Albert-Ludwigs-Universität, Inst. für Informatik
Prof. Dr. Fabian Kuhn
S. Daum, H. Ghodselahi, O. Saukh January 21, 2014

Algorithm Theory, Winter Term 2013/14
Problem Set 6

hand in by Tuesday, January 28, 2014

Exercise 1: Linear-Time Contention Resolution

In class, we looked at the following simple contention resolution problem. There are n processes that need to access a shared resource. Time is divided into time slots and in each time slot, a process i can access the resource if and only if i is the only process trying to access the resource. We have shown that if each process independently tries to access the resource with probability $1/n$ in each time slot, by time $O(n \log n)$, all processes can access the resource at least once with high probability. The goal of the exercise is to improve the algorithm and to get an $O(n)$ time algorithm under the following assumptions.

- As in the lecture, all the processes know n (the number of processes). In the algorithm of the lecture, this is needed because the probability $1/n$ for accessing the channel depends on n.
- If a process tries to access the resource in a time slot, the process afterwards knows whether the access was successful or not.

Give a randomized algorithm which guarantees that with probability at leat $1 - 1/n$, during the first $O(n)$ time slots, each of the n processes can access the channel once.

Hint: You can make use of the following fact. Consider a time interval consisting of at least $e^2 k$ time slots. During the time interval, there are at most k processes trying to access the channel and in each time slot, each of the at most k processes tries to access the channel with probability $1/k$. Then, with probability at least $1 - e^{-k}$, at the end of the interval, at most $k/2$ of the processes have not succeeded to access the channel.

Exercise 2: Running Time of the Contraction Algorithm

We have discussed the randomized contraction algorithm for the minimum cut problem in the lecture. When analyzing the algorithm, we have assumed that each contraction on a graph with n nodes can be done in time $O(n)$. Show that this is indeed possible. That is, give an appropriate (simple) data structure to store the current multi-graph G and an algorithm that contracts a uniformly random edge in time linear in the number of nodes of G.

Exercise 3: Randomized Minimum *s*-*t* Cuts?

Consider adapting the min-cut algorithm (Contraction Algorithm) to the problem of finding an s-t *min-cut* in an undirected graph. In this problem, we are given an undirected graph $G = (V, E)$ together with two distinguished nodes $s \in V$ and $t \in V$. An s-t cut is a partition $A \dot{\cup} B = V$ of V such that $s \in A$ and $t \in B$. The size of a cut (A, B) is the number of edges u, v for which $u \in A$ and $v \in B$; we seek an s-t cut of minimum size. As the algorithm proceeds, the vertex s may get merged into a new vertex as the result of an edge being contracted; we call this vertex the s-vertex (initially

s itself). Similarly, we have a t-vertex. As we run the contraction algorithm, we ensure that we never contract an edge between the s-vertex and the t-vertex (this guarantees that in the end, we get an s-t cut). Show that there are graphs (not multi-graphs) in which the probability that this algorithm finds a minimum s-t cut is exponentially small.