yzmduncan

数据结构之——后缀数组

数据结构 算法 C C++ C#

后缀数组，很精妙的数据结构。

后缀：从母串的某一位置开始到结尾，suffix(i) = Ai,Ai+1...An。

后缀数组：后缀数组SA是个一维数组，它保存1...n的某个排列SA[1],SA[2]...SA[n]，并且保证suffix(SA[i])<suffix(SA[i+1])，也就是将S的n个后缀从小到大排好序后的开头位置保存到SA中。

名次数组：名次数组Rank[i]保存的是以i开头的后缀的排名，与SA互为逆。简单的说，后缀数组是“排在第几的是谁”，名次数组是“你排第几”。

为了方便比较，通常在串的末尾添加一个字符，它是从未出现并且最小的字符。

求解后缀数组的算法主要有两种：倍增算法和DC3算法。在这里使用的是许智磊的倍增算法，复杂度为nlogn。

关于详细求解后缀数组的算法，详见许智磊2004国家集训队论文。

后缀数组的应用：

最长公共前缀：先定义height数组，height[i] = suffix(SA[i-1])和suffix(SA[i])的最长公共前缀，也就是排名相邻的两个后缀的最长公共前缀。

例如，字符串为“aabaaaab”，求后缀“abaaaab”和后缀“aaab”的最长公共前缀，如图 4 所示：

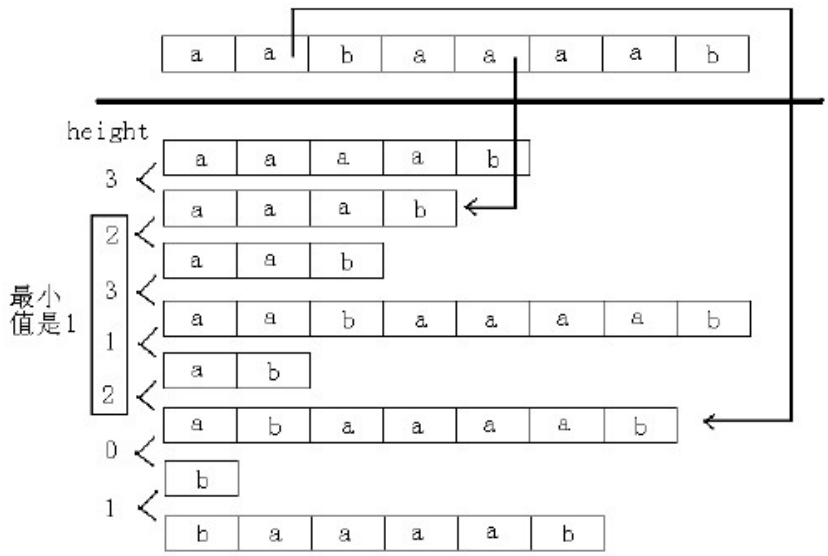

图4

例1：最长公共前缀

函数
- 次小生成树

给定一个串，求任意两个后缀的最长公共前缀。

解：先根据rank确定这两个后缀的排名i和j(i<j)，在height数组i+1和j之间寻找最小值。(可以用rmq优化)

例2：最长重复子串(不重叠)(poj1743)

解：二分长度，根据长度len分组，若某组里SA的最大值与最小值的差>=len，则说明存在长度为len的不重叠的重复子串。

例3：最长重复子串(可重叠)

解：height数组里的最大值。这个问题等价于求两个后缀之间的最长公共前缀。

例4：至少重复k次的最长子串(可重叠)(poj3261)

解：二分长度，根据长度len分组，若某组里的个数>=k，则说明存在长度为len的至少重复k次子串。

例5：最长回文子串(ural1297)

给定一个串，对于它的某个子串，正过来写和反过来写一样，称为回文子串。

解：枚举每一位，计算以这个位为中心的的最长回文子串(注意串长要分奇数和偶数考虑)。将整个字符串反转写在原字符串后面，中间用$分隔。这样把问题转化为求某两个后缀的最长公共前缀。

例6：最长公共子串(poj2774)

给定两个字符串s1和s2，求出s1和s2的最长公共子串。

解：将s2连接到s1后，中间用$分隔开。这样就转化为求两个后缀的最长公共前缀，注意不是height里的最大值，是要满足sa[i-1]和sa[i]不能同时属于s1或者s2。

例7：长度不小于k的公共子串的个数(poj3415)

给定两个字符串s1和s2，求出s1和s2的长度不小于k的公共子串的个数(可以相同)。

解：将两个字符串连接，中间用$分隔开。扫描一遍，每遇到一个s2的后缀就统计与前面的s1的后缀能产生多少个长度不小于k的公共子串，这里s1的后缀需要用单调栈来维护。然后对s1也这样做一次。

例8：至少出现在k个串中的最长子串(poj3294)

给定n个字符串，求至少出现在n个串中k个的最长子串。

将n个字符串连接起来，中间用$分隔开。二分长度，根据长度len分组，判断每组的后缀是否出现在不小于k个原串中。

求解后缀数组的模板：

Cpp代码

```cpp
const int N = 20005;//串A的最大长度
const int MAX = 1000100;//串A的最大值
//int n,m,k;
int SA[N], rank[N], height[N], key[N];
int A[N], C[MAX], t1[N+1], t2[N+1];

//倍增法求sa[]-----待排序的字符串放在r 数组中，r[]为整型数组，从r[0]到r[n-1]，长度为n，且最大值小于m

//约定除r[n-1]外所有的r[i]都大于0，r[n-1]=0
//结果放在sa 数组中，从sa[0]到sa[n-1]
//先对所有后缀的第一个字符进行排序(采用挖坑式的基数排序,即统计每个字符的个数,以便在扫描时总能将字符放入合适的位置),放入sa中
void da(int n, int m)
{
    int i, j, l, s,*t;
```

```
    int *X = t1, *Y = t2;
    memset(C, 0, sizeof(C));
    for (i=0;i<n;i++) C[X[i] = A[i]]++;
    for (i=1;i<m;i++) C[i] += C[i-1];
    for (i=n-1;i>=0;i--) SA[--C[X[i]]] = i;
    for (l=1; l<n; l<<=1)
    {
        for (i=n-l,j=0;i<n;i++) Y[j++] = i;
        for (i=0;i<n;i++) if (SA[i] >= l) Y[j++] = SA[i] - l;
        for (i=0;i<n;i++) key[i] = X[Y[i]];
        memset(C, 0, sizeof(C));
        for (i=0;i<n;i++) C[key[i]]++;
        for (i=1;i<m;i++) C[i] += C[i-1];
        for (i=n-1;i>=0;i--) SA[--C[key[i]]] = Y[i];
        t = X;
        X = Y;
        Y = t;
        X[SA[0]] = j = 0;
        for (i=1;i<n;i++)
        {
            if (Y[SA[i]] != Y[SA[i-1]] || Y[SA[i]+l] != Y[SA[i-1]+l])
                j++;
            X[SA[i]] = j;
        }
        m = j + 1;
        if (m==n) break;
    }

    for (i=0;i<n;i++)
        rank[SA[i]] = i;
    return;
}

//height[i]:suffix(sa[i-1])与suffix(sa[i])的最长公共前缀，即排名相邻的两个后缀的最长公共前缀
void calheight(int n)
{
    int i,j,k=0;
    for(i=0; i<n; i++)
    {
        if (k > 0)
            --k;
        if(rank[i] == 0)
            height[0] = 0;
        else
        {
            j = SA[rank[i] - 1];
            while(A[i+k] == A[j+k])
                k++;
            height[rank[i]] = k;
        }
    }
}
//串A[0]...A[n-1]
da(n,1000001);  //m的最大值不超过1,000,000
calheight(n);
```

查看图片附件

分享到：

差分约束系统——建模与求解 | 数据结构之——RMQ与LCA

2011-03-30 08:59:29 | 浏览 748 | 评论(0) | 分类:编程语言 | 相关推荐 MORE

评论

发表评论

您还没有登录,请您登录后再发表评论