[**后缀数组(O(n)三分实现)**](http://www.cnblogs.com/zhuangli/archive/2008/08/18/1270608.html)

比起WC关于后缀数组的倍增法,要更有效率,由于模版使用的是指针,所以在new和delete处效率会有所消耗

来自NIT的后缀数组模版 注意在字符串间加入特殊符号进行区分 避免LCP越界

SAMLPE测试为HDU 1403~~

#include<iostream>  
#include<cmath>  
#include<cstring>  
using namespace std;  
const int maxn=210000;  
  
char s[maxn];  
int len,k;  
int sa[maxn],rank[maxn],lcp[maxn];  
int num[maxn];  
  
inline bool leq(int a1, int a2, int b1, int b2)  
{  
    return (a1 < b1 || a1 == b1 && a2 <= b2);  
}  
  
inline bool leq(int a1, int a2, int a3, int b1, int b2, int b3)  
{  
    return(a1 < b1 || a1 == b1 && leq(a2, a3, b2, b3));  
}  
  
static void radixPass(int\* a, int\* b, int\* r, int n, int K)  
{  
    int\* c = new int[K + 1];  
    int i,sum;  
    for(i = 0; i <= K; i++)   
        c[i] = 0;  
    for(i = 0; i < n; i++)   
        c[r[a[i]]]++;  
    for(i = 0, sum = 0; i <= K; i++)  
    {  
        int t = c[i];   
        c[i] = sum;   
        sum += t;  
    }  
    for(i = 0; i < n; i++)   
        b[c[r[a[i]]]++] = a[i];  
    delete [] c;  
}  
  
void suffixArray(int\* T, int\* SA, int n, int K)  
{  
    int n0 = (n + 2) / 3, n1 = (n + 1) / 3, n2 = n / 3, n02 = n0 + n2;  
    int\* R = new int[n02 + 3]; R[n02] = R[n02+1] = R[n02 + 2] = 0;  
    int\* SA12 = new int[n02 + 3]; SA12[n02] = SA12[n02 + 1] = SA12[n02 + 2] = 0;  
    int\* R0 = new int[n0];  
    int\* SA0 = new int[n0];  
    int i,j;  
    for(i = 0, j = 0; i < n + (n0 - n1); i++) if(i % 3 != 0) R[j++] = i;  
    radixPass(R , SA12, T + 2, n02, K);  
    radixPass(SA12, R , T + 1, n02, K);  
    radixPass(R , SA12, T , n02, K);  
    int name = 0, c0 = -1, c1 = -1, c2 = -1;  
    for( i = 0; i < n02; i++)  
    {  
        if(T[SA12[i]] != c0 || T[SA12[i] + 1] != c1 || T[SA12[i] + 2] != c2)  
        {  
            name++; c0 = T[SA12[i]]; c1 = T[SA12[i] + 1]; c2 = T[SA12[i] + 2];  
        }  
        if(SA12[i] % 3 == 1) { R[SA12[i] / 3] = name; }  
        else{ R[SA12[i] / 3 + n0] = name; }  
    }  
    if(name < n02)  
    {  
        suffixArray(R, SA12, n02, name);  
        for(int i = 0; i < n02; i++) R[SA12[i]] = i + 1;  
    }  
    else  
        for(int i = 0; i < n02; i++) SA12[R[i] - 1] = i;  
    for(i = 0, j = 0; i < n02; i++) if(SA12[i] < n0) R0[j++] = 3 \* SA12[i];  
    radixPass(R0, SA0, T, n0, K);  
    for(int p = 0, t = n0 - n1, k = 0; k < n; k++)  
    {  
        #define GetI() (SA12[t] < n0 ? SA12[t] \* 3 + 1 : (SA12[t] - n0) \* 3 + 2)  
        int i = GetI();  
        int j = SA0[p];  
        if(SA12[t] < n0 ?  
        leq(T[i], R[SA12[t] + n0], T[j], R[j / 3]) : leq(T[i],T[i + 1],R[SA12[t] - n0 + 1], T[j],T[j + 1],R[j / 3 + n0]))  
        {  
            SA[k] = i; t++;  
            if(t == n02)  
                for(k++; p < n0; p++, k++) SA[k] = SA0[p];  
        }  
        else{  
            SA[k] = j;  
            if(++p == n0)for(k++; t < n02; t++, k++) SA[k] = GetI();  
        }  
    }  
    delete [] R; delete [] SA12; delete [] SA0; delete [] R0;  
}  
  
void LCP(char \*str)   
{   
    int i,j,k;   
    for(k=i=0;i<len;i++)   
    {   
        if( rank[i] == len - 1 ) lcp[ rank[i] ] = k = 0;   
        else   
        {   
            if( k > 0 )    k--;   
            j = sa[ rank[i] + 1];   
            for(;str[i+k] == str[j+k];k++);   
            lcp[rank[i]] = k;   
        }   
    }   
}  
  
int main(){  
  
    int i, n;  
  
    scanf("%s",s);  
    n=strlen(s);  
    s[n]='A'-1;  
    scanf("%s",s+n+1);  
    len=strlen(s);  
    for(i=0;i<len;i++)  
        num[i]=s[i]-'A'+1;  
    suffixArray(num, sa, len, 60);  
    for(i=0;i<len;i++)  
        rank[sa[i]]=i;  
  
    LCP(s);  
  
    int ans=0;  
    for(i=1;i<len;i++)  
        if(sa[i]<n && sa[i-1]>n  ||  sa[i]>n && sa[i-1]<n)  
            if(lcp[i-1]>ans) ans=lcp[i-1];  
  
    printf("%d\n",ans);  
  
    return 0;  
}