Question1:
We can use BFS -> achieve a linear running time

a. Algorithm:

```
Set all nodes to unvisited
visited[size];
BFSQueue = new Queue ();
BFSQueue.enqueue(firstNode);

while ( BFSQueue is not empty) {
Successor = BFSQueue.dequeue();
    If (Successor has not be visited) {
     Visited[successor] = true;
          For (every child of successor) {
               if (child has not been visited) {
                    BFSQueue.enqueue(child);
               }
          }
}
```

b. Proof:
We can using proof by induction to show the BFS will find the shortest path.
-Base case: The 1st iteration of BFS will give a shortest path of 1(this will find all the children of the initial vertex)
-inductive hypothesis: BFS algorithm will find the shortest path for Kth iteration
-inductive case: BFS algorithm finds the shortest path for the K+1th iteration and node X is the one removed in this iteration

Distance (startVertex -> X) <= Distance (startVertex -> parentOfX + 1)
Distance (startVertex -> parentOfX + 1) <= Distance(startVertex -> parentOfX) +1

According to the inductive hypothesis, we know that startvertex -> parentOfX is the shortest path (because it is Kth iteration), therefore, (startVertex -> parentOfX) + 1 is also a shortest path.

c. The worst case scenario is that getting and putting into the queue for every vertex in the tree, in this case we need to perform removal from the queue (O(1)) for all edges and vertices and adding the successors of the vertex to the queue (O(1)) for all vertices. Then the running time would be O(V+E) where V and E is number of vertices and edges -> O(n+m).

Question2:

We can use proof by induction to prove this proposition.

-> Assumption:
Let C be any cycle in G, and we let E be the max weight edge in C, E is necessary for the MST.

->Proof:
We suppose that the max weight edge E belongs to MST *AnyTree*, then if we delete such an edge, the MST *AnyTree* would become disconnected (because of the property of MST -> connected, acyclic). When the MST becomes disconnected, there will be two components (e.g A and B), let A be one of the component, and some edge, for example X, in cycle C would only have one endpoint in A since A and B are cut. Then *NewTree* = *AnyTree* $\cup$ {X} - {E} would also be a spanning tree. And since we know that $W_X < W_E$, then the $Cost_{NewTree} < Cost_{AnyTree}$
-> This is a contradiction, removal of E didn't decrease the cost/weight of the MST, and we have a more optimal choice. Therefore, E is shown to be not necessary

// For grader: Below are just some of my other thoughts on this question.
-> The other way to think about this:
If E is necessary to MST, which means the MST wants to include it, then edge E needs to satisfy the cut property which states that: let S be any subset of the vertices and let E be a min weight such that there exists some other edge that is more maximal than it, and E should also have only one endpoint in S. As shown above, edge E is not the maximum and only contains one endpoint, then it should be necessary to MST. Vice versa, if E is the max weight edge, then I should not be necessary to MST, and MST should exclude it, such a cycle property is also demonstrated above.