This is CS50

CS50's Introduction to Computer Science

OpenCourseWare

Donate (https://cs50.harvard.edu/donate)

David J. Malan (https://cs.harvard.edu/malan/)
malan@harvard.edu
(https://www.facebook.com/dmalan) (https://github.com/dmalan)
(https://www.instagram.com/davidjmalan/) (https://www.linkedin.com/in/malan/)
(https://www.reddit.com/user/davidjmalan)
(https://www.threads.net/@davidjmalan) (https://twitter.com/davidjmalan)

Sort

Problem to Solve

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.

In this problem, you'll analyze three (compiled!) sorting programs to determine which algorithms they use. In a file called `answers.txt` in a folder called `sort`, record your answers, along with an explanation for each program, by filling in the blanks marked `TODO`.

Distribution Code

For this problem, you'll need some "distribution code"—that is, code written by CS50's staff. Provided to you are three already-compiled C programs, `sort1`, `sort2`, and `sort3`, as well as

several `.txt` files for input and another file, `answers.txt`, in which to write your answers. Each of `sort1`, `sort2`, and `sort3` 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. Start by downloading these files.

▶ **Download distribution files**

Hints

Click the below toggles to read some advice!

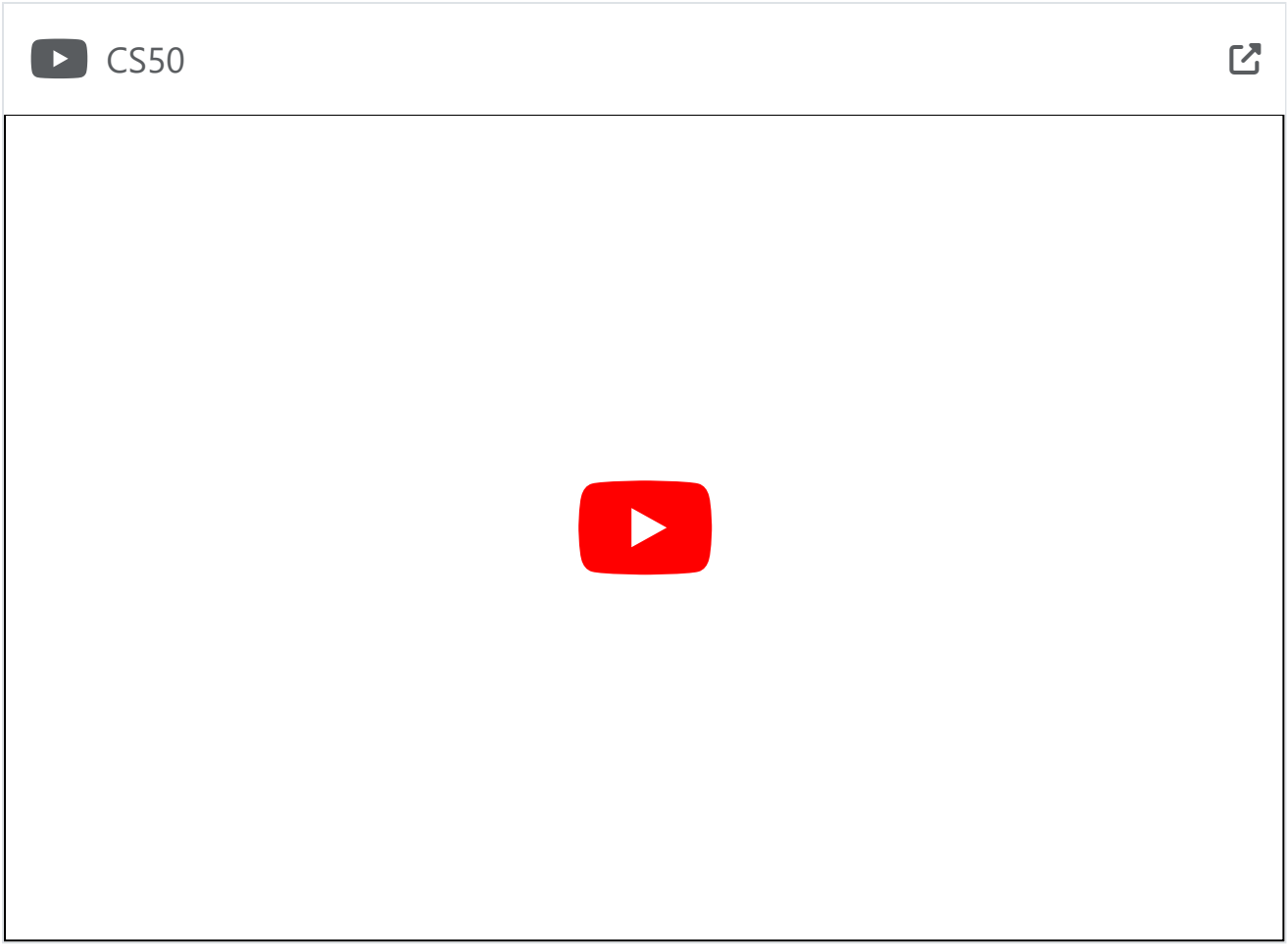
▼ **Explore the `.txt` files**

- 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 `random50000.txt` contains 50000 lines of numbers that are in random order.
- 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.

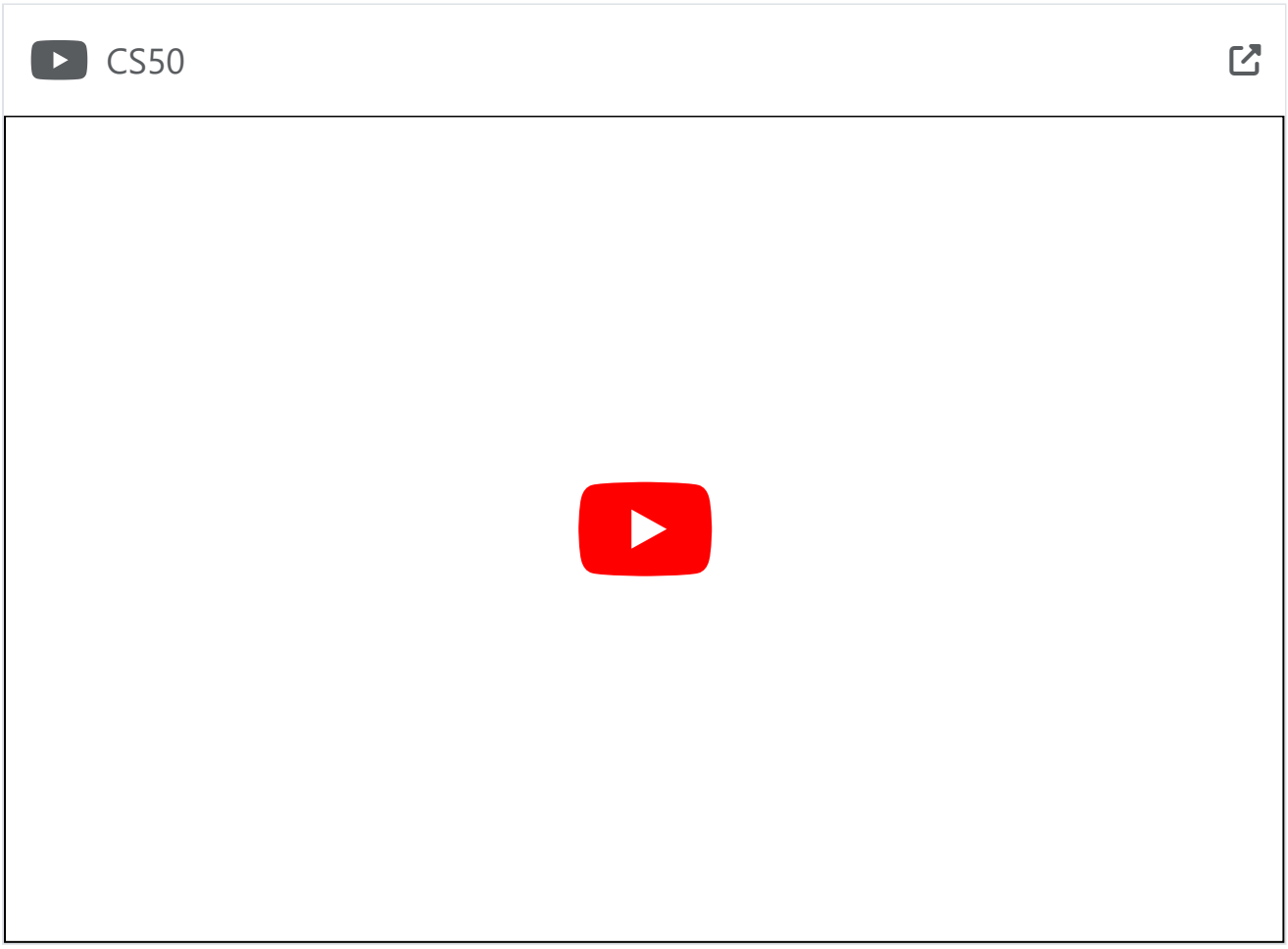
▼ **Time each sort with different inputs**

- 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.

Walkthrough



▼ **Not sure how to solve?**



How to Test

Correctness

```
check50 cs50/problems/2024/x/sort
```

How to Submit

```
submit50 cs50/problems/2024/x/sort
```