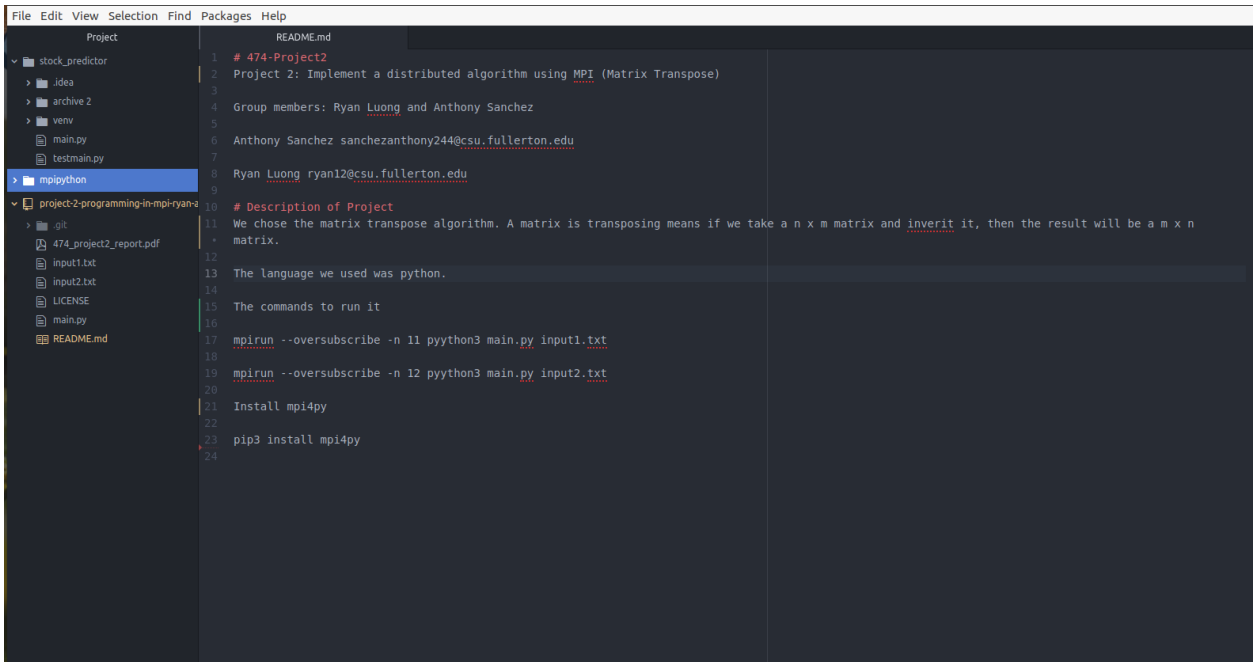
474-Project2
Project 2: Implement a distributed algorithm using MPI (Transpose Matrix)

Group members: Ryan Luong and Anthony Sanchez

sanchezanthony244@csu.fullerton.edu

ryan12@csu.fullerton.edu



Summary:

This document includes group member information, the pseudocode of our chosen algorithm, as well as the result of our project. We chose the matrix transpose algorithm for this project and completed it in Python. A matrix is transposing means if we take a n x m matrix and invert it, then the result will be a m x n matrix. MPI_Broadcast and MPI_Gather will achieve this. We originally worked on a machine learning algorithm but encountered difficulties combining MPI commands with the model. This was a challenging project.

Pseudocode:

```
import mpi and sys

initialize MPI, rank, and size
get the terminal arguments
initialize n, m and elapsed time to 0
```

```
matrix = [], row_list = []

if the rank is 0 {
    start time
    print the message "Original Matrix"
    open the file {
        for each line in the file {
            print the line
            data = []
            for each data in the line {
                if the data is numeric and n and m are not 0 {
                    add data to the data list as an int
                }
                else {
                    if n is 0 {
                        let n euqal that data as an int
                    }
                    else if m is 0 {
                        let m equal that data as an int
                    }
                }
            }
            if data is not empty {
                add it to row_list
            }
        }
    }
    end the time
    add it to elapsed time
    MPI.Barrier()
else {
    MPI.Barrier()
}

broadcast the elapsed time, n, m and row_list to the other processes

if the elapsed time exceeds 60 minutes {
    Abort it
}
```

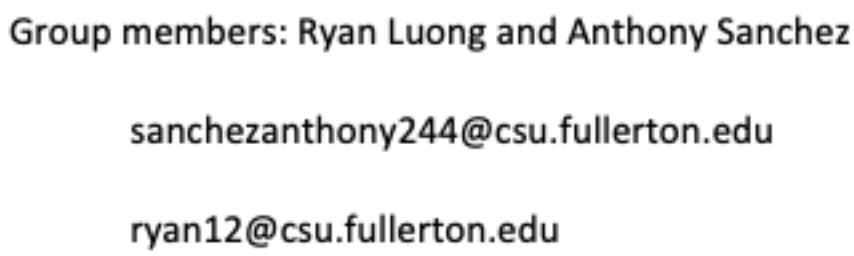
```
for each row in the row_list {
    start the time
    scatter the row to the other processes
    append it to matrix
    end the time
    add it to elapsed time
}

broadcast elapsed time
gather the results of the processes to the root

if the elapsed time exceeds 60 minutes {
    Abort it
}

if the rank is 0 {
    print the message "Tranpose Matrix" and output the whole tranposed
matrix
}
}
```

Screenshots:



```
mpirun --oversubscribe -n 11 python3 main.py input1.txt
Original Matrix
11 11

[ [ 1, 12, 4, 5, 7, 11, 3, 75, 90, 10, 11 ] ,

[ 25, 3, 41, 55, 100, 32, 324, 493, 392, 1, 55 ],

[ 5, 2, 100, 23, 10, 22, 24, 93, 92, 15, 50 ],

[ 2, 15, 1, 5, 13, 2, 34, 43, 32, 5, 54 ],

[ 3, 9, 14, 34, 94, 23, 32, 49, 39, 19, 551 ],

[ 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 ],

[ 15, 17, 22, 55, 43, 20, 14, 54, 85, 14, 32 ],

[ 94, 32, 411, 553, 1001, 322, 3324, 1493, 4392, 12, 515 ],

[ 40, 30, 20, 10, 100, 50, 60, 70, 80, 90, 0 ],

[ 33, 31, 14, 91, 54, 29, 50, 15, 89, 10, 75 ],

[ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 ] ]

Transposed Matrix

[1, 25, 5, 2, 3, 4, 15, 94, 40, 33, 1]

[12, 3, 2, 15, 9, 5, 17, 32, 30, 31, 1]

[4, 41, 100, 1, 14, 6, 22, 411, 20, 14, 1]

[5, 55, 23, 5, 34, 7, 55, 553, 10, 91, 1]

[7, 100, 10, 13, 94, 8, 43, 1001, 100, 54, 1]

[11, 32, 22, 2, 23, 9, 20, 322, 50, 29, 1]

[3, 324, 24, 34, 32, 10, 14, 3324, 60, 50, 1]

[75, 493, 93, 43, 49, 11, 54, 1493, 70, 15, 1]

[90, 392, 92, 32, 39, 12, 85, 4392, 80, 89, 1]

[10, 1, 15, 5, 19, 13, 14, 12, 90, 10, 1]
```

```
mpirun --oversubscribe -n 12 python3 main.py input2.txt
Original Matrix
12 12

[ [ 1, 12, 4, 5, 7, 11, 3, 75, 90, 10, 11, 50 ] ,

[ 25, 3, 41, 55, 100, 32, 324, 493, 392, 1, 55, 60 ],

[ 5, 2, 100, 23, 10, 22, 24, 93, 92, 15, 50, 70 ],

[ 2, 15, 1, 5, 13, 2, 34, 43, 32, 5, 54, 100 ],

[ 3, 9, 14, 34, 94, 23, 32, 49, 39, 19, 551, 10 ],

[ 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 20 ],

[ 15, 17, 22, 55, 43, 20, 14, 54, 85, 14, 32, 30 ],

[ 94, 32, 411, 553, 1001, 322, 3324, 1493, 4392, 12, 515, 300 ],

[ 40, 30, 20, 10, 100, 50, 60, 70, 80, 90, 0, 400 ],

[ 33, 31, 14, 91, 54, 29, 50, 15, 89, 10, 75, 200 ],

[ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 ] ]

Transposed Matrix

[1, 25, 5, 2, 3, 4, 15, 94, 40, 33, 1]

[12, 3, 2, 15, 9, 5, 17, 32, 30, 31, 1]

[4, 41, 100, 1, 14, 6, 22, 411, 20, 14, 1]

[5, 55, 23, 5, 34, 7, 55, 553, 10, 91, 1]

[7, 100, 10, 13, 94, 8, 43, 1001, 100, 54, 1]

[11, 32, 22, 2, 23, 9, 20, 322, 50, 29, 1]

[3, 324, 24, 34, 32, 10, 14, 3324, 60, 50, 1]

[75, 493, 93, 43, 49, 11, 54, 1493, 70, 15, 1]

[90, 392, 92, 32, 39, 12, 85, 4392, 80, 89, 1]

[10, 1, 15, 5, 19, 13, 14, 12, 90, 10, 1]

[11, 55, 50, 54, 551, 14, 32, 515, 0, 75, 1]

[50, 60, 70, 100, 10, 20, 30, 300, 400, 200, 1]
```