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

Homework 7

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 4, at 11:59am

1. (★ level) Linear Programming Fundamentals

For each of the following optimization problems, is there a finite optimal solution that can be found by an LP solver? If the answer is no, identify the reason why (i.e. unbounded, infeasible, or not a linear program).

(a)

$$\max 5x + 3y$$

$$\begin{aligned} x^2 + y &\leq 45 \\ x &\leq 7 \\ x, y &\geq 0 \end{aligned}$$

(b)

$$\max 8x + \frac{1}{13}y + z$$

$$\begin{aligned} x &\leq 5 \\ 4x + 3z &\leq 9 \\ x, y, z &\geq 0 \end{aligned}$$

(c)

$$\max 3x + 3y$$

$$\begin{aligned} 5x - 2y &\geq 0 \\ 2x &\leq 18 \\ x, y &\geq 0 \end{aligned}$$

(d)

$$\max 2x + 2y$$

$$\begin{aligned} x + 2y &\leq 12 \\ x &\leq 6 \\ x + y &\geq 10 \\ x, y &\geq 0 \end{aligned}$$

(e)

$$\max 2x - 3y$$

$$\begin{aligned} 3x + 5y &\geq 13 \\ x &\leq 4 \\ x, y &\geq 0 \end{aligned}$$

2. (★★★ level) Matches for tutoring

A tutoring service has contracted you to work on pairing tutors with tutees. You are given a set of tutors U and a set of tutees V. Everyone has filled out some questionnaires, so you know which tutors are compatible with which tutees (i.e. able to tutor the right subject). Additionally, each tutor i has given a limit c_i on how many tutees they want to work with. Each tutee only gets one tutor.

Describe an efficient algorithm for assigning tutors to tutees, such that as many tutees receive tutoring as possible. Give only the main idea.

3. (★★★★ level) Criminal capture

The Central Intelligence Agency has tasked you with preventing a criminal from fleeing the country. Roads and cities are represented as an unweighted directed graph $G = (V, E)$. We're also given a set $C \subseteq V$ of possible current locations of the criminal, and a set $P \subseteq V$ of all airports out of the country. You want to set up roadblocks on a subset of the roads to prevent the criminal from escaping (i.e. reaching an airport).

Clearly you could just put a roadblock at all the roads to each airport, but there might be a better way: for example, if the criminal is known to be at a particular intersection, you could just block all roads coming out of it. You may assume that roadblocks can be set up instantaneously.

Give an efficient algorithm to find a way to stop the criminal using the least number of roadblocks.

(a) Describe the main idea of your algorithm (no proof or pseudocode necessary).

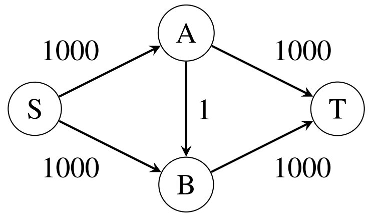
(b) Analyze the asymptotic running time of your algorithm, in terms of $|C|$, $|P|$, $|V|$, and $|E|$.

(c) Unfortunately we were too slow in implementing your roadblocks. The criminal has made it to an airport and is flying from city to city within the country. Your only option now is to shut down entire airports. Naturally, you want to minimize the number of airports you must shut down.

We formulate this as a graph problem where V is now the set of airports, and the (directed) edges E represent flights from one airport to another[1]. We still know a set $C \subseteq V$ of possible current locations of the criminal, and $P \subseteq V$ the set of all airports with flights leaving the country. Give an efficient algorithm to find the minimal set of airports (vertices) to block that will prevent the criminal from reaching P.

4. (★★★★★ level) Another max flow algorithm

Consider the following simple network with edge capacities as shown.



(a) Show that, if the Ford-Fulkerson algorithm is run on this graph, a careless choice of updates might cause it to take 2000 iterations. Imagine if the capacities were a million instead of 2000!

(b) We will now find a strategy for choosing paths under which the algorithm is guaranteed to terminate in a reasonable number of iterations.

Consider an arbitrary directed network $(G = (V, E), s, t, \{c_e\})$ in which we want to find the maximum flow. Assume for simplicity that all edge capacities are at least 1, and define the capacity of an $s - t$ path to be the smallest capacity of its constituent edges. The fattest path from s to t is the path with the most capacity.

[1] Assume flights are frequent enough that we can ignore departure/arrival times.

Show how to modify Dijkstra's algorithm to compute the fattest $s-t$ path in a graph. The full four-part algorithm response is not needed, but provide a convincing justification that your modification finds this path.

(c) Show that the maximum flow in G is the sum of individual flows along at most $|E|$ paths from s to t.

(d) Now show that if we always increase flow along the fattest path in the residual graph, then the Ford-Fulkerson algorithm will terminate in at most $O(|E| \log F)$ iterations, where F is the size of the maximum flow. (Hint: It might help to recall the proof for the greedy set cover algorithm in Section 5.4.)

In fact, an even simpler rule—finding a path in the residual graph using breadth-first search—guarantees that at most $O(|V| \cdot |E|)$ iterations will be needed.

5. (★★★★★ level) Cannibal canisters

You see n canisters, each with their own height h_i and radius r_i (they are perfectly cylindrical). A canister can eat another canister if it has a smaller height and radius. The eaten canister will be instantly consumed, and the eating canister will be tired for the rest of the day, unable to eat anymore. Design an algorithm to make as many canisters as possible get eaten (so they don't try to eat you!) by the end of the day. Your solution should give the optimal order that canisters should eat each other, and the runtime should be $O(n^3)$.

As an example, if there are three canisters with height and radius:

$$h_1 = r_1 = 1.5$$

$$h_2 = r_2 = 2.5$$

$$h_3 = r_3 = 3.5$$

Then your output should be "c_2 eats c_1, c_3 eats c_2" or similar.

(a) For this part, assume that $\forall i, h_i = r_i$. How would you solve the problem? Describe only the main idea and runtime, no need for pseudocode or proof.

(b) Solve the question fully, without the assumption that $h_i = r_i$. Hint: think about bipartite matching.