9.2: DFS on a directed graph:

9.4:

Tree edges: (e,f), (f,d), (d,c), (c,a), (a,b), (g,h), (h,f)

Back edges: (b,c), (b,d), (c,e), (g,e), (h,f)

Forward edges: none

Cross edges: none
9.9:
Add a “time variable” $t$ while visiting vertices in DFS, and assign start and finish times to each vertex. Initially, start and finish times are set to -1 for all vertices. Every time when a vertex $v$ is visited directly after $u$, mark the edge $(u, v)$ as tree edge.
Then, while checking for every non-tree edge $(u, v)$ [in the direction $u$ to $v$, even though it might be an undirected graph]
   a. If $\text{start}(u) \neq -1$ and $\text{finish}(u) = -1$: Backward edge
   b. If $\text{start}(u) > \text{start}(v)$: Forward edge
   c. Else: Cross edge

9.13:
Assumption: a is the starting vertex
Algorithm 9.2 - ARTICPOINTS (modified)

Input: A connected undirected graph $G = (V,E)$.
Output: Array $A[1...count]$ containing the bridges of $G$, if any.

1. Let $s$ be the start vertex.
2. for each vertex $v \in V$
3. mark $v$ unvisited
4. end for
5. pred$m$ $\leftarrow 0$; count $\leftarrow 0$

Procedure $dfs(s)$

1. mark $v$ visited; bridge $\leftarrow$ false; pred$m$ $\leftarrow$ pred$m$ + 1
2. for each edge $(v,w) \in E$
3. REMOVE $(v,w)$
4. $dfs(w)$
5. if $G$ remains connected
6. bridge $\leftarrow$ true
7. end if
8. ADD $(v,w)$
9. end for
10. if bridge then
11. count $\leftarrow$ count + 1
12. $A[count] \leftarrow (v,w)$
13. end if
9.23:
The algorithm will be similar to 9.9. Since BFS visits every adjacent unvisited vertex, there will be no forward edges.

9.30:
The algorithm will be similar to 9.9. Since BFS visits every adjacent unvisited vertex, there will be no forward edges.

9.34:
The proof is by contradiction. We will assume that it is known that BFS tree rooted at r gives shortest paths to all vertices in the graph, if all edge weights are 1 (or any fixed positive number). This is not difficult to show, using the key property that BFS visits all adjacent unvisited vertices at any point. Given this, assume that there exists a spanning tree T' having height shorter than that of BFS tree rooted at r, say T. Therefore, there exists a vertex v, that is at height h' in T' such that h'(v) < h(v) in T. This is a contradiction, because we know that BFS tree contains shortest path from r to v, but r to v in T' is even shorter. Therefore, height of BFS tree is minimum over all spanning trees rooted at r.