AC自动机

```
#include<bits/stdc++.h>
using namespace std;
const int N=1e6+10,M=7e1+10,K=1.5e2+10;
map<int,int> mp;
struct trie {
    int fail;
    int c[26];
    set<int> exist;
}dic[M*K];
int cnt;
char str[N],tmp[K][M];

void ini(){
    mp.clear();
    for(int i=0;i<M*K;i++){
        dic[i].fail=0;
        for(int j=0;j<26;j++){
            dic[i].c[j]=0;
        }
        dic[i].exist.clear();
    }
    cnt=0;

}

void build(char str[],int x) {
    int len = strlen(str);
    int pos = 0;
    for (int i = 0; i < len; i++) {
        int c = str[i] - 'a';
        if (dic[pos].c[c] == 0) {
            dic[pos].c[c] = ++cnt;
        }
        pos = dic[pos].c[c];
    }
    dic[pos].exist.insert(x);
}

void get_fail() {
    queue<int> q;
    for (int i = 0; i < 26; i++) {
        if (dic[0].c[i] != 0) {
            q.push(dic[0].c[i]);
        }
    }
    while (!q.empty()) {
        int x = q.front();
        q.pop();
        int fafail = dic[x].fail;//父节点的fail指针
        for (int i = 0; i < 26; i++) {
            int y = dic[x].c[i];//指向的位置
            if (y != 0) {
                dic[y].fail = dic[fafail].c[i];
                q.push(y);
                for (auto &i:dic[dic[y].fail].exist) {
                    dic[y].exist.insert(i);
                }
            } else {
                dic[x].c[i] = dic[fafail].c[i];
```

```
                }
            }
        }
    }

    void AC_auto() {
        int len = strlen(str);
        int pos = 0;
        for (int i = 0; i < len; i++) {
            int x = str[i] - 'a';
            if (dic[pos].c[x] != 0) {
                pos = dic[pos].c[x];
                int size = dic[pos].exist.size();
                for (auto &i:dic[pos].exist) {
                    mp[i]++;
                }
            } else {
                while (dic[pos].c[x] == 0 && pos != 0) {
                    pos = dic[pos].fail;
                }
                if (pos != 0) {
                    pos = dic[pos].c[x];
                }
            }
        }
    }

    int main() {

        int n;
        while (cin >> n) {

            if(n==0){
                break;
            }
            ini();

            for (int i = 1; i <= n; i++) {
                cin >> tmp[i];
                build(tmp[i], i);
            }
            get_fail();

            cin >> str;
            AC_auto();
            int maxnum = 0;
            for (auto &i:mp) {
                maxnum = max(maxnum, i.second);
            }
            vector<int> vec;
            for (auto &i:mp) {
                if (i.second == maxnum) {
                    vec.push_back(i.first);
                }
            }
            cout << maxnum << endl;
            int size = vec.size();
            for (int i = 0; i < size; i++) {
                cout << tmp[vec[i]] << endl;
            }
        }

        return 0;
    }
```

拓扑优化

```
#include<bits/stdc++.h>
using namespace std;
const int N=2e6+10;
char str[N];
struct trie {
    int c[26];
    int fail;
    int flag;
    int ans;
}dic[N];
int cnt,mp[N],in[N];

void build(char str[],int x) {
    int len = strlen(str);
    int pos = 0;
    for (int i = 0; i < len; i++) {
        int v = str[i] - 'a';
        if (dic[pos].c[v] == 0) {
            dic[pos].c[v] = ++cnt;
        }
        pos = dic[pos].c[v];
    }
    dic[pos].flag = x;
    mp[x] = pos;
}

void get_fail() {
    queue<int> q;
    for (int i = 0; i < 26; i++) {
        if (dic[0].c[i] != 0) {
            q.push(dic[0].c[i]);
        }
    }
    while (!q.empty()) {
        int x = q.front();
        q.pop();
        int fafail = dic[x].fail;
        for (int i = 0; i < 26; i++) {
            int v = dic[x].c[i];
            if (v == 0) {
                dic[x].c[i] = dic[fafail].c[i];
                continue;
            }
            dic[v].fail = dic[fafail].c[i];
            in[dic[v].fail]++;
            q.push(v);
        }
    }
}

void query(char str[]) {
    int len = strlen(str);
    int pos = 0;
    for (int i = 0; i < len; i++) {
        int v = str[i] - 'a';
        pos = dic[pos].c[v];
        dic[pos].ans++;
    }
}

void topo() {
    queue<int> q;
    for (int i = 1; i <= cnt; i++) {
```

```
            if (in[i] == 0) {
                q.push(i);
            }
        }
        while (!q.empty()) {
            int x = q.front();
            q.pop();
            int v = dic[x].fail;
            dic[v].ans += dic[x].ans;
            in[v]--;
            if (in[v] == 0) {
                q.push(v);
            }
        }
    }

    int main() {

        int n;
        scanf("%d", &n);
        for (int i = 1; i <= n; i++) {
            scanf("%s", str);
            build(str, i);
        }
        get_fail();
        scanf("%s", str);
        query(str);
        topo();

        for (int i = 1; i <= n; i++) {
            printf("%d\n", dic[mp[i]].ans);
        }

        return 0;
    }
```