Chapter 3, Problem 3.1: tree edge: solid black; back edge: dotted red.
Chapter 3, Problem 3.2(a,b): tree edge: solid black; back edge: dotted red; forward edge: dashed blue; cross edge: dashed orange.
Chapter 3, Problem 3.5: Algorithm:

```plaintext
procedure reverse_graph(G)
    \( G^R = \text{new\_graph()} \) /* initialize an empty graph */
    for all \( v \in V \):
        add \( v \) to \( G^R \) /* create a new adjacency list for \( v \) */
    for all \( v \in V \):
        for each edge \((v, u) \in E\):
            add edge \((u, v)\) to \( G^R \) /* add \( u \) in the adjacency list for \( v \) in \( G^R \) */
    return \( G^R \)
```

Time analysis: The first loop takes \( O(|V|) \). In the second loop, each edge is scanned once and the reversed edge is added once, and adding an edge using adjacency list takes \( O(1) \), thus the total time is \( O(|E|) \). The overall running time of the algorithm is therefore \( O(|V| + |E|) \), which is linear.

Chapter 3, Problem 3.6(a): Proof. Since each edge is incident upon two vertices in an undirected graph, each edge is counted twice in the total degree of vertices. Thus the sum of all degrees of vertices is equal to the twice the number of edges.

Chapter 3, Problem 3.6(b): Proof. Suppose there are an odd number of vertices whose degree is odd. The sum of degrees of these vertices must be odd. The sum of degrees of remaining vertices must be even since each of the remaining vertex has an even degree. The total degree of all vertices is therefore odd (odd plus even is odd), contradicting (a).

Chapter 3, Problem 3.6(c): No. A counterexample: a digraph with two vertices \( u \) and \( v \), and a single edge \( u \rightarrow v \).