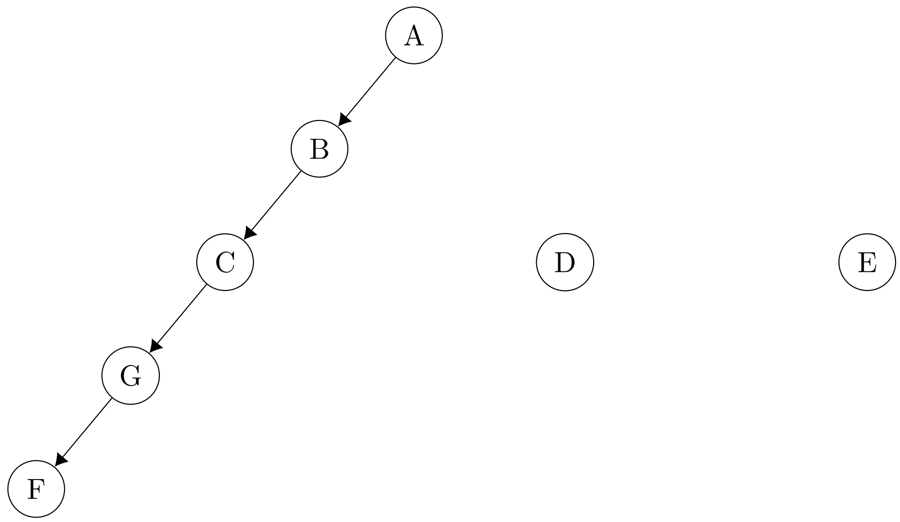
Problem 1.

(a) The BFS tree is shown below.

Figure 1: BFS tree.



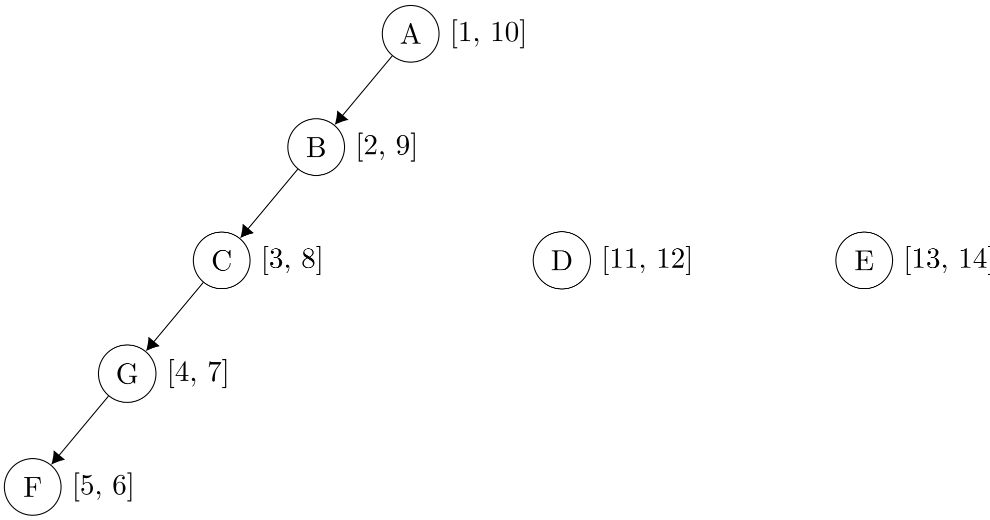
The classification of each edge is given below.

Table 1: Classification of each edge for the BFS tree.

Edge	Classification
(A, B)	Tree edge
(B, C)	Tree edge
(C, A)	Back edge
(C, G)	Tree edge
(D, A)	Cross edge
(D, C)	Cross edge
(D, F)	Cross edge
(E, B)	Cross edge
(E, C)	Cross edge
(E, G)	Cross edge
(F, C)	Back edge
(G, F)	Tree edge

The DFS tree is shown below.

Figure 2: DFS tree.



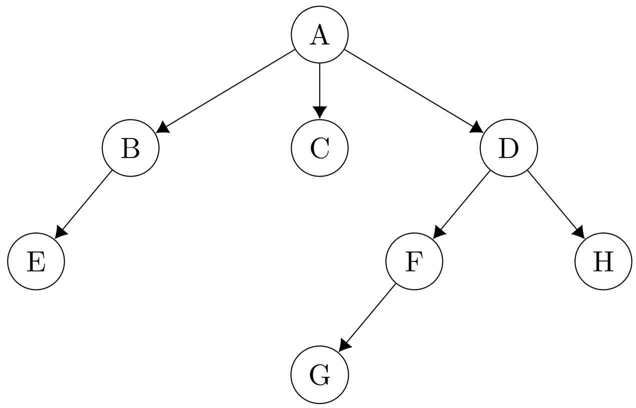
The classification of each edge is given below.

Table 2: Classification of each edge for the DFS tree.

Edge	**Classification**
(A, B)	Tree edge
(B, C)	Tree edge
(C, A)	Back edge
(C, G)	Tree edge
(D, A)	Cross edge
(D, C)	Cross edge
(D, F)	Cross edge
(E, B)	Cross edge
(E, C)	Cross edge
(E, G)	Cross edge
(F, C)	Back edge
(G, F)	Tree edge

(b) The BFS tree is shown below.

Figure 3: BFS tree.



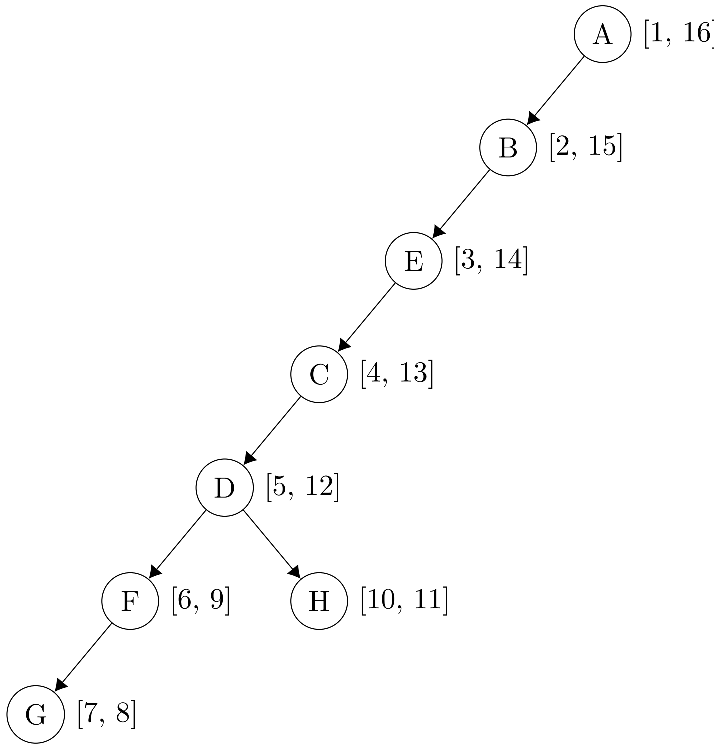
The classification of each edge is given below.

Table 3: Classification of each edge for the BFS tree.

Edge	**Classification**
(A, B)	Tree edge
(A, C)	Tree edge
(A, D)	Tree edge
(B, E)	Tree edge
(C, D)	Cross edge
(D, F)	Tree edge
(D, H)	Tree edge
(E, A)	Back edge
(E, C)	Cross edge
(F, G)	Tree edge
(G, D)	Back edge
(H, C)	Cross edge
(H, G)	Cross edge

The DFS tree is shown below.

Figure 4: DFS tree.



The classification of each edge is given below.

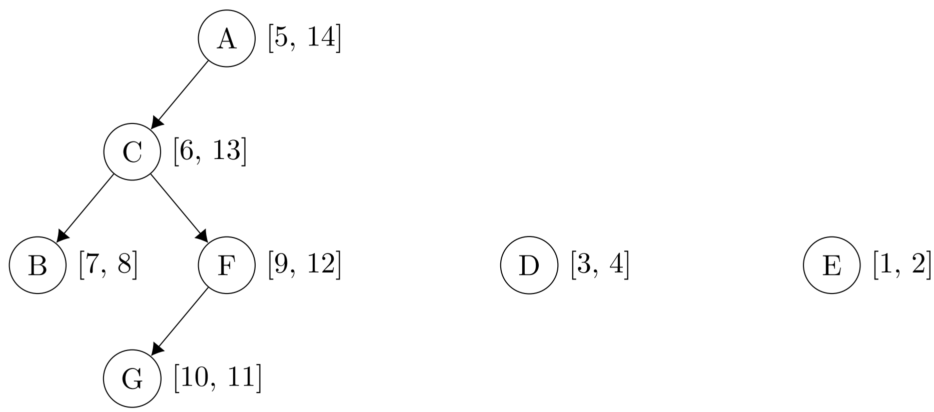
Table 4: Classification of each edge for the DFS tree.

Edge	**Classification**
(A, B)	Tree edge
(A, C)	Forward edge
(A, D)	Forward edge
(B, E)	Tree edge
(C, D)	Tree edge
(D, F)	Tree edge
(D, H)	Tree edge
(E, A)	Back edge
(E, C)	Tree edge
(F, G)	Tree edge
(G, D)	Back edge
(H, C)	Back edge
(H, G)	Cross edge

Problem 2.

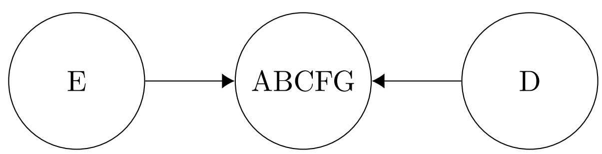
(a) The DFS tree of G^T produced by traversing nodes in decreasing $ftime$ is shown below.

Figure 5: DFS tree of G^T produced by traversing nodes in decreasing $ftime$.



The component graph illustrating the SCCs is shown below.

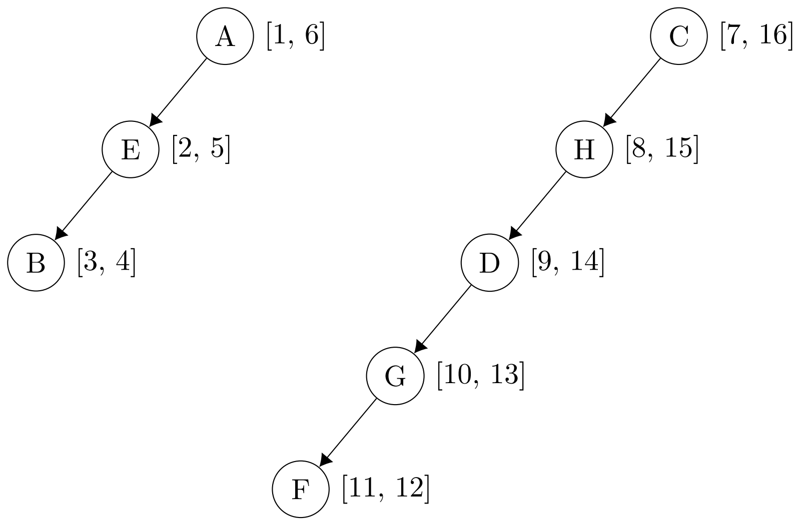
Figure 6: Component graph illustrating the SCCs.



A topological sorting of the component graph is E, D, ABCFG.

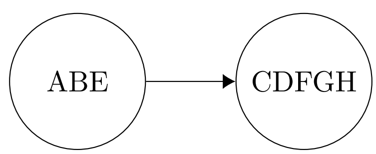
(b) The DFS tree of G^T produced by traversing nodes in decreasing $ftime$ is shown below.

Figure 7: DFS tree of G^T produced by traversing nodes in decreasing $ftime$.



The component graph illustrating the SCCs is shown below.

Figure 8: Component graph illustrating the SCCs.



A topological sorting of the component graph is ABE, CDFGH.

Problem 3.

a. We first select any node of graph G. We then apply BFS or DFS and count the number of nodes reached. If the number of nodes counted via BFS or DFS is not equal to n (the number of nodes in G) then the graph is disconnected.

b. For each vertex v in G:

1. Remove v from G.
2. Apply the method described in the previous question to see if the graph is disconnected. If the graph is disconnected, then v is a cut vertex.
3. Add v back to G.

c. The two DFS trees are shown below. Dotted links indicate non-tree edges.

Figure 9: DFS tree starting from vertex C.

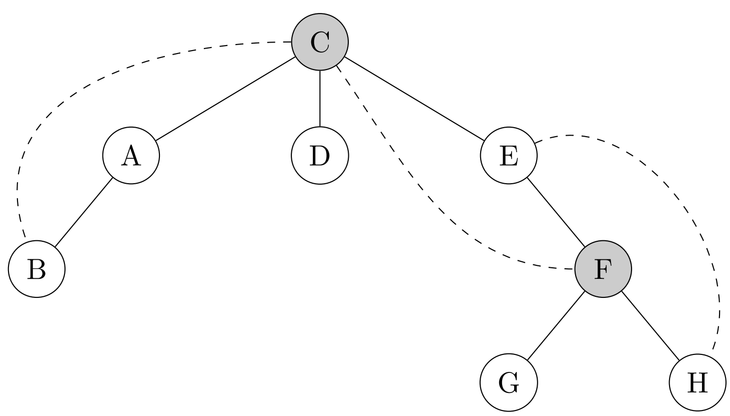
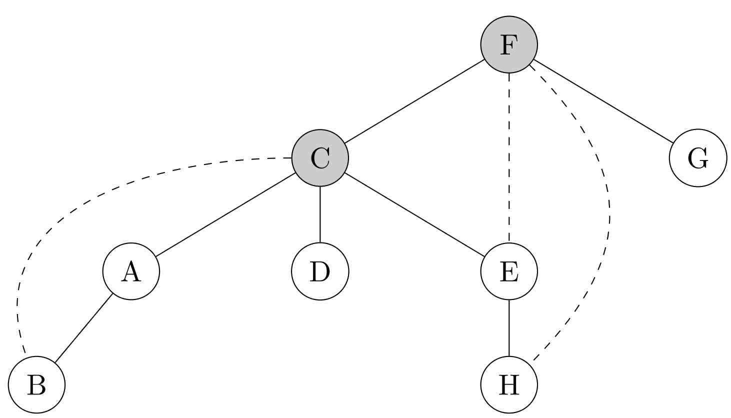


Figure 10: DFS tree starting from vertex F.



d. i) As discussed in the lecture slides, a DFS tree for an undirected graph contains only tree edges and back edges. If the root r of the DFS tree has at least 2 children $c_1, c_2, \ldots, c_n$, there can be no edges between the subtrees rooted at $c_1, c_2, \ldots, c_n$. This is due to the fact that an edge between any of the subtrees rooted at $c_1, c_2, \ldots, c_n$ would be cross edges and we know a DFS tree for an undirected graph cannot contain cross edges. Hence, the removal of the root r will cause the graph to become disconnected. Conversely, if the root r of the DFS tree has 1 child and it were to be removed, the rest of the DFS tree would still be connected. This means the entire graph would still be connected.

ii) If a vertex v is a leaf in the DFS tree, it means all vertices connected to v have already been visited. This means that removing v will not disconnect the original graph since a path exists between all neighbors of v. Hence a leaf in the DFS tree cannot be a cut vertex.

e. **Claim 1:** If u is a cut vertex, then there exists a subtree rooted at a child of u that has no back edges to any ancestor of u.

Proof of Claim 1: Assume for the sake of contradiction that all subtrees rooted at $c_1, \ldots, c_n$ have back edges to some ancestor $v_1, \ldots, v_n$ of u. Then each ancestor v_i is an ancestor of c_i meaning the removal of u will not cause the graph to be disconnected. This means u is not a cut vertex which contradicts the original assumption.

Claim 2: If there exists a subtree rooted at a child of u that has no back edges to any ancestor of u, then u is a cut vertex.

Proof of Claim 2: Note that a DFS on an undirected graph only produces tree edges and back edges meaning cross edges never occur. We also know if a subtree rooted at a child c of u has no back edges to any ancestor of u, then removing u would disconnect c from the rest of the graph. This is because

- Removing u would sever the only path between c and any ancestor of u and
- There can't be a connection between s and any other subtree since such a connection would be a cross edge (which we know can't happen).