Math 154 - Discrete Math and Graph Theory
Homework 5
Due **Monday, March 3rd, 11:59pm**

Instructions: It is highly recommended (though not required) that you type your answers. It is your responsibility to make any handwriting clear and legible for grading. A LaTeX template for the homework is provided on Canvas. Please start each problem on a new page.

We will only be grading some of the problems below for correctness. However, because all of the concepts are important, we will not reveal which problems are being graded for correctness until after the assignment has been submitted. The remaining problems will be graded for completeness (i.e., does it look like there was a good-faith effort to solve the problem?).

Additional textbook questions for practice (not graded): 5.1, 5.15(a), 5.16

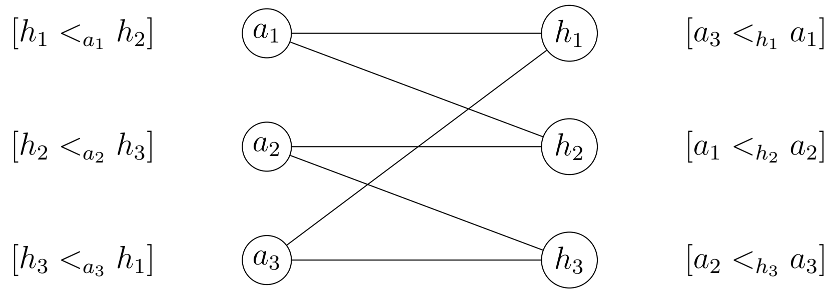
Problems:

1. **Stable matchings**

   In this problem, we will explore a variant of the bipartite matching problem where each vertex has a ranking of the vertices it could be paired with. One real world scenario in which this comes up is in matching medical students to hospital residencies.

   Let's instantiate this problem as follows: We first create a bipartite graph $G(A, H)$ where A is a set of vertices representing the medical applicants and H is a set of vertices representing the hospitals. Add an edge $\{a, h\}$ to the graph if the applicant $a \in A$ has the expertise to work at hospital $h \in H$.[1] Moreover, while each applicant $a \in A$ may have the expertise work at several hospitals (a set $H_a \subseteq H$), they have some opinion of how much they want to work at each hospital. That is, there is a total ordering $<_a$ on the hospitals H_a given by a's preferences: for any two hospitals $h, h' \in H_a$, either a prefers h' (i.e., $h <_a h'$) or a prefers h (i.e., $h' <_a h$). Similarly, each hospital $h \in H$ has a total ordering $<_h$ on the applicants $A_h \subseteq A$ they would most like to hire.

   Below is an example with applicants $A = \{a_1, a_2, a_3\}$ and hospitals $H = \{h_1, h_2, h_3\}$. We can see that applicant a_1 has the expertise to work at hospitals h_1 and h_2, but would prefer to work at h_2. However, hospital h_2 would prefer to hire a_2 over a_1.



[1]The textbook implicitly assumes that the bipartite graph is complete, whereas this definition does not.

The question we would like to answer is how to match applicants with hospitals in such a way that the applicants and hospitals are getting their highest possible preference. Formally, given a matching M in $G(A, H)$, we say that a hospital $h \in H$ *would accept* an applicant $a \in A$ if either: h is unmatched in M; or h prefers a to its currently matched applicant. We define the exact same condition for the applicants: an applicant *would accept* a hospital $h \in H$ if either: a is unmatched in M; or a prefers h to its currently matched hospital. A matching M is *stable* if there is no edge $\{a, h\}$ outside the matching such that: a would accept h; and h would accept a.

In other words, in a stable matching, there may be some other hospitals that an applicant would prefer to work at, but none of those hospitals would prefer to hire that applicant over their currently matched applicant.

Let's first see that stable matchings are meaningfully different from generic maximal matchings that don't respect the preference lists.

(a) Give an example of a graph $G(A, H)$ and preference lists for which a maximal matching is larger than any stable matching.

There are two questions that remain: Does every graph admit a stable matching? If so, can we find it efficiently? We will answer both questions together by giving an explicit algorithm to construct a stable matching.

Our plan will be to start with the empty matching $M = \emptyset$ and iteratively refine it. We will say that an applicant $a \in A$ *tolerates* the current matching M if either: a is unmatched; or a prefers their currently matched hospital over any other hospital that would accept a. Notice that if a tolerates a matching, they are not not necessarily matched with their top choice of hospital: a might prefer to work at some other hospital, but that hospital is already matched to a different applicant that the hospital prefers.

(b) Suppose there is a matching M for which the following two conditions hold: all applicants tolerate the matching; and for every applicant $a \notin V(M)$ not in the matching, there is no hospital $h \in H$ that would accept a. Prove that M is a stable matching.

Our goal now is to try to find some strategy of building a matching so that the matching will have the properties described in part (b). Here is a first attempt at such an iterative strategy: if there is an applicant $a \in A$ not yet in the matching and $h \in H$ would accept a, update our matching to include the edge $\{a, h\}$. It's possible that h is already matched with some other applicant $a' \neq a$; in that case, we simply discard the edge between a' and h.

(c) Given a matching M and any applicant $a \in A$, show that the number of hospitals that would accept a never increases when we update the matching M using the iterative strategy above.

If we repeatedly apply the iterative strategy above, we obtain an algorithm that terminates whenever it's impossible to find an applicant $a \in A$ not yet in the matching that a hospital $h \in H$ would accept. Notice that this satisfies the second condition from part (b).

To satisfy the first condition, we need to make sure that if all applicants tolerate the matching M, they also tolerate the updated matching. We employ the following strategy: if there are multiple hospitals that would accept $a \notin V(M)$, then amongst those hospitals, choose the hospital that a prefers the most.

(d) Let M be a matching tolerated by all $a \in A$. Using the above stategy, show that one iterative update to M results in a matching tolerated by all applicants $a \in A$.

Since we've described an algorithm that satisfies both conditions from part (b), we've now shown that our algorithm produces a stable matching. Let's analyze its runtime:

(e) Starting from the empty matching, what is an upper bound on the number of iterations of the algorithm before the algorithm terminates? Justify your answer. Write your answer in terms of $|A|$ and $|H|$.

(f) Try out the algorithm on the example graph earlier in the problem with $A = \{a_1, a_2, a_3\}$ and $H = \{h_1, h_2, h_3\}$. Starting with the empty matching, show each iterative refinement to arrive at a final stable matching. From the perspective of the applicants, what do you notice about their matches? From the perspective of the hospitals, what do you notice about their matches?

2. **One failure leads to two**

Let $G(A, B)$ be a bipartite graph on $2n$ vertices with $|A| = |B| = n$, and suppose that $G(A, B)$ does not have a perfect matching. Show that there are two distinct subsets of vertices in $G(A, B)$ that violate Hall's condition. That is, show that there are two subsets $X_1 \neq X_2$ such that for $i \in \{1, 2\}$:

- Subset is contained in one of the bipartitions: $X_i \subseteq A$ or $X_i \subseteq B$; and
- Size of the neighborhood is smaller than size of the set: $|X_i| > |N(X_i)|$.