AC自动机

一、基础知识

1. 基本思想

AC自动机可用于解决多模式串匹配问题。该类问题的一般做法如下：

Step1: 读入模式串构造trie

Step2: 对trie的每个节点构造失配指针fail**trie图fail树是重点**

Step3: 匹配如果失配，跳fail边，类似kmp

2. fail树的构建

* 一个节点A的fail指针指向的节点B实际上B所代表的字符串是A代表的字符串的**最长后缀** * fail构成了树结构，会和树的相关算法数据结构结合 * 我们**利用部分已经求出 fail 指针的结点**推导出当前结点的 fail 指针。

- 考虑字典树中当前的节点uu的父节点是pp通过字符c的边指向u
- 假设深度小于u的所有节点的 fail 指针都已求得。那么p的 fail指针显然也已求得。
- 我们跳转到p的 fail指针指向的结点 $fail[p]$
  - 如果结点 $fail[p]$ 通过字母 c连接到的子结点 w存在：
    - 则让u的fail指针指向这个结点 w $fail[u]=w$
    - 相当于在 p和 $fail[p] $后面加一个字符 c 就构成了$fail[u]$
  - 如果不存在
    - 那么我们继续找到 $fail[fail[p]]$指针指向的结点，重复上述判断过程，一直跳 fail 指针直到根节点。
    - 如果真的没有，就令 $fail[u]$= 根节点。

二、模板

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<string>
#include<queue>
#include<algorithm>
#include<cmath>
using namespace std;
const int MAX=1e6,TYPE=26;
struct ac_automation{
    int trie[MAX][TYPE],num[MAX],tot,fail[MAX];
    void readin(char * a)
    {
```

```
        int k=strlen(a),now=0;
        for(int i=0;i<k;i++)
        {
            if(!trie[now][a[i]-'a'])
                trie[now][a[i]-'a']=++tot;
            now=trie[now][a[i]-'a'];
        }
        num[now]++;
    }
    void getfail()
    {
        queue<int> q;
        memset(fail,0,sizeof(fail));
        for(int i=0;i<TYPE;i++)
            if(trie[0][i])
                q.push(trie[0][i]);
        while(!q.empty())
        {
            int k=q.front();
            q.pop();
            for(int i=0;i<TYPE;i++)
                if(trie[k][i])
                {
                    fail[trie[k][i]]=trie[fail[k]][i];
                    q.push(trie[k][i]);
                }
                else
                    trie[k][i]=trie[fail[k]][i];//（1）重点：构造fail图
        }
    }
    int find(char *s)
    {
        int ans=0,p=0;
        for(int i=0;s[i];i++)
        {
            p=trie[p][s[i]-'a'];
            //看似是沿着trie走了一步，实际上已经开始失配跳fail了，见（1）
            for(int j=p;j and ~num[j];j=fail[j])    ans+=num[j],num[j]=-1;
            //查找相同后缀，防止漏查，如abcd匹配{abcd,cd}□防止漏查cd
        }
        return ans;
    }
}ac;
char str[MAX],tem[MAX];
int main()
{
    int k;
    scanf("%d",&k);
    for(int i=1;i<=k;i++)
```

```
    {
        scanf("%s",tem);
        ac.readin(tem);
    }
    scanf("%s",str);
    ac.getfail();
    printf("%d",ac.find(str));
    return 0;
}
```

三、例题与技巧

1.病毒[POI2000]

题意：给出多个01串，判断是否存在一个无限长的01串，使得给出的01串都不是它的子串。

分析与题解

利用给定的01串构造自动机，显然，只要判断是否存在一个01串能够在这个自动机上无限跑下去。即能否在这个**fail图**上找到一个环，使得0节点在这个环上，并且环上无“危险节点”（即给出的01串的末尾节点）。注意这个危险节点，如果一个点通过fail边直接或者间接的指向一个危险节点，那么它也是个危险节点（利用fail的性质：一个节点A的fail指针指向的节点B实际上B所代表的字符串是A代表的字符串的**最长后缀**

在建图建fail边的时候，更新节点的危险属性，即如果$fail[i]$危险，那么i也危险。随后dfs找环即可。

2. AC自动机+树上差分

在进行多模式串匹配的时候，一般情况下可以暴力跳fail边，但是这样做在一些题目中会被卡掉，此时记录自动机上的每个状态被匹配了几次，最后求出每个模式串在 Trie 上的终止节点在 fail 树上的子树总匹配次数可以优化。详情见 【模板AC自动机（二次加强版）.

3.阿狸的打字机 (fail树)

题意：给出总长度数量级约为1e5的多个字符串，1e5次询问,每次询问(x,y):返回第x个字符串在第y个字符串里出现的次数。

分析与题解

利用fail的性质：一个节点A的fail指针指向的节点B实际上B所代表的字符串是A代表的字符串的**最长后缀**，所以如果某个节点A其fail指针直接或者间接（多次跳fail的指向B的结束节点，则显然字符串B是A所代表字符串的后缀，则B为A的子串。所以，这里**将fail边反向**，所有fail边就变成了**fail树**（每个fail边最终都会走到节点0，且每个fail边（正向）只会指向深度小于本身的点，所以构成树结构），于是只要检查B节点的子树中，有多少属于A节点，即可得出B在A中出现过多少次。

于是，一个字符串问题被我们变成了**子树查询问题**，利用树的dfs序，这个问题就变成了简单的单点插入区间求和，这里有个小问题就是此时要离线计算，保证每一个字符串只插入一次，否则多次插入相同的

字符串导致tle

4.[JSOI2007]文本生成器 dp

题意：给定N个**$N \le 60$**由大写字母组成的单词（每个单词长度小于100），并给出一个长度**$M \le 100$**，求出有多少个长度为M的字符串，至少包含一个单词。

分析：求方案数，想到dp至少包含，可以先求出不符合要求的字符串，最后用总方案数减去不可行的方案数。

此时题目转化为，有多少长度为M的字符串不包含所有给定的单词，看起来是不是和病毒那题很像？但这题不再是无限长，而且要求求出方案数，考虑一个十分套路的dp

$dp[i,j] - > dp[i+1,trie[j].ch[k]]$

代表当前长度为i时，已经匹配到j节点的方案数。主要判断危险节点，要绕开(关于危险节点，见1.[POI2000]病毒)

再次注意，这个dp方程看似与fail边无关，但是我们在建立fail图的过程中，对于trie树中添加了许多不存在的边，构成了fail图，所以我们在图上跑dp实际上已经在跳fail了，**fail数组只是在匹配后缀，实际上trie图也是由fail数组构造的**.

这类dp经常与矩阵快速幂结合，见下题。

5.POJ2778(矩阵快速幂)

题意：给定m个长度不超过10 的由ATCG组成的字母串，求出长度为n的不包含这些给定字母串的字符串。**$m \le 10,n \le 2e9$**

分析：是不是看起来和上一题差不多？是不是想用dp看看数据范围，清醒了吗？正常的dp显然不现实。

现在我们换一个角度，从**图**的角度来看这个问题，**fail图**.

我们再来看看上一题写出的状态转移方程，实质上等价于从根节点开始转移，转移i次，并且保证不经过危险节点的方案数。再看看数据范围，模式串的长度个数与构成字母都很少，但n却很大，那么我们将fail图去掉危险节点，是不是我们只要求根节点走n步到达j节点的方案数就可以了？好了，这题的trie图实际上特别小，将trie图用邻接矩阵储存，这题就变成了**矩阵快速幂**；求出A^n,最后的答案就是$\sum {A^n[0,i]}$.

6.HDU2243(矩阵快速幂)

题意：给定m个长度不超过5 的字母串，求出长度**不超过**n的至少包含一个这些给定字母串的字符串。**$m \le 6,n<2^{31}$**

题解$ans=S_n-T_n$

和上一题很像，但是却要求和，于是我们改造一下矩阵 $$ \begin{bmatrix} A & E \\ 0 & E

\end{bmatrix} \times \begin{bmatrix} T_n \\ A \end{bmatrix} = \begin{bmatrix} T_{n+1} \\ A \end{bmatrix} $$ A为trie图的邻接矩阵

7.HDU2825(状压dp)

题意:求长度为n□**包含至少k个**模式串的字符串数量。

题解:好了，现在变成了至少k个，而不是至少一个，我们上面惯用的取反手段失效了。老老实实正面硬刚：子集问题，考虑**状压dp**

$$dp[i + 1][trie[j][x]][K | num[trie[j][x]]] += dp[i][j][K];$$

$$dp[i][j][k] $$表示当前走到了字符串的第i位，位于trie的第j个节点上，此时已经匹配的模式串是k□子集状压）

可以看出这个实际上就是之前最简单的那个dp上加了一维信息，维护子集而已。