[AC自动机学习小结](http://www.cnblogs.com/DreamUp/archive/2010/07/29/1787682.html)

        首先简要介绍一下AC自动机：Aho-Corasick automation，该算法在1975年产生于贝尔实验室，是著名的多模匹配算法之一。一个常见的例子就是给出n个单词，再给出一段包含m个字符的文章，让你找出有多少个单词在文章里出现过。要搞懂AC自动机，先得有模式树（字典树）Trie和KMP模式匹配算法的基础知识。AC自动机算法分为3步：构造一棵Trie树，构造失败指针和模式匹配过程。

        如果你对KMP算法和了解的话，应该知道KMP算法中的next函数（shift函数或者fail函数）是干什么用的。KMP中我们用两个指针i和j分别表示，A[i-j+ 1..i]与B[1..j]完全相等。也就是说，i是不断增加的，随着i的增加j相应地变化，且j满足以A[i]结尾的长度为j的字符串正好匹配B串的前 j个字符，当A[i+1]≠B[j+1]，KMP的策略是调整j的位置（减小j值）使得A[i-j+1..i]与B[1..j]保持匹配且新的B[j+1]恰好与A[i+1]匹配，而next函数恰恰记录了这个j应该调整到的位置。同样AC自动机的失败指针具有同样的功能，也就是说当我们的模式串在Tire上进行匹配时，如果与当前节点的关键字不能继续匹配的时候，就应该去当前节点的失败指针所指向的节点继续进行匹配。

这里有一个帖子讲的特别好，强烈推荐[AC自动机算法详解](http://www.cppblog.com/mythit/archive/2009/04/21/80633.html)

**HDU2222代码：（经典的AC自动机上面的帖子就是以它为例讲的）**

**http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gifhttp://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif代码**

**#include<iostream>**  
**using namespace std;  
struct node{  
 int count;  
 node \*fail;  
 node \*next[26];  
 node(){  
 fail=NULL;  
 count=0;  
 memset(next,NULL,sizeof(next));  
 }  
}\*q[500001];  
char keyword[51];  
char str[1000001];  
int head,tail;  
  
void insert(char \*str,node \*root)  
{  
 node \*p=root;  
 for(int i=0;str[i];i++)  
 {  
 int id=str[i]-'a';  
 if(p->next[id]==NULL)  
 p->next[id]=new node();  
 p=p->next[id];  
 }  
 p->count++;  
}  
  
void build(node \*root)  
{  
 int i;  
 root->fail=NULL;  
 q[head++]=root;  
 while(head!=tail)  
 {  
 node \*temp=q[tail++];  
 node \*p=NULL;  
 for(i=0;i<26;i++)  
 {  
 if(temp->next[i]!=NULL)  
 {  
 if(temp==root)  
 temp->next[i]->fail=root;  
 else{  
 p=temp->fail;  
 while(p!=NULL)  
 {  
 if(p->next[i]!=NULL)  
 {  
 temp->next[i]->fail=p->next[i];  
 break;  
 }  
 p=p->fail;  
 }  
 if(p==NULL)  
 temp->next[i]->fail=root;  
 }  
 q[head++]=temp->next[i];  
 }  
 }  
 }  
}  
  
int query(node \*root)  
{  
 int cnt=0,id;  
 node \*p=root;  
 for(int i=0;str[i];i++)  
 {  
 id=str[i]-'a';  
 while(p->next[id]==NULL && p!=root)  
 p=p->fail;  
 p=p->next[id];  
 if(p==NULL) //yasherp,when s[i]='p',p==NULL**  
 **p=root;  
 node \*temp=p;  
 while(temp!=root && temp->count!=-1)  
 {  
 cnt+=temp->count;  
 temp->count=-1;  
 temp=temp->fail;  
 }  
 }  
 return cnt;  
}  
int main()  
{  
 int n,t;  
 scanf("%d",&t);  
 while(t--)  
 {  
 head=tail=0;  
 node \*root=new node();  
 scanf("%d",&n);  
 getchar();  
 while(n--)  
 {  
 gets(keyword);  
 insert(keyword,root);  
 }  
 build(root);  
 scanf("%s",str);  
 printf("%d\n",query(root));  
 }  
 return 0;  
}**

**PKU1204代码：（14724K 1610MS）**

**http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gifhttp://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif代码**

**#include<string.h>**  
**#include<iostream>**  
**using namespace std;  
#define N 1002**  
**typedef struct tree{  
 int count;  
 struct tree \*fail;  
 struct tree \*next[26];  
 tree(){  
 count=-1;  
 fail=NULL;  
 memset(next,NULL,sizeof(next));  
 }  
}\*Tree,T;  
Tree q[N\*N];  
Tree root;  
char map[N][N];  
int result[N][4],length[N];  
int l,c,w;  
int dir[8][2]={{-1,0},{-1,1},{0,1},{1,1},{1,0},{1,-1},{0,-1},{-1,-1}};  
void insert(char \*str,int k)  
{  
 Tree p=root;  
 for(int i=0;str[i];i++)  
 {  
 int id=str[i]-'A';  
 if(p->next[id]==NULL)  
 p->next[id]=new tree();  
 p=p->next[id];  
 }  
 p->count=k;  
}  
void build()  
{  
 int head=0,tail=0,i;  
 root->fail=NULL;  
 q[head++]=root;  
 while(head!=tail)  
 {  
 Tree temp=q[tail++];  
 Tree p=NULL;  
 for(i=0;i<26;i++)  
 {  
 if(temp->next[i]!=NULL)  
 {  
 if(temp==root)  
 temp->next[i]->fail=root;  
 else{  
 p=temp->fail;  
 while(p!=NULL)  
 {  
 if(p->next[i]!=NULL)  
 {  
 temp->next[i]->fail=p->next[i];  
 break;  
 }  
 p=p->fail;  
 }  
 if(p==NULL)  
 temp->next[i]->fail=root;  
 }  
 q[head++]=temp->next[i];  
 }  
 }  
 }  
}  
void search(int x,int y,int k)  
{  
 int x1=x, y1=y;  
 Tree p=root;  
 while(x1>=0 && x1<l && y1>=0 && y1<c)  
 {  
 int id=map[x1][y1]-'A';  
 while(p->next[id]==NULL && p!=root)  
 p=p->fail;  
 p=p->next[id];  
 if(p==NULL)  
 p=root;  
 Tree temp=p;  
 while(temp!=root && temp->count!=-1)  
 {  
 result[temp->count][0]=x1-length[temp->count]\*dir[k][0];  
 result[temp->count][1]=y1-length[temp->count]\*dir[k][1];  
 result[temp->count][2]=k+'A';  
 temp->count=-1;  
 temp=temp->fail;  
 }  
 x1+=dir[k][0]; y1+=dir[k][1];  
 }  
}  
void slove()  
{  
 int i,j,k;  
 for(i=0;i<l;i++)  
 for(j=0;j<c;j++)  
 if(i==0 || j==0 || i==l-1 || j==c-1)  
 for(k=0;k<8;k++)  
 search(i,j,k);  
}  
int main()  
{  
 char word[N];  
 int i;  
 scanf("%d%d%d",&l,&c,&w);  
 getchar();  
 for(i=0;i<l;i++)  
 gets(map[i]);  
 root=new tree();  
 for(i=0;i<w;i++)  
 {  
 gets(word);  
 insert(word,i);  
 length[i]=strlen(word)-1;  
 }  
 build();  
 slove();  
 for(i=0;i<w;i++)  
 printf("%d %d %c\n",result[i][0],result[i][1],result[i][2]);  
 return 0;  
}**

**用字典树做的是14320K 1485MS见链接**[**http://www.cnblogs.com/DreamUp/archive/2010/07/23/1783410.html**](http://www.cnblogs.com/DreamUp/archive/2010/07/23/1783410.html)

**HDU2896 同HDU2222如出一辙 给定目标串求其中出现的模板串**

**用数组表示的树形链表（静态版本）171MS 23068K**

**http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gifhttp://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif代码**

**#include<iostream>**  
**using namespace std;  
struct node {  
 int fail,count;  
 int next[100];  
 void init(){  
 memset(next,NULL,sizeof(next));  
 fail=-1;  
 count=0;  
 }  
};  
node tire[100000];  
int n,cnt,m;  
char str[10010];  
int mark[1010];  
void insert(char \*s,int d)  
{  
 int root=0;  
 while(\*s)  
 {  
 int t=\*s-30;  
 if(tire[root].next[t]==NULL)  
 {  
 tire[++cnt].init();  
 tire[root].next[t]=cnt;  
 }  
 root=tire[root].next[t];  
 s++;  
 }  
 tire[root].count=d;  
}  
int que[100000],tail,head;  
void build()  
{  
 int q,temp;  
 tail=head=0; que[0]=0;  
 while(tail<=head)  
 {  
 int now = que[tail++];  
 for(int t=0; t<100;t++)  
 {  
 if(tire[now].next[t])  
 {  
 temp=tire[now].next[t];  
 q=tire[now].fail;  
 while(q!=-1 && tire[q].next[t]==NULL)  
 q=tire[q].fail;  
 if(q==-1)  
 tire[temp].fail=0; //指向根**  
 **else**  
 **tire[temp].fail=tire[q].next[t];  
 que[++head]=temp;  
 }  
 }  
 }  
}  
void match(char \*s)  
{  
 int root=0,p;  
 while(\*s)  
 {  
 int t=\*s-30;  
 if(tire[root].next[t])  
 root=tire[root].next[t];  
 else**  
 **{  
 p=tire[root].fail;  
 while(p!=-1 && tire[p].next[t]==NULL)  
 p=tire[p].fail;  
 if(p==-1)  
 root=0;  
 else**  
 **root=tire[p].next[t];  
 }  
 if(tire[root].count)  
 {  
 mark[tire[root].count]=1;  
 p=tire[root].fail;  
 while(p!=NULL && mark[tire[p].count]==0)  
 {  
 mark[tire[p].count]=1;  
 p=tire[p].fail;  
 }  
 }  
 s++;  
 }  
}  
int main()  
{  
 int i,j;  
 cnt=0;  
 tire[0].init();  
 scanf("%d",&n);  
 for(i=1;i<=n;i++)  
 {  
 scanf("%s",str);  
 insert(str,i);  
 }  
 build();  
 scanf("%d",&m);  
 int ans=0;  
 for(i=1;i<=m;i++)  
 {  
 for(j=0;j<=n;j++)  
 mark[j]=0;  
 scanf("%s",str);  
 match(str);  
 int num=0;  
 for(j=1;j<=n;j++)  
 num+=mark[j];  
 if(num)  
 {  
 ans++;  
 printf("web %d:",i);  
 for(j=1;j<=n;j++)  
 if(mark[j])  
 printf(" %d",j);  
 puts("");  
 }  
 }  
 printf("total: %d\n",ans);  
 return 0;  
}**

**在discuss里看了数据，上面程序错了，下面的(用指针实现的动态版本)对了。上面那个程序有问题，但是也AC了，建议加强数据啊**

**3  
abs  
shd  
h  
1  
abshe  
  
2  
ct  
actg  
1**

**贴上代码（不明白temp->count=0;加上就WA，去掉就AC了) 171MS 23516K**

**http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gifhttp://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif代码**

**#include<iostream>**  
**#include<string.h>**  
**using namespace std;  
typedef struct tree{  
 int count;  
 struct tree \*fail;  
 struct tree \*next[100];  
 tree(){  
 count=0;  
 fail=NULL;  
 memset(next,0,sizeof(next));  
 }  
}\*Tree,T;  
int n,cnt,m;  
char str[10010];  
int mark[502];  
Tree root;  
void insert(char \*s,int d)  
{  
 Tree p=root;  
 while(\*s)  
 {  
 int id=\*s-30;  
 if(p->next[id]==NULL)  
 p->next[id]=new tree();  
 p=p->next[id];  
 s++;  
 }  
 p->count=d;  
}  
Tree que[100010];  
int tail,head;  
void build()  
{  
 int i;  
 head=tail=0;  
 root->fail=0;  
 que[head++]=root;  
 while(head!=tail)  
 {  
 Tree temp=que[tail++];  
 Tree p=NULL;  
 for(i=0;i<100;i++)  
 {  
 if(temp->next[i]!=NULL)  
 {  
 if(temp==root)  
 temp->next[i]->fail=root;  
 else**  
 **{  
 p=temp->fail;  
 while(p!=NULL)  
 {  
 if(p->next[i]!=NULL)  
 {  
 temp->next[i]->fail=p->next[i];  
 break;  
 }  
 p=p->fail;  
 }  
 if(p==NULL)  
 temp->next[i]->fail=root;  
 }  
 que[head++]=temp->next[i];  
 }  
 }  
 }  
}  
void match(char \*s)  
{  
 Tree p=root;  
 while(\*s)  
 {  
 int id=\*s-30;  
 while(p->next[id]==NULL && p!=root)  
 p=p->fail;  
 p=p->next[id];  
 if(p==NULL)  
 p=root;  
 Tree temp=p;  
 while(temp!=root)  
 {  
 mark[temp->count]=1;  
 //temp->count=0;**  
 **temp=temp->fail;  
 }  
 s++;  
 }  
}  
int main()  
{  
 int i,j;  
 cnt=0;  
 root=new tree();  
 scanf("%d",&n);  
 for(i=1;i<=n;i++)  
 {  
 scanf("%s",str);  
 insert(str,i);  
 }  
 build();  
 scanf("%d",&m);  
 int ans=0;  
 for(i=1;i<=m;i++)  
 {  
 memset(mark,0,sizeof(mark));  
 scanf("%s",str);  
 match(str);  
 int num=0;  
 for(j=1;j<=n;j++)  
 num+=mark[j];  
 if(num)  
 {  
 ans++;  
 printf("web %d:",i);  
 for(j=1;j<=n;j++)  
 if(mark[j])  
 printf(" %d",j);  
 puts("");  
 }  
 }  
 printf("total: %d\n",ans);  
 return 0;  
}**

**PKU3691 题目大意是说给定一个长度最大为1000的仅由A,T,C,G四个字符组成的字符串，要求改变最少的字符个数，使得得到的字符串中不含有任意预先给定的一系列子串。**

AC自动机+DP（注意这个AC自动机失败节点存在危险的结点的传递）4132K 110MS

DISCUSS帖子<http://acm.pku.edu.cn/JudgeOnline/showmessage?message_id=122852>

题目分析帖子<http://www.cnblogs.com/woodfish1988/archive/2008/11/19/1303492.html> 

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gifhttp://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif代码

#include<stdio.h>  
#include<string.h>  
#define N 1005  
  
struct trie{  
 int cnt,fail;  
 int next[4];  
}Tree[N];  
int que[N];  
int dp[N][N];  
char str[N];  
int nodes;  
  
void init(int t)  
{  
 Tree[t].cnt=0;  
 Tree[t].fail=-1;  
 memset(Tree[t].next,-1,sizeof(Tree[t].next));  
}  
int change(char ch)  
{  
 switch(ch){  
 case'A':return 0;  
 case'G':return 1;  
 case'C':return 2;  
 case'T':return 3;  
 }  
 return 0;  
}  
  
void insert(char \*s)  
{  
 int i=0,t=0;  
 while(\*s)  
 {  
 int id=change(\*s);  
 if(Tree[t].next[id]==-1)  
 {  
 init(nodes);// 初始化结点  
 Tree[t].next[id]=nodes++;  
 }  
 t=Tree[t].next[id];  
 if(Tree[t].cnt!=0)  
 break;  
 s++;  
 }  
 Tree[t].cnt++;  
}  
  
void build()  
{  
 int head=0,tail=0;  
 que[head++]=0;  
 while(head!=tail)  
 {  
 int now=que[tail++];  
 for(int t=0;t<4;t++)  
 {  
 int tmp=Tree[now].next[t];  
 if(tmp!=-1)  
 {  
 if(now==0)  
 Tree[tmp].fail=0;  
 else  
 {  
 int p=Tree[now].fail;  
 while(p!=-1)  
 {  
 if(Tree[p].next[t]!=-1)  
 {  
 Tree[tmp].fail=Tree[p].next[t];  
 //存在危险的结点的传递，详见PKU该题的DISCUSS   
 if(Tree[Tree[p].next[t]].cnt)  
 Tree[tmp].cnt++;  
 break;  
 }  
 p=Tree[p].fail;  
 }  
 if(p==-1)  
 Tree[tmp].fail=0;  
 }  
 que[head++]=tmp;  
 }  
 }  
 }  
}  
  
int len;  
void DP()  
{  
 int i,j,k,t;  
 memset(dp,-1,sizeof(dp));  
 dp[0][0]=0;  
 for(i=0;i<len;i++)  
 {  
 for(j=0;j<nodes;j++)  
 {  
 if(dp[i][j]!=-1)  
 {  
 for(k=0;k<4;k++)  
 {  
 t=j;  
 while(t!=-1)  
 {  
 if(Tree[t].next[k]!=-1)  
 break;  
 t=Tree[t].fail;  
 }  
 if(t==-1)  
 t=0;  
 else  
 t=Tree[t].next[k];  
 //状态转移方程：dp[i+1][Tree[t].next[k]] = MIN (dp[i][j] + (s[i]!=k) )   
 if(Tree[t].cnt==0 && dp[i][j]!=-1)  
 if(dp[i+1][t]>dp[i][j]+(change(str[i])!=k) || dp[i+1][t]==-1)  
 dp[i+1][t]=dp[i][j]+(change(str[i])!=k);  
 }  
 }  
 }  
 }  
}  
  
int main()  
{  
 int i,n;  
 int min,cas=1;  
 char word[30];  
 while(scanf("%d",&n),n)  
 {  
 nodes=1;  
 init(0);  
 for(i=0;i<n;i++)  
 {  
 scanf("%s",word);  
 insert(word);  
 }  
 build();  
 scanf("%s",str);  
 len=strlen(str);  
 DP();  
 min=-1;  
 for(i=0;i<nodes;i++)  
 if(dp[len][i]!=-1 &&(dp[len][i]<min || min==-1))  
 min=dp[len][i];  
 printf("Case %d: %d\n",cas++,min);  
 }  
 return 0;  
}