HW-1

▼ **Reading Assignment: Kleinberg and Tardos, Chapter 1**

Done. Created notes

▼ **Solve Kleinberg and Tardos, Chapter 1, Exercise 1.**

True or false? In every instance of the Stable Matching Problem, there is a stable matching containing a pair (m, w) such that m is ranked first on the preference list of w and w is ranked first on the preference list of m.

The statement is **FALSE**.

Counter Example:

Consider the following preference lists for a set of men, M

m = [w, w', w'']

m' = [w', w, w'']

m'' = [w'', w', w]

For a set of women, W, the preference list of as follows:

w = [m', m, m'']

w' = [m'', m', m]

w'' = [m, m'', m']

Solving the above question using Gale-Shapley algorithm,

free_man = [m, m', m'']

- for m, (m, w) get engaged, free_man is updated to [m', m'']
- for m', (m', w') get engaged, free_man is updated to [m'']
- for m'', (m'', w'') get engaged, free_man is updated to []

Since, the free_man array is empty, the algorithm exits from the while loop and terminates.

It is evident from the counter example above that there exists an instance of the Stable Matching problem where no woman gets a partner who was ranked first in her preference list.

▼ **Solve Kleinberg and Tardos, Chapter 1, Exercise 2.**

True or false? Consider an instance of the Stable Matching Problem in which there exists a man m and a woman w such that m is ranked first on the preference list of w and w is ranked first on the preference list of m. Then in every stable matching S for this instance, the pair (m, w) belongs to S.

The statement is **TRUE.**

Gale-Shapley Algorithm Solution of a given example

Consider the following example which satisfy the aforementioned condition -

- Preference list of a set of men, M

  m = [w, w']

  m' = [w', w]

- Preference list of a set of women, W

  w = [m, m']

  w' = [m', m]

Following the Gale-Shapley algorithm for solving the above question (**MEN PROPOSE**) →

free_man = [m, m']

- for m, (m, w) get engaged, free_man is updated to [m']
- for m', (m', w') get engaged, free_man is updated to []

Since the free_man array is not empty, the while loop ends and the algorithm is terminated with S = [(m, w), (m', w')] when **MEN** propose.

Following the Gale-Shapley algorithm for solving the above question (**WOMEN PROPOSE**) →

free_woman = [w, w']

- for w, (m, w) get engaged, free_woman is updated to [w']
- for w', (m', w') get engaged, free_woman is updated to []

Since the free_woman array is not empty, the while loop ends and the algorithm is terminated with S = [(m, w), (m', w')] when **WOMEN** propose.

Proof by contradiction

The pairs (m, w') and (m', w) form a perfect matching. But, from the aforementioned statement,

- m prefers w over w'
- w prefers m over m'

so, (m, w) will always form a pair in the stable matching S. This is an instability that shows that this is a contradiction. Our initial assumption was wrong and hence (m, w) belongs to S.

Hence, every stable matching matching S, contains the pair (m, w) for the instance of execution of the Stable Matching Problem in which there exists a man m and a woman w such that m is ranked first on the preference list of w and w is ranked first on the preference list of m.

▼ **State True/False: An instance of the stable marriage problem has a unique stable matching if and only if the version of the Gale-Shapely algorithm where the male proposes and the version where the female proposes both yield the exact same matching**

The statement is **TRUE.**

Explanation for truth of CLAIM -

By the inherent correctness of the Gale-Shapley algorithm, we know that when men propose, the resulting stable matching set S is created in a way that favors the men (giving men their best valid partners) and when the women propose, then the Gale-Shapley algorithm favors the women (giving women their best valid partners) . It means that the party proposing always has the advantage. Now, if the stable matching is unique, it would mean that the preferences of both the men and women would be the same. For scenarios where the preference lists are the same, the resulting stable matching set would also be the same, hence the truth in the statement.

Explanation for truth of OPPOSITE CLAIM

OPPOSITE CLAIM - The instance of the Gale-Shapley algorithm where the male propose and where the females propose lead to the exact same matching, then the resulting stable matching set is unique)

Proof by Contradiction

Let us suppose that when men propose, the matching is M = [(m, w), (m', w')] and when women propose, the matching is M' = [(m, w'), (m', w)]. This means that for m, although w' is a valid partner, but w is the BEST VALID PARTNER (we know this from the explanation of the CLAIM). Since m prefers w, over w', (m, w') will be an instability and this is a contradiction. This further proves that if the matchings are the same irrespective of which party proposes, then the resulting stable matching is also going to be unique.

▼ **A stable roommate problem with 4 students a, b, c, d is defined as follows - Each student ranks the other three in strict order of preference. A matching is defined as the separation of the students into two disjoint pairs. A matching is stable if no two separated students prefer each other to their current roommates. Does a stable matching always exist? If yes, give a proof. Otherwise give an example roommate preference where no stable matching exists.**

A stable matching for the Stable Roommate Problem might not always exist. A counter example is shown below -

Consider the following preference order for all the roommates present.

a = (b, c, d)

b = (c, d, a)

c = (a, d, b)

d = (a, b, c)

The following steps happen if we follow the Gale-Shapley algorithm:

- a proposes to b, since b is not occupied, the proposal is accepted and (a, b) become roommates for now.

- c proposes to a, a is already paired up with b (which is a's highest preference). Next, c proposes to d and since d is not occupied, (c, d) become roommates
- so, the pairing for probable roommates now is (a, b) and (c, d).
- but, it is evident from the priority list of b that b has rated c higher than d. This would have resulted in (b, c) to be roommates. But again, for the remaining pair, i.e. (a, d), a prefers c over d and this causes an instability in every pairing option that is available.

The above counter example proves that a stable matching does not always exist for the Stable Roommate Problem.

▼ **Solve Kleinberg and Tardos, Chapter 1, Exercise 3.**

3. There are many other settings in which we can ask questions related to some type of "stability" principle. Here's one, involving competition between two enterprises.

Suppose we have two television networks, whom we'll call $\mathcal{A}$ and $\mathcal{B}$. There are n prime-time programming slots, and each network has n TV shows. Each network wants to devise a *schedule*—an assignment of each show to a distinct slot—so as to attract as much market share as possible.

Here is the way we determine how well the two networks perform relative to each other, given their schedules. Each show has a fixed *rating*, which is based on the number of people who watched it last year; we'll assume that no two shows have exactly the same rating. A network *wins* a given time slot if the show that it schedules for the time slot has a larger rating than the show the other network schedules for that time slot. The goal of each network is to win as many time slots as possible.

Suppose in the opening week of the fall season, Network $\mathcal{A}$ reveals a schedule S and Network $\mathcal{B}$ reveals a schedule T. On the basis of this pair of schedules, each network wins certain time slots, according to the rule above. We'll say that the pair of schedules (S, T) is *stable* if neither network can unilaterally change its own schedule and win more time slots. That is, there is no schedule S' such that Network $\mathcal{A}$ wins more slots with the pair (S', T) than it did with the pair (S, T); and symmetrically, there is no schedule T' such that Network $\mathcal{B}$ wins more slots with the pair (S, T') than it did with the pair (S, T).

The analogue of Gale and Shapley's question for this kind of stability is the following: For every set of TV shows and ratings, is there always a stable pair of schedules? Resolve this question by doing one of the following two things:

(a) give an algorithm that, for any set of TV shows and associated ratings, produces a stable pair of schedules; or

(b) give an example of a set of TV shows and associated ratings for which there is no stable pair of schedules.

Let us consider that Network A reveals a schedule S wherein it is going to air 2 shows (a1, a2) which have the rating of (2, 4) out of a 5 star rating system.

Similarly, Network B reveals a schedule T, where in it is going to air 2 shows (b1, b2) which have the rating of (1, 3) out of a 5 star rating system.

Each network tries to win more slots and hence, for Network A to win, a1(2) is compared with b1(1) → hence, a1(2) wins and Network A wins one slot to telecast show 'a' with rank 2. Now, Network A is looking for ways to grab one more slot timing. But a1(2) does not allow for A to win a slot as its rated less than b2(3). So Network A will want to switch the shows in its schedule, doing which it will win 2 slot timings as → a2(4) paired with b1(1) → A wins slot due to larger rating of a2.

In the same way, for Network B to win, if b1(2) is paired with a1(3), then Network B would switch the shows in its schedule so that it is able to fetch at least 1 slot.

In terms of Gale-Shapley algorithm,

a1 = [2]

a2 = [4]

and

b1 = [1]

b2 = [3]

b1 is paired with a1 i.e. (a1, b1) does not get any slot. Hence, swapping the shows for Network B, leads to (a1, b2) and network B wins a slot. (a1, b2) is an instability.

Symmetrically, if there is a pair (a1, b2) then network A does not win a slot, so it switches the shows in its schedule to make a pair (a2, b2) to win another slot. (a2, b2) is also an instability)

This proves that **no stable pair of schedules are present** for the given set of TV shows and associated ratings.

▼ **Solve Kleinberg and Tardos, Chapter 1, Exercise 4.**

4. Gale and Shapley published their paper on the Stable Matching Problem in 1962; but a version of their algorithm had already been in use for ten years by the National Resident Matching Program, for the problem of assigning medical residents to hospitals.

Basically, the situation was the following. There were m hospitals, each with a certain number of available positions for hiring residents. There were n medical students graduating in a given year, each interested in joining one of the hospitals. Each hospital had a ranking of the students in order of preference, and each student had a ranking of the hospitals in order of preference. We will assume that there were more students graduating than there were slots available in the m hospitals.

The interest, naturally, was in finding a way of assigning each student to at most one hospital, in such a way that all available positions in all hospitals were filled. (Since we are assuming a surplus of students, there would be some students who do not get assigned to any hospital.)

We say that an assignment of students to hospitals is *stable* if neither of the following situations arises.

- First type of instability: There are students s and s', and a hospital h, so that
  - s is assigned to h, and
  - s' is assigned to no hospital, and
  - h prefers s' to s.
- Second type of instability: There are students s and s', and hospitals h and h', so that
  - s is assigned to h, and
  - s' is assigned to h', and
  - h prefers s' to s, and
  - s' prefers h to h'.

So we basically have the Stable Matching Problem, except that (i) hospitals generally want more than one resident, and (ii) there is a surplus of medical students.

Show that there is always a stable assignment of students to hospitals, and give an algorithm to find one.

Since this can be considered as a variation to the Stable Matching Problem, we will be trying to modify the existing stable matching Algorithm to fit our needs.

```
Initially all h ∈ H and s ∈ S are free where H = Hospitals and S = Students
While there is a hospital h with atleast 1 slot and hasn't extended an admission to every student
  Choose such a hospital h with atleast one slot
  Let s be the highest-ranked student in h's preference list to whom h has not yet extended admission to
  If s is free then
    (h, s) become a pair and the number of slots available in hospital h reduce by 1
```

```
  Else s is currently admitted to h'
    If s prefers h' to h then the number of slots in the hospital h remain the same
    Else s prefers h to h' (h, s) become a pair and h' becomes free whereas the number of free slots in the hospital h reduce by 1
  Endif
Endif
Endwhile
Return the association A of hospital-student pairs for all the slots that were available in each hospital h.
```

All points valid in the Stable Matching algorithm apply here as well -

- once a student s, receives an offer from hospital h, that student is never without an offer during the entire execution of the algorithm.
- the quality of students that the hospital extends an offer to keeps getting worse.
- the student keeps getting better and better hospital as the algorithm progresses.

Time complexity analysis - this algorithm takes O(number of slots available) * O(number of hospitals available) time for the execution of the entire algorithm. So, if

> 💡 m = number of slots available
> n = number of hospitals available with each slot
>
> then O(m * n) is the total time complexity for the entire algorithm

To prove that the association A is a stable one, let us consider that there exists an instability in the algorithm and then try to contradict the same -

After running the modified Stable Matching algorithm, we get the association A, where one slot in H is associated with a student in S i.e. (h, s)

Instability 1 → Assume s is assigned to h, s' is assigned to no hospital and h prefers s' over s

s is assigned to h because h extended an offer to s. s' was not assigned to any hospital because no offer was extended to s'. If h would have preferred s' over s, then s' would have been assigned to h and (h, s') would have been a pair, which is not the case. This is a contradiction as (h, s) is a pair. Hence, out assumption is flawed.

Instability 2 → Assume there is an assignment (h, s) and another assignment (h', s'). Now, h prefers s', over s & h' prefers s, over s'

- in a stable setting, (h, s) and (h', s') are the assignments. if h preferred s', over s, then h would have extended an offer to s'. Since the final assignment (h, s) reveals that no such pairing has been achieved, there is a contradiction.
- it is also possible that s' rejected the offer by h because s' was already assigned to another hospital h'' which it preferred over h. It is possible that h'' = h' or h'' is a different hospital altogether but in each case, we arrive at a contradiction.

This means our assumption of instabilities existing is wrong and hence, the assignment as a result of the modified Gale-Shapley algorithm is always stable.

▼ **N men and N women were participating in a stable matching process in a small town named Walnut Grove. A stable matching was found after the matching process finished and everyone got engaged. However, a man named Almazo Wilder, who is engaged with a woman named Nelly Oleson, suddenly changes his mind by preferring another woman named Laura Ingles, who was originally ranked right below Nelly in his preference list, therefore Laura and Nelly swapped their positions in Almanzo's preference list. Your job now is to find a new matching for all of these people and to take into account the new preference of Almanzo, but you don't want to run the whole process from the beginning again, and want to take advantage of the results you currently have from the previous matching. Describe your algorithm for this problem. Assume that no woman gets offended if she got refused and then gets proposed by the same person again**

According to the question above, Almanzo proposes to Laura and rejects Nelly. One of the following 2 things might happen (Almanzo = A, Nelly = N, and Laura = L)

1. L rejects A as she prefers her current partner over A. A goes back to N and N accepts him and (A, N) get engaged again. In this case, we can make use of the previous matchings that were the output of the previous run for Stable

Matching algorithm.

2. L accepts A's proposal, there by rejected her current partner, lets say, A'. Now, A' may not go directly to N.
    a. Consider A's preference list = [w1, w2, w3, w4, ... , N]. In this case, A' would want to propose to all these other women first before proposing to N.
    b. Another case that can arise is that A' preference list has N above other possible partners, in which case, A' proposes to Nelly and (A', N) get engaged.
    c. Also, N becomes single when A leaves N for L. Now, these might be scenarios where other men might have had N higher in their preference lists and would prefer to propose to N rather than their current partners if given a second chance. One thing that can be done is to put the men, including A' that have been rejected by N previously into a free_men list. Also, the wives of newly free rejected_by_N men will be put into the free_women list. This process occurs recursively until
        i. all men and women are in the free list **(worst case scenario)**
        ii. only A' and N are in the free_list and get engaged (in which case we utilize the previous matchings to our advantage - **best case scenario**)

A pseudo algorithm is as follows -

```
a) Initially when Almanzo proposes to Laura and she accepts, Nelly becomes free. Other men would have had Nelly prior to their cur
b) If a man is free, he goes to the free_list(men). For m's partner, w, she also becomes free and goes to the free_list(women)
c) Recursively execute step 2 to create the set of free_list(men) and free_list(women)
d) once we have both the sets of free_list(men) and free_list(women), execute Gale-Shapley on the new free_lists based on their pr

// From the book
Gale-Shapley() {
  Initially all m∈M and w∈W are free
  While there is a man m who is free and hasn’t proposed to every woman
    Choose such a man m
    Let w be the highest-ranked woman in m’s preference list to whom m has not yet proposed If w is free then
      (m, w) become engaged
    Else w is currently engaged to m'
      If w prefers m' to m then m remains free
      Else w prefers m to m' (m, w) become engaged m' becomes free
    Endif Endif
  Endwhile
  Return the set S of engaged pairs
}
```

This algorithm is correct as it has been derived from the existing instance of the Gale-Shapley algorithm and hence always provides a stable matching.