Homework 1 – Due 10/22/2018 at midnight on Canvas

Homework Guidelines

Please make sure you read the collaboration policy, and write the following as the first line of your homework: "I have read and agree to the collaboration policy. ⟨`Your name`⟩." Your homework will not be graded if you do not write this.

Collaboration policy Collaboration on homework problems is permitted, but not encouraged. You are allowed to collaborate with *at most two students enrolled in the class*. You must mention the name of your collaborators clearly on the first page of your submission. Even if you collaborate, you are *expected to write and submit your own solution independent of others*, and your collaboration should be restricted to discussions only. Also, you should be able to explain your solution verbally to the course staff if required to do so. *Collaborating with any one not enrolled in the class, or taking help from any online resources for the homework problems is strictly forbidden.*

The Computer Science Department of UCSC has a zero tolerance policy for any incident of academic dishonesty. If cheating occurs, consequences within the context of the course may range from getting zero on a particular assignment, to failing the course. In addition, every case of academic dishonesty will be referred to the student's college Provost, who sets in motion an official disciplinary process. Cheating in any part of the course may lead to failing the course and suspension or dismissal from the university.

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.

This means that you will submit 4 separate files on Canvas, one for each problem!

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 acknowledgement to the collaboration policy,
- 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 (unless the problems explicitly mention that you don't need to):

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 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.

Following are the problems to be handed in. We will pick three problems out of four problems for grading. Each problem is worth 25 points.

1. (**Truthfulness**)

   We have discussed the stable matching problem at some length now, where the goal is to match a set of men and a set of women without any man-woman pair that would be happier with each other than the person with whom they were assigned. We have also seen a solution to this problem, the Gale-Shapley algorithm, which goes through unmatched men and match them to the highest woman on their preference list that would accept them.

   The setting in which we have been considering the problem assumes that the participants have been honest in their preference lists. However, what if they have not? Can they benefit from lying and get a person they prefer more, if we are running Gale-Shapley?

   More specifically, for a stable matching problem with 3 women and 3 men, let a woman be Alice and three men be Ben, Carl, and David, if Alice prefers Ben to Carl but lies that she prefers Carol to Ben, is it possible that she can be matched with David, whom she prefers to both of Ben and Carl?

   For this problem, either prove that for any set of preference lists, Alice cannot end up with David by lying about Ben and Carl, or provide a set of preference lists where she can. Please state clearly any assumptions you make that are related to your solution.

2. (**TA-course Matching**)

   There are m courses in CS department at UCSC, and each need a certain number of teaching assistants. There are n graduate students(applicants) who applied for TA positions this quarter. The instructor of each course has a preference list of all the applicants, and each applicant also has a preference list of all the courses. We assume that there are more applicants than the number of available TA positions for all the m courses.

We want to find a way of assigning each student as a TA to at most one course, in such a way that all the TA positions would be filled. (Since the number of the applicants is more than the number of positions, there would be some students who won't get assigned to any course.) We say that an assignment of students to courses is stable if neither of the following situations arises.

- The first type of instability that can occur is that there is a course c, and there are students s and s', so that
  - s is assigned to c, and
  - s' is assigned to no course, and
  - the instructor of c favors s' over s.
- The second type of instability that can occur is that there are two courses c,c' and two students s,s' so that
  - s is assigned to c, and
  - s' is assigned to c', and
  - the instructor of c favors s' over s, and
  - s' favors c over c'.

This is similar to the Stable Matching problem, except the courses generally need more than one TA, and the number of applicants is more than the number of available TA positions. Show that there is always a stable assignment of applicants to courses, and give an algorithm to find one. Please give a clear description of your algorithm. Don't forget to prove its correctness and analyze its time and space complexity.

3. (**Tokens in the bag**)

   Suppose we have a lot of tokens, and every token is either red, green or blue. We also have a bag with some tokens in it.

   Let's consider the following procedure:

   - Repeat the following until the bag is empty
     (1) If there are more than two tokens in the bag, take two random tokens out of the bag. Otherwise, empty the bag.
     (2) According the two tokens we got in step (1), we do the following things:
       * **Case 1:** If one of the tokens is red, do nothing.
       * **Case 2:** If both tokens are green, we put one green token and 2 blue tokens back into the bag.
       * **Case 3:** If we got one blue token, and the other token is not red, then we put 3 red tokens back into the bag.

   Assume that we always have enough tokens to put back into the bag, prove via induction that this process always terminates.

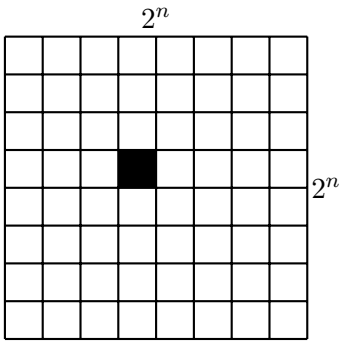
4. (**Perfectly imperfect masterpieces**)

   Bob is a young abstract artist. He wants his next series of works to express the idea of that many things in our life are imperfect, but we should embrace the beauty of them.

   He wants the works to be "perfectly imperfect" squares, which he defines as follows (Also see Figure 1:

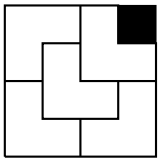
- (**Perfect**) The edge of the square is with length 2^n units where n can be any natural number.
- (**Imperfect**) The square must leave a 1×1 unit square blank.

He also wants the works to be composed of L-shaped 3-unit color blocks. (See Figure 1b). The figure below shows his "perfectly imperfect" ideas (in black and white):



(a) The imperfect perfect square with $n = 3$. The black unit square should be left blank.

(b) An L-shaped 3-unit color block



(c) One of the works in Bob's mind with $n = 2$

Figure 1

Can you design an efficient algorithm to help Bob design the alignment of the L-shaped blocks in the perfectly imperfect squares for any n? Your algorithm should run in time $O(2^{2n})$. You should consider each placement of the L-shaped block as $O(1)$ operation. Prove the time complexity of your algorithm. (For this problem, you don't have to analyze the space complexity.)