EMA971915 - Estruturas de Dados I
Árvores Balanceadas

Carlos A. Astudillo
carlos.astudillo@unesp.br

Universidade Estadual Paulista (Unesp)

1º semestre/2023

Slides baseados em matérial dos professores Hélio Pedrini, Rafael C. S. Schouery e Lehilton Pedrosa (IC/UNICAMP).

Eficiência da busca, inserção e remoção

Qual é o tempo da busca, inserção e remoção em ABBs?

Eficiência da busca, inserção e remoção

Qual é o tempo da busca, inserção e remoção em ABBs?

- depende da altura da árvore…

Eficiência da busca, inserção e remoção

Qual é o tempo da busca, inserção e remoção em ABBs?

- depende da altura da árvore...

Ex: 31 nós

Eficiência da busca, inserção e remoção

Qual é o tempo da busca, inserção e remoção em ABBs?

- depende da altura da árvore...

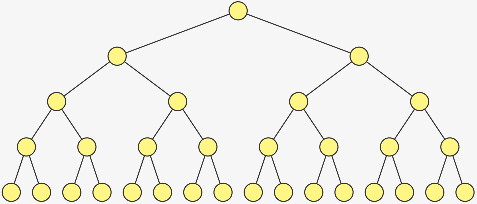
Ex: 31 nós

Eficiência da busca, inserção e remoção

Qual é o tempo da busca, inserção e remoção em ABBs?

- depende da altura da árvore...

Ex: 31 nós



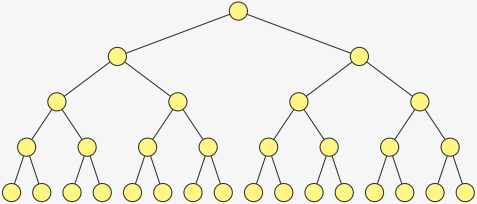
Melhor árvore: $\lceil \lg n + 1 \rceil$

Eficiência da busca, inserção e remoção

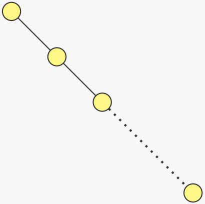
Qual é o tempo da busca, inserção e remoção em ABBs?

- depende da altura da árvore...

Ex: 31 nós



Melhor árvore: $\lceil \lg n + 1 \rceil$



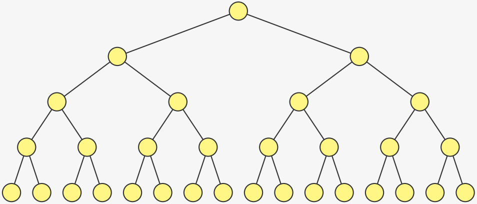
Pior árvore: n

Eficiência da busca, inserção e remoção

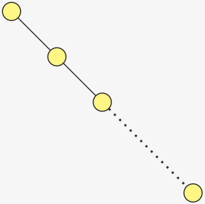
Qual é o tempo da busca, inserção e remoção em ABBs?

- depende da altura da árvore...

Ex: 31 nós



Melhor árvore: $\lceil \lg n + 1 \rceil$



Pior árvore: n

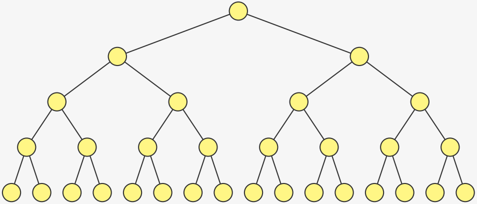
Para ter a pior árvore basta inserir em ordem crescente...

Eficiência da busca, inserção e remoção

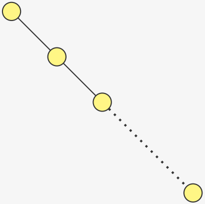
Qual é o tempo da busca, inserção e remoção em ABBs?

- depende da altura da árvore...

Ex: 31 nós



Melhor árvore: $\lceil \lg n + 1 \rceil$



Pior árvore: n

Para ter a pior árvore basta inserir em ordem crescente...

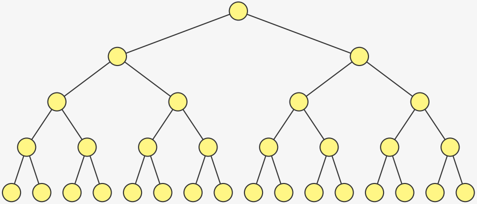
Veremos uma árvore balanceada

Eficiência da busca, inserção e remoção

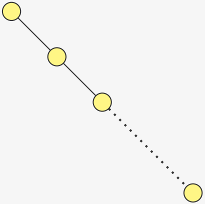
Qual é o tempo da busca, inserção e remoção em ABBs?

- depende da altura da árvore...

Ex: 31 nós



Melhor árvore: $\lceil \lg n + 1 \rceil$



Pior árvore: n

Para ter a pior árvore basta inserir em ordem crescente...

Veremos uma árvore balanceada

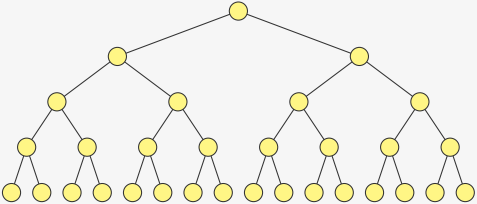
- Não é a melhor árvore possível, mas é “quase”

Eficiência da busca, inserção e remoção

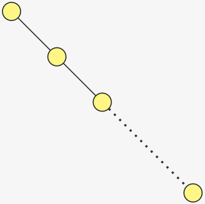
Qual é o tempo da busca, inserção e remoção em ABBs?

- depende da altura da árvore...

Ex: 31 nós



Melhor árvore: $\lceil \lg n + 1 \rceil$



Pior árvore: n

Para ter a pior árvore basta inserir em ordem crescente...

Veremos uma árvore balanceada

- Não é a melhor árvore possível, mas é "quase"
- Operações em $\mathrm{O}(\lg n)$

ABBs balanceadas

Existem também outras ABBs balanceadas:

ABBs balanceadas

Existem também outras ABBs balanceadas:

- Uma árvore balanceada é uma árvore com altura $\mathrm{O}(\lg n)$

ABBs balanceadas

Existem também outras ABBs balanceadas:

- Uma árvore balanceada é uma árvore com altura $\mathrm{O}(\lg n)$

Árvores AVL:

ABBs balanceadas

Existem também outras ABBs balanceadas:

- Uma árvore balanceada é uma árvore com altura $\mathrm{O}(\lg n)$

Árvores AVL:

- A altura das subárvores pode variar de no máximo 1

ABBs balanceadas

Existem também outras ABBs balanceadas:

- Uma árvore balanceada é uma árvore com altura $\mathrm{O}(\lg n)$

Árvores AVL:

- A altura das subárvores pode variar de no máximo 1
- Tem altura $\mathrm{O}(\lg n)$

ABBs balanceadas

Existem também outras ABBs balanceadas:

- Uma árvore balanceada é uma árvore com altura $\mathrm{O}(\lg n)$

Árvores AVL:

- A altura das subárvores pode variar de no máximo 1
- Tem altura $\mathrm{O}(\lg n)$

ABB aleatorizada:

ABBs balanceadas

Existem também outras ABBs balanceadas:

- Uma árvore balanceada é uma árvore com altura $\mathrm{O}(\lg n)$

Árvores AVL:

- A altura das subárvores pode variar de no máximo 1
- Tem altura $\mathrm{O}(\lg n)$

ABB aleatorizada:

- Decide de maneira aleatória como inserir o nó

ABBs balanceadas

Existem também outras ABBs balanceadas:

- Uma árvore balanceada é uma árvore com altura $\mathrm{O}(\lg n)$

Árvores AVL:

- A altura das subárvores pode variar de no máximo 1
- Tem altura $\mathrm{O}(\lg n)$

ABB aleatorizada:

- Decide de maneira aleatória como inserir o nó
  - inserção normal como folha

ABBs balanceadas

Existem também outras ABBs balanceadas:

- Uma árvore balanceada é uma árvore com altura $\mathrm{O}(\lg n)$

Árvores AVL:

- A altura das subárvores pode variar de no máximo 1
- Tem altura $\mathrm{O}(\lg n)$

ABB aleatorizada:

- Decide de maneira aleatória como inserir o nó
  - inserção normal como folha
  - inserção na raiz — rotações trazem o nó até a raiz

ABBs balanceadas

Existem também outras ABBs balanceadas:

- Uma árvore balanceada é uma árvore com altura $\mathrm{O}(\lg n)$

Árvores AVL:

- A altura das subárvores pode variar de no máximo 1
- Tem altura $\mathrm{O}(\lg n)$

ABB aleatorizada:

- Decide de maneira aleatória como inserir o nó
  - inserção normal como folha
  - inserção na raiz — rotações trazem o nó até a raiz
- Altura "média" (esperada): $\mathrm{O}(\lg n)$

ABBs balanceadas

Existem também outras ABBs balanceadas:

- Uma árvore balanceada é uma árvore com altura $\mathrm{O}(\lg n)$

Árvores AVL:

- A altura das subárvores pode variar de no máximo 1
- Tem altura $\mathrm{O}(\lg n)$

ABB aleatorizada:

- Decide de maneira aleatória como inserir o nó
  - inserção normal como folha
  - inserção na raiz — rotações trazem o nó até a raiz
- Altura “média” (esperada): $\mathrm{O}(\lg n)$

Árvores Splay:

ABBs balanceadas

Existem também outras ABBs balanceadas:

- Uma árvore balanceada é uma árvore com altura $\mathrm{O}(\lg n)$

Árvores AVL:

- A altura das subárvores pode variar de no máximo 1
- Tem altura $\mathrm{O}(\lg n)$

ABB aleatorizada:

- Decide de maneira aleatória como inserir o nó
  - inserção normal como folha
  - inserção na raiz — rotações trazem o nó até a raiz
- Altura “média” (esperada): $\mathrm{O}(\lg n)$

Árvores Splay:

- Sobe os nós no caminho da busca/inserção

ABBs balanceadas

Existem também outras ABBs balanceadas:

- Uma árvore balanceada é uma árvore com altura $\mathrm{O}(\lg n)$

Árvores AVL:

- A altura das subárvores pode variar de no máximo 1
- Tem altura $\mathrm{O}(\lg n)$

ABB aleatorizada:

- Decide de maneira aleatória como inserir o nó
  - inserção normal como folha
  - inserção na raiz — rotações trazem o nó até a raiz
- Altura “média” (esperada): $\mathrm{O}(\lg n)$

Árvores Splay:

- Sobe os nós no caminho da busca/inserção
- Nós mais acessados ficam mais próximos da raiz

ABBs balanceadas

Existem também outras ABBs balanceadas:

- Uma árvore balanceada é uma árvore com altura $\mathrm{O}(\lg n)$

Árvores AVL:

- A altura das subárvores pode variar de no máximo 1
- Tem altura $\mathrm{O}(\lg n)$

ABB aleatorizada:

- Decide de maneira aleatória como inserir o nó
  - inserção normal como folha
  - inserção na raiz — rotações trazem o nó até a raiz
- Altura "média" (esperada): $\mathrm{O}(\lg n)$

Árvores Splay:

- Sobe os nós no caminho da busca/inserção
- Nós mais acessados ficam mais próximos da raiz
- Não é balanceada, mas o custo de m inserções/buscas em uma árvore Splay com n nós é $\mathrm{O}((n+m)\lg(n+m))$

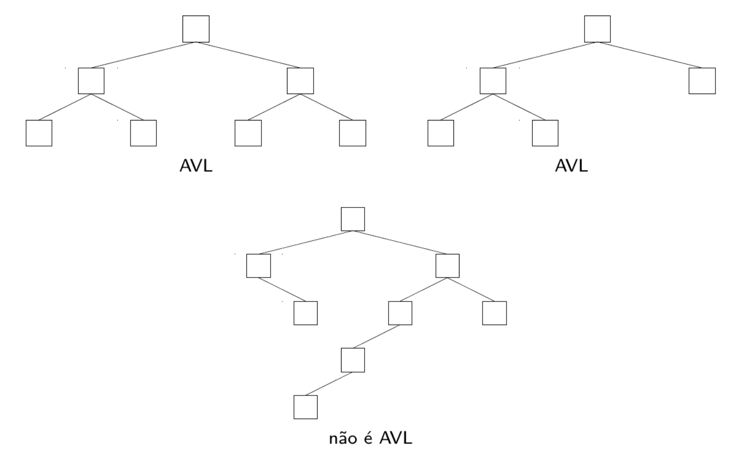
Árvores AVL

- a diferença das alturas entre as subárvores esquerda e direita, em cada nó, não deve ser superior a uma unidade.

Árvores AVL

- a diferença das alturas entre as subárvores esquerda e direita, em cada nó, não deve ser superior a uma unidade.

Exemplos:



Árvores Balanceadas

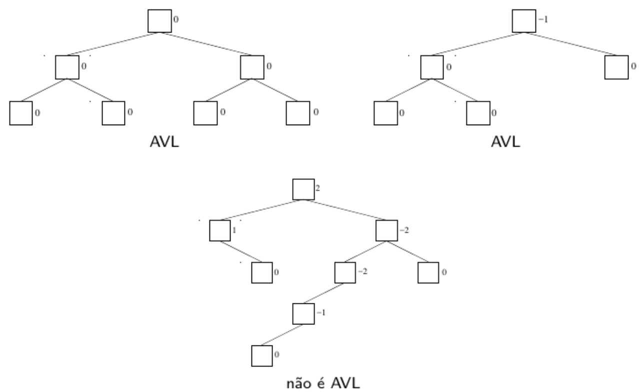
- Fator de balanceamento: o fator de balanceamento de um nó em uma árvore binária é definido como a diferença entre a altura de sua subárvore direita e a altura de sua subárvore esquerda.

Árvores Balanceadas

- Fator de balanceamento: o fator de balanceamento de um nó em uma árvore binária é definido como a diferença entre a altura de sua subárvore direita e a altura de sua subárvore esquerda.
- Dessa forma, cada nó em uma árvore balanceada AVL tem um fator de balanceamento -1, 0 ou 1.

Árvores Balanceadas

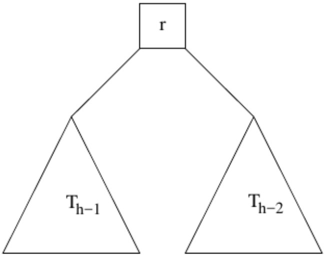
- Fator de balanceamento: o fator de balanceamento de um nó em uma árvore binária é definido como a diferença entre a altura de sua subárvore direita e a altura de sua subárvore esquerda.
- Dessa forma, cada nó em uma árvore balanceada AVL tem um fator de balanceamento -1, 0 ou 1.



Árvores Balanceadas

Seja T_h uma árvore AVL com altura h e número mínimo de nós. Para formar T_h, consideram-se inicialmente os casos triviais:

- se $h = 0$, T_h é uma árvore vazia.
- se $h = 1$, T_h consiste em um único nó.
- quando $h > 1$, escolhe-se um nó r como raiz. Em seguida, escolhe-se T_{h-1} para formar a subárvore esquerda de r e escolhe-se T_{h-2} para formar a subárvore direita de r. Note que é possível intercambiar as subárvores esquerda e direita de r sem alterar o balanceamento dos nós da árvore.

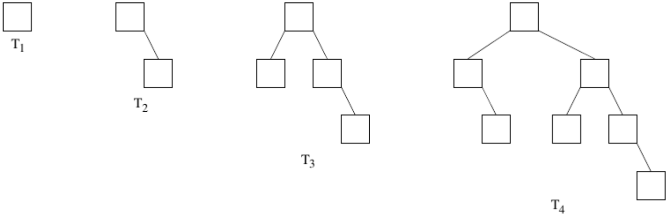


Árvores Balanceadas

- Vamos agora calcular o número mínimo de nós. Para facilitar, observa-se que basta calcular um limite inferior do valor procurado em termos de h.
- Seja $|T_h|$ o número de nós de T_h. Então

$$\begin{cases} |T_h| = 1, & \text{para } h = 1 \\ |T_h| = 2, & \text{para } h = 2 \\ |T_h| = 1 + |T_{h-1}| + |T_{h-2}|, & \text{para } h > 2 \end{cases}$$

Exemplos:



- Mostre exemplos de árvores T_5 e T_6.
- Devido à analogia com a série de Fibonacci, T_h é denominada árvore de Fibonacci.

Árvores Balanceadas

Estrutura de nó:

Árvores Balanceadas

Estrutura de nó:

```
typedef struct No {
  int info;
  int altura;
  struct No *esq, *dir;
} No;
```

Árvores Balanceadas

Criar um nó de uma árvore AVL:

Árvores Balanceadas

Criar um nó de uma árvore AVL:

```c
No *criar_no_AVL(int x) {
  No *no = (No*) malloc(sizeof(No));
  if (no == NULL)
    return NULL;
  no->info = x;
  no->altura = 1;
  no->esq = NULL;
  no->dir = NULL;
  return no;
}
```

Árvores Balanceadas

Altura de uma árvore AVL:

Árvores Balanceadas

Altura de uma árvore AVL:

```
int altura_arvore_AVL(No *raiz) {
  if (raiz == NULL)
    return 0;
  return raiz->altura;
}
```

Árvores Balanceadas

Obter fator de balanceamento de uma árvore AVL :

Árvores Balanceadas

Obter fator de balanceamento de uma árvore AVL :

```
int fator_balanceamento(No *r) {
  if (r == NULL)
    return 0;
  return altura_arvore_AVL(r->dir) - altura_arvore_AVL(r->esq);
}
```

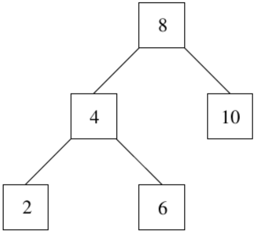
Árvores Balanceadas

Inserção de um nó em uma árvore AVL:

- A inserção é sempre feita como nó-folha da árvore.
- Duas operações são realizadas após a inserção:
  - atualizar a altura dos nós já existentes na árvore.
  - verificar a necessidade de rebalancear a árvore.
- Para a atualização da altura dos nós da árvore, basta, no pior caso, atualizar a altura dos nós que definem o caminho da raiz da árvore ao nó (folha) inserido.
- Quando o nó é inserido na subárvore de um nó de menor altura, não é necessário atualizar a altura de todos os nós daquele caminho (apenas os nós desta subárvore).

Árvores Balanceadas

Suponha a árvore balanceada AVL:



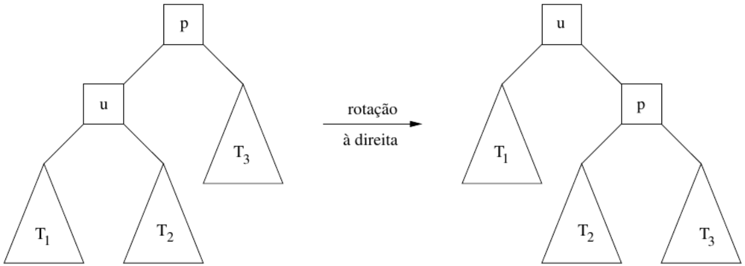
- Os nós 9 e 11 podem ser inseridos diretamente.
- Os nós 3, 5 ou 7 requerem que a árvore seja rebalanceada.
- As operações de rebalanceamento são baseadas em rotações de nós.

Árvores Balanceadas

- Após a inserção de um nó q em uma árvore AVL T, se a árvore não permanecer balanceada, seja o nó p mais próximo às folhas de T que se tornou desbalanceado.
- Sejam $h_E(p)$ e $h_D(p)$ as alturas das subárvores esquerda e direita de p, respectivamente.
- Quatro casos possíveis devem ser analisados.

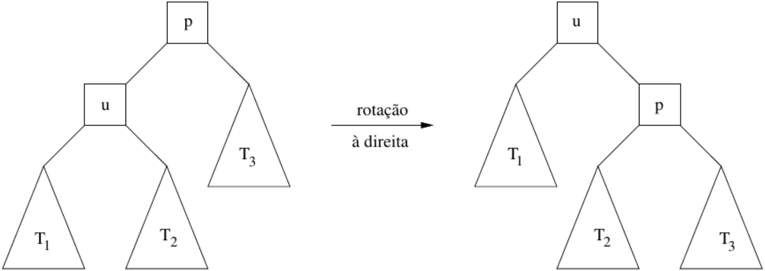
Árvores Balanceadas

- Caso 1: $h_E(p) > h_D(p)$



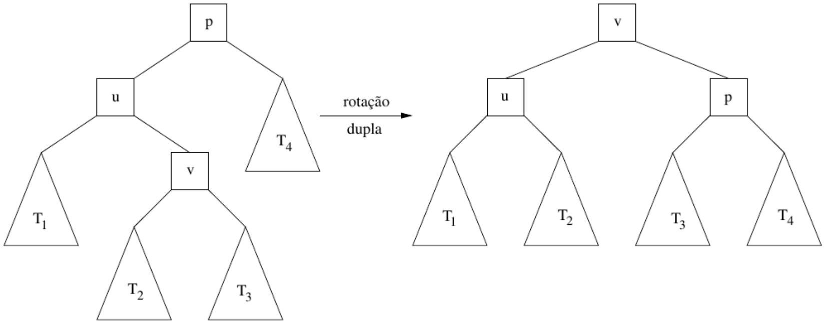
Árvores Balanceadas

- Caso 1: $h_E(p) > h_D(p)$
  - Caso 1.1: $h_E(u) > h_D(u)$



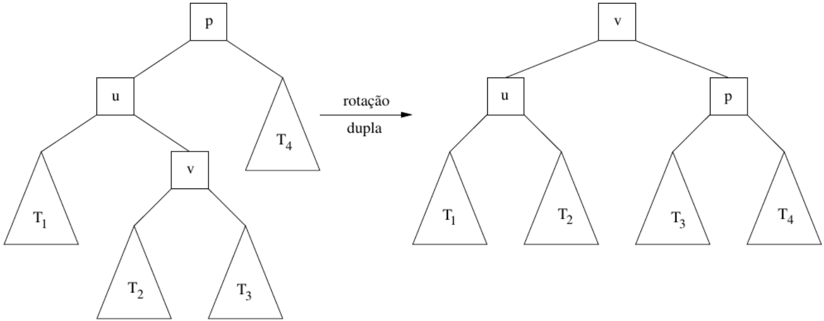
Árvores Balanceadas

- Caso 1: $h_E(p) > h_D(p)$



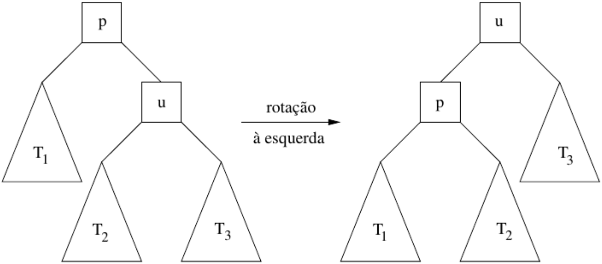
Árvores Balanceadas

- Caso 1: $h_E(p) > h_D(p)$
  - Caso 1.2: $h_E(u) < h_D(u)$



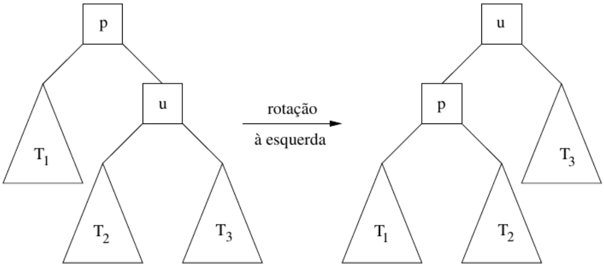
Árvores Balanceadas

- Caso 2: $h_E(p) < h_D(p)$



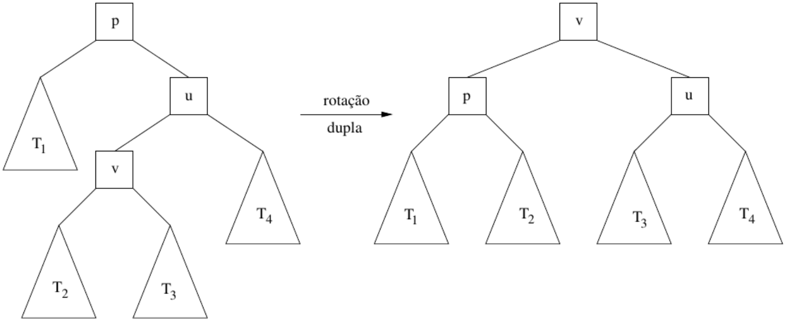
Árvores Balanceadas

- Caso 2: $h_E(p) < h_D(p)$
  - Caso 2.1: $h_E(u) < h_D(u)$



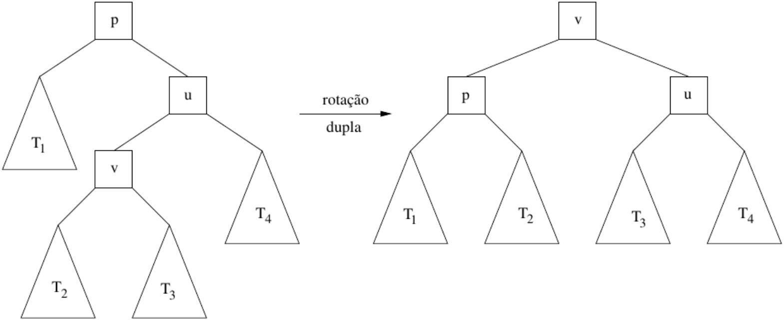
Árvores Balanceadas

- Caso 2: $h_E(p) < h_D(p)$



Árvores Balanceadas

- Caso 2: $h_E(p) < h_D(p)$
  - Caso 2.2: $h_E(u) > h_D(u)$



Árvores Balanceadas

Rotação à esquerda:

Árvores Balanceadas

Rotação à esquerda:

```
No *rotacao_esquerda(No *x) {
  No *y = x->dir;
  No *T2 = y->esq;

  y->esq = x;
  x->dir = T2;

  x->altura = max(altura(x->esq), altura(x->dir)) + 1;
  y->altura = max(altura(y->esq), altura(y->dir)) + 1;

  return y;
}
```

Árvores Balanceadas

Rotação à direita:

Árvores Balanceadas

Rotação à direita:

```
No *rotacao_direita(No *y) {
  No *x = y->esq;
  No *T2 = x->dir;

  x->dir = y;
  y->esq = T2;

  y->altura = max(altura(y->esq), altura(y->dir)) + 1;
  x->altura = max(altura(x->esq), altura(x->dir)) + 1;

  return x;
}
```

Árvores Balanceadas

Inserção:

```
No *inserir_AVL(No *r, int val) {
  int balanceamento;

  /* insercao normal em uma arvore binaria de busca */
  if (r == NULL)
    return criar_no_avl(val);

  if (val < r->info)
    r->esq  = inserir_AVL(r->esq, val);
  else if (val > r->info)
         r->dir = inserir_AVL(r->dir, val);
  else
     return r;

  /* atualizar altura do no ancestral */
  r->altura = 1 + max(altura(r->esq), altura(r->dir));

  /* obter fator de balanceamento do no ancestral */
  balanceamento = fator_balanceamento_avl(r);

  ...
```

Árvores Balanceadas

Inserção:

```
  ...

  /* 4 casos devem ser considerados quando um no se torna desbalanceado */

  /* caso 1.1: rotacao simples (direita) */
  if (balanceamento > 1 && val < r->esq->info)
    return rotacao_direita(r);

  /* caso 2.1: rotacao simples (esquerda) */
  if (balanceamento < -1 && val > r->dir->info)
    return rotacao_esquerda(r);

  /* caso 1.2: rotacao dupla (esquerda e direita) */
  if (balanceamento > 1 && val > r->esq->info) {
    r->esq = rotacao_esquerda(r->esq);
    return rotacao_direita(r);
  }

  /* caso 2.2: rotacao dupla (direita e esquerda) */
  if (balanceamento < -1 && val < r->dir->info) {
    r->dir = rotacao_direita(r->dir);
    return rotacao_esquerda(r);
  }

  return r;
}
```

Árvores Balanceadas

Remoção:

```
No *remover_AVL(No *r, int val) {
  No *temp;
  int balanceamento;

  /* remocao normal em uma arvore binaria de busca */
  if (r == NULL)
    return r;

  if (val < r->info)
    r->esq = remover_AVL(r->esq, val);
  else if (val > r->info)
          r->dir = remover_AVL(r->dir, val);
        else {
          /* no com apenas um ou nenhum filho */
          if ((r->esq == NULL) || (r->dir == NULL)) {
            temp = r->esq ? r->esq : r->dir;

            /* nenhum filho */
            if (temp == NULL) {
              temp = r;
              r = NULL;
            }
            else /* um filho */
              *r = *temp; /* copia conteudo do filho nao vazio */

            free(temp);
          }
          ...
```

Árvores Balanceadas

Remoção:

```
        ...

        else {
    /* no com dois filhos: obter sucessor em in-ordem /*
            temp = minimo(r->dir);

            /* copiar informacao do sucessor em in-ordem */
            r->info = temp->info;

            /* remover sucessor em in-ordem */
            r->dir = remover_AVL(r->dir, temp->info);
          }
        }

  /* arvore com apenas um no */
  if (r == NULL)
    return r;

  /* atualiza altura do no corrente */
  r->altura = 1 + max(altura(r->esq, altura(r->dir));

  /* fator de balanceamento do no */
  balanceamento = fator_balanceamento(r);

  ...
```

Árvores Balanceadas

Remoção:

```
  ...

  /* 4 casos devem ser considerados quando um no se torna desbalanceado */

  /* caso 1.1: rotacao simples (direita) */
  if (balanceamento > 1 && fator_balanceamento(r->esq) >= 0)
    return rotacao_direita(r);

  /* caso 1.2: rotacao dupla (esquerda e direita) */
  if (balanceamento > 1 && fator_balanceamento(r->esq) < 0) {
    r->esq = rotacao_esquerda(r->esq);
    return rotacao_direita(r);
  }

  /* caso 2.1: rotacao simples (esquerda) */
  if (balanceamento < -1 && fator_balanceamento(r->dir) <= 0)
    return rotacao_esquerda(r);

  /* caso 2.2: rotacao dupla (direita e esquerda) */
  if (balanceamento < -1 && fator_balanceamento(r->dir) > 0) {
    r->dir = rotacao_direita(r->dir);
    return rotacao_esquerda(r);
  }

  return r;
}
```

Árvores Balanceadas

Exercício:
A partir de uma árvore AVL inicialmente vazia, construa a árvore resultante fazendo sucessivas inserções dos seguintes nós, na ordem dada:
10, 20,15, 5, 1, 12, 25, 30
A cada inserção, verificar se algum nó ficou desregulado, e em caso afirmativo aplicar a rotação adequada.