HW1 Assignment

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.

Ans.) False.

As there is no guarantee that there exist a pair (m, w) that would rank each other first in their respective preference list, For Example there are 2 men (M1, M2) and 2 women (W1, W2).

And their preference lists

Men's preference list –

M1 – W2, W1.

M2 – W1, W2.

Women's preference list –

W1 – M1, M2.

W2 – M2, M1.

Here no two men and women prefer each other as their first choice.

The Stable Matching's in this scenario would be

Using Men Optimal Approach

M1 – W2.

W2 – M1.

(OR)

Using Women Optimal Approach

M1 – W1.

M2 – W2.

But in an instance, if there exists a pair (M, W) that ranks each other highest in their respectively preference list then in every stable matching they would always be paired. Because if they are not paired with each other, they would form an unstable pair, As Gale – Shapley Algorithms does not allow such pairs to exist, thus in every stable matching the pair (M, W) would always exist.

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.

Ans.) True.

In an instance, if there exists a pair (M, W) that ranks each other highest in their respective preference list then in every stable matching of that instance they would always be paired with each other. Because if they are not paired with each other, they would form an unstable pair, As Gale – Shapley Algorithms does not allow such pairs to exist, thus in every stable matching the pair (M, W) would always exist.

M1 - W1, W2

M2 - W1, W2

and

W1 - M1, M2

W2 - M2, M1

Here M1 and W1 rank each first in their preference list, Thus

Using Men Optimal Approach

M1 – W1

M2 – W2

Using Women Optimal Approach

M1 – W1

M2 – W2

Since both men and women have strong preferences for each other, any deviation from this matching would result in instability.

If M1 were paired with W2, M1 would prefer W1 and W1 would also prefer M1, leading them to break their current matches.

Similarly, if M2 were paired with W1, W1 would still prefer M1, creating instability.

There is one stable matching in this case due to the structure of preferences, where each person has clear preferences that align with the final matching.

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 A and 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— 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 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 A reveals a schedule S and Network B reveals a schedule T. Based on 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 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 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.

Ans.) For any set of TV shows and associated rating, even if the rating of every show is unique, we encounter cycles which makes the algorithm unstable.

To better understand this problem, lets devise an algorithm analogous to Gale – Shapley Algorithm.

Problem Definition and Rules:

Two networks: A, B.

n time slots.

Each show has its own unique rating. (No two shows have the same rating).

Each network has it own schedule and each network can unilaterally changes it schedule to win more time slots.

GALE – SHAPLEY ANALOGOUS ALGORITHM:

1. Each network starts with an initial schedule i.e The start with a fixed schedule.
2. Now network A propose to its changed schedule in order to win more slots
3. Then network B will responded to the changes and devise its own schedule, if it finds that the changes made by network A have impacted its own performance.
4. This will keep on happening until none of the network can make unilateral changes to there schedules in order to win more slots.
5. The algorithm terminates when no network can make a unilateral change that improves its win count, resulting in a stable pair of schedules.

Now let’s discuss about the example where we encounter a cycle.

Let’s say

network A has Show1 and Show2

network B has Show3 and Show4.

The ratings of the shows are:

Network A: [8,6].

Network B: [7,5].

Step 1:

For Slot 1: Network A (8) vs Network B (7). Network A wins.

For Slot 2: Network A (6) vs Network B (5). Network A wins.

Step 2:

Then Network B changes its Schedule, in order to respond to network A schedule.

For Slot 1: Network A (8) vs Network B (5). Network A wins.

For Slot 2: Network A (6) vs Network B (7). Network B wins.

Step 3:

Then Network A changes its Schedule, to respond to network B's change of its schedule.

For Slot 1: Network A (6) vs Network B (5). Network A wins.

For Slot 2: Network A (8) vs Network B (7). Network A wins.

And this keeps on happening endlessly, thus making the algorithm unstable.

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

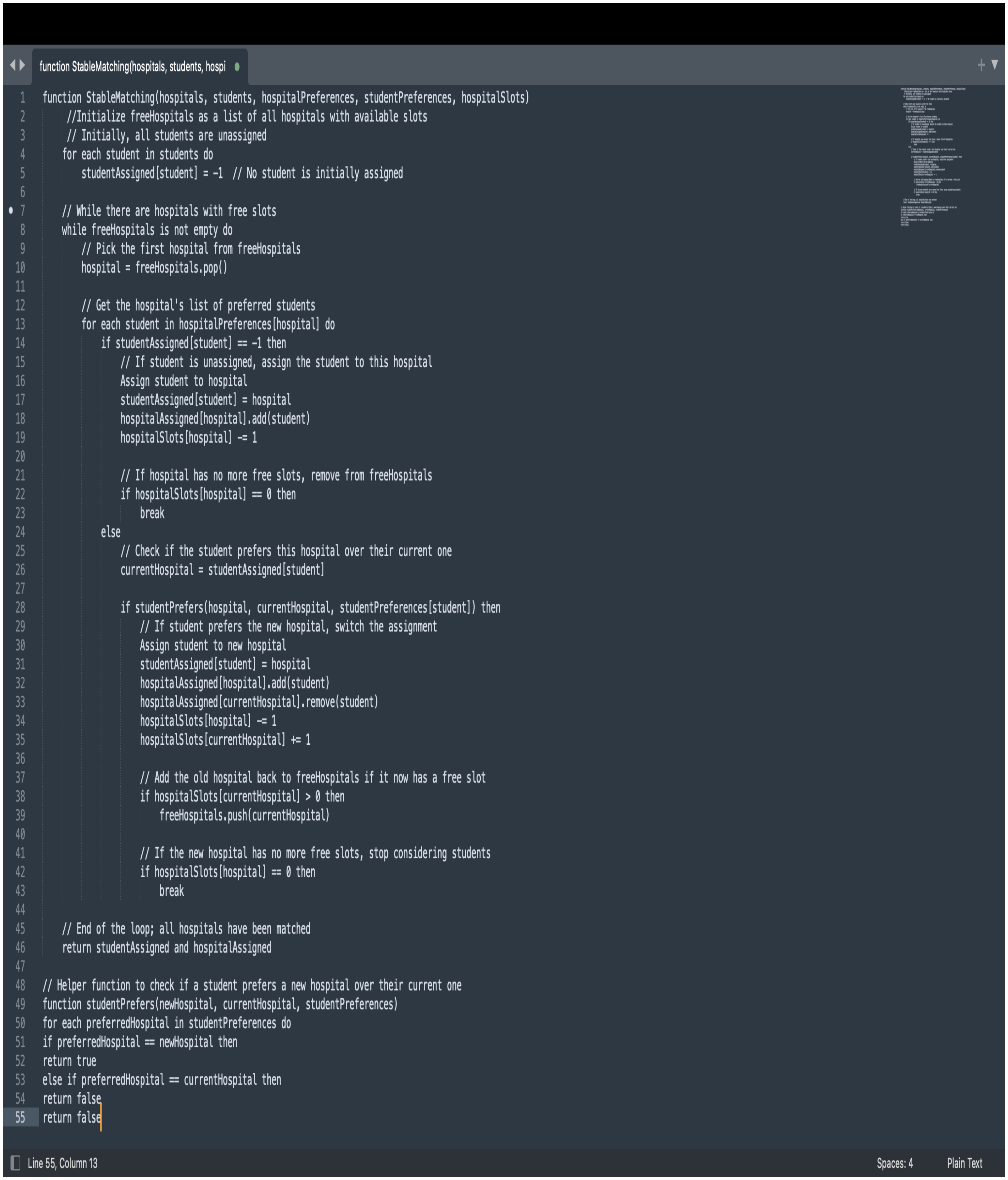
(i) hospitals generally want more than one resident, and
(ii) there is a surplus of medical students. Show that there is always a stable assignment of students to hospitals and give an algorithm to find one.

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

Ans.) We can extend the Gale – Shapley Stable matching algorithm to this problem by making minor changes to it, i.e. here the hospital has multiple slots to accept multiple students. And not all students are assigned a hospital in some cases

So, to handle this scenario, the hospital is the one that proposes to the students and students accept or reject the hospital based on their preference list of the hospitals that they want to study in. We continue this algorithm until all slots in the hospital are filled.

Pseudo Code:



```
function StableMatching(hospitals, students, hospitalPreferences, studentPreferences, hospitalSlots)
     //Initialize freeHospitals as a list of all hospitals with available slots
     // Initially, all students are unassigned
    for each student in students do
        studentAssigned[student] = -1  // No student is initially assigned

    // While there are hospitals with free slots
    while freeHospitals is not empty do
        // Pick the first hospital from freeHospitals
        hospital = freeHospitals.pop()

        // Get the hospital's list of preferred students
        for each student in hospitalPreferences[hospital] do
            if studentAssigned[student] == -1 then
                // If student is unassigned, assign the student to this hospital
                Assign student to hospital
                studentAssigned[student] = hospital
                hospitalAssigned[hospital].add(student)
                hospitalSlots[hospital] -= 1

                // If hospital has no more free slots, remove from freeHospitals
                if hospitalSlots[hospital] == 0 then
                    break
            else
                // Check if the student prefers this hospital over their current one
                currentHospital = studentAssigned[student]

                if studentPrefers(hospital, currentHospital, studentPreferences[student]) then
                    // If student prefers the new hospital, switch the assignment
                    Assign student to new hospital
                    studentAssigned[student] = hospital
                    hospitalAssigned[hospital].add(student)
                    hospitalAssigned[currentHospital].remove(student)
                    hospitalSlots[hospital] -= 1
                    hospitalSlots[currentHospital] += 1

                    // Add the old hospital back to freeHospitals if it now has a free slot
                    if hospitalSlots[currentHospital] > 0 then
                        freeHospitals.push(currentHospital)

                    // If the new hospital has no more free slots, stop considering students
                    if hospitalSlots[hospital] == 0 then
                        break

    // End of the loop; all hospitals have been matched
    return studentAssigned and hospitalAssigned

// Helper function to check if a student prefers a new hospital over their current one
function studentPrefers(newHospital, currentHospital, studentPreferences)
for each preferredHospital in studentPreferences do
if preferredHospital == newHospital then
return true
else if preferredHospital == currentHospital then
return false
return false
```

Line 55, Column 13   Spaces: 4   Plain Text

STEP BY STEP EXPLANATION OF THE ALGORITHM:

1.Initialization:

1. Each hospital h is initially “free” and has a certain number of available slots for students.

2. Each student s is also “free” and unassigned to any hospital.

3. Each hospital and students have a ranked list of preferences for the opposite group.

2.Proposal Phase:

1. Each free hospital h with available slots proposes to the highest-ranked students on its preference list who have not yet rejected it.

2. Each student evaluates the proposal(s) from the hospitals:

3. If the student has no current match, they accept the proposal from the highest-ranked hospital.

4. If the student is already matched to a hospital, they compare the new proposal to their current match. If they prefer the new hospital, they accept the proposal and leave their current match, making a slot free at their previous hospital.

3.Hospital’s Decision:

1. Each hospital h keeps the highest-ranked students from the proposals it receives filling up its available slots.

2. Any lower-ranked students who proposed to the hospital or were proposed to but not accepted are rejected.

4.Repeat Until Stable:

1.This process continues until every hospital has either filled all its available slots or made proposals to all the students on its preference list.

5.Termination:

1.The algorithm terminates when no hospital has an available slot and there are no students left to whom it has not yet proposed.

2.The result is a stable assignment because no student-hospital pair can improve their situation by switching their current match.

Proof by Contradiction:

Let's assume that no stable matching exists. That is every possible matching has at least one unstable pair.

Now let's use the modified version of the gale shapely algorithm to match students with hospitals.

Each hospital h proposes to its most preferred available students.

Students accept or reject based on their preferences for hospitals.

Hospitals can propose to other students if their slots are not yet filled or if a student, they prefer becomes available due to another hospital's proposal.

1.First type of instability: There exists a student s' and hospital h such that:

- h is matched with a student s,
- s' is unassigned,
- Hospital h prefers s' over its assigned student s.

In this case, why didn't h propose to s'? Since s' is unassigned, the algorithm would have allowed h to propose to s' before assigning s. If h preferred s', it would have proposed to s', and s' would have accepted, leaving no instability of this type. Therefore, such a situation cannot arise, contradicting the first type of instability.

2.Second type of instability: There exist students s and s', and hospitals h and h', such that:

- s is assigned to h,
- s' is assigned to h',
- h prefers s' over s,
- s' prefers h over h'.

If such an instability occurred, the algorithm would have allowed h to propose to s' earlier, before proposing to s. If s' preferred h, then s' would have accepted h 's proposal, meaning that s' would have been matched with h, and not h'. Since the algorithm ensures that h and s' have the chance to match earlier in the process, this type of instability also cannot arise.

Therefore through, proof by contradiction we can prove that a stable matching always exists for this modified Gale-Shapely algorithm for hospitals and students.

5.) The Stable Matching Problem, as discussed in the text, assumes that all men and women have a fully ordered list of preferences. In this problem we will consider a version of the problem in which men and women can be indifferent between certain options. As before we have a set M of n men and a set W of n women. Assume each man and each woman ranks the members of the opposite gender, but now we allow ties in the ranking. For example (with n =4), a woman could say that m1 is ranked in first place; second place is a tie between m2 and m3 (she has no preference between them); and m4 is in last place. We will say that w prefers m to m′ if m is ranked higher than m′ on her preference list (they are not tied). With indifferences in the rankings, there could be two natural notions for stability. And for each, we can ask about the existence of stable matchings, as follows.

(a) A strong instability in a perfect matching S consists of a man m and a woman w, such that each of m and w prefers the other to their partner in S. Does there always exist a perfect matching with no strong instability? Either give an example of a set of men and women with preference lists for which every perfect matching has a strong instability; or give an algorithm that is guaranteed to find a perfect matching with no strong instability.

(b)A weak instability in a perfect matching S consists of a man m and a woman w, such that their partners in S are w′ and m′, respectively, and one of the following holds:

– m prefers w to w′, and w either prefers m to m′ or is indifferent between these two choices: or

– w prefers m to m′, and m either prefers w to w′ or is indifferent between these two choices.

In other words, the pairing between m and w is either preferred by both or preferred by one while the other is indifferent. Does there always exist a perfect matching with no weak instability? Either give an example of a set of men and women with preference lists for which every perfect matching has a weak instability; or give an algorithm that is guaranteed to find a perfect matching with no weak instability.

Ans.)

a.) Does there always exist a perfect matching with no strong instability?

Answer: Yes, there always exists a perfect matching with no strong instability.

Algorithm:

The Gale-Shapley algorithm can be adapted to handle the case of ties in preferences while ensuring that there is no strong instability.

Modified Gale-Shapley Algorithm for Strong Stability:

1.Initialize:

•All men and women are initially free.

•Each man proposes to the highest-ranked woman on his list who has not yet rejected him.

•Women can tentatively accept proposals from men but may later change their minds if they receive a better proposal.

2.Proposing Phase:

•Each free man proposes to the highest-ranked woman on his list who has not yet rejected him. If a woman has not been proposed to yet, she tentatively accepts the proposal.

•If a woman has already received a proposal, she compares her current tentative match with the new proposal. If she strictly prefers the new man, she switches to the new man and rejects her current match.

•If the new proposal is tied with the current one, she remains with her current match, thus breaking the tie in favor of stability.

3.Repeat the proposing phase until all men are matched.

4.Termination:

•Once all men are matched, the process terminates, and the result is a perfect matching with no strong instability. No pair of man and woman would strictly prefer each other over their current partners.

Why It Works:

•This algorithm guarantees that if there is a strong preference by a man or a woman over their current match, they will eventually be matched with that partner.

•By breaking ties in favor of the current match, the algorithm ensures that no man and woman will both strictly prefer each other over their current partners.

Thus, there always exists a perfect matching with no strong instability.

b.) Does there always exist a perfect matching with no weak instability?

Answer: No, there does not always exist a perfect matching with no weak instability.

Example: Consider an example with 3 men and 3 women ($n = 3$) and the following preference lists:

•Man 1 (m1) prefers:

1.Woman 1 (w1)

2.Woman 2 (w2) = Woman 3 (w3) (tie)

•Man 2 (m2) prefers:

1.Woman 2 (w2)

2.Woman 1 (w1) = Woman 3 (w3) (tie)

•Man 3 (m3) prefers:

1.Woman 3 (w3)

2.Woman 1 (w1) = Woman 2 (w2) (tie)

•Woman 1 (w1) prefers:

1.Man 1 (m1)

2.Man 2 (m2) = Man 3 (m3) (tie)

•Woman 2 (w2) prefers:

1.Man 2 (m2)

2.Man 1 (m1) = Man 3 (m3) (tie)

•Woman 3 (w3) prefers:

1.Man 3 (m3)

2.Man 1 (m1) = Man 2 (m2) (tie)

Final Stable Matching:

1.Man 1 (m1) with Woman 1 (w1)

2.Man 2 (m2) with Woman 2 (w2)

3.Man 3 (m3) with Woman 3 (w3)

Weak Instability:

In this setup, any matching will have a weak instability because:

•Each man has indifferences in his preference list.

•Each woman has indifferences in her preference list.

For example:

•If man 1 (m1) is matched with woman 1 (w1) and man 2 (m2) is matched with woman 2 (w2), man 1 might prefer woman 2, and woman 2 might be indifferent between man 1 and man 2. This would create a weak instability.

•Similarly, any other matching would result in a weak instability based on the indifferences in preference.

Thus, in this example, every perfect matching has a weak instability.