Name: Lu Min
mlu821@bu.edu

Problem Set 3

1. Mathematical Calculations

I will prove it by two aspects, first, suppose $f\colon \mathbb{R} \to \mathbb{R}$ is continuous by the open set definition. I would like to prove that it is continuous by the $\epsilon - \delta$ definition. Fix $x \in \mathbb{R}$ and $\varepsilon > 0$. Then $(f(x) - \varepsilon, f(x) + \varepsilon)$ is an open set in $\mathbb{R}$. By continuity, $U = f^{-1}((f(x) - \varepsilon, f(x) + \varepsilon))$ is an open set in $\mathbb{R}$. It is easy to see that U contains x, and x is an interior point of U by openness of U. That is, there exists $\delta > 0$ such that $(x - \delta, x + \delta) \subseteq U = f^{-1}((f(x) - \varepsilon, f(x) + \varepsilon))$. Then it follows that $f((x - \delta, x + \delta)) \subseteq (f(x) - \varepsilon, f(x) + \varepsilon)$. So if f is open-set-continuous, then it is $\epsilon - \delta$-continuous.
Second, I suppose $f\colon \mathbb{R} \to \mathbb{R}$ is continuous by the $\epsilon - \delta$ definition. Then take an arbitrary open set $V \subseteq \mathbb{R}$, and we want to prove $f^{-1}(V)$ is open. This is true if $f^{-1}(V)$ is empty, so assume $x \in f^{-1}(V)$. Since $f(x) \in V$ and V is open, there exists some $\varepsilon > 0$ such that $(f(x) - \varepsilon, f(x) + \varepsilon) \subseteq V$. By continuity at x, there exists some $\delta > 0$ such that $(x - \delta, x + \delta) \subseteq f^{-1}(V)$. That is, x is an interior point of $f^{-1}(V)$. Since this is true for arbitrary $x \in f^{-1}(V)$, it follows that $f^{-1}(V)$ is open. So if f is $\epsilon - \delta$-continuous, then it is open-set-continuous.
Consequently, the definition of continuity of a map $f\colon \mathbb{R} \to \mathbb{R}$ as the inverse image of open sets being open is equivalent to the usual $\epsilon - \delta$ definition of continuity.

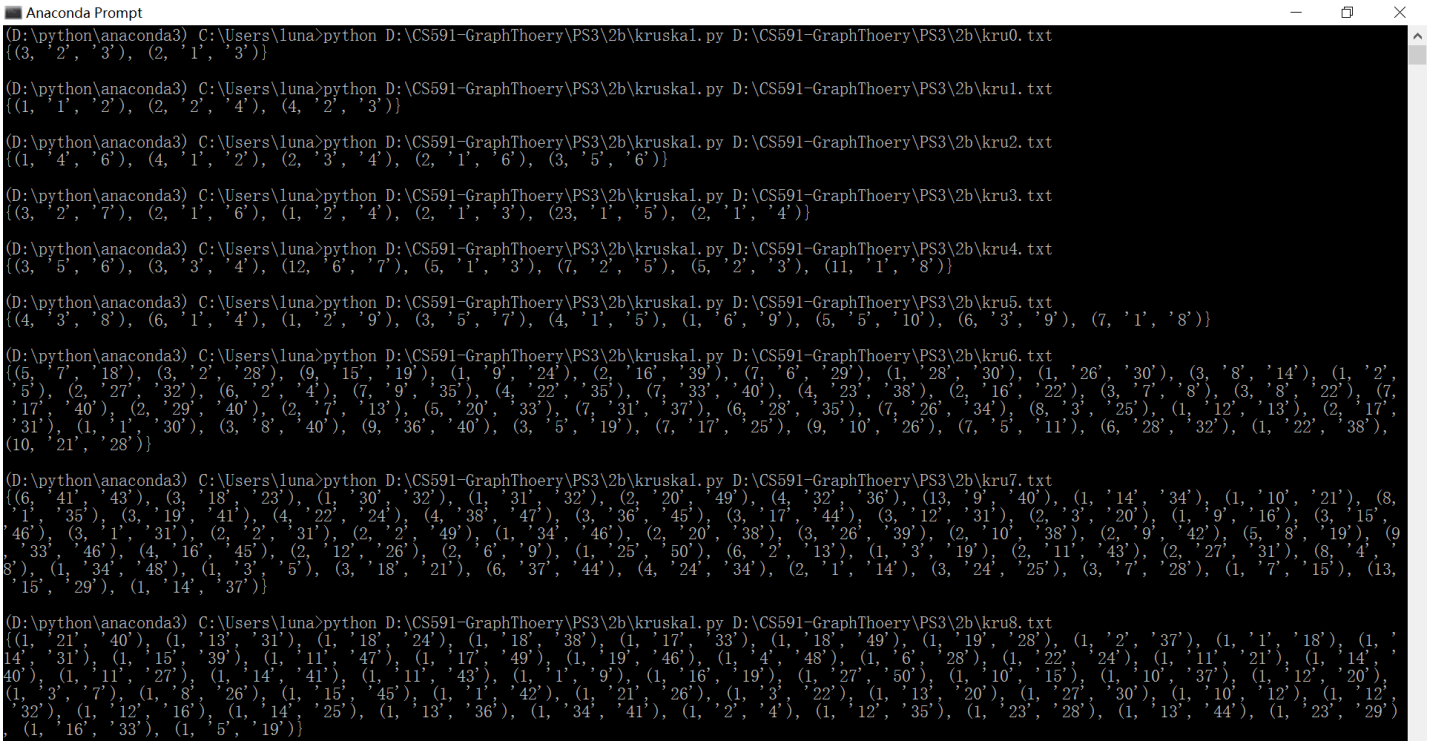
2. Programming

(a) For a point set with 4 nodes, I initialize a set in my code which is [1,2,3,4]. And I get the following outcome in the outcone.txt.

```
The total number of topologies: 355
[[], [2], [1, 2, 3, 4]]
[[], [3], [1, 2, 3, 4]]
[[], [1], [1, 2, 3, 4]]
[[], [4], [1, 2, 3, 4]]
[[], [1, 2], [1, 2, 3, 4]]
[[], [1, 3], [1, 2, 3, 4]]
[[], [1, 4], [1, 2, 3, 4]]
[[], [2, 3], [1, 2, 3, 4]]
[[], [3, 4], [1, 2, 3, 4]]
[[], [2, 4], [1, 2, 3, 4]]
[[], [1, 3, 4], [1, 2, 3, 4]]
[[], [2, 3, 4], [1, 2, 3, 4]]
[[], [1, 2, 4], [1, 2, 3, 4]]
[[], [1, 2, 3], [1, 2, 3, 4]]
[[], [2], [1, 2], [1, 2, 3, 4]]
[[], [2], [2, 3], [1, 2, 3, 4]]
[[], [2], [2, 4], [1, 2, 3, 4]]
[[], [2], [1, 3, 4], [1, 2, 3, 4]]
[[], [2], [2, 3, 4], [1, 2, 3, 4]]
[[], [2], [1, 2, 4], [1, 2, 3, 4]]
[[], [2], [1, 2, 3], [1, 2, 3, 4]]
[[], [3], [1, 3], [1, 2, 3, 4]]
[[], [3], [2, 3], [1, 2, 3, 4]]
[[], [3], [3, 4], [1, 2, 3, 4]]
[[], [3], [1, 3, 4], [1, 2, 3, 4]]
[[], [3], [2, 3, 4], [1, 2, 3, 4]]
[[], [3], [1, 2, 4], [1, 2, 3, 4]]
[[], [3], [1, 2, 3], [1, 2, 3, 4]]
[[], [1], [1, 2], [1, 2, 3, 4]]
[[], [1], [1, 3], [1, 2, 3, 4]]
```

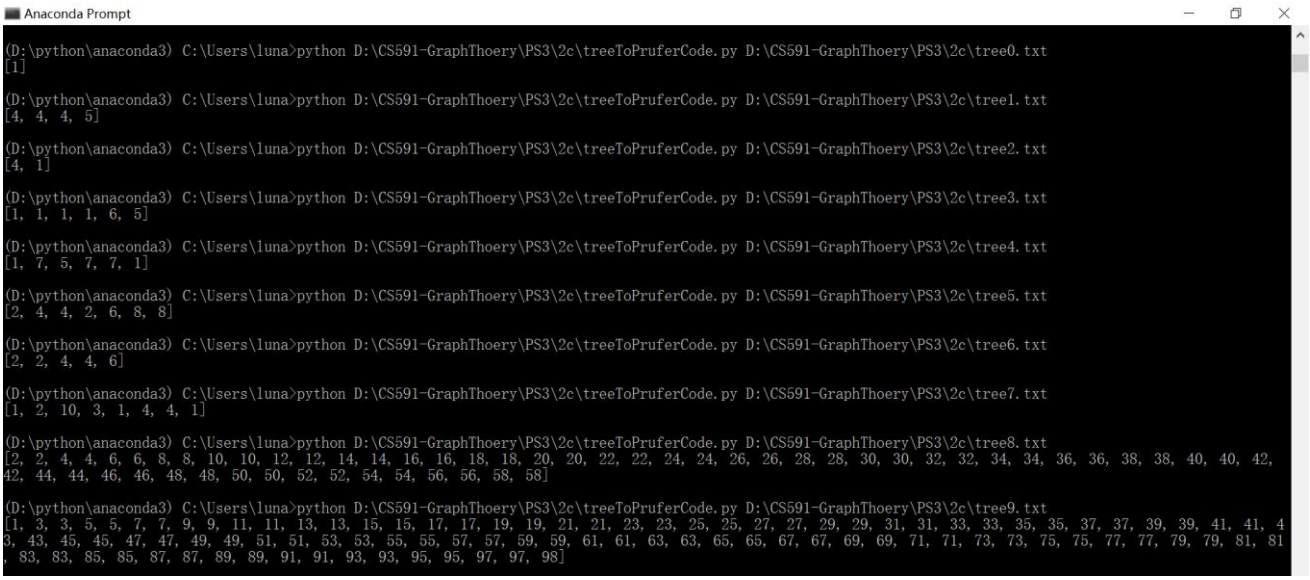
My basic idea is get all combinations of the subsets of X, and delete combinations without empty set and X. Then we need the union and intersection of any two subsets in the combination, so I delete combinations which do not satisfy this condition. Finally, I get the all the topologies of point set with 4 nodes.

(b) I use Kruskal's algorithm to get minimum spanning tree in the graph. Here is the result of 10 graph example up to 100 nodes.

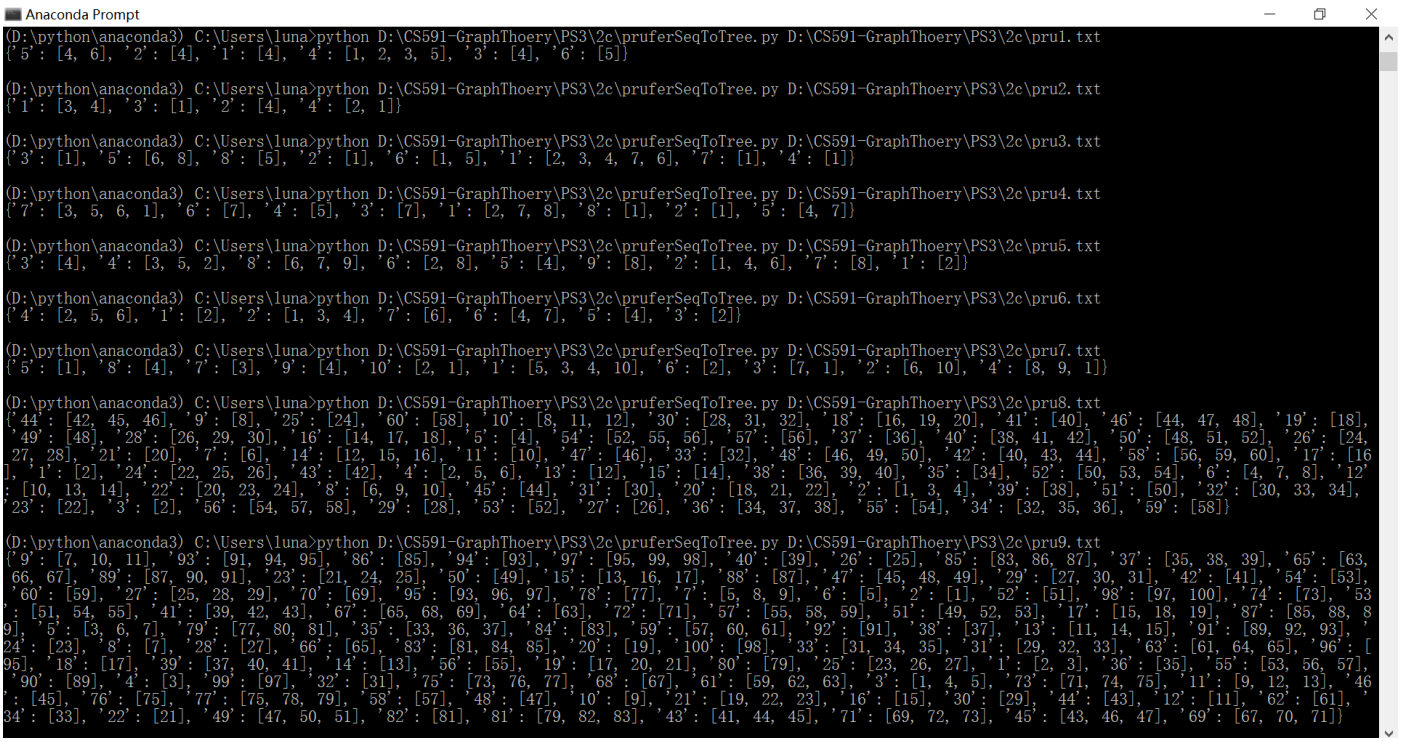


In every spanning tree, for example, we have (5, '1', '2') which means node 1 to node 2 has edge with weight 5. So the above image shows the minimum spanning tree in each graph.

(c) For generating prufer code with a given tree, in my tree, for example, 1,2 means there is an edge between node 1 and node 2. I would make sure that these 10 example are all trees without cycle. My basic idea is for all the nodes, I will first remove the leaf node with least label and cut the connection between this node and its neighbor, and then add the neighbor to the prufer code. Keep doing that until the length of prufer code is number of nodes minus 2.



To construct tree by given prufer sequence, first, I get the number of nodes



is the length of prufer sequence plus 2. For all the nodes, I initialize the degree to 1. Then for each appearance of node in the prufer sequence, I add 1 to the degree of corresponding node. Second, for each node from node 1 with degree of 1, I connect it with node in the prufer sequence. After connecting them, I both decrease 1 on the degree of leaf node and connected node. Finally, I get the tree. I represent it with python dictionary.