Stable Matching Report

Dennis, Thor, Daniel, Nicoline, Maria

September 5, 2017

Results

Our implementation produces the expected results on all input–output file pairs. However, the matches are correct, but the order of the printed results do not match the, provided -out files. This is because we iterate through the array of women when printing instead of the array of men.

We solved this by sorting the results before checking the differences in an SH script.

On input `sm-bbt-in.txt`, we produce the following matching:

Sheldon – Amy, Leonard – Penny, Howard – Bernadette, Rajesh – Priya

Implementation details

The men and women are stored in a combined double array of integers, where the first array denotes the ID of the person and the second array differs between men and women. In this regard, for men, the second array holds a priority list of his preferences whereas for women the second array serves as a "dictionary." This dictionary holds the preference value for all men for a given woman. That is, at a given index i, the dictionary contains the preference value of man i for the given woman. The names of the people are stored in a separate string array and can be looked up with the ID of a person. Also, we have an int array holding the current pointers of every man. These pointers point to the current place in a man's priority queue of women. Furthermore, a boolean array holds a value for each man, indicating whether he is paired or not. Lastly, we use an int array to hold the ID of the current partner for every woman.

We can check/find a free man who has not proposed to every woman in time $O(n)$ because we store this information in a boolean array.

With these data structures, our implementation runs in time $O(n^2)$ on inputs with n men and n women.