University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
P. Schneider

Algorithms and Data Structures
Winter Term 2019/2020
Exercise Sheet 5

Remark: For this exercise, watch the eighth and ninth video lecture.

Exercise 1: Master Theorem for Recurrences

Use the *Master Theorem* for recurrences, to fill the following table. That is, in each cell write $\Theta\big(g(n)\big)$, such that $T(n) \in \Theta\big(g(n)\big)$ for the given parameters $a, b, f(n)$. Assume $T(1) \in \Theta(1)$. Additionally, in each cell note the case you used (1st, 2nd or 3rd by the order given in the lecture). We filled out one cell as an example.

$T(n) = aT(\frac{n}{b}) + f(n)$	$a = 16, b = 2$	$a = 1, b = 2$	$a = b = 3$
$f(n) = 1$	$\Theta(n^4)$, 1st		
$f(n) = n$			
$f(n) = n^4$			

Exercise 2: Peak Element

You are given an array $A[1 \dots n]$ of n integers and the goal is to find a peak element, which is defined as an element in A that is equal to or bigger than its direct neighbors in the array. Formally, $A[i]$ is a peak element if $A[i-1] \leq A[i] \geq A[i+1]$. To simplify the definition of peak elements on the rims of A, we introduce *sentinel-elements* $A[0] = A[n+1] = -\infty$.

(a) Give an algorithm with runtime $\mathcal{O}(\log n)$ which returns the position i of a peak element.

(b) Prove that your algorithm always returns a peak element, give a recurrence relation for the runtime and use it to prove the runtime $\mathcal{O}(\log n)$.

Exercise 3: Binary search

(a) Provide the pseudocode of an algorithm BinarySearch implementing the following informal algorithm description. The input is a *sorted* array $A[0..n-1]$ of keys and a search key k. If there is an index i with $A[i] = k$, the algorithm returns i, else false.

The algorithm first divides the array at some index m which is in the "middle". If $A[m] > k$ we start the algorithm recursively on the left sub-array. If $A[m] < k$ we start the algorithm recursively on the right sub-array. Else we have $A[m] = k$ and return m.

(b) Give a recurrence relation for the runtime of BinarySearch and show it has runtime $\mathcal{O}(\log n)$.

(c) For the data structure "Hierarchy of Arrays" of Exercise Sheet 4, describe an operation Search(k) that takes at most $\mathcal{O}\big((\log n)^2\big)$ time and returns the array number i of an array A_i and an index j such that $A_i[j] = k$, or false if such a pair i, j can not be found. Explain the runtime.