CS 222 Fall 2007, Midterm. The exam is open materials (books CLR and Kleinberg/Tardos, along with any materials distributed through the class website, and any of your own notes).

Try for SHORT, to the point answers. Explain you answers. You do not need formal proofs, but you do need to identify and explain the key points that show your answer is correct.

1. In homework 1, we saw that if M and M' are two different (perfect) stable matchings, and if each man is assigned to woman $F(m, M, M')$, the result is also a stable matching. The book proved something stronger: For a given set of men's and women's preferences, let S be the set of all (perfect) stable matchings based on those preferences, and let $F(m, S)$ be the woman that man m most prefers over *all* of his partners in the matchings in S. Using the above mentioned result from homework 1, show that if each man m is assigned to $F(m, S)$ then the result is a stable matching. The book proves this from scratch, but you MUST use the result from homework 1. A very short answer is possible.

Answer: Consider the following conceptual algorithm:

0) Let M and M' be any two stable matchings in S.

1) Let M'' be the stable matching formed from assigning each man m to $F(m, M, M')$. Remove M and M' from S. M'' is a stable matching by the result of Homework 1.

2) If S is not empty, let M be any stable matching in S; Assign M'' to M' and go to step 1). If S is empty, stop.

The algorithm explicitly considers each stable matching in M, always assigning m to the woman he most prefers between M and his best choice obtained so far. Hence at termination, m is asssigned to $F(m, S)$, and since M'' is a stable matching at each iteration, that assignment is a stable matching.

2. Let f be an maximum s-t flow in a directed graph $G = (V, E)$ found by the Ford-Fulkerson algorithm, and let G_f be the augmentation (residual) graph for G and f. Let (A, B) be the s-t cut where A consists of all nodes reachable by a directed path from s in G_f, so $B = V - A$. In class we proved that (A, B) is a minimum s-t cut. Now define Q as the set of all nodes that can reach t by a directed path in G_f, and let $P = V - Q$.

2a) Give an example to show that Q is not necesarily B.

2b) Is (P, Q) necesarily a minimum s-t cut? Give a clear justification for your answer.

Answer: Consider a graph consisting only of the s-t path $s, v1, v2, v3, t$ where the capacity of the first and last edges is 1, and the capacity of the

other two edges is 2. Then $A = \{s\}$ so $B = \{v1, v2, v3, t\}$, but $Q = \{t\}$. The s-t cut (P, Q) is a minumum cut. First, it is an s-t cut because t certainly is in Q. Now s is in P because there is no s to t directed path in G_f when f is the maximum flow. So (P, Q) is an s-t cut.

To show that it is a minimum s-t cut, consider any edge in $e = (u, v)$ in G directed from P to Q. It must be that $f(e) = c_e$ or e would be a forward edge in G_f and u would be in Q. Similarly, consider an edge $e' = (x, y)$ in G directed from Q to P. It must be that $f(e') = 0$ or else the edge (y, x) would be in G_f and y would be in Q. Hence, the flow across $(P, Q) = \Sigma f(e) : e$ crosses from P to Q, so the flow across $(P, Q) = \Sigma c_e : e$ crosses from P to Q. Hence the capacity of (P, Q) equals the value of the maximum flow f, making (P, Q) a minimum cut.

Another possible way to answer this question is to consider reversing the directed of all edges in G, and assigning the maximum flow f, from the original graph to the corresponding reversed edges. Then argue that the t to s from must be maximum, and that the (Q, P) cut is a minimum t-s cut for this flow. This argument is a bit more subtle. The argument above is more direct.

3. A bipartite graph is a graph G consisting of two sets of nodes, A and B, such that every edge in G has one end in A and one end in B. A *matching* M in G is a set of edges such that no two edges in M share a node. Note that a matching might not be perfect, i.e., there may be nodes that are not touched by an edge in M.

The nodes of A are numbered 1 through $|A|$, with each node getting a distinct number. Similarly, the node in B are numbered 1 through $|B|$, with each node getting a distinct number. Two edges (i, j) and (i', j') *cross* if $i < i'$ but $j' < j$, or $i' < i$ but $j < j'$. Nodes i, i' are in A and j, j' are in B. A *non-crossing* matching is a matching that does not contain any crossing edges. We want to find a *largest* non-crossing matching (one with the largest number of edges). This problem can be solved by Dynamic Programming with the following recurrences.

Let $M(i, j)$ be the size of the largest non-crossing matching that can be found using only edges whose endpoints in A are from 1 to i, and whose endponts in B are from 1 to j. Then, for $i > 0$ and $j > 0$, $M(i, j) =$ Max $[M(i-1, j-1)+1$ if edge (i, j) exists; $M(i-1, j-1)$; $M(i, j-1)$; $M(i-1, j)]$. The base cases are $M(0, j) = M(i, 0) = 0$.

3a) Explain why these recurrences are correct, and how (in what order) they should be evaluated, the time bound for evaluating them, and how an actual matching is constructed.

3b) Explain why the general recurrence can be replaced with $M(i,j) =$ Max $[M(i-1,j-1)+1$ if edge (i,j) exists; $M(i,j-1); M(i-1,j)]$.

3c) Now suppose that each edge (i,j) has a given weight $w(i,j)$ which may be positive or negative or zero. We want to find a non-crossing matching M such that the *sum* of edge weights in M is maximum over all non-crossing matchings. Modify the recurrences to solve that problem, and state its time analysis.

3d) As in 3c) each edge has a weight $w(i,j)$ which can be positive or negative or zero. Now we want to find a non-crossing matching M such that the *product* of the edge weights in M is maximum over all non-crossing matchings. Explain how to solve that problem and its time analysis.

Answer: 3a) Note that nodes i and j can *both* be touched by some edge in a matching (that uses edges whose A node is from 1 to i and whose B node is from 1 to j) *only if* edge (i,j) is used. This follows from the non-crossing requirement. So the recurrences given in the problem statement explicitly enumerate all possibilities for how node i and j are involved in the solution of value $M(i,j)$, and explicitly enumerate the best results in those cases: when both i and j are touched in such a matching; when neither are touched; when only i is touched; when only j is touched. In each of these cases, the recurrence states what is the best possible outcome under that case. If the graph has n A nodes and m B nodes, then there are $(n+1)(m+1)$ values of M that must be calculated. Each takes a constant number of operations, hence the total time is $O(nm)$. TRACEBACK.

3b) The case of $M(i-1,j-1)$ may be removed because the matchings that are considered for $M(i,j-1)$ contain the matchings that are considered for $M(i-1,j-1)$.

3c) Change the general recurrence to $M(i,j) =$ Max $[M(i-1,j-1) +$ $w(i,j)$ if edge (i,j) exists; $M(i,j-1); M(i-1,j)]$.

3d) This is the hardest part. Let $M(i,j)$ be defined as the largest non-negative product possible in a non-crossing matching using edges whose A ends are from 1 to i and whose B ends are from 1 to j. Let $M'(i,j)$ defined as the smallest non-positive product possible in a non-crossing matching using edges whose A ends are from 1 to i and whose B ends are from 1 to j. Note that $M(i,j) \geq 0$ and $M'(i,j) \leq 0$, because one can always choose the empty matching consisting of no edges.

Then $M(i,j) =$ Max $[M(i-1,j-1) \times w(i,j)$ if edge (i,j) exists and $w(i,j) > 0; M'(i-1,j-1) \times w(i,j)$ if (i,j) exists and $w(i,j) < 0; M(i,j-1); M(i-1,j)]$.

Then $M'(i,j) =$ Min $[M'(i-1,j-1) \times w(i,j)$ if edge (i,j) exists and

$w(i,j) > 0; M(i-1,j-1) \times w(i,j)$ if (i,j) exists and $w(i,j) < 0$; $M'(i,j-1); M'(i-1,j)]$.

The key point in this solution is that because an edge weight can be negative, and the product of two negative numbers is positive, we have to determine and maintain the quantity $M'(i,j)$ as well as $M(i,j)$ and use it in the recurrences.

If you think about the recurrences a bit more, you should see that they can be simplified to:

$M(i,j) = \text{Max}\ [M(i-1,j-1) \times w(i,j)$, if (i,j) exists; $M'(i-1,j-1) \times w(i,j)$ if (i,j) exists; $M(i,j-1); M(i-1,j)]$.

$M'(i,j) = \text{Min}\ [M'(i-1,j-1) \times w(i,j)$ if edge (i,j) exists; $M(i-1,j-1) \times w(i,j)$ if (i,j) exists; $M'(i,j-1); M'(i-1,j)]$.