3) Among collection of n accounts, the company will not disclose the identity of the account holders directly without a subpoena, but will tell if two accounts belong to the same user. We need to find if there is a set of more than n/2 of them that are all associated with the same user.

**Algorithm:**

1. Divide the accounts into two equal groups (N1, N2). If a user has to have more than n/2 of the accounts then he has to have more than half of the accounts in any one of the groups of accounts N1 or N2
2. Similarly in group N1 if we split it into two equal groups(N11, N12), then the user must have at least more than half the accounts in either N12 or N12 for him to have more than half the accounts in N1
3. Thus, we can recursively run an algorithm on each of the groups N1 and N2 to see if there exists an user who has more than half the accounts in either N1 or N2
4. However, if a user has more than half the accounts in one group alone doesn’t mean he has more than half the accounts overall. So for just one account of the user we can compare with every other account and check if the count > n/2

Count-check(N[], n) {

If n == 1 then return N

If n == 2 then

If the accounts belong to same user then return any one account

Else

I = Count-check(N[1, n/2], n/2)

J = Count-check(N[(n/2)+1, n], n/2)

If I is not null then

Check account I with all accounts in N[]

if count of similar accounts to I > n/2 then Return I

If J is not null then

Check account J with all accounts in N[]

if count of similar accounts to J > n/2 then Return J

If I and J are both null then Return null

}

**Proof of correctness:**

* As mentioned before, for a user to have more than half the accounts in a group of size n, then the user must have more than half the accounts in any one of the two subgroups of size n/2
* And since we are checking the account of such a user with every other account at that level, we are making sure the user has a total of more than n/2 accounts as even if a user has more than half the accounts in one group alone doesn’t mean he has more than half the accounts overall

**Proof of Termination:**

* Since the function gets called recursively and in each recursive call either an account is returned or null is returned if there exists no account with majority class, the algorithm will terminate.

**Asymptotic Complexity:**

Algorithm has two recursive calls each with a maximum of 2n tests done outside of the recursive calls in each level. Therefore the complexity is

T(n) = 2T(n/2) + 2n

**By Master Theorem:**

Case 2: f(n) = Θ(nk logp n) for k = logb a, Then T(n) = Θ(nk logp+1 n)

In this example: K = 1, p = 0, a =2, b = 2, f(n) = 2n

Therefore, **T(n) = O(n log n)**

**Space Complexity – O(n)**