CS6905 (AGA) Winter 2023 – Assignment 7 (Minor)
Due Tuesday March 28, 2023, by 5pm.

The `GraphWtAL.java` file (on Desire2Learn) provides the `GraphWtAL` class, used to build and store a static undirected graph with edge weights, as described for Assignment 1.

The `UnionFind.java` file (on Desire2Learn) provides a simple implementation of a union-find data structure class, using arrays and path compression.

In this assignment, you need to implement Karger's randomized algorithm to find a global minimum cut.

You need to write a `Karger` class that extends `GraphWtAL`, adding the following:

- an array of edges of the graph, and an integer with the number of edges added
- a constructor that calls the `GraphWtAL` constructor and also sets up the array of edges
- `addEdge(int, int)`: takes a pair of vertex indices at its parameter, and adds an edge between those vertices to the graph with weight 1 (calling `GraphWtAL.addEdge` as needed), and also adds the edge to the array of edges.
- `minCut(int` k`, int` *times*`)`: takes two integers as parameters, where k is the number of pieces that the graph is to be broken into, and *times* is the number of repetitions. You need to implement the Karger algorithm as discussed in class, and run it for *times* repetitions to find the minimum over all the repetitions. Each run should pick random edges and merge vertices until there are k vertices left.

  This method needs to use the provided `UnionFind` class to manage the vertex merges.

  As a source of random numbers, you should use the `Random` class in `java.util.Random`, to construct an object and use its `nextInt()` method. Calling `nextInt(m)` for a `Random` object will return a random integer between 0 and `m-1`.

  This method needs to print out the pieces of the graph that result from the minimum cut, with one line per piece, and each line listing the vertices of that piece. It also needs to set the `mark` value for each vertex to the number of its piece (where the pieces are numbered from 0).

  This method also needs to return an `int` that is the number of edges in the minimum cut.

Your class should also have other methods and classes as appropriate.

Ensure that your code works with the provided `DriverAGA7.java` code, which will be used to test your submitted solution. Organize and comment your code appropriately.

Submit on D2L: your `Karger.java` file, and the I/O from one test run of your solution that you choose to demonstrate your code. This test data should not be I/O provided by the instructor.

Note: since Karger's algorithm is randomized, the results of running the algorithm may sometimes differ.