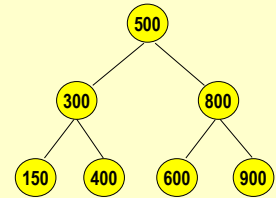


Árvores Balanceadas

Estruturas de Dados - Árvores

Árvores Binárias de Pesquisa

- Apresentam uma relação de **ordem**
- A ordem é definida pela **chave**
- Operações:
 - inserir
 - consultar
 - excluir



Estruturas de Dados - Árvores

Problemas com ABP

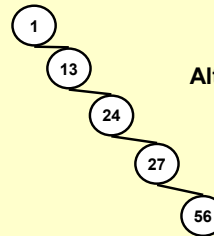
Exemplo:

- Inserção: 10, 5, 15, 20, 25, 30, 35
- Inserção: 1, 13, 24, 27, 56

Estruturas de Dados - Árvores

Problemas com ABP

- Desbalanceamento progressivo
- Exemplo:
 - inserção: 1, 13, 24, 27, 56



Alternativa de solução:

- **Árvores balanceadas**
- **AVL**

Estruturas de Dados - Árvores

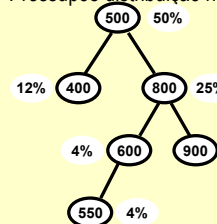
Balanceamento de Árvores

- Distribuição equilibrada dos nós
 - otimizar as **operações de consulta**
 - diminuir o número médio de comparações
- Distribuição
 - **uniforme**
 - **não uniforme**
 - chaves mais solicitadas mais perto da raiz

Estruturas de Dados - Árvores

Por Frequência

- Por frequência de acesso
 - Pressupõe distribuição não uniforme de acessos

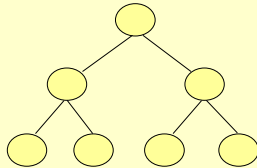


Balanceamento por distribuição de acessos!

Estruturas de Dados - Árvores

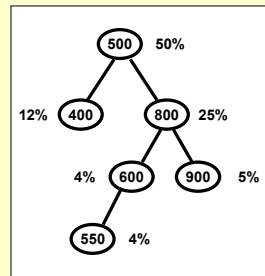
Árvores balanceadas por ALTURA

Uma árvore binária é **completamente balanceada** se a distância média dos nodos até a raiz for mínima

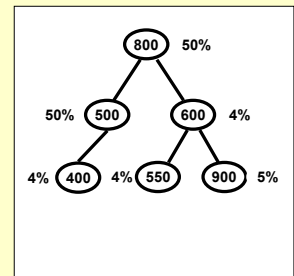


Estruturas de Dados - Árvores

Por Frequência X Por Altura



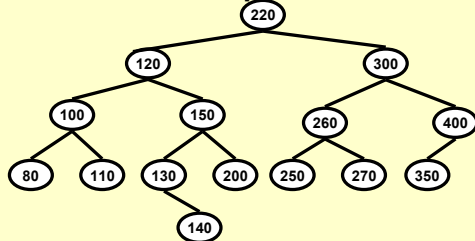
Splay



AVL
Rubro-Negras

Estruturas de Dados - Árvores

Balanceamento por ALTURA



Árvore **não completamente** balanceada

Estruturas de Dados - Árvores

Árvores AVL

Adelson-Velskii e Landis (1962)

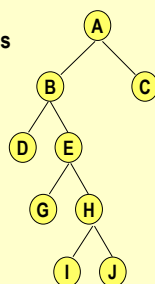
Uma **árvore AVL** é uma **árvore binária de pesquisa (ABP)** construída de tal modo que a altura de sua subárvore direita difere da altura da subárvore esquerda de no **máximo 1**.

Estruturas de Dados - Árvores

Árvores balanceadas por altura

HB(k)-Tree → Height-Balanced k-Tree

- árvore binária
- para qualquer nodo, as alturas de suas duas subárvores não diferem de mais do que k unidades
- cada uma das subárvores do nodo apresenta a propriedade **FATOR(k)**



Estruturas de Dados - Árvores

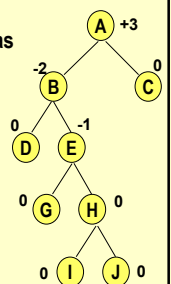
Árvores balanceadas por altura

HB(k)-Tree → Height-Balanced k-Tree

- árvore binária
- para qualquer nodo, as alturas de suas duas subárvores não diferem de mais do que k unidades
- cada uma das subárvores do nodo apresenta a propriedade **FATOR(k)**

Ex: verificar se a árvore ao lado é

FATOR(1) → Não
FATOR(2) → Não
FATOR(3) → Sim



Estruturas de Dados - Árvores

Árvores AVL

Adelson-Velskii e Landis (1962)

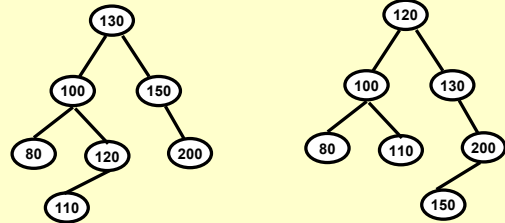
- árvores **FATOR(1)** são chamadas **Árvores AVL**

Uma **árvore AVL** é uma **árvore binária de pesquisa (ABP)** construída de tal modo que a altura de sua subárvore direita difere da altura da subárvore esquerda de no **máximo 1**.

Estruturas de Dados - Árvores

Exercício:

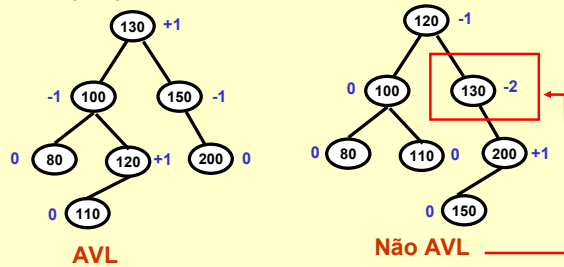
Verifique quais das ABP são AVL:



Estruturas de Dados - Árvores

Exercício: Resposta

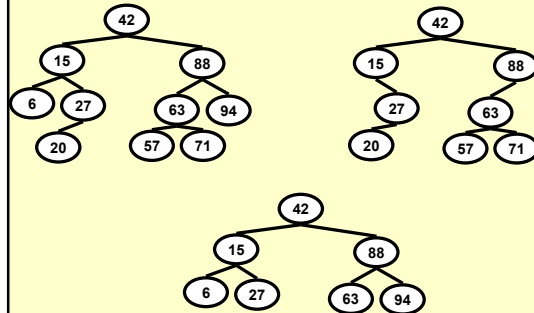
Verifique quais das ABP são AVL:



Estruturas de Dados - Árvores

Exercício

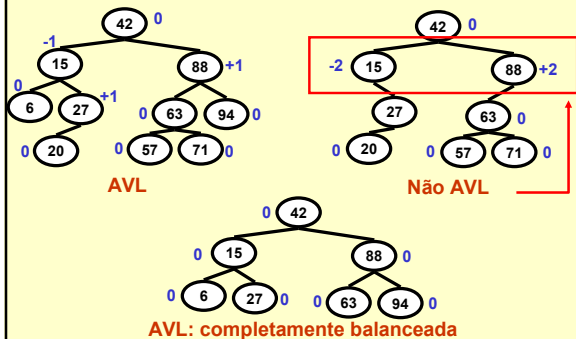
Verifique quais das ABP são AVL:



Estruturas de Dados - Árvores

Exercício: Resposta

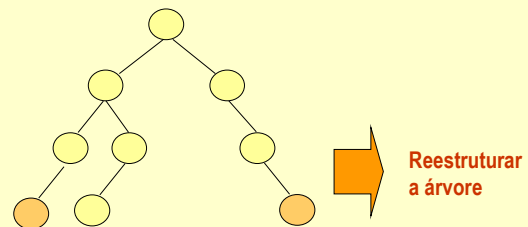
Verifique quais das ABP são AVL:



Estruturas de Dados - Árvores

Operações

- por exemplo: **INSERÇÃO**
- deve ser preservada a propriedade AVL



Estruturas de Dados - Árvores

Operações

- Como manter uma árvore AVL sempre balanceada após uma **inserção** ou **exclusão**?
 - Através de uma operação de **ROTAÇÃO**
- Característica da operação
 - preservar a ordem das chaves
 - basta uma execução da operação de rotação para tornar a árvore AVL novamente

Estruturas de Dados - Árvores

Balanceamento de Árvore AVL com Rotação

☞ Rotação Simples

- à direita
- à esquerda

☞ Rotação Dupla

- à direita
- à esquerda

Estruturas de Dados - Árvores

Rotação Simples DIREITA

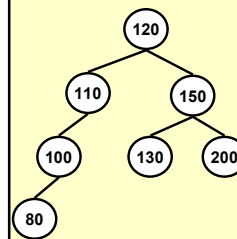
Toda vez que uma subárvore fica com um fator:

- **positivo** e sua subárvore da esquerda também tem um fator **positivo**

ROTAÇÃO SIMPLES À DIREITA

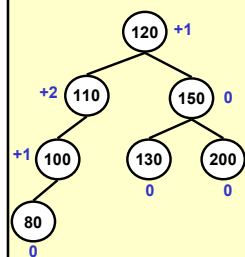
Estruturas de Dados - Árvores

Rotação Direita



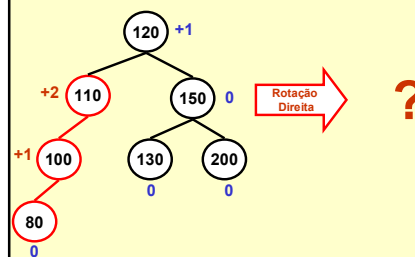
Estruturas de Dados - Árvores

Rotação Direita



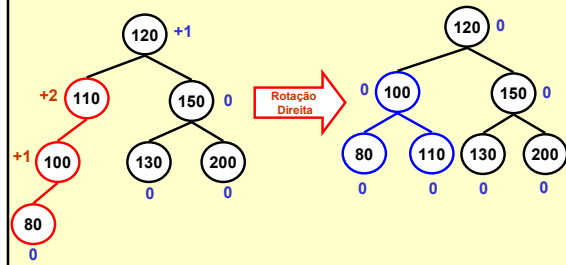
Estruturas de Dados - Árvores

Rotação Direita



Estruturas de Dados - Árvores

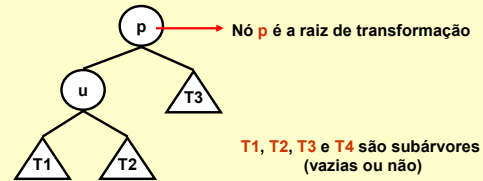
Rotação Direita



Estruturas de Dados - Árvores

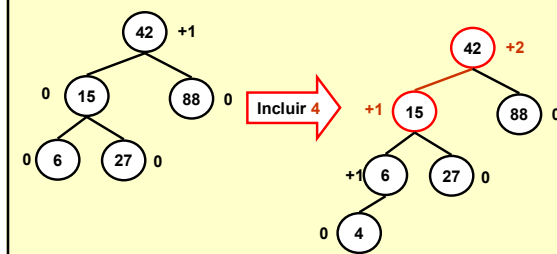
Balanceamento de Árvore AVL com Rotação

Rotação Simples
• à direita



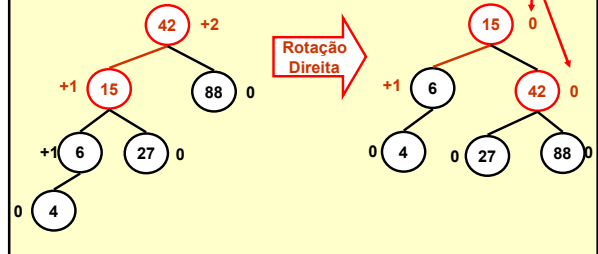
Estruturas de Dados - Árvores

Rotação Direita



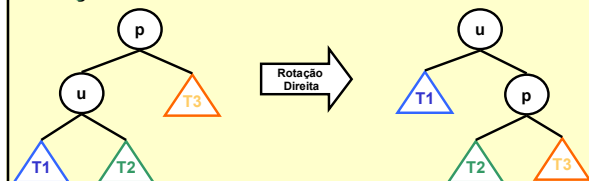
Estruturas de Dados - Árvores

Rotação Direita



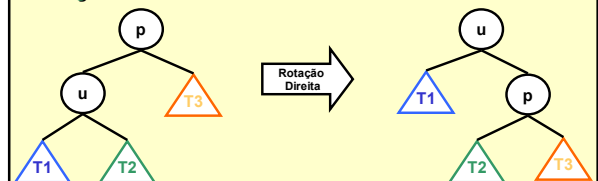
Estruturas de Dados - Árvores

Rotação Direita



Estruturas de Dados - Árvores

Rotação Direita



```
struct TNodoA{
    TipInfo info;
    int FB;
    struct TNodoA *esq;
    struct TNodoA *dir;
};
typedef struct TNodoA pNodoA;
```

```
pNodoA* rotacao_direita(pNodoA* pt){
    pNodoA *ptu;
    ptu = pt->esq;
    pt->esq = ptu->dir;
    ptu->dir = pt;
    pt->FB = 0;
    pt = ptu;
    return pt;
}
```

Estruturas de Dados - Árvores

Rotação Simples ESQUERDA

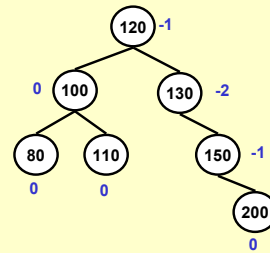
Toda vez que uma subárvore fica com um fator:

- **negativo** e sua subárvore da direita também tem um fator **negativo**

ROTAÇÃO SIMPLES À ESQUERDA

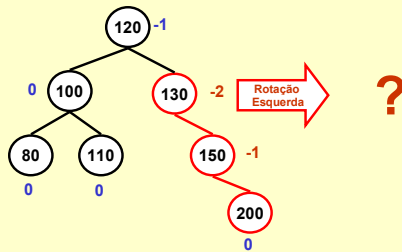
Estruturas de Dados - Árvores

Rotação Esquerda



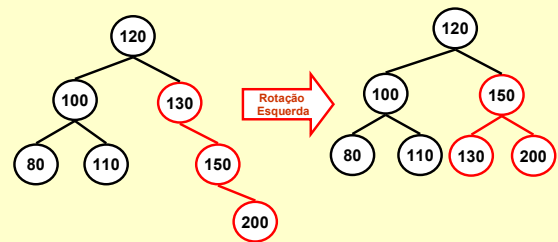
Estruturas de Dados - Árvores

Rotação Esquerda



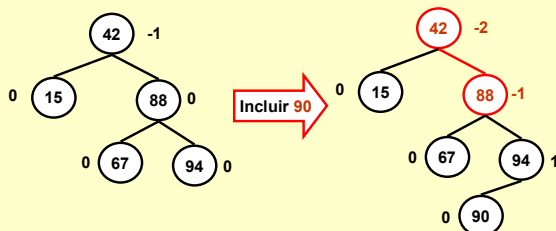
Estruturas de Dados - Árvores

Rotação Esquerda



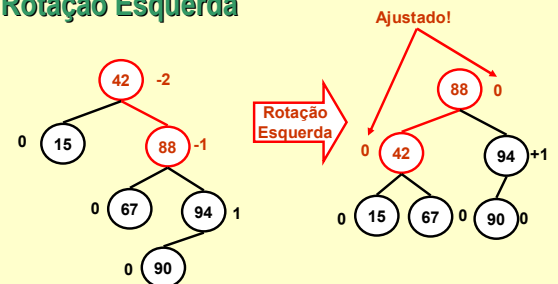
Estruturas de Dados - Árvores

Rotação Esquerda



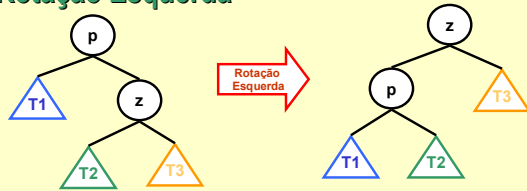
Estruturas de Dados - Árvores

Rotação Esquerda



Estruturas de Dados - Árvores

Rotação Esquerda



Estruturas de Dados - Árvores

Rotação Esquerda



```
struct TNodeA{
    TipInfo info;
    int FB;
    struct TNodeA *esq;
    struct TNodeA *dir;
};
typedef struct TNodeA pNodeA;
```

```
pNodeA* rotacao_esquerda(pNodeA *pt){
    pNodeA *ptu;
    ptu = pt->dir;
    pt->dir = ptu->esq;
    ptu->esq = pt;
    pt->FB = 0;
    pt = ptu;
    return pt;
}
```

Estruturas de Dados - Árvores

Rotação Dupla DIREITA

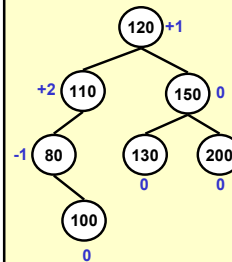
Toda vez que uma subárvore fica com um fator:

- **positivo** e sua subárvore da esquerda tem um fator **negativo**

ROTAÇÃO DUPLA À DIREITA

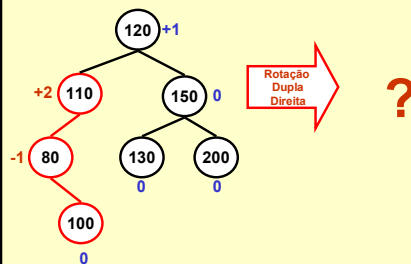
Estruturas de Dados - Árvores

Rotação Dupla Direita



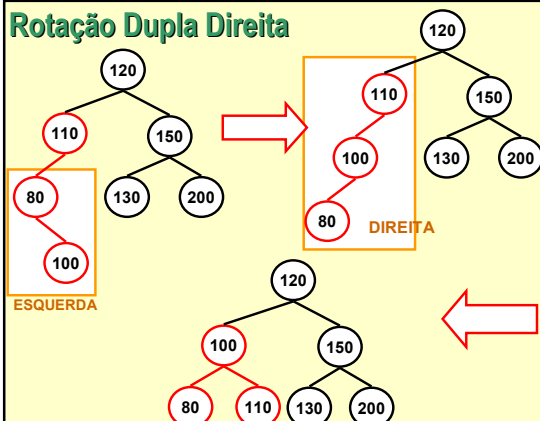
Estruturas de Dados - Árvores

Rotação Dupla Direita



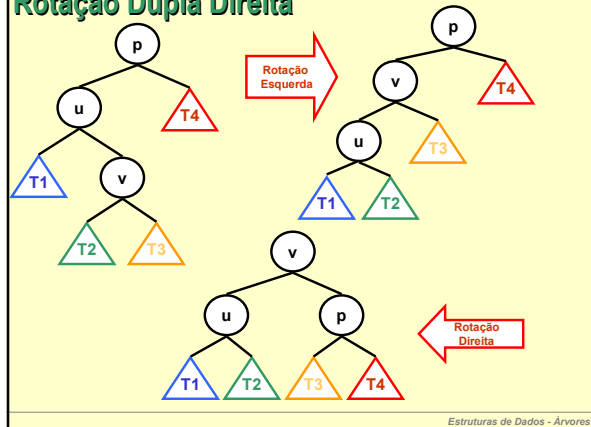
Estruturas de Dados - Árvores

Rotação Dupla Direita



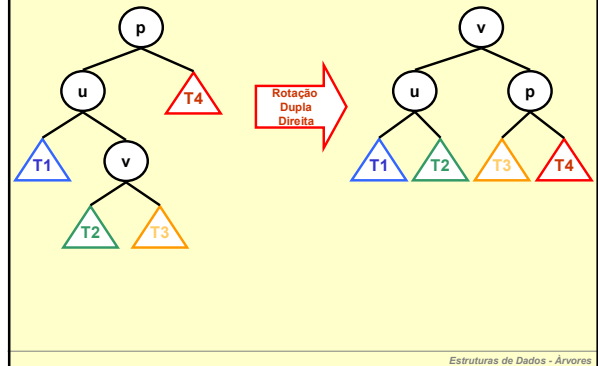
Estruturas de Dados - Árvores

Rotação Dupla Direita



Estruturas de Dados - Árvores

Rotação Dupla Direita



Estruturas de Dados - Árvores

Rotação Dupla Direita

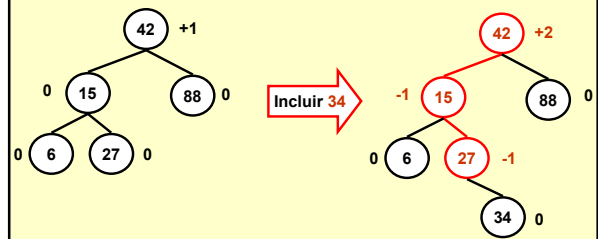
```
struct TNodeA{
    TipInfo info;
    int FB;
    struct TNodeA *esq;
    struct TNodeA *dir;
};
typedef struct TNodeA pNodeA;
```

```
pNodeA* rotacao_dupla_direita (pNodeA* pt){
    pNodeA *ptu, *ptv;
```

```
    ptu = pt->esq;
    ptv = ptu->dir;
    ptu->dir = ptv->esq;
    ptv->esq = ptu;
    pt->esq = ptv->dir;
    ptv->dir = pt;
    if (ptv->FB == 1) pt->FB = -1;
    else pt->FB = 0;
    if (ptv->FB == -1) ptu->FB = 1;
    else ptu->FB = 0;
    pt = ptv;
    return pt;
}
```

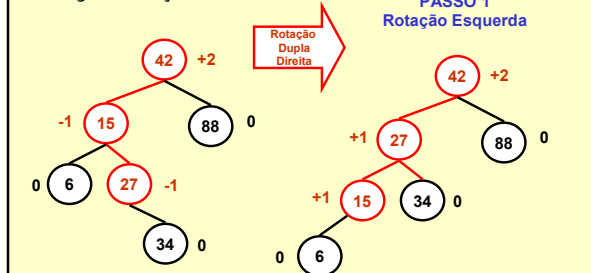
Estruturas de Dados - Árvores

Rotação Dupla Direita



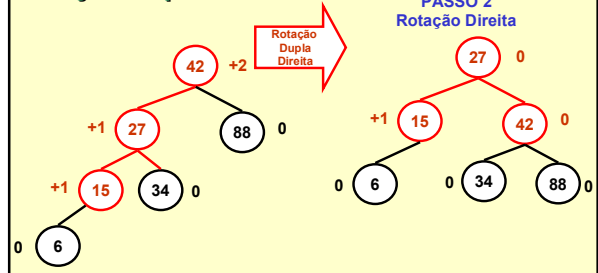
Estruturas de Dados - Árvores

Rotação Dupla Direita



Estruturas de Dados - Árvores

Rotação Dupla Direita



Estruturas de Dados - Árvores

Rotação Dupla ESQUERDA

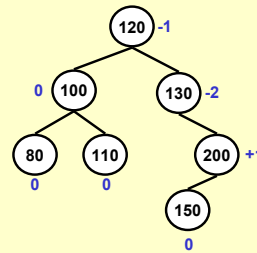
Toda vez que uma subárvore fica com um fator:

- **negativo** e sua subárvore da direita tem um fator **positivo**

ROTAÇÃO DUPLA À ESQUERDA

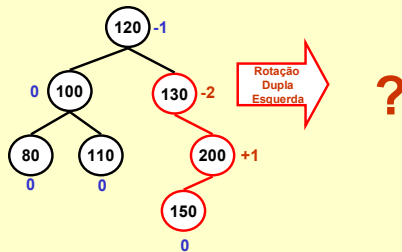
Estruturas de Dados - Árvores

Rotação Dupla Esquerda



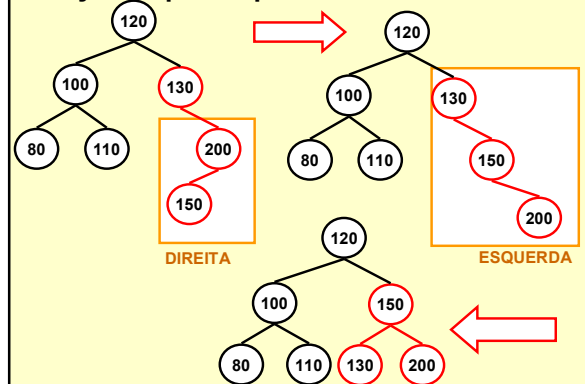
Estruturas de Dados - Árvores

Rotação Dupla Esquerda



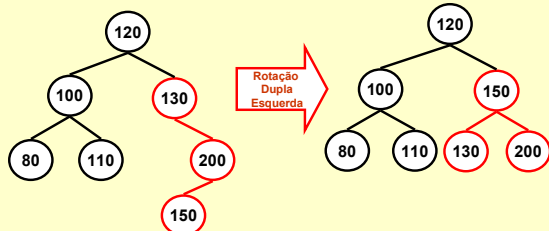
Estruturas de Dados - Árvores

Rotação Dupla Esquerda



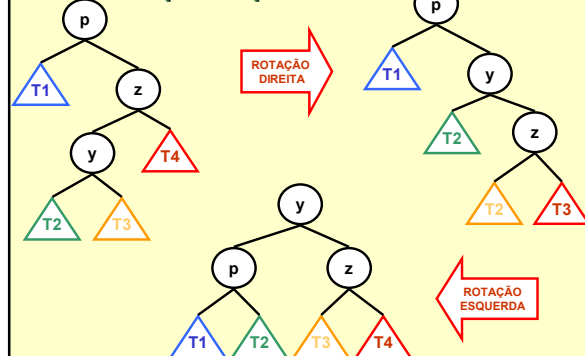
Estruturas de Dados - Árvores

Rotação Dupla Esquerda



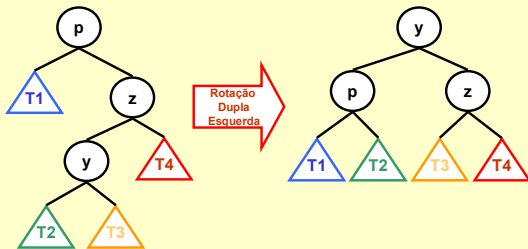
Estruturas de Dados - Árvores

Rotação Dupla Esquerda



Estruturas de Dados - Árvores

Rotação Dupla Esquerda



Estruturas de Dados - Árvores

Rotação Dupla Esquerda

```
struct TNodeA{
    TipoInfo info;
    int FB;
    struct TNodeA *esq;
    struct TNodeA *dir;
};
typedef struct TNodeA pNodeA;
```

```
pNodeA rotacao_dupla_esquerda (pNodeA *pt){
    pNodeA *ptu, *ptv;

    ptu = pt->dir;
    ptv = ptu->esq;
    ptu->esq = ptv->dir;
    ptv->dir = ptu;
    pt->dir = ptv->esq;
    ptv->esq = pt;
    if (ptv->FB == -1) pt->FB = 1;
    else pt->FB = 0;
    if (ptv->FB == 1) ptu->FB = -1;
    else ptu->FB = 0;
    pt = ptv;
    return pt;
}
```

Estruturas de Dados - Árvores

Exercícios

- Inserir em AVL, refazendo a árvore quando tiver rotação e anotando as rotações realizadas:
-50, 40, 30, 45, 47, 55, 56, 1, 2, 3
- Na inserção de quais elementos será necessário fazer rotação?

Estruturas de Dados - Árvores

Inserção de nodos em árvores AVL Alguns Problemas

- Percorre-se a árvore verificando se a chave já existe ou não
 - Em caso positivo, encerra a tentativa de inserção
 - Caso contrário, a busca encontra o local correto de inserção do novo nó
- Verifica-se se a inclusão tornará a árvore desbalanceada
 - Em caso negativo, o processo termina
 - Caso contrário, deve-se efetuar o balanceamento da árvore
- Descobre-se qual a operação de rotação a ser executada
- Executa-se a rotação

Estruturas de Dados - Árvores

Inserção de nodos em árvores AVL

- Como saber se a árvore está balanceada?
 - Verificando se existe um nó "desregulado"
- Como saber se um nó está desregulado?
 - Determina-se as alturas de suas sub-árvores e subtrai-se uma da outra
- Procedimento muito lento!
- Como ser mais eficiente?
 - Para cada nó v de uma árvore, armazena-se uma variável $\text{balanço}(v)$, onde

$$\text{balanço}(v) = \text{altura}(v.f.esq) - \text{altura}(v.f.dir)$$

Estruturas de Dados - Árvores

Inserção de nodos em árvores AVL Alguns problemas...

- Que valores são possíveis para balanço ?
 - 1, 0, 1
- De novo, como ser eficiente no cálculo do balanço ?
 - Dado q , como o nodo inserido.
 - Se q pertencer à sub-árvore esquerda de v e essa inclusão resultar em aumento na altura da sub-árvore, então

$$\text{balanço}(v) := \text{balanço}(v) + 1$$
 - Se $\text{balanço}(v) = 2$, então v está desregulado
 - Se q pertencer à sub-árvore direita de v e essa inclusão resultar em aumento na altura da sub-árvore, então

$$\text{balanço}(v) := \text{balanço}(v) - 1$$
 - Se $\text{balanço}(v) = -2$, então v está desregulado

Estruturas de Dados - Árvores

Inserção de nodos em árvores AVL

Alguns problemas...

- Mas, quando é que a inclusão de **q** causa aumento na altura da sub-árvore **v**?
- Suponha que **q** seja incluído na sub-árvore à esquerda de **v**.
- Para **q** incluído na sub-árvore à direita, considera-se o caso simétrico.

Estruturas de Dados - Árvores

Inserção de nodos em árvores AVL

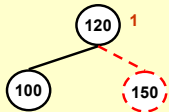
INSERÇÃO A DIREITA

- Se, antes da inclusão:
 - Balanco(v) = 1**, então **Balanco(v)** se tornará 0
 - altura da árvore não foi alterada
 - Por consequência, altura dos outros nós no caminho até a raiz, não se altera também.
 - Balanco(v) = 0**, então **Balanco(v)** se tornará -1
 - altura da árvore foi modificada
 - Por consequência, altura dos outros nós no caminho até a raiz, pode ter sido alterada também.
 - Repetir o processo (recursivamente), com **v** substituído por seu pai.
 - Balanco(v) = -1**, então **Balanco(v)** se tornará -2
 - altura da árvore foi modificada e o nó está desregulado
 - Rotação correta deve ser empregada.
 - Como a árvore será redesenhada, não é necessário verificar os outros nós.

Estruturas de Dados - Árvores

Inserção de nodos em árvores AVL

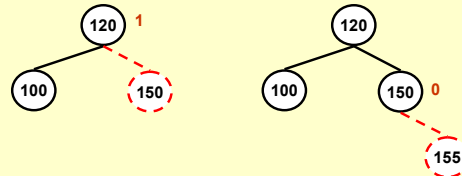
INSERÇÃO A DIREITA



Estruturas de Dados - Árvores

Inserção de nodos em árvores AVL

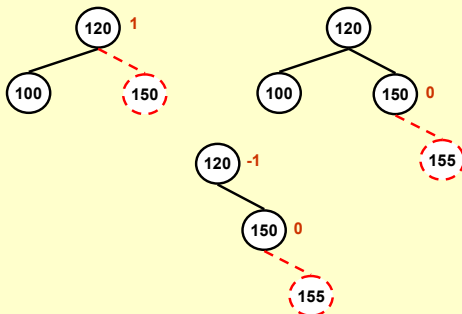
INSERÇÃO A DIREITA



Estruturas de Dados - Árvores

Inserção de nodos em árvores AVL

INSERÇÃO A DIREITA



Estruturas de Dados - Árvores

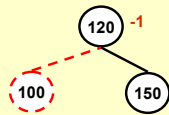
Inserção de nodos em árvores AVL

INSERÇÃO A ESQUERDA

- Se, antes da inclusão:
 - Balanco(v) = -1**, então **Balanco(v)** se tornará 0
 - altura da árvore não foi alterada
 - Por consequência, altura dos outros nós no caminho até a raiz, não se altera também.
 - Balanco(v) = 0**, então **Balanco(v)** se tornará 1
 - altura da árvore foi modificada
 - Por consequência, altura dos outros nós no caminho até a raiz, pode ter sido alterada também.
 - Repetir o processo (recursivamente), com **v** substituído por seu pai.
 - Balanco(v) = 1**, então **Balanco(v)** se tornará 2
 - altura da árvore foi modificada e o nó está desregulado
 - Rotação correta deve ser empregada.
 - Como a árvore será redesenhada, não é necessário verificar os outros nós.

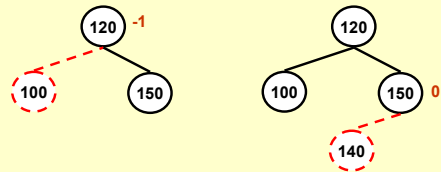
Estruturas de Dados - Árvores

Inserção de nodos em árvores AVL INSERÇÃO A ESQUERDA



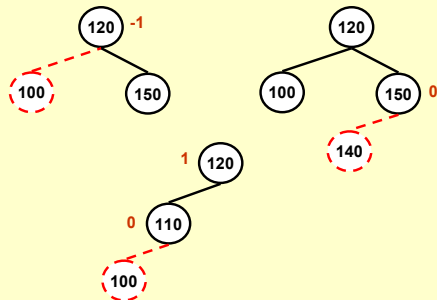
Estruturas de Dados - Árvores

Inserção de nodos em árvores AVL INSERÇÃO A ESQUERDA



Estruturas de Dados - Árvores

Inserção de nodos em árvores AVL INSERÇÃO A ESQUERDA



Estruturas de Dados - Árvores

Inserção de nodos em árvores AVL

```
pNodoA* InsereAVL (pNodoA *a, TipInfo x, int *ok){
/* Insere nodo em uma árvore AVL, onde A representa a raiz da árvore,
x, a chave a ser inserida e h a altura da árvore */
if (a == NULL) {
    a = (pNodoA*) malloc(sizeof(pNodoA));
    a->info = x;
    a->esq = NULL;
    a->dir = NULL;
    a->FB = 0;
    *ok = 1;
}
else if (x < a->info) {
    a->esq = InsereAVL(a->esq,x,ok);
    if (ok) {
        switch (a->FB) {
            case -1: a->FB = 0; *ok = 0; break;
            case 0: a->FB = 1; break;
            case 1: a = Caso1(a,ok); break;
        }
    }
}
}
```

```
struct TNodoA{
    TipInfo info;
    int FB;
    struct TNodoA *esq;
    struct TNodoA *dir;
};
typedef struct TNodoA pNodoA;
```

```
else {
    a->dir = InsereAVL(a->dir,x,ok);
    if (ok) {
        switch (a->FB) {
            case 1: a->FB = 0; ok = 0; break;
            case 0: a->FB = -1; break;
            case -1: a = Caso2(a,ok);
        }
    }
    return a;
}
```

Estruturas de Dados - Árvores

Inserção de nodos em árvores AVL

```
struct TNodoA{
    TipInfo info;
    int FB;
    struct TNodoA *esq;
    struct TNodoA *dir;
};
typedef struct TNodoA pNodoA;
```

```
pNodoA* Caso1 (pNodoA *a , int *ok)
{
    pNodoA *ptu;

    ptu = a->esq;
    if (ptu->FB == 1) {
        printf("fazendo rotacao direita em %d\n",a->info);
        a = rotacao_direita(a);
    }
    else {
        printf("fazendo rotacao dupla direita em %d\n",a->info);
        a = rotacao_dupla_direita(a);
    }
    a->FB = 0;
    *ok = 0;
    return a;
}
```

Estruturas de Dados - Árvores

Inserção de nodos em árvores AVL

```
struct TNodoA{
    TipInfo info;
    int FB;
    struct TNodoA *esq;
    struct TNodoA *dir;
};
typedef struct TNodoA pNodoA;
```

```
pNodoA* Caso2 (pNodoA *a , int *ok)
{
    pNodoA *ptu;

    ptu = a->dir;
    if (ptu->FB == -1) {
        printf("fazendo rotacao esquerda em %d\n",a->info);
        a = rotacao_esquerda(a);
    }
    else {
        printf("fazendo rotacao dupla esquerda em %d\n",a->info);
        a = rotacao_dupla_esquerda(a);
    }
    a->FB = 0;
    *ok = 0;
    return a;
}
```

Estruturas de Dados - Árvores

Remoção de nodos em árvores AVL

- Caso parecido com as inclusões.
- No entanto, nem sempre se consegue solucionar com uma única rotação...

Estruturas de Dados - Árvores

- <http://webpages.ull.es/users/jriera/Docencia/AVL/AVL%20tree%20applet.htm>

Estruturas de Dados - Árvores