HW 11

Due on **Monday, May 4, 2020 at 11pm**

CS/ECE 374 B: Algorithms & Models of Computation, Spring 2020 Version: **1.1**

Submission instructions as in previous homeworks.

31 (100 PTS.) Circuit Switched Networks

Consider the following problem. You are managing a communication network, modeled by a directed graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$. There are c *users* who are interested in making use of this network. User i (for each $i = 1, 2, \ldots, c$) issues a *request* to reserve a specific path P_i in G on which to transmit data. You are interested in accepting as many of these path requests as possible, subject to the following restriction: if you accept both P_i and P_j, then P_i and P_j can not share any modes.

The Path Selection decision problem asks: Given a directed graph $\mathsf{G} = (V, E)$, a set of requests $P_1, \ldots, P_c$ (each of which must be a path in G), and a number k, is it possible to select at least k of the paths so that no two of the selected paths share any nodes?

Path Selection is an NP-COMPLETE problem. This can be verfied via a reduction from Independent Set or Set Packing or 3D Matching. You can find the proof with a simple Google search.

In this problem, however, assume that you were given an access to the oracle, denoted by **orac**, who is willing to tell you, given an instance of Path Selection whether or not it is feasible. Furthermore, the oracle is in a good mood, and willing to answer as many such questions as you might have.

In the optimization version of the problem, you have to compute the maximum number of paths that can be allocated (i.e., you are not given k).

Describe a polynomial time algorithm, that uses **orac**, and computes and outputs the optimal solution (i.e., the maximum number of paths that can be allocated without a conflict, and the paths themselves).

32 (100 PTS.) Clique-Cover

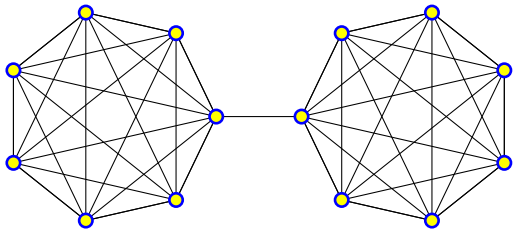
Given an undirected graph $G = (V, E)$, a partition of V into $V_1, V_2, \ldots, V_k$ is said to be a clique cover of size k if each V_i is a clique in G. CLIQUE-COVER is the following decision problem: given G and integer k, does G have a clique cover of size at most k?

32.A. (50 PTS.) Describe a polynomial-time reduction from CLIQUE-COVER to SAT. Does this prove that CLIQUE-COVER is NP-COMPLETE? For this part you just need to describe the reduction clearly and show that it is polynomial-time. No proof of correctness is necessary. *Hint:* Use variablex $x(u, i)$ to indicate that node u is in partition i.

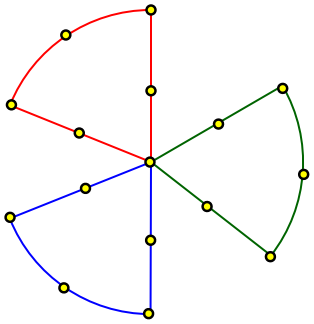
32.B. (50 PTS.) Prove that CLIQUE-COVER is NP-COMPLETE.

33 (100 PTS.) Shapes

33.A. (50 PTS.) For $k > 1$, a (k, k)-*dumbbell* is a graph formed by two disjoint complete graphs (cliques), each one on k vertices, plus an edge connecting the two (i.e., its a graph with $2k$ vertices). See figure for a $(7, 7)$-dumbbell. The DUMB problem is the following: given an undirected graph $\mathsf{G} = (V, E)$ and an integer k, does G contain a (k, k)-dumbbell as a subgraph? Prove that DUMB is NP-COMPLETE.



33.B. (50 PTS.) An undirected graph a ***3-blade-fan*** if it consists of three cycles C_1, C_2, and C_3 of k nodes each and they all share exactly one node. Hence, the graph has $3k - 2$ nodes. The figure below shows a ***3-blade-fan*** of 16 nodes.



Given an undirected graph G with n vertices and m edges and an integer k, the FAN problem asks whether or not there exists a subgraph of G which is a ***3-blade-fan***. Prove that FAN is NP-COMPLETE.

34 (100 PTS.) Undecidable

For each of the following languages, either prove that it is undecidable (by providing a detailed reduction from a known undecidable language), or describe an algorithm that decides this language – your description of the algorithm should be detailed and self contained. (Note, that you cannot use Rice Theorem in solving this problem. If you do not know what Rice's Theorem is, you can read more about it in the lecture notes but it is not required.)

34.A. (25 PTS.) $L_1 = \{\langle M \rangle \mid M \text{ is a Turing machine that halts on at least 2 input strings}\}$

34.B. (25 PTS.) $L_2 = \left\{ \langle M \rangle \;\middle|\; M \text{ is a Turing machine that } \begin{array}{r} \text{accepts at least one input string} \\ \text{and} \quad \text{rejects at least one input string} \end{array} \right\}$

34.C. (25 PTS.) $L_3 = \{\langle R, D \rangle \mid L(R) = L(D)$, where R is a regular expression, and D is a DFA$\}$.

34.D. (25 PTS.) $L_4 = \{\langle M, D \rangle \mid L(M) = L(D)$, where M is a Turing Machine, and D is a DFA$\}$.