Practice Exam for

Algorithms, Data Structures, and Data Abstraction WS 17/18

Wolfgang Mulzer, Katharina Klost

Time: at least 90 minutes

Aufgabe 1 Implementations of an ADT *3+3+4*

Consider the following specification of an abstract data type: Let $\mathcal{U}$ be a totally ordered universe. We would like to store subsets $S \subseteq U$, so that the following operations are possible:

- `insert`(x): Pre: None. Effect: $S \mapsto S \cup \{x\}$.
- `deleteMin`(): Pre: S is not empty. Effect: $S \mapsto S \setminus \{\min S\}$.
- `deleteMax`(): Pre:: S is not empty. Effect: $S \mapsto S \setminus \{\max S\}$.

You may assume that two elements from $\mathcal{U}$ can be compared in constant time. For each of the following data structures, describe briefly how to implement the operations `deleteMin` and `deleteMax` efficiently, and give good asymptotic upper bounds on the running times. If necessary, explain what additional assumptions need to be made.

(a) Sorted doubly linked list with pointers to the first and last element;

(b) AVL-tree; and

(c) binary min-heap.

Aufgabe 2 Hashing *2+4+4*

(a) Hashing with chaining gives *expected* running time $O(1)$ per operation. What does this mean? What can we say about the *worst-case* performance?

(b) Let T be a hash-table with chaining that has m slots and stores a set S with $2m$ entries. Prove the following: assuming that the hash function h behaves randomly, the expected time to find a random element x in the hash-table is $O(1)$. You may assume that $h(x)$ can be computed in $O(1)$ time.

(c) Let T be a hash-table with n slots in which we store a set S with n entries. Let h be a hash function.

What is a *collision* under h? Compute the *total* expected number of collisions in T, assuming that h behaves randomly.

Hint: Consider all possible pairs of entries in S, and use linearity of expectation.

Aufgabe 3 Miscellaneous *3+2+2+3*

(a) What is a cryptographic hash function? Describe two properties and one application.

(b) What is the difference between the static and the dynamic data type of a variable? Give a short example.

(c) Under which circumstances is a data structure with $O(\log n)$ *amortized* time per operation preferable to a data structure with $O(\log n)$ *worst-case* time per operation?

(d) What is an *Abstract Data Type*? Give two example from class and explain the general principle that forms the basis of the notion of an abstract data type.

Aufgabe 4 Skip-Lists *4+3+3*

(a) Show that the expected size of a randomly constructed skip-list with n elements is $O(n)$.

(b) Let L_1 and L_2 be two randomly constructed skip-lists, with element sets K_1 and K_2, respectively. Give an efficient algorithm to obtain a skip-list for the element set $K_1 \cup K_2$ from L_1 and L_2. Analyze the expected running time of your algorithm. (You may use the result from (a) without proof.)

(c) Suppose that in (b) we also know that for all $k_1 \in K_1$, $k_2 \in K_2$ we have $k_1 < k_2$. This means that every element in L_1 comes before every element in K_2. Describe an algorithm that merges L_1 and L_2 in expected time $O(\max\{\log |K_1|, \log |K_2|\})$. Prove the guarantee on the running time.
Hint: For $x \in (0, 1)$, we have $\sum_{i=1}^{\infty} i x^i = x/(1-x)^2$.