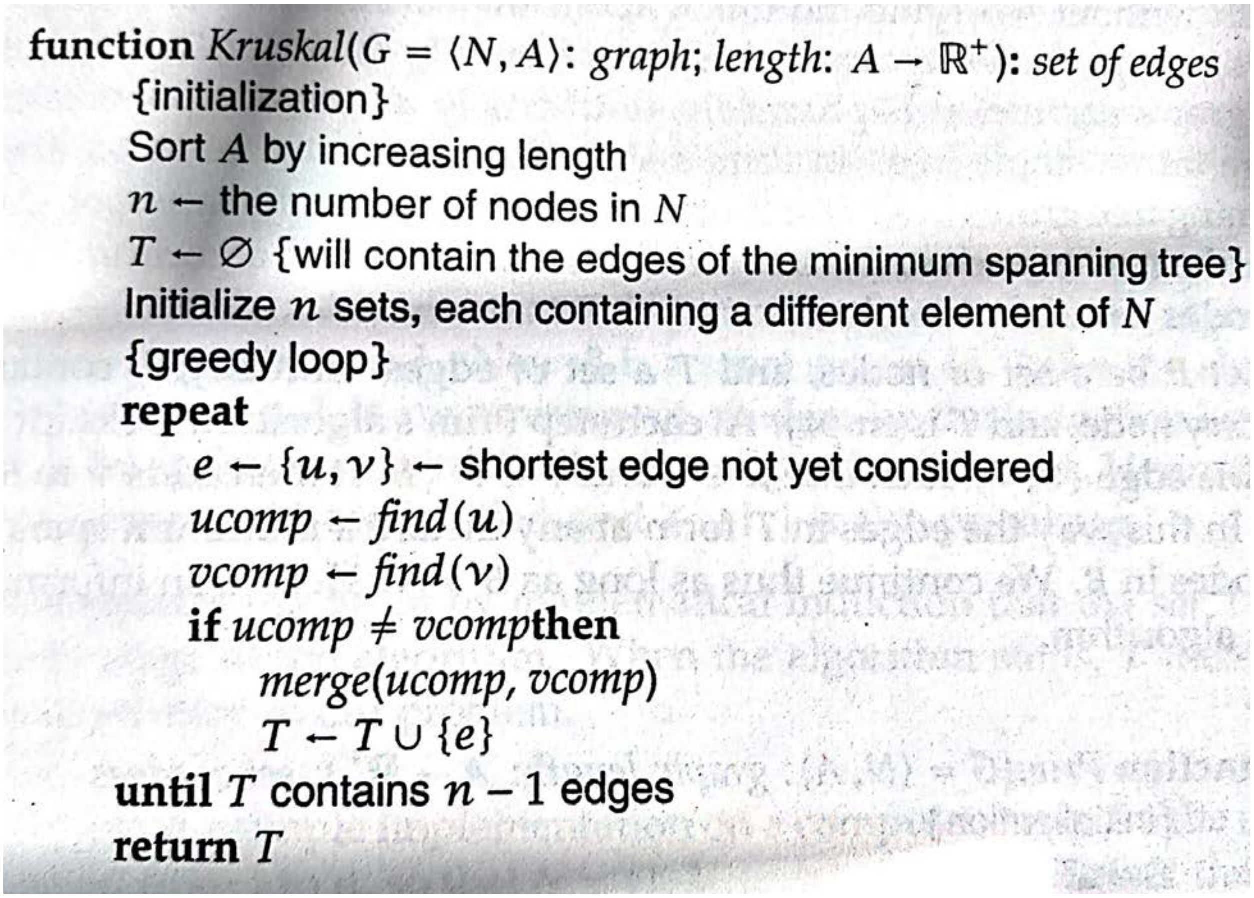
Homework 9

Data Structures II

Kruskal's method for finding a minimal spanning tree of a weighted undirected graph – Implementation

Here is the algorithm



```
function $Kruskal(G = \langle N, A \rangle$: graph; length: $A \rightarrow \mathbb{R}^+$): set of edges
    {initialization}
    Sort $A$ by increasing length
    $n \leftarrow$ the number of nodes in $N$
    $T \leftarrow \varnothing$ {will contain the edges of the minimum spanning tree}
    Initialize $n$ sets, each containing a different element of $N$
    {greedy loop}
    repeat
        $e \leftarrow \{u, v\} \leftarrow$ shortest edge not yet considered
        $ucomp \leftarrow find(u)$
        $vcomp \leftarrow find(v)$
        if $ucomp \neq vcomp$then
            $merge(ucomp, vcomp)$
            $T \leftarrow T \cup \{e\}$
    until $T$ contains $n - 1$ edges
    return $T$
```

Here is what you are given.

1. A single connected component graph : artist_edges.txt with 819,306 edges and 50,515 vertices.
2. A list of weights for each edge given in weights.txt . Each weight in the text file goes with the edge at the corresponding line number in artist_edges.txt
3. You will use Java's PriorityQueue and union-find algorithms to determine the minimum spanning tree.
4. Your code will return an ArrayList of edges comprising the minimum spanning tree and a double value equal to the sum of the weights of the edges on the minimum spanning tree.
5. For union- find, you will reuse the code we had for HW8 (the third algorithm) and a new algorithm using path-compression. Basically you will do this in two different ways. One is straightforward and the other requires implementation. See chapter 8.