Problem 1. Let $G$ be a directed graph with $n$ vertices. For simplicity we identify the vertex set to the set of positive integers $\{1, 2, \ldots, n\}$. To each couple $(i, j)$, with $1 \leq i, j \leq n$, we associate a weight $w_{i,j}$ such that:

(i) $w_{i,j}$ is a non-negative integer if and only if $(i, j)$ is an arc in $G$,

(ii) $w_{i,j}$ is $+\infty$ if and only if $(i, j)$ is not an arc in $G$.

We assume $w_{i,i} = 0$ for all $1 \leq i \leq n$. If $x_1, x_2, \ldots, x_m$ are $m \geq 2$ vertices of $G$ such that $(x_1, x_2), (x_2, x_3), \ldots, (x_{m-1}, x_m)$ are all arcs of $G$, we say that $p = (x_1, x_2, \ldots, x_m)$ is a path in $G$ from $x_1$ to $x_m$; moreover the weight of $p$ is denoted by $w(p)$ and defined by

$$w(p) = w_{x_1,x_2} + w_{x_2,x_3} + \cdots + w_{x_{m-1},x_m}.$$

For each couple $(i, j)$ which is not an arc in $G$ it is natural to ask whether

(1) there is a path in $G$ from $i$ to $j$, and

(2) if such path exists, then compute the minimal weight of such a path.

This question is often referred as ASAP for All-Pair Shortest Paths. The celebrated Floyd-Warshall algorithm solves ASAP by computing a matrix $\text{path}$ as follows:

```
for k = 1 to n
    for i = 1 to n
        for j = 1 to n
            \text{path}[i][j] = \min ( \text{path}[i][j], \text{path}[i][k]+\text{path}[k][j] );
```

after initializing $\text{path}[i][j]$ to $w_{i,j}$. For more details, please refer to the Wikipedia page of the Floyd-Warshall algorithm.

One way to obtain an efficient multi-threaded algorithm for ASAP is to apply a divide and conquer approach. To this end we view $(w_{i,j})$ as an $n \times n$-matrix, denoted by $W$. We also view the targeted result, namely the values $(\text{path}[i][j])$ as an $n \times n$-matrix, denoted by $\overline{W}$.

Before stating the divide and conquer formulation, we introduce a few notations. Let $X, Y$ be square matrices (of the same order) whose entries are non-negative integers or $+\infty$. We denote by

- $XY$ the min-plus product of $X$ by $Y$ (obtained from the usual matrix multiplication by replacing $+$ (resp. $\times$) by $\min$ (resp. $+$))
• $X \lor Y = \min(X, Y)$ the element-wise minimum of the two matrices $X$ and $Y$.

We are ready to state the divide and conquer formulation. If we decompose $W$ into four $n/2 \times n/2$-blocks, namely

$$W = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

then we have

$$\overline{W} = \begin{pmatrix} E & EB\overline{D} \\ G & \overline{D} \lor GB\overline{D} \end{pmatrix}$$

where we have $E = A \lor BDC$ and $G = DCE$. We shall admit that these formulas are correct (even though proving them is not that hard).

**Question 1.** [5 points] Propose an algorithm for computing $\overline{W}$ in the fork-join parallelism model.

**Question 2.** [5 points] Analyze the work, the span and the parallelism of your algorithm.

There exist alternative algorithms for the ASAP problem which rely on the min-plus multiplication. A simple one is based on the observation that $W = W^n$ (and in fact $W^{n-1}$) where $W^n$ is the $n$-th power of $W$ computed for min-plus multiplication using repeated squaring.

**Question 3.** [10 points] Propose such an algorithm. You are welcome to use the literature or simply to use the one suggested above.

**Question 4.** [10 points] Analyze the work, the span and the parallelism of this third algorithm.

The goal of the rest of this problem is to realize a CUDA implementation of an algorithm solving the ASAP problem described in Problem 1.

**Question 5.** [5 points] Among the ASAP algorithms discussed in Problem 1, explain which one is better suited for a CUDA implementation.

**Question 6.** [30 points] Realize a CUDA kernel implementing the min-plus multiplication.

**Question 7.** [25 points] Realize a C/C++ implementation of this algorithm, based on this CUDA kernel. Provide experimental data and performance analysis together with comments.

**Submission instructions.**

**Format:** The answers to the problem questions should be typed.
• If these are programs, input test files and a Makefile (for compiling and running) are required. Please provide a README describing how to compile and test your code. Please submit source code only!

• If these are algorithms or complexity analyzes, \LaTeX is highly recommended; in any case a PDF file must gather all these answers.

All the files should be archived using the UNIX tar command.

Submission: The assignment should be returned to the instructor by email.

Collaboration. You are expected to do this assignment on your own without assistance from anyone else in the class. However, you can use literature and if you do so, briefly list your references in the assignment. Be careful! You might find on the web solutions to our problems which are not appropriate. For instance, because the parallelism model is different. So please, avoid those traps and work out the solutions by yourself. You should not hesitate to contact me if you have any questions regarding this assignment. I will be more than happy to help.

Marking. This assignment will be marked out of 100. A 10 % bonus will be given if your paper is clearly organized, the answers are precise and concise, the typography and the language are in good order. Messy assignments (unclear statements, lack of correctness in the reasoning, many typographical and language mistakes) may give rise to a 10 % malus.