# 1 Assignment No. 8: Disjoint Sets

**Allocated time:** 2 hours

## 1.1 Implementation

You are required to implement **correctly** and **efficiently** the base operations for **disjoint set** (*chapter 21.1* from [1]) and the **Kruskal's algorithm** (searching for the minimum spanning tree, *chapter 23.2* from [1]) using disjoint sets.

You have to use a tree as the representation of a disjoint set. Each tree holds, besides the necessary information, also the *rank* field (i.e. the height of the tree).

The base operations on **disjoints sets** are:

- MAKE_SET (x)
  - creates a set with the element $x$
- UNION (x, y)
  - makes the union between the set that contains the element $x$ and the set that contains the element $y$
  - the heuristic *union by rank* takes into account the height of the two trees so as to make the union
  - the pseudo-code can be found in *chapter 21.3*[1]
- FIND_SET (x)
  - searches for the set that contains the element $x$
  - the heuristic *path compression* links all nodes that were found on the path to $x$ to the root node

## 1.2 Minimal requirements for grading

The lack of any of the minimum requirements (even partially) may result in a lower grade through penalties or refusal to accept the assignment resulting in a grade of 0.

- *Demo*: Prepare a demonstration of correctness for each algorithm implemented. The correctness of each algorithm is demonstrated through a simple example (maximum 10 values).
- The charts created must be easy to evaluate as in grouped and added through the Profiler functions as specified by the assignment requirements. The assignment will not be evaluated if it contains a plethora of ungrouped charts. For example, the comparative analysis implies the grouping of the compared algorithms.

- Interpret the chart and write your observations in the header (block comments) section at the beginning of your *main.cpp* file.
- We do not accept assignments without code indentation and with code not organized in functions (for example where the entire code is in the main function).
- *The points from the requirements correspond to a correct and complete solution, quality of interpretation from the block comment and* ***the correct answer to the questions from the teacher.***

## 1.3 Requirements

### 1.3.1 Correct implementation of MAKE_SET, UNION and FIND_SET (5p)

*Demo*: The correctness of the algorithm must be proved on a small-sized input.

- create (MAKE) 10 sets + show the contents of the sets
- execute the sequence UNION and FIND_SET for 5 elements + show the contents of the sets

### 1.3.2 Correct and efficient implementation for Kruskal's algorithm (2p)

*Demo*: The correctness of the algorithm must be proved on a small-sized input.

- create a graph of 5 nodes and 9 edges + **print the edges**
- apply Kruskal's algorithm + **print the chosen edges**

### 1.3.3 Evaluate the disjoint sets operations (MAKE, UNION, FIND) using Kruskal's algorithm (3p)

**!** Before you start to work on the algorithms evaluation code, make sure you have a correct algorithm!

Once you are sure your program works correctly:

- vary $n$ from 100 to 10000 with a step of 100
- for each n
  - build an **undirected**, **connected**, and **random** graph with random weights on edges (**n** nodes, **n*4** edges)
  - find the minimum spanning tree using Kruskal's algorithm
    - evaluate the computational effort of each individual base operation (MAKE, UNION, FIND– *resulting in a plot with 3 series*) on disjoint sets as the sum of the comparisons and assignments performed; thus, there should be **3 series in the plot**, one for each operation.

# References

[1] Thomas H. Cormen et al. *Introduction to Algorithms*. 2nd. The MIT Press, 2001. ISBN: 0262032937.