Find primes in a given range [L, R]

Sometimes, we are required to find the prime numbers in a given range [L, R], where L and R may be as large as 10^12, however, the number of elements in the range (R-L+1) is less (say 10^6).

In order to find all the primes in the given range, we can use a similar idea to that of the segmented sieve.

Since the number of elements in the range can be stored in memory, so here we are not required to split the given range into different intervals(blocks).

Pseudocode:

```
/* Input n is a positive integer, finds all the prime numbers less than or equal to sqrt(n).*/
function sieveOfEratosthenes(n)

        /*
                Declaring a sieve array whose value for the ith index is false if the ith value is
                composite, and is true if the ith value is prime
        */

        bool  sieve[n + 1], primes
        //   Initializing sieve array to be true
        for i = 1 to n
                sieve[i] = true
        sieve[1] = false

        /*
                Now iterating from 2 and for every prime value marking the multiples of that
                prime up to n as composites i.e false
        */
        for i = 2; i * i<=n; i++
                if sieve[i] equals true
                        //   Storing the prime number sieve[i]
                        primes.insert(sieve[i])
                        //   Marking all multiples of i till n to be false(composites)
                        for j = i * i; j<=n; j += i
                                sieve[j] = false

        return primes
```

```
/*
        Input L and R are positive integers, finds all the prime numbers between L to R(L and
        R inclusive)
*/
function findPrimes(L,R)

        if L > R
                return

        //   Calling simple sieve on n and storing all the prime numbers upto sqrt(n) in primes
        primes  = sieveOfEratosthenes(n)

        start = L
        end  = R
        /*
                Declaring a block array representing the current interval of size ‘end-start+1’,
                initialized to true
        */
        block[end-start+1]

        /*
                Iterating over all prime numbers in primes(unmarked) and marking all
                multiples of the current unmarked number till the end.
        */

        for prm = 0 to primes.size()-1
                /*
                        Finding the smallest number greater than or equal to start
                        that is a multiple of current prime prm[i]
                */
                prm_start = (start / prm[i]) * prm[i]
                if prm_start < prm[i]
                        prm_start = prm_start + prm[i]
                if prm_start == prm[i]
                        prm_start = prm_start + prm[i]

                /*
                        Marking all the multiples of the prime prm[i], here block[i] represents
                        the value i+start, hence in a way compressing the values.
                */
                for itr = prm_start; itr <= end; itr += prm[i]
                        block[itr-start] = false
```

```
for i = start to end
    if block[i-start] equals true
        print(i)
```

Time complexity: $O(n*\log_2(\log_2 n))$, where n is the count of numbers present in the range R-L+1.