Albert-Ludwigs-Universität, Inst. für Informatik
Prof. Dr. Fabian Kuhn
H. Ghodselahi, Y. Maus
January 21, 2015

Algorithm Theory, Winter Term 2014/15
Problem Set 12

hand in (hard copied) by Thursday, 10:00, January 29, 2015, either before the lecture or in the box corresponding to your group in building no. 51.

Exercise 1: Randomized Minimum *s-t* Cuts? (4+1 points)

a) 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.

b) How large can the number of $s-t$ min-cuts in an instance be? Give an example which proves your claim.

Exercise 2: Load Balancing Approximation (4+1+1+1 points)

In class, we considered the following load balancing problem. There are n jobs and m machines; job i requires t_i time units to be executed. The objective is to assign each job to one of the m machines such that the makespan (the maximum total processing time of a single machine) is minimized.
We have seen that if the jobs are sorted such that $t_1 \geq t_2 \geq \cdots \geq t_n$, the following greedy algorithm has an approximation ratio of 3/2. The greedy algorithm goes through the jobs in the given order and always assigns a job to the machine with the least load. In this exercise, we want to understand this algorithm a bit better.
As in the lecture, assume that T is the makespan of the solution constructed by the described algorithm and assume that i is a machine with load T in the greedy solution. Further, assume that $\hat{n}$ is the last job that is scheduled on machine i (in class, we called this job j). Note that even without jobs $\hat{n}+1, \ldots, n$, the makespan of the algorithm is T and clearly the optimal makespan when only considering jobs $1, \ldots, \hat{n}$ cannot be larger than the optimal makespan for all the jobs.

(a) Show that if an optimal solution for jobs $1, \ldots, \hat{n}$ assigns at most 2 jobs to each machine, the algorithm computes an optimal solution (i.e., $T = T^*$).

(b) Show that therefore, either $t_{\hat{n}} \leq T^*/3$ or the greedy algorithm computes an optimal solution.

(c) Based on (a) and (b), conclude that the algorithm has approximation ratio at most 4/3.

(d) Try to find a bad input for the algorithm, where the ratio T/T^* between the makespan of the solution of the algorithm and the optimal makespan is at least a fixed constant larger than 1.
Hint: There is an example with $m = 2$ and $n = O(1)$.