[B - Powers of Two](https://vjudge.net/problem/CodeForces-702B" \t "_blank)

You are given *n* integers *a*1, *a*2, ..., *an*. Find the number of pairs of indexes *i*, *j* (*i* < *j*) that *ai* + *aj* is a power of 2 (i. e. some integer *x* exists so that *ai* + *aj* = 2*x*).

**Input**

The first line contains the single positive integer *n* (1 ≤ *n* ≤ 105) — the number of integers.

The second line contains *n* positive integers *a*1, *a*2, ..., *an* (1 ≤ *ai* ≤ 109).

**Output**

Print the number of pairs of indexes *i*, *j* (*i* < *j*) that *ai* + *aj* is a power of 2.

**Examples**

**Input**

4  
7 3 2 1

**Output**

2

**Input**

3  
1 1 1

**Output**

3

**Note**

In the first example the following pairs of indexes include in answer: (1, 4) and (2, 4).

In the second example all pairs of indexes (*i*, *j*) (where *i* < *j*) include in answer.

**APPROACH 1:**

#include <iostream>

#include <vector>

#include<algorithm>

using namespace *std*;

bool isPowerOf2(int n) {

return (n > 0 && !(n & (n - 1)));

}

int main() {

int t; *cin* >> t;

*vector*<int> arr(t);

for (auto i = 0; i < t; i++)

*cin* >> arr[i];

int count = 0;

for (auto i = 0; i < t; i++)

for (auto j = i + 1; j < t; j++) {

if (isPowerOf2(arr[i] + arr[j]))

count++;

}

*cout* << count << "\n";

return 0;

}

Note: The above approach will give TLE.

APPROACH 2:

### **Efficient Approach**

Count how many pairs are there such that, Ai+ Aj= 2x, can also be re-written or seen as for each Aj, How many Ais are there such that Ai = 2x - Aj

This converts it to a SEARCH PROBLEM.

So, what if we take a count variable as count = 0.

for j = [0, n-1]

  for p = [0, 31]

    count += countElementsInArrayEqualToGivenValue(power(2,p)-A[j]))

return count;

Rather than counting values at runtime, what if we take a frequency Map in beginning and fill it by looping over array once?

long long countPairs(int a[], int n){  
    long long c = 0;   
    map<int, int> mp;  
    for i = [0, n-1]  
        if(mp.containsKey(a[i]))  
           mp[a[i]] = mp[a[i]]+1;  
        else  
           mp[a[i]] = 1;  
    for j = [0, j-1]  
        for p = [0, 30]  
            if( mp.containsKey((1<<p) - a[j]))  
               c = c + mp[(1<<p) - a[j]];  
    return c;  
}

#include <iostream>

#include<algorithm>

using namespace *std*;

int countPairs(int\* arr, int n) {

int c = 0;

for (auto i = 0; i < n; i++)

{

int k = arr[i];

for (auto p = 0; p < 30; p++)

{

long long d = (1 << p) - k;

if (d <= 0)continue;

int lb = *lower\_bound*(arr + i + 1, arr + n, d) - arr;

int ub = *upper\_bound*(arr + i + 1, arr + n, d) - arr;

if (lb == n)continue;

c += (ub - lb);

}

}

return c;

}

int main() {

int t; *cin* >> t;

int\* arr = new int[t];

for (auto i = 0; i < t; i++)

*cin* >> arr[i];

*sort*(arr, arr + t);

auto count = countPairs(arr,t);

*cout* << count << "\n";

return 0;

}