Exercise Session 09

Exercise 1.
Recall the recursive algorithm Fib-Rec(n) which computes n-th Fibonacci number F(n) in time O(2n) by means of a naive implementation the Fibonacci recurrence.

Fib-Rec(n)
if n < 2
    return 0
else
    return Fib-Rec(n 1) + Fib-Rec(n 2)

(a) Implement a top-down memoized version of the above procedure called Memoized-Fib(n).

Initialise ∀ of memo[n] = 0 of size n

memo-Fib($memo, n$)
if memo[n] > 0
    return memo[n]
if n = 1 or n = 0
    return memo[n] = 1
else
    memo[n] = Fib(memo, n-1) + Fib(memo, n-2)
    return memo[n]

(b) Perform the asymptotic analysis of the worst-case running-time of Memoized-Fib(n).

$$T(n) = C_1 n + (C_2 + C_3 + C_5 + C_6 + C_7) \cdot (n - 1) = \Theta(n)$$

(c) Perform the asymptotic analysis of the space used by Memoized-Fib(n) It uses $\Theta(n)$ storage as we store n+1 elements in the memo array

Exercise 2.

T is an empty tree
CreateAndSort-Tree(A, T)
T.root = A[1]
for i = 2 to A.length
    Tree-Insert(A[i], T.root)
Inorder-Tree-Walk(T.root)

$$T(n) = c_1 + (n-1) \cdot (c_2 + c_3) + c_4$$
$$T(c) = 1 + n - 1 + (n-1) \cdot (n \cdot log(n)) + n$$
$$T(c) = n + n^2 log(n) + n = n^2 \cdot log(n)$$

Exercise 3.
Consider the binary search tree T depicted in Figure 2. Delete the node with *key = 10* from T by applying the procedure Tree-Delete(T, z) as described in CLRS.

Exercise 4.
Show the red-black trees that result after successively inserting the keys 41; 38; 31; 12; 19; 8 into an initially empty red-black tree.

Exercise 2.
Given a sequence of elements $A = [a_1, a_2, \ldots, a_n]$ we say that $[a_{i_1}, a_{i_2}, \ldots, a_{i_k}]$ is a subsequence of A if and only if $1 \leq i_1 < i_2 < \cdots < i_k \leq n$. For example, the following are valid subsequences of $A' = [4, 6, 6, 7, 6, 8, 1, 0, 9, 0, 15, 7, 10]$:

$$[6, 6, 6, 8, 15], \qquad [4, 6, 0, 15, 7, 10], \qquad [4, 6, 7, 10]\,.$$

Given a sequence of numbers $A[1\,..\,n]$, we are interested in finding an arbitrary nondecreasing subsequence of A of maximal length (a.k.a., longest nondecreasing subsequence).

(a) Argue why a brute-force enumeration of all subsequences leads to an exponential algorithm.

(b) Let $L(i)$ be the maximal length of a nondecreasing subsequence of $A[1\,..\,i]$ that contains the element $A[i]$. Describe a recurrence defining $L(i)$ for $i = 1\,..\,n$.

(c) Note that $L = \max_{i=1..n} L_i$ is the length of a longest nondecreasing subsequence of A. Describe a bottom-up dynamic programming procedure BottomUp-LNDS(A) that computes L.

(d) Describe a procedure Print-LNDS(A) that prints an arbitrary longest nondecreasing subsequence of A.

Remark: the longest subsequence may not be unique. For instance, the length of the longest non decreasing subsequence for A' is 7 as witnessed by the subsequences $[4, 6, 6, 7, 8, 9, 15]$ and $[4, 6, 6, 7, 8, 9, 10]$.

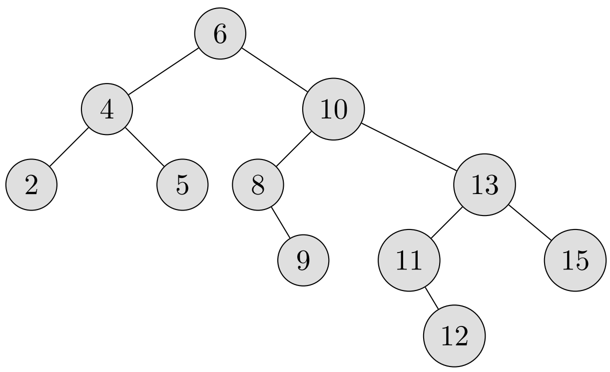

Figure 1: Binary Tree

Exercise 5.
Consider the red-black tree T depicted in Figure 3. Insert first a node with $key = 15$ in T, then delete the node with $key = 8$. Show all the intermediate transformations of the red-black tree with particular emphasis on the rotations.

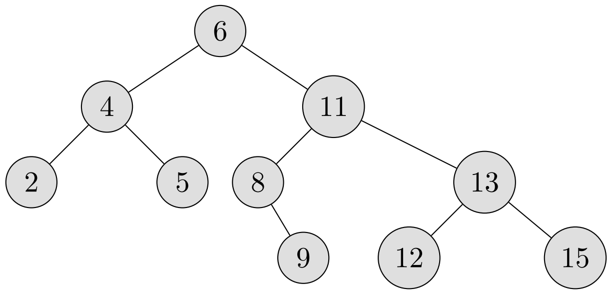

Figure 2: Binary Tree after deletion of 10

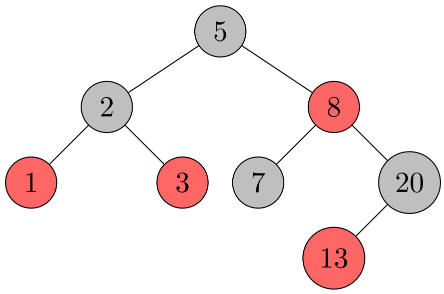

Figure 3: RB-Tree (NIL leaf nodes are omitted from the drawing)