Árvores AVL

Algoritmos e Complexidade

LEI-LCC 2010-2011

MBB

Novembro de 2010

Árvore Balanceada (AVL)

- Calcula-se que o custo médio de performance de não utilizar uma árvore totalmente balanceada para efectuar uma pesquisa é da ordem dos 39%.
- Uma solução consiste em reestruturar a árvore uma vez que ela esteja completamente construída. Há algoritmos razoavelmente simples que permitem construir uma árvore balanceada a partir de uma lista ordenada.
- No entanto, na maior parte das aplicações, o facto de haver alterações frequentes torna este solução pouco prática.
- As árvores AVL (o nome resulta das iniciais dos seus inventores) são casos particulares de árvores de pesquisa binária em que as operações de inserção e remoção são desenhadas para manter a árvore muito próxima de um estado balanceado em cada instante.

Árvore Balanceada (AVL)

- Uma árvore AVL nunca excede $(1.44 \log n)$ em altura, o que implica que, mesmo no pior caso, o tempo de pesquisa numa árvore AVL é da ordem de O(log n).
- Isto é equivalente ao caso médio numa árvore binária de pesquisa construída de forma realmente aleatória.
- Numa árvore perfeitamente balanceada, as sub-árvores de cada nó têm a mesma altura.
- Na prática, isto pode não ser possível mas, construindo a árvore cuidadosamente, é possível garantir que as sub-árvores de cada nó não diferem de mais de uma unidade nas suas alturas.

Árvore Balanceada (AVL): Definição

- Uma árvore AVL é uma árvore binária de pesquisa em que as sub-árvores esquerda e direita da raiz não diferem de mais do que uma unidade nas suas alturas.
- Por sua vez, cada uma destas sub-árvores é também uma árvore AVL.
- Cada nó numa árvore AVL está associado a um factor de balanceamento que pode tomar três valores: esquerda mais alta ($/$), alturas iguais ($-$) ou direita mais alta ($\backslash$).
- Note-se que não é necessário que todas as folhas estejam ao mesmo nível, ou mesmo em níveis adjacentes.

Árvore Balanceada (AVL): Implementação

- Implementação ligada:

```
typedef char TreeEntry;
typedef enum balancefactor \
                { LH , EH , RH } BalanceFactor;
typedef struct treenode Treenode;
struct treenode {
   BalanceFactor bf;
   TreeEntry entry;
   TreeNode *left;
   TreeNode *right;
};
```

Árvore Balanceada (AVL): Inserção

- A operação de inserção é, na maior parte dos casos, exactamente igual à das árvores binárias de pesquisa.
- De facto, desde que a inserção não perturbe a altura da sub-árvore correspondente, o balanceamento da raiz não se altera.
- Mais do que isso, mesmo que a altura da sub-árvore correspondente se altere, em muitos casos apenas será necessário actualizar o factor de balanceamento da raiz.
- Só surgem complicações quando a inserção provoca um aumento da altura de uma sub-árvore que já é mais comprida que a sua complementar.

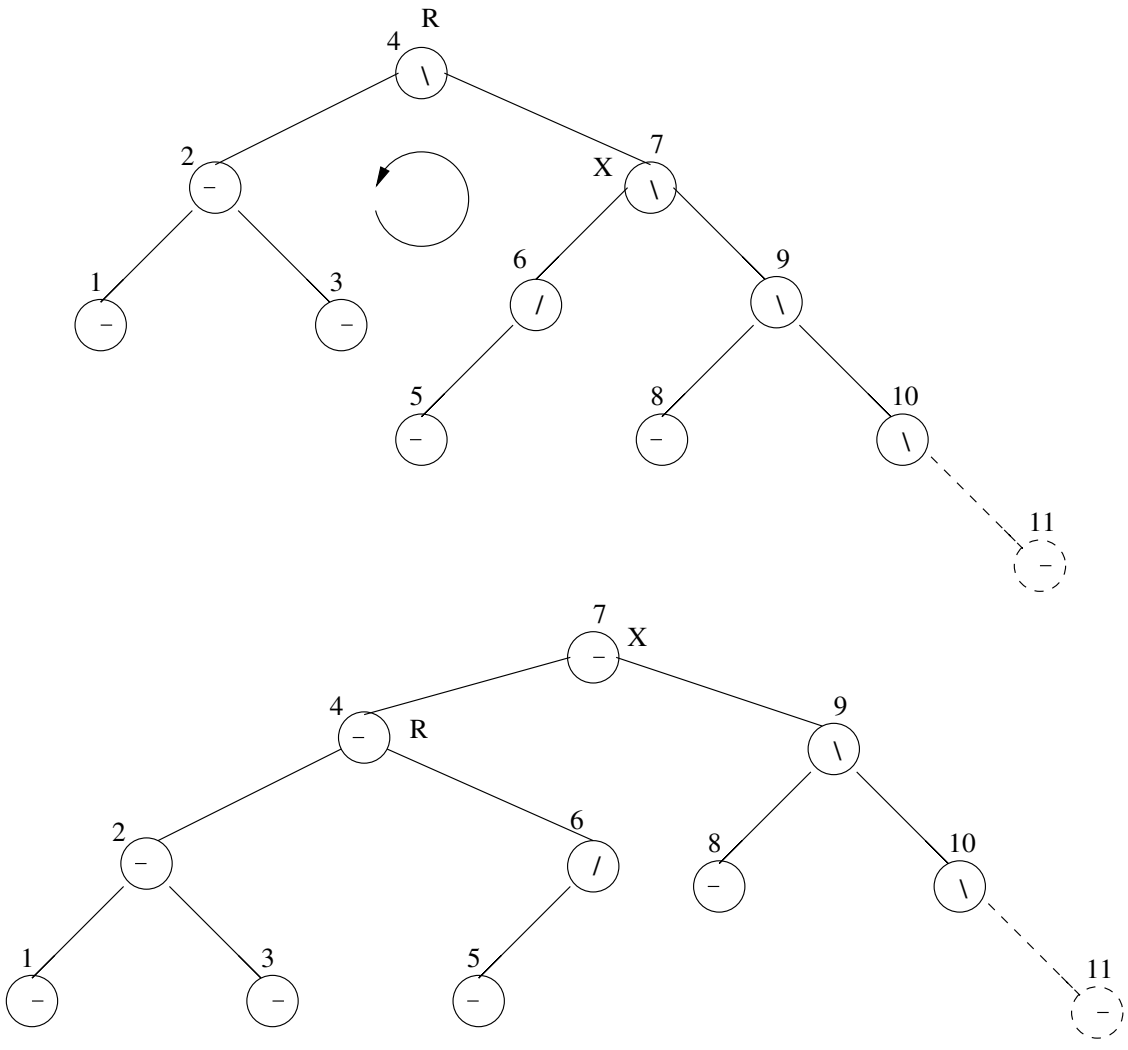
Árvore Balanceada (AVL): Inserção

- Quando isto acontece há duas situações a considerar relativamente ao estado da sub-árvore que causa o problema. Suponhamos que se trata da sub-árvore direita:
  1. Se está desequilibrada para a direita, o problema resolve-se com uma rotação para a esquerda.
  2. Se está desequilibrada para a esquerda, é necessária uma dupla rotação (primeiro à direita e depois à esquerda).
- Note-se que a sub-árvore nunca pode estar balanceada uma vez que sabemos que a sua altura acabou de aumentar.
- No caso do problema surgir na sub-árvore esquerda a solução é simétrica.
- A operação de remoção de uma árvore AVL é bastante complexa e é proposta como exercício.

Árvore Balanceada (AVL): Inserção - 1º Caso

- Inseriu-se um elemento na árvore (com raiz r) e a sub-árvore da direita (com raiz x) passa a ter um desequilíbrio para a direita que causa uma violação das regras de balanceamento.
- A rotação consiste em construir uma nova sub-árvore, com raiz r, sub-árvore esquerda igual à anterior sub-árvore esquerda de r, e sub-árvore direita igual à anterior sub-árvore esquerda de x.
- A árvore AVL final tem x como raiz, a sub-árvore do ponto anterior do lado esquerdo, e , como sub-árvore direita, a original sub-árvore direita de x.
- Os nós x e r da nova árvore têm ambos um factor de balanceamento equilibrado.

Árvore Balanceada (AVL): Inserção - 1º Caso



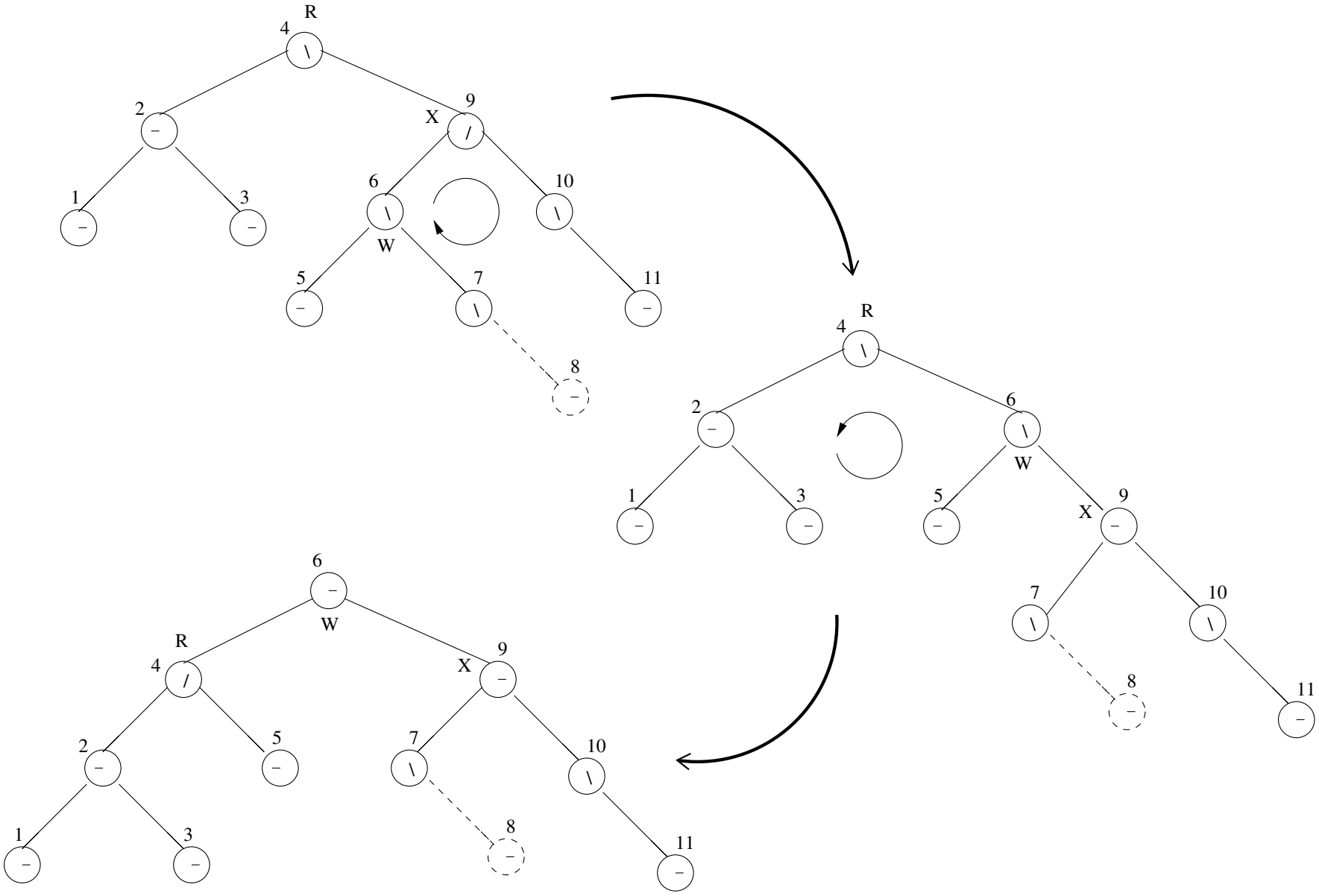
Árvore Balanceada (AVL): Inserção - 2º Caso

- Inseriu-se um elemento na árvore (com raiz r) e a sub-árvore da direita (x) passa a ter um desequilíbrio para a esquerda que causa uma violação das regras de balanceamento. A sub-árvore esquerda de x tem raiz w.
- Na primeira rotação, w passa para o lugar de x que, por sua vez, passa a constituir a raiz da sub-árvore direita de w. A sub-árvore direita original de w é pendurada do lado esquerdo de x.
- Com esta rotação, w passa a ser a raiz de uma sub-árvore desequilibrada para a direita: temos o caso anterior. Daí que seja necessário uma segunda rotação para a esquerda equivalente à descrita anteriormente.

Os novos factores de balanceamento de r e x dependem do factor de balanceamento original de w: (– apenas quando w acaba de ser criado)

w	–	\	/
r	–	/	–
x	–	–	\

Árvore Balanceada (AVL): Inserção - 2º Caso



Árvore Balanceada (AVL): Inserção - 2º Caso