AnthroChassidus Write up

Benjamin Katz

October 2021

1 Code Overview

My code implements a tree based weighted quick union with path compression in order to ascertain the amount of different chassiduses. It starts by calling union on all pairs. This works by calling find on both chassidim(who's nodes are stored in a hashmap) which traverses up the tree to find their roots. At this point path compression is utilized and all the visited grand children of the root are assigned as direct children of the root. If the two chasidim return different roots, the root with the smaller size is set as a child of the root with the larger size and the size variable of the new root is updated. If they already have the same root no action is performed.

2 Lower Bound and Share same Chassidus

At the end of the constructor the program looks at every node and sees if it is a root. It then adjusts a variable that is initialized as the entire population, subtracts the size of the node, and adds one. This leaves us with the lower bound of how many chassiduts there could be (although it is more of an upper bound because it is impossible for there to be less types of chassidut). This visits every node so it is linear time. Share same chassidus simply returns the size of its root.

3 Time Complexity

Since every call to both union and find are constant time and the program must run a set amount of union and finds on every pair of chassidim, the time complexity is linear. Because of path compression the tree is very flat and therefor a node has to make very few traversals in order to find the root therefore find and union are linear.