Algoritmos e Estruturas de Dados I

Árvores AVL

Mirtha Lina Fernández Venero
Sala 529-2, Bloco A
mirtha.lina@ufabc.edu.br
http://professor.ufabc.edu.br/~mirtha.lina/aedi.html

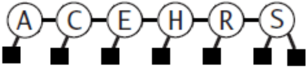
7 de abril de 2018

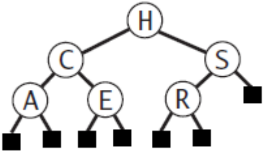
Agenda

Aulas anteriores - Estratégias de Busca - Caso Pior

Técnica	Ordem	Busca	Inserção	Remoção
Busca Sequencial	Não	N	N	N
Busca Binária	Sim	log(N)	N	N
ABB	Sim	h	h	h
???	Sim	log(N)	log(N)	log(N)

- As operações nas ABBs têm custo entre $O(\log N)$ e $O(N)$
- O caso melhor acontece e.g. quando elas são completas
- Por que não "completar" a árvore após inserir/remover?

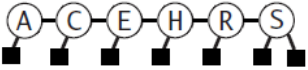


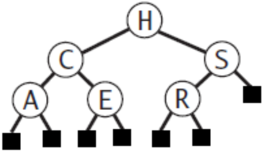


Aulas anteriores - Estratégias de Busca - Caso Pior

Técnica	Ordem	Busca	Inserção	Remoção
Busca Sequencial	Não	N	N	N
Busca Binária	Sim	log(N)	N	N
ABB	Sim	h	h	h
???	Sim	log(N)	log(N)	log(N)

- As operações nas ABBs têm custo entre $O(\log N)$ e $O(N)$
- O caso melhor acontece e.g. quando elas são completas
- Por que não "completar" a árvore após inserir/remover? Custo elevado $O(N)$

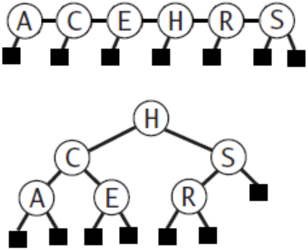




Aulas anteriores - Estratégias de Busca - Caso Pior

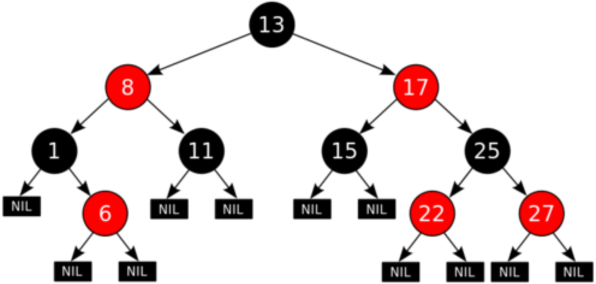
Técnica	Ordem	Busca	Inserção	Remoção
Busca Sequencial	Não	N	N	N
Busca Binária	Sim	log(N)	N	N
ABB	Sim	h	h	h
???	Sim	log(N)	log(N)	log(N)

- Na verdade uma ABB não precisa ser completa para ter custo $O(\log N)$; basta serem **balanceadas**
- Que tipo de **balanceamento**?
- Como balancear de forma eficiente e ao mesmo tempo preservar a ordem simétrica após inserir/remover?



Tipos de Árvores Balanceadas

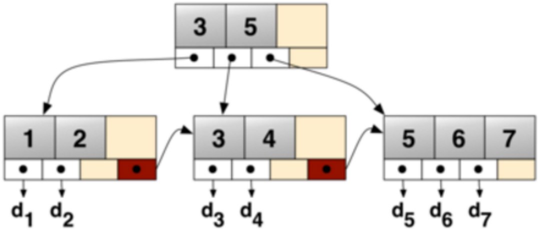
- Balanceamento pela altura (*height balance*): AVL trees, red-black trees



https://en.wikipedia.org/wiki/Red-black_tree

Tipos de Árvores Balanceadas

- Balanceamento pela altura (*height balance*): AVL trees, red-black trees
- Balanceamento perfeito pela altura (*perfect height balance*): 2-3 trees, 2-3-4 trees, B trees (B+, B*)



https://en.wikipedia.org/wiki/B+tree

Tipos de Árvores Balanceadas

- Balanceamento pela altura (*height balance*): AVL trees, red-black trees
- Balanceamento perfeito pela altura (*perfect height balance*): 2-3 trees, 2-3-4 trees, B trees (B+, B*)
- Balanceamento pela "classe"(*rank balance*): WAVL trees [1]

[1] Rank-balanced trees: Haeupler B., Sen, S., Tarjan, R. E., ACM Transactions on Algorithms, 2015

Universidade Federal do ABC

Tipos de Árvores Balanceadas

- Balanceamento pela altura (*height balance*): AVL trees, red-black trees
- Balanceamento perfeito pela altura (*perfect height balance*): 2-3 trees, 2-3-4 trees, B trees (B+, B*)
- Balanceamento pela "classe"(*rank balance*): WAVL trees [1]
- Balanceamento pela frequência de acesso: splay trees [2]

[2] Self-Adjusting Binary Search Trees: Sleator, Daniel D., Tarjan, Robert E. Journal of the ACM, 1985

Tipos de Árvores Balanceadas

- Balanceamento pela altura (*height balance*): AVL trees, red-black trees
- Balanceamento perfeito pela altura (*perfect height balance*): 2-3 trees, 2-3-4 trees, B trees (B+, B*)
- Balanceamento pela "classe"(*rank balance*): WAVL trees [1]
- Balanceamento pela frequência de acesso: splay trees [2]
- Balanceamento pelo número de nós: weight-balanced binary trees [3]

[3] Binary Search Trees of Bounded Balance: Nievergelt, J., Reingold, E. M. SIAM Journal on Computing, 1973

Universidade Federal do ABC

Agenda

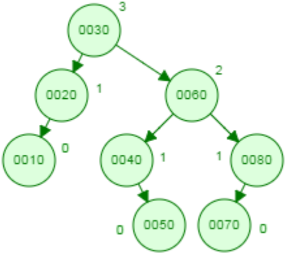
Universidade Federal do ABC

Agenda

Árvores Binárias Balanceadas pela Altura

Altura de um nó: número de passos do mais longo caminho até uma folha

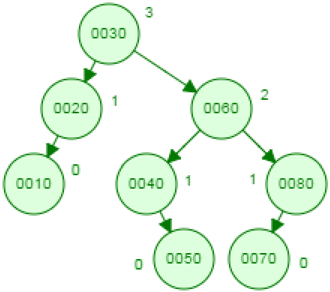
$$h(n) = \begin{cases} -1 & \text{se } n\text{=NULL} \\ max(h(n \rightarrow esq), h(n \rightarrow dir)) + 1 & \text{se } n\neq\text{NULL} \end{cases}$$

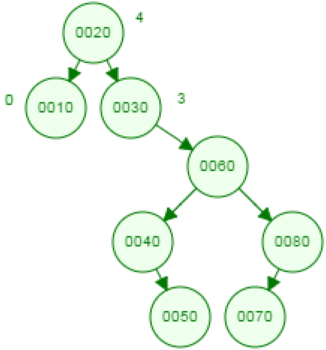


Árvores Binárias **AVL**

Primeiras árvores balanceadas, propostas em 1962 pelos cientistas russos **Georgy Adelson-Velsky** e **Evgenii Landis** [4]

- para cada nó na árvore, a diferença de altura de suas duas subárvores é no máximo 1

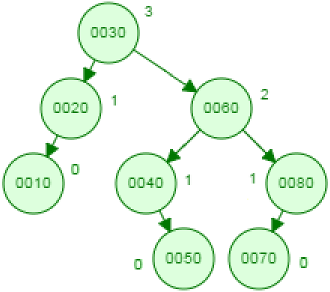


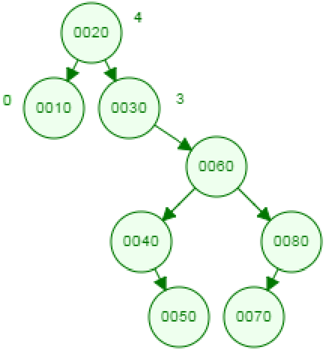


[4] An algorithm for the organization of information. Proc. USSR Academy of Sciences, 146: 263-266, 1962

Árvore AVL e Fator de Balanceamento de um nó n

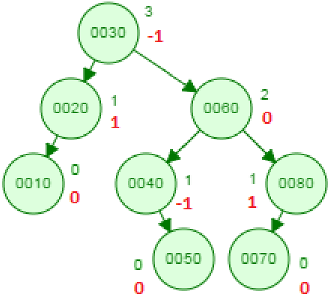
$$FB(n) = h(n \rightarrow esq) - h(n \rightarrow dir)$$

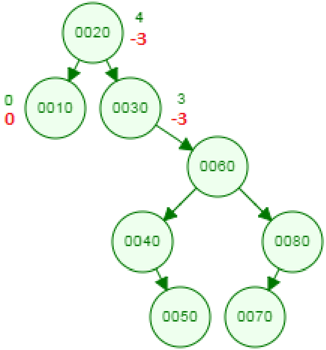




Árvore AVL e Fator de Balanceamento de um nó n

$$FB(n) = h(n \to esq) - h(n \to dir)$$

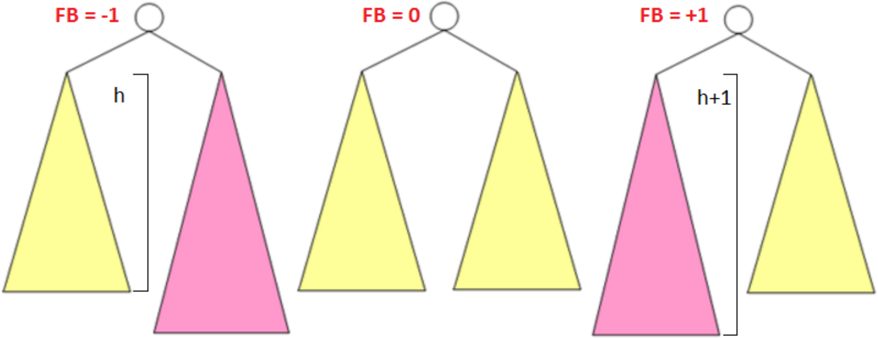




Árvore AVL e Fator de Balanceamento de um nó n

$$FB(n) = h(n \to esq) - h(n \to dir)$$

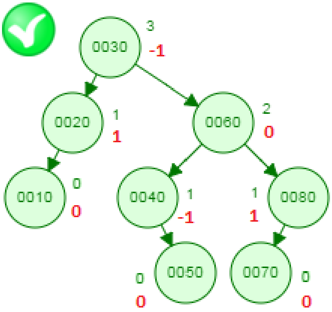
- Uma ABB é **AVL** se para cada nó n, $|\ FB(n)\ | \leq 1$

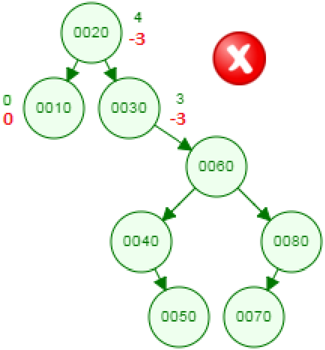


Árvore AVL e Fator de Balanceamento de um nó n

$$FB(n) = h(n \rightarrow esq) - h(n \rightarrow dir)$$

- Uma ABB é **AVL** se para cada nó n, $|\ FB(n)\ | \leq 1$



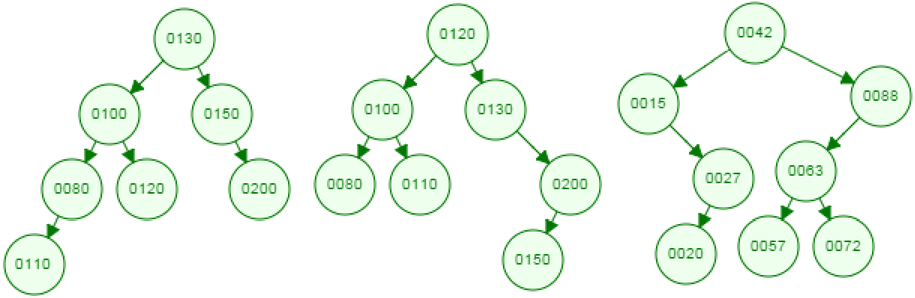


Árvore AVL e Fator de Balanceamento de um nó **n**

$$FB(n) = h(n \rightarrow esq) - h(n \rightarrow dir)$$

- Uma ABB é **AVL** se para cada nó n, $|\ FB(n)\ | \leq 1$

Exercício: Quais das ABBs abaixo são AVL?



Exemplo de Implementação - Árvore AVL

Pode ser implementada armazenando a altura ou o fator de balanceamento

```
typedef struct avlTreeNode avlTreeNode;
struct avlTreeNode {
  int key;
  int height; // unsigned short balanceFactor;

  // pointers to the left and right children of the node
  avlTreeNode *left, *right;
};

void updateHeight(avlTreeNode *n) {
  if (!n) return;
  int hl = (n->left)  ? n->left->height  : -1,
      hr = (n->right) ? n->right->height : -1;
  n->height =  (hl > hr ? hl : hr) + 1 ;
}
```

Exemplo de Implementação - Árvore AVL

Pode ser implementada armazenando a altura ou o fator de balanceamento

```
const int leftheavy = -1, balanced = 0, rightheavy = 1;

int balanceFactor(avlTreeNode *n) {
  if (!n)
    return 0;
  int hl = (n->left)  ? n->left->height  : -1,
      hr = (n->right) ? n->right->height : -1;
  return hl - hr;
}
```

Agenda

Universidade Federal do ABC

Análise das Árvores AVL

Qual a altura máxima h duma árvore AVL com n nós?

Prova: Usar pergunta equivalente: fixando h, qual é a menor árvore AVL (# nós) que pode ser construída com altura h? Seja $N(h)$ o menor número de nós de uma árvore AVL de altura h.

- $N(0) = 1$ e $N(1) = 2$
- Se $h > 1$ as sub-árvores esquerda e direita terão no máximo altura $h - 1$. Na verdade, para fazer com que a árvore tenha o menor número de nós possível (sem violar a condição de AVL) então uma sub-árvore terão altura $h - 1$ e a outra altura $h - 2$. Isto leva à recorrência

$$N(h) = N(h-1) + N(h-2) + 1 \qquad se \;\; h > 1$$

Análise das Árvores AVL - Menor árvore AVL de altura h

$$N(0) = 1; \;\; N(1) = 2$$

$$N(h) = N(h-1) + N(h-2) + 1 \;\; se \; h > 1$$

```
                                         ___ *____
                                        |         |
     *    *      _*_      __*__        _*_      __*__
          |     |   |    |     |      |   |    |     |
          *     *   *    *    _*_     *   *    *    _*_
                    |    |   |   |        |    |   |   |
                    *    *   *   *        *    *   *   *
                                 |                     |
                                 *                     *
```

Universidade Federal do ABC

Análise das Árvores AVL - Menor árvore AVL de altura h

$$N(0) = 1; \;\; N(1) = 2$$

$$N(h) = N(h-1) + N(h-2) + 1 \;\; se \; h > 1$$

Para $h - 1$ temos $N(h-1) = N(h-2) + N(h-3) + 1$, logo

$$N(h) = (\ N(h-2) + N(h-3) + 1\ ) + N(h-2) + 1 \qquad \Rightarrow$$

$$N(h) > 2 * N(h-2) \qquad \Rightarrow$$

$$N(h) > 2 * N(h-2) > 2 * 2 * N(h-4) > \ldots > 2^{h/2}$$

Análise das Árvores AVL - Menor árvore AVL de altura h

$$N(h) > 2^{h/2} \Rightarrow \log_2 N(h) > log_2 2^{h/2} \Rightarrow$$

$$h < 2 * \log_2 N(h) \quad \square$$

Desta forma, para qualquer outra árvore con n nós e altura h

$$n \geq N(h) > 2^{h/2} \Rightarrow \log_2 n > log_2 2^{h/2} \Rightarrow h < 2 * \log_2 n$$

Qual a altura máxima h duma árvore AVL com n nós?

Resposta: $h = O(\log_2 n)$

Análise das Árvores AVL - Menor árvore AVL de altura h

Resolvendo melhor a recorrência (notar que é parecida com a recorrência dos números de Fibonacci)

$$N(0) = 1; \;\; N(1) = 2$$

$$N(h) = N(h-1) + N(h-2) + 1 \;\; se \; h > 1$$

é obtido que

$$N(h) = \varphi^h, \varphi = \frac{1+\sqrt{(5)}}{2} \approx 1.618 \; (the \; golden \; ratio)$$

Calculando $\log_2(\varphi) = 1.44$; portanto, a altura de uma árvore AVL é $\approx 1.44 * \log_2 n$ onde n é o número de nós da árvore.

Universidade Federal do ABC

Agenda

Universidade Federal do ABC

Árvore AVL e Fator de Balanceamento de um nó **n**

- Como manter as árvores AVL balanceadas após uma inserção ou remoção?
- Como preservar a ordem simétrica das ABS?
- Como manter o custo logarítmico das operações?

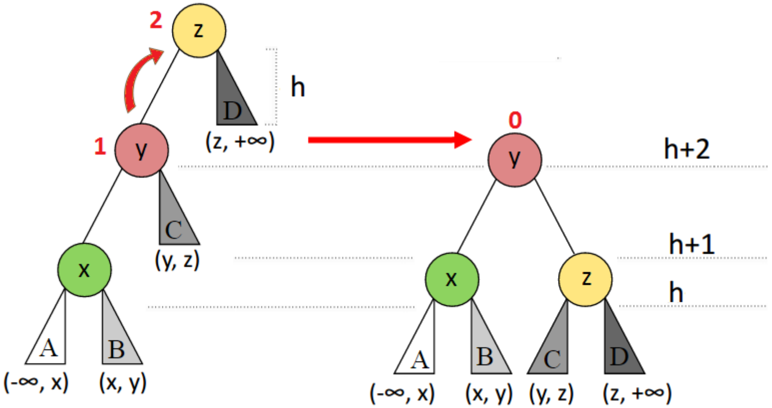
Resposta: usar transformações locais (de baixo custo - $O(1)$) que somente sejam efetuadas no caminho da operação

Rotações: Permitem intercambiar o papel da raiz (nó com FB igual a -2 ou 2) e um dos filhos, preservando a ordem das chaves

- **Simples**: Esquerda ou Direita
- **Dupla**: (Direita-) Esquerda ou (Esquerda-) Direita

Rotação Simples à Direita - Caso LL

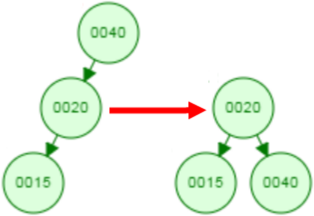
- O desbalanceamento está à esquerda-esquerda
- Troca o papel da raiz e o filho esquerdo preservando a ordem

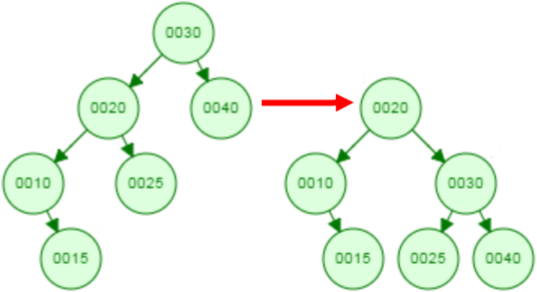


Universidade Federal do ABC

Exemplos Rotação Simples à Direita - Caso LL

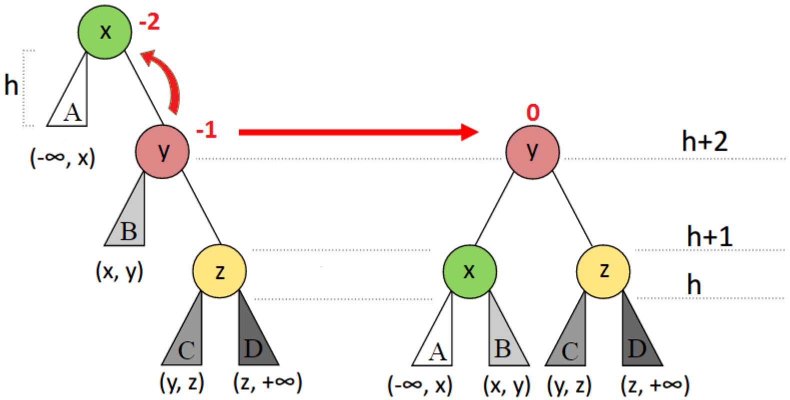
- O desbalanceamento está à esquerda-esquerda: o nó desbalanceado tem FB=2 e o filho esquerdo FB=1 (note que é o mesmo sinal do pai!)
- Rotaciona a raiz e o filho esquerdo em sentido horário





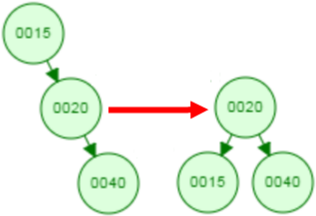
Rotação Simples à Esquerda - Caso RR

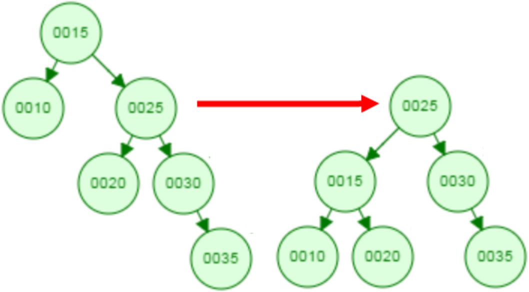
- O desbalanceamento está à direita-direita
- Troca o papel da raiz e o filho direito preservando a ordem



Exemplos Rotação Simples à Esquerda - Caso RR

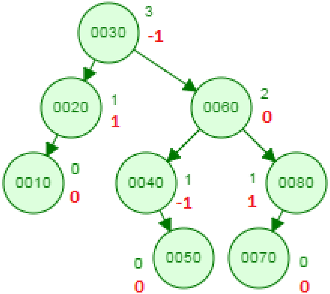
- O desbalanceamento está à direita-direita: o nó desbalanceado tem FB=-2 e o filho direito FB=-1 (note-se que é o mesmo sinal do pai!)
- Rotaciona a raiz e o filho direito em sentido anti-horário

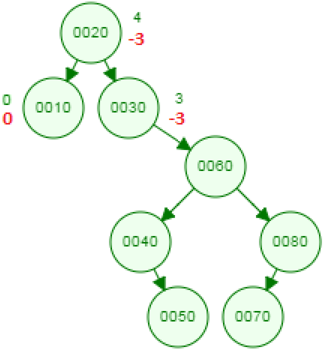




Exercício de Rotação Simples

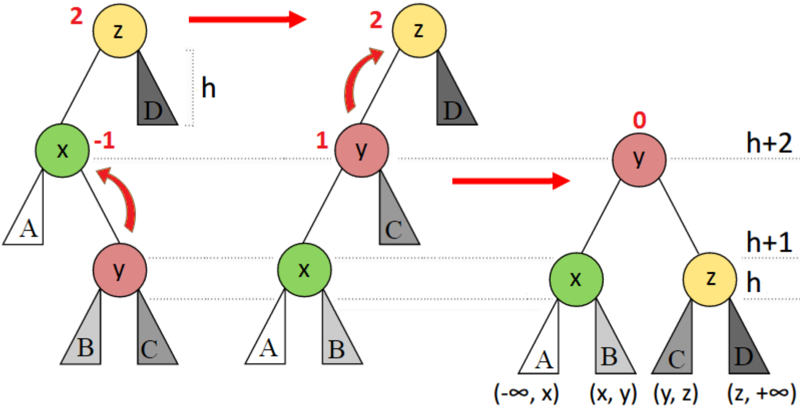
Desenhe as árvores resultantes de aplicar a rotação simples à esquerda no nó **0030** nas árvores abaixo.





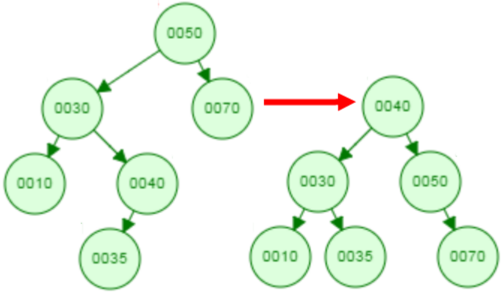
Rotação Dupla à Direita - Caso LR

- O desbalanceamento está à esquerda-direita
- Duas rotações simples: 1ra à esquerda; 2da à direita



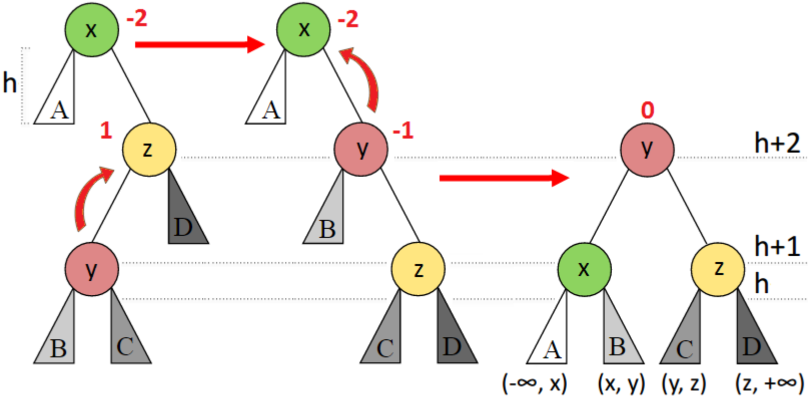
Exemplos Rotação Dupla à Direita - Caso LR

- O desbalanceamento está à esquerda-direita: o nó desbalanceado tem FB=2 e o filho esquerdo FB=-1 (sinal inverso ao do pai!)
- O neto que está à esquerda-direita é "movimentado" duas vezes: primeiro à esquerda e depois à direita



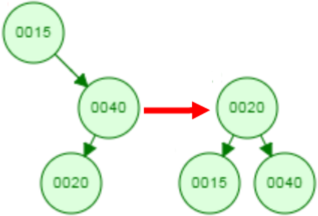
Rotação Dupla à Esquerda - Caso RL

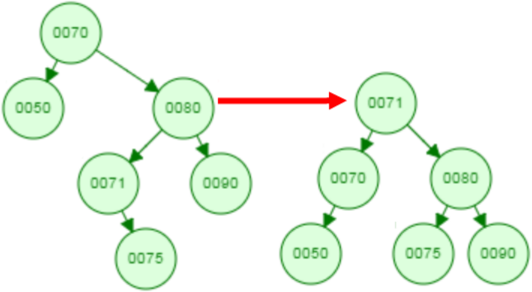
- O desbalanceamento está à direita-esquerda
- Duas rotações simples: 1ra à direita; 2da à esquerda



Exemplos Rotação Dupla à Esquerda - Caso RL

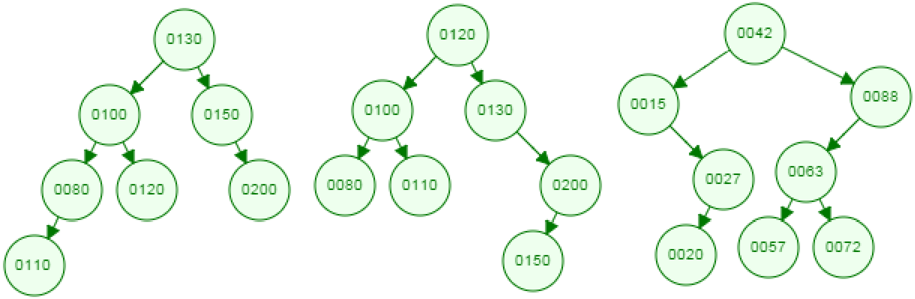
- O desbalanceamento está à direita-esquerda: o nó desbalanceado tem FB=-2 e o filho direito FB=1 (sinal inverso ao do pai!)
- O neto que está à direita-esquerda é "movimentado" duas vezes: primeiro à direita e depois à esquerda





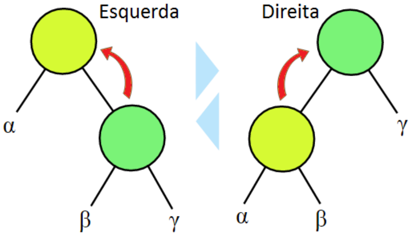
Exercício: Rotações

Balanceie as ABBs não AVL abaixo usando rotações apropriadas



Resumo Rotações

- São simétricas



- Nas simples, o nó não balanceado tem FB com o mesmo sinal do filho mais alto ("simétrico" à rotação). São aplicadas no sentido inverso à maior altura
- Nas duplas, o nó não balanceado tem FB o sinal inverso do filho mais alto ("simétrico" à rotação). São compostas por uma rotação simples e a simétrica.

Universidade Federal do ABC

Agenda

Inserção nas Árvores AVL

Inserir o nó da mesma forma que nas ABBs. No caminho de volta até a raíz, atualizar a altura de cada nó **n**, checar se não cumpre a condição AVL e rotacionar de forma apropriada

Caso 1 A nova chave foi inserida na subárvore de menor altura $\Rightarrow$ a altura **não** muda e a árvore fica balanceada

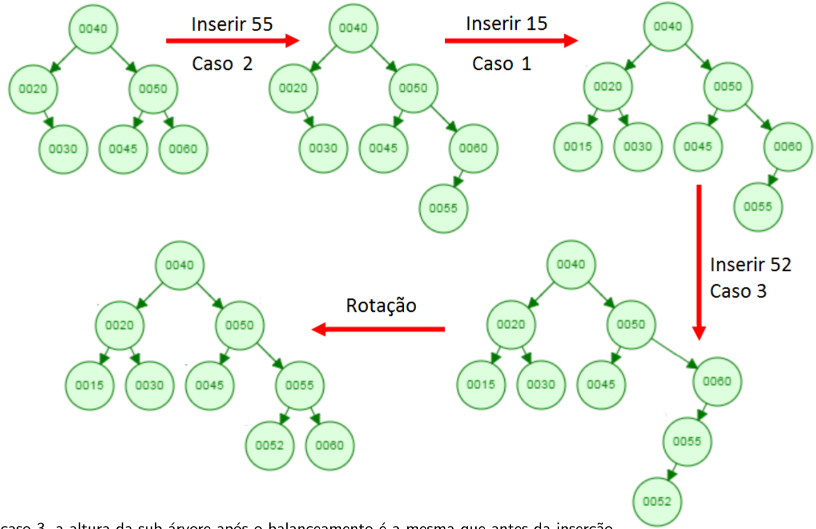
Caso 2 O nó tinha FB $= 0$ antes da inserção $\Rightarrow$ atualizar a altura. É preciso conferir a condição AVL dos antecessores

Caso 3 A nova chave foi inserida na subárvore de maior altura (direita ou esquerda resp) $\Rightarrow$ efetuar a rotação. Não é preciso conferir o balanceamento dos antecessores

Exemplo: Inserir as seguintes chaves numa árvore AVL:
40, 20, 50, 30, 45, 60, 55, 15, 52

Ver animação em AVL Tree Visualization

Exemplo de Inserções nas Árvores AVL



No caso 3, a altura da sub-árvore após o balanceamento é a mesma que antes da inserção

Inserção nas Árvores AVL

Inserir o nó da mesma forma que nas ABBs. No caminho de volta até a raíz, atualizar a altura de cada nó **n**, checar se não cumpre a condição AVL e rotacionar de forma apropriada

Caso 1 A nova chave foi inserida na subárvore de menor altura $\Rightarrow$ a altura **não** muda e a árvore fica balanceada

Caso 2 O nó tinha FB $= 0$ antes da inserção $\Rightarrow$ atualizar a altura. É preciso conferir a condição AVL dos antecessores

Caso 3 A nova chave foi inserida na subárvore de maior altura (direita ou esquerda resp) $\Rightarrow$ efetuar a rotação. Não é preciso conferir o balanceamento dos antecessores

Exercício: Inserir as seguintes chaves numa árvore AVL (usando a ordem lexicográfica): maio, março, novembro, agosto, abril, janeiro, dezembro, fevereiro, julho, junho, outubro, setembro

Exemplo de Implementação - Inserção AV

```
void avlInsert(avlTreeNode **treeRoot,  int key) {
  if (*treeRoot == NULL) {

    // update the root to point at a new Node
    avlTreeNode *newNode = malloc(sizeof(avlTreeNode));
    *treeRoot = newNode;

    if (!newNode)
        return;

    newNode->key = key;
    newNode->left = newNode->right = NULL;
    newNode->height = 0;
    return;
  }

  if (key == (*treeRoot)->key)
    return;
```

Ver código aqui.

Exemplo de Implementação - Inserção AV

```
avlTreeNode *tree = *treeRoot;

if (key < tree->key) { // recursively move to the left
  avlInsert(&tree->left, key);

  // check if the tree must be updated
  if (balanceFactor(tree) == 2) {
    // inserted in the left from node was already heavy on the left

    if (key < tree->left->key)
      caseLLrotateRight(&tree);
    else caseLRrotateLeftRight(&tree);
  }
  else updateHeight(tree);
}
```

Ver código aqui.

Exemplo de Implementação - Inserção AV

```
else { // otherwise recursively move right
  avlInsert(&tree->right, key);

  if (balanceFactor(tree) == -2) {

    if (key > tree->right->key)
      caseRRrotateLeft(&tree);
    else caseRLrotateRightLeft(&tree);
  }
  else updateHeight(tree);
}

*treeRoot = tree;
}
```

Ver código aqui.

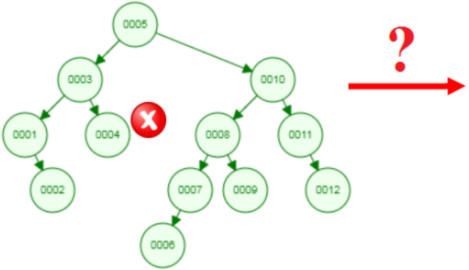
Universidade Federal do ABC

Agenda

Remoção nas Árvores AVL

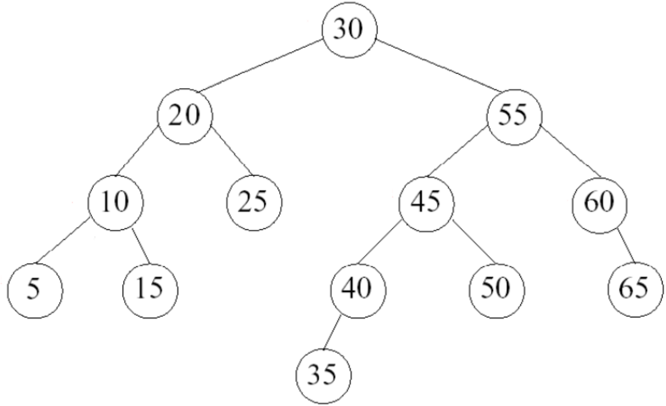
Remover o nó da mesma forma que nas ABBs. No caminho de volta desde o pai da **folha removida** até a raiz, atualizar a altura de cada nó **n**, checar se não cumpre a condição AVL e rotacionar

- O fator de balanceamento pode mudar
- A árvore pode diminuir sua altura
- Pode ser necessário rotacionar todos os nós no caminho de volta!



Exercício: Remoção nas Árvores AVL

Remover da árvore AVL abaixo as seguintes chaves: 5, 20, 50



Agenda

Conclusões - Árvores AVL

- A altura de uma árvore AVL é aproximadamente igual a $1.44 * \log_2 n$ onde n é o número de nós da árvore.
- O balanceamento usa transformações simples, locais, simétricas, de custo constante
- As operações de busca e inserção tem custo $O(\log_2 n)$ no caso médio e também no caso pior

Técnica	Ordem	Busca	Inserção	Remoção
Busca Sequencial	Não	N	N	N
Busca Binária	Sim	log(N)	N	N
ABB	Sim	h	h	h
AVL	Sim	log(N)	log(N)	log(N)

Conclusões - Árvores AVL

Técnica	Ordem	Busca	Inserção	Remoção
Busca Sequencial	Não	N	N	N
Busca Binária	Sim	log(N)	N	N
ABB	Sim	h	h	h
AVL	Sim	log(N)	log(N)	log(N)

- Inserção simples, após inserir **basta uma rotação para tornar a árvore AVL**
- A remoção **pode precisar** $\log n$ **rotações**
- Precisa armazenar a altura ou o fator de balanceamento (mais dois bits por nó)

Conclusões - Árvores AVL

Técnica	Ordem	Busca	Inserção	Remoção
Busca Sequencial	Não	N	N	N
Busca Binária	Sim	log(N)	N	N
ABB	Sim	h	h	h
AVL	Sim	log(N)	log(N)	log(N)

- Inserção simples, após inserir **basta uma rotação para tornar a árvore AVL**
- A remoção **pode precisar** $\log n$ **rotações**
- Precisa armazenar a altura ou o fator de balanceamento (mais dois bits por nó)

É possível melhorar isto? Sim, árvores preto-vermelho

Agenda

Universidade Federal do ABC

Complexidade de algumas Estruturas de Dados

Data Structure	Time								Space
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Array	Θ(1)	Θ(n)	Θ(n)	Θ(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Queue	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Singly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Doubly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Skip List	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n log(n))
Hash Table	N/A	Θ(1)	Θ(1)	Θ(1)	N/A	O(n)	O(n)	O(n)	O(n)
Binary Search Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)
Cartesian Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(n)	O(n)	O(n)	O(n)
B-Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Red-Black Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Splay Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(log(n))	O(log(n))	O(log(n))	O(n)
AVL Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
KD Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)

http://bigocheatsheet.com/

Agenda

Universidade Federal do ABC

Referências Bibliográficas

- Donald Knuth. The Art of Computer Programming, Volume 3: Sorting and Searching, 3rd Ed., Addison-Wesley, 1997, pages 458-475, section 6.2.3: Balanced Trees.
- Jayme L. Szwarcfiter and Lilian Markezon, Estruturas de Dados e seus Algoritmos, 3ra edição, 2010
- Don Spickler, Tutorial AVL TREES
- Wikipedia: AVL tree, Red-black tree, B tree
- AVL Tree Visualization

  https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

Agenda

Inserção nas Árvores AVL usando o FB

Alg: Inserir o nó da mesma forma que nas ABBs. No caminho de volta até a raíz, atualizar o FB de cada nó **n**

Caso 1 Se o nó tinha FB $\neq 0$ antes da inserção (-1 ou 1), e a nova chave foi inserida na subárvore de menor altura (esquerda ou direita resp) $\Rightarrow$ atualizar FB para 0. Como a altura **não** mudou não é preciso conferir o FB dos antecessores

Caso 2 Se o nó tinha FB $= 0$ antes da inserção $\Rightarrow$ atualizar FB para -1 ou 1 dependendo da sub-árvore onde acontece a inserção (direita ou esquerda resp). Como a altura mudou é preciso conferir o FB dos antecessores

Caso 3 Se o nó tinha FB $\neq 0$ antes da inserção (-1 ou 1), e a nova chave foi inserida na subárvore de maior altura (direita ou esquerda resp) $\Rightarrow$ efetuar a rotação. Como a sub-árvore será rebalanceada não é preciso conferir o FB dos antecessores

Exemplo de Implementação da Inserção AVL usando o FB

```
void avlInsert(avlTreeNode **treeRoot,  int key,
                         int *reviseBalanceFactor) {
  if (*treeRoot == NULL) {
    // update the root to point at newNode
    avlTreeNode *newNode = malloc(sizeof(avlTreeNode));
    newNode->key = key;
    newNode->left = newNode->right = NULL;
    newNode->height = 0;
    *treeRoot = newNode;

    // balanceFactor must be checked!
    *reviseBalanceFactor = 1; // true;
    return;
  }
```

Ver código completo aqui.

Exemplo de Implementação da Inserção AVL usando o FB

```
if (key == (*treeRoot)->key){
  *reviseBalanceFactor = 0; // false;
  return;
}

avlTreeNode *tree = *treeRoot;
// indicates a change in node's balanceFactor
int rebalanceCurrNode, balanceFactorOld = balanceFactor(tree);

if (key < tree->key) { // recursively move to the left
  avlInsert(&tree->left, key, &rebalanceCurrNode);

  // check if balanceFactor must be updated.
  if (rebalanceCurrNode) {
```

Ver código completo aqui.

Universidade Federal do ABC

Exemplo de Implementação da Inserção AVL usando o FB

```
// check if balanceFactor must be updated.
if (rebalanceCurrNode) {

  // case 3: went left from node that is already heavy
  // on the left. violates AVL condition; rotate
  if (balanceFactorOld == leftheavy)
    updateLeftTree(treeRoot, reviseBalanceFactor);

  // case 1: inserting in the left on previously balanced
  // node that now will be heavy on left
  else if (balanceFactorOld == balanced) {
    *reviseBalanceFactor = 1; // true;
    updateHeight(*treeRoot); // update the height
  }
```

Ver código completo aqui.

Exemplo de Implementação da Inserção AVL usando o FB

```
            // case 2: scanning left from node heavy on the
            // right. The node will be balanced, the height is the same
            else *reviseBalanceFactor = 0; // false;
        }

        // no balancing occurs; do not ask previous nodes
        else *reviseBalanceFactor = 0; // false;
    }

    // otherwise recursively move right
    else {
        avlInsert(&tree->right, key, &rebalanceCurrNode);
        // check if balanceFactor must be updated.
        if (rebalanceCurrNode) {
```

Ver código completo aqui.

Universidade Federal do ABC

Rebalanceamento da sub-árvore à esquerda

```
void updateLeftTree(avlTreeNode **pRoot, int *reviseBalanceFactor) {
  avlTreeNode *leftChild = (*pRoot)->left;
  int balanceFactorChild = balanceFactor(leftChild);

  if (balanceFactorChild == leftheavy) {
    // left subtree is also heavy
    caseLLrotateRight(pRoot); // need a single rotation
    *reviseBalanceFactor = 0; // false;
  }
  // is right subtree heavy?
  else if (balanceFactorChild == rightheavy) {
    // make a double rotation
    caseLRrotateLeftRight(pRoot);
    // root is now balanced
    *reviseBalanceFactor = 0; // false;
  }
}
```