AC自动机

一个**确定有限状态自动机（DFA）**由五部分组成：

字符集Σ, 状态集合Q，起始状态$start$, 接受状态集合F，转移函数δ

在AC自动机中，状态集合就是字典树的节点，字符集就是a~z等，状态转移函数就是tr[u][c]；开始状态就是字典树的根节点，接收状态就是字典树中标记的字符串结尾节点组成的集合。

AC自动机是以Trie的结构位基础，结合KMP的思想建立的。

AC自动机在初始化时会将若干个模式串丢到一个Trie中，

luogu P3796 【模板】AC自动机（加强版）

```
#include<bits/stdc++.h>
using namespace std;
const int N=156,L=1e6+6;
namespace AC {
    const int SZ=N*80;
    int tot,tr[SZ][26];
    int fail[SZ],idx[SZ],val[SZ];
    int cnt[N];  // 记录第 i 个字符串的出现次数
    void init(){
        memset(fail,0,sizeof(fail)); //失配数组
        memset(tr,0,sizeof(tr)); //Trie
        memset(val,0,sizeof(val)); //val[i]以节点i结尾的字符串个数
        memset(cnt,0,sizeof(cnt));//字符串的个数
        memset(idx,0,sizeof(idx));//节点i结尾的字符串的编号
        tot=0;
    }
    void insert(char *s, int id) {  // id 表示原始字符串的编号
        int u=0;
        for(int i=1;s[i];i++){
            if(!tr[u][s[i]-'a']) tr[u][s[i]-'a']=++tot;
            u=tr[u][s[i]-'a'];  // 转移
        }
        idx[u]=id;  // 以 u 为结尾的字符串编号为 idx[u]
    }
    queue<int> q;
    void build(){
        for(int i=0;i<26;i++)
```

```
            if (tr[0][i]) q.push(tr[0][i]);
        while(!q.empty()){
            int u = q.front();
            q.pop();
            for (int i=0;i<26;i++) {
                if(tr[u][i]){
                    fail[tr[u][i]]=tr[fail[u]][i];  // fail数组：同一字符可以匹配的其他位置
                    q.push(tr[u][i]);
                }else tr[u][i]=tr[fail[u]][i];
            }
        }
    }
    int query(char *t){  // 返回最大的出现次数
        int u=0,res=0;
        for(int i=1;t[i];i++){
            u=tr[u][t[i]-'a'];
            for(int j=u;j;j=fail[j]) val[j]++;
        }
        for (int i=0;i<=tot;i++)
            if(idx[i]) res=max(res,val[i]),cnt[idx[i]]=val[i];
        return res;
    }
}  // namespace AC
int n;
char s[N][100],t[L];
signed main(){
    while (~scanf("%d",&n)){
        if(n==0) break;
        AC::init();  // 数组清零
        for(int i=1;i<=n;i++)
            scanf("%s",s[i]+1),AC::insert(s[i],i);  // 需要记录该字符串的序号
        AC::build();
        scanf("%s",t+1);
        int x=AC::query(t);
        printf("%d\n",x);
        for(int i=1;i<=n;i++)
            if(AC::cnt[i]==x) printf("%s\n",s[i]+1);
    }
    return 0;
}
```

AC自动机的拓扑建图优化

把trie中的fail都想象成一条条有向边，那么如果在某个点进行操作时，连出去的点也会操作(跳fail)。对于同一个字符，向上跳最多n次，且每个字符都要跳那么复杂度会退化为$O(n*m)$。拓扑排序可以让确定我们从深度大的点向上更新，保证每个点只跑一遍。

luoguP5357 【模板】AC自动机（二次加强版）

```
#include<bits/stdc++.h>
using namespace std;
const int N=2e6+7;
char S[N],T[N];
int n,cnt,vis[N],ans,in[N];
```

```
map<int,int>mp;

struct node{
    int ch[26],fail,flag,ans;
    void clear(){memset(ch,0,sizeof(ch)),fail=flag=ans=0;}
}tr[N]; //trie树

queue<int>q;

void insert(char *s,int num){ //在Trie树中插入模式串
    int u=1,len=strlen(s);
    for(int i=0;i<len;i++){
        int v=s[i]-'a';
        if(!tr[u].ch[v]) tr[u].ch[v]=++cnt;
        u=tr[u].ch[v];
    }
    if(!tr[u].flag) tr[u].flag=num;
    mp[num]=tr[u].flag;
}

void getFail(){   //获取失配以及连边的过程
    for(int i=0;i<26;i++) tr[0].ch[i]=1;
    q.push(1);
    while(!q.empty()){
        int u=q.front();
        q.pop();
        int Fail=tr[u].fail;
        for(int i=0;i<26;i++){
            int v=tr[u].ch[i];
            if(!v){tr[u].ch[i]=tr[Fail].ch[i];continue;} //失配
            tr[v].fail=tr[Fail].ch[i];
            in[tr[v].fail]++; //入度+1
            q.push(v);
        }
    }
}

void topo(){ //拓扑排序优化
    for(int i=1;i<=cnt;i++) if(!in[i]) q.push(i);
    while(!q.empty()){
        int u=q.front(),v=tr[u].fail;
        q.pop();
        vis[tr[u].flag]=tr[u].ans; //记录下当前的答案
        in[v]--;
        tr[v].ans+=tr[u].ans; //前缀和思想
        if(!in[v]) q.push(v);
    }
}

void query(char *s){ //查询
    int u=1,len=strlen(s);
    for(int i=0;i<len;i++) u=tr[u].ch[s[i]-'a'],tr[u].ans++;
}

signed main(){
    cin>>n;
    cnt=1;
    for(int i=1;i<=n;i++){
```

```
        cin>>S;
        insert(S,i);
    }
    getFail();
    cin>>T;
    query(T);
    topo();
    for(int i=1;i<=n;i++) printf("%d\n",vis[mp[i]]);
    return 0;
}
```

其他例题

文本生成器

给定n个单词和文章长度m,求包含至少一个单词的文章个数。

```
#include<bits/stdc++.h>
using namespace std;
const int N=1e4+5;
const int mod=1e4+7;

int ch[N][26],ed[N],fail[N];
int n,m,sz,dp[105][N],ans;
char s[N];
queue<int>q;

void insert(char s[]){  //this is the insert operation of AC
    int now=0,len=strlen(s);
    for(int i=0;i<len;++i){
        int x=s[i]-'A';
        if(!ch[now][x]) ch[now][x]=++sz;
        now=ch[now][x];
    }
    ed[now]|=1;
}

void build_fail(){ //fail pointer for all node
    for(int i=0;i<26;++i) if(ch[0][i]) q.emplace(ch[0][i]);
    while(q.size()){
        int fro=q.front();
        q.pop();
        for(int i=0;i<26;++i){
            if(!ch[fro][i]){
                ch[fro][i]=ch[fail[fro]][i];
                continue;
            }
            ed[ch[fro][i]]|=ed[ch[fail[fro]][i]];
            fail[ch[fro][i]]=ch[fail[fro]][i];
            q.emplace(ch[fro][i]);
        }
    }
}

signed main(){
```

```
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    cin>>n>>m;
    for(int i=1;i<=n;++i){
        cin>>s;
        insert(s);
    }
    build_fail();
    dp[0][0]=1;
    for(int i=1;i<=m;++i){ //for the length of i
        for(int j=0;j<=sz;++j){ //for all node in AC
            for(int k=0;k<26;++k){ //for all character
                if(!ed[ch[j][k]]) dp[i][ch[j][k]]=(dp[i][ch[j][k]]+dp[i-1]
[j])%mod; //dp in fail tree
            }
        }
    }
    for(int i=0;i<=sz;++i) ans=(ans+dp[m][i])%mod; //count the number of string
include specified string
    int sum=1;
    for(int i=1;i<=m;++i) sum=sum*26ll%mod; //total number of string if no
restriction
    cout<<(sum-ans+mod)%mod<<"\n";
    return 0;
}
```

参考资料

OI-wiki

https://www.luogu.com.cn/blog/juruohyfhaha/solution-p5357

https://www.luogu.com.cn/blog/juruohyfhaha/ac-zi-dong-ji

https://www.cnblogs.com/cjyyb/p/7196308.html

https://blog.csdn.net/bestsort/article/details/82947639