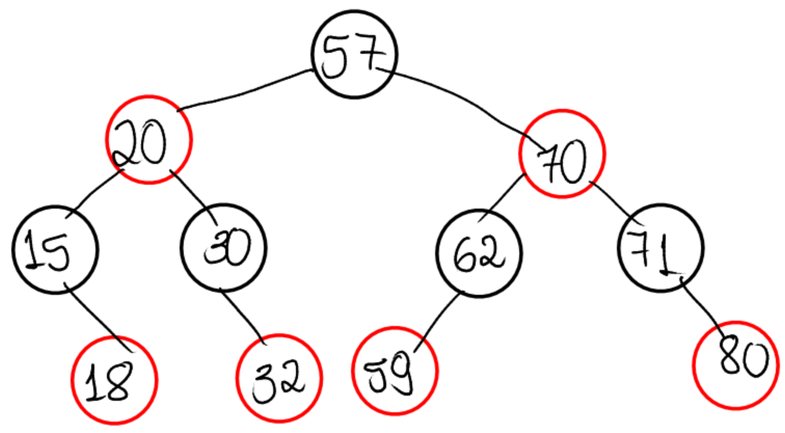
Algoritmo de remoção em árvore Rubro-negra

```
Remove(raiz, valor)
  Se busca(raiz, valor) == verdadeira, então:
    aux = raiz
    i = 0
    Pilha[altura_arvore)

    Comentário: Preenchendo a pilha
    Enquanto aux.valor != valor, faça:
      Pilha[i] = aux
      Se valor > aux.valor, então:
        aux = aux.direita
      Senão, então:
        aux = aux.esquerda
      i++

    Comentário: última posição da pilha que possui elemento
    pos = altura_arvore - qtd_pilha - 1
    aux_valor = aux
    pos_removido

    Comentário: se o nó a ser remvido for rubro
    Se aux.cor == rubro, então:

      Comentário: caso não tenha filhos
      Se aux.esquerda == NULL e aux.direita == NULL, então:
        Se pai.esquerda.valor == valor, então:
          Pai.esquerda == NULL
          Pos_removido = esquerda
        Senão, então:
          Pai.direita == NULL
          Pos_removido = direita
        free(aux)

      Comentário: caso tenha filho da esquerda
      Se aux.esquerda != NULL, então:
        Pai = Pilha[pos]
        aux = aux.esquerda
        Se aux.direita == NULL, então:
          Pai.esquerda = aux
          free(aux_valor)
        Senão, então:
          Enquanto aux.direita != NULL, faça:
            Pai = aux
            aux = aux.direita
          Se aux.esquerda == NULL, então:
            aux.esquerda = aux_valor.esquerda
            aux.direita = aux_valor.direita
            pai.direita = NULL
          Senão, então:
            Pai.direita = aux.esquerda
            aux.esquerda = aux_valor.esquerda
            aux.direita = aux_valor.direita
        Se Pilha[pos].direita.valor == valor, então:
          Pilha[pos].direita = aux
          Pos_removido = direita
        Senão, então:
          Pilha[pos].esquerda = aux
          Pos_removido = esquerda
      free(aux_valor)
```

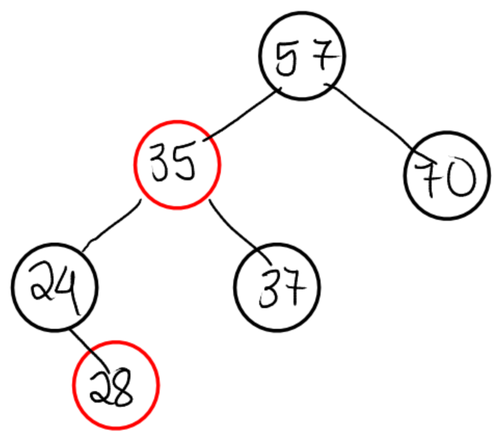


```
    Comentário: caso tenha não tenha filho esquerdo e tem direito
    Senão, se aux.direita != NULL, então:
        Pai = Pilha[pos]
        aux = aux.direita
        Se aux.esquerda == NULL, então:
            Pai.direita = aux
            free(aux_valor)
        Senão, então:
            Enquanto aux.esquerda != NULL, faça:
                Pai = aux
                aux = aux.esquerda
            Se aux.direita == NULL, então:
                aux.direita = aux_valor.direita
                aux.esquerda = aux_valor.esquerda
                Pai.esquerda = NULL
            Senão, então:
                Pai.esquerda = aux.direita
                aux.esquerda = aux_valor.esquerda
                aux.direita = aux_valor.direita
        Se Pilha[pos].direita.valor = valor, então:
            Pilha[pos].direita = aux
            Pos_removido = direita
        Senão, então:
            Pilha[pos].esquerda = aux
            Pos_removido = esquerda
        free(aux_valor)

Se aux.cor == negro, então:
```

Remoção de nó negro

```
    Comentário: caso não tenha filhos
    Se aux.esquerda == NULL e aux.direita == NULL, então:
        Se Pilha[pos].direita.valor == valor, então:
            Pilha[pos].direita = aux
            Pos_removido = direita
        Senão, então:
            Pilha[pos].esquerda = aux
            Pos_removido = esquerda
        free(aux_valor)
        aux = NULL
        Pai = Pilha[pos]

    Comentário: caso tenha filho da esquerda
    Se aux.esquerda != NULL, então:
        Pai = Pilha[pos]
        aux = aux.esquerda
        Se aux.direita == NULL, então:
            Pai.esquerda = aux
            free(aux_valor)
        Senão, então:
            Enquanto aux.direita != NULL, faça:
                pai = aux
                aux = aux.direita
            Se aux.esquerda == NULL, então:
                aux.esquerda = aux_valor.esquerda
                aux.direita = aux_valor.direita
                Pai.direita = NULL
            Senão, então:
                Pai.direita = aux.esquerda
                aux.esquerda = aux_valor.esquerda
                aux.direita = aux_valor.direita
        Se Pilha[pos].direita.valor == valor, então:
            Pilha[pos].direita == aux
            Pos_removido = direita
        Senão, então:
            Pilha[pos].esquerda = aux
            Pos_removido = esquerda
        free(aux_valor)
```

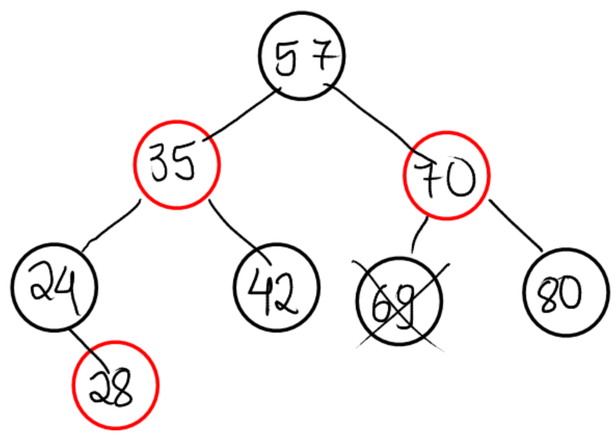
```
    Comentário: caso tenha não tenha filho esquerdo e tem direito
    Se aux.direita != NULL, então:
    | Pai = Pilha[pos]
    | aux = aux.direita
    | Se aux.esquerda == NULL, então:
    | | Pai.direita = aux
    | | free(aux_valor)
    | Senão, então:
    | | Enquanto aux.esquerda != NULL, faça:
    | | | Pai = aux
    | | | aux = aux.esquerda
    | | Se aux.direita == NULL, então:
    | | | aux.direita= aux_valor.direita
    | | | aux_esquerda = aux_valor.esquerda
    | | | Pai.esquerda = NULL
    | | Senão, então:
    | | | Pai.esquerda = aux.direita
    | | | aux.esquerda = aux_valor.esquerda
    | | | aux.direita = aux_valor.direita
    | | Se Pilha[pos].direita.valor == valor, então:
    | | | Pilha[pos].direita = aux
    | | | Pos_removido = direita
    | | Senão, então:
    | | | Pilha[pos].esquerda = aux
    | | | Pos_removido = esquerda
    | | free(aux_valor)
```

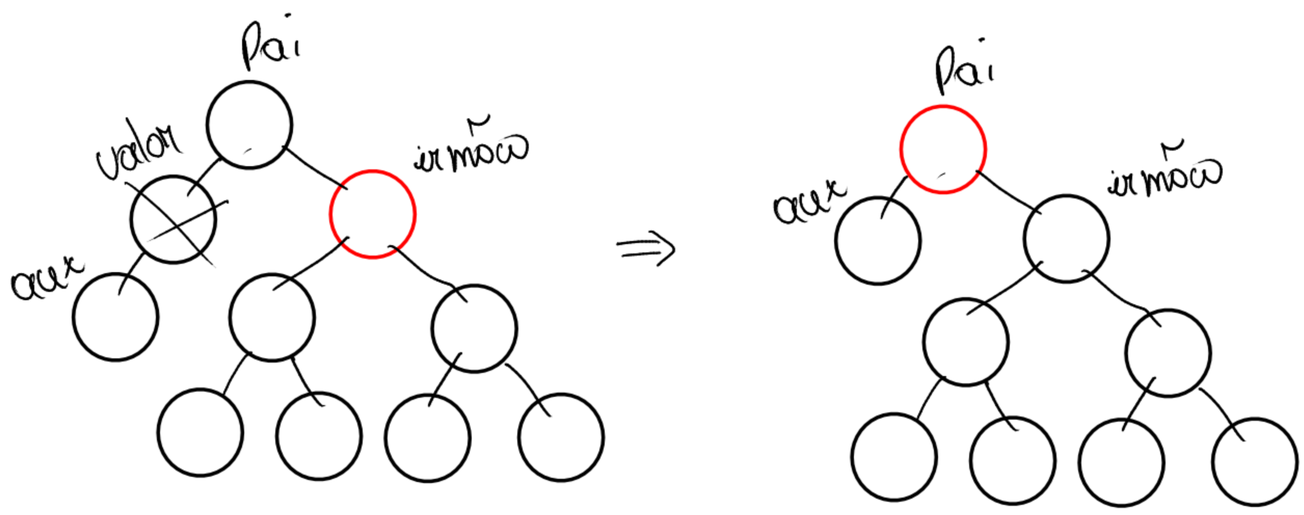


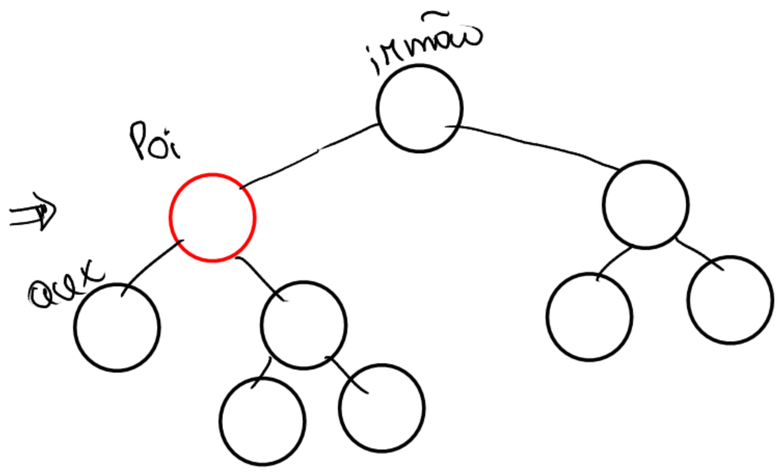

```
    Comentário: tornando a árvore rubro-negra novamente, se necessário
    Se aux.cor == rubro, então:
    | aux.cor = negro

    Senão, então:
    | Para i = pos, enquanto i > 0, faça:
    | | Pai = pilha[pos]
    | | irmão

    | Se i = pos, então:
    | | Se pos_removido == esquerda, então:
    | | | irmão = Pilha[pos].direita
    | | Senão, então:
    | | | irmão = Pilha[pos].esquerda
    | Senão, então:
    | |
    | | Comentário: caso tenha removido na esquerda
    | | Se Pilha[pos-1].esquerda == aux.valor, então:
    | | | irmão = Pilha[pos-1].direita
    | | |
    | | | Se aux.pai == NUll, então:
    | | | | pare
    | | | Senão, então:
    | | | |
    | | | | Comentário: caso 1
    | | | | Se irmão.cor == rubro, então:
    | | | | | Pai.cor = rubro
    | | | | | irmão.cor = negro
    | | | | | RSE(pai)
    | | | | | atualiza_irmão(aux.pai)
```

se o substituto for rubro

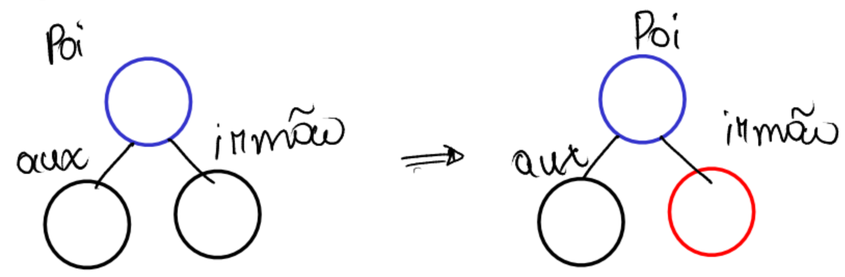
se o substituto for negro



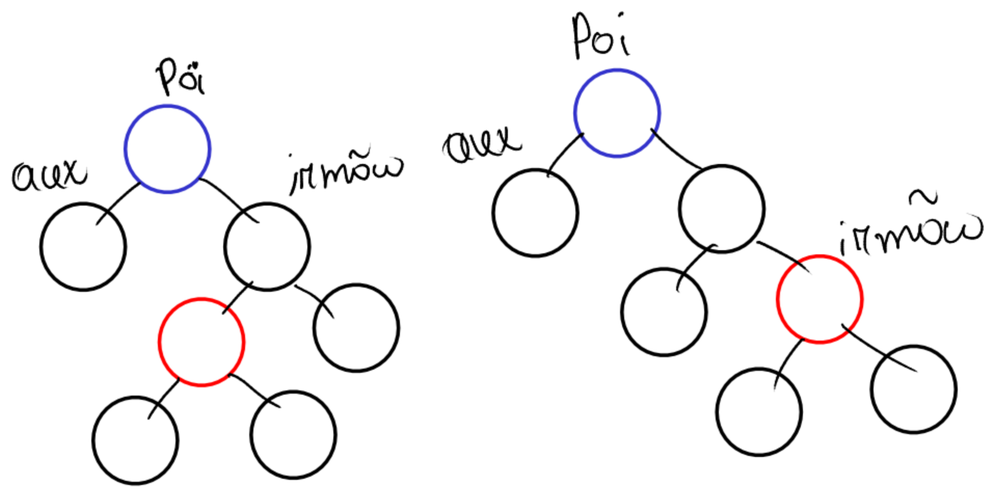




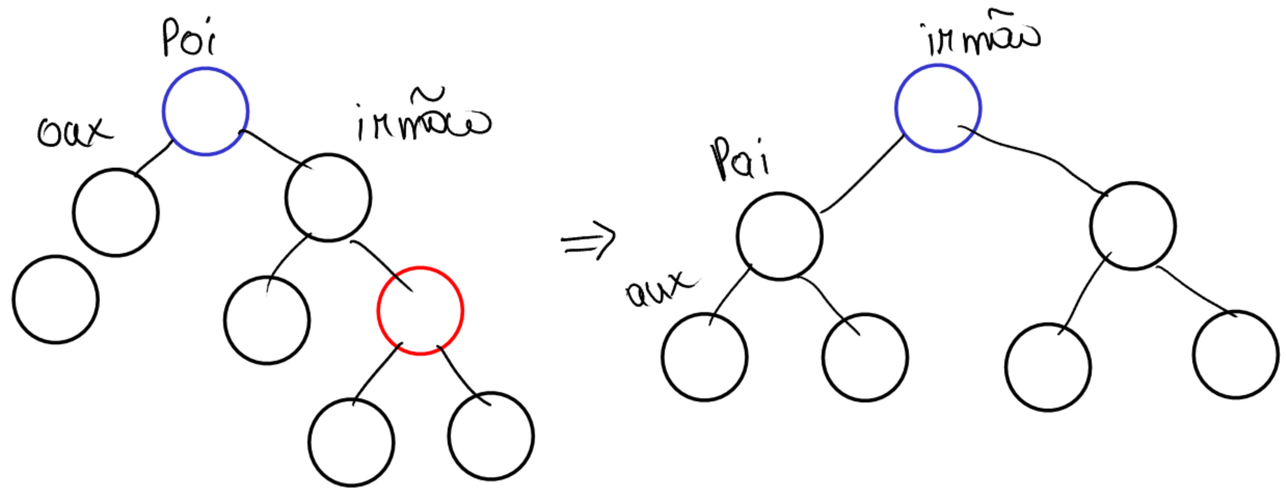
```
Se irmão.cor == negro, então:
    Comentário: caso 2
    Se irmão.direita.cor == negro e irmão.esquerda.cor == negro, então:
        irmão.cor = rubro
        aux = pai

        Se pai.cor == rubro, então:
            pai.cor = negro
        Senão, então:
            continue
```




```
    Comentário: caso 3
    Senão, se irmão.direito.cor == negro, então:
        irmão.esquerdo.cor = negro
        irmão.cor = rubro
        RSD(irmão)
        atualiza_irmão(irmão.pai)
```




```
    Comentário: caso 4
    Senão, se irmão.direito.cor == Rubro,então:
        irmão.cor = pai.cor
        pai.cor = negro
        irmão.direita.cor = negro
        RSE(pai)
        Se irmão.cor == rubro, então:
            irmão.cor = negro
```




```
Senão, então:
    irmão = Pilha[pos-1].esquerda

    se aux.pai == NULL, então:
        pare
    Senão, então:

        Comentário: caso 1
        Se irmão.cor == rubro, então:
            pai.cor = rubro
            irmão.cor = negro
            RSD(pai)
            atualiza_irmão(aux.pai)


        Se irmão.cor == negro, então:

            Comentário: caso 2
            Se irmão.esquerda.cor == negro e irmão.direita.cor == negro, então:
                irmão.cor = rubro
                aux = pai
                se pai.cor == rubro:
                    pai.cor = negro
                Senão, então:
                    continue

            Comentário: caso 3
            Senão, se irmão.esquerdo.cor == negro, então:
                irmão.direito.cor = negro
                irmão.cor = rubro
                RSE(irmão)
                atualiza_irmão(irmão.pai)

            Comentário: caso 4
            Senão, se irmão.esquerdo.cor == rubro, então:
                irmão.cor = pai.cor
                pai.cor = negro
                irmão.esquerdo.cor = negro
                RSD(pai)
                se irmão.cor == rubro, então:
                    irmão.cor = negro
```

```
        free(Pilha[altura_arvore])
        free(pai)
        free(irmão)
        frre(aux)

    Senão, então:
      Mensagem: "O valor não existe na arvore!"
```