ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institute of Theoretical Computer Science
Mohsen Ghaffari, Angelika Steger, David Steurer, Emo Welzl, Peter Widmayer

Algorithms, Probability, and Computing — Exercises KW45 — HS17

General rules for solving exercises

- When handing in your solutions, please write your exercise group on the front sheet:

  Group A/B: Wed 13–15 CAB G 56

  Group C: Wed 16–18 CAB G 52

- This is a theory course, which means: if an exercise does not explicitly say "you do not need to prove your answer", then a formal proof is **always** required.

The following exercises will be discussed in the exercise class on November 9, 2016. Please hand in your solutions not later than November 8.

Exercise 1: Existence vs. Explicit Construction of Matchings

Suppose that we have an algorithm for testing the existence of a perfect matching in a given graph, with running time at most $T(n)$ for any n-vertex graph.

(a) Explain how repeated calls to the algorithm can be used to find a perfect matching if one exists. Estimate the running time of the resulting algorithm.

(b) How can the algorithm be used for finding a maximum matching in a given graph?

Exercise 2: Approximating the Minimum Cut

(Exercise 3.3 from the lecture notes)

You recall that the algorithm BasicMinCut computes a guess for the size of a minimum cut of a (multi)graph G by repeatedly contracting a uniformly random edge until there are only two vertices left and then returning the number of edges running between these two vertices.

As usual, denote the size of a minimum cut of G by $\mu(G)$. We have derived in the lecture that the number L_G which BasicMinCut outputs (on input G) is at least $\mu(G)$, and $\Pr[L_G = \mu(G)] = \Omega(n^{-2})$.

Consider the following slightly modified algorithm BasicMinCut': just like BasicMinCut, it repeatedly contracts a uniformly random edge until there are only two vertices left. But instead of just returning the number of edges between those two vertices in the very end, it returns the smallest degree of any vertex observed during the execution of the algorithm. That is if $G_0, G_1, G_2, \ldots, G_{n-2}$ is the sequence of graphs encountered, with $G_0 = G$ and $|V(G_{n-2})| = 2$, it returns

$$L_G := \min_{0 \leq i \leq n-2} \min_{v \in V(G_i)} \deg(v).$$

Prove that

(a) BasicMinCut' can be implemented so as to run in time $\mathcal{O}(n^2)$,

(b) $L_G \geq \mu(G)$ always holds,

(c) for any fixed $\alpha > 0$, the sucess probability

$$p_\alpha(n) := \min_{G \text{ a graph on } n \text{ vertices}} \Pr[L_G \leq (1+\alpha)\mu(G)]$$

satisfies the recurrence

$$p_\alpha(n) \geq \left(1 - \frac{2}{(1+\alpha)n}\right) p_\alpha(n-1).$$

Using (c), one can prove that for any fixed $\alpha > 0$, $p_\alpha(n) \in \Omega(n^{\frac{-2}{1+\alpha}})$, but this is just calculation and we do not ask you to do this here.

Exercise 3: Bounding the Number of Minimum Cuts

(Exercise 3.2 from the lecture notes)

Prove: a connected (multi)graph G on n vertices cannot have more than $\binom{n}{2}$ minimum cuts.

Exercise 4: Submodularity for Min-Cut

Let (V, E) be an undirected graph. For every subset of vertices S we associate the cut $C(S) := E(S, V \setminus S)$. Define a function $f : 2^V \to \mathbb{R}$ by $f(S) := |C(S)|$ for $S \subseteq V$. In other words $f(S)$ is the size of the cut $C(S)$.

(a) Show that the function f is *sub-modular*, meaning that

$$f(A \cap B) + f(A \cup B) \leq f(A) + f(B), \quad \forall A, B \subseteq V.$$

(b) Let $A, B \subset V$ be sets so that $C(A)$ and $C(B)$ are minimum cuts and suppose that $A \cap B \neq \emptyset$ and $A \cup B \neq V$. Prove that in this case, also $C(A \cap B)$ and $C(A \cup B)$ must be minimum cuts of G.

(c) For two vertices $s, t \in V$ a minimum s-t-cut is the smallest cardinality cut that disconnects s from t. Let $S \subset V$ be such that $C(S)$ is a minimum s-t-cut, $s \in S$ and $|S|$ is as small as possible with the first two properties. Show that S is unique.