Karl Bringmann and Marvin Künnemann **Winter 2017/18**

Exercises for Fine-Grained Complexity Theory

www.mpi-inf.mpg.de/departments/algorithms-complexity/teaching/winter17/fine-complexity/

Exercise Sheet 5 Due: **Tuesday, January 9, 2018**

Total points : 40 + 5 bonus points

You are allowed to collaborate on the exercise sheets, but you have to write down a solution on your own, ***using your own words.*** *Please indicate the names of your collaborators for each exercise you solve. Further, cite all external sources that you use (books, websites, research papers, etc.).*

You need to collect at least 50% of all points on exercise sheets.

Exercise 1 (*10 points*) Recall the **ZeroTriangle** problem from the last exercise sheet and the lecture:

ZeroTriangle: Given a weighted directed graph $G = (V, E, w)$, $|V| = n$ with edge weights $w : E \to \{-n^c, \dots, n^c\}$ (for some $c > 0$), determine if there are three vertices i, j, k such that $w(i, j) + w(j, k) + w(k, i) = 0$ holds.

Recall that in the lecture, we proved a tight reduction from **3SUM** to **ZeroTriangle**.

Now, prove that there is a *(non-tight)* fine-grained reduction in the other direction, from **Zero–Weight Triangle** to **3SUM**, i.e., prove

$$(\textbf{ZeroTriangle}, n^3) \leq_{fgr} (\textbf{3SUM}, n^{1.5}).$$

(Note that we prove only a lower bound of $n^{1.5-o(1)}$, instead of $n^{2-o(1)}$, for ***3SUM*** *here.)*

Exercise 2 (*10 points*) Recall the k–**Clique** problem from the lecture:

k–**Clique**: Given an undirected, unweighted graph G, determine whether G contains a k–clique (i.e., a set of k vertices which are pairwise adjacent).

Show that if $3 \mid k$, then k–**Clique** can be solved in time $O(n^{\frac{\omega k}{3}})$.
What running time can you obtain when $3 \nmid k$?

Note: This is the best running time known for this problem.

Exercise 3 (*10 points*) In the lecture we proved a subcubic reduction from **All–Pairs Negative Triangle** to **Negative Triangle**. Adapt this reduction to obtain a "combinatorial" subcubic reduction from **All–Pairs Triangle** to **Triangle**, i.e. prove

$$(\textbf{All–Pairs Triangle}, n^3) \leq_{fgr} (\textbf{Triangle}, n^3).$$

Why does this reduction not prove an $n^{\omega-o(1)}$ lower bound for **Triangle** under the BMM hypothesis?

Exercise 4 (*15 points*) Consider the following problem on directed acyclic graphs (DAGs):

All–Pairs Lowest Common Ancestor in DAGs (DAG–AP–LCA): Given a DAG $G = (V, E)$, determine for all vertices $i, j \in V$ *any* lowest common ancestor $v \in V$.
Here, a vertex $u \in V$ is a *common ancestor* of the vertices i and j if there is a path in G from u to i and from u to j (i.e., i and j are descendants of u). The vertex u is a *lowest common ancestor* of the vertices i and j if no descendant of u is a common ancestor of i and j.

Note that in DAGs (as opposed to trees), the vertices i and j might have more than one lowest common ancestor. (In this case, the problem just asks for *any* lowest common ancestor.)

a) *(3 points)* Show that **DAG–AP–LCA** has no algorithm running in time $O(n^{\omega-\varepsilon})$ (and no *"combinatorial"* algorithm running in time $O(n^{3-\varepsilon})$) for any $\varepsilon > 0$, unless the BMM hypothesis fails.

This lower bound suggests that to obtain strongly subcubic algorithms, we should use "non-combinatorial" tools, i.e., fast matrix multiplication. Indeed, we will see how to do this in the remaining parts of this exercise.

We will use the following intermediate problem:

BMM–MaxWitness: Given Boolean matrices $A = (a_{ij}), B = (b_{ij}) \in \{0, 1\}^{n\times n}$, determine for all $1 \leq i, j \leq n$ the maximum index k such that $a_{ik} = b_{kj} = 1$ if such a k exists (and $\perp$ otherwise).

b) *(3 points)* Show that the transitive closure of a graph G can be computed using $O(\log(n))$ Boolean matrix multiplications.

c) *(4 points)* Prove that there is a (subcubic) fine-grained reduction from **DAG–AP–LCA** to **BMM–MaxWitness**, i.e., prove

$$(\textbf{DAG–AP–LCA}, n^3) \leq_{fgr} (\textbf{BMM–MaxWitness}, n^3).$$

(Hint: Use a topological sort and part b).)

d) *(5 points)* Solve **BMM–MaxWitness** in subcubic time.

(Hint: Use (rectangular) matrix multiplication on suitably-sized subproblems.)

In total, this yields a strongly subcubic time algorithm for **DAG–AP–LCA**.