Albert-Ludwigs-Universität, Inst. für Informatik
Prof. Dr. Fabian Kuhn
H. Ghodselahi, Y. Maus
February 5, 2015

Algorithm Theory, Winter Term 2014/15
Problem Set 14

This sheet will only be graded for students who have not yet passed the 50% threshold to be admitted for the exam. All points are additional points, i.e., they do not raise the threshold. Note that *online algorithms* and *parallel algorithms* are potential exam topics.
hand in (hard copied) by Thursday, 10:00, February 12, 2015, either before the lecture or in the box corresponding to your group in building no. 51.

Exercise 1: Sparse Subgraph (4* points)

Given a graph $G = (V, E)$ and an integer $k \geq 1$, we define a k-sparse edge set to be a subset $F \subseteq E$ of the edges such that the subgraph $H = (V, F)$ induced by the edges in F has maximum degree at most k. The size of a k-sparse edge set is the number of its edges. The maximum k-sparse subgraph problem (max-k-SSP) asks for a k-sparse edge set of largest possible size.

Assume that you are given the edges one by one in an online fashion. Give a deterministic online algorithm with competitive ratio 2 to solve the maximum k-sparse subgraph problem for $k = 2$ and argue why your algorithm has competitive ratio 2.

Exercise 2: Parallelization, Parentheses (1.5*+3*+1.5* points)

You are given a string S of size n consisting of opening and closing parentheses.
The expression S is supposed to be a string with *balanced parentheses* if each opening parenthesis has a corresponding closing one and the pairs of parentheses are properly nested.
For example, consider the expressions (()()) as a correctly balanced string of parentheses and ())(as an incorrect one.

a) First, provide a (sequential) linear-time algorithm to determine whether S is balanced.

b) Now devise a parallel algorithm to check if the string S is balanced.

Hint: *Similar as in the lecture (the whole lecture is given on Feb. 9. You could refer to the last year lecture) it might be useful to do parallelization with the help of a binary tree. The crucial point is to decide what information processors send up to the next level; of course there is no gain in parallelization if the whole string is sent up the tree.*

c) What are the total asymptotic work T_1 and the span T_∞ of your algorithm? Let the parallel algorithm be executed on p processors, what is the running time T_p of your parallel algorithm asymptotically?