Lab 3: Sort - CS50x 2022

cs50.harvard.edu/x/2022/labs/3/

Lab 3: Sort

You are welcome to collaborate with one or two classmates on this lab, though it is expected that every student in any such group contribute equally to the lab.

Analyze three sorting programs to determine which algorithms they use.

Background

Recall from lecture that we saw a few algorithms for sorting a sequence of numbers: selection sort, bubble sort, and merge sort.

- Selection sort iterates through the unsorted portions of a list, selecting the smallest element each time and moving it to its correct location.
- Bubble sort compares pairs of adjacent values one at a time and swaps them if they are in the incorrect order. This continues until the list is sorted.
- Merge sort recursively divides the list into two repeatedly and then merges the smaller lists back into a larger one in the correct order.

Getting Started

Started CS50x in 2021 or prior and need to migrate your work from CS50 IDE to the new VS Code codespace? Be sure to check out our instructions on how to migrate your files!

Open VS Code.

Start by clicking inside your terminal window, then execute `cd` by itself. You should find that its “prompt” resembles the below.

```
$
```

Click inside of that terminal window and then execute

```
wget https://cdn.cs50.net/2021/fall/labs/3/sort.zip
```

followed by Enter in order to download a ZIP called `sort.zip` in your codespace. Take care not to overlook the space between `wget` and the following URL, or any other character for that matter!

Now execute

```
unzip sort.zip
```

to create a folder called `sort`. You no longer need the ZIP file, so you can execute

```
rm sort.zip
```

and respond with “y” followed by Enter at the prompt to remove the ZIP file you downloaded.

Now type

```
cd sort
```

followed by Enter to move yourself into (i.e., open) that directory. Your prompt should now resemble the below.

```
sort/ $
```

If all was successful, you should execute

```
ls
```

and you should see a collection of `.txt` files alongside executable programs `sort1`, `sort2`, and `sort3`.

If you run into any trouble, follow these same steps steps again and see if you can determine where you went wrong!

Instructions

Provided to you are three already-compiled C programs, `sort1`, `sort2`, and `sort3`. Each of these programs implements a different sorting algorithm: selection sort, bubble sort, or merge sort (though not necessarily in that order!). Your task is to determine which sorting algorithm is used by each file.

- `sort1`, `sort2`, and `sort3` are binary files, so you won’t be able to view the C source code for each. To assess which sort implements which algorithm, run the sorts on different lists of values.
- Multiple `.txt` files are provided to you. These files contain `n` lines of values, either reversed, shuffled, or sorted.
    For example, `reversed10000.txt` contains 10000 lines of numbers that are reversed from `10000`, while `random10000.txt` contains 10000 lines of numbers that are in random order.
- To run the sorts on the text files, in the terminal, run `./[program_name] [text_file.txt]`. Make sure you have made use of `cd` to move into the `sort` directory!
    For example, to sort `reversed10000.txt` with `sort1`, run `./sort1 reversed10000.txt`.

- You may find it helpful to time your sorts. To do so, run `time ./[sort_file] [text_file.txt]`.
    For example, you could run `time ./sort1 reversed10000.txt` to run `sort1` on 10,000 reversed numbers. At the end of your terminal's output, you can look at the `real` time to see how much time actually elapsed while running the program.
- Record your answers in `answers.txt`, along with an explanation for each program, by filling in the blanks marked `TODO`.

Walkthrough

This video was recorded when the course was still using CS50 IDE for writing code. Though the interface may look different from your codespace, the behavior of the two environments should be largely similar!

Hints

> The different types of `.txt` files may help you determine which sort is which. Consider how each algorithm performs with an already sorted list. How about a reversed list? Or shuffled list? It may help to work through a smaller list of each type and walk through each sorting process.

▶ Not sure how to solve?

How to Check Your Answers

Execute the below to evaluate the correctness of your answers using `check50`. But be sure to fill in your explanations as well, which `check50` won't check here!

```
check50 cs50/labs/2022/x/sort
```

How to Submit

In your terminal, execute the below to submit your work.

```
submit50 cs50/labs/2022/x/sort
```