Stable Marriage implementation

Arleth, Johan
Bele, Claudiu
Jacobsen, Andreas
Ulrik, Kenneth Ry

September 12, 2016

1 Results

Our implementation produces the expected results on all input–output file pairs. Having implemented conversion-methods between an woman's internal priority-list and the list containing all people, we are able to precisely identify people from and to either list, ensuring that the right comparisons are made, assuming that a woman's "marriedTo" field at all times contains either the real ID of the man to whom she is currently married or -1.

For example, on input `sm-bbt-in.txt`, we produce the following matching:

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

This is the expected outcome, where initially Rajesh proposes to Bernadette, she accepts, but then ruins Rajesh's heart in priority of her one true love, Howard. Rajesh, the soldier he is, carries on to propose to every woman in the world, but finds rejection in one or another form at every corner, until he finally attempts the proposal of his least favorite Priya, who also doesn't really care for him. However, in the lack of other offers, she must settle with such a man as Rajesh, more specifically Rajesh.

2 Implementation details

2.1 Preferences

The men's preferences are stored in a simple Stack (In our Python implementation, simply a list, upon which we call the method "pop()"). The Stack contains the IDs of all women, of course ordered by the man's preference, such that the first popped Woman is the one he prefers the most.

The women's preferences are stored in an array, where the indexes, using a slight indexing translation, corresponds to the IDs of the men, such that a woman may easily look up a man in her array of priorities. The values held in this array contains an integer defining the value of the woman's preference towards the man identified by the index. The higher the integer value, the higher the preference towards the man.

2.2 Parsing

The implementation parses in stages that have each their own condition for being caught. Once one if-branch has been depleted, it will never be entered again, assuming input-files follow the same format. First, all ID's and their names are recorded and stored in objects. Next, the amount of people is recorded, and a counter based on this value is used when reading all the preferences of the men and women.

Additionally, a challenge occured, where various input files were inconsistent in whether or not the preference lists had an additional space at the end of the line. To cope with this, the line – splitted by spaces – is checked for whether the last element contains the "\n" (newline) character, and if so, that element is removed.