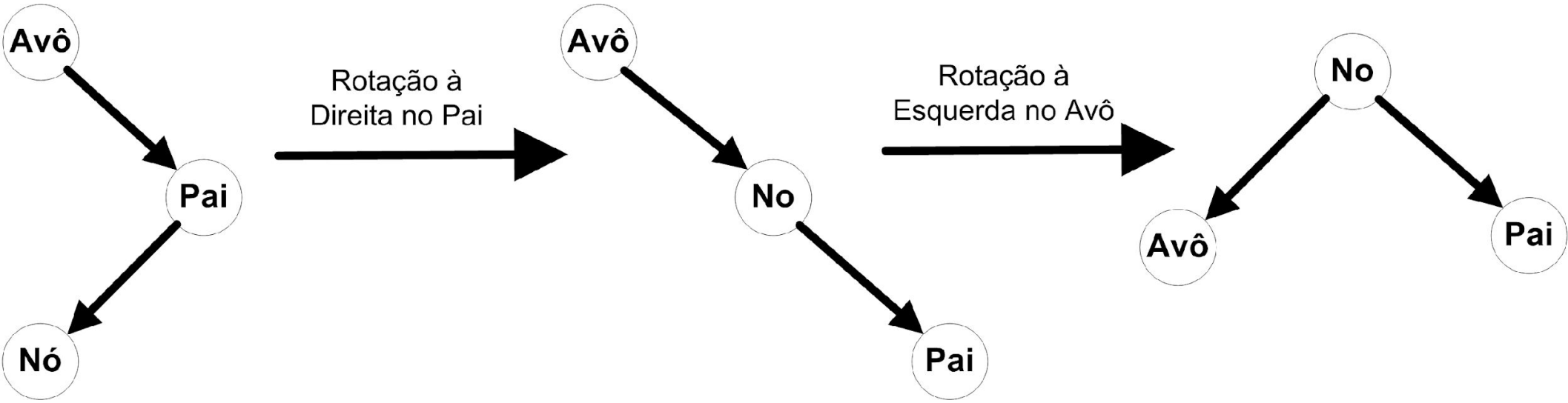
Rotação Simples à Esquerda

- Acontece quando (avô < pai) ***and*** (pai < nó)



Rotação Dupla Direita/Esquerda

- Acontece quando (avô < pai) ***and*** (pai > nó)

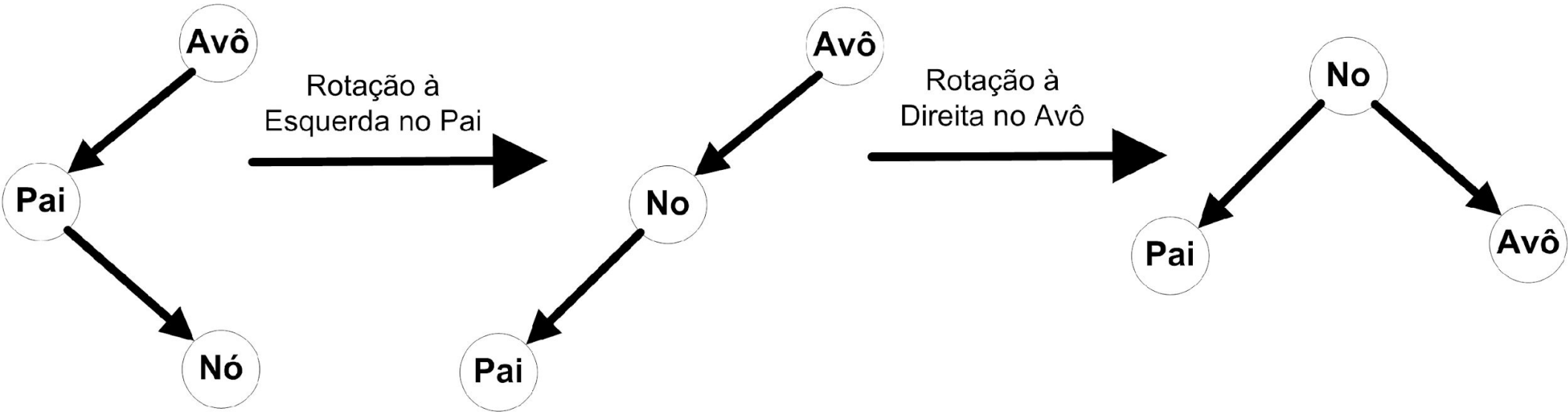


Rotação Simples à Direita

- Acontece quando (avô > pai) ***and*** (pai > nó)

Rotação Dupla Esquerda/Direita

- Acontece quando (avô > pai) ***and*** (pai < nó)



Ideia Básica da Inserção

- Consiste em procurar a folha em que o novo elemento será inserido

  - Se um nó tiver dois filhos pretos (4-nó na 2.3.4), invertemos as cores desse nó e de seus filhos (exceto a raiz que continuará branca porque ela não tem pai para ser gêmeo)

  - Após a inversão de cores, se o pai também for preto, rotacionamos o avô considerando o alinhamento entre avô, pai e nó

  - Após a rotação, o elemento central fica branco e seus novos filhos, pretos

Ideia Básica da Inserção

- Continuar a descida até chegar em uma folha

- A inserção sempre acontece em uma folha que ficará preta porque o novo elemento é gêmeo (na 2.3.4) do pai e do irmão (se esse existir)

- Após a inserção da folha, se o pai for da cor preta, rotacionamos o avô

Exemplo de Inserção

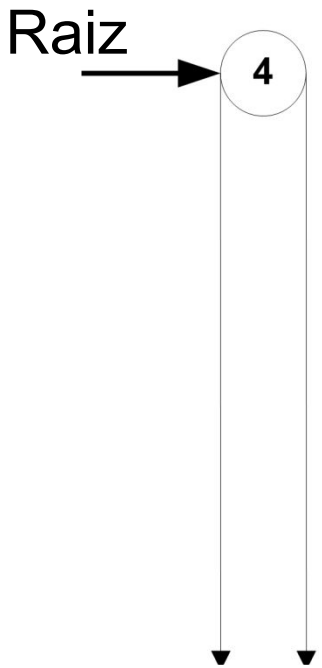
- Crie uma árvore alvinegra através de inserções sucessivas do 4, 35, 10, 13, 3, 30, 15, 12, 7, 40 e 20 respectivamente

Exemplo de Inserção

- Inserindo o 4

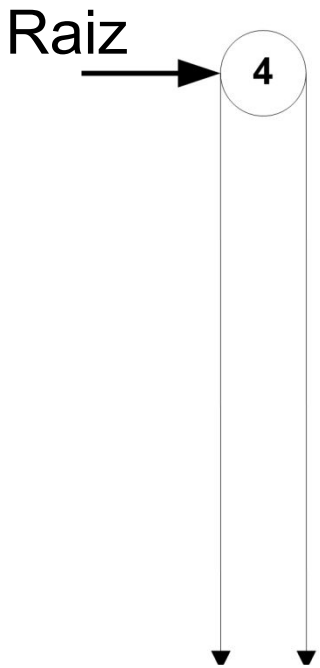
Exemplo de Inserção

- Inserido 4 que será branco pois é o da ``raiz''



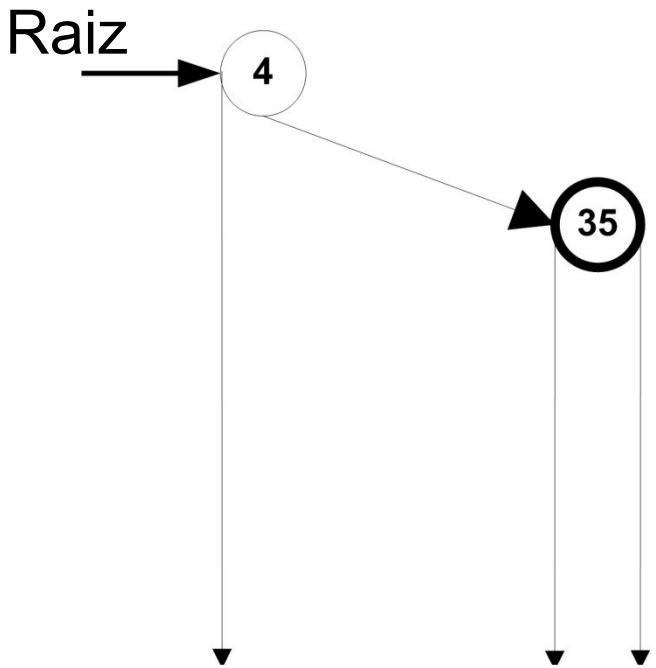
Exemplo de Inserção

- Inserindo o 35



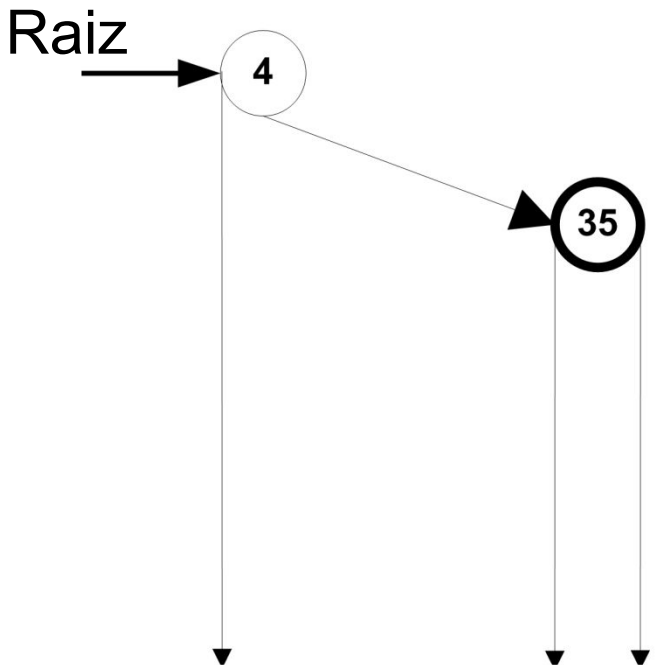
Exemplo de Inserção

- Inserido o 35 (que será preto como todas as folhas)



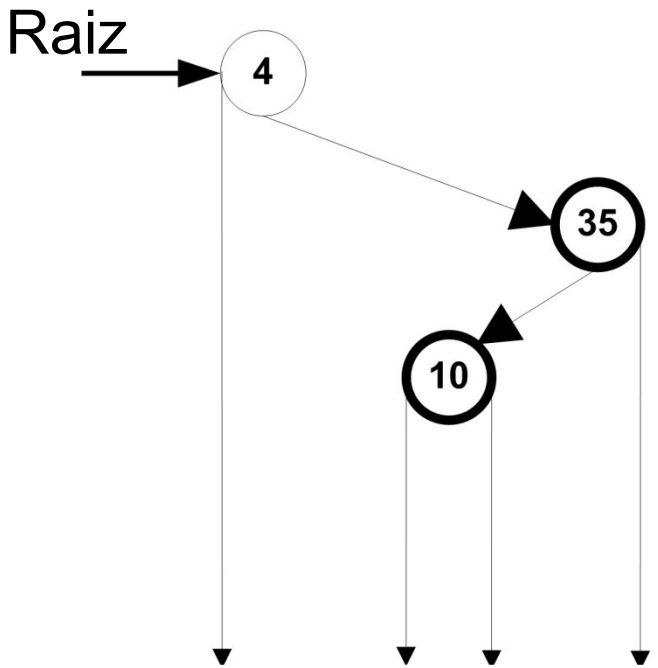
Exemplo de Inserção

- Inserindo o 10



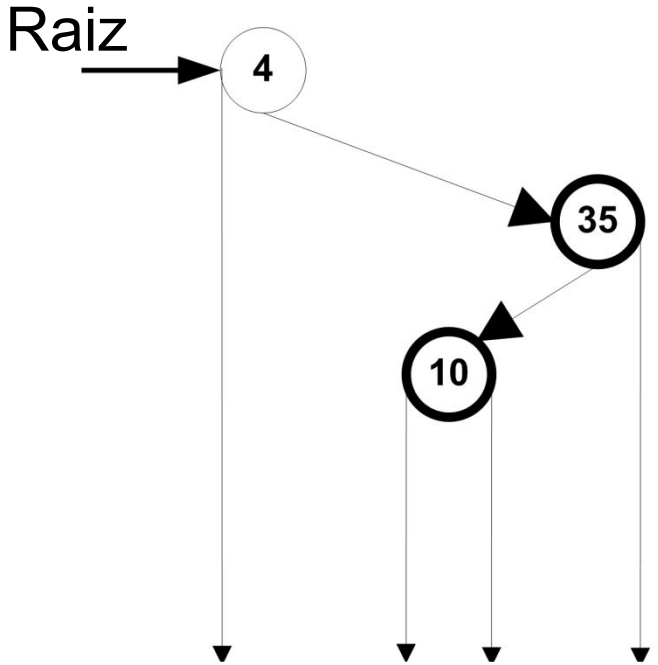
Exemplo de Inserção

- Inserido o 10



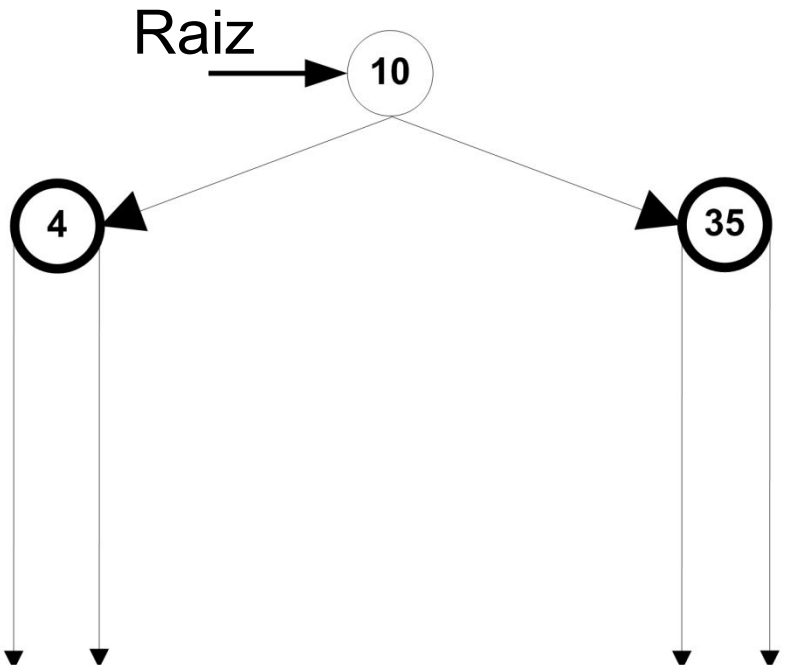
Exemplo de Inserção

- Após a inserção do 10, temos a árvore abaixo. Em seguida, como o pai do 10 tem a cor preta, rotacionamos o avô do 10 - DirEsq(4)



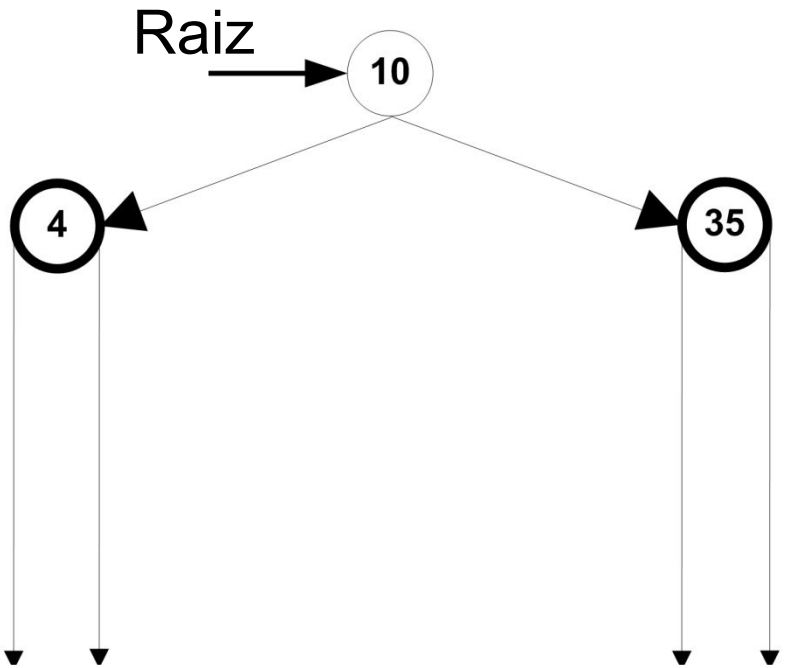
Exemplo de Inserção

- Efetuada a rotação DirEsq no 4, o elemento central fica branco e os filhos, pretos



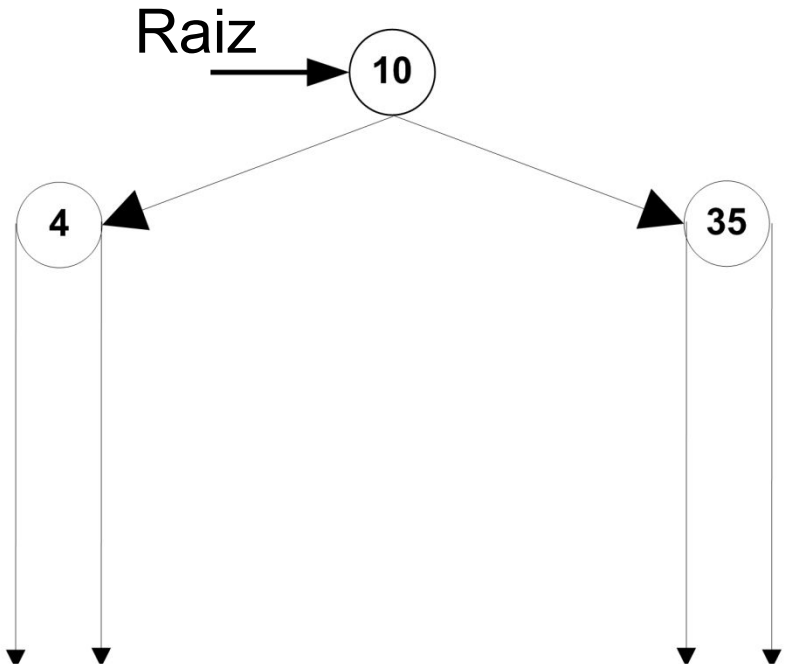
Exemplo de Inserção

- Inserindo o 13, verificamos se o 10 é do tipo 4-nó



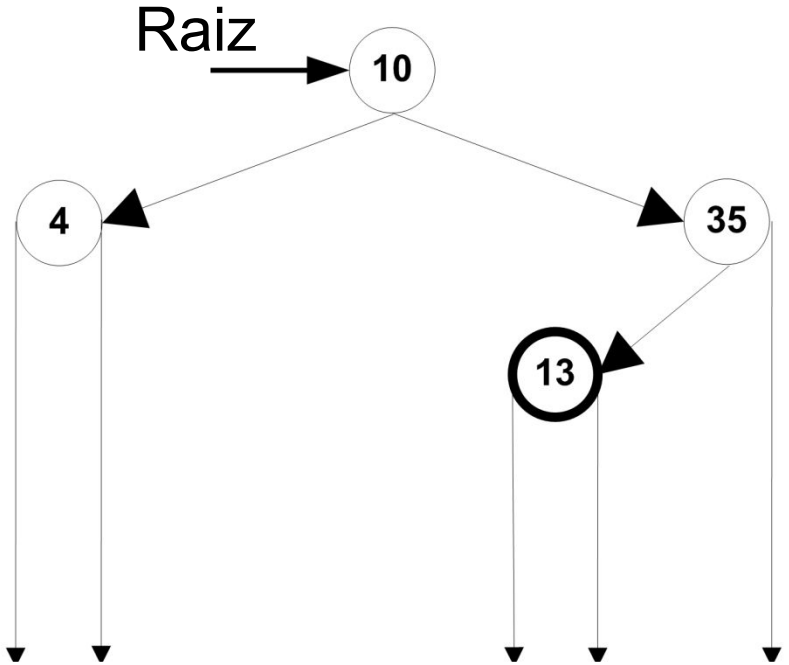
Exemplo de Inserção

- Como o 10 é do tipo 4-nó, invertemos as cores do 4, 10 e 35 (10 continua branco porque é raiz)



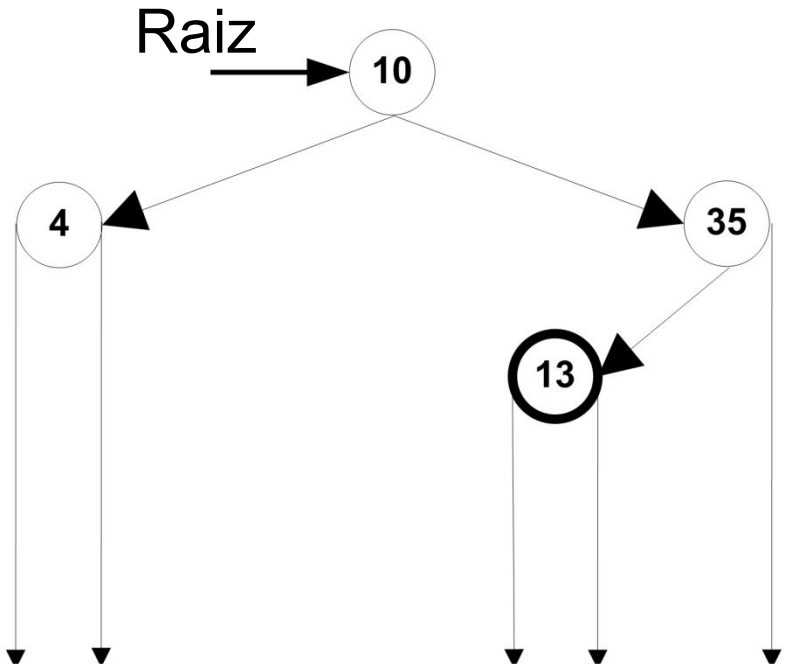
Exemplo de Inserção

- Após a inserção do 13, temos a árvore abaixo. Em seguida, como o pai do 13 tem a cor branca, continuamos



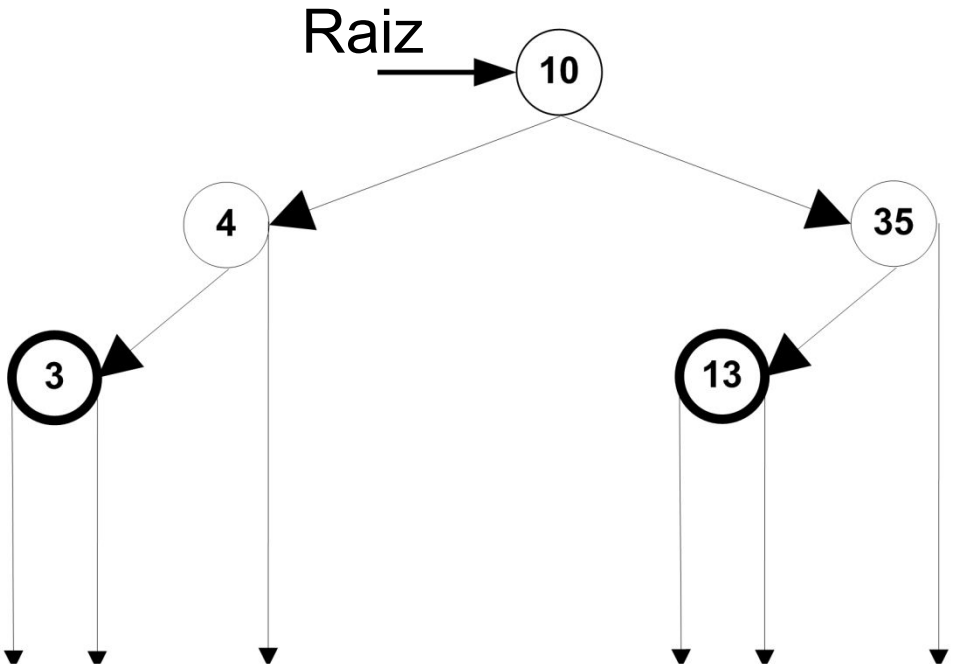
Exemplo de Inserção

- Para inserir o 3, verificamos se o 10 e o 4 são do tipo 4-nó



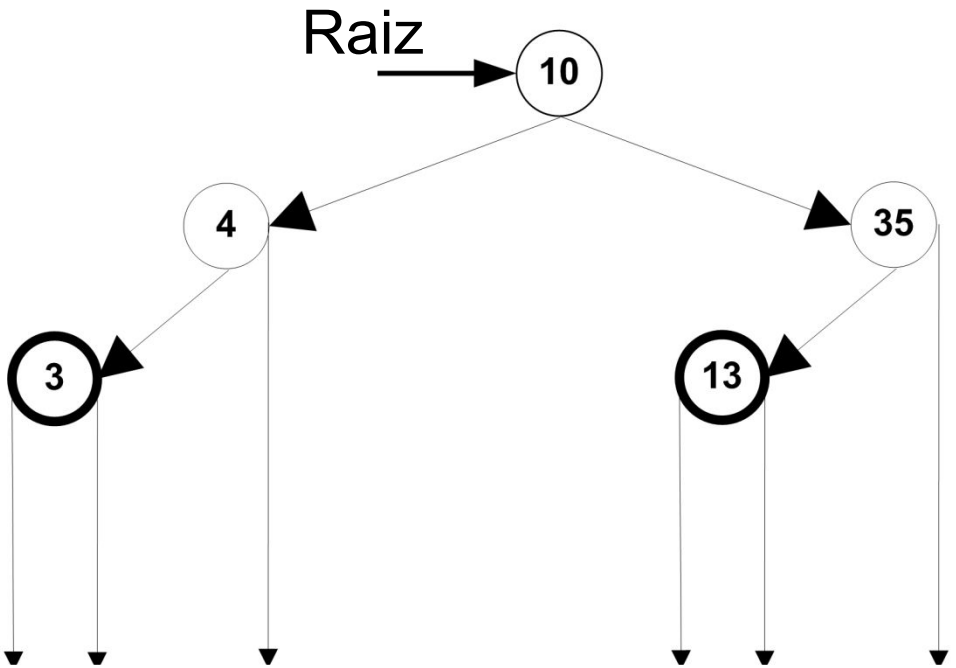
Exemplo de Inserção

- Após a inserção do 3, temos a árvore abaixo. Em seguida, como o pai do 3 tem a cor branca, continuamos



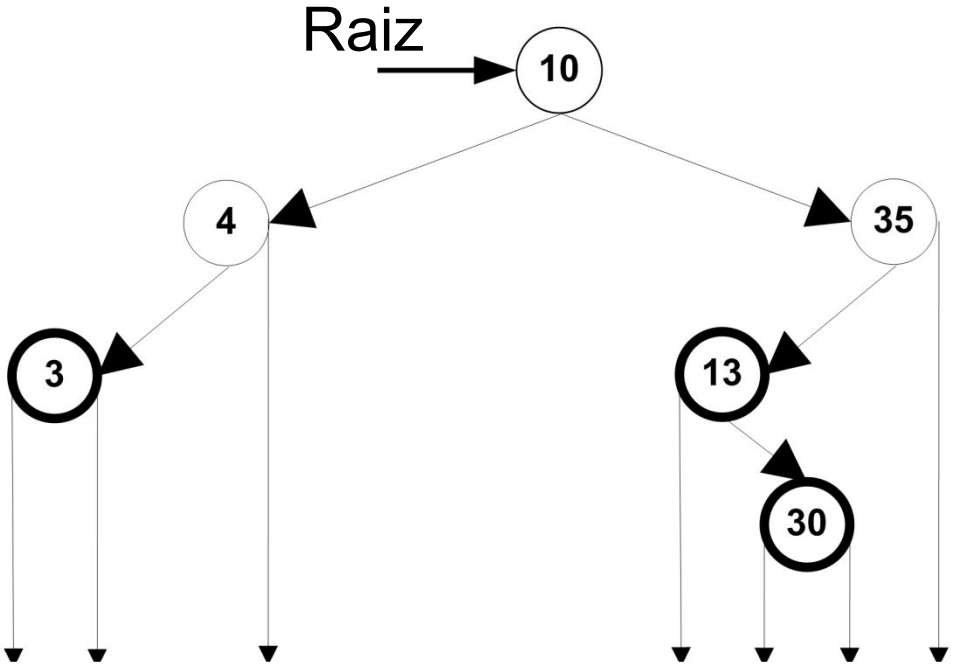
Exemplo de Inserção

- Para inserir o 30, verificamos se o 10, 35 e 13 são do tipo 4 nó



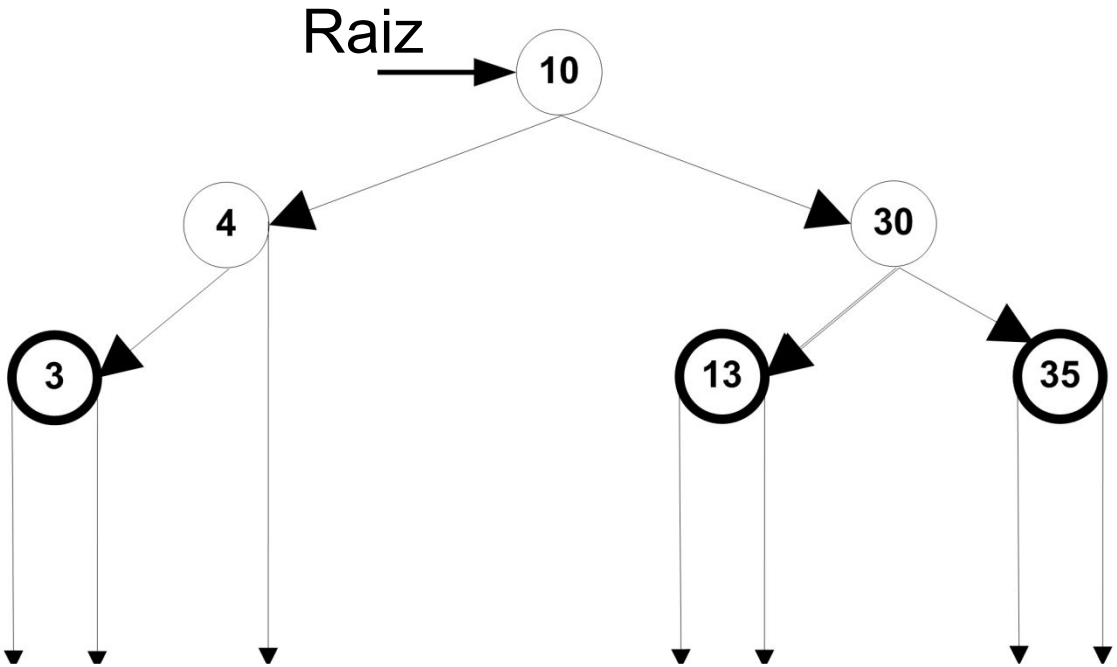
Exemplo de Inserção

- Após a inserção do 30, temos a árvore abaixo. Em seguida, como o pai do 30 tem a cor preta, rotacionamos o avô do 30 - EsqDir(35)



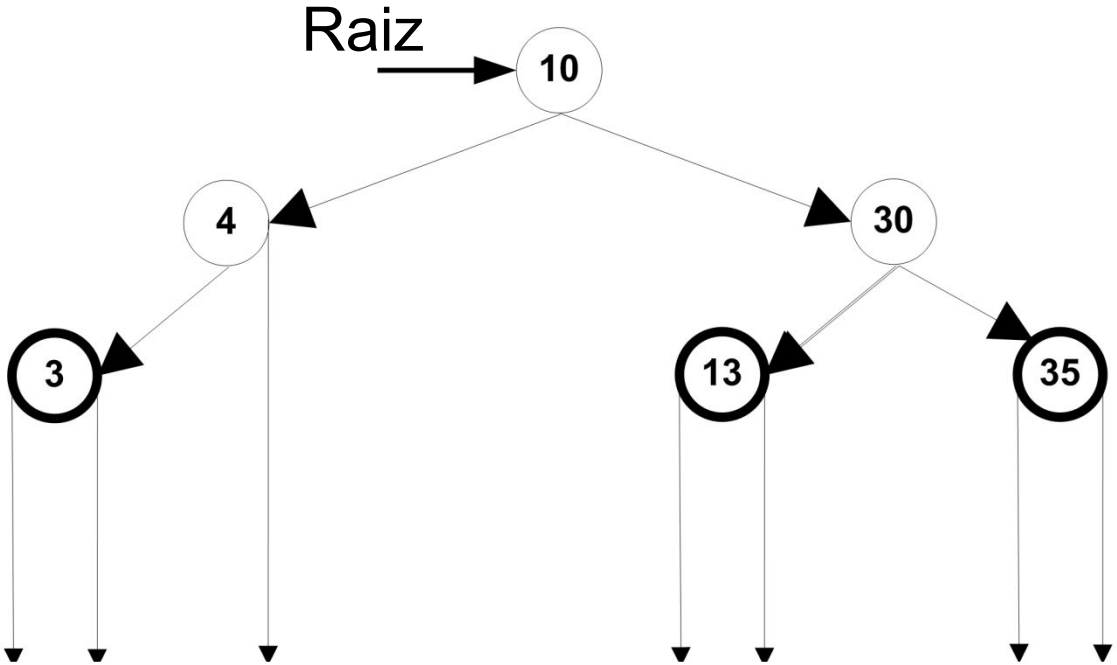
Exemplo de Inserção

- Após a rotação envolvendo (13, 30 e 35), o elemento central (30) será branco e seus filhos (13 e 35), pretos



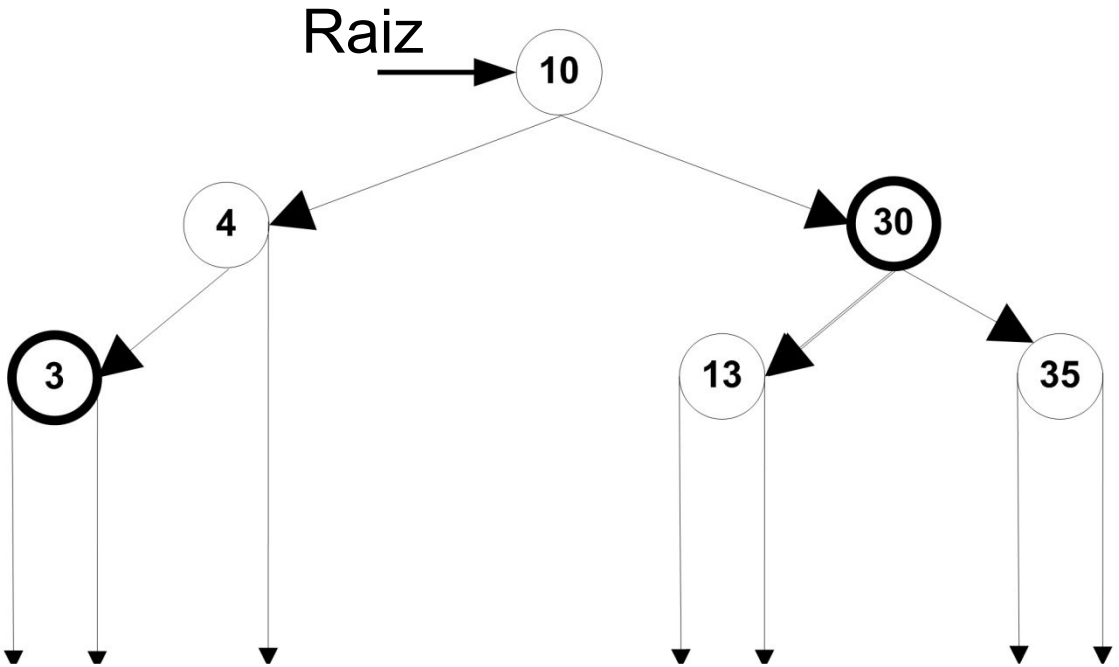
Exemplo de Inserção

- Para inserir o 15, verificamos se o 10 e o 30 são do tipo 4 nó o que acontece com o nó 30



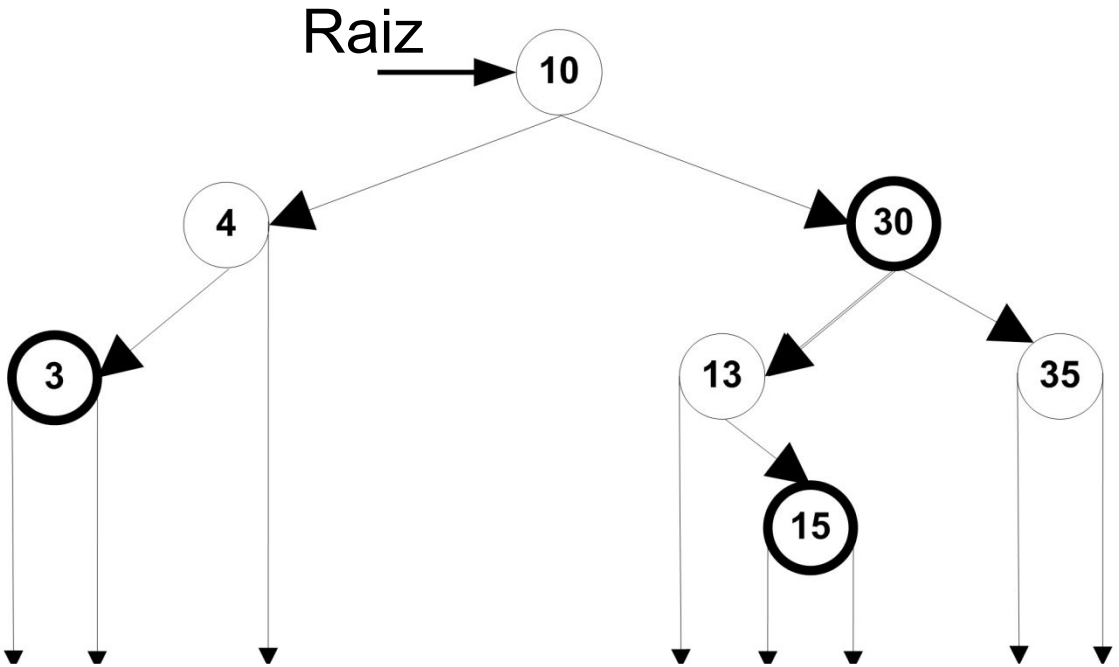
Exemplo de Inserção

- Assim, invertemos a cor do 30, 13 e 35. Em seguida, verificamos se o pai do 30 também tem a cor preta



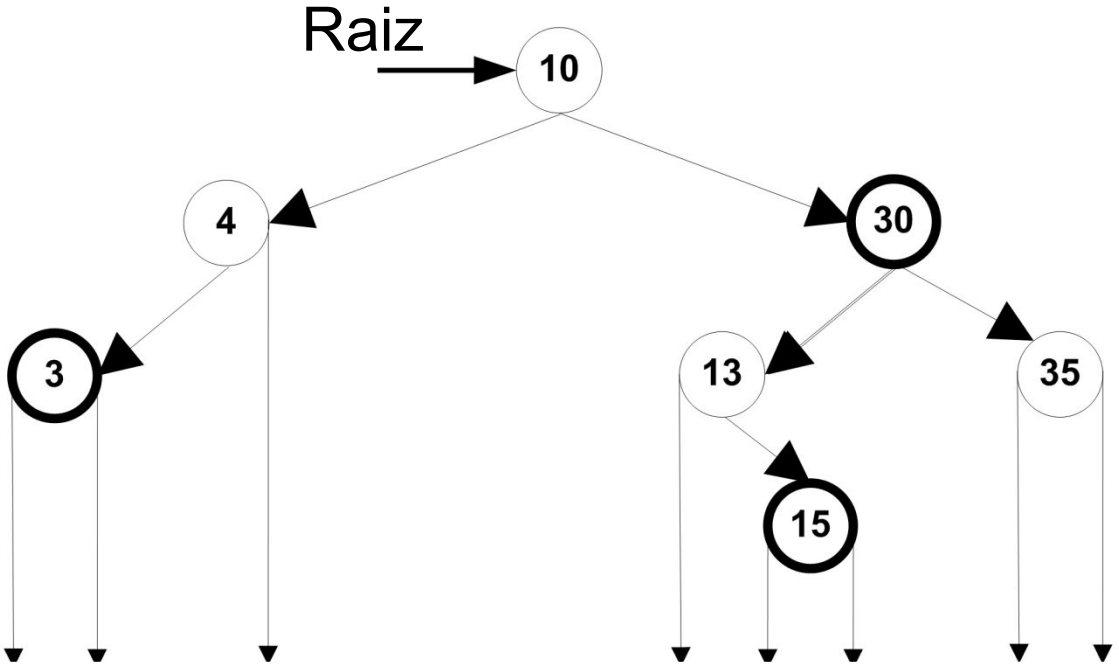
Exercício Resolvido (8)

- Após a inserção do 15, temos a árvore abaixo. Após a inserção do 15, nosso algoritmo verifica se ??????????????????????



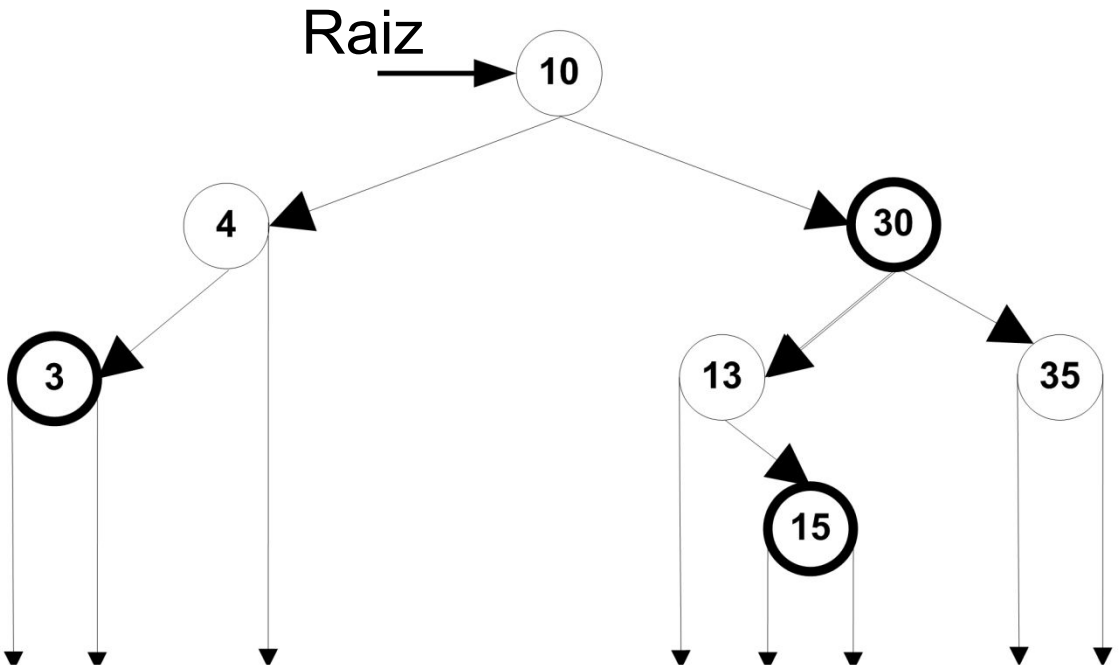
Exercício Resolvido (8)

- Após a inserção do 15, temos a árvore abaixo. Após a inserção do 15, nosso algoritmo verifica se o pai do 15 tem a cor preta



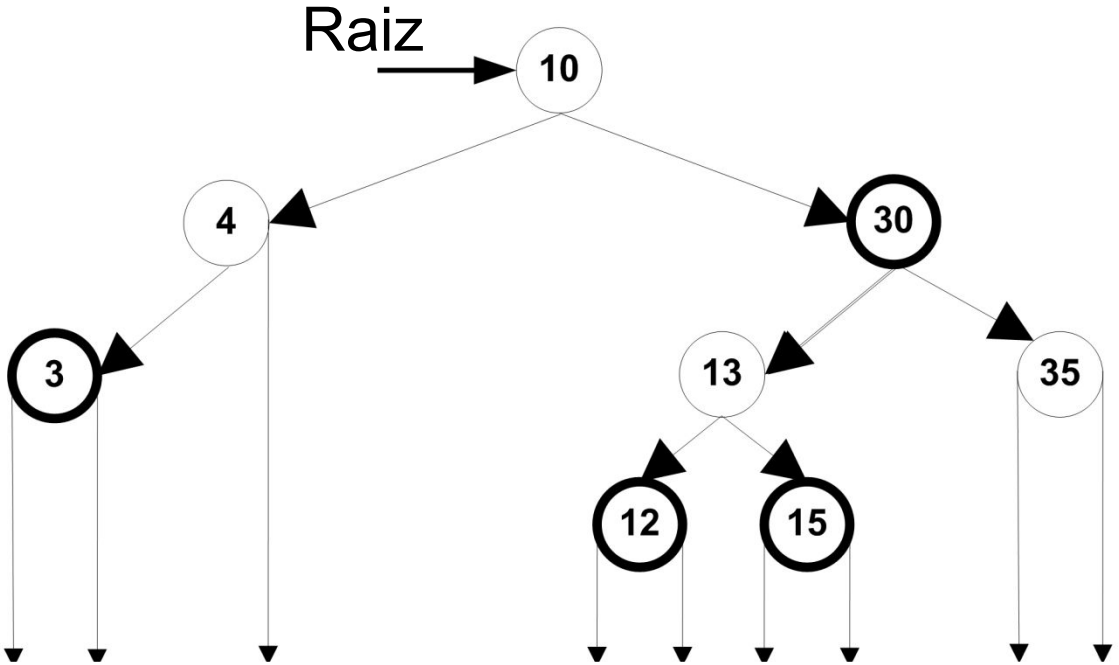
Exemplo de Inserção

- Para inserir o 12, verificamos se o 10, 30 e 13 são do tipo 4-nó



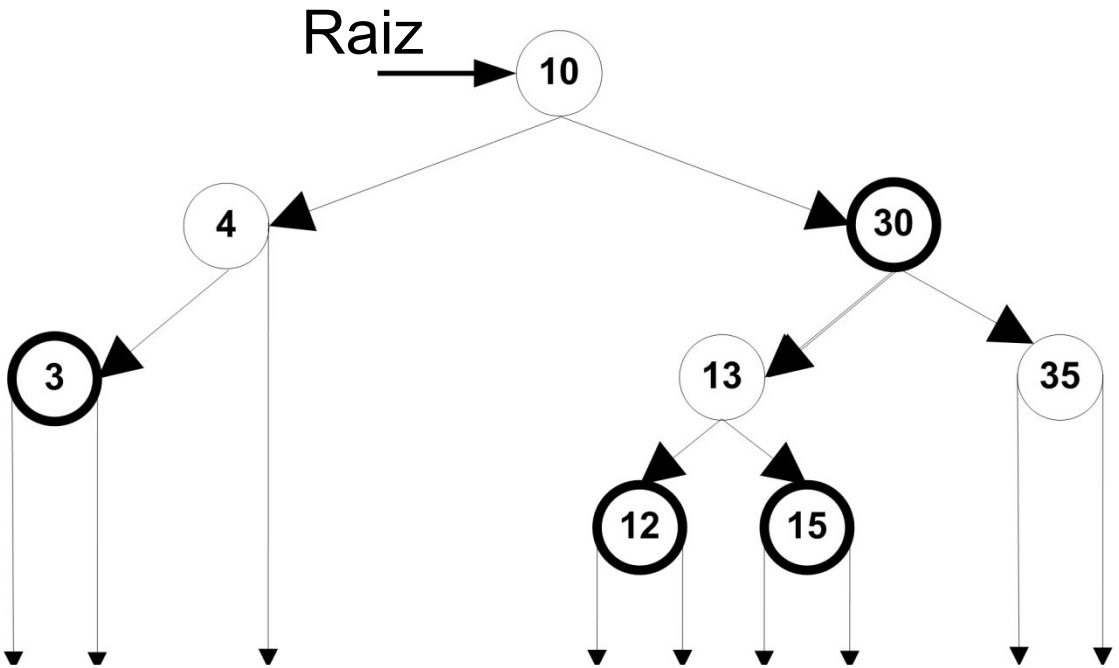
Exemplo de Inserção

- Após a inserção do 12, temos a árvore abaixo. Após a inserção do 12, nosso algoritmo verifica se o pai do 12 tem a cor preta



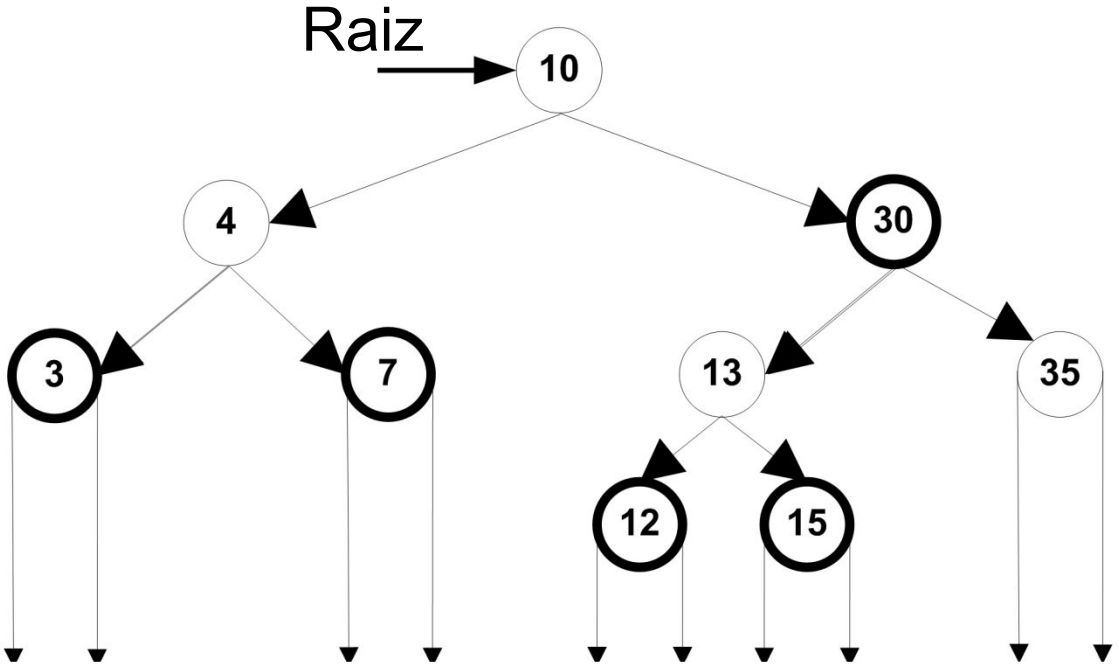
Exemplo de Inserção

- Para inserir o 7, verificamos se os nós 10 e 4 são do tipo 4-nó



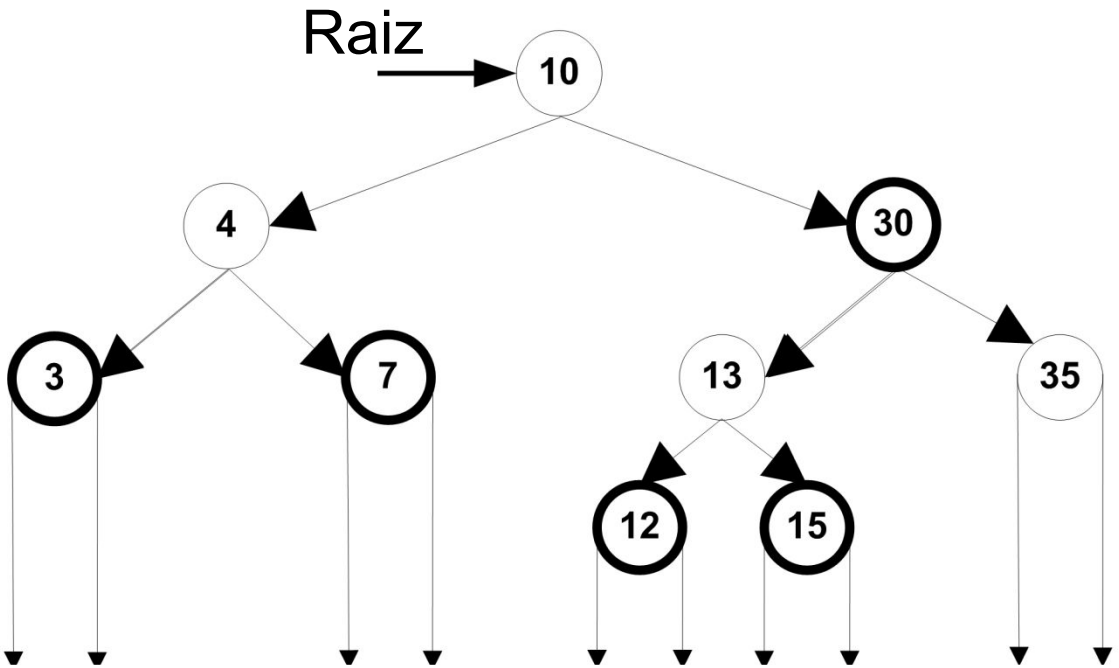
Exemplo de Inserção

- Após a inserção do 7, temos a árvore abaixo. Após a inserção do 7, nosso algoritmo verifica se o pai do 7 tem a cor preta



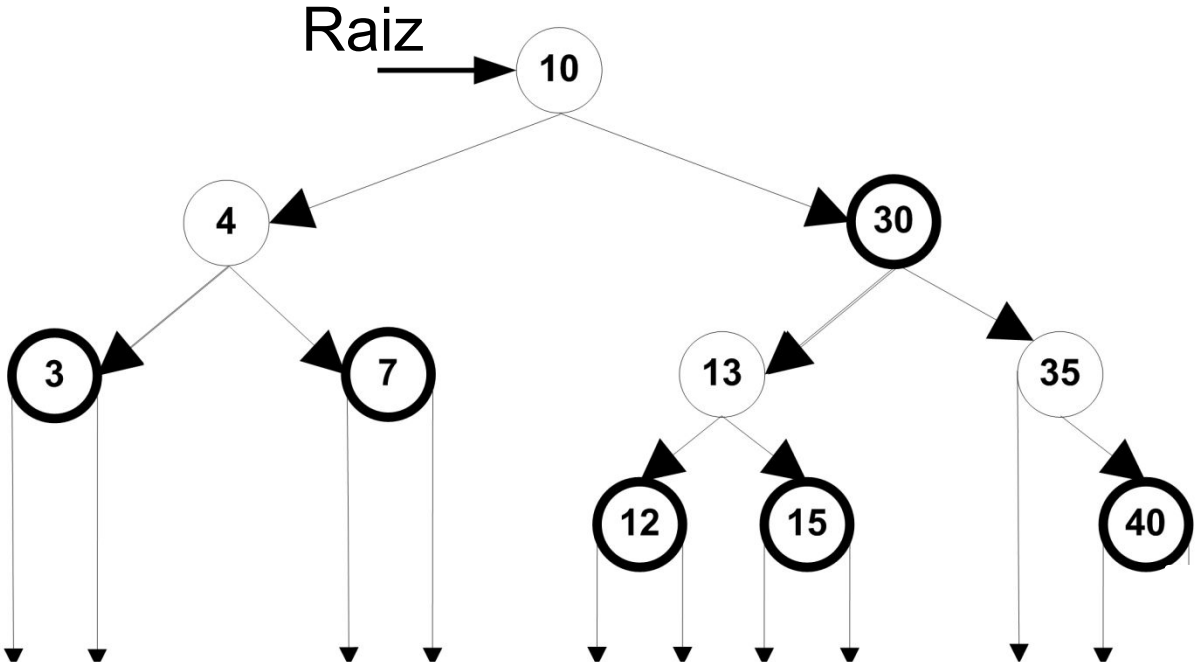
Exemplo de Inserção

- Para inserir o 40, verificamos se os nós 10, 30 e 35 são do tipo 4-nó



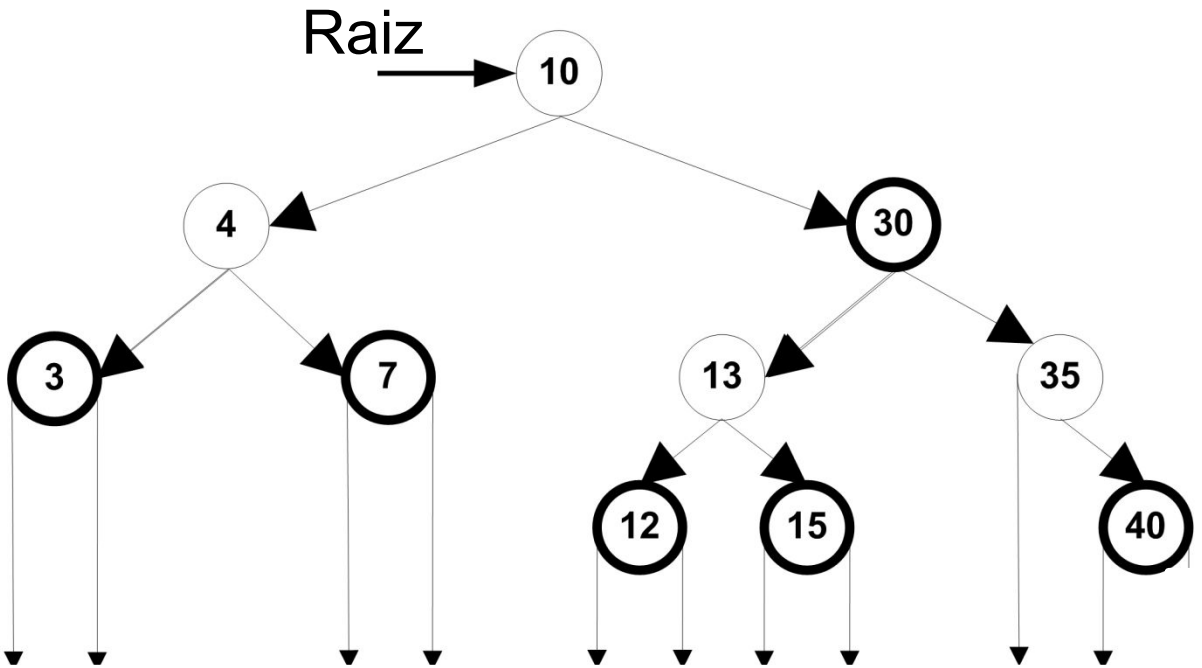
Exemplo de Inserção

- Após a inserção do 40, temos a árvore abaixo. Após a inserção do 40, nosso algoritmo verifica se o pai do 40 tem a cor preta



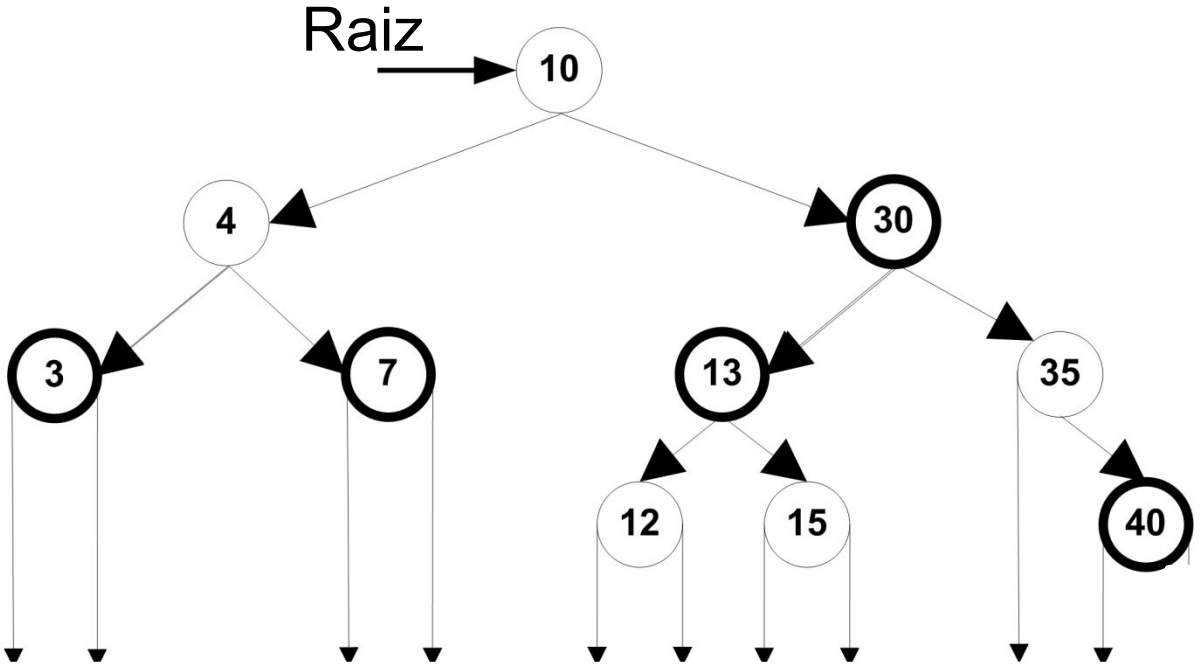
Exemplo de Inserção

- Inserindo o 20, verificamos se os nós 10, 30 e 13 são do tipo 4-nó e isso acontece com o nó 13



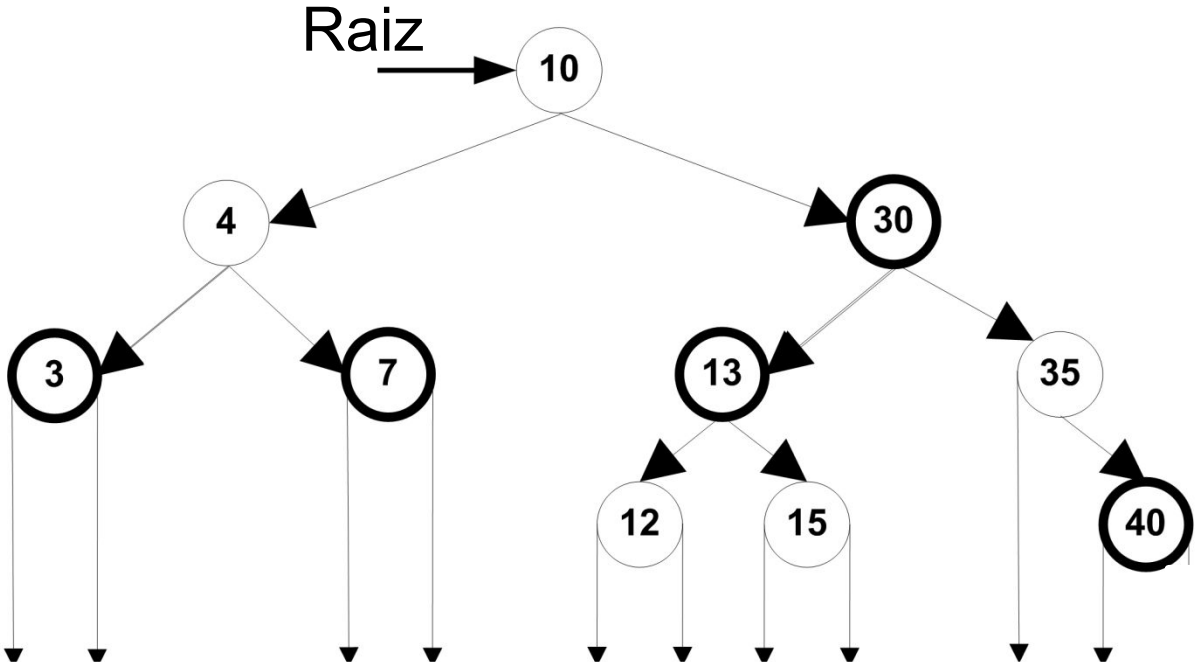
Exemplo de Inserção

- Invertemos as cores dos nós 13 e de seus filhos (12 e 15) e, consequentemente, ficamos como dois nós pretos consecutivos



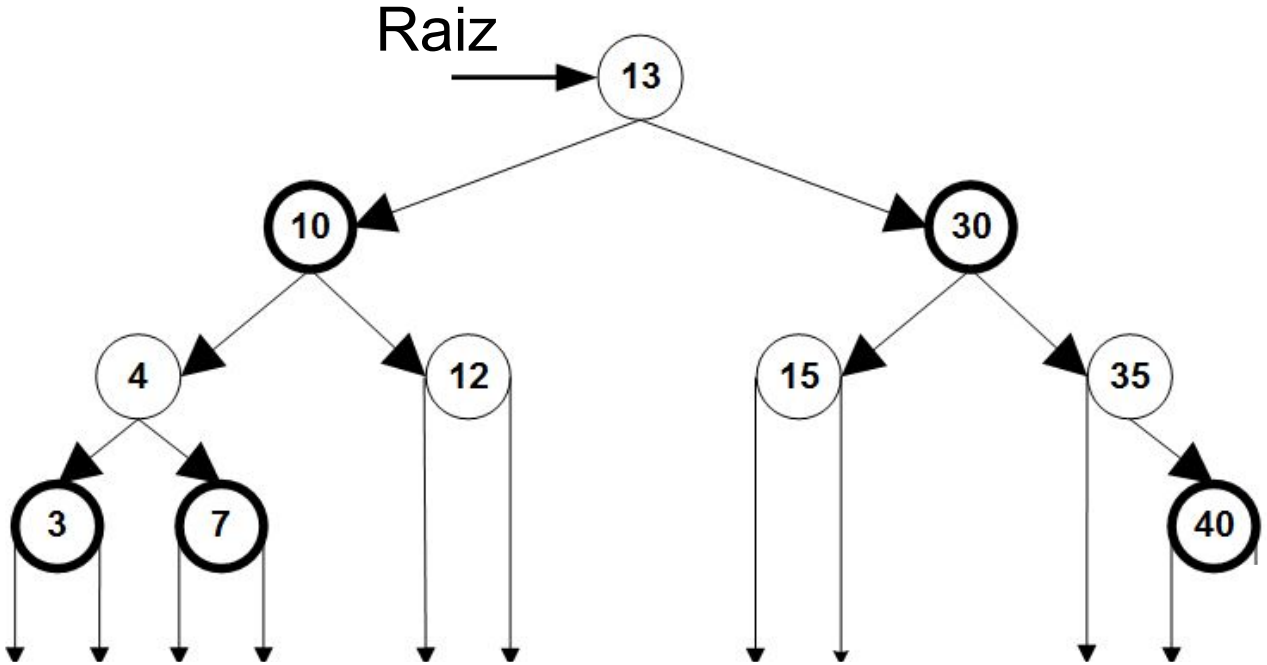
Exemplo de Inserção

- Por isso, dado o alinhamento dos nós 10, 30 e 13, rotacionamos o avô do 13 fazendo uma Dir(30)Esq(10)



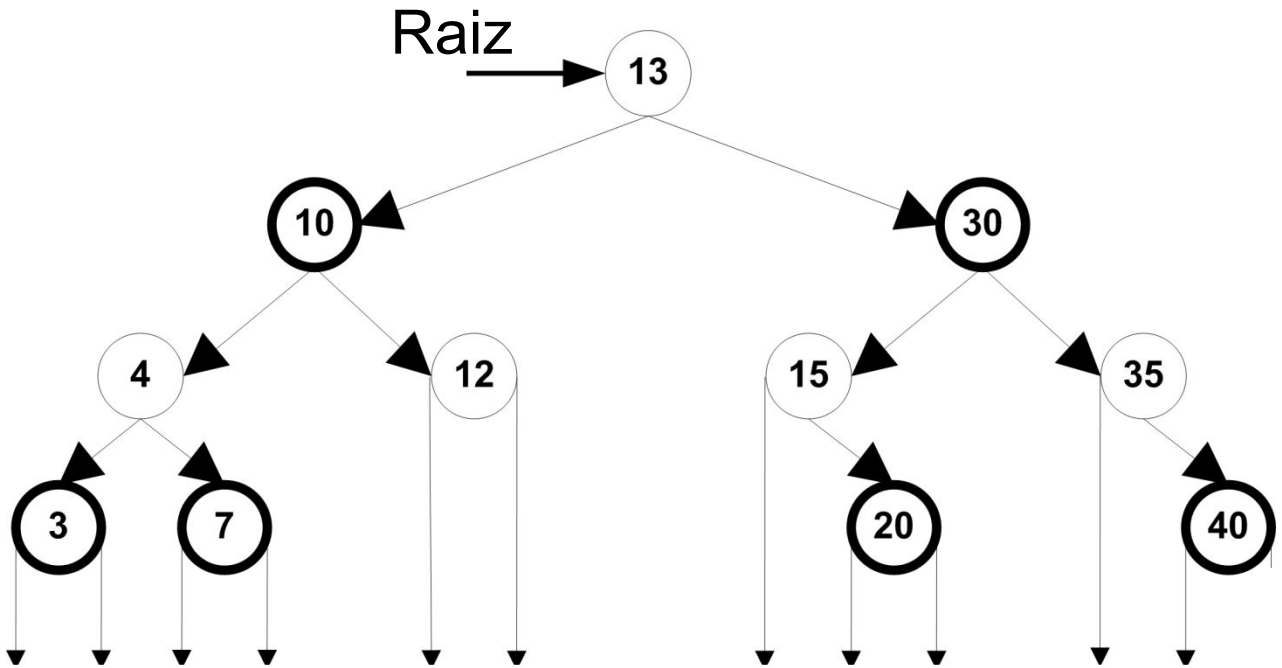
Exemplo de Inserção

- Por isso, dado o alinhamento dos nós 10, 30 e 13, rotacionamos o avô do 13 fazendo uma Dir(30)Esq(10)



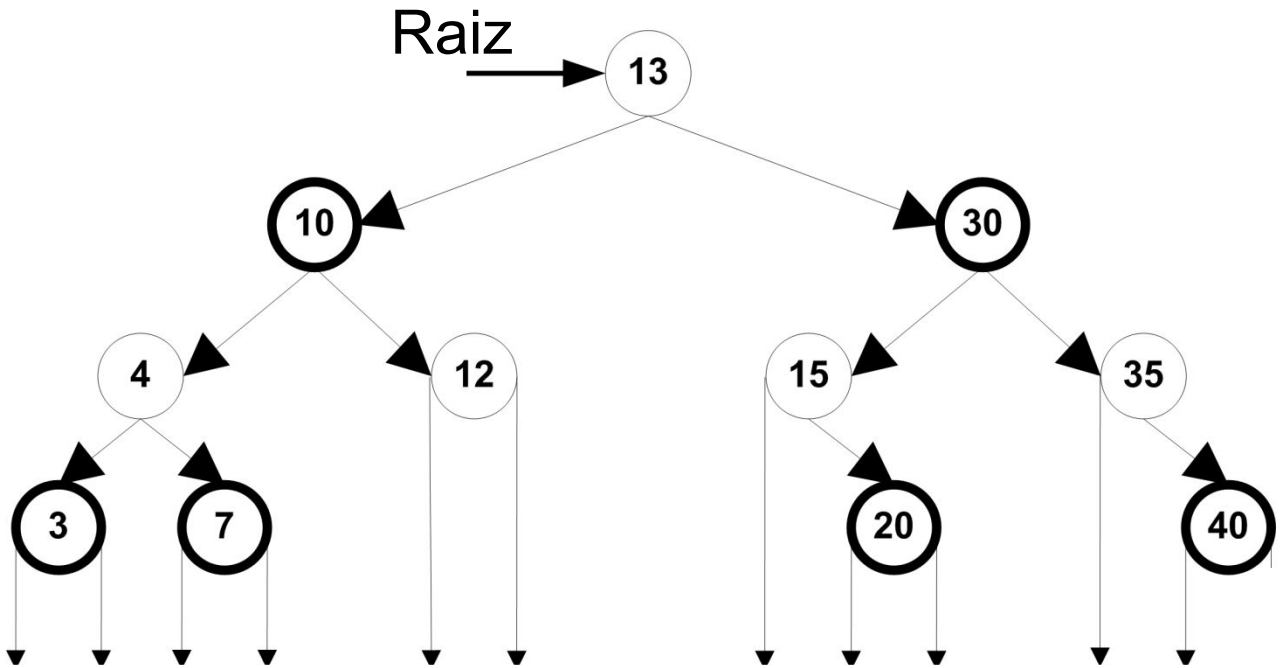
Exemplo de Inserção

- Finalmente, inserimos o 20



Exercício (1)

- Na árvore abaixo, insira o 6



Exercício (2)

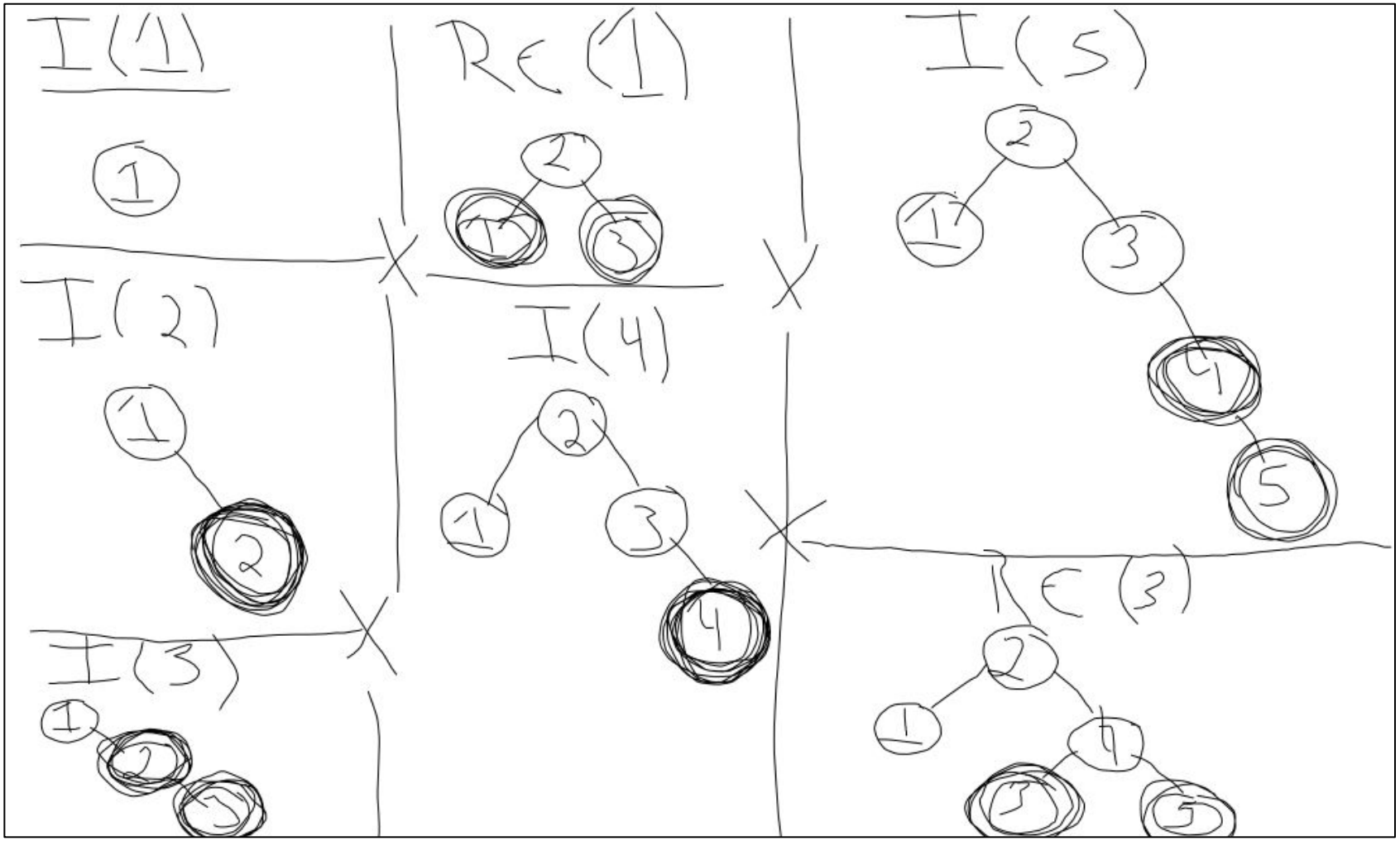
- Crie uma árvore alvinegra através de inserções sucessivas dos números 1 a 20, respectivamente

- Crie uma árvore alvinegra através de inserções sucessivas dos números 20 a 1, respectivamente

- Para cada um dos dois exercícios anteriores, verifique sua resposta usando nosso código para a árvore alvinegra

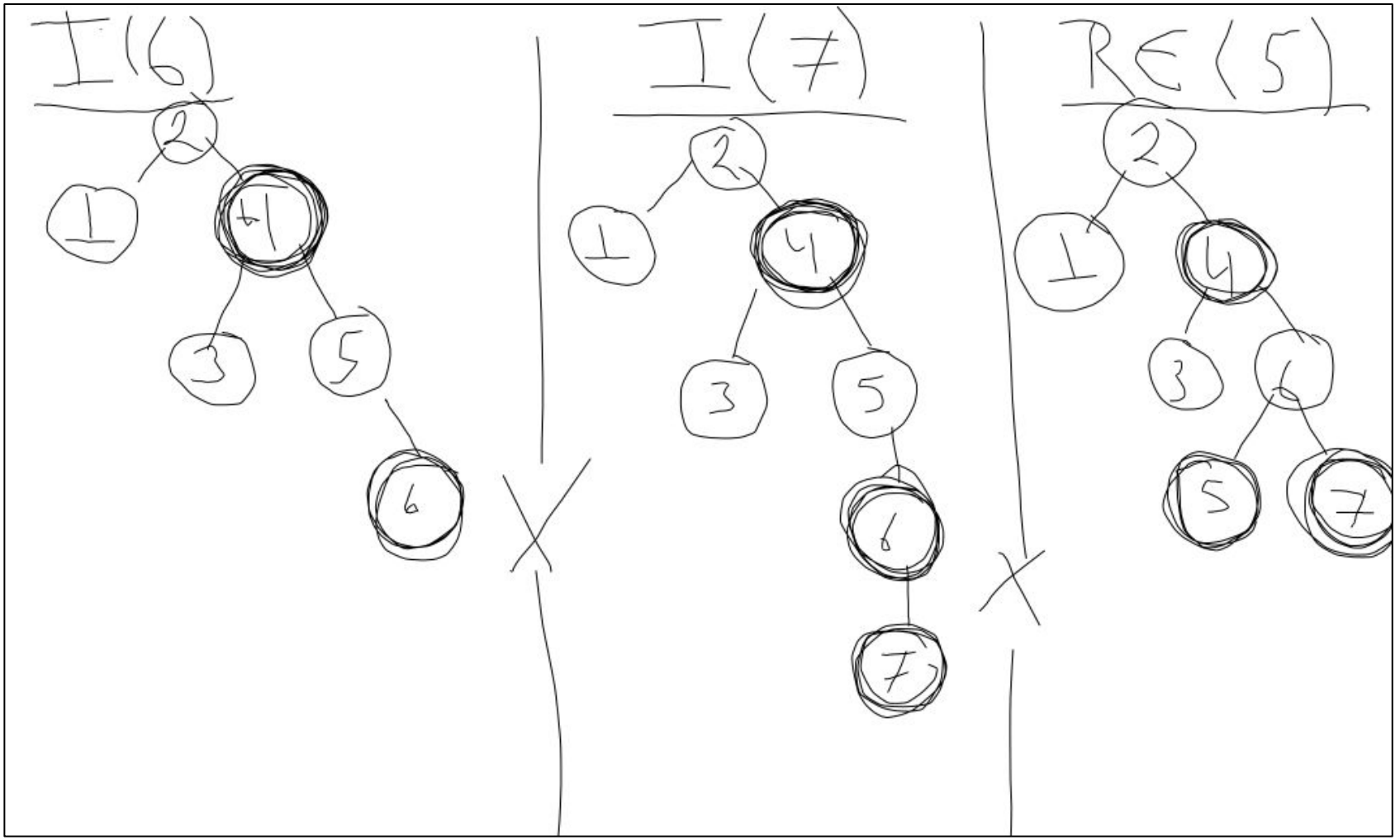
Exercício (2)

- Crie uma árvore alvinegra através de inserções sucessivas dos números 1 a 20, respectivamente



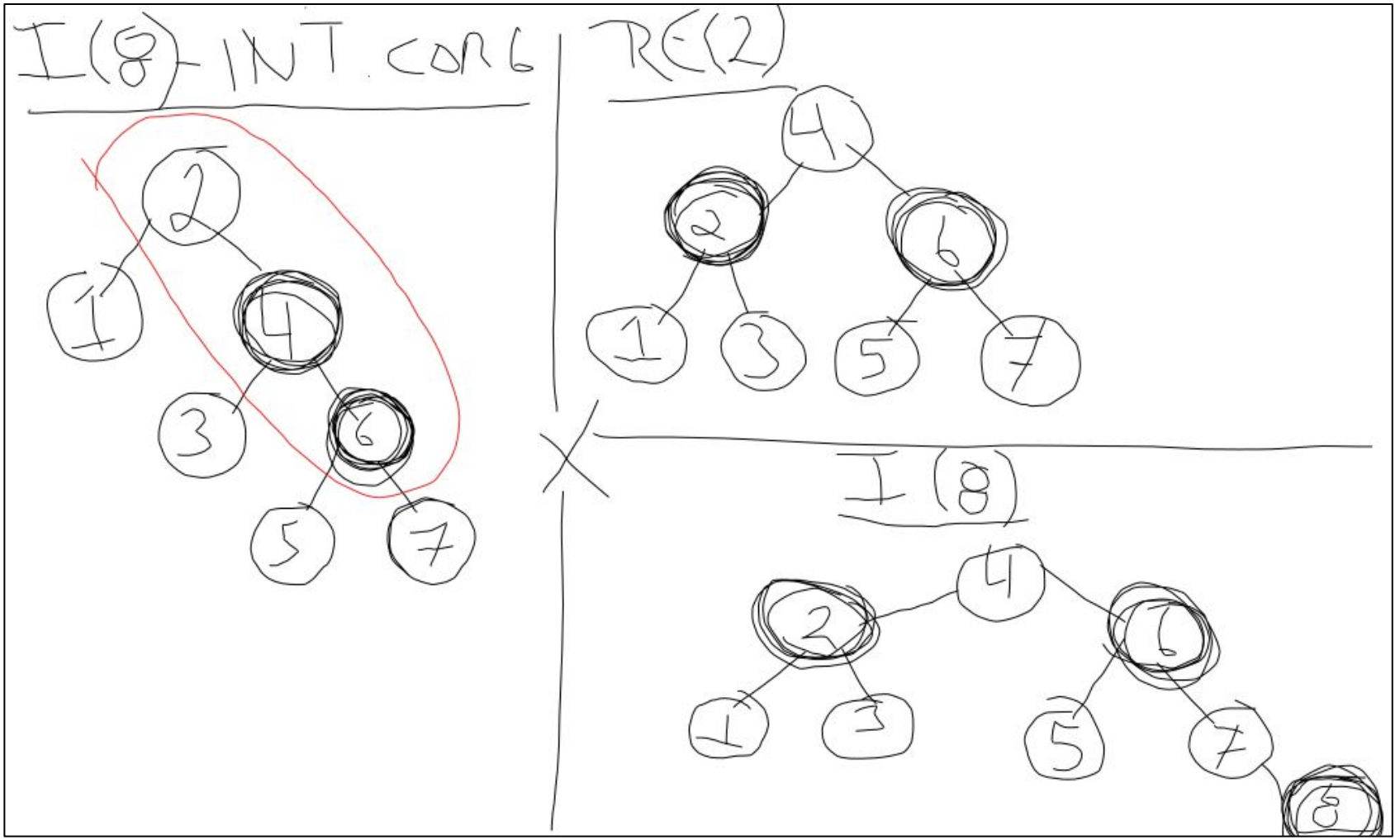
Exercício (2)

- Crie uma árvore alvinegra através de inserções sucessivas dos números 1 a 20, respectivamente



Exercício (2)

- Crie uma árvore alvinegra através de inserções sucessivas dos números 1 a 20, respectivamente

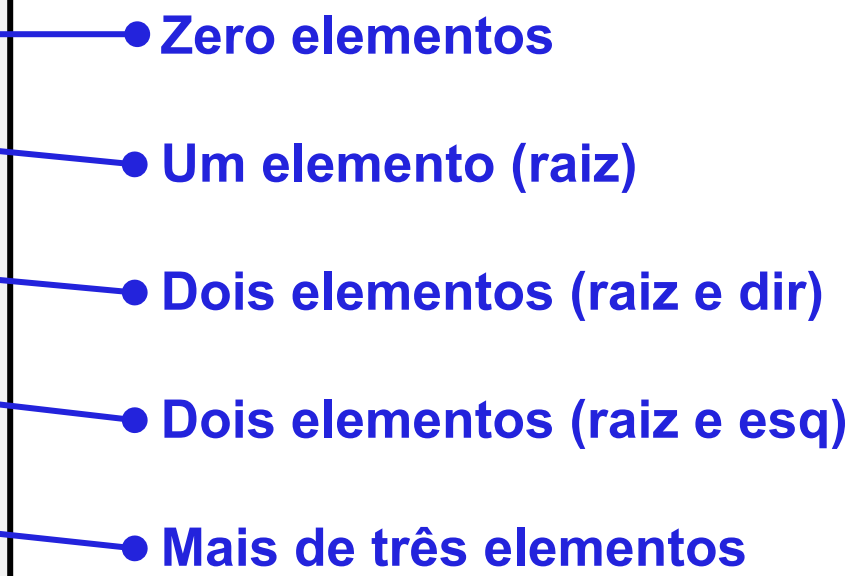


Algoritmo em C-*like*

- A técnica de inserção apresentada será aplicada quando tivermos mais do que três elementos, pois ela depende de teste no pai/avô do novo elemento

- Inserimos “manualmente” os três primeiros elementos

```
void inserir(int elemento) {
  if (raiz == null){
    ...
  } else if (raiz.esq == null && raiz.dir == null){
    ...
  } else if (raiz.esq == null){
    ...
  } else if (raiz.dir == null){
    ...
  } else {
    ...
  }
  raiz.cor = false;
}
```

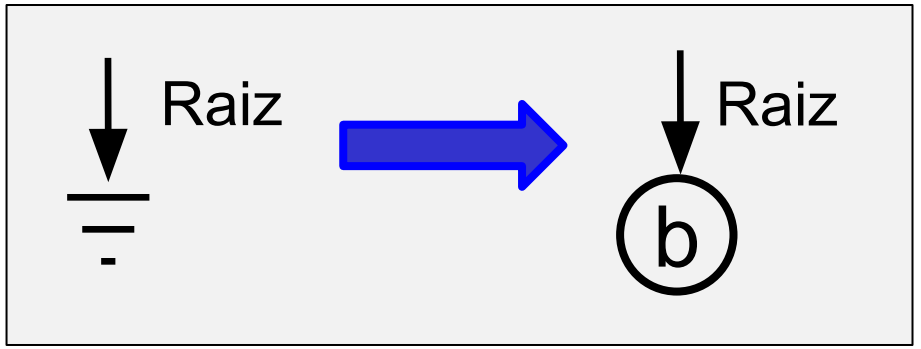


Algoritmo em C-*like*

- Se zero elementos (raiz == null), inserimos o novo elemento na raiz

Zero elementos

```
void inserir(int elemento) {
  if (raiz == null){
    raiz = new NoAN(elemento);
  } else if (raiz.esq == null && raiz.dir == null){
    ...
  } else if (raiz.esq == null){
    ...
  } else if (raiz.dir == null){
    ...
  } else {
    ...
  }
  raiz.cor = false;
}
```

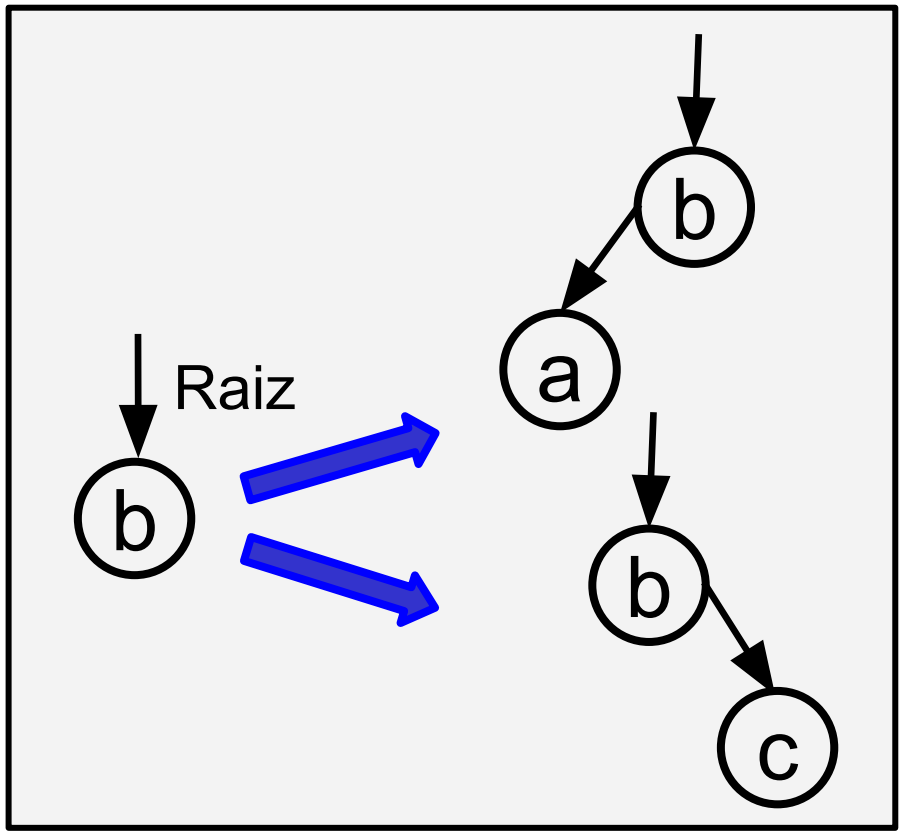


Algoritmo em C-*like*

- Se tivermos um elemento (raiz.esq == null **and** raiz.dir == null), inserimos o novo elemento à esquerda / direita da raiz

```
void inserir(int elemento) {
   if (raiz == null){
      ...
   } else if (raiz.esq == null && raiz.dir == null){
      if (elemento < raiz.elemento){
         raiz.esq = new NoAN(elemento);
      } else {
         raiz.dir = new NoAN(elemento);
      }
   } else if (raiz.esq == null){
      ...
   } else if (raiz.dir == null){
      ...
   } else {
      ...
   }
   raiz.cor = false;
}
```

Um elemento

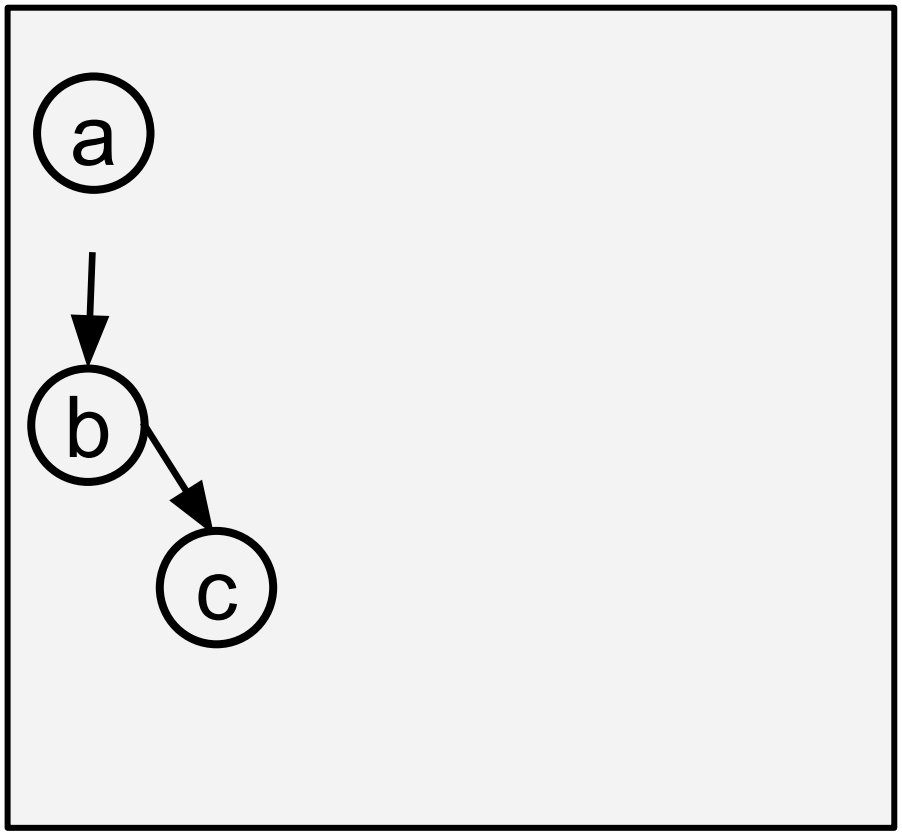


Algoritmo em C-*like*

- Se tivermos dois elementos (raiz e dir), temos três possibilidades de inserção

```
void inserir(int elemento) {
    ...
  } else if (raiz.esq == null){
    if(elemento < raiz.elemento){
      raiz.esq = new NoAN(elemento);
    } else if (elemento < raiz.dir.elemento){
      raiz.esq = new NoAN(raiz.elemento);
      raiz.elemento = elemento;
    } else {
      raiz.esq = new NoAN(raiz.elemento);
      raiz.elemento = raiz.dir.elemento;
      raiz.dir.elemento = elemento;
    }
    raiz.esq.cor = raiz.dir.cor = false;
  }
    ...
  raiz.cor = false;
}
```

Dois elementos (raiz e dir)

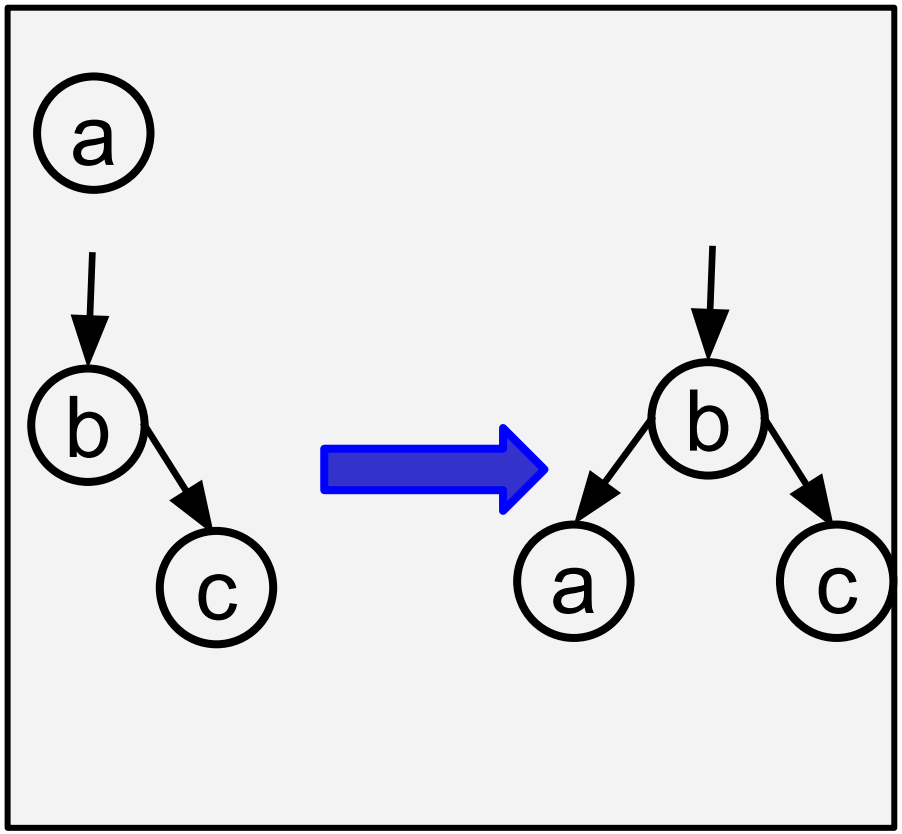


Algoritmo em C-*like*

- Se tivermos dois elementos (raiz e dir), temos três possibilidades de inserção

```
void inserir(int elemento) {
    ...
  } else if (raiz.esq == null){
    if(elemento < raiz.elemento){
      raiz.esq = new NoAN(elemento);
    } else if (elemento < raiz.dir.elemento){
      raiz.esq = new NoAN(raiz.elemento);
      raiz.elemento = elemento;
    } else {
      raiz.esq = new NoAN(raiz.elemento);
      raiz.elemento = raiz.dir.elemento;
      raiz.dir.elemento = elemento;
    }
    raiz.esq.cor = raiz.dir.cor = false;
  }
    ...
  raiz.cor = false;
}
```

Dois elementos (raiz e dir)

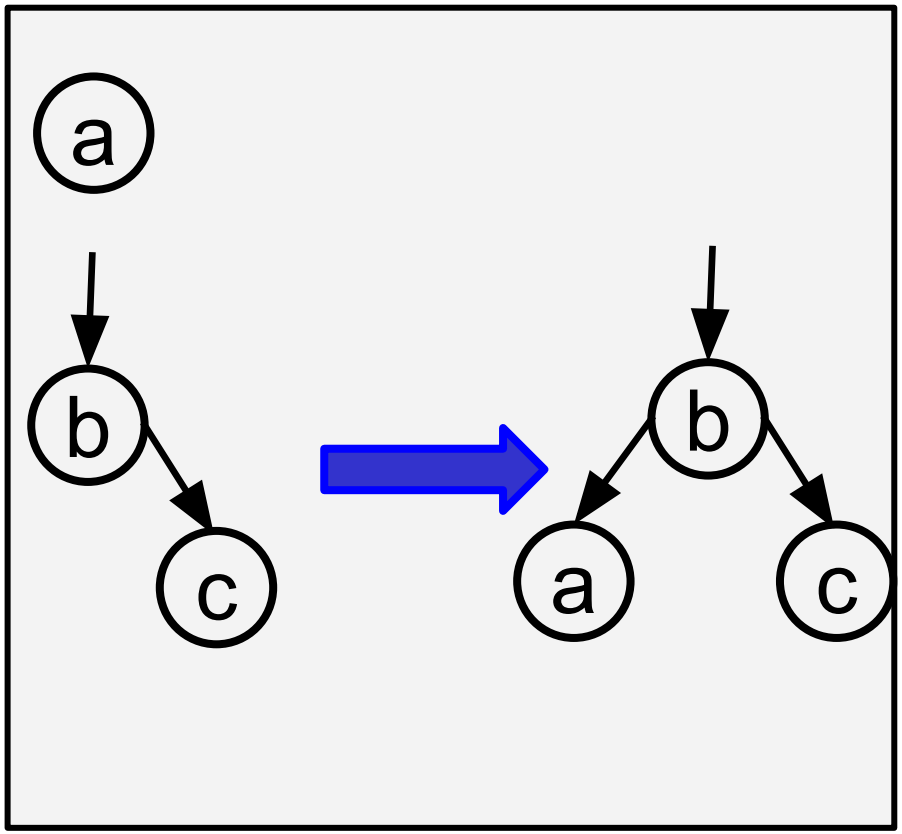


Algoritmo em C-*like*

- Se tivermos dois elementos (raiz e dir), temos três possibilidades de inserção

```
void inserir(int elemento) {
      ...
   } else if (raiz.esq == null){
      if(elemento < raiz.elemento){
         raiz.esq = new NoAN(elemento);
      } else if (elemento < raiz.dir.elemento){
         raiz.esq = new NoAN(raiz.elemento);
         raiz.elemento = elemento;
      } else {
         raiz.esq = new NoAN(raiz.elemento);
         raiz.elemento = raiz.dir.elemento;
         raiz.dir.elemento = elemento;
      }
      raiz.esq.cor = raiz.dir.cor = false;
   }
     ...
   raiz.cor = false;
}
```

Dois elementos (raiz e dir)

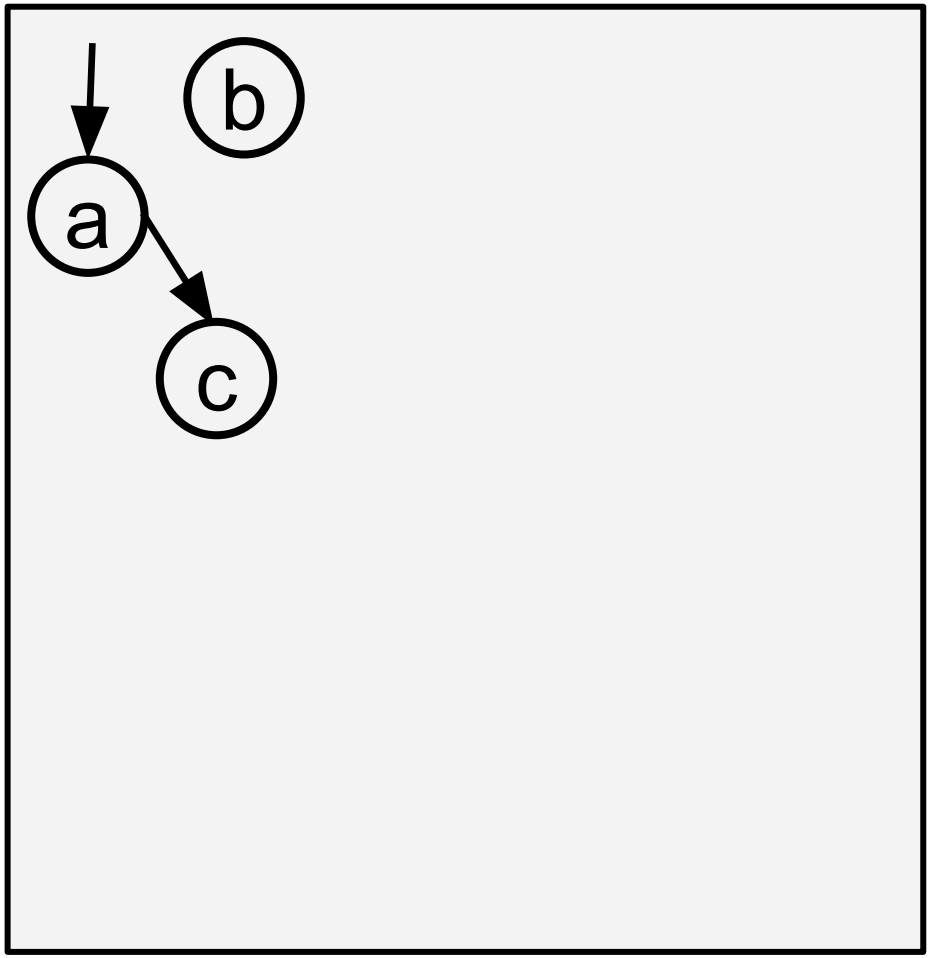


Algoritmo em C-*like*

- Se tivermos dois elementos (raiz e dir), temos três possibilidades de inserção

```
void inserir(int elemento) {
    ...
  } else if (raiz.esq == null){
    if(elemento < raiz.elemento){
      raiz.esq = new NoAN(elemento);
    } else if (elemento < raiz.dir.elemento){
      raiz.esq = new NoAN(raiz.elemento);
      raiz.elemento = elemento;
    } else {
      raiz.esq = new NoAN(raiz.elemento);
      raiz.elemento = raiz.dir.elemento;
      raiz.dir.elemento = elemento;
    }
    raiz.esq.cor = raiz.dir.cor = false;
  }
    ...
  raiz.cor = false;
}
```

Dois elementos (raiz e dir)

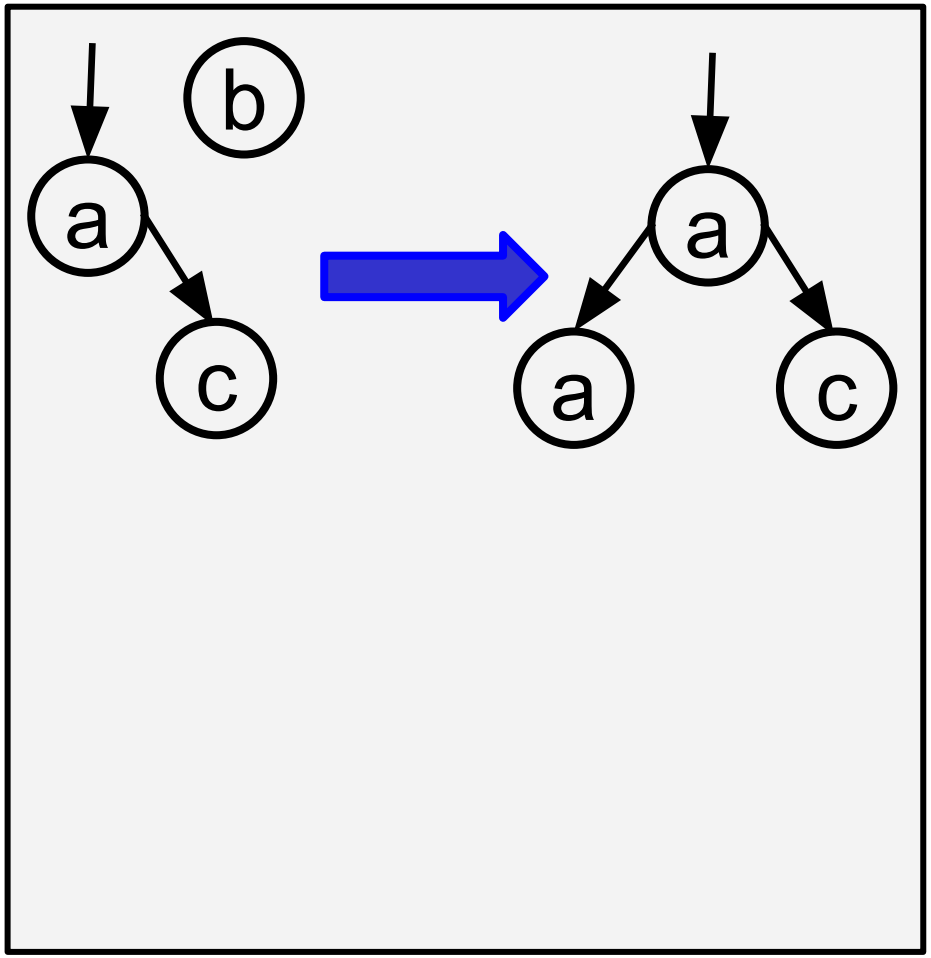


Algoritmo em C-*like*

- Se tivermos dois elementos (raiz e dir), temos três possibilidades de inserção

```
void inserir(int elemento) {
    ...
  } else if (raiz.esq == null){
    if(elemento < raiz.elemento){
      raiz.esq = new NoAN(elemento);
    } else if (elemento < raiz.dir.elemento){
      raiz.esq = new NoAN(raiz.elemento);
      raiz.elemento = elemento;
    } else {
      raiz.esq = new NoAN(raiz.elemento);
      raiz.elemento = raiz.dir.elemento;
      raiz.dir.elemento = elemento;
    }
    raiz.esq.cor = raiz.dir.cor = false;
  }
    ...
  raiz.cor = false;
}
```

Dois elementos (raiz e dir)

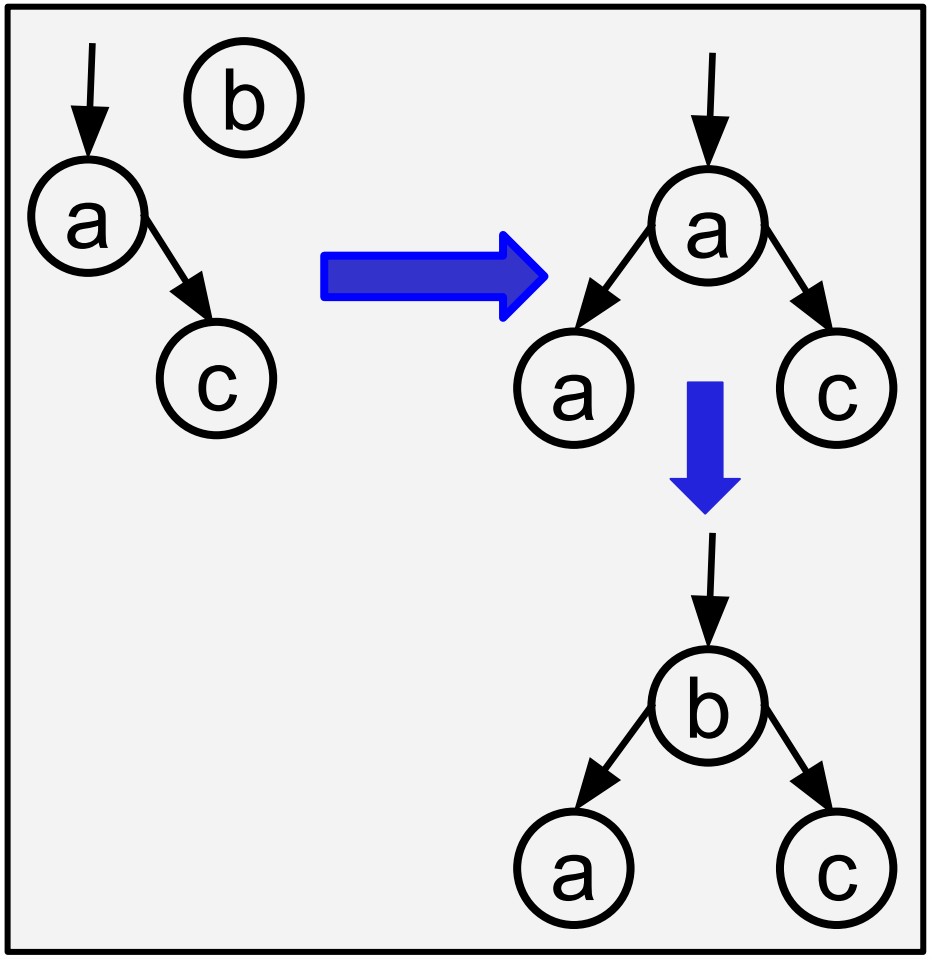


Algoritmo em C-*like*

- Se tivermos dois elementos (raiz e dir), temos três possibilidades de inserção

```
void inserir(int elemento) {
    ...
  } else if (raiz.esq == null){
    if(elemento < raiz.elemento){
      raiz.esq = new NoAN(elemento);
    } else if (elemento < raiz.dir.elemento){
      raiz.esq = new NoAN(raiz.elemento);
      raiz.elemento = elemento;
    } else {
      raiz.esq = new NoAN(raiz.elemento);
      raiz.elemento = raiz.dir.elemento;
      raiz.dir.elemento = elemento;
    }
    raiz.esq.cor = raiz.dir.cor = false;
  }
    ...
  raiz.cor = false;
}
```

Dois elementos (raiz e dir)

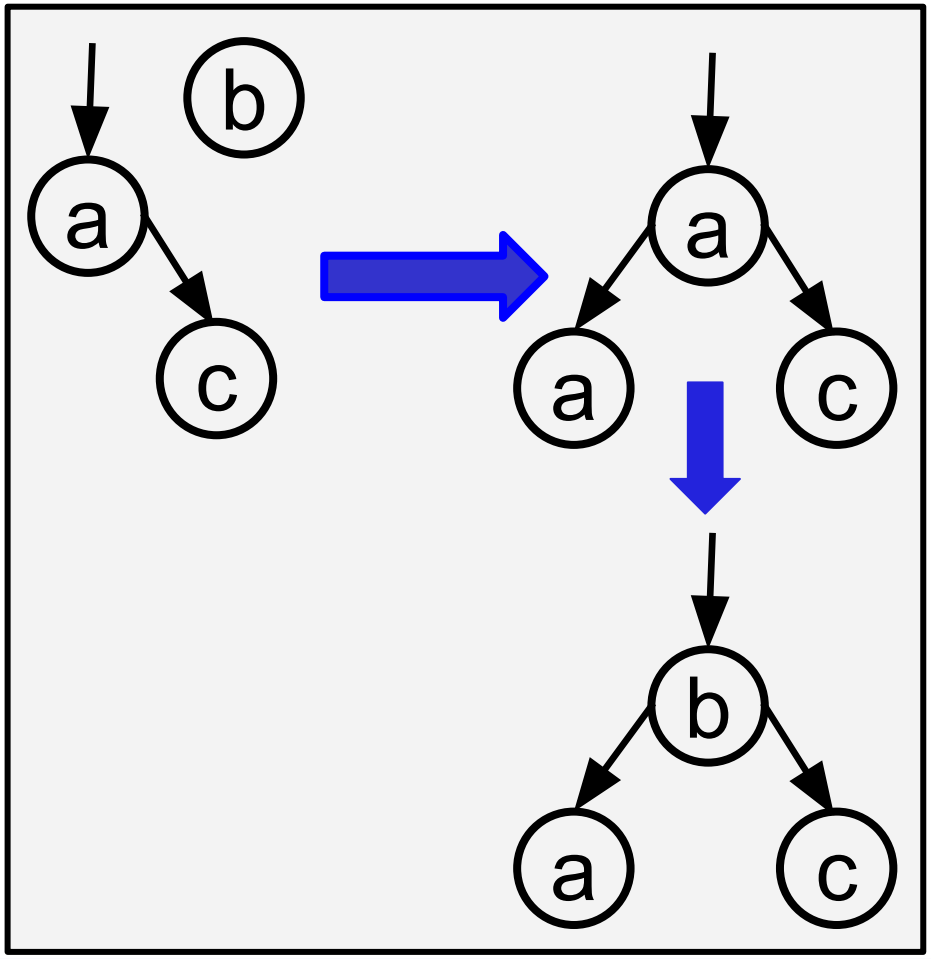


Algoritmo em C-*like*

- Se tivermos dois elementos (raiz e dir), temos três possibilidades de inserção

```
void inserir(int elemento) {
    ...
  } else if (raiz.esq == null){
    if(elemento < raiz.elemento){
      raiz.esq = new NoAN(elemento);
    } else if (elemento < raiz.dir.elemento){
      raiz.esq = new NoAN(raiz.elemento);
      raiz.elemento = elemento;
    } else {
      raiz.esq = new NoAN(raiz.elemento);
      raiz.elemento = raiz.dir.elemento;
      raiz.dir.elemento = elemento;
    }
    raiz.esq.cor = raiz.dir.cor = false;
  }
    ...
  raiz.cor = false;
}
```

Dois elementos (raiz e dir)

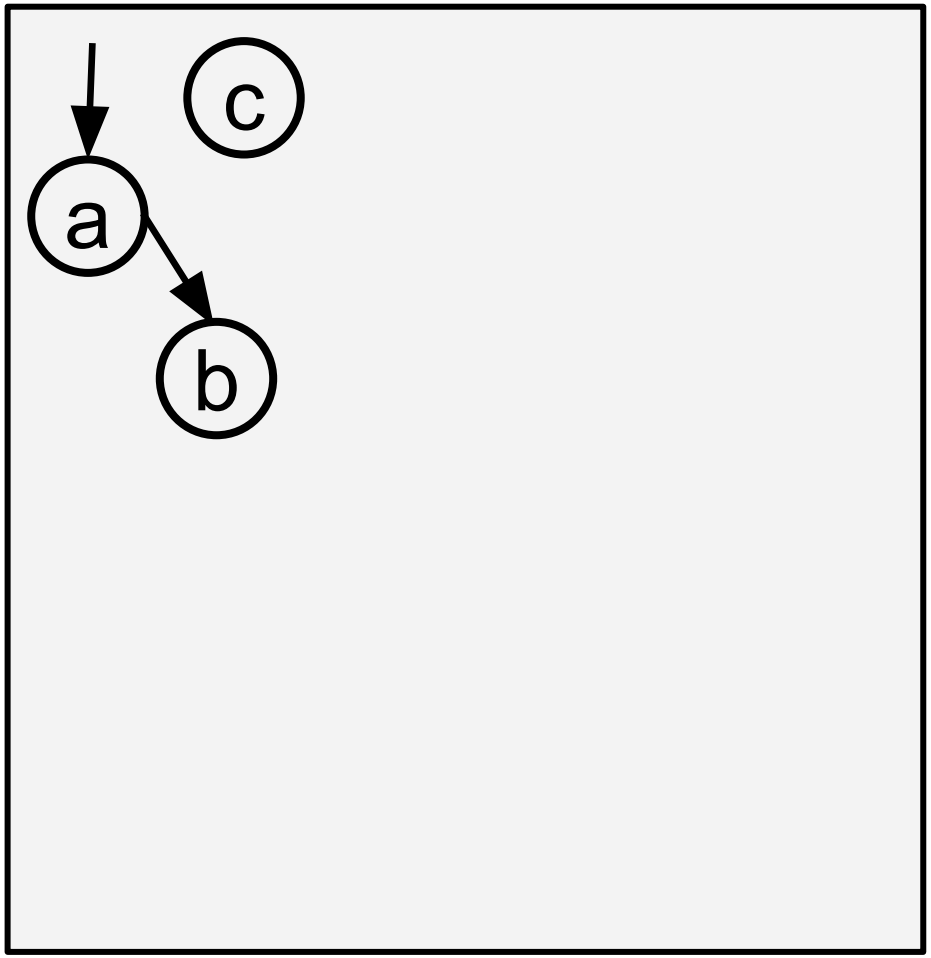


Algoritmo em C-*like*

- Se tivermos dois elementos (raiz e dir), temos três possibilidades de inserção

```
void inserir(int elemento) {
    ...
  } else if (raiz.esq == null){
    if(elemento < raiz.elemento){
      raiz.esq = new NoAN(elemento);
    } else if (elemento < raiz.dir.elemento){
      raiz.esq = new NoAN(raiz.elemento);
      raiz.elemento = elemento;
    } else {
      raiz.esq = new NoAN(raiz.elemento);
      raiz.elemento = raiz.dir.elemento;
      raiz.dir.elemento = elemento;
    }
    raiz.esq.cor = raiz.dir.cor = false;
  }
    ...
  raiz.cor = false;
}
```

Dois elementos (raiz e dir)

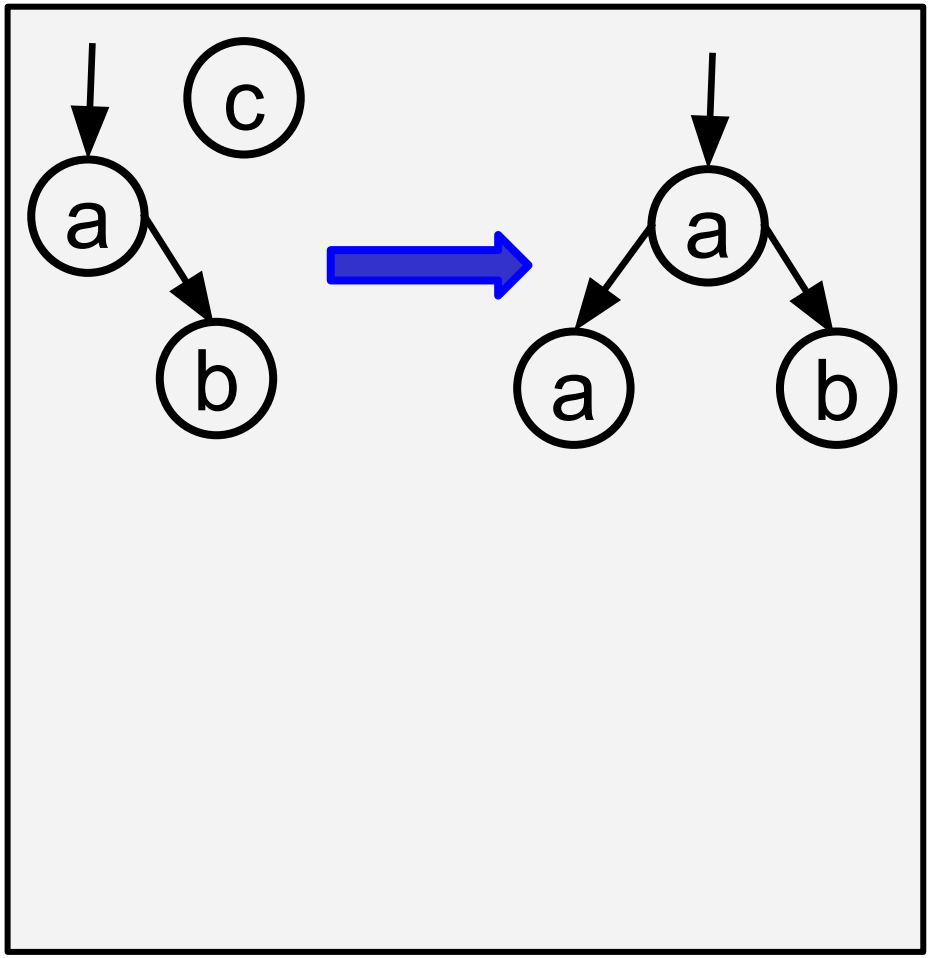


Algoritmo em C-*like*

- Se tivermos dois elementos (raiz e dir), temos três possibilidades de inserção

```
void inserir(int elemento) {
    ...
  } else if (raiz.esq == null){
    if(elemento < raiz.elemento){
      raiz.esq = new NoAN(elemento);
    } else if (elemento < raiz.dir.elemento){
      raiz.esq = new NoAN(raiz.elemento);
      raiz.elemento = elemento;
    } else {
      raiz.esq = new NoAN(raiz.elemento);
      raiz.elemento = raiz.dir.elemento;
      raiz.dir.elemento = elemento;
    }
    raiz.esq.cor = raiz.dir.cor = false;
  }
    ...
  raiz.cor = false;
}
```

Dois elementos (raiz e dir)

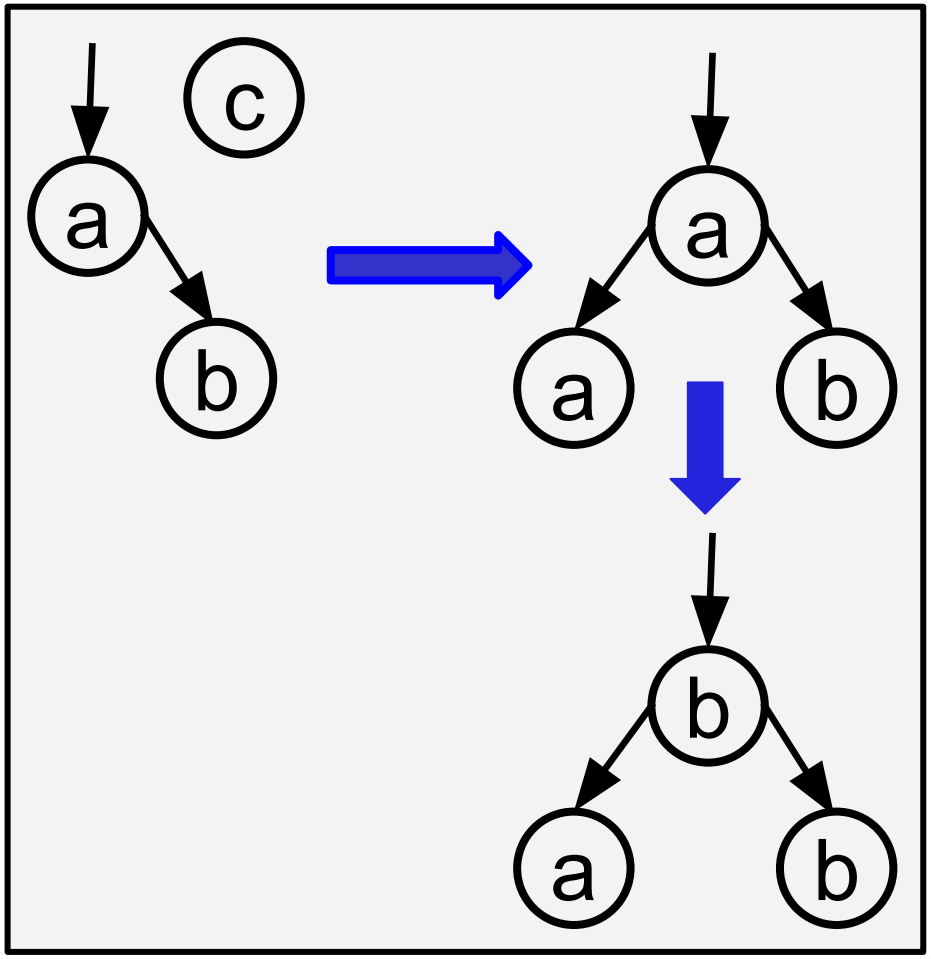


Algoritmo em C-*like*

- Se tivermos dois elementos (raiz e dir), temos três possibilidades de inserção

```
void inserir(int elemento) {
    ...
  } else if (raiz.esq == null){
    if(elemento < raiz.elemento){
      raiz.esq = new NoAN(elemento);
    } else if (elemento < raiz.dir.elemento){
      raiz.esq = new NoAN(raiz.elemento);
      raiz.elemento = elemento;
    } else {
      raiz.esq = new NoAN(raiz.elemento);
      raiz.elemento = raiz.dir.elemento;
      raiz.dir.elemento = elemento;
    }
    raiz.esq.cor = raiz.dir.cor = false;
  }
    ...
  raiz.cor = false;
}
```

Dois elementos (raiz e dir)

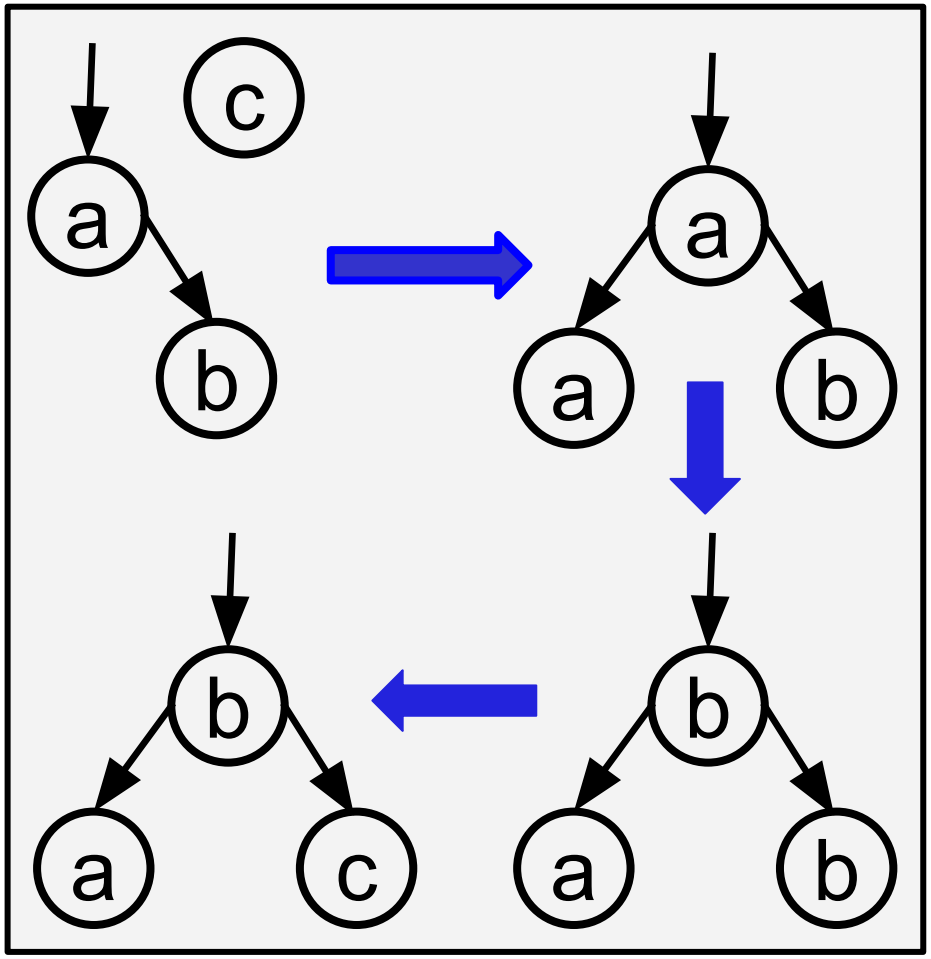


Algoritmo em C-*like*

- Se tivermos dois elementos (raiz e dir), temos três possibilidades de inserção

```
void inserir(int elemento) {
    ...
  } else if (raiz.esq == null){
    if(elemento < raiz.elemento){
      raiz.esq = new NoAN(elemento);
    } else if (elemento < raiz.dir.elemento){
      raiz.esq = new NoAN(raiz.elemento);
      raiz.elemento = elemento;
    } else {
      raiz.esq = new NoAN(raiz.elemento);
      raiz.elemento = raiz.dir.elemento;
      raiz.dir.elemento = elemento;
    }
    raiz.esq.cor = raiz.dir.cor = false;
  }
    ...
  raiz.cor = false;
}
```

Dois elementos (raiz e dir)

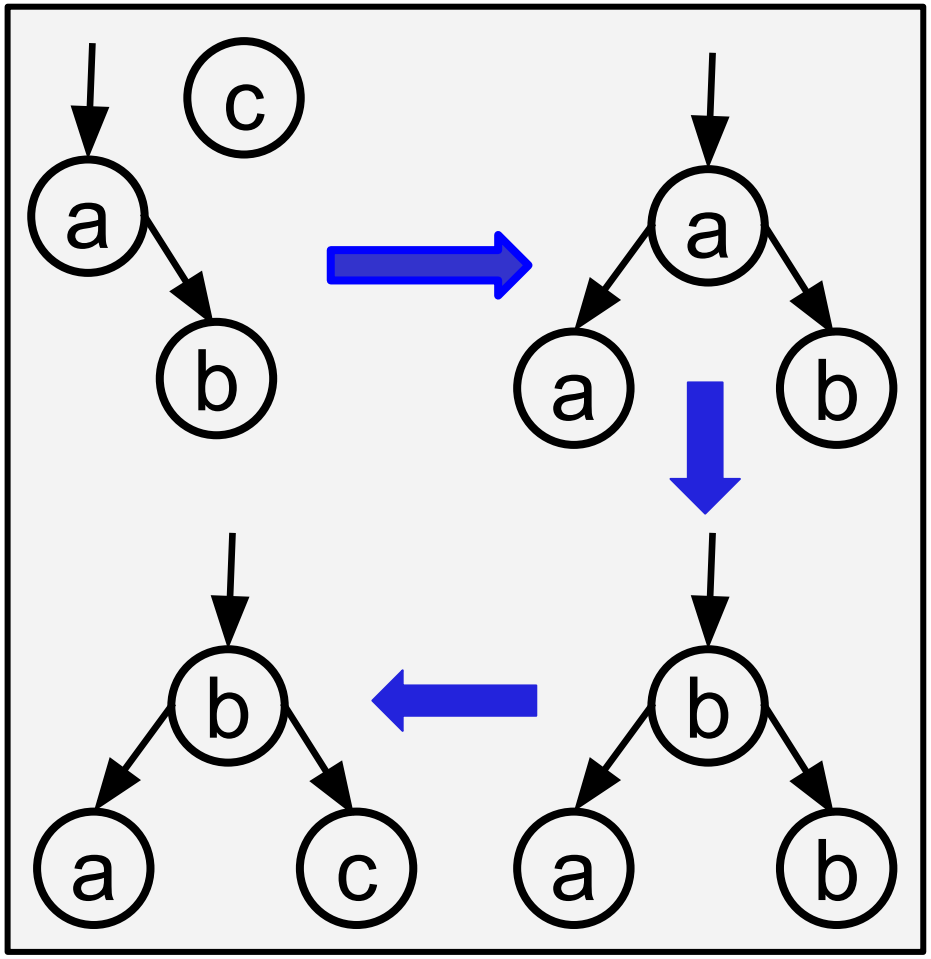


Algoritmo em C-*like*

- Se tivermos dois elementos (raiz e dir), temos três possibilidades de inserção

```
void inserir(int elemento) {
    ...
  } else if (raiz.esq == null){
    if(elemento < raiz.elemento){
      raiz.esq = new NoAN(elemento);
    } else if (elemento < raiz.dir.elemento){
      raiz.esq = new NoAN(raiz.elemento);
      raiz.elemento = elemento;
    } else {
      raiz.esq = new NoAN(raiz.elemento);
      raiz.elemento = raiz.dir.elemento;
      raiz.dir.elemento = elemento;
    }
    raiz.esq.cor = raiz.dir.cor = false;
  }
    ...
  raiz.cor = false;
}
```

Dois elementos (raiz e dir)



Algoritmo em C-*like*

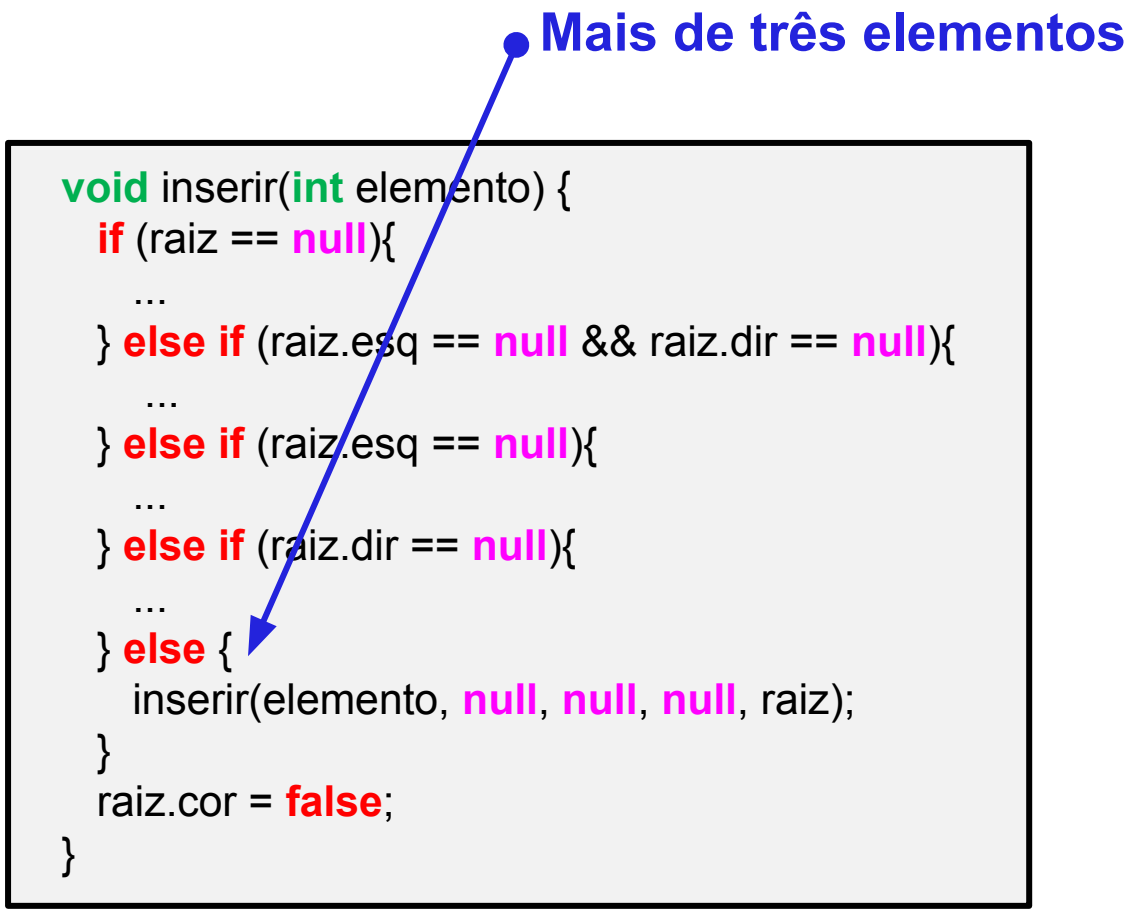
- Se tivermos dois elementos (raiz e esq), temos três possibilidades de inserção

Dois elementos (raiz e esq)

```
void inserir(int elemento) {
      ...
   } else if (raiz.dir == null){
      if(elemento > raiz.elemento){
         raiz.dir = new NoAN(elemento);
      } else if (elemento > raiz.esq.elemento){
         raiz.dir = new NoAN(raiz.elemento);
         raiz.elemento = elemento;
      } else {
         raiz.dir = new NoAN(raiz.elemento);
         raiz.elemento = raiz.esq.elemento;
         raiz.esq.elemento = elemento;
      }
      raiz.esq.cor = raiz.dir.cor = false;
   }
      ...
   raiz.cor = false;
}
```

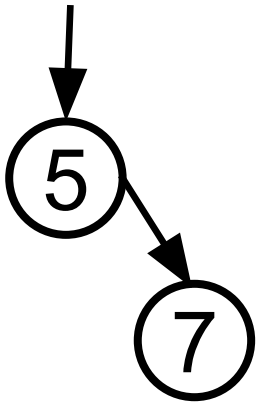
Algoritmo em C-*like*

- Se tivermos três ou mais elementos, executamos a técnica apresentada

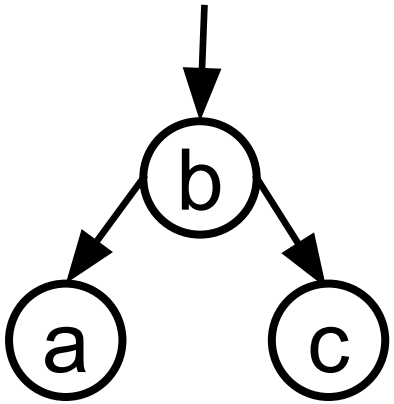


Exercício Resolvido (9)

- Suponha a existência da árvore abaixo:

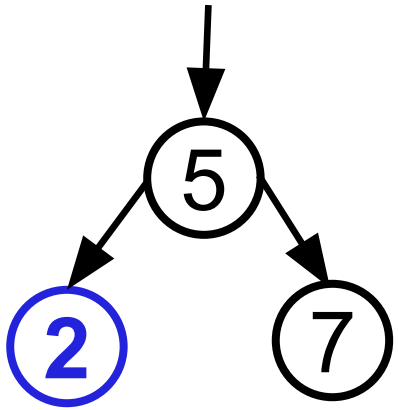


- Mostre o que devemos fazer para que nossa árvore fique como a abaixo se se inserirmos o número: (a) 2; (b) 6; e (c) 8

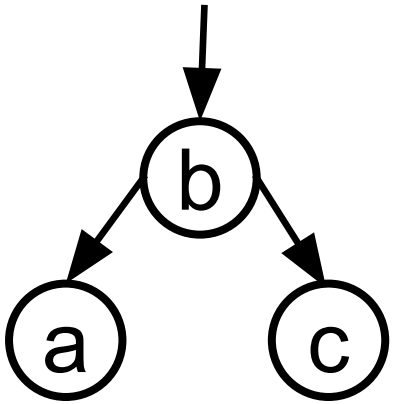


Exercício Resolvido (9)

- Suponha a existência da árvore abaixo:



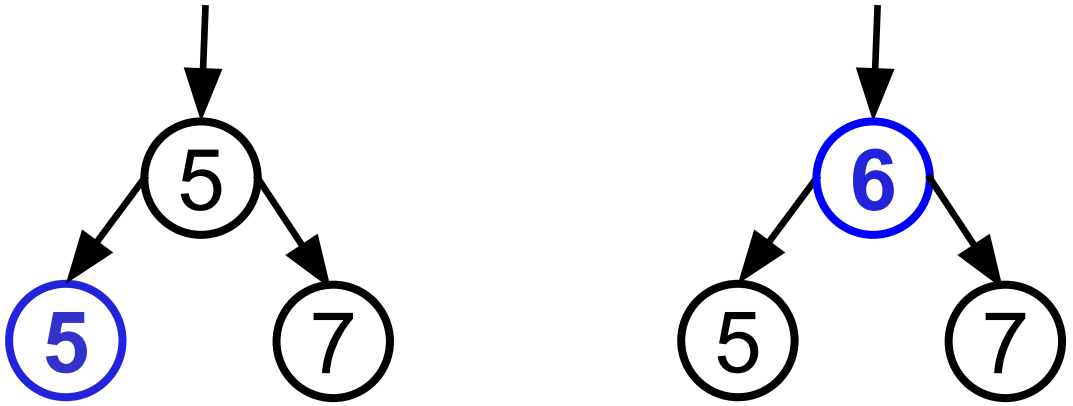
- Mostre o que devemos fazer para que nossa árvore fique como a abaixo se se inserirmos o número: **(a) 2**; (b) 6; e (c) 8



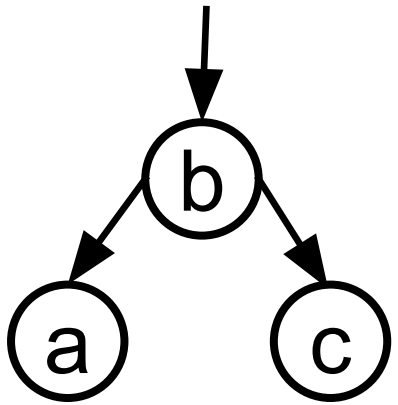
Como (raiz.esq == null e 2 < 5), basta inserir o 2 à esquerda do 5

Exercício Resolvido (9)

- Suponha a existência da árvore abaixo:



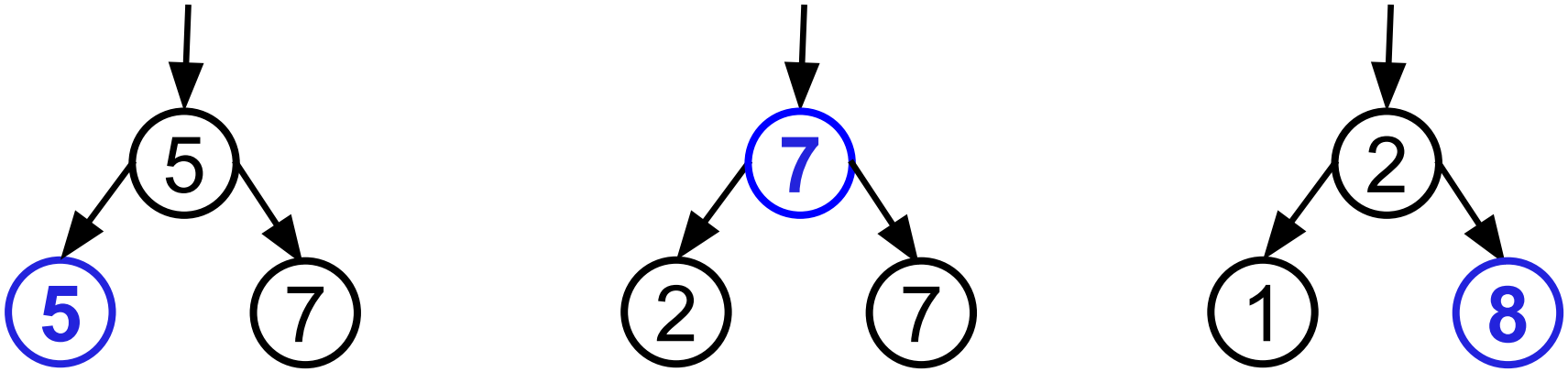
- Mostre o que devemos fazer para que nossa árvore fique como a abaixo se se inserirmos o número: (a) 2; **(b) 6**; e (c) 8



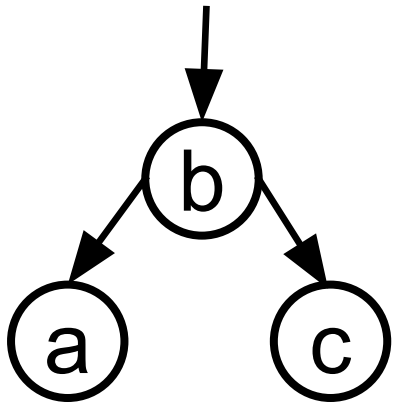
Como (raiz.esq == null e 5 < 6 < 7), basta inserir o 5 à esquerda da raiz e substituir o 6 na raiz

Exercício Resolvido (9)

- Suponha a existência da árvore abaixo:



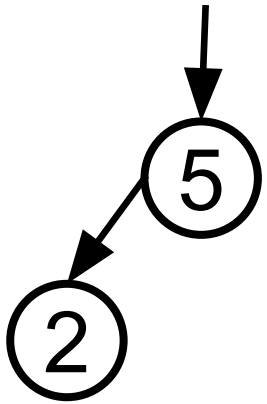
- Mostre o que devemos fazer para que nossa árvore fique como a abaixo se se inserirmos o número: (a) 2; (b) 6; e **(c) 8**



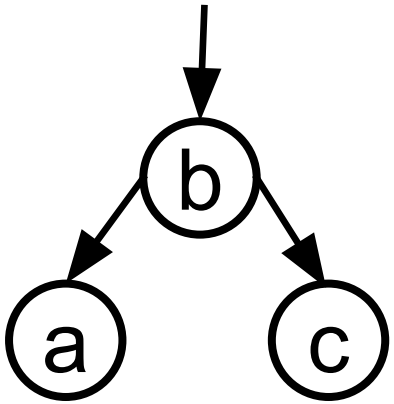
Como (raiz.esq == null e 5 < 7 < 8), basta inserir o 5 à esquerda da raiz, substituir o 7 na raiz e o 8 à direita da raiz

Exercício Resolvido (10)

- Suponha a existência da árvore abaixo:

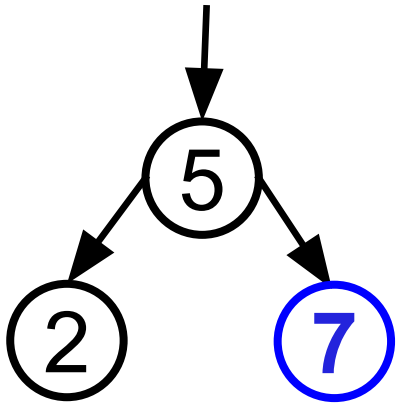


- Mostre o que devemos fazer para que nossa árvore fique como a abaixo se se inserirmos o número: (a) 7; (b) 3; e (c) 1

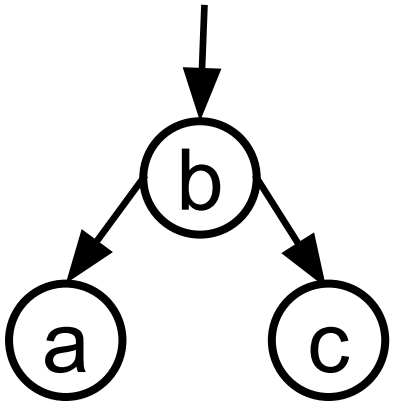


Exercício Resolvido (10)

- Suponha a existência da árvore abaixo:



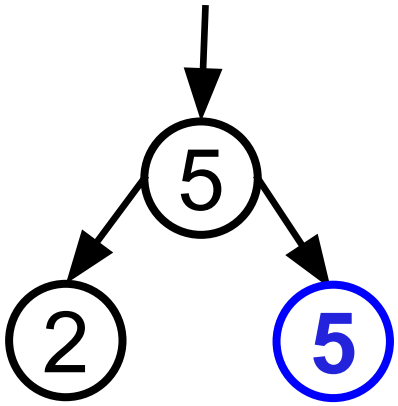
- Mostre o que devemos fazer para que nossa árvore fique como a abaixo se se inserirmos o número: (a) **7**; (b) 3; e (c) 1

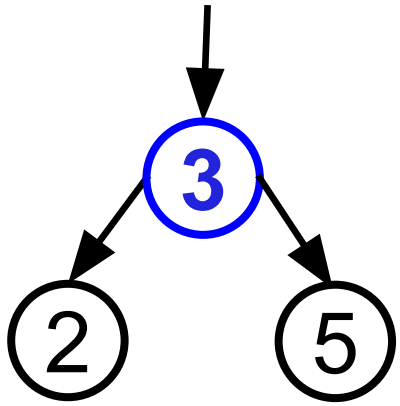


Como (raiz.dir == null e 5 < 7), basta inserir o 7 à direita do 5

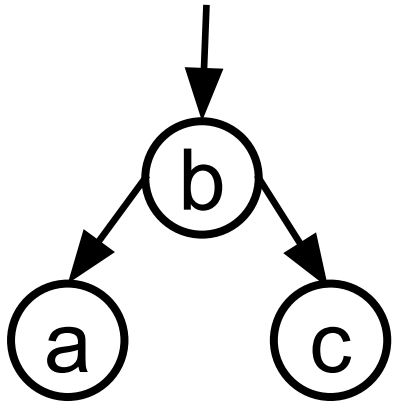
Exercício Resolvido (10)

- Suponha a existência da árvore abaixo:





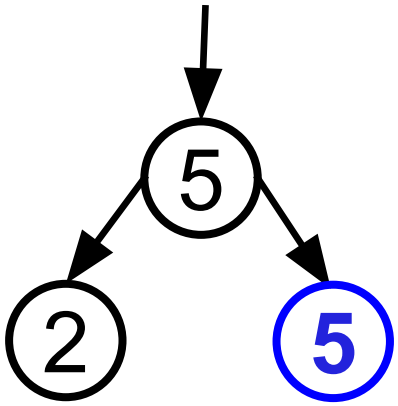
- Mostre o que devemos fazer para que nossa árvore fique como a abaixo se se inserirmos o número: (a) 7; **(b) 3**; e (c) 1

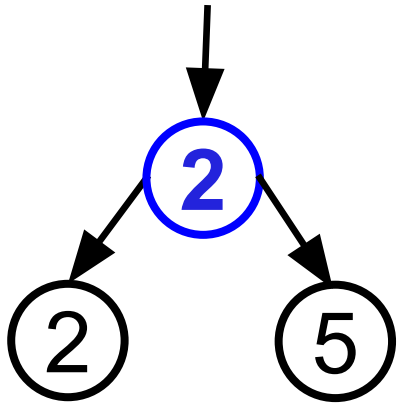


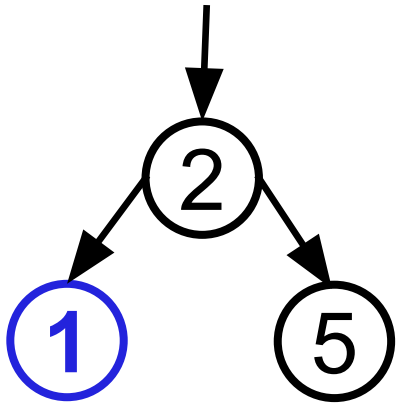
Como (raiz.dir == null e 2 < 3 < 5), basta inserir o 5 à direita da raiz e substituir o 3 na raiz

Exercício Resolvido (10)

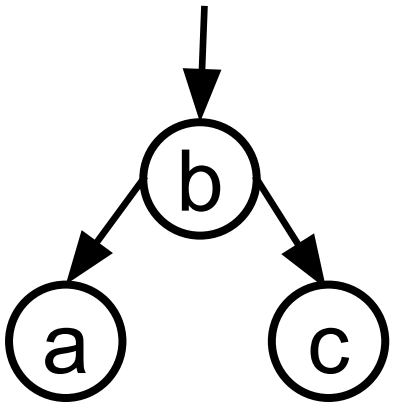
- Suponha a existência da árvore abaixo:







- Mostre o que devemos fazer para que nossa árvore fique como a abaixo se se inserirmos o número: (a) 7; (b) 3; e **(c) 1**



Como (raiz.dir == null e 1 < 2 < 5), basta inserir o 5 à direita da raiz, substituir o 2 na raiz e o 1 à esquerda da raiz

Algoritmo em C-*like*

Ver código em: fonte/unidade08/alvinegra/