TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

Department of Computer Science
Sebastian Wild

Issue Date: 14.06.2017
Version: 2017-06-14 13:12

8th Exercise sheet for Advanced Algorithmics, Summer 17

Hand In: *Until Wednesday, 21.06.2017, 12:00 am, hand-in box in 48-4 or via email.*

Problem 20

30 points

Modify the randomized min-cut algorithm from class as follows. Instead of randomly choosing an edge and contracting it, randomly choose a pair of vertices x and y and identify them into one vertex.

Prove that for some (infinite class of) graphs, the probability that this modified algorithm finds a minimal cut is (at most) exponentially small in the number of vertices n.

Problem 21

30 + 40 points

In this exercise, we consider efficient implementations of `contractionMinCut`.

You may assume that the multigraph is given in one of usual representations for weighted undirected graphs, i.e., either as *adjacency lists* with weights assigned to each successor node or as *adjacency matrix* where the entry $A[i, j] = c(i, j)$ for $\{i, j\} \in E$ and 0 otherwise.

We assume $c(i, j) \leq n^2$ for all edges.

a) Give a detailed implementation for `contractionMinCut` with running time in $\mathcal{O}(n^2)$.

b) Assume you would like to run the independent repetitions of `contractionMinCut` really independently (e.g., in parallel). For that to work smoothly, we require the original graph representation to remain *unchanged*.

   Give an algorithm to simulate one run of `contractionMinCut` where the original input is read-only, and you use only $\mathcal{O}(n)$ extra space. (Yep, simply copying the graph to scratch space is out of the question.)

   What is the best running time bound you can achieve?

   Hint: Draw an input graph and highlight all edges that have been contracted in one run of `contractionMinCut`. Does the result look familiar?