HQ 22

To proove thin MST is NP complete we need to prove that thin MST is NP and NP Hard.

proof that ~~thin MST~~ thin MST is NP

with the help of verification algorithm we will proove that thin MST is NP.

proof P : MST of graph G(V, E)

input instance : a weighted undirected graph G(V, E) and an integer d.

verification algorithm : def verifyThinMST proof (Proof P, graph G , d) :

O(V) — for all vertices v in P :

O(V+E) — if (degree (v) > d)

return ~~False~~ False

return True.

running time : Proof P is a MST of the graph G(V,E), here we choose Kruskal's algorithm and hence complexity will be O(E log E) or O(E log V). The algorithm takes O (V (V + E)) as shown above. Thus total

complexity = O(E log V) + O(V(V+E))

Lemma 1

If G has a Yes instance then there must be a proof P, for which verifyThinMST gives true.

If G has a yes instance ie there exist a minimum spanning tree T, s.t degree of every node in T is atmost d, then according to our proof we have taken one such MST of G using Kruskal's algorithm. Then in the algorithm, if degree of such vertices in the MST is within / atmost d then only we are returning True else if any one vertex also exceeds degree d we are returning False. Thus if G has a yes instance then one such MST must exist as proof P whose degree constraint we are checking in the verification algorithm

Lemma 2

If G has a NO instance, then for every P verify-
-thin MST Proof will give False

If G has a NO instance, ie G do not have a
MST s.t degree of all vertices in MST is atmost d
then our proof will generate a MST ~~but~~ i of
the graph G but our verification algorithm will give
False as degree of that MST (any MST which
P takes are proof from kruskal's algorithm) for
atleast 1 vertex will exceed d. Thus for
every P verify thin MST Proof will give false.

proof that thin MST is ~~AF~~ NP Hard.

To proof that thin MST is NP Hard we will reduce a known NP Hard problem. to ~~A~~ thin MST.

we can reduce Steiner Tree to thin MST but Steiner Tree is not a decision problem.

we can convert Steiner Tree to a decision problem with the help of following StreinerTreeExist ( ) which simply returns True if there exist a minimum weight subtree which contains every marked vertices else False.

def SteinerTreeExist (graph G, markedvertices V') :
~~if (~~
r = SteinerTree (G, V')
~~if (r has v' vertices and r is MST of~~
if (~~all vertices in r are same as v' and~~ |V'| = |V|
and r is MST of G)

return True

return False.

now SteinerTreeExist ( ) will be NP Hard as
we are running SteinerTree ( ) inside the function.
So now we can reduce SteinerTreeExist ( ) to
then MST ( ).

<u>reduction algorithm</u> :

def reduce (G(V,E), marked vertices V') :

G' = new empty graph.
for all vertices v in G : — O(V)
add v' in G'
for all edges E in G : — O(E)
add E' (with weights) in G
if (|V'| = |V|)
d = |V| - 1
else d = 0.
return (G', d).

explanation

given a graph $G(V,E)$ with marked vertices V' we are forming a new graph $G'(V',E')$ which is a copy of G and setting $d = |V|-1$ when number of vertices are equal to ~~the~~ number of marked vertices ~~and~~ else $d=0$.

running time

As shown above running time of above algorithm is ____ $O(V+E)$ rest all are constant time; polynomial time reduction.

Lemma

we will get yes instance of Steiner Tree Exist ~~only~~ iff we get yes instance of ~~thinMST~~ thinMST.

If we get a yes instance of Steiner Tree Exist ie there exist a minimum weight subtree of G that contains V' marked vertices then in the reduce algorithm we are ~~setting~~ copying the same graph G' and now if all vertices are marked then only we are setting d as $|V|-1$ else 0. This is because in any MST of a graph maximum edges that we need is $|V|-1$, by setting $d=|V|-1$ we can ensure that whenever all vertices are marked in Steiner Tree ie it basically gives the MST of the graph then our thin MST also gives ~~a~~ the MST of the same graph and returns a yes instance as $d \geq |V|-1$ will ~~automati~~ trivially be true for any vertices. If maximum edges are |V|-1 then degree of each vertex will never exceed |V|-1 i.e. in a MST the edges will never exceed |V|-1 and degree of a vertex means the number of edges incident on ~~the~~ any edge. (If multi edges or self loops are considered then setting $d = |V|+1$ will return a yes instance as degree of a vertex can never exceed number of vertices +1.)

If we get a NO instance of Steiner Tree Exist ie all vertices of Steiner Tree are not marked ~~or~~ or MST so formed is not a MST of graph G then in the reduction algorithm d will be set to 0. ie if we ~~get no~~ dont have all vertices marked in Steiner Tree then it might not return a MST of the entire graph and thus setting $d=0$ will ensure that in the reduced graph as well even though we will get a MST but degree of each vertex being atmost 0 will always cause thin MST as well to return a NO instance.