字符串 2

AC 自动机

算法简介

一种用于多模式串匹配的自动机，可以近似看成 Trie 树上的 KMP 算法。

算法例题

例题一

洛谷p5357

题意

给你一个文本串 S 和 n 个模式串 T_i，求每个模式串 T_i 在 S 中出现的次数。

题解

建立 AC 自动机后记录每个节点的访问次数，最后拓扑或建立 fail 树统计答案。时间复杂度 $O(|S|+\sum_{i=1}^n |T_i|)$。

```
const int MAXN=2e6+5,MAXS=2e5+5;
int idx[MAXN];
struct AC{
    int ch[MAXS][26],val[MAXS],fail[MAXS],cnt[MAXS],deg[MAXS],sz,tot;
    int ans[MAXS];
    int insert(char *s){
        int len=strlen(s),pos=0;
        _for(i,0,len){
            int c=s[i]-'a';
            if(!ch[pos][c]){
                ch[pos][c]=++sz;
                val[sz]=fail[sz]=0;
                mem(ch[sz],0);
            }
            pos=ch[pos][c];
        }
        if(!val[pos])return val[pos]=++tot;
        else return val[pos];
    }
    void getFail(){
```

```
        queue<int> q;
        _for(i,0,26){
            if(ch[0][i])
            q.push(ch[0][i]);
        }
        while(!q.empty()){
            int u=q.front();q.pop();
            _for(i,0,26){
                if(ch[u][i]){
                    deg[ch[fail[u]][i]]++;
                    fail[ch[u][i]]=ch[fail[u]][i];
                    q.push(ch[u][i]);
                }
                else ch[u][i]=ch[fail[u]][i];
            }
        }
    }
    void topu(){
        queue<int> q;
        _rep(i,1,sz){
            if(!deg[i]&&fail[i])
            q.push(i);
        }
        while(!q.empty()){
            int u=q.front();q.pop();
            if(fail[u]){
                cnt[fail[u]]+=cnt[u];
                deg[fail[u]]--;
                if(!deg[fail[u]])
                q.push(fail[u]);
            }
        }
    }
    void query(char *s){
        int len=strlen(s),pos=0;
        _for(i,0,len){
            pos=ch[pos][s[i]-'a'];
            cnt[pos]++;
        }
        topu();
        _rep(i,1,sz){
            if(val[i])
            ans[val[i]]=cnt[i];
        }
    }
}solver;
char buf[MAXN];
int main()
{
    int n=read_int();
```

```
    _rep(i,1,n){
        scanf("%s",buf);
        idx[i]=solver.insert(buf);
    }
    solver.getFail();
    scanf("%s",buf);
    solver.query(buf);
    _rep(i,1,n)
    enter(solver.ans[idx[i]]);
    return 0;
}
```