Category

A improved exact string matching method for genomic sequencing data

Co-responding Author[1,*], Co-author[2] and Co-Author[2]

[1]Department of XXXXXXX, Address XXXX etc.

[2]Department of XXXXXXX, Address XXXX etc.

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Associate Editor: XXXXXXX

ABSTRACT

Motivation: String matching algorithm plays a vital role in Bioinformatics. In this paper, we proposed an improved pattern matching algorithm based on epsm for biological sequences. According to the feature of biological sequences, the algorithm uses optimized word-size packed string matching instructions. Furthermore, in each test, the algorithm uses a big hash value to decrease byte-by-byte comparisons. Experimental results show that the algorithm is more efficient than epsm especially when the pattern length is short.

1 INTRODUCTION

String matching is an important issue that has been thoroughly studied in computer science. It is applied extensively in bioinformatics. For example, it is used to find similar sequence or locate a segment in a long sequence [1]. Currently, several string matching algorithms are used on biological sequences, such as tvsbs [2], graspm [3] etc. With the development of sequencing techniques, it has become easy to obtain sequences, i.e. the linear arrangement of residues (nucleotides or amino-acids), of DNA, RNA, or protein molecules. So it is meaningful to design more effective string matching algorithms to meet this challenge [4].
Recently, a string matching algorithm called epsm [5] has been used in bioinformatics and had obtained outstanding results. It uses packed string matching technique [6], in which multiple characters are packed into one block-character, so that the characters can be compared in bulk rather than individually. Epsm computes fingerprint values by a hash function using SIMD instructions, which supports parallel execution of some operations via a set of special instructions. However, the max shift distance of epsm algorithm is m-8, where m is the length of the given pattern. It was not an optimal shift distance, especially for biological sequences. In present study, we improve epsm by getting more shift distances and less byte-by-byte comparison calls.

2 MATERIAL AND METHODS

2.1 Algorithm

Epsm algorithm stores all the 8 bytes substrings of the pattern and their position in a hash table. In the searching phase, it checks the last 8 bytes of the current searching window using the hash table to find possible matches. It then checks all the possible positions byte-by-byte. Since the SIMD instructions are used to calculate hash value of packed string, the size of packed strings is set as 8 bytes. When finishing the examination of current window, it jumps forward by m-8 bytes to perform next examination, where m is the length of the pattern.
The basic idea to improve the epsm algorithm is trying to get larger jumping distance and less byte-by-byte comparisons. Our improvements are as follows.
1) Optimization of the size of packed strings for biological sequences. In fact, shift distance of epsm is m-B, here B is the size of packed strings. If take smaller B, we can achieve bigger shift distance. However, smaller B can also result in more hash conflicts and more possible matches which incurs further examinations. We decide the optimal B though experimental method.
2) Reducing byte-by-byte comparison between pattern and text. The epsm uses a 32bit hash value and then mod the size of hash table to determine the entry of hash table and the entrance of byte-by-byte comparison. Intuitively, taking bigger number as the entrance condition will trigger less byte-by-byte comparison. We choose 64bit numbers as the entrance condition of byte-by-byte comparison. In practice, we add a new field fingerprint to record the 64bit number in the definition of node. These fingerprints are initialized during pre-processing phase. As epsm, we use SIMD instructions to quickly calculate hash value of strings, thus 8 bytes substrings are used to generate the fingerprint. In searching phrase, before calling memcmp to perform byte-by-byte comparison, the algorithm examine the 64bit fingerprint of current window against that of substrings of the pattern. By this way, we avoid lots of calls of memcmp.

*To whom correspondence should be addressed.

The data structure of the nodes, i.e., the items of hash table is shown in Fig.1. The algorithm is shown in Fig.2.

2.2 Estimating

We conducted two experiments to estimate the performance of the algorithm. The first one was to evaluate the optimal packed string size on different pattern length. The second one was the comparison to five representative string matching algorithms.
The algorithm was implemented using C. And all experiments were conducted on a PC with Intel(R) Xeon E3-1230 V2 at3.30GHz, 4G memory, running Linux Mint 13. The size of the hash table in the improved algorithm equals to that of epsm algorithm, i.e. 2048.

2.3 Data collection

We compare the improved algorithm against five state-of-art string matching algorithms, i.e., Tvsbs [2], Ufndmq [7], Hashq [8], Fsbndmq [9] and Epsm [5]. All the algorithms executed on four test data sets [10], as follows. To make the experimental results more comparable, we make the data sets with the same size through simple duplication.

Table 1. Table

Data set	type	filename	size
S1	Escherichia coli	Escherichia_ albertii_KF1_uid232181	200MB
S2	Rice	OrySat_Aug2009.fa	200MB
S3	gene sequence of the human genome on chromosome 1	chr1.fa	200MB
S4	amino acid of the Escherichia coli K12MG1655	NC_000913.faa	200MB

This is.

3 RESULTS AND DISCUSSION

3.1 Optimal packed string size

The optimal packed string size is essential to the algorithm's performance. This optimal size would be affected by the length of pattern and the alphabet of the sequences. To determine the optimal size, we tested different length of packed string on different pattern size on different data sets.
The experimental results on dataset S1 are shown in Table 2, where the size of alphabet is 4. And the experimental results on dataset S4 are shown in Table 3, where the size of alphabet is 20.
We can see from Table 2 that the optimal packed string size for gnome sequences is 6 when m is less than 40, otherwise it is 8. Likewise, we can see from Table 3 that the optimal packed string size for protein sequences is 4 when m is less than 256, and it is 8 otherwise.
In the following experiments, we will use these optimal parameters.

Table 2. Table

m	8	12	16	20	24	28	32	36
5	62.85	60.2	35.45	30	27.65	25.95	23.85	23.7
6	**57.05**	**56.85**	**31.7**	**25.3**	**21.25**	**19.15**	**18.05**	**17.55**
7	60.95	58.7	33.2	25.3	21.45	20.55	18.85	18.35
8	62.65	59.1	35.15	25.5	22.7	20.95	19.8	18.65

m	40	44	48	64	128	256	512	1024	2048
5	22.85	21.8	21.1	20.1	19.15	16.5	16.95	18	14.95
6	17.9	17.35	17.05	16.3	15.95	10.6	8.3	7.35	6.6
7	17.15	16.45	16.2	15.55	14.5	9.7	6.1	4.7	3.7
8	**16.9**	**17.05**	**16.15**	**15.1**	**15.4**	**8.85**	**5.65**	**3.7**	**2.85**

Table 3. Table

m	8	12	16	20	24	28	32	36
4	**58.05**	**30.90**	**22.70**	**19.45**	**17.95**	**16.90**	**16.20**	**15.85**
5	68.65	39.00	26.45	21.50	19.35	18.10	17.40	16.55
6	76.75	44.95	29.05	22.55	19.85	18.50	17.40	16.70
7	87.55	53.75	31.55	23.70	20.45	18.70	17.65	16.80
8	110.20	67.30	33.80	24.50	20.30	18.80	17.40	16.85

m	40	44	48	64	128	256	512	1024	2048
4	**15.40**	**15.15**	**14.85**	14.55	**13.30**	**8.00**	5.00	3.90	3.45
5	16.10	15.60	15.40	14.55	13.80	8.90	5.40	3.60	2.80
6	16.10	15.65	15.30	14.55	14.10	8.90	5.35	3.70	2.75
7	16.40	15.90	15.50	14.75	14.25	9.30	5.55	4.10	3.35
8	16.10	15.80	15.25	**14.40**	13.70	8.50	**4.80**	**3.60**	**3.00**

3.2 Efficiency

The searching time on S1, S2, S3 and S4 is shown in Fig. 1. The improved algorithm proposed by present work is marked as new. It is evident that in all of the instances our new algorithm is of the most efficient.
The advantage of the new algorithm decreases when the length of patterns grows. When m is no less than 40 on gnome datasets and m is not less than 256 on protein datasets, the new algorithm takes the same size of packed string as epsm does. Thus the reason of the performance improvement is the filtering before taking byte-by-byte comparison using a big fingerprint. This filter avoids lots of this kind of comparison. And when pattern strings are short, the performance improvement also benefits from the increment of shift distances. For example, on gnome dataset, when m is 12, the shift distance of the epsm is 4, but that of the new algorithm could be 6. But when m is larger than 40, the shift distance of the new algorithm is the same as epsm. The shorter the pattern string is, the new algorithm can get more shift distance than epsm. Therefore, in the experiments with short pattern string, the new algorithm particularly outperforms other algorithms.
At the same time, we can observe from the tables that the new algorithm and epsm obviously outperform other algorithms. This is because the use of SIMD instructions and these results are consistent with the literature [5].

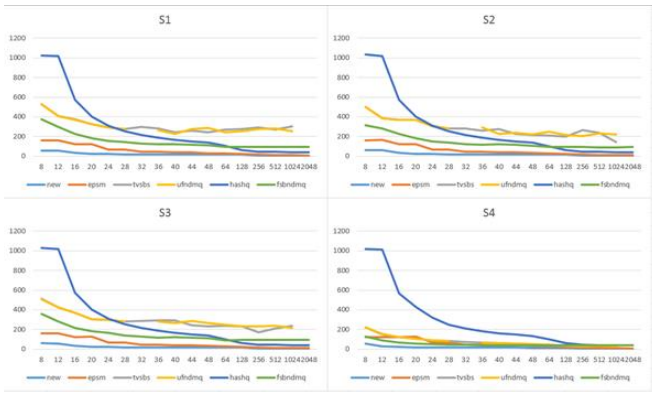

Fig. 1. Relation between τ and *t*. This example has only two continuous Steppers, S_1 and S_2.

4 CONCLUSIONS

In this article, we presented an improved algorithm based on the epsm algorithm for string matching problem on biological sequences. Based on characteristics of biological sequences, the new algorithm took the optimization size of the packed string to generate the hash table. Besides, the new algorithm used a 64bit number as a fingerprint of string, the fingerprint was generated by 8 bytes string. We used the fingerprint filtering many comparisons before taking byte-by-byte comparison. And the experiment results proved using these two optimized strategies, our new algorithm was obviously more efficient than the epsm.

Our further studies would apply the new algorithm to the multiply exactly string matching problem.

ACKNOWLEDGEMENTS

We thank XXX for YYY. This work was supported by the "100-Talent Program" of Chinese Academy of Sciences (Y1SLXb1365; ZZ).

REFERENCES

[1] K.K.Senapati, Sandip Mal, G.Sahoo. RS-A Fast Pattern Matching Algorithm for Biological Sequences. International Journal of Engineering and InnovativeTechnology (IJEIT) Volume 1, Issue 3, March (2012).

[2] R. Thathoo and A. Virmani and S. S. Lakshmi and N. Balakrishnan and K. Sekar. TVSBS: A Fast Exact Pattern Matching Algorithm for Biological Sequences. J. Indian Acad. Sci., Current Sci., vol.91, n.1, pp.47--53, (2006).

[3] S. Deusdado and P. Carvalho. GRASPm. An efficient algorithm for exact pattern-matching in genomic sequences. Int. J. Bioinformatics Res. Appl., vol.5, n.4, pp.385--401, Inderscience Publishers, Inderscience Publishers, Geneva, SWITZERLAND, (2009).

[4] Eric Rivals, Leena Salmela, and Jorma Tarhio. EXACT SEARCH ALGORITHMS FOR BIOLOGICAL SEQUENCES. Algorithms in Computational Molecular Biology: Techniques, Approaches and Applications, John Wiley & Sons, Inc. (Ed.) (2011) 91-111.

[5] S. Faro and M. O. Kulekci. Fast Packed String Matching for Short Patterns. Meeting on Algorithm Engineering and Experiments, ALENEX 2013, (2013).

[6] K. Fredriksson. Faster string matching with super-alphabets. String Processing and Information Retrieval, Springer (2002), pp. 207–214.

[7] B. Durian and J. Holub and H. Peltola and J. Tarhio. Tuning BNDM with q-Grams. Proceedings of the Workshop on Algorithm Engineering and Experiments, ALENEX 2009, pp.29--37, SIAM, New York, New York, USA, (2009).

[8] T. Lecroq. Fast exact string matching algorithms. ipl, vol.102, n.6, pp.229--235, Elsevier North-Holland, Inc., Amsterdam, The Netherlands, The Netherlands, (2007).

[9] H. Peltola and J. Tarhio. Variations of Forward-SBNDM. Proceedings of the Prague Stringology Conference '11, pp.3--14, Czech Technical University, Prague, Czech Republic, (2011).

[10] ftp://ftp.ncbi.nlm.nih.gov/genbank/genomes/Bacteria/