Assignment 3: A Simple Matrix Problem using CUDA

COL 380

March 2025

Due Dates

- Due Date 1: 23:59, March 26, 2025
- Due Date 2: 23:59, April 1, 2025

1 Introduction

In this assignment, you are given N matrices of positive integers. For each matrix, you need to rearrange its elements so that they satisfy a given ordering property.

We use the notation that an $m \times n$ matrix A denoted by $A_{m,n}$ has m rows and n columns, and we use 1-based indexing. Let us define a **sub-matrix**.

Given an $m \times n$ matrix A, the **sub-matrix** $S_{i,j}$ corresponding to an index (i, j) (where $1 \leq i \leq m$ and $1 \leq j \leq n$) is the matrix that consists of all elements of matrix A in the rectangular region from the top-left corner $(1, 1)$ to the position (i, j), while preserving their original order.

Formally, for a given (i, j) and an $m \times n$ matrix A, the **sub-matrix** $S_{i,j}$ is defined as:

$$S_{i,j} = \begin{bmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,j} \\ A_{2,1} & A_{2,2} & \dots & A_{2,j} \\ \vdots & \vdots & \ddots & \vdots \\ A_{i,1} & A_{i,2} & \dots & A_{i,j} \end{bmatrix}$$

where $A_{r,c}$ represents the element at row r and column c of A, and $1 \leq i \leq m$ and $1 \leq j \leq n$)

2 Problem Statement

Given a matrix A, our goal is to modify it by rearranging its elements so that the modified matrix satisfies the following ordering property:

In the modified matrix, for every index (r, c) (where r represents the row and c represents the column), the element at index (r, c) must be the maximum in the sub-matrix $S_{r,c}$, which is defined as the rectangular region that includes all elements from the top-left corner $(1, 1)$ to the position (r, c).

Mathematically, the modified matrix must satisfy the following condition for every index (r, c) of matrix A:

$$A_{r,c} \geq A_{p,q}, \quad \forall\, 1 \leq p \leq r,\, 1 \leq q \leq c.$$

This means that each element $A_{r,c}$ must be the maximum among all elements in the sub-matrix $S_{r,c}$.

For the given N matrices, your task is to modify each of the matrices so that each of them satisfies the property stated above.

2.1 An Example

```
2 # Number of matrices

4 3 # Matrix A: 4 x 3
1  3  2
4  6  5
7  9  8
9  7  1


3 4 # Matrix B: 3 x 4
5  1   4   2
9  3   7   6
8  12  11  10

After modification:
# Matrix A'
1  1  2
3  4  5
6  7  7
8  9  9

# Matrix B'
1  4  5   8
2  7  9   11
6  8  10  12
```

Explanation: In the modified matrices A' and B', for every index (r, c), the element at index (r,c) is the greatest among all elements in the corresponding sub-matrices (the top-left rectangle from $(1, 1)$ to (r, c)). This ensures that the given property is satisfied.

2.2 Task

A simple strategy to solve this problem for a given matrix is to create a frequency array for elements ranging from 0 to the greatest element in the array. Arrange the frequency array in increasing order. Then, create a prefix sum array for the frequency array created. For all the elements with non-zero frequency, use the prefix sum array to find their position in the output matrix. You need to implement an optimized parallel version of this strategy using CUDA and complete the following function:

```
vector<vector<vector<int>>> modify
(vector<vector<vector<int>>> &matrices, vector<int>& ranges);
```

The inputs given to the function are the array of N matrices and the array of the upper bound of the range of elements for each matrix. For $N > 1$, you should modify multiple input matrices in parallel. You must return the output matrices in the same order as the input. For each of the given matrices, the dimensions of the original and modified matrix must be the same, and the multi-set of elements of the original and the modified matrix must be the same.

2.2.1 Input Constraints

- Let M be the array of matrices, and R be the array that contains the upper bound of ranges for the given matrices, with $n = |M|$ $(|M| = |R|)$.
- For each matrix $m \in M$ of dimensions $r \times c$
$$1 \leq r \times c \leq 10^9$$
- For $m = M[k]$ and $r = R[k]$ $\qquad \forall k \in [1, n]$
$$0 < m[i][j] \leq r, \quad \forall i, j \quad (1 \leq i \leq r, \quad 1 \leq j \leq c)$$
- The total number of elements across all the matrices in the input does not exceed 10^9, i.e., $\sum_{m \in M} |m_{r \times c}| \leq 10^9$
- The sum of all ranges in the input does not exceed 10^9, i.e., $\sum_{r \in R} r \leq 10^9$.

3 Starter code

The starter code is provided **here**. It consists of a basic `main.cpp` file that generates the input instances, a header file `modify.cuh`, and a simple checker code. Note that none of the files are final files that will be used to evaluate your submission. You need to implement the function modify(), whose declaration is present in file `modify.cuh` in a file named `modify.cu`.

4 Report

Your report should list your implementation decisions and a discussion on performance in a PDF file named `readme.pdf`. You also need to submit a CSV file named `data.csv` showing your timings for the `modify()` function across different matrix sizes and on different numbers of matrices. Also, explain how you handled various inputs (as listed below).

Format for Performance Analysis: The performance table records timings for the modify() function (in milliseconds up to 2 decimal places) for various kinds of inputs.

$r \times c$	$\lvert M\rvert$	**Max. Value** ($\max_{r \in R} r$)			
		1024	4096	10^5	10^8
$10^3 \times 10^3$	1				
$10^3 \times 10^3$	10				
$10^4 \times 10^4$	1				
$10^4 \times 10^4$	10				
$10^4 \times 10^5$	1				

Here $\lvert M\rvert$ is the number of matrices in the input, and R denotes the array *ranges*, which is given as input. You must maintain the same order of rows and columns in your CSV file called `data.csv`.

5 Execution Instructions

5.1 Module Requirements

Before running the code, you must load **ONLY** the following modules:

- module load compiler/gcc/9.1.0
- module load suite/nvidia-hpc-sdk/20.7/cuda11.0

Run `module purge` before loading any module.

5.2 Compiling and Running the Code

To compile:

```
nvcc main.cpp check.cpp modify.cu -Xcompiler -fopenmp -o check
```

Note: For Haswell (*khas*) nodes with NVIDIA K40m GPUs use the following:

```
nvcc main.cpp check.cpp modify.cu -arch=sm_35 -Xcompiler -fopenmp -o check
```

To run:

```
./check
```

6 Submission Instructions

Submit the following files on Gradescope:

```
|-- modify.cu
|-- readme.pdf
|-- data.csv
```

Note that if you submit more or less than these three files, you will get a 0. Please stick to the given file names.

6.1 Guidelines

- You are not allowed to use external libraries (for ex. Thrust).
- You may refer `Programming Massively Parallel Processors by D. Kirk and W. Hwu` for learning algorithms like prefix sum.
- Use the gen_matrix() function defined in `check.cpp` file to generate matrices. Do not modify the gen_matrix() function.
- You can assume that each matrix would be filled in row-major order with the elements distributed uniformly within the input range.
- You can use std::chrono for timing analysis.

7 Evaluation Criteria

- **Correctness**: The modified matrices should comply with the properties stated.
- **Performance**: Efficiency of the parallel implementation for different kinds of inputs.
- **Report**: Clarity and depth of performance analysis.