Stable Matching Report

Author 1 and Author 2

April 4, 2019

1 Results

Briefly comment the results, did the script say all your solutions were correct? Approximately how long time does it take for the program to run on the largest input? What takes the majority of the time?

Yes, all were correct. With a lot of programs open at the same time.

Start of test 4 at 2,5s. "checking..." at 9.5s. Start of test 5 at 23s. "Checking..." at 34s. Finished at 47s.

The largest input, test 5, took 24s.

2 Implementation details

How did you implement the solution? Which data structures were used? Which modifications to these data structures were used? What is the overall running time? Why?

We created the objects Man and Woman. Both including preference lists, were the Woman's is inverted. The preferenceList was a Map for women, since we used the man's ID as key, and a Deque for the Man.

For CoupleParser we usedone Map¡Integer, Woman/Man¿ each for men and women (two in total). Additionally, we used several arrays to build the input as a matrix and divide the problem into easier problems. We created a seperate Woman-array and a Man-array with their prefs. The reason we created these was to seperately create men and women and then easily add their prefs. This made the code easier to read but decreased the efficiency. However, we still kept the time complexity of $\mathcal{O}(n^2)$.

Overall time was 47s wit multiple programs running. Otherwise we've gotten it to 44s.

3 Oral questions

3.1 Why does your algorithm obtain a stable solution?

It's stable because of two main things. Firstly, the men start asking their most preferred woman until their least favorite. Contrary, the women can theoretically upgrade man until they meet their most preferred man. That's a balance. Best case

3.2 Could there be other stable solutions? Do you have an example/proof of uniqueness?

Obviously a switch between Man and Woman in the algo would make it. Otherwise a consideration would be to let the women

3.3 What is the time complexity and why?

3.4 Is this (the algorithm) how matching is performed in real life? If not, what flaws does it have?

3.5 Are there any applications of the algorithm (as it is or in a slightly shifted version)?