Practice 3

Exercise levels:
(L1) Reproduce: Reproduce basic facts or check basic understanding.
(L2) Apply: Follow step-by-step instructions.
(L3) Reason: Show insight using a combination of different concepts.
(L4) Create: Prove a non-trivial statement or create an algorithm or data structure of which the objective is formally stated.

▶ Lecture 7 Binary Search trees

Binary Search Trees and Traversals

Exercise 1

(a) *(L2)* Insert items with the following keys (in the given order) into an initially empty binary search tree: 30, 40, 24, 58, 48, 26, 11, 13. Draw the tree after any two insertions.

(b) *(L3)* Choose a set of 7 distinct, positive, integer keys. Draw binary search trees of height 2, 5, and 6, excluding NIL-leaves, for your set of keys.

Exercise 2 For each of the algorithms PREORDERTREEWALK, INORDERTREEWALK, and POSTORDERTREE-WALK answer the following questions:

(a) *(L2)* Does the algorithm print the keys stored in a binary search tree in a sorted order? Argue why or give a counter-example.

(b) *(L3)* Does the algorithm print the keys of elements stored in a min-heap in a sorted order? Why or why not?

Exercise 3

(a) *(L3)* The *height* of a node v in a rooted tree T is defined as the number of edges on the longest simple downwards path from the node v to a leaf. Write an algorithm that calculates the heights of all nodes in a binary tree in $O(n)$ time. Do not forget to prove the correctness of your algorithm and to argue that it indeed runs in $O(n)$ time.

(b) *(L3)* The *depth* of a node v in a rooted tree T is defined as the number of edges on the simple path from the root of T to the node v. Write an algorithm that calculates the depths of all nodes in a binary tree and has running time $O(n)$. Do not forget to prove the correctness of your algorithm and to argue that it indeed runs in $O(n)$ time.

Balanced Search Trees

Exercise 4 A *Fibonacci tree* is a binary tree that is defined as follows: a Fibonacci tree F_0 of rank 0 is an empty tree; a Fibonacci tree F_1 of rank 1 is a tree that has 1 node; a Fibonacci tree F_k of rank $k \geq 2$ consists of a root node with F_{k-1} as its left subtree, and F_{k-2} as its right subtree.

(a) *(L2)* Draw the Fibonacci tree of rank 5.

(b) *(L3)* What is the height of a Fibonacci tree of rank k?

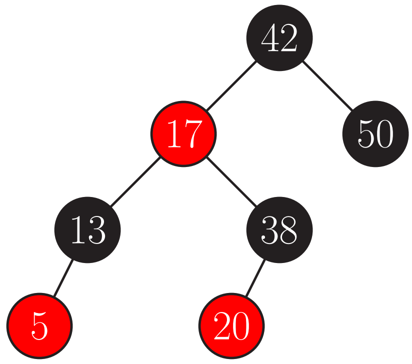
(c) *(L3)* Is every Fibonacci tree an AVL-tree? Why or why not?

(d) *(L4)* Is every Fibonacci tree a red-black tree? If so, give a scheme to color the nodes of a Fibonacci tree “red” and “black” so that it becomes a red-black tree. If not, give a counter-example.

▶ Lecture 8 Augmenting data structures

RB-tree operations

Exercise 5 *(L2)* Given the following red-black tree T:



Show all the changes to the tree when executing the following three operations in sequence:

(a) RB-INSERT$(T, 25)$

(b) RB-INSERT$(T, 18)$

(c) RB-DELETE$(T, 25)$

Augmenting Data Structures

Exercise 6 Different data structures have different advantages and disadvantages.

(a) *(L1)* For each of the data structures give the asymptotic worst-case runtimes for all the operations.

	Sorted array	**Min-heap**	**OS-tree**
INSERT(S, t)			
MINIMUM(S)			
EXTRACT-MIN(S)			
OS-SELECT(S, i)			

(b) *(L3)* Your application never inserts or deletes an element (static data) and you often need to find the element at a given rank. What data structure would be the best to use? Explain your answer.

(c) *(L3)* Your application has very many inserts (dynamic data) and you often need to find the element at a given rank. What data structure would be the best to use? Explain your answer.

(d) *(L3)* Your application has very many inserts and you often need to find the minimum and extract the minimum. What data structure would be the best to use? Explain your answer.

Exercise 7

(a) *(L2)* We augment every node x in a red-black tree T with a field h that stores the height of the node x. Can field h be maintained without affecting the asymptotic performance of any red-black tree operation? Show how, or argue why not.

(b) *(L2)* We augment every node x in a red-black tree T with a field d that stores the depth of the node x. Can field d be maintained without affecting the asymptotic performance of any red-black tree operation? Show how, or argue why not.

Exercise 8 *(L3)* Let $S = \{k_1, \dots, k_n\}$ be a set of distinct integers. Design a data structure for S that supports the following operations in $O(\log n)$ time: $\textsc{Insert}(S, k)$ which inserts the number k into S (you can assume that k is not contained in S yet), and $\textsc{TotalGreater}(S, a)$ which returns the sum of all keys in S that are larger than a, that is, $\sum_{s \in S,\ s > a} s$.
Argue why you can still insert elements in $O(\log n)$ time in your data structure.
Describe the algorithm $\textsc{TotalGreater}(S, a)$, argue its runtime, and prove its correctness.

Exercise 9 *(L3)* Consider an interval tree T containing n intervals with unique endpoints. Describe an $O(\log n)$ time algorithm that, given a number p, returns the interval from T that contains p and has the maximum high endpoint, or NIL if no interval contains p.

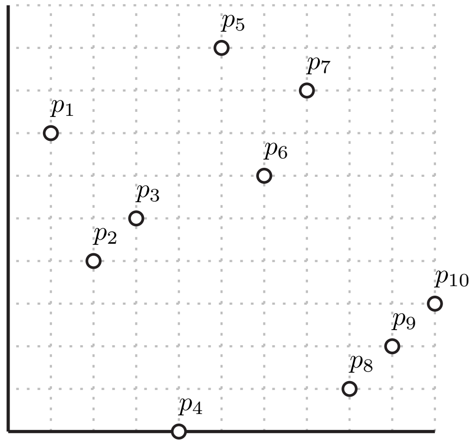
▶ Lecture 9 Range searching

Range Searching

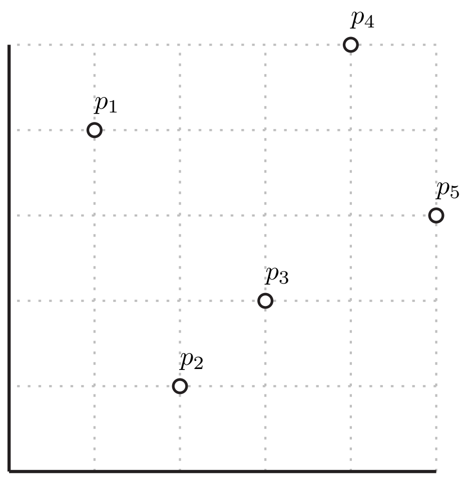
Exercise 10

(a) *(L2)* Build a KD-tree for point set (a). Use the lower median to split the set of points.

(b) *(L2)* Build a 2D range tree (including all associated structures) for point set (b). Use the lower median to split the set of points.



(a)



(b)

Exercise 11 *(L3)* Describe an algorithm that, given a (two-dimensional) KD-tree T, returns the point $p \in T$ with minimum x-coordinate. You may assume that all x-coordinates of the points in T are unique.

Exercise 12 *(L3)* Modify the 1DRangeQuery algorithm to report all the numbers stored in a binary search tree that are outside of a given query range. Prove the correctness of your algorithm, and analyze its running time.