

Davina Doran: davdoran@csu.fullerton

Ashu Singh: ashusingh28@csu.fullerton.edu

Robert Susanto: rsusanto0@csu.fullerton.edu

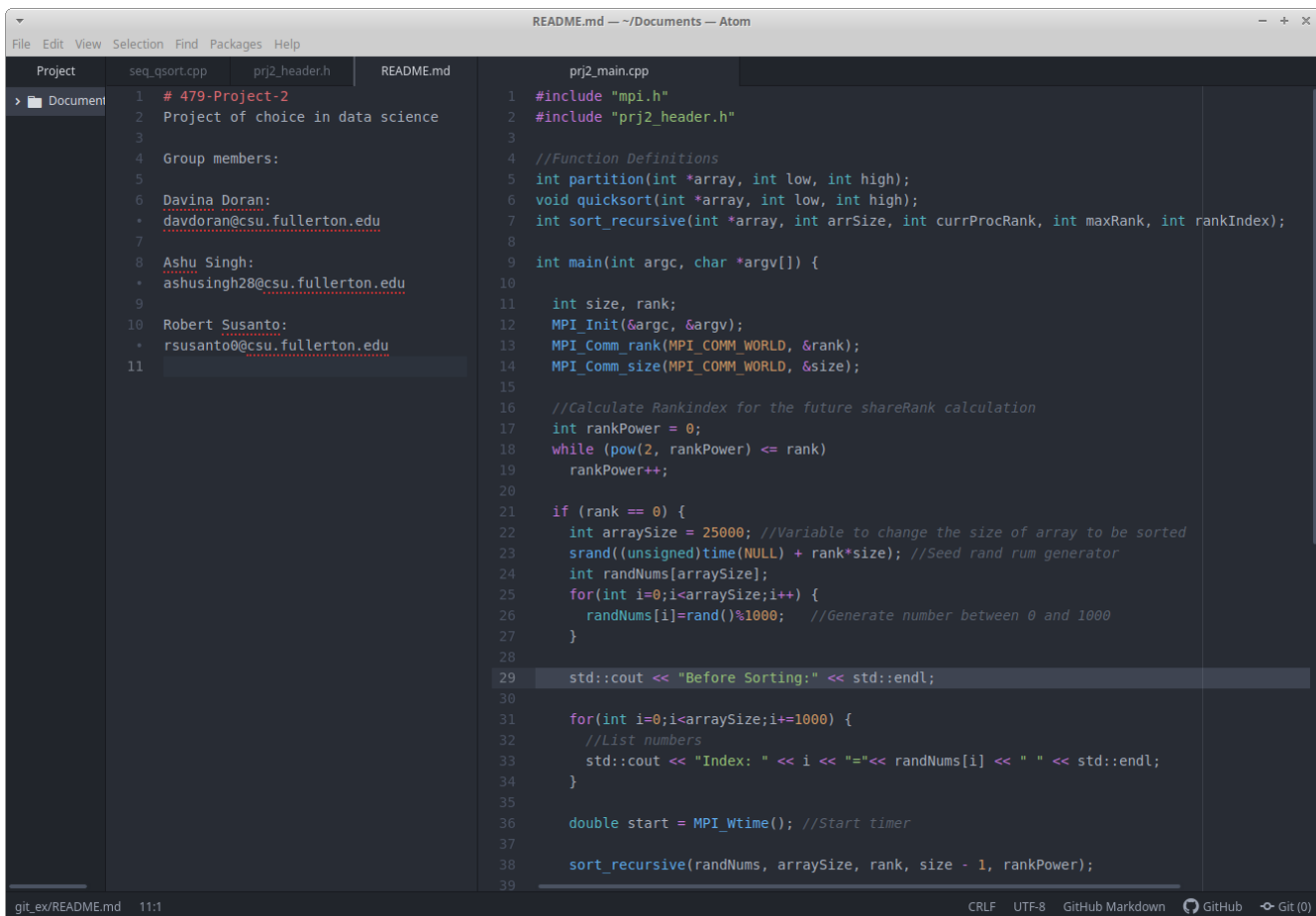
CPSC 479-01

Professor: Doina Bein

05-19-2021

Project 2 Report Document

Full Screen Screenshot (w/ group member names):



The screenshot shows the Atom text editor interface. The left sidebar displays a file explorer with a 'Project' view. The main editor area is split into two panes. The left pane shows the 'README.md' file, which contains the following text:

```
1 # 479-Project-2
2 Project of choice in data science
3
4 Group members:
5
6 Davina Doran:
7   + davdoran@csu.fullerton.edu
8
9 Ashu Singh:
10  + ashusingh28@csu.fullerton.edu
11
12 Robert Susanto:
13   + rsusanto0@csu.fullerton.edu
```

The right pane shows the 'prj2_main.cpp' file, which contains the following C++ code:

```
1 #include "mpi.h"
2 #include "prj2_header.h"
3
4 //Function Definitions
5 int partition(int *array, int low, int high);
6 void quicksort(int *array, int low, int high);
7 int sort_recursive(int *array, int arrSize, int currProcRank, int maxRank, int rankIndex);
8
9 int main(int argc, char *argv[]) {
10
11     int size, rank;
12     MPI_Init(&argc, &argv);
13     MPI_Comm_rank(MPI_COMM_WORLD, &rank);
14     MPI_Comm_size(MPI_COMM_WORLD, &size);
15
16     //Calculate Rankindex for the future shareRank calculation
17     int rankPower = 0;
18     while (pow(2, rankPower) <= rank)
19         rankPower++;
20
21     if (rank == 0) {
22         int arraySize = 25000; //Variable to change the size of array to be sorted
23         srand((unsigned)time(NULL) + rank*size); //Seed rand rum generator
24         int randNums[arraySize];
25         for(int i=0;i<arraySize;i++) {
26             randNums[i]=rand()%1000; //Generate number between 0 and 1000
27         }
28
29         std::cout << "Before Sorting:" << std::endl;
30
31         for(int i=0;i<arraySize;i+=1000) {
32             //List numbers
33             std::cout << "Index: " << i << "=="<< randNums[i] << " " << std::endl;
34         }
35
36         double start = MPI_Wtime(); //Start timer
37
38         sort_recursive(randNums, arraySize, rank, size - 1, rankPower);
39     }
```

How to Run:

Compile: `mpiCC pri2_main.cpp`

Run: `mpirun -np <num of processors> ./a.out` (***) sometimes receives seg fault. Just rerun and that solves it***)

Pseudocode:

MAIN.CPP

main:

```
    if (rank == 0): //leader
        - MPI: Init, rank, size: rank, size
        - create & output array of random numbers: A[n]
        - start clock
        - call sort_recursive(A)
index    - end clock
        - output contents of A & end - start times
    else: // rest of the processes
        - MPI: status, Probe, Get_count //get size of array to recv.
        - allocate memory for subarray: SA
        - MPI_Recv(SA, SASize)
        - sort_recursive(SA)
        - MPI_Send(SA, SASize)
```

HEADER.H

```
int sort_recursive(A, arraySize, currentProcessRank, maxRank, rankIndex):
    - sharedProcess = currentProcessRank + 2^(rankIndex); rankIndex++
    - if (sharedProcess > maxRank): sort chunk sequentially
    - do:
        - pivotIndex = partition(A, j, arraySize - 1); j++
    - while (pivotIndex = j - 1)

    - if(pivotIndex <= arraySize - pivotIndex)
        - MPI_Send(A, pivotIndex - 1)
        - sort_recursive((A + pivotIndex + 1), (arraySize - pivotIndex - 1), currentRank,
                        maxRank, rankIndex)
        - MPI_Recv((A+ pivotIndex + 1), (arraySize - pivotIndex - 1))
```

void quicksort(A, low, high):

```
    - if (low < high):
        - p = partition(A, low, high);
        - quicksort(A, low, p)
        - quicksort(A, p + 1, high)
```

int partition(A, low, high):

```
    - pivot = A[low]; i = low - 1; j = high + 1
    - while(true):
        - do: i = i + 1; while(A[i] < pivot)
        - do: j = j - 1; while(A[j] > pivot)
        - if ( i >= j): return j
    - temp = A[i]; A[i] = A[j]; A[j] = temp
```

Execution Snapshots:

5 Processes

```
Terminal - student@tuffix-vm: ~/Documents/git_ex
File Edit View Terminal Tabs Help
student@tuffix-vm:~/Documents/git_ex$ mpirun -np 5 -oversubscribe ./a.out
Before Sorting:
Index: 0=189
Index: 1000=952
Index: 2000=227
Index: 3000=89
Index: 4000=280
Index: 5000=288
Index: 6000=605
Index: 7000=117
Index: 8000=394
Index: 9000=917
Index: 10000=43
Index: 11000=859
Index: 12000=17
Index: 13000=992
Index: 14000=856
Index: 15000=203
Index: 16000=18
Index: 17000=248
Index: 18000=278
Index: 19000=201
Index: 20000=449
Index: 21000=293
Index: 22000=798
Index: 23000=304
Index: 24000=792
After Sorting:
Index:0=0
Index:1000=41
Index:2000=81
Index:3000=122
Index:4000=160
Index:5000=199
Index:6000=238
Index:7000=279
Index:8000=317
Index:9000=355
Index:10000=397
Index:11000=435
Index:12000=475
Index:13000=516
Index:14000=556
Index:15000=598
Index:16000=638
Index:17000=679
Index:18000=717
Index:19000=761
Index:20000=802
Index:21000=841
Index:22000=879
Index:23000=918
Index:24000=960
Execution time:0.0432882
student@tuffix-vm:~/Documents/git_ex$
```

20 Processes:

```
Terminal - student@tuffix-vm: ~/Documents/git_ex
File Edit View Terminal Tabs Help
student@tuffix-vm:~/Documents/git_ex$ mpirun -np 20 -oversubscribe ./a.out
Before Sorting:
0=363
1000=988
2000=419
3000=732
4000=985
5000=551
6000=707
7000=331
8000=607
9000=641
10000=109
11000=664
12000=329
13000=397
14000=426
15000=863
16000=699
17000=602
18000=623
19000=372
20000=456
21000=972
22000=273
23000=916
24000=870
After Sorting:
0=0
1000=41
2000=82
3000=121
4000=160
5000=200
6000=242
7000=281
8000=322
9000=361
10000=400
11000=438
12000=479
13000=520
14000=559
15000=600
16000=640
17000=680
18000=720
19000=760
20000=798
21000=839
22000=880
23000=920
24000=962
Execution time:0.0871126
student@tuffix-vm:~/Documents/git_ex$
```

Sequential

```
Terminal - student@tuffix-vm: ~/Documents
File Edit View Terminal Tabs Help
Before Sorting:
Index: 0=468
Index: 1000=891
Index: 2000=905
Index: 3000=112
Index: 4000=941
Index: 5000=324
Index: 6000=435
Index: 7000=165
Index: 8000=591
Index: 9000=577
Index: 10000=347
Index: 11000=902
Index: 12000=547
Index: 13000=652
Index: 14000=404
Index: 15000=871
Index: 16000=224
Index: 17000=727
Index: 18000=923
Index: 19000=444
Index: 20000=325
Index: 21000=592
Index: 22000=152
Index: 23000=961
Index: 24000=214
After Sorting:
Index: 0=0
Index: 1000=39
Index: 2000=79
Index: 3000=118
Index: 4000=156
Index: 5000=197
Index: 6000=239
Index: 7000=278
Index: 8000=318
Index: 9000=358
Index: 10000=400
Index: 11000=438
Index: 12000=477
Index: 13000=516
Index: 14000=558
Index: 15000=600
Index: 16000=641
Index: 17000=681
Index: 18000=721
Index: 19000=758
Index: 20000=797
Index: 21000=839
Index: 22000=880
Index: 23000=919
Index: 24000=959
Time measured: 0.011 seconds.
student@tuffix-vm:~/Documents$
```