"REGULAR" SPANNING TREE

THINH D. NGUYEN

ABSTRACT. Given an undirected connected graph $G(V, E)$, constructing a spanning tree is a well studied problem with polynomial time algorithms. If we restrict the spanning tree to be "regular" as defined below, it turns out that this becomes a very hard problem.

1. BACKGROUND AND NOTATION

A spanning tree $T(V, E_T)$ of an undirected connected graph $G(V, E_G)$ is a connected subgraph of G with the same vertex set V and (maybe) different edge set E_T , such that T contains no cycle. The problem of constructing a spanning tree for a graph can be solved by depth-first search or breadth-first search. These are two frequently used techniques in graph algorithms. When the edges of the graph are weighted, we still can utilize Prim, Kruskal or Boruvka algorithms to build a minimum-weight spanning tree. The original unweighted version can be seen as special case of weighted version where all the edge weights are 1. In this paper, we concentrate on unweighted undirected graphs only.

In order to have more beautiful problems in our limited collection of **NP**-complete ones, we add just one requirement to the spanning tree problem. But, we first need some minor concepts. It is an elementary result from the very beginning of graph theory, back to Euler, that every tree must have at least one vertex of degree 1. These are also called *hanging vertex*. All the other vertices of a tree are called *inner vertices*. It is worth noting that since our tree is not rooted, hanging vertex is not leaf. A tree is a 4-"regular" one if apart from its hanging vertices, every inner vertex has degree exactly 4. We want to add the requirement that a spanning tree solution needs to be 4-"regular".

Formally, our computational problem can be defined as follows.

Definition 1. ("REGULAR" SPANNING TREE problem):

Input: an undirected graph $G(V, E_G)$

Output: YES if G has a 4-"regular" spanning tree, otherwise NO.

As mentioned above, we want to have more problems in the still very limited collection of **NP**-complete ones. So, it is our duty to prove our problem's hardness.

Claim 2. *We have that $X3C \leq_p$ "REGULAR" SPANNING TREE*

In this claim, X3C denote the EXACT COVER BY 3-SET which was proved to be **NP**-hard in [1]. In the following section, we describe our constructive reduction in a "bottom-up" manner. The correctness follows easily from the construction.

Key words and phrases. graph, regular, spanning, tree.
Perebor.

2. DESCRIBING THE CONSTRUCTION FROM THE BOTTOM UP

Given an instance of X3C, we will describe how to construct an undirected graph $G(V, E_G)$. Suppose the background universe set is $U = \{1, 2, \dots, n\}$, for each element $j \in U$, we create a vertex $j \in V$. For each 3-set s_i , we create a vertex s_i . Suppose $s_i = \{a, b, c\}$, we add three edges to E_G , namely (s_i, a) , (s_i, b) , (s_i, c) .

Forcing exact cover To force that each element j in U be covered by exactly one 3-set, we connect j to three dummy vertices.

Collect "up" all the 3-set vertices In order to be possibly spanned by a 4-"regular" spanning tree, we need to collect the 3-set vertices "up". We do it in a gradually incremental manner. Whenever there are two or more vertices left to be collected "up", we pick two arbitrary ones and connect the two to a new vertex just above them. Having done this, we have created a binary tree with a root. To be 4-regular, we connect the root to two dummy vertices, and each vertex, which is not a 3-set vertex, to a dummy vertex. We do not add dummy vertices to 3-set vertices because they already have 4 adjacent vertices.

3. PROOF OF CORRECTNESS

Although the correctness follows easily from the construction above, we include the proof here for completeness.

Proof. If the X3C is a YES instance, then we build the "regular" spanning tree from the root. First, take the root and two attached dummy vertices. Then span down the binary tree (actually, we also need to take the dummy vertex at each inner vertex) to the 3-set vertices. Having reached the 3-set vertices, at each set s_i , if s_i is included in the solution to the current X3C, we span down from s_i to the 3 elements of it and also the 9 attached dummy vertices. Note that all the unused 3-set vertices would serve as hanging vertices.

Conversely, if we have a 4-"regular" spanning tree for the constructed $G(V, E_G)$, we have a solution to the X3C instance. Note that each 3-set vertex can be either hanging or inner vertex. If it is inner, we include it to the exact cover solution. If it is hanging, we never include it to the solution. It is obvious to see that the so constructed solution is an exact cover. $\square$

4. CONCLUSION

Garey and Johnson [1] shapes their theory based on previous primal works of Cook, Levin and Karp. Johnson [2] moves on with the guide to this theory. As long as we study a mathematical conjecture, we should encourage ourselves to have moderate amount of labouring hours on popular maths books like these. Then, reading some articles on theory of computing like [3] is a good practice. Having all those done, we could think of the ultimate final for all mathematics sciences.

REFERENCES

1. Michael R. Garey, David S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*
2. David S. Johnson, *The NP-Completeness Column: An Ongoing Guide*. pp.393-405
3. Phan Dinh Dieu, Le Cong Thanh, Le Tuan Hoa, *Average Polynomial Time Complexity of Some NP-Complete Problems*, Theor. Comput. Sci. 46(3): pp.219-237 (1986)
Current address: Department of Mathematics, Moscow State University
Email address: kosmofarmer@yandex.com