原创 【Suffix Array】后缀数组详解

2019-09-30 19:10:27 我是一只计算鸡 阅读数 18 文章标签: 后缀数组 更多

后缀数组，一种处理字符串的有力工具。

后缀：假设字符串的长度为n，那么后缀 i 表示 从 i 到 n 这一段字符串。ababa：后缀 3 为 aba。

后缀数组就是对字符串的所有后缀排序。

很显然，我们可以通过快速排序在$O(nlogn)$的时间复杂度内对 n 个后缀进行排序，但是字符串之间的比较不是$O(1)$的，而是$O(n)$的，所以整

Manber和Myers发明的倍增算法，它的时间复杂度为$O(nlogn)$。

下面详细介绍一下这个倍增算法。

我们有一个aabaaaab的字符串，现在要对这个字符串的后缀排序。我们先看一下过程图。

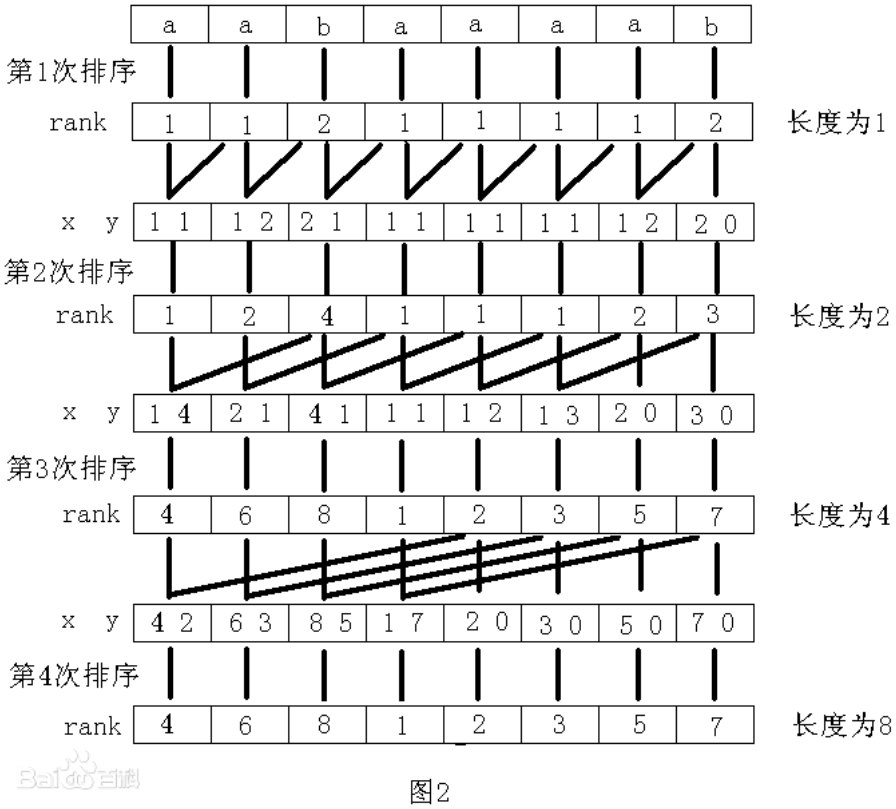

图2

1：对每个后缀的第一个字符排序，根据字典序排序，所以得到了第一次的排序。

2：对每个后缀的前两个字符排序，得到了第二次的排序。

3：对每个后缀的前四个字符排序，既然是倍增就要用倍增的样子。但是我们这次排序的仍然是2个字符。后缀k的前4个字符看成由后缀k的前2个字符和
前2个字符组成。如果要把后缀k和后缀k'比较，应该先比较后缀k的前两个字符和后缀k'的前两个字符，如果相同，再比较后缀k+2的前两个字符和后缀
个字符。所以一直排序的是二元组，这就是为什么要使用基数排序。

4：如果所有的排名都不同的话，就不需要再倍增了。

虽然这个倍增算法很好理解，代码实现起来就很是头大，从头到尾全是数组，建议按照代码手动模拟一遍ababa的后缀排序，就能很好的理解此算法，么知道，因为笔者开始也是一下就看懂了算法思想，但是就是看不懂代码。

```
#include<bits/stdc++.h>
using namespace std;
const int maxn = 1000000 + 7;
```

```
char s[maxn]; // 字符串 s | int sa[maxn], c[maxn], x[maxn], y[maxn], n, m;
// sa[i] 排名为 i 的后缀下标
// c[] 基数排序的桶  x[] 基数排序第一关键字 y[] 基数排序第二关键字
// n 字符串长度 m 关键字种类
int suffix_array()
{
        for(int i = 1; i <= m; i++) c[i] = 0;  // 初始化桶
        for(int i = 1; i <= n; i++) ++c[x[i] = s[i]];  // 统计每个桶中的元素
        for(int i = 2; i <= m; i++) c[i] += c[i-1]; // 前缀和
        for(int i = n; i >= 1; i--) sa[c[x[i]]--] = i;
        for(int k = 1; k <= n; k <<= 1) {  // 倍增
                int num = 0;
                // 后面的后缀没有第二关键字了,所以字典序小,放在前面
                for(int i = n - k + 1; i <= n; i++) y[++num] = i;
                // 通过上一次排序的sa[] 得到第二关键排序
                // 此时已经有k个元素没有第二关键字了,
                // 重新排序时，通过sa[]数组和k 就可以对第二关键字排序
                for(int i = 1; i <= n; i++) if(sa[i] > k) y[++num] = sa[i] - k;
                // 对第一关键字排序
                for(int i = 1; i <= m; i++) c[i] = 0;
                for(int i = 1; i <= n; i++) ++c[x[i]];
                for(int i = 2; i <= m; i++) c[i] += c[i-1];
                for(int i = n; i >= 1; i--) sa[c[x[y[i]]]--] = y[i], y[i] = 0;
                // 交换x , y 便于生成下一次基数排序的第一关键字
                // 此时 y 已经没用了
                swap(x, y);
                x[sa[1]] = 1, num = 1; // 至少有一种关键字
                for(int i = 2; i <= n; i++) { // 按字典序统计不同的排名个数
                        x[sa[i]] = (y[sa[i-1]] == y[sa[i]] && y[sa[i-1] + k] == y[sa[i] + k]) ? num : ++num;
                }
                if(num >= n) break; // 如果 n 个后缀排名都不一样 排序完成
                m = num; // 下一次排序 最多有 m 个不同的关键字
        }
        return 0;
}
int main()
{
        while(scanf("%s", s + 1) == 1) {
                n = strlen(s + 1);
                m = 122;
                suffix_array();
                for(int i = 1; i <= n; i++) printf("%d ", sa[i]);
                printf("\n");
        }
        return 0;
}
```


文章最后发布于: 201