Árvores AVL

Prof. Luciano Antonio Digiampietri

Árvores AVL

São **árvores de busca binária balanceadas** (também chamadas de balanceadas na altura).

- O nome AVL vem de seus criadores **A**delson **V**elsky e **L**andis, cuja primeira referência encontra-se no documento "Algoritmos para organização da informação" de 1962.

Árvores AVL

- Além das características de uma árvore de busca binária, possuem uma propriedade adicional para **garantir o balanceamento**.
- Para todo nó, **a diferença** (em valor absoluto) **entre a altura** da subárvore à direita e à esquerda **será, no máximo, 1** (um).
- As operações de inserção e exclusão devem garantir essa propriedade.

Árvores AVL

- Graças a esta propriedade, **a altura de uma árvore AVL será limitada a O(log n)**, conforme será visto hoje.

Árvores AVL

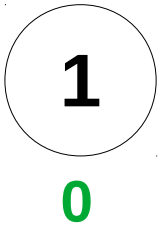
- Definiremos o **balancemanto de um nó** como a diferença entre a altura de sua subárvore à direita e a altura de sua subárvore à esquerda.
- O balanceamento de um nó de uma árvore AVL **sempre valerá -1, 0 ou 1**.

Inserções em Árvore de Busca Binária

- Desejamos inserir três elementos, na seguinte ordem: 1, 2 e 3

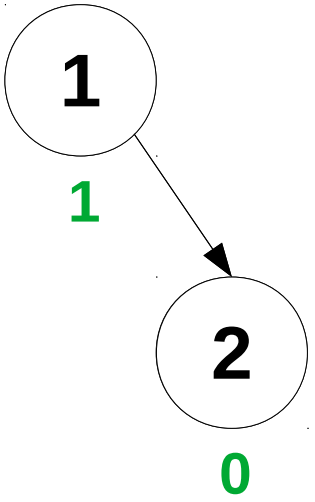
Inserções em Árvore de Busca Binária

- Desejamos inserir três elementos, na seguinte ordem: 1, 2 e 3



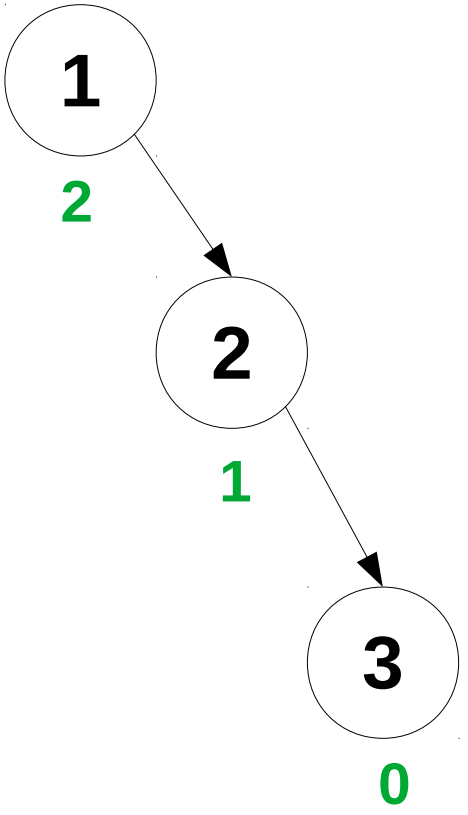
Inserções em Árvore de Busca Binária

- Desejamos inserir três elementos, na seguinte ordem: 1, 2 e 3



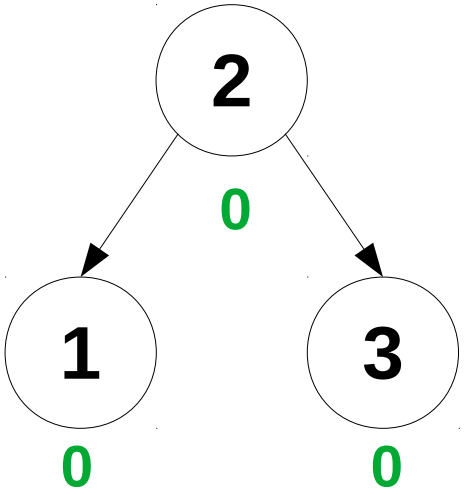
Inserções em Árvore de Busca Binária

- Desejamos inserir três elementos, na seguinte ordem: 1, 2 e 3



Inserções em Árvore AVL

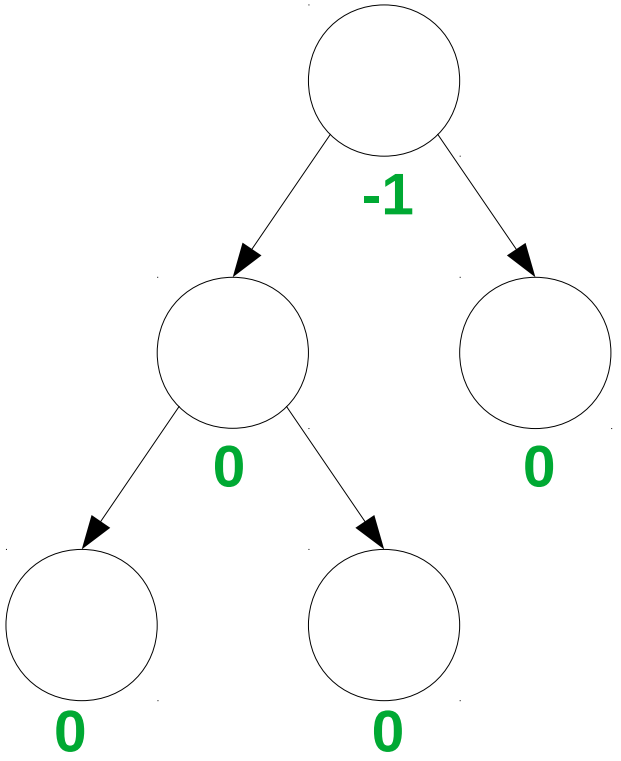
- Desejamos inserir três elementos, na seguinte ordem: 1, 2 e 3



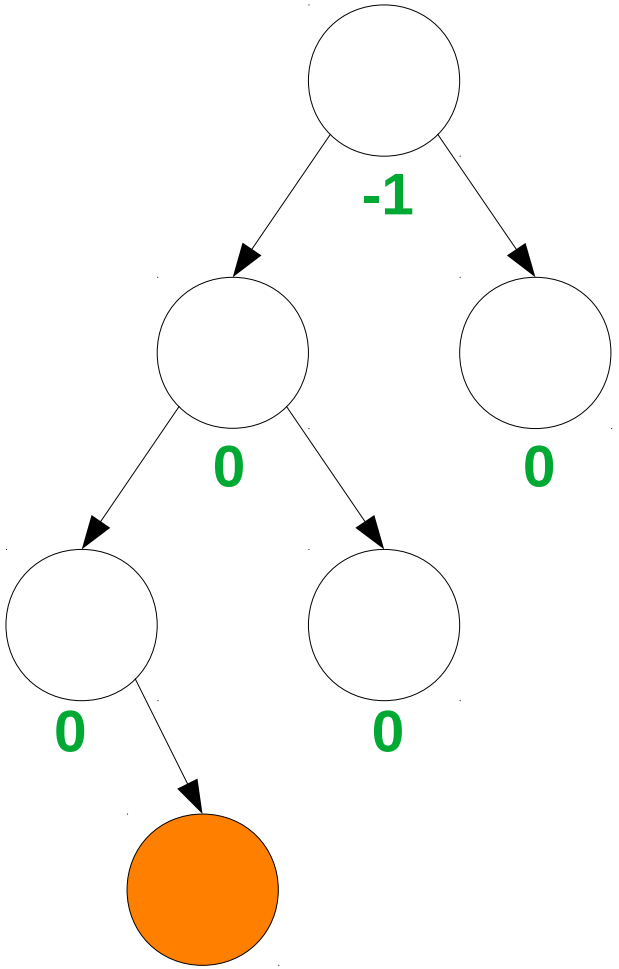
Inserções em Árvore AVL

- Realiza-se a inserção seguindo as regras de árvores de busca binária, de maneira recursiva.
- Durante a volta da recursão, **atualiza-se o balanceamento** de cada nó e verifica se ele **viola a propriedade de um árvore AVL**.
  - Notem que o nó inserido (balanceamento igual a zero) e seu pai não irão violar a regra.
- Se o nó atual violar a propriedade, realiza-se **uma "rotação"** para corrigira a árvore.

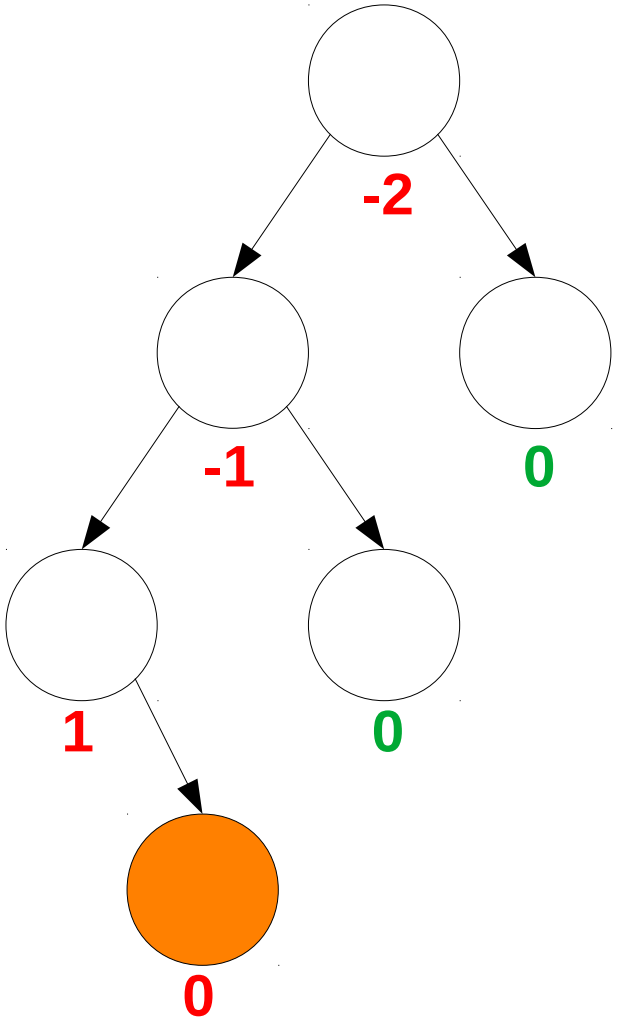
Inserções em Árvore AVL



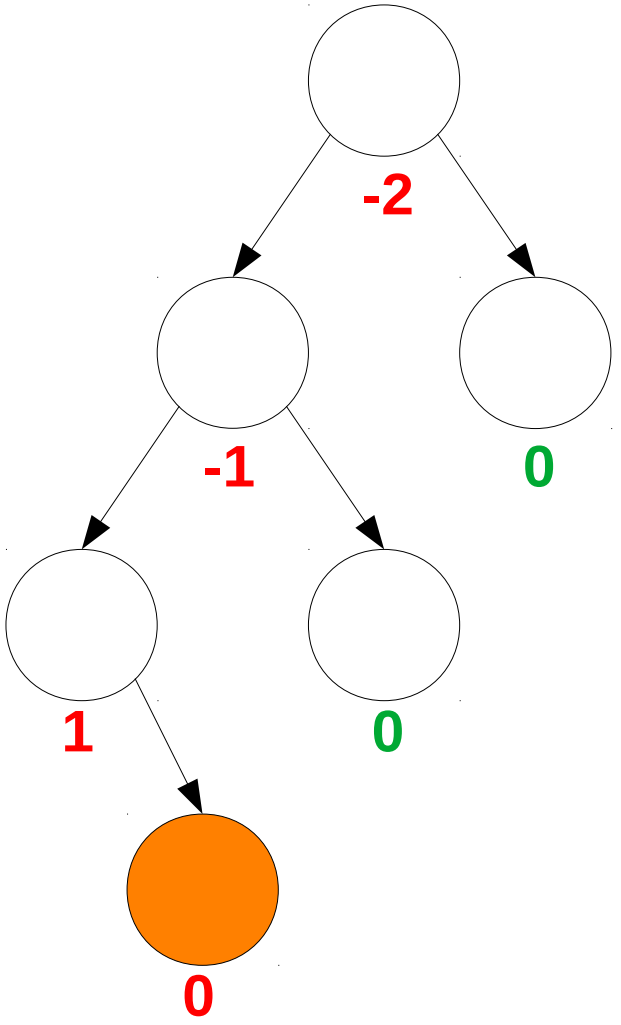
Inserções em Árvore AVL



Inserções em Árvore AVL



Inserções em Árvore AVL

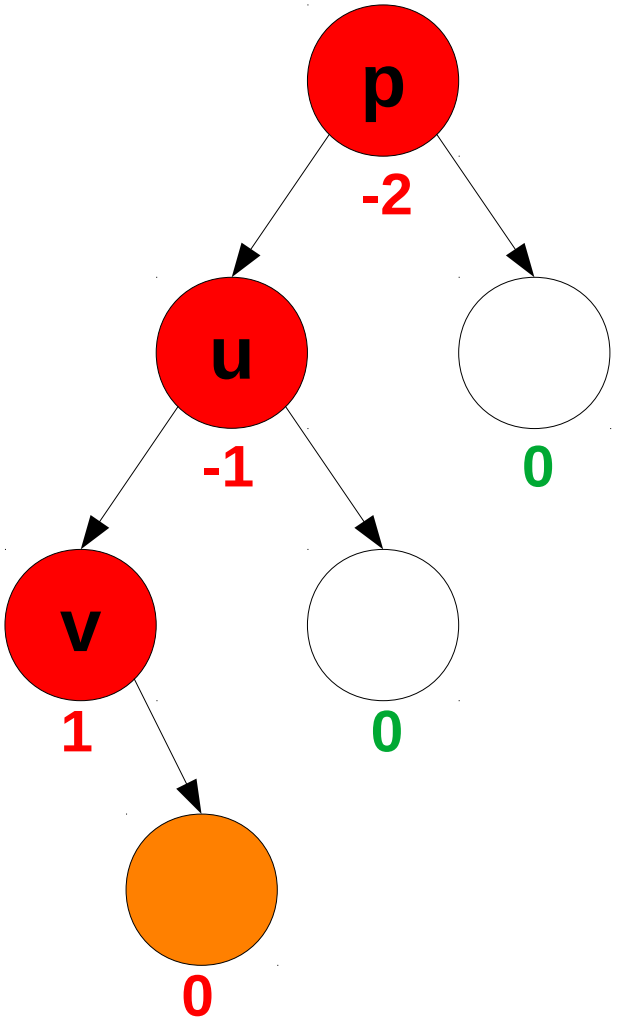


A partir do nó inserido:

O **primeiro nó** com balanceamento igual a -2, se existir, será chamada de **p** (o “problema”)

- Seu filho, no sentido da inserção será chamado de **u**
- Seu neto, no sentido da inserção será chamado de **v**

Inserções em Árvore AVL

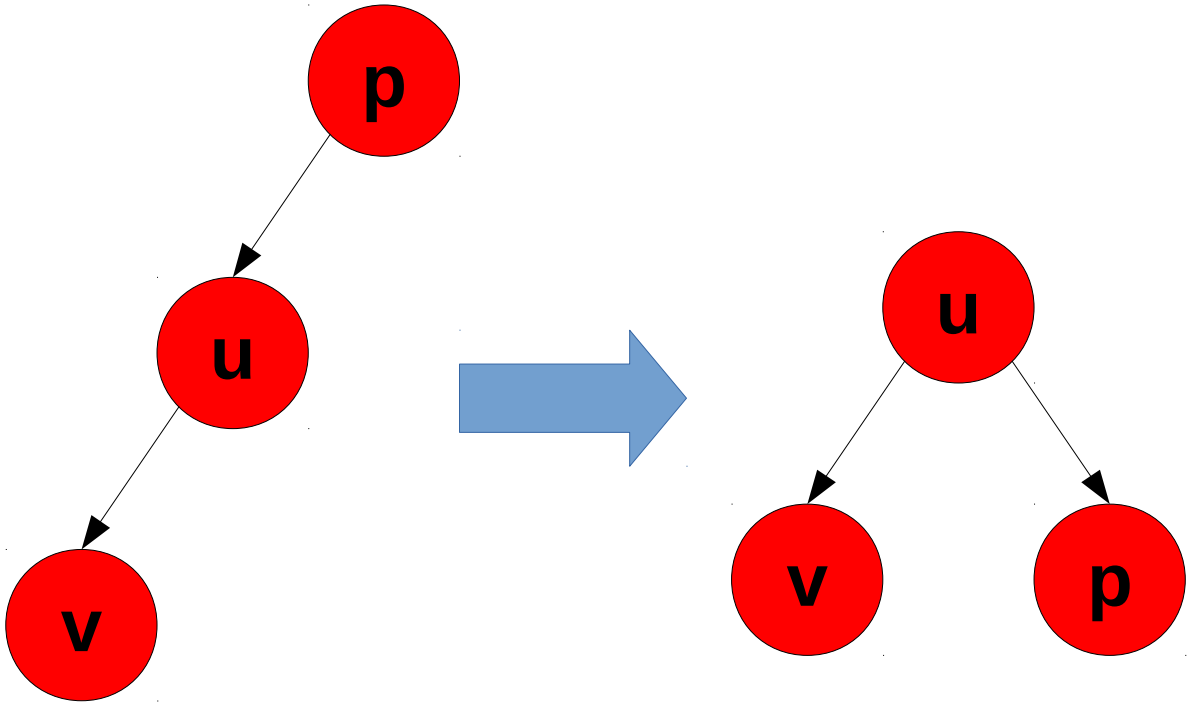


A partir do nó inserido:

O **primeiro nó** com balanceamento igual a -2, se existir, será chamada de **p** (o “problema”)

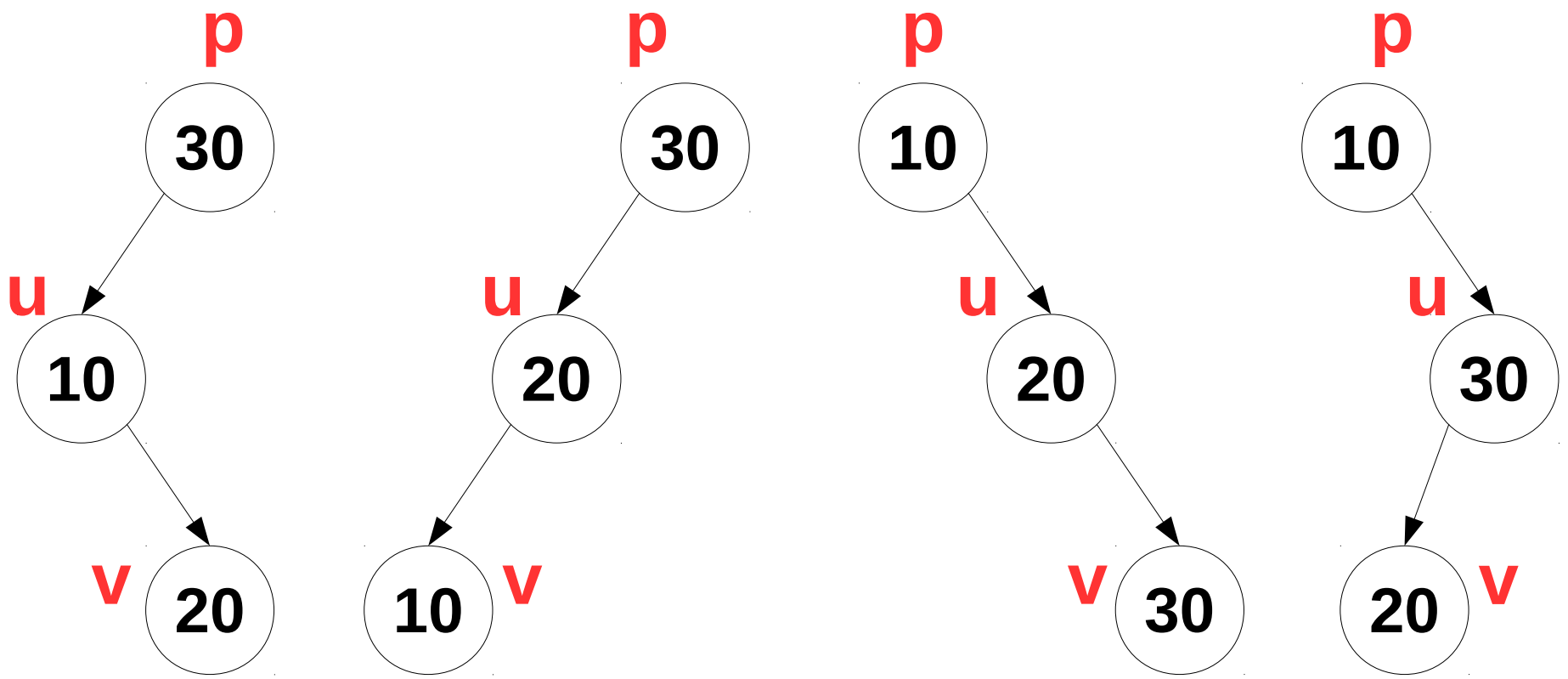
- Seu filho, no sentido da inserção será chamado de **u**
- Seu neto, no sentido da inserção será chamado de **v**

Inserções em Árvore AVL



Inserções em Árvore AVL

- Há quatro configurações possíveis para p, u e v:

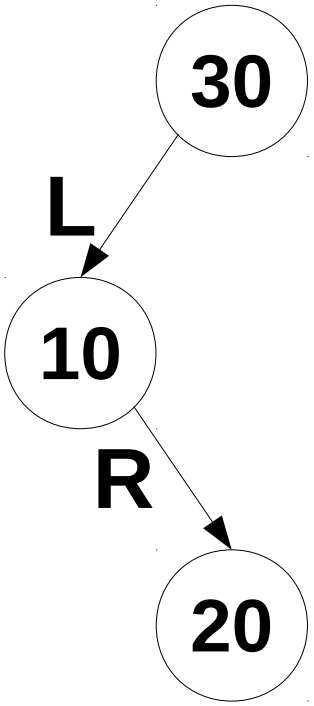


Inserções em Árvore AVL

Posição relativa de u em relação à p e de v em relação à u:

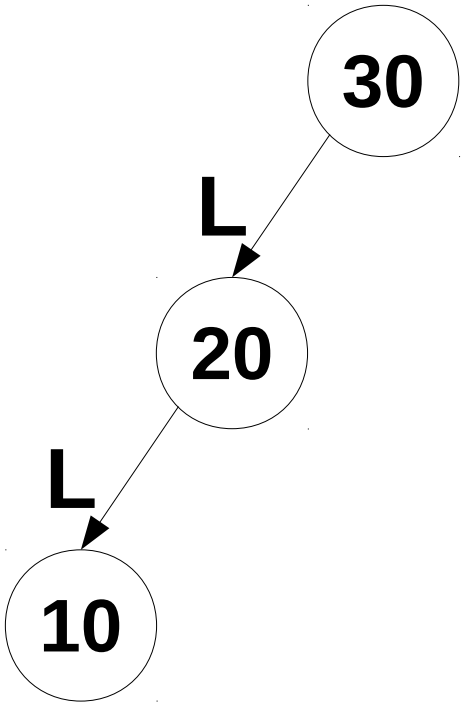
Esquerda-Direita

LR



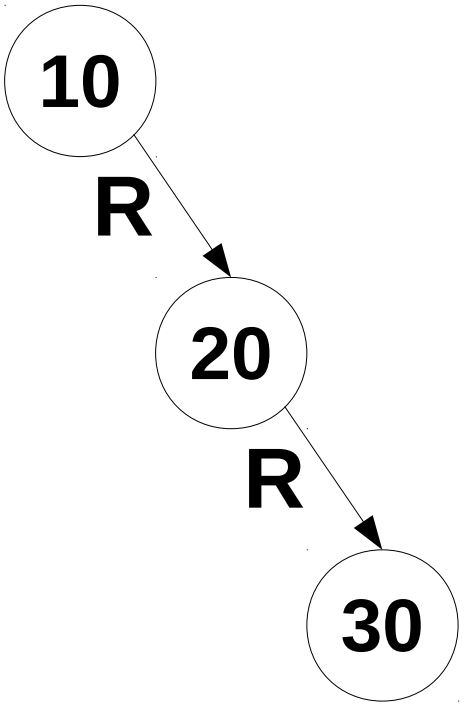
Esquerda-Esquerda

LL



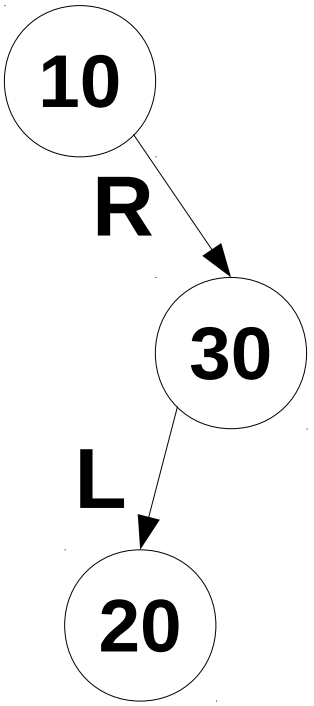
Direita-Direita

RR

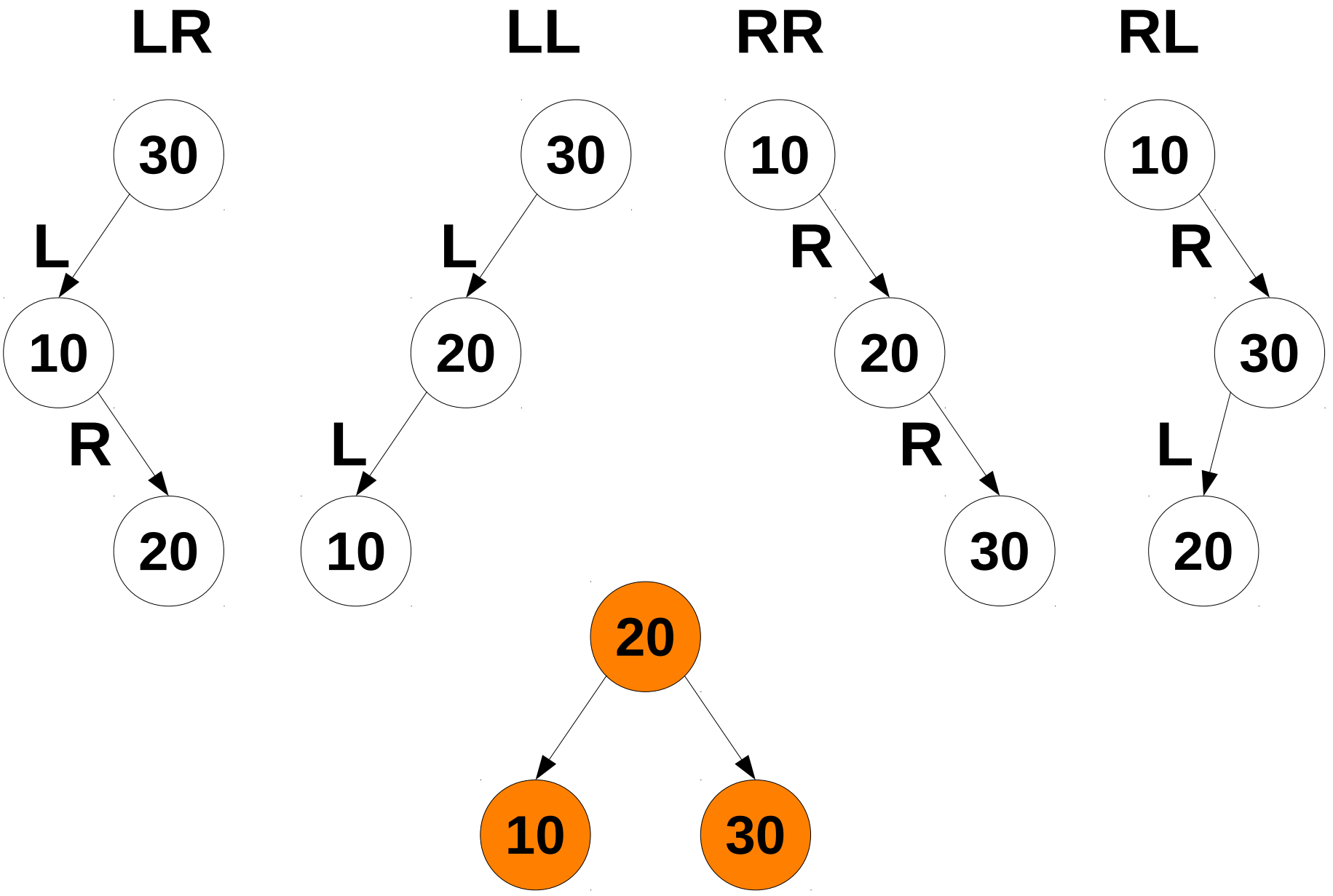


Direita-Esquerda

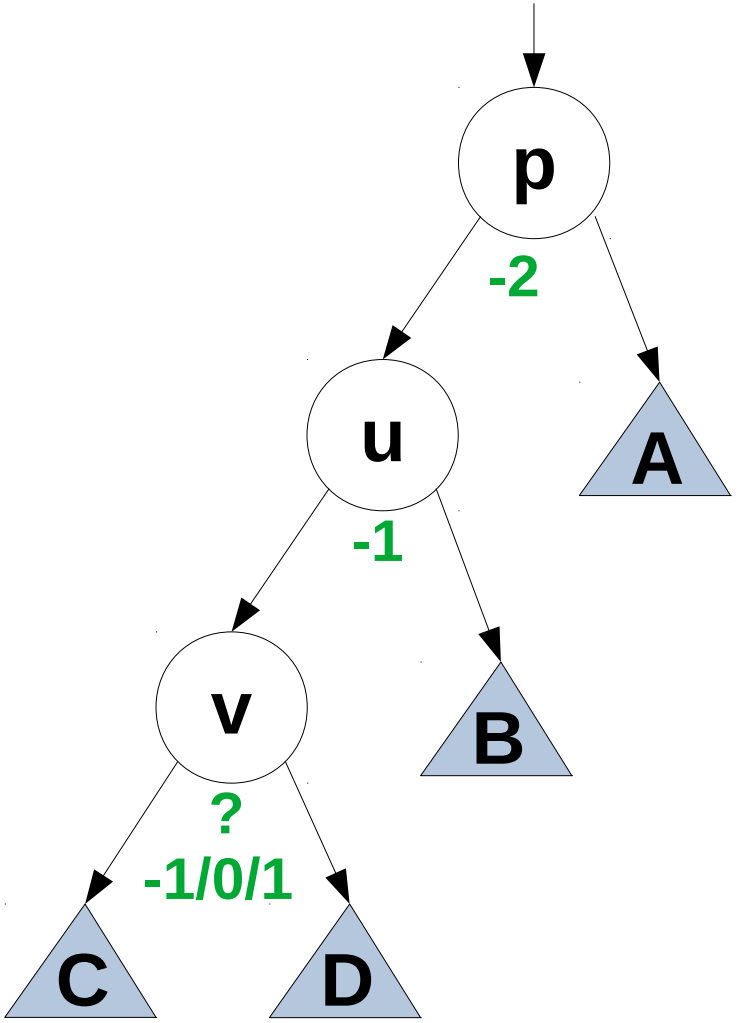
RL



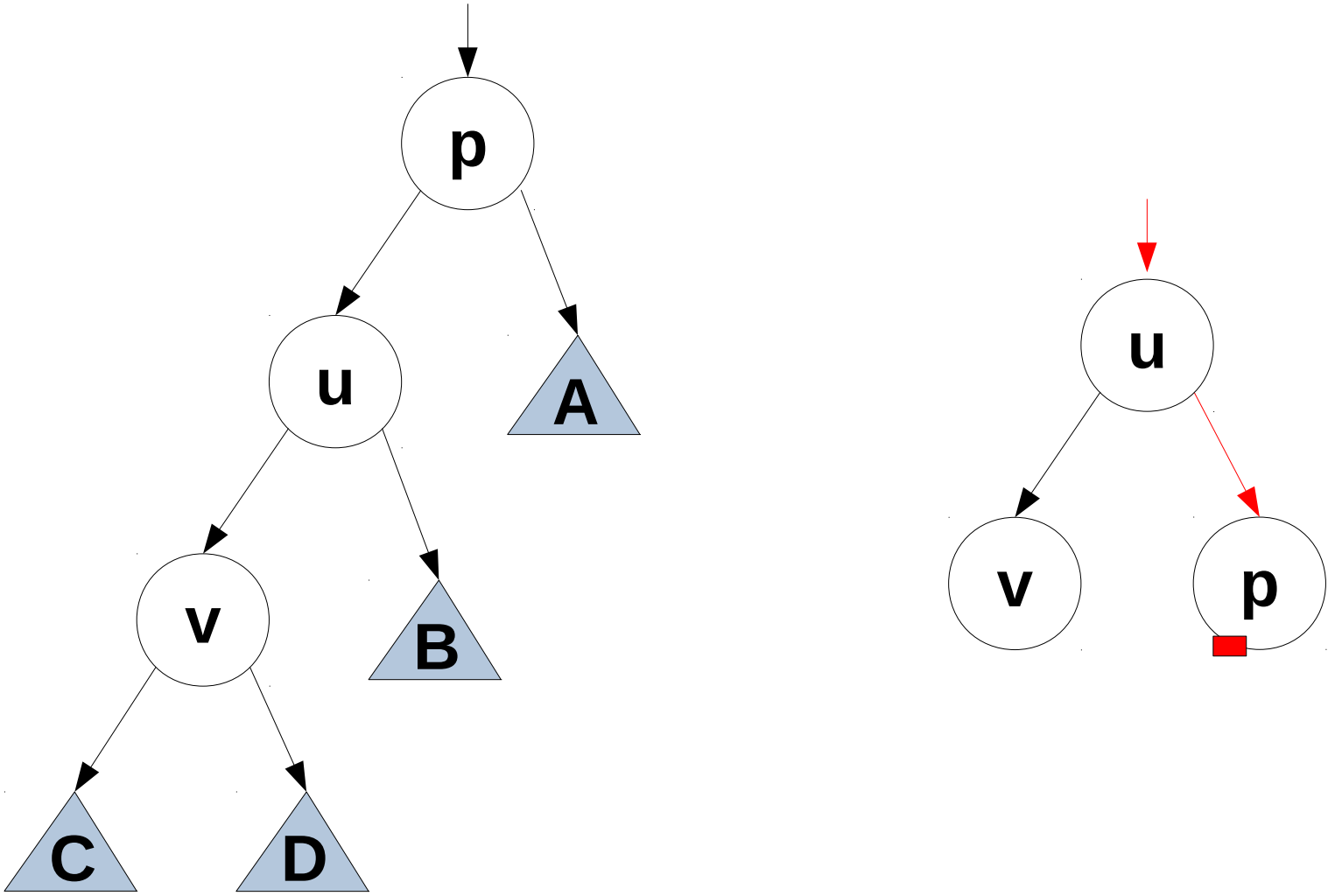
Inserções em Árvore AVL



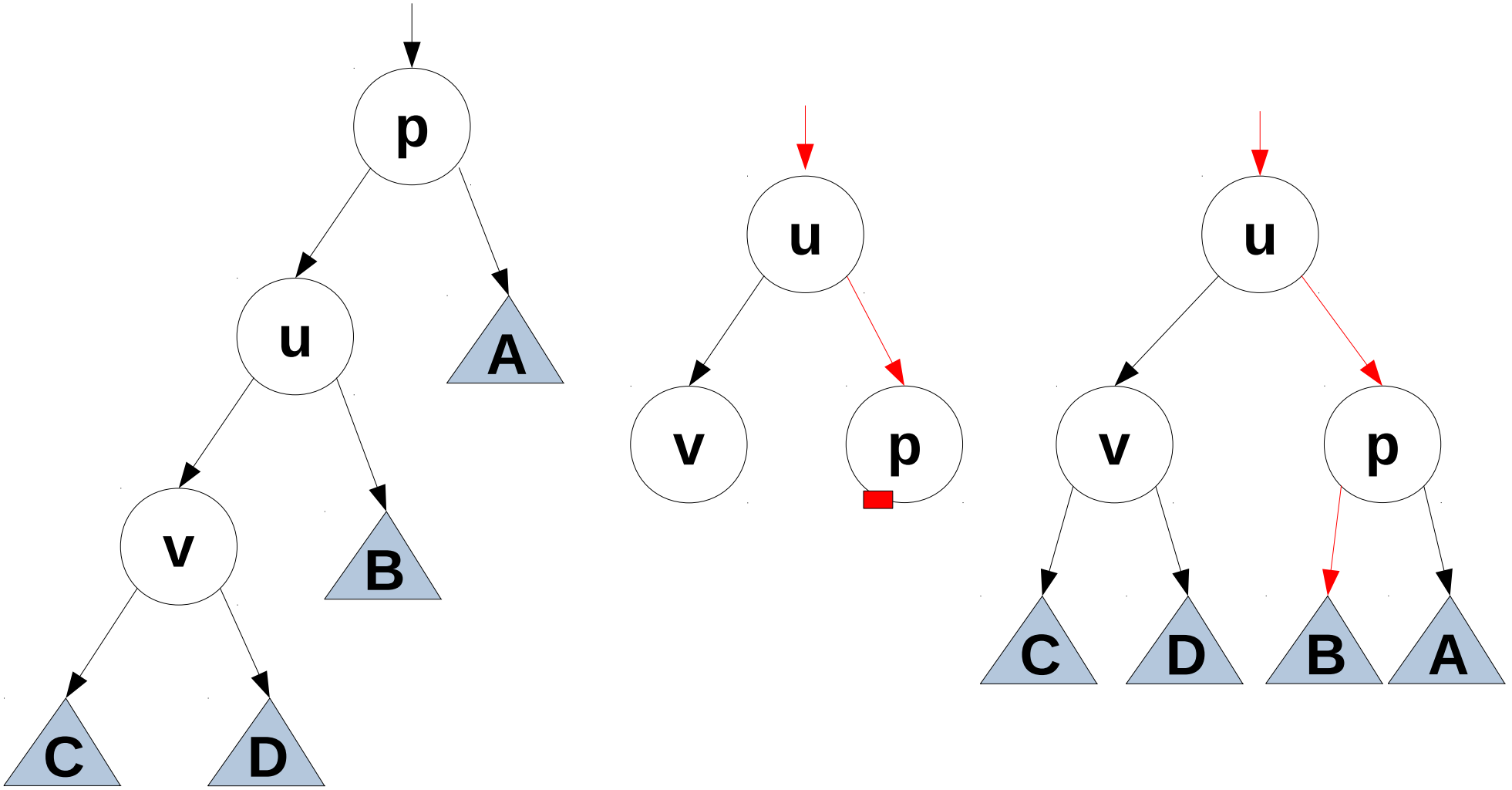
Rotação LL



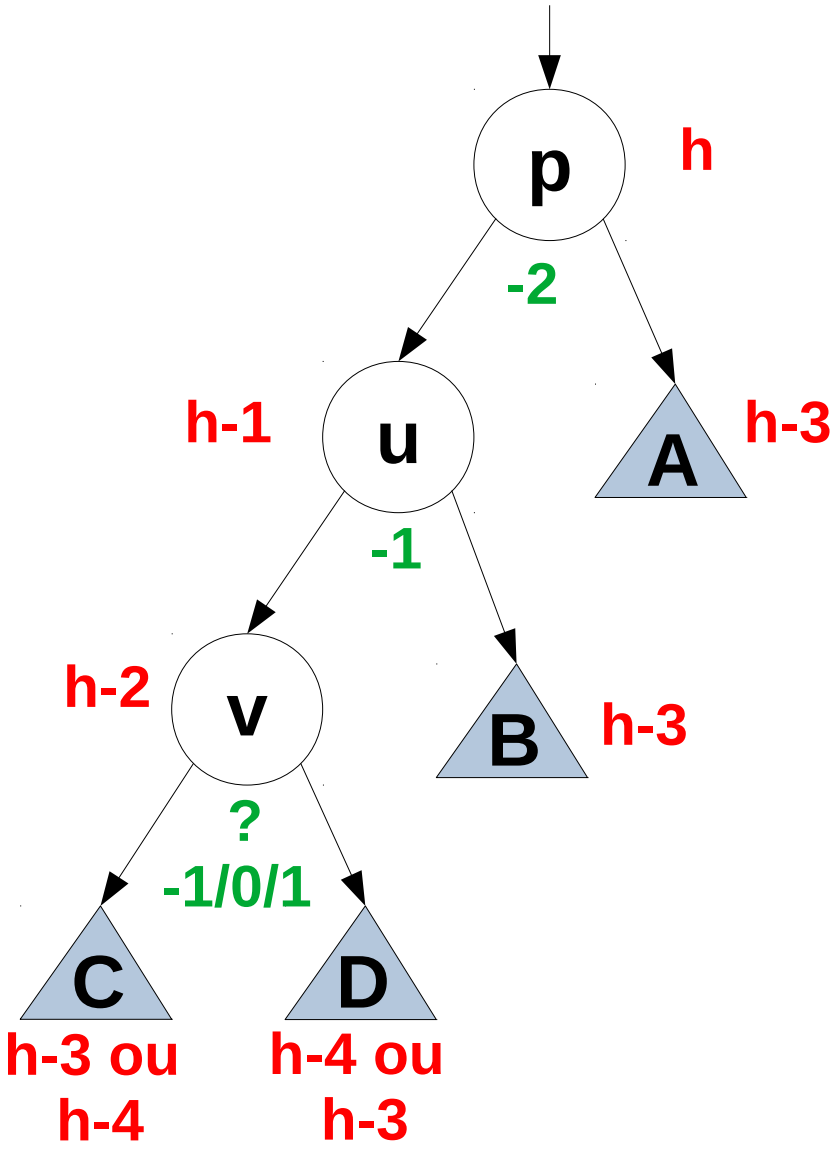
Rotação LL



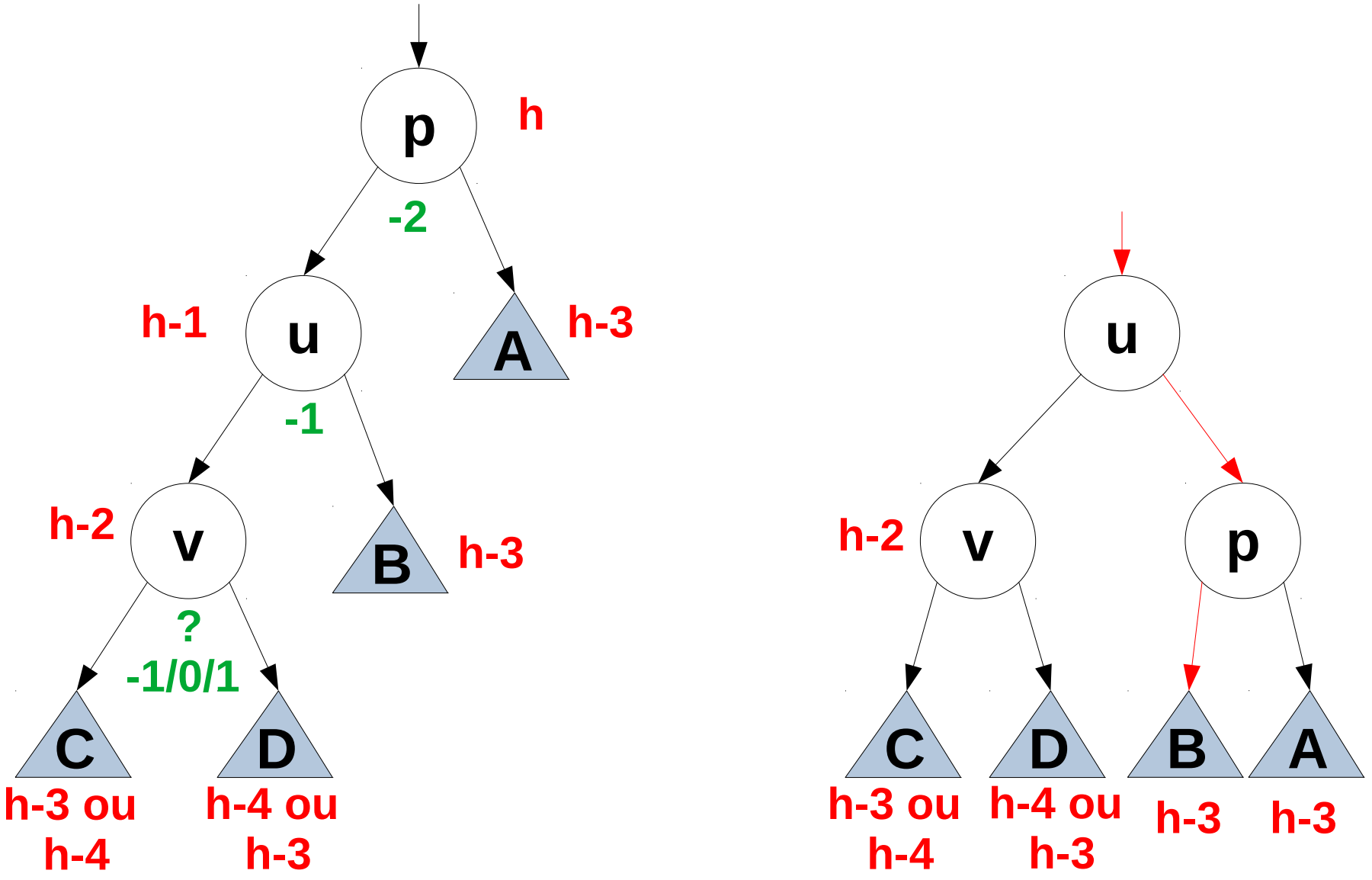
Rotação LL



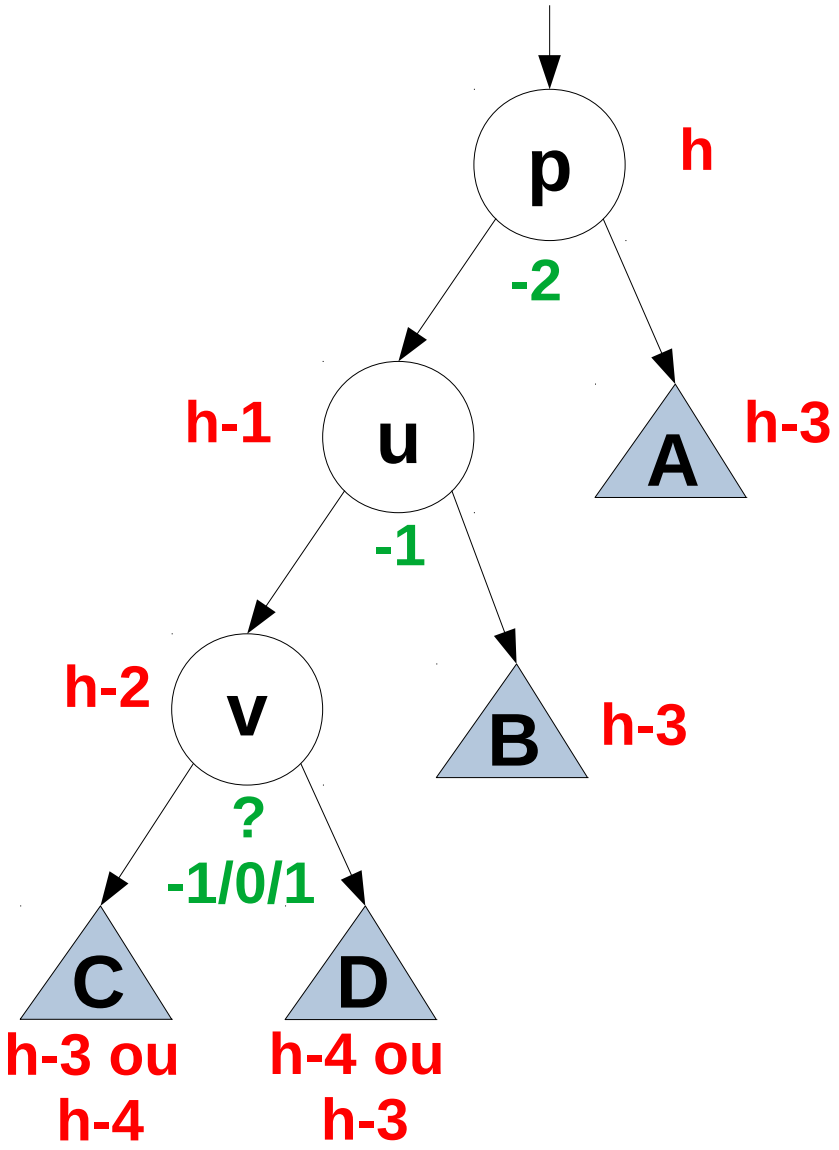
Rotação LL – altura dos nós

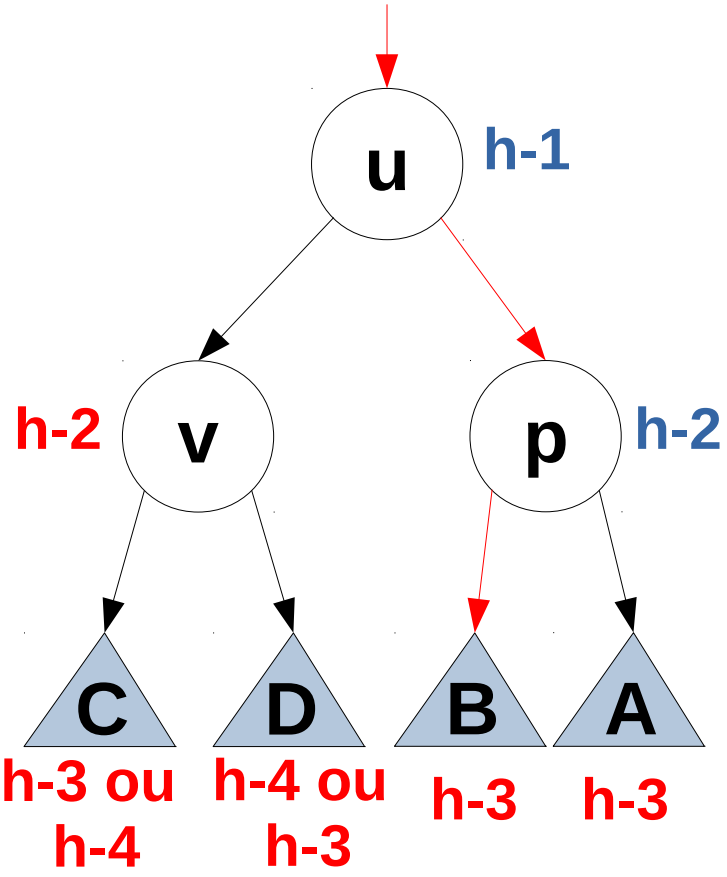


Rotação LL – altura dos nós

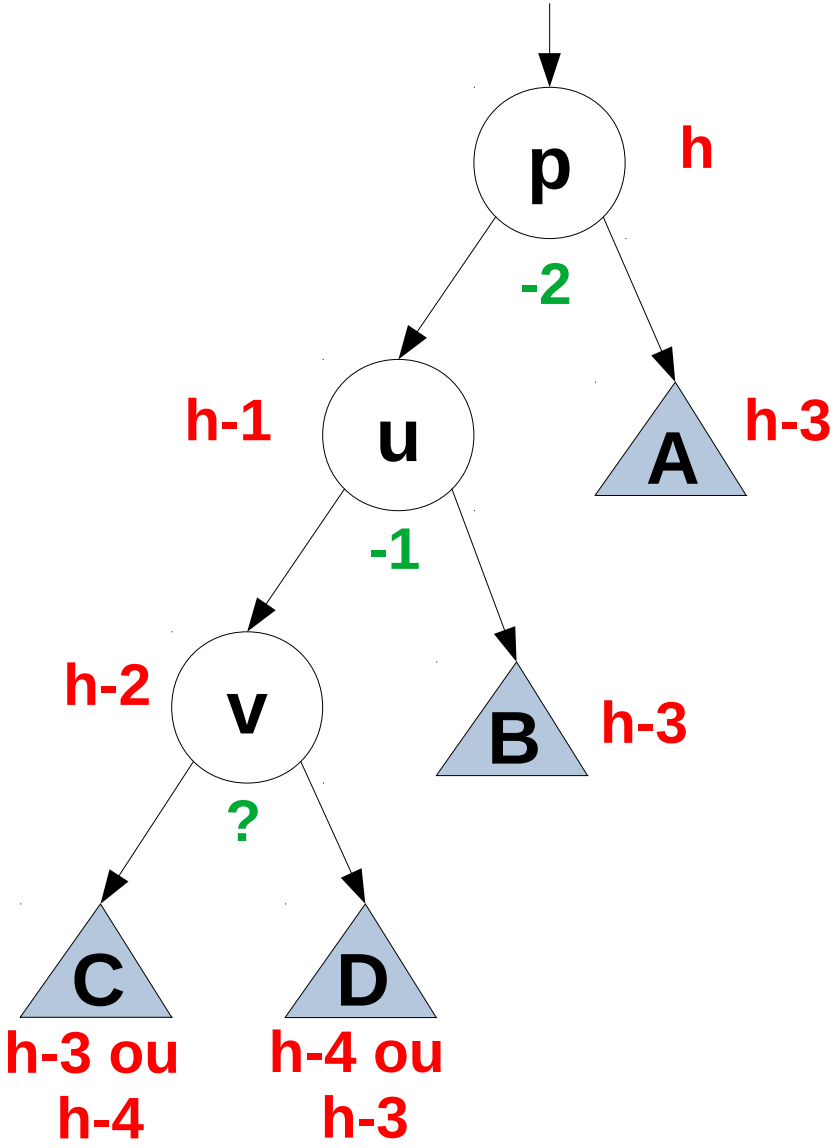


Rotação LL – altura dos nós

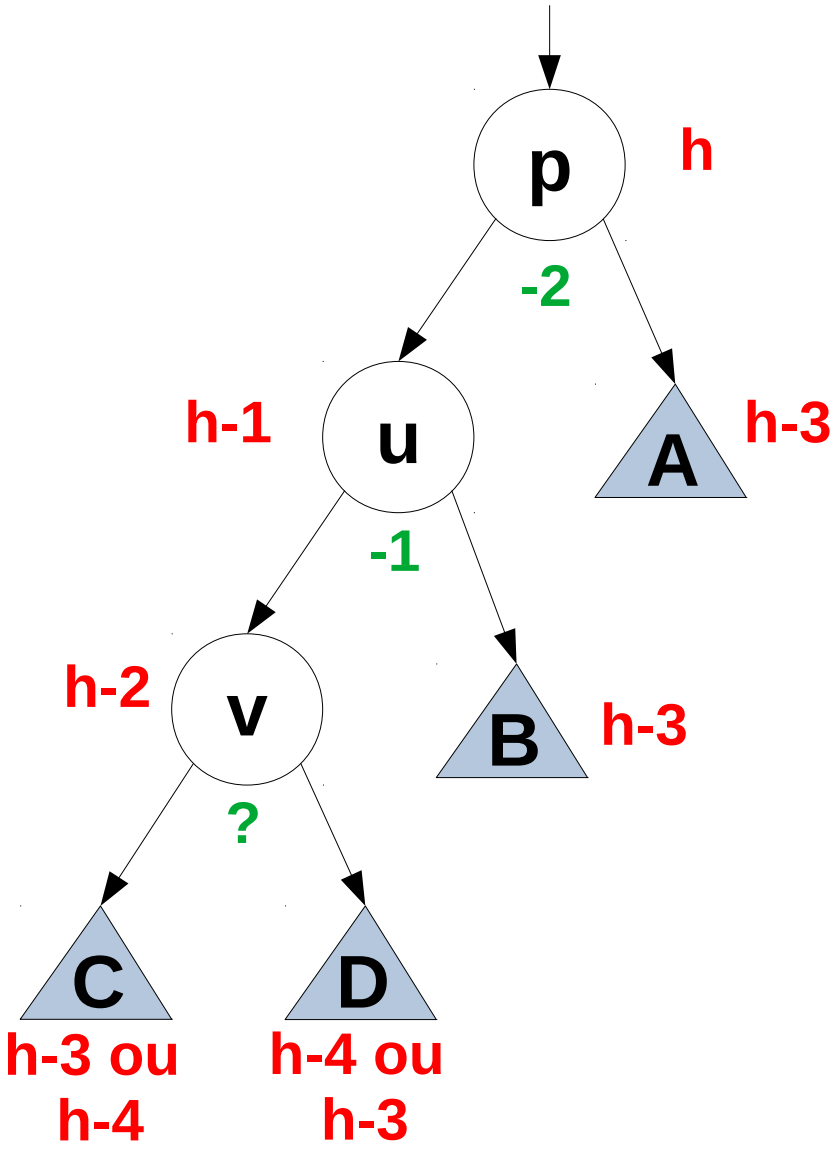


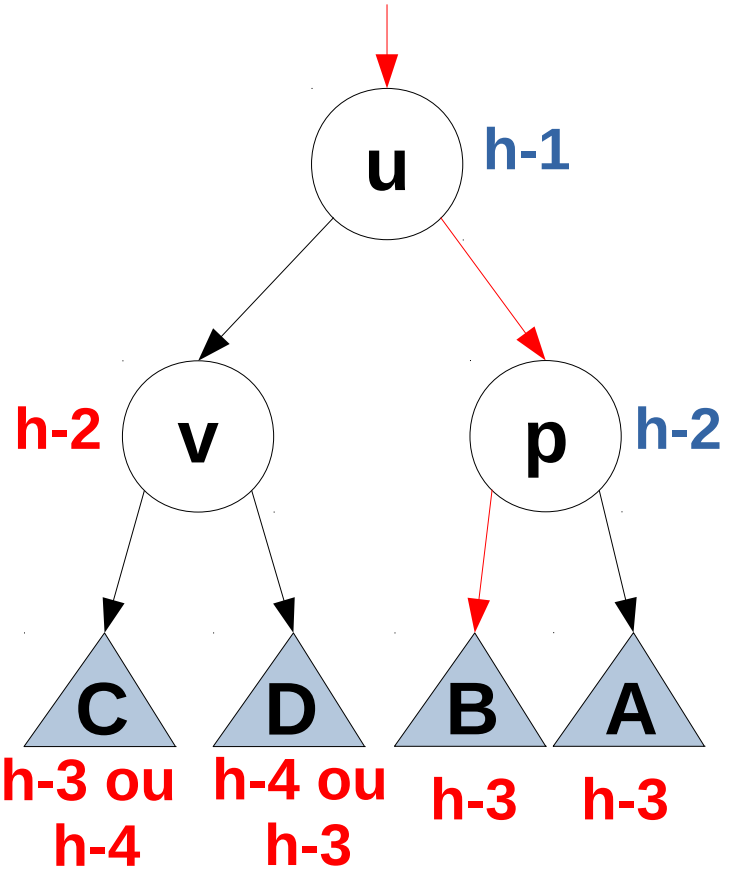


Rotação LL – balanceamento

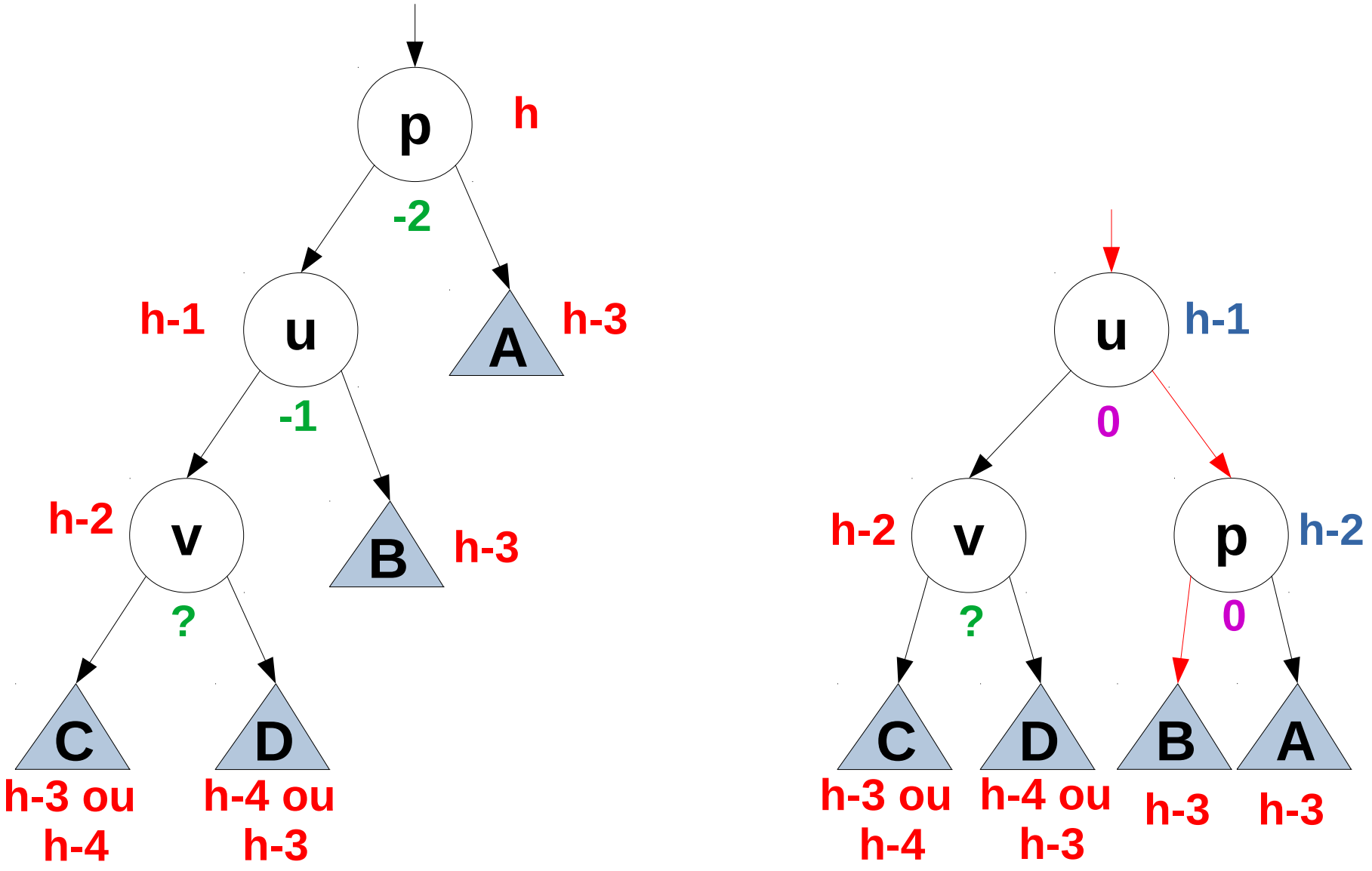


Rotação LL – balanceamento





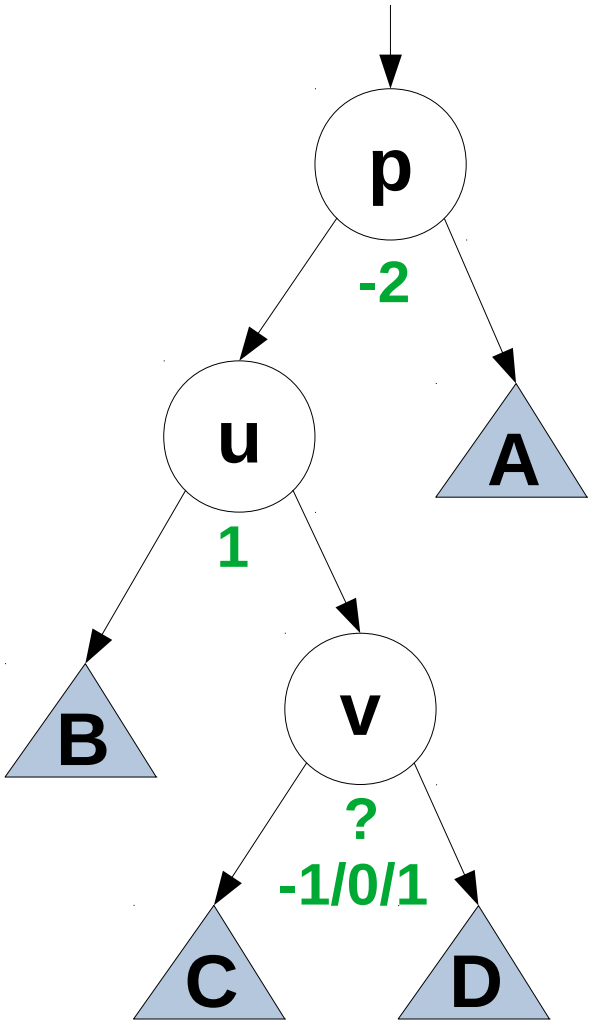
Rotação LL – balanceamento



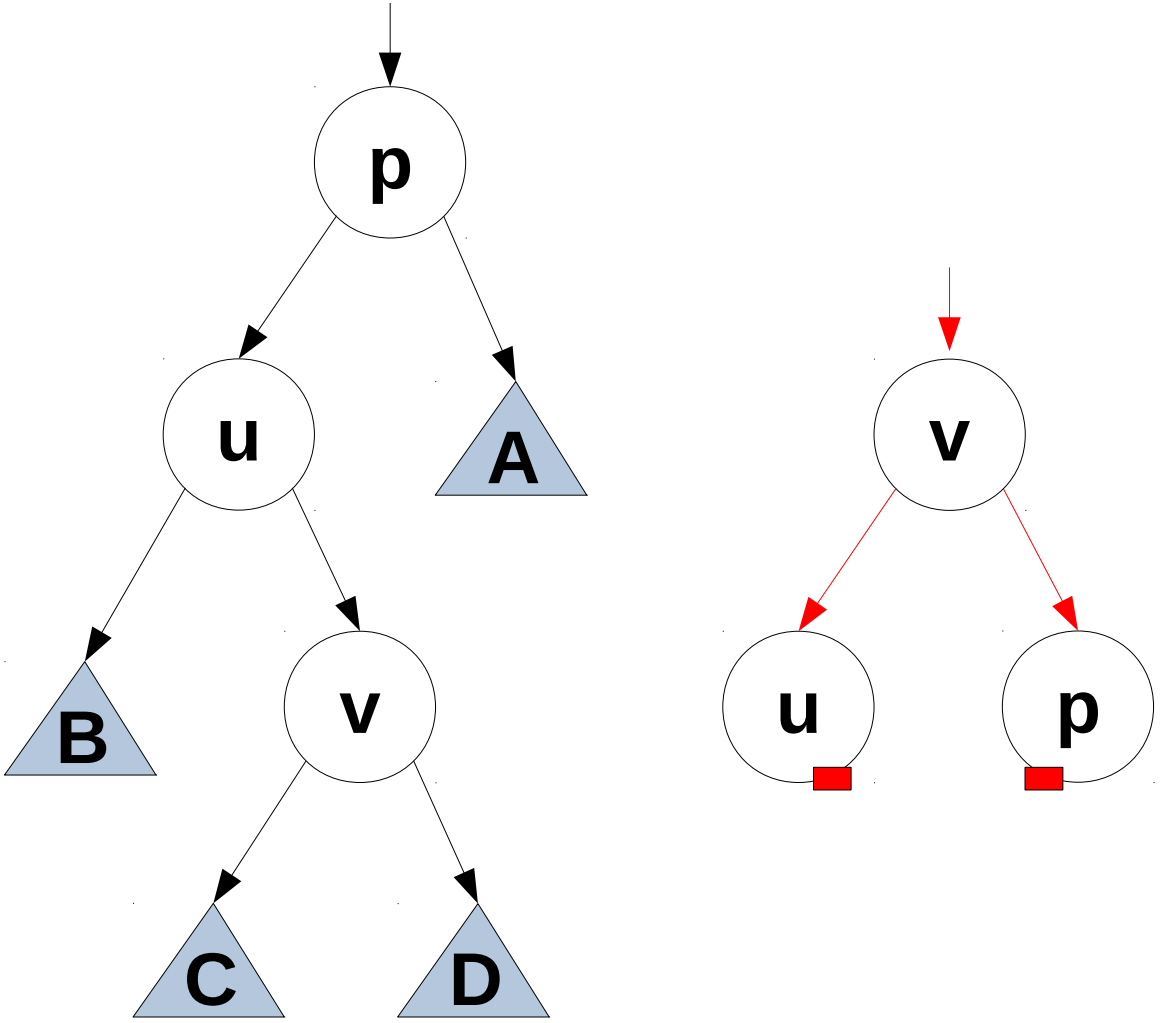
Rotação LL – código

```
PONT rotacaoL(PONT p){
      PONT u, v;
      u = p->esq;
      if(u->bal == -1) {    // LL
            p->esq = u->dir;
            u->dir = p;
            p->bal = 0;
            u->bal = 0;
            return u;
      }
```

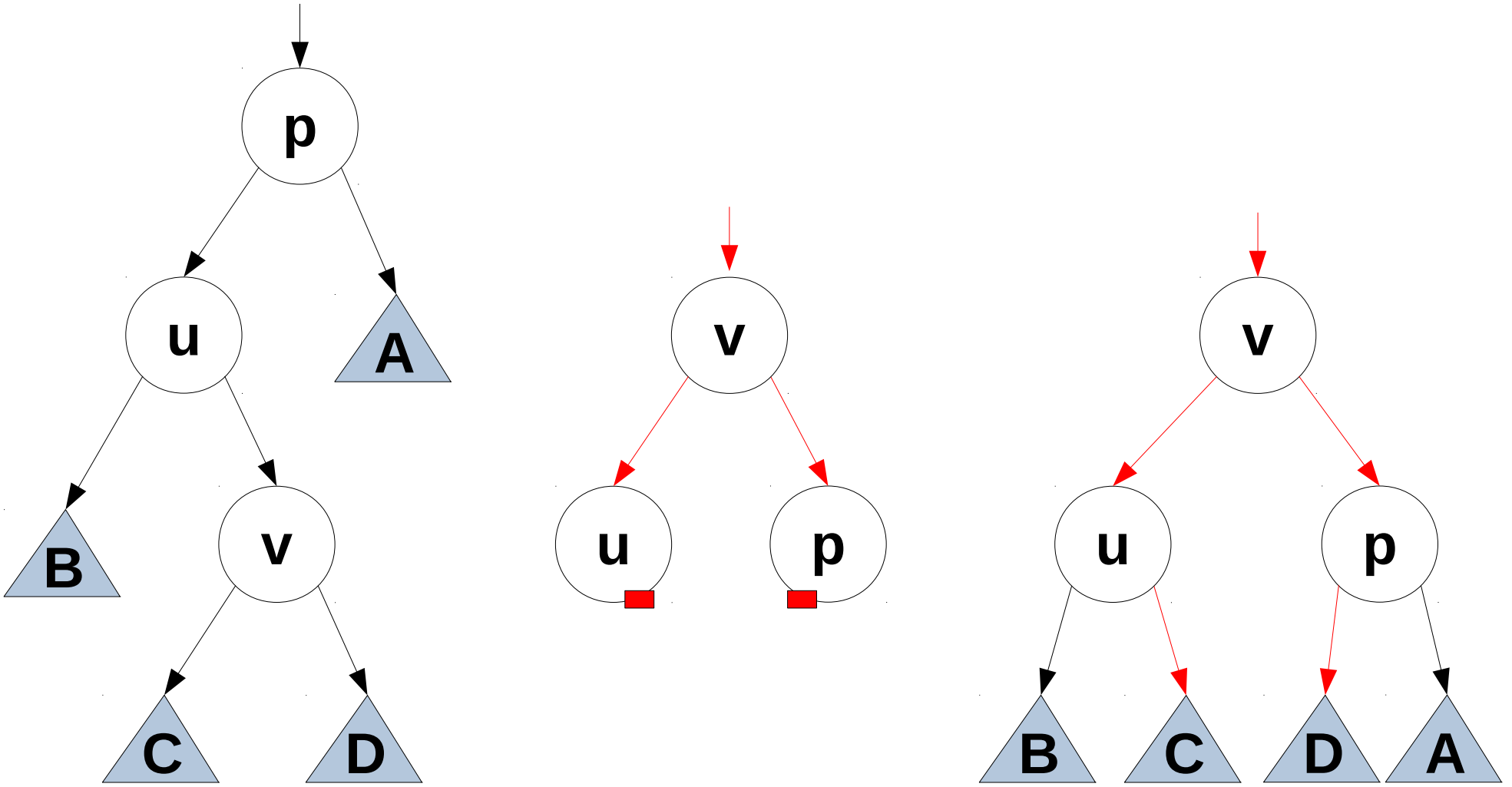
Rotação LR



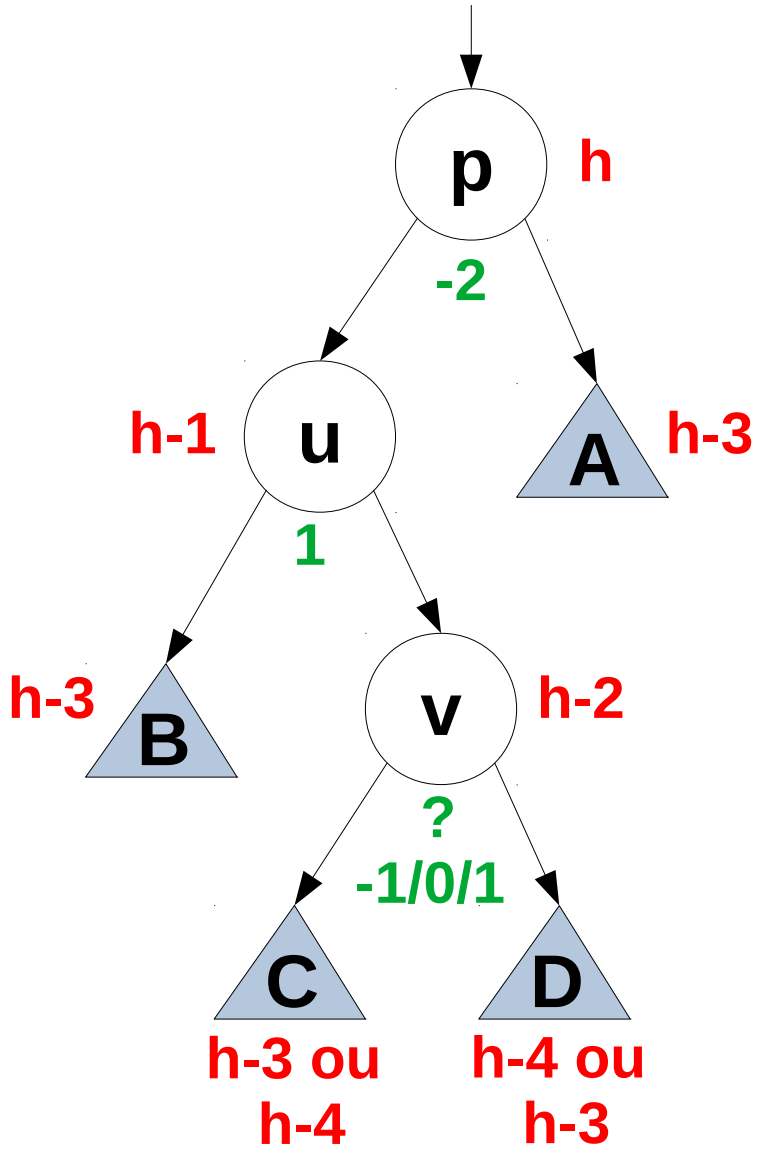
Rotação LR



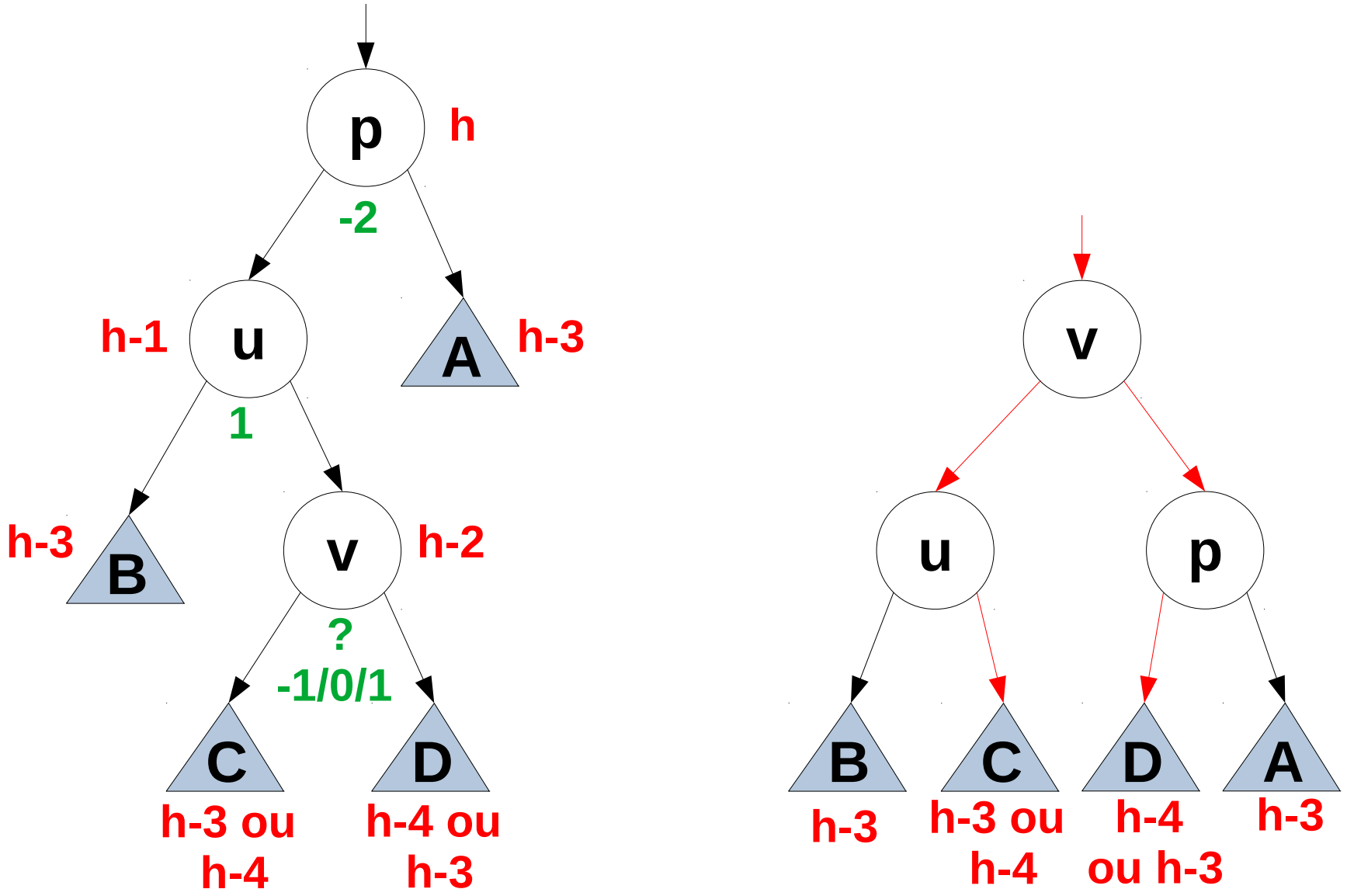
Rotação LR



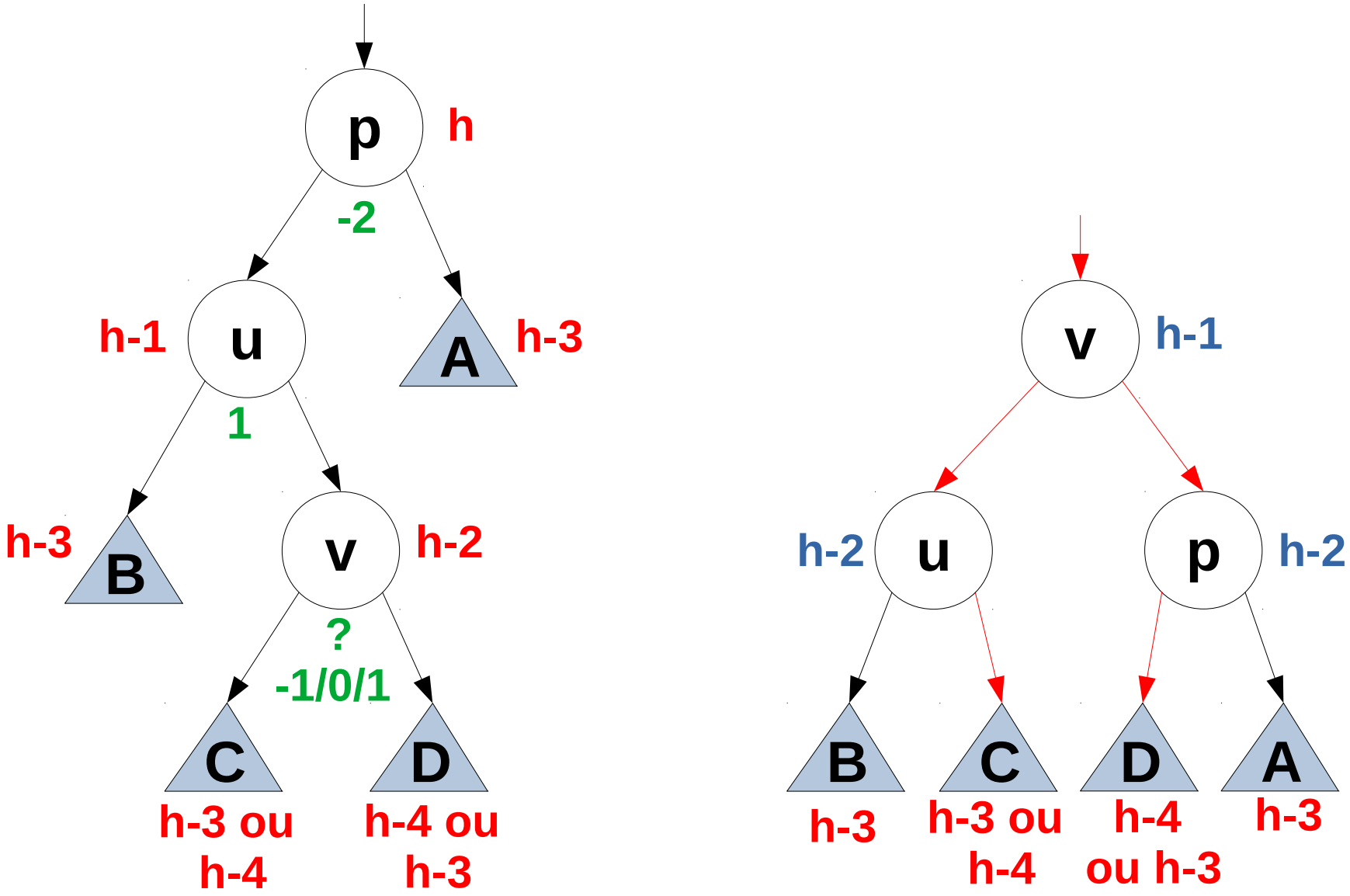
Rotação LR – altura dos nós



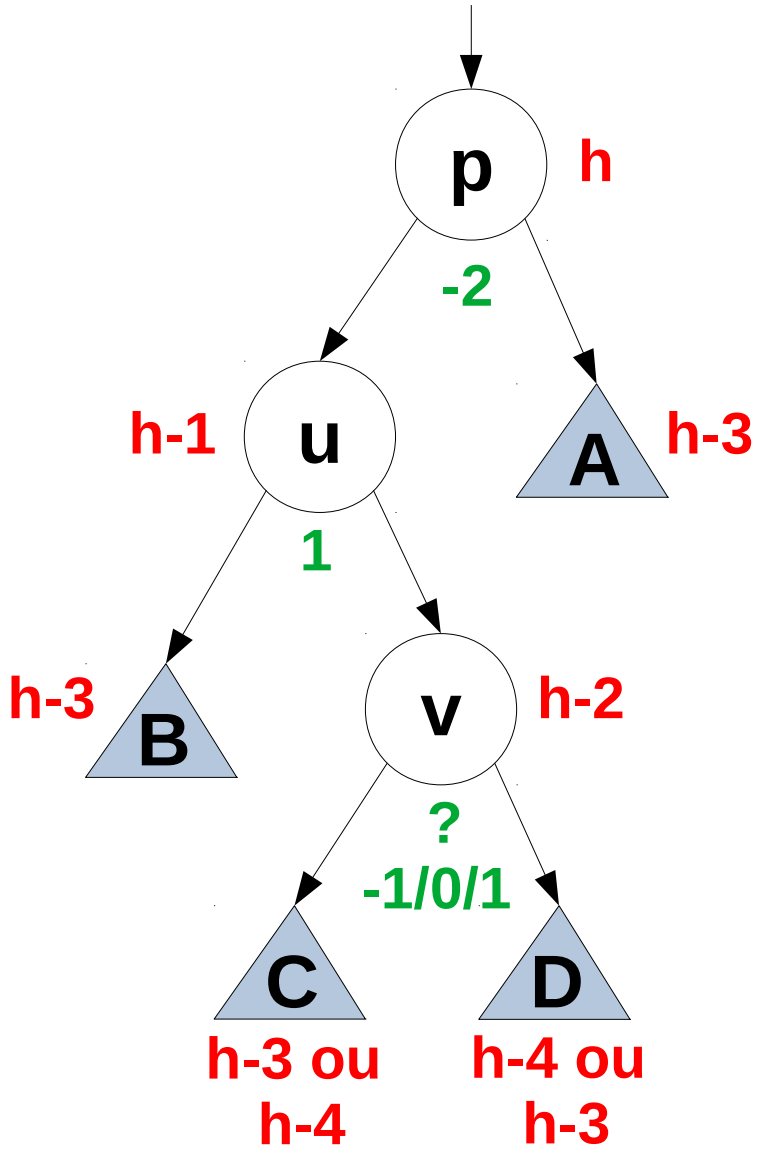
Rotação LR – altura dos nós



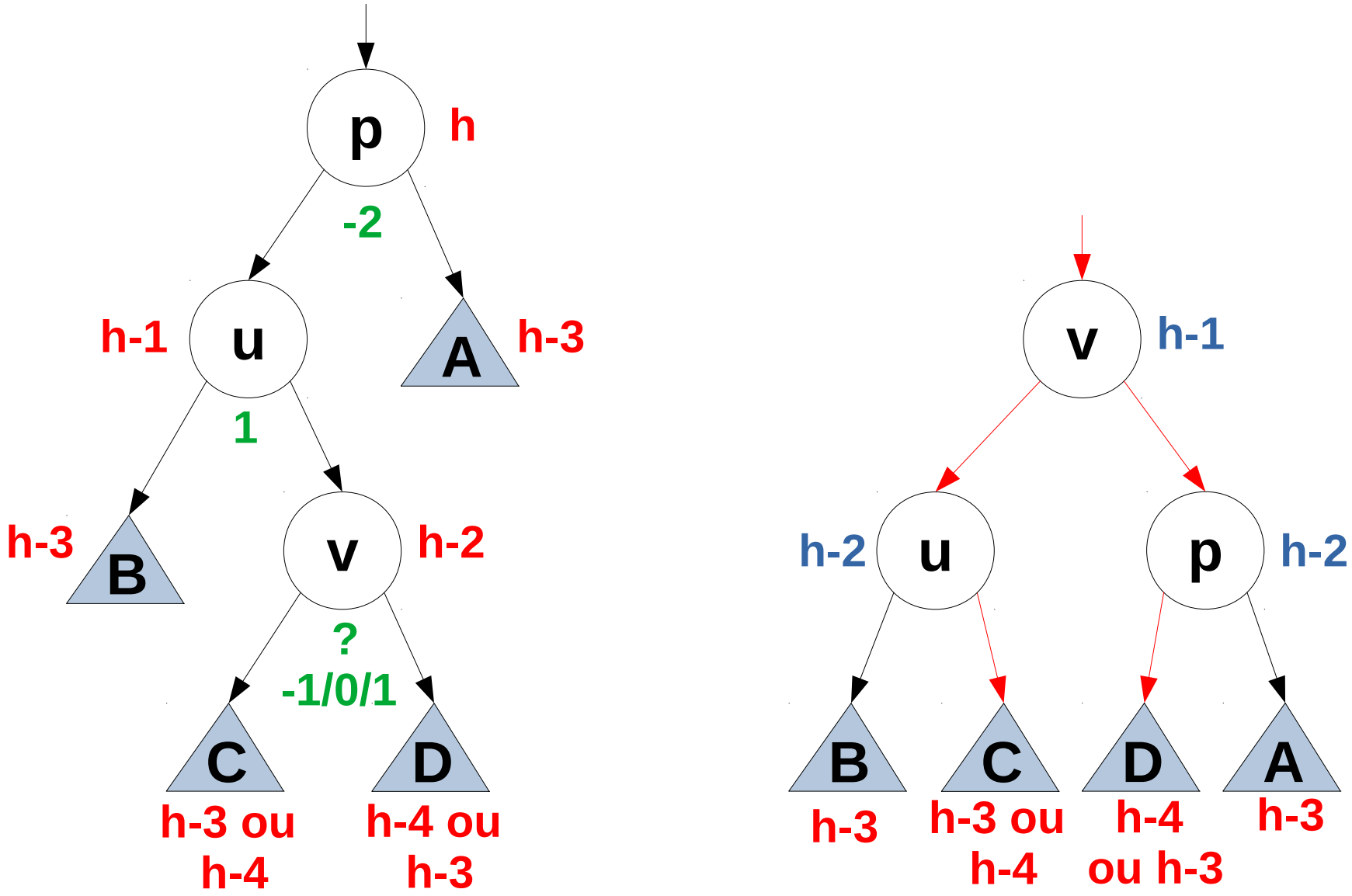
Rotação LR – altura dos nós



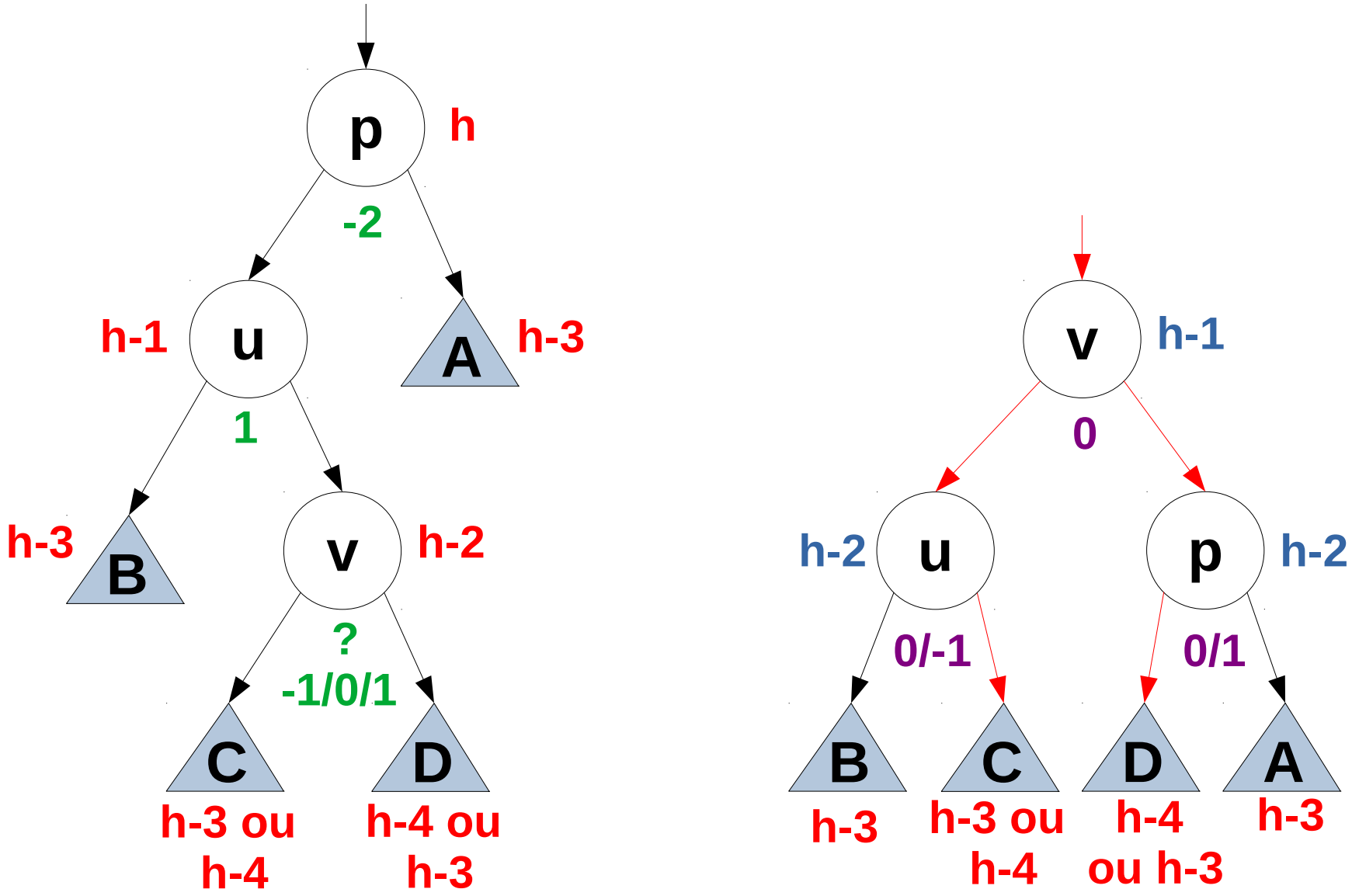
Rotação LR – balanceamento



Rotação LR – balanceamento



Rotação LR – balanceamento



Rotação LR – código

```
PONT rotacaoL(PONT p){
      PONT u, v;
      u = p->esq;
      if (u->bal == 1) {   // LR
            v = u->dir;
            u->dir = v->esq;
            v->esq = u;
            p->esq = v->dir;
            v->dir = p;
            if(v->bal == -1) p->bal = 1;
            else p->bal = 0;
            if(v->bal == 1) u->bal = -1;
            else u->bal = 0;
            v->bal = 0;
            return v;
      }
```

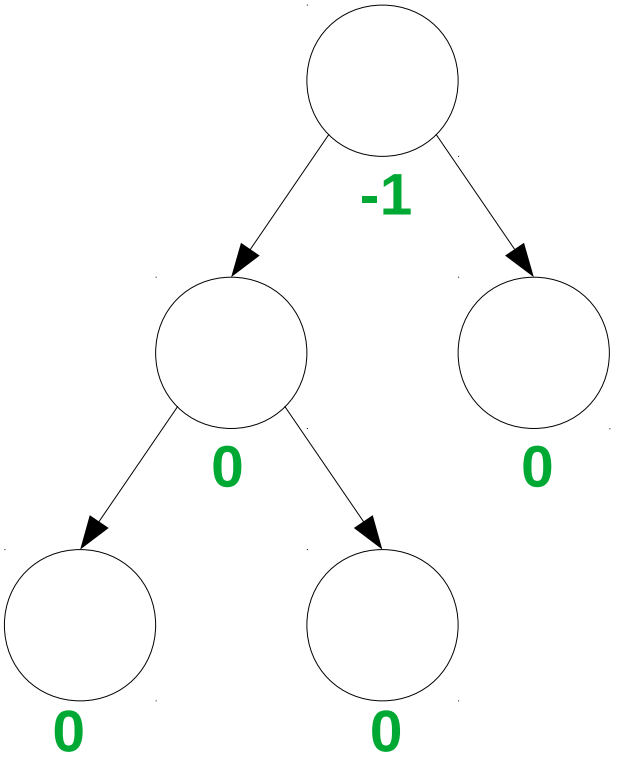
Exclusões em Árvore AVL

- Realiza-se a exclusão seguindo as regras de árvores de busca binária, de maneira recursiva.
- Durante a volta da recursão, **atualiza-se o balanceamento** de cada nó e verifica se ele **viola a propriedade de um árvore AVL**.
- Se o nó atual (durante a volta da recursão) violar a propriedade, realiza-se **uma "rotação"** para corrigir a árvore.

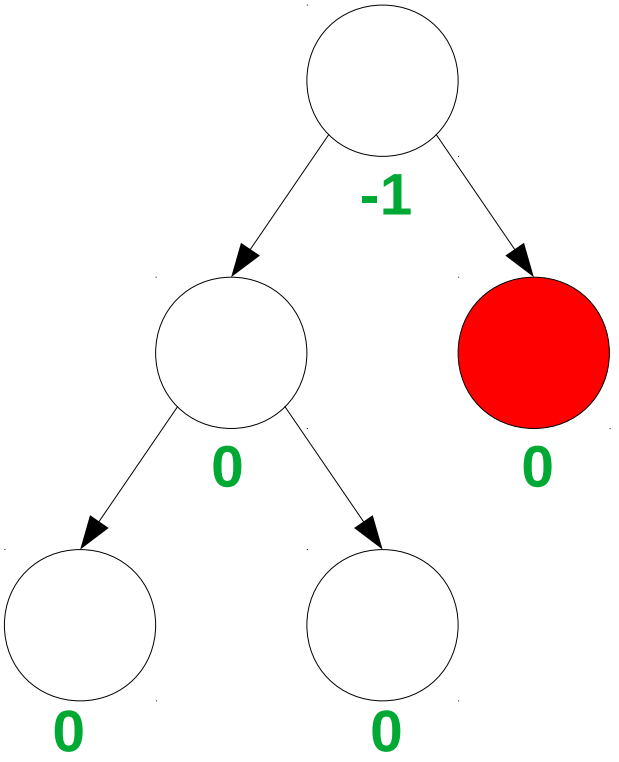
Exclusões em Árvore AVL

- A partir do **nó problema**, temos os mesmos casos da inserção (LL, LR, RR, RL) e dois casos adicionais e simétricos:
  - **Balanceameno do nó u é igual a zero** (independente do balanceamento de p ter se tornado -2 ou +2).
  - Vamos chamar estas rotações de L^0 e R^0

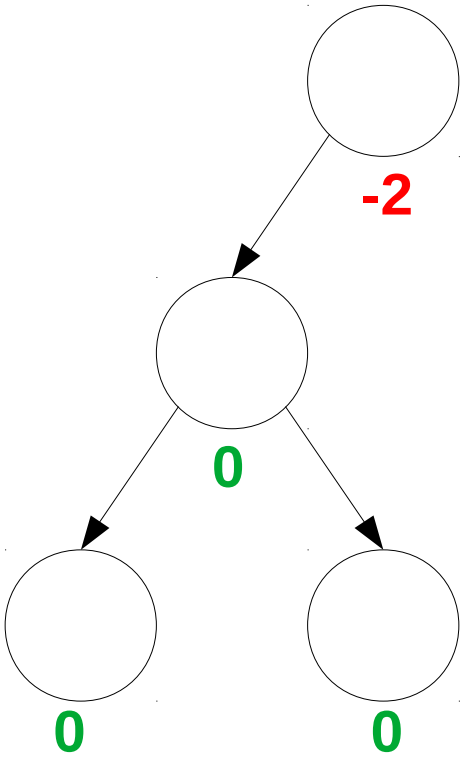
Exclusões em Árvore AVL



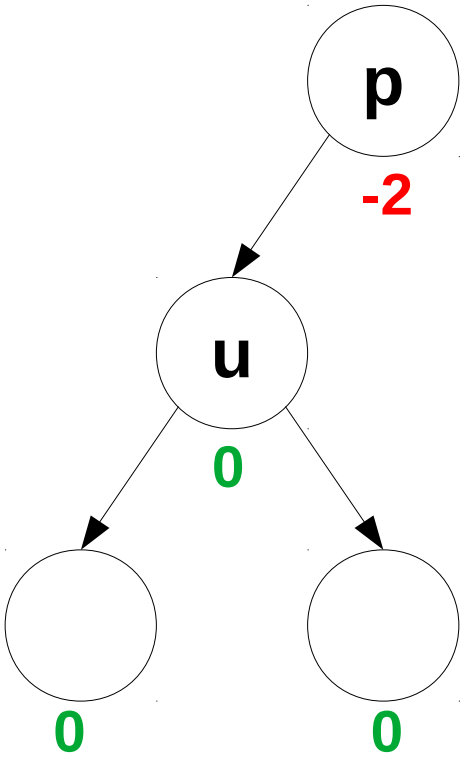
Exclusões em Árvore AVL



Exclusões em Árvore AVL

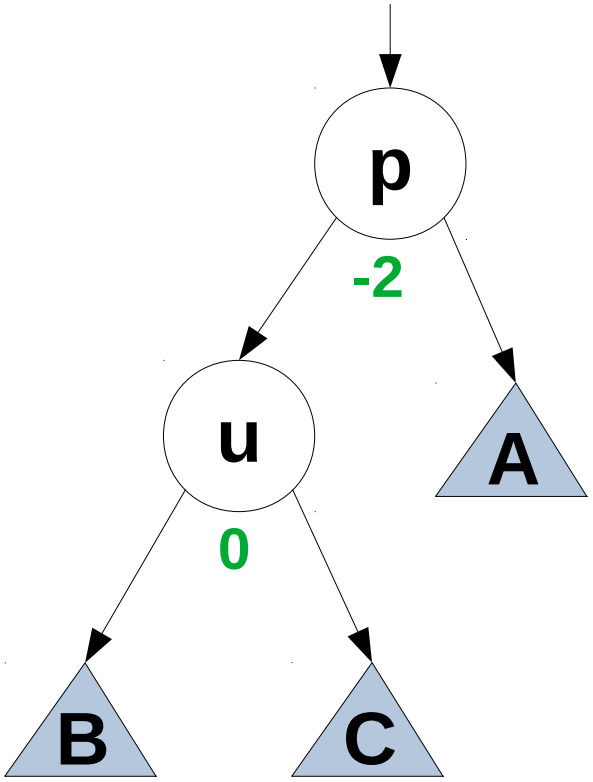


Exclusões em Árvore AVL

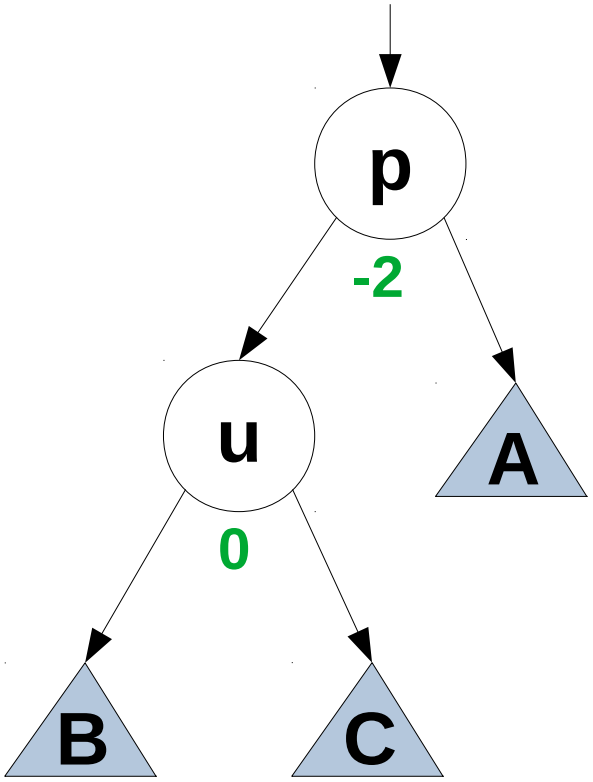


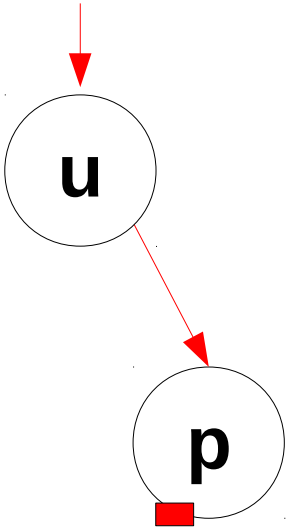
Observe que o nó u fica do lado oposto à exclusão.

Rotação L^0

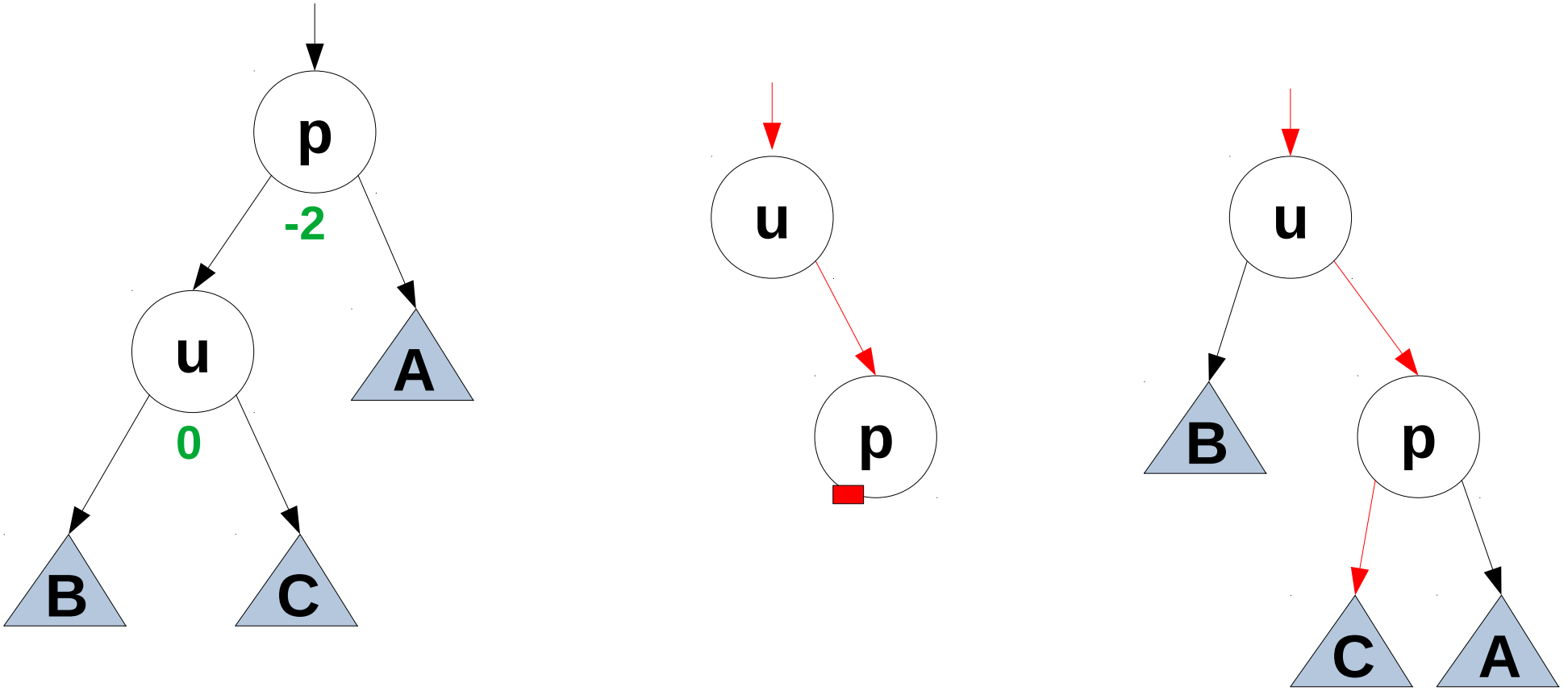


Rotação L^0

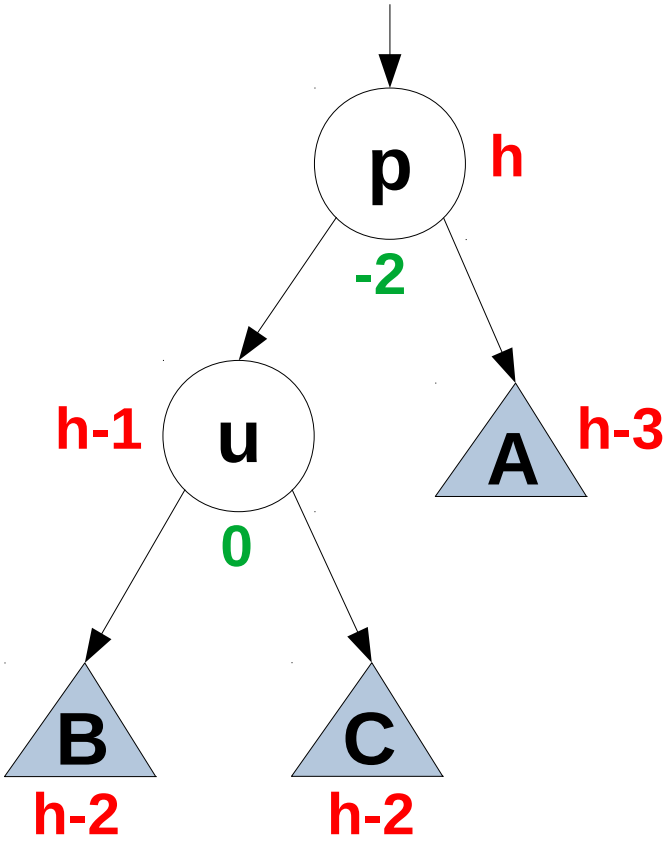




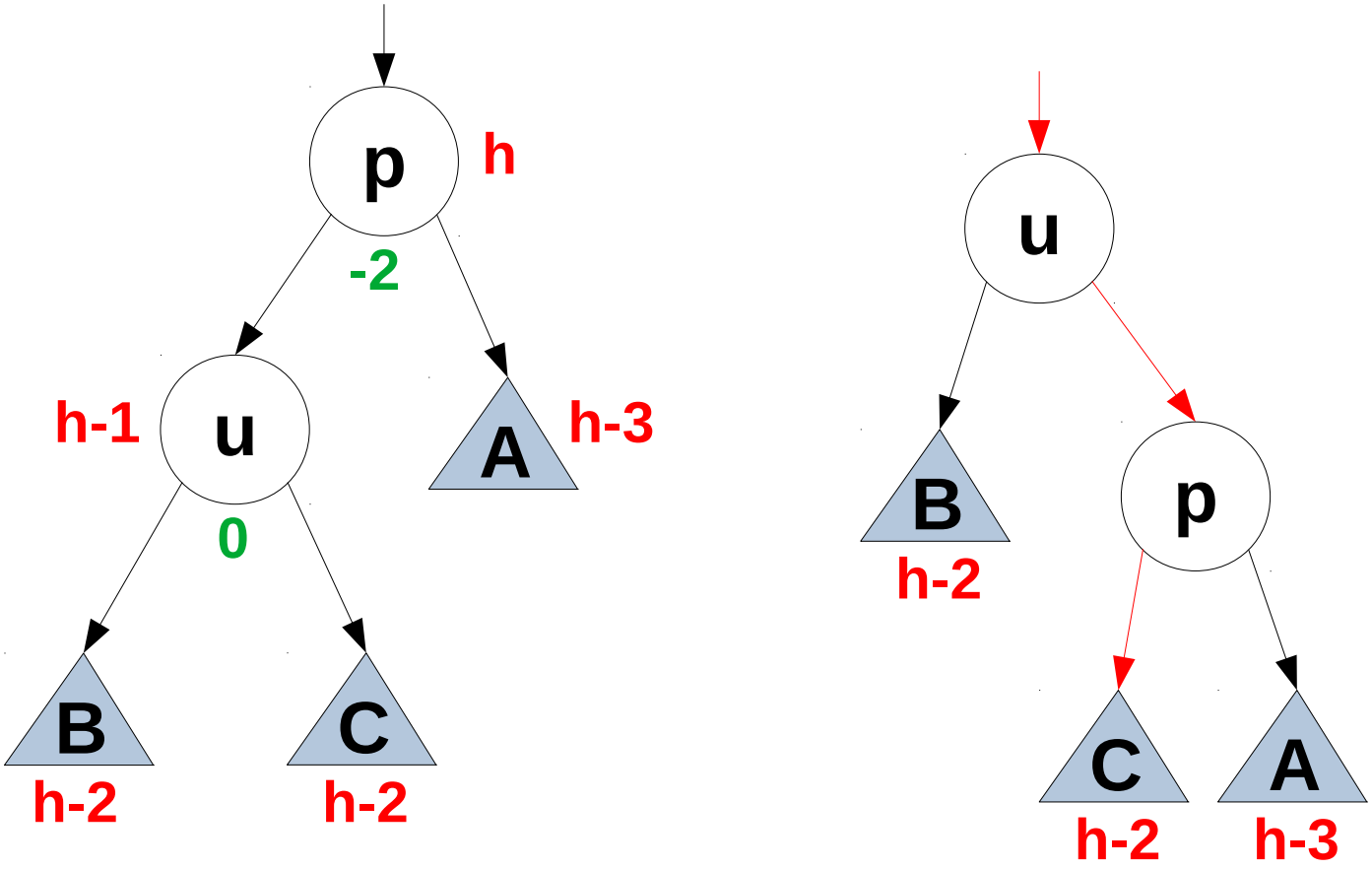
Rotação L^0



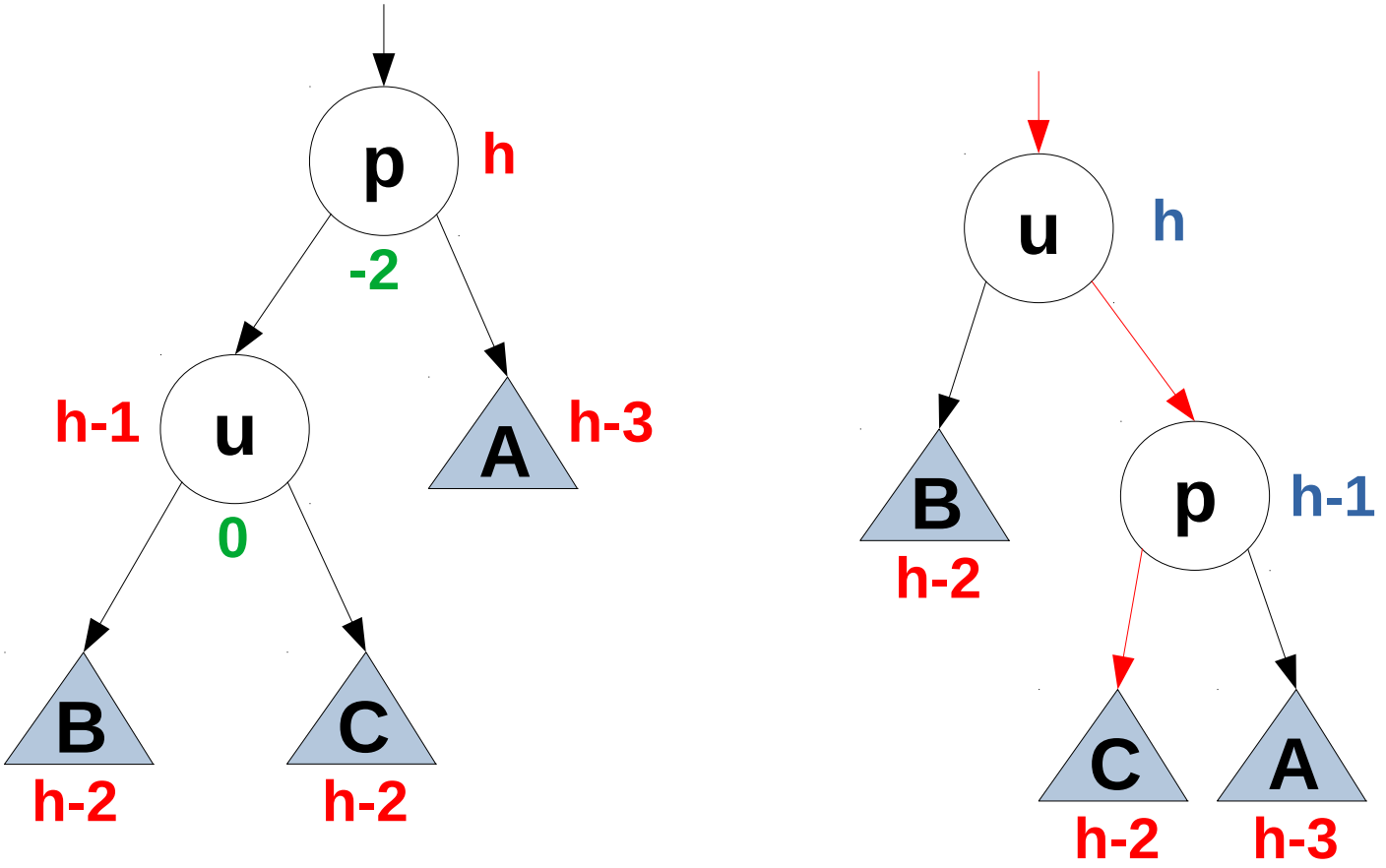
Rotação L^0 – altura dos nós



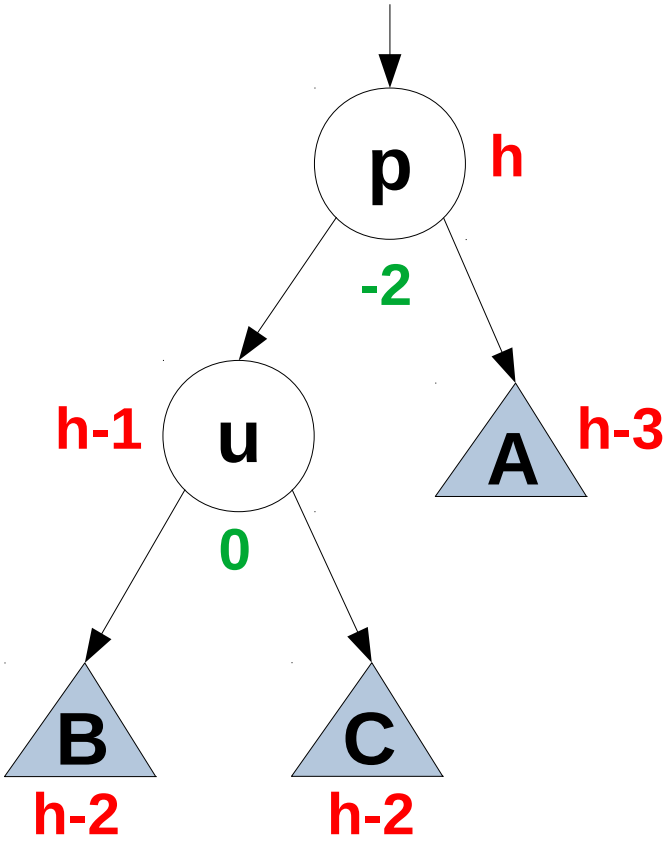
Rotação L^0 – altura dos nós



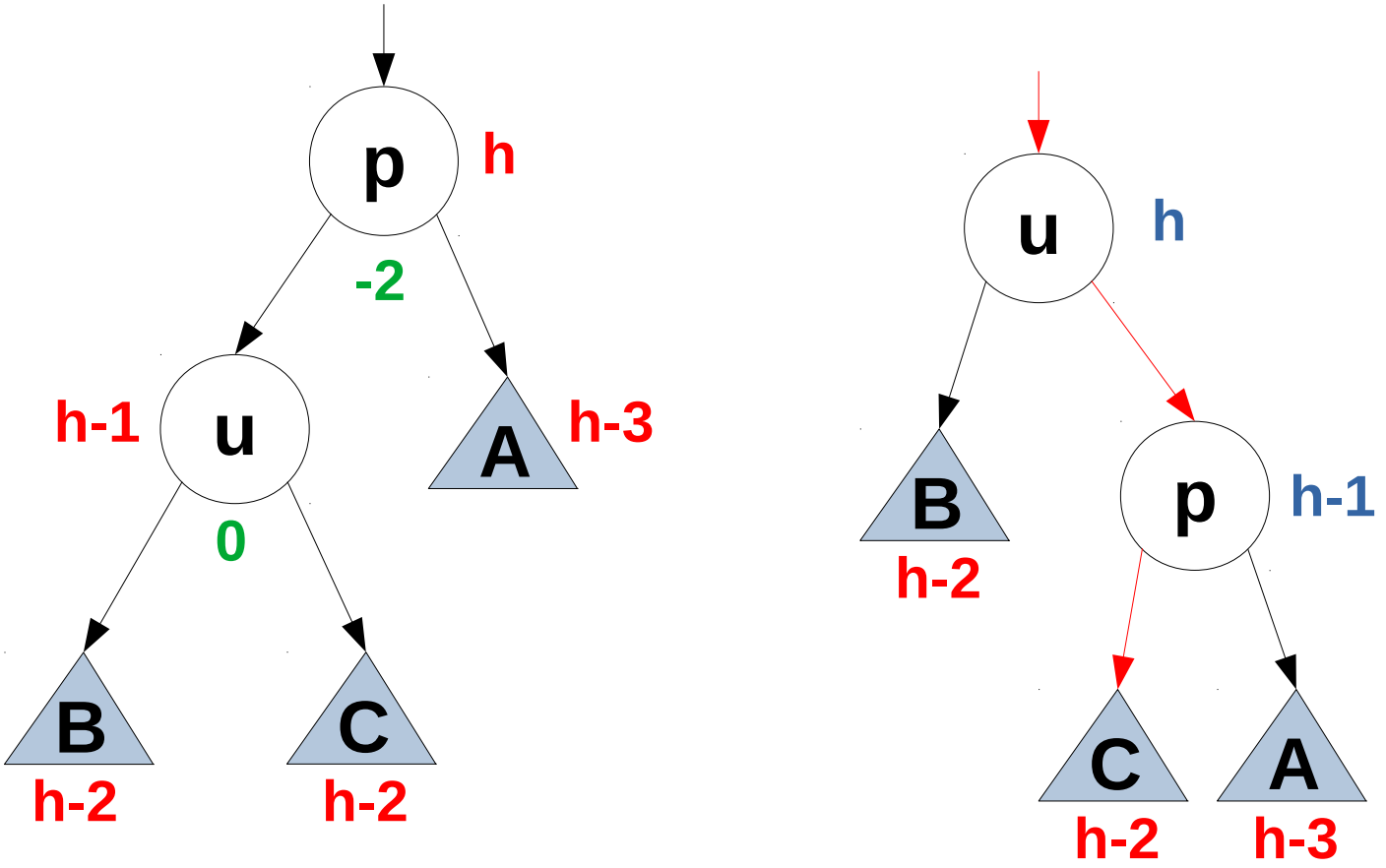
Rotação L^0 – altura dos nós



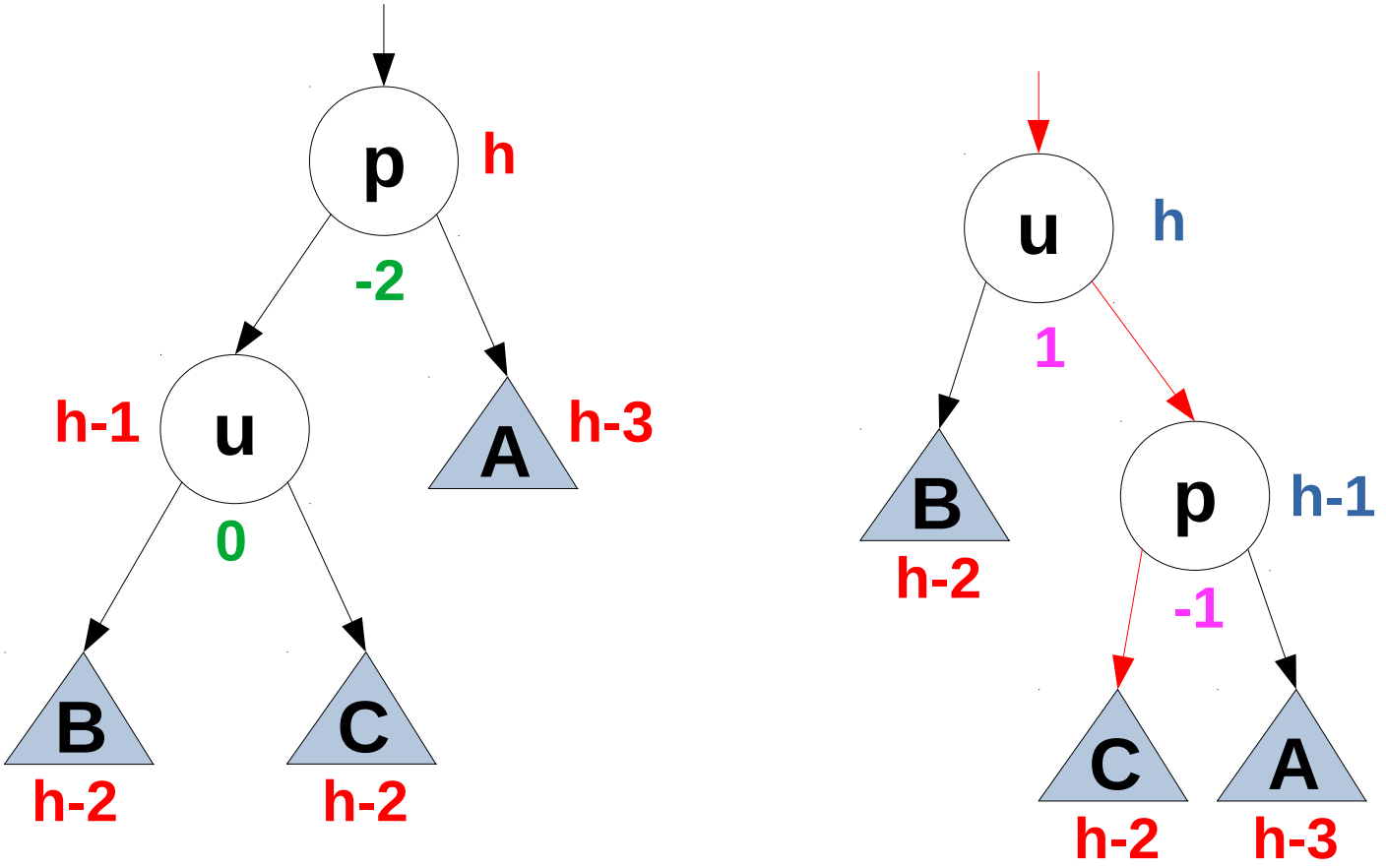
Rotação L^0 – balanceamento



Rotação L^0 – balanceamento



Rotação L^0 – balanceamento



Rotação L^0 – código

```
PONT rotacaoL(PONT p){
      PONT u, v;
      u = p->esq;
      if (u->bal == 0) {   // L0
            p->esq = u->dir;
            u->dir = p;
            // p->bal = -1;
            u->bal = 1;
            return u;
      }
```

Rotação L – Código Completo

```
PONT rotacaoL(PONT p){
      PONT u, v;
      u = p->esq;
      if(u->bal == -1) {    // LL
            p->esq = u->dir;
            u->dir = p;
            p->bal = 0;
            u->bal = 0;
            return u;
      } else if (u->bal == 1) {  // LR
            v = u->dir;
            u->dir = v->esq;
            v->esq = u;
            p->esq = v->dir;
            v->dir = p;
            if(v->bal == -1) p->bal = 1;
            else p->bal = 0;
            if(v->bal == 1) u->bal = -1;
            else u->bal = 0;
            v->bal = 0;
            return v;
      }else{    // caso necessario apenas para a exclusao (u->bal == 0)
            p->esq = u->dir;
            u->dir = p;
            // p->bal = -1;
            u->bal = 1;
            return u;
      }
}
```

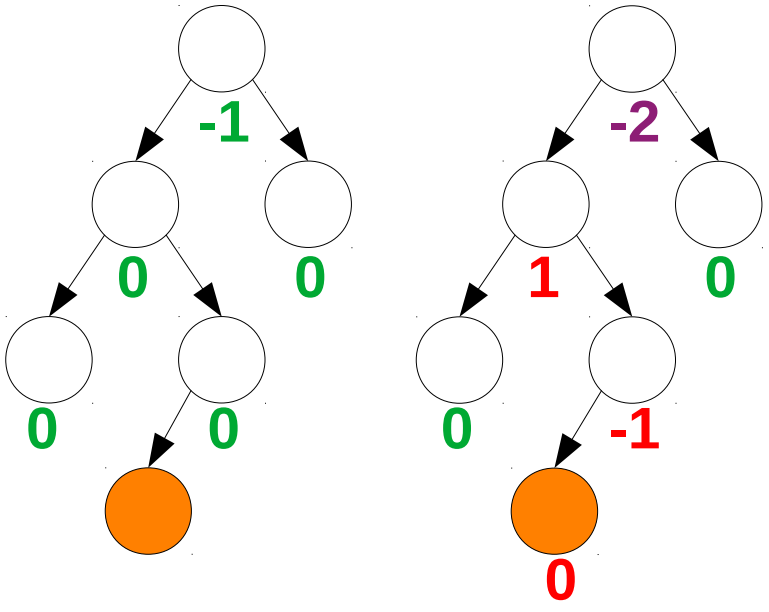
Inserção – Código Completo

```
void inserirAVL(PONT* pp, TIPOCHAVE ch, bool* alterou){
      PONT p = *pp;
      if(!p){
            *pp = criarNovoNo(ch);
            *alterou = true;
      } else {
            if(ch < p->chave) {
                  inserirAVL(&(p->esq), ch, alterou);
                  if(*alterou)
                        switch (p->bal) {
                              case 1 : p->bal = 0;
                              *alterou = false;
                              break;
                              case 0 : p->bal = -1;
                              break;
                              case -1: *pp = rotacaoL(p);
                              *alterou = false;
                              break;
                        }
            } else {
                  inserirAVL(&(p->dir), ch, alterou);
                  if(*alterou)
                        switch (p->bal) {
                              case -1: p->bal = 0;
                              *alterou = false;
                              break;
                              case 0 : p->bal = 1;
                              break;
                              case 1 : *pp = rotacaoR(p);
                              *alterou = false;
                              break;
                        }
            }
      }
}
```

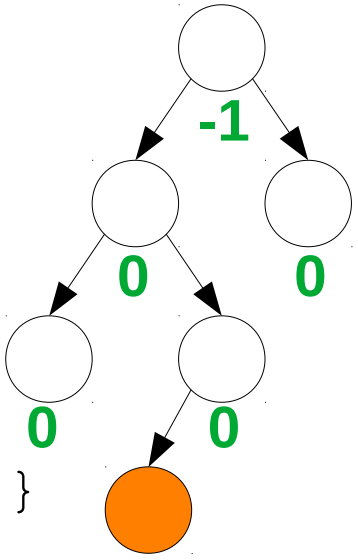
```
void inserirAVL(PONT* pp, TIPOCHAVE ch, bool* alterou){
      PONT p = *pp;
      if(!p){
            *pp = criarNovoNo(ch);
            *alterou = true;
      } else {
            if(ch < p->chave) {
                  inserirAVL(&(p->esq), ch, alterou);
                  if(*alterou)
                        switch (p->bal) {
                              case 1 : p->bal = 0;
                              *alterou = false;
                              break;
                              case 0 : p->bal = -1;
                              break;
                              case -1: *pp = rotacaoL(p);
                              *alterou = false;
                              break;
                        }
            }
```

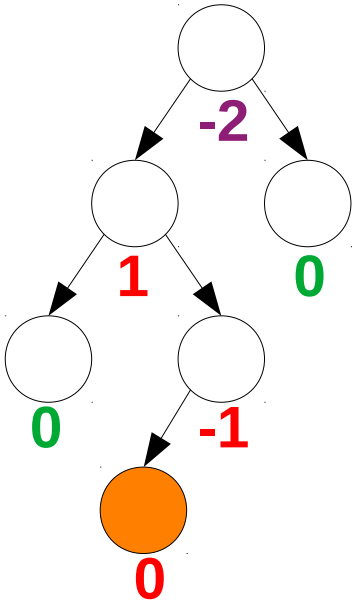
```
            } else {
                inserirAVL(&(p->dir), ch, alterou);
                if(*alterou)
                    switch (p->bal) {
                        case -1: p->bal = 0;
                        *alterou = false;
                        break;
                        case 0 : p->bal = 1;
                        break;
                        case 1 : *pp = rotacaoR(p);
                        *alterou = false;
                        break;
                    }
            }
        }
}
```

```
void inserirAVL(PONT* pp, TIPOCHAVE ch, bool* alterou){
      PONT p = *pp;
      if(!p){
            *pp = criarNovoNo(ch);
            *alterou = true;
      } else {
            if(ch < p->chave) {
                  inserirAVL(&(p->esq), ch, alterou);
                  if(*alterou)
                        switch (p->bal) {
                              case 1 : p->bal = 0;
                              *alterou = false;
                              break;
                              case 0 : p->bal = -1;
                              break;
                              case -1: *pp = rotacaoL(p);
                              *alterou = false;
                              break;
                        }
            }
```



```
} else {
        inserirAVL(&(p->dir), ch, alterou);
        if(*alterou)
                switch (p->bal) {
                        case -1: p->bal = 0;
                        *alterou = false;
                        break;
                        case 0 : p->bal = 1;
                        break;
                        case 1 : *pp = rotacaoR(p);
                        *alterou = false;
                        break;
```





}

Exclusão – Código Completo

```
bool excluirAVL(PONT* raiz, TIPOCHAVE ch, bool* alterou){
  PONT atual = *raiz;
  if (!atual) return false;
  if (atual->chave == ch){
    PONT substituto, pai_substituto;
    if(!atual->esq || !atual->dir) { // tem zero ou um filho
      if(atual->esq) substituto = atual->esq;
      else substituto = atual->dir;
      *raiz = substituto;
      free(atual);
      *alterou = true;
      return true;
    }
    else {   // tem dois filhos
      substituto = maiorAEsquerda(atual,&pai_substituto);
      atual->chave = substituto->chave;
      ch = substituto->chave;
    }
  }
  bool res;
  if (ch > atual->chave) {
    res = excluirAVL(&(atual->dir), ch, alterou);
    if (*alterou){
       switch (atual->bal){
          case 1: atual->bal = 0;
          return true;
          case 0: atual->bal = -1;
          *alterou = false;
          return true;
          case -1: *raiz = rotacaoL(atual);
          if (atual->bal != 0) *alterou = false;
          return true;
       }
    }
  }else{
    res = excluirAVL(&(atual->esq), ch, alterou);
    if (*alterou){
       switch (atual->bal){
          case -1: atual->bal = 0;
          return true;
          case 0: atual->bal = 1;
          *alterou = false;
          return true;
          case 1: *raiz = rotacaoR(atual);
          if (atual->bal != 0) *alterou = false;
          return true;
       }
    }
  }
  return res;
}
```

```
bool excluirAVL(PONT* raiz, TIPOCHAVE ch, bool* alterou){
  PONT atual = *raiz;
  if (!atual) return false;
  if (atual->chave == ch){
    PONT substituto, pai_substituto;
    if(!atual->esq || !atual->dir) { // tem zero ou um filho
      if(atual->esq) substituto = atual->esq;
      else substituto = atual->dir;
      *raiz = substituto;
      free(atual);
      *alterou = true;
      return true;
    }
    else {   // tem dois filhos
      substituto = maiorAEsquerda(atual,&pai_substituto);
      atual->chave = substituto->chave;
      ch = substituto->chave;
    }
  }
```

```
bool res;
if (ch > atual->chave) {
   res = excluirAVL(&(atual->dir), ch, alterou);
   if (*alterou){
      switch (atual->bal){
         case 1: atual->bal = 0;
         return true;
         case 0: atual->bal = -1;
         *alterou = false;
         return true;
         case -1: *raiz = rotacaoL(atual);
         if (atual->bal != 0) *alterou = false;
         return true;
      }
   }
}
```

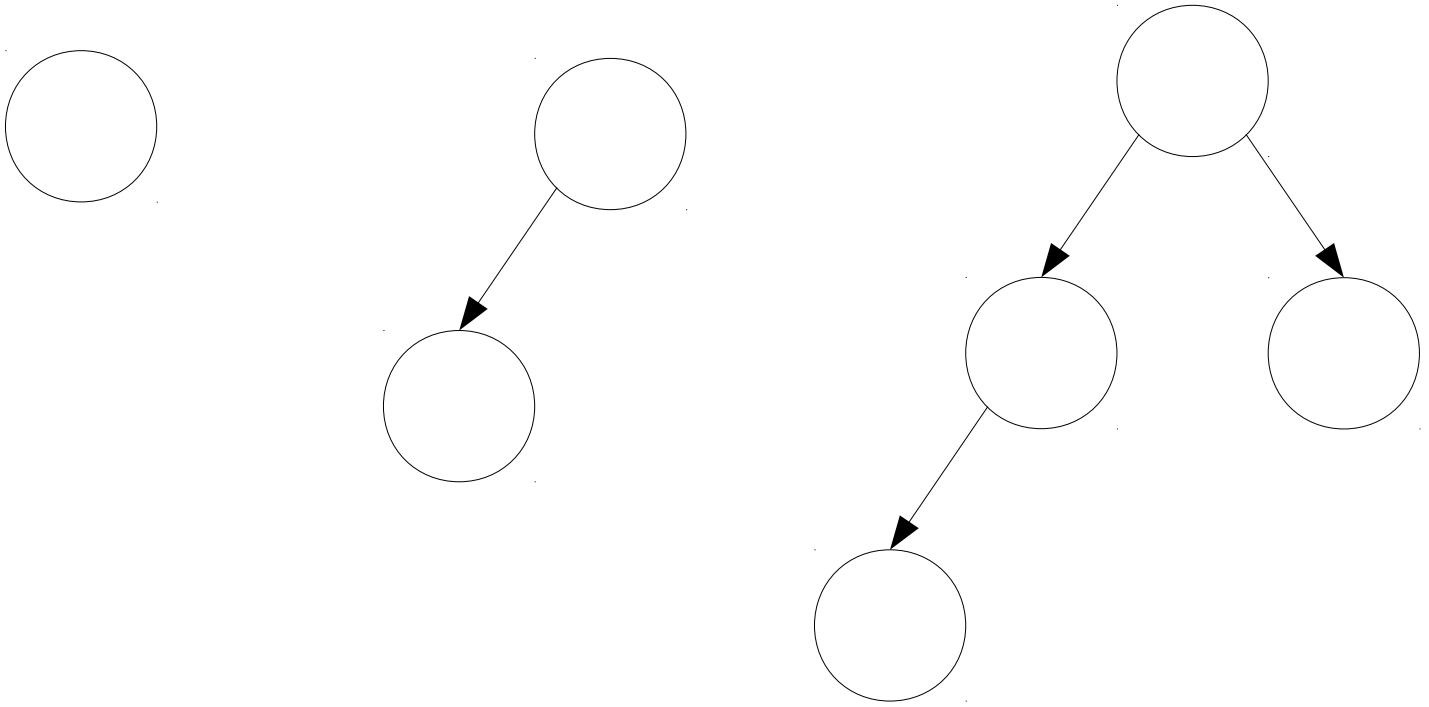
```
else{
    res = excluirAVL(&(atual->esq), ch, alterou);
    if (*alterou){
        switch (atual->bal){
            case -1: atual->bal = 0;
            return true;
            case 0: atual->bal = 1;
            *alterou = false;
            return true;
            case 1: *raiz = rotacaoR(atual);
            if (atual->bal != 0) *alterou = false;
            return true;
        }
    }
}
```

Árvores AVL

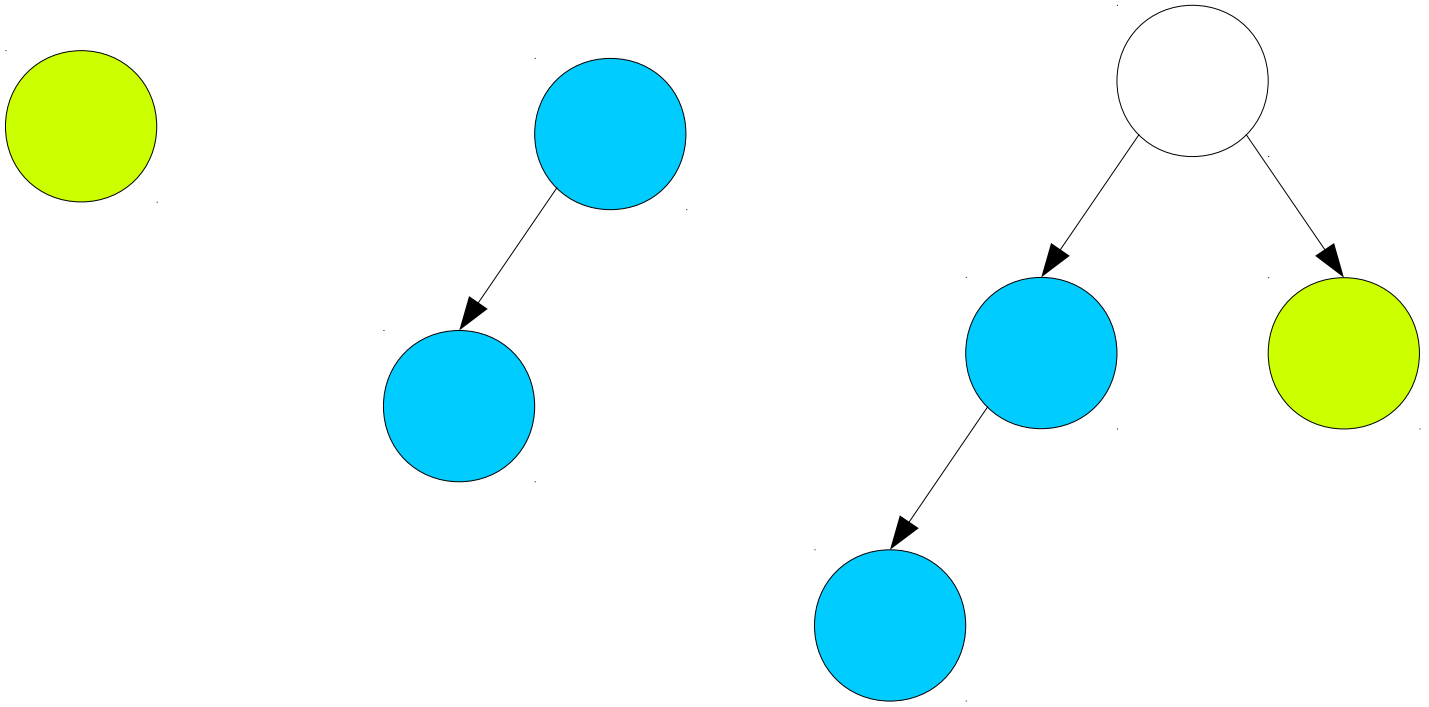
Mas a altura de uma árvore AVL **pertence a O(log n)?**

Vamos verificar quais são as árvores AVL com menor número de nós (para uma dada altura).

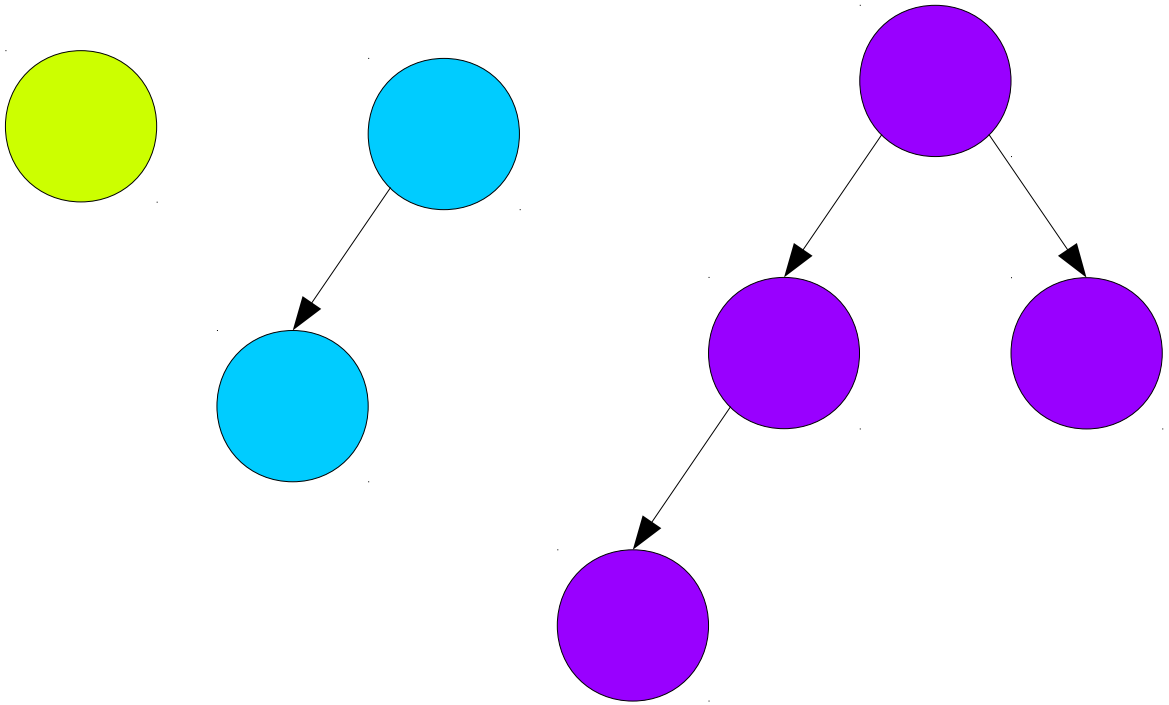
Árvores AVL



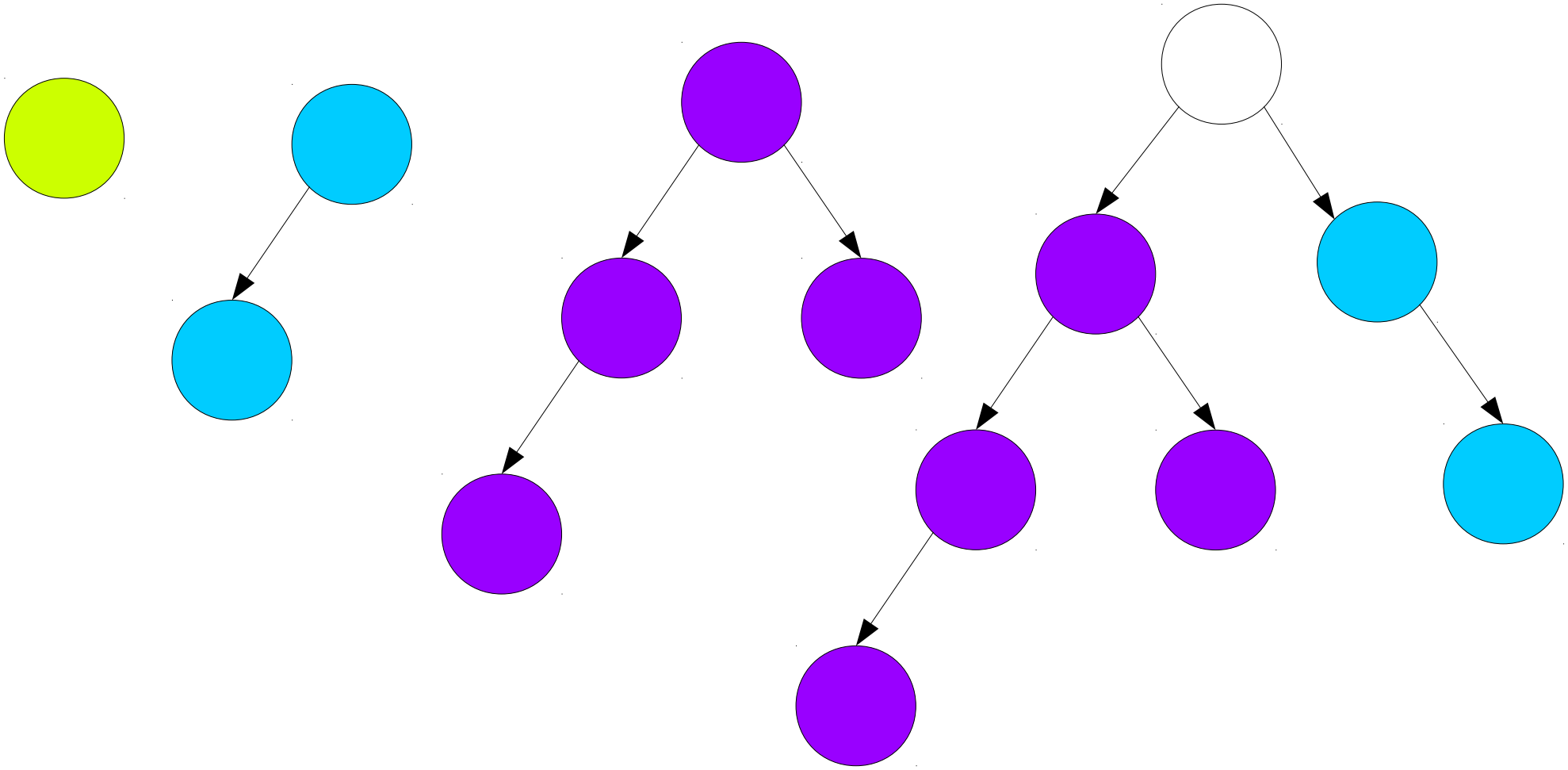
Árvores AVL



Árvores AVL



Árvores AVL



Árvores AVL

O menor número de nós de uma árvore AVL com altura h será:

nós(h) = 1 + nós(h-1) + nós(h-2)

Árvores AVL

O menor número de nós de uma árvore AVL com altura h será:

$min\_nós(h) = 1 + min\_nós(h-1) + min\_nós(h-2)$

$min\_nós(h) \approx 1{,}618034^{h}$

Árvores AVL

Para uma árvore binária completa:

$2^h \leq n < 2^{h+1}$

Para uma árvore AVL:

$1{,}618034^h \leq n < 2^{h+1}$

Assim, a altura será $\lfloor \log_2 n \rfloor \leq h_{AVL} < \log_{1{,}618034} n$

No máximo a altura será igual a $1{,}4404 * \log_2 n$

Árvores AVL

Prof. Luciano Antonio Digiampietri