Implementação

Sim, mas agora é preciso ver como as coisas funcionam na prática.

O primeiro passo é definir o novo registro

```
struct RegIntRN
     int                num
     int                cor
     struct RegIntRN    *esq, *dir
```

Daí, a gente nota que existem 3 casos para a operação de inserção

1. filho vermelho de um pai preto
2. filho vermelho de um pai vermelho, que é o único filho vermelho do avô
3. filho vermelho de um pai vermelho, onde o avô tem outro filho vermelho

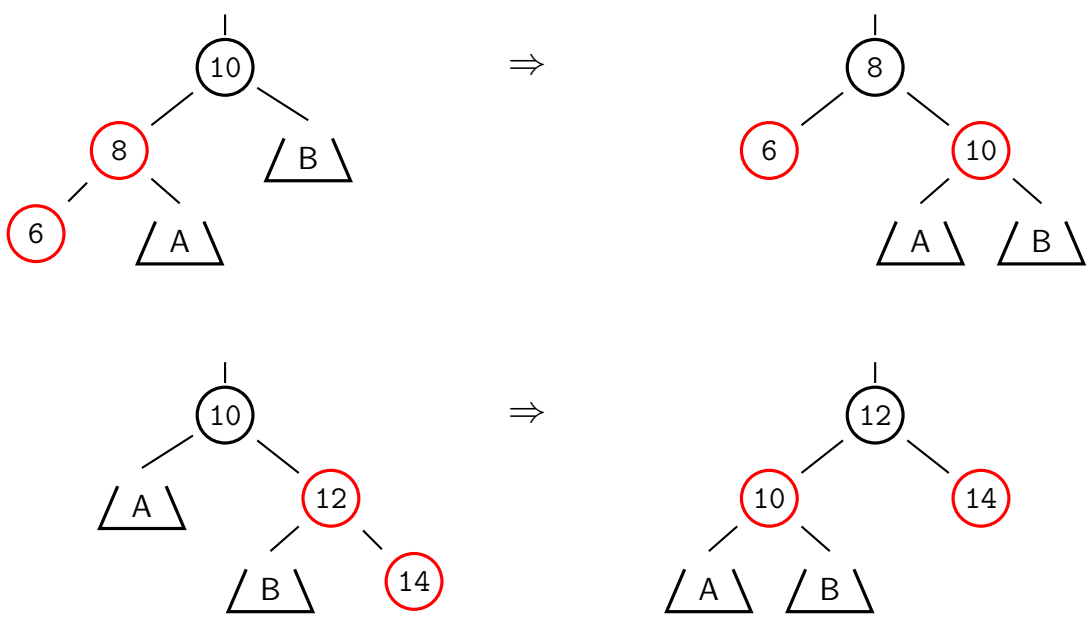
No primeiro caso não há nada a fazer.

E no terceiro caso só o que acontece é uma troca de cores — (*o que pode propagar a ação para cima*)

Então, o único caso que envolve uma pouco mais de trabalho é o segundo.

E no segundo caso, existe um subcaso simples e um subcaso mais complicado.

O subcaso simples é o seguinte



Quer dizer, nos dois casos o pai toma o lugar do avô.

E nós vamos escrever funções auxiliares que implementam essas operações.

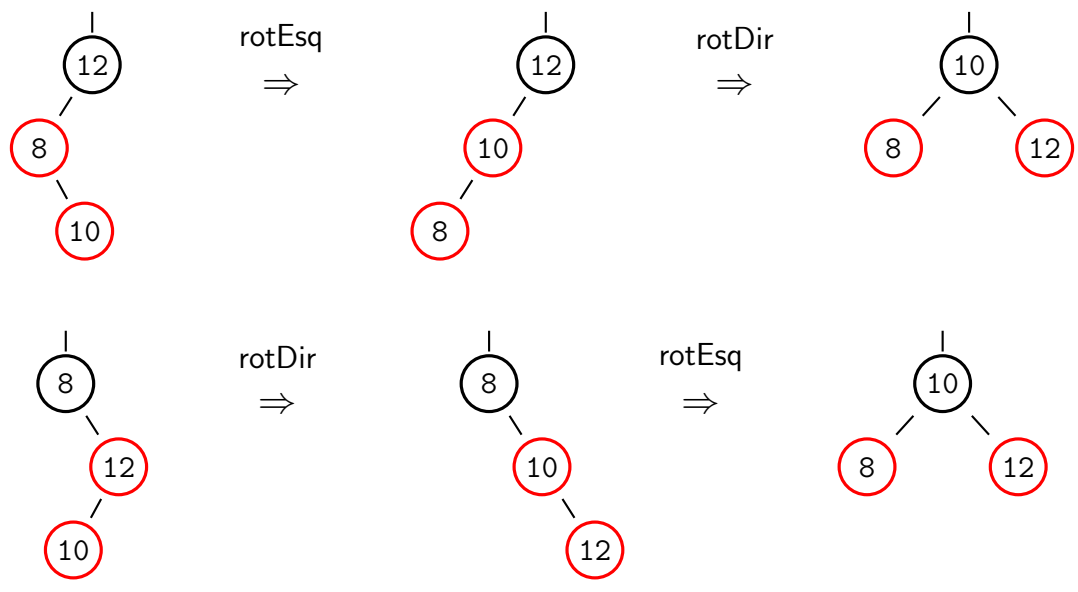
Nos livros, essas operações são chamadas de *rotação*.

E nós vamos utilizar o mesmo nome aqui

```
func rotDir (pai,avo)

      m = pai->dir
      pai->dir = avo                    // o pai toma o lugar do avô
      avo->esq = m

      Retorna (pai)


func rotEsq (pai,avo)

      m = pai->esq
      pai->esq = avo                    // o pai toma o lugar do avô
      avo->dir = m

      Retorna (pai)
```

Legal.

Agora nós podemos ver que o outro caso se resolve com duas rotações



Agora, a gente pode esboçar o procedimento de inserção em três etapas

1. primeiro nós fazemos a descida na árvore até o local da inserção
2. daí, o novo elemento é inserido na árvore — (*com a cor vermelha*)
3. e depois, nós subimos ajustando as coisas para que não exista um pai vermelho com um filho vermelho

E observando como os ajustes são feitos, a gente nota que é mais fácil implementá-los a partir da posição do avô — (*porque?*)

A coisa fica assim

```
func inserçãoRN (k,q)

      Se (q == Nulo)                                    // cria o novo registro
            r = Novo (RegIntRN)
            r->num,cor,esq,dir = k, Verm, Nulo, Nulo
            Retorna (r)

      Sse (q->num > k)                                  // descida pela esquerda

            q->esq = inserçãoRN (k,q->esq)
            Se ((q->esq)->cor == Preto)     Retorna (q)

            neto1  = Nulo;   neto2 = Nulo               // verificação e ajuste
            Se (q->esq->esq != Nulo  E  (q->esq->esq)->cor == Verm)
                  neto1 = q->esq->esq
            Se (q->esq->dir != Nulo  E  (q->esq->dir)->cor == Verm)
                  neto2 = q->esq->dir
            Se (neto1 == Nulo  E  neto2 == Nulo)   Retorna (q)

            Sse ((q->dir)->cor == Verm)                 // caso 3
                  q->cor = Verm
                  (q->esq)->cor = Preto;   (q->dir)->cor = Preto
                  Retorna (q)

            Sse (neto1 != Nulo)                         // caso 2.1
                  q = rotDir (q->esq,q)
                  q->cor = Preto;   (q->dir)->cor = Verm
                  Retorna (q)


Senão
      q->esq = rotEsq (neto2,q->esq)
      q = rotDir (q->esq,q)
      q->cor = Preto;   (q->dir)->cor = Verm
      Retorna (q)
```