Sectumsempra补充模板——zfg部分

目录

1. CRT(非互质)
2. CRT
3. lct
4. lucas定理
5. manacher
6. palindromic_tree
7. treap
8. VirtulTree
9. 后缀数组
10. 后缀自动机（非字典树）
11. 后缀自动机（含字典树）
12. 可持久化并查集
13. 扩展kmp
14. 全局最小割
15. 最小表示法
16. 左偏树 & 斜堆

CRT(非互质)

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<vector>
#define rep(i, l, r) for(int i = l; i <= r; ++i)
using namespace std;
typedef long long ll;
void ExGcd(ll a, ll b, ll &d, ll &x, ll &y){
        if(!b){d = a; x = 1; y = 0;}
        else {
            ExGcd(b, a % b, d ,y, x);
            y -= a/b * x;
        }
    }
struct ModularEquation{
    ll m, r;
    ModularEquation(){}
    ModularEquation(ll m, ll r):m(m), r(r){}
    pair<bool, ModularEquation> operator + (const ModularEquation &t) const{
        ll c = t.r - r;
        ll d, x, y;
        ExGcd(m, t.m, d, x, y);
        if(c % d) return make_pair(0, ModularEquation(0, 0));
        c /= d;
        ll NewM = m /d * t.m, NewR = r + m * x * c;
        NewR %= NewM;
        if(NewR <= 0) NewR += NewM;
        return make_pair(1, ModularEquation(NewM, NewR));
    }
};
vector<ModularEquation> t;
int main(){
    int cas;
    cin >> cas;
    while(cas--){
        ll N;
```

```
        int m;
        cin >> N >> m;
        t.clear();
        rep(i, 1, m) t.push_back(ModularEquation(0, 0));
        for(int i = 0; i < m; ++i) scanf("%I64d", &t[i].m);
        for(int i = 0; i < m; ++i) scanf("%I64d", &t[i].r);
        bool flag = 1;
        for(int i = 1; i < m; ++i){
            pair<bool, ModularEquation> bm = t[i - 1] + t[i];
            if(!bm.first){
                flag = 0;
                break;
            }
            t[i] = bm.second;
        }
        if(!flag){
            printf("0\n");
            continue;
        }
        ModularEquation ans = t[m - 1];
        ans.r %= ans.m;
        if(ans.r <= 0) ans.r += ans.m;
        if(N < ans.r) printf("0\n");
        else cout <<  1 + (N - ans.r) / ans.m << "\n";
    }
    return 0;
}
```

CRT

```
#include<iostream>
#include<cstdio>
#include<cstring>
#define rep(i, l, r) for(int i = l; i <= r; ++i)
using namespace std;
typedef long long ll;
const int maxn = 1e5 + 10;
class CRT{
    private:
        int p[maxn], v[maxn], m;
        ll M, n;
    public:
        void ExGcd(ll a, ll b, ll &x, ll &y){
            if(!b){x = 1; y = 0;}
            else {
                ExGcd(b, a % b, y, x);
                y -= x*(a/b);
            }
        }
        void work(){
            cin >> n >> m;
            rep(i, 1, m) scanf("%d", &p[i]);
            rep(i, 1, m) scanf("%d", &v[i]);
            M = 1;
            rep(i, 1, m) M*= p[i];
            ll t = 0;
            printf("M = %lld\n", M);
            rep(i, 1, m){
                int k = M / p[i];
                ll AnsX, AnsY;
                ExGcd(k, p[i], AnsX, AnsY);
                printf("k = %d p[%d] = %d AnsX = %d\n", k, i,p[i], AnsX);
                t += v[i] * AnsX * k;
            }
            printf("t = %lld M = %lld\n", t, M);
```

```
                if(t <= 0)
                t = (t % M) + M;
            cout << (n - t) / M + 1 << "\n";
        }
}solver;
int main(){
    int cas;
    cin >> cas;
    while(cas--)
        solver.work();
    return 0;
}
```

lct

```
// hdu 4010
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<cmath>
#define maxn 300101
using namespace std;
int ch[maxn][2],fa[maxn],w[maxn],val[maxn],flag[maxn],n,m,last,sign[maxn];
    bool root(int x){
        return ch[fa[x]][0] != x && ch[fa[x]][1] != x;
    }
    void maintain(int x){
        if(!x) return ;
        w[x] = val[x];
        if(ch[x][0] && w[ch[x][0]] > w[x]) w[x] = w[ch[x][0]];
        if(ch[x][1] && w[ch[x][1]] > w[x]) w[x] = w[ch[x][1]];
    }
    void pushdown(int x){
        if(!x) return ;
        if(flag[x]){
            flag[ch[x][0]] ^= 1;
            flag[ch[x][1]] ^= 1;
            swap(ch[x][0], ch[x][1]);
            flag[x] = 0;
        }
        if(sign[x]){
            sign[ch[x][0]] += sign[x]; w[ch[x][0]] += sign[x]; val[ch[x][0]] += sign[x];
            sign[ch[x][1]] += sign[x]; w[ch[x][1]] += sign[x]; val[ch[x][1]] += sign[x];
            sign[x] = 0;
        }
    }
    void rotate(int x, bool d){
        int k = ch[x][d ^ 1];
        fa[ch[x][d ^ 1] = ch[k][d]] = x;
        ch[k][d] = x;
        fa[k] = fa[x];
        if(ch[fa[x]][0] == x) ch[fa[x]][0] = k;
        else if(ch[fa[x]][1] == x) ch[fa[x]][1] = k;
        fa[x] = k ;
        maintain(x);
        maintain(k);
    }
    void splay(int x){
        pushdown(x);
        while(!root(x)){
            int y = fa[x], z = fa[y]; pushdown(z); pushdown(y);pushdown(x);
            bool d1 = x == ch[y][0], d2 = y == ch[z][0];
            if(root(y)) rotate(y, d1);
            else if(d1 == d2) rotate(z, d2), rotate(y, d1);
```

```
            else rotate(y, d1 ), rotate(z, d2);
        }
    }
    void access(int x){
        for(last = 0; x; last = x, x = fa[x] ){
            splay(x);
            ch[x][1] = last;
            maintain(x);
        }
    }
    void makeroot(int x){
        access(x);
        splay(x);
        flag[x] ^= 1;
    }
    void link(int x, int y){
        makeroot(x);
        fa[x] = y;
        access(x);
    }
    void cut(int x){
        access(x);
        splay(x);
        fa[ch[x][0]] = 0;
        ch[x][0] = 0;
        maintain(x);
    }
    int getroot(int x){
        access(x);
        splay(x);
        int p = x;
        for(; ch[p][0]; p = ch[p][0] ) pushdown(p);
        return p;
    }
    int lca(int x, int y){
        access(x);
        access(y);
        return last;
    }
struct edge
{
    int l,r;
}e[maxn];
void work(int x,int v)
{
    val[x]+=v;
    w[ch[x][1]]+=v;val[ch[x][1]]+=v;sign[ch[x][1]]+=v;
    maintain(x);
}
void illegal()
{
    printf("-1\n");
}
int main()
{
    while(cin>>n){
        memset(fa,0,sizeof(fa));memset(ch,0,sizeof(ch));memset(flag,0,sizeof(flag));memset(sign,0,sizeof(sign));
        for(int i=1;i<n;i++)scanf("%d%d",&e[i].l,&e[i].r);
        for(int i=1;i<=n;i++)scanf("%d",&val[i]);
        for(int i=1;i<n;i++)link(e[i].l,e[i].r);
        cin>>m;
        for(int i=1;i<=m;i++){
            int opt,v1,v2,v3;scanf("%d%d%d",&opt,&v1,&v2);
            if(opt==1){
                if(getroot(v1)==getroot(v2))illegal();
                else link(v1,v2);
            }else
```

```
            if(opt==2){
                if(v1==v2||getroot(v1)!=getroot(v2))illegal();
                else {makeroot(v1);cut(v2);}
            }else
            if(opt==3){
                scanf("%d",&v3);
                if(getroot(v2)!=getroot(v3)){
                    illegal();
                    continue;
                }
                int k=lca(v2,v3);
                access(v2);splay(k);
                work(k,v1);
                access(v3);splay(k);
                work(k,v1);
                val[k]-=v1;maintain(k);
            }
            else {
                if(getroot(v1)!=getroot(v2)){
                    illegal();
                    continue;
                }
                int k=lca(v1,v2),ans=0;
                access(v1);splay(k);
                if(ch[k][1])ans=max(ans,max(val[k],w[ch[k][1]]));
                else ans=val[k];
                access(v2);splay(k);
                if(ch[k][1])ans=max(ans,max(val[k],w[ch[k][1]]));
                printf("%d\n",ans);
            }
        }
        printf("\n");
    }
    return 0;
}
```

lucas定理

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<vector>
#define rep(i, l, r) for(int i = l; i <= r; ++i)
using namespace std;
typedef long long ll;
struct data{
    vector<int>v;
    int & operator [](const int x){
        while(x >= v.size())v.push_back(0);
        return v[x];
    }
}fac;
int n, m, p;
void prepare(){
    fac[0] = 1;
    rep(i, 1, p)fac[i] = (ll)fac[i - 1] * i % p;
}
int qp(int a, int b){
    int ret = 1, tmp = a;
    for(; b; b >>= 1){
        if(b & 1)ret = (ll) ret * tmp % p;
        tmp = (ll)tmp * tmp  % p;
    }
    return ret;
}
```

```
int C(int a, int b){
    if(a < b)return 0;
    return (ll)fac[a] * qp((ll)fac[b] * fac[a - b] % p, p - 2) % p;
}
int lucas(int a, int b){
    return b == 0 ? 1 : (ll) C(a % p, b % p) * lucas(a / p, b / p) % p;
}
int main(){
    int cas;
    cin >> cas;
    while(cas--){
        cin >> n >> m >> p;
        prepare();
        printf("%d\n", lucas(n + m, n));
    }
    return 0;
}
```

manacher

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
#define rep(i, l, r) for(int i = l; i <= r; ++i)
using namespace std;
const int maxn =  240101;
class manacher{
    // hdu 3068
    private:
        char s[maxn];
        int len, n,cnt[maxn], f[maxn];
    public:
        int solve(char *str){
            n = strlen(str + 1);
            cnt[0] = cnt[1] = 0;
            s[len = 1] = -1;
            rep(i, 1, n){
                s[++len] = 0;
                s[++len] = str[i];
            }
            s[++len] = -2;
            memset(f, 0, sizeof(int) * (len + 2));
            int p = 0;
            f[1] = f[len] = 1;
            for(int i = 2; i < len; ++i){
                cnt[i] = cnt[i - 1] + (s[i] > 0);
                if(p + f[p] - 1 < i)
                    for(p = i; s[p - f[p]] == s[p + f[p]]; ) ++f[p];
                else {
                    int k = 2 * p - i;
                    f[i] = min(f[k], k - (p - f[p]));
                    for(; s[i - f[i]] == s[i + f[i]]; ) ++f[i];
                    if(i + f[i] > p + f[p]) p = i;
                }
            }
            int ret = 0;
            for(int i = 2; i < len; ++i)
                ret = max(ret, cnt[i + f[i] - 1] - cnt[i - f[i]]);
            return ret;
        }
}solver;
char s[maxn];
int main(){
    while(scanf("%s",s + 1) != EOF)
```

```
        printf("%d\n", solver.solve(s));
    return 0;
}
```

palindromic_tree

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<string>
#include<cstdlib>
#define rep(i,l,r) for(int i=l;i<=r;++i)
using namespace std;
const int maxn=10001;
struct palindromic_tree{
    int ch[maxn][26],len[maxn],pre[maxn],tot,last,s[maxn],maxlen,cnt[maxn];
    palindromic_tree(){
        tot=1;
        len[0]=0;
        len[1]=-1;
        pre[0]=1;
        s[0]=-1;
        //±ØÐë  0ºÅ¼ÚµãÎª³¤¶ÈÎª0µÄ¼ÚµãÎª³¤¶ÈÎª-1µÄ¼ÚµãÎª³¤¶ÈÎª¬ÈôÒª»»»£¬ÔÒò±ØÐë¸ø¸ch[0][i]¸³ÖµÎª1
    }
    int match(int pos){
        while(s[maxlen-len[pos]-1]!=s[maxlen])pos=pre[pos];
        return pos;
    }
    void add(int c){
        s[++maxlen]=c;
        int now=match(last);
        if(!ch[now][c]){
            ++tot;
            pre[tot]=ch[match(pre[now])][c];
            //preºÍch¸³ÖµÓï¾ä²»¿É¼»»»£¬·ñÔò»á¹ò
            ch[now][c]=tot;
            len[tot]=len[now]+2;

        }
        last=ch[now][c];
        //printf("last=%d len=%d now=%d pre=%d\n",last,len[last],now,pre[tot]);
        cnt[last]++;
    }
    void out(){
        rep(i,0,tot)printf("i=%d len=%d pre=%d cnt=%d\n",i,len[i],pre[i],cnt[i]);
    }
    void count(){
        for(int i=tot;i>1;i--)cnt[pre[i]]+=cnt[i];
    }
}t;
string s;
int main(){
    cin>>s;
    rep(i,0,s.size()-1)t.add(s[i]-'a');
    t.out();
    return 0;
}
```

treap

```
#include<iostream>
#include<cstdio>
#include<cstring>
```

```
#include<algorithm>
class Treap{
    int w[maxn], ln[maxn], rn[maxn], size[maxn];
    void maintain(int x){
        size[x] = size[ln[x]] + size[rn[x]] + 1;
    }
    int merge(int x, int y){
        if(!x || !y) return x | y;
        if(rank() % (size[x] + size[y]) < size[x] ){
            rn[x] = merge(rn[x], y);
            maintain(x);
        }
        else {
            ln[y] = merge(x, ln[y]);
            maintain(y);
        }
    }
    pair<int, int> split(int x, int rank) {
        if(!x || !rank) return make_pair(0, x);
        pair<int, int> p;
        if(rank <= size[ln[x]]){
            p = split(ln[x], rank);
            ln[x] = p.second;
            p = make_pair(p.first, x);
            maintain(x);
        }
        else {
            p = split(rn[x], rank - size[ln[x]] - 1)£»
            rn[x] = p.first;
            p = make_pair(x, p.second);
            maintain(x);
        }
        return p;
    }
};
```

VirtulTree

```
/*
bzoj 3572
*/
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<vector>
#define rep(i, l, r) for(int i = l; i <= r; ++i)
using namespace std;
namespace VirtulTree{
    const int logn = 18;
    const int maxn = 3e5 + 10;
    int n, m, fa[maxn][logn + 1], deep[maxn], times, dfn[maxn], stk[maxn], top, rdfn[maxn];
    int rt[maxn], d[maxn], size[maxn], ans[maxn];
    bool mark[maxn];
    vector<int> g[maxn], mp, vt[maxn], OutOrder;
    inline void add(int x, int y){
        g[x].push_back(y);
    }
    void dfs(int x){
        dfn[x] = ++times;
        size[x] = 1;
        for(int i = 0; i < g[x].size(); ++i)
            if(fa[x][0] != g[x][i]){
                fa[g[x][i]][0] = x;
                deep[g[x][i]] = deep[x] + 1;
```

```
            dfs(g[x][i]);
            size[x] += size[g[x][i]];
        }
    rdfn[x] = times;
}
void init(){
    cin >> n;
    rep(i, 2, n){
        int x, y;
        scanf("%d%d", &x, &y);
        add(x, y);
        add(y, x);
    }
    dfs(1);
    rep(i, 1, logn)
        rep(j, 1, n)
            fa[j][i] = fa[fa[j][i - 1]][i - 1];
    cin >> m;
}
bool cmp(int a, int b){
    return dfn[a] < dfn[b];
}
int lca(int x, int y){

    if(deep[x] < deep[y]) swap(x, y);
    int delta = deep[x] - deep[y];
    for(int i = logn; i >= 0; --i)
        if((delta >> i) & 1) x = fa[x][i];
    if(x == y) return x;
    for(int i = logn; i >= 0; --i)
        if(fa[x][i] != fa[y][i]) x = fa[x][i], y = fa[y][i];
    return fa[x][0];
}
void clear(){
    for(int i = 0; i < mp.size(); ++i){
         mark[mp[i]] = 0;
         d[mp[i]] = n + 1;
         vt[mp[i]].clear();
    }
    for(int i = 0; i < OutOrder.size(); ++i)
        ans[OutOrder[i]] = 0;
    OutOrder.clear();
    mp.clear();
    top = 0;
}
int getpos(int x, int step){
        for(int i = logn; i >= 0; --i)
            if((step >> i) & 1) x = fa[x][i];
        return x;
    }
void dfs2(int x){
    ans[rt[x]] += size[x];
    for(int i = 0; i < vt[x].size(); ++i){
        int k = vt[x][i];
        dfs2(k);
        if(rt[k] != rt[x]){
                int len = (deep[k] - deep[x] - 1) - (d[k] - d[x]);
                int p1, p2;
                if(len & 1)  {
                    if(rt[k] < rt[x])
                        p1 = getpos(k, (len >> 1) + 1);
                    else
                        p1 = getpos(k, (len >> 1));
                }
            else
                p1 = getpos(k, len >> 1);
            ans[rt[k]] += size[p1] - size[k];
```

```
            ans[rt[x]] -= size[p1];
        }
        else ans[rt[x]] -= size[k];
    }
}
void getdist(int x){
    for(int i = 0; i < vt[x].size(); ++i){
        int k = vt[x][i];
        if(d[k] > d[x] + deep[k] - deep[x] || (d[k] == d[x] + deep[k] - deep[x] && rt[x] < rt[k])){
            d[k] = d[x] + deep[k] - deep[x];
            rt[k] = rt[x];
        }
        getdist(k);
        if(d[x] > d[k] + deep[k] - deep[x] || (d[x] == d[k] + deep[k] - deep[x] && rt[k] < rt[x])){
            d[x] = d[k] + deep[k] - deep[x];
            rt[x] = rt[k];
        }
    }
}
void DFS(){
    int root;
    for(int i = 0; i < mp.size(); ++i){
        int now = mp[i];
        if(!top) stk[++top] = root = mp[i], root = now;
        else {
            while(!(dfn[stk[top]] <= dfn[now] && dfn[now] <= rdfn[stk[top]])) --top;
            int last = stk[top], delta = deep[now] - deep[last];
            vt[last].push_back(now);
            stk[++top] = now;
        }
    }
    getdist(root);
    getdist(root);
    dfs2(root);
    if(root != 1) ans[rt[root]] += size[1] - size[root];
}
void solve(){
    rep(i, 1, m){
        int k, x;
        scanf("%d", &k);
        clear();
        rep(j, 1, k){
            scanf("%d", &x);
            mp.push_back(x);
            OutOrder.push_back(x);
            mark[rt[x] = x] = 1;
            d[x] = 0;
        }
        sort(mp.begin(), mp.end(), cmp);
        int tmp = mp.size();
        for(int i = 1; i < tmp; ++i){
            int last = mp[i - 1], now = mp[i];
            int LCA = lca(last, now);
            if(!mark[LCA]){
                mark[LCA] = 1;
                d[LCA] =  n + 1;
                mp.push_back(LCA);
            }
        }
        sort(mp.begin(), mp.end(), cmp);
        DFS();
        for(int i = 0; i < OutOrder.size(); ++i)
            printf("%d ", ans[OutOrder[i]]);
        printf("\n");
    }
}
void work(){
```

```
        clear();
        init();
        solve();
    }
}
int main(){
    VirtulTree::work();
    return 0;
}
```

后缀数组

```
#include<iostream>
#include<cstdio>
#include<cstring>
#define rep(i, l, r) for(int i = l; i <= r; ++i)
using namespace std;
const int maxn = 1e5 + 10;
class solver{
    private:
        char s[maxn], MaxChar, MinChar;
        int sa[maxn], rank[maxn], n, assist[maxn], cnt[maxn], height[maxn];
    public:
        void GetSa(int m){
            int *x = rank, *y = assist;
            rep(i, 1, m) cnt[i] = 0;
            rep(i, 1, n) cnt[x[i] = s[i] - MinChar + 1]++;
            rep(i, 1, m) cnt[i] += cnt[i - 1];
            for(int i = n; i >= 1; --i) sa[cnt[x[i]]--] = i;
            for(int k = 1; k <= n; k <<= 1){
                int p = 0;
                rep(i, n - k + 1, n) y[++p] = i;
                rep(i, 1, n) if(sa[i] - k >= 1) y[++p] = sa[i] - k;
                rep(i, 1, m) cnt[i] = 0;
                rep(i, 1, n) cnt[x[i]]++;
                rep(i, 1, m) cnt[i] += cnt[i - 1];
                for(int i = n; i >= 1; --i) sa[cnt[x[y[i]]]--] = y[i];
                p = 1;
                swap(x, y);
                x[sa[1]] = 1;
                rep(i, 2, n)
                    x[sa[i]] = y[sa[i]] == y[sa[i - 1]] && y[sa[i] + k] == y[sa[i - 1] + k] ? p : ++p;
                if(p >= n)break;
                m = p;
            }
            rep(i, 1, n) rank[sa[i]] = i;
        }
        void GetHeight(){
            int p = 0;
            rep(i, 1, n){
                if(p) --p;
                if(rank[i] != 1) while(s[i + p] == s[sa[rank[i] - 1] + p]) ++p;
                height[rank[i]] = p;
            }
        }
        void init(){
            scanf("%s", s + 1);
            n = strlen(s + 1);
        }
        void solve(){
            MaxChar = s[1], MinChar = s[1];
            rep(i, 1, n)
                MaxChar = max(MaxChar, s[i]),
                MinChar = min(MinChar, s[i]);
            GetSa(MaxChar - MinChar + 1);
```

```
        }
}Solver;
```

后缀自动机（非字典树）

```
#include<iostream>
#include<cstdio>
#include<cstring>
#define rep(i, l, r) for(int i = l; i <= r; ++i)
using namespace std;
int MinLen;
const int maxn = 2e5 + 20;
char s[maxn];
int pre[maxn], step[maxn], ch[maxn][52], last, cnt[maxn], g[maxn], cnt2[maxn], tot;
int order[maxn];
/* ×ÖµäÊ÷
void insert(int k){
    int p=last;
    if(ch[p][k]){
        if(step[p]+1==step[ch[p][k]])last=ch[p][k];
        else {
            int q=ch[p][k],nq=++tot;
            memcpy(ch[nq],ch[q],sizeof(ch[nq]));
            step[nq]=step[p]+1;
            pre[nq]=pre[q];
            pre[q]=nq;
            last=nq;
            for(;p&&ch[p][k]==q;p=pre[p])ch[p][k]=nq;
            if(!p&&ch[p][k]==q)ch[p][k]=nq;
        }
        return ;
    }
    int np=++tot;
    last=np;
    step[np]=step[p]+1;
    for(;p&&!ch[p][k];p=pre[p])ch[p][k]=np;
    if(!p&&!ch[p][k])ch[p][k]=np,pre[np]=p;
    else if(step[p]+1==step[ch[p][k]])pre[np]=ch[p][k];
         else {
            int q=ch[p][k],nq=++tot;
            memcpy(ch[nq],ch[q],sizeof(ch[nq]));
            step[nq]=step[p]+1;
            pre[nq]=pre[q];
            pre[q]=pre[np]=nq;
            for(;p&&ch[p][k]==q;p=pre[p])ch[p][k]=nq;
            if(!p&&ch[p][k]==q)ch[p][k]=nq;
         }
}
*/
void add(int k){
    int p = last, np = ++tot;
    step[np] = step[p] + 1;
    last = np;
    for(; p && !ch[p][k]; p = pre[p]) ch[p][k] = np;
    if(!p && !ch[p][k]) ch[p][k] = np, pre[np] = p;
    else if(step[ch[p][k]] == step[p] + 1) pre[np] = ch[p][k];
    else {
        int q = ch[p][k], nq = ++tot;
        memcpy(ch[nq], ch[q], sizeof(ch[q]));
        step[nq] = step[p] + 1;
        pre[nq] = pre[q];
        pre[q] = pre[np] = nq;
        for(; p && ch[p][k] == q; p = pre[p]) ch[p][k] = nq;
        if(!p && ch[p][k] == q) ch[p][k] = nq;
    }
```

```
    ++cnt[np];
}
void clear(){
    rep(i, 0, tot) memset(ch[i], 0, sizeof(ch[i]));
    rep(i, 0, tot) step[i] = pre[i] = cnt[i] = g[i] = cnt2[i] = 0;
    last = tot = 0;
}
void RadixSort(){
    rep(i, 1, tot) g[step[i]]++;
    rep(i, 1, n) g[i] += g[i - 1];
    rep(i, 1, tot) order[g[step[i]]--] = i;
}
int main(){
    while(cin >> MinLen && MinLen){
        scanf("%s", s + 1);
        int n = strlen(s + 1);
        clear();
        rep(i, 1, n)
            add( s[i] <= 'z' && s[i] >= 'a' ? s[i] - 'a' : s[i] - 'A' + 26);
    }
    return 0;
}
```

后缀自动机（含字典树）

```
#include<iostream>
#include<cstdio>
#include<cstring>
#define rep(i, l, r) for(int i = l; i <= r; ++i)
using namespace std;
int MinLen;
const int maxn = 2e5 + 20;
char s[maxn];
int pre[maxn], step[maxn], ch[maxn][52], last, cnt[maxn], g[maxn], cnt2[maxn], tot;
int order[maxn];
/* ×ÖµäÊ÷
void insert(int k){
    int p=last;
    if(ch[p][k]){
        if(step[p]+1==step[ch[p][k]])last=ch[p][k];
        else {
            int q=ch[p][k],nq=++tot;
            memcpy(ch[nq],ch[q],sizeof(ch[nq]));
            step[nq]=step[p]+1;
            pre[nq]=pre[q];
            pre[q]=nq;
            last=nq;
            for(;p&&ch[p][k]==q;p=pre[p])ch[p][k]=nq;
            if(!p&&ch[p][k]==q)ch[p][k]=nq;
        }
        return ;
    }
    int np=++tot;
    last=np;
    step[np]=step[p]+1;
    for(;p&&!ch[p][k];p=pre[p])ch[p][k]=np;
    if(!p&&!ch[p][k])ch[p][k]=np,pre[np]=p;
    else if(step[p]+1==step[ch[p][k]])pre[np]=ch[p][k];
         else {
            int q=ch[p][k],nq=++tot;
            memcpy(ch[nq],ch[q],sizeof(ch[nq]));
            step[nq]=step[p]+1;
            pre[nq]=pre[q];
            pre[q]=pre[np]=nq;
            for(;p&&ch[p][k]==q;p=pre[p])ch[p][k]=nq;
```

```
            if(!p&&ch[p][k]==q)ch[p][k]=nq;
         }
}
*/
void add(int k){
    int p = last, np = ++tot;
    step[np] = step[p] + 1;
    last = np;
    for(; p && !ch[p][k]; p = pre[p]) ch[p][k] = np;
    if(!p && !ch[p][k]) ch[p][k] = np, pre[np] = p;
    else if(step[ch[p][k]] == step[p] + 1) pre[np] = ch[p][k];
    else {
        int q = ch[p][k], nq = ++tot;
        memcpy(ch[nq], ch[q], sizeof(ch[q]));
        step[nq] = step[p] + 1;
        pre[nq] = pre[q];
        pre[q] = pre[np] = nq;
        for(; p && ch[p][k] == q; p = pre[p]) ch[p][k] = nq;
        if(!p && ch[p][k] == q) ch[p][k] = nq;
    }
    ++cnt[np];
}
void clear(){
    rep(i, 0, tot) memset(ch[i], 0, sizeof(ch[i]));
    rep(i, 0, tot) step[i] = pre[i] = cnt[i] = g[i] = cnt2[i] = 0;
    last = tot = 0;
}
void RadixSort(){
    rep(i, 1, tot) g[step[i]]++;
    rep(i, 1, n) g[i] += g[i - 1];
    rep(i, 1, tot) order[g[step[i]]--] = i;
}
int main(){
    while(cin >> MinLen && MinLen){
        scanf("%s", s + 1);
        int n = strlen(s + 1);
        clear();
        rep(i, 1, n)
            add( s[i] <= 'z' && s[i] >= 'a' ? s[i] - 'a' : s[i] - 'A' + 26);
    }
    return 0;
}
```

可持久化并查集

```
#include<iostream>
#include<cstdio>
#include<cstring>
#define rep(i, l, r) for(int i = l; i <= r; ++i)
using namespace std;
const int maxn = 2e5 + 10;
const int logn = 40;
int root[maxn], tot, n, m;
int ln[maxn * logn], rn[maxn * logn], rank[maxn * logn], fa[maxn * logn];
void build(int &x, int l, int r){
    x = ++tot;
    if(l == r) fa[x] = l;
    else {
        int mid = (l + r) >> 1;
        build(ln[x], l, mid);
        build(rn[x], mid + 1, r);
    }
}
void modify(int &x, int last , int l, int r, int p1, int p2){
    x = ++tot;
```

```
    if(l == r){
        fa[x] = p2;
        return ;
    }
    ln[x] = ln[last]; rn[x] = rn[last];
    int mid = (l + r) >> 1;
    if(p1 <= mid) modify(ln[x], ln[last], l, mid, p1, p2);
    else modify(rn[x], rn[last], mid + 1, r, p1, p2);
}
int query(int x, int l, int r, int pos){
    if(l == r) return x;
    int mid = (l + r) >> 1;
    if(pos <= mid) return query(ln[x], l, mid, pos);
    else return query(rn[x], mid + 1, r, pos);
}
int get(int rt, int x){
    int k = query(rt, 1, n, x);
    if(fa[k] == x) return k;
    return get(rt,  fa[k]);
}
void add(int &x, int last , int l, int r, int pos){
    x = ++tot;
    if(l == r)
        rank[x] = rank[last] + 1, fa[x] = fa[last];
    else {
        ln[x] = ln[last];
        rn[x] = rn[last];
        int mid = (l + r) >> 1;
        if(pos <= mid) add(ln[x], ln[last] ,l, mid, pos);
        else add(rn[x], rn[last] ,mid + 1, r, pos);
    }
}
int main(){
    cin >> n >> m;
    build(root[0], 1, n);
    rep(i, 1, m){
        int opt;
        scanf("%d", &opt);
        if(opt == 1){
            int x, y;
            scanf("%d%d", &x, &y);
            root[i] = root[i - 1];
            int r1 = get
            (root[i - 1], x), r2 = get(root[i - 1], y);
            if(fa[r1] == fa[r2]) continue;
            if(rank[r1] > rank[r2]) swap(r1, r2);
            modify(root[i], root[i - 1], 1, n, fa[r1], fa[r2]);
            if(rank[r1] == rank[r2]);
                add(root[i], root[i], 1, n, fa[r2]);
        }
        else if(opt == 2){
            int k;
            scanf("%d", &k);
            root[i] = root[k];
        }
        else {
            int x, y;
            scanf("%d%d", &x, &y);
            root[i] = root[i - 1];
            int r1 = get(root[i], x), r2 = get(root[i], y);
            printf("%d\n", fa[r1] == fa[r2]);
        }
    }
    return 0;
}
```

扩展kmp

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
#define rep(i, l, r) for(int i = l; i <= r; ++i)
using namespace std;
const int maxn = 2e5 + 10;
int n, len, f[maxn];
char s[maxn];
// hdu 43333
void exkmp(){
    int p = 1;
    f[p] = 0;
    rep(i, 2, n){
        f[i] = 0;
        if(p + f[p] - 1 < i)
            while(s[i + f[i]] == s[1 + f[i]]) ++f[i];
        else {
            f[i] = min(f[i - p + 1], p + f[p] - i);
            while(s[i + f[i]] == s[1 + f[i]]) ++f[i];
        }
        if(i + f[i] > p + f[p]) p = i;
    }
}
int main(){
    int cas, cnt1, cnt2, cnt3, val;
    cin >> cas;
    rep(num, 1, cas){
        cnt1 = cnt2 = cnt3 = 0;
        val = 1;
        scanf("%s",s + 1);
        n = strlen(s + 1);
        len = n << 1;
        rep(i, 1, n)s[n + i] = s[i];
        exkmp();
        f[1] = len;
        rep(i, 1, n)
            if(f[i] >= n)
                cnt2++;
            else if(s[f[i] + 1] > s[i + f[i]]) cnt1++;
            else cnt3++;
        cnt1/=cnt2;
        cnt3/=cnt2;
        cnt2 = 1;
        printf("Case %d: %d %d %d\n", num, cnt1, cnt2, cnt3);
    }
    return 0;
}
```

全局最小割

```
#include<iostream>
#include<cstdio>
#include<cstring>
#define maxn 510
#define rep(i,l,r) for(int i=l;i<=r;++i)
using namespace std;
const int INF=1<<29;
struct programer{
    int n,m,d[maxn];
    int w[maxn][maxn],cut,ans;
    bool use[maxn],vis[maxn];
```

```
    void init(){
        cin>>n>>m;
        rep(i,1,m){
            int l,r,v;scanf("%d%d%d",&l,&r,&v);
            w[l][r]+=v;
            w[r][l]+=v;
        }
    }
    int prim(){
        memset(d,0,sizeof(d));
        memset(vis,0,sizeof(vis));
        int s,t;
        cut=0;
        while(1){
            int maxv=-INF,pos;
            rep(i,1,n)if(!use[i]&&!vis[i]&&d[i]>maxv){
                maxv=d[i];
                pos=i;
            }
            if(maxv==-INF){
                rep(i,1,n)if(!use[i]){
                    w[i][s]+=w[i][t];
                    w[s][i]+=w[i][t];
                }
                return t;
            }
            cut=maxv;
            vis[pos]=1;
            s=t;t=pos;
            rep(i,1,n)
                if(!use[i]&&!vis[i])d[i]+=w[pos][i];
        }
    }
    void work(){
        ans=INF;
        rep(i,2,n){
            int k=prim();
            use[k]=1;
            if(cut<ans)ans=cut;
        }
        printf("%d\n",ans);
    }
}program;
int main(){
    program.init();
    program.work();
    return 0;
}
```

最小表示法

```
#include<iostream>
#include<cstdio>
#include<cstring>
#define rep(i, l, r) for(int i = l; i <= r; ++i)
using namespace std;
// zju 2006
int solve(char *s){
    int i = 1, j = 2, len = strlen(s + 1), k = 0;
    while(i <= len && j <= len){
        if(k == len) return min(i, j);
        int p1 = i + k > len ? i + k - len : i + k,
            p2 = j + k > len ? j + k - len : j + k;
        if(s[p1] == s[p2]) ++k;
        else if(s[p1] > s[p2]){
```

```
                i += k + 1;
                if(i == j) ++i;
                k = 0;
            }
            else {
                j += k + 1;
                if(i == j) ++j;
                k = 0;
            }
    }
    return min(i, j);
}
char s[1001];
int main(){
    int n;
    cin >> n;
    rep(i, 1, n){
        scanf("%s", s + 1);
        printf("%d\n",solve(s));
    }
    return 0;
}
```

左偏树 & 斜堆

```
#include<cstdio>
#include<algorithm>
using namespace std;
#define N 1000100
int n,m,w[N],l[N],r[N],d[N],fa[N];bool died[N];char str[9];
int find(int x){return x==fa[x]?x:fa[x]=find(fa[x]);}
int merge(int x,int y){
    if(!x)return y;
    if(!y)return x;
    if(w[x] > w[y]) swap(x,y);
    r[x] = merge(r[x],y);
   /* if(d[r[x]] > d[l[x]]) swap(r[x],l[x]);
    d[x] = d[r[x]] + 1;*/
    swap(l[x], r[x]);
    return x;
}
```