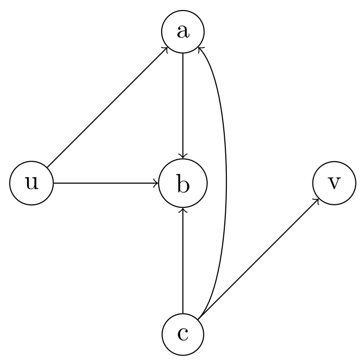
CS201 ASSIGNMENT 2

Name: Aditi Khandelia, Roll No.: 220061

31st October 2023

1.

Ans.



The above graph has no path from u to v or from v to u.
However, on disregarding edge directions, we get a path from u to v and vice versa. The path from u to v is {u,b,c,v}. Likewise, a path from v to u is {v,c,b,u}.

2.

Ans.
We say that a path exists from u to v if there is a sequence of vertices { u_0 , u_1 , u_1 , ... , u_k } where u_0 = u and u_k =v and for all i, (u_i,u_{i+1})∈ E.
Two vertices u and v are said to be related under the relation R iff there is a path from u to v and vice versa.

Claim: Relation R is an equivalence relation on the set of vertices.

Proof:
To prove that the relation is an equivalence relation we need to prove that it is reflexive,symmetric and transitive.

Reflexivity:
Consider the sequence {u}. This is the sequence of vertices where there is no repetition of vertices and the first vertex is the starting vertex and the last vertex is the destination vertex. Thus, there is a path from every vertex to itself.
Thus, uRu $\forall$ u $\in$ V.
Thus, the relation is reflexive.

Symmetry:
Consider any two vertices a and b such that aRb, i.e. there is a path from a to b and vice versa. Lert the first path be the sequence of vertices S_1 and the second path be the sequence of vertices S_2.
Now consider b and a. The path from b to a is S_2 and the path from a to b is S_1.
Thus, bRa whenever aRb $\forall$ a,b $\in$ V.

Transitivity:
Consider three vertices a, b and c $\in$ V such that aRb and bRc.
Thus, there is a path from a to b and from b to a.
There is also a path from b to c and vice versa.
Let the path from a to b be $(u_0 , u_1 , ... , u_k)$ where u_0 = a and u_k = b and $\forall$ i, $(u_i, u_{i+1}) \in$ E.
Similarly let the path from b to c be $(v_0 , v_1 , ... , v_m)$ where v_0 = b and v_m = c and $\forall$ i, $(v_i, v_{i+1}) \in$ E.
Consider the sequence of vertices $(u_0 , u_1 , ... , u_k , v_1 , ... , v_m)$. Here, u_0 = a and v_m = c. Also, consider any two adjacent vertices k and l in the sequence. Since any such pair already belongs to a path, (k,l) $\in$ E. Thus, this sequence of vertices is a path from a to c.
Let the path from b to a be $(w_0 , w_1 , ... , w_n)$ where w_0 = b and w_n = a and $\forall$ i, $(w_i, w_{i+1}) \in$ E.
Similarly let the path from c to b be $(x_0 , x_1 , ... , x_p)$ where x_0 = c and x_p = b and $\forall$ i, $(x_i, x_{i+1}) \in$ E.
Consider the sequence of vertices $(w_0 , w_1 , ... , w_n , x_1 , ... , x_p)$. Here, x_0 = c and x_p = a. Also, consider any two adjacent vertices t and r in the sequence. Since any such pair already belongs to a path, (t,r) $\in$ E. Thus, this sequence of vertices is a path from c to a.
Since there is a path from a to c and vice versa, aRc.

Since the relation is reflexive, symmetric and transitive, this is an equivalence relation on the set of vertices.

3.

Ans.
Given:
Two strongly connected components V_1 and V_2.

Claim:
All the edges between V_1 and V_2 are in the same direction, i.e., either all the edges are from the vertices of V_1 to the vertices of V_2 or from the vertices of V_2 to the vertices of V_1.

Proof:
Consider any two vertices a and b in V_1. Consider any two vertices c and d in V_2.
Similarly,let the path from c to d be S_3 and the path from d to c be S_4.
If possible, suppose there is an edge from a to c and an edge from d to b. Thus, there are edges in both directions.
Now, consider any vertex v_1 in V_1 and any vertex v_2 in V_2.

(i)
Since v_1 and a belong to the same strongly connected component, there is a path from v_1 to a. Let this path, which is a sequence of vertices be S_1=(v_1, v_2 , ... , a). By assumption there is an edge from a to c. Consider the sequence of vertices be (v_1, v_2 , ... , a , c). This is also a path since for any two adjacent vertices t_1 and t_2, (t_1,t_2) $\in$ E. Thus, there is a path from v_1 to c. Let this path be S_1'.
Now, since c and v_2 lie in the same strongly connected component, there is going to be a path from c to v_2. Let this path be S_2= (c,..., v_i,v_2). Again the concatenation of two paths, (v_1, v_2 , ... , a , c) and (c,..., v_i,v_2) produces a path (v_1, v_2 , ... , a , c , ..., v_i,v_2) since any two adjacent vertices represent an edge.
Thus, there is a path from any vertex v_1 in V_1 to any vertex v_2 in V_2.

(ii)
Again consider v_2 and d. Since they belong to the same strongly connected component, there is a path from v_2 to d. By assumption, there is an edge from d to b. As seen in (i), there will thus be a path from v_2 to b.
For any vertex v_1 in V_1, v)1 and b belong to the same strongly connected component, thus there is a path from b to v_1. Again, like in (i), concatenating the path from v_2 to b and from b to v_1 produces a path from v_2 to v_1.

There is a path from v_1 to v_2 and a path from v_2 to v_1.Thus, v_1Rv_2. Since this is true $\forall$ $v_1 \in V_1$ and $\forall$ $v_2 \in V_2$, V_1 and V_2 are the same strongly connected component.
We have reached a contradiction because of an incorrect consumption.
All the edges between V_1 and V_2 are in the same direction, i.e., either all the edges are from the vertices of V_1 to the vertices of V_2 or from the vertices of V_2 to the vertices of V_1.

4.

Ans.

Given:
A directed tree (V,E).

Claim:
For a directed tree |E|=|V|-1 and for any two vertices u,v $\in$ V such that u$\neq$v, if there is a path from u to v, then there is no path from v to u.

Proof:
(i) |E|=|V|-1:

Proof by induction:
P(1):
Consider a directed tree with 1 vertex. Since a directed tree must have atleast one vertex with indegree 0. Thus, the vertex cannot have an edge to itself. Since there are no other edges, it will not have an edge with any other vertex either. Thus, |E|=0=1-1=|V|-1.

P(n):
Suppose for a directed tree with n vertices, |E|=|V|-1.

P(n+1):
Consider a directed tree with (n+1) vertices.
If possible, let there be no vertex with outdegree 0. Thus, $\forall$ v $\in$ V, there is another vertex u $\in$ V such that there is an edge from v to u.
Let the vertex with zero indegree be v_0. Since the outdegree $\neq$ 0, there is a vertex v_1 such that (v_0,v_1) $\in$ E. Similarly, we get v_2 , v_3, ..., v_n. Now, consider v_{n+1}. If it has an edge to v_0, we have no vertex with indegree 0. If it has an edge to any of v_1, v_2,..., v_n, we have a vertex with indegree>1.
Thus, there must be atleast one vertex with outdegree 0.
Let this vertex be a.

Consider the graph G'(V',E') obtained by removing this vertex along with the associated edges. Since the indegree of this vertex is 1, we only remove one edge from the graph to it from some vertex b.

Since this vertex has outdegree 0, it doesn't lie on any path from two vertices in G'. Additionally, since the indegrees of the rest of the vertices remain unchanged, we still have a vertex with indegree 0 and the rest of the vertices have indegree 1.
Thus, G' is a directed tree with n vertices. Thus, |E'|=|V'|-1 $\rightarrow$ |E'|+1=|V'|+1-1 $\rightarrow$ |E|=|V|-1.
Thus, P(n+1) is true.
For a directed tree |E|=|V|-1.

(ii)For any two vertices u,v $\in$ V such that u$\neq$v, if there is a path from u to v, then there is no path from v to u.

If possible, let there be a path from u to v ,(u,u_1,...,u_k,v) as well as a path from v to u, (v, v_1, v_2,...,v_m, u). There are three possible cases: 1) either u or v is a root, 2)root neither lies on the path from u to v nor on the path from v to u and 3)root lies on the paths.

Case 1.
Since there is path from u to v, the indegree of v>0. Similarly, the indegree of u>0. Thus, neither can be the root.

Case 2.
Suppose the root is r. Since the graph is connected and root has indegree 0, there is a path from r to either u or v. Without loss of generality, assume its u. Let the path from r to u be (r, r_1,..., v_i,...,v_m,u).Consider the first common vertex between this path and the path to u from v.Let this vertex be v_i. Thus, we get atleast one vertex v_i, perhaps u itself, such that indegree(v_i)>1 since it has an edge from two different vertices.
This is not possible.

Case 3.
If root lies on either of the paths and $\neq$ u, v, we have an edge on root from atleast one other vertex. Thus, indegree(root)>0. This is not possible.

Thus, all three cases are not possible. This happens because of wrong assumption.
For any two vertices u,v $\in$ V such that u$\neq$v, if there is a path from u to v, then there is no path from v to u.

5.

Ans.
For a binary tree, the root will have two children.
Remove the edges from the root to its children. Thus, we get two graphs. Consider one of these graphs.

We will prove that this graph is a directed tree.
Cconnectivity:
Consider any two vertices a and b in the graph. Since they were originally a part of the directed binary tree T, they were connected. Consider the path from a to b (a,v_0,v_1,...,v_m,b).
If possible, let this path have vertices from the other graph as well. Consider the first such vertex v which has its predecessor u in the first graph. Such a vertex will surely exist since the path begins in the first graph.
Now, since v is in the second graph, its parent will lie in the second graph as well. Thus, v has one edge from a vertex in the first graph, and the otehr from a vertex in the second graph, implying that its indegree>1.Clearly, this is not possible.

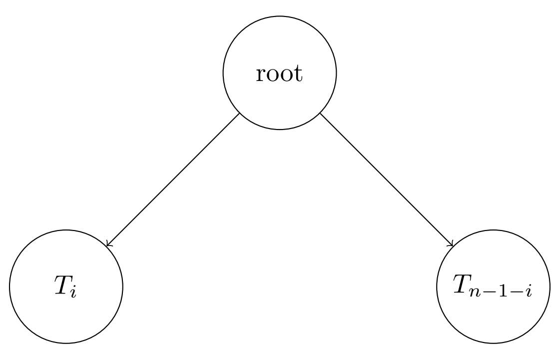
If possible, let the path contain the root. In this case, root will surely have a predecessor since the path begins with a, a vertex in the first graph. However, this means that root has an edge incident on it, implying that its indegree>0, which is not possible.

Thus, any path from a to b has vertices from the first graph only, and since we have not removed any edges from this graph, all the edges connecting these vertices remain intact.
Thus, the graph is connected.

Indegree analysis:
Consider the child c_1 of the root. It originally had indegree 1. However, upon removal of the edge from the root, its indegree is reduced to 0.
Consider any other vertex c in the graph rooted at c_1. Since the graph is connected, there is a path from c_1 to c. Thus, its indegree>0. Additionally, c was originally a part of the directed tree T, its indegree was 1. Since we have not added any edges, the indegree of any vertex can't increase, implying that the indegree of any vertex c=1.

As stated above, we have not added any edges, so outdegree of all vertices <=2.

Each of the two graphs obtained is a directed binary tree. The no. of vertices in the left subtree can vary from 0 to (n-1) for a binary tree with n vertices.



1. i will range from 0 to n−1. Thus, the total no of trees= $\sum_{i=0}^{n-1} G_i$, where G_i denotes the no. of trees when the size of the left subtree is i.

2. The configuration of the left subtree is independent of the configuration of the right subtree for any given i, hence they represent independent events. Thus, for a given i, no. of trees= $T_i \cdot T_{n-1-i}$

Thus, the recurrence relation T_n representing the no. of binary trees of size n is:

$$\sum_{i=0}^{n-1} T_i \cdot T_{n-1-i}$$

6.

Ans.
Let the required generating function $\sum_{n=0} T_n x^n$ be represented by F(x).
Thus, F(x) = T_0 + T_1x + $T_2 x^2$ ...
The base case is a directed tree with 0 vertices. The no. of ways to achieve this is 1.

As proven in the question above,

$$T_n = \begin{cases} 1 & \text{if } n = 0 \\ \sum_{i=0}^{n-1} T_i \cdot T_{n-1-i} & \text{if } n > 0 \end{cases}$$

Now, $(\text{F(x)})^2 = (\sum_{i=0} T_i x^i) \cdot (\sum_{j=0} T_j x^j) = \sum_{i=0} \sum_{j=0} T_i x^i T_j x^j = \sum_{i=0} \sum_{j=0} T_i T_j x^{i+j}$

Consider the n^{th} term of $(F(x))^2$. This is obtained when i+j = n.

Thus, ${(F(x))^2}_n = \sum_{j=0}^{n} T_{n-j} T_j x^n = T_{n+1} x^n$
Hence, $(F(x))^2 = T_1$ + T_2x + $T_3 x^2$...

Multiplying the equation of $(F(x))^2$ with x and subtracting from the equation of $F(x)$:

$$T_0 + T_1\text{x} + T_2x^2 \ldots - \text{x}*(T_1 + T_2\text{x} + T_3x^2\ldots)$$
$$= T_0$$
$$= 1$$
$$\text{Thus, } F(x) - (F(x))^2 = 1$$
$$(F(x))^2 - F(x) + 1 = 0$$
$$F(x) = \tfrac{1-\sqrt{1-4x}}{2x} \text{ or } F(x) = \tfrac{1+\sqrt{1-4x}}{2x}$$
First consider $F(x) = \frac{1+\sqrt{1-4x}}{2x}$
Multiplying the numerator and the denominator with $\frac{1-\sqrt{1-4x}}{1-\sqrt{1-4x}}$, we get:

$$F(x) = \tfrac{2}{1-\sqrt{1-4x}}$$

This is not possible since F(x) is then undefined at x=0. However, F(0)=T_0=1.

Now consider $F(x) = \frac{1-\sqrt{1-4x}}{2x}$
Multiplying the numerator and the denominator with $\frac{1+\sqrt{1-4x}}{1+\sqrt{1-4x}}$, we get:

$$F(x) = \frac{2}{1+\sqrt{1-4x}}$$

This gives F(0)=1=T_0
Thus, $F(x) = \frac{1-\sqrt{1-4x}}{2x}$ is the generating function for getting the no. of binary trees with n vertices.

T_n is given by the coefficient of x^n in F(x).
Coefficient of x^n= (coefficient of x^{n+1} in the numerator)/2.
Using binomial expansion, coefficient of x^{n+1} in the numerator =

$$= -\frac{\frac{1}{2} \cdot \left(\frac{1}{2}-1\right) \cdot \left(\frac{1}{2}-2\right) \cdot \ldots \cdot \left(\frac{1}{2}-n\right)\left(-4^{n+1}\right)}{(n+1)!}$$

$$= -\frac{(3) \cdot (5) \cdot \ldots \cdot (2n-1)\left(-4^{n+1}\right)\left(-1^{n}\right)}{(n+1)!\left(2^{n+1}\right)}$$

$$= \frac{(3) \cdot (5) \cdot \ldots \cdot (2n-1)\left(2^{n+1}\right)\left(-1^{2n+2}\right)(n)!}{(n+1)!\,(n)!}$$

$$= \frac{(1) \cdot (3) \cdot (2) \cdot (5) \cdot \ldots \cdot (2n-1) \cdot (n)\left(2^{n+1}\right)}{(n+1)!\,(n)!}$$

$$= \frac{(2) \cdot (3) \cdot (4) \cdot (5) \cdot \ldots \cdot (2n-1) \cdot (2n) \cdot 2}{(n+1)!\,(n)!}$$

$$= \frac{2n! \cdot 2}{(n+1)!\,(n)!}$$

$$= \frac{2n! \cdot 2}{(n+1)!\,(n)!}$$

$$= \binom{2n}{n}\frac{2}{n+1}$$

Dividing this with 2 from (2x), the final coefficient of x^n in F(x) is given by: $\frac{\binom{2n}{n}}{n+1}$.
Therefore, the no. of binary trees with n vertices:

$$\frac{\binom{2n}{n}}{n+1}$$

7.

Ans.
Given:

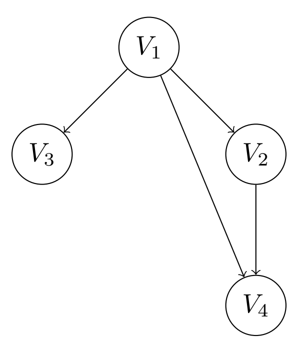
Consider a directed graph G = (V,E). Let V_1, V_2, ..., V_k be its strongly connected components. Define graph H = (V_H,E_H) as: V_H = {1,2,...,k}, and edge (i,j) $\in E_H$ if there is an edge from a vertex in V_i to a vertex in V_j.

Claim:
H is an undirected tree.

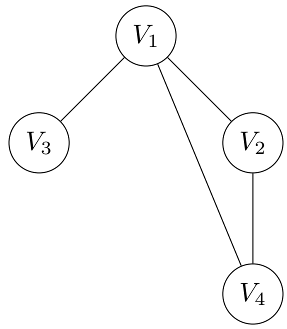
It can be proven that the claim is wrong using a counterexample.
Consider a directed graph with the strongly connected components V_1, V_2, V_3 and V_4. It has already been proven that all the edges from one strongly connected component to another can be in one direction only.
Thus, a possible case is a vertex in V_2 having an edge from a vertex in V_1, a vertex in V_3 having an edge from a vertex in V_1 and vertices in V_4 having edges from a vertex in V_1 and a vertex in V_3.
This arrangement assures that all the four strongly connected components are separate from one another.

1. V_3 is separate from V_1,V_2 and V_4.This is because there are no edges moving out from V_3 to any other strongly connected component.
2. V_4 is separate from V_1,V_2 and V_3.This is because there are no edges moving out from V_4 to any other strongly connected component.
3. V_2 is separate from V_1,V_4 and V_3. The only edge coming out from V_2 is to V_4, which it is separate as proven above. Additionally, V_4 has no edges coming out from it to any other component, implying that there can be no path from V_2 to any another component via V_4.
4. V_1 is separate from V_2,V_4 and V_3. Proven in the cases above.

Thus the following is an allowed G.



The corresponding H is:



Clearly this cannot be a tree since there are two paths from V_1 to V_4. One is (V_1,V_2,$V4$) and the other is (V_1,V_4).
Thus, the claim is wrong.

8.

Ans.
Given:
An undirected graph G(V,E).

Claim:
An undirected graph is a tree if and only if it is connected and does not have any cycle.

Proof:
(i) A tree is connected and does not have a cycle
Since there is a unique path from every vertex to every other vertex in a tree, it is connected.
Additionally, a tree is a graph where there is a unique path between any vertex v and u.
Suppose there are is a cycle in the graph (v_0,v_1,...,v_i,...,v_k,v_0). Consider any vertex v_i in this sequence. Since a path doesn't allow the repetition of vertices, there will be two different paths between v_0 and v_i. One is (v_0,v_1,...,v_i) and the other is (v_i,...,v_k,v_0).
Clearly this is not possible. We reach a contradicition because of a wrong assumption that the tree can have a cycle.
Thus, a tree is connected and does not have a cycle.

(ii) If an undirected graph is connected and doesn't have a cycle, it is a tree.
Consider any two vertices a,b $\in$ V. Since the graph is connected, there is atleast one path from a to b.
If possible, let there be multiple paths between a and b. Let any two paths be (a,v_0,...,v_i,b_0,...,b_m,v_{i+1},...,v_k,b) and (a,v_0,...,v_i,c_0,...,c_n,v_{i+1},...,v_k,b).
Here, b_0 is the first vertex of difference and v_{i+1} is the next common vertex. Since these paths are different (b_0,...,b_m) and (c_0,...,c_n) will surely exist and

have no common vertices.
Consider $(v_i,b_0,...,b_m,v_{i+1},c_n,...,c_0,v_i)$. Here the starting and ending evrtices are the same with no repetition of vertices in the middle of the sequence. Thus, this is a cycle.
This is a contradiciton that arises due to the wrong assumption that there might be multiple paths between any two vertices a,b $\in$ V.
The graph is connected and has a unique path between any two vertices.The graph is thus a tree.

Hence, an undirected graph is a tree if and only if it is connected and does not have any cycle.

9.

Ans.
Given:
A directed graph G(V,E).

Claim:
For a directed graph, all vertices in a cycle of the graph lie in the same strongly connected component.

Proof:
Suppose we have a cycle $(v_0,v_1,...,v_j,...,v_l,...,v_m,v_0)$ where $\forall$ i,v_i $\in$V and (v_i,v_{i+1}) $\in$ E.
Suppose $\exists$ j such that v_j lies in some strongly connected component U and v_j is in the cycle.
Consider any vertex u in U. There is a sequence of vertices which constitutes a path from u to v_j. Additionally, $(v_j,...,v_l)$ is a path from v_j to any vertex v_l in the cycle. Thus, there is a path from any vertex u in U to any vertex v_l in the cycle, obtained by concatenating the two sequences.

Since v_j and u lie in the same strongly connected component, there is a path from v_j to u. Now, a path from any vertex v_l to v_j is $(v_l,...,v_m,v_0,v_1,...,v_j)$. Thus, by concatenating the two paths, we get a path from any vertex in the cycle to any vertex in U.

Thus , all the vertices in the graph lie in the same strongly connected component.

10.

Ans.
Given:
An undirected graph G(V,E).

Claim:
An undirected graph has a spanning tree iff it is connected.

Proof:
(i)If G has a spanning tree, it is connected.
Suppose the spanning tree of the graph is (V,E'), where E'$\subset$ E.
Since this is a tree, it will contain a path from every vertex to every other vertex.
Consider any edge (a,b) in the path.Since (a,b) $\in$ E' and E'$\subset$ E, (a,b) $\in$ E.
The path exists in E as well. Thus, there is a path from every vertex to every other vertex in G, implying that G is connected.

(ii)If G is connected, it has a spanning tree.
We will first prove that a path necessarily exists if a walk exists between two vertices in a graph, where a walk of sequence of vertices composed of edges in the graph, but the repetition of vertices is allowed. Proof is by induction on no. of repeating vertices:
P'(0):
If we have a walk with 0 repeating vertices, the walk will itself be a path.
P'(n):
Assume that a walk with n repeating vertices has a path.
P'(n+1):
Consider a walk with (n+1) repeating vertices. Consider any such vertex v.
Then the walk is (a,b,...,v,u,...,w,v,...,c,d). Consider the sequence of vertices obtained after removing all the vertices between the two occurrences of v in the walk, as well as 1 v.
Since the sequence is still composed of edges in the graph, it is a walk. Additionally, since some of the other repeating vertices might be present in the vertices that were removed, no. of repeating vertices <= n.
We thus now have a walk with <= n repeating vertices. By assumption, there exists a path between a and d.
Thus, it is proven that a path exists between two vertices if a walk exists between them.

Now, we will prove that if G is connected it has a spanning tree.
Proof by Induction on the no. of edges:
P(0):
If the graph is connected and has 0 edges, it contains a single vertex, which is its own spanning tree.
P(n):
Assume that a graph that is connected and has n edges has a spanning tree.
P(n+1):
Consider a graph G(V,E) which is connected and has (n+1) edges.
If the graph has a tree, we are done.
If a connected graph doesn't have a tree, it must have a cycle.
Consider any cycle in the graph (v_0,v_1,...,v_k,v_{k+1},...,v_0).Remove the edge (v_k,v_{k+1}) from the graph to obtain the graph G'(V,E').

Suppose there was a path (u_0,u_1,...,v_k,v_{k+1},...,u_m) between two vertices (u_0,u_m) containing the edge (v_k,v_{k+1}).Replace this edge with (v_{k+1},...,v_0,v_1,...,v_k). We thus get a walk from (u_0,u_m). It has already been proven that a path will exist between (u_0,u_m) since a walk exists between them.
Thus, G' is connected. Additionally, it has n edges. By assumption it has a spanning tree which will be the spanning tree of G as well.

Thus, proven.

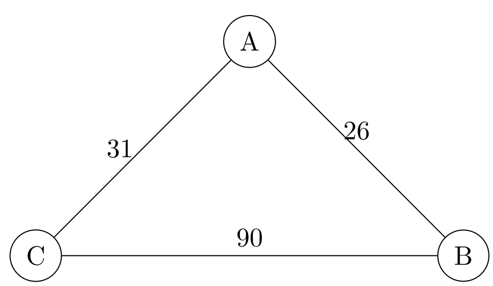
11.

Ans.
Given:
Two graphs G'(V,E') and G(V,E) such that E'⊆E. Thus, G' is a subgraph of G.

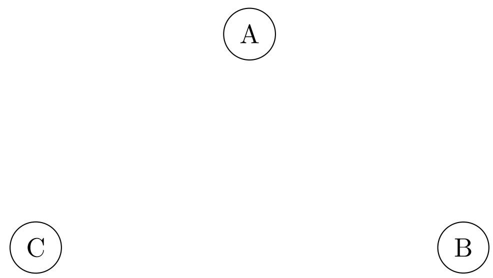
Claim:
When G' is a minimum weight subgraph, G' is a spanning tree.

Proof:
Consider the subgraph G' of G such that E'= $\emptyset$. If G is such that the weights of all the edges are positive, then G' is clearly the minimum weight subgraph of G. Consider the following example:

Graph G



Graph G'



Clearly G' is the minimum weight subgraph of G.However, it cannot be a minimum spanning tree since there is no path from A to B.

However, if the minimum weight subgraph is connected, we get the minimum spanning tree.

Proof by contradiction:
Consider G'.If possible let G' not be a tree. Since it is connected, it must have a cycle. We have already proven in Q. 10 that removing an edge from a cycle doesn't affect the connectivity of a graph. Thus, consider now the graph G"(V,E") obtained by removing one of the edges from one of the cycles of G'.

Since E" $\subseteq$ E' and E'$\subseteq$, and considering the transivitivity of $\subseteq$, we obtain E"$\subseteq$E. Thus, G"(V,E") is also a subgraph of G and since E" is obtained after removing one of the edges from E', weight(G")$\leq$weight(G).

This is a contradiction since we assumed that G' is the minimum weight connected subgraph.

Thus, the minimum weight subgraph can only be a tree.
Suppose the graph has a spanning tree T such that the W(T)$<$W(G').Now, a spanning tree is a subgraph that is a tree. We have already proven that a tree will be connected. Thus, T is a connected subgraph with weight less than G'. We have arrived at a contradiction since we assumed that G' was the minimum weight connected subgraph.

G' is thus the spanning tree with the least weight, implying that it is the minimum spanning tree of the graph.
12.

Ans.
Suppose the tree produced by algorithm is given by T and the minimum spanning tree of G(which will exist in a connected graph according to q.10), is T'.

If possible, let T$\neq$T'.

There must be atleast one edge (a,b) in T such that (a,b) $\notin$ T'. If this is not true, then T$\subset$T' and there is an (a,b)$\in$T' such that (a,b) $\notin$ T. Then, W(T')>= w(a,b) +W(T), which contradicts the assumption that T' is the minimum spanning tree, which is the spanning tree with the minimum weight. Thus,$\exists$ (a,b) such that (a,b)$\in$ T and (a,b)$\notin$ T'.

Consider the iteration of the loop when (a,b) is added to T.Suppose U represents the set of vertices that have been added upto this point.

When (a,b) is added, it is the minimum weight edge from U to V|U(since the algorithm picks the edge with the least weight).According to the algorithm, one of the vertices will lie in U and the other in V|U.Without loss of generality, assume that a$\in$ U.
Since T' is also a tree, there must be a path from a to b in T'. Since a lies in U and v lies in V |U, there will be one edge from some vertex a' in U to some vertex b' in V|U. Thus, T' has the edge (a',b'). Clearly, w((a',b'))>= w((a,b)).

Now consider the graph T" obtained by removing (a',b') from T' and instead replacing it with (a,b).
T" = T'-(a',b') +(a,b). We have already proven that removing an edge from a cycle preserves connectivity. Thus, this graph will also be a tree.
Now,weight of this tree= W(T')-w((a',b'))+w((a,b))<=W(T'), which contradicts the assumption that T' is the minimum spanning tree. This contradiction arises due to wrong assumption.
Thus, T produced by this algorithm is the minimum spanning tree.

13.

Ans.
Given:
The minimum spanning tree T(V,E') and minimum length tour T'(V,E") of a graph G(V,E).

Claim:
W(T)<=W(T')

Proof:
If possible, let W(T)>W(T').
Consider T'. Since the salesman is supposed to travel to all cities, the graph is connected. If T' itself is a tree, we have arrived at a contradiction since T is supposed to have the minimum weight amongst all trees.Otherwise, T' will itself contain a spanning tree according to q10. Let this spanning tree of T' be T".
Since T" is obtained from T' after removing some of the vertices of T', W(T")<=W(T').
Now, T' contains all the vertices of the graph G and T" is the spanning tree of

T'. Thus, T" is a tree containing all the vertices of G→T" is a spanning tree of G.
Now, by assumption, W(T)>W(T'). Additionally, W(T')>=W(T").
Combining the two inequalitites, we get W(T)>W(T"). Thus, T" is a spanning tree with weight less than the weight of the minimum spanning tree. This is clearly a contradiction, arrived at due to wrong assumption.

Weight of a minimum spanning tree of the graph is at most the minimum length tour.

14.

Ans.
Given:
The minimum spanning tree T(V,E') and minimum length tour T'(V,E") of a graph G(V,E).

Claim:
W(T')<=2W(T)

Proof:
This claim can be proven showing that there exists a path P that visits every vertex and returns to the original vertex such that W(P)<=2W(T). Since the minimum length tour has the minimum weight amongst all such paths, W(T')<=W(P).

Such a path P can be obtained through Depth First Traversal of the minimum spanning tree T.

The following procedure is followed, starting from any vertex v:

1. Visit an unvisited neighbour of v, u and mark u visited. If there are no unvisited neighbours, return to the previously visited vertex using the same edge as that used to visit the current vertex.

2. Carry out the Depth First Traversal of u.

3. Go to step 1.

   We will first prove that this algorithm visits all the vertices of the graph, and then prove that each edge is traversed exactly twice in this algorithm.

   Proof that the algorithm visits all the vertices:
   If possible, let us assume that v_i is a vertex reachable from v that doesn't get visited with atleast one neighbour that gets visited. Such a vertex will surely exist since Depth First Traversal surely visits v. Lets assume that the visited neighbour vertex of v_i is v_{i-1}.

When v_{i-1} gets visited, it visits all its unvisited neighbours during Depth First Traversal.Since v_i is an unvisited neighbour at this point, it surely gets visited. Thus, we arrive at a contradiction due to wrong assumption. Thus, Depth First Traversal will visit all the vertices in the graph.

Proof that each edge gets visited exactly twice:
We have already proven that each vertex gets visited atleast once. Since we are traversing a tree, all the edges will have to be traversed atleast once to visit all the vertices.This is because suppose an edge (a,b) doesn't get visited.We have already proven that all the edges in the tree will get visited. Thus, there will be a path from the root to the vertex a, as well as a path from the root to the vertex b, neither of which will contain the edge (a,b).Concatenating these to paths will result in a walk from a to b, which in turn gives a path different than (a,b).Since we also have the edge (a,b), we clearly get a cycle, which is not possible in a tree. Thus, all the edges get visited atleast once.
Additionally, after all the vertices reachable from a particular vertex are traversed, we return to the parent of the vertex using the same edge as that used while visiting the current vertex. So, every edge gets visited twice.
Since after returning the call to the parent of the current vertex, the current vertex has already been marked visited, the algorithm doesn't go back to the current vertex. Thus, the edge is never used again.

So, we can conclude that every edge gets visited exactly twice$\rightarrow$ W(P)=2W(T).
Now, W(T')<=W(P).
$\rightarrow$ W(T')<=2W(T)

Minimum length tour is at most twice the weight of a minimum spanning tree.