Anthrochassidus Algorithm

As instructed, the algorithm contains all of the logic in the constructor, with the methods simply "getting" information already stored in the underlying data structure. After the constructor is ran, a HashMap mapping people interviewed to a HashSet representing members of their chassidus is generated. A HashMap was used as it provides O(1) access to get, put, size, and contains (and the same with HashSet). In addition, a count of how many sets (representing types of chassidus) is stored and updated. Therefore, when the *getLowerBoundOnChassidusTypes* method is called, the algorithm simply returns the total numbers of people (given as a parameter into the constructor) subtracted by the count of how many sets, resulting in the desired lower bound on chassidus types (O(1) runtime). When the *nShareSameChassidus*, a get is called to get the set represting that person's chassidus, and the size of it is returned (both calls are O(1) runtime in a HashMap).

The logic itself is contained in a for loop which loops through each element in the two arrays simultaneously, hence the O(N) logic (for the most part). The logic is divided into four parts:

1) If neither interviewee has data stored in the map, a new set is created containing both people, and each person is added to the map with their ID as the key, and the new set as the value.

2 + 3) If one of the interviewees, A, has data is stored in the map, the ID of the other interviewee, B, is added to the set corresponding to A in the map, and B is added to the map with his/her ID as the key and that set as the value.

4) If both interviewees have data stored in the map, their corresponding sets must be merged, and references must be updated. All the IDs in the set corresponding to B is added to the set corresponding to A, and every reference of B in the map is replaced with a reference to A. Although this slows down the algorithm, the more it happens, the less frequent the next merge will be.