PROBLEM ONE (Trees)

1. Recall the definition of a tree: a tree is a connected, undirected graph which has no cycles. Which of the following definitions are equivalent to this definition of a tree? For each, if it is not equivalent, give a counterexample. If it is equivalent, explain why.

(a) A tree is an undirected graph with no cycles and a cycle is formed if an edge is added to it.

Solution: This is equivalent. I am going to actually prove this, which I do not in any way expect you to do. If you are interested, read on. In the next paragraph, I will show that the original definition implies this property. In the following paragraph, I will show the other implication.

To show this, we will use, instead of the original definition, one which we know to be equivalent (from class): a tree is a graph with the property that every two vertices are connected by a unique path. Suppose that we have a graph \( T \) which satisfies this property (which we will call property (2)). We want to show that it satisfies the property in the statement of part (a) (we will call this property (a)). First of all, since \( T \) satisfies property (2), we know that it must satisfy the original definition of a tree (which we will call property (1)), so it cannot have any cycles. To complete showing that property (a) holds, we must show that if we add any edge to \( T \) then we will form a cycle. Suppose this is not true, i.e., that we can add some edge \( e \) to \( T \) and not form a cycle. Then this new graph \( T' \) (which also has the edge \( e \)) is still connected and has no cycles. Therefore it satisfies property (1). Since property (1) is equivalent to property (2), it must still satisfy property (2). Suppose that the edge that we added connected vertices \( v \) and \( w \). Then we have given a new path on which we can move and go from \( v \) to \( w \). Since \( T \) satisfied property (2) before we added the edge \( e \), there must have already been a path from \( v \) to \( w \). This implies that in the new graph \( T' \), there are at least two ways to move from \( v \) to \( w \). This means that property (2) is false for \( T' \), which is a contradiction, since we already said that it satisfied property (2). Therefore \( T \) satisfies property (a). This means that property (1) implies property (a).

Suppose conversely that some graph \( T \) satisfies property (a). This means, in particular, that \( T \) has no cycles. This is the second part of property (1). To show the other part (that \( T \) must be connected), assume that \( T \) is not connected. We will obtain a contradiction. Since \( T \) is not connected there must exist two vertices, \( v \) and \( w \), such
that there is no path from \( v \) to \( w \). Let us add an edge \( e \) between these two vertices to obtain a new graph \( T' \). Suppose that this creates a cycle in \( T' \). Then we can move from \( v \) to \( w \) on this cycle without using the edge \( e \). This implies that in the original graph \( T \), the vertices \( v \) and \( w \) were connected, which is a contradiction. Therefore, there is no cycle in \( T' \). But this is a contradiction, since \( T \) satisfies property (a). Therefore, \( T \) must be connected. This means that \( T \) satisfies property (1).

(b) A tree is an undirected graph which is connected but is no longer connected if we remove any one vertex and all edges touching it (assuming there is at least one vertex in the graph).

Solution: Not equivalent. Consider the graph which has only two vertices and one edge between them. This graph satisfies the definition of a tree but not the property in question.

(c) A tree is an undirected graph which is connected but is no longer connected if we remove any one edge (assuming there is at least one edge in the graph).

Solution: This is equivalent. Again, I will prove this. First I will show that every graph which satisfies the original definition (which I call property (1)) must satisfy this property (which I call property (c)).

Suppose \( T \) is a graph which satisfies property (1). Then it must be connected. This is the first part of property (c). We need only show that \( T \) is no longer connected if we remove an edge. Suppose that we remove an edge \( e \) (which connected vertices \( v \) and \( w \)) from \( T \) to form a new graph \( T' \). Suppose further, by way of contradiction, that \( T' \) is, in fact, connected. This means that there is a path from \( v \) to \( w \) in the graph \( T' \). This path cannot use the edge \( e \), since \( e \) is not in \( T' \). Therefore, there was a path from \( v \) to \( w \) in the original graph \( T \) which did not use the edge \( e \). But there was already a path from \( v \) to \( w \) in the graph \( T \): the path which consists of only the edge \( e \). Therefore, in \( T \), there were at least two paths connecting \( v \) and \( w \). This means that the property that every pair of vertices in \( T \) are connected by exactly one path (which we call property (2)) is false. This is a contradiction, since property (2) must hold (in fact it is equivalent to property (1)). Therefore \( T' \) must not be connected. This means that \( T \) satisfies property (c).

Suppose conversely that some graph \( T \) satisfies property (c). We will show that it must then satisfy property (1). Since \( T \) satisfies property (c), it must be connected. This is the first part of property (1), so we must only show the second part, that \( T \) has no cycles. Suppose, by way of contradiction, that \( T \) has a cycle. Remove any edge \( e \) (which connects two vertices, say \( v \) and \( w \)) from this cycle to
form a new graph $T'$. Since $T$ satisfies property (c), this new graph $T'$ must not be connected. Therefore there exist two vertices, say $x$ and $y$, which are not connected in $T'$. Since they were connected in $T$, this means that any path between $x$ and $y$ in $T$ must have gone through the edge $e$. Take one such path and call it $P$. It must be that $v$ is connected to either $x$ or $y$ on $P \setminus \{e\}$ (the path $P$ with the edge $e$ taken out). Without loss in generality, we may assume that $v$ is connected to $x$ on $P \setminus \{e\}$. This means that $w$ must be connected to $y$ on $P \setminus \{e\}$. But since the graph $T$ had a cycle containing $v$ and $w$, we must be able to move on this cycle without using the edge $e$ to get from $v$ to $w$. This means that $v$ and $w$ are connected in the graph $T'$. Since $x$ is connected to $v$ in $T'$, $v$ is connected to $w$ in $T'$, and $w$ is connected to $y$ in $T'$, it must be that $x$ is connected to $y$ in $T'$. This is a contradiction, since we already said that $x$ and $y$ are not connected in $T'$. Therefore, $T$ must have no cycles. In other words, $T$ satisfies property (1).

(d) A tree is an undirected graph which is connected but is no longer connected if we remove both one edge and one vertex (assuming there is at least one edge and one vertex in the graph).

Solution: This is not equivalent. Consider the graph below. It satisfies the property in question but it is not a tree. (Note that when we remove the vertex we must also remove all of its incident edges.)

![Diagram of a graph]

2. (a) Explain why every tree is a bipartite graph.

Solution: We can pick a root for the tree and organize the other vertices in levels, as we did for a binary tree. Each vertex at an odd numbered level cannot be connected to any other vertex at an odd numbered level. Similarly, each vertex at an even numbered level cannot be connected to any other vertex at an even numbered level. Therefore, if we split the vertices of our tree into those with an even numbered level and an odd numbered level, we may use these two sets in the definition of a bipartite graph.

(b) Using the tree below, can you write its adjacency matrix in such a way that the fact that the tree is bipartite is evident from the structure of the matrix?
Solution: Label the vertices as I have above (first I have labeled the vertices with odd numbered rows and then vertices with even numbered rows). Then the adjacency matrix $A$ is

\[
A = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\]

Notice that the matrix has a $5 \times 5$ block of zeros in the upper left and a $3 \times 3$ block of zeros in the lower right. Any adjacency matrix of this form must correspond to a bipartite graph (think about why!)

(c) Draw a tree which has an adjacency matrix $A$ with the property that

\[
A^2 = \begin{pmatrix}
1 & 0 & 1 & 0 & 0 & 0 \\
0 & 2 & 0 & 1 & 0 & 0 \\
1 & 0 & 2 & 0 & 1 & 1 \\
0 & 1 & 0 & 3 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1
\end{pmatrix}
\]

Solution: Since we can move from vertex 1 to itself in two steps in only one way, the vertex 1 must have degree 1. Since we can move from vertex 2 to itself in two steps in two ways, the vertex 2 must have degree 2. Similarly, the degree of vertex $i$ is the $i^{th}$ number on the diagonal. This means that the degrees of vertices 1, 2, 3, 4, 5, and 6 are 1, 2, 2, 3, 1, 1. A tree which satisfies this is given below. You can check that the square of its adjacency matrix is the same as that above.
3. In any tree, we call a vertex $v$ an outer vertex if its degree is equal to one. Otherwise, we call $v$ an inner vertex. For each of the below (they are separate problems), draw a tree which satisfies the properties listed. If it is not possible, explain why.

(a) 10 vertices with degrees 1, 2, 3, 4, 1, 1, 2, 3, 4, 1.
Solution: This is impossible. Since the tree has 10 vertices, it must have 9 edges. Therefore the total degrees of all vertices must be 18. But the degrees above add to 22.

(b) 5 inner vertices, 6 outer vertices.
Solution:

(c) 6 inner vertices, 5 outer vertices.
Solution:

(d) 5 inner vertices, 5 outer vertices, all vertices of degree at most 3.
Solution:
(e) 5 inner vertices, 7 outer vertices, all vertices of degree at most 3.
Solution:

\[ \text{Diagram} \]

**PROBLEM TWO (Tree Traversals)**
Consider the following algorithm:

<table>
<thead>
<tr>
<th>line</th>
<th>code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>order(rooted tree ( T )) { |</td>
</tr>
<tr>
<td>1</td>
<td>} |</td>
</tr>
<tr>
<td>2</td>
<td>if ( T \neq \emptyset ) { |</td>
</tr>
<tr>
<td>3</td>
<td>print out root’s number |</td>
</tr>
<tr>
<td>4</td>
<td>order(right subtree) |</td>
</tr>
<tr>
<td>5</td>
<td>order(left subtree) |</td>
</tr>
</tbody>
</table>

1. Draw a rooted tree with 8 vertices and assign the numbers 1 through 8 to the vertices so that the numbers print out in order as the above algorithm executes.

Solution:

\[ \text{Diagram} \]

2. Same question as 1., but after we modify the original algorithm by switching lines 3 and 4.

Solution:
3. Same question as 1., but after we modify the original algorithm by moving line 3 below line 5.

Solution:

```
4

7
6
8
5
3
2
1
```

4. How does this algorithm (and its above variants) differ from the traversal algorithms given in class?

Solution: The are exactly the same, except with “right” switched by “left”. That is, the first is a backward-ordered pre-order traversal, the second is a backward-ordered in-order traversal (and the correct tree in the problem is a backward-ordered binary search tree), and the third is a backward-ordered post-order traversal.