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

Algorithm Theory
Exercise Sheet 2

Due: Tuesday, 2nd of November, 2021, 4 pm

Exercise 1: Computing the Median *(10 Points)*

Let A be an *unsorted* Array of *pairwise distinct* integers of length n. We want to compute the median of A, i.e., the element $m \in A$ that would be in the middle of A if we would sort A (we say the median is the smaller of the two "middle" elements in case A is of even length). We want to accomplish this *deterministically*[1] in time $O(n)$.

Remark: You can not assume that the size of integers in A is constant in n, thus simply sorting A is not possible in $O(n)$ time.

(a) We start with an algorithm that computes a value relatively close to the median. The first step is to partition the elements of A into $k := \lceil \frac{n}{5} \rceil$ consecutive sub-arrays (group) A_i ($i \in \{1, \dots, k\}$) of 5 elements each (the last group A_k may be smaller). Then compute the median m_i of each group A_i. Let m' be the median of $m_1, \dots, m_k$. Show that at least $\frac{n}{5}$ elements in A are smaller than or equal to m' *and* $\frac{n}{5}$ elements in A are larger than or equal to m'. *(3 Points)*

Hint: You may assume that n is divisible by 5.

(b) Give a divide and conquer algorithm to compute the j^{th} largest element of A in time $O(n)$ for some j (which can obviously also be used to compute the median). Argue why your algorithm is correct and why it has the desired running time. *(7 Points)*

Hint: Use part (a) as subroutine. Use that one can show part (a) with the value $\frac{3n}{10}$ instead of $\frac{n}{5}$.

Exercise 2: Fast Fourier Transformation (FFT) *(10 Points)*

Let $p(x) = 8x^7 + 7x^6 + 6x^5 + 5x^4 + 4x^3 + 3x^2 + 2x + 1$. We want to compute the discrete fourier transform $DFT_8(p)$ (where we define $DFT_8(p) := DFT_8(a)$ given that a is the vector of coefficients of p). More specifically, we want you to visualize the steps which the FFT-algorithm performs as follows.

(a) Illustrate the *divide* procedure of the algorithm. More precisely, for the i-th divide step, write down all the polynomials p_{ij} for $j \in \{0, \dots, 2^i - 1\}$ that you obtain from further dividing the polynomials from the previous divide step $i-1$ (we define $p_{00} := p$). *(3 Points)*

(b) Illustrate the *combine* procedure of the algorithm. That is, starting with the polynomials of smallest degree as base cases, compute the $DFT_N(p_{ij})$ bottom up with the recursive formula given in the lecture (where N is the smallest power of 2 such that $\deg(p_{ij}) < N$). *(7 Points)*

Remarks: The base case for a polynomial $p = a$ of degree 0 is $DFT_1(p) = DFT_1(a) = a$. It suffices to give the $p_{ij}(\omega)$ for all N^{th} roots of unity ω, from which $DFT_N(p_{ij})$ can be derived. Use $\sqrt{\cdot}$ instead of floating point numbers if possible (for instance $\omega_8^1 = \frac{i+1}{\sqrt{2}}$ and $\omega_8^3 = \frac{i-1}{\sqrt{2}}$).

[1] That is, the algorithm must always succeed within the claimed running time.