Institut für Informatik
Prof. Dr. Heiko Röglin
Dr. Melanie Schmidt

Randomized Algorithms and
Probabilistic Analysis
Summer 2016

Problem Set 4

Problem 1

Let $(\Omega, \mathbf{Pr})$ be a discrete probability space and let $X, Y : \Omega \to \mathbb{R}$ be discrete random variables. Assume that $\mathbf{Var}(X)$ and $\mathbf{Var}(Y)$ exist, let $a, b \in \mathbb{R}$ be two real values. Prove that:

1. $\mathbf{Var}(aX + b)$ exists and $\mathbf{Var}(aX + b) = a^2 \cdot \mathbf{Var}(X)$.
2. If X and Y are independent, then $\mathbf{Var}(X + Y)$ exists and

$$\mathbf{Var}(X + Y) = \mathbf{Var}(X) + \mathbf{Var}(Y).$$

Problem 2

Can the Markov inequality be improved? Show that for any $a > 1$, it is possible to define a random variable X_a that is non-negative, $\mathbf{E}(X_a)$ exists and

$$\mathbf{Pr}(X_a \geq a \cdot \mathbf{E}(X_a)) = 1/a$$

holds. What bound does Chebychev's inequality give for X_a?

Problem 3 (*)

We assume that we get an array $A = [x_1, \ldots, x_n]$ with n distinct numbers. Quick*Select* finds the kth smallest element in A, i. e. the element that would be at position k when A is sorted increasingly. QuickSelect can in particular compute the median.

It proceeds in a similar fashion as QuickSort by choosing a pivot element x, partitioning the array into the smaller elements, x itself and the larger elements and then recursing. We assume that `Partition(A,i,j,m)` stores $A[m]$ in x, then swaps elements in A in time $c \cdot (j - i + 1)$ for some constant c such that $A[i, \ldots, j]$ first contains all elements that are smaller than x, then x and then all elements that are larger than x. It returns the new position of x. Consider the following non-recursive realization of randomized QuickSelect:

`RQuickSelect`$(A = [x_1, \ldots, x_n],\ k \in \{1, \ldots, n\})$

1. **set** $i = 1$ **and** $j = n$
2. **while** $i \neq j$ **do**
3. Choose m uniformly at random from $\{i, \ldots, j\}$ (thus, $A[m]$ is pivot)
4. **set** $m' =$ `Partition(A,i,j,m)` (m' is the pivot's position in the sorted array)
5. **if** $(m' = k)$ **then set** $i = m'$ **and** $j = m'$
6. **if** $(m' < k)$ **then set** $i = m' + 1$
7. **if** $(m' > k)$ **then set** $j = m' - 1$

8. **return** $A[i]$

To analyze the expected running time of `RQuickselect(A,k)`, we partition the execution into *phases*: A phase consists of iterations of the loop, where the number of elements $j-i+1$ is some value n'. The first phase starts with the first iteration of the loop where $n' = n$. A phase ends when $j - i + 1$ was decreased to at most $(3/4)n'$, or when $j - i + 1 = 1$. Let X_ℓ be the number of iterations of the loop in phase ℓ. Analyze $\mathbf{E}(X_\ell)$ and use your result to bound the expected running time of `RQuickSelect`. For simplicity, assume that $(3/4)^\ell n$ is an integer for $\ell \in \{1, \ldots, \log_{(4/3)} n\}$.