

Sort Test Draft

Shukry Zablah

[2019-02-14 Thu]

Contents

1 Sort Test (Performance Evaluation Project 1)	1
1.1 Sample Use	1
1.2 Supported Algorithms	3
1.3 Todos [5/6]	4
1.4 Helpful Links	4

1 Sort Test (Performance Evaluation Project 1)

1.1 Sample Use

We provide a command line utility to run a variety of sorting algorithms with a variety of options.

To compile we use the following command:

```
gcc sorttest.c -o sorttest -lrt
```

Note that we need to link the `librt.a` library when we compile a program that uses the `clock_gettime` function.

Suppose we want to time the `merge_sort` algorithm on an array of 1000 random values for values under ten thousand, and we want to repeat this 10 times.

We can run the compiled executable in the command line as follows:

```
./sorttest merge 1000 10000 10 TRUE
```

```
merge,1000,10000,10,606291
merge,1000,10000,10,459376
merge,1000,10000,10,487524
```

```
merge,1000,10000,10,463135
merge,1000,10000,10,459790
merge,1000,10000,10,457852
merge,1000,10000,10,464574
merge,1000,10000,10,480770
merge,1000,10000,10,468019
merge,1000,10000,10,459385
Correctly Sorted: 10/10
```

The program prints the algorithm, the length of the input, the range of the values, the number of repetitions, and the elapsed time in nanoseconds in that order. The output is in a csv format.

We used the option `TRUE` in our command in order to run a function that verifies the array is sorted. Normally you would use the `FALSE` option to omit this step.

In order to automate our data collection we create a small bash script called `driver.sh` that contains the commands we will run. (Remember to use `chmod +x <file>` to be able to execute scripts).

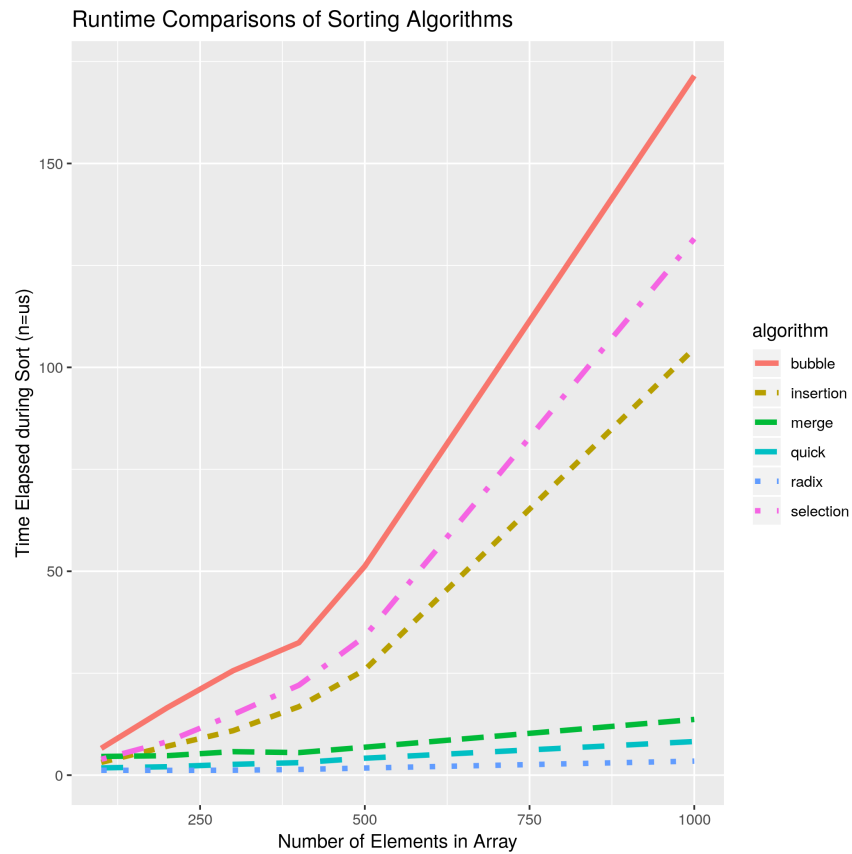
In order to create a csv file that contains all of our collected data we redirect the output of the program into the `test.csv` file.

```
./driver.sh > test.csv
```

After collecting our data we use an R script in order to generate a plot of the elapsed times. Again, we allow the script to be executable for convenience.

```
./analyze.R
```

Now we can check the current directory in order to retrieve the image of our plot.



Remember that we can change:

- the `sorttest.c` file to change the behavior of the timing programs and the way data is created
- the `driver.sh` file to specify `sorttest` input to decide what data to collect
- the `analyze.R` to create different plots and manipulate data

1.2 Supported Algorithms

- bubble
- counting **not working currently**
- insertion
- merge

- quick
- radix
- selection

1.3 Todos [5/6]

- ☒ add initial support for choosing algo, num values, and range
- ☒ support for repetitions (right now the reps arg doesn't matter)
- ☒ support for verification of sorting (as an additional arg)
- ☒ use high precision methods (`clock_gettime`)
- ☒ update readme for sample workflow with bash and R
- ☐ fix counting sort not working

1.4 Helpful Links

<https://www.cs.rutgers.edu/~pxk/416/notes/c-tutorials/gettime.html>