CS 170
Spring 2017

Efficient Algorithms and Intractable Problems
Prasad Raghavendra and Sanjam Garg

Homework 6

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 March 14, at 11:59am

This homework emphasizes dynamic programming problems. In your solutions, note the following:

- When you give the main idea for a dynamic programming solution, you need to explicitly write out a recurrence relation and explain its interpretation.
- When you give the pseudocode for a dynamic programming solution, it is not sufficient to simply state "solve using memoization" or the like; your pseudocode should explain how results are stored.

1. (★★★★★ level) Hacking for Justice (shortcut question)

In an alternate universe, the students of CS170 found a certain problem on HW6 to be extremely difficult. Initially, no one was able to find the solution. However, some subset of the students managed to download the solution PDF to their laptops. These students began to send the PDF to others via email, who then sent the PDF to others, and so on. Eventually, all of the students had the solution PDF. Uh oh!

After much effort, a TA has figured out the full history of the solution PDF sharing, and constructed a directed acyclic graph $G = (V, E)$ to represent it. V represents the students, and an edge E from v_1 to v_2 represents that v_1 sent the solution file to v_2. All sharing is one-way, and you know that there are no cycles.

If the TA could go back in time, and completely block off all communications to/from one student's laptop, which student should be blocked to minimize the number of students who received the PDF? Assume that blocking one person will not cause anyone else to share with more people than they did before. Answer this question in part (c), or try (a) and (b) for inspiration.

One more important detail: the TA notices that for any two different paths from A to B, the two paths differ by at most k vertices, where k is a small number. You can assume k is about equal to $\log(|V| + |E|)$.

A useful bound: the TA also notices that if you take a depth-k BFS search starting at any vertex, you will traverse $O(k)$ vertices in the search, and the BFS search runs in $O(k)$ time. Using this property will make the problem easier.

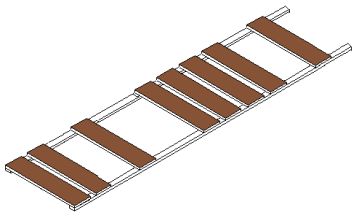
(a) **(10% credit)** Is it always the case that the best student to block is one of the original students who first downloaded the PDF? If so, explain why in 2-3 sentences. If not, give a small counterexample.

(b) **(50% credit)** Write a recursive definition for $F(s)$, the number of students that won't have the PDF if you choose to block s.

(c) **(100% credit)** Design an algorithm to determine which student to block off. Give a 4-part solution.

2. (★★★★ level) Bridge Hop

You notice a bridge constructed of a single row of planks. Originally there had been n planks; unfortunately, some of them are now missing, and you're no longer sure if you can make it to the other side. For convenience, you define an array $V[1..n]$ so that $V[i]$ = TRUE iff the ith plank is present. You're at one side of the bridge, standing still; in other words, your *hop length* is 0 planks. Your bridge-hopping skills are as follows: with each hop, you can increase or decrease your hop length by 1, or keep it constant.



For example, the image above has planks at indices [1, 2, 4, 7, 8, 9, 10, 12], and you could get to the other side with the following hops: [0, 1, 2, 4, 7, 10, 12, 14].

Clarifications: You start at location 0, just before the first plank. Arriving at any location greater than n means you've successfully crossed. Due to your winged shoes, there is no maximum hop length. But you can only hop forward (hop length cannot be negative).

Devise an efficient algorithm to determine whether or not you can make it to the other side.

For this problem, you should know how to do the proof of correctness, but need not include it in your submission. You should submit the main idea, pseudocode, and runtime.

3. (★★★★ level) Longest Palindrome Substring A substring is *palindromic* if it is the same whether read left to right or right to left. For example, "bob" and "racecar" are palindromes, but "cat" is not. Devise an algorithm that takes a sequence $x[1..n]$ and returns the length of the longest palindromic substring. Its running time should be $O(n^2)$.

For this problem, you should know how to do the proof of correctness, but need not include it in your submission. You should submit the main idea, pseudocode, and runtime.

4. (★★★★ level) A Sisyphean Task

Suppose that you have n boulders, each with a positive integer weight w_i. You'd like to determine if there is any set of boulders that together weight exactly k pounds. You may want to review the solution to the Knapsack Problem for inspiration.

For this problem, you should know how to do the proof of correctness, but need not include it in your submission. You should submit the main idea, pseudocode, and runtime.

(a) Design an algorithm to do this.

(b) Is your algorithm polynomial in the *size* of the input? Remember that size is in terms of how many bits we need.

5. (★★★★★ level) Advertising network

You are an advertising network tasked with making bids on Facebook's mobile ad units for your customers. On each day, you make a bid of $\$\theta$, $\theta \in [0,1]$, and win that day's auction with probability θ. If you don't win, then you don't spend any money. As per your contract with your customers, you *need* to win at least k out of n auctions that will occur this fiscal year.

An obvious approach may be to bid \$1 for each of the first k days, then nothing for the rest of the $n-k$ days – this strategy will cost you $\$k$. However, it may not be optimal! Consider this alternative for $n = 30, k = 4$:

- Bid \$0.50 for each of the first 26 days, or until you've won 4 auctions.
- Bid \$1.00 for each of the next 4 days, if you didn't win them yet.
- This strategy will have expected cost \$2.00[1], and worst case cost \$4.00 – a strictly superior strategy.

Give an optimal strategy to minimize expected cost while maintaining your contract, for any k, n. You only need to explain the main idea; no need for proof, runtime, or pseudocode.

Here are some hints:

- You'll need to use probability; specifically, the linearity of expectation [2].

[1] Rounded to the nearest cent.

[2] https://en.wikipedia.org/wiki/Expected_value#Linearity

- Parameterize your strategy as a set of variables $\theta_{n,k}$, and notice that you need to minimize some function that can be written in terms of $\theta_{n,k}$.
- To actually solve said optimization problem, you can assume you have access to a quadratic program solver that can minimize any quadratic function of a single variable, in time that is efficient.

6. (★★★★★★ level) Knightmare (extra credit question)

Give an algorithm to find the number of ways you can place knights on an N by M chessboard such that no two knights can attack each other [3] (there can be any number of knights on the board, including zero knights). The runtime should be $O(2^{3M} \cdot N)$ (or symmetrically, switch the variables).

Note that even though this question is extra credit (and thus only worth 1 pt), the staff *strongly recommends* that you at least attempt it and understand the solution. It will be *very* helpful practice for your upcoming midterm, which we remind you will be Monday March 20, 2017.

[3] Knights attack according to https://en.wikipedia.org/wiki/Knight_(chess)#Movement