Assigned: 2023-11-09

Assignment: Data Structure and STL

A sparse matrix is a matrix in which most of the elements are zero. To save on storage, there are several possible approaches. Sparse formats include Compressed Row Storage, Compressed Column Storage, etc. For more information: http://netlib.org/linalg/html_templates/node90.html .

In this assignment, we are going to use a simpler format. In this format, we only store the non-zero elements. Each element is characterized by <row, col, value>.

The matrix is characterized by a class SparseMatrix, with a usage similar to vector in STL. It is a rectangular matrix with `nrow` rows and `ncol` columns.

The declaration code should be:

```
#include<vector>
using namespace std;

struct element {
        int col;
        int row;
        double value;
};

class SparseMatrix{
private:
        vector<element> data;
        int ncol;  // Number of columns
        int nrow;  // Number of rows
};
```

SparseMatrix has all its non-zero values stored in the variable `data` of type vector<element>. ‘ncol’ and ‘nrow’ are the numbers of columns and rows of the matrix.

Following that, your work consists of the following:

1. Create a default constructor and a copy constructor, use initializer lists (10 pt)
2. Implement the assignment operator (10 pt)
3. Implement the following member functions: (10 pt)
   double get(int i, int j);   // get the i-th row, j-th column element
   void set(int i, int j, double v);   //change the i-th row, j-th column element by value
   double operator()(int I, int j); // shortcut to get i-th row, j-th column. Same as get(int i, int j)
4. Complete the SparseMatrix by adding the operator+ function to add two sparse matrices. (Beware: the two sparse matrices do not have non-zero elements in the same locations). (15 pt)

5. Add the operator* function (matrix-matrix multiplication), SparseMatrix times SparseMatrix, which returns another SparseMatrix. The matrices can have different sizes. (15 pt)
   As a reminder, (A*B)[i,j] = sum_{k=0, ncol-1} A[i,k] * B[k,j] .
6. Add the operator * function, a scalar times SparseMatrix, return another SparseMatrix. (10 pt)
   - If mat is a SparseMatrix, I should be able to execute a*mat and mat*a where a is a scalar. This scalar can be either a float, a double, or an int. Use a template function to achieve this (do not create three versions of the overloaded function). There is a way to restrict templates to only act on a subset of types. Ask GPT or research this on Google.
7. Add an operators "<<" to output a SparseMatrix and ">>" to read a SparseMatrix. (10 pt)
8. Demonstrate that you can read and write a SparseMatrix.
9. Use a random number generator to create a SparseMatrix with random numbers in the range [0,1].
10. Create a main function to test the operators you created (scale * matrix, matrix * matrix, matrix + matrix, and the << operator). Make sure you test all the required functionality. (20 pt)

Create a Makefile, zip your code, output, along with any documents, and submit to Canvas

---

Here is a random generator:

```
// Call this function once at the beginning of the main program to set the seed

void initializeRandomSeed() {

   srand(static_cast<unsigned int>(time(nullptr)));

}

//-------------------------------------------------------------------

// Function to generate a random integer between 0 and M (inclusive)

int generateRandomNumber(int M) {

   return rand() % (M + 1);

}
```