Problem G　ZOJ 3190

一.题目大意

给出你 n 个资源串和 m 个病毒串，要求你计算出*“包含所有资源串但是不含任何病毒串”*的最短串的长度。

二.所用算法

将资源串和病毒串插入到一个 AC 自动机中，在结束节点标明该串的种类（资源串是 1...n，病毒串是-1）。然后用 BFS 在 AC 自动机中找出一条不经过病毒串结束节点的最短路径。

三.Trick

由于资源串 n<10，所以我们可以用位运算来记录状态。再用一个 visit 数组判重。由 BFS 找出的第一个 EOS（End of Search)点。就为最后的答案。

四.参考代码

```
//Result: Accepted    3190    C++    9110MS    16772KB    Wizmann
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <string>
#include <bitset>
#include <vector>
#include <iostream>
#include <algorithm>
#include <queue>

using namespace std;

#define print(x) cout<<x<<endl
#define input(x) cin>>x
#define N 10 //the resources
#define M 1010 //the virus codes
#define ALPHA 2
#define SIZE (50000+M*N)
#define ROOT 0
#define INF 1<<30
#define DPSIZE 1<<11
#define STRLEN 1024
```

```
struct node
{
    int next[ALPHA];
    int end,fail;
};

struct status
{
    int pos,step,binStatus;

    status(int i_pos,int i_step,int i_b)
    {
        pos=i_pos;step=i_step;
        binStatus=i_b;
    }
};

node trie[SIZE];
char inStr[STRLEN];
int n,m;
int ind;
bitset<DPSIZE> dp[SIZE];
int EOS;

void trieInsert(char *str,int endMark)
{
    int ptr=ROOT;
    for(int i=0;str[i];i++)
    {
        int now=str[i]-'0';
        if(!trie[ptr].next[now])
        {
            trie[ptr].next[now]=ind++;
        }
        ptr=trie[ptr].next[now];
    }
    trie[ptr].end=endMark;
}



void makeAC()
{
    queue<int> q;
    for(int i=0;i<ALPHA;i++)
```

```
    {
        if(trie[ROOT].next[i])
        {
            trie[trie[ROOT].next[i]].fail=ROOT;
            q.push(trie[ROOT].next[i]);
        }
    }
    while(!q.empty())
    {
        int now=q.front();
        q.pop();
        for(int i=0;i<ALPHA;i++)
        {
            if(trie[now].next[i])
            {
                int v=trie[now].next[i];
                q.push(v);
                int fail=trie[now].fail;
                trie[v].fail=trie[fail].next[i];
            }
            else
            {
                int fail=trie[now].fail;
                trie[now].next[i]=trie[fail].next[i];
            }
        }
    }
}

int getMinDistance()
{
    queue<status> q;
    q.push(status(ROOT,0,0));
    for(int i=0;i<=ind;i++) dp[i].reset();
    while(!q.empty())
    {
        status now=q.front();
        q.pop();

        for(int i=0;i<ALPHA;i++)
        {
            if(trie[trie[now.pos].next[i]].end>=0)
            {
                status tmp=now;
```

```
                tmp.pos=trie[now.pos].next[i];
                tmp.step++;

                if(trie[tmp.pos].end>0) tmp.binStatus|=(1<<trie[tmp.pos].end);
                if(tmp.binStatus==EOS) return tmp.step;

                if(dp[tmp.pos][tmp.binStatus]) continue;
                dp[tmp.pos][tmp.binStatus]=1;
                q.push(tmp);
            }
        }
    }
    return -1;
}

int main()
{
    freopen("input.txt","r",stdin);
    while(input(n>>m))
    {
        if(m==0&&n==0) break;
        EOS=0;ind=1;
        for(int i=1;i<=n;i++) EOS|=(1<<i);
        memset(trie,0,sizeof(trie));
        //给出一些程序片段和另一些病毒片度
        //求解出包含所有程序片段且不包含病毒片段的最短的序列

        //The next n lines contain the resources
        //The next m lines contain the virus codes,
        for(int i=0;i<n;i++)
        {
            scanf("%s",inStr);
            trieInsert(inStr,i+1);
        }
        for(int i=0;i<m;i++)
        {
            scanf("%s",inStr);
            trieInsert(inStr,-1);
        }
        makeAC();
        print(getMinDistance());
    }
    return 0;
}
```