Introduction to Algorithms	**Problem Set 6**
CS 482 Spring 2004	**Due March 19, 2004**

Please hand in each problem on a separate sheet with your name and NetID on each.

Reading: Section 5.9 (also Sections 5.1-5.4 and 5.7-5.8 from last week).

(1) *(This is Problem 10 at the end of Chapter 5.)* Back in the euphoric early days of the Web, people liked to claim that much of the enormous potential in a company like *Yahoo!* was in the "eyeballs" — the simple fact that it gets millions of people looking at its pages every day. And further, by convincing people to register personal data with the site, it can show each user an extremely targeted advertisement whenever he or she visits the site, in a way that TV networks or magazines couldn't hope to match. So if the user has told *Yahoo!* that they're a 20-year old computer science major from Cornell University, the site can throw up a banner ad for apartments in Ithaca, NY; on the other hand, if they're a 50-year-old investment banker from Greenwich, Connecticut, the site can display a banner ad pitching Lincoln Town Cars instead.

But deciding on which ads to show to which people involves some serious computation behind the scenes. Suppose that the managers of a popular Web site have identified k distinct *demographic groups* $G_1, G_2, \ldots, G_k$. (These groups can overlap; for example G_1 can be equal to all residents of New York State, and G_2 can be equal to all people with a degree in computer science.) The site has contracts with m different *advertisers*, to show a certain number of copies of their ads to users of the site. Here's what the contract with the i^{th} advertiser looks like:

- For a subset $X_i \subseteq \{G_1, \ldots, G_k\}$ of the demographic groups, advertiser i wants its ads shown only to users who belong to at least one of the demographic groups in the set X_i.
- For a number r_i, advertiser i wants its ads shown to at least r_i users each minute.

Now, consider the problem of designing a good *advertising policy* — a way to show a single ad to each user of the site. Suppose at a given minute, there are n users visiting the site. Because we have registration information on each of these users, we know that user j (for $j = 1, 2, \ldots, n$) belongs to a subset $U_j \subseteq \{G_1, \ldots, G_k\}$ of the demographic groups. The problem is: is there a way to show a single ad to each user so that the site's contracts with each of the m advertisers is satisfied for this minute? (That is, for each $i = 1, 2, \ldots, m$, at least r_i of the n users, each belonging to at least one demographic group in X_i, are shown an ad provided by advertiser i.)

Give an efficient algorithm to decide if this is possible, and if so, to actually choose an ad to show each user.

(2) *Ad hoc networks*, made up of low-powered wireless devices, have been proposed for situations like natural disasters in which the coordinators of a rescue effort might want

to monitor conditions in a hard-to-reach area. The idea is that a large collection of these wireless devices could be dropped into such an area from an airplane, and then be configured into a functioning network.

Note that we're talking about (a) relatively inexpensive devices that are (b) being dropped from an airplane into (c) dangerous territory; and for the combination of reasons (a), (b), and (c), it becomes necessary to include provisions for dealing with the failure of a reasonable number of the nodes.

We'd like it to be the case that if one of the devices v detects that it is in danger of failing, it should transmit a representation of its current state to some other device in the network. Each device has a limited transmitting range — say it can communicate with other devices that lie within d meters of it. Moreover, since we don't want it to try transmitting its state to a device that has already failed, we should include some redundancy: a device v should have a set of k other devices that it can potentially contact, each within d meters of it. We'll call this a *back-up set* for device v.

(a) Suppose you're given a set of n wireless devices, with positions represented by an (x, y) coordinate pair for each. Design an algorithm that determines whether it is possible to choose a back-up set for each device (i.e. k other devices, each within d meters), with the further property that, for some parameter b, no device appears in the back-up set of more than b other devices. The algorithm should output the back-up sets themselves, provided they can be found.

(b) The idea that, for each pair of devices v and w, there's a strict dichotomy between being "in range" or "out of range" is a simplified abstraction. More accurately, there's a power decay function $f(\cdot)$ that specifies, for a pair of devices at distance δ, the signal strength $f(\delta)$ that they'll be able to achieve on their wireless connection. (We'll assume that $f(\delta)$ decreases with increasing δ.)

We might want to build this into our notion of back-up sets as follows: among the k devices in the back-up set of v, there should be at least one that can be reached with very high signal strength, at least one other than can be reached with moderately high signal strength, and so forth. More concretely, we have values $p_1 \geq p_2 \geq \cdots \geq p_k$ so that if the back-up set for v consists of devices at distances $d_1 \leq d_2 \leq \cdots \leq d_k$, then we should have $f(d_j) \geq p_j$ for each j.

Give an algorithm that determines whether it is possible to choose a back-up set for each device subject to this more detailed condition, still requiring that no device should appear in the back-up set of more than b other devices. Again, the algorithm should output the back-up sets themselves, provided they can be found.

(3) *(This is Problem 39 at the end of Chapter 5.)*

(a) Suppose you are given a flow network with integer capacities, together with a maximum flow f that has been computed in the network. Now, the capacity of one of the edges e out of the source is raised by one unit. Show how to compute a maximum flow in the resulting network in time $O(m + n)$, where m is the number of edges and n is the number of nodes.

(b) You're designing an interactive image segmentation tool that works as follows. You start with the image segmentation set-up described in Section 5.9, 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 will 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 will 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 practice, this is useful for the following kind of operation: in segmenting a photo of a group of people, perhaps someone is holding a bag that has been accidentally labeled as part of the background. By clicking on a single pixel belonging to the bag, and re-computing an optimal segmentation subject to the new condition, the whole bag will often become part of the foreground.)

In fact, the system should allow the user to perform a sequence of such mouse-clicks $v_1, v_2, \ldots, v_t$; 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.

(Note: Part (a) is a useful primitive for doing this. Also, the symmetric operation of forcing a pixel to belong to the background can be handled by analogous means, but you do not have to work this out here.)