Stable Matching Report

Otto Holmström

March 31, 2020

1 Results

Running the script, all the results come out as correct. Using time logging in the code, the biggest file (5testhugemessy.in) took on average 2.12 seconds to run. The actual algorithm, which has been implemented in its own function, takes however only 7.2 ms to run on average. This indicates that the majority of the time in the program is spent on IO.

2 Implementation details

The Gale-Shapely algorithm was implemented using C++(11) in its on function GS. It takes 3 parameters; a map of all the men, a map of all the women and a map in which to store the final matchings. To store the men and women, C++11 $std :: unordered\_map$ was used for multiple reasons; $unordered\_map$ has on average constant time element access by the use of a hashmap implementation, and we can use the map datastructure to store the man/woman as a key and their preference list as a value.

The preference lists do not change and are thus implemented as simple integer arrays. In the solution the men propose and therefore we store the women directly in the men's preference lists, and the men indices in the women's preference list to allow for constant time comparaison if a woman should dump her current partner when a man proposes.

These two maps of type $< int, int* >$ are supplied to the GS-function, as well as a $std :: unordered\_map < int, int >$ in which we store the matchings. In the function we also create a list of type $std :: list < int >$ to keep track of the next man to propose, as well as a $std :: map < int, int >$ to store the index at which the men are in their personal preference list. Finally, the algorithm is implemented like the pseudocode in the lecture notes.

The time complexity overall is $O(n^2)$; the use of $unordered_map$ and $list$ means that the data structure operations all have constant time complexity and all comparaisons also have constant complexity; thus these don't impact the overall time complexity. Each man has n women in their preference list to propose to and we have in total n men; each loop iteration one man proposes and thus we have at most n^2 proposals, leading to a $O(n^2)$ time complexity.