HW 10

Due on **Wednesday, November 17, 2021 at 10am**

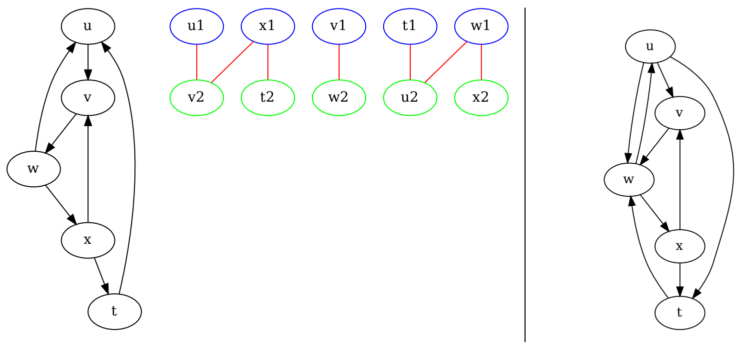
CS 473: Algorithms, Fall 2021 Version: **1.1**

28 (100 PTS.) Matchings - Applications

28.A. (50 PTS.) You are given a directed (non-bipartite) graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$. Let $\mathsf{V}_1 = \{u_1 \mid u \in \mathsf{V}\}$ and $\mathsf{V}_2 = \{u_2 \mid u \in \mathsf{V}\}$. Consider the bipartite graph $\mathsf{G}' = (\mathsf{V}_1 \cup \mathsf{V}_2, \{u_1 v_2 \mid (u, v) \in \mathsf{E}\})$.

Determining whether G admits a Hamiltonian cycle is NP-HARD. An easier task is to decompose G into smaller Hamiltonian cycles? Specifically, one need to compute vertex disjoint cycles $C_1, C_2, \ldots C_k$ of G, such that $C_1, C_2, \ldots C_k$ cover all the vertices in V.

Here are two examples of two such graphs, and the resulting graphs from the reduction:



Present an algorithm, as fast as possible, for computing such cycle decomposition using G'. What is the running time of your algorithm?

28.B. (50 PTS.) The hotel management is facing an issue of installing as few security cameras per floor as possible. Each floor comprises of n rooms and the rooms are connected to each other by a set of hallways H (each hallway connects a set of two rooms). The cameras can be installed only at every room. The goal is to install as few cameras as possible such that the hotel can keep surveillance of all hallways. The hotel insists on a very fast solution than an exact one.

Two cameras can communicate with each other if they can see each other. To avoid the case that thieves steal the cameras themselves, we require that any pair of installed cameras can communicate with each other (using potentially several hops between adjacent cameras).

Design an algorithm, as fast as possible, that gives a good approximation on the minimum number of cameras that needs to be installed that meets all the requirements above. To get any points, your approximation factor must be a constant (and as small as possible)! What is the running time of your algorithm?

29 (100 PTS.) Stable Matchings- Comparing Stable Matchings

In this problem, we discover some more remarkable properties of the set of stable matchings.

29.A. (25 PTS.) In class, we saw that a matching M returned by the Gale-Shapely algorithm is *optimal* for men, i.e., each man gets assigned a partner in M which is the at least as good as any partner he is assigned in any stable matching. Prove that M is *pessimal* for women, i.e., each woman gets assigned a partner in M which is at most as good as any partner she is assigned in any stable matching.

29.B. (25 PTS.) Given two stable matchings M and M', a person (man or woman), ***prefers*** M to M' if they like their partner in M more than their partner in M'. Show that the number of people that prefer M to M' equals the number of people who prefer M' to M.

29.C. (25 PTS.) Given two stable matchings M and M', show that the matching M'' where each man gets the better of his partners in M and M' is also a stable matching.

29.D. (25 PTS.) Given two stable matchings M and M', show that the matching M'' where each man gets the worse of his partners in M and M' is also a stable matching.

30 (100 PTS.) Perfect Matchings – Algebraic Methods

Consider a balanced bipartite graph $\mathsf{G} = (A \cup B, E)$ where $|A| = |B| = n$. Consider the $n \times n$ matrix A where $A_{ij} = x_{ij}$ if there exists the edge $(i, j) \in E$ and $A_{ij} = 0$ otherwise. Here X_{ij} is a unique variable for any i, j.

30.A. (25 PTS.) Show that the number of monomials in $Det(A)$ is equal to the total number of perfect matchings in G.

30.B. (10 PTS.) Let Q be a non-zero univariate polynomial of degree d. Let S be any finite set of integers. Let $s \in S$ be chosen uniformly at random. Show that $\mathbb{P}[Q(s) = 0] \leq d/|S|$.

30.C. (40 PTS.) Let Q be a non-zero n-variate polynomial of degree d. Let S be any finite set of integers. Choose $s_1, s_2, \ldots, s_n \in S$ independently and uniformly at random. Show that $\mathbb{P}[Q(s_1, s_2, \ldots, s_n) = 0] \leq d/|S|$. (Hint: Use induction on the number of variables)

30.D. (25 PTS.) Given a bipartite graph G, using the above, give a $O(n^\omega \log(n))$-time Monte-Carlo algorithm (i.e., an algorithm whose answer can be incorrect with low probability). to determine whether G admits a perfect matching or not, where $O(n^\omega)$ is time taken for matrix multiplication, which is also the time taken to find the determinant of a matrix (currently $\omega \leq 2.38$)