CS 170 Spring 2017 | Efficient Algorithms and Intractable Problems | Prasad Raghavendra and Sanjam Garg

Homework 8

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*: Shortcut 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 try to 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 April 11, at 11:59am

1. (★★ level) Salt

Salt is an extremely valuable commodity. There are m producers and n consumers, each with their own supply $[a_1...a_m]$ and demand $[b_1...b_n]$ of salt.

Note: Solve parts $(b), (c)$ independently of each other.

(a) Each producer can supply to any consumer they choose. Find an efficient algorithm to determine whether it is feasible for all demand to be met.

(b) Each producer is willing to deliver to consumers at most c_i distance away. Each producer i has a distance $d_{i,j}$ from consumer j. Solve part (a) with this additional constraint.

(c) Each producer and consumer now belongs to one of p different countries. Each country has a maximum limits on the amount of salt that can be imported (e_k) or exported (f_k). Deliveries within the same country don't contribute towards this limit. Solve part (a) with this additional constraint.

2. (★★★★ level) Minimum Cost Flows

In the max flow problem, we just wanted to see how much flow we could send between a source and a sink. But in general, we would like to model the fact that shipping flow takes money. More precisely, we are given a directed graph G with source s, sink t, costs l_e, capacities c_e, and a flow value F. We want to find a nonnegative flow f with minimum cost, that is $\sum_e l_e f_e$ that respects the capacities and ships F units of flow from s to t.

(a) Show how the minimum cost flow problem can be solved in polynomial time.

(b) Show how a minimum cost flow solver can solve shortest path problems.

(c) Show how a minimum cost flow solver can solve maximum flow problems.

3. (★★★★ level) Minimum Spanning Trees

Consider the spanning tree problem, where we are given an undirected graph G with edge weights $w_{u,v}$ for every pair of vertices u, v.

An integer linear program that solves the minimum spanning tree problem is as follows:

$$\begin{aligned}
&\text{Minimize} \sum_{(u,v)\in E} w_{u,v}x_{u,v} \\
&\text{subject to} \sum_{\{u,v\}\in E: u\in L, v\in R} x_{u,v} \geq 1 \quad \text{for all partitions of } V \text{ into disjoint nonempty sets } L, R \\
&\qquad x_{u,v} \in \{0,1\}, \quad \forall (u,v) \in E
\end{aligned}$$

(a) Give an interpretation of the purpose of the objective function, decision variables, and constraints.

(b) Is creating the formulation a polynomial time algorithm with respect to the size of the input graph?

(c) Suppose that we relaxed the binary constraint on the decision variables $x_{u,v}$ with the non-negativity constraint:

$$x_{u,v} \geq 0, \quad \forall (u,v) \in E$$

How does the new linear program solution's objective value compare to the integer linear program's? Provide an example (decision variables and objective value) where the above LP relaxation achieves a better objective value than the ILP formulation.

4. (★★★ level) Major Key

You are a locksmith tasked with producing keys $k_1, ..., k_n$ that sell for $p_1, ..., p_n$ respectively. Each key k_i takes g_i grams of gold and s_i grams of silver. You have a total of G gold and S silver to work with, and can produce as many keys of any type as you want within the time and material constraints.

(a) Unfortunately, integer linear programming is an NP-complete problem. Fortunately, you have found someone to instead buy the alloys at an equivalent price! Instead of selling keys, you have decided to focus on melting the prerequisite metals together, and selling the mixture. Formulate the linear program to maximize the profit of the locksmith, and explain your decision variables, objective function, and constraints.

(b) Formulate the dual of the linear program from part (a), and explain your decision variables, objective function, and constraints. The explanations provide economic intuition behind the dual. We will only be grading the dual formulation.
Hint: Formulate the dual first, then think about it from the perspective of the locksmith when negotiating prices for buying G gold and S silver if they had already signed a contract for the prices for the output alloys p_i. Think about the breakeven point, from which the locksmith's operations begin to become profitable for at least one alloy.

5. (★★★ level) Zero-Sum Battle

Two Pokemon trainers are about to engage in battle! Each trainer has 3 Pokemon, each of a single, unique type. They each must choose which Pokemon to send out first. Of course each trainer's advantage in battle depends not only on their own Pokemon, but on which Pokemon their opponent sends out.

The table below indicates the competitive advantage (payoff) Trainer A would gain (and Trainer B would lose). For example, if Trainer B chooses the fire Pokemon and Trainer A chooses the rock Pokemon, Trainer A would have payoff 2.

		Trainer B: ice	grass	fire
Trainer A:	dragon	-10	3	3
	steel	4	-1	-3
	rock	6	-9	2

Feel free to use an online LP solver to solve your LPs in this problem.
Here is an example of a Python LP Solver and its Tutorial.

(a) Write an LP to find the optimal strategy for Trainer A. What is the optimal strategy and expected payoff?

(b) Now do the same for Trainer B. What is the optimal strategy and expected payoff?

6. (★★★ level) Decision vs. Search vs. Optimization

The following are three formulations of the VERTEX COVER problem:

- As a *decision problem*: Given a graph G, return TRUE if it has a vertex cover of size at most b, and FALSE otherwise.
- As a *search problem*: Given a graph G, find a vertex cover of size at most b (that is, return the actual vertices), or report that none exists.
- As an *optimization problem*: Given a graph G, find a minimum vertex cover.

At first glance, it may seem that search should be harder than decision, and that optimization should be even harder. We will show that if any one can be solved in polynomial time, so can the others:

Describe your algorithms precisely; justify correctness and running time. No pseudocode.

Hint for both parts: Call the black box more than once.

(a) Suppose you are handed a black box that solves VERTEX COVER (DECISION) in polynomial time. Give an algorithm that solves VERTEX COVER (SEARCH) in polynomial time.

(b) Similarly, suppose we know how to solve VERTEX COVER (SEARCH) in polynomial time. Give an algorithm that solves VERTEX COVER (OPTIMIZATION) in polynomial time.