CS217 – Algorithm Design and Analysis

Homework 4

Not Strong Enough

April 4, 2020

1

Let T be a minimum spanning tree of G, and let $c \in \mathbb{R}$. Show that T_c and G_c have exactly the same connected components. (That is, two vertices $u, v \in V$ are connected in T_c if and only if they are connected in G_c). You are encouraged to draw pictures to illustrate your proof.

Solution. Since T_c is a subset of G_c, if $u, v \in V$ are connected in T_c, then u, v are connected in G_c.

Now let's assume $u, v \in V$ are not connected in T_c, but are connected in G_c. We call the edge (u, v) x, then $w(x) \leq c$. Also, the path in T from u to v contains an edge whose weight is greater than c. We call it y, then $w(y) > c$. If we add x into T, this forms a cycle which contains both x and y. After that, removing y from T still keeps the connectivity. We name $T' = T + x - y$. Now T' becomes a tree, because it has $|V| - 1$ edges and it's connected. The total weight of T' is $w(T) + w(x) - w(y) < w(T)$, which violates that T is an MST. So u, v are not connected in G_c if they are not connected in T_c.

In conclusion, two vertices $u, v \in V$ are connected in T_c if and only if they are connected in G_c. □

2

Suppose G is connected, and no two edges of G have the same weight. Show that G has exactly one minimum spanning tree

Proof. Suppose there are two different minimum spanning trees, named T and T'.

Now we remove an edge e in T, and the graph becomes two connected parts. If in T' the two parts are connected by e too, then remove another edge in T until the one in T' is different from the one in T.

There should be such a pair: otherwise, each edge in T is the same as that in T', which is a contradiction.

Now consider such pair. Without loss of generality, assume that $w(e) < w(e')$. Then if we replace e' in T' by e, the total weight of T' is less, so T' is not a minimum spanning tree. □

3

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.

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.

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)

$W \leftarrow \{w(e_i) : e_i \in E\}$

$X \leftarrow \varnothing$

$Answer \leftarrow 1$

for $w_i \in W$ in increasing order **do**

$E_{w_i} \leftarrow \{e_i : w(e_i) = w_i\}$

$V' \leftarrow$ connected components of $G' = (V, X)$

$E' \leftarrow \{(A, B) : (x, y) \in E, x \text{ is in component } A, y \text{ is in component } B\}$

$Answer \leftarrow Answer \times$ NumberOfSpanningTreesOfMultigraph(V', E')

$X \leftarrow X \cup E_{w_i}$

end for

return $Answer$

end function

- Effectiveness(polynomial-time):
  It is polynomial-time because the for-loop will be executed $O(n)$ 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.