Assignment No. 8: Disjoint Sets

Allocated time: 2 hours

Implementation

You are required to implement **correctly** and **efficiently** the base operations for **disjoint set** (chapter 21.1 from book[1]) 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 **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 the chapter 21.3 from the book[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

Thresholds

Grade	Requirements
5	Correct implementation for: MAKE_SET, UNION and FIND_SET + demo
7	Correct and **efficient** implementation for: Kruskal's algorithm
9	Evaluate the disjoint sets operations using Kruskal's algorithm
10	Interpretations, discussion

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein. *Introduction to Algorithms*

Evaluation

! Before you start to work on the algorithms evaluation code, make sure you have a correct algorithm! The correctness of the algorithm must be proved on a small-sized input (i.e. create 10 sets and execute the sequence: UNION and FIND_SET for 5 elements and dump the content of the sets).

Once you are sure your program works correctly:

- vary n from 100 to 10000 with step 100;
- for each n
    - build a undirected, connected, 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 as the sum of the comparisons and assignments performed by each individual base operation on disjoint sets.