# Week 7. Problem set

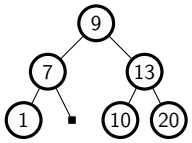
1. Insert the following keys into an initially empty AVL tree [GTG, §11.3]:

   $$2, 3, 29, 5, 11, 23, 13, 17, 19, 7$$

   For each insertion:

   - show the *state of the tree* **after** the insertion
   - specify the *number of rotations* performed during the insertion

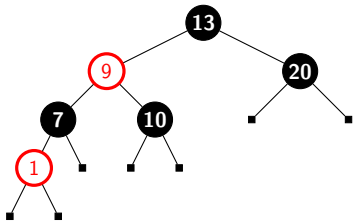
   Depict each tree using the array representation for binary trees. For example, consider the following AVL tree:

   The tree above must be depicted as the following array:

   | 9 | 7 | 13 | 1 | – | 10 | 20 |
   |---|---|---|---|---|---|---|

2. Perform the following operations starting with a reb-black tree [CLRS, §13] presented below:

   - insert $18, 5, 7$
   - insert $25, 20, 24$
   - insert $23, 22, 21$
   - delete $5, 9, 23$
   - delete $22, 31, 1$

   For each triple of operations (insertions or deletions):

   - show the *state of the tree* **after** the operations
   - specify the total *number of rotations* performed during the operations

   Depict each tree using the array representation for binary trees. You **must** color the keys for red nodes with **red** and keys for black nodes with **black** color. For example, the initial red-black tree presented above is depicted as the following array:

   | **13** | **9** | **20** | **7** | **10** | – | – | **1** | – | – | – | – | – | – | – |
   |---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

3. Compare randomly-built binary search trees and randomly chosen binary search trees for size $n = 4$:

   (a) Write down the number of distinct shapes for a binary search trees of size $n = 4$.

   (b) What is the average height of a randomly chosen binary search tree of size $n = 4$?

   (c) Consider $a < b < c < d$. For every permutation of $a, b, c, d$, write down an array representation of a corresponding binary search tree built from that permutation (by inserting keys in the given order).

   (d) What is the average height of a randomly built binary search tree of size $n = 4$?

   (e) Explain, in your own words, why we should not start with a complete binary tree of size $n$ (for any $n$) with "empty" nodes (i.e. each node does not have any key) and then populate it with given keys $k_1, k_2, \ldots, k_n$ to achieve the optimal height of the resulting binary search tree.

# References


[CLRS] Cormen, T.H., Leiserson, C.E., Rivest, R.L. and Stein, C., 2022. *Introduction to algorithms, Fourth Edition.* MIT press.

[GTG] M. T. Goodrich, R. Tamassia, and M. H. Goldwasser. *Data Structures and Algorithms in Java.* WILEY 2014.