AC自动机习题。

前话：AC自动机常用于解决多模式字符串匹配问题，主要运用到了$Trie+kmp$的思想，而kmp是单模式字符串匹配，求$fail$与kmp的$next[]$数组类似的思想，只不过$fail$是记录最大后缀，$next[]$最大前后缀。
都能在$O(n)$复杂度内解决问题。但是暴力的跳$fail$会浪费很多时间，导致一个结点被多次访问，所以考虑树上跑拓扑优化。

1.P3808 【模板】AC自动机（简单版）

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=2e6+5,M=1e6+5,inf=0x3f3f3f3f,mod=1e9+7;
#define mst(a) memset(a,0,sizeof a)
#define lx x<<1
#define rx x<<1|1
#define reg register
#define PII pair<int,int>
#define fi first
#define se second
int tri[N][26],fail[N],cnt[N],id;
void Insert(string s){  //构建字典树. 将模式串插入字典树.
    int rt=0;
    for(int i=0;i<s.size();i++){
         int j=s[i]-'a';
        if(!tri[rt][j]) //如果rt没有j儿子,则新增结点.
            tri[rt][j]=++id;
        rt=tri[rt][j];//rt变为它的j儿子.
    }
    cnt[rt]++;//记录模式串的个数.
}
void getFail(){ //求失配指针fail
    queue<int>q;
    for(int i=0;i<26;i++)   //显然第二层结点没有失配后缀 所以直接指向0
        if(tri[0][i]) fail[tri[0][i]]=0,q.push(tri[0][i]);  //第二层即所有模式串的首字母丢进队列.
    while(!q.empty()){
        int u=q.front();q.pop();
    for(int i=0;i<26;i++){  //如果当前结点u的i儿子存在 则它的fail指针指向(它父亲u失配指针指向的i儿子结点)，然后u的i儿子入队
         if(tri[u][i]) fail[tri[u][i]]=tri[fail[u]][i],q.push(tri[u][i]);
         else tri[u][i]=tri[fail[u]][i];
    }       //否则u的i儿子为它的父亲u失配时指向结点的i儿子结点.
    }
}
int query(string s){    //查询模式串在文本串出现过的次数.
    int u=0,ans=0;
    for(int i=0;i<s.size();i++){
        u=tri[u][s[i]-'a'];
        for(int j=u;j&&~cnt[j];j=fail[j])   //进行匹配直到已经匹配过或者匹配到根.
            ans+=cnt[j],cnt[j]=-1;
```

```
    }
    return ans;
}
int main(){
    int n;
    string s;
    scanf("%d",&n);
    for(int i=0;i<n;i++)
        cin>>s,Insert(s);
    getFail();
    cin>>s;
    printf("%d\n",query(s));
    return 0;
}
```

思路：板子题，不多说，具体看代码。

2.P3796 【模板】AC自动机（加强版）

思路：求出现次数最多的字符串，我们只需用一个$cnt[]$数组改为维护最后一次出现的模式串的编号，然后用$vis[]$数组记录答案。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=155*70,M=1e6+10,inf=0x3f3f3f3f,mod=1e9+7;
#define mst(a) memset(a,0,sizeof a)
#define lx x<<1
#define rx x<<1|1
#define reg register
#define PII pair<int,int>
#define fi first
#define se second
int tri[N][26],fail[N],cnt[N],id,vis[155];
char s[155][100],t[M];
void Insert(char *s,int num){   //构建字典树. 将模式串插入字典树.
    int rt=0,n=strlen(s);
    for(int i=0;i<n;i++){
         int j=s[i]-'a';
        if(!tri[rt][j]) //如果rt没有j儿子,则新增结点.
            tri[rt][j]=++id;
        rt=tri[rt][j];//rt变为它的j儿子.
    }
    cnt[rt]=num;//记录模式串的个数.
}
void getFail(){ //求失配指针fail
    queue<int>q;
    for(int i=0;i<26;i++)   //显然第二层结点没有失配后缀 所以直接指向0
        if(tri[0][i]) fail[tri[0][i]]=0,q.push(tri[0][i]);  //第二层即所有模式串的首字母丢进队列.
    while(!q.empty()){
        int u=q.front();q.pop();
    for(int i=0;i<26;i++){  //如果当前结点u的i儿子存在 则它的fail指针指向(它父亲u失配指针指向的i儿子结点)，然后u的i儿子入队
         if(tri[u][i]) fail[tri[u][i]]=tri[fail[u]][i],q.push(tri[u][i]);
         else tri[u][i]=tri[fail[u]][i];
```

```
    }       //否则u的i儿子为它的父亲u失配时指向结点的i儿子结点.
    }
}
void query(char * s){   //查询模式串在文本串出现过的次数.
    int u=0,n=strlen(s);
    for(int i=0;i<n;i++){
        u=tri[u][s[i]-'a'];
        for(int j=u;j;j=fail[j])
            vis[cnt[j]]++;
    }
}
int main(){
    int n;
    while(scanf("%d",&n)&&n){
    mst(tri),mst(fail),mst(cnt),mst(vis),id=0;
    for(int i=1;i<=n;i++)
        scanf("%s",s[i]),Insert(s[i],i);
    getFail();
    scanf("%s",t);
    query(t);
    int ans=0;
    for(int i=1;i<=n;i++) ans=max(ans,vis[i]);
    printf("%d\n",ans);
    for(int i=1;i<=n;i++)
        if(vis[i]==ans) cout<<s[i]<<endl;
    }
    return 0;
}
```

3.P5357 【模板】AC自动机（二次加强版）

题意：求每个模式串在文本串中出现的次数，可能含重复串。

因为可能含重复串，所以我们需要用一个数组$bj[]$来标记第一次在字典树的出现的模式串对应的编号。因为之前的跳$fail$都是暴力跳的，一个结点可能会被重复访问多次，很浪费时间，所以我们考虑在字典树上跑拓扑，以$fail$为一条出边，进行拓扑状态转移，从入度为0的开始跑，这样每个结点就不会重复访问了。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=5e6+10,M=2e6+10,inf=0x3f3f3f3f,mod=1e9+7;
#define mst(a) memset(a,0,sizeof a)
#define lx x<<1
#define rx x<<1|1
#define reg register
#define PII pair<int,int>
#define fi first
#define se second
int tri[N][26],fail[N],cnt[N],id,vis[200010],bj[N],ans[N],in[N];
char s[200010],t[M];
void Insert(char *s,int num){   //构建字典树. 将模式串插入字典树.
    int rt=0,n=strlen(s);
    for(int i=0;i<n;i++){
         int j=s[i]-'a';
        if(!tri[rt][j]) //如果rt没有j儿子,则新增结点.
```

```
            tri[rt][j]=++id;
        rt=tri[rt][j];//rt变为它的j儿子.
    }
    if(!cnt[rt]) cnt[rt]=num;//记录模式串的个数.
    bj[num]=cnt[rt];
}
void getFail(){ //求失配指针fail
    queue<int>q;
    for(int i=0;i<26;i++)   //显然第二层结点没有失配后缀 所以直接指向0
        if(tri[0][i]) fail[tri[0][i]]=0,q.push(tri[0][i]);  //第二层即所有模式串的首字母丢进队列.
    while(!q.empty()){
        int u=q.front();q.pop();
    for(int i=0;i<26;i++){  //如果当前结点u的i儿子存在 则它的fail指针指向(它父亲u失配指针指向的i儿子结点)，然后u的i儿子入队
         if(tri[u][i]) fail[tri[u][i]]=tri[fail[u]][i],q.push(tri[u][i]),++in[fail[tri[u][i]]];
         else tri[u][i]=tri[fail[u]][i];
    }       //否则u的i儿子为它的父亲u失配时指向结点的i儿子结点.
    }
}
void query(char * s){   //查询模式串在文本串出现过的次数.
    int u=0,n=strlen(s);
    for(int i=0;i<n;i++){
        u=tri[u][s[i]-'a'];
        ++ans[u];//打上标记.
    }
}
void toposort(){
    queue<int>q;
     for(int i=1;i<=id;i++)
        if(!in[i]) q.push(i);
    while(!q.empty()){
        int u=q.front();q.pop();
        vis[cnt[u]]=ans[u];
        int v=fail[u];
        in[v]--;
        ans[v]+=ans[u];
        if(!in[v]) q.push(v);
    }
}
int main(){
    int n;
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
        scanf("%s",s),Insert(s,i);
    getFail();
    scanf("%s",t);
    query(t);
    toposort();
    for(int i=1;i<=n;i++) printf("%d\n",vis[bj[i]]);
    return 0;
}
```

4.P3966 [TJOI2013]单词

思路：把单词用#连接起来成为文本串，然后跑AC自动机即可。

注意打$ans[]$标记的时候，特判一下#，遇到#，当前结点u归0.

坑点：字符串大小应该开大点，之前开得太小，调了好半天。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=3e5+10,M=2e6+10,inf=0x3f3f3f3f,mod=1e9+7;
#define mst(a) memset(a,0,sizeof a)
#define lx x<<1
#define rx x<<1|1
#define reg register
#define PII pair<int,int>
#define fi first
#define se second
int tri[N][26],fail[N],cnt[N],id,vis[200010],bj[N],ans[N],in[N],len;
char s[N],t[M];
void Insert(char *s,int num){   //构建字典树. 将模式串插入字典树.
    int rt=0,n=strlen(s);
    for(int i=0;i<n;i++){
        t[len++]=s[i];
         int j=s[i]-'a';
        if(!tri[rt][j]) //如果rt没有j儿子,则新增结点.
            tri[rt][j]=++id;
        rt=tri[rt][j];//rt变为它的j儿子.
    }
    t[len++]='#';
    if(!cnt[rt]) cnt[rt]=num;//记录模式串的个数.
    bj[num]=cnt[rt];
}
void getFail(){ //求失配指针fail
    queue<int>q;
    for(int i=0;i<26;i++)   //显然第二层结点没有失配后缀 所以直接指向0
        if(tri[0][i]) fail[tri[0][i]]=0,q.push(tri[0][i]);  //第二层即所有模式串的首字母丢进队列.
    while(!q.empty()){
        int u=q.front();q.pop();
    for(int i=0;i<26;i++){  //如果当前结点u的i儿子存在 则它的fail指针指向(它父亲u失配指针指向的i儿子结点)，然后u的i儿子入队
         if(tri[u][i]) fail[tri[u][i]]=tri[fail[u]][i],q.push(tri[u][i]),++in[fail[tri[u][i]]];
          else tri[u][i]=tri[fail[u]][i];
    }       //否则u的i儿子为它的父亲u失配时指向结点的i儿子结点.
    }
}
void query(char * s){   //查询模式串在文本串出现过的次数.
    int u=0,n=strlen(s);
    for(int i=0;i<n;i++){
        if(s[i]=='#'){
            u=0;
            continue;
        }
        u=tri[u][s[i]-'a'];
        ++ans[u];//打上标记.
    }
}
```

```
void toposort(){
    queue<int>q;
     for(int i=1;i<=id;i++)
        if(!in[i]) q.push(i);
    while(!q.empty()){
        int u=q.front();q.pop();
        vis[cnt[u]]=ans[u];
        int v=fail[u];
        in[v]--;
        ans[v]+=ans[u];
        if(!in[v]) q.push(v);
    }
}
int main(){
    int n;
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
        scanf("%s",s),Insert(s,i);
    getFail();
    query(t);
    toposort();
    for(int i=1;i<=n;i++) printf("%d\n",vis[bj[i]]);
    return 0;
}
```