CS 170
Fall 2017

Efficient Algorithms and Intractable Problems
Prasad Raghavendra and Sanjam Garg

Homework 2

Instructions: You are welcome to form small groups (up to 4 people total) to work through the homework, but you **must** write up all solutions by yourself. List your study partners for homework on the first page, or "none" if you had no partners.

If using LaTeX (which we recommend), you may use the homework template linked on this Piazza post to get started.

Begin each problem on a new page. Clearly label where each problem and subproblem begin. The problems must be submitted in order (all of P1 must be before P2, etc). For questions asking you to give an algorithm, respond in what we will refer to as the *four-part format* for algorithms: main idea, pseudocode, proof of correctness, and running time analysis.

Read the Homework FAQ Piazza post on Piazza before doing the homework for more explanation on the four-part format and other clarifications for our homework expectations.

No late homeworks will be accepted. No exceptions. This is not out of a desire to be harsh, but rather out of fairness to all students in this large course. Out of a total of approximately 12 homework assignments, the lowest two scores will be dropped.

Special Questions:

- *Shortcut questions*: Short questions are usually easy questions that give you opportunities to practice basic materials. However, we understand that some of you are very familiar with the topics and do not want to spend too much time on easy questions. Therefore, we design shortcut questions for this purpose. A shortcut question usually has multiple parts that build upon each other and are ordered by their difficulty level. You can work on those in order or start from wherever you like. However you only need to submit the last part you are able to solve. For example, if a question has 5 parts (a, b, c, d, e), you are confident about part e, you should submit part e without any of the previous four parts. If you are confident about d but not sure about e, you should submit d for grading purposes. Please clearly indicate in your submission which part you are submitting.
- *Redemption questions*: It is important that you carefully read the posted solutions, even for problems you got right. To encourage this, you have the option of submitting a redemption file, a few paragraphs in which you explain, for each problem you choose to cover, what you did wrong and what the right idea was in your own words (not cutting and pasting from the solution!), and appending it to your homework. For example, suppose that as you review your solutions to HW1, you realize you had misunderstood question 3 and answered it incorrectly. You would write down what you just learned, and then submit it in your HW2 assignment the following week. Because these are mainly for your benefit, feel free to format them however is most useful for you.
- *Extra credit questions*: We might have some extra credit questions in the homework for people who really enjoy the materials. However, please note that you should do the extra credit problems only if you really enjoy working on these problems and want an extra challenge. It is likely not the most efficient manner in which to maximize your score.

Due Wednesday, September 13, at 4:59pm

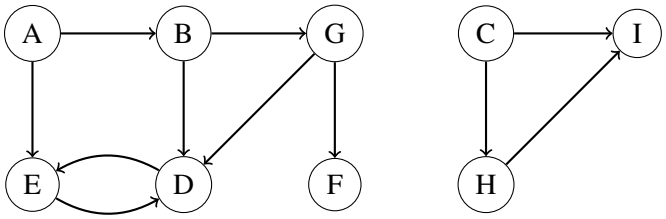
1. (★★ level) Minimal Positive Valued Function

Consider a strictly decreasing function $f : \mathbb{N} \to \mathbb{Z}$, such that $f(i) > f(i+1)$ for all $i \in \mathbb{N}$. Assuming we can evaluate f at any number in constant time, we want to find $n = \min\{i \in \mathbb{N} : f(i) < 0\}$. Design a $O(\log n)$ algorithm to compute n.

(Please turn in a four part solution to this problem.)

2. (★★ level) Graph Basics

For parts (a) and (b), refer to the figure below. For parts (c) through (f), please prove only for simple graphs; that is, graphs that do not have any parallel edges or self-loops.



(a) Run DFS at node A, trying to visit nodes alphabetically (e.g. given a choice between nodes D and F, visit D first).

- List the nodes in the order you visit them (so each node should appear in the ordering exactly once).
- List each node with its pre- and post-number. The numbering starts from 1 and ends at 18.
- Label each edge as **T**ree, **B**ack, **F**orward or **C**ross.

(b) Let $|E|$ be the number of edges in a simple graph and $|V|$ be the number of vertices. Show that $|E|$ is in $O(|V|^2)$.

(c) For each vertex v_i, let d_i be the *degree*- the number of edges incident to it. Show that $\sum d_i$ must be even.

3. (★★★ level) Peak element

Prof. Garg just moved to Berkeley and would like to buy a house with a view on Euclid Ave. Needless to say, there are n houses on Euclid Ave, they are arranged in a single row, and have distinct heights. A house "has a view" if it is taller than its neighbors. For example, if the houses heights were $[3, 7, 5, 9]$, then 7 and 9 would "have a view".

Devise an efficient algorithm to help Prof. Garg find a house with a view on Euclid Ave. (If there are multiple such houses, you may return any of them.)

4. (★★★ level) Exact Change

Your friend from Mars visits you and wants to buy a CS 170 textbook, which is priced at $\$t$ dollars. Unfortunately, Martians have their own currency. You are given an array $A[0..n-1]$ with n elements representing the value of each Martian coin in dollars. The value of each coin is a distinct integer in the range $0 \leq A[i] \leq 170n$. For some reason, your friend brought only 4 coins of each type.

Design an efficient algorithm that determines, given A and t, whether your friend can give you exact change **using exactly 4 coins** (no more or no less). Note: the algorithm should run asymptotically faster than $O(n^2)$.

(Please turn in a four part solution to this problem.)
Hint: Think about properties of exponents.

5. (★★★★ level) Local Maxima

Consider an $n \times n$ matrix M with distinct integer entries. Call an entry M_{ij} a *local maximum* if it is greater than all of its neighbors. More precisely, M_{ij} is a local maximum if for all a, b with $|i-a| \leq 1$ and $|j-b| \leq 1$, $M_{ij} \geq M_{ab}$. In an array A, say that A_i is a *local maximum* of A is $A_i \geq A_{i+1}$ and $A_i \geq A_{i-1}$. Note that if i is the last element in A, then $A_i \geq A_{i-1}$ is sufficient for A_i to be local maximum, and similarly if i is the first element, then $A_i \geq A_{i+1}$ is sufficient.

Suppose that M is guaranteed to have exactly one local maximum and that every column of M, when viewed as an array, contains exactly one local maximum. Show that the local maximum of M can be found in $O(\log^2 n)$ time.

(Please give a four part solution for this problem.)

6. (★★★★★ level) DNA Sequence Alignment

We are given binary strings s, t; s is m bits long, and t is n bits long, and $m < n$. We are also given an integer k. We want to find whether s occurs as a substring of t, but with $\leq k$ errors, and if so, find all such matches. In other words, we want to determine whether there exists an index i such that $s_0, s_1, \ldots, s_{m-1}$ agrees with $t_i, t_{i+1}, t_{i+2}, \ldots, t_{i+m-1}$ in all but k bits; and if yes, find all such indices i.

(a) Describe an $O(mn)$ time algorithm for this string matching problem. Just show the pseudocode; you don't need to give a proof of correctness or show the running time.

(b) Let's work towards a faster algorithm. Suggest a way to choose polynomials $p(x), q(x)$ of degree $m-1, n-1$, respectively, with the following property: the coefficient of x^{m-1+i} in $p(x)q(x)$ is $m - 2d(i)$, where $d(i)$ is the number of bits that differ between $s_0, s_1, \ldots, s_{m-1}$ and $t_i, t_{i+1}, t_{i+2}, \ldots, t_{i+m-1}$.

Hint: use coefficients $+1$ and -1.

(c) Describe an $O(n \lg n)$ time algorithm for this string matching problem, taking advantage of the polynomials $p(x), q(x)$ from part (b).

(d) Now imagine that s, t are not binary strings, but DNA sequences: each position is either A, C, G, or T (rather than 0 or 1). As before, we want to check whether s matches any substring of t with $\leq k$ errors (i.e., $s_0, s_1, \ldots, s_{m-1}$ agrees with $t_i, t_{i+1}, t_{i+2}, \ldots, t_{i+m-1}$ in all but k letters), and if so, output the location of all such matches. Describe an $O(n \lg n)$ time algorithm for this problem.

Hint: encode each letter into 4 bits.

7. (??? **level)** **(Optional) Redemption for Homework 1**

Submit your *redemption file* for Homework 1 on Gradescope. If you looked at the solutions and took notes on what you learned or what you got wrong, include them in the redemption file.