

Standard Code Library

Moyi

East Northern University

September 9102

Contents

一切的开始	2
宏定义	2
速读	2
数据结构	2
线段树	2
树状数组	4
并查集	7
数学	7
素数筛	7
扩展欧几里得	8
类欧几里得	8
逆元	9
组合数	9
快速幂	9
质因数分解	10
公式	10
调和级数部分和	10
一些数论公式	10
一些数论函数求和的例子	10
斐波那契数列性质	11
一些组合公式	11
中国剩余定理	11
博弈	11
图论	12
LCA	12
欧拉路径	13
强连通分量与 2-SAT	15
拓扑排序	17
Tarjan	18
割点	18
桥	18
强连通分量缩点	18
最小生成树及	20
Dijk 最短路	21
计算几何	22
字符串	22
manacher	22
哈希	23
KMP	24
Trie	25
AC 自动机	25
杂项	26

一切的开始

宏定义

- qwq

```
1  #include<bits/stdc++.h>
2  using namespace std;
3
4  // #define int long long
5  #define mst(a) memset(a,0,sizeof(a))
6  #define cf int Tcodeforces, Tcodeforce;cin>>Tcodeforces;for(Tcodeforce = 1; Tcodeforce <= Tcodeforces; Tcodeforce++)
7  typedef long long ll;
8  typedef unsigned long long ull;
9  const ll maxn = 2e5 +7;
10 const ll maxm = 2e5 +7;
11 const ll inf = 0x3f3f3f3f;
12 const ll mod = 1000000007;//1e9+7
13
14 signed main()
15 {
16     freopen("D:/c++source file/intxt/in.txt","r",stdin);
17     ios :: sync_with_stdio(0);
18     cin.tie(0);
19
20     //cerr<<"Time : "<<1000*((double)clock())/(double)CLOCKS_PER_SEC<<"ms";
21     return (0);
22 }
23
24 // -----
```

速读

```
1 //读入整数 可以是负数
2 inline int read(){
3     int x=0,f=1,c=getchar();
4     while(!isdigit(c)){if(c=='-')f=-1;c=getchar();}
5     while(isdigit(c)){x=(x<<1)+(x<<3)+(c^48);c=getchar();}
6     return f==1?x:-x;
7 }
```

杂类输入

- getline

```
1 char s[];
2 int len;
3 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

```
1 int a[maxn];
2 struct node {
3     int len, val;//长度 数量
4     int s, t, pl, el;//第一个 最后一个 前缀上升长度 后缀上升长度
5 }tree[maxn<<2];
6
7 node Merge(node a, node b) {
8     node res;
9     res.pl = a.pl;
10    res.el = b.el;
```

```

11     res.s = a.s;
12     res.t = b.t;
13     res.len = a.len + b.len;
14     res.val = a.val + b.val;
15     if(a.t > b.s) return res;
16     if(a.pl == a.len) res.pl = a.len + b.pl;
17     if(b.el == b.len) res.el = a.el + b.len;
18     res.val += a.el * b.pl;
19     return res;
20 }
21
22 void update(int rt) {
23     tree[rt] = Merge(tree[lson], tree[rson]);
24 }
25
26 void build(int l, int r, int rt) {
27     if(l == r){
28         tree[rt].len = 1;
29         tree[rt].val = 1;
30         tree[rt].s = tree[rt].t = a[l];
31         tree[rt].el = tree[rt].pl = 1;
32         return ;
33     }
34     build(l, mid, lson);
35     build(mid+1, r, rson);
36     update(rt);
37 }
38
39 void modify(int l, int r, int rt, int x, int y) {
40     if(l == r) {
41         tree[rt].s = tree[rt].t = y;
42         return ;
43     }
44     if(x <= mid) modify(l, mid, lson, x, y);
45     else modify(mid+1, r, rson, x, y);
46     update(rt);
47 }
48
49 node query(int l, int r, int rt, int L, int R) {
50     if(L <= l && r <= R) return tree[rt];
51     node tmp;
52     tmp.len = -1;
53     if(L <= mid) tmp = query(l, mid, lson, L, R);
54     if(mid < R) {
55         if(tmp.len == -1) tmp = query(mid+1, r, rson, L, R);
56         else tmp = Merge(tmp, query(mid+1, r, rson, L, R));
57     }
58     return tmp;
59 }

```

+ 加法乘法

```

1  int n, m, md;
2  int a[maxn];
3  struct node{
4      int val;
5      int add, mult;
6  }tree[maxn<<2];
7
8  void buildtree(int l, int r, int rt) {
9      if(l == r) {
10         tree[rt].val = a[l];
11         tree[rt].add = 0;
12         tree[rt].mult = 1;
13         tree[rt].val %= md;
14         return ;
15     }
16
17     buildtree(l, mid, lson);
18     buildtree(mid+1, r, rson);
19     tree[rt].add = 0;
20     tree[rt].mult = 1;

```

```

21     tree[rt].val = tree[lson].val + tree[rson].val;
22     tree[rt].val %= md;
23     return ;
24 }
25 //先乘再加
26 void pushdown(int l, int r, int rt) {
27     tree[lson].val = (tree[lson].val * tree[rt].mult + tree[rt].add * (mid-l+1)) % md;
28     tree[lson].add = (tree[lson].add * tree[rt].mult + tree[rt].add) % md;
29     tree[lson].mult = (tree[lson].mult * tree[rt].mult) % md;
30
31     tree[rson].val = (tree[rson].val * tree[rt].mult + tree[rt].add * (r-mid)) % md;
32     tree[rson].add = (tree[rson].add * tree[rt].mult + tree[rt].add) % md;
33     tree[rson].mult = (tree[rson].mult * tree[rt].mult) % md;
34
35     tree[rt].add = 0;
36     tree[rt].mult = 1;
37     return ;
38 }
39
40 int query(int l, int r, int rt, int L, int R) {
41     if(L > r || R < l) return 0;
42     if(l >= L && r <= R) {return tree[rt].val;}
43
44     pushdown(l, r, rt);
45     return query(l, mid, lson, L, R) + query(mid+1, r, rson, L, R);
46 }
47
48 void add(int l, int r, int rt, int L, int R, int val) {
49     if(L > r || R < l) return ;
50     if(l >= L && r <= R) {
51         tree[rt].add += val;
52         tree[rt].val += val * (r-l+1);
53         tree[rt].val %= md;
54         return ;
55     }
56     pushdown(l,r,rt);
57     add(l, mid, lson, L, R, val);
58     add(mid+1, r, rson, L, R, val);
59     tree[rt].val = tree[lson].val + tree[rson].val;
60     tree[rt].val %= md;
61     return ;
62 }
63
64 void mult(int l, int r, int rt, int L, int R, int val) {
65     if(L > r || R < l) return ;
66     if(l >= L && r <= R) {
67         pushdown(l, r, rt);
68         tree[rt].mult *= val;
69         tree[rt].val *= val;
70         tree[rt].val %= md;
71         return ;
72     }
73
74     pushdown(l, r, rt);
75     mult(l, mid, lson, L, R, val);
76     mult(mid+1, r, rson, L, R, val);
77     tree[rt].val = tree[lson].val + tree[rson].val;
78     tree[rt].val %= md;
79 }

```

树状数组

- 注意：0 是无效下标

```

1  int tree[maxn];
2  int n;
3  int lowbit(int x) {return x & -x;}
4  void add(int pos, int val)
5  {
6      while(pos <= n)
7      {

```

```

8         tree[pos] += val;
9         pos += lowbit(pos);
10    }
11 }
12 int sum(int pos)
13 {
14     int ans = 0;
15     while(pos)
16     {
17         ans += tree[pos];
18         pos -= lowbit(pos);
19     }
20     return ans;
21 }

```

- 区间修改 & 区间查询（单点修改，查询前缀和的前缀和）

```

1  int tree1[maxn], tree2[maxn];
2  int n, q;
3  int lowbit(int x) {return x & -x;}
4  inline int read(){
5      int x=0,f=1,c=getchar();
6      while(!isdigit(c)){if(c=='-')f=-1;c=getchar();}
7      while(isdigit(c)){x=(x<<1)+(x<<3)+(c^48);c=getchar();}
8      return f==1?x:-x;
9  }
10 void add(int pos, int val)
11 {
12     int addval = val * pos;
13     while(pos <= n)
14     {
15         tree1[pos] += val;
16         tree2[pos] += addval;
17         pos += lowbit(pos);
18     }
19 }
20 int sum1(int pos)
21 {
22     // cout<<"sum1:"<<pos<<"=";
23     int ans = 0;
24     while(pos)
25     {
26         ans += tree1[pos];
27         pos -= lowbit(pos);
28     }
29     // cout<<ans<<"\n";
30     return ans;
31 }
32 int sum2(int pos)
33 {
34     // cout<<"sum2:"<<pos<<"=";
35     int ans = 0;
36     while(pos)
37     {
38         ans += tree2[pos];
39         pos -= lowbit(pos);
40     }
41     // cout<<ans<<endl;
42     return ans;
43 }
44 int sum(int pos){return sum1(pos) * (pos + 1) - sum2(pos);}
45 void modify(int l, int r, int val)
46 {
47     add(l, val);
48     add(r+1, -val);
49 }

```

- 二维树状数组单点修改

```

1  int tree[maxn][maxn];
2  int xn, yy;
3  int lowbit(int x) {

```

```

4     return x & -x;
5 }
6 void add(int x, int y, int val) {
7     int my = y;
8
9     while (x <= xn) {
10         y = my;
11
12         while (y <= yy) {
13             tree[x][y] += val;
14             y += lowbit(y);
15         }
16
17         x += lowbit(x);
18     }
19 }
20
21 int getsum(int x, int y) {
22     int ans = 0;
23     int my = y;
24
25     while (x) {
26         y = my;
27
28         while (y) {
29             ans += tree[x][y];
30             y -= lowbit(y);
31         }
32
33         x -= lowbit(x);
34     }
35
36     return ans;
37 }
38 int q_get(int x1, int y1, int x2, int y2) {
39     int ans = 0;
40     ans += getsum(x2, y2);
41     ans -= getsum(x1 - 1, y2);
42     ans -= getsum(x2, y1 - 1);
43     ans += getsum(x1 - 1, y1 - 1);
44     return ans;
45 }

```

- 区间修改区间查询二维树状数组

```

1 int lowbit(int x) {return x & -x;}
2 void add(int x, int y, int val)
3 {
4     // cout<<x<<" "<<y<<" "<<val<<endl;
5     int memoy = y, memox = x;
6     while(x <= n)
7     {
8         y = memoy;
9         while(y <= m)
10         {
11             t1[x][y] += val;
12             t2[x][y] += val * memoy;
13             t3[x][y] += val * memox;
14             t4[x][y] += val * memox * memoy;
15             y += lowbit(y);
16         }
17         x += lowbit(x);
18     }
19 }
20
21 int ask(int x, int y)
22 {
23     int ans = 0;
24     int memoy = y, memox = x;
25     while(x)
26     {
27         y = memoy;

```

```

28     while(y)
29     {
30         ans += (memory+1)*(memox+1)*t1[x][y];
31         ans -= t2[x][y] * (memox + 1);
32         ans -= t3[x][y] * (memory + 1);
33         ans += t4[x][y];
34         y -= lowbit(y);
35     }
36     x -= lowbit(x);
37 }
38 return ans;
39 }
40
41 void range_add(int xx1, int yy1, int xx2, int yy2, int val)
42 {
43     add(xx1,yy1, val);
44     add(xx1, yy2 + 1, -val);
45     add(xx2 + 1, yy1, -val);
46     add(xx2+1,yy2+1,val);
47 }
48
49 int range_ask(int xx1, int yy1, int xx2, int yy2)
50 {
51     int ans = 0;
52     ans += ask(xx1-1,yy1-1);
53     ans -= ask(xx1-1,yy2);
54     ans -= ask(xx2, yy1-1);
55     ans += ask(xx2, yy2);
56     return ans;
57 }

```

并查集

```

1  int n, m;
2  int fa[maxn], rk[maxn];
3  inline void init(){for(int i=0;i<=n;i++){fa[i]=i;rk[i]=1;}}
4  int find(int x) {return fa[x]==x?x:(fa[x]=find(fa[x]));}
5  inline void merge(int i, int j)
6  {
7      int x = find(i), y = find(j);
8      if(rk[x] <= rk[y]) fa[x] = y;
9      else fa[y] = x;
10     if(rk[x] == rk[y] && x != y) rk[y]++;
11 }

```

数学

- gcd

```

1  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

```

1  bool visit[maxn];
2  int k, prime[maxn];
3
4  void initPrime()//init
5  {
6      visit[0] = visit[1] = 1;
7      for(int i = 2; i < maxn; i++)
8      {
9          if(!visit[i]) prime[k++] = i;
10         for(int j = 0; j < k && i*prime[j]<maxn; j++)//遍历素数数组
11         {
12             visit[i * prime[j]] = 1;

```



```

13         if(i % prime[j] == 0) break;
14     }
15 }
16 }

```

● 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 $\text{phi}[i]$ 里。

```

1 //phi(i):= 小于等于 i 的与 i 互质的数的个数
2 const int p_max = 1e5 + 100;
3 int phi[p_max];
4 void get_phi() {
5     static bool vis[p_max];
6     static int prime[p_max], p_sz, d;
7     vis[0] = vis[1] = 1; phi[1] = 1;
8     for(int i = 2; i < p_max; i++) {
9         if (!vis[i]) {
10             prime[p_sz++] = i;
11             phi[i] = i - 1;
12         }
13         for (int j = 0; j < p_sz && (d = i * prime[j]) < p_max; ++j) {
14             vis[d] = 1;
15             if (i % prime[j] == 0) {
16                 phi[d] = phi[i] * prime[j];
17                 break;
18             }
19             else phi[d] = phi[i] * (prime[j] - 1);
20         }
21     }
22 }

```

扩展欧几里得

- 求 $ax + by = \text{gcd}(a, b)$ 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

```

1 //ax+by=gcd(a,b) 是否有解 有解解就是 x
2 bool ex_gcd(int a, int b, int& x, int& y) {
3     if(b == 0) {
4         x = 1;
5         y = 0;
6         return a;
7     }
8     int d = ex_gcd(b, a%b, x, y);
9     int temp = x;
10    x = y;
11    y = temp - a/b * y;
12    return d;
13 }
14 //同余方程 ax+by=c 即 ax=c(mod b) 是否有解
15 bool CongruenceEquation(int a, int b, int c, int& x, int& y) {
16     int d = ex_gcd(a, b, x, y);
17     if(c%d != 0) return 0;
18     int k = c / d;
19     x *= k;
20     y *= k;
21     return 1;
22 }

```

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 $a \geq c$ 或 $b \geq c$ 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 $a \geq c$ 或 $b \geq c$ 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

```

1  int n;
2  ll mod = (int)(1e9+7);
3  inline ll qpow(ll base, ll rk) {
4      ll ans = 1;
5      ll now = (base % mod + mod) % mod;
6      for(; rk; rk >>= 1) {
7          if(rk & 1) {ans *= now; ans %= mod;}
8          now *= now; now %= mod;
9      }
10     return ans;
11 }
12
13 //费马小定理求单个数逆元 mod 必须素数
14 ll get_inv(ll x) {return qpow(x, mod - 2);}
15
16 int inv_n[maxn];
17 //线性求 n 个数的逆元
18 void init_n() {
19     inv_n[1] = 1;
20     for(int i = 2; i <= n; i++) {
21         inv_n[i] = 1ll * (mod - mod / i) * inv_n[mod % i] % mod;
22     }
23 }
24
25 ll a[maxn], inv[maxn], s[maxn], sv[maxn]; //原数组 逆元 前缀积 逆元前缀积
26 //线性求任意 n 个数的逆元
27 void init_any() {
28     s[0] = 1;
29     for(int i = 1; i <= n; i++) s[i] = s[i-1] * a[i] % mod;
30     sv[n] = qpow(s[n], mod-2);
31     for(int i = n; i >= 1; i--) sv[i-1] = sv[i] * a[i] % mod;
32     for(int i = 1; i <= n; i++) inv[i] = sv[i] * s[i-1] % mod;
33 }

```

- 预处理阶乘及其逆元

```

1  LL invf[M], fac[M] = {1};
2  void fac_inv_init(LL n, LL p) {
3      FOR (i, 1, n)
4          fac[i] = i * fac[i-1] % p;
5      invf[n-1] = bin(fac[n-1], p-2, p);
6      FORD (i, n-2, -1)
7          invf[i] = invf[i+1] * (i+1) % p;
8  }

```

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

```

1  inline LL C(LL n, LL m) { // n >= m >= 0
2      return n < m || m < 0 ? 0 : fac[n] * invf[m] % MOD * invf[n-m] % MOD;
3  }

```

快速幂

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

```

1  ll FastPowerMod(ll x, ll n)
2  {
3      if(!n) return 1;
4      ll t = 1;
5      while(n)
6      {
7          if(n&1) t *= x;
8          n >>= 1;
9          x *= x;
10         x = (x + mod) % mod;
11         t = (t + mod) % mod;
12     }
13     return t;
14 }

```

质因数分解

- 前置模板：素数筛
- 带指数

```

1  LL factor[30], f_sz, factor_exp[30];
2  void get_factor(LL x) {
3      f_sz = 0;
4      LL t = sqrt(x + 0.5);
5      for (LL i = 0; pr[i] <= t; ++i)
6          if (x % pr[i] == 0) {
7              factor_exp[f_sz] = 0;
8              while (x % pr[i] == 0) {
9                  x /= pr[i];
10                 ++factor_exp[f_sz];
11             }
12             factor[f_sz++] = pr[i];
13         }
14     if (x > 1) {
15         factor_exp[f_sz] = 1;
16         factor[f_sz++] = x;
17     }
18 }

```

公式

调和级数部分和

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

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

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

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

斐波那契数列性质

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

一些组合公式

- 错排公式: $D_1 = 0, D_2 = 1, D_n = (n-1)(D_{n-1} + D_{n-2}) = n! (\frac{1}{2!} - \frac{1}{3!} + \dots + (-1)^n \frac{1}{n!}) = \lfloor \frac{n!}{e} + 0.5 \rfloor$
- 卡特兰数 (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

```

1 LL CRT(LL *m, LL *r, LL n) {
2     if (!n) return 0;
3     LL M = m[0], R = r[0], x, y, d;
4     FOR (i, 1, n) {
5         d = ex_gcd(M, m[i], x, y);
6         if ((r[i] - R) % d) return -1;
7         x = (r[i] - R) / d * x % (m[i] / d);
8         // 防爆 LL
9         // x = mul((r[i] - R) / d, x, m[i] / d);
10        R += x * M;
11        M = M / d * m[i];
12        R %= M;
13    }
14    return R >= 0 ? R : R + M;
15 }

```

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

```

1  int N, M, S;
2  int fa[maxn][31], dep[maxn];
3  //fa[i][j]: 第 i 个点的第 2^j 个祖先
4  vector<int> G[maxn];
5
6  void dfs(int root, int fno)
7  {
8      fa[root][0] = fno;
9      dep[root] = dep[fno] + 1;
10     //初始化 fa
11     for(int i = 1; i < 31; ++i)
12     {
13         fa[root][i] = fa[fa[root][i-1]][i-1];
14     }
15     //遍历
16     for(auto y : G[root])
17     {
18         if(y == fno) continue;
19         dfs(y, root);
20     }
21 }
22
23 int lca(int x, int y)
24 {
25     if(dep[x] > dep[y]) swap(x, y);
26     int tem = dep[y] - dep[x];
27     for(int i = 0; tem; ++i, tem >>= 1)
28     {
29         if(tem & 1) y = fa[y][i];
30     }
31     if(y == x) return x;
32     for(int i = 30; i >= 0 && y != x; --i)
33     {
34         if(fa[x][i] != fa[y][i])
35         {
36             x = fa[x][i];
37             y = fa[y][i];
38         }
39     }
40     return fa[x][0];
41 }
42
43 void dfs(int u, int fa) {
44     pa[u][0] = fa; dep[u] = dep[fa] + 1;
45     FOR (i, 1, SP) pa[u][i] = pa[pa[u][i-1]][i-1];
46     for (int& v: G[u]) {
47         if (v == fa) continue;
48         dfs(v, u);
49     }
50 }
51
52 int lca(int u, int v) {
53     if (dep[u] < dep[v]) swap(u, v);
54     int t = dep[u] - dep[v];

```

```

13     FOR (i, 0, SP) if (t & (1 << i)) u = pa[u][i];
14     FORD (i, SP - 1, -1) {
15         int uu = pa[u][i], vv = pa[v][i];
16         if (uu != vv) { u = uu; v = vv; }
17     }
18     return u == v ? u : pa[u][0];
19 }

```

欧拉路径

```

1 //欧拉回路 1 有向图 2 无向图
2 int tsk, n, m;
3 int head[maxn], nxt[maxm], to[maxm], tote;
4 int ind[maxn], outd[maxn];
5 void addedge(int u, int v)
6 {
7     nxt[++tote] = head[u];
8     head[u] = tote;
9     to[tote] = v;
10 }
11 int getegs(int rt)
12 {
13     return ind[rt] + outd[rt];
14 }
15
16 int fa[maxn], rk[maxn];
17 vector<int> path;
18 int skt[maxm], top = 0;
19 inline void init(){for(int i=0;i<=n;i++){fa[i]=i;rk[i]=1;}}
20 int find(int x) {return fa[x]==x?x:(fa[x]=find(fa[x]));}
21 inline void merge(int i, int j)
22 {
23     int x = find(i), y = find(j);
24     if(rk[x] <= rk[y]) fa[x] = y;
25     else fa[y] = x;
26     if(rk[x] == rk[y] && x != y) rk[y]++;
27 }
28
29 int vis[maxm];
30 void dfs1(int u)
31 {
32     for(int &e = head[u]; e; e = nxt[e])
33     {
34         if(vis[e>>1]) continue; int t = e;
35         vis[t>>1] = 1;dfs1(to[t]);
36         path.push_back(t&1?-(t>>1):t>>1);
37     }
38 }
39
40 void task1()
41 {
42     tote = 1;
43     mst(vis);
44     for(int i = 1; i <= m; i++)
45     {
46         int u, v;
47         cin>>u>>v;
48         addedge(u,v);
49         addedge(v,u);
50         ind[v]++;outd[u]++;
51         merge(u,v);
52     }
53     int ori = -1, f = 1;
54     for(int i = 1; i <= n; i++)
55     {
56         int qwq = getegs(i);
57         if(qwq == 0) continue;
58         if(qwq & 1) f = 0;
59         if(ori == -1) ori = i;
60         if(find(ori) != find(i)) f = 0;
61     }

```

```

62     if(f==0) {cout<<"NO"<<endl; return;}
63     else cout<<"YES"<<endl;
64     if(ori != -1)
65     {
66         dfs1(ori);
67         for(int i = path.size() - 1; i >= 0; i--)
68         {
69             cout<<path[i]<<" ";
70         }
71         cout<<endl;
72     }
73 }
74
75 void dfs2(int rt)
76 {
77     for(int &e = head[rt]; e; e = nxt[e])
78     {
79         if(vis[e]) continue;int t = e;
80         vis[e] = 1;
81         dfs2(to[e]);
82         skt[++top] = t;
83     }
84 }
85
86 void task2()
87 {
88     tote = 0;
89     for(int i = 1; i <= m; i++)
90     {
91         int u, v;
92         cin>>u>>v;
93         addedge(u,v);
94         merge(u,v);
95         ind[v]++;outd[u]++;
96     }
97     int ori = -1, f = 1;
98     for(int i = 1; i <= n; i++)
99     {
100         int qwq = getegs(i);
101         if(qwq == 0) continue;
102         if(ind[i] != outd[i]) f = 0;
103         if(ori == -1) ori = i;
104         if(find(ori) != find(i)) f = 0;
105     }
106     if(f == 0) {cout<<"NO"<<endl;return;}
107     else cout<<"YES"<<endl;
108     if(ori != -1)
109     {
110         dfs2(ori);
111         for(int i = m; i >= 1; i--) cout<<skt[i]<<" ";
112         cout<<endl;
113         // for(int i = path.size() - 1; i >= 0; i--)
114         // cout<<path[i]<<" ";
115         // cout<<endl;
116     }
117 }
118
119 int S[N << 1], top;
120 Edge edges[N << 1];
121 set<int> G[N];
122
123 void DFS(int u) {
124     S[top++] = u;
125     for (int eid: G[u]) {
126         int v = edges[eid].get_other(u);
127         G[u].erase(eid);
128         G[v].erase(eid);
129         DFS(v);
130     }
131     return;
132 }
133 }
134 }
135

```

```

16 void fleury(int start) {
17     int u = start;
18     top = 0; path.clear();
19     S[top++] = u;
20     while (top) {
21         u = S[--top];
22         if (!G[u].empty())
23             DFS(u);
24         else path.push_back(u);
25     }
26 }

```

强连通分量与 2-SAT

```

1 //2-sat
2
3 int n, m; //n 人 1~n(1&2 同党) m 关系
4 int oth(int x) {return x%2?x+1:x-1;} //另一个人
5 vector<int> e[maxn]; //存边
6
7 int dfn[maxn], low[maxn], dfncnt;
8 int tj_stack[maxn], in_stack[maxn], tp;
9 int scc[maxn], scc_cnt;
10 void tarjan(int rt)
11 {
12     dfn[rt] = low[rt] = ++dfncnt;
13     tj_stack[++tp] = rt, in_stack[rt] = 1;
14     for(int i = 0; i < e[rt].size(); i++)
15     {
16         int y = e[rt][i];
17         if(!dfn[y])
18         {
19             tarjan(y);
20             low[rt] = min(low[rt], low[y]);
21         }
22         else if(in_stack[y]) low[rt] = min(low[rt], dfn[y]);
23     }
24     if(dfn[rt] == low[rt])
25     {
26         ++scc_cnt;
27         while(tj_stack[tp] != rt)
28         {
29             scc[tj_stack[tp]] = scc_cnt;
30             in_stack[tj_stack[tp]] = 0;
31             tp--;
32         }
33         scc[tj_stack[tp]] = scc_cnt;
34         in_stack[tj_stack[tp]] = 0;
35         tp--;
36     }
37 }
38
39 void init()
40 {
41     mst(dfn);
42     mst(low);
43     dfncnt=0;
44     tp = 0;
45     for(int i = 1; i <= n; i++) e[i].clear();
46     mst(scc);
47     scc_cnt = 0;
48     mst(tj_stack);
49     mst(in_stack);
50 }
51
52 int main()
53 {
54     // freopen("D:/c++source file/intxt/in.txt", "r", stdin);
55     ios :: sync_with_stdio(0);
56     cin.tie(0);
57 }

```



```

58 while(cin>>n>>m)
59 {
60     n *= 2;
61     init();
62
63     for(int i = 1; i <= m; i++)
64     {
65         int u, v;
66         cin>>u>>v;
67         e[u].push_back(oth(v));
68         e[v].push_back(oth(u));
69     }
70     for(int i = 1; i <= n; i++)
71     {
72         if(!dfn[i]) tarjan(i);
73     }
74     int f = 0;
75     for(int i = 1; i <= n; i+=2)
76     {
77         if(scc[i] == scc[oth(i)])
78         {
79             cout<<"NIE"<<endl;
80             f = 1;
81             break;
82         }
83     }
84     if(f) continue;
85     for(int i = 1; i <= n; i+=2)
86     {
87         cout<<(scc[i] > scc[oth(i)] ? oth(i) : i)<<endl;
88     }
89 }
90
91
92 //cerr<<"Time : "<<1000*((double)clock())/(double)CLOCKS_PER_SEC<<"ms";
93 return (0);
94 /*
95 先 tarjan 缩点 然后检查行不行 然后直接输出
96 */
97 }
98
99 int n, m;
100 vector<int> G[N], rG[N], vs;
101 int used[N], cmp[N];
102
103 void add_edge(int from, int to) {
104     G[from].push_back(to);
105     rG[to].push_back(from);
106 }
107
108 void dfs(int v) {
109     used[v] = true;
110     for (int u: G[v]) {
111         if (!used[u])
112             dfs(u);
113     }
114     vs.push_back(v);
115 }
116
117 void rdfs(int v, int k) {
118     used[v] = true;
119     cmp[v] = k;
120     for (int u: rG[v])
121         if (!used[u])
122             rdfs(u, k);
123 }
124
125 int scc() {
126     memset(used, 0, sizeof(used));
127     vs.clear();
128     for (int v = 0; v < n; ++v)
129         if (!used[v]) dfs(v);

```

```

32     memset(used, 0, sizeof(used));
33     int k = 0;
34     for (int i = (int) vs.size() - 1; i >= 0; --i)
35         if (!used[vs[i]]) rdfs(vs[i], k++);
36     return k;
37 }
38
39 int main() {
40     cin >> n >> m;
41     n *= 2;
42     for (int i = 0; i < m; ++i) {
43         int a, b; cin >> a >> b;
44         add_edge(a - 1, (b - 1) ^ 1);
45         add_edge(b - 1, (a - 1) ^ 1);
46     }
47     scc();
48     for (int i = 0; i < n; i += 2) {
49         if (cmp[i] == cmp[i + 1]) {
50             puts("NIE");
51             return 0;
52         }
53     }
54     for (int i = 0; i < n; i += 2) {
55         if (cmp[i] > cmp[i + 1]) printf("%d\n", i + 1);
56         else printf("%d\n", i + 2);
57     }
58 }

```

拓扑排序

```

1  vector<int> G[MAXN]; // vector 实现的邻接表
2  int c[MAXN]; // 标志数组
3  vector<int> topo; // 拓扑排序后的节点
4
5  bool dfs(int u) {
6      c[u] = -1;
7      for (int v : G[u]) {
8          if (c[v] < 0)
9              return false;
10         else if (!c[v])
11             if (!dfs(v)) return false;
12     }
13     c[u] = 1;
14     topo.push_back(u);
15     return true;
16 }
17
18 bool toposort() {
19     topo.clear();
20     memset(c, 0, sizeof(c));
21     for (int u = 0; u < n; u++)
22         if (!c[u])
23             if (!dfs(u)) return false;
24     reverse(topo.begin(), topo.end());
25     return true;
26 }
27
28 vector<int> toporder(int n) {
29     vector<int> orders;
30     queue<int> q;
31     for (int i = 0; i < n; i++)
32         if (!deg[i]) {
33             q.push(i);
34             orders.push_back(i);
35         }
36     while (!q.empty()) {
37         int u = q.front(); q.pop();
38         for (int v : G[u])
39             if (--deg[v] == 0) {
40                 q.push(v);
41                 orders.push_back(v);
42             }
43     }
44     return orders;
45 }

```

```

15         }
16     }
17     return orders;
18 }

```

Tarjan

割点

- 判断割点
- 注意原图可能不连通

```

1  int dfn[N], low[N], clk;
2  void init() { clk = 0; memset(dfn, 0, sizeof dfn); }
3  void tarjan(int u, int fa) {
4      low[u] = dfn[u] = ++clk;
5      int cc = fa != -1;
6      for (int& v: G[u]) {
7          if (v == fa) continue;
8          if (!dfn[v]) {
9              tarjan(v, u);
10             low[u] = min(low[u], low[v]);
11             cc += low[v] >= dfn[u];
12         } else low[u] = min(low[u], dfn[v]);
13     }
14     if (cc > 1) // ...
15 }

```

桥

- 注意原图不连通和重边

```

1  int dfn[N], low[N], clk;
2  void init() { memset(dfn, 0, sizeof dfn); clk = 0; }
3  void tarjan(int u, int fa) {
4      low[u] = dfn[u] = ++clk;
5      int _fst = 0;
6      for (E& e: G[u]) {
7          int v = e.to; if (v == fa && ++_fst == 1) continue;
8          if (!dfn[v]) {
9              tarjan(v, u);
10             if (low[v] > dfn[u]) // ...
11                 low[u] = min(low[u], low[v]);
12             } else low[u] = min(low[u], dfn[v]);
13     }
14 }

```

强连通分量缩点

```

1  int n, m;
2  int val_ori[maxn];
3  vector<int> e_ori[maxn], e_scc[maxn];
4
5  int dfn[maxn], low[maxn], dfncnt, tj_stack[maxn], in_stack[maxn], tp;
6  int scc[maxn], scc_cnt, val_scc[maxn]; //属于哪个 scc 一共多少 scc 缩点后权值和
7  //编号从 1 开始
8  void tarjan(int rt)
9  {
10     dfn[rt] = low[rt] = ++dfncnt;
11     tj_stack[++tp] = rt, in_stack[rt] = 1;
12     for(auto y: e_ori[rt])
13     {
14         if(!dfn[y])
15         {
16             tarjan(y);
17             low[rt] = min(low[rt], low[y]);
18         }
19         else if(in_stack[y]) low[rt] = min(low[rt], dfn[y]);
20     }
21     if(dfn[rt] == low[rt])

```

```

22     {
23         ++scc_cnt;
24         while(tj_stack[tp] != rt)
25         {
26             scc[tj_stack[tp]] = scc_cnt;
27             in_stack[tj_stack[tp]] = 0;
28             val_scc[scc_cnt] += val_ori[tj_stack[tp]];
29             tp--;
30         }
31         scc[tj_stack[tp]] = scc_cnt;
32         in_stack[tj_stack[tp]] = 0;
33         val_scc[scc_cnt] += val_ori[tj_stack[tp]];
34         tp--;
35     }
36 }
37
38 int dp[maxn]; //当前节点出发最大权值
39 void search_dp(int rt)
40 {
41     dp[rt] = val_scc[rt];
42     int maxv = 0;
43     for(auto y : e_scc[rt])
44     {
45         if(!dp[y]) search_dp(y);
46         maxv = max(maxv, dp[y]);
47     }
48     dp[rt] = dp[rt] + maxv;
49 }
50
51 int main()
52 {
53     // freopen("D:/c++source file/intxt/in.txt", "r", stdin);
54     ios :: sync_with_stdio(0);
55     cin.tie(0);
56     cin >> n >> m;
57     for(int i = 1; i <= n; i++) cin >> val_ori[i];
58     for(int i = 1; i <= m; i++)
59     {
60         int u, v;
61         cin >> u >> v;
62         e_ori[u].push_back(v);
63     }
64     for(int i = 1; i <= n; i++)
65     {
66         if(!dfn[i])
67         {
68             tarjan(i);
69         }
70     }
71     for(int i = 1; i <= n; i++)
72     {
73         for(auto y : e_ori[i])
74         {
75             if(scc[i] != scc[y]) //注意 tarjan 完了逆拓扑序建反图
76             {
77                 e_scc[scc[i]].push_back(scc[y]);
78             }
79         }
80     }
81     int ans = 0;
82     for(int i = 1; i <= scc_cnt; i++)
83     {
84         if(!dp[i])
85         {
86             search_dp(i);
87             ans = max(dp[i], ans);
88         }
89     }
90     cout << ans << endl;
91
92     // cerr << "Time : " << 1000 * ((double)clock()) / (double)CLOCKS_PER_SEC << "ms";

```

```

93     return (0);
94 }

1  int low[N], dfn[N], clk, B, bl[N];
2  vector<int> bcc[N];
3  void init() { B = clk = 0; memset(dfn, 0, sizeof dfn); }
4  void tarjan(int u) {
5      static int st[N], p;
6      static bool in[N];
7      dfn[u] = low[u] = ++clk;
8      st[p++] = u; in[u] = true;
9      for (int& v: G[u]) {
10         if (!dfn[v]) {
11             tarjan(v);
12             low[u] = min(low[u], low[v]);
13         } else if (in[v]) low[u] = min(low[u], dfn[v]);
14     }
15     if (dfn[u] == low[u]) {
16         while (1) {
17             int x = st[--p]; in[x] = false;
18             bl[x] = B; bcc[B].push_back(x);
19             if (x == u) break;
20         }
21         ++B;
22     }
23 }

```

最小生成树及

```

1  //唯一
2  struct edge{
3      int from, to, val;
4  }e[maxn];
5  int n, m, ans;
6  int fa[maxn];
7  bool cmp(edge a, edge b){return a.val < b.val;}
8  void init()
9  {
10     for(int i = 0; i <= n; i++)
11     {
12         fa[i] = i;
13     }
14     ans = 0;
15 }
16 int find(int x) {return fa[x]==x?x:(fa[x] = find(fa[x]));}
17 bool uniKruskal()
18 {
19     int sum1 = 0, sum2 = 0; //已使用的 可能使用的
20     int p = 0; //相同指针
21     int flag = 0, num = 0;
22     for(int i = 1; i <= m + 1; i++)
23     {
24         if(p < i)
25         {
26             if(sum1 != sum2)
27             {
28                 flag = 1;
29                 break;
30             }
31             sum1 = 0, sum2 = 0;
32             for(int j = i; j <= m + 1; j++)
33             {
34                 if(e[j].val != e[i].val)
35                 {
36                     p = j - 1;
37                     break;
38                 }
39                 if(find(e[j].from) != find(e[j].to))
40                     ++sum2;
41             }
42         }

```

```

43     if(i>m) break;
44     int x = find(e[i].from);
45     int y = find(e[i].to);
46     if(x != y && num != n-1)
47     {
48         num++;
49         sum1++;
50         // merge(x, y);
51         fa[x] = fa[y];
52         ans += e[i].val;
53     }
54 }
55 if(flag) return false;
56 else return true;
57 }
58
59 int main()
60 {
61     freopen("D:/c++source file/intxt/in.txt", "r", stdin);
62     // ios :: sync_with_stdio(0);
63     // cin.tie(0);
64     cf
65     {
66         cin>>n>>m;
67         init();
68         for(int i = 1; i <= m; i++) cin>>e[i].from>>e[i].to>>e[i].val;
69         sort(e+1, e+1+m, cmp);
70         if(uniKruskal()) cout<<ans<<endl;
71         else cout<<"Not Unique!"<<endl;
72     }
73
74
75     //cerr<<"Time : "<<1000*((double)clock())/(double)CLOCKS_PER_SEC<<"ms";
76     return (0);
77 }

```

Dijk 最短路

```

1  namespace Dijk
2  {
3      const int maxm = 5e5;
4      const int inff = 2147483647;
5      int n, m, s;
6      int dis[maxm], vis[maxm]; //距离 是否在集合里
7      struct edge{
8          int v, w;
9          edge(int b, int c):v(b),w(c){}
10     };
11     struct node{
12         int dis, u;
13         bool operator > (const node a) const {return dis>a.dis;}
14     };
15     vector<edge> e[maxm];
16     struct Dijkstra
17     {
18         void init()//输入数据 重置
19         {
20             cin>>n>>m>>s;
21             for(int i = 0; i <= m; i++) e[i].clear();
22             for(int i = 0; i <= n; i++) dis[i] = inff;
23             memset(vis, 0, sizeof(vis));
24             for(int i = 1; i <= m; i++)
25             {
26                 int a, b, c;
27                 cin>>a>>b>>c;
28                 e[a].push_back(edge(b,c));
29             }
30         }
31         void work()//Dijk
32         {
33             dis[s] = 0;

```

```

34     priority_queue<node, vector<node>, greater<node> > q;
35     q.push({0, s});
36     while(!q.empty())
37     {
38         int x = q.top().u;q.pop();
39         if(vis[x]) continue;
40         vis[x] = 1;
41         for(int i = 0; i < e[x].size(); i++)
42         {
43             int to = e[x][i].v, w = e[x][i].w;
44             if(dis[to] > dis[x] + w)
45             {
46                 dis[to] = dis[x] + w;
47                 q.push({dis[to], to});
48             }
49         }
50     }
51 }
52 };
53 }
54
55 int main()
56 {
57     // freopen("D:/c++source file/intxt/in.txt", "r", stdin);
58     ios :: sync_with_stdio(0);
59     cin.tie(0);
60     Dijk::Dijkstra a;
61     a.init();
62     a.work();
63     for(int i = 1; i <= Dijk::n; i++)
64     {
65         cout<<Dijk::dis[i]<<' ' ;
66     }
67
68
69     //cerr<<"Time : "<<1000*((double)clock())/(double)CLOCKS_PER_SEC<<"ms";
70     return (0);
71 }

```

计算几何

字符串

manacher

```

1  namespace Manacher {
2      //记得改 maxn 要开两倍
3      char Manacher_s[maxn];
4      int d[maxn]; //变换后的字符串回文半径
5      int maxlen; //最长回文串长度
6      void manacher(char s[], int n) {
7          memset(d, 0, sizeof(d));
8          int cnt = 0;
9          Manacher_s[cnt++] = '#';
10         for(int i = 0; i < n; i++) {Manacher_s[cnt++] = s[i]; Manacher_s[cnt++] = '#';}
11         n = strlen(Manacher_s);
12         int r = 0, p = 0;
13         for(int i = 0; i < n; i++) {
14             if(i < r) d[i] = min(d[2*p-i], r-i);
15             else d[i] = 1;
16             while(i-d[i]>=0&& i+d[i]<n&& Manacher_s[i-d[i]]==Manacher_s[i+d[i]]) d[i]++;
17             if(d[i]+i-1>r) {r=d[i]+i-1;p=i;}
18         }
19         for(int i = 0; i < n; i++) d[i]--;
20         maxlen = 0;
21         for(int i = 0; i < n; i++) {
22             if(i&1) maxlen = max(maxlen, (d[i]>>1)*2+1);
23             else maxlen = max(maxlen, d[i]);
24         }
25     }
26 }

```

```

25     }
26 }

```

哈希

内置了自动双哈希开关（小心 TLE）。

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  #define ENABLE_DOUBLE_HASH
5
6  typedef long long LL;
7  typedef unsigned long long ULL;
8
9  const int x = 135;
10 const int N = 4e5 + 10;
11 const int p1 = 1e9 + 7, p2 = 1e9 + 9;
12 ULL xp1[N], xp2[N], xp[N];
13
14 void init_xp() {
15     xp1[0] = xp2[0] = xp[0] = 1;
16     for (int i = 1; i < N; ++i) {
17         xp1[i] = xp1[i - 1] * x % p1;
18         xp2[i] = xp2[i - 1] * x % p2;
19         xp[i] = xp[i - 1] * x;
20     }
21 }
22
23 struct String {
24     char s[N];
25     int length, subsize;
26     bool sorted;
27     ULL h[N], hl[N];
28
29     ULL hash() {
30         length = strlen(s);
31         ULL res1 = 0, res2 = 0;
32         h[length] = 0; // ATTENTION!
33         for (int j = length - 1; j >= 0; --j) {
34             #ifdef ENABLE_DOUBLE_HASH
35                 res1 = (res1 * x + s[j]) % p1;
36                 res2 = (res2 * x + s[j]) % p2;
37                 h[j] = (res1 << 32) | res2;
38             #else
39                 res1 = res1 * x + s[j];
40                 h[j] = res1;
41             #endif
42             // printf("%llu\n", h[j]);
43         }
44         return h[0];
45     }
46
47     // 获取子串哈希, 左闭右开区间
48     ULL get_substring_hash(int left, int right) const {
49         int len = right - left;
50         #ifdef ENABLE_DOUBLE_HASH
51             // get hash of s[left...right-1]
52             unsigned int mask32 = ~(0u);
53             ULL left1 = h[left] >> 32, right1 = h[right] >> 32;
54             ULL left2 = h[left] & mask32, right2 = h[right] & mask32;
55             return (((left1 - right1 * xp1[len] % p1 + p1) % p1) << 32) |
56                 (((left2 - right2 * xp2[len] % p2 + p2) % p2));
57         #else
58             return h[left] - h[right] * xp[len];
59         #endif
60     }
61
62     void get_all_subs_hash(int sublen) {
63         subsize = length - sublen + 1;
64         for (int i = 0; i < subsize; ++i)

```



```

65         hl[i] = get_substring_hash(i, i + sublen);
66         sorted = 0;
67     }
68
69     void sort_substring_hash() {
70         sort(hl, hl + subsize);
71         sorted = 1;
72     }
73
74     bool match(ULL key) const {
75         if (!sorted) assert (0);
76         if (!subsize) return false;
77         return binary_search(hl, hl + subsize, key);
78     }
79
80     void init(const char *t) {
81         length = strlen(t);
82         strcpy(s, t);
83     }
84 };
85
86 int LCP(const String &a, const String &b, int ai, int bi) {
87     // Find LCP of a[ai...] and b[bi...]
88     int l = 0, r = min(a.length - ai, b.length - bi);
89     while (l < r) {
90         int mid = (l + r + 1) / 2;
91         if (a.get_substring_hash(ai, ai + mid) == b.get_substring_hash(bi, bi + mid))
92             l = mid;
93         else r = mid - 1;
94     }
95     return l;
96 }
97
98 int check(int ans) {
99     if (T.length < ans) return 1;
100    T.get_all_subs_hash(ans); T.sort_substring_hash();
101    for (int i = 0; i < S.length - ans + 1; ++i)
102        if (!T.match(S.get_substring_hash(i, i + ans)))
103            return 1;
104    return 0;
105 }
106
107 int main() {
108     init_xp(); // DON'T FORGET TO DO THIS!
109
110     for (int tt = 1; tt <= kases; ++tt) {
111         scanf("%d", &n); scanf("%s", str);
112         S.init(str);
113         S.hash(); T.hash();
114     }
115 }

```

KMP

```

1  namespace Kmp {
2      //下标从 0 开始
3      int nxt[maxn]; //前缀函数值
4      int cnt = 0; //可以匹配的点的数量
5      vector<int> res; //可以匹配的点的下标
6      void kmp(int a[], char s[], int n) {
7          int j = a[0] = 0;
8          for (int i = 1; i < n; i++) {
9              while (j && s[i] != s[j]) j = a[j-1];
10             a[i] = j += s[i] == s[j];
11         }
12     }
13     void compare(char s[], char mode[], int la, int lb) {
14         res.clear(); memset(nxt, 0, sizeof(nxt));
15         kmp(nxt, mode, lb);
16         int j = 0;
17         for (int i = 0; i < la; i++) {

```

```

18         while(j && s[i] != mode[j]) j = nxt[j-1];
19         if(s[i] == mode[j]) j++;
20         if(j == lb) res.push_back(i-j+1);
21     }
22     cnt = res.size();
23 }
24 };

```

- 前缀函数 (每一个前缀的最长 border)

```

1 void get_pi(int a[], char s[], int n) {
2     int j = a[0] = 0;
3     FOR (i, 1, n) {
4         while (j && s[i] != s[j]) j = a[j - 1];
5         a[i] = j += s[i] == s[j];
6     }
7 }

```

- Z 函数 (每一个后缀和该字符串的 LCP 长度)

```

1 void get_z(int a[], char s[], int n) {
2     int l = 0, r = 0; a[0] = n;
3     FOR (i, 1, n) {
4         a[i] = i > r ? 0 : min(r - i + 1, a[i - l]);
5         while (i + a[i] < n && s[a[i]] == s[i + a[i]]) ++a[i];
6         if (i + a[i] - 1 > r) { l = i; r = i + a[i] - 1; }
7     }
8 }

```

Trie

```

1 namespace trie {
2     int t[N][26], sz, ed[N];
3     void init() { sz = 2; memset(ed, 0, sizeof ed); }
4     int _new() { memset(t[sz], 0, sizeof t[sz]); return sz++; }
5     void ins(char* s, int p) {
6         int u = 1;
7         FOR (i, 0, strlen(s)) {
8             int c = s[i] - 'a';
9             if (!t[u][c]) t[u][c] = _new();
10            u = t[u][c];
11        }
12        ed[u] = p;
13    }
14 }

```

AC 自动机

```

1 const int N = 1e6 + 100, M = 26;
2
3 int mp(char ch) { return ch - 'a'; }
4
5 struct ACA {
6     int ch[N][M], danger[N], fail[N];
7     int sz;
8     void init() {
9         sz = 1;
10        memset(ch[0], 0, sizeof ch[0]);
11        memset(danger, 0, sizeof danger);
12    }
13    void insert(const string &s, int m) {
14        int n = s.size(); int u = 0, c;
15        FOR (i, 0, n) {
16            c = mp(s[i]);
17            if (!ch[u][c]) {
18                memset(ch[sz], 0, sizeof ch[sz]);
19                danger[sz] = 0; ch[u][c] = sz++;
20            }
21            u = ch[u][c];
22        }
23        danger[u] |= 1 << m;
24    }
25 }

```

```

24     }
25     void build() {
26         queue<int> Q;
27         fail[0] = 0;
28         for (int c = 0, u; c < M; c++) {
29             u = ch[0][c];
30             if (u) { Q.push(u); fail[u] = 0; }
31         }
32         while (!Q.empty()) {
33             int r = Q.front(); Q.pop();
34             danger[r] |= danger[fail[r]];
35             for (int c = 0, u; c < M; c++) {
36                 u = ch[r][c];
37                 if (!u) {
38                     ch[r][c] = ch[fail[r]][c];
39                     continue;
40                 }
41                 fail[u] = ch[fail[r]][c];
42                 Q.push(u);
43             }
44         }
45     }
46 } ac;
47
48 char s[N];
49
50 int main() {
51     int n; scanf("%d", &n);
52     ac.init();
53     while (n--) {
54         scanf("%s", s);
55         ac.insert(s, 0);
56     }
57     ac.build();
58
59     scanf("%s", s);
60     int u = 0; n = strlen(s);
61     FOR (i, 0, n) {
62         u = ac.ch[u][mp(s[i])];
63         if (ac.danger[u]) {
64             puts("YES");
65             return 0;
66         }
67     }
68     puts("NO");
69     return 0;
70 }

```

杂项