# Template

Khoray

October 20, 2023

# Contents

## 4    Misc                                                                                40

# 1 Data Structure

## 1.1 线段树

- 维护的信息 Data 满足幺半群性质（有幺元，结合，封闭）。

- Mapping 是区间修改操作，必须满足可以进行 Composition（用于合并 Lazy 标记），Mapping 需要存在幺元。

```cpp
#include<bits/stdc++.h>
using namespace std;
int mod;
struct info {
    int sum;
    int sz;
};
struct F {
    int add, mul;
};
inline info merge(info a, info b) {
    return {(a.sum + b.sum) % mod, (a.sz + b.sz) % mod};
}
inline info mapping(F f, info a) {
    return {(1ll * a.sum * f.mul % mod + 1ll * f.add * a.sz % mod) % mod, a.sz};
}
// f作用于g
inline F comp(F f, F g) {
    return {(1ll * g.add * f.mul % mod + f.add) % mod, 1ll * f.mul * g.mul % mod};
}
const int N = 1e5 + 1;
struct segmenttree {

    info tr[N << 2];
    F tag[N << 2];
    int treel[N << 2], treer[N << 2];
    void build(int u, int l, int r, vector<int> &a) {
        treel[u] = l;
        treer[u] = r;
        tag[u] = {0, 1};
        if(l == r) {
            tr[u].sum = a[l];
            tr[u].sz = 1;
            return;
        }
        int mid = l + r >> 1;
        build(u << 1, l, mid, a);
        build(u << 1 | 1, mid + 1, r, a);
        tr[u] = merge(tr[u << 1], tr[u << 1 | 1]);
    }

    void pushdown(int u) {
        tr[u << 1] = mapping(tag[u], tr[u << 1]);
        tr[u << 1 | 1] = mapping(tag[u], tr[u << 1 | 1]);
        tag[u << 1] = comp(tag[u], tag[u << 1]);
```

```
46          tag[u << 1 | 1] = comp(tag[u], tag[u << 1 | 1]);
47          tag[u] = {0, 1};
48      }
49
50      info query(int u, int L, int R) {
51          if(treel[u] > R || treer[u] < L) return {0, 0};
52          if(treel[u] >= L && treer[u] <= R) {
53              return tr[u];
54          }
55          pushdown(u);
56          int mid = treel[u] + treer[u] >> 1;
57          return merge(
58              query(u << 1, L, R),
59              query(u << 1 | 1, L, R)
60          );
61      }
62
63      void modify(int u, int L, int R, F f) {
64          if(treel[u] > R || treer[u] < L) return;
65          if(treel[u] >= L && treer[u] <= R) {
66              tr[u] = mapping(f, tr[u]);
67              tag[u] = comp(f, tag[u]);
68              return;
69          }
70          pushdown(u);
71          modify(u << 1, L, R, f);
72          modify(u << 1 | 1, L, R, f);
73          tr[u] = merge(tr[u << 1], tr[u << 1 | 1]);
74      }
75
76      template<typename Check>
77      int max_right(int u, info &s, int l, int r, int L, Check check) {
78          if(r < L) return -1;
79          if(l >= L) {
80              info ss = merge(s, tr[u]);
81              if(check(ss)) {
82                  s = ss;
83                  return -1;
84              }
85              if(l == r) return l;
86          }
87          pushdown(u);
88          int mid = l + r >> 1;
89          int pos = max_right(u << 1, s, l, mid, L, check);
90          if(pos != -1) return pos;
91          return max_right(u << 1 | 1, s, mid + 1, r, L, check);
92      }
93
94
95 };
96
97 segmenttree segt;
98 signed main() {
```

```
 99     int n, m;
100
101     cin >> n >> m;
102     cin >> mod;
103
104     vector<int> a(n + 1);
105     for(int i = 1; i <= n; i++) cin >> a[i];
106
107     segt.build(1, 1, n, a);
108
109     while(m--) {
110         int op; cin >> op;
111         if(op == 1) {
112             int l, r; cin >> l >> r;
113             int mul; cin >> mul;
114             segt.modify(1, l, r, {0, mul});
115         } else if(op == 2) {
116             int l, r, add; cin >> l >> r >> add;
117             segt.modify(1, l, r, {add, 1});
118         } else if(op == 3) {
119             int l, r; cin >> l >> r;
120             info x = segt.query(1, l, r);
121             cout << x.sum << "\n";
122         } else if(op == 4) {
123             int l, val; cin >> l >> val;
124             info u = {0, 0};
125             int pos = segt.max_right(1, u, 1, n, l, [&] (info x) {
126                 return x.sum <= val;
127             });
128             cout << pos << "\n";
129         }
130     }
131
132     return 0;
133 }
```

## 1.2  珂朵莉树

//TODO 我是菜鸡，以后再加！

# 2 Math

## 2.1 欧拉筛/积性函数筛/线性筛

- 积性函数筛，$f(pq) = f(p)f(q), (p, q) = 1$

- calc_f(val, power) 返回 $f(val^{power})$

```
// all in which f[pq] = f[p] * f[q] (gcd(p, q) = 1)
const int N = 1e7 + 5;
int pri[N / 5], notpri[N], prinum, minpri_cnt[N], f[N];

int calc_f(int val, int power) {

}

void init_pri() {
    for(int i = 2; i < N; i++) {
        if(!notpri[i]) pri[++prinum] = i, minpri_cnt[i] = 1, f[i] = calc_f(i, 1);
        for(int j = 1; j <= prinum && pri[j] * i < N; j++) {
            notpri[pri[j] * i] = pri[j];
            if(i % pri[j] == 0) {
                minpri_cnt[pri[j] * i] = minpri_cnt[i] + 1;
                f[pri[j] * i] = f[i] / calc_f(pri[j], minpri_cnt[i]) * calc_f(pri[j],
                        minpri_cnt[i] + 1);
                break;
            }
            minpri_cnt[pri[j] * i] = 1;
            f[pri[j] * i] = f[i] * calc_f(pri[j], 1);
        }
    }
}
```

## 2.2 快速幂

```
int ksm(int a, int b = mod - 2, int MOD_KSM = mod) {
    int ret = 1;
    while(b) {
        if(b & 1) {
            ret = (ll) ret * a % MOD_KSM;
        }
        a = (ll) a * a % MOD_KSM;
        b >>= 1;
    }
    return ret;
}
```

## 2.3 组合数

### 2.3.1 暴力

- 暴力求组合数 $\binom{n}{k}$，时间复杂度 $O(\min(k, n - k))$.

- 前置：快速幂

- 模数必须是质数!

```
1   int binom(int n, int k) {
2       if(n < 0 || k < 0 || k > n) { return 0; }
3       k = min(n - k, k);
4       int u = 1, v = 1;
5       for(int i = 0; i < k; i++) {
6           v = v * (i + 1) % mod;
7           u = u * (n - i) % mod;
8       }
9       return u * ksm(v, mod - 2) % mod;
10  }
```

### 2.3.2　递推

- $O(n^2)$ 递推求，模数随意。

```
1   const int N = 31;
2   int C[N][N];
3
4   void init_C() {
5       C[0][0] = C[1][0] = C[1][1] = 1;
6       for(int i = 2; i < N; i++) {
7           C[i][0] = 1;
8           for(int j = 1; j <= i; j++)
9               C[i][j] = (C[i - 1][j] + C[i - 1][j - 1]) % mod;
10      }
11  }
12
13  int binom(int n, int k) {
14      if(n < 0 || k < 0 || n < k) return 0;
15      return C[n][k];
16  }
```

### 2.3.3　逆元

- 模数必须是质数!

- 前置：快速幂

```
1   const int N = 1e7 + 5;
2   const int mod = 1e9 + 7;
3   int facinv[N], fac[N];
4   void init_fac() {
5       fac[0] = fac[1] = 1;
6       for(int i = 2; i < N; i++) {
7           fac[i] = (ll) fac[i - 1] * i % mod;
8       }
9       facinv[N - 1] = ksm(fac[N - 1], mod - 2);
10      for(int i = N - 2; i >= 0; i--) {
```

```
11          facinv[i] = (ll) facinv[i + 1] * (i + 1) % mod;
12      }
13  }
14  int binom(int n, int k) {
15      if(n < 0 || k < 0 || k > n) { return 0; }
16      return (ll) fac[n] * facinv[n - k] % mod * facinv[k] % mod;
17  }
18
19  int inv[N];
20  void init_inv() {
21      inv[0] = inv[1] = 1;
22      for(int i = 2; i < N; i++) {
23          inv[i] = (ll) (mod - mod / i) * inv[mod % i] % mod;
24      }
25  }
```

## 2.4   ExGCD

- 求解 $ax + by = (a, b)$ 的特解 $x_0, y_0$.

- 通解 $x^* = x_0 + \frac{bk}{(a,b)}, y^* = y_0 - \frac{ak}{(a,b)}$ （ $k \in \mathbb{Z}$ ）.

```
1   // ax + by = gcd(a, b)
2   // return gcd(a, b)
3   // all sol: x = x_0 + k[a, b] / a, y = y_0 - k[a, b] / b;
4   int exgcd(int &x, int &y, int a, int b) {
5       if(!b) {
6           x = 1;
7           y = 0;
8           return a;
9       }
10      int ret = exgcd(x, y, b, a % b);
11      int t = x;
12      x = y;
13      y = t - a / b * y;
14      return ret;
15  }
```

## 2.5   拉格朗日插值

- $f(x)$ 是多项式，并且我们知道一系列连续的点值 $f(l), \cdots, f(r)$ ，求解 $f(n)$。$O(r - l)$

- 前置：逆元组合数

- 模数为质数

```
1   int LagrangeInterpolation(vector<int> &y, int l, int r, int n) {
2       if(n <= r && n >= l) return y[n];
3       vector<int> lg(r - l + 3), rg(r - l + 3);
4       int ret = 0;
5       lg[0] = 1;
6       rg[r - l + 2] = 1;
```

```
 7     for(int i = l; i <= r; i++) {
 8         lg[i - l + 1] = (ll) lg[i - l] * (n - i) % mod;
 9     }
10     for(int i = r; i >= l; i--) {
11         rg[i - l + 1] = (ll) rg[i - l + 2] * (n - i) % mod;
12     }
13     for(int i = l; i <= r; i++) {
14         if(r - i & 1) {
15             ret = (ret - (ll) y[i] * lg[i - l] % mod * rg[i - l + 2] % mod * facinv[i
                   - l] % mod * facinv[r - i] % mod + mod) % mod;
16         } else {
17             ret = (ret + (ll) y[i] * lg[i - l] % mod * rg[i - l + 2] % mod * facinv[i
                   - l] % mod * facinv[r - i] % mod) % mod;
18         }
19     }
20     return ret;
21 }
```

## 2.6  原根

若 $g$ 满足：

$$(g, m) = 1$$

$$\delta_m(g) = \phi(m)$$

则 $g$ 为 $m$ 的原根。找所有原根：

1. 找到最小的原根 $g$。如果一个数 $g$ 是原根，那么 $\forall p | \phi(m) : g^{\frac{\phi(m)}{p}} \neq 1$

2. 找小于 $m$ 与 $\phi(m)$ 互质的数 $k$，则 $g^k$ 也是原根（能覆盖所有原根）个数为 $\phi(\phi(m))$ 个。

题目：找出 $n$ 的所有原根，间隔 $d$ 输出。

```
 1 void solve() {
 2     // d 是间隔输出（对题目无影响
 3     int n, d; cin >> n >> d;
 4     vector<int> pf; // 质因数分解
 5     int pn = phi[n];
 6     while(notpri[pn]) {
 7         int now = notpri[pn];
 8         pf.push_back(now);
 9         while(pn % now == 0) pn /= now;
10     }
11     if(pn != 1) {
12         pf.push_back(pn);
13     }
14     int cnt = 0;
15     int ming = -1;
16     vector<int> ans, vis(n); // 记录答案
17     // 找到最小的原根 min_g
18     for(int i = 1; i < n; i++) {
19         if(__gcd(i, n) != 1) continue;
20         int judge = 1;
21         for(auto &p : pf) {
```

```
22          if(ksm(i, phi[n] / p, n) == 1) {
23              judge = 0;
24              break;
25          }
26      }
27      if(judge) {
28          ming = i;
29          break;
30      }
31  }
32  // 还原出所有原根 g
33  if(ming > 0) {
34      for(int i = 1; i < n; i++) {
35          if(__gcd(i, phi[n]) == 1) {
36              int cur = ksm(ming, i, n);
37              if(!vis[cur]) {
38                  vis[cur] = 1;
39                  ans.push_back(cur);
40              }
41          }
42      }
43  }
44  // 排序输出所有原根
45  cout << ans.size() << '\n';
46  sort(ans.begin(), ans.end());
47  for(auto as : ans) {
48      cnt++;
49      if(cnt % d == 0) {
50          cout << as << ' ';
51      }
52  }
53  cout << '\n';
54 }
```

## 2.7 Ex-Baby-Step-Giant-Step-Algorithm

BSGS

求解 $a^x = b \pmod{p}, (0 \le x < p)$

令 $x = A\lceil\sqrt{p}\rceil - B$，其中 $0 \le A, B \le \lceil\sqrt{p}\rceil$，则有 $a^{A\lceil\sqrt{p}\rceil - B} \equiv b \pmod{p}$，稍加变换，则有 $a^{A\lceil\sqrt{p}\rceil} \equiv ba^B \pmod{p}$。

我们已知的是 $a, b$，所以我们可以先算出等式右边的 $ba^B$ 的所有取值，枚举 $B$，用 'hash'/'map' 存下来，然后逐一计算 $a^{A\lceil\sqrt{p}\rceil}$，枚举 $A$，寻找是否有与之相等的 $ba^B$，从而我们可以得到所有的 $x$，$x = A\lceil\sqrt{p}\rceil - B$。

注意到 $A, B$ 均小于 $\lceil\sqrt{p}\rceil$，所以时间复杂度为 $\Theta\left(\sqrt{p}\right)$，用 'map' 则多一个 log。

exBSGS

其中 $a, p$ 不一定互质。

当 $a \perp p$ 时，在模 $p$ 意义下 $a$ 存在逆元，因此可以使用 BSGS 算法求解。于是我们想办法让他们变得互质。

具体地，设 $d_1 = \gcd(a, p)$。如果 $d_1 \nmid b$，则原方程无解。否则我们把方程同时除以 $d_1$，得到

$$\frac{a}{d_1} \cdot a^{x-1} \equiv \frac{b}{d_1} \pmod{\frac{p}{d_1}}$$

如果 $a$ 和 $\frac{p}{d_1}$ 仍不互质就再除，设 $d_2 = \gcd\left(a, \frac{p}{d_1}\right)$。如果 $d_2 \nmid \frac{b}{d_1}$，则方程无解；否则同时除以 $d_2$ 得到

$$\frac{a^2}{d_1 d_2} \cdot a^{x-2} \, \frac{b}{d_1 d_2} \pmod{\frac{p}{d_1 d_2}}$$

同理，这样不停的判断下去。直到 $a \perp \frac{p}{d_1 d_2 \cdots d_k}$。

记 $D = \prod_{i=1}^{k} d_i$，于是方程就变成了这样：

$$\frac{a^k}{D} \cdot a^{x-k} \equiv \frac{b}{D} \pmod{\frac{p}{D}}$$

由于 $a \perp \frac{p}{D}$，于是推出 $\frac{a^k}{D} \perp \frac{p}{D}$。这样 $\frac{a^k}{D}$ 就有逆元了，于是把它丢到方程右边，这就是一个普通的 BSGS 问题了，于是求解 $x - k$ 后再加上 $k$ 就是原方程的解啦。

注意，不排除解小于等于 $k$ 的情况，所以在消因子之前做一下 $\Theta(k)$ 枚举，直接验证 $a^i \equiv b \pmod{p}$，这样就能避免这种情况。

- 注意，inv 必须由扩欧求！

- 注意开 long long

- 前置：ksm, exgcd 求逆元

```cpp
int bsgs(int a, int b, int p) { //BSGS算法
    unordered_map<int, int> f;
    int m = ceil(sqrt(p));
    b %= p;
    for(int i = 1; i <= m; i++) {
        b = b * a % p;
        f[b] = i;
    }
    int tmp = ksm(a, m, p);
    b = 1;
    for(int i = 1; i <= m; i++) {
        b = b * tmp % p;
        if(f[b]) {
            return (i * m - f[b] + p) % p;
        }
    }
    return -1;
}
int exbsgs(int a, int b, int p) {
    b %= p;
    a %= p;
    if(b == 1 || p == 1) {
        return 0; //特殊情况，x=0时最小解
    }
    int g = __gcd(a, p), k = 0, na = 1;
    while(g > 1) {
        if(b % g != 0) {
            return -1; //无法整除则无解
        }
        k++;
        b /= g;
        p /= g;
```

```
33        na = na * (a / g) % p;
34        if(na == b) {
35            return k; //na=b说明前面的a的次数为0，只需要返回k
36        }
37        g = __gcd(a, p);
38    }
39    int f = bsgs(a, b * inv(na, p) % p, p);
40    if(f == -1) {
41        return -1;
42    }
43    return f + k;
44 }
```

## 2.8 逆元

### 2.8.1 exgcd 求逆元

- 前置：exgcd

- $(x, p) = 1$

```
1 int inv(int x, int p) {
2     int y, k;
3     int gcd = exgcd(y, k, x, p);
4     int moder = p / gcd;
5     return (y % moder + moder) % moder;
6 }
```

### 2.8.2 快速幂求逆元

根据费马小定理：$p \in primes \rightarrow a^{-1} \equiv a^{p-2} \pmod{p}$

### 2.8.3 整数除法取模

如果 $\frac{a}{b} \in \mathbb{N}$, $b \times p$ 可以在计算机中表示，那么 $\frac{a}{b} \bmod p = \frac{a \bmod (p \times b)}{b}$

## 2.9 上下取整

- $b$ 必须为正整数。

```
1 // b must be positive integer
2
3 int updiv(int a, int b) {
4     return a > 0 ? (a + b - 1) / b : a / b;
5 }
6
7 int downdiv(int a, int b) {
8     return a > 0 ? a / b : (a - b + 1) / b;
9 }
```

## 2.10 线性基

- $O(\log x)$ insert

- $O(\log^2 x)$ get-kth

- $O(\log x)$ get-max

- 如果问能否通过选一些数（不能不选）异或得到 0，必须特判。

```cpp
struct linear_basis {
    vector<int> base, kth;
    int size, max_size, builded;
    /// @brief 构造线性基，向量长度是n
    /// @param n
    linear_basis(int n) : base(n), size(0), max_size(n), builded(0) {}
    /// @brief 插入一个数
    /// @param x
    void insert(int x) {
        builded = 0;
        for(int i = max_size - 1; i >= 0; i--) {
            if((x >> i) & 1) {
                if(!base[i]) {
                    base[i] = x, size++;
                    break;
                }
                else x ^= base[i];
            }
        }
    }
    /// @brief 获取最大值
    /// @return 最大值
    int get_max() {
        int ret = 0;
        for(int i = max_size - 1; i >= 0; i--) {
            if(!((ret >> i) & 1) && base[i]) ret ^= base[i];
        }
        return ret;
    }
    /// @brief 查询是否能等于x
    /// @param x
    /// @return 查询是否能等于x
    bool can_eq(int x) {
        int now = 0;
        for(int i = max_size - 1; i >= 0; i--) {
            if(((now >> i) & 1) != ((x >> i) & 1)) {
                if(!base[i]) return false;
                else now ^= base[i];
            }
        }
        return true;
    }

    /// @brief 询问第k大的值，insert后需要先buildk
```

```
45        /// @param k
46        /// @return 第k大的值
47        int get_kth(int k) {
48            if(k >= 1ll << size) return -1;
49            if(!builded) buildk();
50            int ret = 0;
51            for(int i = size - 1; ~i; i--) {
52                if(k >> i & 1) {
53                    ret ^= kth[i];
54                }
55            }
56            return ret;
57        }
58        /// @brief 找到小于等于x的数的个数
59        /// @param x
60        /// @return 小于等于x的数的个数
61        int get_rank(int x) {
62            int tmpsz = size, ret = 0, now = 0;
63            for(int i = max_size - 1; i >= 0; i--) {
64                if(base[i]) tmpsz--;
65                if((x >> i) & 1) {
66                    if(!((now >> i) & 1)) {
67                        ret += 1ll << tmpsz;
68                        if(!base[i]) return -1;
69                        else now ^= base[i];
70                    } else {
71                        if(base[i]) ret += 1ll << tmpsz;
72                    }
73                } else {
74                    if((now >> i) & 1) {
75                        if(!base[i]) return -1;
76                        else now ^= base[i];
77                    }
78                }
79            }
80            return ret;
81        }
82  private:
83        /// @brief 消成上阶梯矩阵
84        void buildk() {
85            builded = 1;
86            kth.resize(size);
87            int cnt = size;
88            for(int i = max_size - 1; ~i; i--) {
89                if(base[i]) {
90                    for(int j = i - 1; ~j; j--) {
91                        if(base[i] >> j & 1) {
92                            base[i] ^= base[j];
93                        }
94                    }
95                }
96            }
97            for(int i = max_size - 1; ~i; i--) {
```

```
98          if(base[i]) {
99              kth[--cnt] = base[i];
100         }
101     }
102   }
103 };
```

## 2.11  高斯消元

- equ 是方程个数，n 是变元个数，答案存在 ans。

- return：无解 (-1)，自由变元个数。

### 2.11.1  解异或线性方程组

```
1   const int N = 1005;
2   template<int N>
3   struct Matrix {
4       bitset<N> mat[N];
5       Matrix() { }
6       bitset<N> &operator [] (int idx) { return mat[idx]; }
7   };
8
9   template<int N>
10  int guass(int n, int equ, Matrix<N> a, vector<int> b, vector<int> &ans) {
11      fill(ans.begin(), ans.end(), 0);
12      vector<int> fre(n + 1);
13      int row, col;
14      for(row = 1, col = 1; col <= n; col++) {
15          if(!a[row][col]) {
16              int sw = 0;
17              for(int i = row + 1; i <= equ; i++) {
18                  if(a[i][col]) {
19                      swap(a[row], a[i]);
20                      swap(b[row], b[i]);
21                      sw = 1;
22                      break;
23                  }
24              }
25              if(!sw) {
26                  fre[col] = 1;
27                  continue;
28              }
29          }
30          for(int i = row + 1; i <= equ; i++) {
31              if(a[i][col]) {
32                  a[i] ^= a[row];
33                  b[i] ^= b[row];
34              }
35          }
36          row++;
37      }
38      if(row <= equ) {
```

```
39        for(int i = row; i <= equ; i++) {
40            if(b[i]) return -1;
41        }
42    }
43    int all = 0;
44    for(col = n; col >= 1; col--) {
45        if(fre[col]) {
46            ans[col] = 1;
47            all++;
48        } else {
49            row--;
50            ans[col] = b[row];
51            for(int i = col + 1; i <= n; i++) {
52                if(a[row][i]) ans[col] ^= ans[i];
53            }
54        }
55    }
56    return all;
57 }
```

### 2.11.2 解 double 线性方程组

```
1  const int N = 1005;
2  template<int N>
3  struct Matrix {
4      bitset<N> mat[N];
5      Matrix() { }
6      bitset<N> &operator [] (int idx) { return mat[idx]; }
7  };
8  template<int N>
9  int guass(int n, int equ, Matrix<N> a, vector<double> b, vector<double> &ans) {
10     fill(ans.begin(), ans.end(), 0);
11     vector<int> fre(n + 1);
12     int row, col;
13     for(row = 1, col = 1; col <= n; col++) {
14         double mx = fabs(a[row][col]);
15         int mxp = row;
16         for(int i = row + 1; i <= equ; i++) {
17             if(fabs(a[row][col]) > mx) {
18                 mx = fabs(a[row][col]);
19                 mxp = i;
20             }
21         }
22         if(mxp != row) {
23             for(int i = col; i <= n; i++) {
24                 swap(a[row][i], a[mxp][i]);
25             }
26             swap(b[row], b[mxp]);
27         }
28         if(fabs(a[row][col]) < eps) {
29             fre[col] = 1;
30         }
```

```
31        for(int i = row + 1; i <= equ; i++) {
32            if(fabs(a[i][col]) > eps) {
33                double k = a[i][col] / a[row][col];
34                for(int j = col; j <= n; j++) {
35                    a[i][j] -= a[row][j] * k;
36                }
37                b[i] -= b[row] * k;
38            }
39        }
40        row++;
41    }
42    // 判断解是否存在
43    if(row <= equ) {
44        for(int i = row; i <= equ; i++) {
45            if(fabs(b[i]) > eps) return -1;
46        }
47    }
48
49    // 回代求解
50    int all = 0;
51    for(col = n; col >= 1; col--) {
52        if(fre[col]) {
53            ans[col] = 0;
54            all++;
55        } else {
56            row--;
57            ans[col] = b[row];
58            for(int i = col + 1; i <= n; i++) {
59                ans[col] -= ans[i] * a[row][i];
60            }
61            ans[col] /= a[row][col];
62        }
63    }
64    return all;
65 }
```

### 2.11.3 解模意义线性方程组

- 时间复杂度 $O(n^3 \log mod)$

```
1  #define mul(a, b) (ll(a) * (b) % mod)
2  #define add(a, b) (((a) += (b)) >= mod ? (a) -= mod : 0) // (a += b) %= P
3  #define dec(a, b) (((a) -= (b)) < 0 ? (a) += mod: 0) // ((a -= b) += P) %= P
4
5  const int N = 1005;
6  const int mod = 1e9 + 7;
7  template<int N>
8  struct Matrix {
9      int mat[N][N];
10     Matrix() { }
11     int* operator [] (int idx) { return mat[idx]; }
12 };
```

```cpp
template<int N>
int guass(int n, int equ, Matrix<N> a, vector<int> b, vector<int> &ans) {
    fill(ans.begin(), ans.end(), 0);
    vector<int> fre(n + 1);
    int row, col;
    for(row = 1, col = 1; col <= n; col++) {
        if(!a[row][col]) {
            int sw = 0;
            for(int i = row + 1; i <= equ; i++) {
                if(a[i][col]) {
                    for(int j = col; j <= n; j++) {
                        swap(a[row][j], a[i][j]);
                    }
                    swap(b[row], b[i]);
                    sw = 1;
                    break;
                }
            }
            if(!sw) {
                fre[col] = 1;
                continue;
            }
        }
        for(int i = row + 1; i <= equ; i++) {
            if(a[i][col]) {
                int k = a[i][col] * ksm(a[row][col]) % mod;
                for(int j = col; j <= n; j++) {
                    dec(a[i][j], a[row][j] * k % mod);
                }
                dec(b[i], b[row] * k % mod);
            }
        }
        row++;
    }
    if(row <= equ) {
        for(int i = row; i <= equ; i++) {
            if(b[i]) return -1;
        }
    }
    int all = 0;
    for(col = n; col >= 1; col--) {
        if(fre[col]) {
            ans[col] = 0;
            all++;
        } else {
            row--;
            ans[col] = b[row];
            for(int i = col + 1; i <= n; i++) {
                dec(ans[col], ans[i] * a[row][i] % mod);
            }
            mul(ans[col], ksm(a[row][col]));
        }
    }
```

```
66    return all;
67 }
```

## 2.12  Miller-Rabin

- 前置：快速幂 (__int128!!!)

- int 范围: 2, 7, 61

- long long 范围: 2, 325, 9375, 28178, 450775, 9780504, 1795265022

- 4E13: 2, 2570940, 211991001, 3749873356

- 3E15: 2, 2570940, 880937, 610386380, 4130785767

- 注意看判断范围是 int 还是 long long

```cpp
1  bool is_prime(ll n) {
2      if(n < 2 || n % 6 % 4 != 1) return (n | 1) == 3;
3      int A[] = {2, 325, 9375, 28178, 450775, 9780504, 1795265022},
4              s = __builtin_ctzll(n - 1), d = n >> s;
5      for(int a : A) { // ^ count t ra i l in g zeroes
6          int p = ksm(a % n, d, n), i = s;
7          while(p != 1 && p != n - 1 && a % n && i--)
8              p = (__int128) p * p % n;
9          if(p != n - 1 && i != s) return 0;
10     }
11     return 1;
12 }
```

## 2.13  Pollard-Rho

- 前置：Miller-Rabin

```cpp
1  inline ll pollard_rho(ll x) {
2      auto f = [&](ll x, ll c, ll n) {
3          return ((__int128) x * x + c) % n;
4      };
5      ll s = 0, t = 0, c = 1ll * rand() % (x - 1) + 1;
6      ll stp = 0, goal = 1;
7      ll val = 1;
8      for(goal = 1;; goal <<= 1, s = t, val = 1) {
9          for(stp = 1; stp <= goal; ++stp) {
10             t = f(t, c, x);
11             val = (__int128)val * abs(t - s) % x;
12             if((stp % 127) == 0) {
13                 ll d = __gcd(val, x);
14                 if(d > 1)
15                     return d;
16             }
17         }
18         ll d = __gcd(val, x);
19         if(d > 1)
```

```
20          return d;
21      }
22  }
23
24  inline void get_factor_a(ll x, vector<ll> &fac) {
25      if(x < 2) return;
26      if(is_prime(x)) {
27          fac.push_back(x);
28          return;
29      }
30      ll p = x;
31      while(p >= x) p = pollard_rho(x);
32      while((x % p) == 0) x /= p;
33      get_factor_a(x, fac), get_factor_a(p, fac);
34  }
```

## 2.14  拆系数 FFT/MTT

- FFT，内含 MTT，比较灵活。

```
1   #define fp(i, a, b) for (int i = a, i##_ = (b) + 1; i < i##_; ++i)
2   #define fd(i, a, b) for (int i = a, i##_ = (b) - 1; i > i##_; --i)
3   using namespace std;
4   #define int long long
5   using ll = int64_t;
6   using db = double;
7   using Poly = vector<ll>;
8   struct cp {
9       db x, y;
10      cp(db real = 0, db imag = 0) : x(real), y(imag){};
11      cp operator+(cp b) const { return {x + b.x, y + b.y}; }
12      cp operator-(cp b) const { return {x - b.x, y - b.y}; }
13      cp operator*(cp b) const { return {x * b.x - y * b.y, x * b.y + y * b.x}; }
14  };
15  using vcp = vector<cp>;
16  const int mod = 1e9 + 7;
17  namespace FFT {
18      const db pi = acos(-1);
19      vcp Omega(int L) { // In order to reduce the accuracy error
20          vcp w(L); w[1] = 1;
21          for (int i = 2; i < L; i <<= 1) {
22              auto w0 = w.begin() + i / 2, w1 = w.begin() + i;
23              cp wn(cos(pi / i), sin(pi / i));
24              for (int j = 0; j < i; j += 2)
25                  w1[j] = w0[j >> 1], w1[j + 1] = w1[j] * wn;
26          }
27          return w;
28      }
29      auto W = Omega(1 << 23); // NOLINT
30      void DIF(cp *a, int n) {
31          cp x, y;
32          for (int k = n >> 1; k; k >>= 1)
```

```
33              for (int i = 0; i < n; i += k << 1)
34                  for (int j = 0; j < k; ++j)
35                      x = a[i + j], y = a[i + j + k],
36                      a[i + j + k] = (x - y) * W[k + j], a[i + j] = x + y;
37          }
38      void IDIT(cp *a, int n) {
39          cp x, y;
40          for (int k = 1; k < n; k <<= 1)
41              for (int i = 0; i < n; i += k << 1)
42                  for (int j = 0; j < k; ++j)
43                      x = a[i + j], y = a[i + j + k] * W[k + j],
44                      a[i + j + k] = x - y, a[i + j] = x + y;
45          const db Inv = 1. / n;
46          fp(i, 0, n - 1) a[i].x *= Inv, a[i].y *= Inv;
47          reverse(a + 1, a + n);
48      }
49  }
50  namespace Polynomial {
51      void DFT(vcp &a) { FFT::DIF(a.data(), a.size()); }
52      void IDFT(vcp &a) { FFT::IDIT(a.data(), a.size()); }
53      int norm(int n) { return 1 << (__lg(n - 1) + 1); }
54
55      // Poly mul
56      vcp &dot(vcp &a, vcp &b) { fp(i, 0, a.size() - 1) a[i] = a[i] * b[i]; return a;
            }
57      Poly &operator+=(Poly &a, Poly b) {
58          a.resize(max(a.size(), b.size()));
59          fp(i, 0, a.size() - 1) a[i] += b[i];
60          return a;
61      };
62      Poly &operator-=(Poly &a, Poly b) {
63          a.resize(max(a.size(), b.size()));
64          fp(i, 0, a.size() - 1) DEC(a[i], b[i]);
65          return a;
66      };
67      Poly operator * (Poly A, Poly B) {
68          int n = A.size() + B.size() - 1;
69          int L = norm(n);
70          Poly res(n); vcp a(L), b(L), c(L), d(L);
71          fp(i, 0, A.size() - 1) a[i] = cp(A[i] & 0x7fff, A[i] >> 15);
72          fp(i, 0, B.size() - 1) b[i] = cp(B[i] & 0x7fff, B[i] >> 15);
73          FFT::DIF(a.data(), L), FFT::DIF(b.data(), L);
74          for (int k = 1, i = 0, j = 0; k < L; j ^= k, k <<= 1)
75              for (; i < k * 2; i++, j = i ^ (k - 1)) {
76                  c[i] = cp(a[i].x + a[j].x, a[i].y - a[j].y) * b[i] * 0.5,
77                  d[i] = cp(a[i].y + a[j].y, -a[i].x + a[j].x) * b[i] * 0.5;
78              }
79          FFT::IDIT(c.data(), L), FFT::IDIT(d.data(), L);
80          for (int i = 0; i < n; i++) {
81              ll x = c[i].x + 0.5, y = c[i].y + 0.5, z = d[i].x + 0.5, w = d[i].y + 0.5;
82              x %= mod;
83              y %= mod;
84              z %= mod;
```

```
 85            w %= mod;
 86            res[i] = (x + ((y + z) << 15) % mod + (w << 30) % mod) % mod;
 87        }
 88        return res;
 89    }
 90    vcp operator * (vcp a, vcp b) {
 91        int n = a.size() + b.size() - 1;
 92        int L = norm(n);
 93        a.resize(L), b.resize(L);
 94        DFT(a), DFT(b);
 95        dot(a, b);
 96        IDFT(a);
 97        return a;
 98    }
 99 }
100 using namespace Polynomial;
```

## 2.15　多项式全家桶 (Number-Theoretic-Transform)

- 注意调整原根 g，模数 mod，N 开 3 到 4 倍数据范围，附录 A

- 注意 resize()

- 注意 Inv/Ln 的时候常数项不能为 0

- 注意 Exp 的时候常数项必须是 0

- 注意这里面的 ksm() 第三个参数是初值而不是模数

```
 1 #define fp(i, a, b) for (int i = (a); i <= (b); i++)
 2 #define fd(i, a, b) for (int i = (a); i >= (b); i--)
 3 const int N = 3e5 + 5, mod = 998244353; // (N = 4 * n)
 4
 5 using ll = int64_t;
 6 using Poly = vector<int>;
 7 /*-----------------------------------------------------------------------*/
 8 // 二次剩余
 9 class Cipolla {
10     int mod, I2{};
11     using pll = pair<ll, ll>;
12 #define X first
13 #define Y second
14     ll MUL(ll a, ll b) const {
15         return a * b % mod;
16     }
17     pll MUL(pll a, pll b) const {
18         return {(a.X * b.X + I2 * a.Y % mod * b.Y) % mod, (a.X * b.Y + a.Y * b.X) %
19             mod};
20     }
21     template<class T> T ksm(T a, int b, T x) {
22         for(; b; b >>= 1, a = MUL(a, a)) if(b & 1) x = MUL(x, a);
23         return x;
24     }
25 public:
```

```
25      Cipolla(int p = 0) : mod(p) {}
26      pair<int, int> sqrt(int n) {
27          int a = rand(), x;
28          if(!(n %= mod)) return {0, 0};
29          if(ksm(n, (mod - 1) >> 1, 1) == mod - 1) return {-1, -1};
30          while(ksm(I2 = ((ll) a * a - n + mod) % mod, (mod - 1) >> 1, 1) == 1) a =
                rand();
31          x = (int) ksm(pll{a, 1}, (mod + 1) >> 1, {1, 0}).X;
32          if(2 * x > mod) x = mod - x;
33          return {x, mod - x};
34      }
35  #undef X
36  #undef Y
37  };
38  /*------------------------------------Modular-----------------------------------*/
39  #define MUL(a, b) (ll(a) * (b) % mod)
40  #define ADD(a, b) (((a) += (b)) >= mod ? (a) -= mod : 0) // (a += b) %= P
41  #define DEC(a, b) (((a) -= (b)) < 0 ? (a) += mod: 0) // ((a -= b) += P) %= P
42  Poly getInv(int L) {
43      Poly inv(L);
44      inv[1] = 1;
45      fp(i, 2, L - 1) inv[i] = MUL((mod - mod / i), inv[mod % i]);
46      return inv;
47  }
48  int ksm(ll a, int b = mod - 2, ll x = 1) {
49      for(; b; b >>= 1, a = a * a % mod) if(b & 1) x = x * a % mod;
50      return x;
51  }
52  auto inv = getInv(N); // NOLINT
53  /*----------------------------------NTT----------------------------------------*/
54  namespace NTT {
55  const int g = 3;
56  Poly Omega(int L) {
57      int wn = ksm(g, mod / L);
58      Poly w(L);
59      w[L >> 1] = 1;
60      fp(i, L / 2 + 1, L - 1) w[i] = MUL(w[i - 1], wn);
61      fd(i, L / 2 - 1, 1) w[i] = w[i << 1];
62      return w;
63  }
64  auto W = Omega(1 << 20); // NOLINT
65  void DIF(int *a, int n) {
66      for(int k = n >> 1; k; k >>= 1)
67          for(int i = 0, y; i < n; i += k << 1)
68              for(int j = 0; j < k; ++j)
69                  y = a[i + j + k], a[i + j + k] = MUL(a[i + j] - y + mod, W[k + j]), ADD
                        (a[i + j], y);
70  }
71  void IDIT(int *a, int n) {
72      for(int k = 1; k < n; k <<= 1)
73          for(int i = 0, x, y; i < n; i += k << 1)
74              for(int j = 0; j < k; ++j)
75                  x = a[i + j], y = MUL(a[i + j + k], W[k + j]),
```

```
 76              a[i + j + k] = x - y < 0 ? x - y + mod : x - y, ADD(a[i + j], y);
 77      int Inv = mod - (mod - 1) / n;
 78      fp(i, 0, n - 1) a[i] = MUL(a[i], Inv);
 79      reverse(a + 1, a + n);
 80  }
 81  }
 82  /*---------------------------------Polynomial 全家桶
        --------------------------------*/
 83  namespace Polynomial {
 84      // basic operator
 85      int norm(int n) {
 86          return 1 << (__lg(n - 1) + 1);
 87      }
 88      void norm(Poly &a) {
 89          if(!a.empty()) a.resize(norm(a.size()), 0);
 90          else a = {0};
 91      }
 92      void DFT(Poly &a) {
 93          NTT::DIF(a.data(), a.size());
 94      }
 95      void IDFT(Poly &a) {
 96          NTT::IDIT(a.data(), a.size());
 97      }
 98      Poly &dot(Poly &a, Poly &b) {
 99          fp(i, 0, a.size() - 1) a[i] = MUL(a[i], b[i]);
100          return a;
101      }
102
103      // MUL / div int
104      Poly &operator*=(Poly &a, int b) {
105          for(auto &x : a) x = MUL(x, b);
106          return a;
107      }
108      Poly operator*(Poly a, int b) {
109          return a *= b;
110      }
111      Poly operator*(int a, Poly b) {
112          return b * a;
113      }
114      Poly &operator/=(Poly &a, int b) {
115          return a *= ksm(b);
116      }
117      Poly operator/(Poly a, int b) {
118          return a /= b;
119      }
120
121      // Poly ADD / sub
122      Poly &operator+=(Poly &a, Poly b) {
123          a.resize(max(a.size(), b.size()));
124          fp(i, 0, b.size() - 1) ADD(a[i], b[i]);
125          return a;
126      }
127      Poly operator+(Poly a, Poly b) {
```

```
128        return a += b;
129    }
130    Poly &operator-=(Poly &a, Poly b) {
131        a.resize(max(a.size(), b.size()));
132        fp(i, 0, b.size() - 1) DEC(a[i], b[i]);
133        return a;
134    }
135    Poly operator-(Poly a, Poly b) {
136        return a -= b;
137    }
138
139    // Poly MUL
140    Poly operator*(Poly a, Poly b) {
141        int n = (int) a.size() + b.size() - 1, L = norm(n);
142        if(a.size() <= 8 || b.size() <= 8) {
143            Poly c(n);
144            fp(i, 0, (int) a.size() - 1) fp(j, 0, (int) b.size() - 1)
145            c[i + j] = (c[i + j] + (ll) a[i] * b[j]) % mod;
146            return c;
147        }
148        a.resize(L), b.resize(L);
149        DFT(a), DFT(b), dot(a, b), IDFT(a);
150        return a.resize(n), a;
151    }
152
153    // Poly inv
154    Poly Inv2k(Poly a) { // |a| = 2 ^ k
155        int n = a.size(), m = n >> 1;
156        if(n == 1) return {ksm(a[0])};
157        Poly b = Inv2k(Poly(a.begin(), a.begin() + m)), c = b;
158        b.resize(n), DFT(a), DFT(b), dot(a, b), IDFT(a);
159        fp(i, 0, n - 1) a[i] = i < m ? 0 : mod - a[i];
160        DFT(a), dot(a, b), IDFT(a);
161        return move(c.begin(), c.end(), a.begin()), a;
162    }
163    Poly Inv(Poly a) {
164        int n = a.size();
165        norm(a), a = Inv2k(a);
166        return a.resize(n), a;
167    }
168
169    // Poly div / mod
170    Poly operator/(Poly a, Poly b) {
171        int k = a.size() - b.size() + 1;
172        if(k < 0) return {0}; // TODO: 这里是否应该是 k<=0??
173        reverse(a.begin(), a.end());
174        reverse(b.begin(), b.end());
175        b.resize(k), a = a * Inv(b);
176        a.resize(k), reverse(a.begin(), a.end());
177        return a;
178    }
179    pair<Poly, Poly> operator%(Poly a, const Poly& b) {
180        Poly c = a / b;
```

```
181          a -= b * c, a.resize(b.size() - 1);
182          return {c, a};
183      }
184
185      // Poly calculus
186      Poly deriv(Poly a) {
187          fp(i, 1, a.size() - 1) a[i - 1] = MUL(i, a[i]);
188          return a.pop_back(), a;
189      }
190      Poly integ(Poly a) {
191          a.push_back(0);
192          fd(i, a.size() - 1, 1) a[i] = MUL(inv[i], a[i - 1]);
193          return a[0] = 0, a;
194      }
195
196      // Poly ln
197      Poly Ln(Poly a) {
198          int n = a.size();
199          a = deriv(a) * Inv(a);
200          return a.resize(n - 1), integ(a);
201      }
202
203      // Poly exp
204      Poly Exp(Poly a) {
205          int n = a.size(), k = norm(n);
206          Poly b = {1}, c, d;
207          a.resize(k);
208          for(int L = 2; L <= k; L <<= 1) {
209              d = b, b.resize(L), c = Ln(b), c.resize(L);
210              fp(i, 0, L - 1) c[i] = a[i] - c[i] + (a[i] < c[i] ? mod : 0);
211              ADD(c[0], 1), DFT(b), DFT(c), dot(b, c), IDFT(b);
212              move(d.begin(), d.end(), b.begin());
213          }
214          return b.resize(n), b;
215      }
216
217      // Poly sqrt
218      Poly Sqrt(Poly a) {
219          int n = a.size(), k = norm(n);
220          a.resize(k);
221          Poly b = {(new Cipolla(mod))->sqrt(a[0]).first, 0}, c;
222          for(int L = 2; L <= k; L <<= 1) {
223              b.resize(L), c = Poly(a.begin(), a.begin() + L) * Inv2k(b);
224              fp(i, L / 2, L - 1) b[i] = MUL(c[i], (mod + 1) / 2);
225          }
226          return b.resize(n), b;
227      }
228
229      // Poly pow
230      Poly Pow(Poly &a, int b) {
231          return Exp(Ln(a) * b); // a[0] = 1
232      }
233      Poly Pow(Poly a, int b1, int b2) { // b1 = b % mod, b2 = b % phi(mod) and b >= n
```

```
          iff a[0] > 0
234     int n = a.size(), d = 0, k;
235     while(d < n && !a[d]) ++d;
236     if((ll) d * b1 >= n) return Poly(n);
237     a.erase(a.begin(), a.begin() + d);
238     k = ksm(a[0]), norm(a *= k);
239     a = Pow(a, b1) * ksm(k, mod - 1 - b2);
240     a.resize(n), d *= b1;
241     fd(i, n - 1, 0) a[i] = i >= d ? a[i - d] : 0;
242     return a;
243 }
244
245 // a0, a1,..., a_{k-1}, f1, f2, ..., fk
246 // fi = sum{i=1,k} fi*a[i-k]
247 // a 是值, f 是递推系数
248 // 常系数齐次线性递推
249 int LinearRecursion(vector<int> &a, vector<int> &f, int n) {
250     int k = a.size();
251     Poly t(k + 1);
252     for(int i = 0 ; i < k; i++) {
253         t[i] = (mod - f[k - i]) % mod;
254     }
255     t[k] = 1;
256     Poly g{0, 1}, ret{1};
257     while(n) {
258         if(n & 1) ret = (ret * g % t).second;
259         g = (g * g % t).second;
260         n >>= 1;
261     }
262     int ans = 0;
263     for(int i = 0; i < k; i++) {
264         ADD(ans, ret[i] * a[i] % mod);
265     }
266     return ans;
267 }
268 Poly mulT(Poly a, Poly b) {
269     int n = a.size(), m = b.size();
270     reverse(b.begin(), b.end());
271     a = a * b;
272     for(int i = 0; i < n; i++) {
273         a[i] = a[i + m - 1];
274     }
275     a.resize(n);
276     return a;
277 }
278 // x是点, a是多项式
279 // x0,x1,...,xn-1
280 vector<int> eval(Poly a, vector<int> x) {
281     int m = max(x.size(), a.size());
282     vector<int> ans(x.size());
283     x.resize(m + 1);
284     vector<Poly> divd(4 * m);
285     function<void(int, int, int)> divide_mul = [&](int l, int r, int id) -> void
```

```
                {
286             if(l == r) {
287                 divd[id] = Poly{1, (mod - x[l]) % mod};
288                 return;
289             }
290             int mid = l + r >> 1;
291             divide_mul(l, mid, id << 1), divide_mul(mid + 1, r, id << 1 | 1);
292             divd[id] = divd[id << 1 | 1] * divd[id << 1];
293         };
294         function<void(int, int, int, Poly)> getans = [&](int l, int r, int id, Poly
                now) -> void {
295             if(l == r) {
296                 if(l < ans.size()) {
297                     ans[l] = now[0];
298                 }
299                 return;
300             }
301             int mid = l + r >> 1;
302             now.resize(r - l + 2);
303             getans(l, mid, id << 1, mulT(now, divd[id << 1 | 1]));
304             getans(mid + 1, r, id << 1 | 1, mulT(now, divd[id << 1]));
305         };
306         divide_mul(0, m - 1, 1);
307         a.resize(m);
308         getans(0, m - 1, 1, mulT(a, Inv(divd[1])));
309         return ans;
310     }
311     // x1,x2,...,xn, y1,y2,...,yn
312     Poly Interpolation(vector<int> x, vector<int> y) {
313         int n = x.size() - 1;
314         vector<int> ans(n + 1);
315         vector<Poly> divd(4 * n);
316         auto divide_mul = [&](auto divide_mul, int l, int r, int id) -> void {
317             if(l == r) {
318                 divd[id] = Poly({mod - x[l], 1});
319                 return;
320             }
321             int mid = l + r >> 1;
322             divide_mul(divide_mul, l, mid, id << 1), divide_mul(divide_mul, mid + 1, r
                , id << 1 | 1);
323             divd[id] = divd[id << 1 | 1] * divd[id << 1];
324         };
325         auto getans = [&](auto getans, int l, int r, int id, Poly now) -> void {
326             if(l == r) {
327                 ans[l] = now[0];
328                 return;
329             }
330             int mid = l + r >> 1;
331             // Poly left = now % divd[id<<1];
332             // Poly right = now % divd[id<<1|1];
333             getans(getans, l, mid, id<<1, (now % divd[id << 1]).second);
334             getans(getans, mid + 1, r, id<<1|1, (now % divd[id << 1 | 1]).second);
335         };
```

```
336          divide_mul(divide_mul, 1, n, 1);
337          Poly a = deriv(divd[1]);
338          getans(getans, 1, n, 1, a);
339          auto get_poly = [&](auto get_poly, int l, int r, int id) -> Poly {
340              if(l == r) {
341                  Poly ret = { (int) (1ll * y[l] * ksm(ans[l], mod - 2) % mod) };
342                  return ret;
343              }
344              int mid = l + r >> 1;
345              return get_poly(get_poly, l, mid, id<<1) * divd[id<<1|1] + get_poly(
                     get_poly, mid + 1, r, id<<1|1) * divd[id<<1];
346          };
347          return get_poly(get_poly, 1, n, 1);
348      }
349  }
350  using namespace Polynomial;
```

- 简约版全家桶:

```
1   #define fp(i, a, b) for (int i = (a); i <= (b); i++)
2   #define fd(i, a, b) for (int i = (a); i >= (b); i--)
3   const int N = 3e5 + 5, mod = 998244353; // (N = 4 * n)
4
5   using ll = int64_t;
6   using Poly = vector<int>;
7   /*-----------------------------------Modular---------------------------------*/
8   #define MUL(a, b) (ll(a) * (b) % mod)
9   #define ADD(a, b) (((a) += (b)) >= mod ? (a) -= mod : 0) // (a += b) %= P
10  #define DEC(a, b) (((a) -= (b)) < 0 ? (a) += mod: 0) // ((a -= b) += P) %= P
11  Poly getInv(int L) {
12      Poly inv(L);
13      inv[1] = 1;
14      fp(i, 2, L - 1) inv[i] = MUL((mod - mod / i), inv[mod % i]);
15      return inv;
16  }
17  int ksm(ll a, int b = mod - 2, ll x = 1) {
18      for(; b; b >>= 1, a = a * a % mod) if(b & 1) x = x * a % mod;
19      return x;
20  }
21  auto inv = getInv(N); // NOLINT
22  /*-----------------------------------NTT-------------------------------------*/
23  namespace NTT {
24  const int g = 3;
25  Poly Omega(int L) {
26      int wn = ksm(g, mod / L);
27      Poly w(L);
28      w[L >> 1] = 1;
29      fp(i, L / 2 + 1, L - 1) w[i] = MUL(w[i - 1], wn);
30      fd(i, L / 2 - 1, 1) w[i] = w[i << 1];
31      return w;
32  }
33  auto W = Omega(1 << 20); // NOLINT
34  void DIF(int *a, int n) {
```

```
35        for(int k = n >> 1; k; k >>= 1)
36            for(int i = 0, y; i < n; i += k << 1)
37                for(int j = 0; j < k; ++j)
38                    y = a[i + j + k], a[i + j + k] = MUL(a[i + j] - y + mod, W[k + j]), ADD
                          (a[i + j], y);
39 }
40 void IDIT(int *a, int n) {
41        for(int k = 1; k < n; k <<= 1)
42            for(int i = 0, x, y; i < n; i += k << 1)
43                for(int j = 0; j < k; ++j)
44                    x = a[i + j], y = MUL(a[i + j + k], W[k + j]),
45                    a[i + j + k] = x - y < 0 ? x - y + mod : x - y, ADD(a[i + j], y);
46        int Inv = mod - (mod - 1) / n;
47        fp(i, 0, n - 1) a[i] = MUL(a[i], Inv);
48        reverse(a + 1, a + n);
49 }
50 }
51 /*----------------------------------Polynomial 全家桶
      --------------------------------*/
52 namespace Polynomial {
53     // basic operator
54     int norm(int n) {
55         return 1 << (__lg(n - 1) + 1);
56     }
57     void norm(Poly &a) {
58         if(!a.empty()) a.resize(norm(a.size()), 0);
59         else a = {0};
60     }
61     void DFT(Poly &a) {
62         NTT::DIF(a.data(), a.size());
63     }
64     void IDFT(Poly &a) {
65         NTT::IDIT(a.data(), a.size());
66     }
67     Poly &dot(Poly &a, Poly &b) {
68         fp(i, 0, a.size() - 1) a[i] = MUL(a[i], b[i]);
69         return a;
70     }
71
72     // MUL / div int
73     Poly &operator*=(Poly &a, int b) {
74         for(auto &x : a) x = MUL(x, b);
75         return a;
76     }
77     Poly operator*(Poly a, int b) {
78         return a *= b;
79     }
80     Poly operator*(int a, Poly b) {
81         return b * a;
82     }
83     Poly &operator/=(Poly &a, int b) {
84         return a *= ksm(b);
85     }
```

```
 86     Poly operator/(Poly a, int b) {
 87         return a /= b;
 88     }
 89
 90     // Poly ADD / sub
 91     Poly &operator+=(Poly &a, Poly b) {
 92         a.resize(max(a.size(), b.size()));
 93         fp(i, 0, b.size() - 1) ADD(a[i], b[i]);
 94         return a;
 95     }
 96     Poly operator+(Poly a, Poly b) {
 97         return a += b;
 98     }
 99     Poly &operator-=(Poly &a, Poly b) {
100         a.resize(max(a.size(), b.size()));
101         fp(i, 0, b.size() - 1) DEC(a[i], b[i]);
102         return a;
103     }
104     Poly operator-(Poly a, Poly b) {
105         return a -= b;
106     }
107
108     // Poly MUL
109     Poly operator*(Poly a, Poly b) {
110         int n = a.size() + b.size() - 1, L = norm(n);
111         if(a.size() <= 8 || b.size() <= 8) {
112             Poly c(n);
113             fp(i, 0, a.size() - 1) fp(j, 0, b.size() - 1)
114             c[i + j] = (c[i + j] + (ll) a[i] * b[j]) % mod;
115             return c;
116         }
117         a.resize(L), b.resize(L);
118         DFT(a), DFT(b), dot(a, b), IDFT(a);
119         return a.resize(n), a;
120     }
121 }
122 using namespace Polynomial;
```

# 3 Math Formula

## 3.1 多项式牛顿迭代

## 3.2 牛顿恒等式

设 $p_k$ is k-th power sum, $p_k = \sum_{i=1}^{n} a_i^k$, $e_k$ 是 $k$ 个 $a$ 的轮换和。

设 $E(x) = 1 + e_1 x + e_2 x^2 + ....$

$$\prod_{i=1}^{n}(1 + a_i x) = E(x)$$

$$(\ln \prod_{i=1}^{n}(1 + a_i x))' = E'(x)/E(x)$$

$$\sum_{i=1}^{n} \frac{a_i}{1 + a_i x} = E'(x)/E(x)$$

$$E'(x)/E(x) = \sum_{i=1}^{n} \sum_{j \geq 0} (-1)^j a_i^{j+1} x^j$$

$$= \sum_{j \geq 0} (-1)^j \left( \sum_{i=1}^{n} a_i^{j+1} \right) x^j$$

$$= \sum_{j \geq 0} (-1)^j p_{j+1} x^j$$

结论：

$$ke_k = \sum_{i=1}^{k} (-1)^{i-1} e_{k-i} p_i \qquad (k \leq n)$$

$$0 = \sum_{i=k-n}^{k} (-1)^{i-1} e_{k-i} p_i \quad (k > n)$$

$$e_1 = p_1$$
$$2e_2 = e_1 p_1 - p_2 = p_1^2 - p_2$$
$$3e_3 = e_2 p_1 - e_1 p_2 + p_3 = \frac{1}{2} p_1^3 - \frac{3}{2} p_1 p_2 + p_3$$
$$4e_4 = e_3 p_1 - e_2 p_2 + e_1 p_3 - p_4 = \frac{1}{6} p_1^4 - p_1^2 p_2 + \frac{4}{3} p_1 p_3 + \frac{1}{2} p_2^2 - p_4,$$

$$e_k(k > n) = 0$$

$$p_1 = e_1$$
$$p_2 = e_1 p_1 - 2e_2 = e_1^2 - 2e_2$$
$$p_3 = e_1 p_2 - e_2 p_1 + 3e_3 = e_1^3 - 3e_1 e_2 + 3e_3$$
$$p_4 = e_1 p_3 - e_2 p_2 + e_3 p_1 - 4e_4 = e_1^4 - 4e_1^2 e_2 + 4e_1 e_3 + 2e_2^2 - 4e_4,$$
$$p_k = (-1)^{k-1} k e_k + \sum_{i=1}^{k-1} (-1)^{k-1+i} e_{k-i} (x_1, \ldots, x_n) p_i (k \leq n)$$
$$p_k = \sum_{i=k-n}^{k-1} (-1)^{k-1+i} e_{k-i} p_i (k > n)$$

## 3.3 生成函数/形式幂级数

### 3.3.1 麦克劳林展开

$$\frac{1}{1-x} = 1 + x + \cdots + x^n + \cdots$$

$$\frac{1}{1+x} = 1 - x + x^2 + \cdots + (-1)^n x^n + \cdots$$

$$\ln(1+x) = x - \frac{x^2}{2} + \cdots + \frac{(-1)^{n-1}x^n}{n} + \cdots$$

$$e^x = 1 + x + \cdots + \frac{x^n}{n!} + \cdots$$

$$(1+x)^\alpha = 1 + \alpha x + \frac{\alpha(\alpha-1)}{2!}x^2 + \cdots + \frac{\alpha(\alpha-1)\cdots(\alpha-n+1)}{n!} + \cdots$$

$$\sin x = \frac{x^3}{3!} + \cdots + \frac{(-1)^n x^{2n+1}}{(2n+1)!} + \cdots$$

$$\cos x = 1 - \frac{x^2}{2!} + \cdots + \frac{(-1)^n x^{2n}}{(2n)!} + \cdots$$

## 3.4 数论公式

### 3.4.1 莫比乌斯反演

一般反演：

$$f(n) = \sum_{d|n} g(d) \Longleftrightarrow g(n) = \sum_{d|n} \mu(d) f(\frac{n}{d})$$

$$f(n) = \sum_{\substack{n|d \\ d \le N}} g(d) \Longleftrightarrow g(n) = \sum_{\substack{n|d \\ d \le N}} \mu(\frac{d}{n}) f(d)$$

gcd 反演结论：

$$[\gcd(i,j) = 1] = \sum_{d|\gcd(i,j)} \mu(d)$$

### 3.4.2 杜教筛

$$g(1)S(n) = \sum_{i=1}^{n} (f * g)(i) - \sum_{i=2}^{n} g(i) S\left(\left\lfloor \frac{n}{i} \right\rfloor\right)$$

### 3.4.3 Min_25 筛/Ex-Eratosthenes-Sieve

第一步：筛出所有质数部分（此处的 $g(n)$ 是 $f(n), n \in \mathbb{P}$ 的通项公式）

$$G_k(n) := \sum_{i=1}^{n} [p_k < \text{lpf}(i) \vee \text{isprime}(i)] \, g(i)$$
$$G_k(n) = G_{k-1}(n) - \left[p_k^2 \le n\right] g(p_k) (G_{k-1}(n/p_k) - G_{k-1}(p_{k-1}))$$

第二部: 筛出质数和合数部分, 我们令 $F_{\mathrm{prime}}(n) = \sum\limits_{\substack{2 \le p \le n \\ p \in \mathrm{prime}}} f(p)$

$$
\begin{aligned}
F_k(n) &= \sum_{i=2}^{n} [p_k \le \mathrm{lpf}(i)] f(i) \\
&= \sum_{\substack{k \le i \\ p_i^2 \le n}} \sum_{\substack{c \ge 1 \\ p_i^c \le n}} f\left(p_i^c\right) \left([c > 1] + F_{i+1}\left(n/p_i^c\right)\right) + \sum_{\substack{k \le i \\ p_i \le n}} f(p_i) \\
&= \sum_{\substack{k \le i \\ p_i^2 \le n}} \sum_{\substack{c \ge 1 \\ p_i^c \le n}} f\left(p_i^c\right) \left([c > 1] + F_{i+1}\left(n/p_i^c\right)\right) + F_{\mathrm{prime}}(n) - F_{\mathrm{prime}}(p_{k-1}) \text{常用!!} \\
&= \sum_{\substack{k \le i \\ p_i^2 \le n}} \sum_{\substack{c \ge 1 \\ p_i^{c+1} \le n}} \left(f\left(p_i^c\right) F_{i+1}\left(n/p_i^c\right) + f\left(p_i^{c+1}\right)\right) + F_{\mathrm{prime}}(n) - F_{\mathrm{prime}}(p_{k-1})
\end{aligned}
$$

```cpp
// sieve first!!

namespace min25 {

ll n, sqr, w[N];
int c;
ll g[N];

inline int id(ll x) {
    return x ? x <= sqr ? c - x + 1 : n / x : 0;
}

void cal_g(ll n_) { // 这里的 g 只能是一个没有系数的单项式!! 不是极性函数f
    n = n_; sqr = sqrt(n_); c = 0;
    for (ll l = 1, r; l <= n; l = r + 1) {
        ll v = w[++c] = n / l; r = n / v;
        g[c] = (v - 1) % mod;
    }
    for (int i = 1; i <= prinum; i++) {
        int p = pri[i];
        if (1ll * p * p > n) break;
        for (int j = 1; 1ll * p * p <= w[j]; ++j)
            g[j] -= g[id(w[j] / p)] - g[id(p - 1)];
    }
}
int cal_s(int n, ll x, int y) { // 用 g 的单项式乘以系数加起来。binom那个地方是填f(p^e)
    if(x <= pri[y]) return 0;
    int ans = (g[id(x)] - g[id(pri[y])] + mod) * n % mod;

    for(int i = y + 1; i <= prinum && pri[i] * pri[i] <= x; i++) {
        ll P = pri[i];
        for(int e = 1; P <= x; e++, P *= pri[i]) {
            ans = (ans + (ll) binom(n + e - 1, n - 1) * (cal_s(n, x / P, i) + (e != 1)
                ) % mod) % mod;
        }
    }
    return ans;
}
```

39 | }

## 3.5　分配问题

1. $n$ 个球放到 $k$ 个盒子，每个盒子只有一种形态。

| $n$ 个球 | $k$ 个盒子 | 盒子可以为空 | 每个盒子内至少有一个球 |
|---|---|---|---|
| 有标号 | 有标号 | $k^n$ | $k!S_2(n,k)$ |
| 有标号 | 无标号 | $\sum_{i=1}^{k} S_2(n,i)$ | $S_2(n,k)$ |
| 无标号 | 有标号 | $C(n+k-1,k-1)$ | $C(n-1,k-1)$ |
| 无标号 | 无标号 | $p(n+k,k)$ | $p(n,k)$ |

其中 $S_2(n,k)$ 为第二类 'Stirling 数 ' ，$p(n,k)$ 为 '分拆数 '

2. $n$ 个球放到 $k$ 个盒子，每个盒子至少一个球，装有 $i$ 个球的盒子有 $f_i$ $(i \geq 1)$ 种形态。
   $F(x)$ 是 $f_i$ 的 $o.g.f.$ ，$E(x)$ 是 $f_i$ 的 $e.g.f.$

| $n$ 个球 | $k$ 个盒子 | 关于 $n$ 方案的生成函数 |
|---|---|---|
| 有标号 | 有标号 | $e.g.f = E(x)^k$ |
| 有标号 | 无标号 | $e.g.f = \frac{1}{k!}E(x)^k$ |
| 无标号 | 有标号 | $o.g.f = F(x)^k$ |
| 无标号 | 无标号 | 不会 |

3. $n$ 个球放到若干盒子，每个盒子至少一个球，装有 $i$ 个球的盒子有 $f_i$ $(i \geq 1)$ 种形态.

| $n$ 个球 | 盒子 | 方案的生成函数 |
|---|---|---|
| 有标号 | 有标号 | $e.g.f = \frac{1}{1-E(x)}$ |
| 有标号 | 无标号 | $e.g.f = \exp(E(x))$ |
| 无标号 | 有标号 | $o.g.f = \frac{1}{1-F(x)}$ |
| 无标号 | 无标号 | $o.g.f = \prod_{i \geq 1}\left(\frac{1}{1-x^i}\right)^{f_i} = \exp\left(\sum_{j \geq 1} \frac{1}{j}F\left(x^j\right)\right)$ |

## 3.6　第二类斯特林数

性质：

1. $\left\{{n \atop k}\right\} = \left\{{n-1 \atop k-1}\right\} + k\left\{{n-1 \atop k}\right\}$ 边界条件 $\left\{{x \neq 1 \atop 0}\right\} = 0, \left\{{0 \atop 0}\right\} = 1$

2. $\left\{{n \atop k}\right\} = \frac{1}{k!}\sum_{i=0}^{k}(-1)^i\binom{k}{i}(k-i)^n$ （用来求第二类斯特林数·行，考虑容斥）

3. $k^n = \sum_{i=0}^{k}\left\{{n \atop i}\right\}k^{\underline{i}}$ （性质 2 的反演）

4. 重要公式：$x^n = \sum_{k=0}^{n}\left\{{n \atop k}\right\}x^{\underline{k}}$ （基于 3，$x$ 个球 $n$ 个盒子）

5. 关于 $n$ 的 $e.g.f. = \dfrac{(e^x-1)^k}{k!}$ （考虑将 $n$ 个物品染成 $k$ 种颜色，每一种物品的指数生成函数为 $e^x - 1$ ）（用于求第二类斯特林数·列）

## 3.7　第一类斯特林数

性质：

1. $\left[\begin{matrix}n\\k\end{matrix}\right] = \left[\begin{matrix}n-1\\k-1\end{matrix}\right] + (n-1)\left[\begin{matrix}n-1\\k\end{matrix}\right]$ 边界条件 $\left[\begin{matrix}x\neq 1\\0\end{matrix}\right] = 0, \left[\begin{matrix}0\\0\end{matrix}\right] = 1$

2. 重要公式：$x^{\overline{n}} = \sum\limits_{k=0}^{n} \left[\begin{matrix}n\\k\end{matrix}\right] x^n$　（$x$ 个球 $n$ 个盒子）（用于计算第一类斯特林数 · 行）

3. 有符号的第一类斯特林数：$S_1(n,k) = (-1)^{n+k}\left[\begin{matrix}n\\k\end{matrix}\right]$

   $x^{\underline{n}} = \sum\limits_{k=0}^{n} S_1(n,k)x^n$

4. 关于 $n$ 的 $e.g.f. = \dfrac{(-\ln(1-x))^k}{k!}$. (列)

## 3.8　分拆数

性质：

1. $p(n,k) = p(n-1,k-1) + p(n-k,k)$

2. $o.g.f. = x^k \prod\limits_{i=1}^{k} \frac{1}{1-x^i}$　（考虑 Ferrers 图中长度为 $i$ 的一列有多少个）

## 3.9　五边形数

性质：

1. $p(n) = \sum\limits_{k=1}^{n} p(n,k)$

2. $o.g.f. = \prod\limits_{i\geq 1} \frac{1}{1-x^i}$

   考虑求 $\ln$ , $o.g.f. = \exp(\sum_{n\geq 1}\sum_{d|n}\frac{x^n}{d})$ $O(n\log n)$

3. 递推公式：

   (a) 考虑 $Q(x) = \prod_{i\geq 1}(1-x^i)$

   (b) $Q(x) = \sum_{n\geq 0}(q_{even}(n) - q_{odd}(n))x^n$　$q_{even/odd}$ 表示有偶数/奇数行，每一行的个数都不相同，大部分 $q_{even}(n)$ 和 $q_{odd}(n)$ 抵消，小部分也就是有 $b$ 行，$n = \frac{b(3b-1)}{2}$ 和 $n = \frac{b(3b+1)}{2}$ 无法抵消。其系数都是 $(-1)^b$

   (c) 则 $Q(x) = 1 + \sum_{i\geq 1}(-1)^i x^{\frac{(3i\pm 1)i}{2}}$

   (d) $P(x) = Q^{-1}(x)$ 则，$p(n) = \sum_{k\geq 1}(-1)^{k-1}\left(p\left(n - \frac{(3k-1)k}{2}\right) + p\left(n - \frac{(3k+1)k}{2}\right)\right)$

## 3.10　Polya

**置换群**

　　$G$ 是置换的集合，$\circ$ 是置换的复合，且 $(G,\circ)$ 为一个群时，称 $(G,\circ)$ 为一个置换群。

旋转群：

　　设 $n$ 元环的 $n$ 个结点分别为 $a_1, a_2, ..., a_n$，旋转操作可以看成 $A = a_1, ..., a_n$ 上的 $n$ 个置换，其中第 $i$ 个置换为 $g_i = [i+1, i+2, ..., n, 1, ..., i]$。

　　设集合 $G = \{g_0, g_1, ..., g_{n-1}\}$，则 $(G,\circ)$ 是一个置换群，称为正 $n$ 边形的旋转群。

**群对集合的作用**

一个操作会将一个对象改变为另一个对象，形式化地：

设 $(G, \circ)$ 是一个群，其单位元为 $e$，$X$ 是一个集合，群 $G$ 对集合 $X$ 的一个作用是一个 $G \times X$ 到 $X$ 的映射 $f$，满足：

- $\forall x \in X. f(e, x) = x$

- $\forall g, h \in G. f(h \circ g, x) = f(h, f(g, x))$ 我们把 $f(g, x)$ 简记成 $g_f(x)$ 或（在没有歧义的情况下记成）$g(x)$ 。

- 设 $n$ 元环的 $n$ 个结点分别为 $a_1, a_2, \ldots, a_n$，令 $A = \{a_1, \ldots, a_n\}$ 。设颜色集合为 $B = \{b_1, b_2, \ldots, b_m\}$ 。给 $n$ 元环染色可以看成 $A$ 到 $B$ 的一个映射 $x : A \to B$，令 $X$ 是所有这些映射的集合，即 $X = \{x \mid x : A \to B\}$ 。

- 令 $G = \{g_0, g_1, \ldots, g_{n-1}\}$ 是正 $n$ 边形的旋转群，定义 $G \times X$ 到集合 $X$ 的映射 $f$，其中 $\forall i = 0..n-1, x \in X. y = f(g_i, x)$ 满足 $\forall j = 1..n. y(a_j) = x(g_i(a_j))$，可以证明 $f$ 是群 $G$ 到集合 $X$ 的一个作用，因此我们简记 $y = f(g_i, x)$ 为 $y = g_i(x)$ 。

- $X$ 上的 $G$ 关系为 $R_G = \{(x, y) \mid x, y \in X \wedge (\exists g \in G. y = g(x))\}$，$x R_G y$ 当且仅当染色方案 $x$ 能通过旋转得到 $y$ 。

要求所有不同的染色方案，即是求 $X$ 上的 $G$-轨道的数量。

**Burnside 引理**

设有限群 $(G, \circ)$ 作用在有限集 $X$ 上，则 $X$ 上的 $G$-轨道数量为

$$N = \frac{1}{|G|} \sum_{g \in G} \Psi(g)$$

其中 $\Psi(g)$ 表示 $g(x) = x$ 的 $x$ 的数量。

**轮换指标**

设 $(G, \circ)$ 是一个 $n$ 元置换的置换群，它的轮换指标为

$$P_G(x_1, x_2, \ldots, x_n) = \frac{1}{|G|} \sum_{g \in G} x_1^{b_1} x_2^{b_2} \ldots x_n^{b_n}$$

$x_i^{b_i}$ 表示 $g$ 这个置换的长度为 $i$ 的环有 $b_i$ 个。

正 $n$ 边形旋转群轮换指标：

$$P_G = \frac{1}{n} \sum_{d \mid n} \varphi(d) x_d^{n/d}$$

正 $n$ 边形二面体群轮换指标 (即可对称):

$$P_G = \frac{1}{2n} \sum_{d \mid n} \varphi(d) x_d^{n/d} + \begin{cases} \frac{1}{2} x_1 x_2^{\frac{n-1}{2}}, & n \text{ 为奇数} \\ \frac{1}{4} \left( x_2^{\frac{n}{2}} + x_1^2 x_2^{\frac{n-2}{2}} \right), & n \text{ 为偶数} \end{cases}$$

正方体置换群：

顶点置换群：

$$P_G = \frac{1}{24} \left( x_1^8 + 8 x_1^2 x_3^2 + 9 x_2^4 + 6 x_4^2 \right)$$

边置换群:

$$P_G = \frac{1}{24} \left( x_1^{12} + 8x_3^4 + 6x_1^2 x_2^5 + 3x_2^6 + 6x_4^3 \right)$$

面置换群:

$$P_G = \frac{1}{24} \left( x_1^6 + 8x_3^2 + 6x_2^3 + 3x_1^2 x_2^2 + 6x_1^2 x_4 \right)$$

**Polya 定理**

集合 $X$ 可以看成是给集合 $A = \{a_1, a_2, \ldots, a_n\}$ 的每个元素赋予式样（颜色, 种类等）的映射的集合

引入表示式样的集合 $B$, 令 $X = \{x \mid x : A \to B\}$, 记为 $B^A$

**式样清单**: $G$ 作用在 $B^A$ 上的 $G$-轨道的集合称为 $B^A$ 关于 $G$ 的**式样清单**, 记为 $F$

种类的权值: 假设 $B$ 上的每个元素 $b$ 都赋予了权值 $w(b)$

$f \in B^A$ 的权值: 定义 $w(f) := \prod_{a \in A} w(f(a))$

$G$-轨道的权值: $w(F) := w(f)$, 任选一个 $f \in F$

定理:

$B^A$ 关于 $G$ 的**式样清单**记为 $\mathcal{F}$, 则

$$\sum_{F \in \mathcal{F}} w(F) = P_G \left( \sum_{b \in B} w(b), \sum_{b \in B} w(b)^2, \ldots, \sum_{b \in B} w(b)^n \right)$$

# 4  Misc

## 4.1  德州扑克

```cpp
#include <bits/stdc++.h>
using namespace std;
struct card{
  char suit;
  int rank;
  card(){
  }
  bool operator <(card C){
    return rank < C.rank || rank == C.rank && suit < C.suit;
  }
};
istream& operator >>(istream& is, card& C){
  string S;
  is >> S;
  if (S[0] == 'A'){
    C.rank = 14;
  } else if (S[0] == 'K'){
    C.rank = 13;
  } else if (S[0] == 'Q'){
    C.rank = 12;
  } else if (S[0] == 'J'){
    C.rank = 11;
  } else if (S[0] == 'T'){
    C.rank = 10;
  } else {
    C.rank = S[0] - '0';
  }
  C.suit = S[1];
  return is;
}
vector<int> hand(vector<card> C){
  sort(C.begin(), C.end());
  set<char> suits;
  for (int i = 0; i < 5; i++){
    suits.insert(C[i].suit);
  }
  if (suits.size() == 1 && C[4].rank - C[0].rank == 4){
    if (C[4].rank == 14){
      return vector<int>{9};
    } else {
      return vector<int>{8, C[4].rank};
    }
  }
  if (suits.size() == 1 && C[3].rank == 5 && C[4].rank == 14){
    return vector<int>{8, 5};
  }
  if (C[0].rank == C[3].rank){
    return vector<int>{7, C[0].rank, C[4].rank};
  }
```

```
50    if (C[1].rank == C[4].rank){
51      return vector<int>{7, C[1].rank, C[0].rank};
52    }
53    if (C[0].rank == C[2].rank && C[3].rank == C[4].rank){
54      return vector<int>{6, C[0].rank, C[3].rank};
55    }
56    if (C[2].rank == C[4].rank && C[0].rank == C[1].rank){
57      return vector<int>{6, C[2].rank, C[0].rank};
58    }
59    if (suits.size() == 1){
60      return vector<int>{5, C[4].rank, C[3].rank, C[2].rank, C[1].rank, C[0].rank};
61    }
62    if (C[1].rank - C[0].rank == 1 && C[2].rank - C[1].rank == 1 && C[3].rank - C[2].
          rank == 1 && C[4].rank - C[3].rank == 1){
63      return vector<int>{4, C[4].rank};
64    }
65    if (C[0].rank == 2 && C[1].rank == 3 && C[2].rank == 4 && C[3].rank == 5 && C[4].
          rank == 14){
66      return vector<int>{4, 5};
67    }
68    if (C[0].rank == C[2].rank){
69      return vector<int>{3, C[0].rank, C[4].rank, C[3].rank};
70    }
71    if (C[1].rank == C[3].rank){
72      return vector<int>{3, C[1].rank, C[4].rank, C[0].rank};
73    }
74    if (C[2].rank == C[4].rank){
75      return vector<int>{3, C[2].rank, C[1].rank, C[0].rank};
76    }
77    if (C[0].rank == C[1].rank && C[2].rank == C[3].rank){
78      return vector<int>{2, C[2].rank, C[0].rank, C[4].rank};
79    }
80    if (C[0].rank == C[1].rank && C[3].rank == C[4].rank){
81      return vector<int>{2, C[3].rank, C[0].rank, C[2].rank};
82    }
83    if (C[1].rank == C[2].rank && C[3].rank == C[4].rank){
84      return vector<int>{2, C[3].rank, C[1].rank, C[0].rank};
85    }
86    if (C[0].rank == C[1].rank){
87      return vector<int>{1, C[0].rank, C[4].rank, C[3].rank, C[2].rank};
88    }
89    if (C[1].rank == C[2].rank){
90      return vector<int>{1, C[1].rank, C[4].rank, C[3].rank, C[0].rank};
91    }
92    if (C[2].rank == C[3].rank){
93      return vector<int>{1, C[2].rank, C[4].rank, C[1].rank, C[0].rank};
94    }
95    if (C[3].rank == C[4].rank){
96      return vector<int>{1, C[3].rank, C[2].rank, C[1].rank, C[0].rank};
97    }
98    return vector<int>{0, C[4].rank, C[3].rank, C[2].rank, C[1].rank, C[0].rank};
99 }
100 int dfs(vector<vector<int>> &alice, vector<vector<int>> &bob, int a, int b, int p){
```

```
101    if (p == 6){
102      if (alice[a] > bob[b]){
103        return 1;
104      } else if (alice[a] < bob[b]){
105        return -1;
106      } else {
107        return 0;
108      }
109    } else {
110      int mx = -1;
111      for (int i = 0; i < 6; i++){
112        if ((a >> i & 1) == 0 && (b >> i & 1) == 0){
113          if (p % 2 == 0){
114            mx = max(mx, -dfs(alice, bob, a | (1 << i), b, p + 1));
115          } else {
116            mx = max(mx, -dfs(alice, bob, a, b | (1 << i), p + 1));
117          }
118        }
119      }
120      return mx;
121    }
122  }
123  int main(){
124    int T;
125    cin >> T;
126    for (int i = 0; i < T; i++){
127      vector<card> a(2);
128      cin >> a[0] >> a[1];
129      vector<card> b(2);
130      cin >> b[0] >> b[1];
131      vector<card> c(6);
132      cin >> c[0] >> c[1] >> c[2] >> c[3] >> c[4] >> c[5];
133      vector<vector<int>> alice(1 << 6), bob(1 << 6);
134      for (int j = 0; j < (1 << 6); j++){
135        if (__builtin_popcount(j) == 3){
136          vector<card> ha = {a[0], a[1]};
137          vector<card> hb = {b[0], b[1]};
138          for (int k = 0; k < 6; k++){
139            if ((j >> k & 1) == 1){
140              ha.push_back(c[k]);
141              hb.push_back(c[k]);
142            }
143          }
144          alice[j] = hand(ha);
145          bob[j] = hand(hb);
146        }
147      }
148      int ans = dfs(alice, bob, 0, 0, 0);
149      if (ans == 1){
150        cout << "Alice" << endl;
151      } else if (ans == -1){
152        cout << "Bob" << endl;
153      } else {
```

```
154        cout << "Draw" << endl;
155      }
156   }
157 }
```

## 4.2   Expression Parser

```cpp
#include<bits/stdc++.h>
using namespace std;
// 运算符
template<typename T>
/// @brief 二元运算符
/// @tparam a, b, sg
/// @return a (sg) b
function<T(T, T, char)> ca = [&] (T a, T b, char sg) {
    // 操作取模在这里改
    if(sg == '+') return a + b;
    if(sg == '-') return a - b;
    if(sg == '*') return a * b;
    if(sg == '/') return a / b;
    return (T)0;
};
// 读入取模在这里改
/// @param 要读入的字符串s 和当前指针pt，都要是引用类型
// 最后pt停在数字的最后一位
function<int(string&, int&)> readint = [&] (string &s, int &pt) {
    int ret = 0, fl = 1;
    if(s[pt] == '-') fl = -1, pt++;
    while(pt < s.length() && isdigit(s[pt])) {
        ret = ret * 10 + s[pt++] - '0';
    }
    pt--;
    return ret * fl;
};
function<double(string&, int&)> readdouble = [&] (string &s, int &pt) {
    int fl = 1;
    string ret;
    if(s[pt] == '-') fl = -1, pt++;
    while(pt < s.length() && (isdigit(s[pt]) || s[pt] == '.')) {
        ret += s[pt++];
    }
    pt--;
    return stod(ret) * fl;
};

// priority['('] = 0;
template<typename T>
T calc(string s, map<char, int> priority, function<T(T, T, char)> calculate,
       function<T(string&, int&)> readnum) {
    int pt = 0;
    string tmp;
    for(int i = 0; i < s.length(); i++) if(s[i] != ' ') tmp += s[i];
```

```cpp
        s = tmp;
        vector<T> stk;
        vector<char> sgn;
        auto calc_single = [&] () {
            char sg = sgn.back();
            sgn.pop_back();
            T v = stk.back();
            stk.pop_back();
            T u = stk.back();
            stk.pop_back();
            stk.push_back(calculate(u, v, sg));
        };
        int mode = 0;
        // 0 is number, 1 is sign
        while(pt < s.length()) {
            if(mode == 0) {
                if(isdigit(s[pt]) || s[pt] == '-') {
                    stk.push_back(readnum(s, pt));// 读数字
                    mode = 1;
                } else if(s[pt] == '(') {
                    sgn.push_back(s[pt]);
                }
            } else {
                if(s[pt] != ')') {
                    while(sgn.size() && priority[sgn.back()] >= priority[s[pt]]) {
                        calc_single();
                    }
                    sgn.push_back(s[pt]);
                    mode = 0;
                } else if(s[pt] == ')') {
                    while(sgn.back() != '(') {
                        calc_single();
                    }
                    sgn.pop_back();
                }
            }
            pt++;
        }
        while(sgn.size()) {
            calc_single();
        }
        assert(stk.size() == 1);
        return stk[0];
};

int main() {
    // 设定优先级
    map<char, int> prio;
    prio['+'] = prio['-'] = 1;
    prio['*'] = 2;
    prio['/'] = 2;
    string s; cin >> s;
    cout << calc<double>(s, prio, ca<double>, readdouble);
```

```
98    return 0;
99  }
```

## 4.3 原根表

```
// 原根表: p = r * 2^k + g
prime r k g
3 1 1 2
5 1 2 2
17 1 4 3
97 3 5 5
193 3 6 5
257 1 8 3
7681 15 9 17
12289 3 12 11
40961 5 13 3
65537 1 16 3
786433 3 18 10
5767169 11 19 3
7340033 7 20 3
23068673 11 21 3
104857601 25 22 3
167772161 5 25 3
469762049 7 26 3
998244353 119 23 1
1004535809 479 21 3
2013265921 15 27 31
2281701377 17 27 3
3221225473 3 30 5
75161927681 35 31 3
77309411329 9 33 7
206158430209 3 36 22
2061584302081 15 37 7
2748779069441 5 39 3
6597069766657 3 41 5
39582418599937 9 42 5
79164837199873 9 43 5
263882790666241 15 44 7
1231453023109121 35 45 3
1337006139375617 19 46 3
3799912185593857 27 47 5
4222124650659841 15 48 19
7881299347898369 7 50 6
31525197391593473 7 52 3
180143985094819841 5 55 6
1945555039024054273 27 56 5
4179340454199820289 29 57 3
```

## 4.4 debuger

```
#include<bits/stdc++.h>
```

```
2   using namespace std;
3
4   #define out(args...) { cout << "Line " << __LINE__ << ": [" << #args << "] = [";
        debug(args); cout << "]\n"; }
5
6   template<typename T> void debug(T a) { cout << a; }
7
8   template<typename T, typename...args> void debug(T a, args...b) {
9       cout << a << ", ";
10      debug(b...);
11  }
12
13  template<typename T>
14  ostream& operator << (ostream &os, const vector<T> &a) {
15      os << "[";
16      int f = 0;
17      for(auto &x : a) os << (f++ ? ", " : "") << x;
18      os << "]";
19      return os;
20  }
21
22  template<typename T>
23  ostream& operator << (ostream &os, const set<T> &a) {
24      os << "{";
25      int f = 0;
26      for(auto &x : a) os << (f++ ? ", " : "") << x;
27      os << "}";
28      return os;
29  }
30
31  template<typename T>
32  ostream& operator << (ostream &os, const multiset<T> &a) {
33      os << "{";
34      int f = 0;
35      for(auto &x : a) os << (f++ ? ", " : "") << x;
36      os << "}";
37      return os;
38  }
39
40  template<typename A, typename B>
41  ostream& operator << (ostream &os, const map<A, B> &a) {
42      os << "{";
43      int f = 0;
44      for(auto &x : a) os << (f++ ? ", " : "") << x;
45      os << "}";
46      return os;
47  }
48
49  template<typename A, typename B>
50  ostream& operator << (ostream &os, const pair<A, B> &a) {
51      os << "(" << a.first << ", " << a.second << ")";
52      return os;
53  }
```

```
54
55  template<typename A, size_t N>
56  ostream& operator << (ostream &os, const array<A, N> &a) {
57      os << "{";
58      int f = 0;
59      for(int i = 0; i < N; i++) {
60          os << (f++ ? ", " : "") << a[i];
61      }
62      os << "}";
63      return os;
64  }
```

## 4.5 卡题时看一眼

- meet in the middle

- sqrt (分块，小的暴力大的 ds 维护)

- 多源点 BFS

- flow

- 容斥

- dp

- 构造

- 直接暴力复杂度可能是对的

- $\gcd(x, y) = \gcd(x, y - x)$

- 1e9 以内因子数最多的数的因子只有 1344 个,1e12:6720,1e18:103680

- MST,DIJ

- 快读，longlong, __int128, 取模