CS 601: Advanced Algorithms

Assignment by: Group5

Apurva Siddappa Raj (rv6455) and Kavitha Dilli babu(sk9653)

Solution 1:

A Gale-Shapley aims to match one man with one woman. But assigning medical students to hospital focus to match one hospital with multiple residents.

INITIALIZE M to empty matching.
For a hospital h that has open positions: Offer a position to the next student on h preference
s ← first student on h's list
If s is free: s accepts the offer (Add h–s to matching M)
Number of positions at h decreases by 1
Else if s is already committed to the hospital h:
If S prefers h' to h:
S remains committed to h'
Else:
S accepts h offer
Number of positions at h decreases by 1
Number of positions at h' increases by 1

Proving there is always a stable assignment of students to hospitals:

Considering the first type of instability, Here h makes offers in its preference, and any s who is free accepts an offer. Since s' isn't assigned to a hospital, s' received no offers. If h preferres s' to s, h would have made an offer to s' before making an offer to s, so the first type of instability can't happen.

In second type of instability, h made offers in its preference. h preferred s' to s, h would have made an offer to s' before making an offer to s. If no such offer was made, then h must have preferred s to s'. Since h' is match for s', s' prefers h' to some other h and s' must prefer h' to h, this shows a contradiction to the assumption that s prefers h to h'.

Therefore, since neither instability can occur, assignment generated by the algorithm must be a stable assignment of students to hospitals.

Solution 2:

a. Yes, There is always a perfect matching with no strong instability.

According to the definition of a strong instability in a perfect matching problem is each man M and each woman W prefers others to their partner. Since we already know It's not possible to have a strong instability in a perfect matching.

We can prove this by contradiction by assuming the below Perfect Matching has had an instability.

So, (M1, W1) and (M2, W2) have instability (M1, W2) because M1 prefers W2 to W1 and W2 also prefers M1 to M2.

If M1 proposed to W2 there should be two possible outcomes yes or no. Following are the two answers which leads to contradiction.

Since M1 ended up being matched with W1

M1 proposed to W1 before proposing to W2, which is a contradiction to assumption that W2 prefers M1 to M2.

Or M1 did propose to W2 before proposing to W1 but was rejected by W2, then following two scenarios would have occurred if that is true.

=> W2 was free and accepted M1 proposal but later rejected M1 for some other Man M' or W2 was already engaged to some man M' where prefers over M1

In either way, W2 ended up with M2, that is W2 prefers M2 over M' over M1 which is also a contradiction

b. No, there is not always a perfect matching without a weak instability. Assume a case where we have a set of men and a woman. M = {M1 and M2} and W = {W1 and W2}. M1 and M2 both prefer W1 to W2 and both W1 and W2 are indifferent to M1 to M2.

Men's preference list

	1st	2nd
M1	W1	W2

M2	W1	W2

Women's preference list:
Women do not have an ordered preference list since both W1 and W2 are indifferent to M1 and M2.

The possible perfect matching from the above example

1. (M1, W1); (M2, W2) and
2. (M1, W2); (M2, W1)

(M1, W2) from case1 and (M1, W1) from case 2 have a weak instability.

Solution 3:

There does not exist an unstable assignment. The GS algorithm generates the stable pairs. Here, n Teaching Assistants are to be assigned to n courses with one course having exactly one TA. Every TA and course do not form an unstable pair for any preference lists.

First, each course is assigned to its top-ranked choice. Next, each TA who has received at least two proposals keeps a (tentatively) top-ranked proposal and rejects the rest. Then, each course that has been null is assigned to its top-ranked choice among the TA's. Again each TA who has at least two proposals (including ones from previous rounds) keeps a top-ranked proposal and rejects the rest. The process repeats until no course has a TA to propose to or each TA has at most one proposal. At this point the algorithm terminates and each course is assigned to a TA. No course is assigned to more than one TA. Since each TA is allowed to keep only one course at any stage, no TA is assigned to more than one course. Therefore, the algorithm terminates in a matching.

Solution 4:

Gale-Shapely algorithm in Java- women proposes to men

```
package advalgo;

public class GaleShapley
{
   private int N, engagednum;
   private String[] women;
```

```
private String[] men;
private String[][] womenPref;
private String[][] menPref;
private String[] menscompanion;
private boolean[] womencomitted;


public GaleShapley(String[] m, String[] w, String[][] mp, String[][] wp)
{
   N = wp.length;
   engagednum = 0;
   men = m;
   women = w;
   menPref = mp;
   womenPref = wp;
   womencomitted = new boolean[N];
   menscompanion = new String[N];
   Matching();
}

private void Matching()
{
   while (engagednum < N)
   {
      int free;
      for (free = 0; free < N; free++)
         if (!womencomitted[free])
            break;

      for (int i = 0; i < N && !womencomitted[free]; i++)
      {
         int index = menIndexOf(womenPref[free][i]);
         if (menscompanion[index] == null)
         {
             menscompanion[index] = women[free];
            womencomitted[free] = true;
            engagednum++;
         }
         else
         {
            String currentPartner = menscompanion[index];
            if (morePreference(currentPartner, women[free], index))
            {
```

```
                menscompanion[index] = women[free];
                womencomitted[free] = true;
                womencomitted[womenIndexOf(currentPartner)] = false;
            }
        }
    }
    }
    printMatches();
  }

  private boolean morePreference(String curPartner, String newPartner, int
index)
  {
    for (int i = 0; i < N; i++)
    {
      if (menPref[index][i].equals(newPartner))
        return true;
      if (menPref[index][i].equals(curPartner))
        return false;
    }
    return false;
  }

  private int womenIndexOf(String str)
  {
    for (int i = 0; i < N; i++)
      if (women[i].equals(str))
        return i;
    return -1;
  }

  private int menIndexOf(String str)
  {
    for (int i = 0; i < N; i++)
      if (men[i].equals(str))
        return i;
    return -1;
  }

  public void printMatches()
  {
    System.out.println("The final match : \n");
    for (int i = 0; i < N; i++)
```

```
        {
            System.out.println(menscompanion[i] +" "+ "and" +" "+ men[i]);
        }
    }

    public static void main(String[] args)
    {
        System.out.println("Implementing Gale Shapley Marriage Algorithm such
that women propose men \n");

        String[] m = {"Victor", "Wyatt", "Xavier", "Yancey", "Zeus"};

        String[] w = {"Amy", "Bertha", "Clare", "Diane", "Erika"};


        String[][] mp = {{"Bertha", "Amy", "Diane", "Erika", "Clare"},
                {"Diane", "Bertha", "Amy", "Clare", "Erika"},
                {"Bertha", "Erika", "Clare", "Diane", "Amy"},
                {"Amy", "Diane", "Clare", "Bertha", "Erika"},
                {"Bertha", "Diane", "Amy", "Erika", "Clare"}};

        String[][] wp = {{"Zeus", "Victor", "Wyatt", "Yancey", "Xavier"},
                {"Xavier", "Wyatt", "Yancey", "Victor", "Zeus"},
                {"Wyatt", "Xavier", "Yancey", "Zeus", "Victor"},
                {"Victor", "Zeus", "Yancey", "Xavier", "Wyatt"},
                {"Yancey", "Wyatt", "Zeus", "Xavier", "Victor"}};

        GaleShapley gs = new GaleShapley(m, w, mp, wp);
    }
}
```

Output:

Implementing Gale Shapley Marriage Algorithm such that women propose men

The final match :

Diane and Victor
Clare and Wyatt
Bertha and Xavier
Erika and Yancey
Amy and Zeus

Time Complexity:

In this Stable Matching problem there are 5 women and 5 men. For the input given in the program each of the women could be matched against their most preferred partner without running the inner loop iterating over the preference list of each woman more than once.

But the worst case scenario would be to check the stable matching of each woman against all the men in her preference list. In this case the inner loop would execute for a maximum number of M times since it iterates all the men in each of the women's preference list. So, there are at most N^2 proposals and each takes O(1) times.

Therefore the time complexity of this program is T(n)=O(n2).