Develop a UF ("union-find") client that takes an integer value n from the command line to determine the number of "sites." Then generates random pairs of integers between 0 and n-1, calling connected() to determine if they are connected then union() if not. Loop until all sites are connected then print the number of connections generated. Package your program as a static method count() that takes n as argument and returns the number of connections; and a main() that takes n from the command line, calls count() and prints the returned value.

To what extent can you confirm the hypothesis that the number of pairs generated to accomplish this (i.e. to reduce the number of components from n to 1) is ~ 1/2 n log n where log n is the natural logarithm of n.

You can use any UnionFind implementation you like but if you need one, I have uploaded an implementation of WeightedQuickUnion with PathCompression in the class repository (called WQUPC). Here's a link which hopefully will work: [WQUPC.java](https://github.com/rchillyard/INFO6205/blob/master/src/main/java/edu/neu/coe/info6205/union_find/WQUPC.java)