Adam Frenkel
AnthroChassidus

My implementation of AnthroChassidus works primarily by implementing weighted quick-find from Sedgewick and adding on path compression myself. This yields the desired results of the constructor running at nearly O(n) time and the other methods running at O(1) time. I ran a test to prove that the constructor does run at O(n) time, and the results showed that there was a rate of growth of around 2.1 as n doubled; in other words the program runs at linear time. For completeness, here are the average results:

n	Nanoseconds	Rate of Growth
250	500611.0	0.9
500	499569.3	1.0
1000	496027.7	1.0
2000	576027.7	1.2
4000	715417.0	1.2
8000	1064305.3	1.5
16000	1626152.7	1.5
32000	3055541.7	1.9
64000	5344527.7	1.7
128000	10498472.3	2.0
256000	21387292.0	2.0
512000	51241249.7	2.4
1024000	141703597.3	2.8
2048000	391504847.3	2.8
4096000	874015666.7	2.2
8192000	2045418944.3	2.3
16384000	4465101486.0	2.2
32768000	9271979097.0	2.1
65536000	19046416278.0	2.1

The nitty gritty details of my implantation are the following: I begin by initializing a weighted quick union of size n. This is actually quite an expensive operation as it requires creating and initializing every value of two arrays of size n. One array keeps track of the root that the person belongs to (ie: the root which contains the known members of his chassidus), and the other keeps track of the size of the chassidus that he's apart of. Then the constructor goes through all of array a and b checking for valid input and then calling union, inorder to make all the subtrees.

Union works just as it does in the textbook, find is called twice and then the smaller sub-tree is attached to the larger one. The key addition that I made was that I implemented path compression in find to bring union to very near to O(1) as per Sedgewick (page 231). This means that all the union calls to union take O(n) time, rather than O(logn) had I used regular union. After this I go through every element and call find in order to save the root of every element for easy O(1) time later on any call to nShareSameChassidus. Addititonally, because the union Find keeps track of the count of total subtrees, one simply returns that value to get the lower bound on Chassidus type, which takes O(1) time.