The Theory of Computation
Bernard M.E. Moret
Addison-Wesley-Longman, 1998

Solution Manual
Chapter 8, Problem 8.21

1. In order to "multiply" the problem here, we need to do more than merely copy the problem over and over, since the result would not even have an identification tree of any kind. We want more classes, because we want to feed to the claimed approximation algorithm a suitably larger instance. But we need all new classes tied to existing ones somehow, since all will be in the same tree. In particular, we need tests to distinguish not only between new classes in a "copy," but also tests to distinguish between "copies." Assume that the claimed algorithm guarantees a distance K approximation and let k be the smallest power of 2 such that $2^p = k > K$. We can solve an arbitrary instance $C, \mathcal{T}$ (where C is the set of classes and $\mathcal{T}$ the set of tests) exactly by using $\mathcal{A}$ as follows.
   We create a new instance $C', \mathcal{T}'$. C' is the disjoint union of $2^p = k$ copies of C; that is, if C equals $\{a, b \ldots, z\}$, then C' equals $\{a_1, a_2, \ldots, a_k, b_1, \ldots, b_k, \ldots, z_1, \ldots, z_k\}$. $\mathcal{T}'$ is a single copy of $\mathcal{T}$ where each class has been replaced by its k copies, plus p additional tests, call them $T_1, \ldots, T_p$, as follows.

$$T_1 = \{a_1, \ldots, a_{k/2}, b_1, \ldots, b_{k/2}, \ldots, z_1, \ldots, z_{k/2}\}$$

$$T_2 = \{a_1, a_2, \ldots, a_{k/4}, a_{1+k/2}, a_{2+k/2}, \ldots, a_{3k/4},$$
$$b_1, b_2, \ldots, b_{k/4}, b_{1+k/2}, b_{2+k/2}, \ldots, b_{3k/4}, \ldots, z_1, \ldots, z_{3k/4}\}$$

$$\ldots$$

$$T_p = \{a_1, a_3, a_5, \ldots, a_{k-1}, b_1, b_3, b_5, \ldots, b_{k-1}, \ldots, z_1, z_3, \ldots, z_{k-1}\}$$

   Note that these p tests among themselves distinguish any copy from any other copy, but cannot distinguish any two elements in the same copy. We claim that an optimal identification tree for this new instance uses these p tests in the first p levels (all complete), then has $2^p = k$ copies of the optimal identification tree for the original instance hanging from level p; if the total path length of the optimal identification tree for the original instance with n objects was m, the total path length of this new tree is $M = km + knp$. (Equivalently, we can use the optimal identification tree for the original instance and replace each of its leaves by a complete tree of p levels using tests $T_1, \ldots, T_p$ to distinguish between the 2^p copies of each class associated with that leaf. Yet other optimal trees exist, but they all share the property that all p tests, $T_1, \ldots, T_p$, appear on every path from the root.) A simple, but tedious, induction proof verifies that claim.

Now we have the desired effect. Applying $\mathscr{A}$ to the new instance produces a tree with total path length not exceeding $M + K$; since every class exists in 2^p copies and since only the p tests T_i can distinguish between copies, all of them must appear on every path, so that we can assume that they make up the top p levels of the tree. (If they do not, we can move them up the tree until they do.) Now, each of the k subtrees hanging from level p is a copy of the original instance and its optimal solution requires a subtree of total path length m. Since the total path length of the tree created by $\mathscr{A}$ does not exceed $M + K = km + knp + K$ and since the top p levels account for the knp term, the k copies have a total cost of $km + K$, with $K < k$; since each copy costs at least m, one copy must cost exactly m and thus has been solved optimally.

2. The problem here is that a straightforward replication of the graph into $k + 1$ copies will produce a disconnected graph, one that does not have any spanning tree. We can either correct that problem by introducing additional edges to link the copies (somewhat difficult because of the degree bound) or avoid it altogether by scaling the edge costs rather than the entire graph. If we multiply the cost of each edge in the original graph by $k + 1$ we multiply the cost of the optimal solution by the same factor. Denote the cost of the optimal solution in the original problem by c; the cost of the optimal solution in the scaled problem is then $(k + 1)c$. The answer returned on the scaled problem by an approximation algorithm with guaranteed distance of k has cost at most $(k + 1)c + k$. In terms of the original problem, then the cost of the solution returned (the exact same tree, but now with original edge costs) is at most $c + k/(k + 1)$; since that cost must be an integer, it is in fact at most c and thus must be optimal.

3. We again need to be careful in "multiplying" the problem. Mere copies would not work, since colors would be reusable from copy to copy. We still want copies, but must connect them so as to prevent reuse in other copies of any color used in a copy. The solution is to connect the copies in the analog of a complete graph—a complete graph forces a different color for each of its vertices and our similar construction will force a different set of colors for each copy. In other words, we treat each copy like a vertex of some higher-level graph, which we make completely connected. This operation can be viewed as multiplying the complete graph on n vertices (for n copies) by the graph in hand, with the result that each vertex of the first graph is replaced by a copy of the second. Wherever the first graph had an edge $\{u, v\}$, we connect every vertex of the copy corresponding to u to every vertex of the copy corresponding to v.

   Denote this operation by $G' = K_n[G]$, where G is our original graph and K_n denotes, as usual, the complete graph on n vertices. We claim that, if the chromatic number of G is $\chi(G)$, then the chromatic number of G' is $\chi(G') = n\chi(G)$. Observe that no color used in one copy of G can be used in any other copy, since every vertex of one copy of G is connected to all vertices of all other copies (due to the fact that our pattern graph is a complete graph). But, if that's the case, whatever legal coloring is used for one copy (and it must take at least $\chi(G)$ colors) uses its own set of colors, so that the minimum number of colors needed for G' is $d\chi(G)$; it is easy to see that these many colors suffice.

   Now that we have established this claim, the proof follows the usual lines. Given a claimed approximation algorithm $\mathscr{A}$ with constant distance guarantee d, we solve an arbitrary instance $G = (V, E)$ of chromatic number as follows. We create the new graph $G' = K_{d+1}[G]$ and run $\mathscr{A}$ on it; it returns a coloring with at most $\chi(G') + d = (d + 1)\chi(G) + d$ colors. Now, we know that each copy uses a different set of colors and that each copy requires at least $\chi(G)$ colors; so one of the copies must have been colored with exactly $\chi(G)$ colors.