COMP3143
Data Structures and Algorithms

AVL-Trees (Part 1: Single Rotations)

Balance Binary Search Tree

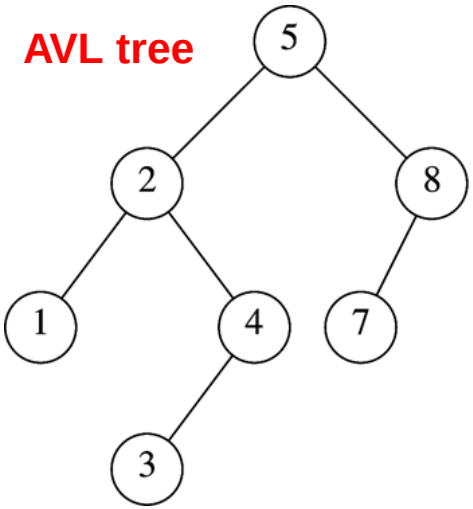
- Worst case height of **binary search tree**: N-1
  - Insertion, deletion can be O(N) in the worst case
- We want a tree with small height
- Height of a binary tree with N nodes is at least $\Theta(\log N)$
- Goal: keep the height of a binary search tree O(log N)
- Balanced binary search trees
  - Examples: AVL tree, red-black tree

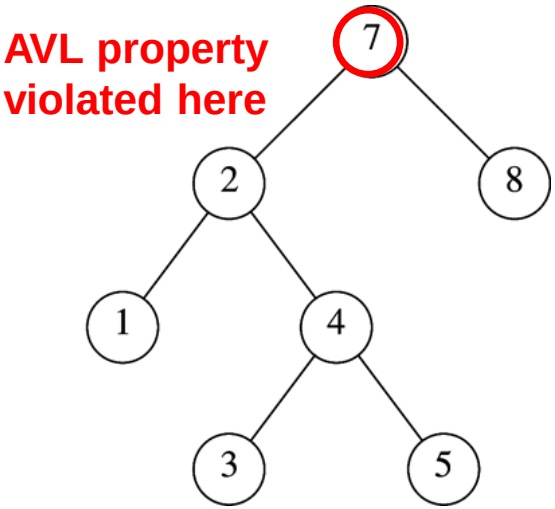
Balanced Tree?

- Suggestion 1: the left and right subtrees of root have the same height
  - But the left and right subtrees may be linear lists!
- Suggestion 2: every node must have left and right subtrees of the same height
  - Only complete binary trees satisfy
  - Too rigid to be useful
- Our choice: for each node, the height of the left and right subtrees can differ at most 1

AVL Tree (invented by Adelson-Velsky and Landis in 1962)

- An AVL tree is a binary search tree in which
  - for *every* node in the tree, the height of the left and right subtrees differ by at most 1.
- Height of subtree: Max # of edges to a leaf
- Height of an **empty subtree: -1**
  - Height of one node: 0





Georgy Adelson-Velsky

14 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

Georgy Maximovich Adelson-Velsky (Russian: Гео́ргий Макси́мович Адельсо́н-Ве́льский; name is sometimes transliterated as **Georgii Adelson-Velskii**) (8 January 1922 – 26 April 2014) was a Soviet and Israeli mathematician and computer scientist.

Born in Samara, Adelson-Velsky was originally educated as a pure mathematician. His first paper, with his fellow student and eventual long-term collaborator Alexander Kronrod in 1945, won a prize from the Moscow Mathematical Society.[1] He and Kronrod were the last students of Nikolai Luzin, and he earned his doctorate in 1949 under the supervision of Israel Gelfand.[2]

He began working in artificial intelligence and other applied topics in the late 1950s.[1] Along with Evgenii Landis, he invented the AVL tree in 1962. This was the first known balanced binary search tree data structure.[3]

Beginning in 1963, Adelson-Velsky headed the development of a computer chess program at the Institute for Theoretical and Experimental Physics in Moscow. His innovations included the first use of bitboards (a now-common method for representing game positions) in computer chess.[4] The program defeated Kotok-McCarthy in the first chess match between computer programs, also in 1966,[4] and it evolved into Kaissa, the first world computer chess champion.[5]

In August 1992, Adelson-Velsky moved to Israel, and he resided in Ashdod.[1]

He worked as a professor in the department of Mathematics and Computer Science, Bar Ilan University.

Adelson-Velsky died on 26 April 2014, aged 92, in his apartment in Giv'atayim, Israel.[6]

Selected publications [edit]

- Adel'son-Vel'skiĭ, G. M.; Kronrod, A. S. (1945), "On a direct proof of the analyticity of a monogenic function", *Doklady Akademii Nauk SSSR*, New Series, **50**: 7–9, MR 0051912.
- Adel'son-Vel'skiĭ, G. M.; Landis, E. M. (1962), "An algorithm for organization of information", *Doklady Akademii Nauk SSSR*, **146**: 263–266, MR 0156719.
- Adel'son-Vel'skiĭ, G. M.; Arlazarov, V. L.; Bitman, A. R.; Životovskiĭ, A. A.; Uskov, A. V. (1970), "On programming a computer for playing chess", *Akademiya Nauk SSSR I Moskovskoe Matematicheskoe Obshchestvo*, **25** (2 (152)): 221–260, MR 0261965. Translated as "Programming a computer to play chess", *Russian Mathematical Surveys* 25: 221–262, 1970, doi:10.1070/RM1970v025n02ABEH003792

Evgenii Landis

Article Talk

From Wikipedia, the free encyclopedia

Evgenii Mikhailovich Landis (Russian: Евге́ний Миха́йлович Ла́ндис, *Yevgeny Mikhaylovich Landis*; 6 October 1921 – 12 December 1997) was a Soviet mathematician who worked mainly on partial differential equations.

Life [edit]

Landis was born in Kharkiv, Ukrainian SSR, Soviet Union. He was Jewish. He studied and worked at the Moscow State University, where his advisor was Alexander Kronrod, and later Ivan Petrovsky. In 1946, together with Kronrod, he rediscovered Sard's lemma, unknown in USSR at the time.

Later, he worked on uniqueness theorems for elliptic and parabolic differential equations, Harnack inequalities, and Phragmén–Lindelöf type theorems. With Georgy Adelson-Velsky, he invented the AVL tree data structure (where "AVL" stands for **A**delson-**V**elsky **L**andis).

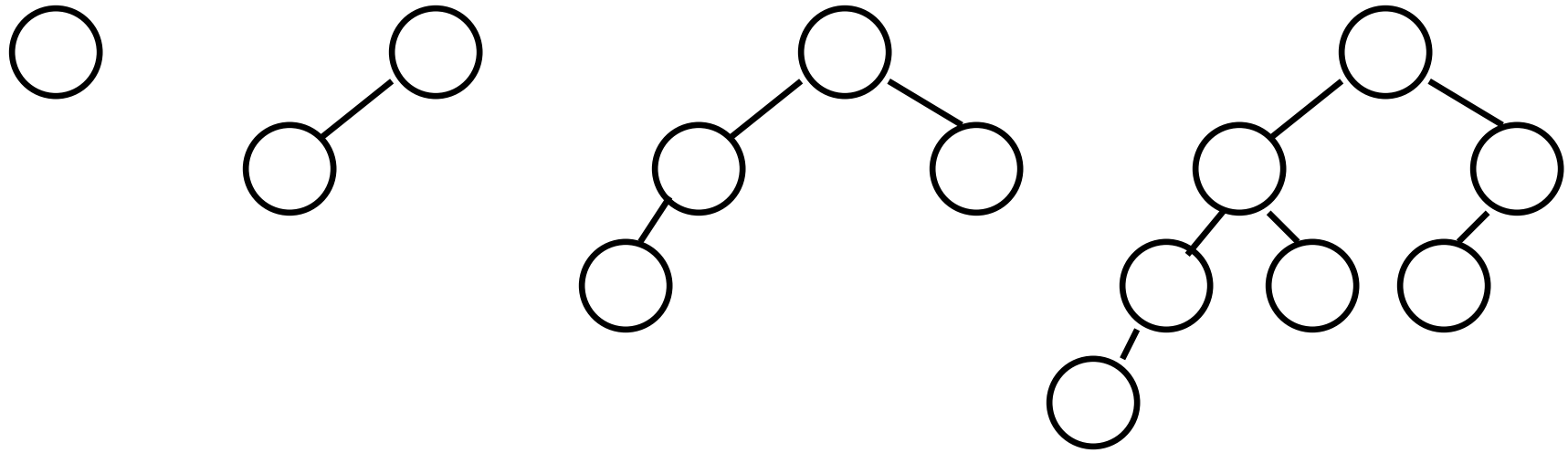
He died in Moscow. His students include Yulij Ilyashenko.

External links [edit]

- Evgenii Landis at the Mathematics Genealogy Project
- Biography of Y.M. Landis at the International Centre for Mathematical Sciences.

AVL Tree with Minimum Number of Nodes

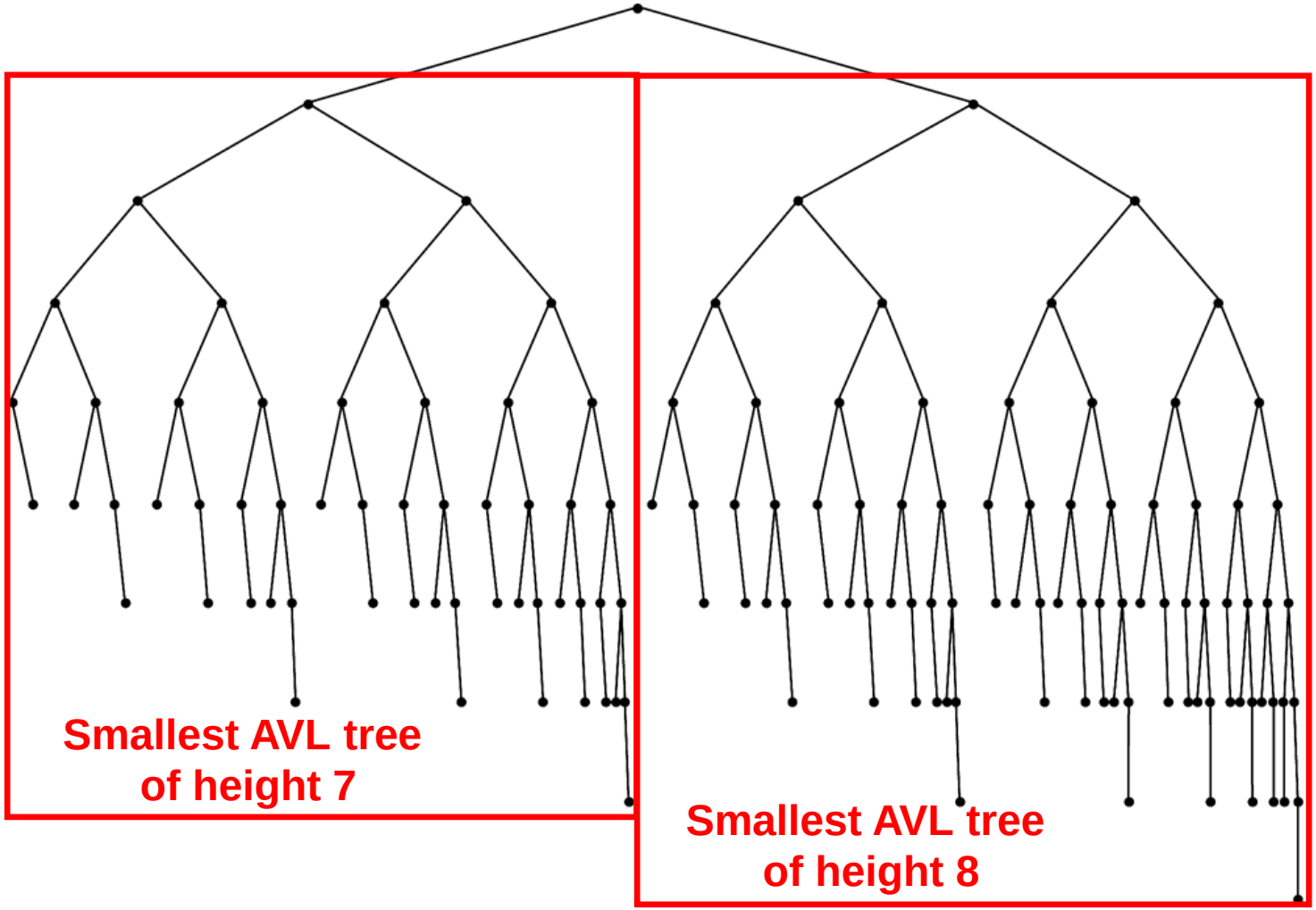
We use N_h to denote the minimum number of nodes in an AVL tree with height of tree h.



$N_0 = 1$ $N_1 = 2$ $N_2 = 4$ $N_3 = N_1 + N_2 + 1 = 7$

height of left=?

Height right=?



Smallest AVL tree of height 9

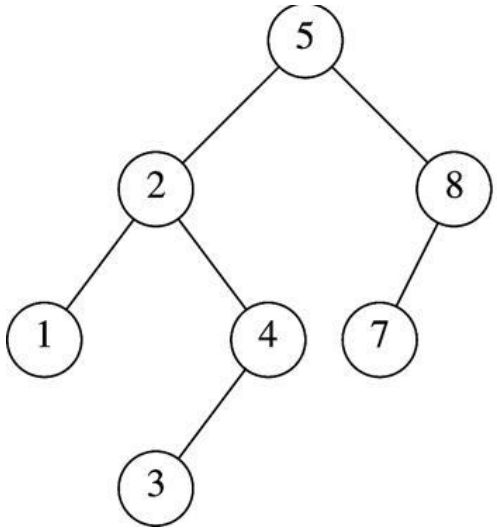
Height of AVL Tree

- Denote N_h the minimum number of nodes in an AVL tree of height h
- $N_0 = 1$, $N_1 = 2$ (base)
  $N_h = N_{h-1} + N_{h-2} + 1$ (recursive relation)
- $N > N_h = N_{h-1} + N_{h-2} + 1$ (N_h is an increasing function)

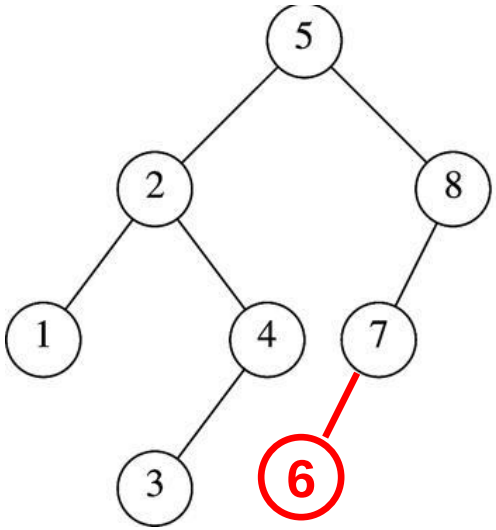
  $> 2\ N_{h-2} > 4\ N_{h-4} > \ldots > 2^i\ N_{h-2i}$
- If h is even, let $i = h/2 - 1$. The equation becomes $N > 2^{h/2-1} N_2$
  $\Rightarrow N > 2^{h/2-1} * 4 \Rightarrow h = O(\log N)$
- If h is odd, let $i = (h-1)/2$. The equation becomes $N > 2^{(h-1)/2} N_1$
  $\Rightarrow N > 2^{(h-1)/2} * 2 \Rightarrow h = O(\log N)$
- Thus, many operations (i.e. searching) on an AVL tree will take O(log N) time

Insertion in AVL Tree

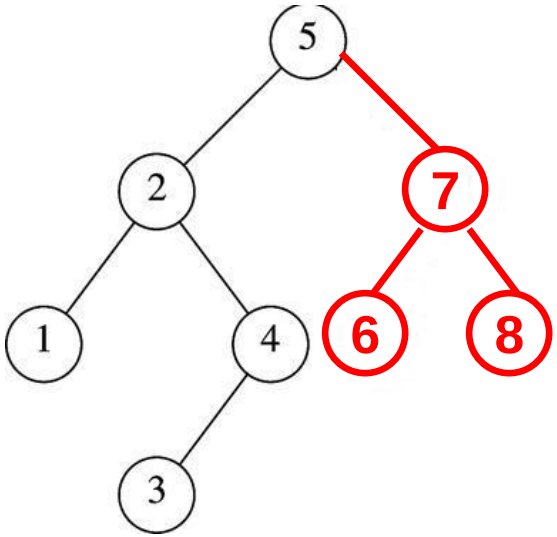
- Basically follows insertion strategy of binary search tree
  - But may cause violation of AVL tree property
- Restore the destroyed balance condition if needed



Original AVL tree



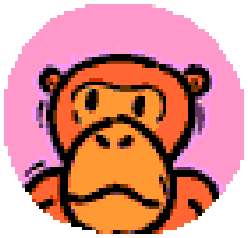
Insert 6
Property violated

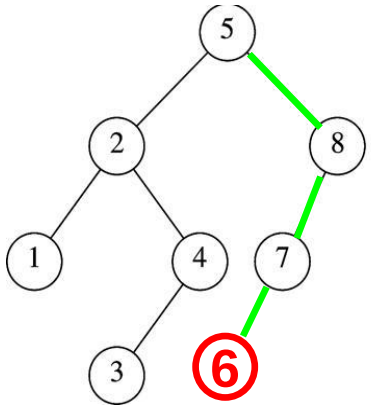


Restore AVL property

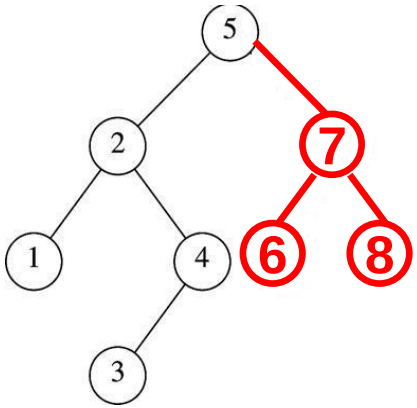
Some Observations

- After an insertion, only nodes that are on the path from the insertion point to the root might have their balance altered
  - Because only those nodes have their subtrees altered
- Rebalance the tree at the deepest such node guarantees that the entire tree satisfies the AVL property





Nodes 5,8,7 might have balance altered



Rebalance node 7 guarantees the whole tree be AVL

Different Cases for Rebalance

- Denote the node that must be rebalanced by α
  - Case 1: an insertion into the left subtree of the left child of α
  - Case 2: an insertion into the right subtree of the left child of α
  - Case 3: an insertion into the left subtree of the right child of α
  - Case 4: an insertion into the right subtree of the right child of α
- Cases 1&4 are mirror image symmetries with respect to α, as are cases 2&3

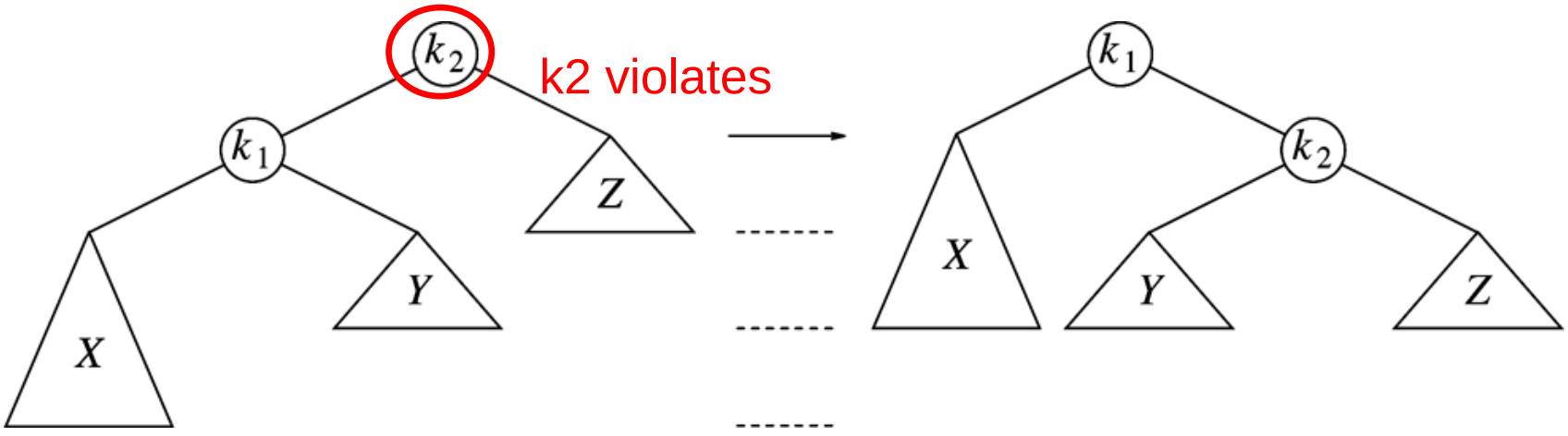
Rotations

- Rebalance of AVL tree is done with simple modification to tree, known as rotation
- Insertion occurs on the "outside" (i.e., left-left or right-right) is fixed by single rotation of the tree
- Insertion occurs on the "inside" (i.e., left-right or right-left) is fixed by double rotation of the tree

Insertion Algorithm

- First, insert the new key as a new leaf just as in an ordinary binary search tree
- Then trace the path from the new leaf towards the root. For each node x encountered, check if heights of left(x) and right(x) differ by at most 1
  - If yes, proceed to parent(x)
  - If not, restructure by doing either a single rotation or a double rotation
- Note: once we perform a rotation at a node x, we won't need to perform any rotation at any ancestor of x.

Single Rotation to Fix Case 1 (left-left)



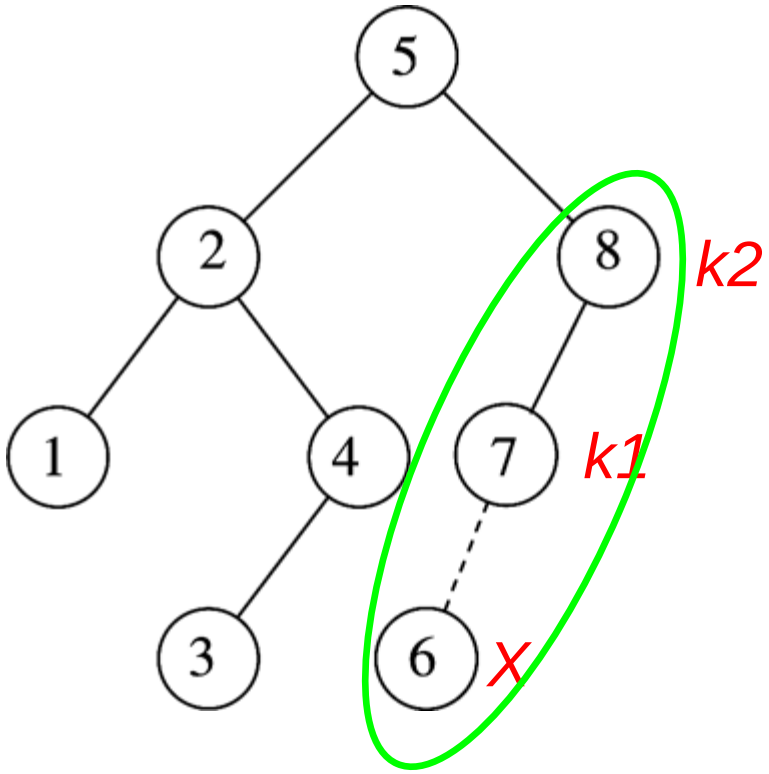
An insertion in subtree X,
AVL property violated at node k2

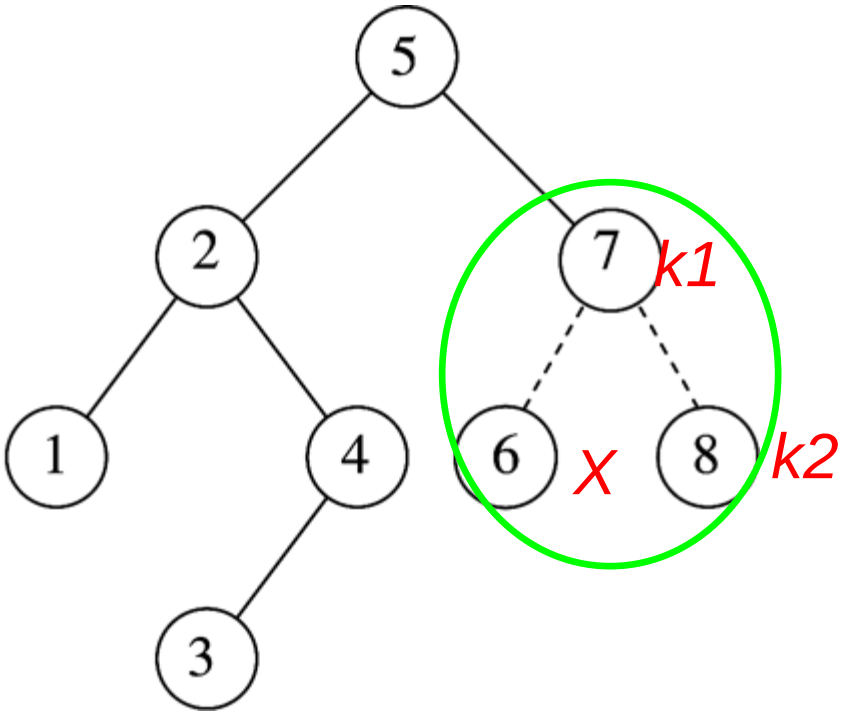
Solution: single rotation

AVL-property quiz:

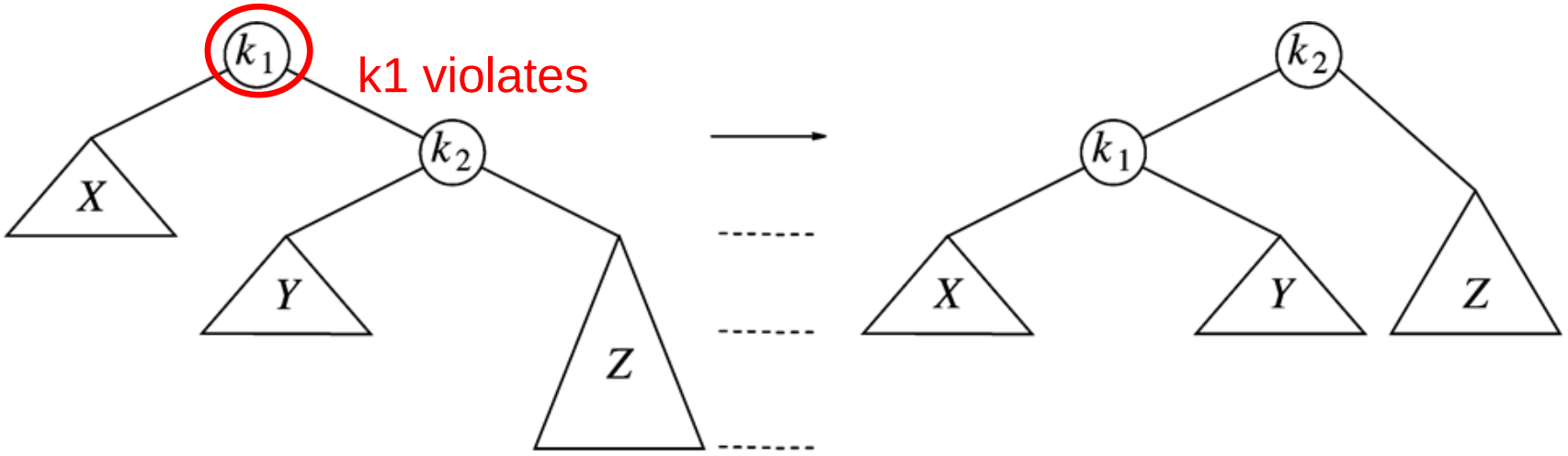
1. Can Y have the same height as the new X?
2. Can Y have the same height as Z?

Single Rotation Case 1: Example





Single Rotation to Fix Case 4 (right-right)

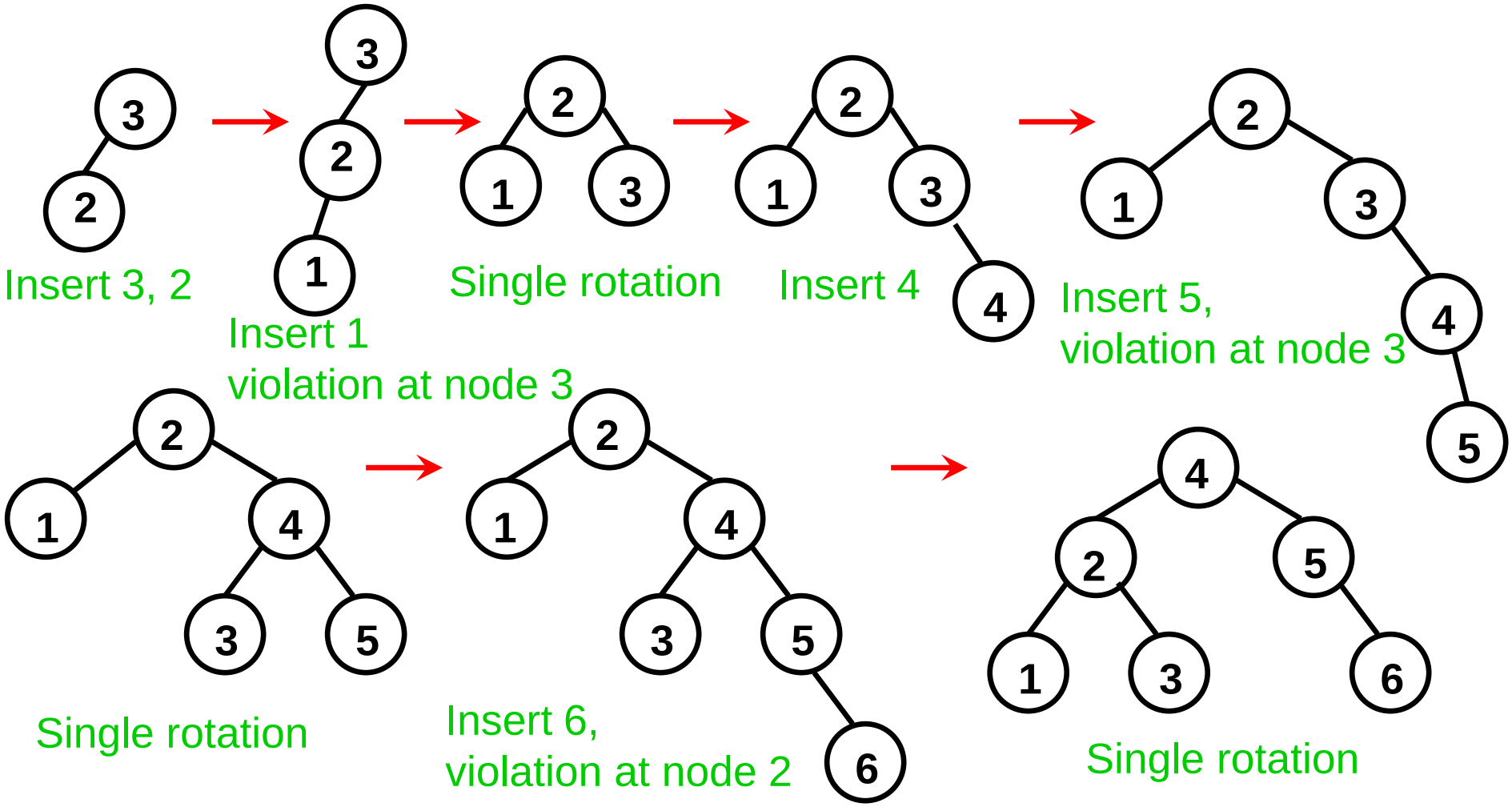


An insertion in subtree Z

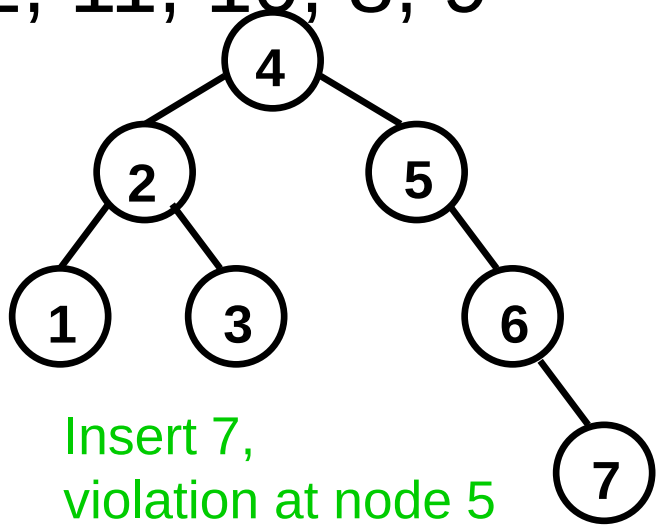
- Case 4 is a symmetric case to case 1
- Insertion takes O(Height of AVL Tree) time, Single rotation takes O(1) time

Single Rotation Example

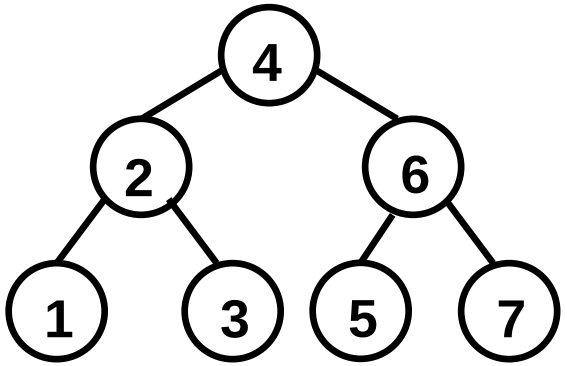
- Sequentially insert 3, 2, 1, 4, 5, 6 to an AVL Tree



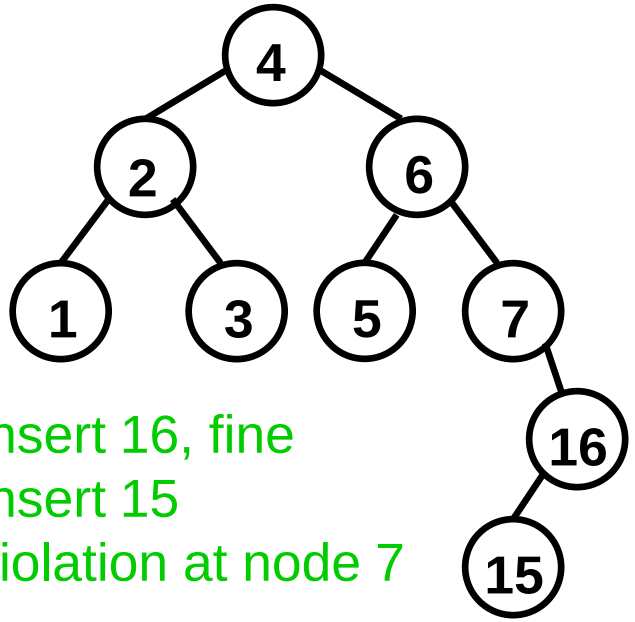
- If we continue to insert 7, 16, 15, 14, 13, 12, 11, 10, 8, 9



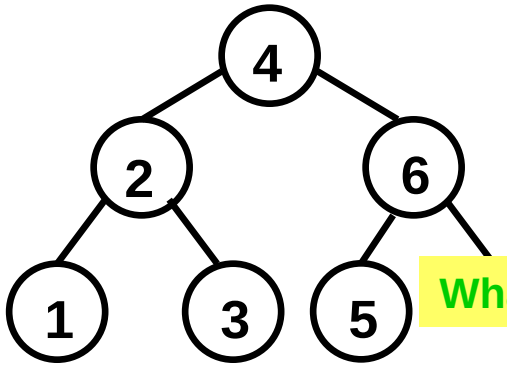
Insert 7,
violation at node 5



Single rotation



Insert 16, fine
Insert 15
violation at node 7

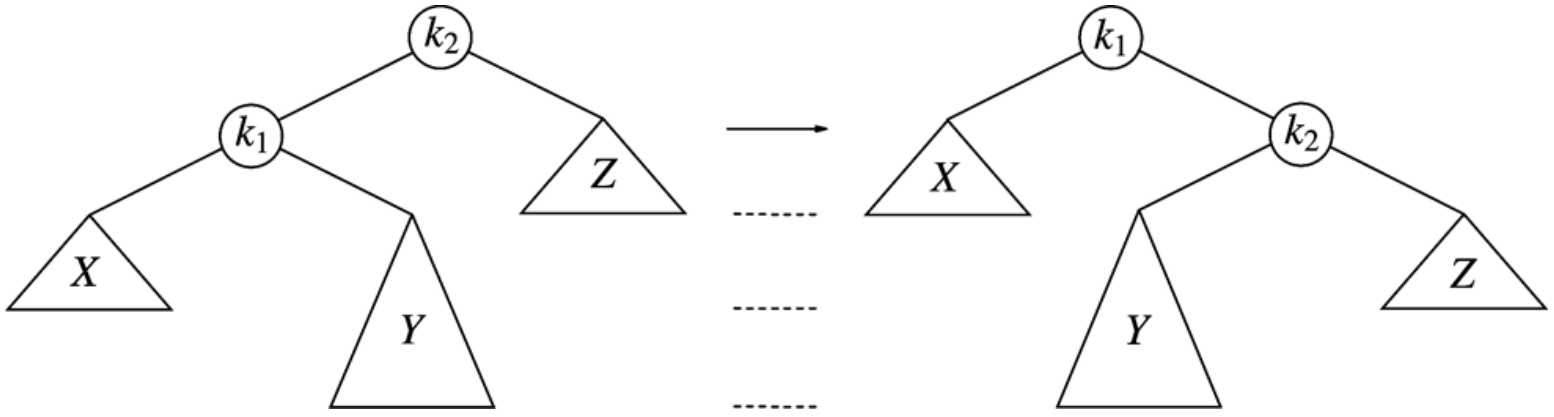


Single rotation

But….
Violation remains

What is the result?

Single Rotation Fails to fix Case 2&3



Case 2: violation in k2 because of insertion in subtree Y (Y is right sub-tree of left child of k2)

Single rotation result

- Single rotation fails to fix case 2&3
- Take case 2 as an example (case 3 is a symmetry to it)
  - The problem is subtree Y is too deep
  - Single rotation doesn't make it any less deep

Single Rotation Fails

- What shall we do?
- We need to rotate twice
  - Double Rotation