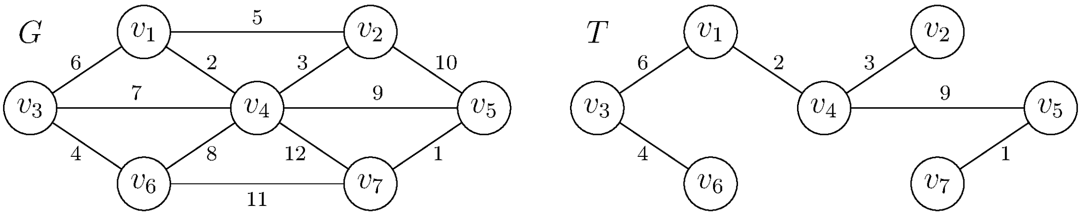
TND004 Data Structures
Lab 5 Part B
Minimum Spanning Trees

Overview

The main goal of the second part of this lab is to implement the algorithms by Prim and Kruskal presented during lecture 12 for the following graph problem:

> *Minimum Spanning Tree* (MST):
> Given a connected weighted undirected graph $G = (V, E)$,
> construct a spanning tree for G of minimum total weight.

Input and output for this problem is examplified in figure 1.
In particular, the output is essentially $n - 1$ tree edges.



Figur 1: A connected weighted undirected graph G and a mimimum spanning tree T for this graph (in this case, the tree is unique).

The following files can be copied from the course directory ~~`S:\TN\D\004\lab\lab5b`~~.

- `list.*` : classes for adjacency lists.
- `graph.*` : class for undirected graphs.
- `heap.h` : class for generic heaps.
- `edge.*` : class for "explicit" edges (needed for the heap).
- `dsets.*` : class for disjoint sets.
- `main.cpp` : meny-driven test program.
- ~~`lab5b.cbp` : project file for CodeBlocks~~.
- `graph1.txt` : data for the graph in figure 1.
- `graph2.txt` : data for the graph in figure 9.48 from the course book.

The testprogram should be familiar by now ...

Exercise

Copy the files to your computer. Then implement the following member functions from class Graph (that is, implement the algorithms presented during the lecture):

- `void Graph::mstPrim() const`
- `void Graph::mstKruskal() const`

Both member functions should print out the edges of a minimum spanning for the current graph along with the corresponding total weight.

Finally, implement *weighted union* and *find with path compression* by modifying the code for the following member functions from class DSets:

- `void DSets::join(int r, int s)`
- `int  DSets::find(int x)`

Currently only *simple union* and *simple find* are implemented by these functions.

Presenting your solution

Demonstrate your program to the lab assistent. No print-outs are required.