CMPE 300 - Analysis of Algorithms
Spring 2016
Programming Project

Due: 15.05.2016 23:59

Introduction

In this project, you are going to experience parallel programming with C/C++ using MPI library. You are going to implement a parallel algorithm for solving the stable marriage problem.

An instance of size n of the stable marriage problem (also known as stable matching problem) involves two disjoint sets of size n, which we will call men and women. Each person has a preference list containing all members of the opposite sex in the order of preference. A matching is a correspondence between the elements in the two sets, men and women. When there is a match between two elements, we will say that they are partners. A matching is unstable if there exists a man and a woman who are not partners and both would be happier if they were partners. Otherwise, the matching is called stable.

Consider the following example. Below are the tables of preferences for men (m_1, m_2, m_3) and women (w_1, w_2, w_3). A matching is given on the right where (m_i, w_i) denotes that the man m_i is matched with the woman w_i.

m_1	2	1	3
m_2	1	2	3
m_3	2	3	1

Preference table for men

w_1	3	2	1
w_2	1	2	3
w_3	2	1	3

Preference table for women

(m_1, w_2)
(m_2, w_1)
(m_3, w_3)

Matching

Let us verify that the matching is stable. Let us consider m_1 and m_2. They are both matched with their first preferences. Now consider m_3 who is matched with w_3. m_3 would prefer w_2 but w_2 prefers m_1 over m_3. Therefore, there are no men and women who are not partners, and they would be happier if they were partners.

We could also verify the stability of the matching by considering women. w_1 would prefer m_3 over m_2 but m_3 wouldn't prefer w_1 over his current partner w_3. w_2 is matched with her first choice. w_3 would prefer m_1 and m_2 over m_3 but m_1 and m_2 wouldn't prefer her over their current partners.

Actually it is sufficient to consider each member of one sex for the verification. Now let us consider an unstable matching.

(m_1, w_1)
(m_2, w_2)
(m_3, w_3)

m_1 is matched with w_1 and m_2 is matched with w_2. Both m_1 and w_2 would prefer each other over their current partners which proves that the matching is not stable.

If you forget about men, women and marriage, the problem is actually to find a matching between two disjoint sets of equal size considering their preferences. The problem has different versions like stable roommates problem where we try to match $2n$ people with each other or college admission problem where multiple students are matched with a college.

Gale Shapley Algorithm

In 1962, Gale and Shapley proved that a stable matching always exists (there may be many stable matchings in fact) and they presented an algorithm which always gives a stable matching. The algorithm can be seen as a sequence of proposals from men to women. During the algorithm, we will call each person as free or engaged.

Each men proposes to the women on his list, in the order of preference until he becomes engaged. When a free woman receives a proposal, she accepts and becomes engaged. If she is engaged, then she compares the new proposer with her fianc. If she prefers the new proposer over her fianc, she breaks her current engagement and becomes engaged to the current proposer. Note that her old fianc is free now and may propose to the next women in his list. The algorithm terminates when everyone is engaged.

Note that during the algorithm's execution, a man may alternate between being engaged and free. Once a woman is engaged, she can not become free but her fianc may change. A man who is engaged more than once, becomes engaged to a less favorable woman each time and a woman who is engaged more than once, becomes engaged to a more favorable man.

Below is the pseudocode for the algorithm.

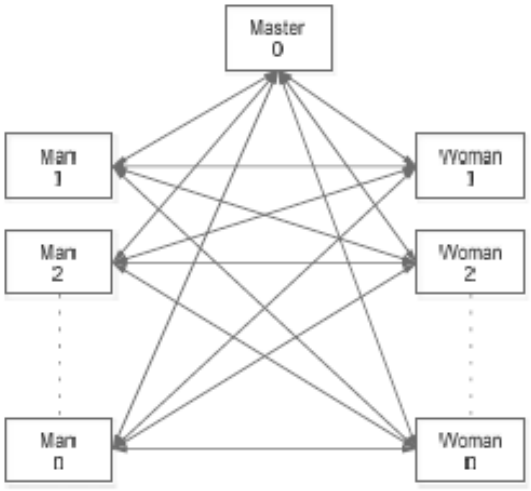
```
Function: Gale-Shapley

    Assign each man and woman to be free
    while There exists a free man m do
        w ← first woman on m's list to whom m has not yet proposed
        if w is free then
            Assign m and w to be engaged
        else
            if w prefers m to her fianc m' then
                Assign m and w to be engaged and m' to be free
            else
                w rejects m
            end if
        end if
    end while
    Output the matching
end Gale-Shapley
```

The order in which the free men propose does not change the outcome
of the algorithm. For any given instance, the algorithm terminates and always gives the same stable matching. Runtime of the algorithm is $O(n^2)$. If you are interested, you may further read about the algorithm in [1]. You may ask whether the matching is optimal for the individuals. Note that this is different from stability. In fact, the algorithm is man-optimal, that is each man has the best partner that he can have in any stable matching. The reason behind this is that the men are the ones who are proposing. The algorithm would be woman optimal if women were the ones who proposed. A nice blog entry which would familiarize you about the stable marriage problem and discusses the optimality of the algorithm can be found here.

[1]. Dan Gusfield and Robert W. Irving. 1989. The Stable Marriage Problem: Structure and Algorithms. MIT Press, Cambridge, MA, USA.

Parallel Gale-Shapley Algorithm

For this project, you are going to implement the parallel version of the Gale-Shapley algorithm using the MPI environment. For the MPI environment and tutorials please visit the web page. You can find many other tutorials in the web. For an instance of the problem with n men and n women, there will be 1 master and $2n$ slave proccessors each proccessor representing a person, $2n + 1$ proccessors in total. The aim of this project is not to obtain an algorithm with a better runtime but to to learn MPI messaging routine. The possible communication between the proccessors is depicted below.



Each man will be proposing to some women and the women either will accept or reject the proposal in response. Details of the implementation are given below.

Implementation

1. You will read the input from a text file and print the result in another text file. Input file will contain the number of men/women, the preference list of each man followed by the preference list of each woman. The output file will contain the list of couples. An example input and output file are as follows:

Input file	Output file
3	
3 2 1	
1 2 3	$1-2$
2 1 3	$2-1$
3 2 1	$3-3$
1 2 3	
2 1 3	

2. The whole input shouldn't be read/stored by each proccessor. Input should be read by the master proccessor and the master proccessor should send each person his/her preference list and any other information needed. Any functioning of the program regarding the whole program such as printing the output should be done by the master proccessor.

3. When the proccessors are exchanging information, be careful to avoid deadlock. For instance, a woman who is waiting for a proposal may not be the first choice of any man. An engaged man may or may not become free afterwards. You should decide when to terminate with the help of the master proccessor.

4. You are not allowed to use MPI_Status object and its methods. (except its appearance in receive methods) You will not get any point if you don't obey this rule.

5. The names of the input and output files will be given on the command line, as well as the number of proccessors. An example execution for a Ubuntu user that runs on 7 processors and uses input.txt and output.txt files would be:
   mpiexec -n 7 ./project.exe input.txt output.txt You may assume that the correct number of proccessors is given.

6. Several input and output files to test your code will be provided in the course website.

Bonus Part

For the bonus part of this project, you are required to implement parallel version of college admission problem [2]. There may be m colleges and n students where each college has some quota q_i. Colleges and students have a list of preference. Total number of available quotas is more than the number of students. An assignment of students to colleges will be unstable if there is a student and a college who are not matched but would prefer each other over their current matches. A real-life application of the algorithm is the National Resident Mathcing Problem and a tutorial is available at the web site. Lloyd Shapley and Alvin Roth won the Nobel Prize in Economic Sciences in 2012 extending this idea.

[2] Gale, David, and Lloyd S. Shapley. "College admissions and the stability of marriage." The American Mathematical Monthly 69.1 (1962): 9-15.

Submission

1. The deadline for submitting the project is May 15, 23:59. The deadline is strict. We will have demo sessions the following days. In the demo session, you are required to bring your own laptop and explain how you implemented the project.
2. This is an individual project. Your code should be original. Any similarity between submitted projects or to a source from the web will be accepted as cheating.
3. Your code should be sufficiently commented and indented.
4. You should write a header comment in the source code file and list the following information.
   /*
   Student Name: Ali Veli
   Student Number: 2008123456
   Compile Status: Compiling/Not Compiling
   Program Status: Working/Not Working
   Notes: Anything you want to say about your code that will be helpful in the grading process.
   */
5. You must prepare a document about the project, which is an important part of the project. You should talk about your approach in solving the problem. Follow the guidelines given in the "Programming Project Documentation" link on `http://www.cmpe.boun.edu.tr/~gungort/informationstudents.htm`.
6. Submit your project and document as a compressed archive (zip, rar, ...) to the following link: `https://www.dropbox.com/request/K4LYusljxK3s16JuqJcU`. Your archive file should be named in the following format: "name_surname.ext". You don't need to submit any hard copies.
7. If you have any further questions, send an e-mail to *ozlem.salehi@boun.edu.tr*.