# 引言

## 研究背景

随着计算机的普及和软件的发展，每个人都变成了数据的产生者，大家通过网络共享数据，但信息共享不但没有减少数据，反而使得数据急速膨胀。各大公司的数据规模在不断增大，而数据规模的增大往往导致查找时间增加，如何在大规模文本数据中高效的查找，筛选出有用信息，如何对大量文本信息进行高效的存储，都是目前研究的热点问题。

对于大规模字符串，目前的主要研究方向是查找和模式匹配。生物信息学的基因序列比对是一个典型的模式匹配问题。基因数据库可以看成一个大规模的字符串，给定的基因序列可以看成一个信息串，基因序列比对就可以看成在一个大规模字符串中查找给定字符串的过程。此类问题的关键在于高效的索引构造和查找算法，现有的基因序列比对软件主要采用后缀树【】和后缀数组【】等后缀相关的数据库来构造基因数据库的索引。

后缀树是由Weiner 于1973年提出的一种能够有效解决字符串匹配和查询的数据结构。它是一颗有向的把所有后缀保存并压缩的字典树，因此后缀树在解决某个字符串s1是否在另外一个字符串s2中出现，指定字符串s1在字符串s2中出现的次数，两个字符串s1，s2最长的公共前缀等问题上有很好的性能。虽然后缀树也可以在O(n)时间内构造【】，但是其空间开销大且对大字母集的效率不太理想。

后缀数组是由U. Manber和G. Myers提出的一种代替后缀树的数据结构。它存储的是所有后缀按照字典顺序进行排序后的后缀首字符在原字符串中的位置，也就是说，后缀数组的第i个数就是字典序第i小的后缀编号。后缀数组是后缀树一个精美的代替品比后缀树容易编程实现，能够实现后缀树的很多功能且时间复杂度并不逊色，而且它比后缀树所占用的内存空间少很多。因此后缀数组比后缀树更有应用前景。

目前行业内的专家们已经提出了很多线性时空内构造后缀数组的算法，这些算法基本上都是单机的内存算法，虽然对于单机内存不能计算的数据量可以通过外存构造算法解决，但是单机的外存构造算法相对于单机内存构造算法要慢的多。因此利用充分利用多机内存并行高效计算后缀数组就变得非常迫切。本文讲述的就是一种多机并行计算广义后缀数组的算法。

## 研究现状

U. Manber和G. Myers在文献【】中首次提出了后缀数组，其定义如下：设是一个长度为的字符串即，定义从i位置开始的后缀，定义后缀数组为，那么是第小的后缀在中的位置。因此根据后缀数组的定义可得。 自从后缀数组的概念被提出之后，许多研究人员也着手研究后缀数组的构造算法，从文献【】可知，根据算法的特点可以将这些构造算法大致分为三类：倍增算法、归纳排序算法、分治递归算法。下面将概括的描述下这三类算法。

倍增算法：倍增算法利用名次数组和后缀数组可以在时间内转化的性质，先求名次数组，再来计算后缀数组。名次数组计算的方法是：在第一轮排序中对长度为的字符串进行排序，记录其名次数组；第二轮将上一轮的名次数组做为输入，对长度为的字符串进行排序，如此循环下去，直到轮结束。此时的名次数组就是最终的名次数组，根据名次数组求出后缀数组。每一轮的排序采用基数排序时间复杂度为，一共需要次，因此倍增算法的时间复杂度为。比较有名的倍增算法有算法【】，算法【】等等，但是由于采用了一些额外的辅助数据结构，这类算法的缺点就是空间消耗较大。

归纳排序算法：首先按照某种特定的规则从所有子集中选出符合规则的后缀，然后把选出的后缀子集排序之后，再归纳出那些未被选出的后缀顺序。这类算法不采用递归的方法，从文献【】可知，这类算法在某些情况下性能非常好，但是不够稳定。其最坏的时间复杂度是，但在实践应用中性能比一些O(n)的算法还要好。原因是这类算法属于轻量级的算法，所占用的空间较小，同时这类算法大都采用迭代的方式来排序子集，容易实现。比较有名的MP【】算法，KA【】算法都属于归纳排序算法。

分治递归算法：首先按照某种特定的规则从所有子集中选出符合规则的后缀，把选出的后缀子集进行排序命名，形成一个更短的字符串，如果该字符串不存在相同元素，则可以直接求出该字符串的排名顺序，否则就递归求该字符串的排名顺序。求出该字符串的排名顺序之后，回溯求出上一代的未被选取的后缀的排名，这样一步步求出整个字符串所有后缀的排名。分治递归算法与归纳排序算法有些类似，不同之处在于归纳排序采用迭代的方式来实现，而分治递归算法采用递归的方式来实现。典型的分治递归算法有KA【】算法，KS【】算法。

除上述几类算法之外，之后又出现了同时用归纳排序和递归来构建后缀数组的算法，比如SAIS算法。该算法兼用了归纳排序算法和分支递归算法的优点，既保证了时间复杂度为O(n)，空间消耗又比单纯的分治递归小。据文献【】可知，理论上构造后缀数组的空间复杂度最低为5n，而SAIS算法的空间复杂度只有5.12n，可见该算法的优美。本文后面会详细介绍该算法。

目前针对多文件的后缀数组（广义后缀数组）构造，如果内存够用，则在每个文件后加一个不出现在字母表并比所有字母表中的元素小的哨兵，合并所有文件，然后放入内存中计算；如果内存不够，则利用单机的广义后缀数组的构造算法比如说eSAIS【】，SAScan【】等等。由于SAScan算法性能较好，构造时利用的硬盘空间还不到eSAIS的一半，且速度比SAScan还高，本文后面将会详细的描述一下SAScan是如何构造广义的后缀数组的。

## 论文主要工作

本文首先深入讲述了当前主流的后缀数组构造算法的思想，并对采用了分治递归和归纳排序的SAIS算法和构建广义后缀数组的SAScan算法进行了深入的分析。

其次以SAIS算法为基础，提出了一种并行计算广义后缀数组的方法PSAIS。针对于PSAIS算法在实现过程中内存占用过高的问题，通过利用通信数据只更新其前继字符全局排名的特性，大大减少了内存空间的占用。

同时讲述了多机情况下的负载均衡问题。针对于文件过多和机器硬件不统一的情况，提出了解决方案，提高了算法的运行速度。

最后，通过实验结果进行分析，指出了该技术的特点和局限性，并提出了改进的方法。

## 论文组织结构

本论文共分为五章，从理论到实现，再到优化，全面讲述了PSAIS的构造和优化方法。

第一章讲述了该论文的研究背景和研究意义，指出了并行计算广义后缀数组的必要性。同时描述了主流后缀数组构造算法的思想和当前广义后缀数组的构造方法。

# 后缀数组常用内存算法和MPI介绍

## 常用符号和术语

## 归纳排序和递归排序

### SAIS算法

## MPI介绍

## 本章小结

# 第三章 PSAIS算法介绍和实现

## PSAIS算法自定义变量

## PSAIS算法的归纳排序和分治递归

## 本章小结

# 第四章 实验结果和算法效率分析

## 实验环境

## 实验内容和效率分析

## 实验总结

# 第五章 总结与展望

## 实验总结

## 研究展望