
Project - 1 **MA517M-Basic Programming Laboratory** Last Date: 09 November 2025
Name **Roll No.: MA25M107**

Sparse Matrix Implementation using C++ and Classes

1. Design a C++ class **SparseMatrix** to efficiently store and manipulate sparse matrices using the **Compressed Sparse Row (CSR)** format. Each matrix should be represented by:

$$A = (\text{values}, \text{cols}, \text{ptr})$$

where:

- **values[k]** stores the nonzero entries of the matrix row by row,
- **cols[k]** stores the corresponding column indices,
- **ptr[i]** stores the index in **values** where row i begins.

For example, for a 3×3 matrix

$$A = \begin{pmatrix} 10 & 0 & 0 \\ 0 & 20 & 30 \\ 40 & 0 & 50 \end{pmatrix}$$

we have:

$$\text{values} = [10, 20, 30, 40, 50], \quad \text{cols} = [0, 1, 2, 0, 2], \quad \text{ptr} = [0, 1, 3, 5].$$

2. Implement the following functionalities in the **SparseMatrix** class:
- A constructor that builds the sparse matrix from a dense $m \times n$ 2D array (or vector of vectors).
 - A member function to display the matrix in both dense and sparse (CSR) formats.
 - A method **toDense()** that converts the CSR matrix back to a 2D array.
3. Implement operator overloading for the following operations:
- **operator+** : Sparse matrix addition
 - **operator-** : Sparse matrix subtraction
 - **operator*** : Sparse matrix–vector multiplication

The multiplication should be implemented as:

$$(Ax)_i = \sum_{j=1}^n a_{ij}x_j,$$

but only iterating over the nonzero elements of A .

4. Add functions to compute:

- The **1-norm** of the matrix:

$$\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^m |a_{ij}|,$$

- The **infinity norm**:

$$\|A\|_\infty = \max_{1 \leq i \leq m} \sum_{j=1}^n |a_{ij}|.$$

5. Implement a function `JacobiSolve(Vector b, int maxIter, double tol)` that solves:

$$Ax = b$$

using the **Jacobi Iterative Method** with CSR-based matrix operations. The iteration formula is:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right)$$

6. Create a test program that:

- Defines a sparse matrix A with at least 50% zeros,
- Defines a right-hand side vector b ,
- Solves $Ax = b$ using the Jacobi method,
- Prints the number of iterations and final residual $\|Ax - b\|_2$.

7. (Optional Extension:) Implement a friend function to compare two sparse matrices using operator overloading:

$$A == B \iff \text{All entries and structure are identical,}$$

$$A < B \iff \|A\|_1 < \|B\|_1,$$

$$A > B \iff \|A\|_1 > \|B\|_1.$$

Project - 2 : Word Shuffle Game Using C++ Classes

Problem Statement

Design and implement a **Word Shuffle Game** using **C++ classes**. The program should allow the user to unscramble letters to form meaningful words. The project should utilize object-oriented programming concepts such as classes, objects, encapsulation, and methods to handle word selection, shuffling, and user interaction.

Project Requirements

1. Create a **Word** class to represent a word in the game.
 - (a) Include a method to randomly select a word from a predefined list.
 - (b) Include a method to shuffle the letters of the word.
 - (c) Include a method to display the shuffled word to the user.
2. Create a **Game** class to manage gameplay.

- (a) Display options for the user: **Play** or **Solution**.
 - (b) If the user chooses **Play**, allow them to enter guesses.
 - (c) Validate user input and provide feedback if the guess is correct or incorrect.
 - (d) Keep track of the number of attempts.
 - (e) Allow the user to quit by entering a special command (e.g., **Q/q**) and confirm before exiting.
- 3. If the user chooses **Solution**, display the original word and the correct sequence of letters.
 - 4. Ensure proper encapsulation of word logic and gameplay operations within the respective classes.

Suggested Class Structure

1. **Word Class:**

- Data member: `string originalWord`, `string shuffledWord`
- Methods: `selectWord()`, `shuffleWord()`, `displayWord()`

2. **Game Class:**

- Data member: `Word` object, user choice, attempt counter
- Methods: `playGame()`, `showSolution()`, `processGuess(string guess)`, `confirmQuit()`