[解题报告]I Long Long Message

[Source]

http://202.114.18.202:8080/judge/contest/view.action?cid=6236#problem/I

[Description]

给定两个字符串 A 和 B ，求最长公共子串。

[Solution]

这是一道后缀数组的模板题。下面是罗穗骞的国家集训队论文中的内容。（论文讲了后缀数组的定义，求法，应用等，建议去看原论文）

字符串的任何一个子串都是这个字符串的某个后缀的前缀。求 A 和 B 的最长公共子串等价于求 A 的后缀和 B 的后缀的最长公共前缀的最大值。如果枚举 A 和 B 的所有的后缀，那么这样做显然效率低下。由于要计算 A 的后缀和 B 的后缀的最长公共前缀，所以先将第二个字符串写在第一个字符串后面，中间用一个没有出现过的字符隔开，再求这个新的字符串的后缀数组。观察一下，看看能不能从这个新的字符串的后缀数组中找到一些规律。以 A=“ aaaba ”，B=“ abaa ”为例，如图 8 所示。

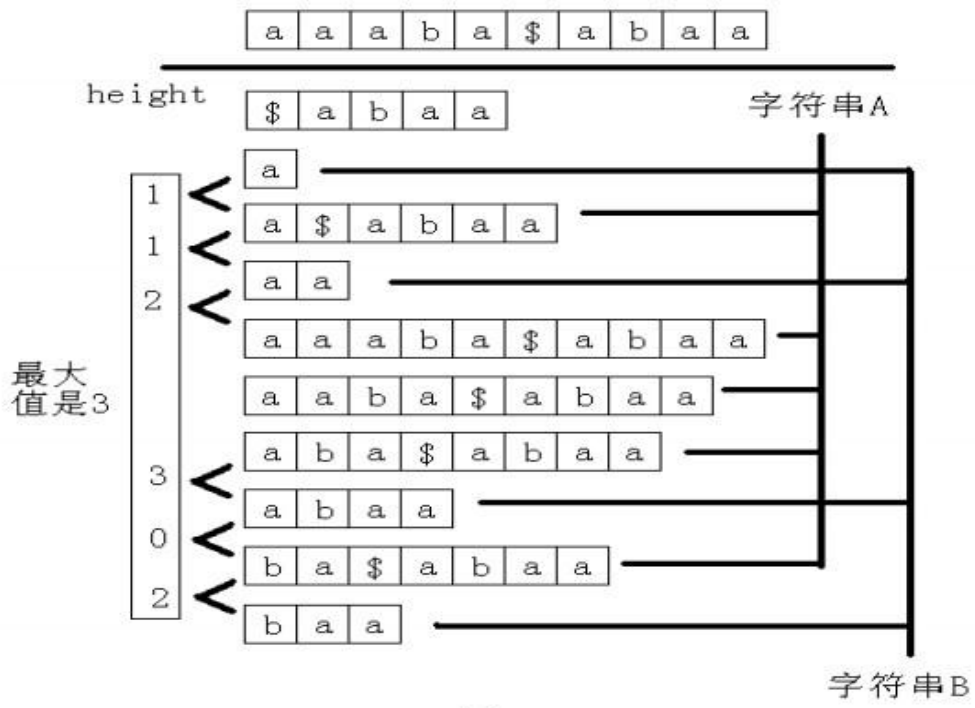

图8

那么是不是所有的 height 值中的最大值就是答案呢？不一定!有可能这两个后缀是在同一个字符串中的，所以实际上只有当 suffix(sa[i-1])和 suffix(sa[i]) 不是同一个字符串中的两个后缀时，height[i]才是满足条件的。而这其中的最大值就是答案。记字符串 A 和字符串 B 的长度分别为|A|和|B|。求新的字符串的后缀数组和 height 数组的时间是 O(|A|+|B|) ，然后求排名相邻但原来不在同一个字符串中的两个后缀的 height 值的最大值，时间也是 O(|A|+|B|)，所以整个做法的时间复杂度为 O(|A|+|B|) 。时间复杂度已经取到下限，由此看出，这是一个非常优秀的算法。

[Code]

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<cmath>
#include<algorithm>

using namespace std;
const int N=200005;
char s[N];
int n, sa[N], height[N], rank[N], tmp[N], top[N];

void makesa()
{
	int i, j, len, na;
	na = (n < 256 ? 256 : n);
	memset(top, 0, na * sizeof(int));
	for (i = 0; i < n ; i++) top[ rank[i] = s[i] & 0xff ]++;
	for (i = 1; i < na; i++) top[i] += top[i - 1];
	for (i = 0; i < n ; i++) sa[ --top[ rank[i] ] ] = i;
	for (len = 1; len < n; len <<= 1)
	 {
		for (i = 0; i < n; i++)
		{
			j = sa[i] - len; if (j < 0) j += n;
			tmp[ top[ rank[j] ]++ ] = j;
		}
		sa[ tmp[ top[0] = 0 ] ] = j = 0;
		for (i = 1; i < n; i++)
		{
			if (rank[ tmp[i] ] != rank[ tmp[i-1] ] ||
				rank[ tmp[i]+len ]!=rank[ tmp[i-1]+len ])
			top[++j] = i;
```

```
                sa[ tmp[i] ] = j;
            }
            memcpy(rank, sa , n * sizeof(int));
            memcpy(sa , tmp, n * sizeof(int));
            if (j >= n - 1) break;
        }
}

void lcp()
{
        int i, j, k;
        for (j = rank[height[i=k=0]=0]; i < n - 1; i++, k++)
        while (k >= 0 && s[i] != s[ sa[j-1] + k ])
        height[j] = (k--), j = rank[ sa[j] + 1 ];
}

int main()
{
        int l1,i;
        scanf("%s",s);
        l1=strlen(s);
        s[l1]='0';
        scanf("%s",s+l1+1);
        n=strlen(s)+1;
        makesa();
        lcp();
        int ans=0;
        for (i=2; i<n; i++)
          if (sa[i]<l1 && sa[i-1]>l1 || sa[i]>l1 && sa[i-1]<l1)
          {
                ans=max(ans,height[i]);
          }
        printf("%d\n",ans);
}
```