CSCI 270 Problem Set 6 Solutions

Fall 2025

Problem 1. *For an input graph $G = (V, E)$, let T and T' be two different (possibly very different) spanning trees. Prove the following two exchange properties:*

1. *For every edge $e \in T \setminus T'$, there exists an edge $e' \in T' \setminus T$ such that $T \cup \{e'\} \setminus \{e\}$ is also a spanning tree.*

2. *For every edge $e' \in T' \setminus T$, there exists an edge $e \in T \setminus T'$ such that $T \cup \{e'\} \setminus \{e\}$ is also a spanning tree. (Notice that the quantifier order for e, e' is switched compared to the previous subproblem.)*

Solution.

1. Let $e \in T \setminus T'$ be arbitrary. Because T is a tree, $T \setminus \{e\}$ is a forest with exactly two trees, i.e., two connected components $S, \bar{S}$. Because T' is a spanning tree, it must contain at least one edge across this cut $(S, \bar{S})$. Let e' be an arbitrary such edge. Because S and $\bar{S}$ were disconnected in $T \setminus \{e\}$, there was no other edge across the cut; in particular, e' is not in T, so it is an edge in $T' \setminus T$. Adding e' to $T \setminus \{e\}$ reconnects the two components, and thus produces a spanning tree again.

2. Let $e' \in T' \setminus T$ be arbitrary. Consider $T \cup \{e'\}$. As $e' \notin T$ and T is connected, it must be that $T \cup \{e'\}$ contains a cycle C. Let $e \in C \setminus \{e'\}$ be any edge in $C \cap T$. Then $T \cup \{e'\} \setminus \{e\}$ is connected, as any $u \to v$ path in (V, T) can have the edge e replaced with $C \setminus \{e\}$ if necessary. Furthermore, it must be that $T \cup \{e'\} \setminus \{e\}$ is acyclic. If it were not, and contained a cycle C', then T must have also contained a cycle, by replacing the edge e' in C' with $C \setminus \{e'\}$ if necessary.

□

Problem 2. *In this problem, we will consider what happens in stable matching when we are guaranteed additional structure on the preferences.*

1. *Suppose that there is a "most popular" professor, whom all students rank first. Describe who this professor must be matched with in* every *stable matching, and explain why.*

2. *Use your insight from the previous part to analyze the case when all students have identical rankings. Prove that in this case, there is a* unique *stable matching, and describe it.*

3. *For the case of the previous problem, describe a simpler algorithm than Gale-Shapley which never has to break any assigned professors and students, and computes the unique stable matching. Prove that your algorithm outputs a stable matching. (Your proof may heavily draw on the previous subproblems.)*

Solution. 1. This professor p^* must be matched with their favorite student, s^*. Any other matching includes the unstable pair $\{(p^*, s), (p, s^*)\}$ for some other professor p and student s. Namely, s^* prefers professor p^* to p, by definition of p^* as most-popular, and p^* prefers s^* to their current student, by definition of s^*.

2. The unique stable matching is as follows: the top-ranked professor gets their first choice student. Among all the remaining students, the second-ranked professors gets their first choice. Among the remaining students after that, the third-ranked professor gets their first choice. And so on.

   We will prove this is the only stable matching by induction on n, the number of students & professors. The base case of $n = 1$ is immediate; there is a single stable matching given by matching the only professor with the only student. Now suppose the claim holds up to n students & professors, and let there be $n + 1$ many students & professors. Let $(p_1, \ldots, p_{n+1})$ be the professors sorted in descending order by the common preference order of students.

   By part (1.), in every stable matching, p_1 must be matched with their top-choice student. Thus, there is a unique student, say s_1, such that every stable matching is of the following form: p_1 pairs with s_1, and the other pairs form a stable matching between the remaining men and women.

   In the absence of p_1 and s_1, all $(n+1) - 1 = n$ students continue to share the same preference order $(p_2, p_3, \ldots, p_{n+1})$ over the remaining n professors. Invoking the inductive hypothesis to this instance reveals that it has a unique stable matching, obtained in the form we described above. Adding the match (p_1, s_1) to this unique stable matching gives the unique stable matching for the entire instance, completing the argument.

3. By the previous subproblem, the unique stable matching is the one in which the top-ranked professor gets their first choice student, the second-ranked professor gets their first choice among all the remaining students, and so on. This establishes correctness of the following algorithm, which simply implements the previous instructions.

```python
def soln(professors, students, popularity):
    # popularity[p] = agreed-upon popularity of professor p
    M = []
    available_students = set(students)
    professors = sorted(professors, key=lambda x: popularity[x],
                        reverse=True)
    for p in professors:
        favorite_student = None
        for curr_student in available_students:
            if (
                favorite_student is None
                or p.ranking(curr_student) > p.ranking(favorite_student)
            ):
                favorite_available_student = curr_student
        M.append((p, favorite_student))
        available_students.remove(favorite_student)
    return M
```

□

Problem 3. *We will look at another variant of "restricted" Stable Matching instances. Suppose that for each pair of professor p and student s, there is a happiness value $H(p, s)$ if the two were paired up. Importantly, both p and s would experience the* same *happiness $H(p, s)$ if they were matched. To avoid tie breaking issues, we assume that all the $H(p, s)$ values are different. Now, each professor/student will rank the other side by decreasing happiness they would get.*

1. *Show that for this type of input, there is again a simpler algorithm for finding a stable matching, which does not need to break any assigned pairs. Describe such an algorithm, and prove that it finds a stable matching.*

2. *Suppose that our goal were to maximize the total happiness of all people, i.e., the sum of all happinesses of the matches that the matching contains. Give and explain an example where neither Gale-Shapley nor your algorithm from the previous part finds the happiness-maximizing matching.*

Proof. 1. We give an algorithm to compute the stable matching and then show that it is the unique stable matching. Consider Algorithm 1:

Algorithm 1

```
1: Initialize $P$ to be the set of all $n$ professors, and $S$ the set of all $n$ students.
2: while $P \neq \emptyset$ do
3:     Let $(p, s)$ be the pair with $p \in P$ and $s \in S$ with greatest happiness $H(p, s)$. {Note that
       since the $H(p, s)$ are all different, the pair $(p, s)$ is unique.}
4:     Match $p$ to $s$.
5:     Let $P = p \setminus \{p\}$ and $S = S \setminus \{s\}$, i.e., remove $p, s$ from further consideration.
6: end while
```

We first prove that the algorithm produces a stable matching. Consider a pair (p, s) that was not matched by the algorithm; instead, the algorithm matched (p, s') and (p', s). Without loss of generality,[1] assume that (p, s') were matched by the algorithm before (p', s). When (p, s') were matched, the set S contained both s and s'. Since the algorithm chose the pair with greatest happiness, we know in particular that $H(p, s') > H(p, s)$. So p prefers s' over s, and as a result (p, s) cannot be an instability. Because this holds for all pairs (p, s), the algorithm outputs a stable matching.

[1]This phrase means that the other case — here, that (p', s) were matched first — is exactly identical in terms of how the proof works, and we're just arbitrarily "breaking ties" in our proof to avoid an unnecessary case distinction. Writing "without loss of generality" when you're focusing on one of two (or more) exactly identical cases in a proof is a common way to save duplicating work. Keep in mind that many people — including scientists writing papers — sometimes write "without loss of generality" wrongly, including in cases where they actually make a mistake.

Next, we prove the uniqueness of the stable matching. (This was not technically required of you, but the proof is fairly easy, so we'll share it anyway.) Let (p, s) be the *first* pair matched in the above algorithm. Then, $H(p, s)$ is larger than all other values; in particular, p prefers s over all other students, and s prefers p over all other professors. Similar to Problem 2(a), this means that every stable matching must contain the pair (p, s). Then, very similar to Problem 2(b), we can do an induction proof to show that the stable matching is unique.

2. Consider two professors and two students. The happiness is given as follows: $H(p_1, s_1) = 10, H(p_1, s_2) = 11, H(p_2, s_1) = 0, H(p_2, s_2) = 9$. There is a unique stable matching (since we proved uniqueness in Part (a)), and it must be matching p_1 with s_2, sacrificing p_2 and s_1 for a total happiness of 11. Instead, matching (p_1, s_1) and (p_2, s_2) would have resulted in total happiness 19.

□