Homework 1 – Due Thursday, September 1, 2016 on Canvas

Homework Guidelines

Please make sure you read and sign the collaboration policy before you start working on your homework. (We will grade your homework only if we have a signed copy on file.) Refer to the general information handout for the homework policy and additional options.

Collaboration policy Collaboration on homework problems, with the exception of programming assignments and reading quizzes, is permitted, but not encouraged. If you choose to collaborate on some problems, you are allowed to discuss each problem with at most 2 other students currently enrolled in the class. Before working with others on a problem, you should think about it yourself for at least 45 minutes. Finding answers to problems on the Web or from other outside sources (these include anyone not enrolled in the class) is strictly forbidden.

You must write up each problem solution by yourself without assistance, even if you collaborate with others to solve the problem. You must also identify your collaborators. If you did not work with anyone, you should write "Collaborators: none." It is a violation of this policy to submit a problem solution that you cannot orally explain to an instructor or TA.

How to submit your solutions Each problem must be **typed** up separately (in at least an 11-point font) and submitted in the appropriate assignment box on the Canvas website as a PDF file. You are strongly encouraged, but not required, to format your problem solutions in LaTeX. Template HW files and other LaTeXresources are posted on the course webpage. LaTeXis a free, open-source scientific document preparation system. Most technical publications in CS are prepared using this tool.

You might want to acquire a LaTeX manual or find a good online source for LaTeX documentation. The top of each problem should include the following:

- your name,
- the question number, e.g., Problem 1-2 for "homework 1, problem 2",
- the names of all people you worked with on the problem (see the handout "Collaboration and Honesty Policy"), indicating for each person whether you gave help, received help or worked something out together, or "Collaborators: none" if you solved the problem completely alone.

Solution guidelines For problems that require you to provide an algorithm, you must give the following:

1. a precise description of the algorithm in English and, if helpful, pseudocode,
2. a proof of correctness,
3. an analysis of running time and space.

You may use algorithms from class as subroutines. You may also use any facts that we proved in class.

You should be as clear and concise as possible in your write-up of solutions. Understandability of your answer is as desirable as correctness, because communication of technical material is an important skill. A simple, direct analysis is worth more points than a convoluted one, both because it is simpler and less prone to error and because it is easier to read and understand. Points might be subtracted for illegible handwriting and for solutions that are too long. Incorrect solutions will get from 0 to 30% of the grade, depending on how far they are from a working solution. Correct solutions with possibly minor flaws will get 70 to 100%, depending on the flaws and clarity of the write up.

Assigned Problems

Exercises (Do not hand in) Chapter 1, Problems 1-3, 5. Chapter 2, Problems 3-5, 7.

Problems to be handed in, 10 points each

1. (**Resident Matching**, 2-page limit – your solutions should fit on two sides of 1 page) Chapter 1, problem 4. Please give a clear description of your algorithm. Don't forget to prove its correctness and analyze its time and space complexity.

2. (**Stable Matching**, 2-page limit)

   (a) (**Truthfulness in Stable Matching**) Chapter 1, problem 8.
   Hint: Try playing with several specific examples of preference lists.

   (b) (**Unique Matching**) Give an algorithm that takes an instance of the stable matching problem as input and decides if there is *exactly one* stable matching for this instance (that is, the algorithm outputs either "unique stable matching", or "more than one stable matching").
   Hint: You may want to read the section on "Extensions" at the end of KT Section 1.1.

3. (**Asymptotics**) Complete the Quiz "Homework 2 (Online component)" on Canvas.

4* (**Optional, no collaboration**, 2-page limit) You just got a job offer and you are trying to negotiate your salary. Possible salaries are integers from 1 to n. Your new company is willing to pay some (unknown to you) salary s or any salary lower than s. In one round of negotiations, you can name any integer g between 1 and n. If $g > s$, the company will say "too high". Otherwise, the company will accept g as your salary. Your goal is to ensure that you will be paid s, a fair salary you deserve.

   One way to ensure that you will get paid s is to name all integers, starting from n and going down by 1 in each subsequent round of negotiations, until the company accepts your proposed salary g. But if you follow this strategy, you might have to go through n rounds of negotiations.

   If you are allowed to change your mind (that is, ask for a higher salary) after the company accepted your offer, you might want to try binary search as your strategy. (You should know how many rounds this will take in the worst case, as a function of n.) However, it might be embarrassing to change your mind so many times.

   (a) Suppose you are allowed to change your mind exactly once. Describe a strategy for ensuring a fair salary, s, that uses $o(n)$ rounds of negotiation (as few as you can).

   (b) Now suppose you are allowed to change your mind k times, where $k > 1$. Describe a strategy for ensuring a fair salary, s, with as few negations as you can. Let $f_k(n)$ denote the number of rounds you use, as a function of n. (The answer from part (a) is your $f_1(n)$.) For each k, you should be able to get an asymptotically better solution than for $k-1$: that is, make sure that $f_k(n) = o(f_{k-1}(n))$.