Assignment-1 : MPI (Message Passing Interface)

Due : 26th January 2024 (11:55pm)

11th January 2024

1 Introduction

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 strongly recommended owing to the amount of documentation available online)

- You may only use MPI library routines for communicating and coordinating between the MPI processes. You cannot use any form of shared-memory parallelism. The idea is to develop a program that could be deployed on a large message-passing system
- We'll be doing the final evaluation on Linux, so make sure you don't use platform-specific libraries in your solutions. Please use standard libraries only
- The code you submit will be run alongside a Profiler to check if you are parallelizing the code and not submitting a sequential solution.
- Your programs should execute with any number of processes between 1 and 12. For those who are not able to run with 12 processes, use the following command (for C++): `mpirun -np 12 --use-hwthread-cpus --oversubscribe ./a.out`
- Attempt any 1 from **Q1A** and **Q1B**. **Q2** and **Q3** are mandatory

2 Question - 1A (20 points)

The Multibrot set is a fractal that is defined as the set of all complex numbers c for which the sequence z does not diverge to infinity where z is defined as the sequence of complex numbers $\{z_0, z_1, z_2, ...\}$ such that $z_0 = 0$ and $z_{i+1} = z_i^d + c$

The magnitude of z is its distance from the origin. If the magnitude of z ever becomes greater than 2 its subsequent values will grow without bound and we know that c is not a point in the Multibrot Set. If we iterate K times and find

that the magnitude of z_K is still less than or equal to 2, we can conclude c is a point in the Multibrot set.

You have to write a program using MPI that computes the Multibrot set after K iterations for $N * M$ points spaced uniformly in the region of the complex plane bounded by $-1.5 - \iota, -1.5 + \iota, 1 + \iota, 1 - \iota$ in a distributed manner

Input

The input contains only one line with 4 integers - the number of points along x axis: N, the number of points along y axis: M, the exponent: D, and the number of iterations: K.

Output

You are expected to output a Grid of $M \times N$, ones and zeros with one indicating the number is in the Multibrot set and zero indicating if it s not after K iterations.

Constraints

$N \geq 2$ and $M \geq 2$
$N * M * K * D \leq 1000000$

Sample Input

```
16 16 2 1000
```

Sample Output

```
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0
0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0
0 0 0 1 0 1 1 1 1 1 1 1 1 0 0 0 0 0
0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
0 0 0 1 0 1 1 1 1 1 1 1 1 0 0 0 0 0
0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0
0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
```

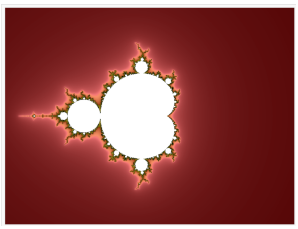

Figure 1: d=2

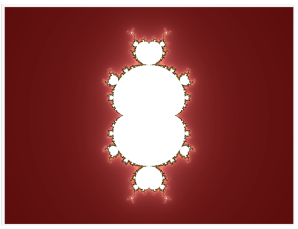

Figure 2: d=3

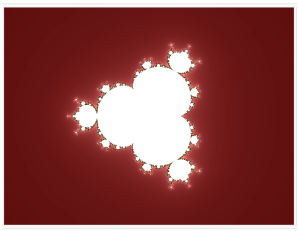

Figure 3: d=4

3 Question - 1B (20 points)

On an $N \times N$ chessboard, you want to place N queens such that no queen attacks another. A queen attacks any other queen that is in the same row, column or diagonal. Given an integer N, write an MPI program that counts the number of ways to place N queens on the $N \times N$ chessboard.

Input

The program takes a single constant N (the size of the chessboard) as input.

Output

The program should print the number of ways to place the N queens on the chessboard.

Constraints

$1 \leq N \leq 12$

Sample Input

```
4
```

Sample Output

```
2
```

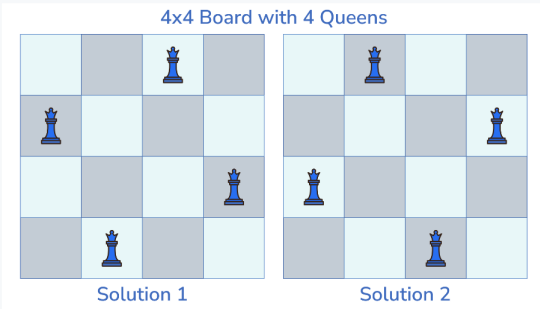

Figure 4: 2 solutions

4 Question - 2 (40 points)

The Floyd-Warshall algorithm is used to find the shortest paths between all pairs of nodes in a weighted graph. Write a parallel program to calculate the final adjacency matrix which represents shortest distance between all pairs.

Input

The first line contains a single integer N, the number of vertices in the graph. Each of the next N lines contain N space separated integers. These represent the $N \times N$ adjacency matrix of the graph.

$A[i][j]$ = weight of edge from node i to node j. (It will be -1 if there exists no edge from node i to node j)

Output

The program should output the final $N \times N$ adjacency matrix which represents shortest distance between all pairs. Output should contain N lines and each line should have N space separated integers such that $A[i][j]$ is the shortest distance from node i to node j.

Constraints

$1 <= N <= 1000$
$1 <= A[i][j] < 10^9$

5 Question - 3 (40 points)

The Game of Life is a cellular automaton devised by mathematician John Conway. It consists of a grid of cells, each of which can be in one of two states: alive or dead. The state of each cell evolves over time according to the following set of rules based on the neighboring cells.

1. Any live cell with fewer than two live neighbours dies, as if by underpopulation.
2. Any live cell with two or three live neighbours lives on to the next generation.
3. Any live cell with more than three live neighbours dies, as if by overpopulation.
4. Any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.

Refer this link for more information about the Game of Life or to simulate it on any initial configuration. You will be given an initial configuration of a grid of $N \times M$ cells. Write a program using MPI that simulates T generations in the Game of Life using this initial configuration.

Input

The first line of input contains three space separated integers N, M, T representing the size of the grid and the number of generations to simulate.

The next N lines would contain M space separated integers on each line, representing a grid of N rows and M columns. A value of 0 in this grid represents a "dead" cell and a value of 1 represents an "alive" cell.

Output

Print an $N \times M$ grid of space separated integers that represents the state of the game after T generations. Dead cells would be represented by 0 and alive cells would be represented by 1.

Constraints

$N >= 2, M >= 2$
$N * M * T <= 1000000$

Sample Input 1

```
10 10 7
0 0 1 0 0 0 0 0 0 0
1 0 1 0 0 0 0 0 0 0
0 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
```

Sample Output 1

```
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 1 0 1 0 0 0 0 0
0 0 0 1 1 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
```

Sample Input 2

```
10 10 50
0 0 1 0 0 0 0 0 0 0
1 0 1 0 0 0 0 0 0 0
0 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
```

Sample Output 2

```
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 1 1
```

6 Submission

We'll be automating the checking, so make sure you deal with all the edge cases and comply with all the Input-Output specifications and the directory structure given in the submission guidelines below. Not following the given requirements would result in a heavy penalty.

Your submission is expected to be a ⟨**RollNumber**⟩**.tar.gz** file containing a directory with the same name as your roll number that holds the following files:

- A directory for each of the mentioned problems with the name: ⟨**ProblemNumber**⟩ containing one source file named ⟨**ProblemNumber**⟩ (e.g. 1.cpp, 1.py)
- You are also required to submit a report, named README.md in the root directory of your project, with the following data for every problem you attempt:
  - The total time complexity of your approach
  - The total message complexity of your approach
  - The space requirements of your solution
  - The performance scaling as you went from 1 to 12 processes (use a large enough test case for this)

Example structure

```
2020102029
├── 1A
│   └── 1A.cpp
├── 2
│   └── 2.cpp
├── 3
│   └── 3.cpp
└── README.md
```