Lance Michael O. Co

2 BS CS-DGDD

CSCI 30 B / CS 110 B

1. Algorithm: COUNT ( L , a )

Input: An array L

An account a

Output: The number of accounts that a owns in L

For i in L do

If COMPARE ( L[i] , a )= true do

Counter ← Counter+1

1. Since COMPARE has a cost of 1 and all other operations are free, the Big O notation only depends on how many times COMPARE is used. Since COMPARE is used n times, COUNT is an O(n) notation.
2. Algorithm: CHECK (L)

Input: An array L, with n elements

Output: If n/2 accounts are owned by the same entity, return the account. Otherwise, return NULL

1. If L is empty or L’s length is 1 return NULL
2. Split L into n subproblems, with each subproblem having a size of 1
3. Using the first subproblem of L, run COMPARE on the rest of the subproblems of L
   1. If COMPARE returns TRUE, append the subproblems together
   2. If COMPARE returns FALSE, do nothing
   3. If COMPARE reaches the end without having returned TRUE, FIRST L becomes NULL.
4. Run CHECK recursively using the next non-concatenated subproblem of the rest of L.
5. Once all the subproblems have run through CHECK, if the number of elements in any of the concatenated subproblems is greater than n/2, return the first element of said subproblem.
   1. If none of the subproblems are greater than n/2, return NULL.
   2. It is correct for the base cases since if the array is empty, there are no accounts for any entity to own, thus CHECK is NULL. If the array still only has 1 element after CHECK is run on it, then it can be said that it is a unique account. As such, it is not necessary to check if that owner owns n/2 accounts since he only has 1 account.
   3. It is divided into n subproblems where each subproblem has a length of 1. This is due to the possibility that each account has a unique owner.
   4. By grouping together the subproblems that share the same owner, it is now possible to count each array of accounts with the same owner to check if the owner owns more than n/2 accounts or not.
   5. T(n)=nT(1)+O(n)
   6. Master Theorem