Algorithmen und Datenstrukturen
SS 2018
Blatt 13

Mario Löscher, Antonio Rodríguez-Sánchez,
Philipp Zech, Alexander Maringele

2018–06–19

Exercise 1 (3 Points): Maps

1. What is the basic idea of a map and where is it used? (2 Pts)
2. Name and describe two ways to handle collisions (2.5 Pts)
3. What are Hash- and compressions functions used for? (3 Pts)
4. Think about the ADT of maps. Name at least 6 of the functions (2.5 Pts)
5. You have a hash table with size of 7 which handles collisions by separate chaining strategy. As a hash function, use the given table from below. In order to calculate the position of the corresponding element, further use Key mod 7 to reduce the hash. Enter the string "turingaward"character wise and give the hash table after all insert operations where done. (5 Pts)

Value	Key	Value	Key
a	0	n	13
b	1	o	14
c	2	p	15
d	3	q	16
e	4	r	17
f	5	s	18
g	6	t	19
h	7	u	20
i	8	v	21
j	9	w	22
k	10	x	23
l	11	y	24
m	12	z	25

Exercise 2 (3 Points): Complexity

1. Estimate the running-time complexity of the program Algorithm 1 in big-O notation:
2. The number of operations executed by algorithms A and B is $8n \log n$ and $2n^3$, respectively. Determine the minimal value of n_0 such that A is better than B for $n \geq n_0$.
3. Show that $n \log n$ is $\Omega(n)$.
4. Explain why the plot of the function n^c is a straight line with slope c on a log-log scale.

Algorithm 1 Complexity example Function

```
Input: array $a$ of size $n$
sum = 0
for $i = 1$ to $n/2$ do
    for $j = 1$ to $n$ do
        for $k = 1$ to $n - (n - 2)$ do
            $sum \mathrel{+}= a[i] + a[j] - a[k]$
        end for
    end for
end for
```

Exercise 3 (3 Points): Priority Queues

1. List and describe the methods that characterize the priority queue ADT (4.5 Pts)
   (no need to give generic container methods).
2. The lecture slides contained an example of a min heap where the root is the minimum element. The following array represents a max heap (largest element is the root) stored in an array in the manner discussed in the lecture. Show the tree represented by this array. (3.5 Pts)

0	1	2	3	4	5
95	77	88	11	45	85

3. Show the contents of the array after 105 is inserted into the original heap using the algorithm for `insert` discussed in the lecture. (3.5 Pts)
4. Show the contents of the array after `removeMax` for getting the maximum element of the heap (analogous to `removeMin` for getting the minimum element as discussed in the slides) is executed **twice** on the **original** heap. (3.5 Pts)

Exercise 4 (4 Points): Recursion

1. What is a recursive function? (1.5 Pts)
2. Describe the terms Recursionstart and Recursionstep (2 Pts)
3. List and describe the four forms of recursion (4 Pts)
4. Using the given function rec, draw the resulting calling tree including the intermediate steps (3 Pts)
5. Rewrite rec to be tail recursive (3 Pts)
6. Describe the expected runtime of function rec (1.5 Pts)

```
int rec(int a) {
        if (a % 2 || a > 8) {
                return a;
        }

        return rec(a+1) + rec(a+2);
}
```

Exercise 5 (3 Points): Trees

1. List and describe the methods that characterize the Tree ADT. (5 Pts) (no need to give generic container methods)
2. Define the depth of a position in a tree. (5 Pts)
3. Write pseudo code to compute the height of a tree t. (5 Pts)

Exercise 6 (4 Points): Find a solution to a problem

Assume two given collections A, B with positive integers and a positive integer c. The goal is to answer if there are positive integers $a \in A$ and $b \in B$ such that $a \times b = c$.

1. State the obvious (brute force) algorithm to find a solution in pseudo code. (5 Pts)
2. What is the runtime complexity of this approach? Give reasons for your answer. (5 Pts)
3. Describe a more efficient algorithm using suitable abstract data types and data structures. (5 Pts)