Assignment 1: MPI Programming

Distributed Systems

January 17, 2022
Due: 11:55 PM, January 27, 2022

1 Introduction

This assignment deals with parallel programming using the message passing model. In this assignment you will construct parallel solutions to certain problems and implement them using the Message Passing Interface: MPI in any programming language of your choice (C/C++ is recommended).

- Documentation: `https://www.rookiehpc.com/mpi/docs/index.php`
- References: `https://pages.tacc.utexas.edu/~eijkhout/pcse/html/index.html`, `http://condor.cc.ku.edu/~grobe/docs/intro-MPI-C.shtml`
- Tutorials: `https://mpitutorial.com/tutorials/`, `https://www.rookiehpc.com/mpi/exercises/index.php`
- All your programs should run for minimum 1 and maximum 11 processes.
- For those who are not able to run their programs with 11 or less process use following command (for C++):
  mpirun -np 11 –use-hwthread-cpus –oversubscribe ./a.out

2 Problems

You are supposed to use the given template for implementing each of the problems. Additionally, your program will be given two arguments, path to an input file and an output file from which your program will obtain the input and output the result into respectively, like the following (for C++):
mpirun -np 11 ./a.out <input_file> <output_file>.

2.1 Problem 1 (10 points)

Write a program to check if a number is prime.

Input: First line will contain the integer N.

Output: Print YES if the integer is prime, else print NO.

Constraints: $1 <= N <= 10^9$

Example:

Sample Input:
11

Sample Output:
YES

2.2 Problem 2 (40 points)

Given an undirected binary weighted graph G, find the number of cliques (of all possible weights) in the graph with sizes 3 and 4 (Refer to the sample input and output for clear understanding).

Input: The first line contains two integers N and E, representing the number of vertices and the number of edges in the graph, respectively.
Following E lines contain 3 integers (2 vertices and a weight) each, representing an edge between these two vertices with a binary weight (0 or 1).
Note :
2 cliques are different if they have at least 1 vertex which is not common. There will be only 1 edge between any 2 vertices.

Output: Print a total of 11 lines. Each line should contain 3 space separated integers - ⟨size of clique⟩ ⟨weight of clique⟩ ⟨number of such cliques⟩

Constraints: $2 <= N <= 10^2$, $1 <= E <= 10^3$

Example:

Sample Input:
7 11
1 2 0
2 3 0
2 4 1
3 4 0
3 5 1
1 5 1
1 6 0
6 7 1
7 5 1
7 1 0
6 5 1
1 3 0
2 5 1

Sample Output:
3 0 1
3 1 2
3 2 5
3 3 1
4 0 0
4 1 0
4 2 0
4 3 1
4 4 1
4 5 0
4 6 0

2.3 Problem 3 (50 points)

Let A be a full rank square matrix of N rows and N columns. Write a program to find the solution to the set of linear equations $Ax = b$ using the **Gaussian Elimination algorithm followed by back substitution**. You can assume that A is partitioned into rows and the partitions are stored across the processes.

Input: First line will contain N, the size of the square matrix. The next N rows will contain N + 1 floating-point values each, where the value in the i^{th} row at the j^{th} place is the A[i][j] value when $j \leq N$ and b[i] if j is N + 1.

Output: A vector x which is the solution to $Ax = b$ containing real numbers having absolute error less than 1e-6.

Constraints: $1 <= N <= 10^2$

Example:

Sample Input:
3
1.0 5.0 11.0 -1.0
2.0 9.0 29.0 -2.0
-3.0 10.0 1.0 3.0

Sample Output:
-1.0 0 0

3 Submission Instructions

Your submission is expected to be a **<RollNumber>.zip** file containing a directory with the same name as your roll number that holds the following files:

- A program file for each of the mentioned problems with the name: **<RollNumber>_<ProblemNumber>.cpp** (file extension depends on your chosen language)
- A brief report describing and analyzing your solution as: **README.md**

NOTE : It is recommended to do the assignment in C/C++. We won't answer language specific doubts for any language other than C/C++. If you choose to use a language **other than C/C++**, then you should also submit a file along with their codes which includes the following:

- How to install all the dependencies to execute your code
- How to execute your code
- Any warnings that we have to ignore

The TAs should be able to execute your code on their systems for evaluation. Modify the template provided below in the language of your choice.

Example structure

```
2018101077
├─2018101077_1.cpp
├─2018101077_2.cpp
├─2018101077_3.cpp
└─README.md
```

There will be automated evaluation. Therefore, please strictly adhere to the expected format and handle appropriate edge cases, or you will be penalized.

NOTE: Strict actions would be taken against anyone found involved in any kind of plagiarism either from the internet or from other students. If we find any of the codes implementing the question in serial manner or using other algorithms than mentioned in the question, then it will result in serious penalties.

4 Appendix: Template for C++

```cpp
#include <iostream.h>
#include <mpi.h>
```

```
using namespace std;

int main( int argc, char **argv ) {
    int rank, numprocs;

    // initiate MPI
    MPI_Init( &argc, &argv );

    // get size of the current communicator
    MPI_Comm_size( MPI_COMM_WORLD, &numprocs );

    // get current process rank
    MPI_Comm_rank( MPI_COMM_WORLD, &rank );

    /*synchronize all processes*/
    MPI_Barrier( MPI_COMM_WORLD );
    double start_time = MPI_Wtime();

    // enter your code here

    MPI_Barrier( MPI_COMM_WORLD );
    double end_time = MPI_Wtime() - start_time;
    double maxTime;
    // get max program run time for all processes
    MPI_Reduce( &end_time, &maxTime, 1, MPI_DOUBLE,
      MPI_MAX, 0, MPI_COMM_WORLD );
    if ( rank == 0 ) {
        cout<<"Total time (s): "<<maxTime<<"\n";
    }

    // shut down MPI and close
    MPI_Finalize();
    return 0;
}
```