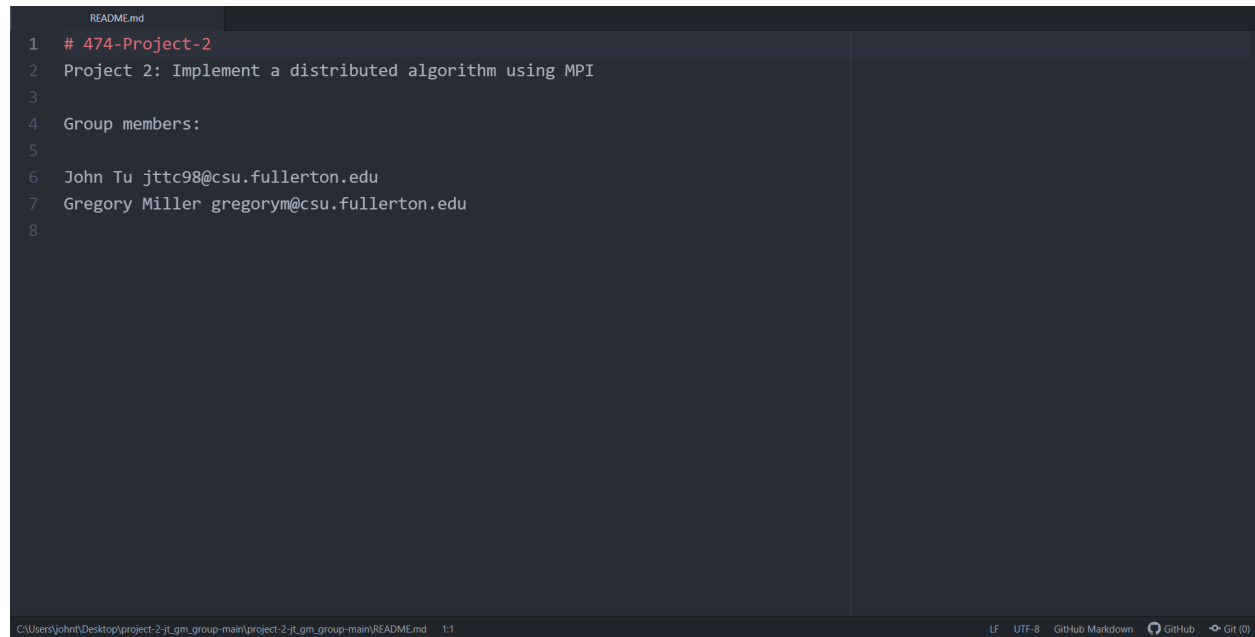CPSC 474 Project 2 Report: Sparse Matrix Transpose

John Tu (jttc98@csu.fullerton.edu)
Gregory Miller (gregorym@csu.fullerton.edu)

Screenshot for README.md:

Pseudocode:

```
def main(argc, argv):
        Create communicator object
        Create a mpi rank variable
        Create a mpi size variable
        Create variables for number of rows and columns

        Create a list for the text file values (file_list)
        Create a list for values in rows (list_row)
        Create a list for values in columns (list_col)
        Create a list for values for each row and column (list_val)
        Create index values for rows, columns and values

        if size < 2:
                print("There needs to be at least 2 processes.")
                Exit program
        if argc != 2:
                print("Invalid number of arguments.")
                Exit program
```

```
print("I am rank {} out of {}".format(rank, size))
if rank == 0:
        Open file in read mode
        for lines in file:
                Split the line
                Append the line into list
        Close file

        # For each tuple of nonzero values, obtain the row, column, and value of the
        sparse matrix
        # and then store them in three separate lists.
        For i in range length of file_list:
                Obtain the row and append it into list_row
                Obtain the column and append it into list_col
                Obtain the value and append it into list_val

        # Get the number of rows and columns by obtaining the maximum values.
        n = max(list_row)+1
        m = max(list_col)+1

        Create a matrix of size n by m with all values initialized to zero.


        Now begin constructing the original matrix based on the row and column location.
        Then, print out the completed matrix.
        for i in range length of list_val:
                r = list_row at index i
                c = list_col at index i
                matrix[r][c] = list_val at index i
        for i in range from 0 to n:
                print(matrix[i])

        Send over the number of rows and columns for the restored sparse matrix, then
        call wait() on the rows and columns, along with the matrix itself over to any
        nonzero process for transposing.

elif rank == 1:
        Receive the rows and columns with mpi_comm.irecv()
        Declare a variable n_req and get the number of rows received via wait().
        Declare a variable m_req and get the number of columns received via wait().
        print("Number of rows received from process 0:", n_req)
        print("Number of columns received from process 0:", m_req)
```

Receive the whole matrix from process 0.
Get the whole matrix received via wait() and assign to mat_req.

```
for j in range(0, n_req):
        print(mat_req[j])
print("Now transposing the matrix...")
```
Create a transposed matrix of size len(mat_req[0]) by len(mat_req) with all values initialized to mat_req[j][i], where j is 0 to len(mat_req) and i is 0 to len(mat_req[0]).
```
print("Transpose complete. Here's the result:")
# Print the completed transposed sparse matrix.
for j in range(0, m_req):
        print(transposed[j])
else:
        pass
```

How to run the code:
(NOTE: Be sure to have mpi4py installed or else the program will not compile!)
1) Download the file for project 2 on github and save it to a location that has access to mpiexec (mpi4py) functionality.
2) cd into the project folder location
3) Then type the following command: mpiexec -n [number] python mpi_sparse_transpose.py [text file name].txt (Replace [number] with the number of processors you would like to use and [text file name] with the text file you would like to open. For example, 'mpiexec -n 5 python mpi_sparse_transpose.py sample_sparse1.txt' will use 5 processors and open sample_sparse1.txt as input text file.
        Note: To change the number of processors used, change the number after -n. Also there are other example text files, to view those, so be sure to change [text file name] if you would like to try out other inputs.
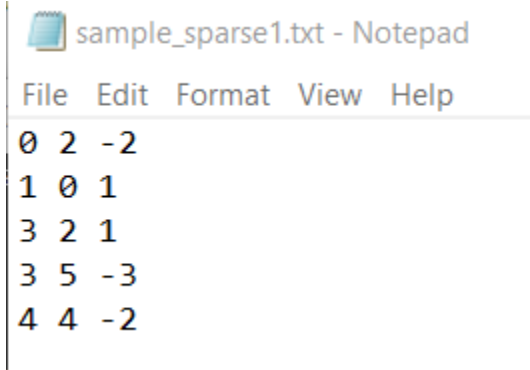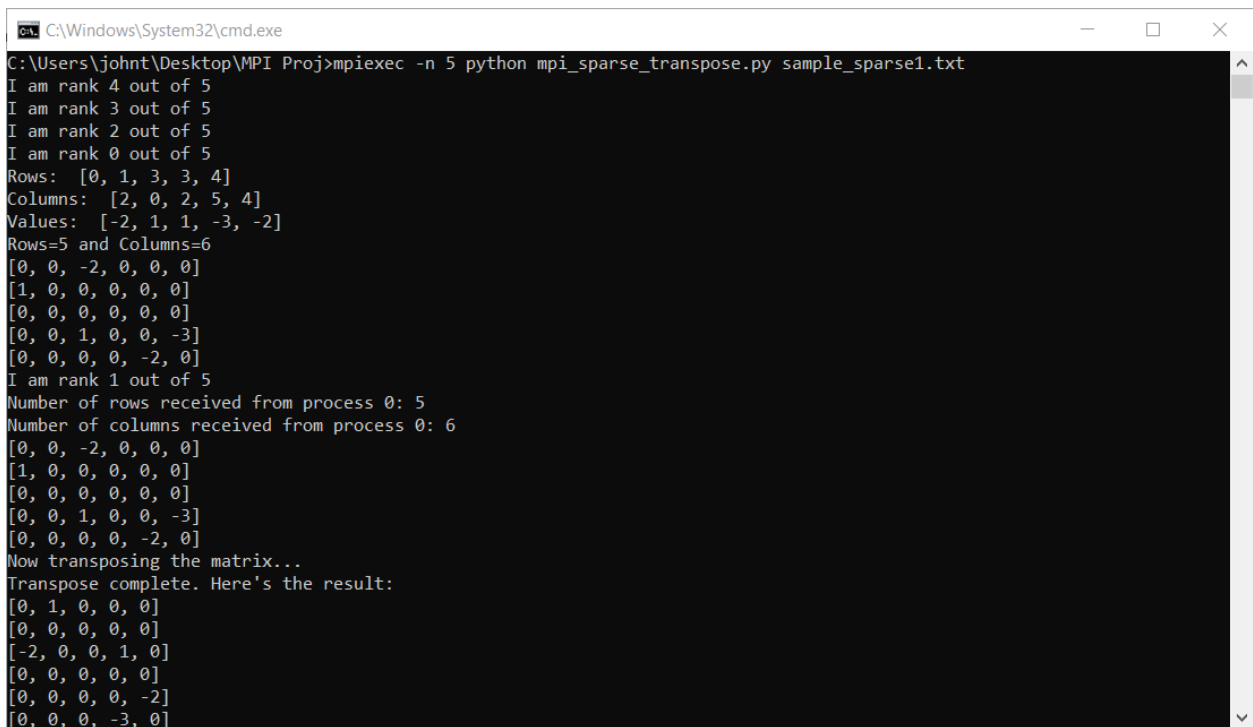
Screenshots for Test Cases:

sample_sparse1.txt (Input)

```
sample_sparse1.txt - Notepad

File  Edit  Format  View  Help
0 2 -2
1 0 1
3 2 1
3 5 -3
4 4 -2
```

sample_sparse1.txt (Output)

```
C:\Windows\System32\cmd.exe                                                    —    □    ×

C:\Users\johnt\Desktop\MPI Proj>mpiexec -n 5 python mpi_sparse_transpose.py sample_sparse1.txt
I am rank 4 out of 5
I am rank 3 out of 5
I am rank 2 out of 5
I am rank 0 out of 5
Rows:    [0, 1, 3, 3, 4]
Columns:  [2, 0, 2, 5, 4]
Values:  [-2, 1, 1, -3, -2]
Rows=5 and Columns=6
[0, 0, -2, 0, 0, 0]
[1, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0]
[0, 0, 1, 0, 0, -3]
[0, 0, 0, 0, -2, 0]
I am rank 1 out of 5
Number of rows received from process 0: 5
Number of columns received from process 0: 6
[0, 0, -2, 0, 0, 0]
[1, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0]
[0, 0, 1, 0, 0, -3]
[0, 0, 0, 0, -2, 0]
Now transposing the matrix...
Transpose complete. Here's the result:
[0, 1, 0, 0, 0]
[0, 0, 0, 0, 0]
[-2, 0, 0, 1, 0]
[0, 0, 0, 0, 0]
[0, 0, 0, 0, -2]
[0, 0, 0, -3, 0]
```

sample_sparse2.txt (Input)

sample_sparse2.txt (Output)


```
Microsoft Windows [Version 10.0.18363.1916]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Grego>mpiexec -n 5 python mpi_sparse_transpose.py sample_sparse2.txt
I am rank 2 out of 5
I am rank 4 out of 5
I am rank 3 out of 5
I am rank 0 out of 5
Rows:   [0, 0, 1, 1, 2, 2]
Columns:  [0, 1, 2, 3, 4, 5]
Values:  [1, 2, 3, 4, 5, 6]
Rows=3 and Columns=6
[1, 2, 0, 0, 0, 0]
[0, 0, 3, 4, 0, 0]
[0, 0, 0, 0, 5, 6]
I am rank 1 out of 5
Number of rows received from process 0: 3
Number of columns received from process 0: 6
[1, 2, 0, 0, 0, 0]
[0, 0, 3, 4, 0, 0]
[0, 0, 0, 0, 5, 6]
Now transposing the matrix...
Transpose complete. Here's the result:
[1, 0, 0]
[2, 0, 0]
[0, 3, 0]
[0, 4, 0]
[0, 0, 5]
[0, 0, 6]
```

sample_sparse3.txt (Input)

sample_sparse3.txt - Notepad

File Edit Format View Help

```
7 3 90
12 3 57
11 10 84
18 6 99
0 0 26
14 20 82
5 13 85
15 4 78
16 2 51
16 1 100
3 9 60
6 16 94
9 19 94
17 2 14
17 7 97
10 2 88
2 20 67
8 23 26
14 2 50
```

sample_sparse3.txt (Output)

```
C:\Windows\System32\cmd.exe                                              —    □    ×

C:\Users\johnt\Desktop\MPI Proj>mpiexec -n 5 python mpi_sparse_transpose.py sample_sparse3.txt
I am rank 3 out of 5
I am rank 4 out of 5
I am rank 2 out of 5
I am rank 0 out of 5
Rows:  [7, 12, 11, 18, 0, 14, 5, 15, 16, 16, 3, 6, 9, 17, 17, 10, 2, 8, 14]
Columns:  [3, 3, 10, 6, 0, 20, 13, 4, 2, 1, 9, 16, 19, 2, 7, 2, 20, 23, 2]
Values:  [90, 57, 84, 99, 26, 82, 85, 78, 51, 100, 60, 94, 94, 14, 97, 88, 67, 26, 50]
Rows=19 and Columns=24
[26, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 67, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 60, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 85, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 94, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 90, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 26]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 94, 0, 0, 0, 0, 0]
[0, 0, 88, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 84, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 57, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 50, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 82, 0, 0, 0]
[0, 0, 0, 0, 78, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 100, 51, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 14, 0, 0, 0, 0, 97, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 99, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
I am rank 1 out of 5
Number of rows received from process 0: 19
```

```
Number of columns received from process 0: 24
[26, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 67, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 60, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 85, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 94, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 90, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 26]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 94, 0, 0, 0]
[0, 0, 88, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 84, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 57, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 50, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 82, 0, 0, 0]
[0, 0, 0, 0, 78, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 100, 51, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 14, 0, 0, 0, 0, 97, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 99, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Now transposing the matrix...
Transpose complete. Here's the result:
[26, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 100, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 88, 0, 0, 0, 50, 0, 51, 14, 0]
[0, 0, 0, 0, 0, 0, 0, 90, 0, 0, 0, 0, 57, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 78, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 99]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 97, 0]
```

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 60, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 84, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 85, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 94, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 94, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 67, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 82, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 26, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

C:\Users\johnt\Desktop\MPI Proj>
```