CS7800: Advanced Algorithms. Fall 2017
Homework 7

Instructor: Jonathan Ullman   TA: Albert Cheu

Due Friday, November 3rd at 11:59pm
(Email to `neu.cs7800@gmail.com`)

Homework Guidelines

Collaboration Policy. Collaboration on homework problems is permitted, however it will serve you well on the exams if you solve the problems by yourself. If you choose to collaborate, you may discuss the homework with at most 2 other students currently enrolled in the class. Finding answers to problems on the Web or from other outside sources (these include anyone not enrolled in the class) is strictly forbidden.

You must write up each problem solution by yourself without assistance, even if you collaborate with others to solve the problem. You must also identify your collaborators. If you did not work with anyone, indicate that on your submission. If asked, you must be able to explain your solution to the instructors.

Preparing and Submitting Solutions. You must type your solutions using LaTeX. Please use an 11-pt or larger font. Please submit both the source and PDF files using the naming conventions `lastname_hw7.tex` and `lastname_hw7.pdf`. Your name must be on the first page of the PDF.

Solution guidelines For problems that require you to provide an algorithm, you must give the following: (1) a precise description of the algorithm in English and, if helpful, pseudocode, (2) a proof of correctness, (3) an analysis of running time. You may use any facts from class in your analysis and you may use any algorithms from class as subroutines in your solution.

You should be as clear and concise as possible in your write-up of solutions. Communication of technical material is an important skill, so clarity is as important as correctness. A simple, direct analysis is worth more points than a convoluted one, both because it is simpler and less prone to error and because it is easier to read and understand. Points might be subtracted for solutions that are too long.

Problem 1 (Tournaments, 10 points)**.**

In order to determine the America's best jouster, and revive interest in the venerable sport of jousting, you have hosted a jousting tournament. The tournament was a round-robin, so every jouster competed against every other jouster, and each joust results in a unique winner. Unfortunately, the final results had *cycles* where jouster i beat jouster j, jouster j jouster k, but jouster k beat jouster i. As a result, you are having trouble figuring out who is truly America's best jouster. Your brilliant idea is to invent reasons to disqualify a small number of players to ensure that among the players who remain, there are no cycles.

We can formalize this problem as follows: there is a directed graph $G = (V, E)$ where the vertices V represent jousters, and there is an edge $(i \to j) \in E$ if jouster i beat jouster j. Because every jouster competed against every other jouster, resulting in a unique winner, for every pair (i, j), exactly one of $(i \to j)$ or $(j \to i)$ is in E. The problem is to take a directed graph such as this one and find the minimum size subset of vertices S such that if we remove all vertices in S, and all edges incident on S, then the remaining graph has no cycles. For example, if there are 4 jousters and the results of the $\binom{4}{2} = 6$ jousts are

$$1 \to 2 \quad 1 \to 4 \quad 2 \to 3 \quad 3 \to 1 \quad 4 \to 2 \quad 4 \to 3,$$

which contains the cycle $(3 \to 1 \to 2 \to 3)$, then disqualifying player 3 leaves just

$$1 \to 2 \quad 1 \to 4 \quad 4 \to 2,$$

which contains no cycles. We call this problem MinimumDisqualifications.

(a) Formulate MinimumDisqualifications as a decision problem.

(b) Prove that MinimumDisqualifications is in *NP*.

(c) Prove that MinimumDisqualifications is *NP*-hard.

Problem 2 (Maximum Saturated Flow, 10 points)**.**

We discussed at great length the fact that maximum flow problem can be solved in polynomial time. However, consider a slight modification of the maximum flow problem where we additionally require that every edge carries either no flow or is saturated. That is, for every edge e, either $f(e) = 0$ or $f(e) = c(e)$. We call such a flow *saturated*.

(a) Formulate MaxSaturatedFlow as a decision problem.

(b) Prove that MaxSaturatedFlow is in *NP*.

(c) Prove that MaxSaturatedFlow is *NP*-complete.[1]

[1] **Hint:** Try reducing from SubsetSum, which we will see in *NP*-complete on tuesday.