# 算法选择

|  |  |  |
| --- | --- | --- |
| 改进 | 新算法 |  |
| 目前最快的 | epsm | S. Faro and M. O. Kulekci. Fast Packed String Matching for Short Patterns. Meeting on Algorithm Engineering and Experiments, ALENEX 2013, (2013). |
| 生物学的 | tvsbs | 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). |
| 前缀搜索 | Ufndmq | 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). |
| 后缀搜索 | Hashq | 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). |
| 子串搜索 | Fsbndmq | 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). |

为什么选择这几种算法？ 2013, (2013).出现的EPSM，之前2012，2011有没有相应的算法？或者EPSM已经比结过的，作为间接比较？有没有遗漏专门针对生物信息的匹配算法？

2012年有个bsdmq算法，但是是epsm的作者发的([S. Faro](http://www.dmi.unict.it/~faro/smart/authors?author=S.%20Faro) and [T. Lecroq](http://www.dmi.unict.it/~faro/smart/authors?author=T.%20Lecroq). A Fast Suffix Automata Based Algorithm for Exact Online String Matching. Implementation and Application of Automata - 17th International Conference, CIAA 2012, Lecture Notes in Computer Science, n.7381, pp.149--158, Springer-Verlag, Berlin, (2012).)

Epsm之前的算法是2011年的ssecp，它是开始使用硬件指令得到不错的改进效果的算法，epsm的实验就是拿它来对比(O. Ben-Kiki and P. Bille and D. Breslauer and L. Gasieniec and R. Grossi and O. Weimann.Optimal Packed String Matching. IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2011, LIPIcs, n.13, pp.423--432, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, (2012).)

选择tvsbs是因为它是生物学的算法，以前找的时候没有看到比2006年更晚的，应该有遗漏，我再找找。

后面三个选择是因为算法实现类别来选的，前缀、后缀和子串。

# 测试集选择

## 大肠杆菌ecoil

## 水稻 OrySat\_Aug2009.fa

## 人类基因组1号染色体的基因序列chr1.fa

## 氨基酸数据待定

章张老师能否帮助确认一个氨基酸数据集。

# 对比试验表格

## 实验一，变量为模式串长度m，

测4组，即上面的测试集.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| len(m) | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 | 4096 | 8192 |
| new |  |  |  |  |  |  |  |  |  |  |
| epsm |  |  |  |  |  |  |  |  |  |  |
| tvsbs |  |  |  |  |  |  |  |  |  |  |
| ufndmq |  |  |  |  |  |  |  |  |  |  |
| hashq |  |  |  |  |  |  |  |  |  |  |
| fsbndmq |  |  |  |  |  |  |  |  |  |  |

我建议从m=4开始。m=｛4，8，12，16，32 64 128 256 512 1024，2048｝

M值的划分最初我是按epsm来的，他对长度16以上和16下做了不同的优化，所以改进的算法也是从m=16开始测的，因此不能保证长度4、8和12效果同样好。不过结果待确认。

对上述实验分别进行改进二、改进2+三、改进2+四进行测试。

1. 几个算法横向比较，取200m；
2. 本算法在三种数据下和自己比；
3. 同上。

## ~~实验二，变量为匹配串长度n~~

~~可以测多组，每组的模式串长为实验一里描述~~

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ~~len(n)~~ | ~~4m~~ | ~~20m~~ | ~~100m~~ | ~~200m~~ | ~~1000m~~ |
| ~~new~~ |  |  |  |  |  |
| ~~epsm~~ |  |  |  |  |  |
| ~~tvsbs~~ |  |  |  |  |  |
| ~~ufndmq~~ |  |  |  |  |  |
| ~~hashq~~ |  |  |  |  |  |
| ~~fsbndmq~~ |  |  |  |  |  |

这个实验应该没什么必要。

## ~~实验三，变量为串字典大小，即DNA的4和AMINO ACID的20~~

|  |  |  |
| --- | --- | --- |
|  | ~~4~~ | ~~20~~ |
| ~~new~~ |  |  |
| ~~epsm~~ |  |  |
| ~~tvsbs~~ |  |  |
| ~~ufndmq~~ |  |  |
| ~~hashq~~ |  |  |
| ~~fsbndmq~~ |  |  |

## 实验四，改进二的块读取大小，4-8之间

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 块读取大小 | 4 | 5 | 6 | 7 | 8 |
| new |  |  |  |  |  |
| epsm |  |  |  |  |  |
| tvsbs |  |  |  |  |  |
| ufndmq |  |  |  |  |  |
| hashq |  |  |  |  |  |
| fsbndmq |  |  |  |  |  |

hash(p,seed)

1. crc = \_mm\_crc32\_u64(seed,p)
2. filter = crc&mask

pre(p,m,t,n)

1. for i ← 1 to count of all string do
2. if suffix of s equals prefix of p
3. then shift [ hash(s,seed) ] ← idx
4. for i ← 0 to m – σ do
5. filter ← hash(p+I,seed)
6. if i == m – σ
7. then endfilter = filter
8. then shift [ filter ] = 1
9. else
10. if shift [ filter ] > m - i – σ
11. then shift [ filter ] = m - i – σ

search(p,m,t,n)

1. endptr ← t – σ
2. ptr ← p – σ
3. while ptr < endptr
4. filter ← hash(ptr,seed)
5. if filter == endfilter
6. then compare(p,ptr,m)
7. ptr += shift [ filter ]

hash函数是用来计算定长字符串对应的数值，它先将定长字符串转成ulong，再调用sse得到ulong的CRC值，最后将其与上掩码，得到int数值。

Pre函数是用来计算偏移值数组的，

Search函数负责对串采样，然后根据计算的hash值决定下一步前进多少，如果为一，检查是否完全匹配。