Number Theory I

Agenda

- Primes
- Divisors
- Prime Factors
- Sieve of Eratosthenes

Primes

prime numbers are whole numbers greater than 1, that have only two factors or divisors: 1 and the number itself

2	3	5	7	11
13	17	19	23	29
31	37	41	43	47
53	59	61	67	71
73	79	83	89	97

Primes

- How to check if a number (n) is prime or not?
- The simplest way is to iterate through the numbers from 2 to n-1 and check if there is a divisor among them or not
- This runs in O(n) time complexity. Can we do better?

```
bool isPrime(ll n){
    for(ll i=2; i<=n-1; i++)
        if(n%i==0)
            return false;
    return true;
}
```

Primes (Square Root Rule)

Let **n = a * b**

- **min(a,b) <= sqrt(n)**
  Example: 16 = 1 * 16
  16 = 2 * 8
  16 = 4 * 4
- If **n** is divisible by **a**, so **n** is also divisible by **n/a** which is **b**

Primes (Square Root Rule)

- How to check if a number (n) is prime or not?
- Using the square root rule, we need to iterate only through elements from 2 to sqrt(n) and check if there is a divisor among them or not
- This runs in O(sqrt(n)) time complexity which is faster that O(n)

```
bool isPrime(ll n){
    for(ll i=2; i*i<=n; i++)
        if(n%i==0)
            return false;
    return true;
}
```

Divisors

Using the same square root rule, we can get the divisors for n in O(sqrt(n))

```
vector <ll> getDivisors(ll n){
    vector <ll> divs;
    for(ll i=1; i*i<=n; i++){
        if(n%i==0){
            divs.push_back(i);
            if(i!=n/i) divs.push_back(n/i);
        }
    }
    return divs;
}
```

Prime Factors

- Prime Factors for a number (n) are a set of prime numbers that decompose this number

- Examples:
  - 8 → 2^3
  - 15 → 3^1 * 5^1
  - 14 → 2^1 * 7^1
  - 11 → 11^1
  - 60 → 2^2 * 3^1 * 5^1

Prime Factors

How to find the prime factors for a number (n)?

- Using the square root rule, we can iterate through elements from 2 to sqrt(n), and keep dividing n over its divisors. This is runs in O(sqrt(n)) time complexity
- Example N=60:
  - 60 is divisible by 2 → N=60/2=30 , {2^1}
  - 30 is divisible by 2 → N=30/2=15 , {2^2}
  - 15 is divisible by 3 → N=15/3=5 , {2^2, 3^1}
  - 5 is divisible by 5 → N=5/5=1, {2^2, 3^1, 5^1}

Prime Factors

```
vector<pair<ll, ll>> factorize(ll n){
    vector<pair<ll, ll>> fact;
    for(ll i=2; i*i<=n; i++){
        int cnt=0;
        while(n%i==0){
            n/=i;
            cnt++;
        }
        if(cnt) fact.push_back({i,cnt});
    }
    if(n>1) fact.push_back({n,1});
    return fact;
}
```

Sieve of Eratosthenes

How to find all prime numbers from 1 to n?

- The main idea of Sieve of Eratosthenes is a number is prime until it's proven to be not prime
- Sieve of Eratosthenes works as follows:
  1. Initially all numbers are prime except 1
  2. For each number from 2 to sqrt(n): if it's still prime, mark all its multiples as not prime
- The time complexity of Sieve is O(n log (log n))

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120

Prime numbers

Sieve of Eratosthenes

```
vector<bool> sieve(ll n){
    vector<bool>isPrime(n+1,1);
    isPrime[0]=isPrime[1]=0;
    for(ll i=2; i*i<=n; i++)
        if(isPrime[i])
                for(ll j=i*i; j<=n; j+=i)
                    isPrime[j]=0;
    return isPrime;
}
```

To Solve

- T-primes
- Almost Prime
- Number into Sequence