Project 2: Data Science project

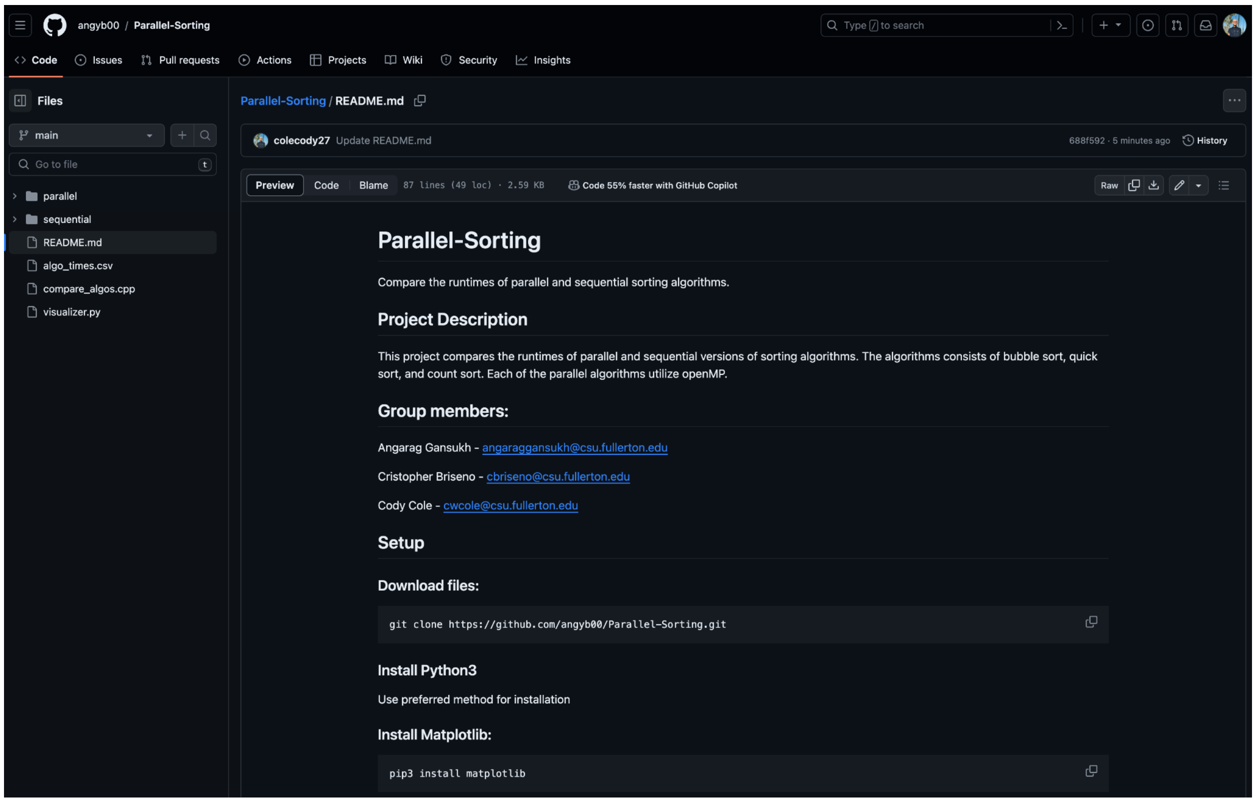
Project Summary

This project compares the runtimes of parallel and sequential versions of three sorting algorithms. The algorithms consist of **bubble sort, quick sort, and count sort.** Each algorithm has a parallel and sequential approach, making for a total of **6 sorting functions.** Each of the parallel algorithms utilize openMP to successfully parallelize.

All 6 sorting functions are passed three random arrays consisting of **100, 1,000, and 10,000** elements. The run times for each function are tracked and visualized by a python script. The visualization is a two-line plot. One line is the sequential algorithm and the other is the parallel.

Team Members

Angarag Gansukh - angaraggansukh@csu.fullerton.edu
Cristopher Briseno - cbriseno@csu.fullerton.edu
Cody Cole - cwcole@csu.fullerton.edu



Parallel-Sorting

Compare the runtimes of parallel and sequential sorting algorithms.

Project Description

This project compares the runtimes of parallel and sequential versions of sorting algorithms. The algorithms consists of bubble sort, quick sort, and count sort. Each of the parallel algorithms utilize openMP.

Group members:

Angarag Gansukh - angaraggansukh@csu.fullerton.edu

Cristopher Briseno - cbriseno@csu.fullerton.edu

Cody Cole - cwcole@csu.fullerton.edu

Setup

Download files:

```
git clone https://github.com/angyb00/Parallel-Sorting.git
```

Install Python3

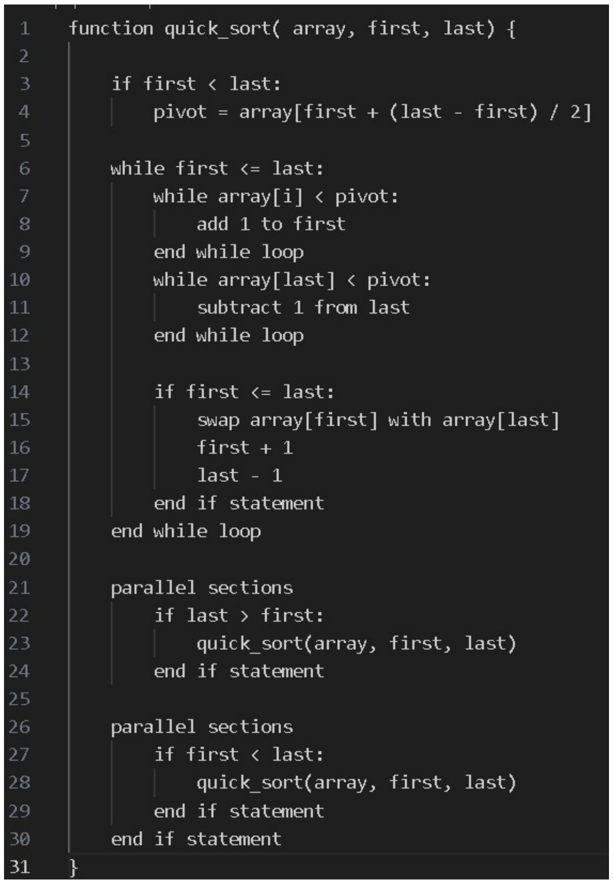
Use preferred method for installation

Install Matplotlib:

```
pip3 install matplotlib
```

Pseudocode

Quick Sort



```
function quick_sort( array, first, last) {

    if first < last:
        pivot = array[first + (last - first) / 2]

    while first <= last:
        while array[i] < pivot:
            add 1 to first
        end while loop
        while array[last] < pivot:
            subtract 1 from last
        end while loop

        if first <= last:
            swap array[first] with array[last]
            first + 1
            last - 1
        end if statement
    end while loop

    parallel sections
        if last > first:
            quick_sort(array, first, last)
        end if statement

    parallel sections
        if first < last:
            quick_sort(array, first, last)
        end if statement
    end if statement
}
```

Bubble Sort

```
Void bubble_sort(int array){
start for loop with i going from 0 to N and incrementing i by 1 every iteration to
start odd even phase
        if i is even, enter even phase
                start parallel omp for loop with j going from 1 to N incrementing j by 2
every iteration and with 16 threads
                        if array[j - 1] is greater than array[j]
                                then swap the two elements
        if i is odd, enter odd phase
                start parallel omp for loop with j going from 1 to N - 1 and incrementing
j by 2 every iteration and with 16 threads
                        if array[j] > array[j+1]
                                then swap the two elements
}
```

Count Sort

```
void par_count_sort(int array){
   // Get largest element in unsorted array which will be the size of the array of
values

   // Iterate through nums and increment each value in count array
   // Each thread will get an index in a block fashion
   Start parallel omp for loop
       Have each thread increment the index of the value

   // Iterate through each count and insert values into sorted array
   Start parallel omp for loop
       Iterate through the array, inserting the values found in the other array of
values
}
```

Execution Instructions

Setup

Download files:

```
git clone https://github.com/angyb00/Parallel-Sorting.git
```

Install Python3

Use preferred method for installation

Install Matplotlib:

```
pip3 install matplotlib
```

Compilation

This project utilizes 16 threads for the completion of the parallel algorithms. In this case, it will be necessary to login to the school's remote server. A sample output of running times has been provided within the project's root folder ("algo_times.csv"). NOTE: *To see the visualization of the times, without gathering a new list of times, run:

```
 python3 visualizer.py
```

Login to remote server

```
ssh -l <username> ecs.fullerton.edu
```

Compile Parallel Algorithms

Navigate to parallel folder:

```
cd parallel
```

Compile:

```
g++ -c par_algos.cpp
```

Compile Sequential Algorithms

Navigate to parallel folder:

```
cd sequential
```

Compile:

```
g++ -c seq_algos.cpp
```

Compile and Run Driver Code

Navigate to root folder:

```
cd ..
```

Compile:

```
g++ -c compare_algos.cpp
```

Link files:

```
g++ sequential/seq_algos.o parallel/par_algos.o compare_algos.o
```

Run

```
./a.out
```

Run visualization

A csv file, ("algo_times.csv"), of the running times has been generated within the project folder. This cannot be run on the remote server. This can be copied over to your local device by any means but below will be a supplemental option. The only file necessary to make the visualization is "visualizer.py". So make sure this file or the project has been downloaded onto the local device.

Print and copy the contents of CSV file found on the remote desktop in the root folder ("PARALLEL-SORTING"):

```
cat algo_times.csv
```

Locate the pre-existing CSV file containing samples outputs within the root folder ("PARALLEL-SORTING")

Paste the contents found on the remote desktop in the local CSV file ("algo_times.csv"):

```
nano algo_times.csv
```

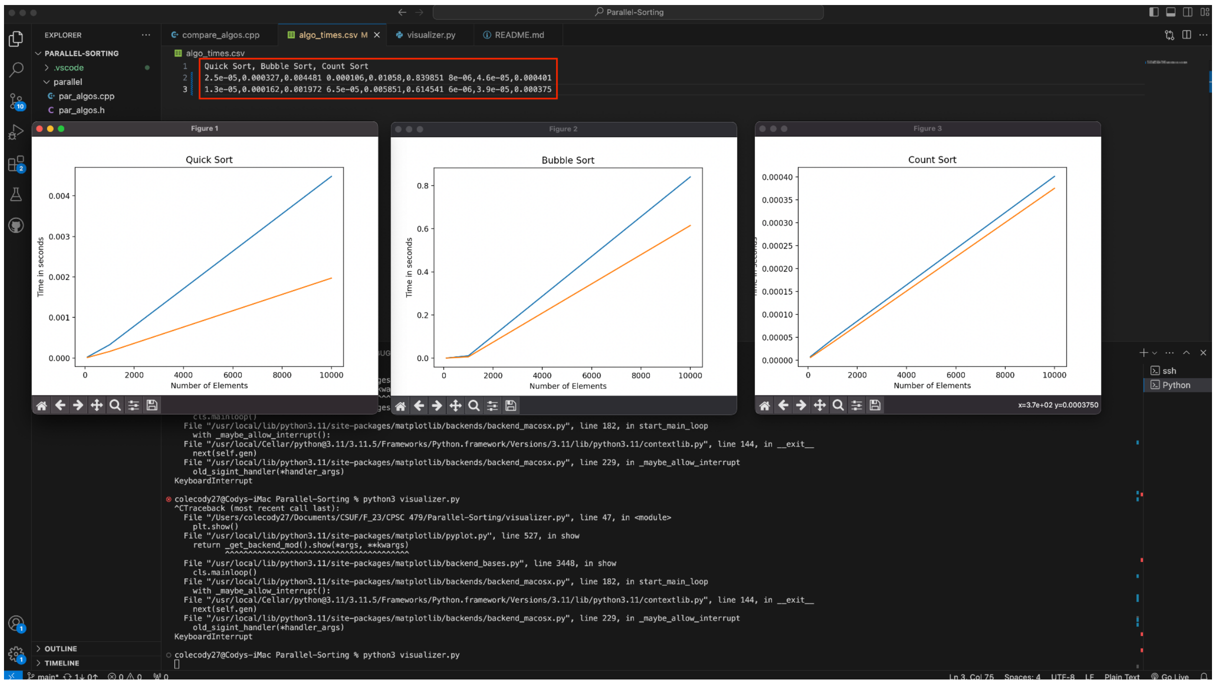
Run:

```
python3 visualizer.py
```

Executed Code Screenshots

NOTE: Each of the 6 sorting functions are giving 3 random arrays consisting of **100, 1,000, and 10,000 elements** upon each execution.

Execution 1



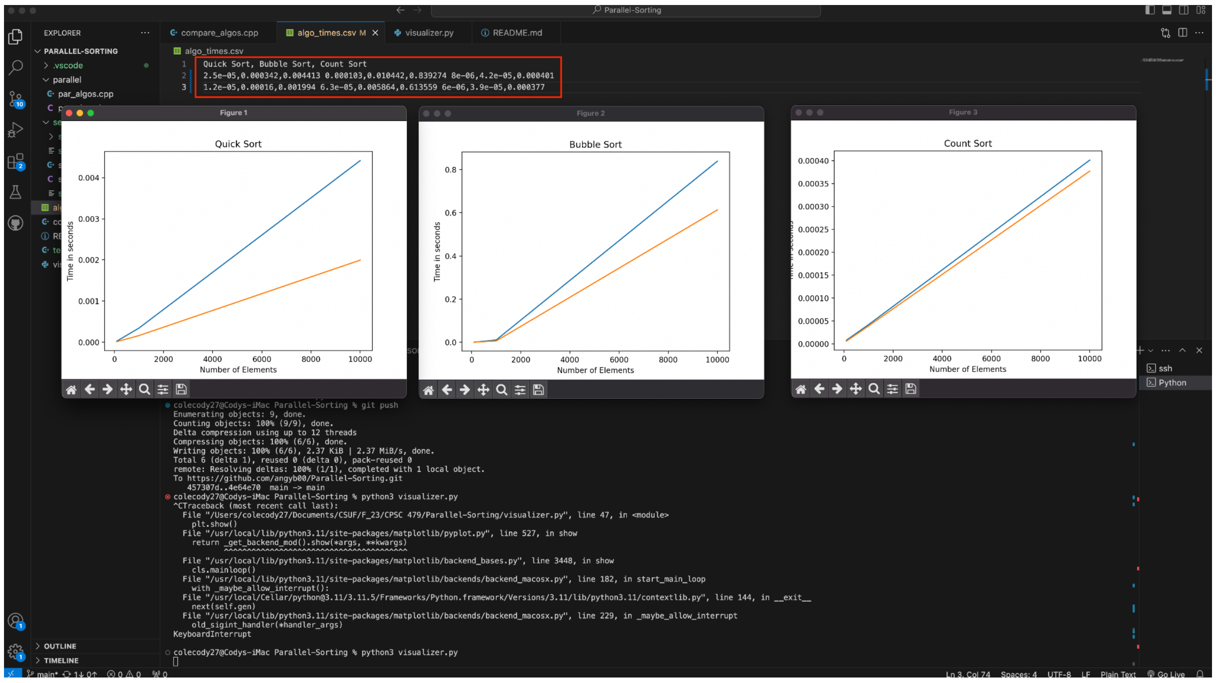
```
algo_times.csv
Quick Sort, Bubble Sort, Count Sort
2.5e-05,0.000327,0.004481 0.000106,0.01058,0.839851 8e-06,4.6e-05,0.000401
1.3e-05,0.000162,0.001972 6.5e-05,0.005851,0.614541 6e-06,3.9e-05,0.000375
```

```
    cls.mainloop()
  File "/usr/local/lib/python3.11/site-packages/matplotlib/backends/backend_macosx.py", line 182, in start_main_loop
    with _maybe_allow_interrupt():
  File "/usr/local/Cellar/python@3.11/3.11.5/Frameworks/Python.framework/Versions/3.11/lib/python3.11/contextlib.py", line 144, in __exit__
    next(self.gen)
  File "/usr/local/lib/python3.11/site-packages/matplotlib/backends/backend_macosx.py", line 229, in _maybe_allow_interrupt
    old_sigint_handler(*handler_args)
KeyboardInterrupt

colecody27@Codys-iMac Parallel-Sorting % python3 visualizer.py
^CTraceback (most recent call last):
  File "/Users/colecody27/Documents/CSUF/F_23/CPSC 479/Parallel-Sorting/visualizer.py", line 47, in <module>
    plt.show()
  File "/usr/local/lib/python3.11/site-packages/matplotlib/pyplot.py", line 527, in show
    return _get_backend_mod().show(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/site-packages/matplotlib/backend_bases.py", line 3448, in show
    cls.mainloop()
  File "/usr/local/lib/python3.11/site-packages/matplotlib/backends/backend_macosx.py", line 182, in start_main_loop
    with _maybe_allow_interrupt():
  File "/usr/local/Cellar/python@3.11/3.11.5/Frameworks/Python.framework/Versions/3.11/lib/python3.11/contextlib.py", line 144, in __exit__
    next(self.gen)
  File "/usr/local/lib/python3.11/site-packages/matplotlib/backends/backend_macosx.py", line 229, in _maybe_allow_interrupt
    old_sigint_handler(*handler_args)
KeyboardInterrupt

colecody27@Codys-iMac Parallel-Sorting % python3 visualizer.py
```

Execution 2



```
algo_times.csv
Quick Sort, Bubble Sort, Count Sort
2.5e-05,0.000342,0.004413 0.000103,0.010442,0.839274 8e-06,4.2e-05,0.000401
1.2e-05,0.00016,0.001994 6.3e-05,0.005864,0.613559 6e-06,3.9e-05,0.000377
```

```
colecody27@Codys-iMac Parallel-Sorting % git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 12 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 2.37 KiB | 2.37 MiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/angyb00/Parallel-Sorting.git
   457307d..4e64e70  main -> main
colecody27@Codys-iMac Parallel-Sorting % python3 visualizer.py
^CTraceback (most recent call last):
  File "/Users/colecody27/Documents/CSUF/F_23/CPSC 479/Parallel-Sorting/visualizer.py", line 47, in <module>
    plt.show()
  File "/usr/local/lib/python3.11/site-packages/matplotlib/pyplot.py", line 527, in show
    return _get_backend_mod().show(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/site-packages/matplotlib/backend_bases.py", line 3448, in show
    cls.mainloop()
  File "/usr/local/lib/python3.11/site-packages/matplotlib/backends/backend_macosx.py", line 182, in start_main_loop
    with _maybe_allow_interrupt():
  File "/usr/local/Cellar/python@3.11/3.11.5/Frameworks/Python.framework/Versions/3.11/lib/python3.11/contextlib.py", line 144, in __exit__
    next(self.gen)
  File "/usr/local/lib/python3.11/site-packages/matplotlib/backends/backend_macosx.py", line 229, in _maybe_allow_interrupt
    old_sigint_handler(*handler_args)
KeyboardInterrupt

colecody27@Codys-iMac Parallel-Sorting % python3 visualizer.py
```