Q6) a

// X, Y, ■ be three set of integers from the range [-M -- M] having almost N elements. These are considered as global.

// UniqueElements(): returns the number of unique elements in X ∪ Y.

```
def UniqueElements():
    P_x = a_0 + a_1 x + a_2 x^2 + ... + a_{N-1} x^{N-1}   where a_i ∈ X
    P_y = b_0 + b_1 x + b_2 x^2 + ... + b_{N-1} x^{N-1}   where b_i ∈ Y
    P_z = FFTMultiply(P_x, P_y)   // taken from Jeff Erickson
                                  // O(nlogn)
    // finding the elements present in both X and Y
    // let h[] be a hashset, commonElecount be a counter
    ∀x in X:                      // O(n)
        h[x] = h[x] + 1
    ∀y in Y:                      // O(n)
        h[y] = h[y] + 1
        if h[y] > 1
            commonEleCount = CommonEle Count + 1
    maxDegree = degree of P_z
    return maxDegree - commonEle Count
```

Analysis

Let T(n) be the time complexity of UniqueElements()

$$T(n) = O(n\log n) + O(n) + O(n)$$

So, $T(n) \cong O(n\log n)$

Explaination

The concept of this algorithm is from basic set theory operation

$$|X \cup Y| = |X| + |Y| - |X \cap Y|$$

in the algorithm

maxDegree = $|X| + |Y|$

commonEleCount = $|X \cap Y|$

using polynomial multiplication we are finding $|X| + |Y|$ and using hashset we are counting common elements of set X and Y.

So, finally number of elements in $(X \cup Y)$ is returned.

Q.6) b

// X, Y, Z be three set of integers from the range $(-M, \ldots, M)$ having at most N elements. These are considered as global.

// triplet(): will return true if there is a triplet x from X, y from Y, and z from Z such that $x+y+z=0$

1. def triplet():
2. $\forall x$ in X
   $x = x + M$ // O(n)
3. $\forall y$ in Y
   $y = y + M$ // O(n)
4. $\forall z$ in Z
   $z = z + M$ // O(n)
5. P_x = polynomial representing set X where the powers of x in the polynomial are the elements of X and the all the coefficients are 1.
6. P_y = polynomial representing set Y where the powers of x in the polynomial are the elements of Y and all the coefficients are 1.
7. P_z = polynomial representing set Z where the powers of x in the polynomial are the elements of Z and all the coefficients are 1.
8. P_a = FFTMultiply (P_x, P_y)
9. P_f = FFTMultiply (P_a, P_z)

   } taken from Jeff Ericksons book, where FFTMultiply has O(nlogn) complexity.

10. if (P_f contains x^{3M} as a polynomial term)
    then triplet-exist-
    return TRUE
11. else
    return FALSE

Analysis

Let $T(n)$ is the time complexity of triplet()

$$T(n) = 3 * O(n) + 2 * O(n \log n) + O(d)$$

here d = maximum degree of the polynomial P_f

~~For polynomial P_f it contains the elements of X, Y, Z as powers of x, so while multipl~~

For polynomial P_f, the powers of its terms gets added (where those powers belongs from set X, Y, Z).

The range of numbers in set X, Y, Z lies in [0...2M] as they are modified in line 2, 3, 4 of the algorithm.

Thus $O(d) \cong O(2m) \cong O(m) \cong O(n)$

Thus $T(n) \cong O(n \log n)$

Explaination

FFTMultiply function can't handle negative numbers, so to make them positive M is added to all elements of set X, Y, Z as shown in line 2, 3, 4.

We are looking for triplet x, y, z where $x+y+z=0$ in P_f, the degree is $(x'+y'+z')$ as it is obtained by multiplying P_x, P_y, P_z. [where x', y', z' are the modified elements of set X, Y, Z]

now, $x' + y' + z' = (x+M) + (y+M) + (z+M)$

$$x' + y' + z' = x + y + z + 3M$$

if $x+y+z = 0$ then

$$x' + y' + z' = 3M$$

Therefore if x^{3M} term exists in P_f then triplet exists.