

Problem E - Make My Array Equal

- **Question Analysis:**

আমাকে একটা **array A** দেওয়া আছে। এই **array A** এর সব **elements** আমাকে **Equal** করতে হবে।

Operation (any number of times) :

যেকোনো ২টা **index i, j choose** কর, যেখানে ($i \neq j$)

$A_i = A_i + A_j$ করতে হবে।

Possible হলে, 'Yes', else 'No' print করতে হবে।

- **Observation:**

1. যেহেতু , আমাদের **operation** হচ্ছে , **addition** - তাহলে বাড়বে মান, কমবে না। তাই **Possible Way** হচ্ছে **Maximum element** টা-কেই **equal** মান ধরা।
 2. কখনো কি, এই **Maximum element** এর থেকে বড় মানে সবগুলো কে নিয়ে যাওয়া সম্ভব?
না। - কারণ, একবার কিন্তু মান **Change** হয়ে গেলে , এর আগের মানের **Help** পাওয়া **Possible** না।
 3. $A_i = 0$, হলে, সেটা কে **easily max value** - তে নিয়ে যেতে পারি।
 4. যদি, কখনো **array** - তে **0** এবং, **maximum** মান ছাড়া অন্যকোনো মান থাকে, তাহলে কি কখনো সব **elements** আমাকে **Equal** করা **possible?** না। কারণ, এখানে **Ultimately, maximum element** এর **help** নিয়েই, **operation** করতে হচ্ছে। যদি অন্য মান থাকে, তাহলে $value + maximum > maximum$ হয়ে যাচ্ছে, (**Obs-2**) হয়ে যাচ্ছে।
-

- **Implementation:**

শুধু **check** করতে হবে, **0 & maximum value** ছাড়া অন্য **extra positive value** আছে কিনা।

না থাকলে -> **Yes.**

থাকলে -> **No.**

Set, Map Use করা যেতে পারে। অথবা, **vector** এর, **erase and unique method.**

- **Code:**

```
#include <bits/stdc++.h>
using namespace std;

#define ll long long

int main(){
    int tc=1;
    cin >> tc;
    while(tc--){
        int n;
        cin >> n;
        set<int> st;
        for(int i=0; i<n; i++){
            int x;
            cin >> x;
            if(x!=0) st.insert(x);
        }

        if(st.size()>1) cout << "NO\n";
        else cout << "YES\n";
    }

    return 0;
}

/* Author: Hridoy Barua (CS Instructor Phitron) */
```

Problem F - Equal Pairs (Easy)

- **Question Analysis:**

আমাকে একটা **array A** দেওয়া আছে। আমাকে **f(A)** - বের করতে হবে, তাও আবার **possible maximum f(A)**।

Operation -

- যত **Ai = 0** আছে, সবগুলোকে - যেকোনো **positive value** দিয়ে **replace** করতে হবে।

f(A) হবে - **number of pairs (i,j)**, যেখানে **i < j**, **Ai = Aj**.

এখন, এমনভাবে **operation** টা করতে হবে যাতে **f(A)** **maximum** হয়।

- **Observation:**

1. **Finally**, সব **0** **replace** হয়ে যাবে।
2. আমি কোনো মান দিয়ে **replace** করব। **Greedyly** - যেই মান **already non-zero** এবং এর **Frequency** বেশি।
3. কেন? সব **0** **same** একটা **element** দিয়েই কেন করব?

Proof:

যদি **value x** এর **frequency** হয় **n**. তাহলে -

$$\begin{aligned}\text{Pair নেওয়ার combination} &= nC2 \\ &= n! / 2! (n-2)! \\ &= n(n-1)/2\end{aligned}$$

4. এখন, ধর -

1 1 0 0 0 0 3 3 - এখানে, **0** কে অর্ধেক অর্ধেক করে, **1 1 1 1 3 3 3 3** বানান থেকে, পুরো **1 1 1 1 1 1 3 3** অথবা, **1 1 3 3 3 3 3 3** বানান বেশি **optimal**। কারণ, এতে **maximum frequency** বাড়বে, এতে **n(n-1) => 6*5 = 30** যা, আগের **maximum frequency 4*3=12** থেকে বেশি।

- **Implementation:**

যেই **element** এর **frequency** বেশি, তার **frequency** আরো বাড়ব, **0** ওই **element** দিয়ে **replace** করে।

তারপর যত **same element** এর **frequency** আছে, তাদের **pair combination count** করে **ans** এ যোগ করব।
Map Use করতে পারি।

- **Code:**

```
#include <bits/stdc++.h>
using namespace std;

#define ll long long

ll count(ll n){
    return n*(n-1)/2;
}

int main(){
    int tc=1;
    cin >> tc;
    while(tc--){
        int n;
        cin >> n;
        map<int,int> mp;
        int mx=0, cnt=0, zero=0;
        for(int i=0; i<n; i++){
            int x;
            cin >> x;
            if(x==0) zero++;
            else mp[x]++;
            if(mp[x]>cnt){
                mx=x;
                cnt=mp[x];
            }
        }
        ll ans = count(zero+cnt);
        for(auto [val,cnt]: mp){
            if(val!=mx) ans += count(cnt);
        }
        cout << ans << "\n";
    }
    return 0;
}

/* Author: Hridoy Barua (CS Instructor Phitron) */
```

Problem G - Non-Primes 101

- **Question Analysis:**

আমাকে একটা **array A** দেওয়া আছে।


বের করতে হবে এমন ২ টা **pair of index (i, j)** - যেখানে,

- $i \neq j$
- $A_i + A_j$ non-prime number.

Multiple answer হতে পারে। **possible** না হলে **-1 output** করতে হবে।

- **Some Key Takeaways:**

even + even = even 

odd + odd = even 

even + odd = odd 

odd + even = odd 

- **Observation:**

1. যদি, দুইটা **even number** পাই তাহলে, এদের **Sum even**, যা কখনো **Prime** হবে না। তাহলে এদের **index answer**
2. যদি, দুইটা **odd number** পাই (২ টাই 1 ব্যতিত) তাহলে, এদের **Sum even**, যা কখনো **Prime** হবে না। তাহলে এদের **index answer**
২ টাই 1 হলে, $1 + 1 = 2$, যা **Prime**। বাকি **odd** এর ক্ষেত্রে এমন সমস্যা নাই।
3. যদি ১টা **even** ও ১টা **odd** হয়, তাহলে শুধু এদের **check** করলেই হলো যে, এদের **sum prime** কিনা। হলে **-1**, নাহলে **(i, j)**.

- **Implementation:**

Prime check - এর জন্য, **Sieve of Eratosthenes** দিয়ে **Precompute** করে রাখতে পারি। এতে নিজের নিজের **code use** করতে পার।

Observation এর - 3 টা, **case** - ই, **just check** করতে পারি।

এই **Corner case** টা নিয়ে সর্বক থাকতে হবে।

২ টাই 1 হলে, $1 + 1 = 2$, যা **Prime**

• Code:

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
vector<bool> is_prime(1000, true);
void Sieve(ll n) { // TC: O(nloglogn)
    for (ll p=2; p<=n; p++){
        if (is_prime[p]){
            // prime.push_back(p);
            for (ll i=p*p; i<=n; i+=p){
                is_prime[i] = false;
            }
        }
    }
}

int main(){
    Sieve(500);
    int tc=1;    cin >> tc;
    while(tc--){
        int n;    cin >> n;
        int e1=-1, e2=-1, o1=-1, o2=-1;
        vector<int> a(n);
        for(int i=0; i<n; i++){
            cin >> a[i];
            if(a[i]&1){
                if(o1==-1) o1=i;
                else if(!(a[o1]==1 && a[i]==1)) o2=i;
            }
            else{
                if(e1==-1) e1=i;
                else e2=i;
            }
        }

        if(e1!=-1 && e2!=-1) cout << e1+1 << " " << e2+1 << endl;
        else if(o1!=-1 && o2!=-1) cout << o1+1 << " " << o2+1 << endl;
        else if(e1!=-1 && o1!=-1 && !is_prime[a[e1]+a[o1]])
            cout << e1+1 << " " << o1+1 << endl;
        else cout << -1 << endl;
    }
    return 0;
}

/*    Author: Hridoy Barua (CS Instructor Phitron)    */
```