EECS 560 Lab 9: Disjoint Sets

Due date
11/1/2020 Sunday, 11:59 pm

Objective
Get familiar with disjoint sets ADT implementation using vector from lab1 with C++. Work on basic operations of disjoint sets.

Specification of the ADT
Refer to chapter 8 in Textbook and implement the code for methods in figure 8.6 of disjoint sets class skeleton. The implementation for some of these methods available as mentioned below:

1. Figure 8.7: Explicit constructor to initialize the disjoint sets.
2. Figure 8.9: Public member functions **int find(int x) const** to find the "name" of the set that containing the element x, note that you **don't** perform path compression here.
3. Figure 8.16: Public member functions **int find(int x)** to find the "name" of the set that containing the element x, while performing the find operation, you should perform path compression.

Requirements to completed from text from template

1. Change the disjoint sets class into "**myDisjointSet**".
2. Implement public member functions **void unionSets(int root1, int root2)** defined in figure 8.7, you should union two trees by **size**, discussed in section 8.4. **Don't** confuse with union by height as shown in figure 8.14. Breaking tie by attaching root2 to root1.
3. Implement public member functions **void print(int i)** that prints the path from element i to its roots. You **don't** perform path compression during printing;
4. Implement public member functions **int getNumberElements() const** that returns the maximum number of elements you can put in the disjoint sets.
5. Implement public member functions **int getSetSize(int i) const** that returns the size of the set that contains element i, note that size of set is a positive number.
6. Any other internal (private) functions you need to implement the previous functions.

Testing and Grading

We will test your implementation using a tester main function, on a number of instances that are randomly generated. We will release the tester main function, several instances (will be different from the grading instances but will have the same format), and expected output together with the lab instruction via Blackboard. You code will be compiled under Ubuntu 20.04 LTS using g++ version 9.3.0 (default) with C++11 standard.

The command line we are going to use for compiling your code is: "g++ -std=c++11 main.cpp" (note that main.cpp will try to include the .hpp file you submit, and your .hpp file needs to be property implemented to compile successfully).

Your final score will be the percentage your program passes the grading instances. Note that if your code does not compile (together with our tester main function), you will receive 0. Therefore, it is very important that you ensure your implementation can be successfully compiled before submission.

Submission and Deadline

Please submit your implementation as one .hpp file, with file name "myDisjointSet_[YourKUID].hpp". For example, if my KU ID is c123z456, my submission will be a single file named "myDisjointSet_c124z456.hpp". Submissions that do not comply with the naming specification will not be graded. Please submit through Blackboard.