最强大脑

时间限制： C/C++语言 1000MS；其他语言 3000MS
内存限制： C/C++语言 65536KB；其他语言 589824KB
题目描述：
人脑对于长度特别长的字符串的处理速度是有限的，但是最强大脑挑战的就是人脑的极限，现在有这样一项挑战，给出一个很长的字符串S，和一个较短的字符串T，请你求出对于每一个前缀[1,r]内有多少个T字符串。

输入

第一行一个字符串S。
第二行一个字符串T。两个字符串保证均只含小写字母。（1≤|S|≤500000, 1≤|T|≤100）

输出

输出仅包含|S|个正整数，分别表示[1,r]内有多少个T字符串。(1<=r<=|S|)

样例输入

```
ababac
ab
```

样例输出

```
0 1 1 2 2 2
```

规则

请尽量在全场考试结束10分钟前调试程序，否则由于密集排队提交，可能查询不到编译结果
点击“调试”亦可保存代码
编程题可以使用本地编译器，此页面不记录跳出次数

```
#include<stdio.h>
#include<algorithm>
#include<string>
#include<iostream>
#include<string.h>
using namespace std;

typedef long long ll;
typedef unsigned long long ull;
const int MAX = 5e6 + 5;

const int base = 95;
char a[MAX], b[MAX];
unsigned long long getval(char ch)
{
    if (ch >= 'a' && ch <= 'z')return ch - 'a' + 1;
    return ch - 'A' + 27;
}

ull sum[MAX];
int main(){

    while (scanf("%s %s", &a, &b) != EOF) {
```

```
        int n = strlen(b), m = strlen(a);
        sum[0] = getval(a[0]);
        for (int i = 1; i < m; i++)sum[i] = sum[i - 1] * base + getval(a[i]);
        ull ha = 0, hb = 0, f = 1;
        for (int i = 0; i < n; i++)f = f * base;
        for (int i = 0; i < n; i++)
            ha = (ha * base + getval(b[i]));

        int ans = 0;
        if(m<n){
            for(int i=0;i<m;i++){
                printf("0");
                if(i!=m-1)printf(" ");
            }
            printf("\n");
        }
        else{
            for (int i = 0; i < n - 1; i++) {
                printf("%d", ans);
                if (i != m - 1)printf(" ");
            }
            if (ha == sum[n - 1])ans++;
            printf("%d", ans);
            if (n-1 != m - 1)printf(" ");
            for (int i = n; i < m; i++)
            {
                if (ha == sum[i] - sum[i - n] * f)ans++;
                printf("%d", ans);
                if (i != m - 1)printf(" ");
            }
            printf("\n");
        }

    }
    //printf("%d\n", ans);
    return 0;
}

/*
ababac
ab
0 1 1 2 2 2
 */
```

或和异或

时间限制： C/C++语言 1000MS；其他语言 3000MS
内存限制： C/C++语言 65536KB；其他语言 589824KB
题目描述：
首先给出一个长度为k=2^n（其中n为正整数）的序列A={a1，a2...a_{k-1},ak}，我们定义一个值v，这个值是由如下计算方法得到的，首先将A序列的第 i 位和第 i+1 位进行 OR 操作得到新数列A ‘，然后再对A’ 序列操作，将A’ 序列的第 i 位和第 i+1 位进行 XOR 操作得到A ‘’ ，对A ‘’ 按照第一次操作方式进行OR操作，因为序列长度为2^n,所以显然进行n次操作之后就只剩下一个数字了，此时这个数字就是v。
例如A={1，2，3，4}，第一次操作之后为{1|2=3，3|4=7}，第二次操作后，{3^7=4},所以v=4。
但是显然事情并没有那么简单，给出A序列后，还有m个操作，每个操作表示为“a b”，表示将A[a]改为b，之后再对A序列求v值。（注意每一次对序列的修改的永久的，即第i次修改是建立在前i-1次修改之上的）

输入

输入第一行包含两个正整数n，m，分别表示A序列的长度为2^n,操作数量为m。（1<=n<=17,1<=m<=10^5）
输入第二行包含n个正整数，中间用空格隔开，表示A序列。（0<=ai<=2^30）
接下来有m行，每行包含两个正整数a，b，表示一次操作，即把A[a]变成b。

输出

输出包含m行，第i行表示进行了第i次操作之后，A序列的v值。

样例输入

```
2 4
1 2 3 4
1 4
3 4
1 3
1 3
```

样例输出

```
1
2
7
7
```

```
#include<stdio.h>
#include<algorithm>
#include<string>
#include<iostream>
#include<string.h>
#include<cmath>
using namespace std;

int n2;
int a[800000];
int v[200000];
int init(int s, int l,int r){
    // printf("%d %d %d\n", s,l, r);
    if(l==r){
        a[s]=v[r];
        return 0;
    }
    int mid = (l+r)/2;

    int h= init(s*2,l,mid);
    init(s*2+1, mid+1,r);
    if(h%2==0){
        a[s] = a[s*2]|a[s*2+1];
    }
    else{
```

```
        a[s] = a[s*2]^a[s*2+1];
    }
    return h+1;
}

int modify(int s, int l,int r,int t1,int t2){
    if(l==r){
        a[s]=t2;
        return 0;
    }
    int mid = (l+r)/2;

    int h;
    if(t1<=mid){
        h = modify(s*2,l,mid, t1,t2);
    }
    else{
        h = modify(s*2+1,mid+1,r, t1,t2);
    }
    if(h%2==0){
        a[s] = a[s*2]|a[s*2+1];
    }
    else{
        a[s] = a[s*2]^a[s*2+1];
    }
    return h+1;
}

int main(){
    int n,m;
    while (scanf("%d %d", &n, &m) != EOF) {
        n2=pow(2,n);
        for(int i=1;i<=n2;i++){
            scanf("%d",&v[i]);
        }
        init(1,1,n2);

        for(int i=0;i<m;i++){
            int t1,t2;
            scanf("%d %d",&t1, &t2);
            modify(1,1,n2,t1,t2);
            printf("%d\n",a[1]);
            if(i==m-1)printf("\n");
            else printf(" ");
        }

    }
    //printf("%d\n", ans);
    return 0;
}
/*
2 4
1 2 3 4
1 4
3 4
1 3
1 3
```

```
1
2
7
7
*/
```