

ECE368 Programming Assignment #4

Due Tuesday, March 27, 2018, 11:59pm

Description

This project is to be completed on your own. You will implement the following two sorting algorithms:

- Quicksort
- Mergesort

The goal here is to have efficient implementations of the abovementioned two algorithms.

Functions you will have to write and submit:

All mentioned functions and their support functions, if any, must reside in the program module `sorting.c`. You should declare these two functions in `sorting.h`.

```
void Quick_Sort(long *Array, int Size)
```

```
void Merge_Sort(long *Array, int Size)
```

These functions take in an Array of long integers and sort them. Size specifies the number of integers to be sorted.

You may declare and define other help functions in `sorting.c`. However, these help functions should not be declared in `sorting.h`. It is best that these helper functions be declared as `static`. Do not name these help functions with a prefix of two underscores “_”.

Other functions you may want to write:

The first two functions `Load_Into_Array` and `Save_From_Array`, are not for sorting, but are needed to transfer the integers to be sorted to and from the files. You may reuse your code from Programming Assignment 2.

```
long *Load_Into_Array(char *Filename, int *Size)
```

The size of the binary file whose name is stored in the char array pointed to by `Filename` should determine the number of long integers in the file. The size of the **binary** file should be a multiple of `sizeof(long)`. You should allocate sufficient memory to store all long integers in the file into an array and assign to `*Size` the number of integers you have in the array. The function should return the address of the memory allocated for the long integers.

You may assume that all input files that we will use to evaluate your code will be of the correct format.

Note that we will not give you an input file that stores more than `INT_MAX` long integers (see `limits.h` for `INT_MAX`). If the input file is empty, an array of size 0 should still be created and `*Size` be assigned 0.

```
int Save_From_Array(char *Filename, long *Array, int Size)
```

The function saves Array to an external file specified by Filename in **binary format**. The output file and the input file have the same format. The integer returned should be the number of long integers in the Array that have been successfully saved into the file.

If the size of the array is 0, an empty output file should be created.

You will have to write a main function in `sorting_main.c` so that you can test and evaluate the performance of your implementations of these sorting algorithms. You should model your main function after the main function written in Programming Assignment 2. Use the option “-q” to invoke quicksort and “-m” to invoke mergesort.

Report you will have to write:

You should write a 1-page report that contains the following items:

- A concise description of the optimization that you have included in your code. A statement such as “we use the median-of-three as the pivot” for example, would be sufficient.
- A tabulation of the run-times obtained from running your code on some sample input files.
- A summary of the space complexity of your sorting routines, i.e., the complexity of the additional memory required by your routines.

Your report should be in PDF or plain text format.

Submission:

The project requires the submission (electronically) of `sorting.h`, `sorting.c`, and the report through Blackboard. You should create a zip file called `proj4.zip` that contains all of the above and submit the zip file. For example, you can create a zip file with the following command:

```
zip proj4.zip sorting.c sorting.h report.pdf
```

If you want to submit the report through the textbox in Blackboard, you may create a zip file as follows:

```
zip proj4.zip sorting.c sorting.h
```

Grading:

The grade depends on the correctness of your program, the efficiency of your program, the clarity of your program documentation and report.

The report will account for 10% of the entire grade. The quicksort function will account for 45% of the entire grade and the mergesort function will account for 45% of the entire grade. A function is deemed correct only if it can complete sorting within a pre-specified time limit.

Bonus points will be awarded to submissions that are among the fastest. Up to 10 points will be awarded for the quicksort function and up to 10 points will be awarded for the mergesort function.

Given:

We provide an executable to generate input files for the evaluation of your implementations. The executable is called `generate`. The executable is compiled on `ecegrid`. Therefore, you have to save it on `ecegrid` and run it on `ecegrid`. You may have to change the mode of the file so that you can run it. To make it an executable,

```
chmod u+x generate
```

To run the program,

```
./generate size filename
```

where `size` is an integer specifying how many long integers you want, and `filename` is the name of the file to store these integers. For example,

```
./generate 1000000 1M.b
```

generates a file with 1 million long integers and stores them in `1M.b`.

When you are working on your program, please use the following command to compile your program (assuming that you have `sorting.c`, `sorting.h`, and `sorting_main.c`). The flag `-g` allows you to debug your program.

```
gcc -g -std=c99 -Wall -Wshadow -Wvla -pedantic sorting.c sorting_main.c -o proj4
```

When we evaluate your submission, we will use the following command to compile your program:

```
gcc -std=c99 -Wall -Wshadow -Wvla -pedantic -O3 sorting.c sorting_main.c -o proj4
```

We have dropped the `-g` flag and added an optimization flag `-O3`. Note that it is upper-case letter `O` (not number zero). After you have debugged your program, you should compile your program with the optimization option and you should observe a difference in the run-time of your program.

We will call the program to perform quicksort as follows:

```
./proj4 -q input.b output.b
```

We will call the program to perform mergesort as follows:

```
./proj4 -m input.b output.b
```

Note that you do not have to submit `sorting_main.c`. We will provide the `sorting_main.c` file in our evaluation of your submission.

Getting started:

Copy over the files from the Blackboard website. Check out the Blackboard webpage for any updates to these instructions. *Start sorting!*