Software Plagiarism Detection (软件剽窃检测)

No matter you admit it or not, there has always been the battle between the instructor and the students. No matter you admit it or not, there has always been the battle between the good side of your heart and the bad side of your heart. Without monitoring and punishment, any person may be tempted to do something wrong. Therefore, all of us should be careful when alone. In Chinese, it is called "君子必慎其独" ("the superior man must be watchful over himself when he is alone"). You may argue that, "If no one knows, why do I need to supervise myself to be good?" Ok…, maybe no one knows, maybe there is someone who DOES know and make records, maybe it's he, maybe it's she, maybe it's YOU…

Come back to our topic on the battle between instructors and students. In programming courses, some students tend to copy others' source code and submit with little modification, which makes instructors much worried. Finally, smart instructors found a solution to fight back: check the plagiarism in programs automatically using another program. This even led to a new research area with software [1] and excellent research papers [2]. Although we do not have time to finish plagiarism detection software during our course, we can have a taste of the multiple pattern searching algorithm used for plagiarism detection. Rabin–Karp algorithm or Karp–Rabin algorithm is suitable for multiple pattern searching by hash function and thus suitable for detecting plagiarism [2][3]. From previous exercises, we already know the KMP matching algorithm [4].

Requirement:

(1) Implement Karp–Rabin and KMP algorithms;

(2) Pick 3-5 source files from protobuf (you may remove the spaces and tabs for better matching);

(3) Choose at least 10 patterns of 10-20 characters in length;

(4) Search all the patterns in all the source files using Karp–Rabin and KMP algorithms to compare the performance (i.e., report the total runtime of each algorithm), and think about the reason for the different performance of the two algorithms on multiple pattern searching;

(5) Use design patterns to make your program follow the open-closed principle as much as possible. For example, you may use Strategy pattern for easy adoption of new matching algorithms, etc.

[1] http://theory.stanford.edu/~aiken/moss/.

[2] S. Schleimer, D. S. Wilkerson, and A. Aiken, "Winnowing: Local Algorithms for Document Fingerprinting", in *Proc. SIGMOD*, 2003, pp. 76-85.

[3] http://en.wikipedia.org/wiki/Rabin–Karp_string_search_algorithm.

[4] http://en.wikipedia.org/wiki/Knuth–Morris–Pratt_algorithm.

In the end, we need to point out that, even there are battles between instructors and students, the final goal of all is the same: students want to learn something and become strong; instructors want the students to learn something and become strong. So, stop the battle and finish the homework without plagiarism.