1

Suppose you have a polynomial-time algorithm that, given a multigraph H, computes the number of spanning trees of H. Using this algorithm as a subroutine, design a polynomial-time algorithm that, given a weighted graph G, computes the number of minimum spanning trees of G.

Algorithm 1 Compute the number of minimun spanning tree of G

---

function NumberOfSpanningTreesOfMultigraph(V, E)
    some black magic...
end function
function NumberOfMST(V, E, w)
    $W \leftarrow [w(e_i) : e_i \in E]$
    sort W so that $i < j \iff w_i < w_j$
    $X \leftarrow \varnothing$
    $Answer \leftarrow 1$
    for $w_i \in W$ **do**
        $E_{w_i} \leftarrow \{e_i : w(e_i) = w_i\}$
        $V' \leftarrow$ connected components of $G' = (V, X)$
        $E' \leftarrow \{(A, B) : (x, y) \in E, x$ is in component A, y is in component $B\}$
        $Answer \leftarrow Answer \times$ NumberOfSpanningTreesOfMultigraph(V', E')
        $X \leftarrow X \cup E_{w_i}$
    end for
    return $Answer$
end function

The algorithm above can compute the number of MST of G in polynomial time if we can get the number of spanning trees of multigraphs in polynomial time.

- Effectiveness(polynomial-time):
  It is polynomial-time because the for-loop will be executed $O(|E|)$ times, and every statement in the loop can be done in polynomial time.
- Correctness:
  Consider the process of Kruskal's Algorithm to find an MST, [problem 1] shows that T_{w_i} will always have the same connected components. Thus, by adding edges of same weights into that graph, the resulting connected components will always be the same. This observation indicates that there will be multiple MSTs if and only if we can add E_{w_i} into the graph in different ways, and for every w_i, the influence on answer is independent.
  Thus at each time we add edges of same weights into the graph together, and find the number of ways these edges can connect the current graph. The number of MSTs is the product of numbers of ways for every w_i.