CSE 548 Homework Assignment 5 Solutions/Hints

1 (Textbook *22.2-7*)
Claim. In a breadth-first search of an undirected graph, the following properties hold:

- There are no back edges and no forward edges.
- For each tree edge $\{u, v\}$, we have $d[v] = d[u] + 1$.
- For each cross edge $\{u, v\}$, we have $d[v] = d[u]$ or $d[v] = d[u] + 1$.

Now, we run the breadth-first search on the graph and label all vertices u with an even $d[u]$ as babyfaces and all vertices v with odd $d[v]$ as "heels". If there is an edge between two nodes of the same label, then we answer NO; otherwise, we answer YES. Note that if the graph is not bipartite, then there must be an edge between two babyfaces vertices or an edge between two "heels" vertices. So, our algorithm will report NO correctly.

Conversely, if the algorithm outputs NO, then G must have an edge $\{u, v\}$ with both $d[u]$ and $d[v]$ even or both odd. However, by the above claim, the only possible such cases happen when $\{u, v\}$ is a cross edge with $d[u] = d[v]$. So, the graph G contains a simple cycle of odd length: Let t be the lowest common ancestor of both u and v. Then, the path from t to u using tree edges, plus the edge $\{u, v\}$, plus the path from v to t following the reversed tree edges, form a cycle of odd length (because the distance from t to u and the distance from t to v are equal). It follows that G is not bipartite.

2 (Textbook *22.3-13*, modified)
Suppose we perform a DFS on the graph G starting with vertex u. If there is a forward or cross edge (v, w) in the DFS tree rooted at u (i.e., both v and w are descendants of u), then G has two paths from u to w: one uses the tree edges, the other uses tree edges from u to v followed by the forward or cross edge (v, w).

Conversely, assume G is not singly connected so that there are two paths from some node u to another node w. Then, we claim that if we perform a DFS on G starting with u, then the DFS tree with root u must contain a forward edge or a cross edge. To see this, we observe first that w must be in the DFS tree rooted at u, since w is reachable from u. Then, there must be a path $\pi : u \rightarrow w$ that contains at least one non-tree edge. Let the first non-tree edge in π be (x, y). Since this is the first non-tree edge in π, π must contain a subpath $\pi_1 : u \rightarrow x$ that consists of only tree edges. Therefore, (x, y) cannot be a back edge. (Suppose (x, y) were a back edge. Then, y would be an ancestor of x and would be in π_1,

and π would visit y twice and not a simple path.) It follows that (x, y) must be either a forward edge or a cross edge.

In addition, we observe that a forward or cross edge (x, y) can be recognized by checking that y is BLACK when the edge (x, y) is examined in line 4 of the procedure DFS-VISIT (see page 604 of the textbook). The observation suggests a $O(|V| \cdot |E|)$ algorithm to determine whether G is singly connected:

a. For each $u \in V$, run DFS-VISIT(u). (Note that, for each u, we only run DFS- VISIT(u), and when its finished, we do not check whether there are still white vertices in G. That is, we only visit all nodes in the DFS tree rooted at u. We do not visit other trees.)

b. During the execution of DFS-VISIT(u), we add a line between line 7 and line 8:
else if color[v]=BLACK **then** return NO .

c. If we finish all DFS-VISIT(u) without returning NO, then return YES.

3 (Textbook *22.4-3*, modified)

An undirected graph is acyclic (i.e., a forest) if and only if a DFS yields no back edges.

- If there is a back edge, there is a cycle.
- If there is no back edge, then by Theorem 22.10, there are only tree edges. Hence, the graph is acyclic.

Thus, we can run DFS: if we find a back edge, there is a cycle.

- Time: $O(V)$. If we ever see $|V|$ distinct edges, we must have seen a back edge because (by Theorem B.2 on p.1174) in an acyclic (undirected) forest, $|E| \leq |V| - 1$.

4 (Textbook *22-3 Euler tour*)

a It is clear that if there is an Euler cycle, then each time the cycle passes through a vertex u, it adds one in-degree and one out-degree to u. So, u must have the same in-degree as the out-degree.

Conversely, if a graph G has the property that each vertex u has the same in-degree as the out-degree, then we can create an Euler tour as follows: Starting from an arbitrary vertex, travel along edges so that each edge is used at most once, until it returns to the starting vertex. This can always be done because each vetex has the same number of in-edges and out-edges, and so the traveling route would not stop at any node other than the initial vertex. Now, this traveling creates a cycle C_0. Consider the graph G_1, which is the graph G with the edge in C_0 removed (denoted by $G \backslash C_0$), we note that G_1 has the property that each vertex has the same in-degree as the out-degree. In addition, each connected

component of G_1 must contain at least one vertex in C_0 (otherwise, this component would have been a separate component in the original graph G). Now, for each connected component of G_1, recursively find an Euler tour of it. Merge all these tours with cycle C_0, they become an Euler tour of G.

b The following pseudocode takes a connected, directed graph G with all vertices of equal positive in-degree and out-degree and returns an Euler circuit C of G.

EULER-CIRCUIT(G)
Choose a vertex $v \in V(G)$
Construct a cycle C in G with v as one vertex
while length$(C) < |E(G)|$ **do**
    Choose a vertex w on C of positive degree in $G \backslash C$
    Construct a cycle C_1 in $G \backslash C$ through w
    Merge C_1 to C at w to obtain a longer cycle C_2
    let $C \leftarrow C_2$
end
return C