ACM暑期集训字符串-KMP算法+序列自动机

序列自动机

$next[i][j]$表示在原串s第i位后面的第一个j出现的位置，设串长为n,字符集大小为a，预处理时间复杂度为$O(n*a)$，代码如下。

例题 2019南昌邀请赛网络赛 M题-Subsequence

Give a string *S* and *N* string T_i , determine whether T_i is a subsequence of *S*.

If ti is subsequence of *S*, print `YES`,else print `NO`.

If there is an array $\{K_1, K_2, K_3, \cdots, K_m\}$so that $1 \leq K_1 < K_2 < K_3 < \cdots < K_m \leq N$and $S_{k_i} = T_i$, $(1 \leq i \leq m)$(1≤*i*≤*m*), then T_i is a subsequence of S*S*.

Input

The first line is one string *S*,length(*S*) ≤ 100000

The second line is one positive integer N,$N \leq 100000$

Then next *n* lines,every line is a string T_i*T***i*, length(T_i) ≤ 1000

Output

Print N* lines. If the i*-th T_i is subsequence of *S*, print `YES`, else print `NO`.

样例输入

```
abcdefg
3
abc
adg
cba
```

样例输出

```
YES
YES
NO
```

AC代码

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
```

```
char s[N],p[N];
int nex[N][30],now[N];
//nex[i][j]:=表示第i个字符后面第一次出现字符j（a-z用0-25表示）的位置。
//我们从后往前求，now[j]:=字符j从后往前数最晚出现的位置（now数组初始化为-1）
//序列自动机就是这个nex数组的使用，很简单，一下子就可以学会了
void init()
{
    memset(now,-1,sizeof(now));
    int len=strlen(s);
    for(int i=len-1; i>=0; i--)
    {
        for(int j=0; j<26; j++)
        {
            nex[i][j]=now[j];
        }
        now[s[i]-'a']=i;
    }
}

int main()
{
    int n,len,loc,flag;
    scanf("%s",s);
    init();
    scanf("%d",&n);
    for(int i=1; i<=n; i++)
    {
        scanf("%s",p);
        len=strlen(p);
        loc=now[p[0]-'a'];
        if(loc==-1)
            printf("NO\n");
        else
        {
            flag=0;
            for(int i=1; i<len; i++)
            {
                loc=nex[loc][p[i]-'a'];
                if(loc==-1)
                {
                    flag=1;
                    break;
                }
            }
            if(!flag)
                printf("YES\n");
            else
                printf("NO\n");
        }
    }
}
```

KMP算法

暴力匹配

假设现在我们面临这样一个问题：有一个文本串S，和一个模式串P，现在要查找P在S中的位置，怎么查找呢?

如果用暴力匹配的思路，并假设现在文本串S匹配到 i 位置，模式串P匹配到 j 位置，则有：

如果当前字符匹配成功（即S[i] == P[j]），则i++，j++，继续匹配下一个字符；如果失配（即S[i]! = P[j]），令i = i - (j - 1)，j = 0。相当于每次匹配失败时，i 回溯，j 被置为0。

理清楚了暴力匹配算法的流程及内在的逻辑，咱们可以写出暴力匹配的代码，如下：

```
//复杂度为 sLen*pLen
int ViolentMatch(char* s, char* p)
{
    int sLen = strlen(s);
    int pLen = strlen(p);

    int i = 0;
    int j = 0;
    while (i < sLen && j < pLen)
    {
        if (s[i] == p[j])
        {
            //①如果当前字符匹配成功（即S[i] == P[j]），则i++，j++
            i++;
            j++;
        }
        else
        {
            //②如果失配（即S[i]! = P[j]），令i = i - (j - 1)，j = 0
            i = i - j + 1;
            j = 0;
        }
    }
    //匹配成功，返回模式串p在文本串s中的位置，否则返回-1
    if (j == pLen)
        return i - j;
    else
        return -1;
}
```

KMP算法

1.定义

Knuth-Morris-Pratt 字符串查找算法，简称为 “KMP算法”，常用于在一个文本串S内查找一个模式串P 的出现位置，这个算法由Donald Knuth、Vaughan Pratt、James H. Morris三人于1977年联合发表，故取这3人的姓氏命名此算法。

博客，视频链接

博客<https://blog.csdn.net/gao506440410/article/details/81812163>

B站视频1<https://www.bilibili.com/video/av11866460?from=search&seid=5943607834453890070>

B站视频2<https://www.bilibili.com/video/av16828557?from=search&seid=5943607834453890070>

//博客图文好多，~~我懒得做~~。。。给大家几个链接看看。。。

//课前可以稍微看看博客稍微了解，我上课会结合例子给大家画图讲解

//这个视频讲的是真的好，清晰明白，我都是模仿他讲的，不过还是给大家讲一遍

//可以直接看视频就能懂。。。今天还是很简单的

模板

```
#include<bits/stdc++.h>
using namespace std;

const int N=1e4+7;

char Find[10100],a[1000100];
int len=strlen(Find);
int llen=strlen(a);
int NEXT[N];

void getNEXT() {
    int i = 0, j = NEXT[0] =-1;
    while (i<len){
        if (j == -1 || Find[i] == Find[j]) {
            i++;j++;
            NEXT[i] = j;
        }
        else {
            j = NEXT[j];
        }
    }
}


int FirstKmp() {//在a中找到第一个Find的位置
    int i = 0, j = 0;
    while (i < llen) {
        if (j == -1 || a[i] == Find[j]) {
            i++, j++;
            if (j == len) {
```

```
                return i - j;
            }
        }
        else j = NEXT[j];
    }
    return -1;
}


int main()
{

        scanf("%s",a)
        scanf("%s",Find);
        len=strlen(Find);
        llen=strlen(a);
        memset(NEXT,0,sizeof(NEXT));
        getNEXT();
        cout<<FirstKmp()<<endl;
    }

return 0;
}
```