Lab 2, Medians

Due date: Monday, October 9, 11:59pm.

Lab Assignment

Assignment Preparation

This lab is also a pair programming lab due to the number of algorithms you need to

The Task

For this assignment, implement and test the running time of the following algorithms for finding the median

- The sort-based algorithm for finding the median. Implement any fast sorting algorithm from scratch (do **not** use existing library implementations), and use it to compute the median.
- The Randomized Median algorithm. This algorithm picks *some* number to split the array on in O(1) time.
- Three versions of the Fast Median algorithm which splits the array into $\frac{n}{k}$ chunks of k elements each, finds the median of each chunk and chooses the median of these medians as the pivot point. Implement this algorithm for the following values of k:
  - $k = 3$
  - $k = 5$ (this is the algorithm we discussed in class)
  - $k = 7$ (this is actually the originally proposed fast median computation algorithm)

In implementing these algorithms you are only allowed to use the following language features:

- `Integer` variables
- One-dimensional `Integer` arrays (you can use Python's NumPy arrays instead of lists)
- Comparison operations.
- Arithmetic operations.
- Control structures of your programming language (loops and conditionals) as well as function calls and user-defined functions.

You are **not allowed** to use **any vector arithmetics** operations[1].

Note: You are implementing the algorithms to test them side-by-side in their form that most resembles our pseudo-code. The restrictions above are designed to achieve just that. You goalr is to time your implementations, not Java's or Python's implementations of these operations.

In addition, implement a **testing harness** that allows you to collect the information about the run time of your implementation. The testing harness, at a minimum, shall consist of the following:

- A procedure for generating a random array of length n.
- An ability to archive any generated arrays (this is needed to make sure that your algorithms are tested on the same set of matrix multiplication operations).
- An ability to time the execution of each of the median finding algorithms you implement.
- The actual harness: a procedure that takes as input a set of *experimental parameters*, performs the desired run-time experiment (see below), and collects the run-time data.
- Archiving of the results: the ability to save the results of the experiment in a way that avails them for further analysis (hint: CSV files work well).

Running Experiments

With the implementations of the Median Finding algorithms, and with the test harness you build to test them, you will stage a run-time experiment designed to understand how the performance of the algorithms changes with the increase in the input size.

The experiment you will be staging has the following design.

[1] They already implement the algorithms you want to build.

1. Array sizes. A list of array sizes organized similarly to the list of matrix sizes you used in `Lab 1`. No need to limit array sizes to powers of 2 though.

2. Repetitions. Because some of the algorithms work in randomized way you need a lot of repetitions per size to actually understand the correct run-time behavior of your implementations. The repeat parameter will control for that.

3. Measurement. For each array size and for each repetition step, generate an array of the appropriate size and compute find the median using all the methods. Aggregate the results by computing the average running time, and standard deviation of the running time for each of the implementations.

4. Reporting. For each algorithm you should obtain a set of average runtimes for each input size considered. Plot these runtime numbers. If possible, fit a curve to them to see if they follow the expected algorithm run-times.

5. Analysis. Observe the results. Answer the following questions.

- What is the observed runtime behavior for each algorith?
- Which algorithm is the fastest in your experiment? Which is the slowest?
- What are the asymptotic trends for each algorithm?

Specifically, we are interested in seeing how the fast median algorithm for $k = 3$ behaves.

6. Report. Prepare a properly formatted, and professionally presentable report describing your work in this lab. The report shall contain the precise description of the experiment your team has implemented, it shall contain the experimental results both in tabular and in graphical form, and it shall contain your analysis. **Both the obtained results, and the quality of writing will be graded!**

Note. Your test harness shall support the steps 1-3 described above.

Deliverables and Submission

You have two main deliverables: your code, and the report. Additionally, submit a simply `README` file.

README. Your `README` file shall contain the list of students in the team, as well as information about your implementation: language chosen, and the insturctions for how to run your implementation: both in order to perform a single matrix multiplication operation, and in order to run your test harness.

Code. Submit all your code in a `lab02.zip` or `lab02.tar.gz` archive.

Report. Submit your report as a PDF document named `lab02-report.pdf`

Note. Additional instructions concefning the internals of your implementation (which would make it possible for us to test your work automatically) will be released on Wednesday, October 4.

Use `handin` to submit all your deliverables

```
$ handin dekhtyar lab02 <files>
```

Good Luck!