CENG 218
Design and Analysis of Algorithms

Izmir Institute of Technology

Lecture 10: Balanced search trees

Balanced search trees

A balanced search-tree is a data structure for which a height of $O(\lg n)$ is guaranteed for search/insert/delete.

Examples:

- <u>2-3 trees</u>
- 2-3-4 trees
- <u>AVL trees</u>
- B-trees
- Red-black trees

covered in this course

2-3 trees

Allow 1 or 2 keys per node.

- 2-node: one key, two children
- 3-node: two keys, three children

Perfect balance: Every path from root to null link has same length.

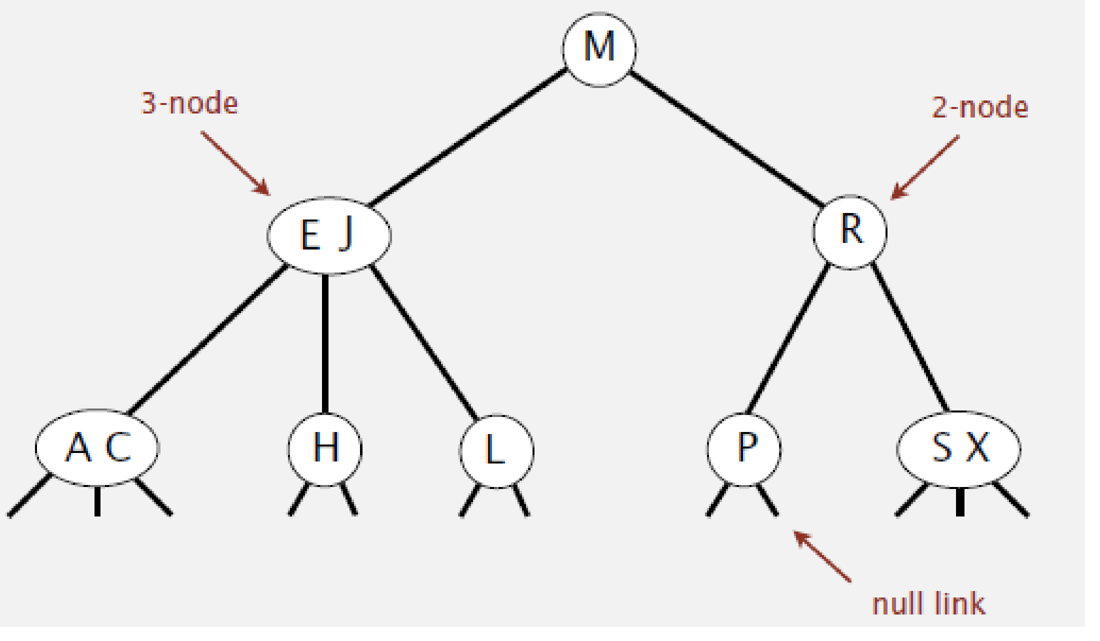


Figure source: Algorithms, 4th edition by Sedgewick and Wayne

2-3 trees

Symmetric order: Inorder traversal yields keys in ascending order.

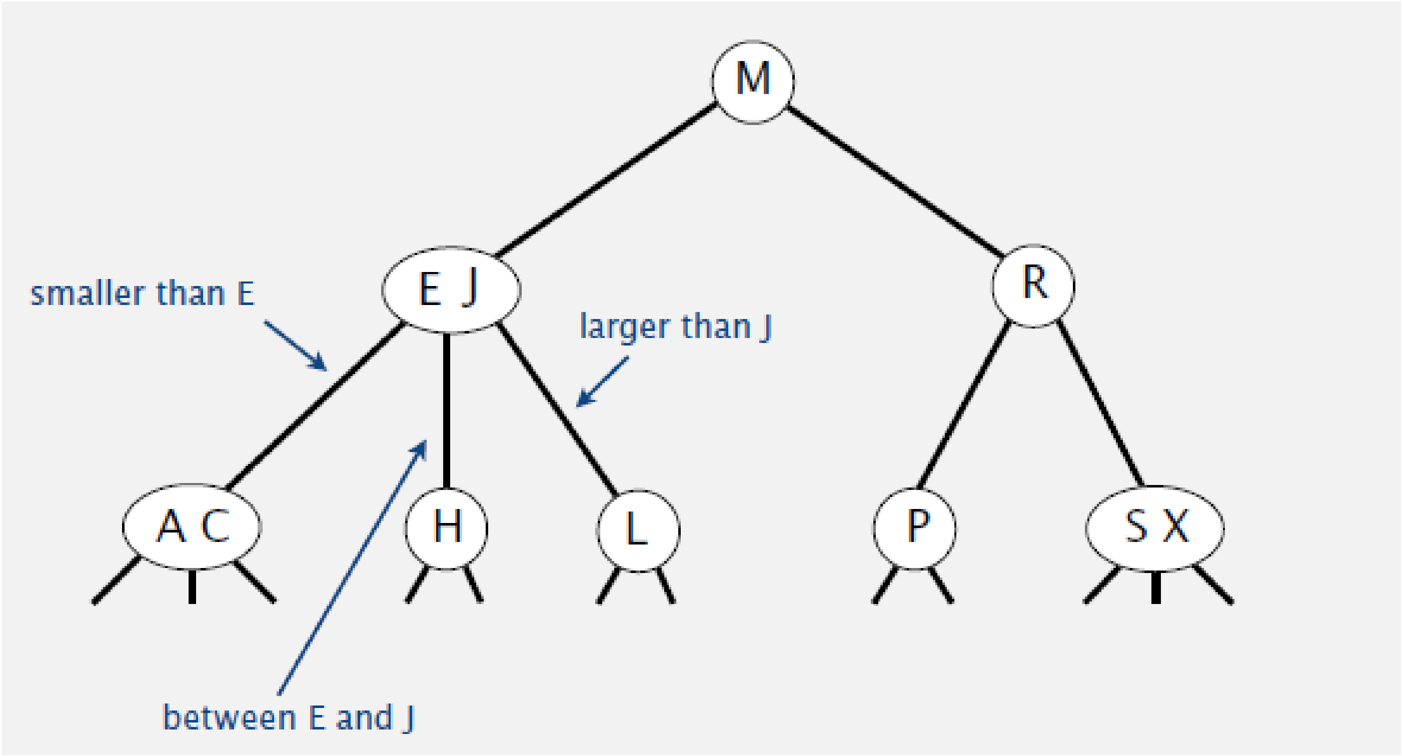


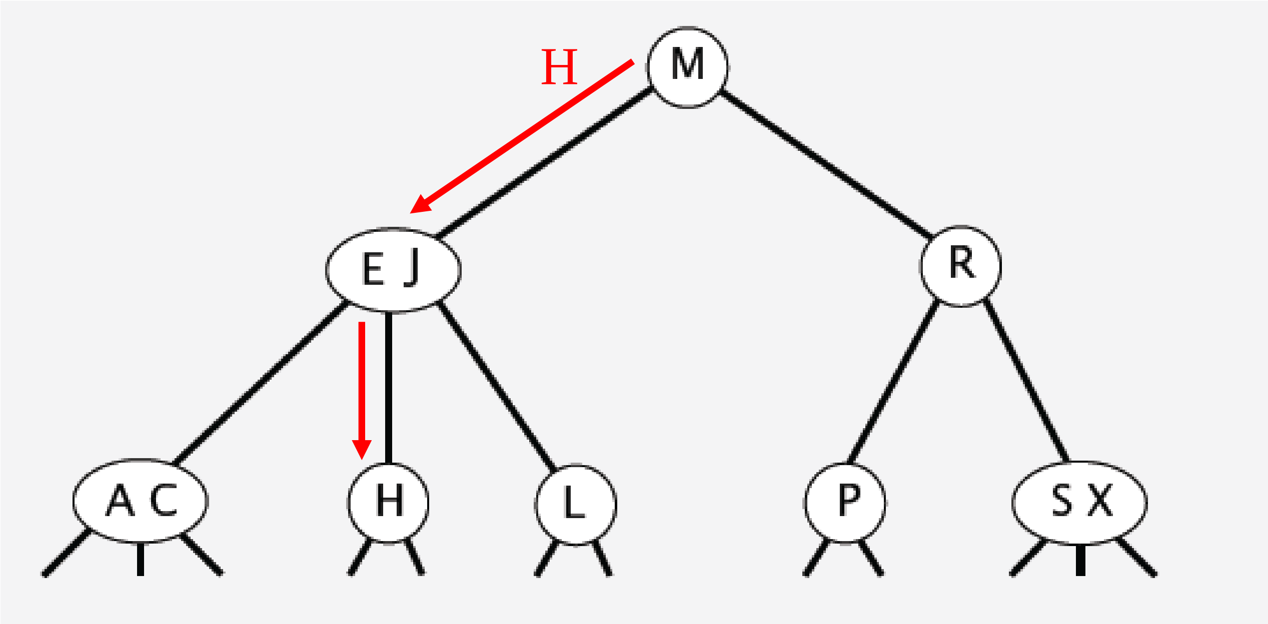
Figure source: Algorithms, 4th edition by Sedgewick and Wayne

2-3 trees

Search:

- Compare search key against keys in node.
- Follow associated link (recursively).

Example: Search for H

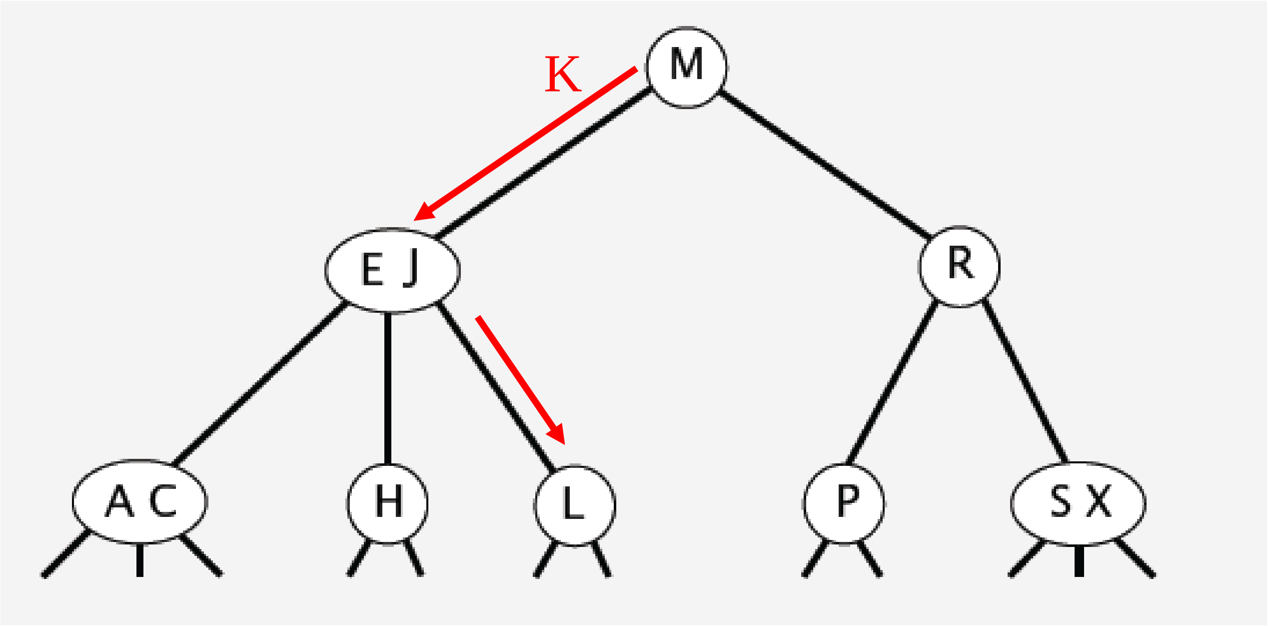


2-3 trees

Insert into a 2-node:

- Search for key.
- Replace 2-node with 3-node.

Example: Insert K

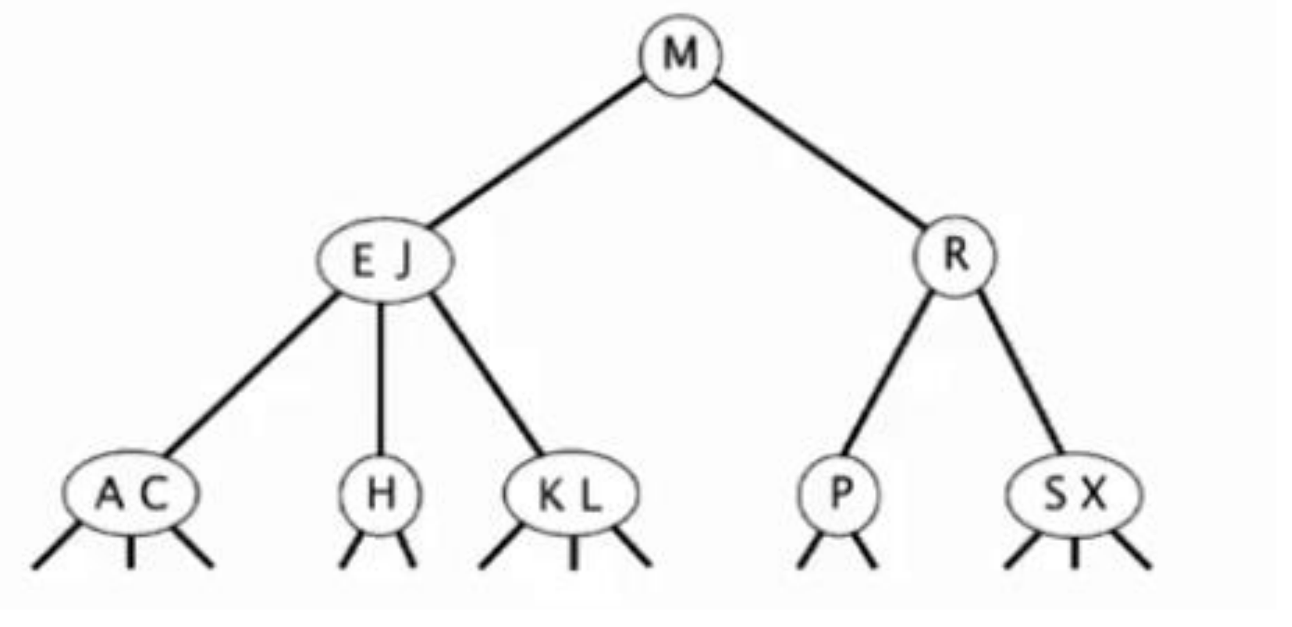


2-3 trees

Insert into a 2-node:

- Search for key.
- Replace 2-node with 3-node.

Example: K is inserted.

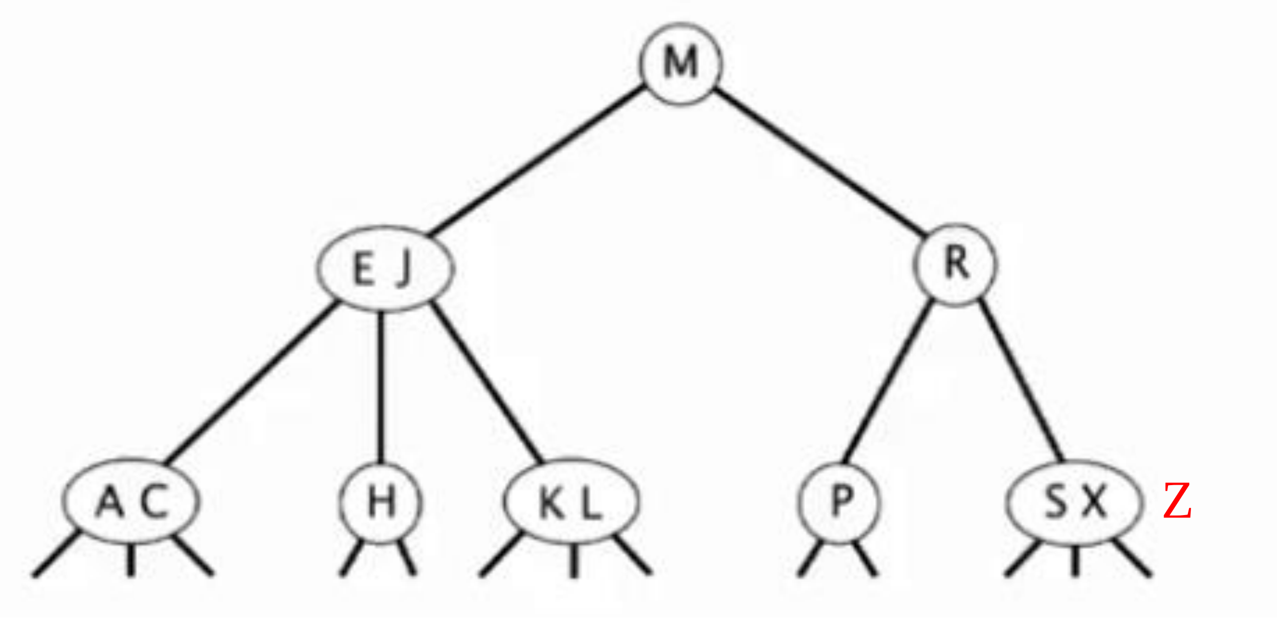


2-3 trees

Insert into a 3-node at the bottom:

- Add new key to 3-node to create a temporary 4-node.
- Move middle key in 4-node into parent (recursively).

Example: Insert Z

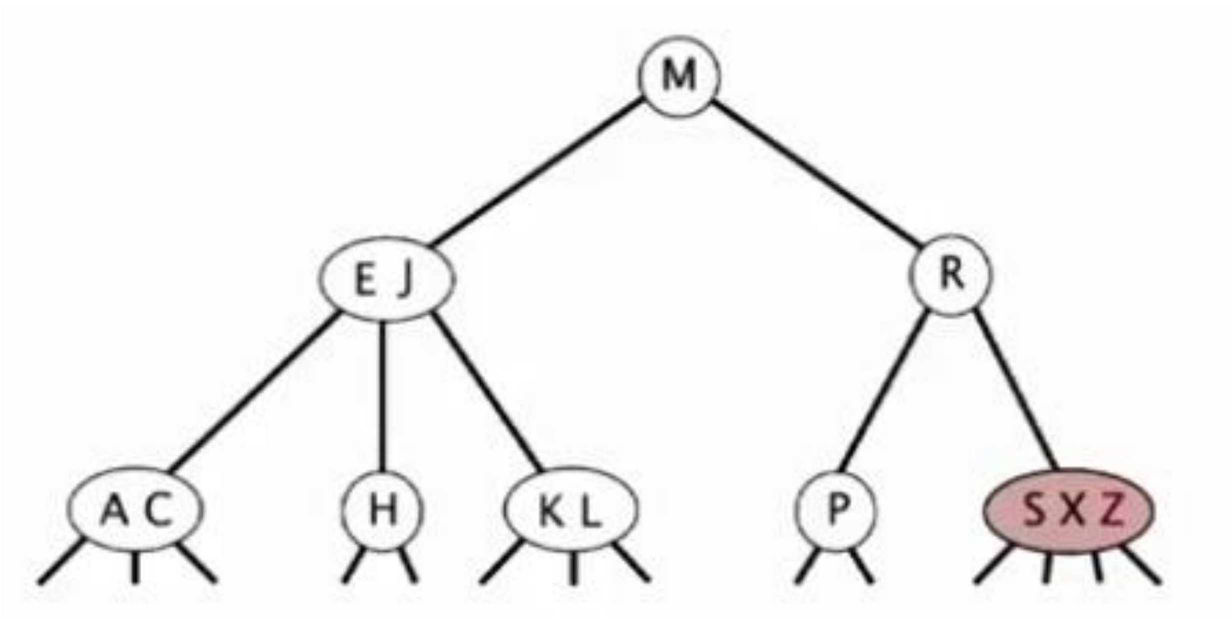


2-3 trees

Insert into a 3-node at the bottom:

- Add new key to 3-node to create a temporary 4-node.
- Move middle key in 4-node into parent (recursively).

Example: Insert Z

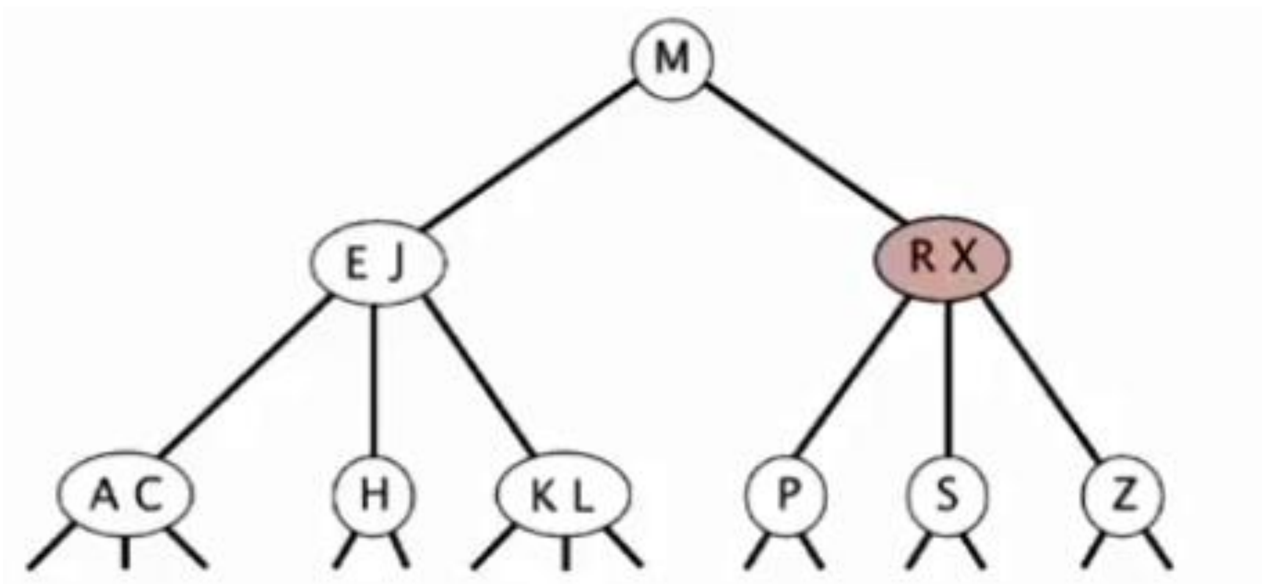


2-3 trees

Insert into a 3-node at the bottom:

- Add new key to 3-node to create a temporary 4-node.
- Move middle key in 4-node into parent (recursively).

Example: Z is inserted.

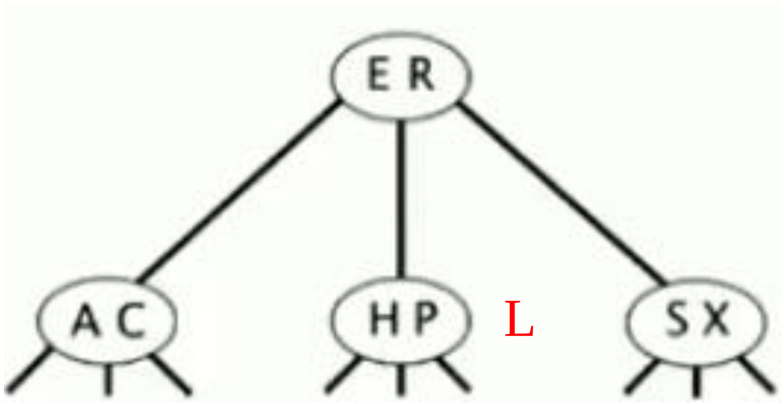


2-3 trees

Insertion when all nodes are 3-node:

- Move middle key in 4-node into parent (recursively).
- If you reach the root, split it into three 2-nodes.

Example: Insert L.

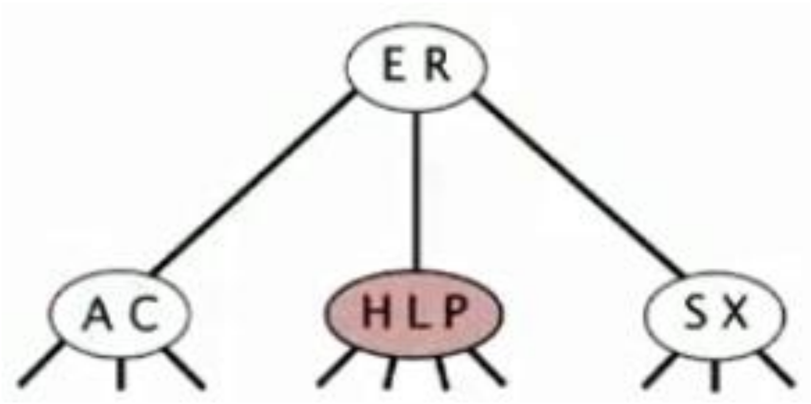


2-3 trees

Insertion when all nodes are 3-node:

- Move middle key in 4-node into parent (recursively).
- If you reach the root, split it into three 2-nodes.

Example: Insert L.

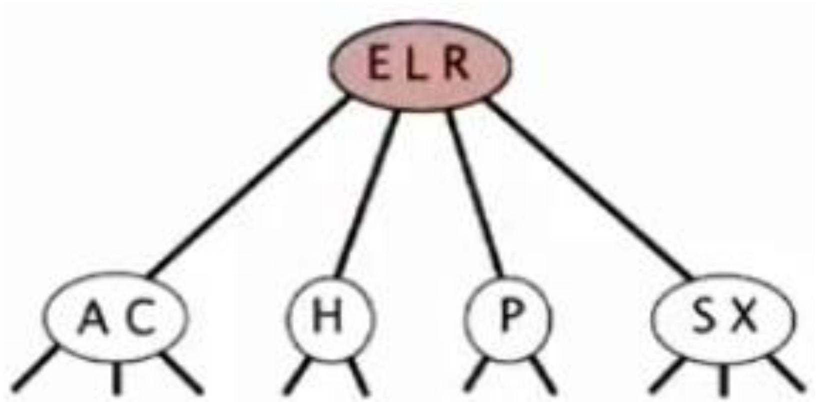


2-3 trees

Insertion when all nodes are 3-node:

- Move middle key in 4-node into parent (recursively).
- If you reach the root, split it into three 2-nodes.

Example: Insert L.

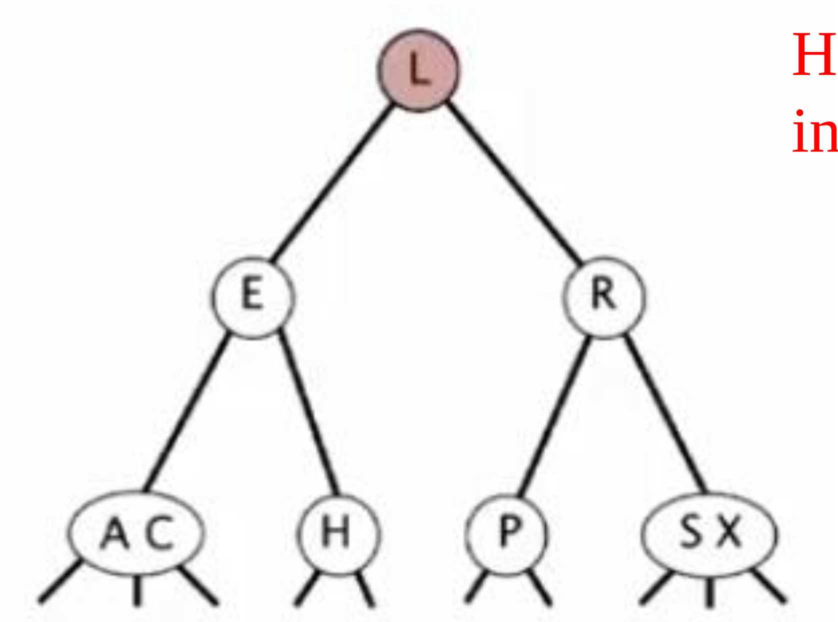


2-3 trees

Insertion when all nodes are 3-node:

- Move middle key in 4-node into parent (recursively).
- If you reach the root, split it into three 2-nodes.

Example: L is inserted.



Height of the tree increases by 1.

Comparison of elementary symbol-table implementations

implementation	worst-case cost (after N inserts)			average case (after N random inserts)		
	search	insert	delete	search hit	insert	delete
sequential search (unordered list)	N	N	N	N/2	N	N/2
binary search (ordered array)	lg N	N	N	lg N	N/2	N/2
BST	N	N	N	1.39 lg N	1.39 lg N	?
2-3 tree	c lg N	c lg N	c lg N	c lg N	c lg N	c lg N

c constants depend on implementations

Figure source: Algorithms, 4th edition by Sedgewick and Wayne

AVL trees

- AVL tree is another balanced binary search tree.
- At any time, depth difference between the right and left subtrees are computed ($depth_{left}$ - $depth_{right}$).
- If the difference goes up to two, left/right rotations are performed to bring the tree back into balance.

Note: AVL stands for Adelson-Velsky and Landis, names of the inventors of the tree

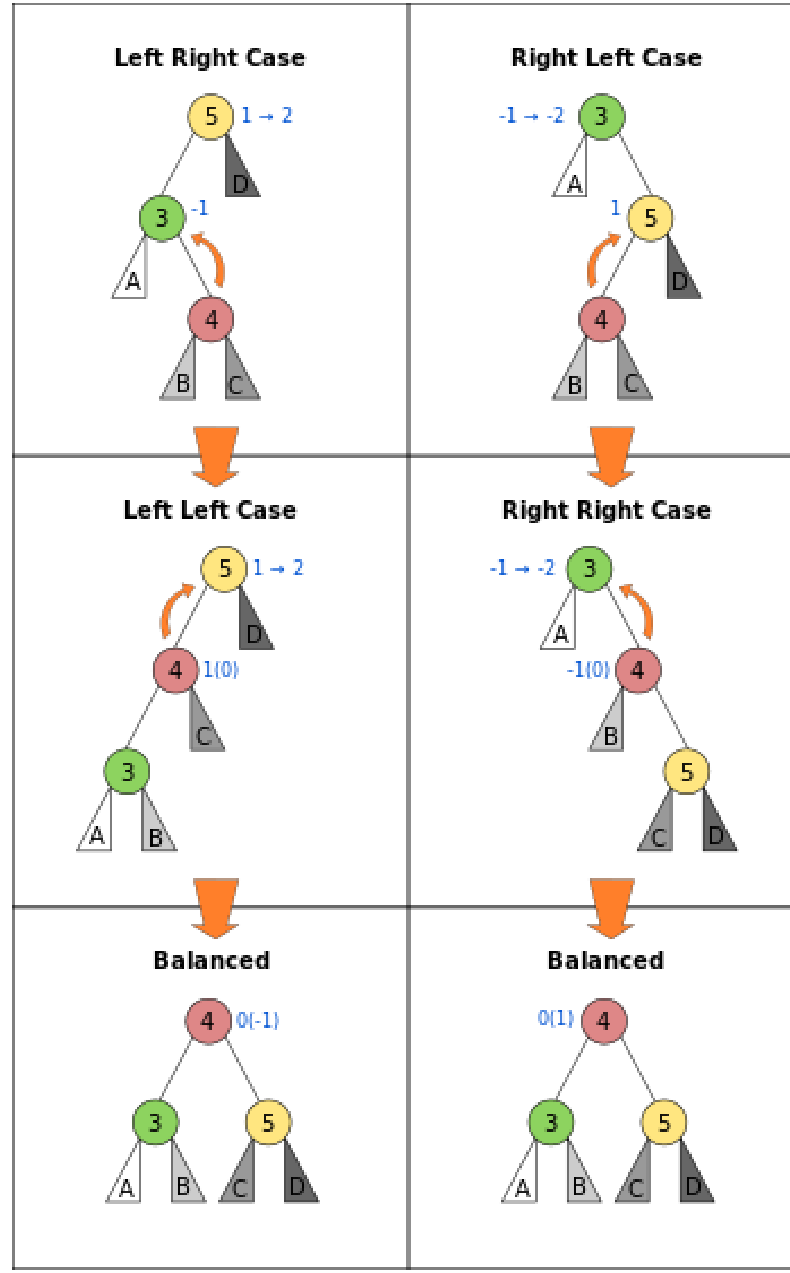
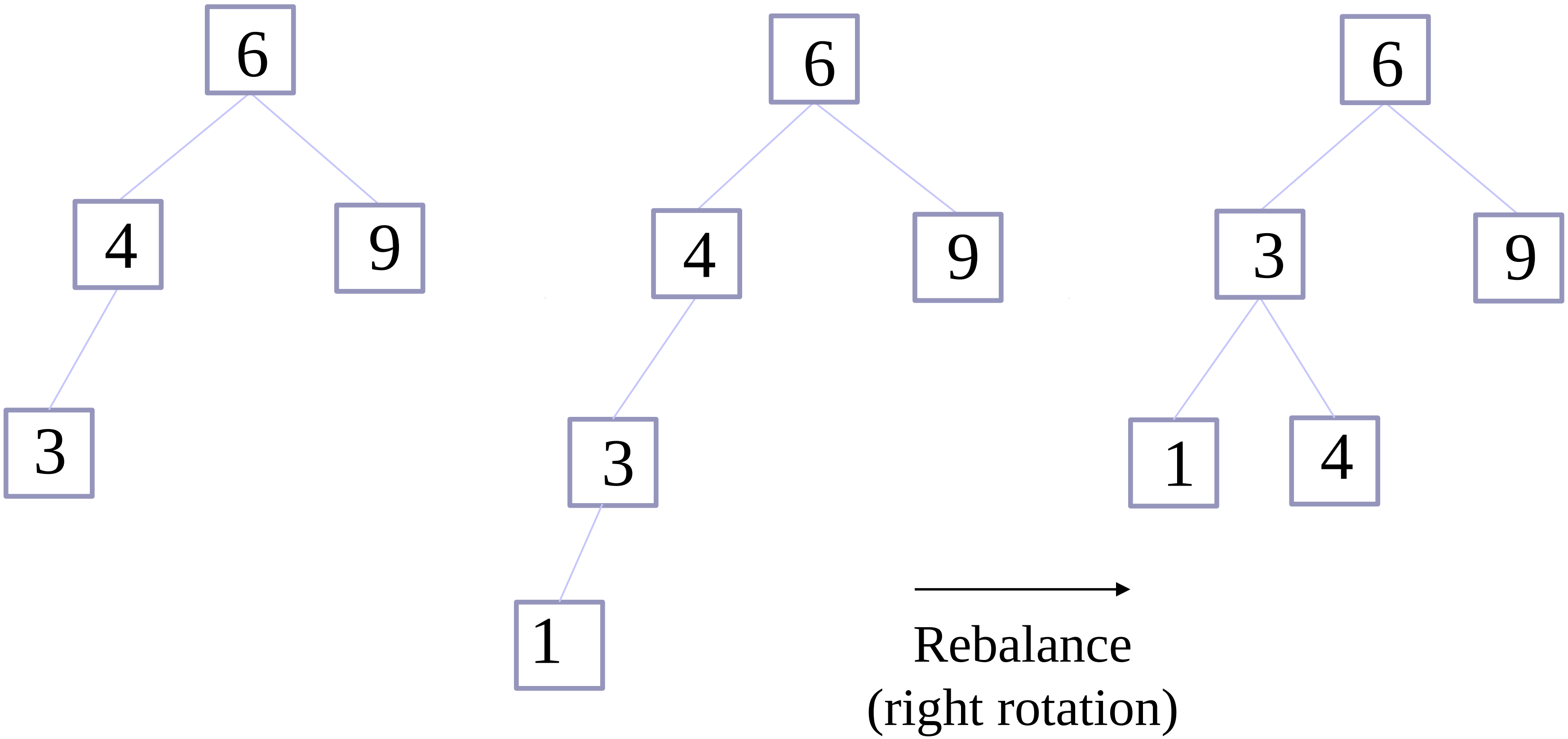


Figure source: Wikipedia

AVL trees

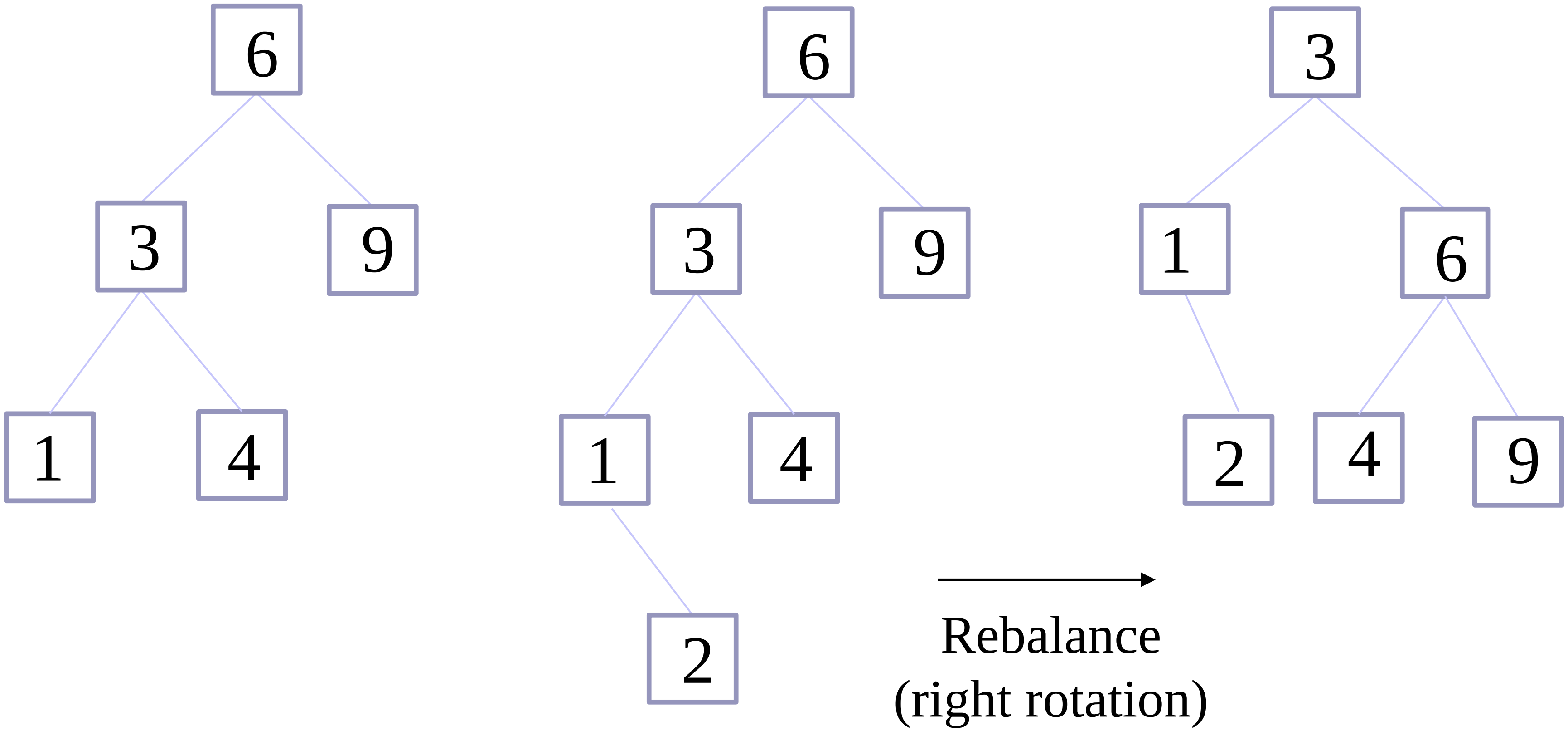
Example: Insert 1



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

AVL trees

Example: Insert 2

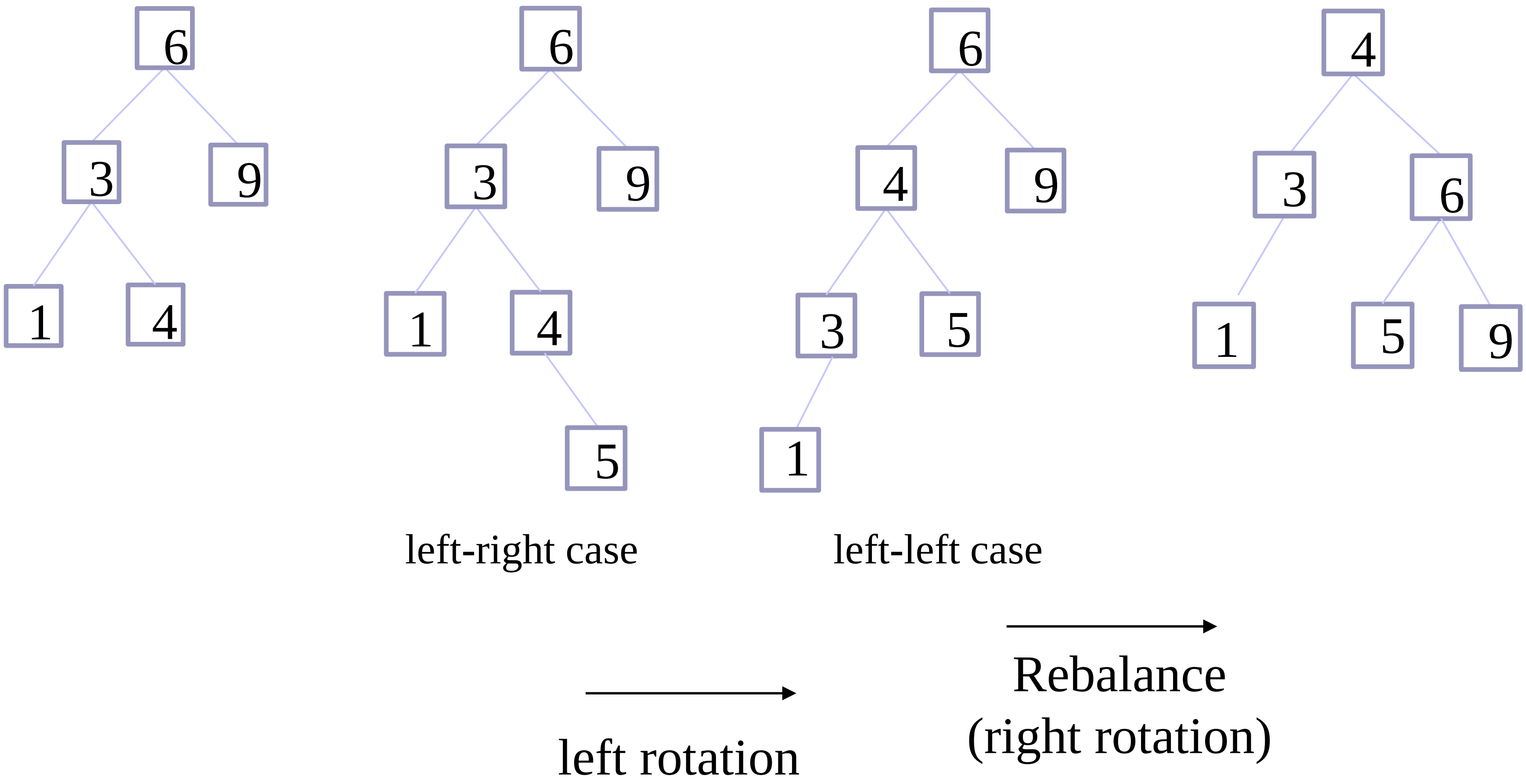


Visualization:

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

AVL trees

Example: Insert 5



Analysis

- If depth difference occurs, perform rotation and terminate.

- **Running time:** $O(\lg n)$ with $O(1)$ rotations.

- Delete operation also has same asymptotic running time and number of rotations as Insert operation.

Summary of symbol-table implementations

implementation	worst-case cost (after N inserts)			average case (after N random inserts)		
	search	insert	delete	search hit	insert	delete
sequential search (unordered list)	N	N	N	N/2	N	N/2
binary search (ordered array)	lg N	N	N	lg N	N/2	N/2
BST	N	N	N	1.39 lg N	1.39 lg N	?
2-3 tree	c lg N	c lg N	c lg N	c lg N	c lg N	c lg N
AVL tree	1.44 lg N	1.44 lg N	1.44 lg N *	1.00 lg N *	1.00 lg N *	1.00 lg N *

* http://pages.cs.wisc.edu/~ealexand/cs367/NOTES/AVL-Trees/index.html