CS180 Homework 1

Due: April 19, 11:59pm

1. (15 pt) In the class we showed that the number of iterations (number of proposals) in the Gale-Shapley algorithm is upper bounded by $\leq n^2$ for n men and n women, since each man can only make $\leq n$ proposals. However, we haven't shown a lower bound on number of iterations. To show the lower bound is also in the order of n^2, please give a way to construct the preference lists for n men and n women such that the Gale-Shapley algorithm will run for $\Theta(n^2)$ iterations. (For simplicity, you can assume that the algorithm always chooses the unmatched man with the smallest index at each iteration).

2. (20 pt) Gale and Shapley published their paper on the Stable Matching Problem in 1962; but a version of their algorithm had already been in use for ten years by the National Resident Matching Program, for the problem of assigning medical residents to hospitals.

   Basically, the situation was the following. There were m hospitals, each with a certain number of available positions for hiring residents. There were n medical students graduating in a given year, each interested in joining one of the hospitals. Each hospital had a ranking of the students in order of preference, and each student had a ranking of the hospitals in order of preference. We will assume that there were more students graduating than there were slots available in the m hospitals.

   The interest, naturally, was in finding a way of assigning each student to at most one hospital, in such a way that all available positions in all hospitals were filled. (Since we are assuming a surplus of students, there would be some students who do not get assigned to any hospital.)

   We say that an assignment of students to hospitals is stable if neither of the following situations arises.

   - First type of instability: There are students s and s', and a hospital h, so that
     - s is assigned to h,and
     - s' is assigned to no hospital, and
     - h prefers s' to s.
   - Second type of instability: There are students s and s', and hospitals h and h', so that
     - s is assigned to h,and
     - s' is assigned to h', and
     - h prefers s' to s, and
     - s' prefers h to h'.

   So we basically have the Stable Matching Problem, except that (i) hospitals generally want more than one resident, and (ii) there is a surplus of medical students.

   Show that there is always a stable assignment of students to hospitals, and give an algorithm to find one.

3. (20 pt) The Stable Matching Problem, as discussed in the text, assumes that all men and women have a fully ordered list of preferences. In this problem we will consider a version of the problem in which men and women can be indifferent between certain options. As before we have a set M of n men and a set W of n women. Assume each man and each woman ranks the members of the opposite gender, but now we allow ties in the ranking. For example (with $n = 4$), a woman could say that m_1 is ranked in first place; second place is a tie between m_2 and m_3 (she has no preference between them); and m_4 is in last place. We will say that w prefers m to m' if m is ranked higher than m' on her preference list (they are not tied).

   With indifferences in the rankings, there could be two natural notions for stability. And for each, we can ask about the existence of stable matchings, as follows.

   (a) A *strong instability* in a perfect matching S consists of a man m and a woman w, such that each of m and w prefers the other to their partner in S. Does there always exist a perfect matching with no strong instability? Either give an example of a set of men and women with preference lists for which every perfect matching has a strong instability; or give an algorithm that is guaranteed to find a perfect matching with no strong instability.

(b) A *weak instability* in a perfect matching S consists of a man m and a woman w, such that their partners in S are w' and m' , respectively, and one of the following holds:

- m prefers $w$to w', and w either prefers m to m' or is indifferent between these two choices; or
- w prefers m to m', and m either prefers w to w' or is indifferent between these two choices.

In other words, the pairing between m and w is either preferred by both, or preferred by one while the other is indifferent. Does there always exist a perfect matching with no weak instability? Either give an example of a set of men and women with preference lists for which every perfect matching has a weak instability; or give an algorithm that is guaranteed to find a perfect matching with no weak instability.

4. (10 pt) Take the following list of functions and arrange them in ascending order of growth rate. That is, if function $g(n)$ immediately follows function $f(n)$ in your list, then it should be the case that $f(n)$ is $O(g(n))$.

- $f_1(n) = \frac{n^4}{\log n}$
- $f_2(n) = \sqrt{2n}$
- $f_3(n) = n^2/2$
- $f_4(n) = 2^{3\log n}$
- $f_5(n) = n(\log n)^{100}$
- $f_6(n) = 2^{\log n - \log\log n}$

5. (20 pt) You're doing some stress-testing on various models of glass jars to determine the height from which they can be dropped and still not break. The setup for this experiment, on a particular type of jar, is as follows. You have a ladder with n rungs, and you want to find the highest rung from which you can drop a copy of the jar and not have it break. We call this the highest safe rung.

It might be natural to try binary search: drop a jar from the middle rung, see if it breaks, and then recursively try from rung $n/4$ or $3n/4$ depending on the outcome. But this has the drawback that you could break a lot of jars in finding the answer.

If your primary goal were to conserve jars, on the other hand, you could try the following strategy. Start by dropping a jar from the first rung, then the second rung, and so forth, climbing one higher each time until the jar breaks. In this way, you only need a single jar – at the moment it breaks, you have the correct answer – but you may have to drop it n times (rather than log n as in the binary search solution). So here is the trade-off: it seems you can perform fewer drops if you're willing to break more jars. To understand better how this tradeoff works at a quantitative level, let's consider how to run this experiment given a fixed "budget" of $k \geq 1$ jars. In other words, you have to determine the correct answer – the highest safe rung – and can use at most k jars in doing so.

(a) Suppose you are given a budget of $k = 2$ jars. Describe a strategy for finding the highest safe rung that requires you to drop a jar at most $f(n)$ times, for some function $f(n)$ that grows slower than linearly. (In other words, it should be the case that $\lim_{n\to\infty} f(n)/n = 0$.)

(b) Now suppose you have a budget of $k > 2$ jars, for some given k. Describe a strategy for finding the highest safe rung using at most k jars. If $f_k(n)$ denotes the number of times you need to drop a jar according to your strategy, then the functions $f_1, f_2, f_3, \ldots$ should have the property that each grows asymptotically slower than the previous one: $\lim_{n\to\infty} f_k(n)/f_{k-1}(n) = 0$ for each k.

6. (15 pt) Given K sorted linked lists, each one contains n numbers in the ascending order. Give an algorithm to merge these K lists into a single sorted list. Your algorithm should run in $O(nK \log K)$ time.

★ Homework assignments are due on the exact time indicated. Please submit your homework using the Gradescope system. Email attachments or other electronic delivery methods are not acceptable. To learn how to use Gradescope, you can:

  - 1. Watch the one-minute video with complete instructions from here:
    https://www.youtube.com/watch?v=-wemznvGPfg
  - 2. Follow the instructions to generate a PDF scan of the assignments:
    http://gradescope-static-assets.s3-us-west-2.amazonaws.com/help/submitting_hw_guide.pdf
  - 3. **Make sure you start each problem on a new page.**

★ We recommend to use LaTeX, LyX or other word processing software for submitting the homework. This is not a requirement but it helps us to grade the homework and give feedback. For grading, we will take into account both the correctness and the clarity. Your answer are supposed to be in a simple and understandable manner. Sloppy answers are expected to receiver fewer points.

★ Unless specified, you should justify your algorithm with proof of correctness and time complexity.