Mod 7 Lab: Which sorting algorithm is which?

Description

Professor Jake has goofed: he implemented 5 sorting algorithms (bubblesort, selectionsort, insertionsort, mergesort, and quicksort), but labeled the algorithms `alg_a`, `alg_b`, `alg_c`, `alg_d`, and `alg_e`.

He wrote these algorithms in C and lost the source code, so now he doesn't know which algorithm is which! All he has is the executable `time_sorts.out`, which uses the following input and output:

- input
  - a file containing numbers to be sorted
- output
  - a table tabulating how long each sorting algorithm took to run

Can you help the professor firgure out which algorithm corresponds to which sorting technique?

Generating lists

You will need to generate lists of varying lengths and patterns to decipher which algorithm corresponds to which sort. The file `generate_numbers.py` can help. By default, it generates 10,000 random integers and saves them in a file.

```
$ python3 ./generate_numbers.py # creates unsorted_10000.txt
$ ls                            # list files in working directory
answers.py  generate_numbers.py  time_sorts.out  unsorted_10000.txt
$
```

Note the use of `python3` to run a python script in Mimir.

Feel free to modify `generate_numbers.py` as needed.

The maximum number of items `time_sorts` can sort is 100,000.

Timing Sorts

The file `time_sorts.out` takes a file of numbers as an input and outputs a table of running times for each algorithm. Note that this is a compiled file: it will only run on Mimir, because it contains instructions specific to the software and hardware Mimir uses to execute code (Ubuntu on an AWS server). We call files like this "non-portable": they cannot be trivially ported between different computers and operating systems, unlike Python code.

These algorithms will likely be faster than the algorithms you have written in Python: non-portable code can be optimized for the machine it is running on.

First, we must edit the permissions of `time_sorts.out` with the command `chmod` so we can execute the file:

```
$ chmod 555 ./time_sorts.out
$
```

`chmod` aside

You can skip this section - it is provided for those curious about file permissions in Unix.

From the system perspective, there are three types of users who may try to access this file:

- **u** - the **u**ser who owns the file
- **g** - members of the user's **g**roup
- **o** - **o**ther users on this computer

The three numbers 555 specify the permission levels for each group (all the same in this case). Each number represents three binary flags - permission to **r**ead, **w**rite, and e**x**ecute a file. **w**rite access tends to be the most dangerous, but we can give the **u**ser write permission by running `chmod 755`:

- **u** permission - 7
  - binary - 111 (r/w/x)
  - **u**ser can **r**ead, **w**rite, and e**x**ecute this file
- **g** permission - 5
  - binary - 101 (r/w/x)
  - **g**roup can **r**ead and e**x**ecute this file
- **o** permission - 5
  - binary - 101 (r/w/x)
  - **o**thers can **r**ead and e**x**ecute this file

Back to `time_sorts`

Now, we can time the algorithms on a specific file by running `time_sorts.out` with that file as a parameter:

```
$ ./time_sorts.out unsorted_10000.txt
====================
n = 10000
--------------------
alg_a   0.091 s
alg_b   0.002 s
alg_c   0.002 s
alg_d   0.378 s
alg_e   0.161 s
--------------------
```

Your turn! Figure out which algorithm is which by feeding them different inputs. Some things to consider:

- Are the algorithms quadratic or nlogn for random numbers?
- What kinds of input that should provide best and worst cases for each algorithms?

Lab Format

You must complete this assignment in the Mimir IDE (available on the page where we typically download starter code). This assignment includes pre-compiled executable code (`time_sorts.out`) that will only run on Mimir - it will not run properly on your local computer.

`time_sorts.out` may become corrupted as you work on this problem. If it stops working correctly, you can reload the original starter code by going to **Mimir > Load Starter Code** in the Mimir IDE.

Submitting

Submit your answers as a dictionary in the file `answers.py`. For intance, if you think all 5 algorithms are bubble sort, you should include the following dictionary in `answers.py`:

```
answers = { 'alg_a': 'bubble',
            'alg_b': 'bubble',
            'alg_c': 'bubble',
            'alg_d': 'bubble',
            'alg_e': 'bubble'}
```

The file `answers.py` in the starter code contains more information and an automatic checker to make sure you use valid strings for the sorting algorithm names.

Students must submit to Mimir **individually** by the due date (typically, two days after lab at 11:59 pm EST) to receive credit.

Grading

Your grade will be based on the number of correct algorithms you identify: 20 points per correct algorithm.

Feedback

If you have any feedback on this assignment, please leave it here.

We check this feedback regularly. It has resulted in:

- A simplified, clear **Submitting** section on all assignments
- A simplified, clear **Grading** section on all assignments
- Clearer instructions on several assignments (particularly in the recursion module)