

CS 359 Assignment - 3

Name: Somya Mehta

Roll No.: 190001058

Problem Statement:

C program which counts the number of primes between 1 and N, using MPI for parallel execution.

Approach/Logic:

If we didn't apply a parallel approach then we would have done it iteratively by running a for loop and then checking for each number if it's a prime or not.

So, if we apply parallel computing the main idea behind the solution is that we would divide the range $[1, N]$ into parts and then we will distribute the parts to all the processors available. Then, each processor will linearly solve the problem for their part as stated above. Then, the master process will receive the final computed answers from all the processors in an array. Finally, the Master process will sum all the values received in its array and output the final result.

CODE :

```
// Name: Somya Mehta
// Roll No: 190001058

#include <stdio.h>
#include <stdlib.h>
#include <mpich/mpi.h>

// Function to check if given number is prime

int is_prime(int n) {
    if (n == 1) return 0;
    int flag = 1;
    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0) {
            flag = 0;
            break;
        }
    }
    return flag;
}

// Compute number of primes for each processor
int NumberOfPrime(int myRank, int n_elements_per_process) {
    int left = myRank * n_elements_per_process + 1;
    int right = left + n_elements_per_process;
    int counter = 0;
    for (int i = left; i < right; i++) {
```

```

        if (is_prime(i)) counter++;
    }

    return counter;
}

int main(int argc, char** argv) {
    int myRank, noOfProcessors;

    int n, n_elements_per_process;

    MPI_Status status;

    MPI_Init(&argc, &argv);

    MPI_Comm_rank(MPI_COMM_WORLD, &myRank);

    MPI_Comm_size(MPI_COMM_WORLD, &noOfProcessors);

    if (myRank == 0) {
        printf("\n Enter the upper limit: ");

        scanf("%d", &n);

        n_elements_per_process = n / noOfProcessors;
    }

    // Broadcast to other processors
    MPI_Bcast(&n_elements_per_process, 1, MPI_INT, 0, MPI_COMM_WORLD);

    // Sright remaining numbers to last processor
    if (myRank == 0) {
        int remaining = n % noOfProcessors;

        MPI_Send(&remaining, 1, MPI_INT, noOfProcessors - 1, 0,
MPI_COMM_WORLD);
    }

    if (myRank == noOfProcessors - 1) {
        int extra;

```

```

        MPI_Recv(&extra, 1, MPI_INT, 0, 0, MPI_COMM_WORLD, &status);
        n_elements_per_process += extra;
    }

    // Create array in processor 0 to receive all answers
    int* recvlen = NULL;

    int counter = NumberOfPrime(myRank, n_elements_per_process);
    if (myRank == 0) recvlen = (int*)malloc(noOfProcessors * sizeof(int));

    MPI_Gather(&counter, 1, MPI_INT, recvlen, 1, MPI_INT, 0,
MPI_COMM_WORLD);

    if (myRank == 0) {
        int res = 0;

        for (int i = 0; i < noOfProcessors; i++) res += recvlen[i];
        printf("\n Number of prime numbers in [1, %d] are %d \n", n, res);
    }

    MPI_Finalize();

    return 0;
}

```

TIME COMPLEXITY :

Number of processors: p

Number of elements: n

Since we are checking for each number i if it is prime or not in $\text{root}(i)$ operations, for processor number k , number of operations

$$= \text{root}(k * x + 1) + \text{root}(k * x + 2) + \dots + \text{root}(k * x + y)$$

where $x = n / p$

$y = x + n \% p$ for last processor

$= x$ for other processors

Since all processors will be executing in parallel and the last processor takes the most of the time, the total running time of the parallel program will be equal to the running time of the last processor.

So, time taken $= \text{root}((p-1)*x + 1) + \dots + \text{root}(n)$

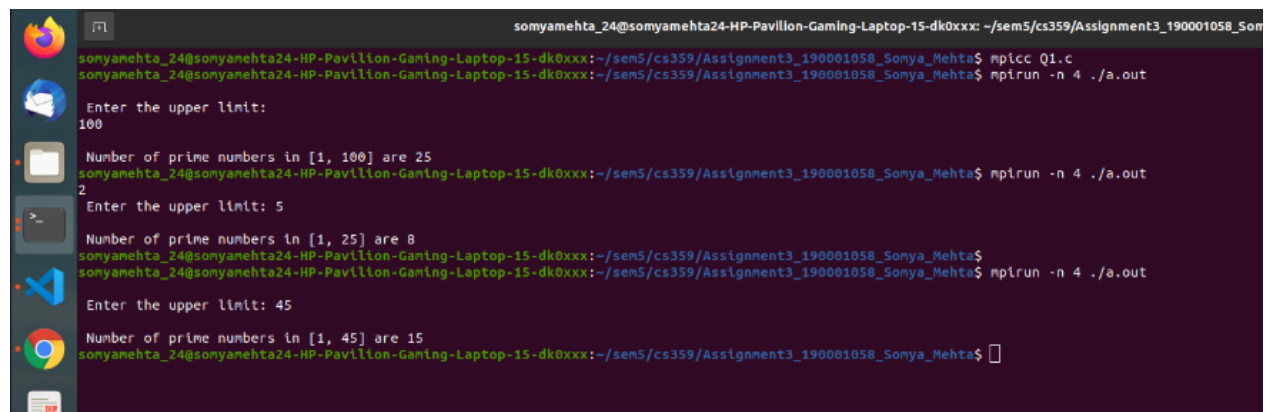
$$< \text{root}(n) + \dots + \text{root}(n)$$

(Since last processor has $x + n \% p$ numbers to work upon, there will be at most $n/p + (p-1)$ terms)

$$< \text{root}(n) * (n/p + p - 1)$$

$$= O((n/p + p) * \text{root}(n))$$

SCREENSHOT :



```
sonyamehta_24@sonyamehta24-HP-Pavillon-Gaming-Laptop-15-dk0xxx: ~/sem5/cs359/Assignment3_190001058_Son
sonyamehta_24@sonyamehta24-HP-Pavillon-Gaming-Laptop-15-dk0xxx:~/sem5/cs359/Assignment3_190001058_Sonya_Mehta$ mpicc Q1.c
sonyamehta_24@sonyamehta24-HP-Pavillon-Gaming-Laptop-15-dk0xxx:~/sem5/cs359/Assignment3_190001058_Sonya_Mehta$ mpirun -n 4 ./a.out
Enter the upper limit:
100
Number of prime numbers in [1, 100] are 25
sonyamehta_24@sonyamehta24-HP-Pavillon-Gaming-Laptop-15-dk0xxx:~/sem5/cs359/Assignment3_190001058_Sonya_Mehta$ mpirun -n 4 ./a.out
Enter the upper limit: 25
Number of prime numbers in [1, 25] are 8
sonyamehta_24@sonyamehta24-HP-Pavillon-Gaming-Laptop-15-dk0xxx:~/sem5/cs359/Assignment3_190001058_Sonya_Mehta$ mpirun -n 4 ./a.out
sonyamehta_24@sonyamehta24-HP-Pavillon-Gaming-Laptop-15-dk0xxx:~/sem5/cs359/Assignment3_190001058_Sonya_Mehta$ mpirun -n 4 ./a.out
Enter the upper limit: 45
Number of prime numbers in [1, 45] are 15
sonyamehta_24@sonyamehta24-HP-Pavillon-Gaming-Laptop-15-dk0xxx:~/sem5/cs359/Assignment3_190001058_Sonya_Mehta$
```