A new 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: Exact string matching algorithm using SIMD instructions.

### Abstract

String matching algorithm plays the vital role in the Bioinformatics. In this research, we propose a fast pattern matching algorithm for biological sequences:The SSE Hash Shift algorithm.The algorithm use specialized word-size packed string matching instructions, based on the Intel streaming SIMD extensions (SSE) technology to compute the shift distance.The experimental results show the algorithm is superior to other algorithms especially when the length of the pattern is short then 64 bytes.

Key words: Exact string matching, SSE, biological sequences

### Introduction

Pattern matching is one of the most important research subjects that have been studied in computer science.With the development of sequencing techniques, it has become easy to obtain the sequence, i.e. the linear arrangement of residues (nucleotides or amino-acids), of DNA, RNA, or protein molecules. So we need a more effective string mathcing algorithm to tackle this challenge [1].

There are two types of exact pattern matching: single pattern matching and multiple patterns matching, here we focus on the single pattern matching.

Classical single pattern matching algorithm includes KMP algorithm and BM algorithm. Other algorithm is mostly improved on these classical algorithms In BM algorithm [2], pattern string moves from left to right, while character is compared from right to left. When it loses matching, the predefined excursion function adopts the max value to decide right shift value. The efficiency of an algorithm depends on two phases: the pre-processing phase and the searching phase. The aim of a good algorithm is to minimize the work done during each attempt and to maximize the length of the shifts.

In the recent years, a new solutions called packed string matching technique [3] has been used on the problem. In the packed string matching technique, multiple characters are packed into one larger word, so that the characters can be compared in bulk rather than individually. And in the modern CPU, SIMD instructions exist in many recent microprocessors supporting parallel execution of some operations on multiple data simultaneously via a set of special instructions working on limited number of special registers. And epsm [4] had obtain good results. In this case, we use Intel streaming SIMD extensions (SSE) technology to achive the goal.

### Material and methods

##### Algorithm

In this section we would describe the SSE Hash Shift algorithm. The algorithm is designed to be faster for medium biological sequences. The algorithm is shown in Fig. 1.

The hash function used for calculating the CRC of a fixed-length string using the following SSE instruction, \_mm\_crc32\_u64(). And then we use a mask to get a 32bit number. The pre function was to calculate shift distance of the given fixed-length string. In the search phrase, it was responsible for sampling the text string by using hash function. And determing the shift distance based on the hash value, and check whether matched when the value equal one. So in most cases, we could jump the distance of m in a great probability. Unlile epsm , it can only jump the distance of m-8 [4].

##### Estimating

We would conduct two experiments, the first one was the comparison between the five representive string matching algorithms, the second one was the comparison between different length of packed string and the third one was the comparison between memcmp function and INT64MEMCMP which compare the string using 64bytes.

We expect the algorithm was faster then the other algorithms for the biological sequences, especially for the medium one(the length of the pattern string was less than 64 bytes and greater than 8 bytes).

##### Data collection

The algorithm was implemented using C. And all experiments were conducted on a PC with Intel(R) Xeon E3-1230 V2@3.30GHz and 4G memory. The host operating system was Linux Mint 13.

In this case, we chose five different type of string matching algorithms as a comparison experiment.

- Tvsbs [5], which represent the pattern matching algorithm for biological sequences;

- Ufndmq [6], which represent the prefix searching algorithm;

- Hashq [7], which represent the suffix searching algorithm;

- Fsbndmq [8], which represent the substring searching algorithm;

- Epsm [4], which was the fastest pattern matching algorithm at the present.

And the test data suit contained four types of gene data:

- the Escherichia coli, ecoil;

- the rice, OrySat\_Aug2009.fa;

- the gene sequence of the human genome on chromosome 1, chr1.fa;

- the amino acid of the Escherichia coli K12MG1655, NC\_000913.faa.

We assume these test data and algorithms were representative.

### Results and discussion

##### Efficiency

From experimemtal results shown in the Table 1, Table 2, Table 3 and Table 4, we could see our algorithm obtained the best performance when m<64 and m>8 And beat the other four algorithms such as tvsbs, ufndmq, hashq and fsbndmq in all cases. The only challenger was epsm algorithm, which perform better when m≥64.

This results were as expected. Because the advantage of our algorithm faded when the length of pattern string grew.

##### Parameter

From experimental results shown in the Table 5, Table 6 and Table 7, we could see the influence of length of packed string. In general, we chose 8 when m≥64 and 6 the other way around.

##### INT64

This experiments does not obtain good results. Table 8, table 9 and table 10 showed there were no difference between INT64MEMCMP and MEMCMP function.

### Conclusions

We presented a new exact string matching algorithm using Intel SSE instructions. Experiments show the algorithm obtained better performance for medium biological sequences.

### ~~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).

### Tables

Table 1.

Experimental results for searching different length of patterns on a 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 | **146.55** | **95.9** | **72.35** | **58.2** | **49.25** | **43.1** | **38.9** | **35.9** |  |
| 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 | **33.7** | **32.25** | **30.95** | 29.2 | 26.4 | 41.25 | 32.1 | 17.65 | 11.65 |
| 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 2.

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 | **148.05** | **96.3** | **71.95** | **58.5** | **49.8** | **44.05** | **39.6** | **36.4** |  |
| 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 | **34.4** | **33.35** | **31.7** | 30 | 26.95 | 42.9 | 32.45 | 18.1 | 11.45 |
| 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 3.

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 | **146.5** | **95.9** | **71.75** | **57.85** | **49.25** | **42.9** | **39.05** | **35.9** |  |
| 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 | **34** | **32.2** | **31.2** | 29.7 | 27.2 | 44.1 | 33.1 | 18.3 | 12.15 |
| 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 4.

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 | 128.15 | **84.6** | **64.05** | **52.2** | **44.4** | **39.95** | **36.25** | **33.6** |  |
| 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 | **32** | **30.65** | **29.9** | 28.1 | 29 | 40.4 | 29.6 | 17.05 | 10.8 |
| 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 |

Table 5.

Experimental results for searching different length of patterns on a 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 | 149.9 | 98.45 | 73.95 | 60.3 | 51.25 | 45.35 | 40.6 | 38.1 |  |
| 6 | **146.55** | **95.9** | **72.35** | **58.2** | **49.25** | **43.1** | **38.9** | **35.9** |  |
| 8 | 163.85 | 99.8 | 72.8 | 58.25 | 49.6 | 43.3 | 39.45 | 36.95 |  |
|  |  |  |  |  |  |  |  |  |  |
| m | 40 | 44 | 48 | 64 | 128 | 256 | 512 | 1024 | 2048 |
| 5 | 35.35 | 34.6 | 32.6 | 30.15 | **26.2** | 42.35 | 44.15 | 37.35 | 32.75 |
| 6 | **33.7** | **32.25** | **30.95** | **29.2** | 27.45 | 41.05 | 35.4 | 21.15 | 15.6 |
| 8 | 34.4 | 32.9 | 31.85 | 29.3 | 26.4 | **41.25** | **32.1** | **17.65** | **11.65** |

Table 6.

Experimental results for searching different length of patterns on a rice 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 | 149.7 | 98.1 | 73.3 | 59.65 | 50.9 | 44.85 | 40.85 | 37.55 |  |
| 6 | **148.05** | **96.3** | **71.95** | **58.5** | **49.8** | **44.05** | **39.6** | **36.4** |  |
| 8 | 163.35 | 99.05 | 72.5 | 58.15 | 49.4 | 43.55 | 39.45 | 36.75 |  |
|  |  |  |  |  |  |  |  |  |  |
| m | 40 | 44 | 48 | 64 | 128 | 256 | 512 | 1024 | 2048 |
| 5 | 35.65 | 33.65 | 32.7 | 30.5 | **26.65** | 44.55 | 44.3 | 35.25 | 30.9 |
| 6 | **34.4** | **33.35** | **31.7** | 30 | 28.55 | 43.95 | 36.05 | 22.1 | 15.7 |
| 8 | 34.85 | 33.35 | 32.1 | **29.4** | 26.95 | **42.9** | **32.45** | **18.1** | **11.45** |

Table 7.

Experimental results for searching different length of patterns on a human 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 | 149.45 | 97.35 | 73.85 | 59.8 | 51.3 | 45.15 | 40.85 | 38.3 |  |
| 6 | **146.5** | **95.9** | **71.75** | **57.85** | **49.25** | **42.9** | **39.05** | **35.9** |  |
| 8 | 164.35 | 99.45 | 73.05 | 58.7 | 49.65 | 43.75 | 40 | 37.3 |  |
|  |  |  |  |  |  |  |  |  |  |
| m | 40 | 44 | 48 | 64 | 128 | 256 | 512 | 1024 | 2048 |
| 5 | 35.5 | 34.2 | 32.95 | 30.75 | 26.9 | 44.3 | 44.7 | 36.45 | 33 |
| 6 | **34** | **32.2** | **31.2** | **29.7** | 27.9 | **42.75** | 34.9 | 21.4 | 15.05 |
| 8 | 35.1 | 33.7 | 32.5 | 29.95 | **27.2** | 44.1 | **33.1** | **18.3** | **12.15** |

Table 8.

Experimental results for searching different length of patterns on a ecoli genome sequence using two kinds of string comparing function. Running times are expressed in millisecond.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| m | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 |
| int64cmp | 164.15 | 72.95 | 39.75 | 30 | 27.3 | 43.55 | 32.95 | 18.2 | 11.5 |
| memcmp | 164.4 | 73 | 39.6 | 29.8 | 27.2 | 42.05 | 32.6 | 17.95 | 11.25 |

Table 9.

Experimental results for searching different length of patterns on a rice genome sequence using two kinds of string comparing function. Running times are expressed in millisecond.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| m | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 |
| int64cmp | 164.3 | 72.9 | 39.7 | 30 | 27 | 43.35 | 32.6 | 17.75 | 12.3 |
| memcmp | 164.45 | 73.05 | 39.5 | 29.65 | 26.7 | 41.7 | 32.2 | 18.1 | 11.6 |

Table 10.

Experimental results for searching different length of patterns on a human genome sequence using two kinds of string comparing function. Running times are expressed in millisecond.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| m | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 |
| int64cmp | 164 | 73.1 | 39.85 | 30.55 | 27.25 | 44.5 | 33 | 18.4 | 12.15 |
| memcmp | 164.4 | 73.45 | 39.85 | 30.05 | 27.35 | 43.45 | 32.85 | 18 | 12.35 |

### Figure legends

Fig. 1 The algorithm.

### Figures

Fig. 1

