IE523 B,OB,ONL: Financial Computing Fall, 2020

Programming Assignment 4: The Stable Marriage Problem

Due Date: 2 October, 2020

©Prof. R.S. Sreenivas

The *Stable Marriage Problem* is a catchy title/version of what is known in economics/computer-science as the *Stable Matching Problem.* You can read a lot more about this problem and its implications at the following Wikipedia Link, or from the following (classic) texts [1, 2]. In its original form – we have two equal-sized set of participants (say, *Men* and *Women*), and a *matching* is a one-to-one mapping between these two sets (i.e. a "marriage").

Each member of these two sets has a *preference*, which is a ordering of the members of the other set. That is, each man (resp. woman) has an ordering that explicates his preference over the set of women (resp. men). A matching is unstable if a man A and a woman a, not married to each other, mutually prefer each other to their spouses. As Knuth notes – an unstable marriage leaves the door open for a "*liaison dangereuse*," which we wish to avoid. This could be the case when[1]

- A is married to b,
- a is married to B,
- A prefers a to b, and
- a prefers A to B.

A matching is *stable* if this situation does not occur. Given a set of preferences of the men and women, it is on interest to find a stable marriage/pairing (or, the "best" stable marriage) between the men and women.

This problem is quite commonplace in a variety of social-choice instances. Several college-admissions, assignment of hospitals/residents, etc. follow the structure enunciated above. There is a lot that has been said on this problem (cf. [2]). For instance, it is possible to construct instances involving n-many men and women, with $2^{n/2}$-many stable solutions. There are other instances, where swapping-partners in a systematic manner, does not lead to any stable solution, etc. You can read about all this at leisure.

We are going to concentrate on what is known as the "*Gale-Shapley Algorithm*[2]," which is described in pseudo-code format in figure 1. There are several things that you can prove about this algorithm (cf. [2] for details):

[1]We denote the men with the upper-case alphabet, while women are denoted by the lower-case.

[2]Lloyd Shapley won the 2012 Nobel Memorial Prize for Economics for this work. (Late) David Gale is a well-known figure in game-theory and economics.

1. If a woman a is un-engaged from A at some point in the execution, then no stable matching can contain the pair Aa.
2. If A prefers a to his fiancée, it means that a has rejected him for another.
3. A woman's situation never worsens throughout the course of the algorithm.
4. The list of preferences of each man never becomes empty.
5. The final matching obtained (when the while-loop terminates) is always stable.
6. The algorithm gives the *optimal* result for each man – that is, each man has the best possible marriage. If there is another stable marriage for a man, and his partner is different in that marriage, then she will be worse than the one he is paired with currently.
7. The result is also the *worst* solution for the women – that is, if there is another stable marriage for a woman, her partner, if different, will be superior to the one assigned by the algorithm.

You could reverse the roles of the men and women in the algorithm (i.e. the women do the proposing, instead of the men) then the results 6 and 7, listed above will be reversed.

```
Boolean Gale-Shapley Algorithm()
1: Get the preferences for the men and women.
2: Assign each person to be un-engaged, initially.
3: while some man $M$ is un-engaged do
4:    Let $w$ be the first woman on $M$'s preference-list to whom $M$ has not yet proposed.
5:    if $w$ is un-engaged then
6:       Assign $M$ and $w$ to be engaged to each other.
7:    else
8:       if $w$ prefers $M$ to her fiancé $\overline{M}$ then
9:          Assign $M$ and $w$ to be engaged to each other. Declere $\overline{M}$ to be un-engaged.
10:      else
11:         $w$ rejects $M$ (and $M$ remains un-engaged).
12:      end if
13:   end if
14: end while
```

Figure 1: Pseudo-code for the Gale-Shapley algorithm.

For this programming exercise you will write a C++ program, along the lines of the hint shown in figure 2, that solves the Stable Marriage Problem using the Gale-Shapley algorithm of figure 1.

```
#include <iostream>
#include <vector>
#include <fstream>

using namespace std;

class stable_marriage_instance
{
    // Private
    int no_of_couples;
    vector <vector <int> > Preference_of_men;
    vector <vector <int> > Preference_of_women;
    vector <int> match_for_men;
    vector <int> match_for_women;

    // private member function: checks if anybody is free in boolean "my_array"
    // returns the index of the first-person who is free in "my_array"
    // if no one is free it returns a -1.
    int anybody_free(vector <bool> my_array)
    {
        // fill the necessary code for this function
    }

    // private member function: if index1 is ranked higher than index2
    // in a preference array called "my_array" it returns "true"; otherwise
    // it returns "false"
    bool rank_check (vector <int> my_array, int index1, int index2)
    {
        // fill the necessary code for this function
    }

    // private member function: implements the Gale-Shapley algorithm
    void Gale_Shapley()
    {
        vector <bool> is_man_free;
        vector <bool> is_woman_free;
        vector <vector <bool> > has_this_man_proposed_to_this_woman;

        int man_index, woman_index;

        // initializing everything
        for (int i= 0; i < no_of_couples; i++)
        {
            // do the necessary initialization here
        }

        // Gale-Shapley Algorithm
        while ( (man_index = anybody_free(is_man_free)) >= 0)
        {
            // fill the necessary code here
        }
    }

    // private member function: reads data
    void read_data(int argc, const char * argv[])
    {
        // fill the necessary code here.  The first entry in the input
        // file is the #couples, followed by the preferences of the men
        // and preferences of the women.  Keep in mind all indices start
        // from 0.
    }

    // private member function: print solution
    void print_soln()
    {
        // write the appropriate code here
    }

public:
    void solve_it(int argc, const char * argv[])
    {
        read_data(argc, argv);

        Gale_Shapley();

        print_soln();
    }
};

int main(int argc, const char * argv[])
{
    stable_marriage_instance x;
    x.solve_it(argc, argv);
}
```

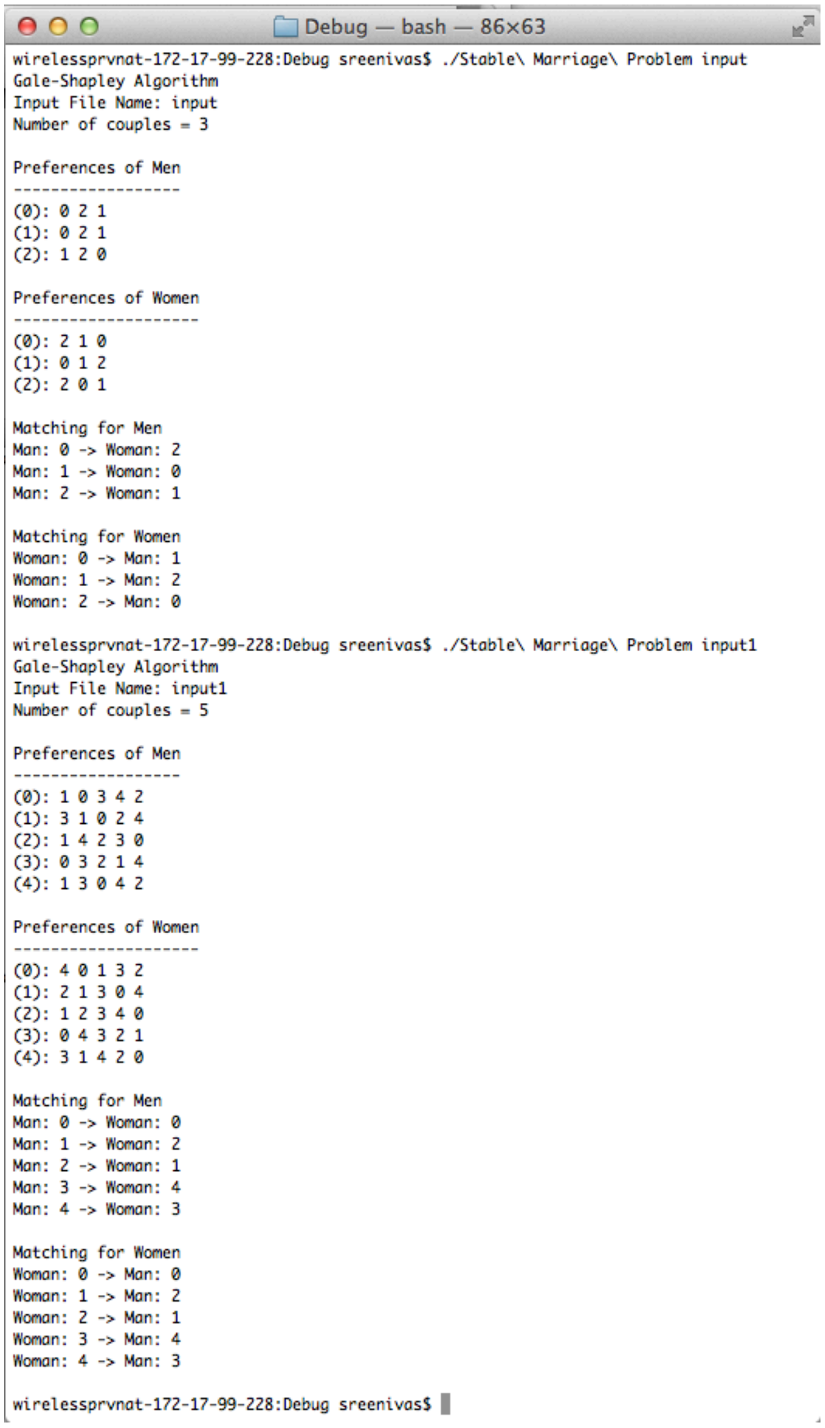
Figure 2: f20_prog4_hint.cpp: C++ code to help you with Programming Assignment #4.

Your code will read the name of the input file on the command-line. The format of the input file is as follows: the first-line contains the number of couples, this is followed by the preference list of the men, and then the preference list of the women. It is important to note that if we have n couples the men/women are referred to using their indices $\{0, 1, \ldots, n-1\}$. The Compass website contains two sample input files `input` and `input1`. Figure 3 shows a sample output.

References

[1] D. Gusfield and R.W. Irving. *The Stable Marriage Problem: Structure and Algorithms*. The MIT Press, 1989.

[2] D.E. Knuth. *Stable Marriage and Its Relation to Other Combinatorial Problems*. American Mathematical Society, 1991.



Debug — bash — 86×63

```
wirelessprvnat-172-17-99-228:Debug sreenivas$ ./Stable\ Marriage\ Problem input
Gale-Shapley Algorithm
Input File Name: input
Number of couples = 3

Preferences of Men
------------------
(0): 0 2 1
(1): 0 2 1
(2): 1 2 0

Preferences of Women
--------------------
(0): 2 1 0
(1): 0 1 2
(2): 2 0 1

Matching for Men
Man: 0 -> Woman: 2
Man: 1 -> Woman: 0
Man: 2 -> Woman: 1

Matching for Women
Woman: 0 -> Man: 1
Woman: 1 -> Man: 2
Woman: 2 -> Man: 0

wirelessprvnat-172-17-99-228:Debug sreenivas$ ./Stable\ Marriage\ Problem input1
Gale-Shapley Algorithm
Input File Name: input1
Number of couples = 5

Preferences of Men
------------------
(0): 1 0 3 4 2
(1): 3 1 0 2 4
(2): 1 4 2 3 0
(3): 0 3 2 1 4
(4): 1 3 0 4 2

Preferences of Women
--------------------
(0): 4 0 1 3 2
(1): 2 1 3 0 4
(2): 1 2 3 4 0
(3): 0 4 3 2 1
(4): 3 1 4 2 0

Matching for Men
Man: 0 -> Woman: 0
Man: 1 -> Woman: 2
Man: 2 -> Woman: 1
Man: 3 -> Woman: 4
Man: 4 -> Woman: 3

Matching for Women
Woman: 0 -> Man: 0
Woman: 1 -> Man: 2
Woman: 2 -> Man: 1
Woman: 3 -> Man: 4
Woman: 4 -> Man: 3

wirelessprvnat-172-17-99-228:Debug sreenivas$ 
```

Figure 3: Sample Output.