1. (a) **Solution:**

Proof by induction.

We begin by showing that at any point during BFS, the set

$$T = \{\{v, v.p\} | v \in V, v.color \neq White\}$$

is a spanning tree for the non-white nodes. When BFS starts, the only non-white node is s and $s.p = NIL$, so T is a spanning tree for them. The next time a node v gets colored gray, its parent is a node u that was already gray (because all nodes in the queue are gray). Hence, the edge $\{u, v\}$ gets added to T. Since u was part of the tree before the new edge was added, T remains connected. Also, since v must have been white before the new edge was added, it was not previously part of the tree. Therefore, adding $\{u, v\}$ does not create a cycle. Hence T is still a spanning tree for the non-white nodes.

Now we simply need to show that, in a connected graph, there are no white nodes at the end of BFS. If this is the case, then what we have already proved implies that T spans the entire graph at the end of BFS. Assume for the sake of contradiction that there is a white node at the end of BFS. Then, there must be a white node v that is a neighbour of a non-white node u (since we know that the graph is connected and there is at least one non-white node). If u is non-white, there must be a point during BFS when it gets dequeued. But at this point, all white neighbours of u (including v) get coloured gray, so we have a contradiction.

(b) **Solution:** Yes, T is a minimum spanning tree for (G, w). We can prove this using the same induction as in (a): When BFS starts, the only non-white node is s and $s.p = NIL$, so T is a trivial minimum spanning tree for the only non-white node s. Assume that T is a MST for the non-white nodes, and so sub-graph of some MST for G after exploring some of the vertices in G. Suppose u is the next vertex that is dequeued. Consider an edge (u, v), where v is a white node. This edge is a cross edge because all white nodes, including v, are disconnected from all non-white nodes, including u, in T. (u, v) also has minimum weight among those edges with one white endpoint and one non-white endpoint because nodes are processed in order of their distance from s. Hence, (u, v) is a safe edge and after we add the edge, T is still a subset of some MST for G.

2. (a) Suppose v is an articulation point in G, and that removing v disconnects G into k connected components $C_1, C_2, \ldots, C_k$, where $k \geq 2$.

   - If DFS starts at v, it will examine every vertex in some connected component C_i without reaching any vertex in another connected component (because the only way to get from one connected component to another is to go through v). This means v will have one child for each connected component.
   - If DFS does not start at v, it begins in some connected component C_i and will eventually reach v. Some of v's children may be other vertices in C_i but other children will belong to other connected components. None of these will have any back edge to an ancestor of v because this would connect a vertex from one connected component to another—a contradiction.

   Now, suppose v is the root of T_{DFS} and has more than one child. Let $u_1, \ldots, u_k$ be the children of v in T_{DFS}. Because DFS does not generate any *cross* edges or *forward* edges when running on an undirected graph, there cannot be any edge from any vertex in u_i's subtree to any vertex in u_j's subtree, for all $u_i \neq u_j$—else they would not be separate children of v. So v is an articulation point: removing v from G creates k connected components.

   Finally, suppose v is not the root of T_{DFS} and there is some child u of v in T_{DFS} such that no descendant of u has a back edge to any proper ancestor of v. If we remove v from G, then no vertex in u's subtree has an edge to any vertex outside of u's subtree—because T_{DFS} does not contain any cross or forward edges, as noted above. Hence, by definition, v is an articulation point.

   (b) Suppose $e = \{u, v\}$ is a bridge. Then $G - e$ contains two connected components. So $G - e$ contains *no* path from u to v, which means there is no cycle containing e in G.

   Suppose $e = \{u, v\}$ is not a bridge. By definition, $G - e$ is connected. By the definition of connected graphs, there exists a path P from u to v (that does not contain the edge e) in $G - e$. Therefore, there is a cycle in G consisting of the path P and the edge e.

Practice Questions

The following questions will **NOT** be marked. Please do **NOT** submit your answer for these questions. Note that you are **expected to work on these questions**, although you are **not required to submit your solutions**. Some test or exam questions can be related to these practice questions, although of course they are expected to be easier than the practice questions.

- First, we modify DFS to compute the m values, as defined in the hint.

```
DF-m(G):
    for u ∈ V:
        u.m ← u.f ← u.d ← ∞
        u.p ← NIL
    t ← 0   # global
    for u ∈ V:
        if u.d = ∞:   # u is "undiscovered"
            m ← Visit-m(G, u)
            if m < u.m:
                u.m ← m

Visit-m(G, u):
    u.m ← u.d ← t ← t + 1
    for v ∈ G.Adj[u]:
        if v.d = ∞:
            v.p ← u
            m ← Visit-m(G, v)
            if m < u.m:
                u.m ← m
    u.f ← t ← t + 1
    return u.m
```

Note that DF-m runs in worst case time $\Theta(m + n)$, just like regular DFS.

Now, we use the results from previous parts to find articulation points and bridges.

For articulation points, first run DF-$m(G)$ to compute the m values. Then, use the result from part (a) directly: the root r of the depth first tree is an articulation point iff r has more than one child; for every vertex $v \neq r$ in the depth first tree, v is an articulation point iff v has some child u with $u.m \geqslant v.d$ (indicating that no vertex in u's subtree has a back edge to a proper ancestor of v). This can be done with a single pre-order traversal of T_{DFS}, which takes time $\Theta(n)$ in addition to DFS, for a total of $\Theta(n + m)$.

For bridges, note that every back edge encountered during DFS is part of a cycle—so it is not a bridge. And every tree edge $\{v.p, v\}$ is on a cycle iff v's subtree has some back edge to an ancestor of v—in which case $v.p.m < v.p.d$. To identify the bridges, first run DF-$m(G)$ and mark every back edge as "cyclic" (part of some cycle). Then traverse the depth first tree T_{DFS} and mark every edge $\{v.p, v\}$ where $v.p.m < v.p.d$ as "cyclic." The bridges are all of the edges that were *not* marked "cyclic" and it takes time $\Theta(m)$ to identify all of them, following the $\Theta(n)$ time for traversing T_{DFS} and the $\Theta(n+m)$ time for DFS. The total time is therefore $\Theta(n + m)$.

- Denote the cycle as $C = (V_C, E_C)$, and assume that there exists an MST that includes e (otherwise, if such an MST does not exist, the proposition is proven automatically). Let $T = (V, E_T)$. Then there must exist an edge $e' \in E_C$ that is not in E_T, because T is acyclic (by definition of a spanning tree). (Note: if C is the only cycle in G then any edge in E_C and not E_T can be picked as e'; if C is not the only cycle in G, then we need to pick such an e' that it does not create another cycle in T, i.e., pick the e' that keeps T' connected.) Then since e is the maximum-weight edge in C, we have $w(e') \leqslant w(e)$. Therefore if we remove edge e from T and replace it with e' then we get a new subgraph $T' = T - \{e\} \cup \{e'\}$, which satisfies the following properties:

  - T' is a **tree**, since we don't change the number of edges in the tree (a connected undirected graph with n vertices is a tree if and only if it has exactly $n - 1$ edges, adding an edge would create a cycle, deleting an edge would disconnect it into a forest).
  - T' is a **spanning** tree, since the cycle C has only one edge e missing, which still covers all vertices in C, other vertices of G are covered by the unmodified part of T.
  - T' is a **minimum** spanning tree, because the total weight of T' is $w(T') = w(T) - w(e) + w(e') \leqslant w(T)$ because $w(e') \leqslant w(e)$. Since T is an MST, T' must be an MST.

  Hence we have an MST T' of G which does not include edge e, as desired.