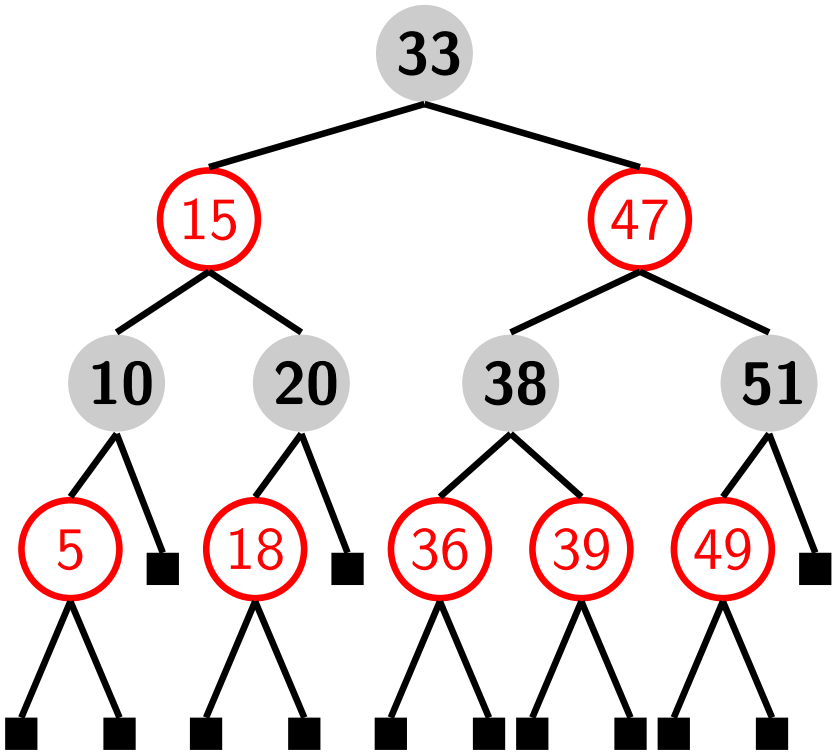
Árvore rubro-negra

Aula de hoje

Nesta aula veremos

- Árvores rubro-negras

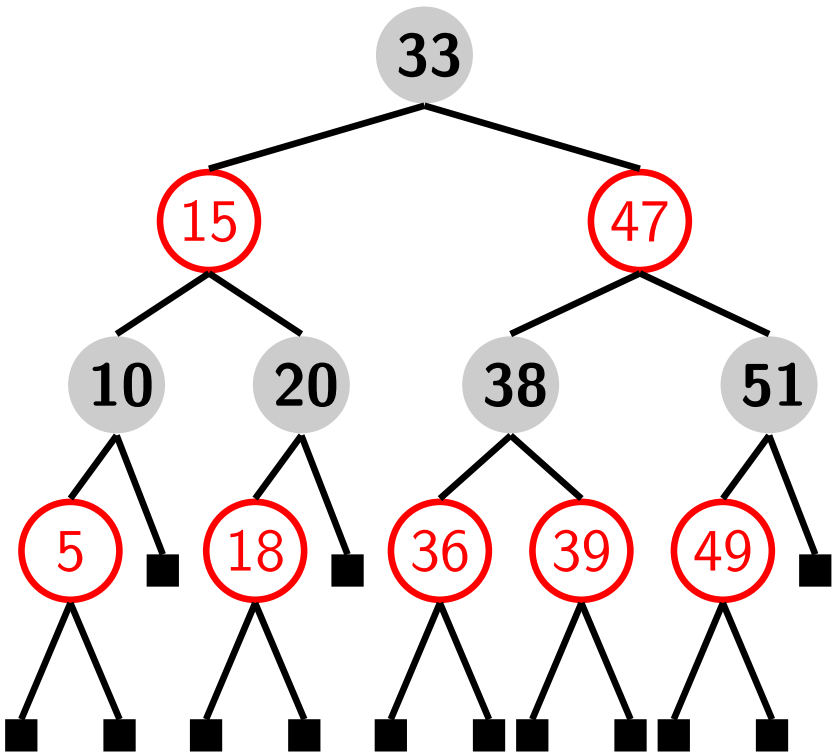
Árvore rubro-negra



Propriedades:

- 1 - Todo nó é vermelho ou preto
- 2 - A raiz é preta
- 3 - Todas as folhas são pretas, considerando inclusive as folhas vazias
- 4 - Se um nó é vermelho então seus filhos são pretos
- 5 - Todo caminho da raiz até qualquer folha tem sempre o mesmo número de nós pretos

Árvore rubro-negra



Critério de balanceamento:

- o caminho da raiz para a folha mais longe não é mais do que duas vezes maior que o caminho para a folha mais perto

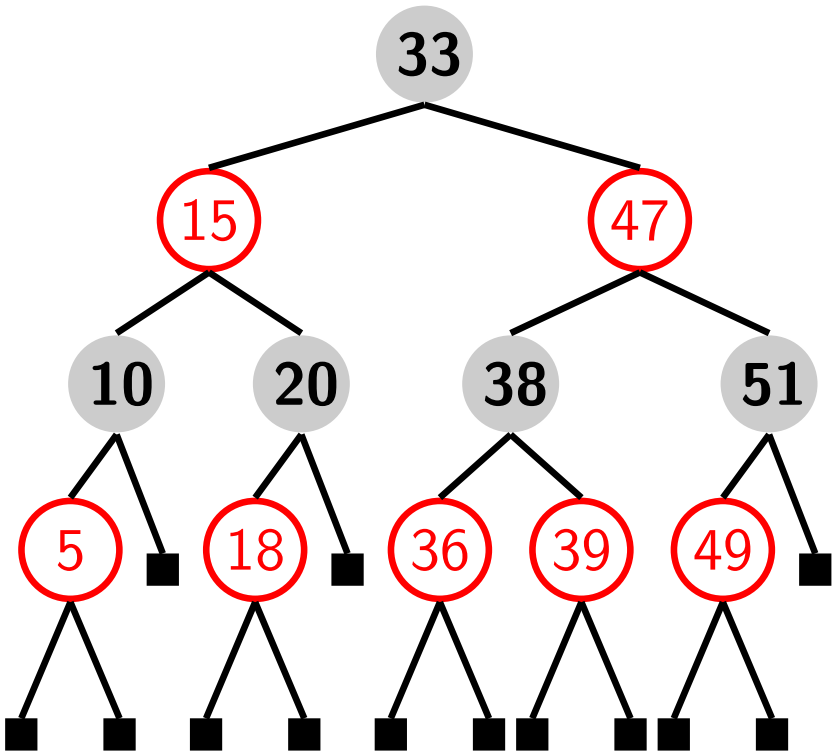
Árvore rubro-negra

Características

- Uma árvore rubro-negra com n nós tem altura menor ou igual a $2\log(n+1)$.
- Uma busca numa árvore leva um tempo $O(\log n)$.
- Inserções e retiradas podem violar as propriedades 'rubro-negras'.
- Para restabelecer as propriedades, recorre-se a rotação e recoloração dos nós

Árvore rubro-negra

Queremos inserir (1)

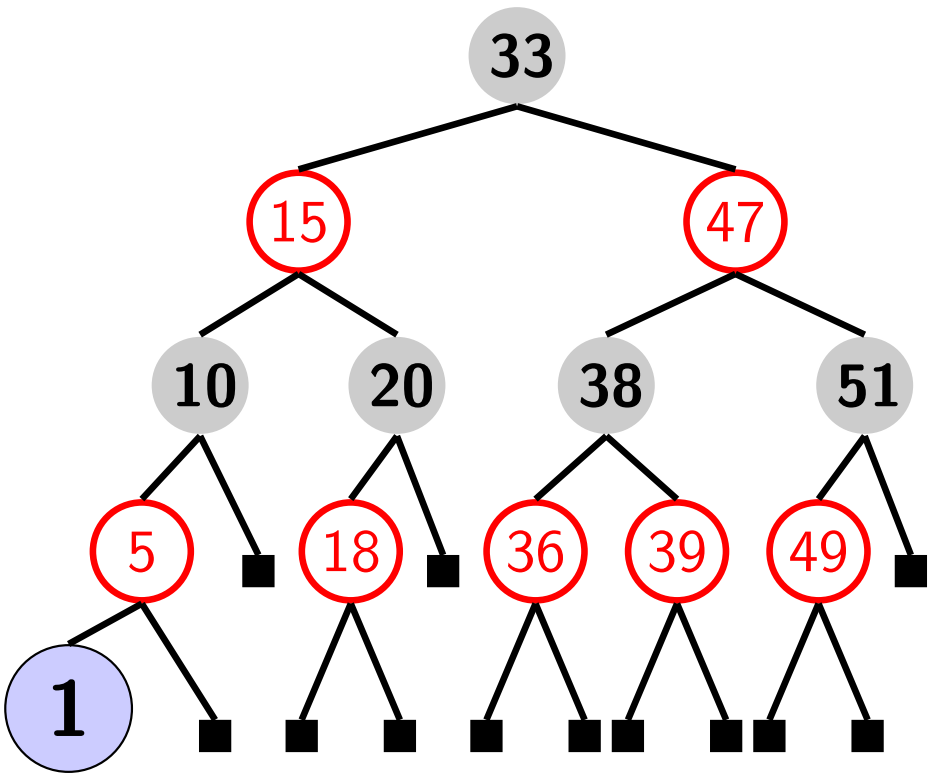


Operação de inserção

1. Insere usando algoritmo da árvore de busca binária

Árvore rubro-negra

Queremos inserir (1)

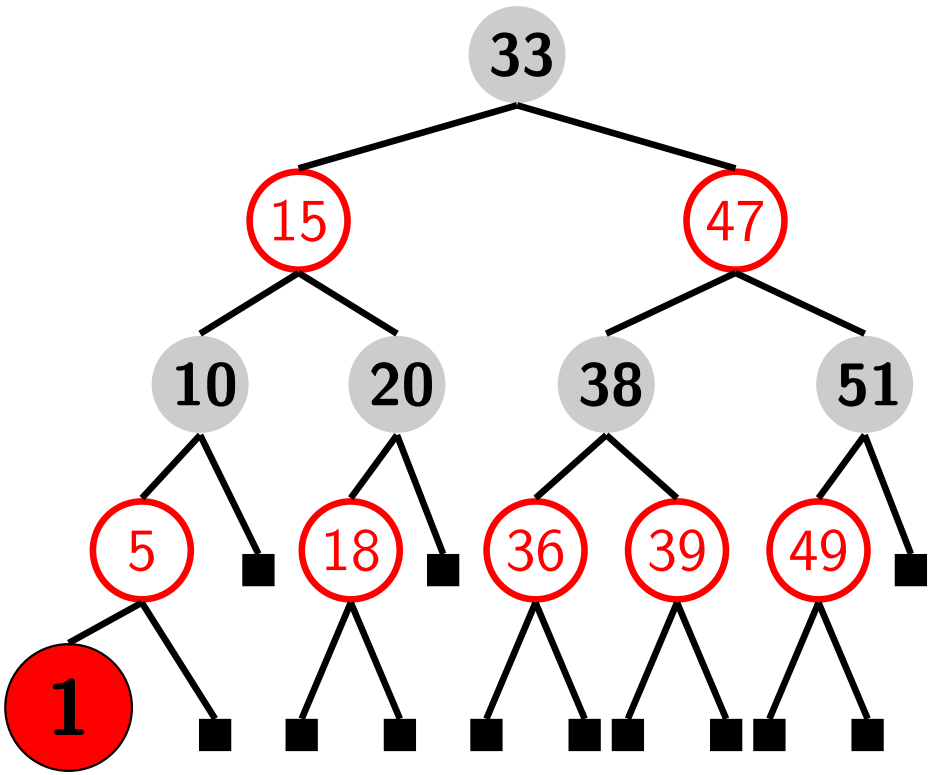


Operação de inserção

1. Insere usando algoritmo da árvore de busca binária

Árvore rubro-negra

Queremos inserir 1

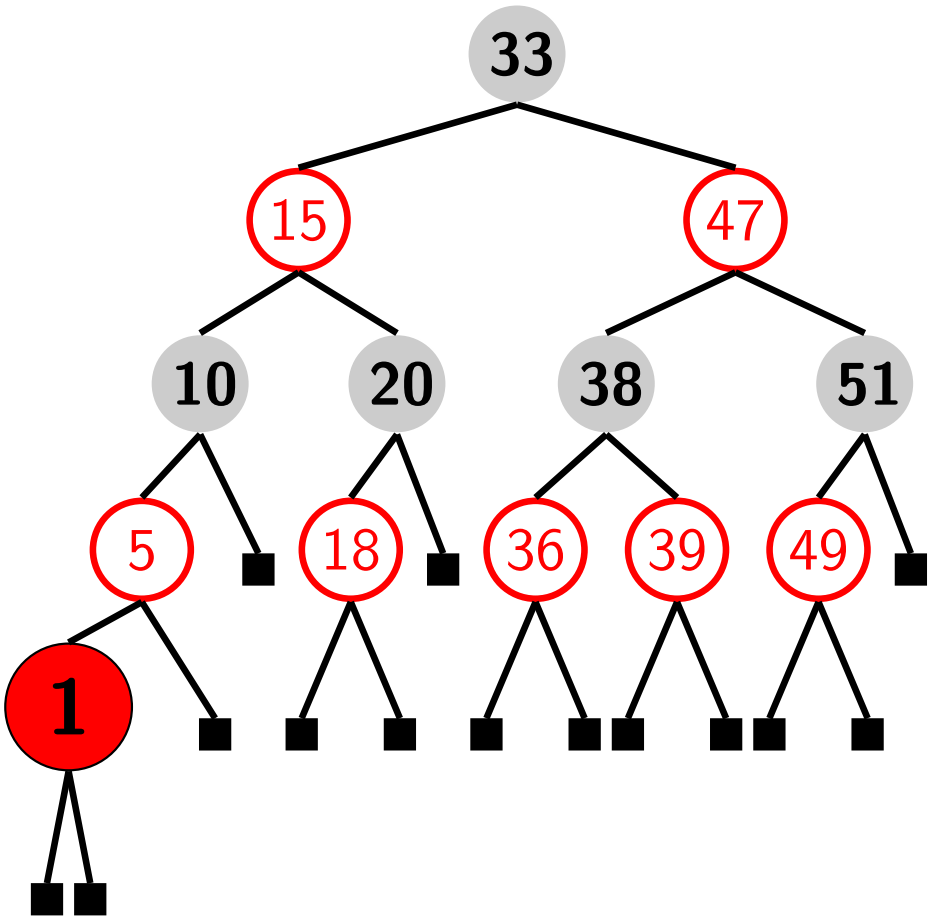


Operação de inserção

1. Insere usando algoritmo da árvore de busca binária
2. Colore o nó inserido de vermelho

Árvore rubro-negra

Queremos inserir 1

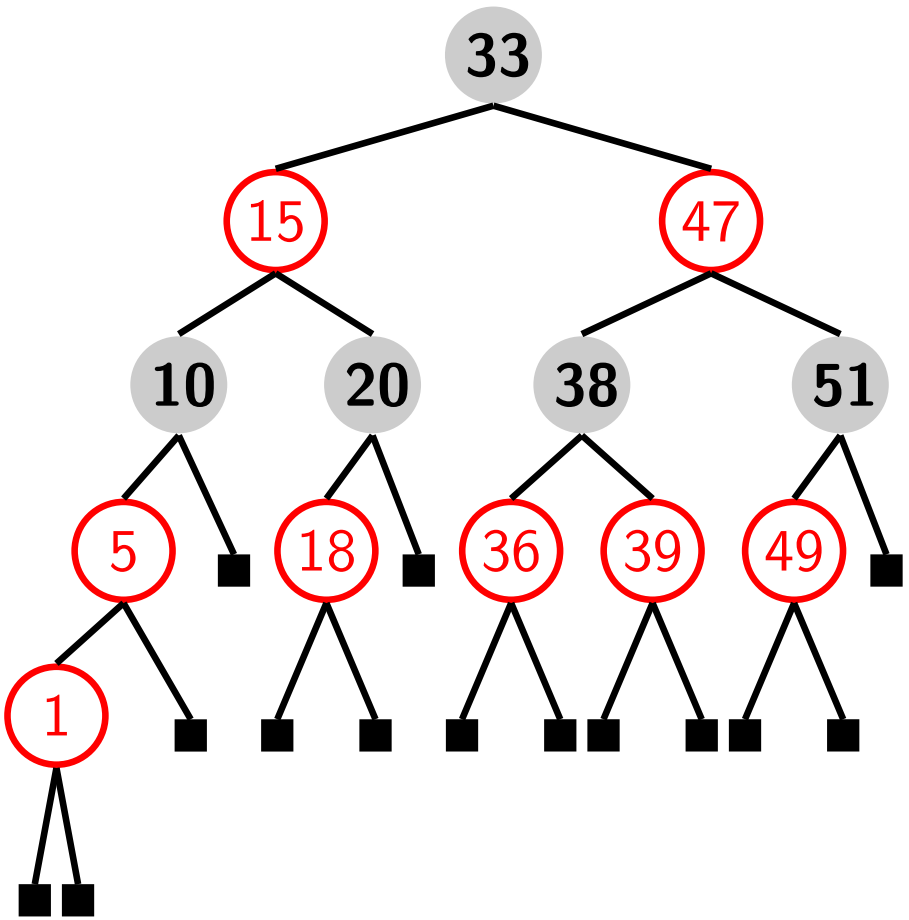


Operação de inserção

1. Insere usando algoritmo da árvore de busca binária
2. Colore o nó inserido de vermelho
3. Adiciona dois nós folhas-preto vazios
4. Próximo passo depende das cores dos nós vizinhos

Árvore rubro-negra

Rebalancear árvore após inserção de (1)



Operação de rebalanceamento

1. Próximo passo depende das cores dos nós vizinhos
   - Propriedade 4 (filhos de nó vermelho são pretos) é ameaçada por adicionar nó vermelho, pintar um nó preto em vermelho ou por uma rotação
   - Propriedade 5 (todos os caminhos da raiz para folha têm o mesmo número de nós pretos) é ameaçada ao adicionar nó preto, mudar a cor de um nó ou por uma rotação

Estrutura de dados

Cada nó da árvore tem uma cor, preto ou vermelho e sabe quem é seu nó-pai e os dois filhos: direita e esquerda.

```
1 class Node {
2     int cor; // 0 é BLACK preto, 1 é RED vermelho
3     int id;
4
5     Node esq, dir;
6     Node pai;
7 }
```

Avôs e tios

Usaremos a noção de avô e tio.

```
1 Node avo(Node n) {
2   if ((n != null) && (n.pai != null))
3     return n.pai.pai;
4   else
5     return null;
6 }
```

```
1  Node tio(Node n) {
2    Node a = avo(n);
3    if (a == null)
4      return null; // sem avo, sem tio
5
6    if (n.pai == a.esq)
7        return a.dir;
8    else
9        return a.esq;
10 }
```

Inserção: caso 1

Se (n) é raiz, pinta de preto.

```
1 void insercao_caso1(Node n)
      {
2 if (n.pai == null)
3   n.pai.cor = BLACK;
4 else
5   insercao_caso2(n);
6 }
```

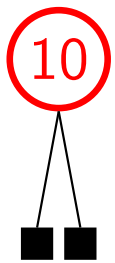
Exemplo: árvore está vazia, insere (10).

Inserção: caso 1

Se (n) é raiz, pinta de preto.

```
1 void insercao_caso1(Node n)
      {
2 if (n.pai == null)
3   n.pai.cor = BLACK;
4 else
5   insercao_caso2(n);
6 }
```

Exemplo: árvore está vazia, insere (10). Inicialmente insere como vermelho.

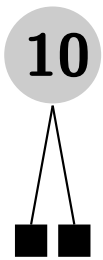


Inserção: caso 1

Se (n) é raiz, pinta de preto.

```
1 void insercao_caso1(Node n)
      {
2 if (n.pai == null)
3   n.pai.cor = BLACK;
4 else
5   insercao_caso2(n);
6 }
```

Exemplo: árvore está vazia, insere (10). Agora aplica caso 1.

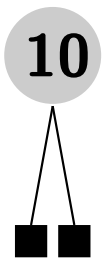


Inserção: caso 2

Se o pai (p) do nó atual (n) é preto, então não faz nada, senão passa para o caso 3.

```
1 void insercao_caso2(Node n)
      {
2 if (n.pai.cor == BLACK)
3 // árvore ainda está válida
4   return;
5 else
6 // conflito: pai e nós são
      rubros
7   insercao_caso3(n);
8 }
```

Exemplo: inserir (15) na árvore a seguir:

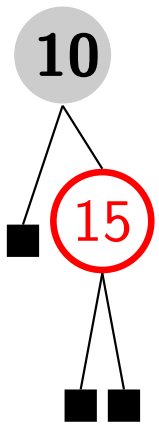


Inserção: caso 2

Se o pai (p) do nó atual (n) é preto, então não faz nada, senão passa para o caso 3.

```
1 void insercao_caso2(Node n)
      {
2 if (n.pai.cor == BLACK)
3 // árvore ainda está válida
4   return;
5 else
6 // conflito: pai e nós são
      rubros
7   insercao_caso3(n);
8 }
```

Exemplo: inserir (15) na árvore a seguir.

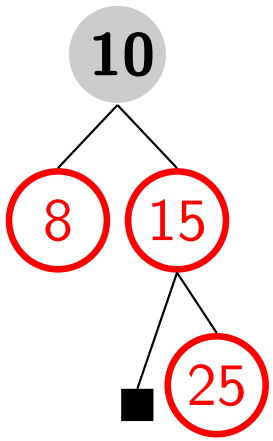


Inserção: caso 3

- Se pai (p) e tio (t) são vermelhos, então são pintados de preto e avô (g) de vermelho.

```
1  void insercao_caso3(Node n) {
2   Node t = tio(n), g;
3
4   if ((t != null)&&(t.cor == RED)){
5     n.pai.cor = BLACK;
6     t.cor = BLACK;
7     g = avo(n);
8     g.cor = RED;
9     insercao_caso1(g);
10  } else {
11    insercao_caso4(n);
12  }
13 }
```

Logo após inserir (25) :

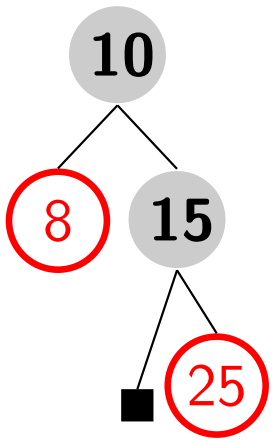


Inserção: caso 3

- Se pai (p) e tio (t) são vermelhos, então são pintados de preto e avô (g) de vermelho.

```
1  void insercao_caso3(Node n) {
2   Node t = tio(n), g;
3
4   if ((t != null)&&(t.cor == RED)){
5     n.pai.cor = BLACK;
6     t.cor = BLACK;
7     g = avo(n);
8     g.cor = RED;
9     insercao_caso1(g);
10  } else {
11    insercao_caso4(n);
12  }
13 }
```

Usando inserção caso 3: linha **5**

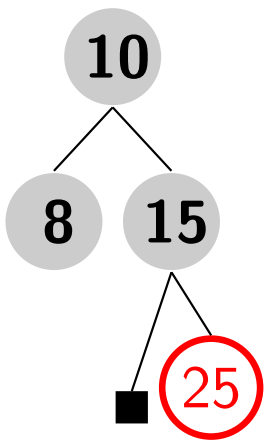


Inserção: caso 3

- Se pai (p) e tio (t) são vermelhos, então são pintados de preto e avô (g) de vermelho.

```
1  void insercao_caso3(Node n) {
2   Node t = tio(n), g;
3
4   if ((t != null)&&(t.cor == RED)){
5      n.pai.cor = BLACK;
6      t.cor = BLACK;
7      g = avo(n);
8      g.cor = RED;
9      insercao_caso1(g);
10  } else {
11     insercao_caso4(n);
12  }
13 }
```

Usando inserção caso 3: linha **6**

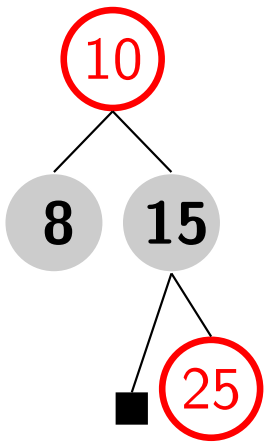


Inserção: caso 3

- Se pai (p) e tio (t) são vermelhos, então são pintados de preto e avô (g) de vermelho.

```
1  void insercao_caso3(Node n) {
2   Node t = tio(n), g;
3
4   if ((t != null)&&(t.cor == RED)){
5      n.pai.cor = BLACK;
6      t.cor = BLACK;
7      g = avo(n);
8      g.cor = RED;
9      insercao_caso1(g);
10  } else {
11     insercao_caso4(n);
12  }
13 }
```

Usando inserção caso 3: linha **8**

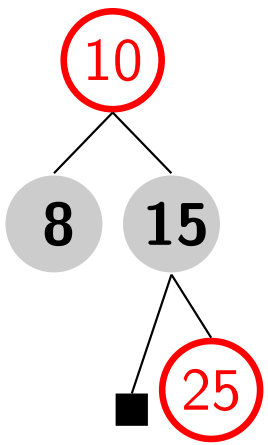


Inserção: caso 3

- Se pai (p) e tio (t) são vermelhos, então são pintados de preto e avô (g) de vermelho.

```
1  void insercao_caso3(Node n) {
2   Node t = tio(n), g;
3
4   if ((t != null)&&(t.cor == RED)){
5      n.pai.cor = BLACK;
6      t.cor = BLACK;
7      g = avo(n);
8      g.cor = RED;
9      insercao_caso1(g);
10  } else {
11     insercao_caso4(n);
12  }
13 }
```

Se avô (g) for a raiz, isso viola a condição de que **raiz** tem que ser de cor **preta**.



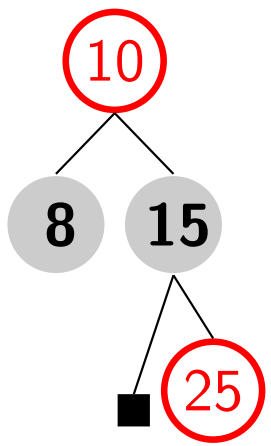
Corrigir com `insercao_caso1(g)`.

Inserção: caso 3

- Se pai (p) e tio (t) são vermelhos, então são pintados de preto e avô (g) de vermelho.

```
1  void insercao_caso3(Node n) {
2   Node t = tio(n), g;
3
4   if ((t != null)&&(t.cor == RED)){
5      n.pai.cor = BLACK;
6      t.cor = BLACK;
7      g = avo(n);
8      g.cor = RED;
9      insercao_caso1(g);
10  } else {
11     insercao_caso4(n);
12  }
13 }
```

Alternativa: Se avô (g) for filho de nó vermelho, isso viola a condição de que todo filho de nó vermelho é de cor **preta**.



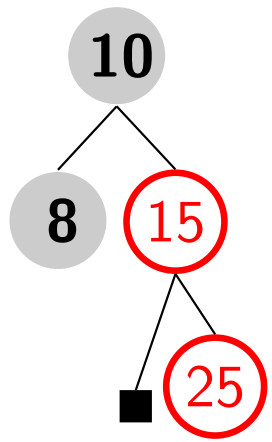
Corrigir com `insercao_caso1(g)`.

Inserção: caso 3

- Se pai (p) e tio (t) são vermelhos, então são pintados de preto e avô (g) de vermelho.

```
1  void insercao_caso3(Node n) {
2   Node t = tio(n), g;
3
4   if ((t != null)&&(t.cor == RED)){
5      n.pai.cor = BLACK;
6      t.cor = BLACK;
7      g = avo(n);
8      g.cor = RED;
9      insercao_caso1(g);
10  } else {
11     insercao_caso4(n);
12  }
13 }
```

Se tio (t) for preto, temos `insercao_caso4(n)`.

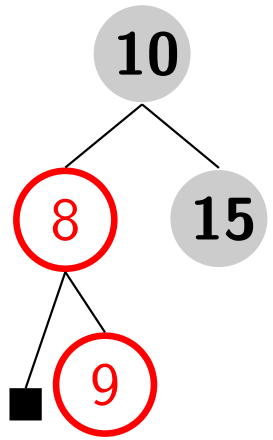


Inserção: caso 4

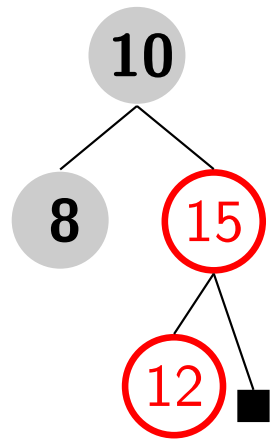
- Se pai (p) for rubro e tio (t) for negro, então ...
- ... correção com rotação à direita ou esquerda em (p)

```
1  void insercao_caso4(Node n) {
2   Node a = avo(n), p = n.pai;
3
4   if (n == p.dir && p == a.esq) {
5      rotacao_esq(p);
6      n = n.esq;
7   }else if (n == p.esq && p == a.dir){
8      rotacao_dir(p);
9      n = n.dir;
10  }
11    insercao_caso5(n);
12 }
```

Subcaso: esq. dir.



Subcaso: dir. esq.

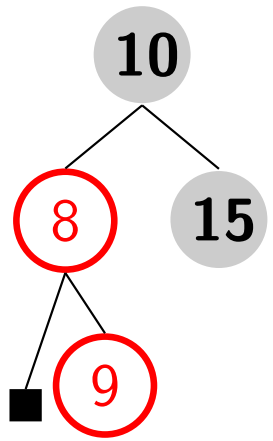


Inserção: caso 4: subcaso pai à esq., filho à dir.

- Se pai (p) for rubro e tio (t) for negro ...
- ... e nó atual (n) está à direita de seu pai
- ... e pai está à esquerda de avô
- Correção com rotação à esquerda em (p)

```
1  void insercao_caso4(Node n) {
2   Node a = avo(n), p = n.pai;
3
4   if (n == p.dir && p == a.esq)  {
5      rotacao_esq(p);
6      n = n.esq;
7   }else if (n == p.esq && p == a.dir){
8      rotacao_dir(p);
9      n = n.dir;
10  }
11    insercao_caso5(n);
12 }
```

Subcaso: filho (n) à direita de pai (p) à esquerda de avô (a)

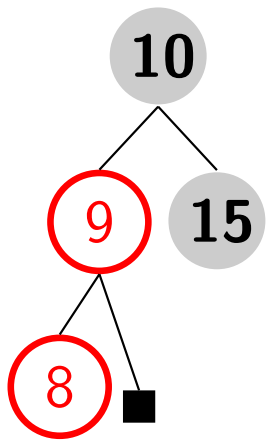


Inserção: caso 4: subcaso pai à esq., filho à dir.

- Se pai (p) for rubro e tio (t) for negro ...
- ... e nó atual (n) está à direita de seu pai
- ... e pai está à esquerda de avô
- Correção com rotação à esquerda em (p)

```
1  void insercao_caso4(Node n) {
2   Node a = avo(n), p = n.pai;
3
4  if (n == p.dir && p == a.esq) {
5      rotacao_esq(p);
6      n = n.esq;
7   }else if (n == p.esq && p == a.dir){
8      rotacao_dir(p);
9      n = n.dir;
10  }
11    insercao_caso5(n);
12 }
```

Subcaso: rotação à esquerda de pai (p)

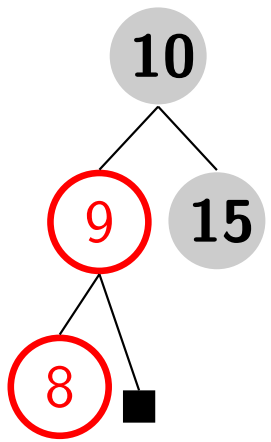


Inserção: caso 4: subcaso pai à esq., filho à dir.

- Se pai (p) for rubro e tio (t) for negro ...
- ... e nó atual (n) está à direita de seu pai
- ... e pai está à esquerda de avô
- Correção com rotação à esquerda em (p)

```
1  void insercao_caso4(Node n) {
2   Node a = avo(n), p = n.pai;
3
4  if (n == p.dir && p == a.esq) {
5      rotacao_esq(p);
6      n = n.esq;
7   }else if (n == p.esq && p == a.dir){
8      rotacao_dir(p);
9      n = n.dir;
10  }
11     insercao_caso5(n);
12 }
```

Como violação de regra sobre filhos de rubro continua, ir à esquerda no `insercao_caso5`.

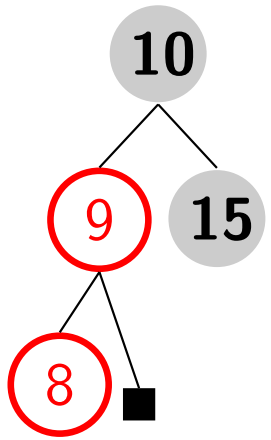


Inserção: caso 5

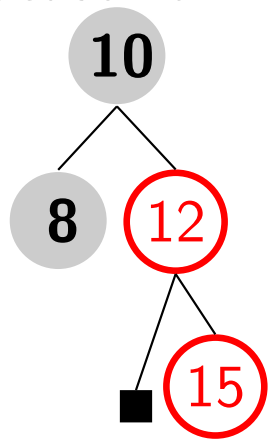
- Se pai (p) for rubro e tio (t) for negro
- e pai e filho estão para mesmo lado (dir dir ou esq esq), então ...
- ... correção: **cores** do pai e avô são **trocadas**
- ... e **rotação no avô** (a)

```
1  insercao_caso5(Node n) {
2   Node a = avo(n);
3   n.pai.cor = BLACK;
4   a.cor = RED;
5
6   if (n == n.pai.esq)
7     rotacao_dir(a);
8   else
9     rotacao_esq(a);
10 }
```

Subcaso: esq. esq.



Subcaso: dir. dir.

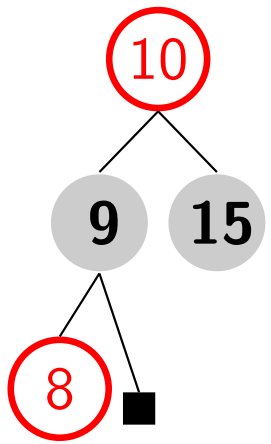


Inserção: caso 5 – subcaso pai e filho à esquerda

- Se pai (p) for rubro e tio (t) for negro
- e pai e filho estão para mesmo lado (dir dir ou esq esq), então ...
- ... correção: **cores** do pai e avô são **trocadas**
- ... e **rotação no avô** (a)

```
1  insercao_caso5(Node n) {
2   Node a = avo(n);
3   n.pai.cor = BLACK;
4   a.cor = RED;
5
6   if (n == n.pai.esq)
7     rotacao_dir(a);
8   else
9     rotacao_esq(a);
10 }
```

Troca cores do pai e avô:



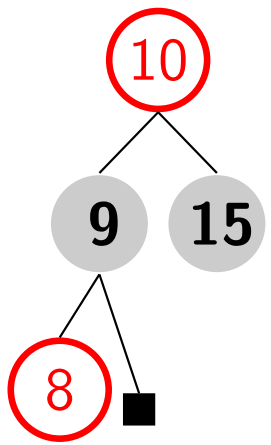
Mas raiz tem que ser preta.

Inserção: caso 5 – subcaso pai e filho à esquerda

- Se pai (p) for rubro e tio (t) for negro
- e pai e filho estão para mesmo lado (dir dir ou esq esq), então ...
- ... correção: **cores** do pai e avô são **trocadas**
- ... e **rotação no avô** (a)

```
1  insercao_caso5(Node n) {
2   Node a = avo(n);
3   n.pai.cor = BLACK;
4   a.cor = RED;
5
6   if (n == n.pai.esq)
7     rotacao_dir(a);
8   else
9     rotacao_esq(a);
10 }
```

Para raiz ser preta, rotação à direita no avô:

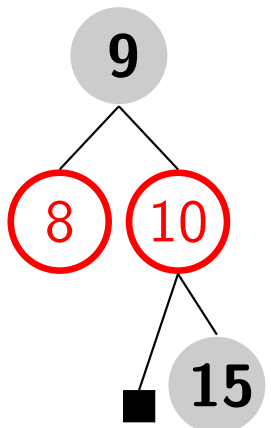


Inserção: caso 5 – subcaso pai e filho à esquerda

- Se pai (p) for rubro e tio (t) for negro
- e pai e filho estão para mesmo lado (dir dir ou esq esq), então ...
- ... correção: **cores** do pai e avô são **trocadas**
- ... e **rotação no avô** (a)

```
1  insercao_caso5(Node n) {
2   Node a = avo(n);
3   n.pai.cor = BLACK;
4   a.cor = RED;
5
6   if (n == n.pai.esq)
7     rotacao_dir(a);
8   else
9     rotacao_esq(a);
10 }
```

Rotação à direita no avô:



Inserção: observações

- Folha vazia é conhecida como **sentinela**
- É possível formalização sem sentinelas, mas algoritmo fica mais complexo
- Custo no pior caso da inserção $O(\log n)$
- Custo da recoloração e do rebalanceamento ?