CS 170
Spring 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:

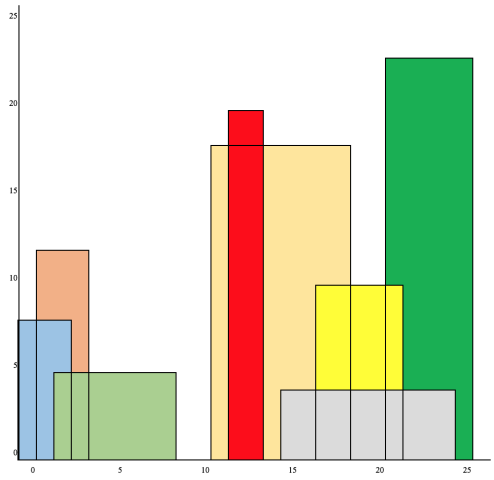
- *Shortcut questions*: Short questions are usually easy questions that give you opportunities to practice basic materials. However, we understand that some of you are very familiar with the topics and do not want to spend too much time on easy questions. Therefore, we design shortcut questions for this 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.

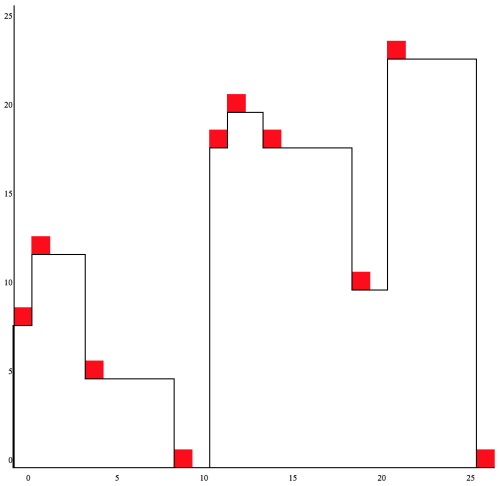
Due Tuesday, February 7, at 11:59am

1. (★★★★ level) Million Dollar View

A $70 million penthouse has recently been listed for sale in San Francisco. What used to be a perfect view of the Golden Gate Bridge has been marred by a recent development of towers and tech office buildings. Your job is to determine if the penthouse is worth the price, by obtaining the skyscrapers' outline.

Suppose you are given the left and right positions of each of the buildings as well as the height in an array A i.e. $[(lt_1, rt_1, ht_1), (lt_2, rt_2, ht_2), \cdots, (lt_n, rt_n, ht_n)]$. The left and right positions are on the x-axis and the height is on y-axis. The output will be the key coordinates of the outline. These key coordinates are left points of the horizontal lines that form the outline. Thus we want the left point of the line as the x coordinate and the height as the y coordinate, i.e. $[(lt1, ht1), (lt2, ht2)...]$. Input building positions are unsorted. See below for an example.





Input buildings: $[(0,3,8),(1,4,12),(2,9,5),(11,19,18),(12,14,20),(15,25,4),(17,22,10),(21,26,23)]$
Output blocks: $[(0,8),(1,12),(4,5),(9,0),(11,18),(12,20),(14,18),(19,10),(21,23),(26,0)]$
Design an efficient algorithm to compute the outline.
Assume that the base of buildings can get arbitrarily larger than n.
$W = \max(rt_i - lt_i) \quad \forall i \in \{1,\ldots,n\}, W \geq n.$

2. (★★★ level) Majority Elements

An array $A[1\ldots n]$ is said to have a *majority element* if more than half of its entries are the same. Given an array, the task is to design an efficient algorithm to tell whether the array has a majority element, and, if so, to find that element. The elements of the array are not necessarily from some ordered domain like the integers, and so there can be **no** comparisons of the form "is $A[i] > A[j]$?". (Think of the array elements as GIF files, say.) However you *can* answer questions of the form: "is $A[i] = A[j]$?" in constant time.

Can you give a linear-time algorithm?

(Hint: Here's a divide-and-conquer approach:

- Pair up the elements of A arbitrarily, to get about $n/2$ pairs
- Look at each pair: if the two elements are different, discard both of them; if they are the same, keep just one of them
- If $|A|$ is odd, there will be one unpaired element. What should you do with this element?

Show that after this procedure there are at most $n/2$ elements left, and that they have a majority element if A does.)

Note: for this problem, assume you don't have access to a good hash function – i.e. perhaps it takes $O(n)$ to hash an item. In the real world, this is usually not a restriction you need to deal with; but for the pedagogical purposes of this problem assume you can only check for equality of the items quickly.

3. (★★★ level) Triple Sum

Design an efficient algorithm for the following problem. We are given an array $A[0..n-1]$ with n elements, where each element of A is an integer in the range $0 \leq A[i] \leq 10n$. We are also given an integer t. The algorithm must answer the following yes-or-no question: do there exist indices i, j, k such that $A[i]+A[j]+A[k] = t$?

Design an $O(n \log n)$ time algorithm for this problem.

Hint: define a polynomial of degree $O(n)$ based upon A, then use FFT for fast polynomial multiplication.

4. (★★★★ level) Shortcut Question: Sherlock

Sherlock Holmes is trying to write a computer antivirus program. He starts by modeling his problem. He thinks of computer RAM as being a binary string s_2 of length m. He thinks of a virus as being a binary string s_1 of length $n < m$. His program needs to find all occurrences of s_1 in s_2 in order to get rid of the virus. Even worse, though, these viruses are still damaging if they differ slightly from s_1. So he wants to find all copies of s_1 in s_2 that differ in at most k locations for arbitrary $k \leq n$.

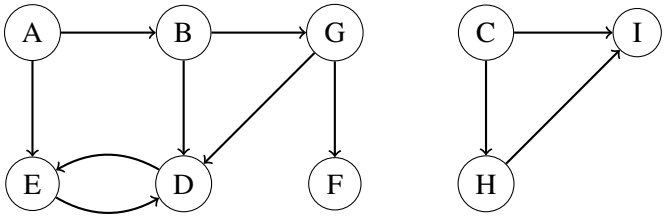
(a) Find an $O(nm)$ algorithm for this problem. Give only the main idea; no four-part solution necessary.

(b) Find polynomials p_1 of degree $n-1$ and p_2 of degree $m-1$ such that the coefficient of x^{n-1+i} of $p_1 p_2$ is $n-2u(i)$ where $u(i)$ is the number of mismatched bits between s_1 shifted by i and s_2. Show how to generate p_1 and p_2 in $O(n+m)$ time.

(c) Give the main idea and runtime for an $O(m \log m)$ algorithm for solving the problem for any k. Again, no four-part solution is necessary, but please clearly describe the algorithm and give pseudocode if you think it will enhance our understanding of your solution.

5. (★★★ 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) How many strongly connected components are there?

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

(d) 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.

(e) An undirected graph G is called *bipartite* if we can separate its vertices into two subsets A and B such that every edge in G must cross between A and B. Show that a graph is bipartite if and only if it has no odd cycles.
Hint: Consider a *spanning tree* of the graph, which is a subset of the graph's edges which allow it to be a tree on all of its vertices.

(f) A directed acyclic graph G is *semiconnected* if for any vertices A and B there is either a path from A to B or a path from B to A. Show that G is semiconnected if and only if there is a directed path that visits all of the vertices of G.

Optional redemption file

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