max planck institut informatik

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Karl Bringmann and Sebastian Krinninger **Summer 2016**

Exercises for Complexity Theory of Polynomial-Time Problems

https://www.mpi-inf.mpg.de/departments/algorithms-complexity/teaching/summer16/poly-complexity/

Exercise sheet 5 Due: **Monday, July 11, 2016**

Total points: 40

Either email an electronic version of your assignment submission to gjindal@mpi-inf.mpg.de or give it to Gorav in his office at Room 425, Building E1.3. If Gorav is not in his office then you can just slide your submission under the door of his office.

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*) Let A and B be two matrices having integer entries in $\{-M, -M+1, \ldots, M-1, M\}$. Prove that the Min-plus matrix product of A and B can be computed in time $O(M^2 \cdot n^\omega)$. Here ω is the exponent of matrix multiplication.

Now consider the following problem called **Maximum Weight Triangle**: Given a weighted undirected graph with integer edge weights in $\{-M, -M+1, \ldots, M-1, M\}$, find the weight of the maximum weight triangle. Generalize $O(M^2 \cdot n^\omega)$ time algorithm for Min-plus matrix product to **Maximum Weight Triangle**.

Note : These problems can also be solved in time $(M \cdot n^\omega \cdot \mathrm{poly}(\log M, \log n))$.

Exercise 2 (*9 points*) k-**Clique** is the problem of deciding whether a given undirected unweighted graph 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 $3 \nmid k$?

Hint *: Reduce it to detecting a triangle in a graph with $O(n^{\frac{k}{3}})$ vertices.*

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

Exercise 3 (*10 points*) **MaxCut** is the following problem.

Given an unweighted undirected graph $G = (V, E)$, find a partition of the vertices cutting as many edges as possible. More precisely, a cut is any subset U of the vertices V, and we define its value $C(U)$ as

$$\begin{aligned} C(U) &:= |\{\{u, v\} \in E \mid u \in U \text{ and } v \in V \setminus U\}| \\ &= \text{Number of edges in } G \text{ with one end point in } U \text{ and the other end point in } V \setminus U. \end{aligned}$$

In **MaxCut** the task is to compute the quantity $\max_{U \subseteq V} C(U)$. Note that this problem can be trivially solved in time $(\text{poly}(n) \cdot 2^n)$.

Show that it can be solved in time $O(\text{poly}(n) \cdot 2^{\frac{\omega n}{3}})$.

Hint *: Use similar ideas as in Exercise 2 to construct a weighted graph with* $O(2^{\frac{n}{3}})$ *vertices and edge weights bounded by* $\text{poly}(n)$ *and then use the Maximum Weight Triangle algorithm from Exercise 1 on this weighted graph. Even if you can not solve the Exercise 1, you can assume the algorithm of Exercise 1 is given to you to solve this exercise.*

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

Exercise 4 (*11 points*) Consider the following problem called **PairsSubstringMatch**.

PairsSubstringMatch : We are given n strings $S_1, S_2, \ldots, S_n$, each of length $O(n \cdot \text{poly} \log n)$. We are also given n^2 pairs of strings $(P_1, Q_1), (P_2, Q_2), \ldots, (P_{n^2}, Q_{n^2})$, where each P_i and Q_i is of length $O(\text{poly} \log n)$. We say that (P_i, Q_i) *matches* S_j if both P_i and Q_i are substrings of S_j. The task is to report all i where there exists a j such that (P_i, Q_i) matches S_j. (This j can be different for different i's.)

Show that a $O(n^{3-\epsilon})$ time combinatorial algorithm (for some $\epsilon > 0$) for **PairsSubstringMatch** implies a $O(n^{3-\delta})$ time combinatorial algorithm (for some $\delta > 0$) for **BMM** (Boolean Matrix Multiplication).

Note : A *substring* of a string S is another string T that occurs contiguously in S . This is not to be confused with *subsequence*, which is a generalization of substring. For example, "2345" is a substring of "1234567" but "1346" is only a subsequence of "1234567" and not a substring.

Hint *: First note that* ***BMM*** *can be a seen as a graph reachability problem on a three layered graph with each layer having n vertices. Now reduce this graph reachability problem to* ***PairsSubstringMatch****. For this, try to encode the neighbourhoods of the n middle layer vertices in the strings* $S_1, S_2, \ldots, S_n$*. The pairs* (P_i, Q_i) *should correspond to queries in the graph reachability problem.*