Stable Matching Report

Amanda Sauerberg-Hansen, Kamila Garczyńska, Nedas Šurkus, Niclas Abelsen, Tomas Babkine-Di Caprio

September 7th, 2022

Results

Our implementation produces the expected results for all input files.

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

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

Implementation details

We store the unmarried men in the stack `unmarried_men` (line 27) and use pop to remove the first element when they are engaged to a woman (line 46), or append them back to the stack when their engagement is broken (line 43). When the stack of unmarried men is empty, we return `man_fiancée` which contains a stack of all men sorted to all women, record the output in *.test.out.txt and the program exits.

InI the dictionary `man_fiancée` (line 28) the dictionary key corresponds to the mens name and the value is initialized to None. When a man is engaged to a woman, we store the woman's name in the value corresponding to the man's name index.

Conversely, we also store the women's engagements to men in the `woman_current` (line 29) dictionary where the keys correspond to the women's name and the value is initialized to None. When a woman is engaged to a man, we store the man's name at her key in the dictionary.

To retrieve the next woman the man will propose to, we store the man's name as key and a number as value in a dictionary called `next_proposals` (line 30). This number corresponds to the men's preference and we use this value to retrieve the women's name. We increment the value stored on the man's key. In line 35 we use this value to access the name of the woman which will be proposed to then in line 40 we use this name to access the woman's preference in constant time and keep track of which woman each man has proposed to.

To keep track of the next women that any given man will propose to, we store the current index of the men's preference list as values in the dictionary `next_proposals` (line 35). The values are initialized to 0, since this is the first index of the men's preference list, and will increment by one after each proposal. We use this to access the name of the woman that is being proposed to's, we can use this name to access her preferences in constant time.

Since we iterate over the algorithm's logic until every man has proposed to every woman that isn't engaged and that through the use of dictionaries we can access that information in constant time, our algorithm runs in $O(n^2)$ time. Where n is the number of men and women.

How to run our solution

1) How to test our solution on a single file:
    a) From the terminal, run

    ```
    python3 stable_matching.py <path to data folder>/<name of file>.txt
    ```

    b) Output of the test will be in `data/<name of file>.out.txt`

2) How to run all tests:
   The testScript is required to be in the data folder where all the input files are.
    a) From the terminal, run the shell script

    ```
    ./testScript.sh
    ```