Algorithms: Assignment 5

Due date: October 5 (Thursday)

Problem 1 (3 points)
Consider the problem of finding the kth smallest element of an array $A[1..n]$, that is, the element that would occupy the kth position after sorting the array. For example, if the array is <6, 4, 8, 2, 10, 0> and $k = 3$, then the kth smallest element is 4, since it is the third element in the sorted array <0, 2, 4, 6, 8, 10>.

Write an algorithm for finding the kth smallest element of a given array. Its average-case complexity should be *significantly better* than the complexity of sorting. Thus, sorting the whole array and then returning the kth element is not an appropriate solution.

Problem 2 (3 points)
Consider a computer environment where the control flow of a program can split three ways after a single comparison $a_i : a_j$, according to whether $a_i < a_j$, $a_i = a_j$, or $a_i > a_j$. Argue that the number of these three-way comparisons required to sort an n-element array is $\Omega(n \lg n)$.

Problem 3 (4 points)
Suppose that $A[1..n]$ is an array of integer numbers, and some value k occurs at least $\lfloor n/2 \rfloor + 1$ times in this array. Write an efficient algorithm for finding this value and give the running time of your algorithm.

Problem 4 (bonus)
This problem is optional, and it allows you to get 2 bonus points toward your final grade for the course. You cannot submit this bonus problem after the deadline.

We consider an array $A[1..n]$ and define a segment sum from p to r, where $1 \leq p \leq r \leq n$, as follows:

$$sum(p, r) = \sum_{p \leq i \leq r} A[i].$$

In other words, it is the sum of all array elements in the segment $A[p..r]$. Note that the total number of distinct segments is $\frac{n(n+1)}{2}$. Write a *linear-time* (that is, $\Theta(n)$) algorithm that determines the maximum over all segment sums.