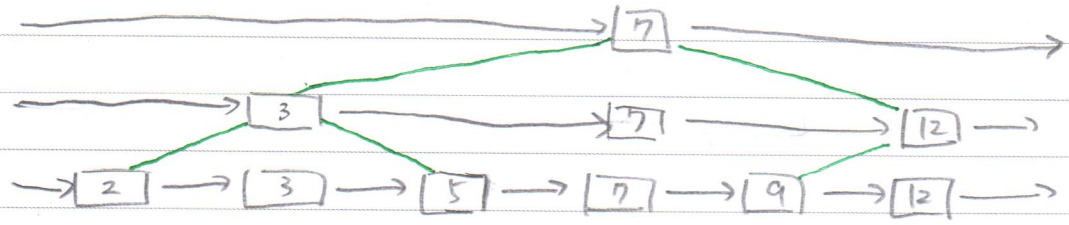
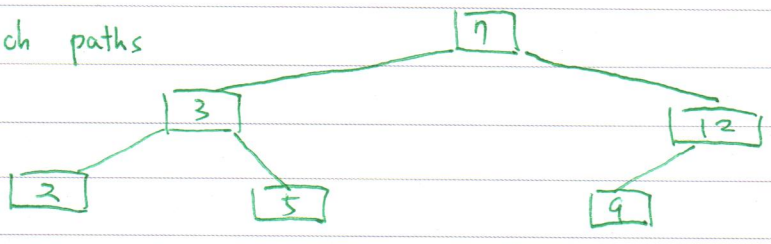
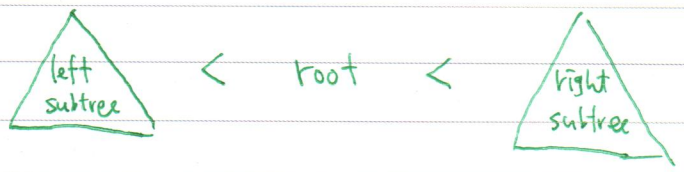
* skip list to BST



the search paths





for every subtree : called binary search tree

* search for key on BST

```
BST-Search (k, T) {
    mid = root of T
    if (k < mid)
        return BST-Search (k, T.left);
    if (k > mid)
        return BST-Search (k, T.right);
    else
        return mid.value;
}
```

worst case : O(h) w/ h being height

* insert : similar to search

* remove :

leaf : simple

one child : simple

two children : find right-most decendent of left-subtree O(h)

Subject : 2012

No. : 10-2

Date : ...../...../.....

* binary search trees

restriction	loose			strict
worst search time	$O(n)$ >	$O(h)$	$\approx$ $O(h)$	> $O(\log n)$
maintenance time after insertion	$O(1)$ <	$O(1)$	< $O(1)$	< $O(n)$
worst height	$O(n)$ >	$O(\log n)$	$\approx$ $O(\log n)$	> $O(\log n)$
	arbitrary BST	RB tree	AVL tree	complete BST

* AVL tree (1962)

Adelson Velskii Landis

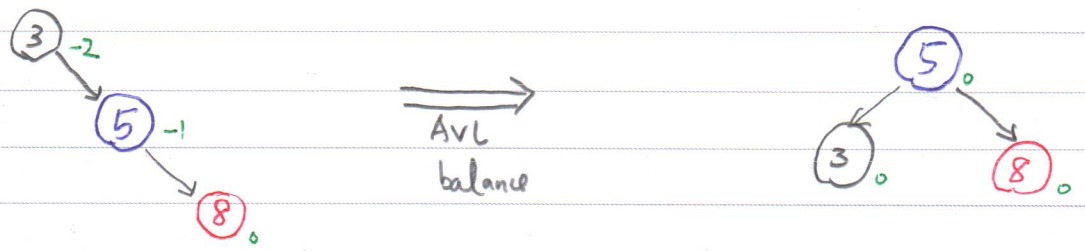
a BST such that

$$\left| \underbrace{\text{height}(T_L)}_{h_L} - \underbrace{\text{height}(T_R)}_{h_R} \right| \leq 1 \quad \text{for every subtree}$$

$$BF(T) = h_L - h_R = \begin{cases} 1 \\ 0 \\ -1 \end{cases}$$

balance factor

* insert (3), (5), (8) to AVL



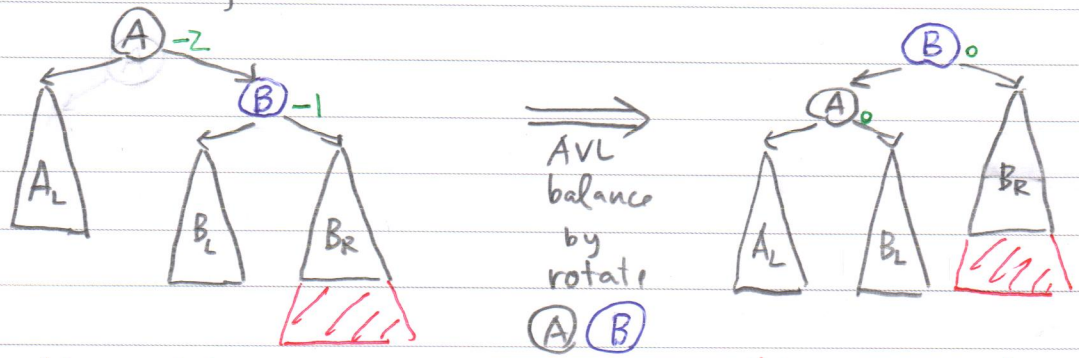
operation : rotate (3) (5)

Subject : 2012

No. : 10-3

Date : ...../...../.....

* case RR during insertion



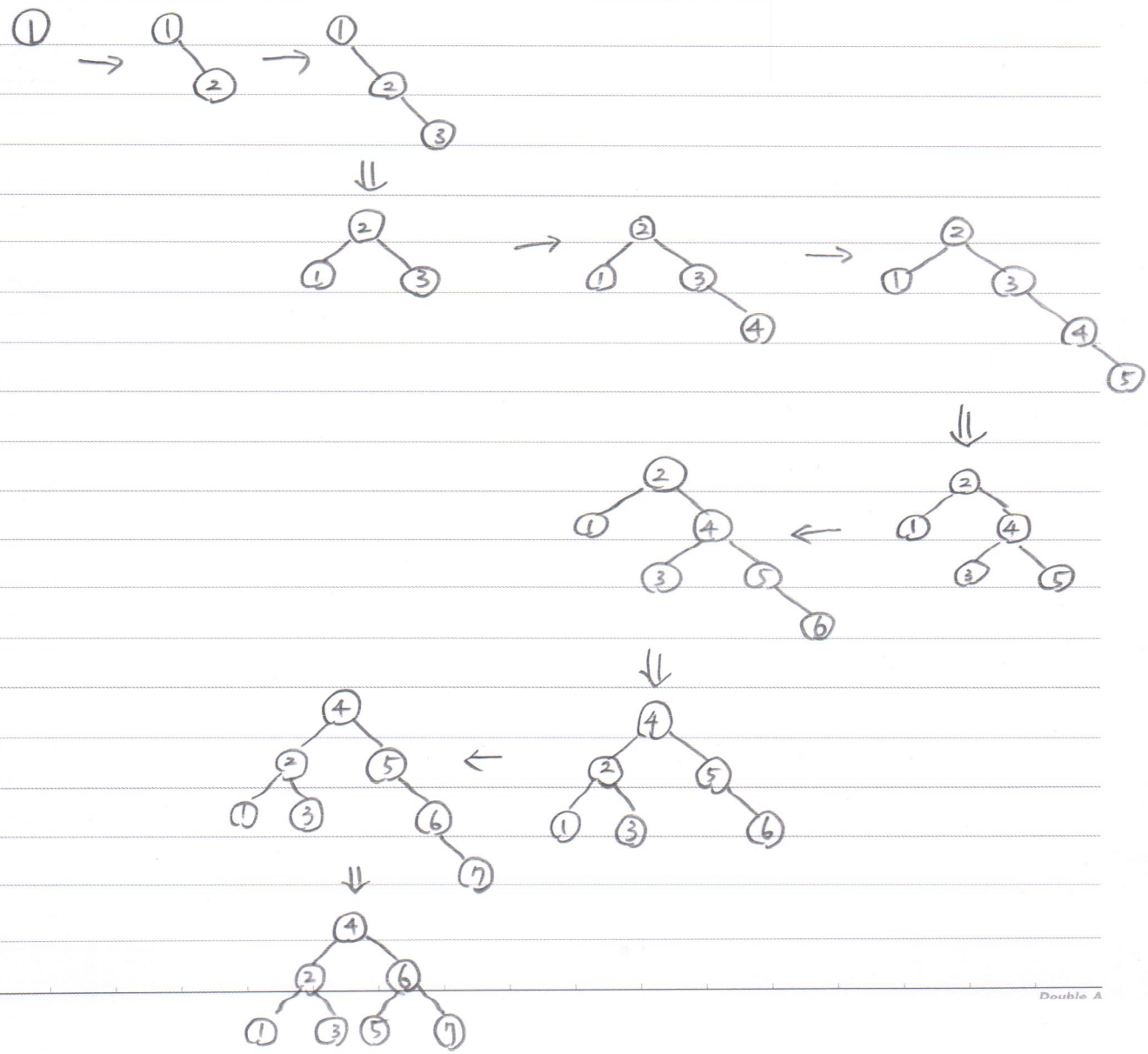
(height ↑)

(height same)

no more rotations

similar for LL

* insert 1, 2, 3, 4, 5, 6, 7



Double A