# CRACK A HACK

## Magic Value

Course: Algorithmic Problem Solving

Course code: 17ECSE309

By:

Sujaykumar Kulkarni

USN: 01FE15BEC203

# Introduction:

Magic Value for a given array B -> [$B_1$, $B_2$, $B_3$,……, $B_m$] is defined as follows:

1. For, $1 \le i \le j \le m$, define V( i, j) = i * gcd( $B_i$, $B_{i+1}$, $B_{i+2}$,……., $B_j$)

2. Calculate $V_{min}$ and $V_{max}$.

3. Magic value of array B = ($V_{max}$ - $V_{min}$) * m.

Problem:

Given, input array A, compute its possible sub arrays and then, find magic values of all the sub arrays and print the sum of magic values.

Note: Number of subarrays from array A = n * (n + 1) ÷ 2

# Implemented Solution:

For all possible subarrays we need to find the magic value. So, consider the following equation,

$$\text{ans} = \sum_{1 \le i \le j \le n} \left[ v_{\max}(i,j) - v_{\min}(i,j) \right] \cdot (j - i + 1)$$

Here, ( $j - i + 1$ ) is length of subarray.

Now, $V_{min}$ is nothing but ‘gcd’ of the sub array, and $V_{max}$ is obtained by calculating for V( i , j ). Then answer is found by summing all the magic values.

## Code: c++

```
#include <iostream>
#include <algorithm>
#define MOD 1000000007
#define MX 200003
using namespace std;
typedef long long ll;
ll a[MX+2];
ll n;
ll gcd(ll a, ll b){
        if (a == b)
                return a;
        if (a == 0)
                return b;
        return gcd(b%a, a);
}
int main() {
   ios_base::sync_with_stdio(0);
   cin.tie(0); cout.tie(0);
   cin >> n;
   for(ll i=1; i<=n; i++){
      cin >> a[i];
```

```
    }
    ll sum=0;
    for(ll i=1; i<=n; i++){                 /* Magic Value */
        ll Gcd=0;
        ll ma=0LL;
        bool flag=0;
        for(ll j=i; j<=n; j++){
            if(a[j]==0)
                flag=1;
            if(!flag)
                Gcd=gcd(Gcd,a[j]);
            else
                Gcd=0;
            ma = max(ma,(j-i+1LL)*a[j]);
            sum += (ma-Gcd)*(j-i+1LL);
            if(sum>=MOD)
                sum %= MOD;
        }
    }
    cout << sum%MOD << "\n";
    return 0;
}
```

## Efficiency:

The implemented code has complexity of $O(n^2)$, It's not an efficient way tough. An efficient implementation is using segment tree data structure with lazy propagation.

## References:


https://en.wikipedia.org/wiki/Greatest_common_divisor

https://en.wikipedia.org/wiki/Segment_tree

https://www.hackerearth.com/practice/notes/segment-tree-and-lazy-propagation/