Project 2 Milestone
The Stable Marriage Problem

Kristen Lawler

April 11, 2017

1 Abstract

The stable marriage problem addresses the following issue: suppose you have a population that can be split in half by some defining characteristic (for example gender). If each member of this population were to have a preferential list of possible mates, then how could they be paired together so that the resulting matchings are stable?

A common solution to this problem follows an iterative process of proposals and acceptances or rejections. In order to model this process in JAVA, a Suitor object was created and utilized to represent each individual in the population. For all individuals, aside from the user, a random list of preferential mates was generated. Then, an iterative process was used so as to pair up all individuals while considering these lists of preference.

Though many solutions do exist to this problem, such as the Gale-Shapley algorithm, most are traditionally programmed in C++ or Python, making this solution unique in its own right. This problem is particularly useful since its solution can be applied to many real life situations. These include residency assignments in hospitals, college housing assignments, college acceptances, and on-line dating applications.

2 Introduction

A solution to the stable marriage problem must result in a "stable" matching of all individuals within the population. In order to achieve this stability, we must have either that each individual is paired with their preferential mate, or that the mate they prefer is paired with someone higher on his or her own preference list. In order to ensure this is achieved, we select one "side" of the population to be the proposers. Beginning with their most preferential mate, each individual on the chosen side of the population proposes to a mate on the other. The mate either accepts the proposal or rejects it based upon their own preferences. This process repeats until all individuals have a mate.

The motivation of this project is to utilize JAVA so as to emulate the previously mentioned process. This methodology of matching has been mathematically proven to result in stable pairings so long as the population size is even, and no one individual is "hated" by all potential mates. This specific program allows for the user to act as a member in the population. As such, it takes the user's input for a list of preferences. It also takes in a user input for the number of suitors in the population.

3 Detailed System Description So Far

The first step in writing a program that will address and solve the stable marriage problem was to create an object that could be used to represent each individual in the population. The Suitor object was therefore created. Each suitor has an I.D attribute. This is used to specify individuals during the proposal process. Suitors also contain an array which holds their preferential list of mates. For all individuals aside from the user, this list is randomly generated using a private method called permutation. These objects contain several other attributes which are utilized during the matching process. The Suitor object was used to generate individuals on both sides of the population.

Suitor
identity: int matches: int proposal: int holder: int[] rank: int[]
Suitor(x: int, m: int) Suitor(int x, int m, int[] p) permutation(): int[]

After getting the user's input for how many mates there should be in the population (essentially one half of the total population size) which we will refer to as n, the program generates two arrays of the specified size holding n many Suitor objects each with a different I.D. (0 to $n-1$). These arrays represent the separated halves of the population. The user then inputs their list of preference referring to each suitor by its I.D., and is placed as the first element in the array of all possible "mates" of the "proposers". Since the user specifies their own input, there are two different constructors for the Suitor object.

4 Next Steps

The goal at this point is to finish the algorithm that simulates one side of the population proposing to the other until all candidates are matched. Currently, I have an outline of how to use the attributes of the various Suitor objects to do so, but have not yet successfully written the entire algorithm. Once this is completed, the final step will be to output to user's match and to perhaps also output the average preference that was achieved for each side of the population by this matching.

5 Bibliography

D. Gale and L.S. Shapley, "College Admissions and the Stability of Marriage," American Mathematical Monthly 69 (1962), pp. 9–14.

Gusfield, Dan, and Robert W. Irving. The Stable Marriage Problem: Structure and Algorithms. Cambridge Mass.: MIT, 1989. Print. Foundations of Computing Ser.

Hunt, William. "The Stable Marriage Problem." West Virginia University, Morgantown.

Sahai, Anant. "The Stable Marriage Problem: An Application of Induction in Understanding Algorithms." University of California, Berkley.