**22(a)**

ThinMST: Given a weighted undirected graph G and integer d, is there a minimum spanning tree T of G such that the degree of every node in T is at most d.

The **UHEMPATH** The Hamiltonian Path Problem asks whether there is a route in a directed graph from a beginning node to an ending node, visiting each node exactly once

Aim: **UHEMPATH** <= **ThinMST and ThinMST have polynomial time Verification**

**A: ThinMST has polynomial time verification-**

Proof - Given Spanning Tree **T( U, E )**

def MODIFIED\_KRUSKAL(G, d):

// d is the utmost possible degree of a vertex.

A = ∅

For each vertex v ∈ G.V:

MAKE-SET(v)

For each edge (u, v) ∈ G.E ordered by increasing order by weight(u, v)

if FIND-SET(u) ≠ FIND-SET(v) and **degree ( u ) < d and degree( v ) < d**:

A = A ∪ {(u, v)}

UNION(u, v)

return A

// MAKE-SET FIND-SET UNION are the operation of disjoint set as taught in class

def **verification( (**G, d), Proof T **)**

// U where U set of vertex

1. H = MODIFIED\_KRUSKAL( G, d )
2. if T == H:
3. return True
4. else:
5. return False

**Complexity analysis for verification:**

The ModifiedKruskal algorithm finds the minimum spanning tree while maintaining the constraint that the degree of every vertex must be utmost d.

We are traversing through every vertex checking the degree of every node.

*hence the complexity of verification is* ***O( E log V ) which is polynomial time verification***

**Correctness:**

* *For any* ***YES*** *instance of (****G****, d) ,there must be a Proof T such that* ***verification((****G, d), Proof T****)*** *returns* ***TRUE****.*

An YES instance of ThinMST can be defined as there exists a Minimum spanning tree in G such that degree of every vertex is at most d.

AsProof T is defined as a spanning tree in G, there exists a spanning tree H which will be equal to the given spanning tree T, i.e. H will be a minimum spanning tree of at most degree d.

Hence there exists a P such that **verification( (**G, d), Proof T**)** will return **TRUE.**

* *For a* ***NO*** *instance of (G, d),* ***verification( (****G, d), Proof T****)*** *returns false for all proofs.*

A **NO** instance of (G,d) is defined as there does not exist a MST in G such that degree of every node is utmost d.

Since, there is no **minimum** ST which satisfies the condition, it's trivial to conclude that there will not be a ST which has all vertices with degree utmost d.

Hence, ***verification( (****G, d), Proof T****)*** will always return FALSE, for all proofs.

**22(b)**

B**-Prove the Problem is NP-Hard i.e HEMPATH <= ThinMST**

Consider the following algorithm.

def **reduce(G)**:

1. Graph G is reduced to H the same as it is.
2. Set d is the utmost degree of the spanning tree, Set d=2, and set the weight of each vertex to constant c.
3. Return the transformed graph as H and d.

**Complexity analysis Reduce:**

Line **(1)** in the reduction step takes **O(V+E)** time as you need to copy the entire adjacency matrix.

Line**(2)** new variable is created that is **O(1)**

Hence, the reduction takes ***O(V+E)*** time i.e. it's a poly-time reduction.

**Statement of correctness lemma for the reduction:**

*Given an undirected graph* ***G, UHAMPATH*** *returns* ***TRUE*** *for the original graph* ***G****(i.e. there is a Hamiltonian path in* ***G****), iff* **ThinMST** *returns* ***TRUE*** *for the reduced graph (****H, d)****(i.e. there is a Spanning Tree in* ***H*** *such that every node have degree utmost d****. [H,d=reduce(G)][ d=2 ]***

**Forward proof(=>):**

Say graph **G** has some Hamiltonian path **P.** The reduction step will set d=2. Let this reduced graph be **H i.e. H,d=reduce(G).**

When **H** is given as input to **ThinMST**,

As the number of nodes and edges are the same in H and G, if there exists a hamiltonian path P in G then there exists some hamiltonian path H’ in H. As H’ is a hamiltonian path, hence H’ is also a spanning tree and each node in H’ has a degree of utmost 2. Since all the weights in H are equal to some constant c, hence all the spanning trees are also minimum spanning trees**,**  Thus, **ThinMST** will also return **TRUE**.

**Backward proof(<=):**

If y is a “TRUE” instance of **ThinMST**, then x is a “TRUE” instance of ***UHAMPATH.***

***OR,*** *we can prove its contrapositive statement:*

If x is a “FALSE” instance of ***UHAMPATH***, then y will also be a “FALSE” instance of **ThinMST*.***

Say graph **G** does not have any Hamiltonian path**, i.e. G is a FALSE** instance of ***UHAMPATH.***

**Let's** Assume that ThinMST returns true. **This** implies there exists a spanning tree with utmost degree 2, that is a spanning line graph or hamiltonian path, that contradicts our assumption.

Thus we can clearly notice that in **H,** Since there does not exist a spanning tree of utmost degree 2 Hence, ***ThinMST*** will also return **FALSE**.

Hence, the above reduction algorithm is a poly-time reduction from **UHAMPATH** to **ThinMST.**

From above statements A and B we can conclude that **ThinMST** has a polynomial time verification (i.e. NP ) and an NP-hard problem( **UHEMPATH** ) can be reduced into ThinMST in polynomial time ( NP-Hard ) hence the given problem is **NP COMPLETE**.