Math 108 Homework 3 Solutions
Ben Bond
February 23, 2016

Problem 1
Solution by Spencer Yee

Let \( M \) be the set of edges that are in either \( M_1 \) or \( M_2 \) or both of them that contain at least one vertex in \( S \) or \( T \). Since \( M_1 \) covers all vertices of \( S \) and \( M_2 \) covers all vertices of \( T \), \( M \) has at least one edge incident to every vertex in \( S \) and \( T \). Now, in \( M \), for any vertex \( v \) in \( S \cup T \), there is either one incident edge or two. There are two ways for there to be exactly one incident edge \( e \). The first is that \( e \) is in \( M_1 \cap M_2 \). The other possibility is that \( e \) is in \( M_1 \Delta M_2 \), and the other vertex incident to \( e \) is not in \( S \cup T \). In both cases, we add \( e \) to \( M_3 \). We repeatedly perform this process while there is still a vertex \( v \) in \( S \cup T \) that is not covered by \( M_3 \) and has exactly one incident edge in \( M \), and at the end \( M_3 \) covers all vertices in \( S \cup T \) with exactly one incident edge from \( M \).

Now we need to cover the vertices in \( S \cup T \) with two incident edges in \( M \). Each of these vertices was covered in both \( M_1 \) and \( M_2 \), but with different edges. We need to decide which of the two edges to add to \( M_3 \). From any vertex \( v_1 \) with two incident edges, take an edge and follow it to \( v_2 \), and if \( v_2 \) is incident to another edge follow that edge to \( v_3 \), and so on. Eventually, \( v_n \) must share an edge with \( v_1 \), forming a cycle, or \( v_n \) must only have one incident edge, or \( v_n \) must share an edge with a vertex not in \( S \cup T \). If \( v_n \) shares an edge with \( v_1 \), then add that edge and every other edge in the cycle to \( M_3 \). Note that every vertex from \( v_1 \) to \( v_n \) is now incident to exactly one edge in \( M_3 \). For the second case, if \( v_n \) has only one incident edge, then that edge has already been added to \( M_3 \), and we add every other edge before it to \( M_3 \), potentially going past \( v_1 \). Therefore, these edges added to \( M_3 \) cover all vertices in the path \( \cdots v_1 v_2 \cdots v_n \) with exactly one incident edge from \( M \). For the final case, if \( v_n \) shares an edge with a vertex not in \( S \cup T \), add that edge and every other edge before it to \( M_3 \). These edges also cover all vertices in the path just explored with exactly one incident edge from \( M \).

For both cases, the reasoning behind this is the same: if one end of a path ends in a vertex in \( S \cup T \) with one incident edge, then the other end must have a vertex sharing an edge with a vertex not in \( S \cup T \), and vice versa, so the two cases are really the same. To see why this is true, without loss of generality, assume that the vertex with exactly one incident edge is in \( S \). Thus the first edge added to \( M_3 \) is from \( M_1 \), and again every other edge is also from \( M_1 \). Thus, the last edge added to \( M_3 \) from this path must be incident to a vertex in \( S \). If the edge is incident to a vertex \( v \) in \( T \) as well, then there is one more edge \( e \) in the path that connects \( v \) to \( A \setminus S \), and this edge was not added to \( M_3 \). The one last edge cannot connect \( v \) to a vertex \( v' \) in \( S \), because otherwise the match for \( v' \) from \( M_1 \) would connect \( v \) to another vertex in \( T \) (which cannot \( v \), because otherwise \( v \) would only have one incident edge and could not be in the middle of the path), which contradicts the fact that \( e \) was the last edge in the path. Thus all the vertices in \( S \cup T \) in this path are covered in \( M_3 \). If the last edge added to \( M_3 \) from this path instead connects a vertex in \( S \) to a vertex in \( B \setminus T \), then there are no more edges in the path and all vertices in \( S \cup T \) in this path are covered in \( M_3 \).

We can repeatedly perform this process while there is still a vertex \( v \) in \( S \cup T \) that is not covered by \( M_3 \) and has two incident edges in \( M \). Finally, \( M_3 \) will cover all vertices in \( S \cup T \), since for every vertex in \( S \cup T \), \( M_3 \) contains either its only incident edge from \( M \), or one of its two incident edges from \( M \).
Problem 2

Solution by Alfred Xue

We describe the unique stable matching as follows. Let $w_1$ denote the top ranked woman, $w_2$ denote the second ranked woman, and so on... Then we argue the unique stable matching is as follows:

Match $w_1$ to her top choice. Match $w_2$ to her top remaining choice, $w_3$ to her top remaining choice, and so on.

We first prove that this is a stable matching. We do this by proving that every woman is in a stable matching (which also proves that every man is in a stable matching, since if there exists a man not in a stable matching then there must also exist a woman not in a stable matching).

We prove by contradiction. Assume that there exists a woman $w_m$ not in a stable matching. Then there must exist a man who she ranked higher than her current partner who also ranked her higher than his current partner. But if she ranked the man higher than her current partner, then either the man is matched with a woman from $w_1, \ldots, w_{m-1}$, or the man was unmatched when $w_m$ selected her top remaining choice. If the man was unmatched when $w_m$ selected her top remaining choice, then the woman could not select her current partner, since the man is ranked higher than her current partner, a contradiction. Thus every woman is in a stable matching, which proves that this method produces a stable matching.

We next prove that this is a unique stable matching by contradiction. Assume that there exists a distinct stable matching. Then there must exist some woman that does not have the same matching as ours. Let $w_m$ be the woman ranked highest by the men that does not have the same matching as ours, and $m_m$ be the man that women is matched to in ours. Then the woman could not be matched with a man matched to someone in $w_1, \ldots, w_{m-1}$, since they are matched to $w_1, \ldots, w_{m-1}$. Then they must be matched with someone of lower rating by $w_m$, then $m_m$. But then that matching is not stable, since $w_m$ ranked $m_m$ higher, and $m_m$ ranked $w_m$ higher than whoever he is matched to, since it cannot be $w_1, \ldots, w_{m-1}$. Thus any other matching is not stable.

Problem 3

Solution by Anshul Samar

First, arrange all bit strings in a Christmas tree pattern. Let $\omega \in \{0,1\}^n$. Say it is in row $i$ and column $k$ of the Christmas tree (rows and columns are zero-indexed). Let $f(\omega)$ return the bit string in row $i$ and column $n - k - 1$ of the Christmas tree. First note that because each row in the Christmas tree form a chain, we automatically satisfy property (ii) as we are staying within the row. Second, note that $\omega$ has $k$ zeros while $f(\omega)$ has $n - k$ zeros. Therefore, $\nu(\omega) + \nu(f(\omega)) = n$ satisfying property 3. Lastly note, that since $f(\omega)$ is in row $i$ and column $n - k$, $f(f(\omega))$ is in row $i$ and column $n - n + k = k$, i.e. $f(f(\omega)) = \omega$.

Our function therefore satisfies the desired properties.

Problem 4

Solution by Zhaolin Ren

Fix a $n \in \mathbb{N}$. Consider the Christmas tree pattern of order $n$, which enumerates all the distinct $n$-bit binary strings. Given any row $i$, looking at the elements $\sigma_1, \sigma_2, \ldots, \sigma_s$ (listed from the left), we have $\sigma_1 \leq \sigma_2 \cdots \leq \sigma_s$. Thus, we cannot take more than two elements from each row of the $n$-Christmas tree. Let us now construct a maximum set $A$ that contains no chain of size 3.

Consider any column, say $k$, in the $n$-Christmas tree. Column $k$ contains all strings with $k$ 1’s in them, so has exactly $\binom{n}{k}$ elements. Note that the rows are centered amongst the columns, so the middle two columns...
are \( \lfloor n/2 \rfloor \) and column \( \lfloor n/2 \rfloor + 1 \). Let us put all these elements in these columns in a set \( A \). \( A \) is the maximum set with no 3-chain we can construct, because we can never take more than 2 elements from each row, and taking elements from the middle 2 columns guarantee that we always get 2 elements from a row when it contains 2 or more elements, and 1 element from a row when it contains just 1 element. Thus, the largest set \( A \) comprises all \( n \)-bit binary strings with \( \lfloor n/2 \rfloor \) 1's and \( (\lfloor n/2 \rfloor + 1) \) 1's.