CPSC 365 / ECON 365: Algorithms Yale University

Problem Set 5

Out: March 15, 2022 Due: April 5, 2022 at 2:30 pm EST

Instructions

Write your solution to the following problems carefully. Submit the PDF of your solution via Gradescope. Please start your solution to every question on a new page. Make sure to assign the correct page in your document corresponding to each problem. We recommend writing your solution in Latex. Handwritten solutions are accepted if they are clearly legible. Write your name and SID in your answer to Problem 0; do not write them anywhere else in your solution.

Collaboration policy: You may collaborate and discuss these problems with other students. However, you must first make an honest effort to solve these problems by yourself. You may discuss hints and try to solve problems together. But you may not explicitly provide the solution to a problem to other students; nor receive the solution from them. You must write your solution independently, and make sure you understand your solution. You must list all your collaborators (anyone with whom you discussed any part of these problems). You may consult the course textbook and other references related to the class. However, you are forbidden from searching for these problems on the Internet. You must list any resources you consult (beyond the course textbook). You must also follow the Yale academic integrity policy.

0 Your Information

On the first page of your submission, include the following information (but do not include these anywhere else in your solution). Also certify that you have followed the collaboration policy. Your solution to the remainder of the problems should start on a new page.

(a) Your name.

(b) Your SID.

(c) A list your collaborators and any outside resources you consulted for this problem set. If none, write “None”.

(d) Certify that you have followed the academic integrity and collaboration policy as written above.

(e) How many hours did you spend in this problem set?

1 Max Flow

Figure 1 shows a flow network on which an $s-t$ flow has been computed. The capacity of each edge appears as a label next to the edge, and the number is boxes give the amount of flow sent on each edge. (Edges without boxed numbers have no flow being sent on them.)

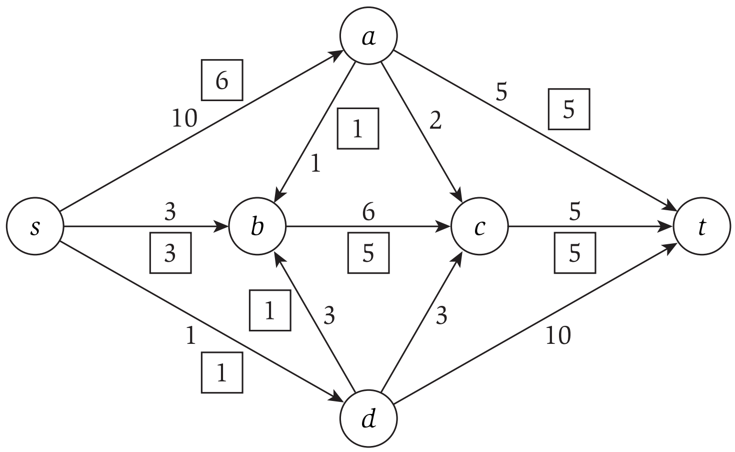

Figure 1: Graph for Problem 1.

(a) What is the value of this flow? Is this flow a maximum $s-t$ flow in this graph?

(b) Find an augmenting path in the residual graph, and use it to augment the flow. Draw the new flow, and say what its value is. Is it a maximum $s-t$ flow in this graph?

(c) Find a minimum $s-t$ cut in the graph in Figure 1. What is its capacity?

2 Min Cut

Let $G = (V, E)$ be an arbitrary flow network, with a source $s \in V$, a sink $t \in V$, and a positive integer capacity c_e on every edge $e \in E$. Let (A, B) be a minimum $s-t$ cut in G with respect to these capacities $\{c_e : e \in E\}$. Now suppose we add 1 to every capacity, so $c'_e = c_e + 1$ for $e \in E$.

Prove or disprove the following statement:

"(A, B) is still a minimum $s-t$ cut with respect to the new capacities $\{c'_e : e \in E\}$."

(If the statement above is true, prove it. If it is false, provide a counterexample.)

3 Repairing a Flow

Suppose you are given a directed graph $G = (V, E)$, with a source $s \in V$, a sink $t \in V$, and with a positive integer capacity c_e for every edge $e \in E$. You are also given a maximum $s - t$ flow f, specified by the flow value f_e (which is an integer) along each edge e. Assume the flow f is *acyclic*, which means there is no cycle C in G on which all edges $e \in C$ carry a positive flow $f_e > 0$.

Now suppose we pick a specific edge $e^* \in E$ and reduce its capacity by 1 unit: $c'_{e^*} = c_{e^*} - 1$. (The other edges have the same capacity as before: $c'_e = c_e$ for $e \neq e^*$.) We want to find a maximum flow in G with the new capacity c'. We can do so by computing the flow from scratch, but there is a faster way by modifying the flow f.

Design an algorithm to find a maximum $s - t$ flow in the graph G with the new capacity c'. Justify the correctness of the algorithm and analyze the running time. To get full credit, your algorithm should run in time $O(|V| + |E|)$.

4 Scheduling Doctors

Suppose you run a medical consulting firm Doctors Without Weekends to help hospitals with various scheduling problems. They've just come to you with the following problem:

For each of the next $n \geq 1$ days, the hospital has determined the number of doctors that they want on hand: On day $i \in [n] = \{1, \ldots, n\}$, there must be *exactly* $p_i \geq 1$ doctors in the hospital.

There are $k \geq 1$ doctors, and each doctor $j \in [k] = \{1, \ldots, k\}$ has provided a set $L_j \subseteq [n]$ of days on which they are willing to work.

Your goal is to find an assignment of the days on which each doctor must work, to satisfy the constraints above. Specifically, you want to return to each doctor j a set $L'_j \subseteq [n]$ of the days they have to work, with the following properties:

(A) $L'_j \subseteq L_j$ (so that doctor j only works on days they are willing); and

(B) Each $i \in [n]$ is in exactly p_i lists among $L'_1, \ldots, L'_k$ (so there are p_i doctors on day i).

Design an algorithm that takes as input the numbers $(p_1, \ldots, p_n)$ and the lists $(L_1, \ldots, L_k)$, and does one of the following two things:

(i) Return a list of sets $(L'_1, \ldots, L'_k)$ satisfying the properties (A) and (B) above, if one exists; or

(ii) If there is no such assignment, report that the problem is impossible.

Justify the correctness of your algorithm, and analyze the running time. Your algorithm should run in time polynomial in n and k.

Hint: Formulate this as a problem that we have seen in class. You must prove why the original problem is equivalent to the problem you construct (i.e., prove that there is such an assignment if and only if your construction satisfies some property).

5 Rearranging a Matrix

Let M be an $n \times n$ matrix with entries $m_{ij} \in \{0, 1\}$ for $i, j = 1, \ldots, n$.

Suppose we are allowed to swap two rows or two columns of M: Swapping rows i and j means we swap the values m_{ik} and m_{jk} for $k = 1, \ldots, n$. Swapping columns i and j means we swap the values m_{ki} and m_{kj} for $k = 1, \ldots, n$.

We say M is *rearrangeable* if it is possible to swap some of the pairs of rows and some of the pairs of columns (in any sequence) so that after all the swapping, all the diagonal entries of M are equal to 1: $m_{ii} = 1$ for $i = 1, \ldots, n$.

(a) Give an example of a matrix M that is *not* rearrangeable, but M has at least one entry in each row equal to 1, and at least one entry in each column equal to 1.

(b) Design an algorithm that takes as input the matrix $M \in \{0, 1\}^{n \times n}$, and determines whether M is rearrangeable. Justify the correctness of your algorithm, and analyze the running time. Your algorithm should run in time polynomial in n.

Hint: Formulate this as a problem that we have seen in class. You must prove why the original problem is equivalent to the problem you construct (i.e., prove that M is rearrangeable if and only if your construction satisfies some property).