University of Washington
Department of Computer Science and Engineering
CSE 421, Winter 2019

January 14, 2019

Homework 1, Due Wednesday, January 16, 1:29 pm, 2019

Turnin instructions: Electronics submission on canvas using the CSE 421 canvas site. Each numbered problem is to be turned in as a separate PDF, with subparts submitted together. E.g., Problems 2a and 2b should be turned in on the together.

Problem 1a (10 points):

Let $I = (M, W)$ be an instance of stable matching. Suppose that m is w's first choice and w is m's first choice in the ranked lists. Are m and w guaranteed to be matched in a solution? Justify your answer.

Problem 1b (10 points):

Let $I = (M, W)$ be an instance of stable matching. Suppose that m is w's last choice and w is m's last choice in the ranked lists. Is it possible for m and w to be matched in a solution? Justify your answer.

Problem 2a (10 points):

Prove that for every n, there is an instance of stable matching on sets M and W with $|M| = |W| = n$ where there is a unique stable matching.

Problem 2b (10 points):

Prove that the stable matching problem may have an exponential number of solutions. To be specific, show that for every n, there is an instance of stable matching on sets M and W with $|M| = |W| = n$ where there are at least c^n stable matchings. (There is a solution with $c = \sqrt{2}$, but you can use a different constant.)

Programming Problem 3a (15 points) :

Implement the stable matching algorithm. Write an input generator which creates completely random preference lists, so that each M has a random permutation of the W's for preference, and vice-versa. The goodness of a match for an individual can be measured by the position in the preference list of the match. The overall goodness for the M's would be the sum over each m, of his rank for the matching w. Similarly, the goodness for the W's can be defined.
You are free to write in any programming language you like. The quality of your algorithm may be graded (but you can use the one in the book), but the actual quality of the code will not be graded. The expectation is that you write the algorithmic code yourself - but you can use other code or libraries for supporting operations. You may use a library to generate random permutations (although this can be done as a four-line algorithm.) Submit your code as a PDF.
Make sure that you test your algorithm on small instance sizes, where you are able to check results by hand.

Programming Problem 3b (15 points) :

As the size of the problem increases - how does the goodness change for M and W? (It is probably easiest to normalize by dividing the goodness by n, the number of pairs.) Submit a write up about how the goodness varies with the input size based on your experiments. Can you determine the asymptotic growth rate? Is the result better for the M's or W's? You will probably need to run your algorithm on inputs with n at least 1,000 to get interesting results.

Problem 4:

The egg-drop problem is: You are given two eggs, and access to a 100-story building. Both eggs are identical. The aim is to find out the highest floor from which an egg will not break when dropped out of a window from that floor. If an egg is dropped and does not break, it is undamaged and can be dropped again. However, once an egg is broken, that's it for that egg. If an egg breaks when dropped from floor n, then it would also have broken from any floor above that. If an egg survives a fall, then it will survive any fall shorter than that. The question is: What strategy should you adopt to minimize the number egg drops it takes to find the solution?. (And what is the worst case for the number of drops it will take?)

Problem 4a (15 points):

To make this more precise, let F be a function from $\{1, \ldots, n\}$ to $\{0, 1\}$, where for some k, $F(j) = 0$ for $j \leq k$ and $F(j) = 1$ for $j > k$. Give an algorithm that finds the value of k with as few evaluations of F as possible with the restriction that at most two of the evaluations of F give the result of 1. Give an algorithm that is as efficient as possible and give the asymptotic (big-Oh) run time.

Problem 4b (5 points):

Generalize your algorithm to allow three evaluations of F to yield the value 1. Give an updated algorithm and run-time.