Solutions to Extra Credit Problems

1. We represent the color of a node by swapping the left and right pointers of a red node and leaving alone the pointers of a black node. We can distinguish the left from the right child by comparing keys (that of the left child is smaller), which allows us to test in constant time whether the pointers are swapped and hence whether the node is red or black.

   This idea works for nodes with two children; we need to handle nodes with zero or one child differently. A node with one child must be black; its only child must be red. (Otherwise, the rules of red-black trees will be violated.) For a node with no children, we can make one of its left and right pointers point to itself (instead of being null) if the node is red.

2. (a) I will reduce the minimum path cover problem to the maximum-cardinality bipartite matching problem, rather than the maximum flow problem, using the same construction as in the problem hint. Given an acyclic graph G with vertex set $\{1, 2, \ldots, n\}$ and edge set E, construct a bipartite graph G' with vertex set $\{x_1, x_2, \ldots, x_n\} \cup \{y_1, y_2, \ldots, y_n\}$ and edge set $\{(x_i, y_j) | (i, j) \in E\}$. Find a maximum-cardinality matching M in G'. This takes $O(n^{1/2}m)$ time using the Hopcroft-Karp bipartite matching algorithm, where $m = |E|$. Let $P = \{(i, j) | (x_i, y_j) \in M\}$. I claim that the edge set P defines a set of paths in G. This is true because M is a matching, which means that any vertex j in G has at most one incoming edge (i, j) in P (corresponding to the at most one edge $(x_i, y_j) \in M$) and at most one outgoing edge (j, k) in P (corresponding to the at most one edge $(x_j, y_k) \in M$). The paths defined by P are all simple and contain no cycles, since G itself is acyclic. These paths form a path cover of G; each vertex of G incident to no edges of P is in a path by itself, consisting of one vertex and no edges. The number of paths in the cover is $n - |P|$, since each additional edge in P reduces the number of paths, initially n, by one. Conversely, the edges in any path cover correspond to a matching in G'. It follows that a minimum-cardinality matching in G' gives a minimum path cover in G. The time to construct P from M is $O(n)$, so the total time to find a minimum path cover is $O(n^{1/2}m)$.

   (b) The problem of finding a minimum path cover in a directed graph with cycles is NP-hard, because the Hamiltonian cycle problem can be reduced to it. Thus the polynomial-time algorithm in (a) must fail on graphs with cycles unless P=NP. The reason it fails is that if G has cycles, the edge set P in G corresponding to a matching M in G' may contain cycles. Set P can still be converted into a path cover (by deleting an edge from each cycle), but P need not be minimum even if M is maximum, because the tight relationship between the size of M and the size of the path cover breaks down.

Exercise: Find a efficient approximation algorithm for the minimum path cover problem in the cyclic case, an algorithm that runs in polynomial time and minimizes the number of paths to within a factor of two (or better, if you can do better).

3. Construct a directed graph G whose vertices are the literals x and $\neg x$ for each variable x in the formula, and with two edges $(\neg x, y)$ and $(\neg y, x)$ for every clause $(x \vee y)$. (Double negatives cancel, so that for example a clause $(x \vee \neg y)$ corresponds to the two edges $(\neg x, \neg y)$ and (y, x). Find the strongly connected components of G. Then the formula is satisfiable if and only if no variable x has both x and $\neg x$ in the same strong component. This algorithm runs in time linear in the number of clauses in the formula.

   We can verify the correctness of this algorithm as follows. There is a path from x to y in G if and only if there is a path from $\neg y$ to $\neg x$, as can be proved by induction on the path length. If there is a path from x to y, then in any satisfying assignment, if x is true then so is y. Again this can be proved by induction on the path length. If x and $\neg x$ are in the same strong conponent, there is a path from x to $\neg x$ and a path from $\neg x$ to x, which means that x must be both true and false in any satisfying assignment, which means that there is no satisfying assignment. Conversely, suppose that x and $\neg x$ are in different strong components for every variable x. Choose any strong component S with no incoming edges from any other component. Then S will have a dual component $\overline{S}$ containing the negations of all literals in S, with no edges from $\overline{S}$ to other components. Make all the literals in S false (and all the literals in $\overline{S}$ true). This assigns truth values to all the variables in S and $\overline{S}$ and makes all the clauses corresponding to the edges incident to S and $\overline{S}$ true. Delete S and $\overline{S}$. Repeat until all variables have truth assignments, and all clauses are true.