维普资讯 http://www.cqvip.com

Tandem repeat 查找方法比较*)

徐恒宇　王　镝　王国仁　郑若石
（东北大学信息科学与工程学院　沈阳 110005）

摘　要　Tandem repeat 在基因组成和进化中起到非常重要的作用，查找和分析 Tandem repeat 已经成为当前生物信息学的一个前沿领域和研究焦点。目前在这一研究领域存在多类解决方法，主要有基于 LZ 分解技术的方法和最近兴起的基于后缀树索引的方法。本文选取了两种时间复杂度达到 O(*n*log*n*)数量级的代表性的方法，对这两种方法进行了全面的综述，并对它们的性能进行了系统的比较和分析。

关键词　Tandem repeat，LZ 分解，后缀树

Methods of Finding Tandem Repeat in String

XU Heng-Yu　WANG Di　WANG Guo-Ren　ZHENG Ruo-Shi
(School of Information Science & Engineering, Northeastern University, Shenyang 110005)

Abstract　Tandem repeat takes such an important role in gene composition and evolution that the search and analysis of tandem repeat have become one of the front domain and research focus. There are multiple methods in recent works, mainly including two methods. One is based on LZ decomposition, and the other is based on suffix tree index. This paper summarize these two O(*n*log*n*) methods and takes a thorough analysis and comparison of their performance.

Keywords　Tandem repeat, LZ decomposition, Suffix tree

1　引言

20 世纪后期，生物科学技术迅猛发展，无论从数量上还是从质量上都极大地丰富了生物科学的数据资源。尤其是随着基因测序技术的快速发展以及人类基因组项目[1,2]的启动，人们对各种生物序列（包括 DNA、RNA、蛋白质）的结构进行了非常深入的研究并得到了大量的测序结果。

对 DNA 序列的试验分析表明序列中含有大量的 tandem repeat，例如，在人类 DNA 序列中 tandem repeat 的片段在整个序列中的比例超过 50%[1]。近年来对 DNA 序列中 tandem repeat 的分析试验表明 tandem repeat 在基因组成和进化中起到非常重要的作用[3,4]，同时它是产生多种疾病的病因[5,6]，如：脊髓延髓性肌萎缩和弗里德赖希（氏）共济失调症，强直性肌营养不良等多种基因遗传疾病[7,8]。因此对 DNA 序列中 tandem repeat 的识别和分析就成为生物信息学中的一个非常重要的研究领域。

对于一个字符串 S，长度为 n，tandem repeat 是其中的子串，其形式为 $\alpha\alpha$（α 为非空子串）。若 S 中有 m 个连续的 α 构成一个子串，则称其为 tandem repeats，称 α 为模式。S 中可能含有多达 $n^4/4$ 个 tandem repeat，识别所有这些 tandem repeat 需要的时间由这些 tandem repeat 的个数 z 决定，早期的一些算法识别所有 tandem repeat 需要 $O(n\log n + z)$ 的时间复杂度，它们通过传统的字符串比较和查找算法进行 tandem repeat 的查找，如 Crochemore 算法，LZ 分解，least common ancestor 算法，Boyer-moore 算法和 Z 算法[9~11]。后来随着后缀树存储结构越来越广泛地应用，一些基于后缀树的查找方法也被采用了，但是都没有达到 O(*n*log*n*) 甚至是线性的时间复杂度[5]。而后有一些方法通过限制 tandem repeat 的输出个数（仅输出 pri- mitive tandem repeat），使算法的复杂度达到 O(*n*log*n*)[11~13]。如果一个 tandem repeat 的模式 α 不能表示成任意其他的 tandem repeat，则称其为 primitive tandem repeat。

目前在这一研究领域有多种比较实用的查找方法，其中具有代表性的两类方法是基于 LZ(Lempel_Ziv)分解技术的查找方法和基于后缀树索引结构的查找方法。本文详细介绍上面提到的这两类有代表性的方法，分别实现了这两种算法，并在实验结果的基础上对它们作出了性能分析和比较。

本文第 2 节详细介绍了基于 LZ 分解的查找方法，第 3 节详细介绍了基于后缀树索引结构的查找方法，第 4 节对两种方法进行了理论分析和实验比较，最后对论文做出了简要的总结。

2　基于 Lempel_Ziv(LZ)分解的查找方法

2.1　基本概念介绍

在这一节中首先详细介绍 tandem repeat 相关的概念，前面已经给出了 tandem repeat 及原始(primitive)tandem repeat 的概念。接下来介绍字符串中 tandem repeat 的表示方法，假设一个字符串 S 的长度为 n，则 $S[i\cdots j]$ 表示 S 中从位置 i 开始到位置 j 结束的子串，其中 $0<i\leqslant j\leqslant n$，$S$ 中的一个 tandem repeats $\alpha\cdots\alpha=S[i\cdots i+m-1]$，可以表示为一个三元组 (i,α,k)，其中 α 为模式(pattern)，k 为 α 重复次数，如果 $k=2$，则其为 tandem repeat，可以简写为 (i,α) 或 (i,l)，其中 $l=2|\alpha|$，叫

*)基金项目：教育部高等学校优秀青年教师教学科研奖励计划基金资助项目；国家自然科学基金(60273079)资助。徐恒宇　硕士研究生，研究方向为生物信息数据库。王　镝　博士研究生，研究方向为生物信息数据库。王国仁　教授，博士生导师，主要研究方向为数据库理论和技术、分布与并行数据库、Web 数据管理技术、生物信息管理、面向对象数据库。郑若石　硕士研究生，研究方向为生物信息数据库。

维普资讯 http://www.cqvip.com

作 tandem repeat 对。

下面给出两个关于 tandem repeat 的集合 Occurrence 和 Vocabulary。Occurrences 集合包含 S 中的所有的 tandem repeat 对。在 Vocabulary 集合中每个类型的 tandem repeat 只出现一次。Tandem repeat 的类型是由它的模式决定的，只有 $\alpha=\alpha'$ 时，tandem repeat $\alpha\alpha=S[i\cdots i+m-1]$ 和 $\alpha'\alpha'=S[i'\cdots i'+m'-1]$ 才属于同一类型。为了方便，可以用某一个特定的 tandem repeat 来代表它所属的类型。如可以用 (i,α) 来代表 $\alpha\alpha$ 所属的类型。对于 Vocabulary 集合中的任意一个类型 $\alpha\alpha$，Occurrences 集合中的每一个和它类型相同的 tandem repeat 都叫做 $\alpha\alpha$ 的一次出现，其中在序列中出现位置最小的那个 tandem repeat 叫做最左出现。例如给定字符串 abaabaabbaaabaaba $，Occurrences 集为{(1,6),(2,6),(3,2),(3,6),(6,2),(8,2),(10,2),(11,2),(11,6),(12,6),(14,2)}，而{(1,6),(2,6),(3,2),(3,6),(8,2)}为一个 Vocabulary 集，类型(3,2)在 Occurrences 集合中的出现是(3,2),(6,2),(10,2),(11,2),(14,2)。其最左出现为(3,2)。

在 S 中，如果 $(i,m),(i+1,m),\cdots,(j,m)$ $(0<i<j\leqslant n)$ 都是 tandem repeat 对，则序列 $i,i+1,\cdots,j$ 叫作长度为 m 的 tandem repeat 的一个 run。对于任意的 $i<j$，当且仅当从 i 开始的长度为 m 的 tandem repeat 的一个 run 包含了 j 时，称 tandem repeat (i,m) 覆盖了 tandem repeat (i,m)。由此 (j,m) 可以由 (i,m) 经过一系列的右转来得到(设 ω 为一个子串，a 为一个字母，则子串 ωa 叫作子串 $a\omega$ 的一个右转)。而且，在这个过程中每次右转所产生的子串也是一个长度为 m 的 tandem repeat。

这里给出 S 中 tandem repeat 的覆盖集 P，当且仅当 Vocabulary 中任意一个 tandem repeat 在 Occurrence 中的多次出现至少有一次可以被 P 中的某个 tandem repeat 覆盖(cover)，则称 P 为一个覆盖集，需要指出的是覆盖集 P 不是唯一的。当且仅当 Vocabulary 中的任意一个 tandem repeat 在 Occurrence 中的最左出现可以被 P 中的某个 tandem repeat 覆盖，称 P 为一个最左覆盖集，最左覆盖集同样不是唯一的。在上面的例子中{(1,6),(8,2),(11,2)}为一个包含集，{(1,6),(3,2),(8,2)}为一个最左包含集。

2.2 LZ 分解

对字符串的 LZ 分解是很多字符串查找和匹配算法的基础，LZ 分解产生的分解块有着很多特殊性质，下面将介绍 LZ 分解和 tandem repeat 在 LZ 分解块上的性质。

对长度为 n 的字符串 S 的任意一个位置 i，定义 m_i 为子串 $S[i\cdots n]$ 中满足下面条件的所有前缀的最大长度：

$S[i,\cdots,i+m-1]$ 为 $S[i\cdots n]$ 的前缀，长度为 m，且此前缀 $S[i,\cdots,i+m-1]$ 在 S 中的某位置 $j(j<i)$ 出现过，即 $S[i,\cdots,i+m-1]=S[j,\cdots,j+m-1]$

即 $m_i=\max(m), m\in\{m\mid \exists j(j<i\wedge S[j,\cdots,j+m-1]=S[i,\cdots,i+m-1])\}$

令 s_i 为满足 $S[i,\cdots,i+m_i-1]=S[j,\cdots,j+m_i-1]$ 的所有 j 的最小值，即在 S 中第一个出现的和最长前缀 $S[j,\cdots,j+m_i-1]$ 相等的子串的开始位置。即：

$s_i=\min(j), j\in\{j\mid 0\leqslant j<i\wedge S[j,\cdots,j+m_i-1]=S[i,\cdots,i+m_i-1]\}$

字符串 S 的 LZ 分解就是它的一个位置索引列表 $i_1,i_2,\cdots,i_k$，其值归纳定义如下：

$i_1=1, i_{B+1}=i_B+\max(1,l_{iB}), i_B\leqslant n$，子串 $S[i_B,\cdots,i_{B+1}-1]$(其中 $1\leqslant B\leqslant k$)就叫做 LZ 分解的第 B 个块。S 的 m_i 值，s_i 值，还有 LZ 分解值的一个例子如下表所示。

text	a	b	a	a	b	a	a	b	b	a	a	a	b	a	a	b	a	$
i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
l_i	0	0	1	5	4	3	2	1	3	2	6	6	5	4	3	2	1	0
s_i	0	0	1	1	2	3	1	2	2	3	3	1	2	3	1	2	1	0

LZ 分解：

a	b	a	a b a a b	b a a	a b a a b a	$
1	2	3	4	5	6	7

2.3 tandem repeat 在 LZ 分解块上的性质

本节将介绍 tandem repeat 在 LZ 分解块上的几个重要性质，tandem repeat 查找算法就是基于这些性质依次扫描每个 LZ 分解块最终得到所有的 tandem repeat。

性质 1 任何一个 tandem repeat 的右半边最多存在于连续的两个 LZ 分解块上。

性质 2 任何一个类型的 tandem repeat 的最左出现至少存在于两个连续的 LZ 分解块上。

性质 3 任何一个类型的 tandem repeat 的最左出现的中心若位于 LZ 分解块 B 上，则其必满足下面一种情况：1) 此最左出现的左端点在 B 块上而右端点在 $B+1$ 块上；2) 此最左出现的左端点在 $B-1$ 块上或者在更左边的块上，同时右端点在 B 块上或是 $B+1$ 块上。

以上介绍的这些性质的详细证明过程请参考文[14]，下面再简要介绍一下算法中要用到的另一个概念。

对于字符串 S 中任意的两个位置 i,j，定义从 i 开始和从 j 开始的最长的相同子串的长度为 i 和 j 的前向最大相同前缀。同样地，定义以 i 为结尾和以 j 为结尾的最长的相同子串的长度为 i 和 j 的反向最大相同后缀。

2.4 基本算法

虽然遍历每个 LZ 分解块，按照性质 1 来进行过滤就能查找出所有的 tandem repeat，但是此种方法的过滤效率低，算法的查找速度慢。本文根据性质 3 的两个情况来找到前面介绍的最左包含集，同时将待查序列存储在后缀树上。然后从最左包含集出发，通过后缀链的游走在后缀树上标出所有的 tandem repeat 的结束位置，最后经过一个后缀树的遍历就能得到所有的 tandem repeat。

下面给出查找最左包含集的算法，此算法分两步，分别对应性质 3 中的两种情况。下面列出的算法假设当前处理的是第 B 块 LZ 分解块，在每个分解块上分别应用本算法就能得到完整的最左覆盖集。此算法是基于 LZ 分解查找 tandem repeat 的核心部分，由于篇幅关系，我们略去算法其他部分。

```
算法a:
    h = |B|
    h1 = |B+1|
    For k form 1 to |B|do
    Begin
        Let q = h1 - k
        计算位置 h1 和 q 开始的前向最大相同前缀，用 k1 表示
        计算位置 h1-1 和 q-1 开始的后向最大相同前缀，用 k2 表
```

维普资讯 http://www.cqvip.com

```
    示
    If(k1+ k2>=k and k1>0 )
      输出 tandem repeat(max(q- k2, q-k+1),2k)
  End
算法b:
  h = |B|
  h1= |B+1|
  For k from 1 to |B|+|B+1| do
  Begin
    Let q = h+ k
    计算位置 h 和 q 开始的前向最大相同前缀,用 k1 表示
    计算位置 h-1 和 q-1 开始的后向最大相同前缀,用 k2 表示
      If(k1+ k2>=k and k1>0 and k2>0and max(h-k2, h
        -k+1)+k < h1)
        输出 tandem repeat(max(h- k2, h-k+1),2k)
  End
```

3 基于后缀树的查找方法

3.1 基本概念介绍

本节介绍几个重要的概念,基于后缀树的查找算法就是围绕这几个概念展开的:

字符串 S 中的一个 tandem repeat $\alpha\alpha = S[i,\cdots,i+2|\alpha|-1]$如果满足条件:$S[i+|\alpha|] \neq S[i+2|\alpha|]$,则称其为分支 tandem repeat。

和分支 tandem repeat 概念相对应的就是非分支 tandem repeat。

设 ω 为一个子串,a 为一个字母,则子串 $a\omega$ 叫作子串 ωa 的一个左转。

下面介绍和上面概念相关的一个重要的定理。

定理 1 任意一个非分支 tandem repeat$(i,a\omega)$都是一个以 $i+1$ 为起始位置的另一个 tandem repeat$(i,\omega a)$的左转。这个以 $i+1$ 为起始位置的 tandem repeat 可能是分支也可能是非分支 tandem repeat。

由定理 1 可知,一个 tandem repeat 或者是一个分支 tandem repeat,或者是由一个分支 tandem repeat 经过一系列左转过程得到的非分支 tandem repeat。对于字符串 S 中的任何一个 tandem repeat$(i+1,\omega a)$,如果 $S[i] = a$,则存在一个以 i 开始的 tandem repeat$(i,a\omega)$。因此,从一个分支 tandem repeat 出发,经过一系列左转过程我们可以得到一系列的非分支 tandem repeat。

3.2 后缀树预处理

后缀树是一种非常重要的数据结构,它在与字符串处理相关的各种领域里有着非常广泛的应用[16~18],详细介绍后缀树的文章有文[19~21]等。

$T(S)$代表字符串 S 对应的后缀树,u,v 分别代表后缀树 $T(S)$上的两个节点(叶子节点或中间节点)。$L(u)$代表节点 u 在后缀树中的路径(从根节点到 u 节点的路径所代表的子串)。$D(u) = |L(u)|$ 为这个路径的长度。设 v 为叶子,如果 $L(v)=S[i\cdots n]$,则置 v 的标号为 i,定义中间节点 u 的 leaf_list 为它所包含的所有叶子节点标号的一个集合列表,简写为 $LL(u)$。下图 1 就是字符串 *mississippi* 的带有 LL 的后缀树。

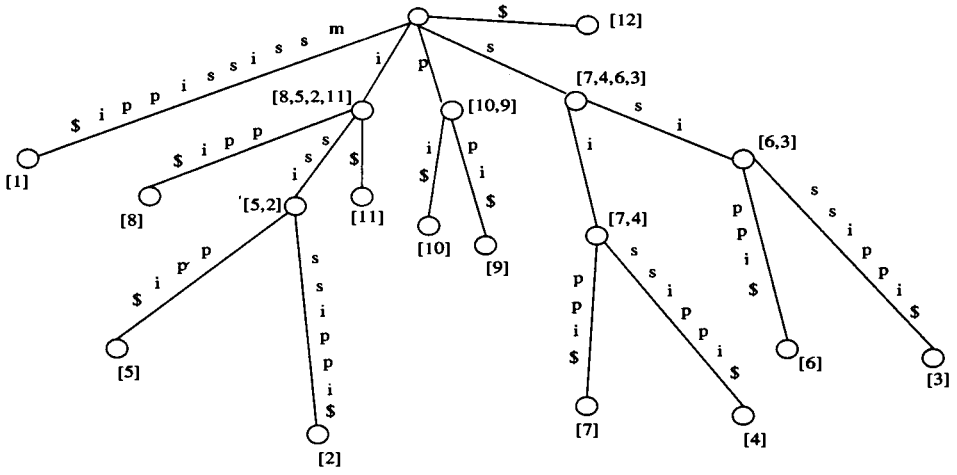

图 1 标注好的后缀树

3.3 tandem repeat 在后缀树上的性质

下面介绍 tandem repeat 在后缀树上的几个性质[22~25]。

定理 2 字符串 S 中的两个位置 $i,j,1\leqslant i<j\leqslant n$,令 $m=j-i$,则下面两个断言是等价的:1)S 中存在一个以位置 i 为起点的长度为 $2m$ 的 tandem repeat$(i,2m)$;2)$T(S)$中存在一个节点 u 满足 $D(u)\geqslant m$,使得 i,j 同时存在于 leaf_list(u)中。

从定理 2 可以得到分支 tandem repeat 在后缀树上的性质,即定理 3。

定理 3 字符串 S 中的两个位置 $i,j,1\leqslant i<j\leqslant n$,令 $m=j-i$,则下面两个断言是等价的:1)S 中存在一个以位置 i 为起点的长度为 $2m$ 的分支 tandem repeat(i,2m);2) $T(S)$中存在一个节点 u 满足 $D(u)=m$,使得 i,j 同时存在于 leaf_list(u)中。但是对任何 v 满足 $D(v)>m$,i,j 不同时存在于 leaf_list(v)中,即对任何 u 的孩子 v 而言,i,j 不同时存在于 leaf_list(v)中。

3.4 基本算法

由定理 3 首先得到查找所有分支 tandem repeat 的基本算法,然后对所有找到的分支 tandem repeat 进行左转操作就可以得到所有的 tandem repeat。下面就是查找所有分支 tandem repeat 的基本算法:

```
tandem repeat 查找算法:
  All nodes are unmarked;
  while (not all node are marked)
    select an unmarked internal node v. ;
    mark v ;
    collect the leaf-list of v, LL(v);
    for each leaf-label i in LL(v)
      if j =i + D(v) is in LL(v) and S[i]! =S[i+2D(v)]
        (i,2S(v)) is a branching tandem repeat;
        Laft_rotation(i,2S(v));

左转算法 Laft_rotation(poistion,length):
i = positon-1;
While(S[i]==[i+length] )
```

维普资讯 http://www.cqvip.com

(i,2S(v))is a non-branching tandem repeat;
i——;

4 两种方法的分析比较

在这一节,我们将从理论和实验结果两方面分析比较这两个算法。

4.1 理论分析

下面先从理论上简单分析两种算法的时间复杂度,第一种算法首先在 O(n)的时间复杂度内对字符串进行了 LZ 分解处理[26],同时在线性的时间内能找到任意给定的两个位置开始的最大相同前缀。因此查找最左包含集这部分算法的时间复杂度为 O(n),同样在 O(n)的时间内能在后缀树上标注出所有的 tandem repeat,最后再遍历后缀树得到所有的 tandem repeat,共需要时间 O(nlogn+z)。

第二种算法在 O(nlogn)的时间复杂度内找到所有的分支 tandem repeat,然后对每个分支 tandem repeat 进行左转操作来找到所有的 tandem repeat,整个算法的时间消耗也是 O(nlogn+z)。

4.2 实验结果

本文对两种算法进行测试的时候选择的数据是含有 tandem repeat 为 58%的蠕虫的染色体 chr1 的片断和含有 tandem repeat 为 72%的人类染色体 chr22 的片断[27]。

两种算法都是用 C++在 Windows2003 Server 上实现的,运行环境为 DELL P4 2.0 服务器(2G RAM)。两种算法的测试结果分别在图 2 和图 3 中。算法 1 是基于 LZ 分解的算法,算法 2 是基于后缀树的算法。

从实验结果可知,两种算法时间开销随着测试数据的增长而增加,且算法的时间消耗和数据集的选择无关,这是由于这两个算法都是基于遍历的形式来产生结果的,对数据含有的 tandem repeat 的个数不敏感。

同时第一种算法的时间消耗和第二种算法的时间消耗随着数据的变大差距也变大,这是由于第一种算法在进行了 LZ 分解以后,先遍历字符串序列找到最左包含集并在后缀树上进行标注,这时不输出已经识别的这部分结果,随后由得到的最左包含集出发,通过后缀链在后缀树上遍历并标注出所有的 tandem repeat,然后再遍历后缀树输出所有的 tandem repeat。显然如果在第一步输出了已经得到的部分结果,在随后的遍历后缀树的过程中还要判断哪些结果已经被输出了。所以第一种算法重复遍历后缀树,造成了时间开销的增加。下面的数据显示由于第一种算法对后缀树的附加遍历,其查找时间比第二种算法大得多。

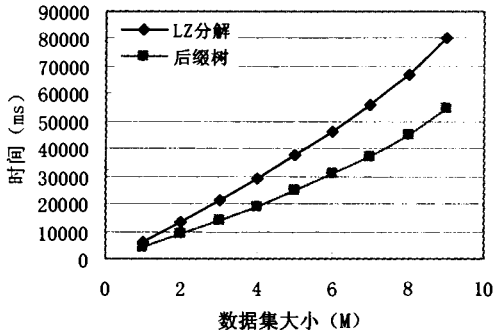

图 2 蠕虫 chr1 数据结果

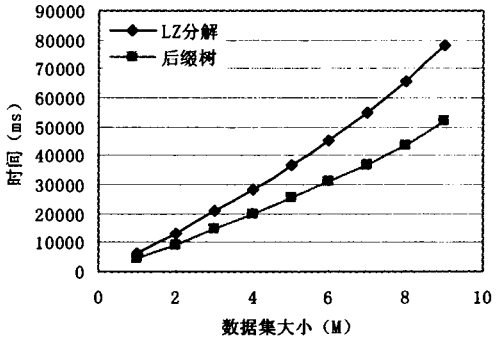

图 3 人类 chr22 数据结果

表 1 和表 2 为两种算法的预处理部分的时间消耗,算法一的预处理就是 LZ 分解部分,算法二的预处理就是对后缀树进行标注部分。表 1 的数据是蠕虫 chr1 的结果,表 2 的数据是人类 ch22 的结果,可见,算法一中 LZ 分解的时间消耗要稍微小于算法二中对后缀树进行预处理所消耗的时间。

表 1 两种算法的预处理部分的时间消耗(蠕虫 chr1)

	1M	2 M	3 M	4 M	5 M	6 M	7 M	8 M	9 M
算法一	628.5	1083.0	2174.1	2992.9	3827.3	4718.5	5565.1	6444.6	7199.9
算法二	1272.21	2628.3	4003.1	5445.9	6898.9	8393.4	9915.2	11559.2	13220.8

表 2 两种算法的预处理部分的时间消耗(人类 ch22)

	1M	2 M	3 M	4 M	5 M	6 M	7 M	8 M	9 M
算法一	617.3	1368.4	2098.4	2914.4	3652.0	4493.9	5320.0	6164.7	7013.4
算法二	1284	2659.9	4073.4	5558.3	7117.1	8695.7	10361.8	11884	13594.2

表 3 和表 4 为两种算法的查询部分的时间消耗,表 3 的数据是蠕虫 chr1 的结果,表 4 的数据是人类 ch22 的结果,可见,算法一中查询部分消耗的时间要远远大于算法二中查询部分消耗的时间。这样就导致了算法一在总时间上比算法二要慢很多。

表 3 两种算法的查找部分的时间消耗(蠕虫 chr1)

	1M	2 M	3 M	4 M	5 M	6 M	7 M	8 M	9 M
算法一	3710.1	7838.8	12251.8	16980.5	21734.4	26715.6	31014.3	36209.2	41437.6
算法二	1025.6	2303.1	3522.9	4822.3	6080.4	7573.2	8843.4	10255.4	11735.5

(下转第 192 页)

维普资讯 http://www.cqvip.com

System. loadLibrary("vtkImagingJava");

载入所需库，生成动态链接库 DLL(Dynamic Link Library)，对调用的方法用关键字 native 做本地声明，具体实现由 C 语言完成，生成对应的 class 文件和头文件。

(5)图像处理算法的实现。图像处理算法采用了邻域运算操作。该操作涉及到两个矩阵，像素矩阵和模板。像素矩阵对应图像，模板对应相应操作。将待处理像素点同它周围若干像素点像素值经过运算后得到的结果，替代原有像素点的像素值。

结论与展望 基于 Web 服务的图像处理系统 WIP 采用分布式架构和服务实现技术，很好地解决了现有图像处理系统中处理速度和图像存储问题，适应性与灵活性增强，维护开销减少。该系统作为底层运行平台，加上某些领域的特殊性，可应用至宇航领域，地理环境领域，电视广告媒体的图像处理。从理论研究到系统推广应用有一个漫长历程，WIP 系统主要针对医学图像处理，其它领域的应用开发正在扩展研究中。WIP 系统的实现，为 WSRF 规范和网格计算应用奠定了基础，利用已有计算资源和图像资源，可以更好地建立一个健壮的、可靠的、全球化的图像处理系统。

参 考 文 献

1 W3C Web Service 相关标准. http://www.w3c.org,2003-06

2 UDDI 相关标准. http://www.uddi.org

3 BEA Weblogic 技术资料. http://e-docs.bea.com/

4 Ali Akbar 著，邱仲潘等译. BEA Weblogic Server 管理指南. 机械出版社

5 Castleman K R 著，朱志刚等译. 数字图像处理. 电子工业出版社

(上接第 175 页)

表 4 两种算法的查找部分的时间消耗(人类 chr2)

	1M	2 M	3 M	4 M	5 M	6 M	7 M	8 M	9 M
算法一	3829.7	7975.1	12408.1	17255.4	22188.4	26938.2	32120.4	36908.5	42305.4
算法二	1023.8	2305.5	3631.7	4951.5	6406.1	7815.1	9207.9	10656.1	12359.9

结论 本文描述了生物信息领域的一个新问题及近年来出现的一些解决方法，综述了在 tandem repeat 查找领域里的两种高效的方法，并对各方法做了简要的实验分析。

近年，随着基因测序工作的蓬勃发展，研究者可以在全球范围内获得大量的研究数据。因此如何快速地发现已有的生物数据所隐含的生物学意义就变得十分重要了。尤其当前的一些应用中，如基因致病的研究，人类免疫的研究都要求对 DNA 序列中的 tandem repeat 进行发现和识别，为了进一步提高查找的速度，就出现了上述的多种查找方法。为了加快查找算法的时间和空间复杂度，就要深入研究现有的算法并加以改进和创新，这就是本文写作的最初的想法。

参 考 文 献

1 The Human Genome Project (HGP). http://www.nhgri.nih.gov/HGP/

2 Collins F, Patrinos A, Jordan E, Chakravarti A, et al. New goals for the us human genome project: 1998-2003. Science, 1998, 282(5389): 682～689

3 Campuzano V, Montermini L, Molto M D, et al. Friedreich's ataxia: autosomal recessive disease caused by an intronic gaa triplet repeat expansion. Science, 1996, 271: 1423～1427

4 Huntington's disease collaborative research group. A novel gene containing a trinucleotide repeat that is expanded and unstable on huntington's disease chromosomes. Cell, 1993, 72: 971～983

5 Du J, Zhu Y, Shanmugam A, Kenter A L. Analysis of immunoglobulin sgamma3 recombination breakpoints by pcr: implications for the mechanism of isotype switching. Nucleic Acids Research, 1997, 25: 3066～3073

6 Fu Y H, Pizzuti A, Fenwick R, et al. An unstable triplet repeat in a gene related to myotonic muscular dystrophy. Science, 1992, 255: 1256～1258

7 La Spada A R, Wilson E M, Lubahn D B, et al. Androgen receptor gene mutations in x-linked spinal and bulbar muscular atrophy. Nature, 1991, 352: 77～79

8 Verkek A J M H, Pieretti M, Sutcliffe J S, et al. Identification of a gene (fmr-1) containing a cgg repeat coincident with a breakpoint cluster region exhibiting length variation in fragile x syndrome. Cell, 1991, 65: 905～914

9 Main M G, Lorentz R J. An O(nlogn) algorithm for finding all repetitions in a string. J. Algorithms, 1984, 5: 422～432

10 Landau G M, Schmidt J P. An algorithm for approximate tandem repeats. In: A. Apostolico, M. Cro-chemore, Z. Galil, and U. Manber, eds. Combinatorial Pattern Matching: 4th Annual Symposium, CPM93. Padova, Italy, June 1993. Proc. number 684 in Lecture Notes in Computer Science, Berlin, Springer Verlag, 1998, 3. 120～133

11 Stoye J, Gusfield D. Simple and flexible detection of contiguous repeats using a suffix tree. In: M. Farach, ed. Combinatorial Pattern Matching: 9th annual symposium, CPM98. Piscataway, New Jersey, USA, July 1998. Proc. number 1448 in Lecture Notes in Computer Science, Berlin, Springer Verlag, 1998. 140～152

12 Crochemore M. An optimal algorithm for computing the repetitions in a word. Inf. Process. Lett., 1981, 12(5): 244～250

13 Apostolico A, Preparata F P. Optimal off-line detection of repetitions in a string. Theor. Comput. Sci., 1983, 22: 297～315

14 Gusfield D, Stoye J. Linear time algorithms for finding and representing all the tandem repeats in a string: [Report CSE-98-4]. Department of Computer Science, University of Califonia, Davis, 1998

15 Stoye J, Gusfield D. Simple and flexible detection of contiguous repeats using a suffix tree. Department of Computer Science. University of California, Davis, CA 95616, USA

16 Apostolico A. The myriad virtues of subword trees. In: A. Apostolico, Z. Galil, eds., Combinatorial Algorithms on Words, vol. F12 of NATO ASI Series, Springer, Berlin, 1985. 85～96

17 Crochemore M, Rytter W. Text Algorithms, Oxford University Press, New York, NY, 1994

18 Gus. eld D. Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cambridge University Press, New York, NY, 1997

19 Maab M. Suffix trees and their applications. Sept. 1999, 13

20 Hunt E. PJama stores and suffix tree indexing for bioinformatics applications. In: Proc. of ECOOP'00, 2000

21 Grossi R, Italiano G F. Suffix Trees and their Applications in String Algorithms. In: Proc. 1st South American Workshop on String Processing (WSP 1993), Sep. 1993. 57～76

22 Crochemore M. An optimal algorithm for computing the repetitions in a word, Inform. Process. Lett., 1981, 12 (5): 244～250

23 Apostolico A, Preparata F P. Optimal oJ-line detection of repetitions in a string. Theoret. Comput. Sci., 1983, 22: 297～315

24 Kosaraju S R. Computation of squares in a string, In: M. Crochemore, D. Gus. eld, ed. Proc. Fifth Ann. Symp. on Combinatorial Pattern Matching, CPM 94, vol. 807 of Lecture Notes in Computer Science, Springer, Berlin, 1994. 146～150

25 Gus. eld D. Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cambridge University Press, New York, NY, 1997

26 Rodeh M, Pratt V R, Even S. Lineat algorithm for data compression via string matching. J. AVM, 1981, 28(1): 16～24

27 ftp://ncbi.nlm.nih.gov/genbank