

CS 222 Fall 2007 HW 3, Due Oct. 26.

In our formulation of maximum flow, we only have upper capacities C_e on the flow in each edge e . However, it is possible to put a lower capacity L_e on each edge e , requiring that $f(e)$ satisfy the constraint that $L_e \leq f(e) \leq C_e$. The network-flow problem can be modified to determine if such a flow is possible, and to find a maximum flow if so (this is detailed in the book). For the next problem, you do not need to know the details of that modification, but you can assume that such a modification is possible, and use it in your answers, if that is helpful.

1. Table Rounding problem: You are given an n by m table of numbers between 0 and 1 along with row and column totals. Your objective is to round each entry to either 0 or 1 (you have complete freedom in this) so that each resulting row and column total is itself rounded to one of its two nearest integers, and so that the table total is rounded to one of its two nearest integers. Any number which was originally an integer should not change. For example,

.2	.4	0	.2		0.8		0	0	0	0	0	0
.4	0	.2	.4		1		0	0	1	0	1	
.8	.8	.8	0		2.4	====>	1	1	1	0	3	
.6	0	.2	.2		1		1	0	0	0	1	

2	1.2	1.2	.8		5.2		2	1	2	0	5	

It turns out that such a rounding is always possible, but you don't have to prove that.

Devise an algorithm, based on network flow, that takes in a table (as described above) and returns a rounding of it (as described above), or declares that no rounding is possible (that outcome should not actually ever happen). The solution must be general, not just be described for the example above. You might be able to prove that a rounding is always possible, by making some observations about your algorithm. That is not required, but you might try it out.

2. Suppose we have the height and weight of every person in the class, and we organize this information into a two dimensional table T where there are

n height intervals, and m weight intervals. In the table, the (i, j) entry $T(i, j)$ indicates how many people fall into height category i and weight category j . Suppose that some of the particular cell entries are kept secret (so they are empty), but the number of people falling into each height category is public, as is the number that fall into each weight category. That is, the n row sums and the m column sums of the table are published, along with some, but not all, of the individual cell values of the table.

Devise an algorithm, based on network flow, that finds a way to assign values to the empty cells so that every row now adds up to its given row total, and every column now adds up to its given column total.

3. The node cover problem

Let G be an undirected graph with each node i given weight $w(i) > 0$. A set of nodes S is a *node cover* of G if every edge of G is incident to at least one node of S . The *weight* of a node cover S is the summation of the weights, denoted $w(S)$, of the nodes in S ; the weighted node cover problem is to select a node cover with minimum weight.

If G is a bipartite, then the minimum weight node cover can be found by network flow. Explain how to do this. Hint: The minimum cut is the key, rather than the maximum flow.

4. Let f and f' be two different maximum s-t network flows in a graph G , where the total flow into any node v is the same in f as in f' (this may be the case for example in your solution to problem 2). Let $e = (u, v)$ be a directed edge where $f'(e) > f(e)$. Show that in the residual graph G_f for flow f , there must be a directed cycle that includes the forward edge (u, v) . Similarly, in the residual graph $G_{f'}$ for f' , there must be a directed cycle that includes the backward edge (v, u) .

Now suppose we want to know if there is another s-t flow f'' in G where the total flow into any node is the same as in f , and where $f''(e) > f'(e)$, assuming that $f'(e) < C_e$. We claim we can solve this question with the following method: Let $G_{f'}$ be the residual graph for flow f' . Then remove the edge (v, u) from $G_{f'}$ and find the maximum flow value in that $G_{f'}$, from v to u . (That is, let v be the source node, and u be the terminal node in that flow computation.)

Problem: How does the flow value computed in the above method answer the question of whether such an f'' exists? That is, complete the description of the above method, and explain why the method is correct.