CPSC 474 Project 2

Group Members:

Cesar Sepulveda cesar237@csu.fullerton.edu

Screenshot of README file:

README.md

```
1  # 474-Project-2
2  Project 2: Implement a distributed algorithm using MPI
3
4  Group members:
5
6  Cesar Sepulveda cesar237@csu.fullerton.edu
7
```

Description/How to Run the Program:

The goal of this program is to compress a sparse matrix. It will take as input a large matrix whose contents are largely zero and use mpi commands to find nonzero values and their positions within the matrix. The program is written in C++ and uses MPI commands to solve a problem. There are two files that are required for this program to run: sparse.cpp and input.txt. The file sparse.cpp contains all the code as well as the mpi commands, and is the file that needs to be compiled. The text file input.txt is where the program reads in the matrix that it will use. Two shortcomings with this program are that it needs to use four processors to run correctly, and the name of the input file is hardcoded. When running the program, the first thing it will do is ask the user for the number of rows in the matrix, followed by the number of columns in the matrix. After this, the program will check if the correct number of processors have been assigned; if they were, the program then runs as normal, and if not, it will output an error message.

How to compile and run:

- mpic++ sparse.cpp -o sparse
- mpirun -n 4 ./sparse

Pseudocode:

```
int main (){
  int rank                  //Initializing variables.
  int size
  int count1
  int count2
  int count 3
  int n
  int m

  MPI_Init(NULL, NULL)              //Starting the MPI envirnment
  MPI_Comm_rank(MPI_COMM_WORLD, &rank)        //Obtaining the rank and size
  MPI_Comm_rank(MPI_COMM_WORLD, &size)

  if rank is equal to 0 and size is not equal to 4:
    output error message        //Checking for the correct number of processors
    return 0
  end if

  if rank is equal to zero:
    Output: "Enter the number of rows n:"
    input n                     //Asking user to enter the amount of rows(n) and columns(m).
    Output: "Enter the number of columns m:"
    input m
  end if

 MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD); //Broadcasting the rows
 MPI_Bcast(&m, 1, MPI_INT, 0, MPI_COMM_WORLD);//Broadcasting the columns

  vector<vector<int>> vec = readInput(n,m);          //Reading in the matrix
  int  array list1[n*m];                        //Initializing list arrays
  int array list2[n*m];
  int array list3[n*m];

  if rank is equal to 1             //Checking that the processor rank is 1
    count1 = 0
    for i = 0, i < vec.size(), i = i+1:    //Looping through the matrix rows
      if i mod size is equal to 0 or i mod size is equal to 1://Using a mod operation to split work
        for j = 0, j < vec[i].size(), j = j+1:      //Looping through the columns
          if vec[i][j] is not equal to 0:           //Checking is a value is not 0
```

```
            list1[count1] = vec[i][j]          //Adding a value and a position to a list
            list1[count1+1] = i
            list1[count1+2] = j
            count1 = count1+3
          end if
        end for
      end if
    end for
  end if

MPI_Bcast(&count1, 1, MPI_INT, 1, MPI_COMM_WORLD);//Bradcasting the array
MPI_Bcast(&list1, count1, MPI_INT, 1, MPI_COMM_WORLD);//Broadcasting array size

  if rank is equal to 2          //Checking that the processor rank is 1
    count2 = 0
    for i = 0, i < vec.size(), i = i+1:    //Looping through the matrix rows
      if i mod size is equal to 2: //Using a mod operation to split work
        for j = 0, j < vec[i].size(), j = j+1:     //Looping through the columns
          if vec[i][j] is not equal to 0:          //Checking is a value is not 0
            list2[count2] = vec[i][j]              //Adding a value and a position to a list
            list2[count2+1] = i
            list2[count2+2] = j
            count2 = count2+3
          end if
        end for
      end if
    end for
  end if

MPI_Bcast(&count2, 1, MPI_INT, 2, MPI_COMM_WORLD);//Bradcasting the array
MPI_Bcast(&list2, count2, MPI_INT, 2, MPI_COMM_WORLD);//Broadcasting array size

  if rank is equal to 3          //Checking that the processor rank is 1
    count3 = 0
    for i = 0, i < vec.size(), i = i+1:    //Looping through the matrix rows
      if i mod size is equal to 3:         //Using a mod operation to split work
        for j = 0, j < vec[i].size(), j = j+1:     //Looping through the columns
          if vec[i][j] is not equal to 0:          //Checking is a value is not 0
            list3[count3] = vec[i][j]              //Adding a value and a position to a li
            list3[count3+1] = i
```

```
            list3[count3+2] = j
            count3 = count3+3
          end if
        end for
      end if
    end for
  end if

  MPI_Bcast(&count3, 1, MPI_INT, 3, MPI_COMM_WORLD);//Bradcasting the array
  MPI_Bcast(&list3, count3, MPI_INT, 3, MPI_COMM_WORLD);//Broadcasting array size

  if rank is equal to 0:          //Checking that the processor rank is 0
    displayMatrix(vec);                 //Displaying the matrix
    displayList(list1, count1, 1);      //Displaying nonzero values and their position
    displayList(list2, count2, 2);
    displayList(list3, count3, 3);
  end if

  MPI_Finalize()                //Exiting the MPI environment
  return 0                      //Exiting program
}
```

Screenshots of Code Executing: