

# 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*<sup>1</sup>) and the **Kruskal's algorithm** (searching for the minimum spanning tree, *chapter 23.2(?)*) 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 **disjoint 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(?)*
- 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 Requirements

### 1.2.1 Correct implementation of MAKE\_SET, UNION and FIND\_SET (5p)

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

---

<sup>1</sup>Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein. *Introduction to Algorithms*

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

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**

### 1.2.3 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.