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

Randomized Algorithms and
Probabilistic Analysis
Summer 2016

Problem Set 3

Problem 1

Assume we have a fair k-sided die that rolls each number $i \in \{1, \ldots, k\}$ with equal probability. Let X be the random variable for the number of one roll. Let Z be the random variable for the sum of two independent rolls. What are $E[X]$ and $E[Z]$?

Problem 2

1. A lively monkey types $26^5 \cdot 42 + 4$ letters (=499017796 letters) on a keyboard. We assume that the keyboard has only upper-case letters and that each of the 26 letter is chosen uniformly at random. What is the expected number of times that the word `RANDOM` appears?
2. We flip a fair coin $n + \log_2 n - 1$ times, assume that n is a power of two. We get a sequence $x_1, x_2, \ldots, x_{n+\log_2 n-1}$ with $x_i \in \{H, T\}$. We say that $x_i, \ldots, x_{i+\ell-1}$ is an *ℓ-sequence* if $x_i = x_{i+1} = \ldots = x_{i+\ell-1}$ (all heads or all tails). What is the expected number of ℓ-sequences for $\ell = 1 + \log_2 n$?

Problem 3

Hint for the following two tasks: Use that $\sum_{i=1}^{n} \frac{1}{i} = H_n = \Theta(\log n)$.

1. Assume that we have n images, and that n is a multiple of k. Each image shows a portrait of a person, and there are k different persons. There are n/k images of each person. We want to have a collection with exactly one (arbitrary) picture of each person and use the following randomized algorithm: We keep choosing a picture uniformly at random (with replacement) until we have pictures of k different people. If we already have a picture of a person, we discard the chosen picture, otherwise we add it to our picture collection. What is the expected numer of times that we choose a picture until we have our collection of k pictures of different persons?
2. We want to sort n distinct numbers that are stored in array A. We use *GuessSort*: We pick two indices $i, j \in \{1, \ldots, n\}$ uniformly at random from all possible pairs (i, j) with $i < j$. If $A[i] > a[j]$, we swap the elements, otherwise, we do nothing. What is the expected number of swaps that this algorithm does until the array is sorted?