Applications of Disjoint Set Data Structures

October 7, 2020

We describe here an application that can be solved easily using a disjoint-set data structure: the computation of a minimum-weight spanning tree of a weighted graph.

Weighted graphs

A **weighted graph** is a finite set of nodes connected by edges which have positive real numbers as weights. For example, the following is a weighted graph with 5 nodes and 6 edges: We will assume that

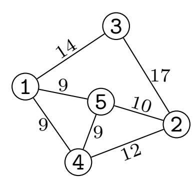


Figure 1: A weighed graph which is connected

- the nodes of a graph with n nodes are labeled with numbers from 1 to n.
- a text file which stores the representation of a weighted graph in the following way:
  - The first line contains the value of n (an integer)
  - The following lines contain 3 numbers separated by whitespace:
    $i \quad j \quad w$
    to indicate that the graph has an edge from node i to node j with weight w.

  We assume that the edges are enumerated in increasing order of weight. For example, the weighted graph from Fig. 1 can be stored and read from a text file with the following content:

```
5
1 4 9
1 5 9
4 5 9
2 5 10
2 4 12
2 3 17
```

Minimum weight spanning trees

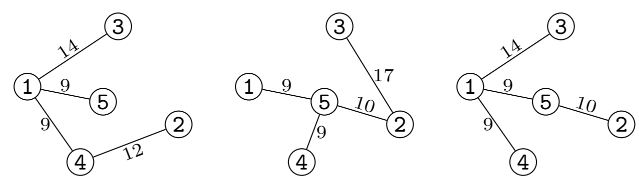
A graph is **connected** if there is a path between every two nodes in the graph. Fo example the weighted graph from Fig. 1 is connected.

A **spanning tree** of a weighted and connected graph G is a set T of edges of G such that

1. Every node of G is an endpoint of an edge in T
2. T has no loops.

The **weight** $w(T)$ of T is the sum of weights of edges in T.

For example, the following are spanning trees of the graph in Fig. 1:



$w(T_1) = 44$ $w(T_2) = 45$ $w(T_3) = 42$

A **minimum weight spanning tree** (or **MWST**) of G is a spanning tree of G whose weight has minimum possible value. For example, T_3 is a MWST of the graph from Fig. 1.

A MWST of a connected and weighted graph G with n nodes can be found with Kruskal algorithm:

Start with the initial partition $S = \{\{1\}, \{2\}, \ldots, \{n\}\}$, $T = \emptyset$ and $W = 0$
for each edge (i, j, w) of G, in increasing order of weights **do**
  if i, j are not in the same component of S
    add (i, j, w) to T
    Union(i, j)
    $W = W + w$
  end if
end for
return T, W

Homework 1

Implement a program that reads from a text file `graph.txt` the representation of a connected weighted graph G and computes a MWST of G. The program will print the weight and the list of edges of the MWST.

Suggestion: implement a disjoint set-data structure using the explanations from Lecture 2, and use it to implement Kruskal algorithm.

Homework 2

Indicate other possible applications of disjoint-set data structures.