Lecture 3
Math & Number Theory - 2

Anik Sarker[1]

Postgraduate Student,
Department of CSE, BUET
ECE Building, West Palasi, Dhaka-1205, Bangladesh

Abstract. This lecture is a part of competitive programming training lectures prepared for Eastern University, Dhaka. This lecture introduces some Math and Number Theory based concepts : Median Properties, Harmonic Number Properties, Range Prime Factorization, Modular Multiplicative Inverse, ${}^nP_r$ and ${}^nC_r$ calculation, Binary Search Technique.

1 Prime Factorization for each $i \in [1, n]$

1. Do normal sieve, insert each i in the list for j
2. The total size of all the lists will be $\leq O(n\ logn)$
3. Calculate Phi function, Divisor count, Divisor sum for each $i \in [1, n]$

```
// Calculates all the prime factors for each i in [1,n]
const int MAXN = 100005;
bool mark[MAXN];
vector<int> Factors[MAXN];
void SieveToPrimeFactorization(int n){
    for(int i=2; i<=n; i++){
        if(mark[i] == false){
            for(int j=2*i; j<=n; j+=i){
                Factors[j].push_back(i);
                mark[j] = true;
            }
        }
    }
}
```

2 Median Properties

- Let $A = \{a_1, a_2, \ldots, a_n\}$
- If n odd, $a_{\frac{n+1}{2}}$ is median
- If n even, both $a_{\frac{n}{2}}$ and $a_{\frac{n}{2}+1}$ are medians
- **Solution** to **this classic problem** is median.

3 Harmonic Number Property 2:

* There can be $2 * \sqrt(n)$ distinct values among all $\frac{n}{i}$ for i $\in$ [1, n]
* **Solution** to **this classic problem** uses this trick.

4 Modular multiplicative inverse

- We know $(a * b) \mod m = ((a \mod m) * (b \mod m)) \mod m$
- But $\frac{a}{b} \mod m \neq \frac{a \mod m}{b \mod m} \mod m$
- $\frac{1}{a} \mod m$ is not even defined
- Because modulo operation can be applied only on integers.
- x is called modular multiplicative inverse of a modulo m if

$$(a * x) \mod m = 1$$

- Modular inverse may not exist for some (a, m) **(SKIP)**
- There may be more than 1 modular inverse for some (a, m) **(SKIP)**
- There will be **exactly 1 modular inverse** if a and m is coprime.

```
// returns modular inverse in O(mod)
ll ModularInverseSlow(ll n, ll mod){
    for(int i=0; i<mod; i++){
        if( (n * i) % mod == 1) return i;
    }
}
```

5 Fermat's Little Theorem :

- for **prime** m, $\frac{1}{a} \mod m = a^{m-2} \mod m$
- What if m **not prime** : will discuss later.

```
// returns modular inverse in O(log(mod))
ll ModularInverseFast(ll n, ll mod){
    return FastPowerCalc(n, mod-2, mod);
}
```

6 ${}^nP_r$ and ${}^nC_r$ calculation

- ${}^nP_r = \frac{n!}{(n-r)!}$
- ${}^nC_r = \frac{n!}{r!\ (n-r)!}$

```
const int MAXN = 100005;
const int mod = 1000000007;

// Do initial precalculation
ll Fact[MAXN];
ll invFact[MAXN];
void PreCalc(int n){
    Fact[0] = invFact[0] = 1;
    for(int i=1; i<=n; i++){
        Fact[i] = (Fact[i-1] * i) % mod;
        invFact[i] = ModularInverseFast(Fact[i], mod);
    }
}

// Answer per query in O(1)
ll nCr(ll n, ll r){
    ll Up = Fact[n];
    ll Down1 = invFact[r];
    ll Down2 = invFact[n-r];

    ll ret = (Up * Down1) % mod;
    ret = (ret * Down2) % mod;
    return ret;
}
```

7 Binary Search

1. **Problem Statement :**
   Given a function f and an value p.
   - find maximum x such that $f(x) \leq p$
   - find minimum x such that $f(x) \geq p$
2. **Condition :** monotonic
   - $f(x-1) \leq f(x) \leq f(x+1)$
   - $f(x-1) \geq f(x) \geq f(x+1)$

8 Problemset Discussion

- Long - 1
- Long - 2
- Long - 3
- Onsite - 1