Assignment 3

1 Overview

This is the second MPI programming project. You will write a parallel program that expects two inputs: a number of radians that represents the angle that the sun makes with the horizon at a fixed time of day when it is shining from the west of the given region, and the name of a file that contains a model of a topographical map containing elevation information. The program will determine locations on the map that are in the shade.

2 Project Description

A large, rectangular geographical region is modeled as an $n \times m$ grid of square prisms (i.e., boxes) of varying heights whose bases are fixed-size squares of dimensions 1×1 meter each. The grid is represented by an $n \times m$ matrix whose elements represent the heights of these square-based prisms in multiples of 1 meter. For example, if the matrix element $M[i, j]$ has the value 4, then $M[i, j]$ represents a square prism that measures $1 \times 1 \times 4$ meters high. Heights are non-negative, real-valued numbers. The grid is oriented so that row 0 is in the north, and column 0 is in the west. Henceforth we call these prisms ***boxes***.

The sun is assumed to shine from the west at an angle of θ radians, $0 < \theta \leq \pi/2$. When $\theta = \pi/2$ it is directly overhead, so every box is in full sun. If the angle θ is greater than $\pi/2$, the sun is shining from the east, which, in terms of how it shades the grid, is equivalent to saying that the sun makes an angle of $\pi - \theta$ with the east side of the grid. Or think of it as making an angle of $\pi - \theta$ with the grid obtained by reversing the grid's columns, i.e., swapping columns 0 and $n - 1$, columns 1 and $n - 2$, and so on.

The matrix is in a binary file whose name will be supplied as the second command line argument. The first command line argument is a fixed decimal number representing the angle of the sun with respect to the west side of the grid, expressed in radians. It is supposed to be in the range $(0, \pi)$, i.e., the slope cannot be 0 or π. The binary file's first number is an integer whose value is n and its second number is an integer whose value is m. After these two integers, it contains $n \times m$ non-negative doubles that represent the matrix in row major order. If either argument is missing or if the slope is outside of the valid range, the program should report the error with a suitable usage message and then exit.

The program determines which box's top surfaces are in the sun or shade. This is a binary function - ***if at least half of the top of the box is in the shade, then the entire top is considered to be shaded, otherwise it is not shaded***. The program's output will be an $n \times m$ matrix of 0's and 1's in which a 1 represents a box in the sun and a 0 represents a box in the shade. This matrix will be written to standard output as a plain text file in which each box's value will be separated from adjacent boxes' values by a single blank in the output, e.g.

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

A box in the matrix will be, by definition, entirely in the shade if there is an obstruction to its west that prevents the sun from reaching it at the given angle. This implies that a box (i, j) is in the shade if there is a box $(i, j - k)$ where $0 \leq j - k$ and the box $(i, j - k)$ has an elevation so high that it blocks the light from the sun. If we let $h(i, j)$ denote the height of box (i, j). then, if $\theta = \pi/4$ for example, then box (i, j) would be in shade if $h(i, j - 1) \geq h(i, j) + 0.5$, as this implies that the box in position $(i, j - 1)$ is 0.5 meters higher than the box at (i, j) and a 45 degree angle has a slope of 1, which makes at least half of the top of (i, j) in shade. Your first task is to figure out the relationship between the angle and the heights of the boxes that

would put a box into the shade. When $\theta \leq \pi/2$, all boxes along the west border of the grid will be in sun, as there are no obstructions to their west, and the remaining boxes are in sun or shade depending on the topography and the angle of the sun. A symmetric statement about the east border is true when the angle is greater than $\pi/2$.

You are to write a parallel program using MPI that solves this problem. You must follow the steps of Foster's task-channel methodology to determine a decomposition of the problem into parallel processes. Your program documentation should explain how you arrived at your decomposition.

Hints.

- You must decide how to agglomerate. The wrong method will make your solution harder.
- The structure of your main program will be similar to those we have discussed in class. However you agglomerate, the pieces of the matrix need to be read by a single process and distributed to others.
- Design your algorithm before coding. Figure out how to determine what is in shade and what is in sun.
- The input file is in binary format. You should use the matrix utilities in the `demos/common` to create matrices and convert them to binary format. There is a program named `matrix2binary.c` that can be given a matrix specified as plain text on standard input and which outputs a binary file that is suitable for input to this program.

3 Grading Rubric

The program will be graded based on the following rubric:

- The program must compile and run on any `cslab` host. If it does not compile and link on any `cslab` host, it loses loses 80%.
- **Correctness** (60%)
  - The program should do exactly what is described above. Incorrect output, incorrectly formatted output, missing output, or output containing other characters are all errors.
  - The program should produce the same output no matter how many processes it is given, except for the elapsed time.
  - It should handle errors correctly.
- **Performance** (15%)

  There are various ways in which you can make this program more or less efficient in both time and work; the solutions that waste no cycles will be rewarded. The faster the program runs, the higher the grade. But too much redundancy will be penalized. The total work is a factor in the performance.
- **Design and clarity** (10%)

  Is the program organized clearly? Are functions and variables designed in an appropriate way? Are the tasks divided appropriately? If not, the program will lose points.
- **Compliance with the Programming Rules** (15%)

  Are all of the rules stated in that document observed? Programs that violate them will lose points accordingly.

4 Submitting the Homework

This is due by the end of the day (i.e. 11:59PM, EST) on November 4, 2019. Follow these instructions precisely to submit it.

Your program should be a single source code file, written in C. It must have a `.c` suffix. It does not matter what base name you give it because the `submithwk` program will change its base name anyway to the form `username_hwk3.c` (or more accurately, `username_hwk3.`*XXX*, where *XXX* is the suffix you gave it.)

Assuming this file is in your current working directory and is named `myhwk3.c` you submit by entering the command

```
submithwk_cs49365 -t  3  myhwk3.c
```

The `-t` tells the command it is a plain text file. The program will copy your file into the `hwk3` sub-directory

```
/data/biocs/b/student.accounts/cs493.65/hwks/hwk3/
```

and if it is successful, it will display the message, "`File username_hwk3.c successfully submitted.`"

You can only run this command on a cslab host. You cannot run it on eniac, so remember to ssh into a cslab host before doing this!

You will not be able to read this file, nor will anyone else except for me. But you can double-check that the command succeeded by typing the command

```
ls -l /data/biocs/b/student.accounts/cs493.65/hwks/hwk3/
```

and making sure you see a non-empty file there.

If you put a solution there and then decide to change it before the deadline, just replace it by the new version. Once the deadline has passed, you cannot do this. I will grade whatever version is there at the end of the day on the due date. You cannot resubmit the program after the due date.

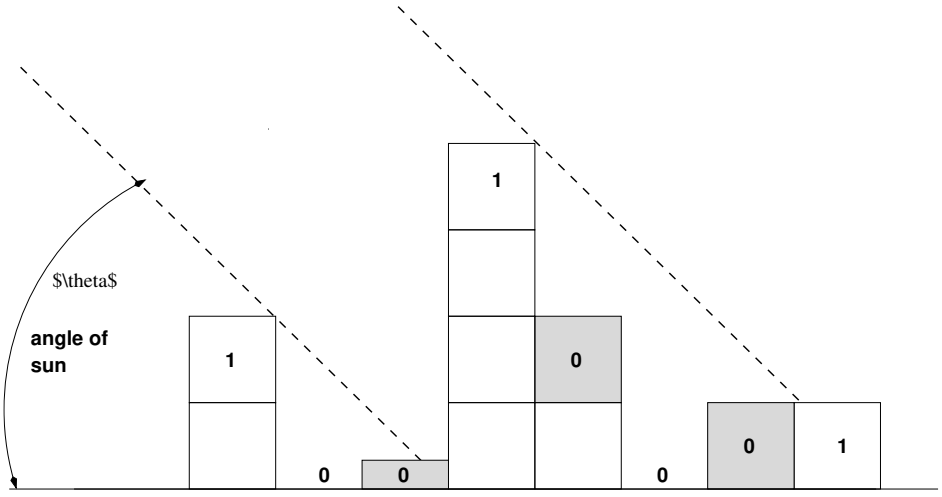

Figure 1: Example of how the sun shades prisms on the grid. The boxes labeled with 0's are in shade; those with 1's are in sun. This illustrates how a box that is more than 50% shaded is marked as fully shaded.