CS 105 (Winter 2005)
Homework 2 Solutions

Problem 1

Part (a) We shall prove that a graph G with distinct edge weights has a unique MST. Suppose that G has two different MSTs, T and T'. Let $e \in T' - T$. As shown in class, $\forall\, f \in \mathrm{cyc}_T(e),\ T + e - f$ is a spanning tree of G. Since T is minimum, this means $wt(e) \geq wt(f)$. Since weights are distinct, we have $wt(e) > wt(f)$. Therefore e is the heaviest edge in the cycle $\mathrm{cyc}_T(e)$. The Red Rule implies that e cannot belong to the MST T', a contradiction.

Everyone successfully found a graph with distinct edge weights and two different second-best-MSTs (2BMSTs), so I shall skip this.

Part (b) We shall show that a 2BMST of G can be obtained by doing just a single edge switch to the MST. In fact we shall show something stronger: that *every* 2BMST can be obtained this way.

Let T_2 be a 2BMST of G. There must exist an edge in $E(G) - T_2$ to which the Red Rule does *not* apply, or else no edge in $E(G) - T_2$ could be an MST edge and T_2 would have to be an MST, which it isn't. Let $e \in E(G) - T_2$ be such an edge. Since the Red Rule does not apply to e, it is *not* the heaviest edge in $\mathrm{cyc}_{T_2}(e)$. In other words $\exists\, f \in \mathrm{cyc}_{T_2}(e)$ such that $wt(f) > wt(e)$. Therefore, the spanning tree $T_2 + e - f$ is lighter than T_2, whence it must be the unique MST of G.

This shows that $T_2 = \mathrm{MST}(G) - e + f$, as desired.

Part (c) We must determine $\max[u, v]$ for all pairs of vertices u and v in time $O(n^2)$. There are many different ways of doing this. Amazingly, I received ten different solutions to this problem and they were all more-or-less correct!

Pick a vertex r of T and run a DFS of T from r, keeping track of the heaviest edge on the path from r to the current vertex. This can be done by, for instance, passing along an extra "max-so-far" parameter in a recursive implementation of DFS. Just before recursing on the children of a vertex $v \neq r$, fill in the table entries $\max[r, v]$ and $\max[v, r]$. The entire operation takes $O(n)$ time, as T has $n - 1$ edges and n vertices.

The above procedure computes $\max[r, v]$ and $\max[v, r]$ for a particular r. Repeat the procedure n times, letting r range over all the vertices of the graph. Clearly this fills in the $\max$ table completely, correctly, and in $O(n^2)$ time.

Part (d) To find a 2BMST of a given graph G, first find the MST T of G using Prim's algorithm. This takes $O(m + n \log n)$ time. Then, compute the $\max[u, v]$ array with respect to T, using the above $O(n^2)$ algorithm. Finally, looping over all pairs u, v of vertices, find one that minimizes the quantity $(wt(u, v) - wt(\max[u, v]))$; this takes $O(n^2)$ time.

By the result of part (b), $T - \max[u, v] + \{u, v\}$ is a 2BMST of G.

Problem 2

Part (a) We shall prove that an MST of a graph is also a bottleneck spanning tree (BNST). Suppose not; i.e., suppose T is an MST of a graph G and that there is a BNST T' such that

$$\max\{wt(e) : e \in T\} \;=\; \alpha \;>\; \beta \;=\; \max\{wt(e') : e' \in T'\}\,.$$

Let e_m be an edge in T of weight α. The above condition implies that $e_m \notin T'$ and that e_m is the heaviest edge in $\mathrm{cyc}_{T'}(e_m)$, because all other edges in that cycle are of weight β or less. By the Red Rule, e_m cannot belong to any MST, a contradiction.

Part (b) Given a graph G and a value β, we wish to determine, in linear time, whether G has a BNST with maximum edge weight $\leq \beta$. This is almost trivial: just delete all edges in G with weight $> \beta$ to obtain a subgraph G'. Then G has a BNST of the required type iff G' is connected. To check this connectedness condition, use any linear time search algorithm, such as DFS, on G'.

Part (c) We wish to find a BNST of a given graph connected graph G in linear time. The key step is to call the algorithm from part (b) passing it the *median* edge weight, w_m, of G. Recall that median finding can be done in linear time, so this does not require sorting the edges of G by weight.

Let $E_>$ denote the set of edges with weight $> w_m$. If G happens to have a BNST of weight $\leq w_m$, we can safely delete all edges in $E_>$; note that this gets rid of about half the edges of G. On the other hand, if G does not have such a BNST, then we can use DFS to find a spanning forest of $G - E_>$ and, in linear time, contract all such components to single vertices. Note that this also gets rid of about half the edges of G. Therefore, in either case, we may safely recurse on the remaining graph, for a total running time of something like

$$O(m + m/2 + m/4 + \cdots) \;=\; O(m)\,.$$

We keep track of all spanning forest edges we find during this algorithm and return the union of these edges as our BNST.

We claim that this algorithm is correct. Proof to come...

Problem 3

Almost everyone solved this problem perfectly. The main observation is that the arrangement of the m rooks on an $n \times n$ chessboard translates naturally into an m-edge bipartite graph on n left vertices $\{u_1, \ldots, u_n\}$ and n right vertices $\{v_1, \ldots, v_n\}$. We put an edge between u_i and v_j iff there is a rook at position (i, j) on the board.

Now, a subset of rooks has the property that no two rooks attack each other iff the corresponding subset of edges in the graph has the property that no two edges share a vertex, i.e., iff the corresponding subset of edges is a matching. Therefore the problem is simply maximum bipartite matching in disguise and we can use the algorithm we studied in class.

Problem 4

Again, almost everyone found a counterexample showing that Professor Nixon's algorithm (call it A_{NIX}) is not a 2-approximation. A few people gave constant-sized examples which are not very satisfactory, for reasons I have discussed in class. Others came up with an infinite family of examples, each of which had A_{NIX} producing a vertex cover of just over twice the optimal size. Only two students came up with really solid lower bounds on the approximation guarantee of A_{NIX}.

Khanh Do Ba gave a very original solution using Gronwall's Theorem (look it up on `mathworld.wolfram.com`) which showed that A_{NIX} is an $\Omega(\log \log n)$-approximation. Paritosh Kavathekar went one step futher and showed that in fact A_{NIX} is an $\Omega(\log n)$-approximation. This is optimal. Paritosh's solution cannot be improved because A_{NIX} happens to be equivalent to the greedy set cover algorithm applied to the vertex cover problem; therefore, it is an $O(\log n)$-approximation.

Construct a bipartite graph with n left vertices L and with several right vertices R; we shall eventually see what $|R|$ is. We think of R as partitioned into subsets $R_1, R_2, \ldots, R_n$, where $|R_i| = \lfloor n/i \rfloor$. Assume that each of these sets of vertices is totally ordered in some arbitrary way. Add edges to this graph as follows: for each $i \in \{1, \ldots, n\}$, connect the first vertex of R_i to the first i vertices of L, the next vertex of R_i to the next i vertices of L and so on. Thus, after all such edges have been added, the graph (G, say) will have the following properties:

1. Each vertex in R_i has degree exactly i.
2. For each $i \in \{1, \ldots, n\}$, each vertex in L is connected to at most one vertex in R_i. Also, the first vertex in L is connected to exactly one vertex in R_i.

We claim that if the tie-breaking amongst vertices of equal degree is done by an adversary, then A_{NIX} can be made to return $R_1 \cup \cdots \cup R_n$ as its choice for a vertex cover of G. To see this, let G_k be the subgraph of G induced by $L \cup (R_1 \cup \cdots \cup R_k)$, where $k \in \{1, \ldots, n\}$. Then, the above properties ensure that the maximum degree in G_k is exactly k. Therefore, assuming that at some point A_{NIX} is dealing with the graph G_k, an adversary can force A_{NIX} to pick all the vertices in R_k. Having done this, A_{NIX} is now left with the graph G_{k-1}.

Since A_{NIX} starts with the graph $G = G_n$, it can thus be made to pick all of R_n, then R_{n-1}, and so on down to R_1, causing it to return the vertex cover R. The size of this cover is

$$|R_1| + \cdots + |R_n| \;=\; \sum_{i=1}^{n} \left\lfloor \frac{n}{i} \right\rfloor \;>\; \sum_{i=1}^{n} \frac{n}{2i} \;=\; \frac{nH_n}{2} \;=\; \Omega(n \log n)\,,$$

whereas G has the much smaller vertex cover L of size n. Therefore, the approximation ratio of A_{NIX} is at least as bad as $\Omega(\log n)$.