

AC自动机

```
#include <bits/stdc++.h>
using namespace std;
struct Tree{ //字典树
    int fail;//失配指针
    int vis[26];//子节点的位置
    int end;//标记有几个单词以这个节点结尾
}AC[1000000]; //Trie树
int cnt=0; //Trie的指针

inline void Build(string s){
    int l=s.length();
    int now=0; //字典树的当前指针
    for(int i=0;i<l;++i){ //构造Trie树
        if(!AC[now].vis[s[i]-'a']) AC[now].vis[s[i]-'a']=++cnt;
        now=AC[now].vis[s[i]-'a'];
    }
    AC[now].end+=1; //标记单词结尾
}

void Get_fail(){
    queue<int> Q;
    for(int i=0;i<26;++i){ //第二层的fail指针提前处理一下
        if(AC[0].vis[i]!=0){
            AC[AC[0].vis[i]].fail=0;
            Q.push(AC[0].vis[i]);
        }
    }
    while(!Q.empty()){
        int u=Q.front();
        Q.pop();
        for(int i=0;i<26;++i){
            if(AC[u].vis[i]!=0){
                AC[AC[u].vis[i]].fail=AC[AC[u].fail].vis[i];
                Q.push(AC[u].vis[i]);
            }
            else AC[u].vis[i]=AC[AC[u].fail].vis[i];
        }
    }
}

int AC_Query(string s){ //AC自动机匹配
    int len=s.length();
    int now=0,ans=0;
    for(int i=0;i<len;++i){
        now=AC[now].vis[s[i]-'a']; //向下一层
        for(int t=now;t&&AC[t].end!=-1;t=AC[t].fail){
            ans+=AC[t].end;
            AC[t].end=-1;
        }
    }
    return ans;
}
```

```
int main(){
    int n;
    string s;
    cin >> n;
    for(int i=1;i<=n;++i){
        cin >> s;
        Build(s);
    }
    AC[0].fail=0;//结束标志
    Get_fail();//求出失配指针
    cin >> s;//文本串
    cout << AC_Query(s) << endl;
    return 0;
}
```