1. In this problem, we represent a cycle in a graph by the set of edges in the cycle. Each edge $e$ is defined by the two endnodes of $e$. Then we can define the symmetric difference of two cycles in the graph as the set of edges that are in exactly one of the two cycles. More generally, the symmetric difference of a set of cycles is the set of edges which appear an odd number of times in the set of edges in the cycles.

In a plane embedding of a planar graph $G$, let $F$ be the set of cycles defined by the interior faces of the embedding of $G$. That is, each cycle in $F$ bounds one interior face of the embedding. Prove that every cycle in $G$ is the symmetric difference of some set of faces in $F$.

Answer: Let $C$ be a cycle in $G$. Clearly, in the embedding of $G$, $C$ encloses some set $I$ of finite faces, i.e., the faces in the interior (with respect to the external face) of $C$, including the faces which have an edge in $C$. Each edge in a face in $I$ is part of one or two faces in $I$. In particular, an edge $e$ is part of one face in $I$ if and only if $e$ is on $C$. Each face in $I$ defines a cycle in $F$, so the symmetric difference of the set of cycles in $F$ is precisely the set of edges on $C$.

2a. Convince yourself that a graph is bipartite if and only if every cycle has even length, and if and only if the nodes of the graph can be properly colored with two colors. Proper coloring means that no adjacent nodes have the same color. Are you convinced? You don’t have to turn in a written justification.

Answer: Yes I am convinced, because: a) If the graph $G$ can be properly two-colored, say with red and green, we can divide the nodes into the red and the green nodes, and since the coloring is proper, the only edges in $G$ go between a red and a green node. Therefore, $G$ is bipartite.

b) Conversely, if the graph is bipartite, then give the nodes on one side of the graph the color red, and give the nodes on the other side the color green.

Also c) If a graph is bipartite, then (after properly two-coloring it) every cycle must alternate between a red and a green node, and hence every cycle must be of even length.

Conversely d) If every cycle is of even length, then we will be able to properly two-color the graph as follows. Consider a DFS of $G$. The DFS partitions the edges of $G$ into the tree edges of the DFS and the back edges
of the DFS. Clearly, we can properly two-color the tree $T$ defined by the tree edges of the DFS - just alternate the colors as you go down a path in the tree. Every node in $G$ is in $T$, so this colors all the nodes of $G$. We need to show that it is a proper coloring of $G$. Now every edge not in $T$ is a back edge, and hence it forms a cycle when added to $T$. That cycle is in $G$ and hence it is of even length, and hence it connects two nodes of different color. Hence the coloring of $G$ is proper. So, $G$ must be bipartite.

2b. Prove that the faces of any plane embedding of a planar graph $G$ can be two-colored if and only if $G$ is Eulerian. The faces include the external face as well as the interior faces.

A hint here is that problem 1 is useful, as are the characterizations of a bipartite graph. Also, think dual.

Answer: First, recall that the above is the original form of the question. In fact, it is a true statement, as shown next. The issue that Tom pointed out is that it is uninteresting, that only trivial graphs obey the conditions of the theorem. Still, it is correct, and here is a proof:

A graph is bipartite if and only if it’s nodes can be two-colored, and also if and only if all cycles have an even number of nodes (edges). The problem of coloring the faces of a planar graph $G$ is the same as the problem of coloring the nodes of its dual graph $G^*$. So the faces of $G$ can be two-colored if and only if the nodes of $G^*$ can be two-colored, if and only if $G^*$ is bipartite, if and only if all cycles of $G^*$ have an even number of edges. Now if all cycles in $G^*$ have an even number of edges, then consider a cycle formed around one node $v$ in $G$. The number of edges in that cycle equals the degree of $v$ so the degree of $v$ is even.

To get the converse, we need the concept of a fundamental cycle. In the case of a planar graph, the set of cycles enclosing faces is a set of fundamental cycles. Then every other cycle is formed by the symmetric difference of fundamental cycles (as shown in problem 1), and so if every fundamental cycle has an even number of edges, then every cycle has an even number of edges. So if every node in $G$ has even degree, every fundamental cycle in $G^*$ has an even number of edges, so all cycles in $G^*$ have an even number of edges, so $G^*$ is bipartite and can be two-colored. QED

Now consider the updated version of the problem defined in the following email: I adapted this question from a theorem in the book. Finite Graphs and Networks, by Busacker and Saaty. The exact statement there is: “A necessary and sufficient condition for a map to be properly colorable (i.e.,
regions with a common edge colored differently) with two colors is that every vertex have even degree greater or equal to 2.”

We need to define a ‘map’ formally. Recall that a face in an embedding is a cycle that does not have an edge in its interior. With that, a map is an embedding of a planar graph so that every point on the plane (i.e., point here does not mean vertex) inside the external face, is contained in or on some face. This definition of map should rule out degenerate cases. Still, what is unstated in the Busacker and Saaty claim is whether the exterior region is also included. If the exterior region is not included, then the statement is untrue. On the other hand if the exterior region is included, then the statement is true, but uninteresting since it only applies in trivial cases (thanks to Tom for pointing out these facts). My guess is that what they intended was that the exterior region is included, and they didn’t realize the limited application of the theorem.

I think the following related statement is true and proving or disproving it is more interesting than the original question:

The interior faces of a map are properly colorable with two colors if and only if all of the interior nodes have even degree. A node is interior if it is not on an edge that is part of the exterior face of the map.

So, resolve that question in your homework. And if you have any other comments about the original question, write those up as well.

Answer: The proof of this is essentially the same as the proof of the original statement, but now we should define $G^*$ as the dual graph of $G$ minus the node in the dual graph that represents the exterior face of $G$, and all of the edges to that node.

2. Read the nicer proof of the contraction Lemma posted on the class website.

Then read the proof of Lemma 5.1 in Thomassen’s paper (i.e., Thomassen’s proof of the contraction Lemma). In that proof, he states ”But it is easy to see that that the smallest component of this graph is a subgraph of $G_1$ ... ”. I don’t find this easy to see. Give a full explanation for why this is true. The nicer proof of the contraction Lemma might help. If you see a relationship between these two proofs of the contraction Lemma, explain that as well.

3a. At the bottom of page 232, he states ”If $G/e$ contains a subdivision of $K_5$ or $K_{3,3}$ then it is easy to see that $G$ does also”. In class I got stuck at this point, but Chelsea pointed out the explanation for the case of when
$G/e$ contains a subdivision of $K_5$. Give a similar justification for the case of $K_{3,3}$.

3b. Read the full proof of what is marked as Theorem * on page 232 of Thomassen’s paper. Then answer the following multiple choice question:

I understand the proof: YES NO

Correct Answer: YES

4. Completion of the Algorithm to determine if $G$ is a line graph, and if so, to determine the root graph $H$.

Before continuing with the method, it is helpful to give a proof of the following claim from HW 4:

Lemma: Any clique $C$ can be in at most one proper decomposition of $G$, and, if $C$ is part of a proper decomposition $D(C)$, it’s easy to find all of the cliques in $D(C)$. Further, it’s easy to determine whether $C$ is part of some proper decomposition.

Proof: Suppose first that $C$ is a clique in some proper decomposition. We will show that $C$ is in only one proper decomposition, called $D(C)$, and we will show how to find all the cliques of $D(C)$. Let $G'$ be obtained by removing all the edges (but not the nodes) of $C$ from $G$. If $G'$ contains no edges, then $D(C)$ consists of $C$ alone. Otherwise, since $G$ is a connected graph, there is a node $v$ in $C$, adjacent to some other nodes (call them $N_v$) in $G'$. But, $v$ is already in one clique $C$, and so, in every proper decomposition containing $C$, the nodes $v \cup N_v$ are forced to be together in a single clique of $G'$. If not, then $v$ will be in more than two cliques, and no resulting decomposition could be proper. Hence, on the assumption that $G$ has a proper decomposition containing $C$, $N_v$ must be a clique in $G'$, and $v \cup N_v$ is forced to be a clique in every proper decomposition that $C$ is in.

This observation can be repeated until a proper clique decomposition of $G$ has been found. After each new clique is located, its edges are removed, and $G'$ is the graph of the remaining edges. Then either $G'$ has no edges, in which case a proper decomposition has been found, or there is a node $v$ that is already in exactly one clique, where $v$ is incident with other nodes (called $N_v$) in $G'$. $N_v \cup v$ must form a clique in $G'$, or else $G$ has no proper decomposition containing $C$.

Note that each clique found in this way, is forced to be in every proper decomposition containing $C$. To make this clear, let $C, C_1, C_2, ..., C_r$ be the cliques chosen. We showed that $C_1$ has to be in every proper decomposition containing $C$. Similarly, $C_2$ has to be in every proper decomposition
containing C and $C_1$, hence in every proper decomposition containing C. This reasoning continues, proving that $C, C_1, C_2, ..., C_r$ is the only possible proper decomposition containing C. Hence, if G has a proper decomposition containing C, it has only one, and the above method will find it. We will refer to the above method as Algorithm LD, and the proper decomposition containing C as $D(C)$.

The fact that LD always finds the same $D(C)$ is somewhat surprising, since, at any step in LD, there may be several choices for the next clique. It doesn’t matter: all choices lead finally to the same decomposition $D(C)$. However, it is not true that G can have only one proper clique decomposition, but it can have only one that contains C.

Now that we have established the Lemma from HW 4, we can continue with the development of the algorithm to determine if $G$ is a line graph. The problem is how to algorithm begins. Clearly, we would like to find a clique C that is guaranteed to be in some proper decomposition of G, if any exist. Then Algorithm LD will either find $D(C)$, or conclude that no proper decomposition of G exists. But how can we find such a clique $C$? We are now ready for the second idea.

Lemma Z: For any edge e in G, there is a set of at most two cliques of G, each containing e, such that any proper clique decomposition of G contains one of these cliques. Further, these two cliques are easily found.

Before proving Lemma Z, consider how it solves the question of how the algorithm should begin. Let $C_1$ and $C_2$ be the two candidate cliques that contain e, i.e., that might be in a proper clique decomposition $D$ of $G$. If $D$ exists, e must appear in one of its cliques, and so Lemma Z implies that either $C_1$ or $C_2$ is in $D$, i.e., $D = D(C_1)$ or $D = D(C_2)$. So a proper decomposition of G can be found, if one exists, by running algorithm LD at most twice: once starting with $C_1$, and once with $C_2$. If both attempts end in failure, then there is no proper decomposition of G.

To prove Lemma Z, for any edge $e = (v,w)$ in $G$, let $S(e)$ denote the set of nodes, other than v and w, that are adjacent to both v and w.

We will use $|S(e)|$ to find one, and sometimes two of the desired cliques $C_1$ and $C_2$. This is done by considering several cases.

Case 1: If $|S(e)| = 0$ then the only clique in $G$ containing e is e itself, so if $G$ has a proper decomposition, e must be part of it; $C_1$ is e itself.

Case 2: If $|S(e)| = 1$, then there are only two cliques in $G$ containing e:
$C_1$, the edge $e$ itself, or $C_2$, the clique formed by $v,w$ and $S(e)$. If $G$ has a proper clique decomposition, one of these two cliques must be part of it.

Case 3: $|S(e)| \geq 2$. This is the most complex case.

Now complete the analysis of this case, and finish the proof of Lemma Z. Then summarize the algorithm to determine if $G$ is a line graph and if so, to find the root graph $H$.

The answer is already posted on the class website.

5. When we discussed the museum traversal problem in class, we said that we could use depth-first-search to find a way to traverse each hallway in the museum, once in each direction. Now that we have defined depth-first-search, explicitly show how DFS can be used to find such a hallway traversal.

Answer: It is a simple property of DFS that it traverses each edge in both directions, when we correctly interpret DFS as a sequence of edge traversals. When the DFS discovers a back edge from $u$ to $v$, interpret that as a traversal from $u$ to $v$ and then a return traversal from $v$ to $u$. For a tree edge $(u,v)$, the first traversal of $(u,v)$ is when the recursive DFS($v$) is called from an invocation of DFS($u$), and the second traversal of $(u,v)$ is then the recursion backs up from $v$, i.e., when the DFS($v$) call is finished.

6. On page 185 in the posted notes on finding the biconnected components of a graph in linear time, Step 2 of Algorithm 5.3 says to “put edge $(v, w)$ on the STACK, if it is not already there.” There is a footnote to explain how to determine if $(v, w)$ is already in STACK.

Suppose we delete the phrase ”if it is not already there” from Step 2. Would the algorithm still work (i.e., output the edges that make up a biconnected component)? What is the real purpose of that check?

Answer: Originally, I thought (and wrote): “When edges are output by the algorithm, those edges still define a biconnected component, but an edge might be appear twice in the output. As far as I can see, the only purpose of the optimization in the algorithm is to avoid this duplication.”

However, in grading HW 5, some of the students explained that in fact the algorithm would not work correctly. One problem is that if the we leave the popping part of the algorithm the same, then the algorithm would only pop until it sees the first copy of an edge, and that would might leave a second copy in the stack to be popped where it does not belong. That problem could be solved by changing the algorithm to pop until both copies are found, if
there are two copies in the stack. So, for this we would need a data structure
to record how many copies are in the stack, in order to be within the right
time bound. But a more serious problem is that if we push whenever we see
an edge, the second copy could go into the stack at a point where it would be
popped and reported to be part of a biconnected component that it is not in.
An example of this is in Figure 5.12 of the DFS notes. The first time that
edge (4,7) is put into the stack is when the DFS is at node 7 and sees (4,7) as
a back edge. Then when the DFS backs up to node 4, it detects that node 4
is a cut node, and so pops the stack down to edge (4,5). That removes edge
(4,7) from the stack. But, because (4,7) is a back edge, we know that 7 has
not been seen on L(4), and so the DFS will look at (4,7). If (4,7) is put onto
the stack, it will be popped when the DFS backs up from node 4 to node 3,
because 3 is a cut node. But that would report that edge (4,7) is part of the
biconnected component that includes (3,4), which is not true. So this shows
that pushing both copies is wrong. It also shows that the statement in line
2 of algorithm 5.3 is in error. It is not correct to push an edge “if it is not
there already”, since (4,7) is not in the stack when the above scenario pushes
it for the second time. However, the more detailed rules in the footnote do
work, and would avoid pushing (4,7) for the second time in this example. So
a better statement for line 2 would have been: “if it has never been in the
stack”. Thanks to Roy Adams for seeing all of this.