Problem 1. Answer the following questions:

(a) Describe the reason for randomizing the Partition procedure used in the Randomized-Quicksort algorithm.

(b) Suppose we use RANDOMIZED-SELECT to find the maximum element of the array A = [3, 2, 9, 0, 7, 5, 4, 8, 6, 1]. Indicate which element would have to be the first chosen as pivot, such that the maximum element would be found immediately during the first call (before the algorithm makes any other recursive calls).

(c) What is the difference between the MAX-HEAP property and the binary search tree property?

(d) Illustrate the operation of Radix sort on the following list of 3-digit numbers.

732

390

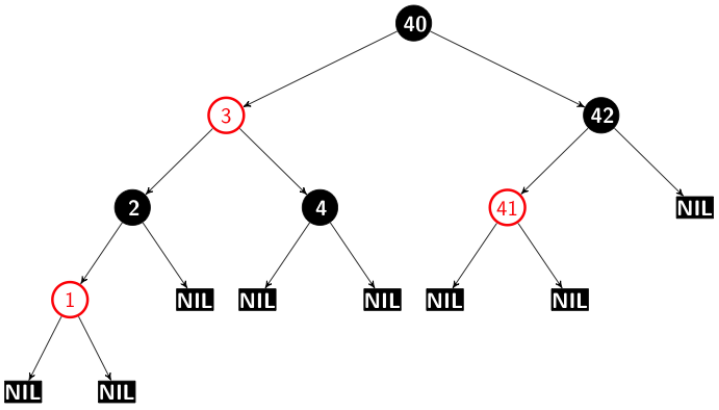
210

881

436

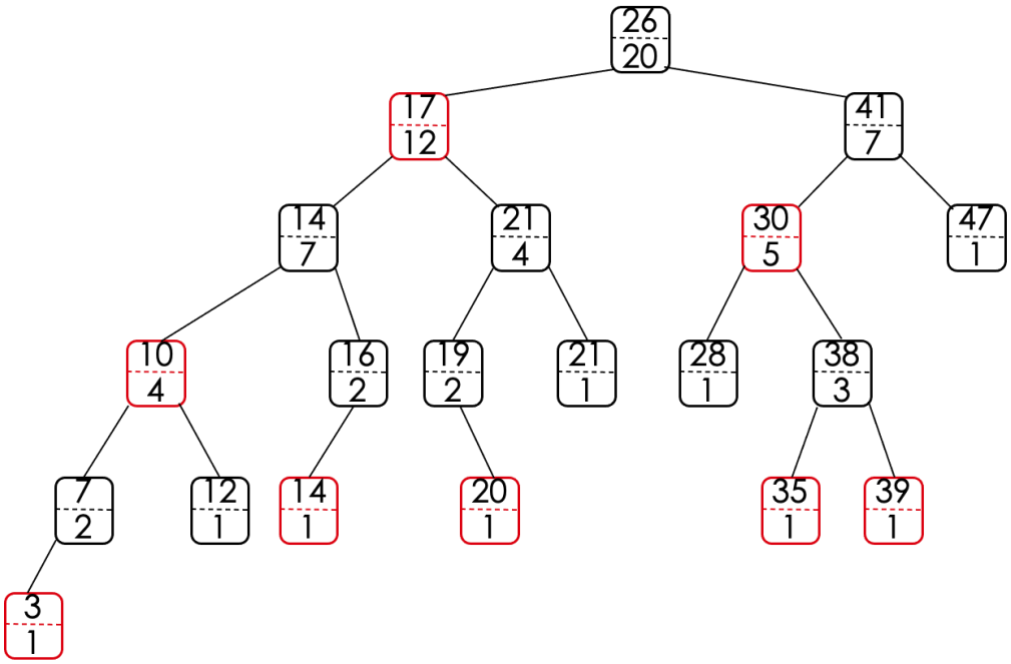
945

(e) Label each node in the following red-black tree with their black heights (black nodes have dark background).

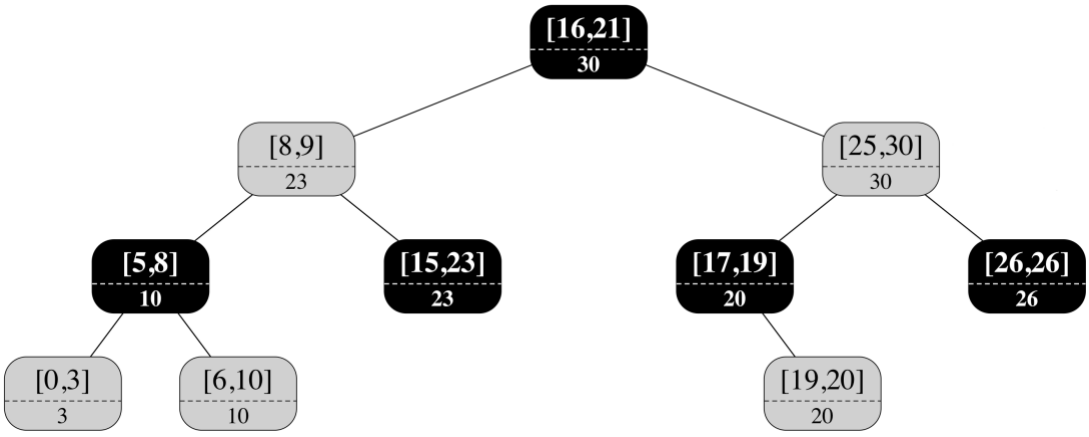


Problem 2.

(a) Show how OS-RANK(T, x) operates on the red-black tree shown in the figure below and the node x with $x.key = 47$.



(b) Highlight and indicate on the tree **all** the changes that would result from inserting the interval [14, 45] into the tree below (black nodes are darker in color).



(c) Describe what red-black tree property would be affected by inserting the interval [1, 45] into the interval tree from part (b).

Problem 3. For each of the following statements indicate only whether it is true or false by circling True or False.

(a) True False

Counting sort is a sort in place algorithm.

(b) True False

The following array constitutes a max-heap: A = [23, 17, 14, 6, 13, 10, 1, 5, 7, 12].

(c) True False

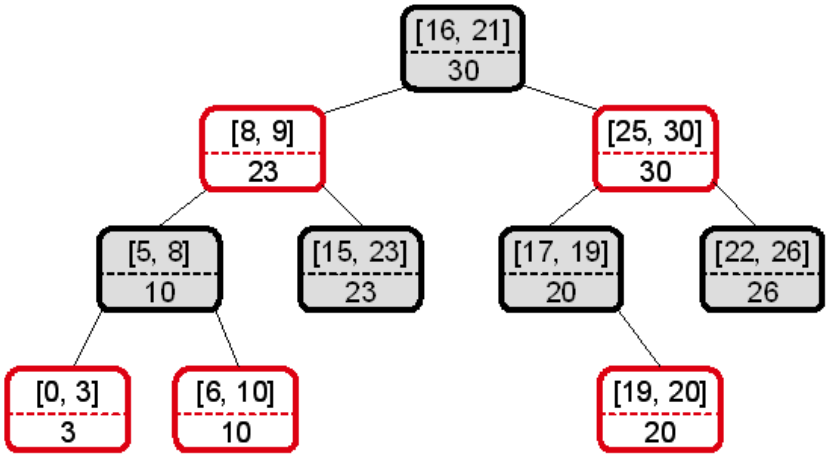
The min-heap property be used to print out the keys of an *n*-node heap in sorted order in *O(n)* time.

(d) True False

The black height of nodes in a red-black tree can be maintained as fields in the nodes of the tree without affecting the asymptotic performance of any of the red-black tree operations.

(e) True False

The following tree constitutes a valid interval-tree. (Black nodes have a gray background).



Problem 4. Write pseudocode for a **recursive** algorithm `RBT_CountB(T)` that takes as input a red-black tree and returns the number of black nodes in the tree (including NIL nodes). **Note:** the algorithm **cannot use any global variables** and **cannot take any other parameters** than a pointer to a node of the tree (initially the root).

Problem 5.

(a) Suppose you are playing a game where you are throwing a modified dice, which has only 3 values: two sides show a 1, two sides show a 2 and two sides show a 3. If the dice comes up with an odd value you gain $7 and if it comes up even you lose $2. What is the expected value of your earnings if playing this game?

(b) In a bin you have 15 cards of three different colors: 6 red, 5 yellow and 4 blue. You select a card at random and you win money based on the following rule: red ($8), yellow ($6), blue ($3). What is the expected value of your winnings?

Problem 6. Write pseudocode for a **divide-and-conquer** algorithm `ReverseArray` that reverses the order of elements of an array A of *n* integers. The algorithm should run in Theta(nlgn) in the average case. For instance, for an array `A = [4 7 2 9]` the output would be `[9 2 7 4]`.

Problem 7. Let *a*, *b* and *c*, be arbitrary nodes in subtrees α, β, and γ in the figure below. Show the tree that results when a left rotation is performed on node *A* and indicate how the **depths** of *a*, *b* and *c* change after this transformation.