Programming Challenges I

Session 4

CSCI 485/4930 - Spring 2018

Primality Test

Primality Test – Naïve Algorithm O(N)

```
bool isprime(int n) {
    if(n < 2) return false;

    for(int i=2; i<n; i++)
        if(n%i == 0)
            return false;

    return true;
}
```

Primality Test – $O(\sqrt{N})$

```
bool isprime(int n) {
    if(n < 2)
        return false;

    for(int i=2; i <= n/i; i++)
        if(n%i == 0)
            return false;

    return true;
}
```

100	
2	50
4	25
5	20
10	**10**
20	**5**
25	**4**
50	**2**

Note: i <= n/i is equavelant to i <= sqrt(n)

Divisors Generation

Divisors Generation – O($\sqrt{N}$)

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

100
2
50
4
25
5
20
10

Factorization

$$6 = 2 \times 3$$

$$12 = 2 \times 2 \times 3 = 2^2 \times 3$$

$$16 = 2 \times 2 \times 2 \times 2 = 2^4$$

$$250 = 2 \times 5 \times 5 \times 5 = 2 \times 5^3$$

$$510{,}510 = 2 \times 3 \times 5 \times 7 \times 11 \times 13 \times 17$$

$$42{,}059 = 137 \times 307$$

Factorization – O($\sqrt{N}$)

```
vector<int> getFactors(int n){
    vector<int> ret;
    for(int i=2; i<=n/i; i++){
        while(n % i == 0){
            ret.push_back(i);
            n /= i;
        }
    }
    if(n != 1){
        ret.push_back(n);
    }
    return ret;
}
```

Sieve of Eratosthenes

	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 Algorithm

```
const int N = 1e6;
bool notprime[N + 5];
vector<int> primes;

void seive(){
    notprime[0] = notprime[1] = 1;
    for(int i=1; i<=N/i; i++){
        if(notprime[i])
            continue;
        if(int j=i*i; j<=N; j+=i)
             notprime[j] = 1;
    }
    for(int i=2; i<=N; i++)
        if(!notprime[i])
            primes.push_back(i);
}
```

Greatest Common Divisor

Given 2 integers a and b, with b ≠ 0, there exist a unique integers q and r such that

$$a = qb + r$$

Where r >= 0 and r < d

Euclidean Algorithm

```
int gcd(int a, int b){
    if( a % b == 0)
        return b;
    else
        return gcd(b, a % b);
}
```

Euclidean Algorithm Proof

gcd(a, b) == gcd(b,a) == gcd(b, a-b)

Let g be gcd(a, b)

a = q1 * g + 0,   b = q2 * g + 0

a – b = (q1 – q2) * g + 0   → g divides a – b

Assume there exist k > g

b = q3 * k + 0   a-b = q4 * k + 0

a = (q3 – q4) * k + 0   → k divides a and b which contradicts g = gcd(a, b)

Euclidean Algorithm Proof

gcd(b, a-b) == gcd(b, a-2b) == gcd(b, a-3b) == …… == **gcd(b, a-qb)** == **gcd(b, a%b);**

Remember:
a = qb + r → r = a - qb

LCM

```
int lcm(int a, int b){
    return a / gcd(a, b) * b;
}
```

Codeforces – 230B T-primes

https://ideone.com/Uj006K