Programming Assignment #1: Multi-Threading Mergesort

Class: COEN 346 Section Y
Submitted on 11 February 2025

Names & IDs:
- Clarence Zhen (40166293)
- David Makary (40189198)
- Justin Ma

High-Level Description of the Assignment’s Objectives and Code

In this assignment, we are tasked to create a `python` program that sorts the contents of a file called `input.txt` which contains a series of numbers (`int` or `float`) separated by newlines `\n`. We must use multithreading techniques learned in class and during the lab session and the mergesort algorithm to sort the numbers. Finally, we will output a log describing how each thread has been spawned and how each has contributed to the sorting process. The output must be written into a text file called `output.txt`.

`main.py`

Contains the mergesort algorithm and miscellaneous functions that aid in formatting numbers, reading `input.txt`, and creating binary numbers. There are two versions of the `mergeSort` function defined in this file: One is with multithreading that requires a `list[Thread]` and the other is just the regular algorithm shown in the lab session.

The way how multithreading is achieved is by creating new threads wherever `mergeSort` is being called. That also includes recursive calls as well. When a thread is created, it will be appended to the `list[Thread]` for further control if needed. However, **the appended threads are destroyed once the recursive call has finished**. So, only one thread remains at the end of the sort. Joining the threads to the parent became impossible in this case due to this phenomenon. (This can be attributed to how `Thread` was created)

`thread.py`

A class that inherits the properties and functions of `threading.thread` class. This might be overdone for this assignment as a result of experimenting with multithreading. Moreover, it includes a logger that outputs its contents into the `output.txt`.

Conclusion

In this programming assignment, we were able to create and make use of multiple threads to improve the performance of an algorithm. Although the performance benefit may not show with such a small subset of numbers given by the assignment’s example, as the input size gets bigger, we could expect the performance to be better.

Contributions

Clarence Zhen (40166293)	Creation of `threads.py`
David Makary (40189198)	Modifying the original mergesort to a multithreaded mergesort algorithm
Justin Ma	Creation of miscellaneous functions