#include<stdio.h>  
#include<string.h>  
#include<algorithm>  
using namespace std;  
struct node  
{  
    bool End;  
    node\* Next[26];  
};  
char s[1000005];  
node\* root;  
void Clear(node\* p)  
{  
    p->End=false;  
    memset(p->Next,NULL,sizeof(p->Next));  
}  
void Insert()  
{  
    int i;  
    node \*p=root;  
    for(i=0; s[i]; i++)  
    {  
        if(p->Next[s[i]-'a']==NULL)  
        {  
            p->Next[s[i]-'a']=(node\*)malloc(sizeof(node));  
            Clear(p->Next[s[i]-'a']);  
        }  
        p=p->Next[s[i]-'a'];  
    }  
    p->End=true;  
}  
int main()  
{  
    int n,m,i;  
    root=(node\*)malloc(sizeof(node));  
    Clear(root);  
    scanf("%d%d",&n,&m);  
    for(i=0; i<n; i++)  
    {  
        scanf("%s",s);  
        Insert();  
    }  
    return 0;  
}  
  
孙洪嘉 2019/3/12 19:26:02  
  
bool query()  
{  
    int i;  
    node \*p=root;  
    for(i=0; s[i]; i++)  
    {  
        if(p->Next[s[i]-'a']==NULL)  
            return false;  
        p=p->Next[s[i]-'a'];  
    }  
    if(p->End)  
        return true;  
    return false;  
}  
  
孙洪嘉 2019/3/12 19:29:47  
  
void dfs(node\* p,int deep)  
{  
    if(p->End)  
    {  
        s[deep]=0;  
        printf("%s\n",s);  
    }  
    int i;  
    for(i=0; i<26; i++)  
    {  
        if(p->Next[i])  
        {  
            s[deep]=i+'a';  
            dfs(p->Next[i],deep+1);  
        }  
    }  
}  
  
孙洪嘉 2019/3/12 19:37:32  
  
字典：按照字典序存储单词  
查字典方法：遍历 O(n\*len)  
查字典方法：二分 O(log(n)\*len)  
向字典中添加单词：维持字典序顺序->移动后面所有单词  
  
字典树  
统计,排序,保存大量的字符串  
利用字符串的公共前缀来减少查询时间，最大限度地减少无谓的字符串比较  
指针 数组  
数值域 指针域  
数值域存储有无以该结点结尾的单词 多少单词通过该节点(前缀)  
指针域存储所有指向儿子结点的指针  
根节点数值域为空，不包含字符  
根节点到某个结点，路径上经过的字符连接起来，就是对应的字符串  
一个字符串，第i+1个字符是第i个字符的子结点  
  
#include<stdio.h>  
#include<string.h>  
#include<algorithm>  
using namespace std;  
struct node  
{  
    bool End;  
    node\* Next[26];  
};  
char s[1000005];  
node\* root;  
void Clear(node\* p)  
{  
    p->End=false;  
    memset(p->Next,NULL,sizeof(p->Next));  
}  
void Insert()  
{  
    int i;  
    node \*p=root;  
    for(i=0; s[i]; i++)  
    {  
        if(p->Next[s[i]-'a']==NULL)  
        {  
            p->Next[s[i]-'a']=(node\*)malloc(sizeof(node));  
            Clear(p->Next[s[i]-'a']);  
        }  
        p=p->Next[s[i]-'a'];  
    }  
    p->End=true;  
}  
bool query()  
{  
    int i;  
    node \*p=root;  
    for(i=0; s[i]; i++)  
    {  
        if(p->Next[s[i]-'a']==NULL)  
            return false;  
        p=p->Next[s[i]-'a'];  
    }  
    if(p->End)  
        return true;  
    return false;  
}  
void dfs(node\* p,int deep)  
{  
    if(p->End)  
    {  
        s[deep]=0;  
        printf("%s\n",s);  
    }  
    int i;  
    for(i=0; i<26; i++)  
    {  
        if(p->Next[i])  
        {  
            s[deep]=i+'a';  
            dfs(p->Next[i],deep+1);  
        }  
    }  
}  
int main()  
{  
    int n,m,i;  
    root=(node\*)malloc(sizeof(node));  
    Clear(root);  
    scanf("%d%d",&n,&m);  
    for(i=0; i<n; i++)  
    {  
        scanf("%s",s);  
        Insert();  
    }  
    dfs(root,0);  
    for(i=0; i<m; i++)  
    {  
        scanf("%s",s);  
        if(query())  
            printf("Yes\n");  
        else  
            printf("No\n");  
    }  
    return 0;  
}  
  
  
malloc 申请一段内存  
头文件 stdlib.h malloc.h  
void\* malloc(size\_t);  
  
int \*p;  
int\* p;  
int n;  
p=&n;  
\*p=n;