CS201 Assignment 2

Pragati Agrawal

October 2023

Question 1

Consider the directed graph G, having 4 vertices: $\{1, 2, 3, 4\}$ and 3 edges, as:

- $1 \to 2$
- $2 \to 3$
- $4 \to 3$

Figure 1: Left: Directed graph G, Right: Undirected graph G

Now, consider the vertices $u = 1$ and $v = 4$. Using the edges, one can clearly see that there is no path from $u \to v$ or $v \to u$. But if we consider the graph to be undirected, then a path exists from $u \to v$ and also from $v \to u$.

Question 2

Relation R on vertices of a directed graph is defined as uRv if there is a path from u to v and vice versa. Now to show that the relation R is an equivalence relation:

- *Symmetric*: In any graph, we know that if uRv, then a path exists from $u \to v$ and also from $v \to u$, i.e. vRu. A relation R is called symmetric iff $uRv \Rightarrow vRu$. This is directly implied from the definition of R. Hence R is *symmetric*.
- $Reflexive$: If we consider the graph to be simple, then no edge exists from $u \to u$. So the path from u to u, i.e. uRu would be just u. Hence R is $reflexive$.
  Otherwise, if there exists a path (actually a cycle) uRu, i.e. $(u = v_0, v_1, v_2, ..., v_k = u)$ then the same path can be back-tracked to reach from $(u = v_k, v_{k-1}, ..., v_1, v_0 = u)$. So uRu, and hence, R is $reflexive$.

- *Transitive* : In the graph, let uRv and vRw. Transitiveness implies uRw, so we need to show that paths $u \to w$ and $w \to u$ exist. We have assumed that paths exist from $u \to v$ and also $v \to w$. So a path exists from $u(\to v) \to w$, hence $u \to w$. Similarly, since paths exist from $v \to u$ and also $w \to v$, we can see that the path $w(\to v) \to u$, hence $w \to u$ exists. Hence uRv and vRw implies uRw and the relation is transitive.

Hence R is reflexive, symmetric and transitive. Therefore R is an equivalence relation.

Question 3

Since R is an equivalence relation, it splits the set V into disjoint equivalence classes. These equivalence classes are each strongly connected components. Now, we take two such equivalence classes V_1 and V_2. We know that each equivalence class is a **maximal subset** of elements of the set following the relation.
To prove: All the edges from V_1 to V_2 are in the same direction.
Assume, on the contrary, that they are not. Then we will have at least one edge from vertices in V_1 to V_2 and also at least one edge from vertices in V_2 to V_1. Let these edges be from $v' \to w'$ and $w'' \to v''$, such that $v', v'' \in V_1$ and $w', w'' \in V_2$.
Claim: A path exists from every vertex $v \in V_1$ to every other vertex $w \in V_2$. Conversely, a path also exists from every vertex $w \in V_2$ to every other vertex $v \in V_1$.
Consider any vertex $v \in V_1$ and $w \in V_2$. By our assumption, there exists at least one edge from a vertex $v' \in V_1 \to w' \in V_2$. Since V_1 and V_2 are strongly connected components, we have a path from $v \to v'$ in V_1. Then another path from $v' \to w'$. And also a path from $w' \to w$ in V_2. This gives us a path from $v \in V_1 \to w \in V_2$.
The converse path can be shown similarly.
Hence there exists a path from $v \to w$ and another path from $w \to v$ in the graph, and this is true $\forall v \in V_1$ and $\forall w \in V_2$.
Using the above claim, we can say that there is a path from every vertex in V_1 to every other vertex in V_2, and also from every vertex in V_2 to every vertex in V_1, i.e., all vertices in $V_1 \cup V_2$ are strongly connected. Therefore the set $V_1 \cup V_2$ is another equivalence class of the relation R. But we know that equivalence classes are maximal subsets. So a union of two equivalence classes cannot be another equivalence class of the same relation. Hence we arrive at a contradiction on our assumption.
Therefore, all edges from V_1 to V_2 are in the same direction if V_1 and V_2 are strongly connected components.

Question 4

A *directed tree* is defined as a graph that is connected and every vertex of the graph has indegree exactly one, except one which has indegree zero.

To prove:
(i) For a directed tree $|E| = |V| - 1$.
(ii)In a directed tree, for any two vertices u and v, $u \neq v$, if there is a path from u to v then there is no path from v to u.

(i) For a directed tree $|E| = |V| - 1$.
Proof:

If the graph has just one vertex, then we know there is no edge, i.e. $|V| = 1$, then $|E| = 0$. So the equation is satisfied, $|E| = |V| - 1$.
Now, consider the case when the graph has more than one vertex. Since the graph is connected, every vertex must have atleast one directed edge, either outgoing or incoming. Otherwise it cannot be connected to the graph. We know that every vertex, other than one (the *root*) has indegree equal to 1, and the root has indegree 0.
Since the **total number of edges in a graph is equal to the total number of incoming edges, which is equal to the total number of outgoing edges.** So, for a directed tree, the total number of incoming edges can be counted as, every vertex other than root has 1 incoming edge, and the root has 0 incoming edge. So the total number of edges $|E| = |V| - 1$.
(ii)In a directed tree, for any two vertices u and v, $u \neq v$, if there is a path from u to v then there is no path from v to u.
Proof: We know that a directed tree is connected, and also it has $|E| = |V| - 1$. So, if we ignore the edge directions, it must be an undirected tree, or simply a tree. We also know that a tree satisfies the property that a unique path exists between its any two distinct vertices. Now, in the undirected version of our directed tree, consider the path $P(u, v)$ as $u = u_0, u_1, u_2, ..., u_k = v$ and $u \neq v$. By the definition of tree, no other path can exist between u and v along undirected edges.
Now, if these edges are directed, either no path will exist between u and v, or a path will exist either from u to v, or from v to u. This is because if all the edges are in the direction $(u_i \rightarrow u_{i+1}), \forall i \in \{0, 1, 2, ..., k-1\}$, then the path will exist from $u \rightarrow v$, and this path P cannot exist from $v \rightarrow u$. Also, no other path exists between u and v.
Similarly, if all the edges are in the direction $(u_{i+1} \rightarrow u_i), \forall i \in \{k-1, k-2, k-3, ..., 1, 0\}$, then the path will exist from $v \rightarrow u$, and this path P cannot exist from $u \rightarrow v$. Also, no other path exists between u and v.
Otherwise, if all the edges do not point in the same direction, no directed path will exist from $u \rightarrow v$ or from $v \rightarrow u$.
Hence, since a single undirected path exists between any two distinct vertices, so if a directed path exists from $u \rightarrow v$, then in the directed tree, a directed path from $v \rightarrow u$ cannot exist.

Question 5

T_n is the number of distinct drawings of binary trees of n vertices.
We know that if we have just one vertex, then only one drawing is possible, i.e. $T_1 = 1$.
Now, if we have two vertices, then the other vertex can either be the left child or the right child. So two drawings are possible, i.e. $T_2 = 2$.
Now, for three vertices, using the counting techniques, we take one vertex as the *root*, and then we can either put 0 vertex on the left side and 2 vertices on the right side, or 1 vertex on both the sides, or 2 vertices on the left side and 0 vertex on the right. Now, the 2 vertices combination itself can be drawn in T_2 ways. Also, 0 child means no child on that side. This can be done in 1 way. So
$T_3 = T_0 * T_2 + T_1 * T_1 + T_2 * T_0 = 1 * 2 + 1 * 1 + 2 * 1 = 5$.
Similarly, $T_4 = T_0 * T_3 + T_1 * T_2 + T_2 * T_1 + T_3 * T_0 = 1 * 5 + 1 * 2 + 2 * 1 + 5 * 1 = 14$.
Continuing this in a similar way, we can observe that the total number of possible drawings is the sum of products of all possible combinations for the left and right sides. So if the left side has j nodes, the right side must have $n - 1 - j$ nodes, and this configuration can be drawn in $T_j * T_{n-1-j}$ ways, for each $j \in \{0, 1, 2, ..., n-1\}$. So we find the recurrence relation for T_n, $(n \geq 1)$ as:

$$T_n = \sum_{j=0}^{n-1} T_j * T_{n-1-j}$$

(For the recurrence relation, we define $T_0 = 1$, and $n \geq 1$.)

Question 6

$$T_n = \sum_{j=0}^{n-1} T_j * T_{n-1-j}$$

To solve this T_n using Generating functions, we define the generating function

$$F(x) = \sum_{n=0}^{\infty} T_n x^n$$

$F(x) = 1 + x + 2x^2 + 5x^3 + 14x^4 + ...\infty$.
Now, $F(x) * F(x) = (\sum_{n=0}^{\infty} T_n x^n) * (\sum_{n=0}^{\infty} T_n x^n)$
$F(x) * F(x) = F^2(x) = 1 + 2x + 5x^2 + 14x^3 + ...\infty$
So, the coefficient of x^n in $F(x)$ is equal to coefficient of x^{n-1} in $F^2(x)$, for $n \geq 1$.
Therefore, $T_n = coeff(x^{n-1})$ in $F^2(x)$, $n \geq 1$.
So if we multiply $F^2(x)$ by x and then add 1 to it, we will obtain the expression of $F(x)$.
$1 + x * F^2(x) = 1 + x + 2x^2 + 5x^3 + 14x^4 + ...\infty$.
i.e., $1 + x * F^2(x) = F(x)$ or $xF^2(x) - F(x) + 1 = 0$.
Using the $Sridharacharya\ formula$ to solve the quadratic equation, we get:

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

i.e., $2xF(x) = 1 \pm \sqrt{1 - 4x}$. Now, for which sign to choose, put $x = 0$.
Since $F(0) = 1$, the $-$ sign would give us the desired result. So,

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

Now, the binomial expansion of
$(1 - 4x)^{1/2} = 1 + (-2x) + (-2x^2) + (-4x^3) + (-10x^4) + (-28x^5) + ...\infty$.
The general (n)th term of the binomial expansion is

$$g_n((1 - 4x)^k) = \frac{(k) * (k - 1) * (k - 2) * ....((n + 1)\_terms) * (-4x)^{n+1}}{(n + 1)!}$$

where $k = \frac{1}{2}$ and $n \geq 0$. So $g_0 = -2x$, $g_1 = -2x^2$ and so on. There is an additional first term 1.
Substituting $k = \frac{1}{2}$ and simplifying, we get

$$g_n((1 - 4x)^k) = \frac{(\frac{1}{2}) * (\frac{-1}{2}) * (\frac{-3}{2}) * .... * (\frac{-2n+1}{2}) * (-4x)^{n+1}}{(n + 1)!}$$

$$= \frac{(-1)^n * 1 * 3 * 5 * .... * (2n-1) * (-1)^{n+1}(4x)^{n+1}}{2^{n+1} * (n+1)!}$$
$$= \frac{(-1)^{2n+1} * 1 * 3 * 5 * .... * (2n-1) * (2)^{2n+2} * x^{n+1}}{2^{n+1} * (n+1)!}$$
$$= \frac{(-1) * 1 * 2 * 3 * 4 * 5 * .... * (2n-1) * 2n * (2)^{2n+2} * x^{n+1}}{2 * 4 * 6 * 8 * .... * 2n * 2^{n+1} * (n+1)!}$$
$$= \frac{(-1) * (2n)! * (2)^{2n+2} * x^{n+1}}{2^n * n! * 2^{n+1} * (n+1)!} = \frac{(-1) * 2 * x^{n+1} * \binom{2n}{n}}{(n+1)}$$

So,

$$(1-4x)^{1/2} = 1 + \sum_{n=0}^{\infty} \frac{(-1) * 2 * x^{n+1} * \binom{2n}{n}}{(n+1)} = 1 - \sum_{n=0}^{\infty} \frac{2 * x^{n+1} * \binom{2n}{n}}{(n+1)}$$

Substituting this in the expression of $F(x)$, we get:

$$F(x) = \frac{1-\sqrt{1-4x}}{2x} = \frac{1-(1-\sum_{n=0}^{\infty} \frac{2*x^{n+1}*\binom{2n}{n}}{(n+1)})}{2x} = \sum_{n=0}^{\infty} \frac{2 * x^{n+1} * \binom{2n}{n}}{(n+1)(2x)}$$

$$F(x) = \sum_{n=0}^{\infty} \frac{\binom{2n}{n}}{n+1} x^n$$

Therefore,

$$T_n = \frac{1}{n+1}\binom{2n}{n} = \frac{(2n)!}{(n!)^2(n+1)}$$

Question 7

Given the directed graph G with k strongly connected components, we need to prove that an undirected graph H, having k vertices, and formed as mentioned in the question, will be a tree. But this statement is not necessarily true always. Below is a counter example:

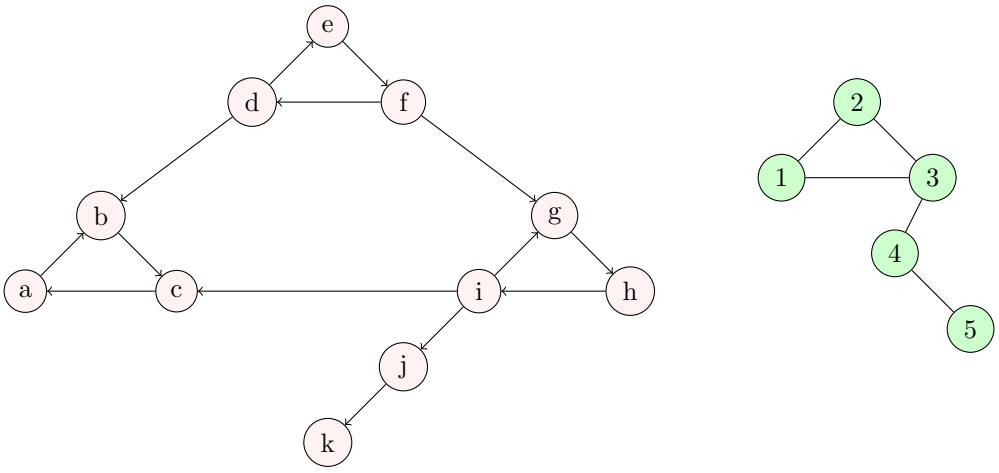

Figure 2: Left: Directed graph G, Right: Undirected graph H

In the above diagram, the directed graph G has 11 vertices, 14 edges and $k = 5$ strongly connected components, $V_1 = \{a, b, c\}, V_2 = \{d, e, f\}, V_3 = \{g, h, i\}, V_4 = \{j\}, V_5 = \{k\}$. These components are connected by edges $\{(d \to b), (f \to g), (i \to c)\}$ ensuring that V_1, V_2 and V_3 remain strongly connected components (if the edge $(d \to b)$ had been $(b \to d)$, then $V_1 \cup V_2 \cup V_3$ would have become a single strongly connected component).
The undirected graph H has $k = 5$ vertices, and edges according to if there is an edge from a vertex in V_i to a vertex in V_j.
So, in G, there is an edge from vertices in $V_2 \to V_1, V_2 \to V_3, V_3 \to V_1, V_3 \to V_4, V_4 \to V_5$. Therefore, there are edges $E_H = \{(1, 2), (2, 3), (3, 1), (3, 4), (4, 5)\}$ We can clearly see that the graph thus formed H is **not** a tree, as it has 2 distinct paths between vertices 1 and 2 . Hence the statement is false.

Question 8

We need to prove that an undirected graph is a tree if and only if it is connected and does not have any cycle.
Proof of part 1: *If any undirected graph is connected and does not have a cycle, then it must be a tree.*
Consider a connected undirected graph G that does not have a cycle. Assume on the contrary that it is not a tree. A tree is defined as a connected undirected graph that has a unique path between every pair of vertices.
So, our graph G does not have a unique path between every pair of vertices. So there will be at least one pair, let (u, v) be those two vertices of G, such that at least two distinct paths P_1 and P_2 exist between u and v in G. Since the paths are distinct, there must be at least one vertex in P_1 that is not in P_2 or vice versa.
Now, if we travel from u to v along say P_1, and return back from v to u along P_2, then there must have been a cycle formation somewhere on these paths. This is because the path P_2 will either end at some ancestor of u or some child of u, or u itself. In the first case, a cycle will form between that ancestor and v. In the second case, the cycle will be formed between that child and v. In the last case, the cycle will be formed between u and v themselves.
Hence if there are two distinct paths between two vertices, then it must have a cycle. But we are given that the graph does not have any cycle. Therefore, our assumption, that it is not a tree, is incorrect.
Proof of part 2: *If an undirected graph is a tree, then it must be connected and can't have any cycle.*
A tree is connected by its definition. Now we need to prove that it can not have a cycle.
Consider an undirected connected graph G that is a tree, but on the contrary, also has at least one cycle C.
Consider any two vertices a and b of the graph that are a part of that cycle C. So we can say that a path P_1 exists from $a \to b$, and another different path P_2 exists from $b \to a$. Since the graph is undirected, the same path P_2 also leads from $a \to b$, and $P_1 \neq P_2$. Hence, we have two distinct paths between a and b. This is a contradiction to our assumption that the graph is a tree. Hence, G cannot have any cycle.
Therefore an undirected graph is a tree iff it is connected and does not have any cycle.

Question 9

We want to prove that for a directed graph, all vertices in a cycle of the graph lie in the same strongly connected component. Consider a directed graph G, and it has a cycle C of k vertices. Assume on the contrary, that the k vertices of the cycle lie in different strongly connected components. Let G have total p strongly connected components, and let the k vertices form part of q strongly connected components ($q \leq p$).
Claim: All the strongly connected components that are connected by the k vertices of the cycle, are part of a single strongly connected component.
If $q = 2$, i.e. the cycle's vertices lie in 2 strongly connected components, then using the statement of question 3, we can say that since the edges from vertices in V_1 to V_2 and then from V_2 to V_1 form a cycle, they can't be all in the same direction. We will have at least one edge from a vertex in $V_1 \to V_2$ and at least one edge from a vertex in $V_2 \to V_1$. Therefore they are part of the same strongly connected component, as we can reach any vertex to any other vertex, through either the component edges, or through the cycle path.
If $q > 2$ then also we can reach any vertex from any other vertex in the q strongly connected components.
Consider any vertex v of a strongly connected component V_1. From v, we can reach every vertex $u \in V_1$, and now through the cycle edge from V_1, we can reach to some other strongly connected component, say V_i. So we can reach to all vertices in V_i from v. Now continue the same process, move from V_i through the cycle edges to some other component V_j. So all vertices in V_j are reachable from $v \in V_1$ and $v' \in V_i$. Eventually, we will reach some vertex in V_1, since the edges form part of a cycle.
Therefore we can see that all vertices are reachable from every other vertex in the q strongly connected components. Hence, they are a single strongly connected component.

Question 10

We wish to prove that an undirected graph has a spanning tree iff it is connected.
Proof of part 1: *An undirected connected graph has a spanning tree.*
If the undirected connected graph has $|E| = |V| - 1$, then it must be a tree. Hence G itself is a spanning tree. Now consider an undirected connected graph that is not a tree. Using the statement of question 8, we can say that if an undirected connected graph does not have any cycle, then it must be a tree. Conversely, if an undirected connected graph is not a tree, it must contain one or more cycles.
Consider any cycle $C = (v_0, v_1, v_2, ..., v_k, v_0)$ in the graph. Now WLOG, remove any edge, let it be $v_0 \to v_1$ from the graph to obtain G'.
Claim: G' is connected.
This is because all the paths not passing through the edge $v_0 \to v_1$ remain unaffected. And for those passing through this edge, we can equivalently take the path $(v_0, v_k, v_{k-1}, ..., v_2, v_1)$ to reach from $v_0 \to v_1$, and then continue the rest of the path. Hence G' is connected, but has one edge less. Now note that C no longer remained a cycle.
Since we know that in an undirected tree, $|E| = |V| - 1$, we will continue removing an edge from each cycle, until we remove the appropriate number of edges from G to obtain a connected graph, that satisfies the above relation. The final reduced graph is the spanning tree.
Proof of part 2: *An undirected graph that has a spanning tree is connected.*
Consider an undirected graph that has a spanning tree. A spanning tree of a graph $G = (V, E)$ is a subgraph of G that is a tree. So, by definition, a spanning tree is connected. Since the spanning

tree $G' = (V', E')$ is a subgraph of the original graph G, such that $V' = V$ but $E' \subseteq E$. So in the subgraph G' itself, all the vertices V of G are connected, since G' is a tree. Now, if we add more edges in the subgraph G' to make it G, it will still remain connected, since all these edges are also present in G. Therefore G is connected if it has a spanning tree.

Question 11

We wish to prove that minimum weight subgraph is a minimum spanning tree of the graph.
We know that the **minimum weight spanning tree (MST)** of a weighted graph is a spanning tree whose weight is the least amongst all spanning trees of the graph.
Also, since the weight of a subgraph is equal to the sum of weights of edges in the subgraph, the **minimum weight subgraph (MWS)** would be the subgraph whose sum of the weights of the edges is minimum, and all the vertices are connected.
Claim: A MWS must be a spanning tree of the graph G.
A subgraph of graph $G = (V, E)$ is a graph $G' = (V, E')$ with $E' \subseteq E$, i.e. $|E'| \leq |E|$. Now, since we want to retain connectedness, the minimum number of edges in the subgraph G' will be $|E'| \geq |V| - 1$. This is because every vertex of the graph will have at least one edge from some previous vertex of the graph, except the root vertex. Now, since the subgraph is of minimum weight among all subgraphs, it must have the minimum number of edges, i.e. $|E'| = |V| - 1$. This satisfies the conditions for a tree, hence G' is a tree, therefore the subgraph G' is a spanning tree of G.
Now, we will prove that the MWS must be the MST.
Assume on the contrary, that it is not the MST. But we have proved that it must be a spanning tree. Now consider any spanning tree S of G, that is not the MST. Then by definition of MST, $weight(S) > weight(MST)$, as it has the minimum weight among all spanning trees. Since MWS is also a spanning tree, therefore, $weight(MWS) > weight(MST)$. But, by definition, the MWS has the minimum weight among all subgraphs, and a MST is also a subgraph. Hence we arrive at a contradiction. Therefore our assumption was incorrect. So if the minimum weight subgraph is a spanning tree, then it must be the minimum weight spanning tree of the graph.

Question 12

We wish to prove that the spanning tree constructed by the given algorithm is a minimum spanning tree (MST).
After i iterations of the loop, we know that U_i is the set of vertices present in the spanning tree T we are constructing. Let each edge be represented by e, and define the set containing the edges present in our spanning tree T as W_i. So the tree formed $T = (U, W, w')$.
Before entering the loop, $U_0 = \{v\} \Rightarrow |U_0| = 1$ and $W_0 = \phi \Rightarrow |W_0| = 0$.
Now, after the first iteration, $U_1 = \{v, v'\} \Rightarrow |U_1| = 2$ and let $e_1 = edge(v, v')$. So $W_1 = \{e_1\} \Rightarrow |W_1| = 1$.
Similarly, after i iterations, $U_i = \{v, v', v'', ..., v^i\} \Rightarrow |U_i| = (i + 1)$ and $W_i = \{e_1, e_2, ...e_i\} \Rightarrow |W_i| = i$, as in each iteration, we are adding exactly one vertex and one edge to our spanning tree T.
We are given that after each iteration, T is a spanning tree. We wish to prove that it must be the MST. We claim that every W_i belongs to the set of edges E' of some MST $T' = (V', E', w')$ the graph $G = (V, E, w)$.

***Claim**: Every $W_i \subseteq E'$, where $T = (V', E', w')$ is any minimum spanning tree of G, for $i \in \{0, 1, 2, ...n-1\}$.*
Proof: We will give a proof by induction on i.
Base case: $i = 0$. $W_0 = \phi \Rightarrow W_0 \subseteq T$, since empty set is a subset of every set.
Assume the claim holds true for i. So let T be the MST such that W_i is contained in its edges' set. Now, consider for $i+1$.

1. If $e_{i+1} \in E'$ of T, then $W_{i+1} \in E'$ of the same MST T. Hence claim holds true.

2. If not, then let $e_{i+1} = (u, v)$ such that $u \in U_i$ and $v \notin U_i$. Now, since $e_{i+1} \notin E'$ of T, therefore, there must exist a unique path $P(u, v) = (u = v_0, v_1, v_2, ..., v_n = v)$ in T. Also, there must be some edge $e_x = (u', v')$ in $P(u, v)$, such that $u' \in U_i$ and $v' \notin U_i$, since $u \in U_i$ and $v \notin U_i$. Now, we add the edge e_{i+1} and remove e_x from the graph to obtain another tree T'. The edge $e_x \notin \{e_1, e_2, ...e_i\}$, since it had not been picked up by our algorithm yet, which means its weight must have been greater than weights of e_1 to e_i.

   ***Claim:** T' is a minimum spanning tree of G.*
   First we will show that T' is a spanning tree. We can clearly see that number of edges in T' is still equal to the number of edges in T, which was $(n-1)$. Since T was a spanning tree, on adding the edge e_{i+1}, a cycle was formed, and it did not remain a tree. Now, we remove one edge e_x from the cycle. This will make it again a tree. This is because for all paths not passing through the edge e_x, the path will remain unaffected in the tree, and for those passing through it, we can take the path through the remaining part of cycle that had been formed. Hence connectedness is retained and T' is a spanning tree.
   Now, the algorithm is such that in the 3rd step, it chooses the minimum weight edge from any vertex in the subgraph, to any other vertex not included till now in the subgraph. Therefore, $weight(e_{i+1}) \leq weight(e_x)$, as it was chosen by the algorithm. Now, let's analyse the weight of T':

$$weight(T') = weight(T) + weight(e_{i+1}) - weight(e_x)$$
$$\Rightarrow weight(T') \leq weight(T)$$

   Therefore, T' is a minimum spanning tree.

Observe that $W_{i+1} = \{e_1, e_2, e_3, ...e_i, e_{i+1}\} \subseteq T'$, where T' is any MST of G. As this holds for all $0 \leq i \leq n-1$, therefore claim shown.

Now, consider the case for $i = n-1$, so $W_{n-1} = \{e_1, e_2, e_3, ...e_{n-1}\}$. Using the above claim, we know that $W_{n-1} \subseteq E_T$, for some MST T. Also, $|W_{n-1}| = n-1$ and edges in tree $|E_T| = n-1$. Therefore the sets are equal, i.e. $W_{n-1} = T$, or the subgraph H formed by the algorithm $H = (U_{n-1}, W_{n-1}, w)$ is a MST of the graph $G = (V, E)$.

Question 13

We wish to prove that the weight of a minimum spanning tree of the graph is at most the minimum length tour.
Let $G = (V, E)$ be the given undirected graph. Consider a tour T of the graph G whose length is l. Now, since it visits all the vertices, it will have some edges $E' \subseteq E$, such that the subgraph formed by $G' = (V, E')$ must be connected. Now, in the tour T, each edge $e \in E'$ will be traversed one or more times. So the length of the tour will be at least the sum of the weights of the edges of G', i.e.

$l \geq weight(G')$.
Since G' is a subgraph of G, its weight must be at least the weight of the minimum weight (connected) subgraph G_{min} , i.e. $weight(G') \geq weight(G_{min})$.
As we have proved in **question 11** that the minimum weight subgraph is the minimum spanning tree, so the weight of the minimum weight (connected) subgraph G_{min} is equal to the weight of the minimum spanning tree (MST). Therefore, $weight(G') \geq weight(G_{min}) = weight(MST)$.
Using the inequalities shown above, we can say that

$$l \geq weight(G') \geq weight(G_{min}) = weight(MST)$$

$$\Rightarrow weight(MST) \leq l$$

But since we had chosen any arbitrary tour T, and the inequality comes true for any tour T, it will be also true for the minimum length tour, since it is also a special type of tour. Therefore, the weight of the minimum spanning tree of the graph is atmost the length of the minimum length tour.

Question 14

We wish to prove that the minimum length tour is at most twice the weight of a minimum spanning tree.
Consider any minimum spanning tree (MST) of the graph $G = (V, E)$. As shown in **question 11**, since the MST is also the minimum weight subgraph of G, it connects all the vertices of G and has the minimum number of edges. So a traversal of the MST will give us a tour T of the graph G.

We state and prove some claims that will help us prove the question.
***Claim**: Every MST will have at least one vertex of degree= 1, given it has 2 or more vertices.*
Proof: Assume contrary. Since we have a tree, if any vertex has $degree = 0$, then it has no edge from the tree vertices. This means that vertex would not be connected to the tree, which is not possible.
Now assume all vertices have degree ≥ 2. We know that in a tree, $\sum_{v \in V} degree(v) = 2|E| = 2(|V| - 1)$. But if all vertices have degree ≥ 2, then $\sum_{v \in V} degree(v) \geq 2|V|$, which is a contradiction to our assumption that the graph was a tree. Hence every MST has atleast one vertex of degree=1.
***Claim**: If we remove any vertex v of degree=1 from any MST, then the resulting graph will also remain a spanning tree of the remaining vertices.*
Proof: Note that the graph is still connected, as any path including that edge must end at v as $(degree(v) = 1)$, but we have removed that vertex. And for any path not including that edge, the path remains unaffected. So the graph is still connected for the remaining vertices. Now, for $|V|$ vertices in the MST, $|E| = |V| - 1$. In the resulting subgraph, we have removed exactly one vertex and one edge. So $|V'| = |V| - 1$ and $|E'| = |E| - 1 = |V| - 2 = |V'| - 1$. Hence it is a tree.
Since it was a subgraph of G, therefore the resulting graph is still a spanning tree of G.
***Claim**: If we remove a vertex v of degree $= 1$ from a MST T of $G = (V, E)$, the resulting spanning tree T' is a MST of a graph of remaining vertices $G' = (V', E')$.*
Proof: Let the removed edge be $e = (u, v)$. Now consider some other arbitrary spanning tree T'' of G'. If we add the vertex v and edge e into T'', we must get a spanning tree T_1 of G, because the graph will remain connected, and $|E| = |V| - 1$.
Since T was a MST of G, therefore $weight(T_1) \geq weight(T)$. Now, $weight(T_1) = weight(T'') + weight(e)$. Also, $weight(T) = weight(T') + weight(e)$. Therefore, $weight(T'') + weight(e) \geq weight(T') + weight(e) \Rightarrow weight(T'') \geq weight(T')$. This is true for any spanning tree T'' of G', therefore $T'$is a MST of G'.

Now, consider any MST T of G.
Claim: *Then there always exists a tour τ of length l of G, such that $l \leq 2 * weight(T)$, where T is a MST of G.*
Proof: We will prove this by induction on the number of edges $|E|$ in G.
Base case: If $|E| = 1$, then in the tour, we can go from the first vertex a to the other b, and then come back via the same edge. The tour will be seen as $\tau = (a \rightarrow b \rightarrow a)$. and its length l will be $l = 2 * weight(e) = 2 * weight(T)$ for those two vertices, since T will have only that edge e in it.
Now assume the claim holds true for a graph G having $m - 1$ edges. We will prove it for m edges.
Consider $G = (V, E), |E| = m, m \geq 1$ and a MST T of G. Now, we know from the above claims that there will exist a vertex v of degree= 1, and if we remove it from T, we will still get a MST T' of $G' = (V', E'), |E'| = m - 1$. By induction hypothesis, a tour τ exists of G', such that $l_\tau \leq 2 * weight(T')$. Since G' had the parent u of v in it, let u be at some arbitrary position x in the tour τ. So $\tau = (v_1, v_2, ..., v_x = u, ..., v_m, v_1)$. Now, consider adding that vertex v and its edge e to the graph to obtain G. For this we can create a new tour τ' in G, as $\tau' = (v_1, v_2, ..., u, v, u, ...v_m, v_1)$.
Let's analyse the length of this tour:
$l_{\tau'} = l_\tau + weight(e) + weight(e)$, for the two extra edges $(u \rightarrow v)$ and $(v \rightarrow u)$ in τ'.
Therefore, $l_{\tau'} = l_\tau + 2 * weight(e) \Rightarrow l_{\tau'} \leq 2 * weight(T') + 2 * weight(e)$. Since $T = T' \cup \{e\}$, therefore, $weight(T) = weight(T') + weight(e)$. Using this in the inequality, we get $l_{\tau'} \leq 2 * weight(T)$.
Since T was a MST of G, the claim is proved.
Now, let the length of the minimum length tour τ_{min} be l_{min}. Since by definition, it has the minimum length among all possible tours of G, $l_{min} \leq l_{\tau'}$. Therefore, combining the two inequalities:

$$l_{min} \leq l_{\tau'} \leq 2 * weight(MST)$$

$$\Rightarrow l_{min} \leq 2 * weight(MST)$$

Therefore, the minimum length tour has length at most twice the weight of the minimum spanning tree.