Sieve of Eratosthenes

Finding primes in Range [1:n] without using Sieve of Eratosthenes

We can check if each number is a prime or not. To check if the number ‘i’ is prime we will traverse all the numbers till [2, sqrt(i)] can check if they divide n or not. Similarly, we do this for all the n numbers.

Time Complexity: O(n sqrt n)

Space Complexity: O(1)

```
bool check_prime(int n) {
    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0)
            return false;
    }
    return true;
}
void get_primes_till_n(int n) {
    for (int i = 2; i <= n; i++) {
        if (check_prime(i)) {
            cout << i << " ";
        }
    }
}
```

Sieve of Eratosthenes

Algorithm: We start from 2, and on each encounter of a prime number, we mark its multiples as composite.

Time Complexity: O(n log log n)

Space Complexity: O(n)

```
void primeSieve(int n) {
    int prime[n+1] = {0};
    for (int i = 2; i <= n; i++) {
        if (prime[i] == 0) {
            for (int j = i * i; j <= n; j += i) {
                prime[j] = 1;
            }
        }
    }

    for (int i = 2; i <= n; i++) {
        if (prime[i] == 0) {
            cout << i << " ";
        }
    } cout << endl;
}
```

Prime Factorization using Sieve

Explanation:

while(num ! = 1):

We keep on dividing it with its smallest prime factor.

The smallest prime factor is pre-calculated using a slightly modified prime sieve.

Since we start from 2 and go on, we mark the first multiple as the spf.

Preprocessing for Sieve: O(n log log n)

Time Complexity for factorization: O(log n)

Space Complexity: O(n)

```
void primefactor(int n) {

    int spf[n+1] = {0};
    for (int i = 2; i <= n; i++) {
        spf[i] = i;
    }
    for (int i = 2; i <= n; i++) {
        if (spf[i] == i) {
            for (int j = i * i; j <= n; j += i) {
                if (spf[j] == j) {
                    spf[j] = i;
                }
            }
        }
    }
    while (n != 1) {
        cout << spf[n] << " ";
        n = n / spf[n];
    }
}
```

Additional Question:

Find primes in the given range