poj2001

```
#include<bits/stdc++.h>
int tot, trie[6000][26], vis[6000];
char word[1002][23];
int insert(char str[], int rt) {
    int len = strlen(str);
    for (int i = 0; i < len; i++) {
        int x = str[i] - 'a';
        if (trie[rt][x] == 0) {
            trie[rt][x] = ++tot;
            memset(trie[trie[rt][x]], 0, sizeof(trie[trie[rt][x]]));
        }
        rt = trie[rt][x];
        vis[rt]++;
    }
}
void finds(char str[], int rt) {
    int len = strlen(str);
    for (int i = 0; i < len; i++) {
        int x = str[i] - 'a';
        rt = trie[rt][x];
        printf("%c", str[i]);
        if (vis[rt] == 1) {
            printf("\n");
            return;
        }
        if (i == len - 1) {
            printf("\n");
            return;
        }
    }
}
int main() {
    int num = 0;
    tot = 0;
    int rt = ++tot;
    memset(trie[rt], 0, sizeof(trie[rt]));
    memset(vis, 0, sizeof(vis));
    while (scanf("%s", word[++num]) != EOF)
        insert(word[num], rt);
    for (int i = 1; i <= num; i++) {
        printf("%s ", word[i]);
        finds(word[i], rt);
    }
}
```

hdu1251

```
#include<bits/stdc++.h>
#define ll long long
#define Sca(a) scanf("%d",&a)
#define Scall(a) scanf("%lld",&a)
#define Pri(a) printf("%d",&a)
```

```
#define Prill(a) printf("%lld",&a)
#define FAST_IO ios::sync_with_stdio(false)
using namespace std;
const int INF = 0x3f3f3f3f;
const ll INFL = 0x3f3f3f3f3f3f3f3f;
using namespace std;

int trie[1000010][26];    //数组形式定义字典树，值存储的是下一个字符的位置
int num[1000010] = {0};  //附加值，以某一字符串为前缀的单词的数量
int pos = 1;

void Insert(char word[]) {  //在字典树中插入某个单词
    int i;
    int c = 0;
    for (i = 0; word[i]; i++) {
        int n = word[i] - 'a';
        if (trie[c][n] == 0) //如果对应字符还没有值
            trie[c][n] = pos++;
        c = trie[c][n];
        num[c]++;
    }
}
int Find(char word[]) {  //返回以某个字符串为前缀的单词的数量
    int i;
    int c = 0;
    for (i = 0; word[i]; i++) {
        int n = word[i] - 'a';
        if (trie[c][n] == 0)
            return 0;
        c = trie[c][n];
    }
    return num[c];
}

int main() {
    char word[11];
    while (gets(word)) {
        if (word[0] == NULL) //空行。gets读入的回车符会自动转换为NULL。
            break;
        Insert(word);
    }
    while (gets(word))
        printf("%d\n", Find(word));
    return 0;
}
```

HDU 4825

```
#include<bits/stdc++.h>
using namespace std;
#define MAXN 100005
#define ll long long
int n,m;
int trie[32*MAXN][2];
ll val[32*MAXN];
int tot;
void insert(ll d)
```

```
{
    int root=0;
    for(int i=32;i>=0;i--)
    {
        int id=(d>>i)&1;//获得这一个bit位的值
        if(!trie[root][id]) trie[root][id]=++tot;
        root=trie[root][id];
    }
    val[root]=d;
}
ll query(ll d)
{
    int root=0;
    for(int i=32;i>=0;i--)
    {
        int id=(d>>i)&1;
        if(trie[root][id^1]) root=trie[root][id^1];
        else root=trie[root][id];
    }
    return val[root];
}
int main()
{
    int cas;
    scanf("%d",&cas);
    for(int tt=1;tt<=cas;tt++)
    {
        scanf("%d%d",&n,&m);
        memset(trie,0,sizeof trie);
        for(int i=0;i<n;i++)
        {
            int d;
            scanf("%d",&d);
            insert(d);
        }
        printf("Case #%d:\n",tt);
        for(int i=0;i<m;i++)
        {
            ll d;
            scanf("%lld",&d);
            printf("%lld\n",query(d));
        }
    }
    return 0;
}
```

zoj2339

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<stack>
#include<cstdio>
#include<string>
#include<cstring>
#include<vector>
// #include<map>
```

```
#include<cmath>
#include<list>
#define ll long long
#define N 100010
#define pb push_back
#define Sca(a) scanf("%d",&a)
#define mem0(a) memset(a,0,sizeof(a))
#define mem1(a) memset(a,-1,sizeof(a))
#define Scal(a) scanf("%ld",&a)
#define Scall(a) scanf("%lld",&a)
#define Pri(a) printf("%d",a)
#define Pril(a) printf("%ld",a)
#define Prill(a) printf("%lld",a)
#define FAST_IO ios::sync_with_stdio(false)
using namespace std;
const int INF = 0x3f3f3f3f;
const ll INFL = 0x3f3f3f3f3f3f3f3f;
using namespace std;
template <class T>void tomax(T&a, T b) {
    a = max(a, b);
}
template <class T>void tomin(T&a, T b) {
    a = min(a, b);
}
int main(){
    int T;
    cin>>T;
    priority_queue<ll,vector<ll>,greater<ll> >q;
    while(T--){
        int n;
        Sca(n);
        for(int i=0;i<n;i++){
            int s;
            Sca(s);
            q.push(s);
        }
        ll ans=0,x,y;
        while(!q.empty()){
            x=q.top();
            q.pop();
            if(q.empty()) break;
            y=q.top(),q.pop();
            ans+=x+y;
            q.push(x+y);
        }
        Prill(ans);
        printf("\n");
        if(T!=0) printf("\n");
    }
}
```

hdu2222

```
#include <map>
```

```cpp
#include <set>
#include <list>
#include <queue>
#include <deque>
#include <stack>
#include <string>
#include <time.h>
#include <cstdio>
#include <math.h>
#include <iomanip>
#include <cstdlib>
#include <limits.h>
#include <string.h>
#include <iostream>
#include <fstream>
#include <algorithm>
using namespace std;

#define LL long long
#define MIN INT_MIN
#define MAX INT_MAX
#define PI acos(-1.0)
#define FRE freopen("input.txt","r",stdin)
#define FF freopen("output.txt","w",stdout)
#define N 10005
#define M 1000005
const int kind = 26;
struct node {
    node *fail;//失败指针
    node *next[kind];//Trie每个节点的子节点（最多26个子母）
    int cnt;//是否为该单词的最后一个节点
    node () {//构造函数初始化
        fail = NULL;
        cnt = 0;
        memset(next,NULL,sizeof(next));
    };
};
char str[55];//输入的单词
char ss[M];//输入的模式串
queue<node*> qq;//队列构造失败指针
//建Trie树
void BuildTree (node *root) {
    node *p = root;
    int i = 0,id;
    while (str[i]) {
        id = str[i] - 'a';
        if (p->next[id] == NULL) {
            p->next[id] = new node();
        }
        p = p->next[id];
        i++;
    }
    (p->cnt)++;
}
//bfs求失败指针
void bfs(node *root) {
    while (!qq.empty()) qq.pop();
    int i;
```

```
    root->fail = NULL;
    qq.push(root);root->fail = NULL;
    node *tmp,*p;
    while (!qq.empty()) {
        tmp = qq.front();
        qq.pop();
        p = NULL;
        for (i = 0; i < 26; i++) {
            if (tmp->next[i] != NULL) {
                p = tmp->fail;
                while (p && !p->next[i]) {
                    p = p->fail;
                }
                if (!p) tmp->next[i]->fail = root;
                else tmp->next[i]->fail = p->next[i];
                qq.push(tmp->next[i]);
            }
        }
    }
}
//AC自动机主程序
int AC_run(node *root) {
    int i = 0,ans = 0,id;
    node *p = root;
    while (ss[i]) {
        id = ss[i] - 'a';
        while (!p->next[id] && p != root) {
            p = p->fail;
        }
        p = p->next[id];
        if (!p) p = root;
        node *tmp = p;
        while (tmp != root && tmp->cnt != -1) {
            ans += tmp->cnt;
            tmp->cnt = -1;
            tmp = tmp->fail;
        }
        i++;
    }
    return ans;
}
int main () {
    int t;
    scanf("%d",&t);
    while (t--) {
        int n;
        scanf("%d",&n);
        node *root = new node();
        while (n--) {
            scanf("%s",str);
            BuildTree(root);
        }
        bfs(root);
        scanf("%s",ss);
        printf("%d\n",AC_run(root));
    }
    return 0;
}
```