Chapter 16
Efficient String Matching Algorithm for Searching Large DNA and Binary Texts

Abdulrakeeb M. Al-Ssulami
King Saud University, Saudi Arabia

Hassan Mathkour
King Saud University, Saudi Arabia

Mohammed Amer Arafah
King Saud University, Saudi Arabia

ABSTRACT

The exact string matching is essential in application areas such as Bioinformatics and Intrusion Detection Systems. Speeding-up the string matching algorithm will therefore result in accelerating the searching process in DNA and binary data. Previously, there are two types of fast algorithms exist, bit-parallel based algorithms and hashing algorithms. The bit-parallel based are efficient when dealing with patterns of short lengths, less than 64, but slow on long patterns. On the other hand, hashing algorithms have optimal sublinear average case on large alphabets and long patterns, but the efficiency not so good on small alphabet such as DNA and binary texts. In this paper, the authors present hybrid algorithm to overcome the shortcomings of those previous algorithms. The proposed algorithm is based on q-gram hashing with guaranteeing the maximal shift in advance. Experimental results on random and complete human genome confirm that the proposed algorithm is efficient on various pattern lengths and small alphabet.

1. INTRODUCTION

The problem of finding the occurrences of a predefined pattern in big data or very long sequences is classical problem in computer science and it has been studied intensively by researchers due to the growing of data which are produced day-by-day. There are many applications that depend on the pat-

DOI: 10.4018/978-1-7998-1204-3.ch016

tern matching, e.g., biological sequences which are produced every day by high throughput sequencing technologies (Morozova & Marra, 2008), natural language processing, or retrieving a specific data from very large databases. Bioinformatics is one of the fields where the pattern matching problem is used widely to search for specific pattern exactly or approximately (Barton, Iliopoulos, & Pissis, 2014; Simone Faro & Lecroq, 2009a; S. Faro & Lecroq, 2012; Kalsi, Peltola, & Tarhio, 2008) where the DNA is considered as a string and the four symbols *A*, *C*, *G*, and *T*, represent the four nucleotides Adenine, Cytosine, Guanine, and Thymine, respectively. Computer networks is the other area where this problem is applied. The binary string matching used in the intrusion detection systems to detect malwares within the heavy load traffic and searching IP addresses in routers swiftly (Chu, Huang, Tsai, & Hsieh, 2008; Liu, Huang, Chen, & Kao, 2004; Tuck, Sherwood, Calder, & Varghese, 2004).

The importance of speeding-up the searching process stems from that there is a need to find the occurrences of a particular pattern and whether the pattern exist or not in terabytes of data. Searching such amount of data offline by uploading the data to memory is not possible due to the memory limitations. In computer networks, Antiviruses are installed on routers or personal computers for detecting malwares so speeding-up the inspecting process increases the performance of the systems.

The problem is formulated as follows. Given a string T of length n and a pattern P of length m, the problem is to find the number of repetitions of P in T. In practice $m \ll n$. Let Σ denotes the set of symbols, then Σ^* denotes the language from which the pattern and the text are drawn and we write $P,T \in \Sigma^*$.

Two types of alphabets are considered in this paper, the DNA alphabet, $\Sigma=\{A,C,G,T\}$, and binary alphabet, $\Sigma=\{0,1\}$. The text and the pattern are represented by arrays of one dimension and the two arrays are indexed starting from 0. We say that the pattern P occurs in the text T, if $P[i]=T[j-m+i+1]$ where $0 \leq i < m$ and $\exists j$ such that $m-1 \leq j < n$.

2. LITERATURE REVIEW

There are so many exact string matching algorithms which are varying between character-based comparison(Al-Ssulami, 2015; Boyer & Moore, 1977; Deusdado & Carvalho, 2009; Franek, Jennings, & Smyth, 2007; Horspool, 1980; James H. Morris & Pratt, 1970; Knuth, James H. Morris, & Pratt, 1977; Lecroq, 2007; Sunday, 1990; Tarhio & Peltola, 1997), automata-based (Allauzen, Crochemore, & Raffinot, 1999; Allauzen & Raffinot, 2000; Blumer et al., 1985; Fan, Yao, & Ma, 2009; He, Fang, & Sui, 2005; Simon, 1994), and bit-parallel based (Baeza-Yates & Gonnet, 1992; Branislav, Durian, Holub, Peltola, & Tarhio, 2010; Cantone, Faro, & Giaquinta, 2010; Chen, Huang, & Lee, 2014; Simone Faro & Lecroq, 2009b; Mohanty & Tragoudas, 2014; Navarro & Raffinot, 2000; Peltola & Tarhio, 2003, 2014; Wu & Manber, 1992). Majority of character-based algorithms depend on the previously published ideas in (Boyer & Moore, 1977; James H. Morris & Pratt, 1970; Knuth et al., 1977) (See Simone Faro & Lecroq, 2013). The automata-based string matching depends on the idea of the finite automata. The algorithms of this type build the minimal deterministic finite automat then the text is parsed with the automata in time complexity of $O(n)$ in case of using an array of size $m|\Sigma|$; otherwise, the search phase costs $O(n\log|\Sigma|)$. The deterministic finite automata DFA is built for the pattern P of length m in time and space complexity of $O(m|\Sigma|)$ if an array is used (See (Simone Faro & Lecroq, 2013)). The bit-parallel based algorithms take the advantage of parallelism the bits inside the computer word. This technique