Árvores

Árvores *Red-Black* – Parte II: Remoção

Prof. Edson Alves - UnB/FGA

2018

Sumário

Remoção

Remoção em árvores red-black

- De forma semelhante ao que acontece com as árvores binárias de busca (pois as árvores *red-black* são árvores binárias de busca), a remoção deve ser tratada em três casos distintos
- Os casos dependente do número de filhos que não sejam folhas do nó N a ser removido: nenhum, um ou dois
- O caso de dois filhos pode ser removido a um dos dois casos anteriores, usando a mesma estratégia da remoção por cópia: basta substituir a informação do nó N pela informação do nó mais à direita D da sub-árvore à esquerda e proceder com a remoção fazendo $N = D$
- Assim, a partir deste ponto, será considerado que o nó N a ser removido tem, no máximo, um filho C que não seja folha

Remoção: caso trivial

- O caso trivial da remoção ocorre quando a informação a ser removida não consta na árvore
- Neste caso não há o que remover, e a rotina deve ser encerrada
- Para identificar tal caso, é necessário o auxílio de uma função auxiliar, que localiza o ponteiro do nó a ser removido
- Caso esta rotina não localize o nó, o ponteiro retornado será nulo e a remoção será encerrada
- Outra rotina auxiliar útil é a que reduz o caso 3 (o nó N a ser removido tem dois filhos que não são folhas) para o caso 1 ou o caso 2
- Esta rotina deve retornar o ponteiro do novo nó a ser removido

Implementação das rotinas auxiliares

```
142     Node * find(Node *node, const T& info)
143     {
144         if (node == nullptr or node->info == info)
145             return node;
146 
147         return info < node->info ? find(node->left, info) :
148             find(node->right, info);
149     }
150 
151     // Troca de informação com o nó mais à direita da sub-árvore
152     // à esquerda de N, e retorna um ponteiro para D
153     Node * swap_info(Node *N)
154     {
155         auto D = N->left;
156 
157         while (D->right)
158             D = D->right;
159 
160         std::swap(N->info, D->info);
161         return D;
162     }
```

Implementação da rotina de remoção

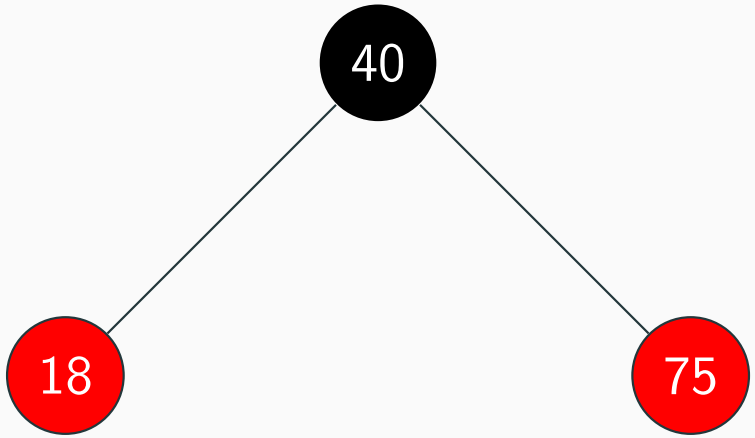
```
163
164 public:
165     void erase(const T& info)
166     {
167         Node *N = find(root, info);
168
169         if (N == nullptr)       // Caso trivial
170             return;
171
172         // Reduz o caso 3 ao caso 1 ou ao caso 2
173         if (N->left and N->right)
174             N = swap_info(N);
175
176         erase(N);
177     }
178
```

Remoção de nó vermelho

- Se N é vermelho, pela propriedade 2 seu filho C deve ser preto
- A substituição de N por C mantém as propriedades da árvore *red-black*
- Primeiramente, se C vir a ocupar a raiz da árvore, esta será preta, mantendo a propriedade 2
- O pai P de N é necessáriamente preto, uma vez que N é vermelho, de modo que a propriedade 4 não é violada
- Por fim, como o nó removido é vermelho, o número de nós pretos nos caminhos não se altera, mantendo a propriedade 5
- Observe que este caso só ocorre se ambos filhos de N são folhas (o caso de apenas uma folha violaria a propriedade 5 de uma árvore *red-black*)

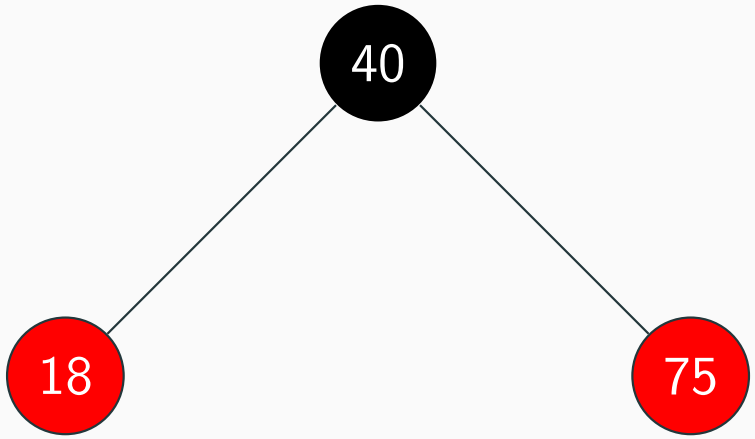
Exemplo de remoção de nó vermelho

Informação a ser removida: 40



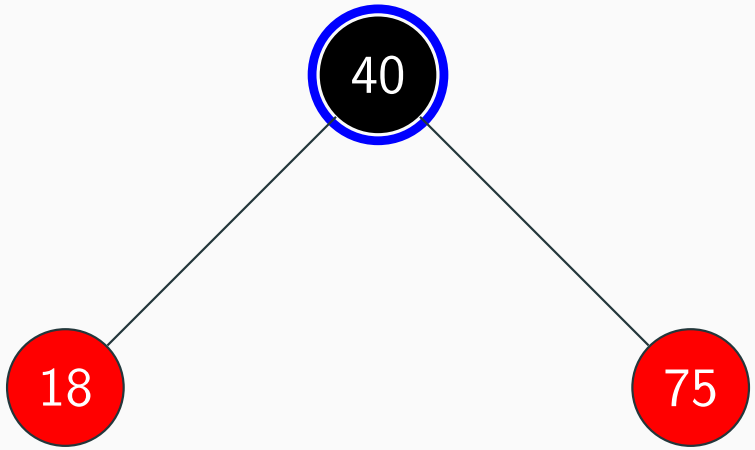
Exemplo de remoção de nó vermelho

40 tem dois filhos que não são folhas



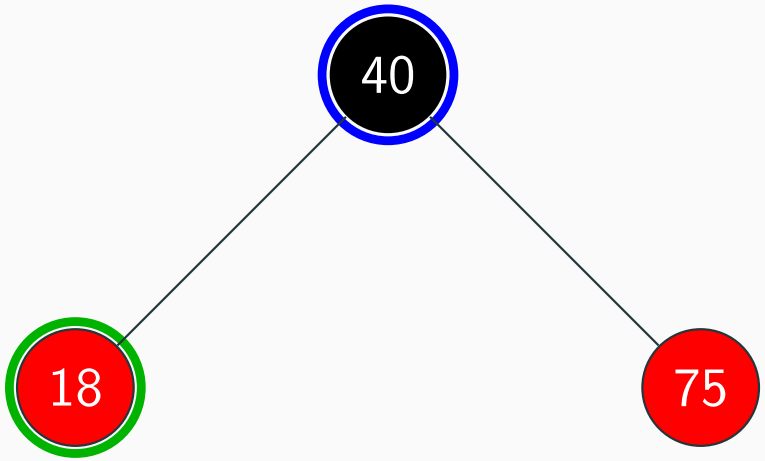
Exemplo de remoção de nó vermelho

Redução ao caso 1 ou ao caso 2



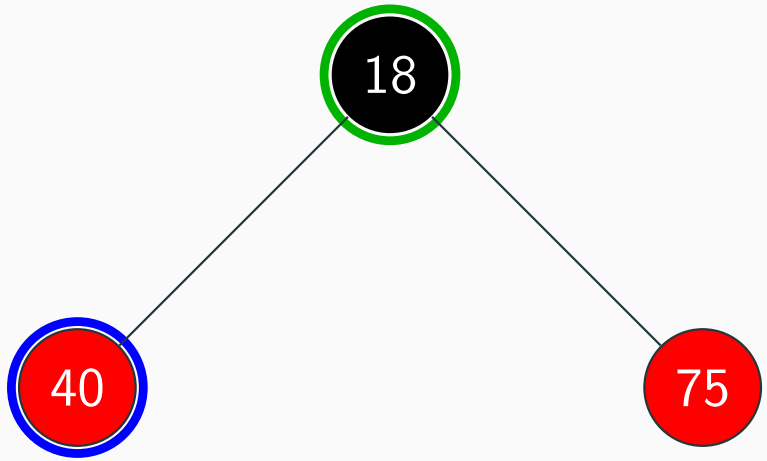
Exemplo de remoção de nó vermelho

Redução ao caso 1 ou ao caso 2



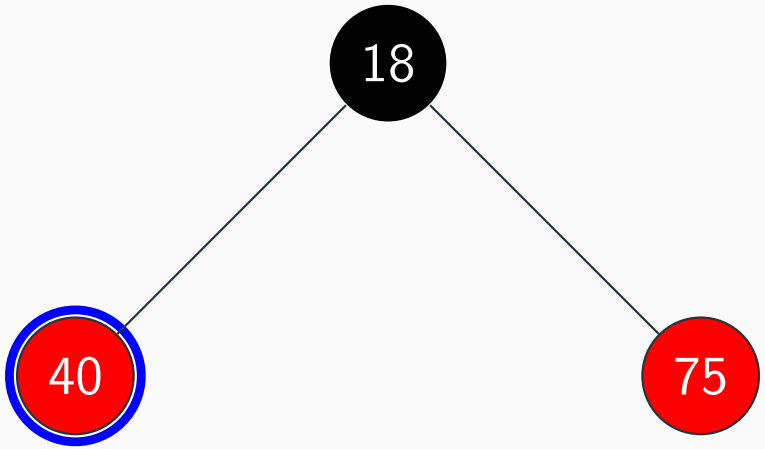
Exemplo de remoção de nó vermelho

Redução ao caso 1 ou ao caso 2



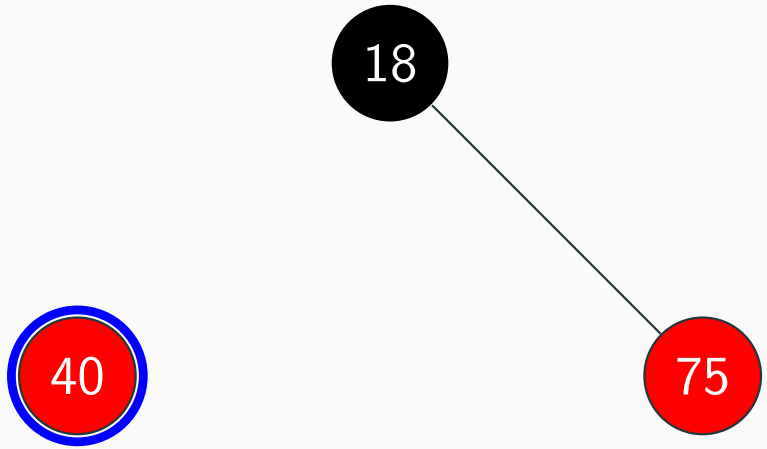
Exemplo de remoção de nó vermelho

Remoção de nó vermelho sem filhos que não sejam folhas



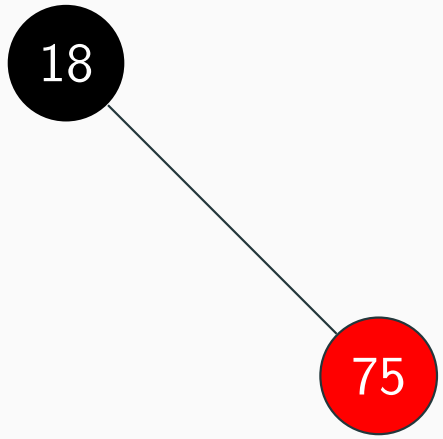
Exemplo de remoção de nó vermelho

Remoção de nó vermelho sem filhos que não sejam folhas



Exemplo de remoção de nó vermelho

Remoção de nó vermelho sem filhos que não sejam folhas



Implementação da remoção de nó vermelho

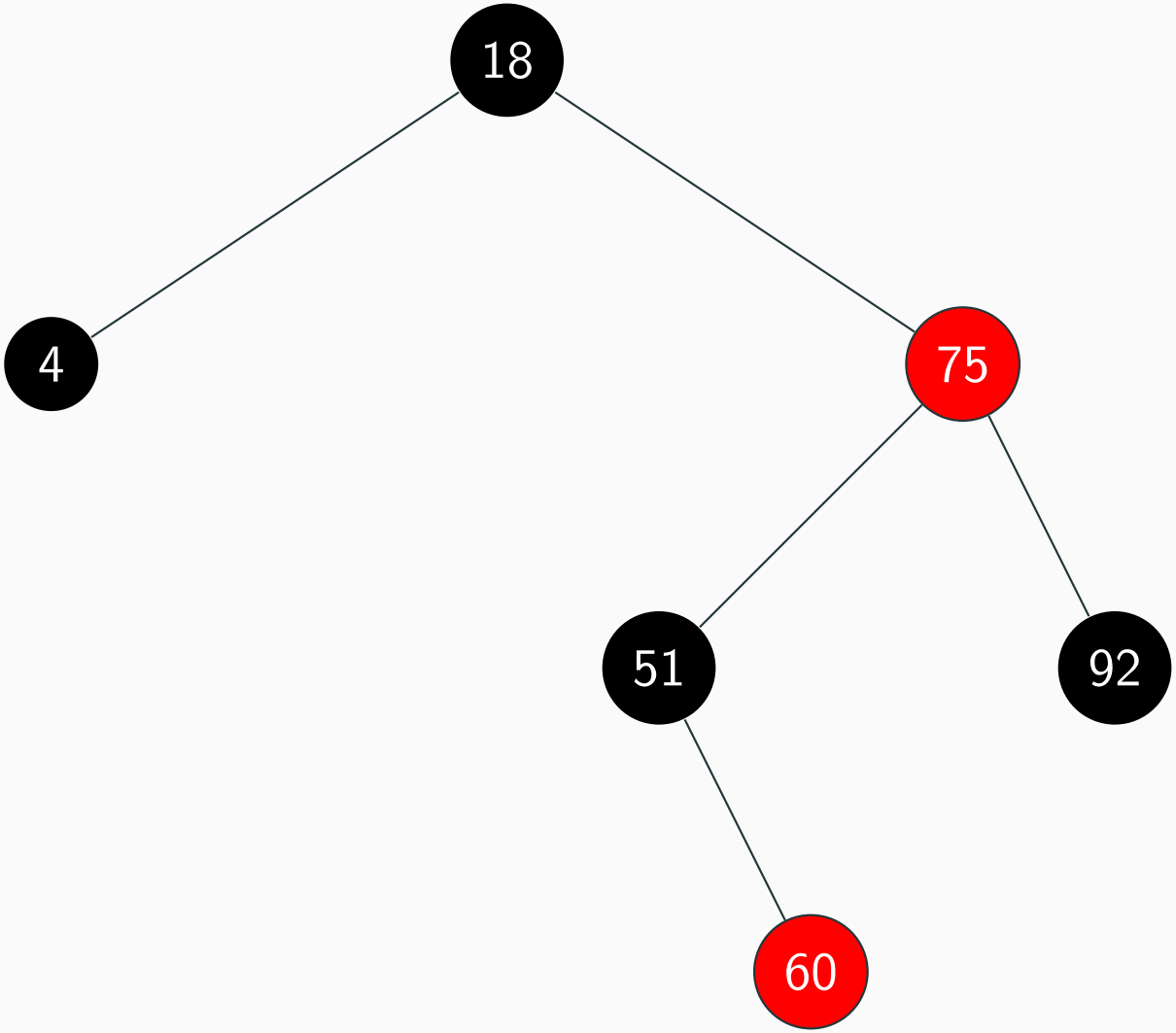
```
194
195     void erase(Node *N)
196     {
197         auto C = N->left ? N->left : N->right;
198
199         swap_nodes(N, C);
200
201         // Se N é vermelho não há nada mais a fazer
202         if (N->color == Node::RED)
203         {
204             delete N;
205             return;
206         }
207
```

Remoção de nó preto com filho vermelho

- Se N é preto e seu filho C é vermelho, a remoção de N viola a propriedade 5, devido a redução no número de nós pretos
- Além disso, a promoção de um nó vermelho pode levar a violação da propriedade 4
- Uma forma de preservar ambas propriedades é recolorir C como um nó preto
- Isto restaura a violação da propriedade 5, pois a perda de N agora é compensada com a adição de um novo nó preto
- O fato de C assumir a cor preta evita que a propriedade 4 seja violada

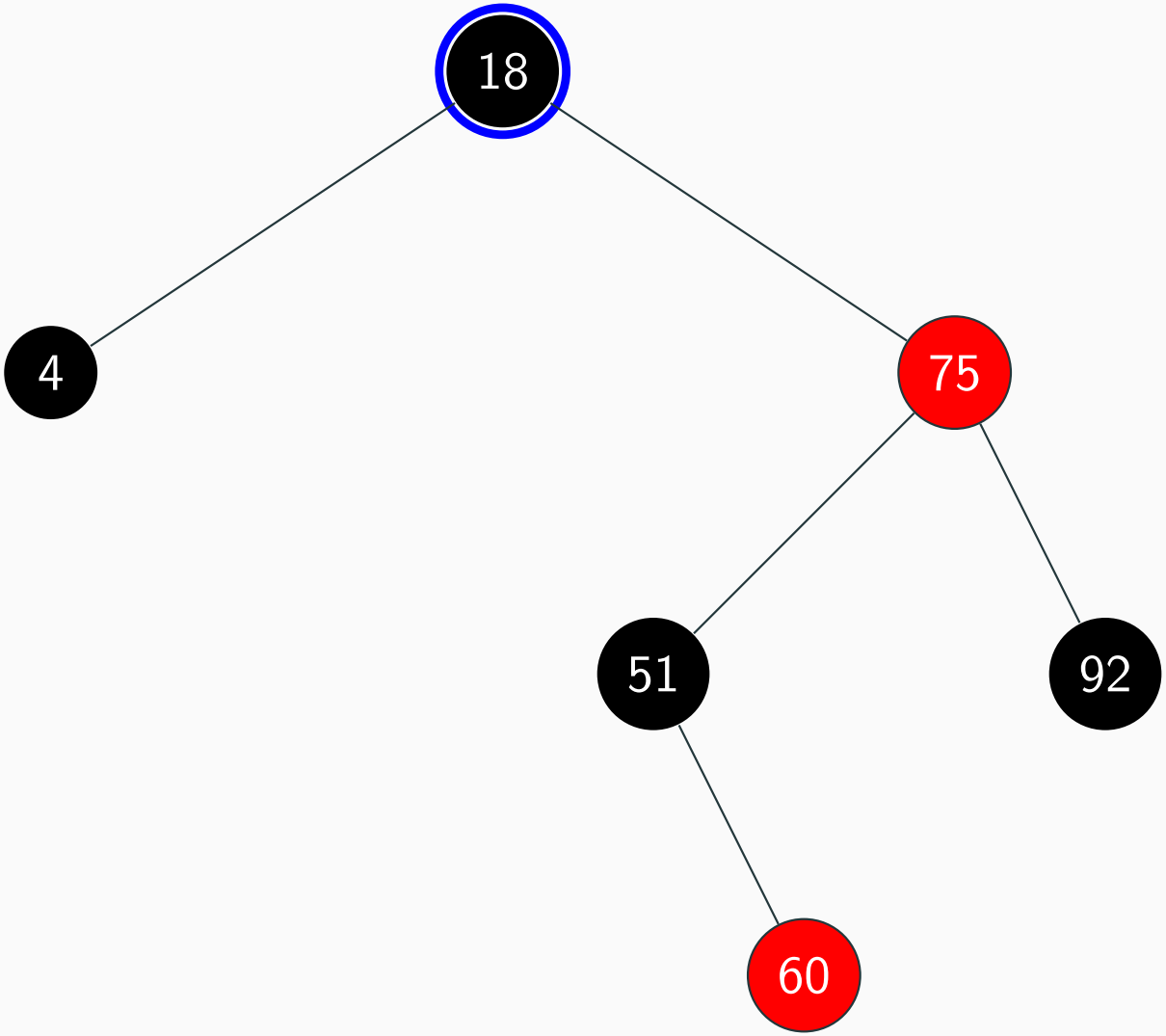
Exemplo de remoção de nó vermelho

Informação a ser removida: 51



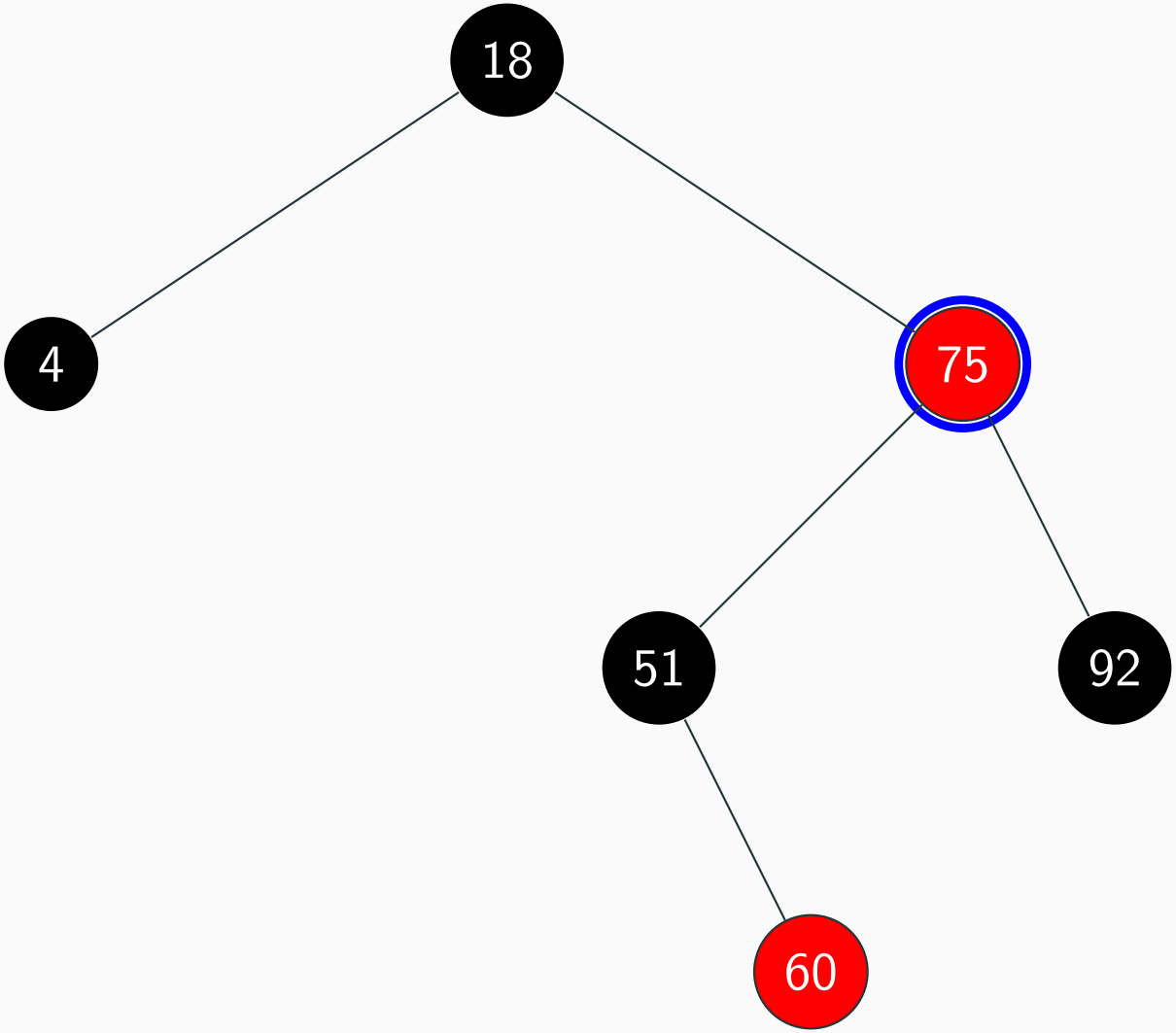
Exemplo de remoção de nó vermelho

Informação a ser removida: 51



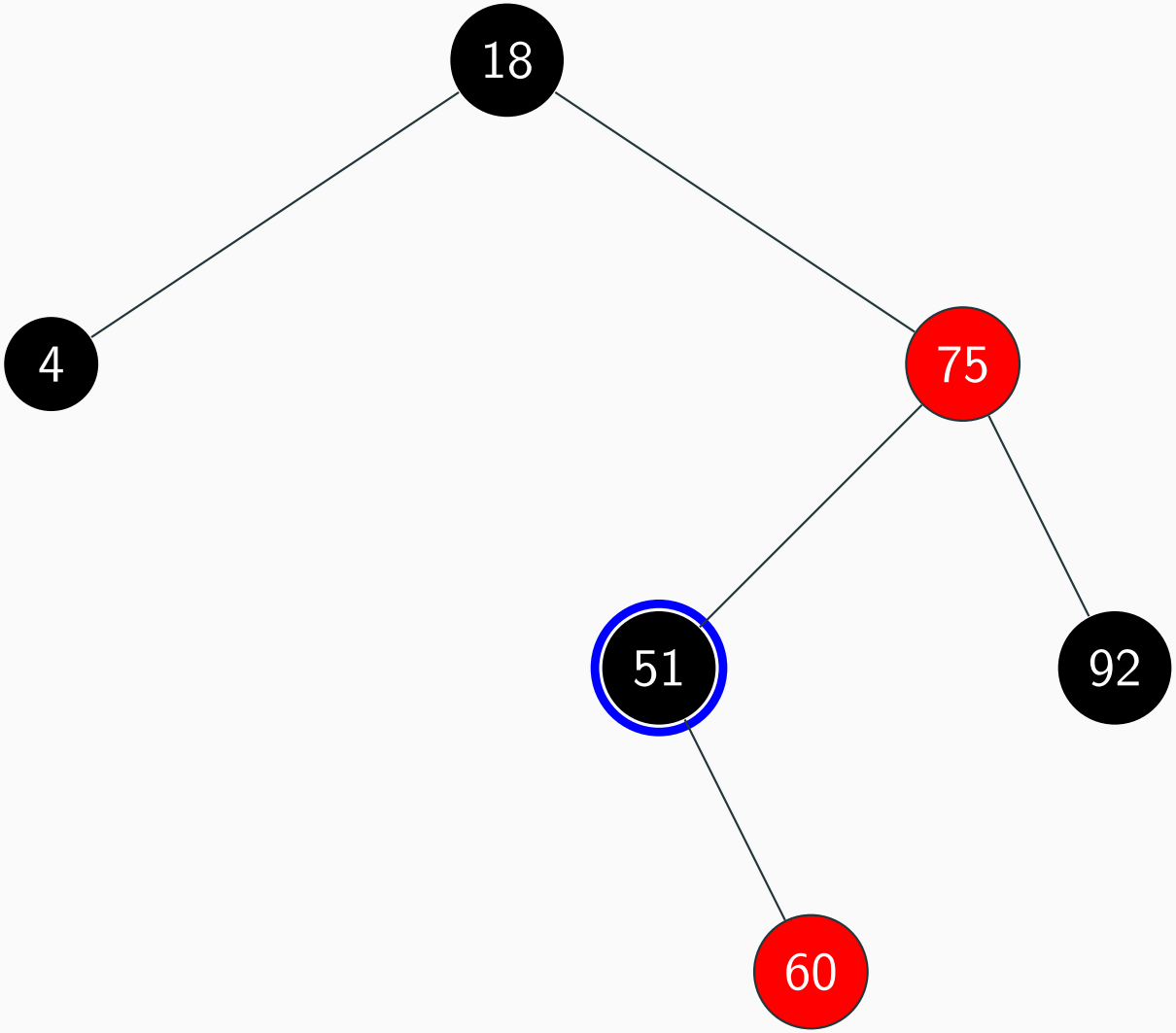
Exemplo de remoção de nó vermelho

Informação a ser removida: 51



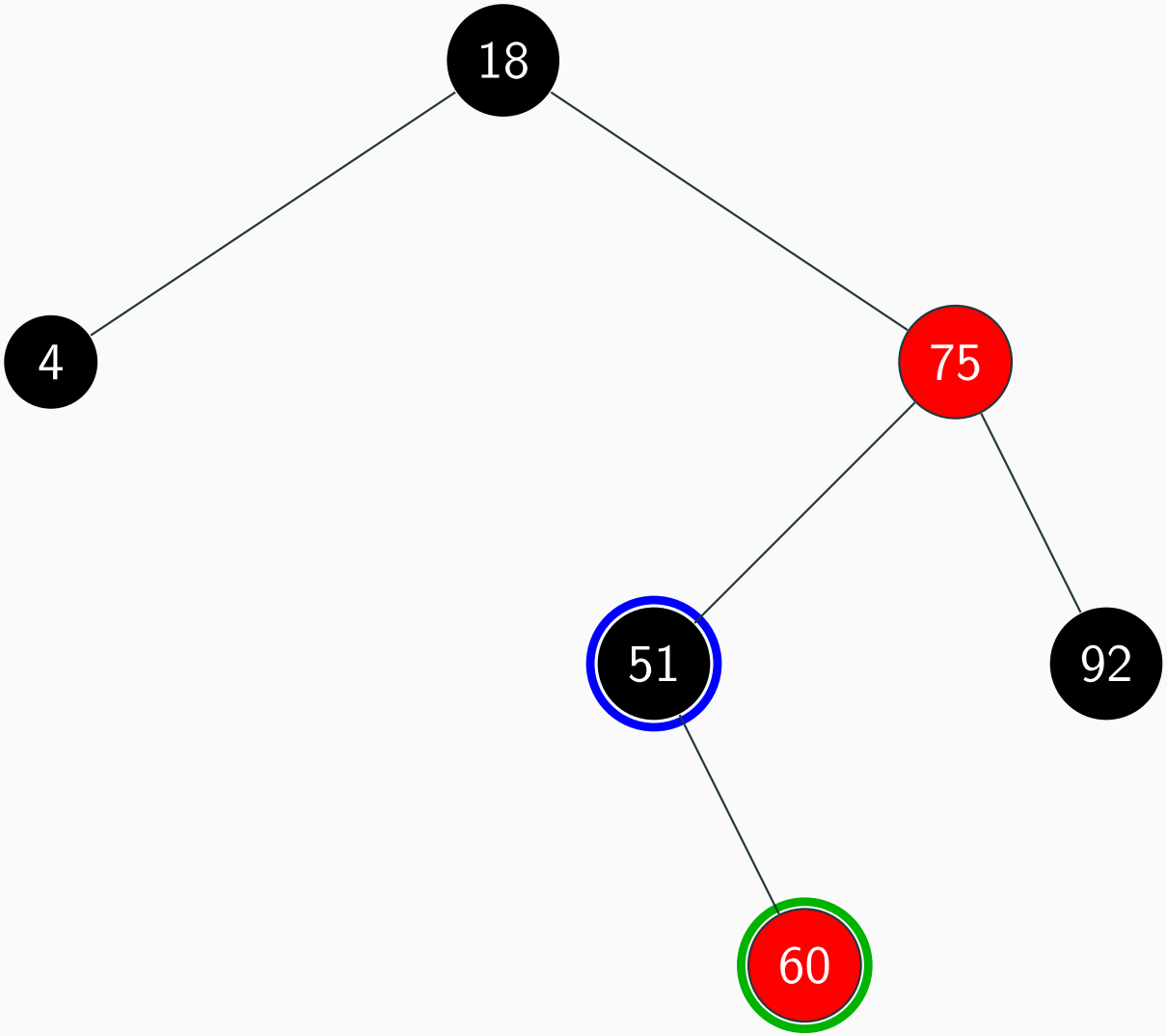
Exemplo de remoção de nó vermelho

Informação a ser removida: 51



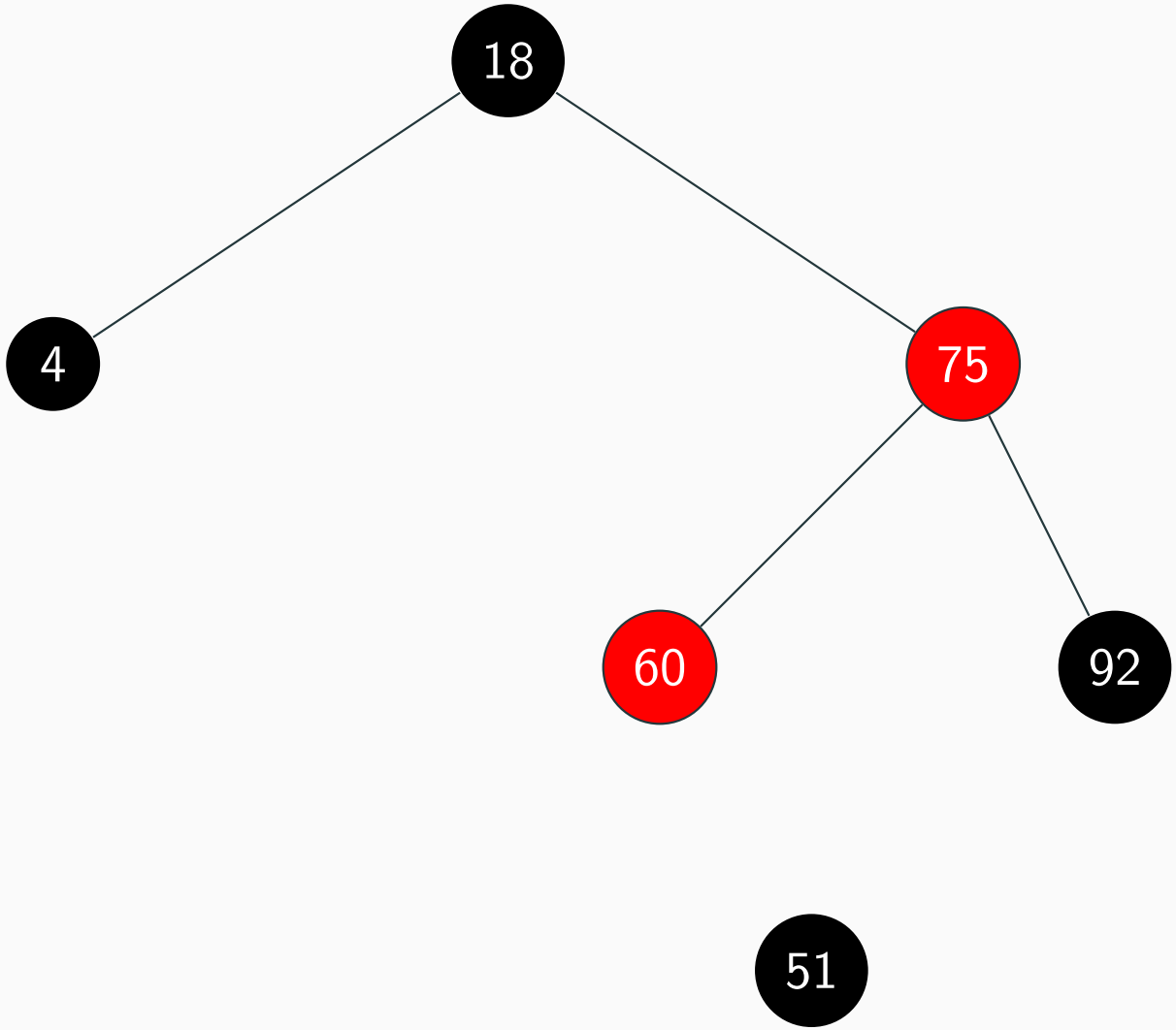
Exemplo de remoção de nó vermelho

Nó preto com filho vermelho



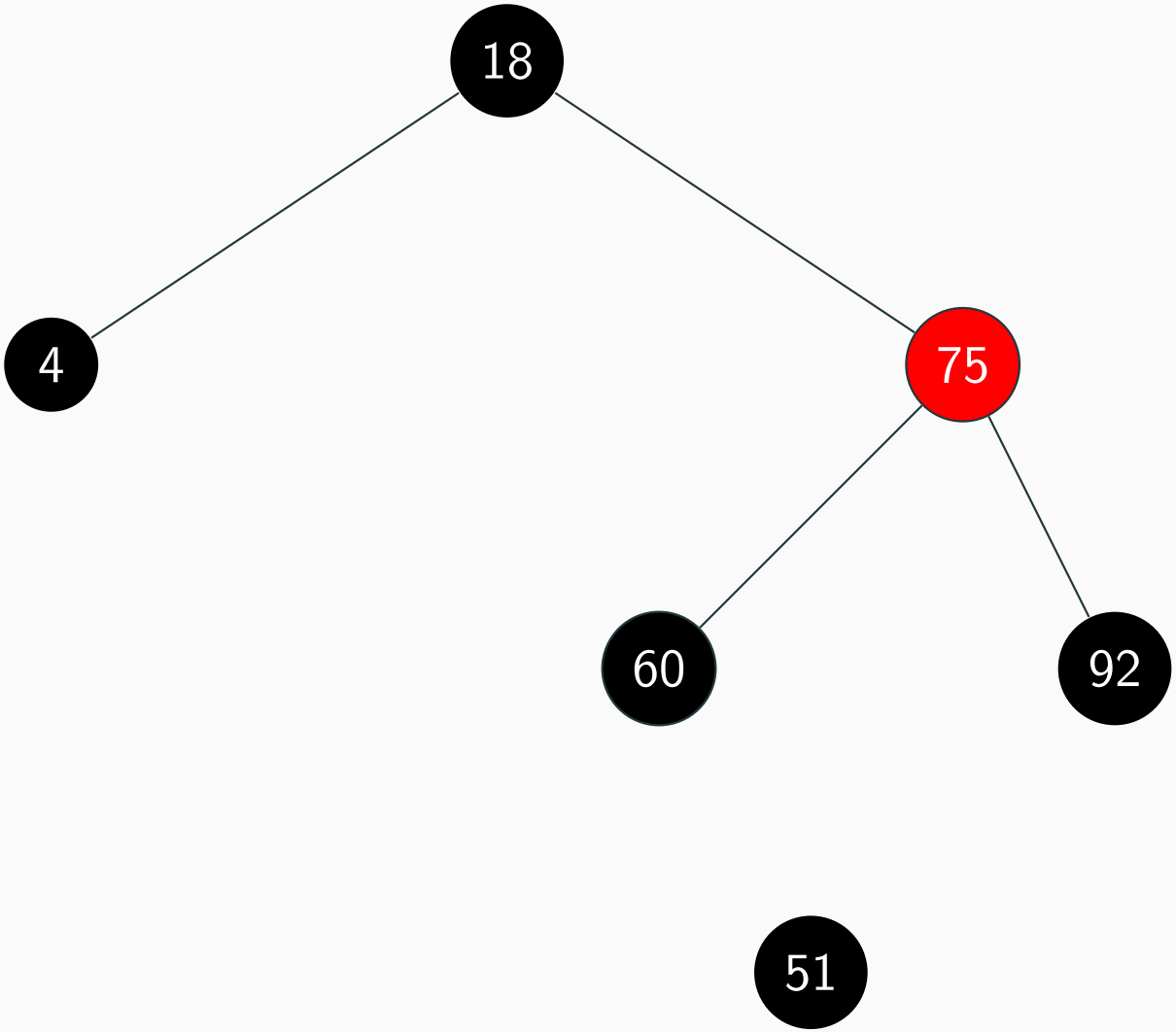
Exemplo de remoção de nó vermelho

Nó preto com filho vermelho



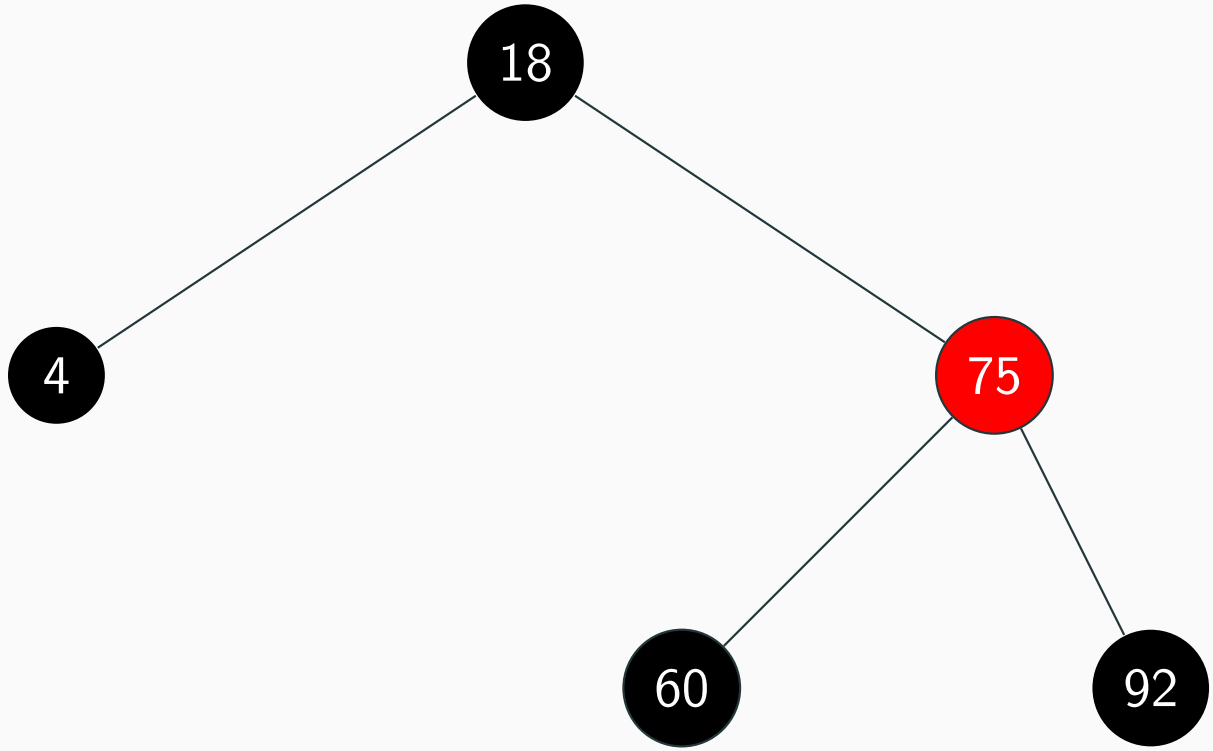
Exemplo de remoção de nó vermelho

Nó preto com filho vermelho



Exemplo de remoção de nó vermelho

Nó preto com filho vermelho



Implementação da remoção de nó preto com filho vermelho

```
208         // N é preto e o filho não-folha C é vermelho
209         if (C and C->color == Node::RED)
210         {
211             C->color = Node::BLACK;
212             delete N;
213             return;
214         }
215
```

Nó preto com filho preto

- O caso onde ambos N e C são pretos é o mais complexo dentre todos os que envolvem um nó com, no máximo, um filho não-folha
- A remoção de um nó preto viola a propriedade 5 das árvores *red-black*
- Há múltiplos cenários possíveis, cada um tendo que ser tratado adequadamente
- Observe que este caso ocorre apenas quando ambos filhos de N são folhas
- Isto porque se N tivesse apenas uma folha preta, o fato do outro filho não ser folha violaria a propriedade 5

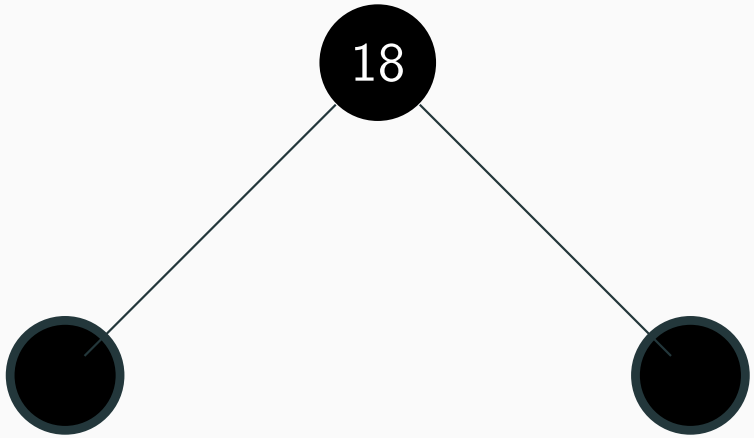
Cenário A: após a troca de N e C, C é raiz

- Este é o cenário mais simples
- A remoção de um nó preto da posição raiz subtrai igualmente uma unidade de todos os caminhos da raiz às folhas
- Assim a propriedade 5 fica preservada
- As demais propriedades também se mantém: como as folhas são pretas, a raiz também será preta

```
216         // N é preto, C é folha e assume a posição raiz
217         if (root == C)
218         {
219             delete N;
220             return;
221         }
222
```

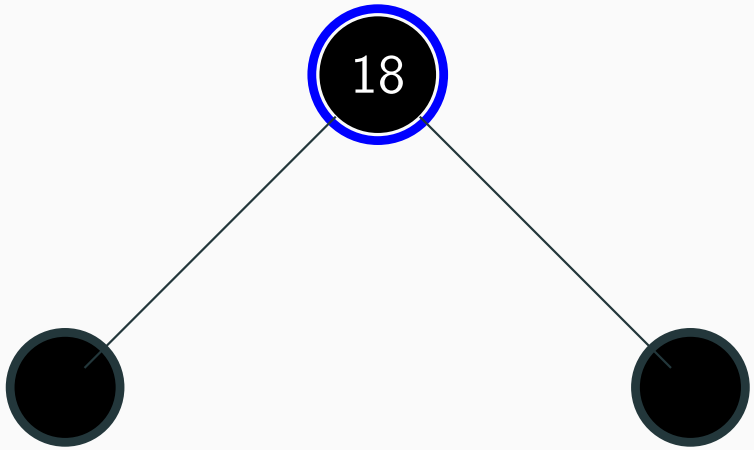
Exemplo de remoção no cenário A

Informação a ser removida: 18



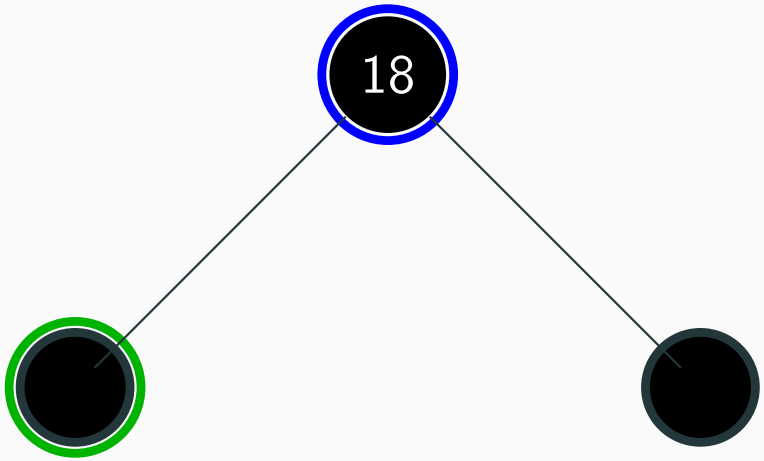
Exemplo de remoção no cenário A

Informação a ser removida: 18



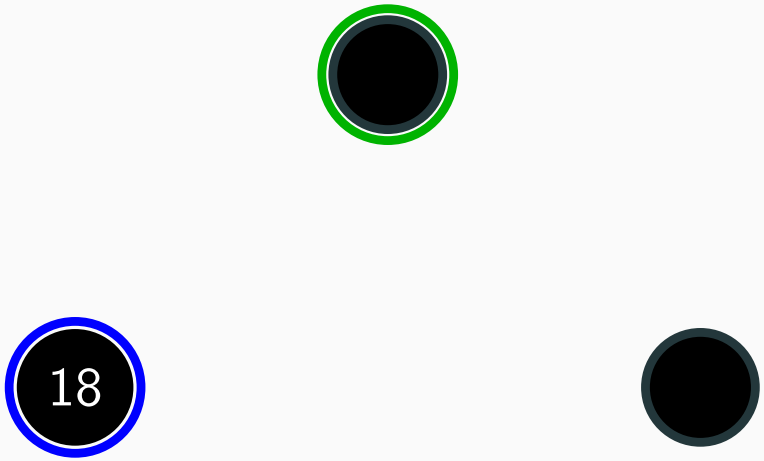
Exemplo de remoção no cenário A

Informação a ser removida: 18



Exemplo de remoção no cenário A

Informação a ser removida: 18



Exemplo de remoção no cenário A

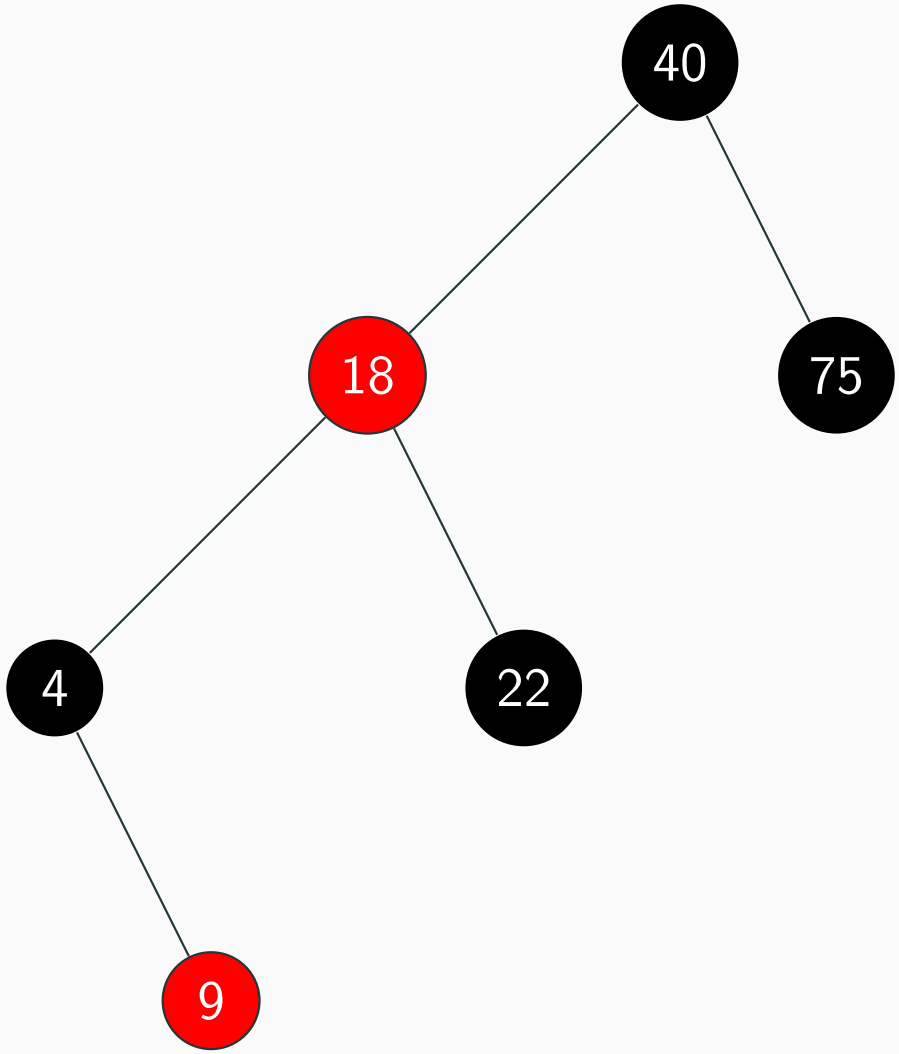
Informação a ser removida: 18

Cenário B: Irmão vermelho

- Se o nó a ser removido tem um irmão S vermelho, é preciso promover um reposicionamento dos nós, além de uma troca de cores entre o pai P e o irmão S
- Este cenário não pode ser resolvido diretamente: o resultado deste reposicionamento levará a um dos próximos cenários
- Primeiramente as cores de P e S devem ser trocadas
- Uma rotação de S em torno de P, o torna o novo avô de N
- Ao final deste processo uma das duas subárvores terá caminho da raiz até uma folha uma unidade menor do que a outra, violando a propriedade 5
- Porém o caso a ser tratado mudou: agora N tem pai vermelho, com irmão preto
- Este cenário será abordado mais adiante

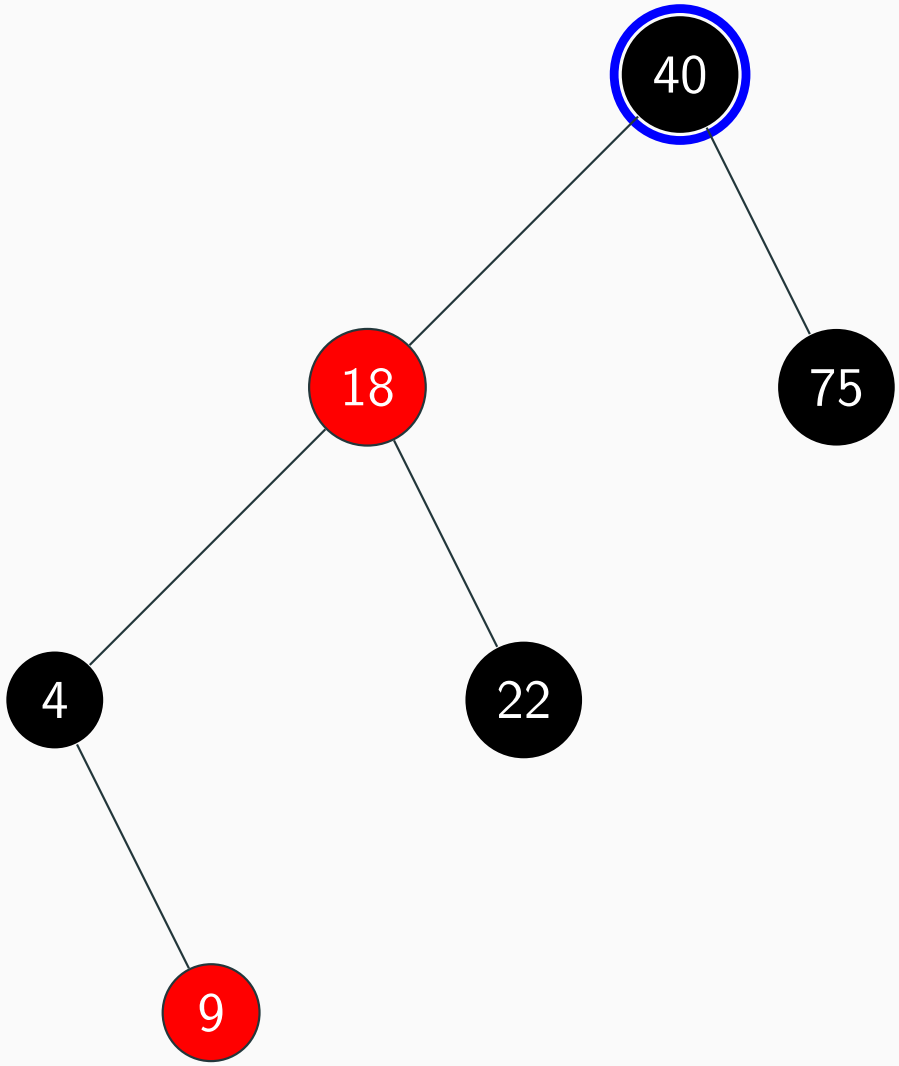
Exemplo de remoção no cenário B

Informação a ser removida: 75



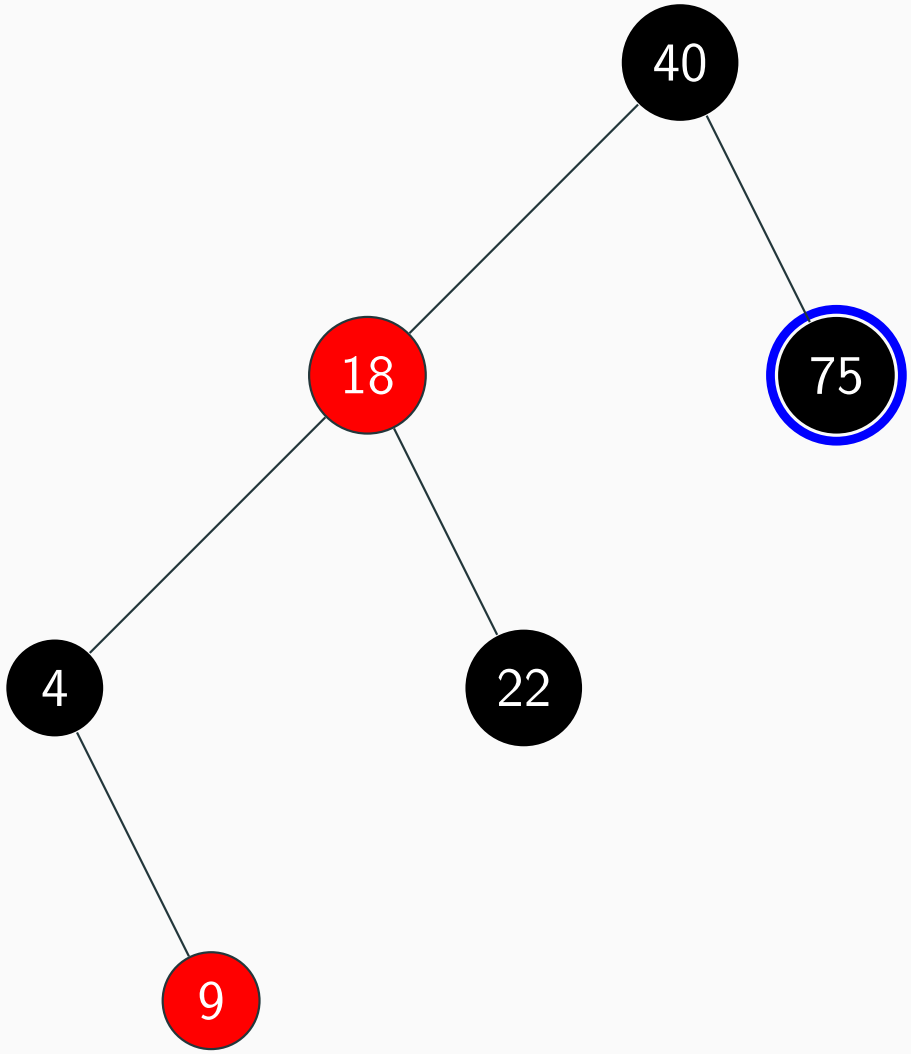
Exemplo de remoção no cenário B

Informação a ser removida: 75



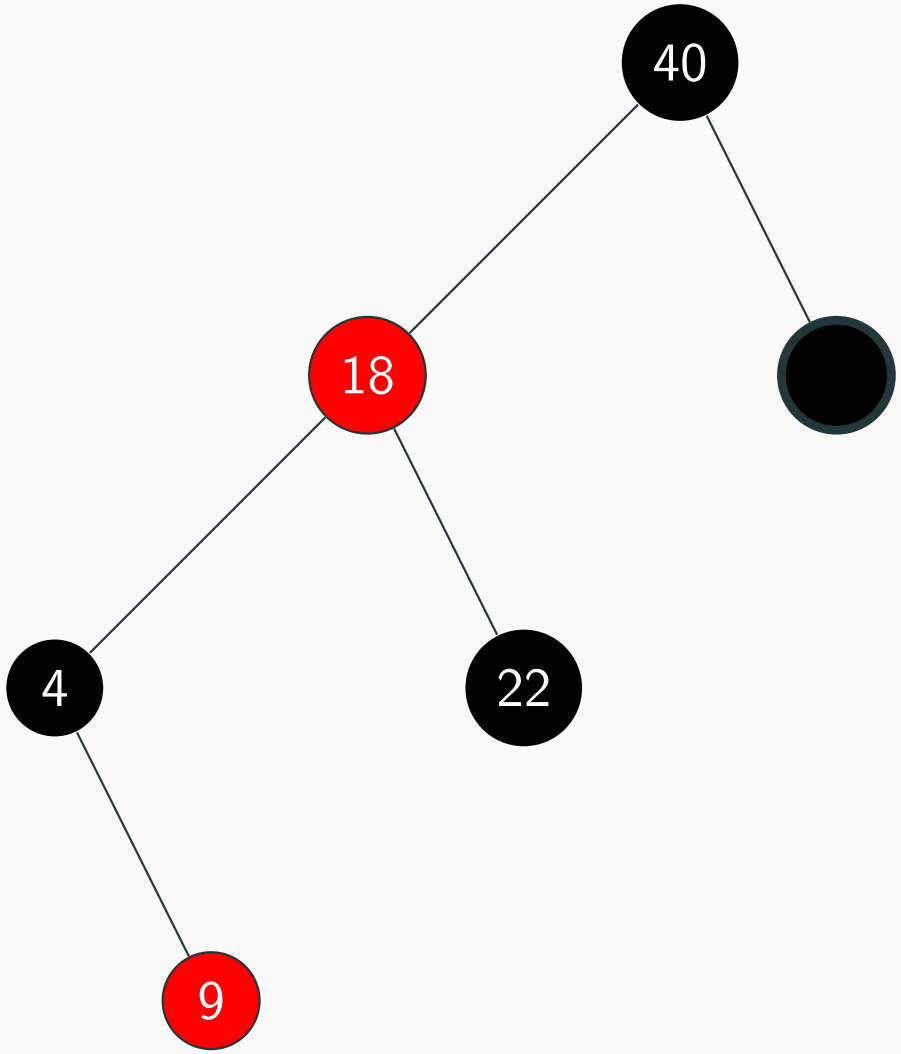
Exemplo de remoção no cenário B

Informação a ser removida: 75



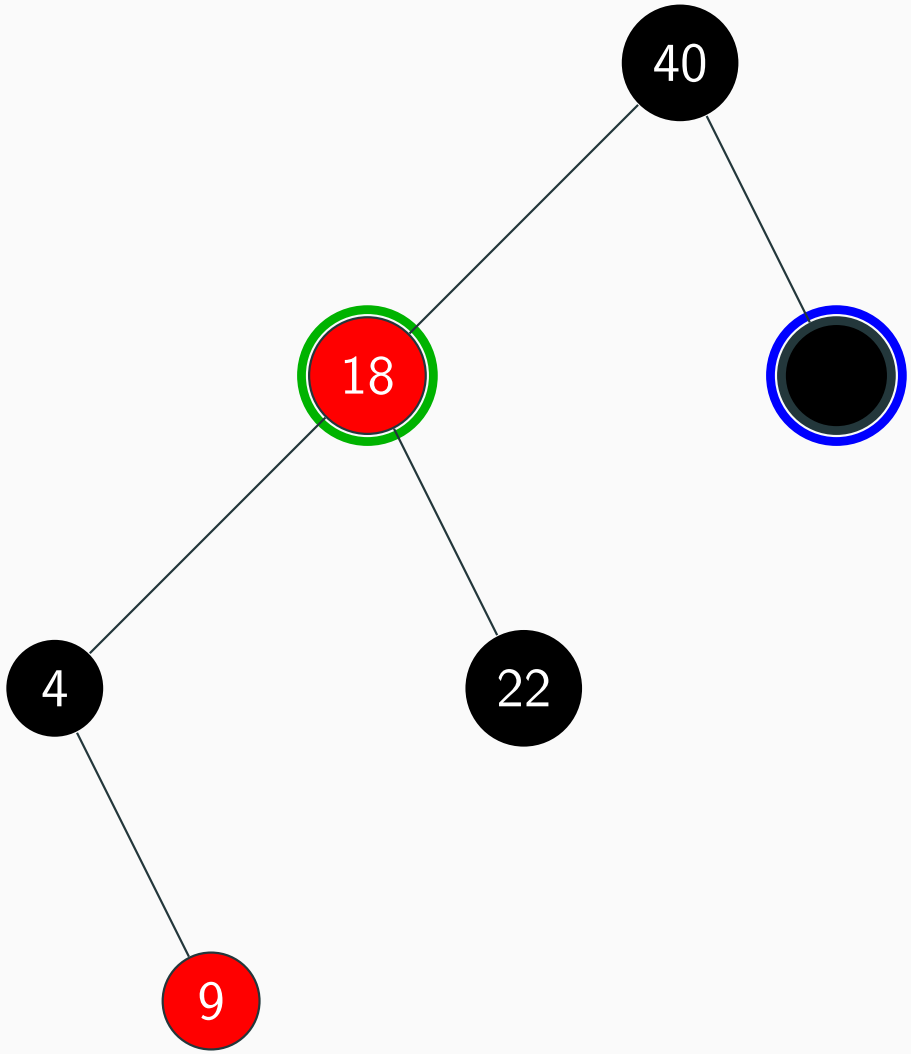
Exemplo de remoção no cenário B

Informação a ser removida: 75



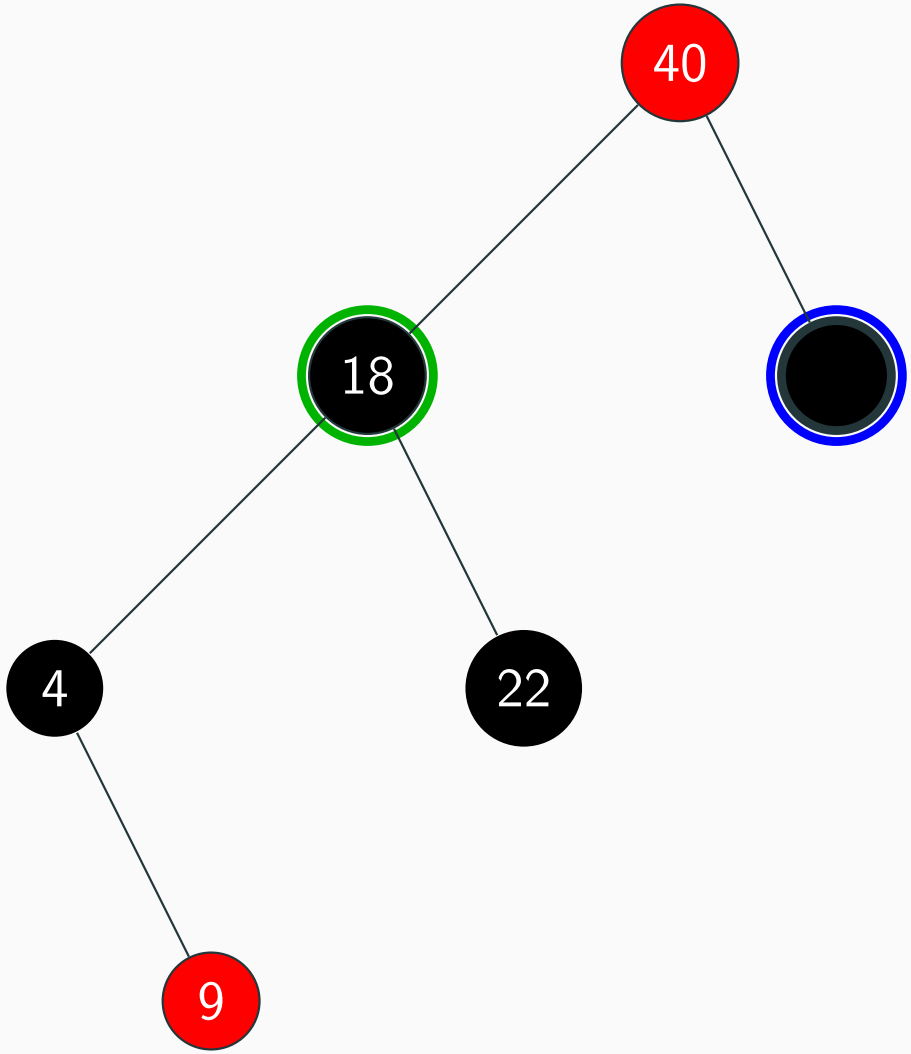
Exemplo de remoção no cenário B

Nó preto, irmão vermelho



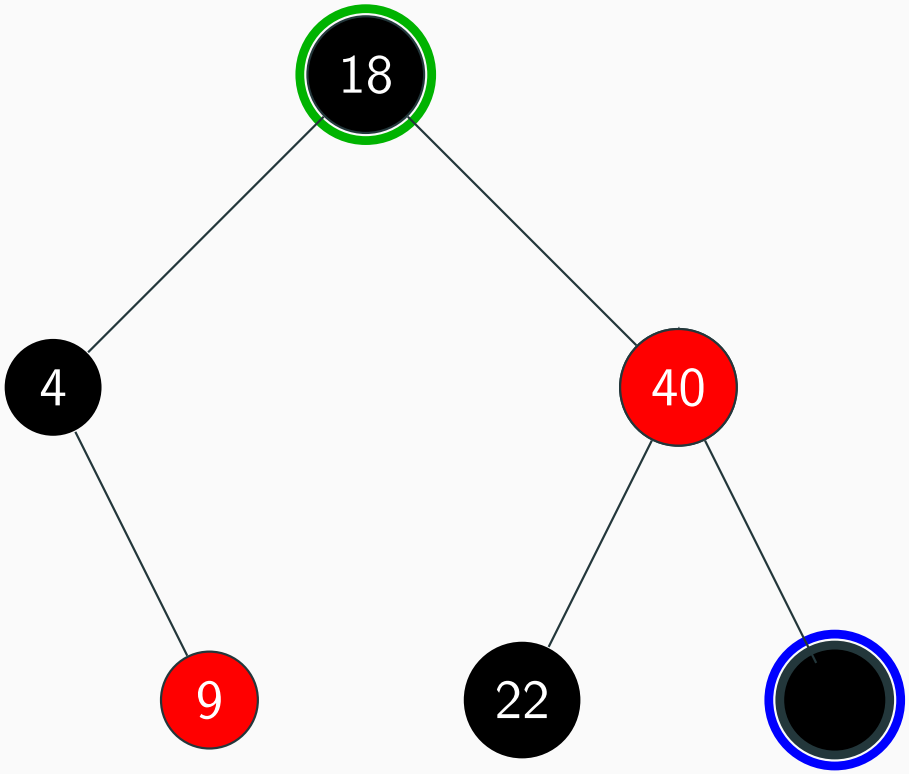
Exemplo de remoção no cenário B

Nó preto, irmão vermelho



Exemplo de remoção no cenário B

Nó preto, irmão vermelho



Implementação do cenário B

```
222
223         // N tem irmão S vermelho: este caso não se encerra no if
224         auto P = parent(N);
225         auto S = sibling(N);
226
227         if (S and S->color == Node::RED)
228         {
229             P->color = Node::RED;
230             S->color = Node::BLACK;
231
232             if (N == P->left)
233                 rotate_left(grandparent(S), P, S);
234             else
235                 rotate_right(grandparent(S), P, S);
236
237             if (parent(S) == nullptr)
238                 root = S;
239         }
240
```

Referências

1. Red-Black Trees, acesso em 27/03/2019.
2. Wikipédia. *Red-Black Tree*, acesso em 27/03/2019.[1]

[1] https://en.wikipedia.org/wiki/Red-black_tree