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

1. Log into ide.cs50.io (https://ide.cs50.io/) using your GitHub account.
2. In your terminal window, run `wget https://cdn.cs50.net/2020/fall/labs/3/lab3.zip` to download a Zip file of the lab distribution code.
3. In your terminal window, run `unzip lab3.zip` to unzip (i.e., decompress) that Zip file.
4. In your terminal window, run `cd lab3` to change directories into your `lab3` directory.

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 `random100000.txt` contains 100000 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]`.
  - 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`.

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.

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/2020/fall/sort
```

How to Submit

1. Download just your `answers.txt` file by control-clicking or right-clicking on the file in CS50 IDE's file browser and choosing **Download**.
2. Go to CS50's Gradescope page (https://www.gradescope.com/courses/157004).
3. Click "Lab 3: Sort".
4. Drag and drop your `answers.txt` file to the area that says "Drag & Drop". Be sure it has the correct filename!
5. Click "Upload".

You should see a message that says "Lab 3: Sort submitted successfully!"