CMPE300 - Analysis of Algorithms
Spring 2020
MPI Programming Project

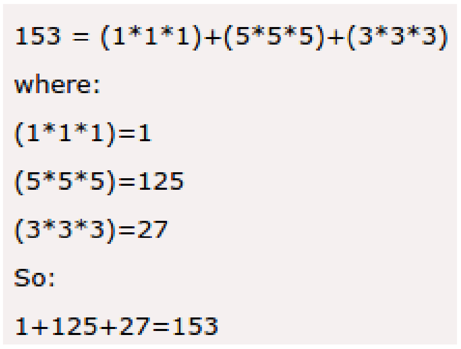
due: 05.06.2020

1 Introduction

In this project, you are going to experience parallel programming with C/C++/Python using MPI library. You will implement a parallel algorithm for finding Armstrong numbers.

An Armstrong number is a number that is equal to the sum of its own digits each raised to the power of the number of digits. For example 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 153, 370, 371, 407, 1634, 8208, 9474, 54748, 92727, 93084, 548834, ... are Armstrong numbers.

Below is the demonstration of why 153 (number of digits = 3) and 1634 (number of digits = 4) are Armstrong numbers.



```
153 = (1*1*1)+(5*5*5)+(3*3*3)
where:
(1*1*1)=1
(5*5*5)=125
(3*3*3)=27
So:
1+125+27=153
```

1634 = (1*1*1*1) + (6*6*6*6) + (3*3*3*3) + (4*4*4*4)

1634 = 1 + 1296 + 81 + 256

2 Finding Armstrong Numbers

In this project, our aim is to implement finding Armstrong numbers using MPI. This way, we will be able to divide the task of finding Armstrong numbers, into several processes, and observe and practice parallel programming.

Write a program for n+1 processors in the master-worker programming style (1 master, n workers).

The master process will create an array of numbers, from 1 to A. You will divide the array and send A/n (random) numbers to each worker, which will check for Armstrong numbers in their partition. When they have finished processing, they will send their results (the Armstrong numbers in their partition) to the master process, which will sort and print these results (the Armstrong numbers between 1 and A) into armstrong.txt.

Suppose A = 20, and we work with 5 processors (n=4):

- **Master Process:** Will create an array of numbers from 1 to 20
  array = {1,2,3,4,5,.......19,20}
  array should be divided into n=4 parts randomly (all 4 partitions should have 5 random numbers. When the partitions are combined, all numbers between 1..A should be covered)
- **Worker Process 1:** Will receive the first partition of the array (with A/n = 20/4 = 5 numbers)
  array1 = {5, 12, 7, 1, 6}
  Will find the Armstrong numbers in its partition
  Will send the found Armstrong numbers in its partition to Master process
- **Worker Process 2:** Will receive the second partition of the array (with A/n = 20/4 = 5 numbers)
  array2 = {3, 18, 19, 2, 10}
  Will find the Armstrong numbers in its partition
  Will send the found Armstrong numbers in its partition to Master process
- **Worker Process 3:** *array3* = {16, 8, 11, 20, 14}
  ...
- **Worker Process 4:** *array4* = {15, 9, 13, 4, 17}
  ...

After completion of finding the Armstrong numbers in each partition, each process should calculate the sum of their Armstrong numbers, and print the results in the console, as in the example below:

Sum of Armstrong numbers in Process 1 = ...

Sum of Armstrong numbers in Process 2 = ...
Sum of Armstrong numbers in Process 3 = ...
Sum of Armstrong numbers in Process 4 = ...

After calculation of the sums by each process, each process should send their sum to the next process (1 to 2, 2 to 3, and so on), each of which will add the received sum to their own sum, resulting in a collective sum of Armstrong numbers. The last worker (Process 4 in this case) should send the final sum to master, which will print the result:

MASTER: Sum of all Armstrong numbers = ...

You are expected to examine the behaviour of the system with the following configurations.

1. **A = 1000, 5 processors (n = 4)**
2. **A = 10000, 5 processors (n = 4)**
3. **A = 100000, 5 processors (n = 4)**
4. **A = 1000, 11 processors (n = 10)**
5. **A = 10000, 11 processors (n = 10)**
6. **A = 100000, 11 processors (n = 10)**

In each configuration, examine the time elapsed and the number of tasks each process completed.

3 Submission

1. The deadline for submitting the project is **05.06.2020 - 23:59**. The deadline is strict.
2. This is an individual project. Your code should be original. Any similarity between submitted projects or to a source from the web will be accepted as cheating.
3. Your code should be suficiently commented and indented.
4. You should write a header comment in the source code file and list the following information.
   /*
   Student Name: Ali Veli

Student Number: 2008123456
Compile Status: Compiling/Not Compiling
Program Status: Working/Not Working
Notes: Anything you want to say about your code that will be helpful in the grading process.
/

5. You must prepare a document about the project, which is an important part of the project. You should talk about your approach in solving the problem. Follow the guidelines given in the ”Programming Project Documentation” link on `https://www.cmpe.boun.edu.tr/~gungort/informationstudents.htm`.

6. Submit your project and document as a compressed archive (zip or rar) to Moodle. Your archive file should be named in the following format: ”name surname.ext”. You don’t need to submit any hard copies.

7. If you have any further questions, send an e-mail to the course assistant: *zeynep.yirmibesoglu@boun.edu.tr*