Weijia Xu

Professor Arun

Algorithms & Structured Programming CS455

Oct.08, 2023

1. **(8 pts) Implement the disjoint-set-collection data structure from scratch as a class in Python.  Your implementation is not required to be the most efficient in time or in space complexity, though it should not be grossly inefficient in either. (Implementing the implementation described in class is just fine.)**

**A black screen with white text

Description automatically generated**

* 1. The constructor method should take one argument, n, and initialize the DSC data structure to contain singleton sets, {{0}, {1}, {2}, ..., {n-1}}.
  2. Support the method find(i) which returns the id of the set containing the element i.
  3. Support the method union(x, y) which assumes that x and y are in different sets and merges those two sets.
  4. Test this class to find the connected components of the graph on 7 nodes specified by the set of edges below: {{0, 1}, {2, 3}, {3, 5}, {4,6}}. Output the set of nodes in each connected component. (You don't need to output the edges in each connected component.)

1. **(8 pts) Use your data structure to find the connected components in a given graph. A screenshot of a computer screen

   Description automatically generated**
   1. Implement using your DSC data structure an algorithm to find the connected components in a given graph. (In fact you can implement it in such a way that 1.4 reuses it for the basic testing.)
   2. Implement a method that generates graphs that are suited to testing your implementation in 2.1. It takes a list of positive integers [n1, n2, ..., nk] as an argument and generates a graph with k connected components containing n1, n2, ..., nk nodes respectively. Within each connected component, make its graph fully connected, i.e. connect every pair of nodes within a component by an edge.
   3. Use your generator in 2.2 to test your implementation in 2.1 on one graph generated from each of the configurations below: [10]\*10, [100,500], [250,5, 10]. In each case automate the functional testing of your implementation in 2.1 by comparing the sets of nodes in each connected component that the generator produces (it can return this information) with the sets of nodes in each connected component that your implementation of 2.1 finds.