Marburg University

Department of Computer Science
Prof. Dr. Sebastian Wild

Date: 2026-05-22
Version: 2026-05-22 17:09

Exercise Sheet 5 for Advanced Data Structures (Summer 2026)

Hand In: *Until 2026-05-29 18:00, on ILIAS.*

Problem 1

60 points

Recall that *splay trees* have the *amortized working set property*. Informally, it is cheaper to access elements that were recently accessed; formally, if we access an element x, and since the last time element x was accessed or inserted we accessed t distinct elements, then the access costs at most an amortized $O(\log t)$ time.

We now want a *worst case* version of this property for some dictionary data structure (*not necessarily a BST!*). In particular, design a dictionary that has the following characteristics:

- It should use $O(n)$ space to store n keys.
- It should be able to search for a key x in $O(\log t)$ time *in the worst case*, where t is the number of distinct keys accessed since the key x was last inserted or searched for.
- It should support insertions and deletions in $O(\log n)$ time worst case.

Partial credit will be given for incomplete solutions and sensible failed attempts if described convincingly.

Hint: Consider using multiple binary search trees of different sizes as *caches* for recently accessed keys.