Problem Set 10 for CS 170

Formatting

Please use the following format for the top of the solution you turn in, with one line per item below (in the order shown below):

> <your username on cory.eecs>
> <your full name>
> `CS170, Spring 2003`
> `Homework #10`
> `Section` <your section number>
> `Partners:` <your list of partners>

(Remember to write your section number, not the name of your TA or the time of your section.) This will make it easier for us to sort and process your homeworks. Thank you!

Note

When asked for an algorithm you must give (1) a brief informal description of the algorithm, (2) a precise description using pseudo-code, (3) an informal argument for termination and correctness of the algorithm, and (4) an analysis of the running time of the algorithm.

Exception! (added 4/21) For Problems 1 and 2, regarding the running time: it suffices to state that your algorithm is polynomial time (it had better be!), and briefly say why.

New! In addition, when specifying a linear programming problem, please first list a table of variables and their intended meaning, before the system of linear inequalities itself.

Problem 0. [Any questions?] (5 points)

What's the one thing you'd most like to see explained better in lecture or discussion sections? A one-line answer would be appreciated.

(Sometimes we botch the description of some concept, leaving people confused. Sometimes we omit things people would like to hear about. Sometimes the book is very confusing on some point. Here's your chance to tell us what those things were.)

Problem 1. [Henron] (30 points)

You've been hired by Henron, a new startup with this great idea to make a killing by creating a market for chicken futures. Henron has relationships with a set of n suppliers and a set of m purchasers. The i-th supplier can supply up to $s[i]$ chickens this year, and the j-th purchaser would like to buy up to $b[j]$ chickens this year. Henron is the middleman

and makes \$1 off each chicken that is sold. Thus, the more chickens that are sold, the more money you make!

However, there's a complication. Due to federal restrictions on interstate trafficking in avian life forms, supplier i can only sell chickens to a purchaser j if they are situated at most 100 miles apart. Assume that you're given a list L of all the pairs (i, j) such that supplier i is within 100 miles of purchaser j. You will be given $n, m, s[1..n], b[1..m], L$ as input. Your job will be to compute the maximum number of chickens that can be sold this year.

(a) Formulate this as a network flow problem.

(In other words, show how to solve this problem, using a network flow algorithm as a subroutine.)

(If you can show your graph, that might help the readers.)

(b) Formulate this as a linear programming problem.

(In other words, show how to solve this problem, using a linear programming algorithm as a subroutine.)

(c) Let's assume you don't care about the running time of your algorithm. Which formulation would be better, network flow or linear programming?

HINT: You can't sell fractional chickens. (Can you imagine the mess?)

Problem 2. [Optimal Gerrymandering] (25 points)

It will soon be election time in Upper Moldavia, and things don't look good for the Republicrat party: the majority of Upper Moldavians plan to vote Democan next year. However, the Republicrat party has a secret weapon up their sleeve: they control the committee that will be drawing up the map of voting districts[1]. The election will be winner-take-all: in each district, whichever party gets the most votes receives one representative elected to the Moldavian Senate. Your job will be to find an algorithm the Republicrats could use to maximize the number of representatives elected for their party next election.

There are n counties in Upper Moldavia. Exactly r_i of the residents in county i are Republicrat, and the other d_i residents are Democan. No one votes for a third party. In your power as mapmaker, you must build m voting districts by amalgamating pieces from the various counties. A district can be made up of many portions of many counties, and each portion can be obtained by drawing off any fraction $f_{i,j}$ of the i-th county towards the j-th district (thereby adding $f_{i,j} \cdot r_i$ Republicrats and $f_{i,j} \cdot d_i$ Democans to the j-th district). For instance, district 1 might be composed of $\frac{1}{2}$ of county A and $\frac{1}{3}$ of county B; district 2 might include $\frac{1}{3}$ of county A and all of county C; and district 3 might include $\frac{1}{6}$ of county A and $\frac{2}{3}$ of county B.

You are subject only to the restrictions that each of the m districts you draw up must contain at least 10^6 voters, and that every voter is in exactly one district. Your algorithm will receive as input the values n, m, r_i, d_i. You must find the districting assignment (given

[1] Hey, if it was good enough for Governor Gerry, maybe it's good enough for Upper Moldavia...

by the $f_{i,j}$'s) that maximizes E, the number of Republicrat representatives that will be elected under your assignment.
Clarification! (added 4/22) Any algorithm that is polynomial in n and m will suffice. Your algorithm does not need to be polynomial in lgm.

HINT: binary search, linear programming.

Problem 3. [A Faster Network Flow Algorithm] (40 points)

Let G be a connected, weighted graph, with source s and sink t. Let f denote the capacity of the maximum flow in G. The *capacity* of an augmenting path is the smallest of the weights on the edges along the path. By "show", I mean "give an informal argument."

(a) Show that the maximum flow in G can always be found by some sequence of at most $|E|$ augmenting paths.

(In other words, show that if Ford-Fulkerson gets *really* lucky at every stage, it is always possible for Ford-Fulkerson to terminate after at most $|E|$ iterations.)

HINT: Choose the paths *after* finding the maximum flow.

(b) Show that, if we consider this set of $\leq |E|$ augmenting paths, the average path-capacity is at least $f/|E|$.

(c) Show that there is some augmenting path in G with capacity at least $f/|E|$.

(d) Show that, if we take the best augmenting path at each iteration, then $O(|E| \lg f)$ iterations of the Ford-Fulkerson algorithm suffice. (By "best", I mean the augmenting path with largest capacity.)

HINT: After picking the first augmenting path, how much (as measured in capacity not yet found) of the maximum flow remains to be found? How about after the second augmenting path? And so on.

HINT: $(1 - 1/|E|)^{|E|} < 1/2$.

(e) Show that we can find the best augmenting path in G efficiently.

HINT: I know of at least two ways to do this quickly, one giving a $O(|E| \lg f)$ running time, and the other $O(|E| \lg |E|)$. Either runtime would be acceptable.

(f) Using the above ideas, describe an efficient algorithm for computing the maximum flow in G. Your algorithm should be faster than the $O(|V| \cdot |E|^2)$ time method given in the reader, assuming f is not too large.