CSE Ph.D. Qualifying Exam, Fall 2025
High-Performance Computing

Instructions:

Please answer three of the following four questions. All questions are graded on a scale of 10. If you answer all four, all answers will be graded and the three lowest scores will be used in computing your total. **This exam is closed-book, closed-notes.**

Questions:

1. **Interconnection Network.** A new static interconnection network is defined as follows: Let the number of processors $P = 2^d$, and the rank of each processor is denoted by a d-bit binary string just as in the hypercube. However, two processors are connected if and only if the bit representations of their ranks differ in exactly k bits.

   (a) (10%) Show that for even k, the network is disconnected.

   (b) (50%) Let k be odd and assume d is sufficiently larger than k. Find a path between two processors whose ranks differ in one bit.

   (c) (10%) Estimate the network diameter for odd k, assuming d is sufficiently larger than k.

   (d) (30%) Argue whether this network is superior or inferior to the hypercube.

2. **Analyzing sparse matrix-sparse matrix multiplication (SpGEMM).** In this problem, we will analyze distributed-memory parallel versions of *Gustavson's row-wise algorithm* for multiplying two sparse matrices A and B to produce some output matrix C (i.e., $C = AB$) (see Algorithm 1 and Figure 1). At a high level, Gustavson's row-wise algorithm iterates through the rows of the matrix, performing only the necessary arithmetic operations where both operands are nonzero.

   In this problem, we will consider the case where the input sparse matrices A and B are square, i.e., where both have n rows and columns. Furthermore, we will assume that A and B each have have $c > 0$ nonzeroes per row/column, uniformly distributed across the row and column. That is, the number of nonzeroes in A and B are as follows: $nnz(A) = nnz(B) = cn$.

   In this problem, we will use p to denote the number of processors, τ to denote the communication latency, μ to denote the inverse bandwidth, and assume arithmetic operations have unit cost. You may assume p is a perfect square, the number of rows $n \gg p$, and that p divides n evenly.

   For all of the following problems, you may express your answer in theta notation.

   (a) (20%) Given input matrices A and B with $c > 0$ nonzeroes per row/column as described above, how many flops (arithmetic operations) does it take to compute AB?

   (b) (20%) Consider the straightforward 1D distribution of A and B where each processor has one block row (n/p matrix rows) of A and B. Let $A_0, A_1, \ldots, A_{p-1}$ (resp. $B_0, B_1, \ldots, B_{p-1}$) denote the block rows of A (resp., B). The algorithm proceeds in p rounds, where in round i, processor p_i broadcasts block row B_i, and all processors multiply their block row of A with B_i.

Algorithm 1 Gustavson's SPGEMM algorithm

```
// Input: A, B ∈ R^{n×n}
// Output: C ∈ R^{n×n} (C initialized all to 0)
for i = 0, ..., n − 1 do
    for all k s.t. a_{i,k} is nonzero do
        for all j s.t. b_{k,j} is nonzero do
            c_{i,j} += a_{i,k} · b_{k,j}
        end for
    end for
end for
```

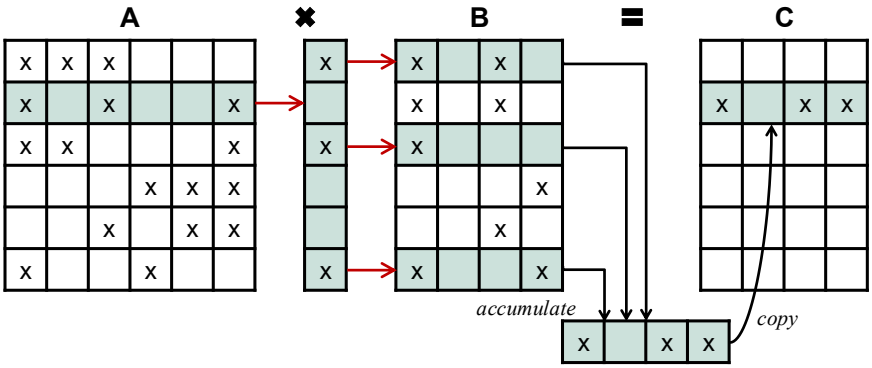

Figure 1: Basic SpGEMM on row order (Gustavson, 1978).

What is the *communication time* of the 1D parallel version of Gustavson's row-wise SpGEMM algorithm? (Hint: the answer should be related to the number of nonzeroes).

(c) (30%) Now consider a 2D SpGEMM algorithm based on Cannon's matrix-multiplication algorithm for dense matrices. In Cannon's algorithm, p processors are arranged in a $\sqrt{p} \times \sqrt{p}$ logical mesh. Each processor gets a submatrix of dimensions $(n/\sqrt{p}) \times (n/\sqrt{p})$.
What is the *communication time* of the 2D parallel version of Gustavson's row-wise SpGEMM algorithm based on Cannon's algorithm? (Hint: the answer should be related to the number of nonzeroes).

(d) (30%) Finally, consider a 2D SpGEMM algorithm based on the SUMMA algorithm for matrix multiplication. In the SUMMA algorithm, processors are again arranged in a logical $\sqrt{p} \times \sqrt{p}$ mesh. In each round, processors in the active processor column broadcast their block column of A along their respective processor rows. Similarly, processors in the active processor row broadcast their block row of B within their respective processor columns.
What is the *communication time* of the 2D parallel version of Gustavson's row-wise SpGEMM algorithm based on SUMMA? (Hint: the answer should be related to the number of nonzeroes).

3. **Communication primitives.** In distributed-memory parallel systems, the **all-gather** collective operation is an essential operation provided, for example, by the MPI library.

(a) Describe the functionality of the all-gather operation and provide a formal definition of its input/output behavior in terms of process ranks and data segments.

(b) Describe two distinct algorithms for implementing all-gather. For each algorithm:

i. Describe its communication pattern.

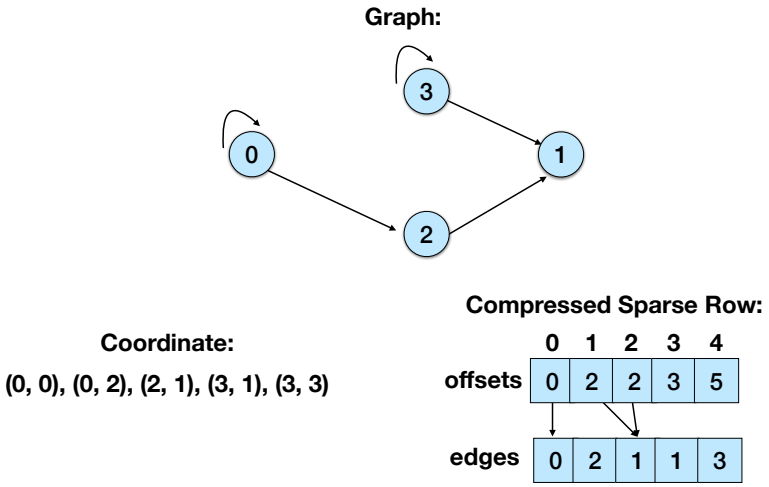

Figure 2: An example of a graph stored in coordinate (COO) format (left) and Compressed Sparse Row (CSR) format (right).

ii. Analyze its **latency cost** (number of communication steps).

iii. Analyze its **bandwidth cost** (total volume of data transferred).

iv. Discuss its scalability with respect to the number of processes p and message size m.

4. **Converting coordinate (COO) format to compressed sparse row (CSR) format.** An unweighted graph (V, E) stored in *coordinate* form maintains the edges E in a sorted list of edges of the form (u, v) where each endpoint $u, v \in V$. On the other hand, a graph stored in *compressed sparse row* format maintains two arrays `offsets` and `edges`. The `offsets` array has length $|V| + 1$, while the `edges` array has length $|E|$. Each entry of the `offsets` array stores an index indicating the starting position of each vertex's edges. Figure 2 illustrates an example graph in COO and CSR formats.

   Assume that the graph in COO format is block-distributed onto p processors (i.e., each processor has $|E|/p$ edges) and that $|E| = \Omega(|V|)$. Given latency/bandwidth communication constants τ and μ, design a distributed-memory parallel algorithm to convert from COO to CSR format that takes $O(|E|/p)$ computation time and $O(|E|/p + (\tau + \mu) \log p)$ communication time. Show that your algorithm achieves the desired bounds.