University of Washington
Department of Computer Science and Engineering
CSE 421, Winter 2024

February 23, 2024

Homework 8, Due Friday, March 1, 11:59 pm, 2024

Turnin instructions: Electronic submission on gradescope using the CSE 421 gradescope site. Submit the assignment as a PDF, with separate pages for different numbered problems. Problems consisting of multiple parts (e.g., 2a, 2b) can be submitted on the same page.
For all of these problem, provide a justification as to why your algorithm correctly solves the problem. Give and justify the run time for your algorithms.

Problem 1 (10 Points):

(Kleinberg-Tardos, Exercise 9, Page 419) Network flow issues come up in dealing with natural disasters and other crises, since major unexpected events often require the movement and evacuation of large numbers of people in a short amount of time.

Consider the following scenario. Due to large-scale flooding in a region, paramedics have identified a set of n injured people distributed across the region who need to be rushed to hospitals. There are k hospitals in the region, and each of the n people needs to be brought to a hospital that is within a half-hour's driving time of their current location (so different people will have different options for hospitals, depending on where they are right now).

At the same time, one does not want to overload any one of the hospitals by sending too many patients its way. The paramedics are in touch by cell phone, and they want to collectively work out whether they can choose a hospital for each of the injured people in such a way that the load on the hospitals is balanced: Each hospital receives at most $\lceil n/k \rceil$ people.

Give a polynomial-time algorithm that takes the given information about the people's locations and determines whether this is possible.

Problem 2 (10 Points):

Suppose you are given a bipartite graph G with n vertices on each side and m edges, and a matching M with $n-1$ edges that is contained in the graph. Give an $O(n+m)$ time algorithm to check whether or not the graph has a matching of size n.

Problem 3 (10 Points):

(Kleinberg-Tardos, Exercise 23, Page 428) Suppose you're looking at a flow network G with source s and sink t, and you want to be able to express something like the following intuitive notion: Some nodes are clearly on the *source side* of the main bottlenecks; some nodes are clearly on the *sink side* of the main bottlenecks; and some nodes are in the middle. However, G can have many minimum cuts, so we have to be careful in how we try making this idea precise.

Here is one way to divide the nodes of G into three categories of this sort:

- We say a node v is upstream if, for all minimum $s-t$ cuts (A, B), we have $v \in A$ – that is, v lies on the source side of every minimum cut.
- We say a node v is downstream if, for all minimum $s-t$ cuts (A, B), we have $v \in B$ – that is, v lies on the sink side of every minimum cut.
- We say a node v is central if it is neither upstream nor downstream; there is at least one minimum $s-t$ cut (A, B) for which $v \in A$, and at least one minimum $s-t$ cut (A, B) for which $v \in B'$.

Give an algorithm that takes a flow network G and classifies each of its nodes as being upstream, downstream, or central. The running time of your algorithm should be within a constant factor of the time required to compute a single maximum flow.

Problem 4 (10 Points):

(Based on Kleinberg-Tardos, Exercise 19, Page 425)
A hospital has k doctors $D_1, \ldots D_k$ and wants to set up a schedule for n days, with the requirement that p_i doctors are available on day i. Each doctor D_j provides a list of days L_j he or she wants to work.

a) Describe a polynomial time algorithm that will determine if there is a schedule that provides p_i doctors on day i and assigns doctors to days they want to work.

b) If there is no assignment that satisfies all of the constraints, the hospital wants to minimize the maximum number of days that a doctor is assigned outside of his or her preference list. In other words, find the smallest $c \geq 0$ such that p_i doctors are assigned on day i and each doctor is assigned at most c days outside of their preference list. Describe a polynomial time algorithm finds such an assignment.

Problem 5 (10 Points):

(Based on Kleinberg-Tardos, Exercise 35, Page 436) This problem is to extend the image segmentation algorithm (from Section 7.10) to allow a user to force individual pixels to be assigned to either the foreground or the background. $q(A, B)$ is given the text and the course slides for lecture 20.

You start with the image segmentation setup described in Section 7.10, with n pixels, a set of neighboring pairs, and parameters a_i, b_i. and p_{ij}. We will make two assumptions about this instance. First, we suppose that each of the parameters a_i, b_i. and p_{ij} is a non-negative integer between 0 and d, for some number d. Second, we suppose that the neighbor relation among the pixels has the property that each pixel is a neighbor of at most four other pixels (so in the resulting graph, there are at most four edges out of each node).

You first perform an *initial segmentation* (A_0, B_0) so as to maximize the quantity $q(A_0, B_0)$. Now, this might result in certain pixels being assigned to the background when the user knows that they ought to be in the foreground. So, when presented with the segmentation, the user has the option of mouse-clicking on a particular pixel v_1, thereby bringing it to the foreground. But the tool should not simply bring this pixel into the foreground; rather, it should compute a segmentation (A_1, B_1) that maximizes the quantity $q(A_1, B_1)$ subject to the condition that v_1 is in the foreground.

In fact, the system should allow the user to perform a sequence of such mouse-clicks $v_1, v_2, \ldots, v_i$; and after mouse-click v_i, the system should produce a segmentation (A_i, B_i) that maximizes the quantity $q(A_i, B_i)$ subject to the condition that all of $v_1, v_2, \ldots, v_i$ are in the foreground.

Give an algorithm that performs these operations so that the initial segmentation is computed within a constant factor of the time for a single maximum flow, and then the interaction with the user is handled in $O(dn)$ time per mouse-click.