1. Since G is connected and undirected, then G is a tree if G is acyclic.
WTS that G is acyclic if the BFS and DFS trees for G are the same.
Now we prove by contradiction: suppose G contains a circle, with vertices $\{V_1, V_2, \cdots, V_n\}$, and suppose this circle contains edges $\{(V_1,V_2),(V_2,V_3),\cdots,(V_{n-1},V_n),(V_n,V_1)\}$.

① We think of BFS first:
Suppose we have firstly reached some vertex in the circle for the first time, and we call it V_i. Since V_{i-1} & V_{i+1} are both 1 edges away from V_i, ($V_{i-1}=V_n$ if $i=1$, and $V_{i+1}=V_1$ if $i=n$), then V_{i-1} and V_{i+1} are both children of V_i, and they are siblings. Also, since V_i is the first vertex to be visited in the graph, and V_1–V_n are connected, then every vertex in V_1–V_n except V_i is a descendant of V_i.

② We then think of DFS:
(i) some other vertex V_j, where $j \neq i$, has been already visited. In this case, since there is a path from V_j to V_i, then V_i is a decendant of V_j.
(ii) V_i is also the first vertex to be visited in DFS: since V_{i-1} & V_{i+1} are connected in a cycle, and none of V_{i-1} – V_{i+1} has been visited, and there is at least one path $(V_{i+1}, (V_{i+1},V_{i+2}), V_{i+2}, \cdots, V_{n-1}, (V_{n-1},V_n), V_n, (V_n,V_1), V_1, \cdots, V_{i-1})$ connecting V_{i+1} and V_{i-1} without V_i. then either V_{i-1} is a descendant of V_{i+1}, or V_{i+1} is a descendant of V_{i-1}.
We see that neither of (i) and (ii) makes V_2 & V_n siblings.

Thus, ① and ② cannot be happen at the same time. Contradiction occurs here, so our assumption must be wrong, and G must be acyclic.

Now, since G is an acyclic undirected connected graph, G is automatically a tree. Proved

2. (a) We construct a new graph, G', which is based on G, such that for every original node v in G, we put two nodes, (v, no) and (v, yes) into G'. Here, "no" and "yes" means whether a black edge has been passed at the time we reach v. Namely, in G', we have $2*n$ # nodes. ← Construct G' should take $O(2n) = O(n)$, since we only duplicate nodes in G for 2 times.

Now we want to find some paths beginning at (s, no), to either (t, yes) or (t, no). To do so, we can use BFS.

We start at the node (s, no), we mark (s, no) as visited.

Now we use BFS to iterate through G' from (s, no), and insert each newly visited into our BFS TREE.

Then, at any node $v_i \in V(G')$, there are 2 situations:

(i): if we are at (v_i, no): Then we observe all not yet visited nodes v_j, which (v_i, no) can go to within 1 step. If the edge (v_i, v_j) is black in G, we insert (v_j, yes) to our BFS TREE, as a child of (v_i, no). If the edge (v_i, v_j) is white in G, we insert (v_j, no) to our BFS TREE, as a child of (v_i, no). Also, we mark the newly inserted nodes as visited.

(ii): if we are at (v_i, yes): Then we observe all not visited nodes v_j, which (v_i, yes) can go to within 1 step. If the edge (v_i, v_j) is black in G, we do nothing. If the edge (v_i, v_j) is white in G, we insert (v_j, yes) to our BFS TREE, as a child of (v_i, yes), ← We see that during our BFS, each $(v_i, yes/no)$ are visited only once.

Also, we mark the newly inserted nodes as visited. So for each edge in G, it will only be visited for at most 3 times: $(v_i, no) \to (v_j, no)$, $(v_i, no) \to (v_j, yes)$, or $(v_i, yes) \to (v_j, yes)$. $O(n+m)$.

If the node (t, no) or (t, yes) is hit for the first time, then we stop our BFS, and the path from root (s, no) to the leaf $(t, no)/(t, yes)$ is our shortest path. ← traversing back from a leaf to root is $O(n)$

Else: if after all iterations of our BFS are done and $(t, no)/(t, yes)$ is not shown in our BFS TREE, then there is no path from s to t with at most one black edge included. ← $O(1)$, since it's the else part.

Therefore, our algorithm is $O(n+m)$.

2(b) Similar to 2(a), we construct G', which is based on G, such that for every node $V_i \in V(G)$, there are $(V_i, 0)$, $(V_i, 1)$, $(V_i, 2)$, $(V_i, 3)$, $(V_i, 4)$ in our G'. The second element in each tuple indicates how many steps V_i is far away from node s, with module 5. ← To construct G', we only need to copy the original nodes in G for 5 times. $O(5n)$

Now we want to determine if there is a path in G' from $(s,0)$ to $(t,0)$. If so, we want to get the shortest such path. We can use BFS to iterate though G' from $(s,0)$, and insert each newly visited node into our BFS TREE.

To begin with, we mark $(s,0)$ as visited. ← $O(1)$

Then, at any node $(V_i, k) \in G'$, we do the following:

if there is an edge from V_i to V_j in G, and $(V_j, (k+1) \bmod 5)$ is not visited yet, we add $(V_j, (k+1) \bmod 5)$ as a child of (V_i, k) in our BFS TREE, and mark $(V_j, (k+1) \bmod 5)$ as visited.

If $(t,0)$ appears in our BFS TREE for the first time, then we have found the shortsest path from s to t with the number of steps as a multiple of 5. We terminate our BFS process, and the shortest path is the path from root $(s,0)$ to the leaf $(t,0)$ in our BFS TREE. ← traversing back from a leaf to root is $O(n)$

If our BFS has ended and $(t,0)$ is not shown, then there is no such path. ← $O(1)$

for each edges (V_i, V_j) in G, it will be visited for at most 5 times:

$((V_i,0),(V_j,1))$, $((V_i,1),(V_j,2))$, $((V_i,2),(V_j,3))$, $((V_i,3),(V_j,4))$, $((V_i,4),(V_j,5))$, $((V_i,5),(V_j,0))$

which is $O(5m)$.

In total: $O(m+n)$.

2(c): Similar to 2(a) and 2(b), we construct G', where each node is of the format (V_i, V_j) s.t. V_i keeps track of Alice's position and V_j keeps track of Bob's position.

Then, in G', the nodes are: $\{(V_i, V_j) \mid V_i, V_j \in V(G)\}$. ← $|V(G')| = n^2$, $O(n^2)$ to construct G'.

We want to find a shortest path from (s,t) to (t,s), and we can use BFS for this.

To begin with, we mark (s,t) as visited. ← $O(1)$

Then, at any node $(V_i, V_j) \in G'$, we do the following:

focus on these 2 sets $E_a \in G$ and $E_b \in G$ s.t. all elements in E_a are within one edge away from V_i (include i), and all elements in E_b are within one edge away from V_j (include V_j).

For all V_p in E_a:

Now we have 2 situations:

(i). if V_p and V_i have different colors:

observe the set $\{(V_p, V_q) \mid V_q \in E_b \wedge V_q$ & V_j have differen colors$)$, we insert all not-yet-visited nodes from this set into our BFS TREE as children of (V_i, V_j).

Mark all these new inserted nodes as visited.

(ii). if V_p and V_i have the same color:

observe the set $\{(V_p, V_q) \mid V_q \in E_b \wedge V_q$ & V_i have the same colors$)$, we insert all not-yet-visited nodes from this set into our BFS TREE as children of (V_i, V_j). # notice here (V_i, V_q) & (V_p, V_j) also count.

Mark all these newly inserted nodes as visited.

If (t,s) appears for the first time, we terminate our BFS, and the path from root (s,t) to the leaf (t,s) is our shortest path. ← $O(n^2)$ to traverse at most n^2 nodes.

If BFS has finished and (s,t) not shown in our BFS TREE: path not exists. ← $O(1)$

In our BFS, since every node can only be visited once, we have $O(n^2)$.

Also, in our BFS, we go from (V_i, V_j) to (V_p, V_q) only if there are

corresponding edges from v_i to v_p and v_j to v_q in G. Since there are m edges in G, then each time we pick next node, there are at most $O(m^2)$ ways ($v_i \to v_p$ at most $O(m)$ ways, $v_j \to v_q$ at most $O(m)$ ways). Namely, there can be at most $O(m^2)$ edges in G', and each can only be visited once.

In total: $O(m^2 + n^2) = O((m+n)^2)$

3. Since there's no bridge in G, each vertex in G is inside some circle, so that if one edge is deleted, we can go from the other side of the circle. Then we pick one arbitrary $v \in V(G)$ and process DFS on G. We denote the edges appeared in the BFS TREE as $E(T)$. In the BFS TREE, we can always get access to any other vertex from the root, and we are always able to get access to any vertex from its ancestors. Now, consider $E/E(T)$, since all vertices in G are in some circle, so we can use some edges from $E/E(T)$ to complete the other half of the circle, which goes from the descendants to ancestors. Namely, there exists a path from every vertex to any others, so the graph is strongly connected.

Pseudocode:

```
G = (V, E)
Then we do DFS on G:
DFS(G, T, u): // T is the DFS TREE.
    u.visited()
    T.insert(u)
    for v ∈ {v | (u,v) ∈ E}:
        if !v.has_been_visited():
            DFS(G, T, v)
```

$O(m+n)$. Since every vertex and edge can only be visited once.

```
T = [] // empty tree        ← O(1)
DFS(G, T, v)                ← O(m+n) as shown above
for (u,v) in E(T):
    set a direct edge from u to v
for (u,v) in E\E(T):
    set a direct edge from v to u.
```

$O(n)$, since there are n vertices.

In total: $O(1)+O(m+n)+O(n)=O(m+n)$ linear.