A improved exact string matching method for genomic sequencing data

XXX1, Zhang Zhang1,\*

1CAS Key Laboratory of Genome Sciences and Information, Beijing Institute of Genomics, Chinese Academy of Sciences, Beijing 100029, China

2School of Computer Science and Technology, XXX

\***Corresponding authors**:

Zhang Zhang (zhangzhang@big.ac.cn): CAS Key Laboratory of Genome Sciences and Information, Beijing Institute of Genomics, Chinese Academy of Sciences, No.7Beitucheng West Road, Building G, Chaoyang District, Beijing100029, China, Tel& Fax: +86-10-82995427

Running head: Improved exact string matching algorithm using SIMD instructions.

### Abstract

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.

Key words: Exact string matching, SIMD, biological sequence, epsm algorithm

### 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] [10]. Currently, several string matching algorithms are used on biological sequences, such as tvsbs [2] [5], graspm [3] [9] 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] [1].

Recently, a string matching algorithm called epsm [5] [4] has been used in bioinformatics and had obtained outstanding results. It uses packed string matching technique [6] [3], 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.

### Material and methods

##### 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.

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.

##### 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.

##### Data collection

We compare the improved algorithm against five state-of-art string matching algorithms, i.e., Tvsbs [5], Ufndmq [6], Hashq [7], Fsbndmq [8] and Epsm [4]. All the algorithms executed on four test data sets [11], as follows. To make the experimental results more direct viewing, we make the data sets with the same size through simple duplication.

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |

### Results and discussion

##### 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 1, where the size of alphabet is 4. And the experimental results on dataset S4 are shown in Table 2, where the size of alphabet is 20.

We can see from Table1 that the optimal packed string size for DNA sequences is 6 when m is less than 40, and it is 8 when m is no less than 40. Likewise, we can see from Table2 that the optimal packed string size for **protein** sequences is 4 when m is less than 40, and it is 8 when m is no less than 48.

In the following experiments, we will use these optimal parameters.

##### Efficiency

The searching time on S1, S2, S3 and S4 is shown in table 3 to table 6. Our new algorithm is marked as *new.* It is easy to find out that our new algorithm is the most efficient.

The advantage of the new algorithm fades away when length of pattern string grows. When m is no less than 40, the *new* takes the same size of packed string as the epsm does. So the reason why the *new* faster than the epsm is that it has filtered some possibilities by a fingerprint before taking comprising byte-by-byte. It will avoid a lot of the comparison. And when the size of packed string is short, the ascension of the *new* is mainly because of the increment of jumping distance. For example, when m equals 12, the jumping distance of the epsm is 4, but the *new* can achieve 6. The length of pattern string more short, this kind of improvement becomes more obvious. Therefore, in the condition of short pattern string, the new algorithm has significant improvement in efficiency.

At the same time, we have observed that the *new* and the epsm is far more efficient than other algorithms. That is because we use the SIMD instruction and this result agrees to the paper about the epsm.

在S1,S2,S3,S4上的搜索时间在分别在表3-6中显示。我们的新算法用new进行标识。可以从表中看出，在所有实例中，新算法是最高效的。

新算法的优势随着模式串的长度增加逐渐减小。当m>40的时候，新算法和epsm采用相同大小的块字符。所以新算法比epsm高效的原因在于进行逐字节比较之前，采用一个大的指纹进行了过滤，从而避免了很多的这种比较。而在模式串较短的情况下，新算法效率的提升还在于跳跃距离的增加。比如当m=12的时候，epsm的跳跃距离为4，而新算法的跳跃距离可以达到6. 而当m>40的时候，新算法的跳跃和epsm相同。这种增加随着模式串越短越明显，所以在模式串相对较短的情况下，新算法对于性能的提升更加的明显。

同时我们可以看到，EPSM和新算法明显的比其它算法高效，这是因为采用了SIMD instruction。这些结果与epsm文献是一致的。

### Conclusions

We improve the epsm algorithm in biological sequences. And the experiments show the algorithm obtain the best performance.

### Acknowledgments

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

[1] 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.

[2] D. M. Sunday. “A very fast substring search algorithm”, Communications of the ACM, 33(8):132-142, 1990.

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

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

[5]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).

[6] 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).

[7] 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).

[8] 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).

[9]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).

[10] 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).

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

### Tables

Table 1.

Experimental results for searching different length of patterns on an **ecoli** genome sequence using different length of packed string. Running times are expressed in millisecond, best results have been boldfaced and padded.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 2.

Experimental results for searching different length of patterns on a **protein** genome sequence using different length of packed string. Running times are expressed in millisecond, best results have been boldfaced and padded.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| m | 8 | 12 | 16 | 20 | 24 | 28 | 32 | 36 |
| 4 | **58.05** | **30.9** | **22.7** | **19.45** | **17.95** | **16.9** | **16.2** | **15.85** |
| 5 | 68.65 | 39 | 26.45 | 21.5 | 19.35 | 18.1 | 17.4 | 16.55 |
| 6 | 76.75 | 44.95 | 29.05 | 22.55 | 19.85 | 18.5 | 17.4 | 16.7 |
| 7 | 87.55 | 53.75 | 31.55 | 23.7 | 20.45 | 18.7 | 17.65 | 16.8 |
| 8 | 110.2 | 67.3 | 33.8 | 24.5 | 20.3 | 18.8 | 17.4 | 16.85 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| m | 40 | 44 | 48 | 64 | 128 | 256 | 512 | 1024 | 2048 |
| 4 | **15.4** | **15.15** | **14.85** | 14.55 | **13.3** | **8** | 5 | 3.9 | 3.45 |
| 5 | 16.1 | 15.6 | 15.4 | 14.55 | 13.8 | 8.9 | 5.4 | 3.6 | 2.8 |
| 6 | 16.1 | 15.65 | 15.3 | 14.55 | 14.1 | 8.9 | 5.35 | 3.7 | 2.75 |
| 7 | 16.4 | 15.9 | 15.5 | 14.75 | 14.25 | 9.3 | 5.55 | 4.1 | 3.35 |
| 8 | 16.1 | 15.8 | 15.25 | **14.4** | 13.7 | 8.5 | **4.8** | **3.6** | **3** |

Table 3.

Experimental results for searching different length of patterns on an **ecoli** genome sequence using six algorithm. Running times are expressed in millisecond, best results have been boldfaced and padded.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| m | 8 | 12 | 16 | 20 | 24 | 28 | 32 | 36 |
| new | **57.05** | **56.85** | **31.7** | **25.3** | **21.25** | **19.15** | **18.05** | **17.55** |
| epsm | 158.05 | 157.8 | 123.55 | 124.75 | 65.05 | 66.1 | 46.35 | 47.15 |
| tvsbs | 531.29 | 409.43 | 372.43 | 325.86 | 292 | 276.29 | 300.43 | 278.86 |
| ufndmq | 527.86 | 407.29 | 370 | 324.86 | 291 | 275.57 | - | 262.57 |
| hashq | 5022.86 | 1015.71 | 574.43 | 400.86 | 307.71 | 253.14 | 216.14 | 187 |
| fsbndmq | 376.57 | 296.14 | 226.14 | 184 | 152.71 | 142.14 | 126.29 | 120.57 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| m | 40 | 44 | 48 | 64 | 128 | 256 | 512 | 1024 | 2048 |
| new | **16.9** | **17.05** | **16.15** | **15.1** | **15.4** | **8.85** | **5.65** | **3.7** | **2.85** |
| epsm | 37.2 | 37.5 | 31.3 | 25.9 | 21.45 | 17.95 | 13.4 | 10.95 | 9.15 |
| tvsbs | 245 | 256.71 | 244.29 | 268.71 | 277.14 | 293.86 | 270.57 | 301 | - |
| ufndmq | 225 | 274.86 | 286.71 | 243 | 254.57 | 274.86 | 280.43 | 254.71 | - |
| hashq | 167.14 | 149.57 | 138 | 103.86 | 63.71 | 45.29 | 43.71 | 40.29 | 40.29 |
| fsbndmq | 120.57 | 113.86 | 113.14 | 94.14 | 94.29 | 95.86 | 94 | 97.14 | 96.71 |

Table 4.

Experimental results for searching different length of patterns on a **rice** genome sequence using six algorithm. Running times are expressed in millisecond, best results have been boldfaced and padded.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| m | 8 | 12 | 16 | 20 | 24 | 28 | 32 | 36 |
| new | **62.9** | **60** | **31.45** | **24.6** | **21.15** | **19.75** | **18.3** | **17.55** |
| epsm | 161.5 | 164.45 | 122.6 | 124.25 | 65.15 | 65.9 | 46.75 | 47.4 |
| tvsbs | 502 | 385.86 | 371.86 | 369.71 | 306.14 | 283 | 281.86 | 260.71 |
| ufndmq | 500.29 | 385.57 | 370.29 | 368.43 | 304.14 | 283.29 | - | 293 |
| hashq | 5033 | 1016.14 | 570.86 | 399.71 | 308.71 | 252.71 | 213.71 | 186.43 |
| fsbndmq | 317 | 280.29 | 224.43 | 180 | 148.14 | 139.71 | 124.57 | 116.57 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| m | 40 | 44 | 48 | 64 | 128 | 256 | 512 | 1024 | 2048 |
| new | **16.55** | **16.3** | **15.75** | **14.9** | **15** | **8.8** | **5.55** | **3.75** | **3.05** |
| epsm | 37.4 | 38.1 | 31.75 | 26.25 | 21.8 | 18.7 | 13.7 | 11.75 | 9.65 |
| tvsbs | 273.43 | 224 | 216 | 211.57 | 198.43 | 263.43 | 235.71 | 144.86 | - |
| ufndmq | 228 | 239.57 | 219.29 | 247.14 | 213 | 202 | 229.29 | 219.29 | - |
| hashq | 165.29 | 149.86 | 136.71 | 102 | 62.14 | 44.86 | 42.86 | 40 | 39.57 |
| fsbndmq | 121.43 | 117.86 | 107.43 | 93.43 | 92.14 | 94.43 | 91.71 | 91.43 | 93.71 |

Table 5.

Experimental results for searching different length of patterns on a **human** genome sequence using six algorithm. Running times are expressed in millisecond, best results have been boldfaced and padded.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| m | 8 | 12 | 16 | 20 | 24 | 28 | 32 | 36 |
| new | **61.35** | **57.4** | **31.45** | **25.65** | **21.55** | **19.7** | **18.65** | **17.5** |
| epsm | 163.05 | 159.2 | 123.45 | 125.6 | 65.5 | 66.8 | 47.1 | 47.35 |
| tvsbs | 509.71 | 426.86 | 370.86 | 302.86 | 299.71 | 281.86 | 284.29 | 293.71 |
| ufndmq | 505.43 | 424.29 | 369 | 301.29 | 297.29 | 280.43 | - | 281.43 |
| hashq | 5028.29 | 1015.71 | 571.71 | 400.57 | 308.86 | 252.29 | 214.57 | 186.57 |
| fsbndmq | 360.29 | 281.29 | 215.14 | 182.86 | 166.14 | 137.14 | 126.86 | 118.57 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| m | 40 | 44 | 48 | 64 | 128 | 256 | 512 | 1024 | 2048 |
| new | **16.9** | **16.55** | **15.85** | **15.6** | **15.45** | **8.95** | **5.9** | **3.95** | **3.05** |
| epsm | 37.9 | 38.2 | 32.1 | 26.65 | 22.25 | 18.85 | 14.05 | 11.75 | 9.9 |
| tvsbs | 293.43 | 241.43 | 229.43 | 237.14 | 231.57 | 169.86 | 211 | 239.29 | - |
| ufndmq | 265.57 | 284.43 | 263.43 | 249.71 | 234.14 | 233.29 | 240.14 | 216.14 | - |
| hashq | 167 | 149 | 137.86 | 102.57 | 62.57 | 44.29 | 43 | 40.29 | 39.57 |
| fsbndmq | 120.14 | 117.14 | 113.43 | 91.14 | 96.29 | 96.14 | 95.43 | 96.29 | 93.43 |

Table 6.

Experimental results for searching different length of patterns on a **protein** genome sequence using six algorithm. Running times are expressed in millisecond, best results have been boldfaced and padded.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| m | 8 | 12 | 16 | 20 | 24 | 28 | 32 | 36 |
| new | **58.05** | **30.9** | **22.7** | **19.45** | **17.95** | **16.9** | **16.2** | **15.85** |
| epsm | 123 | 122.8 | 122.55 | 125.6 | 64.85 | 65.85 | 46.25 | 46.85 |
| tvsbs | 222.29 | 152 | 120.86 | 103.57 | 90.86 | 81.71 | 70.57 | 64.43 |
| ufndmq | 222.57 | 152.43 | 120.29 | 103.71 | 90.43 | 81 | - | 66.14 |
| hashq | 5018 | 1010.14 | 566.14 | 431.43 | 317.29 | 249.71 | 211.43 | 184.71 |
| fsbndmq | 126.86 | 86.86 | 68.57 | 57.86 | 52.14 | 46.57 | 43.57 | 40.29 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| m | 40 | 44 | 48 | 64 | 128 | 256 | 512 | 1024 | 2048 |
| new | **15.4** | **15.15** | **14.85** | **14.4** | **13.7** | **8.5** | **4.8** | **3.6** | **3** |
| epsm | 37.25 | 37.6 | 31.5 | 26 | 21.8 | 18.6 | 13.4 | 10.7 | 8.9 |
| tvsbs | 59.14 | 54.71 | 50.86 | 42.14 | 31.57 | 29.29 | 31.57 | 28 | - |
| ufndmq | 60 | 56 | 50 | 43 | 31.86 | 28.71 | 31.43 | 29.14 | - |
| hashq | 163 | 147 | 133.71 | 100.14 | 59.71 | 42.71 | 41.43 | 39.43 | 39.43 |
| fsbndmq | 39.86 | 40.43 | 41.29 | 41.43 | 40.14 | 39.86 | 39.29 | 38 | 39 |

### Figure legends

Fig. 1 The node

Typedef struct node

{

struct node \*next;

intpos;

int64 fingerprint;

} NODE;

In the node, the fingerprint is introducing by the improved algorithm, which mark the fingerprint of sub-string of length 8 bytes on current position. It is not exist in the epsm algorithm.

### Figures

Fig. 2 The algorithm

