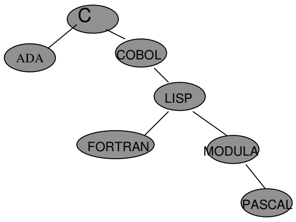
Section 3.2

An Example of AVL rebalancing

Summer 1997 question 4

- Insert the following keys in turn (and in the order shown) into
  - (a) A Binary search tree
  - (b) An AVL tree
  - (c) A Red-Black Tree (See next section for this)
- C, Ada, Cobol, Lisp, Modula, Pascal, Fortran

Solution (a) The binary search tree



Solution: (b) The AVL Tree

New Identifier	After Insertion	After Rebalancing
C		

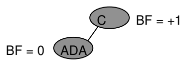
New Identifier | After Insertion | After Rebalancing

C

NO REBALANCING NEEDED

New Identifier | After Insertion | After Rebalancing

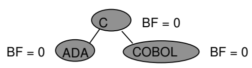
ADA



NO REBALANCING NEEDED

New Identifier | After Insertion | After Rebalancing

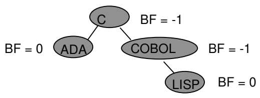
COBOL



NO REBALANCING NEEDED

New Identifier | After Insertion | After Rebalancing

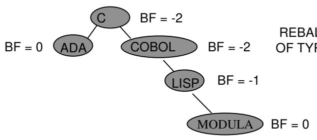
LISP



NO REBALANCING NEEDED

New Identifier | After Insertion | After Rebalancing

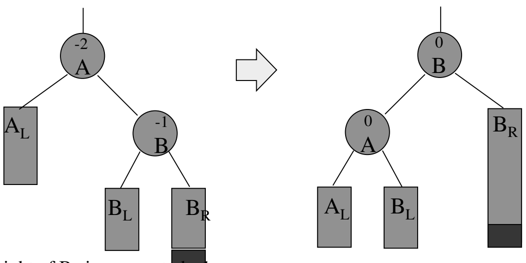
MODULA



REBALANCING NEEDED OF TYPE RR

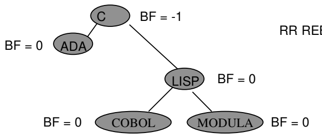
AVL Trees - RR Rotation

Unbalanced following insertion | Rebalanced subtree

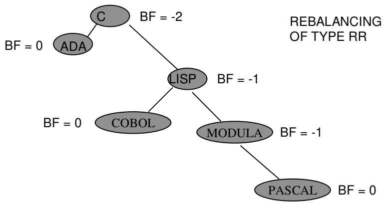


Height of B_R increases to h+1

New Identifier	After Insertion	After Rebalancing
MODULA		RR REBALANCING

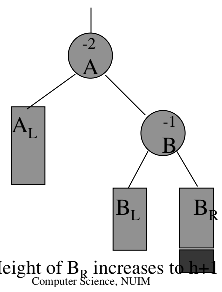


New Identifier	After Insertion	After Rebalancing
PASCAL		REBALANCING NEEDED OF TYPE RR

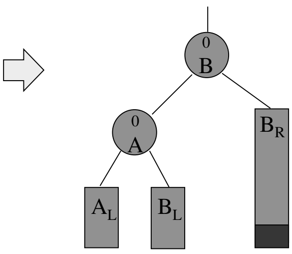


AVL Trees - RR Rotation

Unbalanced following insertion



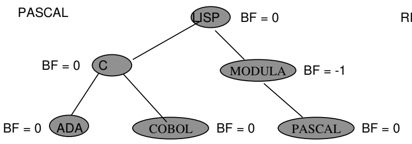
Rebalanced subtree



Height of B_R increases to h+1

New Identifier | After Insertion | After Rebalancing

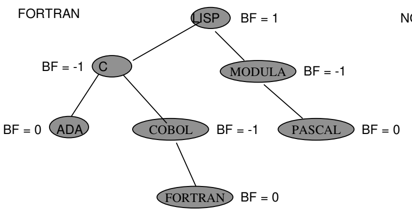
PASCAL



RR REBALANCING

New Identifier | After Insertion | After Rebalancing

FORTRAN



NO REBALANCING NEEDED