Prof. Dr. Susanne Albers
Dr. Suzanne van der Ster
Dario Frascaria
Lehrstuhl für Theoretische Informatik
Institut für Informatik
Technische Universität München

Fall Semester
October 19, 2015

Randomized Algorithms

Exercise Sheet 1

Due: October 26, 2015

Exercise 1.1 (5 points)
Consider the following way to pick a random permutation π of the set of integers $A_n = \{1, 2, \ldots, n\}$:

1. Run Randomized Quicksort (RandQS)
2. Let T be the tree corresponding to the execution of Randomized Quicksort
3. Let π be induced by the level-order traversal of T (meaning that the nodes are visited in increasing order of level numbers and in a left-to-right order within each level)

Find an example which shows that π is not uniformly distributed over the space of all permutations of the elements in A_n.

Exercise 1.2 (5 points)
Given an unbiased coin and n elements, how can one generate a random permutation of these elements?

Exercise 1.3 (10 points)
Suppose you are given a biased coin which lands heads with some unknown probability p. How can we use this coin in order to generate an unbiased coin-flip, i.e., such that $P[\mathrm{H}] = P[\mathrm{T}] = 1/2$?
(H stands for heads and T stands for tails.)
Hint: Consider two consecutive flips of the biased coin.

What is the expected number of coin-flips needed to obtain the outcome H or T?

Exercise 1.4 (10 points) Consider the sorting algorithm *linear insertion sort* for sorting an array of n elements. It starts by comparing the first and second element; if they are not in the correct order, they are swapped. Then the third element is considered. It is compared to the second element and swapped if needed, then (if it was swapped) it is compared to the first element (and swapped if needed). Iteratively, the element in position k is handled by swapping it downward until the first k elements are in sorted order.

Determine the expected number of swaps needed with a linear insertion sort when the input is a random permutation of n distinct numbers.

Exercise 1.5 (10 points)
Suppose that at each step of the min-cut algorithm described in [MR], instead of choosing a random edge for contraction, two vertices are chosen at random and are merged into a single vertex. Show that there exist inputs for which the probability that the modified algorithm finds a min-cut is exponentially small.