Standard Code Library

Moyi

East Northern University

September 9102

Contents

一切的开始

宏定义

- qwq

```cpp
#include<bits/stdc++.h>
using namespace std;

// #define int long long
#define mst(a) memset(a,0,sizeof(a))
#define cf int Tcodeforces, Tcodeforce;cin>>Tcodeforces;for(Tcodeforce = 1; Tcodeforce <= Tcodeforces; Tcodeforce++)
typedef long long ll;
typedef unsigned long long ull;
const ll maxn = 2e5 +7;
const ll maxm = 2e5 +7;
const ll inf = 0x3f3f3f3f;
const ll mod = 1000000007;//1e9+7

signed main()
{
    freopen("D:/c++source file/intxt/in.txt","r",stdin);
    ios :: sync_with_stdio(0);
    cin.tie(0);

    //cerr<<"Time : "<<1000*((double)clock())/(double)CLOCKS_PER_SEC<<"ms";
    return (0);
}

// --------------------------------------------------------------------------
```

速读

```cpp
//读入整数 可以是负数
inline int read(){
    int x=0,f=1,c=getchar();
    while(!isdigit(c)){if(c=='-')f=-1;c=getchar();}
    while(isdigit(c)){x=(x<<1)+(x<<3)+(c^48);c=getchar();}
    return f==1?x:-x;
}
```

杂类输入

- getline

```cpp
char s[];
int len;
cin.getline(s, len);
```

数据结构

线段树

```
#define mid ((l + r) / 2)
#define lson (rt << 1)
#define rson (rt << 1 | 1)
```

- 普适

```cpp
int a[maxn];
struct node {
    int len, val;//长度 数量
    int s, t, pl, el;//第一个 最后一个 前缀上升长度 后缀上升长度
}tree[maxn<<2];

node Merge(node a, node b) {
    node res;
    res.pl = a.pl;
    res.el = b.el;
```

```
    res.s = a.s;
    res.t = b.t;
    res.len = a.len + b.len;
    res.val = a.val + b.val;
    if(a.t > b.s) return res;
    if(a.pl == a.len) res.pl = a.len + b.pl;
    if(b.el == b.len) res.el = a.el + b.len;
    res.val += a.el * b.pl;
    return res;
}

void update(int rt) {
    tree[rt] = Merge(tree[lson], tree[rson]);
}

void build(int l, int r,int rt) {
    if(l == r){
        tree[rt].len = 1;
        tree[rt].val = 1;
        tree[rt].s = tree[rt].t = a[l];
        tree[rt].el = tree[rt].pl = 1;
        return ;
    }
    build(l, mid, lson);
    build(mid+1, r, rson);
    update(rt);
}

void modify(int l, int r, int rt, int x, int y) {
    if(l == r) {
        tree[rt].s = tree[rt].t = y;
        return ;
    }
    if(x <= mid) modify(l, mid, lson, x, y);
    else modify(mid+1, r, rson, x, y);
    update(rt);
}

node query(int l, int r, int rt, int L, int R) {
    if(L <= l && r <= R) return tree[rt];
    node tmp;
    tmp.len = -1;
    if(L <= mid) tmp = query(l ,mid, lson, L, R);
    if(mid < R) {
        if(tmp.len == -1) tmp = query(mid+1, r, rson, L, R);
        else tmp = Merge(tmp, query(mid+1, r, rson, L, R));
    }
    return tmp;
}
```

+ 加法乘法

```
int n, m, md;
int a[maxn];
struct node{
    int val;
    int add, mult;
}tree[maxn<<2];

void buildtree(int l, int r, int rt) {
    if(l == r) {
        tree[rt].val = a[l];
        tree[rt].add = 0;
        tree[rt].mult = 1;
        tree[rt].val %= md;
        return ;
    }

    buildtree(l, mid, lson);
    buildtree(mid+1, r, rson);
    tree[rt].add = 0;
    tree[rt].mult = 1;
```

```
    tree[rt].val = tree[lson].val + tree[rson].val;
    tree[rt].val %= md;
    return ;
}
//先乘再加
void pushdown(int l, int r, int rt) {
    tree[lson].val = (tree[lson].val * tree[rt].mult + tree[rt].add * (mid-l+1)) % md;
    tree[lson].add = (tree[lson].add * tree[rt].mult + tree[rt].add) % md;
    tree[lson].mult = (tree[lson].mult * tree[rt].mult) % md;

    tree[rson].val = (tree[rson].val * tree[rt].mult + tree[rt].add * (r-mid)) % md;
    tree[rson].add = (tree[rson].add * tree[rt].mult + tree[rt].add) % md;
    tree[rson].mult = (tree[rson].mult * tree[rt].mult) % md;

    tree[rt].add = 0;
    tree[rt].mult = 1;
    return ;
}

int query(int l, int r, int rt, int L, int R) {
    if(L > r || R < l) return 0;
    if(l >= L && r <= R) {return tree[rt].val;}

    pushdown(l, r, rt);
    return query(l, mid, lson, L, R) + query(mid+1, r, rson, L, R);
}

void add(int l, int r, int rt, int L, int R, int val) {
    if(L > r || R < l) return ;
    if(l >= L && r <= R) {
        tree[rt].add += val;
        tree[rt].val += val * (r-l+1);
        tree[rt].val %= md;
        return ;
    }
    pushdown(l,r,rt);
    add(l, mid, lson, L, R, val);
    add(mid+1, r, rson, L, R, val);
    tree[rt].val = tree[lson].val + tree[rson].val;
        tree[rt].val %= md;
    return ;
}

void mult(int l, int r, int rt, int L, int R, int val) {
    if(L > r || R < l) return ;
    if(l >= L && r <= R) {
        pushdown(l, r, rt);
        tree[rt].mult *= val;
        tree[rt].val *= val;
        tree[rt].val %= md;
        return ;
    }

    pushdown(l, r, rt);
    mult(l, mid, lson, L, R, val);
    mult(mid+1, r, rson, L, R, val);
    tree[rt].val = tree[lson].val + tree[rson].val;
        tree[rt].val %= md;
}
```

树状数组

- 注意：0 是无效下标

```
int tree[maxn];
int n;
int lowbit(int x) {return x & -x;}
void add(int pos, int val)
{
    while(pos <= n)
    {
```

```
        tree[pos] += val;
        pos += lowbit(pos);
    }
}
int sum(int pos)
{
    int ans = 0;
    while(pos)
    {
        ans += tree[pos];
        pos -= lowbit(pos);
    }
    return ans;
}
```

- 区间修改 & 区间查询（单点修改，查询前缀和的前缀和）

```
int tree1[maxn], tree2[maxn];
int n, q;
int lowbit(int x) {return x & -x;}
inline int read(){
    int x=0,f=1,c=getchar();
    while(!isdigit(c)){if(c=='-')f=-1;c=getchar();}
    while(isdigit(c)){x=(x<<1)+(x<<3)+(c^48);c=getchar();}
    return f==1?x:-x;
}
void add(int pos, int val)
{
    int addval = val * pos;
    while(pos <= n)
    {
        tree1[pos] += val;
        tree2[pos] += addval;
        pos += lowbit(pos);
    }
}
int sum1(int pos)
{
    // cout<<"sum1:"<<pos<<"=";
    int ans = 0;
    while(pos)
    {
        ans += tree1[pos];
        pos -= lowbit(pos);
    }
    // cout<<ans<<"\n";
    return ans;
}
int sum2(int pos)
{
    // cout<<"sum2:"<<pos<<"=";
    int ans = 0;
    while(pos)
    {
        ans += tree2[pos];
        pos -= lowbit(pos);
    }
    // cout<<ans<<endl;
    return ans;
}
int sum(int pos){return sum1(pos) * (pos + 1) - sum2(pos);}
void modify(int l, int r, int val)
{
    add(l ,val);
    add(r+1, -val);
}
```

- 二维树状数组单点修改

```
int tree[maxn][maxn];
int xn, yy;
int lowbit(int x) {
```

```
    return x & -x;
}
void add(int x, int y, int val) {
    int my = y;

    while (x <= xn) {
        y = my;

        while (y <= yy) {
            tree[x][y] += val;
            y += lowbit(y);
        }

        x += lowbit(x);
    }
}

int getsum(int x, int y) {
    int ans = 0;
    int my = y;

    while (x) {
        y = my;

        while (y) {
            ans += tree[x][y];
            y -= lowbit(y);
        }

        x -= lowbit(x);
    }

    return ans;
}
int q_get(int x1, int y1, int x2, int y2) {
    int ans = 0;
    ans += getsum(x2, y2);
    ans -= getsum(x1 - 1, y2);
    ans -= getsum(x2, y1 - 1);
    ans += getsum(x1 - 1, y1 - 1);
    return ans;
}
```

- 区间修改区间查询二维树状数组

```
int lowbit(int x) {return x & -x;}
void add(int x, int y, int val)
{
    // cout<<x<<" "<<y<<" "<<val<<endl;
    int memoy = y, memox = x;
    while(x <= n)
    {
        y = memoy;
        while(y <= m)
        {
            t1[x][y] += val;
            t2[x][y] += val * memoy;
            t3[x][y] += val * memox;
            t4[x][y] += val * memox * memoy;
            y += lowbit(y);
        }
        x += lowbit(x);
    }
}

int ask(int x, int y)
{
    int ans = 0;
    int memoy = y, memox = x;
    while(x)
    {
        y = memoy;
```

```
        while(y)
        {
            ans += (memoy+1)*(memox+1)*t1[x][y];
            ans -= t2[x][y] * (memox + 1);
            ans -= t3[x][y] * (memoy + 1);
            ans += t4[x][y];
            y -= lowbit(y);
        }
        x -= lowbit(x);
    }
    return ans;
}

void range_add(int xx1, int yy1, int xx2, int yy2, int val)
{
    add(xx1,yy1, val);
    add(xx1, yy2 + 1, -val);
    add(xx2 + 1, yy1, -val);
    add(xx2+1,yy2+1,val);
}

int range_ask(int xx1, int yy1, int xx2, int yy2)
{
    int ans = 0;
    ans += ask(xx1-1,yy1-1);
    ans -= ask(xx1-1,yy2);
    ans -= ask(xx2, yy1-1);
    ans += ask(xx2, yy2);
    return ans;
}
```

并查集

```
int n, m;
int fa[maxn], rk[maxn];
inline void init(){for(int i=0;i<=n;i++){fa[i]=i;rk[i]=1;}}
int find(int x) {return fa[x]==x?x:(fa[x]=find(fa[x]));}
inline void merge(int i, int j)
{
    int x = find(i), y = find(j);
    if(rk[x] <= rk[y]) fa[x] = y;
    else fa[y] = x;
    if(rk[x] == rk[y] && x != y) rk[y]++;
}
```

数学

- gcd

```
ll Gcd(ll a,ll b){return b==0?a:Gcd(b,a%b);}
```

素数筛

- 线性筛

visit[] 第 i 个数是不是质数,1 表示质数， prime[] 质数集合 2 3 5 7 ， k 质数数量, 筛到 maxn

```
bool visit[maxn];
int k, prime[maxn];

void initPrime()//init
{
    visit[0] = visit[1] = 1;
    for(int i = 2; i < maxn; i++)
    {
        if(!visit[i]) prime[k++] = i;
        for(int j = 0; j < k && i*prime[j]<maxn; j++)//遍历素数数组
        {
            visit[i * prime[j]] = 1;
```

```
            if(i % prime[j] == 0) break;
        }
    }
}
```

- 线性筛 + 欧拉函数

欧拉函数（Euler’s totient function），即 $\varphi(n)$，表示的是小于等于 n 和 n 互质的数的个数。存在 phi[i] 里。

```
//phi(i):= 小于等于 i 的与 i 互质的数的个数
const int p_max = 1e5 + 100;
int phi[p_max];
void get_phi() {
    static bool vis[p_max];
    static int prime[p_max], p_sz, d;
    vis[0] = vis[1] = 1; phi[1] = 1;
    for(int i = 2; i < p_max; i++) {
        if (!vis[i]) {
            prime[p_sz++] = i;
            phi[i] = i - 1;
        }
        for (int j = 0; j < p_sz && (d = i * prime[j]) < p_max; ++j) {
            vis[d] = 1;
            if (i % prime[j] == 0) {
                phi[d] = phi[i] * prime[j];
                break;
            }
            else phi[d] = phi[i] * (prime[j] - 1);
        }
    }
}
```

扩展欧几里得

- 求 $ax + by = gcd(a, b)$ 的一组解
- 如果 a 和 b 互素，那么 x 是 a 在模 b 下的逆元
- 注意 x 和 y 可能是负数

```
//ax+by=gcd(a,b) 是否有解 有解解就是 x
bool ex_gcd(int a, int b, int& x, int& y) {
    if(b == 0) {
        x = 1;
        y = 0;
        return a;
    }
    int d = ex_gcd(b, a%b, x, y);
    int temp = x;
    x = y;
    y = temp - a/b * y;
    return d;
}
//同余方程 ax+by=c 即 ax=c(mod b) 是否有解
bool CongruenceEquation(int a, int b,int c, int& x, int& y) {
    int d = ex_gcd(a,b,x,y);
    if(c%d != 0) return 0;
    int k = c / d;
    x *= k;
    y *= k;
    return 1;
}
```

类欧几里得

- $m = \lfloor \frac{an+b}{c} \rfloor$.
- $f(a, b, c, n) = \sum_{i=0}^{n} \lfloor \frac{ai+b}{c} \rfloor$: 当 $a \geq c$ or $b \geq c$ 时，$f(a, b, c, n) = (\frac{a}{c})n(n + 1)/2 + (\frac{b}{c})(n + 1) + f(a \bmod c, b \bmod c, c, n)$；否则 $f(a, b, c, n) = nm - f(c, c - b - 1, a, m - 1)$。
- $g(a, b, c, n) = \sum_{i=0}^{n} i\lfloor \frac{ai+b}{c} \rfloor$: 当 $a \geq c$ or $b \geq c$ 时，$g(a, b, c, n) = (\frac{a}{c})n(n + 1)(2n + 1)/6 + (\frac{b}{c})n(n + 1)/2 + g(a \bmod c, b \bmod c, c, n)$；否则 $g(a, b, c, n) = \frac{1}{2}(n(n + 1)m - f(c, c - b - 1, a, m - 1) - h(c, c - b - 1, a, m - 1))$。

- $h(a,b,c,n) = \sum_{i=0}^{n} \lfloor \frac{ai+b}{c} \rfloor^2$: 当 $a \geq c$ or $b \geq c$ 时，$h(a,b,c,n) = (\frac{a}{c})^2 n(n+1)(2n+1)/6 + (\frac{b}{c})^2 (n+1) + (\frac{a}{c})(\frac{b}{c})n(n+1) + h(a \bmod c, b \bmod c, c, n) + 2(\frac{a}{c})g(a \bmod c, b \bmod c, c, n) + 2(\frac{b}{c})f(a \bmod c, b \bmod c, c, n)$；否则 $h(a,b,c,n) = nm(m+1) - 2g(c, c-b-1, a, m-1) - 2f(c, c-b-1, a, m-1) - f(a,b,c,n)$。

逆元

- 如果 p 不是素数，使用拓展欧几里得
- 前置模板：快速幂 / 扩展欧几里得

```
int n;
ll mod = (int)(1e9+7);
inline ll qpow(ll base, ll rk) {
    ll ans = 1;
    ll now = (base % mod + mod) % mod;
    for(; rk; rk >>= 1) {
        if(rk & 1) {ans *= now; ans %= mod;}
        now *= now; now %= mod;
    }
    return ans;
}

//费马小定理求单个数逆元 mod 必须素数
ll get_inv(ll x) {return qpow(x, mod - 2);}

int inv_n[maxn];
//线性求 n 个数的逆元
void init_n() {
    inv_n[1] = 1;
    for(int i = 2; i <= n; i++) {
        inv_n[i] = 1ll * (mod - mod / i) * inv_n[mod % i] % mod;
    }
}

ll a[maxn], inv[maxn], s[maxn], sv[maxn];//原数组 逆元 前缀积 逆元前缀积
//线性求任意 n 个数的逆元
void init_any() {
    s[0] = 1;
    for(int i = 1; i <= n; i++) s[i] = s[i-1] * a[i] % mod;
    sv[n] = qpow(s[n], mod-2);
    for(int i = n; i >= 1; i--) sv[i-1] = sv[i] * a[i] % mod;
    for(int i = 1; i <= n; i++) inv[i] = sv[i] * s[i-1] % mod;
}
```

- 预处理阶乘及其逆元

```
LL invf[M], fac[M] = {1};
void fac_inv_init(LL n, LL p) {
    FOR (i, 1, n)
        fac[i] = i * fac[i - 1] % p;
    invf[n - 1] = bin(fac[n - 1], p - 2, p);
    FORD (i, n - 2, -1)
        invf[i] = invf[i + 1] * (i + 1) % p;
}
```

组合数

- 如果数较小，模较大时使用逆元
- 前置模板：逆元-预处理阶乘及其逆元

```
inline LL C(LL n, LL m) { // n >= m >= 0
    return n < m || m < 0 ? 0 : fac[n] * invf[m] % MOD * invf[n - m] % MOD;
}
```

快速幂

- 如果模数是素数，则可在函数体内加上 `n %= MOD - 1;`（费马小定理）

```cpp
ll FastPowerMod(ll x, ll n)
{
    if(!n) return 1;
    ll t = 1;
    while(n)
    {
        if(n&1) t *= x;
        n >>= 1;
        x *= x;
        x = (x + mod) % mod;
        t = (t + mod) % mod;
    }
    return t;
}
```

质因数分解

- 前置模板：素数筛
- 带指数

```cpp
LL factor[30], f_sz, factor_exp[30];
void get_factor(LL x) {
    f_sz = 0;
    LL t = sqrt(x + 0.5);
    for (LL i = 0; pr[i] <= t; ++i)
        if (x % pr[i] == 0) {
            factor_exp[f_sz] = 0;
            while (x % pr[i] == 0) {
                x /= pr[i];
                ++factor_exp[f_sz];
            }
            factor[f_sz++] = pr[i];
        }
    if (x > 1) {
        factor_exp[f_sz] = 1;
        factor[f_sz++] = x;
    }
}
```

公式

调和级数部分和

$$S = \frac{1}{1} + \frac{1}{2} + ... + \frac{1}{n} S = ln(n) + eulr + \frac{1}{2n} elur = 0.57721\ 56649\ 01532\ 86060$$

一些数论公式

- 当 $x \geq \phi(p)$ 时有 $a^x \equiv a^{x \bmod \phi(p) + \phi(p)} \pmod p$
- $\mu^2(n) = \sum_{d^2|n} \mu(d)$
- $\sum_{d|n} \varphi(d) = n$
- $\sum_{d|n} 2^{\omega(d)} = \sigma_0(n^2)$，其中 ω 是不同素因子个数
- $\sum_{d|n} \mu^2(d) = 2^{\omega(d)}$

一些数论函数求和的例子

- $\sum_{i=1}^{n} i[gcd(i,n) = 1] = \frac{n\varphi(n) + [n=1]}{2}$
- $\sum_{i=1}^{n} \sum_{j=1}^{m} [gcd(i,j) = x] = \sum_{d} \mu(d) \lfloor \frac{n}{dx} \rfloor \lfloor \frac{m}{dx} \rfloor$
- $\sum_{i=1}^{n} \sum_{j=1}^{m} gcd(i,j) = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{d|gcd(i,j)} \varphi(d) = \sum_{d} \varphi(d) \lfloor \frac{n}{d} \rfloor \lfloor \frac{m}{d} \rfloor$
- $S(n) = \sum_{i=1}^{n} \mu(i) = 1 - \sum_{i=1}^{n} \sum_{d|i, d<i} \mu(d) \overset{t=\frac{i}{d}}{=} 1 - \sum_{t=2}^{n} S(\lfloor \frac{n}{t} \rfloor)$
  - 利用 $[n=1] = \sum_{d|n} \mu(d)$
- $S(n) = \sum_{i=1}^{n} \varphi(i) = \sum_{i=1}^{n} i - \sum_{i=1}^{n} \sum_{d|i, d<i} \varphi(i) \overset{t=\frac{i}{d}}{=} \frac{i(i+1)}{2} - \sum_{t=2}^{n} S(\frac{n}{t})$

  - 利用 $n=\sum_{d|n}\varphi(d)$
- $\sum_{i=1}^{n}\mu^2(i)=\sum_{i=1}^{n}\sum_{d^2|n}\mu(d)=\sum_{d=1}^{\lfloor\sqrt{n}\rfloor}\mu(d)\lfloor\frac{n}{d^2}\rfloor$
- $\sum_{i=1}^{n}\sum_{j=1}^{n}gcd^2(i,j)=\sum_d d^2\sum_t\mu(t)\lfloor\frac{n}{dt}\rfloor^2$
  $\overset{x=dt}{=}\sum_x\lfloor\frac{n}{x}\rfloor^2\sum_{d|x}d^2\mu(\frac{x}{d})$
- $\sum_{i=1}^{n}\varphi(i)=\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}[i\perp j]-1=\frac{1}{2}\sum_{i=1}^{n}\mu(i)\cdot\lfloor\frac{n}{i}\rfloor^2-1$

斐波那契数列性质

- $F_{a+b}=F_{a-1}\cdot F_b+F_a\cdot F_{b+1}$
- $F_1+F_3+\cdots+F_{2n-1}=F_{2n},F_2+F_4+\cdots+F_{2n}=F_{2n+1}-1$
- $\sum_{i=1}^{n}F_i=F_{n+2}-1$
- $\sum_{i=1}^{n}F_i^2=F_n\cdot F_{n+1}$
- $F_n^2=(-1)^{n-1}+F_{n-1}\cdot F_{n+1}$
- $gcd(F_a,F_b)=F_{gcd(a,b)}$
- 模 n 周期（皮萨诺周期）
  - $\pi(p^k)=p^{k-1}\pi(p)$
  - $\pi(nm)=lcm(\pi(n),\pi(m)),\forall n\perp m$
  - $\pi(2)=3,\pi(5)=20$
  - $\forall p\equiv\pm1\pmod{10},\pi(p)|p-1$
  - $\forall p\equiv\pm2\pmod{5},\pi(p)|2p+2$

一些组合公式

- 错排公式：$D_1=0,D_2=1,D_n=(n-1)(D_{n-1}+D_{n-2})=n!(\frac{1}{2!}-\frac{1}{3!}+\cdots+(-1)^n\frac{1}{n!})=\lfloor\frac{n!}{e}+0.5\rfloor$
- 卡塔兰数（n 对括号合法方案数，n 个结点二叉树个数，$n\times n$ 方格中对角线下方的单调路径数，凸 $n+2$ 边形的三角形划分数，n 个元素的合法出栈序列数）：$C_n=\frac{1}{n+1}\binom{2n}{n}=\frac{(2n)!}{(n+1)!n!}$

中国剩余定理

- 无解返回 -1
- 前置模板：扩展欧几里得

```
LL CRT(LL *m, LL *r, LL n) {
    if (!n) return 0;
    LL M = m[0], R = r[0], x, y, d;
    FOR (i, 1, n) {
        d = ex_gcd(M, m[i], x, y);
        if ((r[i] - R) % d) return -1;
        x = (r[i] - R) / d * x % (m[i] / d);
        // 防爆 LL
        // x = mul((r[i] - R) / d, x, m[i] / d);
        R += x * M;
        M = M / d * m[i];
        R %= M;
    }
    return R >= 0 ? R : R + M;
}
```

博弈

- Nim 游戏：每轮从若干堆石子中的一堆取走若干颗。先手必胜条件为石子数量异或和非零。
- 阶梯 Nim 游戏：可以选择阶梯上某一堆中的若干颗向下推动一级，直到全部推下去。先手必胜条件是奇数阶梯的异或和非零（对于偶数阶梯的操作可以模仿）。
- Anti-SG：无法操作者胜。先手必胜的条件是：
  - SG 不为 0 且某个单一游戏的 SG 大于 1 。
  - SG 为 0 且没有单一游戏的 SG 大于 1。
- Every-SG：对所有单一游戏都要操作。先手必胜的条件是单一游戏中的最大 step 为奇数。
  - 对于终止状态 step 为 0
  - 对于 SG 为 0 的状态，step 是最大后继 step +1

  - 对于 SG 非 0 的状态，step 是最小后继 step +1
- 树上删边：叶子 SG 为 0，非叶子结点为所有子结点的 SG 值加 1 后的异或和。

尝试：

- 打表找规律
- 寻找一类必胜态（如对称局面）
- 直接博弈 dp

图论

LCA

- 倍增

```
int N, M, S;
int fa[maxn][31], dep[maxn];
//fa[i][j]: 第 i 个点的第 2^j 个祖先
vector<int> G[maxn];

void dfs(int root,int fno)
{
    fa[root][0] = fno;
    dep[root] = dep[fno] + 1;
    //初始化 fa
    for(int i = 1; i < 31; ++i)
    {
        fa[root][i] = fa[fa[root][i-1]][i-1];
    }
    //遍历
    for(auto y : G[root])
    {
        if(y == fno) continue;
        dfs(y, root);
    }
}

int lca(int x, int y)
{
    if(dep[x] > dep[y]) swap(x,y);
    int tem = dep[y] - dep[x];
    for(int i = 0; tem; ++i, tem >>= 1)
    {
        if(tem&1) y = fa[y][i];
    }
    if(y == x) return x;
    for(int i = 30; i >= 0 && y != x; --i)
    {
        if(fa[x][i] != fa[y][i])
        {
            x = fa[x][i];
            y = fa[y][i];
        }
    }
    return fa[x][0];
}
```

```
void dfs(int u, int fa) {
    pa[u][0] = fa; dep[u] = dep[fa] + 1;
    FOR (i, 1, SP) pa[u][i] = pa[pa[u][i - 1]][i - 1];
    for (int& v: G[u]) {
        if (v == fa) continue;
        dfs(v, u);
    }
}

int lca(int u, int v) {
    if (dep[u] < dep[v]) swap(u, v);
    int t = dep[u] - dep[v];
```

```
    FOR (i, 0, SP) if (t & (1 << i)) u = pa[u][i];
    FORD (i, SP - 1, -1) {
        int uu = pa[u][i], vv = pa[v][i];
        if (uu != vv) { u = uu; v = vv; }
    }
    return u == v ? u : pa[u][0];
}
```

欧拉路径

```
//欧拉回路 1 有向图 2 无向图
int tsk, n, m;
int head[maxn], nxt[maxm], to[maxm], tote;
int ind[maxn], outd[maxn];
void addedge(int u, int v)
{
    nxt[++tote] = head[u];
    head[u] = tote;
    to[tote] = v;
}
int getegs(int rt)
{
    return ind[rt] + outd[rt];
}

int fa[maxn], rk[maxn];
vector<int> path;
int skt[maxm], top = 0;
inline void init(){for(int i=0;i<=n;i++){fa[i]=i;rk[i]=1;}}
int find(int x) {return fa[x]==x?x:(fa[x]=find(fa[x]));}
inline void merge(int i, int j)
{
    int x = find(i), y = find(j);
    if(rk[x] <= rk[y]) fa[x] = y;
    else fa[y] = x;
    if(rk[x] == rk[y] && x != y) rk[y]++;
}

int vis[maxm];
void dfs1(int u)
{
    for(int &e = head[u]; e; e = nxt[e])
    {
        if(vis[e>>1]) continue;int t = e;
        vis[t>>1] = 1;dfs1(to[t]);
        path.push_back(t&1?-(t>>1):t>>1);
    }
}

void task1()
{
    tote = 1;
    mst(vis);
    for(int i = 1; i <= m; i++)
    {
        int u, v;
        cin>>u>>v;
        addedge(u,v);
        addedge(v,u);
        ind[v]++;outd[u]++;
        merge(u,v);
    }
    int ori = -1, f = 1;
    for(int i = 1; i <= n; i++)
    {
        int qwq = getegs(i);
        if(qwq == 0) continue;
        if(qwq & 1) f = 0;
        if(ori == -1) ori = i;
        if(find(ori) != find(i)) f = 0;
    }
```

```
    if(f==0) {cout<<"NO"<<endl; return;}
    else cout<<"YES"<<endl;
    if(ori != -1)
    {
        dfs1(ori);
        for(int i = path.size() - 1; i >= 0; i--)
        {
            cout<<path[i]<<" ";
        }
        cout<<endl;
    }
}

void dfs2(int rt)
{
    for(int &e = head[rt]; e; e = nxt[e])
    {
        if(vis[e]) continue;int t = e;
        vis[e] = 1;
        dfs2(to[e]);
        skt[++top] = t;
    }
}

void task2()
{
    tote = 0;
    for(int i = 1; i <= m; i++)
    {
        int u, v;
        cin>>u>>v;
        addedge(u,v);
        merge(u,v);
        ind[v]++;outd[u]++;
    }
    int ori = -1, f = 1;
    for(int i = 1; i <= n; i++)
    {
        int qwq = getegs(i);
        if(qwq == 0) continue;
        if(ind[i] != outd[i]) f = 0;
        if(ori == -1) ori = i;
        if(find(ori) != find(i)) f = 0;
    }
    if(f == 0) {cout<<"NO"<<endl;return;}
    else cout<<"YES"<<endl;
    if(ori != -1)
    {
        dfs2(ori);
        for(int i = m; i >= 1; i--) cout<<skt[i]<<" ";
        cout<<endl;
        // for(int i = path.size() - 1; i >= 0; i--)
        // cout<<path[i]<<" ";
        // cout<<endl;
    }
}
```

```
int S[N << 1], top;
Edge edges[N << 1];
set<int> G[N];

void DFS(int u) {
    S[top++] = u;
    for (int eid: G[u]) {
        int v = edges[eid].get_other(u);
        G[u].erase(eid);
        G[v].erase(eid);
        DFS(v);
        return;
    }
}

```

```
void fleury(int start) {
    int u = start;
    top = 0; path.clear();
    S[top++] = u;
    while (top) {
        u = S[--top];
        if (!G[u].empty())
            DFS(u);
        else path.push_back(u);
    }
}
```

强连通分量与 2-SAT

```
//2-sat

int n, m;//n 人 1~n(1&2 同党) m 关系
int oth(int x) {return x%2?x+1:x-1;}//另一个人
vector<int> e[maxn];//存边

int dfn[maxn], low[maxn], dfncnt;
int tj_stack[maxn], in_stack[maxn], tp;
int scc[maxn], scc_cnt;
void tarjan(int rt)
{
    dfn[rt] = low[rt] = ++dfncnt;
    tj_stack[++tp] = rt, in_stack[rt] = 1;
    for(int i = 0; i < e[rt].size(); i++)
    {
        int y = e[rt][i];
        if(!dfn[y])
        {
            tarjan(y);
            low[rt] = min(low[rt], low[y]);
        }
        else if(in_stack[y]) low[rt] = min(low[rt], dfn[y]);
    }
    if(dfn[rt] == low[rt])
    {
        ++scc_cnt;
        while(tj_stack[tp] != rt)
        {
            scc[tj_stack[tp]] = scc_cnt;
            in_stack[tj_stack[tp]] = 0;
            tp--;
        }
        scc[tj_stack[tp]] = scc_cnt;
        in_stack[tj_stack[tp]] = 0;
        tp--;
    }
}

void init()
{
    mst(dfn);
    mst(low);
    dfncnt=0;
    tp = 0;
    for(int i = 1; i <= n; i++) e[i].clear();
    mst(scc);
    scc_cnt = 0;
    mst(tj_stack);
    mst(in_stack);
}

int main()
{
    // freopen("D:/c++source file/intxt/in.txt","r",stdin);
    ios :: sync_with_stdio(0);
    cin.tie(0);

```

```
    while(cin>>n>>m)
    {
        n *= 2;
        init();

        for(int i = 1; i <= m; i++)
        {
            int u, v;
            cin>>u>>v;
            e[u].push_back(oth(v));
            e[v].push_back(oth(u));
        }
        for(int i = 1; i <= n; i++)
        {
            if(!dfn[i]) tarjan(i);
        }
        int f = 0;
        for(int i = 1; i <= n; i+=2)
        {
            if(scc[i] == scc[oth(i)])
            {
                cout<<"NIE"<<endl;
                f = 1;
                break;
            }
        }
        if(f) continue;
        for(int i = 1; i <= n; i+=2)
        {
            cout<<(scc[i] > scc[oth(i)] ? oth(i) : i)<<endl;
        }
    }


    //cerr<<"Time : "<<1000*((double)clock())/(double)CLOCKS_PER_SEC<<"ms";
    return (0);
    /*
    先 tarjan 缩点 然后检查行不行 然后直接输出
    */
}
```

```
int n, m;
vector<int> G[N], rG[N], vs;
int used[N], cmp[N];

void add_edge(int from, int to) {
    G[from].push_back(to);
    rG[to].push_back(from);
}

void dfs(int v) {
    used[v] = true;
    for (int u: G[v]) {
        if (!used[u])
            dfs(u);
    }
    vs.push_back(v);
}

void rdfs(int v, int k) {
    used[v] = true;
    cmp[v] = k;
    for (int u: rG[v])
        if (!used[u])
            rdfs(u, k);
}

int scc() {
    memset(used, 0, sizeof(used));
    vs.clear();
    for (int v = 0; v < n; ++v)
        if (!used[v]) dfs(v);
```

```
    memset(used, 0, sizeof(used));
    int k = 0;
    for (int i = (int) vs.size() - 1; i >= 0; --i)
        if (!used[vs[i]]) rdfs(vs[i], k++);
    return k;
}

int main() {
    cin >> n >> m;
    n *= 2;
    for (int i = 0; i < m; ++i) {
        int a, b; cin >> a >> b;
        add_edge(a - 1, (b - 1) ^ 1);
        add_edge(b - 1, (a - 1) ^ 1);
    }
    scc();
    for (int i = 0; i < n; i += 2) {
        if (cmp[i] == cmp[i + 1]) {
            puts("NIE");
            return 0;
        }
    }
    for (int i = 0; i < n; i += 2) {
        if (cmp[i] > cmp[i + 1]) printf("%d\n", i + 1);
        else printf("%d\n", i + 2);
    }
}
```

拓扑排序

```
vector<int> G[MAXN];  // vector 实现的邻接表
int c[MAXN];          // 标志数组
vector<int> topo;     // 拓扑排序后的节点

bool dfs(int u) {
  c[u] = -1;
  for (int v : G[u]) {
    if (c[v] < 0)
      return false;
    else if (!c[v])
      if (!dfs(v)) return false;
  }
  c[u] = 1;
  topo.push_back(u);
  return true;
}

bool toposort() {
  topo.clear();
  memset(c, 0, sizeof(c));
  for (int u = 0; u < n; u++)
    if (!c[u])
      if (!dfs(u)) return false;
  reverse(topo.begin(), topo.end());
  return true;
}
```

```
vector<int> toporder(int n) {
    vector<int> orders;
    queue<int> q;
    for (int i = 0; i < n; i++)
        if (!deg[i]) {
            q.push(i);
            orders.push_back(i);
        }
    while (!q.empty()) {
        int u = q.front(); q.pop();
        for (int v: G[u])
            if (!--deg[v]) {
                q.push(v);
                orders.push_back(v);
```

```
            }
        }
        return orders;
    }
```

Tarjan

割点

- 判断割点
- 注意原图可能不连通

```
int dfn[N], low[N], clk;
void init() { clk = 0; memset(dfn, 0, sizeof dfn); }
void tarjan(int u, int fa) {
    low[u] = dfn[u] = ++clk;
    int cc = fa != -1;
    for (int& v: G[u]) {
        if (v == fa) continue;
        if (!dfn[v]) {
            tarjan(v, u);
            low[u] = min(low[u], low[v]);
            cc += low[v] >= dfn[u];
        } else low[u] = min(low[u], dfn[v]);
    }
    if (cc > 1) // ...
}
```

桥

- 注意原图不连通和重边

```
int dfn[N], low[N], clk;
void init() { memset(dfn, 0, sizeof dfn); clk = 0; }
void tarjan(int u, int fa) {
    low[u] = dfn[u] = ++clk;
    int _fst = 0;
    for (E& e: G[u]) {
        int v = e.to; if (v == fa && ++_fst == 1) continue;
        if (!dfn[v]) {
            tarjan(v, u);
            if (low[v] > dfn[u]) // ...
            low[u] = min(low[u], low[v]);
        } else low[u] = min(low[u], dfn[v]);
    }
}
```

强连通分量缩点

```
int n, m;
int val_ori[maxn];
vector<int> e_ori[maxn], e_scc[maxn];

int dfn[maxn], low[maxn], dfncnt, tj_stack[maxn], in_stack[maxn], tp;
int scc[maxn], scc_cnt, val_scc[maxn];//属于哪个 scc 一共多少 scc 缩点后权值和
//编号从 1 开始
void tarjan(int rt)
{
    dfn[rt] = low[rt] = ++dfncnt;
    tj_stack[++tp] = rt, in_stack[rt] = 1;
    for(auto y : e_ori[rt])
    {
        if(!dfn[y])
        {
            tarjan(y);
            low[rt] = min(low[rt], low[y]);
        }
        else if(in_stack[y]) low[rt] = min(low[rt], dfn[y]);
    }
    if(dfn[rt] == low[rt])
```

```
    {
        ++scc_cnt;
        while(tj_stack[tp] != rt)
        {
            scc[tj_stack[tp]] = scc_cnt;
            in_stack[tj_stack[tp]] = 0;
            val_scc[scc_cnt] += val_ori[tj_stack[tp]];
            tp--;
        }
        scc[tj_stack[tp]] = scc_cnt;
        in_stack[tj_stack[tp]] = 0;
        val_scc[scc_cnt] += val_ori[tj_stack[tp]];
        tp--;
    }
}

int dp[maxn];//当前节点出发最大权值
void search_dp(int rt)
{
    dp[rt] = val_scc[rt];
    int maxv = 0;
    for(auto y : e_scc[rt])
    {
        if(!dp[y]) search_dp(y);
        maxv = max(maxv, dp[y]);
    }
    dp[rt] = dp[rt] + maxv;
}

int main()
{
    // freopen("D:/c++source file/intxt/in.txt","r",stdin);
    ios :: sync_with_stdio(0);
    cin.tie(0);
    cin>>n>>m;
    for(int i = 1; i <= n; i++) cin>>val_ori[i];
    for(int i = 1; i <= m; i++)
    {
        int u, v;
        cin>>u>>v;
        e_ori[u].push_back(v);
    }
    for(int i = 1; i <= n; i++)
    {
        if(!dfn[i])
        {
            tarjan(i);
        }
    }
    for(int i = 1; i <= n; i++)
    {
        for(auto y : e_ori[i])
        {
            if(scc[i] != scc[y])//注意 tarjan 完了逆拓扑序建反图
            {
                e_scc[scc[i]].push_back(scc[y]);
            }
        }
    }
    int ans = 0;
    for(int i = 1; i <= scc_cnt; i++)
    {
        if(!dp[i])
        {
            search_dp(i);
            ans = max(dp[i], ans);
        }
    }
    cout<<ans<<endl;

    // cerr<<"Time : "<<1000*((double)clock())/(double)CLOCKS_PER_SEC<<"ms";
```

```
    return (0);
}
```

```
int low[N], dfn[N], clk, B, bl[N];
vector<int> bcc[N];
void init() { B = clk = 0; memset(dfn, 0, sizeof dfn); }
void tarjan(int u) {
    static int st[N], p;
    static bool in[N];
    dfn[u] = low[u] = ++clk;
    st[p++] = u; in[u] = true;
    for (int& v: G[u]) {
        if (!dfn[v]) {
            tarjan(v);
            low[u] = min(low[u], low[v]);
        } else if (in[v]) low[u] = min(low[u], dfn[v]);
    }
    if (dfn[u] == low[u]) {
        while (1) {
            int x = st[--p]; in[x] = false;
            bl[x] = B; bcc[B].push_back(x);
            if (x == u) break;
        }
        ++B;
    }
}
```

最小生成树及

```
//唯一
struct edge{
    int from, to, val;
}e[maxn];
int n, m, ans;
int fa[maxn];
bool cmp(edge a, edge b){return a.val < b.val;}
void init()
{
    for(int i = 0; i <= n; i++)
    {
        fa[i] = i;
    }
    ans = 0;
}
int find(int x) {return fa[x]==x?x:(fa[x] = find(fa[x]));}
bool uniKruskal()
{
    int sum1 = 0, sum2 = 0;//已使用的 可能使用的
    int p = 0;//相同指针
    int flag = 0, num = 0;
    for(int i = 1; i <= m + 1; i++)
    {
        if(p<i)
        {
            if(sum1 != sum2)
            {
                flag = 1;
                break;
            }
            sum1 = 0, sum2 = 0;
            for(int j = i; j <= m+1; j++)
            {
                if(e[j].val != e[i].val)
                {
                    p = j-1;
                    break;
                }
                if(find(e[j].from) != find(e[j].to))
                ++sum2;
            }
        }
```

```
        if(i>m) break;
        int x = find(e[i].from);
        int y = find(e[i].to);
        if(x != y && num != n-1)
        {
            num++;
            sum1++;
            // merge(x, y);
            fa[x] = fa[y];
            ans += e[i].val;
        }
    }
    if(flag) return false;
    else return true;
}

int main()
{
    freopen("D:/c++source file/intxt/in.txt","r",stdin);
    // ios :: sync_with_stdio(0);
    // cin.tie(0);
    cf
    {
        cin>>n>>m;
        init();
        for(int i = 1; i <= m; i++) cin>>e[i].from>>e[i].to>>e[i].val;
        sort(e+1,e+1+m, cmp);
        if(uniKruskal()) cout<<ans<<endl;
        else cout<<"Not Unique!"<<endl;
    }


    //cerr<<"Time : "<<1000*((double)clock())/(double)CLOCKS_PER_SEC<<"ms";
    return (0);
}
```

Dijk 最短路

```
namespace Dijk
{
    const int maxm = 5e5;
    const int inff = 2147483647;
    int n, m, s;
    int dis[maxn], vis[maxn];//距离 是否在集合里
    struct edge{
        int v, w;
        edge(int b, int c):v(b),w(c){}
    };
    struct node{
        int dis, u;
        bool operator > (const node a) const {return dis>a.dis;}
    };
    vector<edge> e[maxm];
    struct Dijkstra
    {
        void init()//输入数据 重置
        {
            cin>>n>>m>>s;
            for(int i = 0; i <= m; i++) e[i].clear();
            for(int i = 0; i <= n; i++) dis[i] = inff;
            memset(vis, 0, sizeof(vis));
            for(int i = 1; i <= m; i++)
            {
                int a, b, c;
                cin>>a>>b>>c;
                e[a].push_back(edge(b,c));
            }
        }
        void work()//Dijk
        {
            dis[s] = 0;
```

```
            priority_queue<node, vector<node>, greater<node> > q;
            q.push({0, s});
            while(!q.empty())
            {
                int x = q.top().u;q.pop();
                if(vis[x]) continue;
                vis[x] = 1;
                for(int i = 0; i < e[x].size(); i++)
                {
                    int to = e[x][i].v, w = e[x][i].w;
                    if(dis[to] > dis[x] + w)
                    {
                        dis[to] = dis[x] + w;
                        q.push({dis[to], to});
                    }
                }
            }
        }
    };
}

int main()
{
    // freopen("D:/c++source file/intxt/in.txt","r",stdin);
    ios :: sync_with_stdio(0);
    cin.tie(0);
    Dijk::Dijkstra a;
    a.init();
    a.work();
    for(int i = 1; i <= Dijk::n; i++)
    {
        cout<<Dijk::dis[i]<<' ';
    }


    //cerr<<"Time : "<<1000*((double)clock())/(double)CLOCKS_PER_SEC<<"ms";
    return (0);
}
```

计算几何

字符串

manacher

```
namespace Manacher {
      //记得改 maxn 要开两倍
      char Manacher_s[maxn];
      int d[maxn];//变换后的字符串回文半径
      int maxlen;//最长回文串长度
      void manacher(char s[], int n) {
            memset(d,0,sizeof(d));
            int cnt = 0;
            Manacher_s[cnt++] = '#';
            for(int i = 0; i < n; i++) {Manacher_s[cnt++]=s[i];Manacher_s[cnt++]='#';}
            n = strlen(Manacher_s);
            int r = 0, p = 0;
            for(int i = 0; i < n; i++) {
                  if(i < r) d[i] = min(d[2*p-i], r-i);
                  else d[i] = 1;
                  while(i-d[i]>=0&&i+d[i]<n&&Manacher_s[i-d[i]]==Manacher_s[i+d[i]]) d[i]++;
                  if(d[i]+i-1>r) {r=d[i]+i-1;p=i;}
            }
            for(int i = 0; i < n; i++) d[i]--;
            maxlen = 0;
            for(int i = 0; i < n; i++) {
                  if(i&1) maxlen = max(maxlen, (d[i]>>1)*2+1);
                  else maxlen = max(maxlen, d[i]);
            }
```

```
        }
    }
```

哈希

内置了自动双哈希开关（小心 TLE）。

```
#include <bits/stdc++.h>
using namespace std;

#define ENABLE_DOUBLE_HASH

typedef long long LL;
typedef unsigned long long ULL;

const int x = 135;
const int N = 4e5 + 10;
const int p1 = 1e9 + 7, p2 = 1e9 + 9;
ULL xp1[N], xp2[N], xp[N];

void init_xp() {
    xp1[0] = xp2[0] = xp[0] = 1;
    for (int i = 1; i < N; ++i) {
        xp1[i] = xp1[i - 1] * x % p1;
        xp2[i] = xp2[i - 1] * x % p2;
        xp[i] = xp[i - 1] * x;
    }
}

struct String {
    char s[N];
    int length, subsize;
    bool sorted;
    ULL h[N], hl[N];

    ULL hash() {
        length = strlen(s);
        ULL res1 = 0, res2 = 0;
        h[length] = 0;  // ATTENTION!
        for (int j = length - 1; j >= 0; --j) {
        #ifdef ENABLE_DOUBLE_HASH
            res1 = (res1 * x + s[j]) % p1;
            res2 = (res2 * x + s[j]) % p2;
            h[j] = (res1 << 32) | res2;
        #else
            res1 = res1 * x + s[j];
            h[j] = res1;
        #endif
            // printf("%llu\n", h[j]);
        }
        return h[0];
    }

    // 获取子串哈希，左闭右开区间
    ULL get_substring_hash(int left, int right) const {
        int len = right - left;
    #ifdef ENABLE_DOUBLE_HASH
        // get hash of s[left...right-1]
        unsigned int mask32 = ~(0u);
        ULL left1 = h[left] >> 32, right1 = h[right] >> 32;
        ULL left2 = h[left] & mask32, right2 = h[right] & mask32;
        return (((left1 - right1 * xp1[len] % p1 + p1) % p1) << 32) |
               (((left2 - right2 * xp2[len] % p2 + p2) % p2));
    #else
        return h[left] - h[right] * xp[len];
    #endif
    }

    void get_all_subs_hash(int sublen) {
        subsize = length - sublen + 1;
        for (int i = 0; i < subsize; ++i)
```

```
            hl[i] = get_substring_hash(i, i + sublen);
        sorted = 0;
    }

    void sort_substring_hash() {
        sort(hl, hl + subsize);
        sorted = 1;
    }

    bool match(ULL key) const {
        if (!sorted) assert (0);
        if (!subsize) return false;
        return binary_search(hl, hl + subsize, key);
    }

    void init(const char *t) {
        length = strlen(t);
        strcpy(s, t);
    }
};

int LCP(const String &a, const String &b, int ai, int bi) {
    // Find LCP of a[ai...] and b[bi...]
    int l = 0, r = min(a.length - ai, b.length - bi);
    while (l < r) {
        int mid = (l + r + 1) / 2;
        if (a.get_substring_hash(ai, ai + mid) == b.get_substring_hash(bi, bi + mid))
            l = mid;
        else r = mid - 1;
    }
    return l;
}

int check(int ans) {
    if (T.length < ans) return 1;
    T.get_all_subs_hash(ans); T.sort_substring_hash();
    for (int i = 0; i < S.length - ans + 1; ++i)
        if (!T.match(S.get_substring_hash(i, i + ans)))
            return 1;
    return 0;
}

int main() {
    init_xp();  // DON'T FORGET TO DO THIS!

    for (int tt = 1; tt <= kases; ++tt) {
        scanf("%d", &n); scanf("%s", str);
        S.init(str);
        S.hash(); T.hash();
    }
}
```

KMP

```
namespace Kmp {
    //下标从 0 开始
    int nxt[maxn];//前缀函数值
    int cnt = 0;//可以匹配的点的数量
    vector<int> res;//可以匹配的点的下标
    void kmp(int a[], char s[], int n) {
        int j = a[0] = 0;
        for(int i = 1; i < n; i++) {
            while(j && s[i] != s[j]) j = a[j-1];
            a[i] = j += s[i] == s[j];
        }
    }
    void compare(char s[], char mode[], int la, int lb) {
        res.clear(); memset(nxt, 0, sizeof(nxt));
        kmp(nxt, mode, lb);
        int j = 0;
        for(int i = 0; i < la; i++) {
```

```
            while(j && s[i] != mode[j]) j = nxt[j-1];
            if(s[i] == mode[j]) j++;
            if(j == lb) res.push_back(i-j+1);
        }
        cnt = res.size();
    }
};
```

- 前缀函数（每一个前缀的最长 border）

```
void get_pi(int a[], char s[], int n) {
    int j = a[0] = 0;
    FOR (i, 1, n) {
        while (j && s[i] != s[j]) j = a[j - 1];
        a[i] = j += s[i] == s[j];
    }
}
```

- Z 函数（每一个后缀和该字符串的 LCP 长度）

```
void get_z(int a[], char s[], int n) {
    int l = 0, r = 0; a[0] = n;
    FOR (i, 1, n) {
        a[i] = i > r ? 0 : min(r - i + 1, a[i - l]);
        while (i + a[i] < n && s[a[i]] == s[i + a[i]]) ++a[i];
        if (i + a[i] - 1 > r) { l = i; r = i + a[i] - 1; }
    }
}
```

Trie

```
namespace trie {
    int t[N][26], sz, ed[N];
    void init() { sz = 2; memset(ed, 0, sizeof ed); }
    int _new() { memset(t[sz], 0, sizeof t[sz]); return sz++; }
    void ins(char* s, int p) {
        int u = 1;
        FOR (i, 0, strlen(s)) {
            int c = s[i] - 'a';
            if (!t[u][c]) t[u][c] = _new();
            u = t[u][c];
        }
        ed[u] = p;
    }
}
```

AC 自动机

```
const int N = 1e6 + 100, M = 26;

int mp(char ch) { return ch - 'a'; }

struct ACA {
    int ch[N][M], danger[N], fail[N];
    int sz;
    void init() {
        sz = 1;
        memset(ch[0], 0, sizeof ch[0]);
        memset(danger, 0, sizeof danger);
    }
    void insert(const string &s, int m) {
        int n = s.size(); int u = 0, c;
        FOR (i, 0, n) {
            c = mp(s[i]);
            if (!ch[u][c]) {
                memset(ch[sz], 0, sizeof ch[sz]);
                danger[sz] = 0; ch[u][c] = sz++;
            }
            u = ch[u][c];
        }
        danger[u] |= 1 << m;
```

```
    }
    void build() {
        queue<int> Q;
        fail[0] = 0;
        for (int c = 0, u; c < M; c++) {
            u = ch[0][c];
            if (u) { Q.push(u); fail[u] = 0; }
        }
        while (!Q.empty()) {
            int r = Q.front(); Q.pop();
            danger[r] |= danger[fail[r]];
            for (int c = 0, u; c < M; c++) {
                u = ch[r][c];
                if (!u) {
                    ch[r][c] = ch[fail[r]][c];
                    continue;
                }
                fail[u] = ch[fail[r]][c];
                Q.push(u);
            }
        }
    }
} ac;

char s[N];

int main() {
    int n; scanf("%d", &n);
    ac.init();
    while (n--) {
        scanf("%s", s);
        ac.insert(s, 0);
    }
    ac.build();

    scanf("%s", s);
    int u = 0; n = strlen(s);
    FOR (i, 0, n) {
        u = ac.ch[u][mp(s[i])];
        if (ac.danger[u]) {
            puts("YES");
            return 0;
        }
    }
    puts("NO");
    return 0;
}
```

杂项