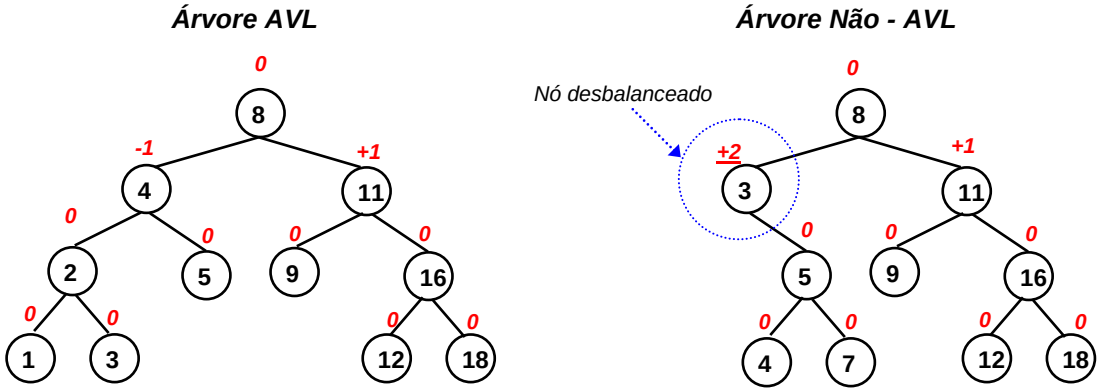
Árvores AVL

G.M. **A**delson-**V**elskii e E.M. **L**andis (1962) apresentaram uma árvore de busca binária que é balanceada com respeito a altura das sub-árvores.
Uma árvore AVL é uma árvore balanceada cuja diferença em altura entre a sub-árvore esquerda e a sub-árvore direita (**em qualquer nó**) deve ser, no máximo, de um nível.
Numa árvore AVL, cada chave é única e as operações de inserção/ remoção tentam manter o equilíbrio da árvore.
Definição: Uma árvore binária vazia é sempre balanceada por altura. Se **T** não é vazia e T_{esq} e T_{dir} são sub-árvores da esquerda e direita, então **T** é balanceada por altura se:

1. T_{esq} e T_{dir} são balanceadas por altura;
2. h_D-h_E = ±1, sendo h_D e h_E a altura de T_{esq} e T_{dir}, respectivamente.

Habitualmente, em cada nó, existe um campo adicional que indica a situação de equilíbrio desse nó, chamado de fator de balanceamento. Entretanto, a cada inserção ou remoção o valor do fator de balanceamento de alguns nós precisar ser alterado.

- ***0*** : equilibrada;
- ***−1*** : sub-árvore esquerda tem mais um nível que a da direita;
- ***1*** : sub-árvore direita tem mais um nível que a da esquerda.



Definição: O fator de balanceamento de um determinado nó em uma árvore binária é definido como sendo h_D-h_E, sendo h_D e h_E a altura da sub-árvore da direita e esquerda respectivamente.

Estrutura do Nó para a Árvore AVL

A estrutura do nó agora necessita de mais alguns atributos de controle, necessários para o controle de balanceamento da árvore AVL.

Esq	*Elem*	***FatBal***	***Pai***	*Dir*

Novos atributos:

- **FatBal:** armazena o fator de balanceamento daquele nó dentro da árvore AVL.
- **Pai:** guarda a referência para o nó pai. No caso da raiz da árvore, o pai é igual a **null.**

```
typedef struct no_AVL AVL;

struct no_AVL {
      int info;
      int fb; // fator de balanceamento
      AVL *pai;
      AVL *esq;
      AVL *dir;
};
```

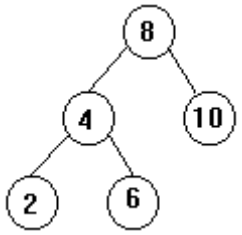
Inserção em uma Árvore AVL

O que pode acontecer quando um novo nó é inserido numa árvore balanceada ?

Dada uma raiz r com subárvores L (left) e R (right), e supondo que a inserção deve ser feita na sub-árvore da esquerda. Podemos distriguir 3 casos:

1. Se hL = hR, então L e R ficam com alturas diferentes mas continuam balanceadas.
2. Se hL < hR, então L e R ficam com alturas iguais e balanceamento foi melhorado.
3. Se hL > hR, então L fica ainda maior e balanceamento foi violado.

Na árvore abaixo:



- Nós 9 ou 11 podem ser inseridos sem balanaceamento . Subárvore com raiz 10 passa a ter uma subárvore e subárvore com raiz 8 vai ficar melhor balanceada !

- Inserção dos nós 3, 5 ou 7 requerem que a árvore seja rebalanceada!

Fator de Balanceamento (FB) de um nó

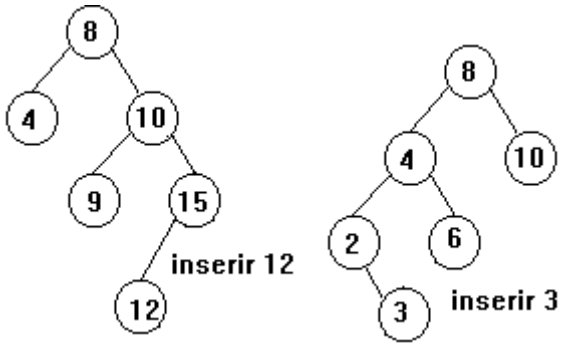
É a altura da subárvore direita do nó menos a altura da subárvore esquerda do nó

```
int altura(Arvore t)
{
    int altE, altD:
    AVL *q, *temp;
    if (t == NULL)
        return -1;
    else
    {
       altE = altura(t->esq);
       altD = altura(t->dir);
       if (altE < altD)
         return altD + 1;
       else
         return altE + 1;
    }
}
```

Rebalanceamento:

Os problemas podem ser mapeados para dois casos:

Tipo 1: o nó raiz de uma subárvore tem FB 2 (ou -2) e tem um filho com FB 1 (-1) o qual tem o mesmo sinal que o FB do nó pai. Exemplos:



- solução: rotação simples sobre o nó de FB=2 (-2).
- Rotações são feitas à esquerda quando FB positivo e à direita quando FB negativo.

Algoritmo para rotação à direita sobre o nó p

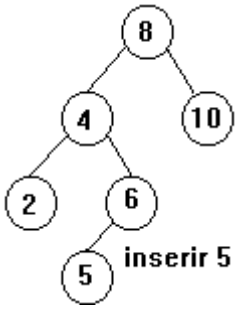
```
void rotacao_direita(AVL *p)
{
    AVL *q, *temp;
    q = p->esq;
    temp = q->dir;
    q->dir = p;
    p->esq = temp;
}
```

Algoritmo para rotação à esquerda sobre o nó p

```
void rotacao_esquerda(AVL *p)
{
    AVL *q, *temp;
    q = p->dir;
    temp = q->esq;
    q->esq = p;
    p->dir = temp;
}
```

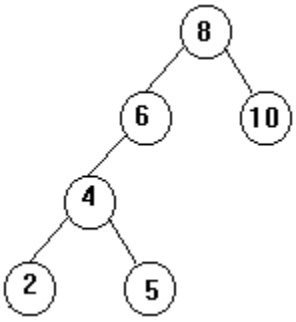
Tipo 2: o nó raiz de uma subárvore tem FB=2 (ou -2) e tem uma um filho com FB=-1 (1) o qual tem o sinal oposto ao FB do nó pai.

Exemplo: Caso (-2) (1)
FB do nó que contém 8: -2
FB do nó que contém 4: 1

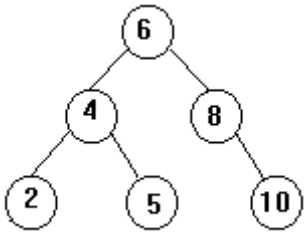


- **solução: duas rotações**
  1. primeiro roda-se o nó com FB=1 (-1) na direção apropriada
  2. depois roda-se o nó que tinha FB=-2 (2) na direção oposta

Rotação de 4 à esquerda



Rotação de 8 à direita



Exercícios

1. Implemente as rotações em uma árvore AVL.
2. Implemente a operação de inserção em uma árvore AVL.
3. Faça o teste usando a árvore AVL acima identificando as rotações aplicadas a cada nível.
4. Faça a simulação da árvore AVL inserindo a seqüência de números a seguir:
   `10, 15, 2, 20, 12, 14, 3, 6, 1, 23`
5. Utilizando a árvore gerada no exercício anterior, realize a remoção dos seguintes nós:
   `2, 20, 14, 15.`