EAST WEST UNIVERSITY

Department of CSE

Course Code: CSE246
Course Title: Algorithms
Lab Assignment Report

Assignment: 2
Assignment Title: Sieve Method for prime numbers

Submitted by-
Noushin Pervez
Id: 2020-1-60-189
Section: 3
Summer 2022

Submitted to-
Jesan Ahammed Ovi
Senior Lecturer
Department of Computer Science and Engineering
East West University

Submission Date: 07-09-2022

Problem Statement: Sieve Method for Prime Numbers

Description:
Given a number n, find prime numbers in a given range.

The sieve of Eratosthenes is one of the most efficient ways to find all prime numbers smaller than or equal to n.

Explanation:
At first, we create a vector of size 100000 with all the values set to 0. Here, initially all the numbers are unmarked. We assume all numbers are prime numbers.

```
vector<int>prime(100000, 0);
```

We create a function. We iterate i from 2 to n and on each encounter of a prime number (if the number is unmarked or 0), we mark its multiples as 1. We are starting the loop from 2 because 2 is the first prime number.

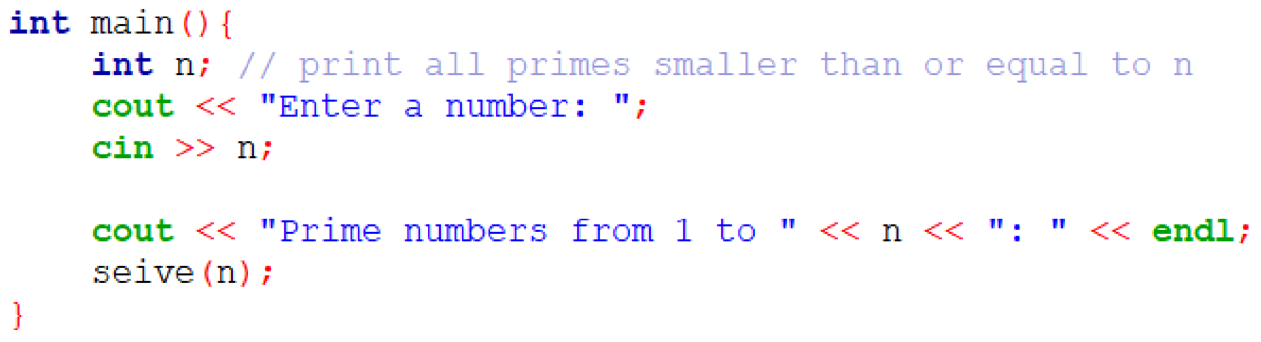
To mark the multiples of a number, we iterate from the square of the prime number (i * i) to n and increment it by i because we need to cross out all the multiples of i.

```
void seive(int n){
    for(int i = 2; i <= n; i++){
        if(prime[i] == 0){
            for(int j = i * i; j <= n; j += i)
                prime[j] = 1; // mark the multiples of numbers
        }
    }
```

To print all the prime numbers, we iterate from 2 to n and check if the number is unmarked (if the number is 0), then print it.

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

In the main function, we input a number n and call the function with parameter n.



```
int main(){
    int n; // print all primes smaller than or equal to n
    cout << "Enter a number: ";
    cin >> n;

    cout << "Prime numbers from 1 to " << n << ": " << endl;
    seive(n);
}
```

The algorithm works as follows: A number is a prime number if none of the smaller prime numbers divides it. Since we iterate over the prime numbers in order, we already marked all numbers which are divisible by at least one of the prime numbers.
Hence if we reach a cell and it is not marked, then it is not divisible by any smaller prime number and therefore has to be a prime number.

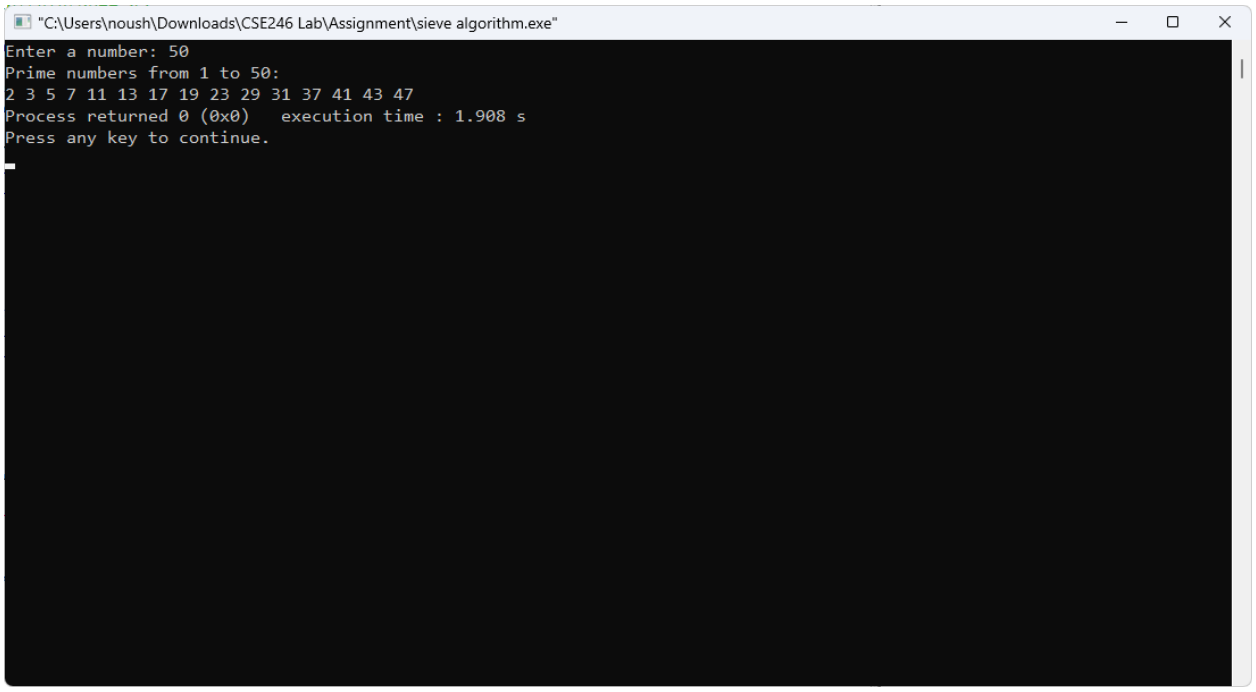
Time Complexity: O(n log log n)

Code:

```
#include<bits/stdc++.h>

using namespace std;

// initially all the numbers are unmarked
vector<int>prime(100000, 0); // creating a vector of size 100000 with all values as 0

void seive(int n){
   for(int i = 2; i <= n; i++){
      if(prime[i] == 0){
         for(int j = i * i; j <= n; j += i)
            prime[j] = 1; // mark the multiples of numbers
      }
   }

   // print the unmarked numbers
   for(int i = 2; i <= n; i++){
      if(prime[i] == 0)
         cout << i << " ";
   }
}
```

```
int main(){
    int n; // print all primes smaller than or equal to n
    cout << "Enter a number: ";
    cin >> n;

    cout << "Prime numbers from 1 to " << n << ": " << endl;
    seive(n);
}
```

Output:
We take 50 as input and print all the prime numbers from 1 to 50.


"C:\Users\noush\Downloads\CSE246 Lab\Assignment\sieve algorithm.exe"

```
Enter a number: 50
Prime numbers from 1 to 50:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47
Process returned 0 (0x0)   execution time : 1.908 s
Press any key to continue.
```