

Java Tools to Help Understanding Pattern Matching Techniques *

Christian Charras – Thierry Lecroq

Institut de Recherche sur l'Enseignement des Mathématiques,

and,

Atelier Biologie Informatique Statistiques et Sociolinguistique,

Faculté des Sciences et des Techniques,

Université de Rouen,

76821 Mont Saint-Aignan Cedex, France,

{Christian.Charras,Thierry.Lecroq}@dir.univ-rouen.fr

Abstract

Java is an emerging tool to help traditional educational methods. Two web sites, including Java applets are presented. These sites are dedicated to pattern matching. The first one is intending for theoretical and practical computer scientists and presents exact string matching algorithms. The second one can help both biologists and computer scientists to understand the basic techniques for aligning sequences.

Keywords: Java, Pattern matching, String matching, Internet, Web, Education, Security, Computational biology, Sequence comparison.

*This work was partially supported by the project “Informatique et Génomes” of the french CNRS.

1 Introduction

The emergence of the Internet and the multiplication of Web servers come with publications of document classes on the network.

In computer science the vast majority of the resources on the network are pre-existing documents which are available either in PostScript form or in HTML (Hyper Text Markup Language) form.

For documents in HTML form even if they try to take advantage of the hypertext functionalities, they are often very simple. The algorithms do not get any presentation using the dynamic possibilities of the language.

However the conception of educating software is an important concern among computer scientists. Unfortunately, most studies have been launched before the Internet boom and the arrival of Java. Thus, in many cases, one would have to face too many difficulties to adapt available tools on the network. Furthermore those tools often use too aggressive technologies to be shared, except by Intranet servers.

The generalization of HTML which is now the universal language for the hypertext since the arrival of CSS (Cascading Style Sheets) and the integration of Java to Web clients, gives today the best technical solution to conceive dynamic open documents:

- Java is secured as well as at the static as at the dynamic level. This enables Java applets to be accepted on nearly all sites;
- Java inherits both from modular programming tradition and from object oriented conception;
- Java has a growing number of ready-to-use shared units;
- Up to now the different versions of Java have a good ascending compatibility;
- Nowadays most constructors support Java.

The integration of Java units in Web documents can be done at different levels (see figure 1).

Level 3 avoids the constraints that a Java applet has to communicate only with the machine where the HTTP (Hyper Text Transport Protocol) which emits the code resides. Levels 2 and 3 seem more appealing than level 1 since they enable to incorporate existing applications but they imposed

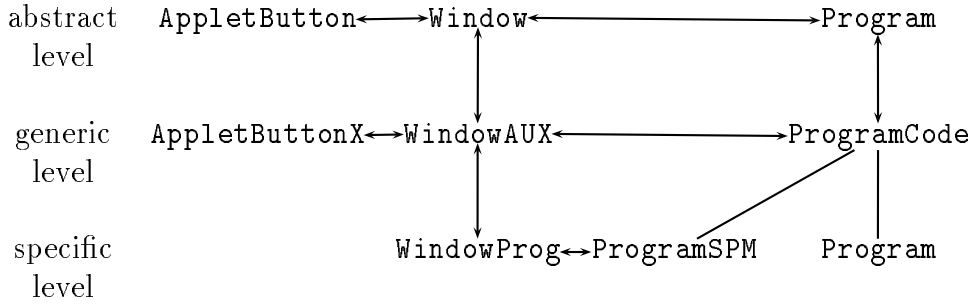


Figure 2: General architecture of the different systems

From one application to another, the algorithms arborescence does not need to be modified.

For “Exact String Matching Algorithms” a class `ProgramSPM` is derived from `ProgramCode` and the classical algorithms are derived from `SPM`. Only very specific algorithms need specific methods.

The GUI level requires more changes depending on the chosen illustration.

2 Description

We developed two web sites dedicated to pattern matching. The first one called “Exact String Matching Algorithms” [2] is intended to computer scientists who want to understand the basic notions of string matching. The second one is called “Sequence Comparison” [3] and is intended to biologists, computer scientists, linguists or mathematicians who want to understand the basic notions for aligning two sequences. Both sites are organized as follows: the different notions are explained and Java applets can be used to illustrate the notions with specific values given by the user. Every user will then be able to choose the set of values that he feels the more valuable to grasp the difficult notions used by the algorithms. We will now describe the two sites.

2.1 Exact String Matching Algorithms

Exact string matching consists in finding all the occurrences of a word x in a text y . A large number of solutions have already been proposed to solve this problem (see [4] or [6]). This various solutions can differ totally and

understanding one does not necessarily help to understand another one. They can use combinatoric or heuristic techniques. Understanding these techniques can help to understand algorithm solving more elaborate problems. It seems interesting to offer a convivial tool to understand them.

In our system the user can choose among the thirty exact string matching algorithms already implemented. He can then enter a text and a word (default text is `gcatcgagagagtagtacg` and default word is `gcagagag` see Figure 3(a)). The text and the word alphabet is restricted to the lower case letters. A button enables the user to start the search and another button to stop it at any time. The search phase is then shown attempt by attempt: for one attempt the text is displayed and the word, which all characters are materialized by a dot, is aligned with the relevant position in the text. In each attempt the different character comparisons are shown in the following way:

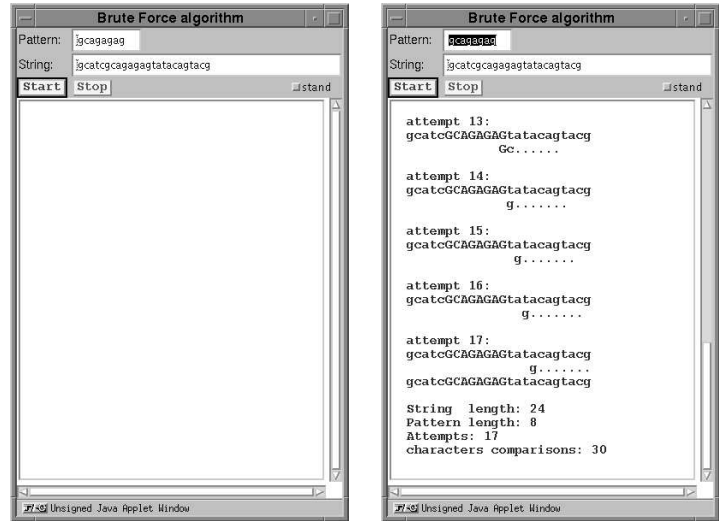
- matches are shown by displaying the word letter in upper case;
- mismatches are shown by displaying the word letter in lower case.

An occurrence of the word in the text is shown by displaying the corresponding text characters in upper case. At the end of the search phase, the system gives the number of attempts and the number of character comparisons performed during the search phase (see Figure 3(b)). The user can visualize the complete history of the search.

The system is written in Java. It is dedicated mainly to exact string matching algorithms but is easily extensible to a large family of algorithms. Moreover it is completely straightforward to implement any string matching algorithms providing that it is written in a specific way.

2.2 Sequence comparison

Pairwise sequence comparison consists in finding the minimal number of edit operations to transform a sequence x of length m into a sequence y of length n . The usual technique to solve this problem is based on dynamic programming and uses a two-dimensional matrix of size $m \times n$. The solution is usually given by an alignment of the two sequences. This technique is essential in computational biology: when a new sequence is found it is necessary to check if similar sequences are already known and stored in a data bank. Most of



(a) (b)

Figure 3: The window for the Brute Force string matching algorithm: (a) before a run; (b) after a run.

the tools daily used by the biologists such as Blast [1] or FastA [5] use some dynamic programming techniques.

Biologists often align two sequences together or even a sequence alone against a complete bank of sequences without understanding the basic notions of the process.

The user can choose among the different kinds of alignment methods. He then can give his values to both sequences x and y . The computation can then be run and it gives the values in the two-dimensional matrix and writes down all the corresponding alignments (see figure 4).

3 Conclusion

The two tools presented above have been successively used during a school dedicated to data processing of genomic information intended for researchers in biology. The public was not familiar with the environment but was able easily to use both systems.

But the main interest of this new technology is that the knowledges are

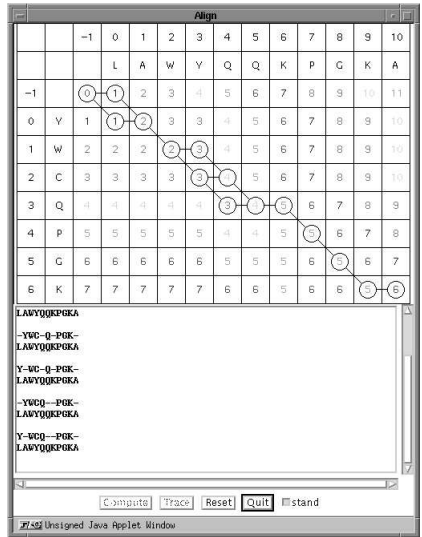


Figure 4: The window for the computation of the Levenshtein distance.

freely available world-wide. We know that people from all around the world use our applets.

References

- [1] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. A basic local alignment search tool. *J. Mol. Biol.*, 215:403–410, 1990.
- [2] C. Charras and T. Lecroq. Exact string matching algorithms. URL:<http://www.dir.univ-rouen.fr/~charras/string/>, 1996.
- [3] C. Charras and T. Lecroq. Sequence comparison. URL:<http://www.dir.univ-rouen.fr/~charras/seqcomp/>, 1996.
- [4] M. Crochemore and W. Rytter. *Text algorithms*. Oxford University Press, 1994.
- [5] W. R. Pearson and D. J. Lipman. Improved tools for biological sequence comparison. *Proc. Nat. Acad. Sci. U.S.A.*, 85:2444–2448, 1988.
- [6] G. A. Stephen. *String searching algorithms*. World Scientific Press, 1994.