CMP1054 - Estruturas de Dados I

Árvores Binárias de Pesquisa Balanceadas

Prof. José Olimpio Ferreira

Balanceamento de Árvores Binárias de Pesquisa

- Árvores Balancedas
  - Árvores ótimas
    - Cheia → Altura k > 0 → Nº de nós = $2^k - 1$
      - Altura da árvore → número de níveis (1, 2, 3, ..., k) da árvore
        - Níveis → 1, 2, 3, 4, 5, ...
        - Número de links do caminho mais longo da raiz (inclusive) até uma folha.
- Árvores binárias de pesquisa
  - Construídas aleatoriamente através de inserção são quase otimamente balanceadas
  - Gasta-se tempo da ordem de $O(\log(n))$
    - Inserção/Remoção/Busca
  - Problemas
    - Operações de inserção e remoção frequentemente não são aleatórias

Dr. José Olimpio Ferreira

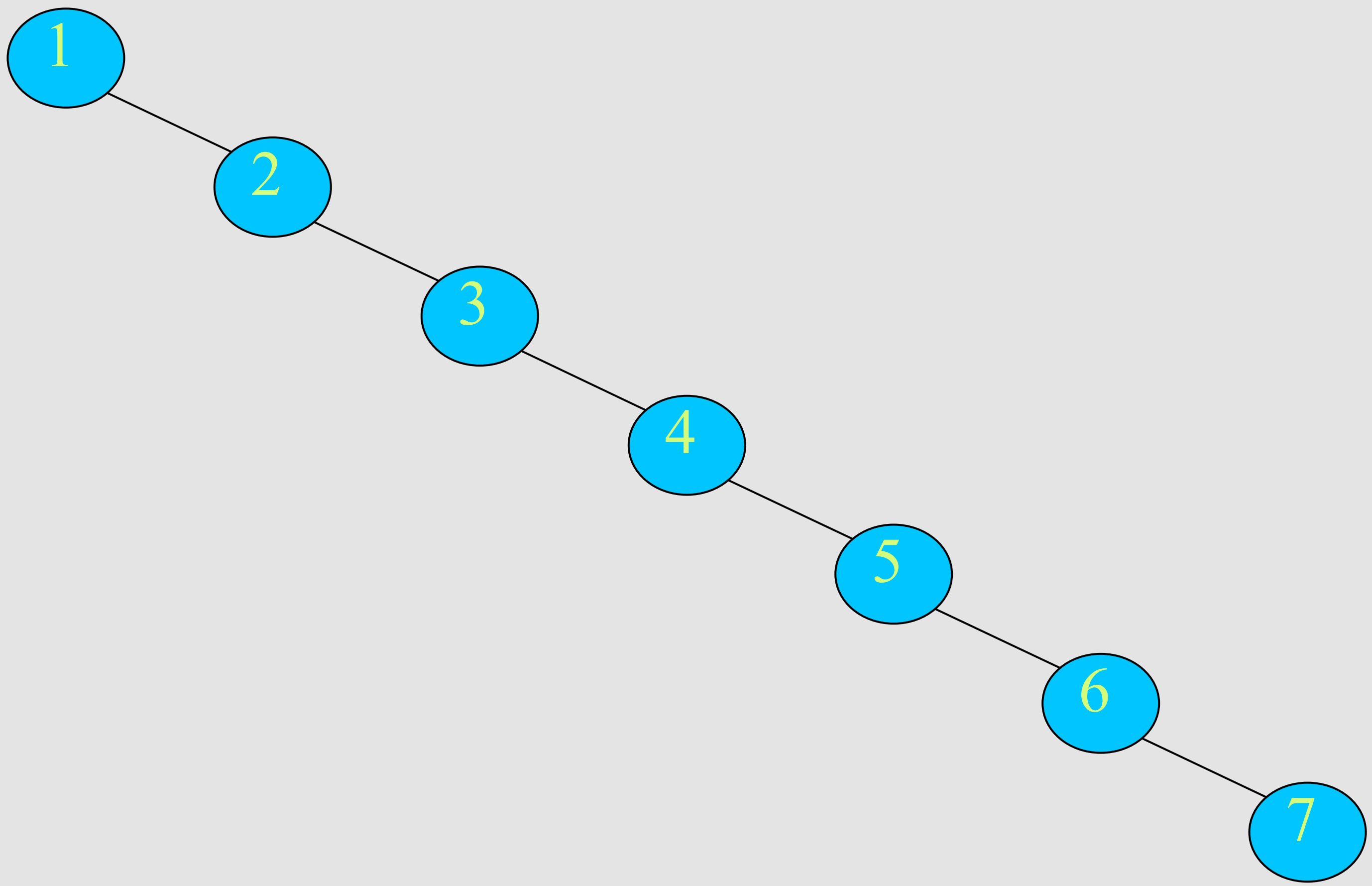
Balanceamento de Árvores Binárias de Pesquisa

- Pesquisas utilizando árvores binárias de busca aumentam o desempenho em relação a busca em listas encadeadas ou vetores.
- No entanto, à medida em que a árvore vai sendo modificada (através de inserções e remoções), ela pode ficar degenerada.
- Balanceamento de árvores visam maximizar o desempenho de buscas em árvores binárias

Dr. José Olimpio Ferreira

Exemplos de árvores binárias de busca

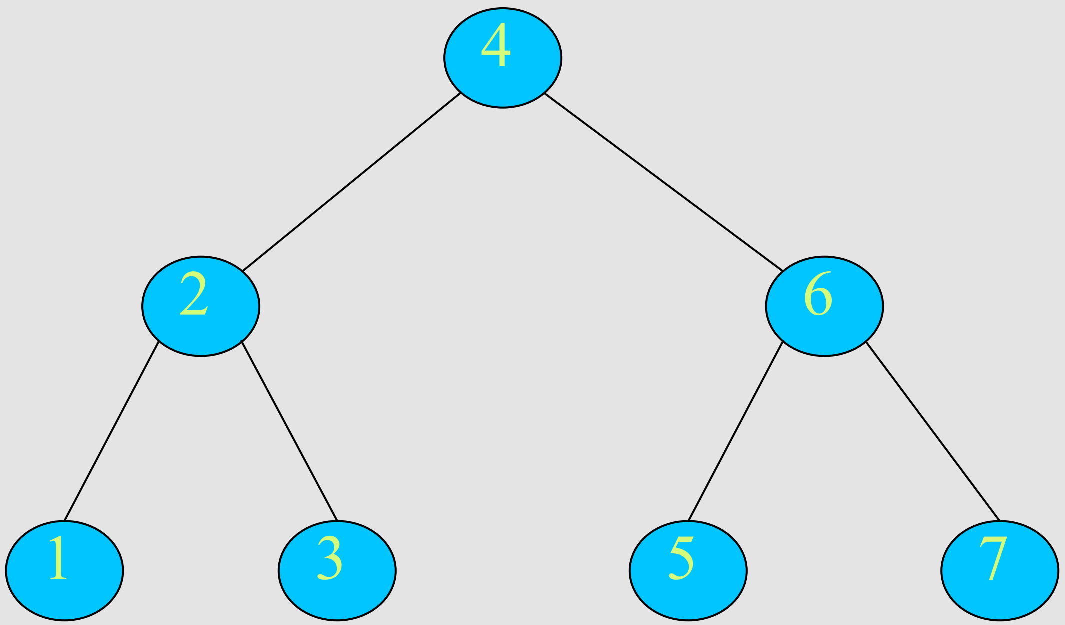
- A: Inserção de 1, 2, 3, 4, 5, 6 e 7
- Pior caso: O(n)



Dr. José Olimpio Ferreira

Exemplos de árvores binárias de busca

- B: Inserção de 4, 2, 6, 1, 3, 5 e 7, nesta ordem
- Pior caso: O(log n)



Dr. José Olimpio Ferreira

Balanceamento de Árvores Binárias de Pesquisa

Altura	Nós em um nível	Nós em todos os níveis
1	$2^0 = 1$	$1 = 2^1 - 1$
2	$2^1 = 2$	$2^2 - 1 = 3$
3	$2^2 = 4$	$2^3 - 1 = 7$
4	$2^3 = 8$	$2^4 - 1 = 15$
11	$2^{10} = 1024$	$2^{11} - 1 = 2047$
20	$2^{19} = 524288$	$2^{20} - 1 = 1048575$
h	$\mathbf{2^{h-1}}$	$\mathbf{2^h - 1 = n}$

- Se tivéssemos 1.048.575 elementos, poderíamos armazená-los em uma árvore binária perfeitamente balanceada com 20 níveis.
- Em outras palavras, poderíamos localizar um elemento qualquer dentro dessa árvore com no máximo 20 comparações.

Dr. José Olimpio Ferreira

Balanceamento de Árvores Binárias de Pesquisa

- Uma boa condição de balanceamento deve possuir as seguintes características:
- Assegurar que a altura da árvore com n nós seja O(log n)
- Deve poder ser mantida balanceada com um custo baixo.
- Para fazer o balanceamento de árvores são usadas as chamadas rotações

Dr. José Olimpio Ferreira

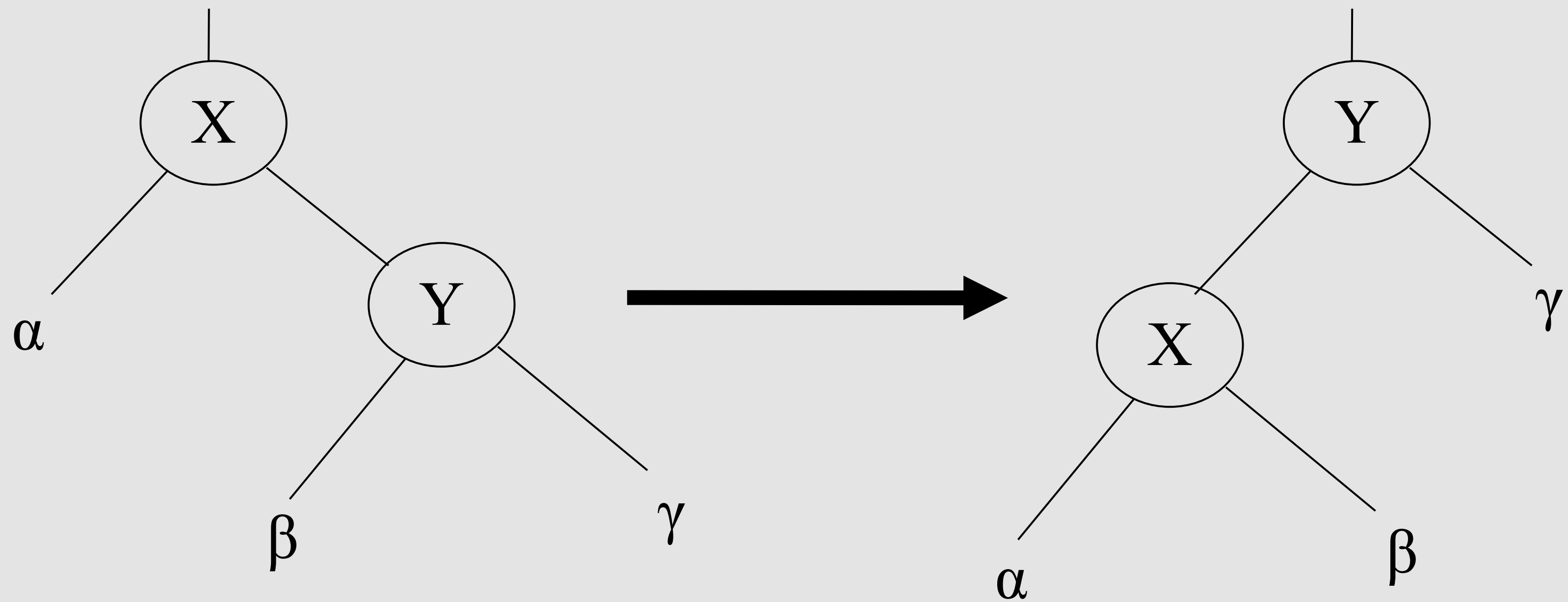
Técnicas de Balanceamento em Árvores Binárias de Pesquisa

Dr. José Olimpio Ferreira

- Técnicas de Balanceamento em Árvore Binária de Pesquisa
  - Árvores Vermelho-Preto (Red-Black)
  - Árvores AVL
    - Rotações
      - Rotação à esquerda e Rotação à direita
      - Operação local que muda a estrutura de ponteiros e preserva a propriedade da árvore de pesquisa binária

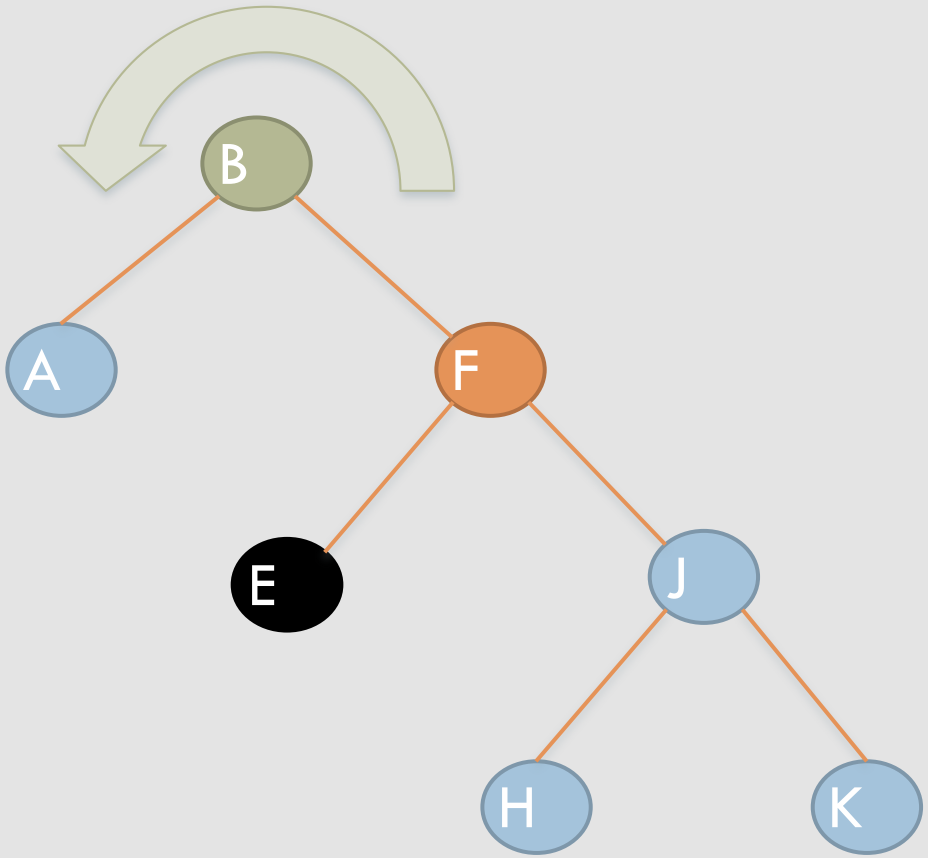
Rotação à esquerda

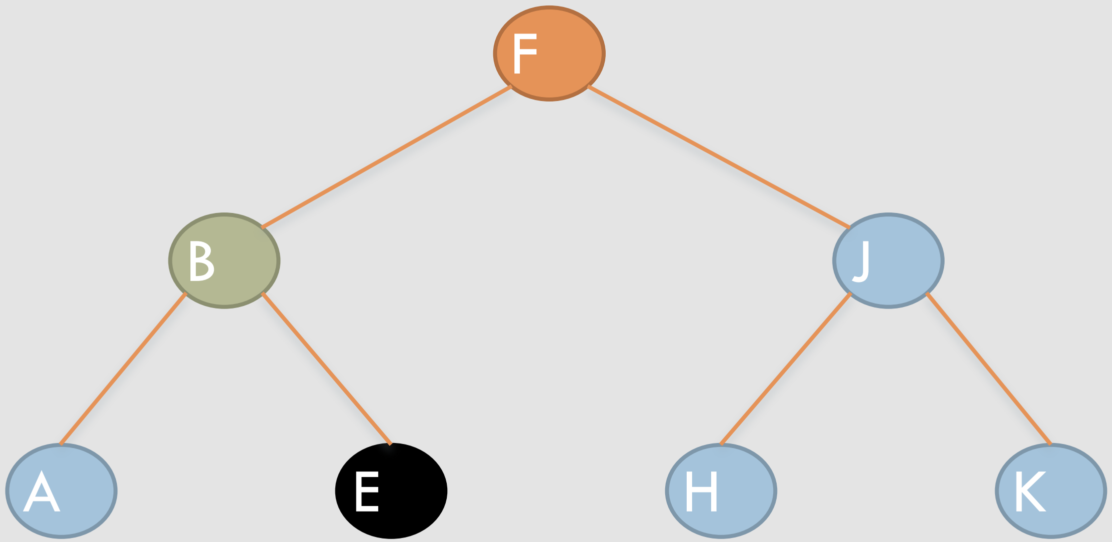
- Rotação a esquerda em um nó X (raiz da subárvore)
  - Supõe-se que o filho da direita, Y, não seja nulo
  - Faz o pivô em torno da ligação de X para Y
    - Faz de Y a nova raiz da sub-árvore
    - X torna-se o filho da esquerda de Y
    - O filho da esquerda de Y (o β) torna-se o filho da direita de X



Dr. José Olimpio Ferreira

Rotação à esquerda





Dr. José Olimpio Ferreira

Estruturas

- Estrutura auto referenciada

```
tipo
    abp = registro (   PAI  ^abp,
                       ESQ  ^abp,
                       DIR  ^abp,
                       nome literal,   {Este é o campo CHAVE}
                       endereco literal,
                       telefone literal
                     )
```

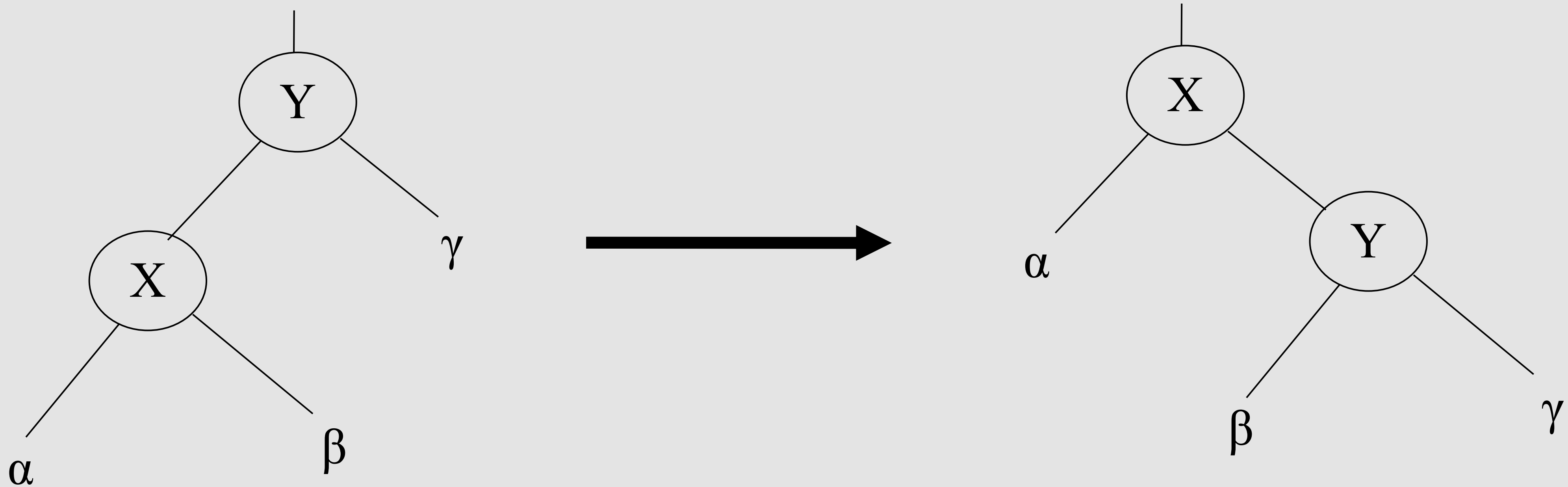
Dr. José Olimpio Ferreira

```
sub-rotinaROTAÇÃO_À_ESQUERDA ( R, X)
 I  declare R, X, Y ponteiro
 I  Y <-- X^.DIR       {Estabelece Y}
 I  X^.DIR <-- Y^.ESQ  {Coloca a sub-árvore à esquerda de Y (beta) como sub-árvore à direita de X}
 I  se Y^.ESQ ≠ nulo     {Estabelece o novo pai de beta}
 I  I  então
 I  I    Y^.ESQ^.PAI <-- X
 I  fim se
 I  Y^.PAI <-- X^.PAI  {Liga o pai de X em Y}
 I  se X^.PAI = nulo      {Estabelece o novo pai de Y}
 I  I  então
 I  I    R <-- Y
 I  I  senão
 I  I    se X = X^.PAI^.ESQ  {se X é filho a esquerda}
 I  I    I  então
 I  I    I    X^.PAI^.ESQ <-- Y {Y passa a ser filho do pai de X}
 I  I    I  senão
 I  I    I    X^.PAI^.DIR <-- Y  {Y passa a ser filho do pai de X}
 I  I    fim se
 I  fim se
 I  Y^.ESQ <-- X       {Y recebe X como filho à esquerda}
 I  X^.PAI <-- Y       {X recebe Y como pai}
 fim sub-rotina
```

Dr. José Olimpio Ferreira

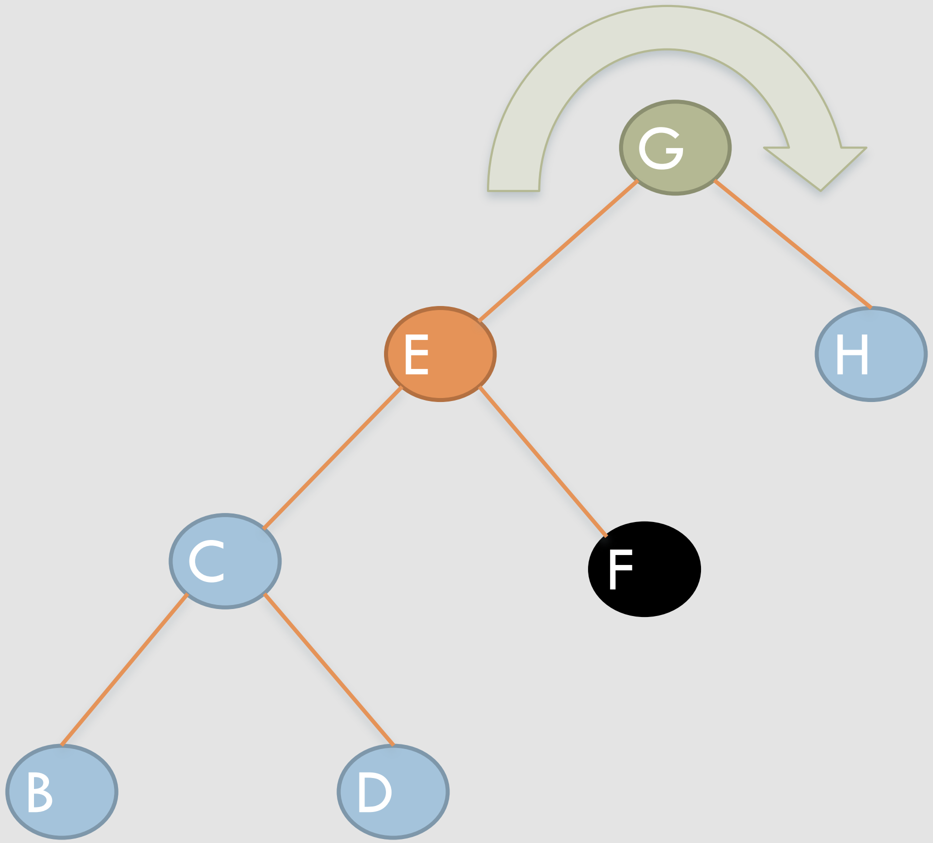
Rotação à direita

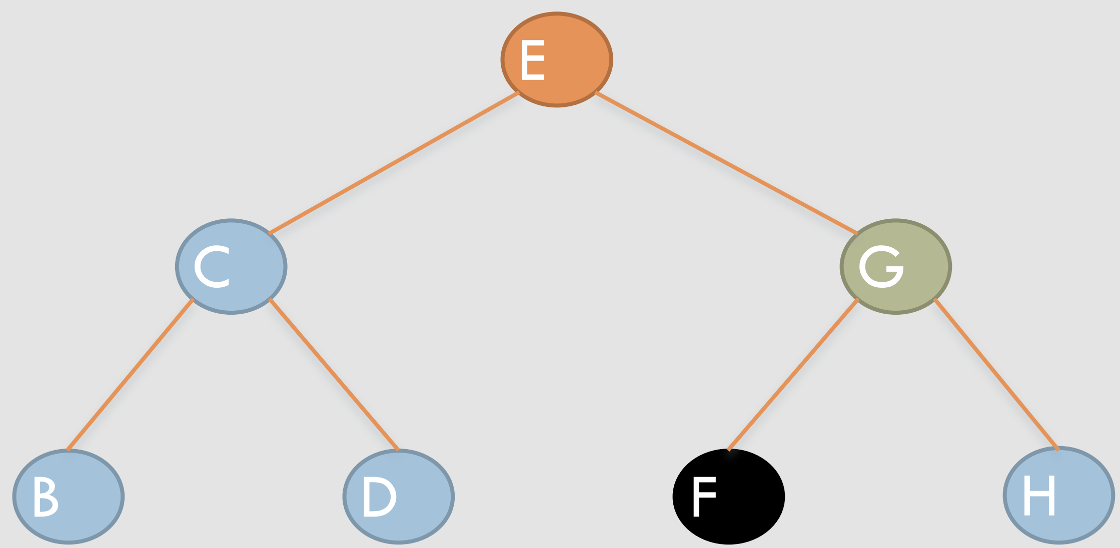
- Rotação a direita em um nó Y (raiz da subárvore)
  - Supõe-se que o filho da esquerda, X, não seja nulo
  - Faz o pivô em torno da ligação de Y para X
    - Faz de X a nova raiz da sub-árvore
    - Y torna-se o filho da direita de X
    - O filho da direita de X (β) torna-se o filho da esquerda de Y



Dr. José Olimpio Ferreira

Rotação à direita





Dr. José Olimpio Ferreira

Dr. José Olimpio Ferreira

```
                                           e/s e
sub-rotinaROTAÇÃO_À_DIREITA ( R, Y)
I  declare R, X, Y ponteiro
I  X <-- Y^.ESQ       {Estabelece X}
I  Y^.ESQ <-- X^.DIR  {Coloca a sub-árvore à direita de X (beta) como sub-árvore à esquerda de Y}
I  se Y^.ESQ ≠ nil     {Estabelece o novo pai de beta}
I  I  então
I  I    Y^.ESQ^.PAI <-- Y
I  fim se
I  X^.PAI <-- Y^.PAI  {Liga o pai de Y em X}
I  se Y^.PAI = nil     {Estabelece o novo pai de X}
I  I  então
I  I    R <-- X
I  I  senão
I  I    se Y = Y^.PAI^.ESQ   {se Y é filho a esquerda}
I  I    I  então
I  I    I    Y^.PAI^.ESQ <-- X  {X passa a ser filho do pai de Y}
I  I    I  senão
I  I    I    Y^.PAI^.DIR <-- X   {X passa a ser filho do pai de Y}
I  I    fim se
I  fim se
I  X^.DIR <-- Y       {X recebe Y como filho à direita}
I  Y^.PAI <-- X       {Y recebe X como pai}
fim sub-rotina
```