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

Algorithms and Data Structures
Summer Term 2019
Sample Solution Exercise Sheet 6

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		
$f(n) = n$			
$f(n) = n^4$			

Sample Solution

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

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 $\mathcal{O}(\log n)$ (measured in the number of read operations on the array) 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.

Sample Solution

(a) **Algorithm 1** `Peak-Element`(A, ℓ, r)

> **if** $\ell = r$ **then return** $A[\ell]$ ▷ base case
> $m \leftarrow \lceil \frac{\ell+r}{2} \rceil$
> **if** $A[m] \leq A[m+1]$ **then**
>   **return** `Peak-Element`$(A, m+1, r)$
> **else if** $A[m] \leq A[m-1]$ **then**
>   **return** `Peak-Element`$(A, \ell, m-1)$
> **else return** $A[m]$ ▷ peak element found

A call of `Peak-Element`$(A, 1, n)$ returns a peak element in A.

(b) We show the invariant that during each call of `Peak-Element`(A, ℓ, r), we have $A[\ell-1] \leq A[\ell]$ and $A[r] \geq A[r+1]$. Since $A[0], A[n+1] = -\infty$, this is obviously true for `Peak-Element`$(A, 1, n)$. During sub-calls of `Peak-Element`(A, ℓ, r) this condition is maintained by the If-conditions and the recursive calls and the appropriate sub-array. This implies that we have found a peak element when $\ell = r$ (at the latest, but we may find one earlier).

During every recursive step, the considered sub-array is at most half the size of the previous one, thus the algorithm terminates eventually. Additionally, in each recurse step we make at most one recursive sub-call. Furthermore, in each recursive step we read at most 5 array entries. Thus we have $T(n) \leq T(n/2) + 5$ (reads), which solves to $T(n) \in \mathcal{O}(\log n)$ using the Master Theorem.