Homework 11 - Efficiency

Bookmark this page

In this assignment, you are asked to improve the execution time of a Java program that attempts to detect plagiarism in a corpus of documents.

In completing this assignment, you will:

- Apply the techniques discussed in the lessons to improve the efficiency of code
- Get more experience understanding and modifying existing code
- Enhance your understanding of the behavior and efficiency of various data structures

Background

Plagiarism detection is a very difficult problem to solve, but a simple approach is to just look for common words and phrases between documents. If two (or more) documents contain many of the same phrases, then there is a good possibility that one author copied from the other.

The program you will improve detects common phrases of size *windowSize* in a corpus of documents, and then report pairs of documents for which the number of common phrases is greater than some *threshold*, sorted by the number of common phrases.

Getting Started

Download the **PlagiarismDetector.java** source code.

The *detectPlagiarism* method takes the name of the directory containing the corpus of documents, as well as the *windowSize* and *threshold* parameters, and returns a Map that lists the pairs of documents, with the Map keys sorted by the number of matches. If two or more pairs have the same number of matches, the order in which they are stored is not specified.

Aside from the *detectPlagiarism* method, the *PlagiarismDetector* also has helper methods to:

- Read a file and store its contents in a List of Strings
- Create the distinct phrases, each of which is of size *windowSize*
- Find the common phrases between two documents
- Sort the results

There is also a *main* method that you can use to run the *detectPlagiarism* method (be sure to specify the directory containing the corpus), which then reports the execution time in seconds. Note that this measures the "wall clock" or actual elapsed time, and not the time that the program is actually running on the CPU, but assuming you are the only person using the computer, and there are not too many other processes running in the background, it should give you a good idea of the execution time of the code.

You can, of course, create your own corpus as you modify this program, but we will evaluate the performance of your program using the corpus we provided in **corpus.zip** with *windowSize* of 4 and *threshold* of 5.

A modified version of this problem (and the corpus of documents) was originally used at the University of Chicago and presented as a Nifty Assignment at SIGCSE 2008 (http://nifty.stanford.edu/2008/franke-catch-plagiarists/), and the implementation was modified by the PennX instruction staff. We will assume that this implementation is correct for our purposes.

Activity

Your goal in this assignment is to improve the execution time of the *detectPlagiarism* method and the methods it uses, using the various techniques seen in the lessons.

In particular, you should consider:

- Does the code use the correct data structures? If it's using a List, maybe it should use a Set, or the other way around
- Does the code use efficient algorithms? Perhaps there are quicker ways to accomplish the same things
- Does the code do unnecessary work? Perhaps some code can be rewritten or even removed if it is doing work that's already been done

Keep in mind that you are asked to focus only on improving the execution time of the code. It is okay if your changes have a negative effect on things like memory usage or other aspects of quality, *except* for correctness, of course: your changes must not change the output of the code!

Please do not change the signature of the *detectPlagiarism* method. You may, however, modify or even remove any of the other methods as you see fit, as long as the code still works correctly. Likewise, you may add new methods or classes, though if you add new classes, please add them to *PlagiarismDetector.java* and do not create new .java files. Please be sure that your *PlagiarismDetector* class is in the default package (i.e., that there is no "package" declaration at the top), and -- perhaps this goes without saying -- please implement all code in Java. :-)

In making changes to the code, you may **not** assume that:

- The program will always use the same set of documents that was provided to you, so you cannot pre-compute results
- The number and size of documents in the corpus will always be the same as the one that was provided to you, so you cannot hardcode values
- The *windowSize* and *threshold* will always have the values specified above
- The code is always executed on a multi-core/multi-processor machine, so you can't know for certain that using threads will help

In some cases, you may need to make a decision that is somewhat dictated by the input, e.g. it may be better to use one data structure for small window sizes and a different one for large ones. In such cases, choose the one that works best for the input values specified above since that is what we will use to measure the execution time of your code.

Helpful Hints

We do **not** recommend that you try to make a lot of changes to the code and then hope that (a) it's faster and (b) it still works.

Remember one of the rules of thumb: "measure, don't guess." In addition to testing the overall execution time using the *main* method we provided, write some additional code that measures the execution time (e.g., for individual methods or parts of the code) and then check that each change you make is making the code faster.

But also make sure that the output is still the same! Check the contents of the Map and be sure that you haven't accidentally introduced any bugs.

You might want to consider making backup copies of the code as you modify it, in case you accidentally break it and/or make it slower. Better yet, use a version control system such as Git/GitHub so you can go back to previous versions of your code if necessary.

Before You Submit

Please be sure that:

- your *PlagiarismDetector* class is in the default package, i.e. there is no "package" declaration at the top of the source code
- your *PlagiarismDetector* class compiles and you have not changed the signature of the *detectPlagiarism* method
- you have not created any additional .java files

Assessment

Part of the challenge of this assignment is knowing when you can stop improving the code. Your new implementation will be considered "correct" when the new execution time is less than or equal to 40% of the original execution time.

Execution time depends greatly on the CPU, operating system, other processes, etc. so you may notice fluctuation in the execution times as you are evaluating your code. For this assignment, we will assess your submission based on the improvement measured on the Codio grading platform that we have used for other assignments. You can access it by clicking the "Begin Submission" button below. You may notice that the improvement on Codio is different than it is on your computer, but we will standardize on that platform for grading this assignment.

Of course, your code must still produce the same output as the original code, regardless of the values of the arguments to the *detectPlagiarism* method. You will be given a score of 0 on this assignment if the output is incorrect with respect to the original implementation, no matter how fast the code runs.

If the execution time of your code on the Codio platform is less than 40% of the original execution time, you will receive a score of 100 for this assignment. If your execution time is greater than 40%, you will lose two points for every 1% over 40%. For instance, you will receive 98 if your execution time is 41% of the original, 96 if it is 42%, 90 if it is 45%, and so on.

On Codio, the original code should take around 95 seconds to complete, so your code should take a little under 40 seconds on Codio in order to earn a score of 100 on this assignment. We strongly recommend that you perform the "Run Autograder" step on Codio to measure the execution time and correctness of your code before submitting it. See the "Submitting this assignment" instructions on the Codio platform after you click the "Begin Submission" button below.

Please note that your code must finish running in under 60 seconds on the Codio platform in order to receive a grade. If it takes more than 60 seconds to complete, the Codio autograder will time out and you will not receive a score for this assignment. So be sure to check the execution time on Codio before completing your submission.

Unlike other assignments, you may not get your score right away for this assignment. Please be patient as it may take 2-3 minutes for your score to appear in Codio and edX.

Homework #11 Submission (External resource)

(100.0 points possible)

This link will take you to Codio so that you may submit the assignment.

BEGIN SUBMISSION

English

Download on the App Store

ANDROID APP ON Google play

POWERED BY OPEN edX