DSA 5005 – Computing Structures – Summer 2019

Project 1 - Due on July 3rd 2019

Introduction:

A rectangular array of numbers is called a *matrix*. Sometimes a matrix is referred as a table, 2-dimensional array, or array of arrays. Consider the following matrix below. It has 5 rows (denoted *n*) and 8 columns (denoted *m*). As you can see the row and column numbers start with a 0.

	0	1	2	3	4	5	6	7
0	100			900		500		
1					200			300
2		400					800	
3			200					
4	1600				700			

The empty cells have the same *common value* (it can be assumed as 0). The above matrix is said to be *sparse* because the total number of common values is significantly large in comparison to other non-common values in the matrix. In the example above, we have a total of 40 values in the matrix (8 × 5), 10 non-common values, and 30 common values.

You can store such a sparse matrix in different formats which will lead to less space complexities. The one method which we are going to be implementing here is called the *Compressed Sparse Row(CSR)* format. In this format, the matrix *n x m* is stored in 3 one dimensional arrays – valueArray, IA and JA.

- valueArray – contains all the non-common values; this array will be of length numNZV which is the number of non-common/non-zero values present in the given input matrix
- IA – contains the number of non-common values in a row
- JA – contains the column number of the non-common value present; length is numNZV

Let us take the example from above and walk through this.

valueArray = [100, 900, 500, 200, 300, 400, 800, 200, 1600, 700]
IA = [0, 3, 5, 7, 8]
JA = [0, 3, 5, 4, 7, 1, 6, 2, 0, 4]

In this project, you will create appropriate C++ classes (see below) to create the compressed sparse matrix representation data structure along with matrix operations on the compressed sparse matrix.

Your project implementation: As part of this project, you will create a class names CSR as described below. This class will have the fields which we used above to store the matrix and also methods necessary.

```
class CSR
{
    protected:
        int noRows; //Number of rows of the original matrix
        int noCols; //Number of columns of the original matrix
        int noNonSparseValues;
        int* valueArray;
        int* IA;
        int* JA;

    public:
        CSR ();
        CSR (int n, int m, int numNZV);
        void setSparseRow (int pos, int r, int c, int v);
        CSR* Add (CSR& M); //Matrix Add
        void display_matrix(); //display in matrix form
        void display_valueArray();
        void display_JA();
        void display_IA();
        //other methods as you deem fit - may include setters,
getters, operator overloading and the ostream operator methods…
};
```

Your main function must look like this:

```
int main()
{
    int n, m, numNZV;
    int testValue, count = 0;

    //matrixA
    cin >> n >> m >> numNZV; //receive the n, m and numNZV for matrix A

    CSR* matrixA = new CSR(n, m, numNZV);

    cout << "MATRIX A ---- Rows, Cols and number of non-common
values: " << matrixA->getRows() <<
        ", " << matrixA->getCols() << ", " << matrixA-
>getnumNZV() << endl;

    //TODO: read in the matrix A from the input file and store
it in the given format

    cout << "The valuesArray for matrix A are : ";
    matrixA->display_valueArray();
    cout << "The JA for matrix A are : ";
    matrixA->display_JA();
    cout << "The IA for matrix A are : ";
    matrixA->display_IA();
    cout << "The matrix A is : " << endl;
    cout << *matrixA; //overload the ostream operator
```

```
    // matrixB
    count = 0;
    cin >> n >> m >> numNZV; //receive the n, m and numNZV for matrix B

    CSR* matrixB = new CSR(n, m, numNZV);

    cout << endl;
    cout << "MATRIX B ---- Rows, Cols and number of non-common
values: " << matrixB->getRows() <<
        ", " << matrixB->getCols() << ", " << matrixB-
>getnumNZV() << endl;

    //TODO: read in the matrix A from the input file and store
it in the given format

    cout << "The valuesArray for matrix B are : ";
    matrixB->display_valueArray();
    cout << "The JA for matrix B are : ";
    matrixB->display_JA();
    cout << "The IA for matrix B are : ";
    matrixB->display_IA();
    cout << "The matrix B is : " << endl;
    cout << *matrixB; //overload the ostream operator

    //addition of the two matrices
    CSR* matrixC = new CSR(n, m, numNZV);
    matrixC = (*matrixA) + (*matrixB); //overloaded '+' operator

    cout << endl;
    cout << "MATRIX C = A + B" << endl;
    cout << "The valuesArray for matrix C are : ";
    matrixC->display_valueArray();
    cout << "The JA for matrix C are : ";
    matrixC->display_JA();
    cout << "The IA for matrix C are : ";
    matrixC->display_IA();
    cout << "The matrix C is : " << endl;
    cout << *matrixC; //overload the ostream operator

    return 0;
}
```

You must ensure that your program outputs the output in the **exact** format provided in the sample outputs. The input file will be of the following format:

```
5 8 10
100 0 0 900 0 500 0 0
0 0 0 0 200 0 0 300
0 400 0 0 0 0 800 0
0 0 200 0 0 0 0 0
1600 0 0 0 700 0 0 0
```

The first line reads - 5 rows, 8 columns and 10 being the number of non-sparse values in the matrix. In the same format, you will have the second matrix as well following this. You are also required to overload the ostream operator to display the matrices and the '+' operator for adding the two CSR objects. Example inputs and outputs for the above given program have been posted in canvas under the Project 1 tab.

The code must be submitted in GradeScope(more instructions on this will be made as an announcement) where they will be auto graded with the sample set of inputs and outputs and will be officially graded later with more exhaustive input files.

Redirected Input: Redirected input provides you a way to send a file to the standard input of a program without typing it using the keyboard. To use redirected input in Visual Studio environment, follow these steps: After you have opened or created a new project, on the menu go to project, project properties, expand configuration properties until you see Debugging, on the right you will see a set of options, and in the command arguments type <"input filename". The < sign is for redirected input and the input filename is the name of the input file (including the path if not in the working directory). A simple program that reads a matrix can be found below.

```
#include <iostream>

using namespace std;

int main ()
{

      int r,c,cv,nsv;
      int val;

      cin >> r >> c >> cv >> nsv;
      for (int i=0; i < r; i++) {
         for (int j=0; j < c; j++) {
           cin >> value;
           cout << value << " ";
         }
         endl;
      }
      return 0;
}
```

Constraints:

1. In this project, the only header you will use is #include <iostream>.
2. None of the projects is a group project. Consulting with other members of this class on programming projects is strictly not allowed and plagiarism charges will be imposed on students who do not follow this.