CS7800: Advanced Algorithms. Fall 2016
Homework 1

Instructor: Jonathan Ullman, TA: Mehraneh Liaee

Due Friday, September 23 at 11:59pm
(Email to m.liaee2050+CS7800@gmail.com)

- Problems 1-3 are meant as a review of undergraduate discrete math and algorithms. They shouldn't take you too long, but I recommend starting these right away to make sure that you have the appropriate background for this course.
- You must type your solutions using LaTeX. Please submit both the source and PDF files using the naming conventions `lastname_hw1.tex` and `lastname_hw1.pdf`.
- Strive for clarity and conciseness in your solutions, emphasizing the main ideas over low-level details. I recommend looking at the introduction in Jeff Erickson's textbook for advice on writing up solutions to algorithms problems.
- Do not share written solutions, and remember to cite all collaborators and sources of ideas. Sharing written solutions, and getting solutions from outside sources such as the Web or students not enrolled in the class is strictly forbidden.

Review Problems

Problem 1 (Review of Asymptotic Growth)**.** Arrange the following list of functions 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 $O(g(n))$. (You do not need to provide proofs.)

$$\begin{array}{lll} f_1(n) = 4n^2 + n\log_8^2(n) & f_2(n) = 2^{n^2} & f_3(n) = \sqrt{1024n\log_2(n)} \\ f_4(n) = n + 40\sqrt{n} & f_5(n) = 10^n & f_6(n) = 3n\log_2(n) \\ f_7(n) = n^2\log_8 n & f_8(n) = 4096\log_2^4(n) & f_9(n) = n^{\log_2(3)} \end{array}$$

Problem 2 (Review of Basic Proof Techniques)**.**

(a) Prove the following statement by induction: Given an unlimited supply of 4 and 7 cent coins, one can make exact change for any amount greater or equal to 20 cents.

(b) Prove the following statement by contradiction: Let G be a simple graph on $2n$ nodes with no self-loops (i.e. there are no edges connecting a node to itself and there is at most one edge connecting any pair of nodes). Prove by contradiction that if every node in G has degree at least n, then G is a connected graph.

Problem 3 (Review of Basic Algorithms)**.** You are buying a house in Brookline. Since you like to be known for finding good deals, it's crucial to you that you find a house that is cheaper than each of your neighbors to the left and right. Unfortunately, the Brookline housing market is out of control[1], so you can't know the price of a house until you visit its open house and scout out the other interested buyers. Since you don't have time to go to every open house, you will need to carefully select which open houses to attend in order to find an affordable house.

More formally, there are n houses with prices $p_1, \dots, p_n$. A *locally cheapest house* is a house i such that $p_i \leq p_{i+1}$ and $p_i \leq p_{i-1}$. If $i = 1$ then we only require $p_1 \leq p_2$ and if $i = n$ then we only require $p_n \leq p_{n-1}$. We will represent the process of going to an open house and scouting out the competition as a function RevealPrice(i) that takes as input $1 \leq i \leq n$ and returns p_i.

Design an algorithm that finds a *locally cheapest house* using at most $O(\log(n))$ calls to RevealPrice. Give a clear description of your algorithm, prove that it correctly finds a locally cheapest house, and analyze its running time.

Stable Matching

NB: In class we used the doctors-hospitals metaphor to discuss stable matching. I am going to continue to use that in this problem. Note that Kleinberg-Tardos uses the men-women metaphor and thus different notation (e.g. m and w instead of d and h) but other than these superficial differences, the two problems are exactly the same.

Problem 4 (Stable Matching)**.** When we formulated the stable matching problem in class, we assumed that all doctors and hospitals have strict preferences. Suppose that hospitals and doctors are allowed to have *indifferences*. For example, some doctor d could have preferences $h_1 \succ_d (h_2 \sim_d h_3) \succ_d h_4$, meaning that h_1 is most preferred, second place is a tie between h_2 and h_3, and last place is h_4. With indifferences there are two natural notions of instability.

[1] Now you all know how I spent the summer.

(a) A *strong instability* in a perfect matching μ consists of two doctor-hospital pairs (d,h) and (d',h') in μ such that 1) d strictly prefers h' to h and 2) h' strictly prefers d to d'. Does there always exist a perfect matching with no strong instabilities? Either give an example of a set of doctors and hospitals 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 instabilities.

(b) A *weak instability* in a perfect matching μ consists of two doctor-hospital pairs (d,h) and (d',h') in μ such that 1) d prefers h' to h or is indifferent, 2) h' prefers d to d' or is indifferent, and 3) at least one of the two preferences in the first to conditions is strict. Does there always exist a perfect matching with no weak instabilities? As above, either give an algorithmic proof or construct a counterexample.

Greedy Algorithms

Problem 5 (Greedy Algorithms: Interval Subcovers)**.**

A *cover* of the unit interval $[0,1]$ is a collection of half-open intervals $C = \{[a_1,b_1),\dots,[a_m,b_m)\}$ such that $(0,1) \subseteq \bigcup_{i=1}^{m}[a_i,b_i)$. In other words, every point $x \in [0,1]$ is contained in some interval $[a_i,b_i) \in C$.[2] Given a collection of half-open intervals $I = \{[a_1,b_1),\dots,[a_n,b_n)\}$, our goal is to find the *smallest cover C of* $[0,1]$ *contained in I*. For example, given intervals $I = \{[0,.6),[.2,.8),[.4,1.1)\}$ the set of intervals $C = \{[0,.6),[.4,1.1)\}$ is a cover of $[0,1]$ and a subset of I, and because no single interval I is a cover of $[0,1]$, C is also a smallest cover in I.

In this problem you will design an efficient greedy algorithm that takes a collection of n intervals I and finds a smallest cover in I or reports that none exists. For simplicity, you may assume that all endpoints $a_1,b_1,\dots,a_n,b_n$ are distinct.

(a) Consider the following greedy algorithm: at every step, add the longest interval that is not already covered by the intervals selected so far. Give an example of a collection of intervals where this approach fails to find the smallest cover.

(b) Give a polynomial time greedy algorithm that takes the endpoints $a_1,b_1,\dots,a_n,b_n$ as input and returns the smallest cover. Faster algorithms will receive slightly more credit than slower algorithms.

(c) What is the running time of your algorithm?

(d) Prove that your algorithm always finds the minimum cover. You may find it helpful to read to the end of this series of questions before you begin.

(e) Given a collection of intervals I, a *spread* for I is a set points contained in $[0,1]$ such that each point is contained in some interval in I but no interval in I contains two of the points. Prove that if there exists a spread of size k for I, then every cover of $[0,1]$ contained in I must have size at least k. For example, $\{.1,.9\}$ is a spread of size 2 for the example at the beginning of this question.

(f) Prove that for every collection of intervals I, the size of the largest spread is equal to the size of the smallest cover of $[0,1]$ contained in I. Thus, the *largest spread problem* is "dual" to the *smallest subcover problem*.[3]

[2] Using half-open intervals that don't include the right endpoint b_i is just to avoid certain technical nuisances.

[3] Hint: Modify the greedy algorithm from part **(b)** to produce a cover and a spread of the same size.