模板之用了就秃

WUST

So Like Coding? You Baldy

October 17, 2020

Contents

0 Header

0.1 pbds

0.1.1 head

```
#include <bits/extc++.h>
#pragma comment(linker, "/STACK:102400000,102400000")
using namespace __gnu_pbds; // tree, gp_hash_table, trie
using namespace __gnu_cxx; // rope
tree<TYPE, null_type, less<>, rb_tree_tag, tree_order_statistics_node_update> tr;
// 可并堆
#include <ext/pb_ds/priority_queue.hpp>
using namespace __gnu_pbds;
__gnu_pbds::priority_queue<int,greater<int>,pairing_heap_tag> q[maxn];
//q[i].join(q[j]) 将j堆并入i
```

0.2 FastIO

0.2.1 快速 IO

```
// 适用于正负整数
template <class T>
inline bool scan(T &ret){
    char c;
    int sgn;
    if (c = getchar(), c == EOF) return 0; //EOF
    while (c != '-' && (c < '0' || c > '9')) c = getchar();
    sgn = (c == '-') ? -1 : 1;
    ret = (c == '-') ? 0 : (c - '0');
    while (c = getchar(), c >= '0' && c <= '9') ret = ret * 10 + (c - '0');
    ret *= sgn;
    return 1;
}

template <class T>
inline void write(int x, int digit=10) {
    static int sta[35];
    int top = 0;
    do {
      sta[top++] = x % digit, x /= digit;
    } while (x);
    while (top) putchar(sta[--top] + 48);
}

inline int read() {
    int x = 0;
    char ch = getchar();
    while (ch > '9' || ch < '0')ch = getchar();
    while (ch >= '0' && ch <= '9') {
        x = x * 10 + ch - '0';
        ch = getchar();
    }
    return x;
}
```

0.2.2 快速幂

```
// 精确快速乘
ll mul(ll a, ll b, ll mod) {
    a %= mod; b %= mod;
    ll res = 0;
    while (b > 0) {
        if (b & 1) {
            res = (res + a);
            if (res >= mod) res -= mod;
        }
        a = (a + a);
        if (a >= mod) a -= mod;
        b >>= 1;
    }
    return res;
}

// O(1)快速乘
ll mul(ll a, ll b, ll M) { return (a * b - (ll)((long double)a * b / M) * M + M) % M; }

//int128
ll ans = ((__int128) a * b) % p;

// 10进制快速幂, 直接读入%s,c 预处理字符串len
char c[1000005];
int len;
ll qp(ll a) {
    len --;
    a %= mod;
    ll s = a;
    ll res = 1;
    while (len >= 0) {
        ll cur = s;
        for (int i = 1; i <= c[len] - '0'; ++i) {
            res = res * s % mod;
        }
        for (int i = 1; i < 10; ++i) {
            cur = cur * s % mod;
        }
        s = cur;
        len --;
    }
    return res;
}
```

0.2.3 Python 读入

```
// python一行读入
a,b = map(int, input().split())

a = []
for i in input().split():
    a.append(int(i))


f.write('{0} {1}\n'.format(1, 2))
```

0.2.4 特殊 IO

```
// 代替gets
scanf("%[^\n]%*c", ss)

// 高精分数输出
void print(ll x, ll y, int need) { // need 表示小数位数
    int a[need];
    printf("%lld.", x / y);
    ll t = x % y;
    for (int i = 0; i < need; i++) {
        t *= 10;
        a[i] = t / y;
        t = t % y;
    }
    if (t * 10 / y >= 5)a[need - 1]++;
    int i = need - 1;
    while (a[i] == 10) {
        a[i] = 0;
        a[--i]++;
    }
    for (int i = 0; i < need; i++) putchar(a[i]);
    putchar('\n');
}

// type mx
// int 10
// int64 19
// int128 38
template<class T, int g = 10>
void print(T x) {
    vector<char> a(38);
    if (x < 0) {x = -x;putchar('-');}
    if (x == 0) {putchar('0');return;}
    int tot = 0;
    while (x > 0) {
        a[tot++] = x % g;
        x /= 10;
    }
    for (int i = tot - 1; i >= 0; --i) putchar('0' + a[i]);
}
```

0.3 header

```
// Editor -> Live Templates
// add template group acm
// add template main
// C++ Declaration

#include <bits/stdc++.h>
#define ll long long
#define x first
#define y second
#define sz size()
#define all(x) x.begin(), x.end()
using namespace std;

typedef pair<int, int> pii;
typedef vector<int> vi;
typedef vector<long long> vl;
```

```

template <class T>
inline bool scan(T &ret){
    char c;
    int sgn;
    if (c = getchar(), c == EOF) return 0; //EOF
    while (c != '-' && (c < '0' || c > '9')) c = getchar();
    sgn = (c == '-') ? -1 : 1;
    ret = (c == '-') ? 0 : (c - '0');
    while (c = getchar(), c >= '0' && c <= '9') ret = ret * 10 + (c - '0');
    ret *= sgn;
    return 1;
}

const ll mod = 1e9+7;
const int maxn = $MAXN$;
const int inf = 0x3f3f3f3f;

ll qp(ll x, ll n, ll mod = ::mod) {
    ll res = 1; x %= mod;
    while (n > 0) {
        if (n & 1) res = res * x % mod;
        x = x * x % mod;
        n >>= 1;
    }
    return res;
}

int main(int argc, char* argv[]) {
    $END$
    return 0;
}

// C++ Expression debug
freopen("data.in","r",stdin);
freopen("data.out","w",stdout);
clock_t ST = clock();
cerr << "time: " << ((clock()-ST)*1000.0 / CLOCKS_PER_SEC) << "ms" << endl;


// C++ Expression tkase
int T;
scanf("%d", &T);
for (int kase = 1; kase <= T; ++kase) {
$END$
}
```

0.3.1 comp

```
// 1 create directory comp
// 2 create directory comp/test
// 3 create duipai.cpp
// add text

#include<bits/stdc++.h>
using namespace std;
int main(){
    int i;
```

```
  for (i=1;;i++){
        printf("The result of No. %d Case is:  ",i);
        system("python3 rand.py");
        system("./std < test/data.in > test/std.out");
        system("./my < test/data.in > test/my.out");
        if (system("diff test/std.out test/my.out")){
            printf("Wrong Answer\n");
            return 0;
        }
        else printf("Accepted\n");
    }
    return 0;
}


// 4 create duipai.sh
#!/bin/bash
g++ std.cpp -o std
g++ my.cpp -o my
python3 rand.py
./a.out

// 5 create rand.py
# coding=utf-8
from random import randint, choice, shuffle
# with open("../cmake-build-debug/data.in", "w") as f:
with open("test/data.in", "w") as f:
    n = randint(1, 10)
    m = randint(1, 10)
    f.write(f"{n} {m}")

// 6 terminal: g++ duipai.cpp
// 7 terminal: sudo chmod 777 duipai.sh
// 8 add my.cpp and std.cpp
// 9 ./duipai.sh
```

0.3.2 vim

```
syntax on
set nu
set tabstop=4
set shiftwidth=4
set background=dark

map <C-A> ggVG"+y
map <F5> :call Run()<CR>
func! Run()
    exec "w"
    exec "!g++ -Wall % -o %<"
    exec "!./%<"
endfunc
```

1 Math

1.1 群论

1.1.1 普通筛

```
int prim[maxn], tot;
bool v[maxn];
void init() {
    for (int i = 2; i < maxn; ++i) {
        if (!v[i]) prim[++tot] = i;
        for (int j = i + i; j < maxn && j <= (ll)i * i; j += i) v[j] = 1;
    }
}
```

1.1.2 线性筛

```
int mn[maxn], prim[maxprime];
int tot = 0;
void init() {
    for (int i = 2; i < maxn; ++i) {
        if (!mn[i]) prim[++tot] = i, mn[i] = i;
        for (int j = 1; j <= tot && i * prim[j] < maxn; ++j) {
            mn[i * prim[j]] = prim[j];
            if (!(i % prim[j])) break;
        }
    }
}

// 素数个数表
// 10 4
// 1e5 9592 (1e4)
// 2e5 17984 (2e4)
// 5e5 41538 (5e4)
// 1e6 78498 (8e4)
// 2e6 148933 (1.5e5)
// 1e7 664579 (7e5)
// 3e7 1857859 (2e6)
// 1e8 5761455 (6e6)
```

1.1.3 Pollard Rho

```
typedef pair<ll, ll> PLL;
namespace Factor {
    const int N = 1010000;
    ll C, fac[10010], a[1001000];
    int cnt, prime[N], p[N], psize, _cnt;
    ll _e[100], _pr[100];
    vector<ll> d;

    // 快速乘
    inline ll mul(ll a, ll b, ll p) {
        if (p <= 1000000000) return a * b % p;
        else if (p <= 100000000000011) return (((a * (b >> 20) % p) << 20) + (a * (b &
            ((1 << 20) - 1)))) % p;
        else {
            ll d = (ll) floor(a * (long double) b / p + 0.5);
            ll ret = (a * b - d * p) % p;
```

```
16              if (ret < 0) ret += p;
17              return ret;
18          }
19      }
20
21      // 素数筛
22      void prime_table() {
23          int i, j, tot, t1;
24          for (i = 1; i <= psize; i++) p[i] = i;
25          for (i = 2, tot = 0; i <= psize; i++) {
26              if (p[i] == i) prime[++tot] = i;
27              for (j = 1; j <= tot && (t1 = prime[j] * i) <= psize; j++) {
28                  p[t1] = prime[j];
29                  if (i % prime[j] == 0) break;
30              }
31          }
32      }
33
34      void init(int ps) {
35          psize = ps;
36          prime_table();
37      }
38
39      // 快速幂
40      ll powl(ll a, ll n, ll p) {
41          ll ans = 1;
42          for (; n; n >>= 1) {
43              if (n & 1) ans = mul(ans, a, p);
44              a = mul(a, a, p);
45          }
46          return ans;
47      }
48
49      bool witness(ll a, ll n) {
50          int t = 0;
51          ll u = n - 1;
52          for (; ~u & 1; u >>= 1) t++;
53          ll x = powl(a, u, n), _x = 0;
54          for (; t; t--) {
55              _x = mul(x, x, n);
56              if (_x == 1 && x != 1 && x != n - 1) return 1;
57              x = _x;
58          }
59          return _x != 1;
60      }
61
62      // n 是否为素数
63      bool miller(ll n) {
64          if (n < 2) return 0;
65          if (n <= psize) return p[n] == n;
66          if (~n & 1) return 0;
67          for (int j = 0; j <= 7; j++) if (witness(rand() % (n - 1) + 1, n)) return 0;
68          return 1;
69      }
70
71      ll gcd(ll a, ll b) {
72          ll ret = 1;
73          while (a != 0) {
74              if ((~a & 1) && (~b & 1)) ret <<= 1, a >>= 1, b >>= 1;
```

```
75              else if (~a & 1) a >>= 1;
76              else if (~b & 1) b >>= 1;
77              else {
78                  if (a < b) swap(a, b);
79                  a -= b;
80              }
81          }
82          return ret * b;
83      }
84
85      // 求 n 的一个素因子
86      ll rho(ll n) {
87          for (;;) {
88              ll X = rand() % n, Y, Z, T = 1, *lY = a, *lX = lY;
89              int tmp = 20;
90              C = rand() % 10 + 3;
91              X = mul(X, X, n) + C;
92              *(lY++) = X;
93              lX++;
94              Y = mul(X, X, n) + C;
95              *(lY++) = Y;
96              for (; X != Y;) {
97                  ll t = X - Y + n;
98                  Z = mul(T, t, n);
99                  if (Z == 0) return gcd(T, n);
100                 tmp--;
101                 if (tmp == 0) {
102                     tmp = 20;
103                     Z = gcd(Z, n);
104                     if (Z != 1 && Z != n) return Z;
105                 }
106                 T = Z;
107                 Y = *(lY++) = mul(Y, Y, n) + C;
108                 Y = *(lY++) = mul(Y, Y, n) + C;
109                 X = *(lX++);
110             }
111         }
112     }
113
114     void _factor(ll n) {
115         for (int i = 0; i < cnt; i++) {
116             if (n % fac[i] == 0) n /= fac[i], fac[cnt++] = fac[i];
117         }
118         if (n <= psize) {
119             for (; n != 1; n /= p[n]) fac[cnt++] = p[n];
120             return;
121         }
122         if (miller(n)) fac[cnt++] = n;
123         else {
124             ll x = rho(n);
125             _factor(x);
126             _factor(n / x);
127         }
128     }
129
130     void dfs(ll x, int dep) {
131         if (dep == _cnt) d.push_back(x);
132         else {
133             dfs(x, dep + 1);
```

```
134             for (int i = 1; i <= _e[dep]; i++) dfs(x *= _pr[dep], dep + 1);
135         }
136     }
137
138     void norm() {
139         sort(fac, fac + cnt);
140         _cnt = 0;
141         for (int i = 0; i < cnt; ++i)
142             if (i == 0 || fac[i] != fac[i - 1]) _pr[_cnt] = fac[i], _e[_cnt++] = 1;
143             else _e[_cnt - 1]++;
144     }
145
146     vector<ll> getd() {
147         d.clear();
148         dfs(1, 0);
149         return d;
150     }
151
152     // 返回所有因子
153     vector<ll> factor(ll n) {
154         cnt = 0;
155         _factor(n);
156         norm();
157         return getd();
158     }
159
160     // 返回所有素因子
161     vector<PLL> factorG(ll n) {
162         cnt = 0;
163         _factor(n);
164         norm();
165         vector<PLL> d;
166         for (int i = 0; i < _cnt; ++i) d.push_back(make_pair(_pr[i], _e[i]));
167         return d;
168     }
169
170     // a 是否为 p 的原根
171     bool is_primitive(ll a, ll p) {
172         vector<PLL> D = factorG(p - 1);
173         for (int i = 0; i < (int) D.size(); ++i) if (powl(a, (p - 1) / D[i].first, p)
                == 1) return 0;
174         return 1;
175     }
176
177     // a 关于 g 的阶
178     int findorder(ll a, ll p) {
179         vector<PLL> D = factorG(p - 1);
180         int t = p - 1;
181         for (int i = 0; i < (int) D.size(); ++i) {
182             while (t % D[i].first == 0 && powl(a, t / D[i].first, p) == 1) t /= D[i].
                    first;
183         }
184         return t;
185     }
186 }
```

1.1.4 解乘法逆元

```
1 void exgcd(ll a, ll b, ll c, ll d, ll &x, ll &y) {
```

```
    ll z = (a + b - 1) / b;
    if (z <= c / d) {
        x = z;
        y = 1;
        return;
    }
    a -= (z - 1) * b; c -= (z - 1) * d;
    exgcd(d, c, b, a, y, x);
    x += (z - 1) * y;
}

// 求 a/b mod p = x mod p, 优先 b 小
pair<ll, ll> invInv(ll p, ll x) {
    ll b, y;
    exgcd(p, x, p, x - 1, b, y);
    return {b * x - p * y, b};
}
```

1.1.5 EulerPhi

```
//计算欧拉phi函数, phi(n)且与n互素的正整数个数

//单点欧拉 O(sqrt(n))
ll euler(ll n) {
    ll rea = n;
    for (ll i = 2; i * i <= n; i++)
        if (n % i == 0) {
            rea = rea - rea / i;
            while (n % i == 0) n /= i;
        }
    if (n > 1) rea = rea - rea / n;
    return rea;
}


// maxn 为根号最值
// maxprime 为 maxn 内素数个数
// 素数线筛 + 单点求值
int prim[maxprime], cnt;
bool v[maxn];
void init() {
    v[0] = v[1] = 1;
    for (int i = 2; i < maxn; i++) {
        if (!v[i]) prim[++cnt] = i;
        for (int j = 1; j <= cnt && i * prim[j] < maxn; j++) {
            v[i * prim[j]] = 1;
            if (!(i % prim[j])) break;
        }
    }
}
ll phi(ll n) {
    ll rea = n;
    for (ll i = 1; prim[i] * prim[i] <= n; i++)
        if (n % prim[i] == 0) {
            rea = rea - rea / n;
            while (n % prim[i] == 0) n /= prim[i];
        }
    if (n > 1) rea = rea - rea / n;
```

```
    return rea;
}


// 线筛素数+欧拉函数
int phi[maxn], prim[maxprime], cnt;
bool v[maxn];
void init() {
    phi[1] = 1;
    for (int i = 2; i < maxn; ++i) {
        if (!v[i]) prim[++cnt] = i, phi[i] = i - 1;
        for (int j = 1; j <= cnt; ++j) {
            if (i * prim[j] >= maxn) break;
            v[i * prim[j]] = 1;
            if (i % prim[j] == 0) {
                phi[i * prim[j]] = phi[i] * prim[j]; break;
            } else phi[i * prim[j]] = phi[i] * phi[prim[j]];
        }
    }
}
```

1.1.6 求素因子

```
vector<pair<ll, int> > getFactors(ll x) {
    vector<pair<ll, int>> fact;
    for (int i = 1; prim[i] <= x / prim[i]; i++) {
        if (x % prim[i] == 0) {
            fact.emplace_back(prim[i], 0);
            while (x % prim[i] == 0) fact.back().second++, x /= prim[i];
        }
    }
    if (x != 1) fact.emplace_back(x, 1);
    return fact;
}
```

1.1.7 gcd

```
ll gcd(ll a,ll b) {while(b^=a^=b^=a%=b);return a;}

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

ll exgcd(ll a, ll b, ll &x, ll &y) {
    ll d = a;
    if (b) d = exgcd(b, a % b, y, x), y -= x * (a / b);
    else x = 1, y = 0;
    return d;
}

// ax + by = c
// x = x + k*dx
// y = y - k*dx
// 当x和y都非负时返回1, x, y即为当前最小非负整数解 (优先x)
bool solve(ll a, ll b, ll c, ll &x, ll &y, ll &dx, ll &dy) {
    x = y = dx = dy = 0;
    if (a == 0 && b == 0) return 0;
    ll x0, y0;
    ll d = exgcd(a, b, x0, y0);
    if (c % d != 0) return 0;
```

```
    dx = b / d, dy = a / d;
    x = (x0 % dx * ((c / d) % dx) % dx + dx) % dx;
    y = (c - a * x) / b;
    // 删掉这一句返回x的最小非负整数解
    if (y < 0) return 0;
    return 1;
}
```

1.1.8 区间筛

```
bool f[maxlen];
bool sieve[maxn];

// maxn 至少为 sqrt(R), 预处理
void init() {
    for (int i = 2; i < maxn; i++) sieve[i] = true;
    for (int i = 2; i * i < maxn; i++) {
        if (sieve[i]) {
            for (int j = i * 2; j < maxn; j += i) {
                sieve[j] = false;
            }
        }
    }
}
// 计算 [L,R] 素性, f[i] 为 1 表示 i+L 为素数
void cal(ll L, ll R) {
    int len = R - L + 1;
    for (int i = 0; i < len; i++) f[i] = true;
    if (1 - L >= 0) f[1 - L] = false;
    for (ll i = 2; i * i < R; i++) {
        if (sieve[i]) {
            for (ll j = max(1ll * 2, (L - 1 + i) / i) * i; j <= R; j += i) f[j - L] =
                false;
        }
    }
}
```

1.1.9 欧拉降幂

```
const int maxn = 1e7+50;

int prim[maxn], vis[maxn];
int tot, phi[maxn];
struct node {
    ll res;
    bool v;
};

node qpow(ll A, ll B, ll C) {
    ll re = 1;
    bool flag = true;
    while (B) {
        if (B & 1) {
            if ((re *= A) >= C) flag = 0;
            re = re % C;
        }
        B = B >> 1;
        if (B) {
```

```
20              if (A >= C) flag = 0;
21              A %= C;
22              if ((A *= A) >= C) flag = 0;
23              A %= C;
24          }
25      }
26      return node{re, flag};
27  }
28
29  void init(int n) {
30      phi[1] = 1;
31      for (int i = 2; i <= n; i++) {
32          if (!vis[i]) {
33              prim[++tot] = i;
34              phi[i] = i - 1;
35          }
36          for (int j = 1; j <= tot && prim[j] * i <= n; j++) {
37              vis[i * prim[j]] = 1;
38              if (i % prim[j] == 0) {
39                  phi[i * prim[j]] = phi[i] * prim[j];
40                  break;
41              } else phi[i * prim[j]] = phi[i] * (prim[j] - 1);
42          }
43
44      }
45  }
46
47
48  inline ll Euler(ll x) {
49      if (x < maxn) return phi[x];
50      return 0;
51  }
52
53  node f(ll a, ll k, ll p) {
54      if (p == 1) return node{0, 0};
55      if (k == 0) return node{a % p, a < p};
56      ll ep = Euler(p);
57      node tmp = f(a, k - 1, ep);
58      if (__gcd(a, p) == 1)return qpow(a, tmp.res, p);
59      if (!tmp.v) {
60          tmp.res += ep;
61      }
62      return qpow(a, tmp.res, p);
63  }
64
65  int main() {
66      ll a, k, p;
67      init(1e7+2);
68      int T;
69      scanf("%d", &T);
70      for (int kase = 1; kase <= T; ++kase) {
71          // k次a次方模p的值
72          scanf("%lld%lld%lld", &a, &k, &p);
73          if (k == 0) printf("%lld\n", 1 % p);
74          else printf("%lld\n", f(a, k - 1, p).res);
75      }
76      return 0;
77  }
```

1.1.10 dirichlet

```
#define poly vector<int>

// mod need to be prim
namespace dirichlet {
    int inv[maxn], cnt[maxn];
    bool vis[maxn];
    void init(int n) {
        cnt[1] = 1;
        for (int i = 2; i <= n; ++i) {
            if (!vis[i]) {
                cnt[i] = 1;
                for (int j = 2, k = i * j; k <= n; ++j, k += i) {
                    vis[k] = true;
                    if (!cnt[k] && cnt[j]) cnt[k] = cnt[j] + 1;
                }
            }
        }
        inv[1] = 1;
        // no need to n (1e6 need 128)
        for (int i = 2; i <= n; ++i) inv[i] = 1ll * (mod - mod / i) * inv[mod % i] %
            mod;
    }

    // no need to init, nlogn
    poly mul(poly a, poly b) {
        int n = max(a.size(), b.size()) - 1;
        while ((int)a.size() <= n) a.push_back(0);
        while ((int)b.size() <= n) b.push_back(0);
        poly d(n + 1);
        for (int i = 1; i <= n; ++i) {
            for (int j = 1, k = i; k <= n; k += i, ++j) d[k] = (d[k] + 1ll * a[i] * b[j
                ]) % mod;
        }
        return d;
    }

    // nlogn
    poly qp(poly a, int k) {
        k %= mod;
        if (k < 0) k += mod;
        int n = a.size() - 1;
        poly g(n + 1);
        poly tf = a;
        for (int i = 1; i <= n; ++i) a[i] = 1ll * a[i] * cnt[i] % mod * k % mod;
        for (int i = 1; i <= n; ++i) {
            int v = g[i];
            g[i] = 1ll * g[i] * inv[cnt[i]] % mod + (i == 1);
            for (int j = 2, t = i + i; t <= n; ++j, t += i)
                g[t] = (g[t] + 1ll * g[i] * a[j]) % mod;
            for (int j = 2, t = i + i; t <= n; ++j, t += i)
                g[t] = (g[t] + 1ll * (mod - tf[j]) * v) % mod;
        }
        return g;
    }

    // nlogn
    inline poly ln(const poly& a) {
```

```
56          int n = a.size() - 1;
57          poly b(n + 1);
58          for (int i = 2; i <= n; ++i) b[i] = 1ll * a[i] * cnt[i] % mod;
59          for (int i = 2; i <= n; ++i) {
60              for (int j = 2, k = i * j; k <= n; ++j, k += i) b[k] = (b[k] - 1ll * b[i] *
                    a[j]) % mod;
61              b[i] = 1ll * b[i] * inv[cnt[i]] % mod;
62              if (b[i] < 0) b[i] += mod;
63          }
64          return b;
65      }
66
67      // nlogn
68      inline poly exp(poly a) {
69          int n = a.size() - 1;
70          poly b(n + 1);
71          for (int i = 2; i <= n; ++i) a[i] = 1ll * a[i] * cnt[i] % mod;
72          b[1] = 1;
73          for (int i = 1; i <= n; ++i) {
74              b[i] = 1ll * b[i] * inv[cnt[i]] % mod;
75              for (int j = 2, k = i * j; k <= n; ++j, k += i) b[k] = (b[k] + 1ll * b[i] *
                    a[j]) % mod;
76          }
77          return b;
78      }
79  }
80  using namespace dirichlet;
```

1.2 同余

1.2.1 逆元

```
1   /*
2   1.费马小定理
3   条件:mod为素数
4   */
5   ll inv(ll x) { return qp(x, mod - 2); }
6
7   /*
8   2.扩展欧几里得
9   条件:gcd(a,mod)==1
10  如果gcd(a,mod)!=1 返回-1
11  */
12  ll inv(ll a, ll p) {
13      ll g, x, y;
14      g = exgcd(a, p, x, y);
15      return g == 1 ? (x + p) % p : -1;
16  }
17
18  /*
19  3.公式
20  a/b%mod=c
21  ->a%(b*mod)/b=c
22  */
23
24  /*
25  4.逆元打表
26  p是模
27  p要求是奇素数
```

```
*/
ll inv[maxn];

void getinv(int n, ll p) {
    inv[1] = 1;
    for (int i = 2; i <= n; i++) inv[i] = (p - p / i) * inv[p % i] % p;
}

// log逆元
ll dlog(ll g, ll b, ll p) {
    ll m = sqrt(p - 1);
    map<ll, ll> powers;
    for (long j = 0; j < m; j++) powers[qp(g, j, p)] = j;
    long gm = qp(g, -m + 2 * (p - 1), p);
    for (int i = 0; i < m; i++) {
        if (powers[b]) return i * m + powers[b];
        b = b * gm % p;
    }
    return -1;
}
```

1.2.2 中国剩余定理

```
void exgcd(ll a, ll b, ll &x, ll &y) {
    if (b) exgcd(b, a % b, y, x), y -= x * (a / b);
    else x = 1, y = 0;
}

// x % m[i] = a[i]
ll crt(const vector<ll>& m, const vector<ll>& a) {
    ll M = 1, ans = 0;
    int n = a.size();
    for (int i = 0; i < n; i++) M *= m[i];
    for (int i = 0; i < n; i++) {
        ll x = 0, y = 0, Mi = M / m[i];
        exgcd(Mi, m[i], x, y);
        ans = (ans + Mi % M * x % M * a[i] % M + M) % M;
    }
    if (ans < 0) ans += M;
    return ans;
}
```

1.2.3 扩展中国剩余定理

```
ll exgcd(ll a, ll b, ll &x, ll &y) {
    ll d = a;
    if (b) d = exgcd(b, a % b, y, x), y -= x * (a / b);
    else x = 1, y = 0;
    return d;
}

// x % m[i] = a[i]
ll excrt(const vector<ll> &m, const vector<ll> &a) {
    ll M = m[0], R = a[0];
    int n = a.size();
    for (int i = 1; i < n; i++) {
        ll x = 0, y = 0, d = exgcd(M, m[i], x, y);
        if ((R - a[i]) % d) return -1;
```

```
//         x = mul((R - a[i]) / d, x, m[i]);
        x = (R - a[i]) / d * x % m[i];
        R -= M * x;
        M = M / d * m[i];
        R %= M;
    }
    if (R < 0) R += M;
    return R;
}
```

1.2.4 BSGS

```
// 求 a^x = b % p
int BSGS(int a, int b, int p) {
    map<int, int> hash;
    b %= p;
    int t = (int) sqrt(p) + 1;
    for (int j = 0; j < t; j++) {
        int val = 1ll * b * qp(a, j, p) % p;
        hash[val] = j;
    }
    a = qp(a, t, p);
    if (a == 0) return b == 0 ? 1 : -1;
    for (int i = 0; i <= t; i++) {
        int val = qp(a, i, p);
        int j = hash.find(val) == hash.end() ? -1 : hash[val];
        if (j >= 0 && i * t - j >= 0) return i * t - j;
    }
    return -1;
}
```

1.2.5 exBSGS

```
unordered_map<int, int> Hash;

int exBSGS(int a, int b, int p) {
    a %= p, b %= p;
    if (b == 1) return 0;
    if (!b && !a) return 1;
    if (!a) return -1;
    if (!b) {
        int ret = 0, d;
        while ((d = __gcd(a, p)) != 1) {
            ++ret, p /= d;
            if (p == 1) return ret;
        }
        return -1;
    }
    int ret = 0, A = a, B = b, P = p, C = 1, d;
    while ((d = __gcd(A, P)) != 1) {
        if (B % d) return -1;
        P /= d, B /= d;
        C = 1ll * C * (A / d) % P;
        ++ret;
        if (C == B) return ret;
    }
    Hash.clear();
    int f = 1, t = sqrt(P) + 1;
```

```
    for (int i = 0; i < t; i++) {
        Hash[1ll * f * B % P] = i;
        f = 1ll * f * A % P;
    }
    int tf = f;
    f = 1ll * f * C % P;
    for (int i = 1; i <= t; i++) {
        if (Hash.find(f) != Hash.end()) return ret + i * t - Hash[f];
        f = 1ll * f * tf % P;
    }
    return -1;
}
```

1.2.6 模素数二次同余方程

```
// 要求模为素数, 输入n, mod, 返回 x^2 % mod = n, 可解任意一元二次方程

bool Legendre(ll a, ll p) {
    return qp(a, p - 1 >> 1, p) == 1;
}

ll modsqr(ll a, ll p) {
    ll x;
    ll i, k, b;
    if (p == 2) x = a % p;
    else if (p % 4 == 3) x = qp(a, p + 1 >> 2, p);
    else {
        for (b = 1; Legendre(b, p); ++b);
        i = p - 1 >> 1;
        k = 0;
        do {
            i >>= 1;
            k >>= 1;
            if (!((1LL * qp(a, i, p) * qp(b, k, p) + 1) % p)) k += p - 1 >> 1;
        } while (!(i & 1));
        x = 1ll * qp(a, i + 1 >> 1, p) * qp(b, k >> 1, p) % p;
    }
    return min(x, p - x);
//    if(p-x<x) x=p-x;
//    if(x==p-x) printf("%d\n",x);
//    else printf("%d %d\n",x,p-x);
}
```

1.3 线代

1.3.1 线性基

```
struct Base {
#define TYPE ll
    static const int len = 64;
    bool rel; int sz;
    TYPE a[len];

    void init() {
        rel = sz = 0;
        memset(a, 0, sizeof a);
    }

```

```
12      TYPE &operator[](int x) {
13          return a[x];
14      }
15
16      TYPE operator[](int x) const {
17          return a[x];
18      }
19
20
21      void ins(ll x) {
22          for(int i = 63; i >= 0; i --) {
23              if((x >> i) & 1) {
24                  if(!d[i]) return void(d[i] = x);
25                  x ^= d[i];
26              }
27          }
28      }
29
30      void insert(TYPE t) {
31          for (int i = len - 1; i >= 0; --i) {
32              if (!(t >> i & 1)) continue;
33              if (a[i]) t ^= a[i];
34              else {
35                  for (int j = 0; j < i; ++j) if (t >> j & 1) t ^= a[j];
36                  for (int j = i+1; j < len; ++j) if (a[j] >> i & 1) a[j] ^= t;
37                  a[i] = t;
38                  ++sz;
39                  return;
40              }
41          }
42          rel = true;
43      }
44
45      bool check(TYPE x) {
46          for (int i = len - 1; i >= 0; i--)
47              if ((x >> i) & 1) {
48                  if (a[i]) x ^= a[i];
49                  else return false;
50              }
51          return true;
52      }
53      TYPE mx() {
54          TYPE res = 0;
55          for (int i = len - 1; i >= 0; --i) {
56              if ((res ^ (a[i])) > res) res ^= a[i];
57          }
58          return res;
59      }
60  //    vector<TYPE> v;
61  //    void basis() {for (int i = 0; i < len; ++i) if (a[i]) v.push_back(a[i]);}
62  //    TYPE k_th(TYPE k) {
63  //        k -= rel;
64  //        if(k >= (((TYPE)1) << sz)) return -1;
65  //        TYPE ans = 0;
66  //        for(int i = 0; i < (int)v.size(); i ++) if(k & (((TYPE)1) << i)) ans ^= v[i];
67  //        return ans;
68  //    }
69  //    void init()
70  //    {
```

```
71  //         sz = 0;
72  //         for(int i = 0; i < len; i ++) if(a[i])
73  //                 for(int j = 0; j < i; j ++)
74  //                     if(a[i] & (1ll << j)) a[i] ^= a[j];
75  //         for(int i = 0; i < len; i ++) if(a[i]) a[sz ++] = a[i];
76  //     }
77      friend Base intersection(const Base &a, const Base &b) {
78          Base ans = {}, c = b, d = b;
79          for (int i = 0; i < len; i++) {
80              TYPE x = a[i];
81              if (!x)continue;
82              int j = i;
83              TYPE T = 0;
84              for (; j >= 0; --j) {
85                  if ((x >> j) & 1)
86                      if (c[j]) {
87                          x ^= c[j];
88                          T ^= d[j];
89                      }
90                      else break;
91              }
92              if (!x)ans[i] = T;
93              else {
94                  c[j] = x;
95                  d[j] = T;
96              }
97          }
98          return ans;
99      }
100
101 #undef TYPE
102 };
103
104 // 前缀线性基
105 struct LinearBasis {
106     int f[20], g[20];
107
108     void ins(int x, int idx) {
109         for (int i = 19; ~i; i--) {
110             if ((x >> i) & 1) {
111                 if (f[i]) {
112                     if (g[i] <= idx) {
113                         x ^= f[i];
114                         f[i] ^= x;
115                         swap(g[i], idx);
116                     }
117                     else x ^= f[i];
118                 } else {
119                     f[i] = x;
120                     g[i] = idx;
121                     break;
122                 }
123             }
124         }
125     }
126
127     int query(int l) {
128         int res = 0;
129         for (int i = 19; ~i; i--)
```

```
            if (g[i] >= l)
                res = max(res, res ^ f[i]);
        return res;
    }
} base[maxn];
```

1.3.2 高斯消元

```
int n;
double b[maxn], c[maxn][maxn];
//c: 系数矩阵, b: 常数

void guass() {
    for (int i = 1; i <= n; i++) {
        for (int j = i; j <= n; j++)
            if (fabs(c[j][i]) > eps) {
                for (int k = 1; k <= n; k++) swap(c[i][k], c[j][k]);
                swap(b[i], b[j]);
            }
        for (int j = 1; j <= n; j++) {
            if (i == j) continue;
            double rate = c[j][i] / c[i][i];
            for (int k = i; k <= n; k++) c[j][k] -= c[i][k] * rate;
            b[j] -= b[i] * rate;
        }
    }
}
```

1.3.3 BM

```
//Berlekamp-Massey
typedef vector<int> VI;
namespace linear_seq
{
#define rep(i,a,n) for (int i=a;i<n;i++)
#define SZ(x) ((int)(x).size())
#define pb(x) push_back(x)
    const ll mod=1e9+7;
    ll powmod(ll a,ll b){ll res=1;a%=mod; assert(b>=0); for(;b;b>>=1){if(b&1)res=res*a%
        mod;a=a*a%mod;}return res;}
    const int N=10010;
    ll res[N],base[N],_c[N],_md[N];
    vector<int> Md;
    void mul(ll *a,ll *b,int k)
    {
        rep(i,0,k+k) _c[i]=0;
        rep(i,0,k) if (a[i]) rep(j,0,k) _c[i+j]=(_c[i+j]+a[i]*b[j])%mod;
        for (int i=k+k-1;i>=k;i--) if (_c[i])
                rep(j,0,SZ(Md)) _c[i-k+Md[j]]=(_c[i-k+Md[j]]-_c[i]*_md[Md[j]])%mod;
        rep(i,0,k) a[i]=_c[i];
    }
    int solve(ll n,VI a,VI b){
        ll ans=0,pnt=0;
        int k=SZ(a);
        assert(SZ(a)==SZ(b));
        rep(i,0,k) _md[k-1-i]=-a[i];_md[k]=1;
        Md.clear();
        rep(i,0,k) if (_md[i]!=0) Md.push_back(i);
```

```
28          rep(i,0,k) res[i]=base[i]=0;
29          res[0]=1;
30          while ((1ll<<pnt)<=n) pnt++;
31          for (int p=pnt;p>=0;p--) {
32              mul(res,res,k);
33              if ((n>>p)&1) {
34                  for (int i=k-1;i>=0;i--) res[i+1]=res[i];res[0]=0;
35                  rep(j,0,SZ(Md)) res[Md[j]]=(res[Md[j]]-res[k]*_md[Md[j]])%mod;
36              }
37          }
38          rep(i,0,k) ans=(ans+res[i]*b[i])%mod;
39          if (ans<0) ans+=mod;
40          return ans;
41      }
42      VI BM(VI s){
43          VI C(1,1),B(1,1);
44          int L=0,m=1,b=1;
45          rep(n,0,SZ(s)){
46              ll d=0;
47              rep(i,0,L+1) d=(d+(ll)C[i]*s[n-i])%mod;
48              if(d==0) ++m;
49              else if(2*L<=n){
50                  VI T=C;
51                  ll c=mod-d*powmod(b,mod-2)%mod;
52                  while (SZ(C)<SZ(B)+m) C.pb(0);
53                  rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
54                  L=n+1-L; B=T; b=d; m=1;
55              } else {
56                  ll c=mod-d*powmod(b,mod-2)%mod;
57                  while (SZ(C)<SZ(B)+m) C.pb(0);
58                  rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
59                  ++m;
60              }
61          }
62          return C;
63      }
64      int gao(VI a,ll n)
65      {
66          VI c=BM(a);
67          c.erase(c.begin());
68          rep(i,0,SZ(c)) c[i]=(mod-c[i])%mod;
69          return solve(n,c,VI(a.begin(),a.begin()+SZ(c)));
70      }
71  };//linear_seq::gao(VI{},n-1)
```

1.3.4 exBM

```
1   // given first m items init[0..m-1] and coefficents trans[0..m-1] or
2   // given first 2 *m items init[0..2m-1], it will compute trans[0..m-1]
3   // for you. trans[0..m] should be given as that
4   //      init[m] = sum_{i=0}^{m-1} init[i] * trans[i]
5   struct LinearRecurrence
6   {
7       using int64 = long long;
8       using vec = std::vector<int64>;
9
10      static void extand(vec& a, size_t d, int64 value = 0)
11      {
```

```
12          if (d <= a.size()) return;
13          a.resize(d, value);
14      }
15      static vec BerlekampMassey(const vec& s, int64 mod)
16      {
17          std::function<int64(int64)> inverse = [&](int64 a) {
18              return a == 1 ? 1 : (int64)(mod - mod / a) * inverse(mod % a) % mod;
19          };
20          vec A = {1}, B = {1};
21          int64 b = s[0];
22          for (size_t i = 1, m = 1; i < s.size(); ++i, m++)
23          {
24              int64 d = 0;
25              for (size_t j = 0; j < A.size(); ++j)
26              {
27                  d += A[j] * s[i - j] % mod;
28              }
29              if (!(d %= mod)) continue;
30              if (2 * (A.size() - 1) <= i)
31              {
32                  auto temp = A;
33                  extand(A, B.size() + m);
34                  int64 coef = d * inverse(b) % mod;
35                  for (size_t j = 0; j < B.size(); ++j)
36                  {
37                      A[j + m] -= coef * B[j] % mod;
38                      if (A[j + m] < 0) A[j + m] += mod;
39                  }
40                  B = temp, b = d, m = 0;
41              }
42              else
43              {
44                  extand(A, B.size() + m);
45                  int64 coef = d * inverse(b) % mod;
46                  for (size_t j = 0; j < B.size(); ++j)
47                  {
48                      A[j + m] -= coef * B[j] % mod;
49                      if (A[j + m] < 0) A[j + m] += mod;
50                  }
51              }
52          }
53          return A;
54      }
55      static void exgcd(int64 a, int64 b, int64& g, int64& x, int64& y)
56      {
57          if (!b)
58              x = 1, y = 0, g = a;
59          else
60          {
61              exgcd(b, a % b, g, y, x);
62              y -= x * (a / b);
63          }
64      }
65      static int64 crt(const vec& c, const vec& m)
66      {
67          int n = c.size();
68          int64 M = 1, ans = 0;
69          for (int i = 0; i < n; ++i) M *= m[i];
70          for (int i = 0; i < n; ++i)
```

```
71          {
72              int64 x, y, g, tm = M / m[i];
73              exgcd(tm, m[i], g, x, y);
74              ans = (ans + tm * x * c[i] % M) % M;
75          }
76          return (ans + M) % M;
77      }
78      static vec ReedsSloane(const vec& s, int64 mod)
79      {
80          auto inverse = [](int64 a, int64 m) {
81              int64 d, x, y;
82              exgcd(a, m, d, x, y);
83              return d == 1 ? (x % m + m) % m : -1;
84          };
85          auto L = [](const vec& a, const vec& b) {
86              int da = (a.size() > 1 || (a.size() == 1 && a[0])) ? a.size() - 1 : -1000;
87              int db = (b.size() > 1 || (b.size() == 1 && b[0])) ? b.size() - 1 : -1000;
88              return std::max(da, db + 1);
89          };
90          auto prime_power = [&](const vec& s, int64 mod, int64 p, int64 e) {
91              // linear feedback shift register mod p^e, p is prime
92              std::vector<vec> a(e), b(e), an(e), bn(e), ao(e), bo(e);
93              vec t(e), u(e), r(e), to(e, 1), uo(e), pw(e + 1);
94              ;
95              pw[0] = 1;
96              for (int i = pw[0] = 1; i <= e; ++i) pw[i] = pw[i - 1] * p;
97              for (int64 i = 0; i < e; ++i)
98              {
99                  a[i] = {pw[i]}, an[i] = {pw[i]};
100                 b[i] = {0}, bn[i] = {s[0] * pw[i] % mod};
101                 t[i] = s[0] * pw[i] % mod;
102                 if (t[i] == 0)
103                 {
104                     t[i] = 1, u[i] = e;
105                 }
106                 else
107                 {
108                     for (u[i] = 0; t[i] % p == 0; t[i] /= p, ++u[i])
109                         ;
110                 }
111             }
112             for (size_t k = 1; k < s.size(); ++k)
113             {
114                 for (int g = 0; g < e; ++g)
115                 {
116                     if (L(an[g], bn[g]) > L(a[g], b[g]))
117                     {
118                         ao[g] = a[e - 1 - u[g]];
119                         bo[g] = b[e - 1 - u[g]];
120                         to[g] = t[e - 1 - u[g]];
121                         uo[g] = u[e - 1 - u[g]];
122                         r[g] = k - 1;
123                     }
124                 }
125                 a = an, b = bn;
126                 for (int o = 0; o < e; ++o)
127                 {
128                     int64 d = 0;
129                     for (size_t i = 0; i < a[o].size() && i <= k; ++i)
```

```
130                     {
131                         d = (d + a[o][i] * s[k - i]) % mod;
132                     }
133                     if (d == 0)
134                     {
135                         t[o] = 1, u[o] = e;
136                     }
137                     else
138                     {
139                         for (u[o] = 0, t[o] = d; t[o] % p == 0; t[o] /= p, ++u[o])
140                             ;
141                         int g = e - 1 - u[o];
142                         if (L(a[g], b[g]) == 0)
143                         {
144                             extand(bn[o], k + 1);
145                             bn[o][k] = (bn[o][k] + d) % mod;
146                         }
147                         else
148                         {
149                             int64 coef = t[o] * inverse(to[g], mod) % mod * pw[u[o] -
                                    uo[g]] % mod;
150                             int m = k - r[g];
151                             extand(an[o], ao[g].size() + m);
152                             extand(bn[o], bo[g].size() + m);
153                             for (size_t i = 0; i < ao[g].size(); ++i)
154                             {
155                                 an[o][i + m] -= coef * ao[g][i] % mod;
156                                 if (an[o][i + m] < 0) an[o][i + m] += mod;
157                             }
158                             while (an[o].size() && an[o].back() == 0) an[o].pop_back();
159                             for (size_t i = 0; i < bo[g].size(); ++i)
160                             {
161                                 bn[o][i + m] -= coef * bo[g][i] % mod;
162                                 if (bn[o][i + m] < 0) bn[o][i + m] -= mod;
163                             }
164                             while (bn[o].size() && bn[o].back() == 0) bn[o].pop_back();
165                         }
166                     }
167                 }
168             }
169             return std::make_pair(an[0], bn[0]);
170         };
171 
172         std::vector<std::tuple<int64, int64, int>> fac;
173         for (int64 i = 2; i * i <= mod; ++i)
174         {
175             if (mod % i == 0)
176             {
177                 int64 cnt = 0, pw = 1;
178                 while (mod % i == 0) mod /= i, ++cnt, pw *= i;
179                 fac.emplace_back(pw, i, cnt);
180             }
181         }
182         if (mod > 1) fac.emplace_back(mod, mod, 1);
183         std::vector<vec> as;
184         size_t n = 0;
185         for (auto&& x : fac)
186         {
187             int64 mod, p, e;
```

```
188             vec a, b;
189             std::tie(mod, p, e) = x;
190             auto ss = s;
191             for (auto&& x : ss) x %= mod;
192             std::tie(a, b) = prime_power(ss, mod, p, e);
193             as.emplace_back(a);
194             n = std::max(n, a.size());
195         }
196         vec a(n), c(as.size()), m(as.size());
197         for (size_t i = 0; i < n; ++i)
198         {
199             for (size_t j = 0; j < as.size(); ++j)
200             {
201                 m[j] = std::get<0>(fac[j]);
202                 c[j] = i < as[j].size() ? as[j][i] : 0;
203             }
204             a[i] = crt(c, m);
205         }
206         return a;
207     }
208
209     LinearRecurrence(const vec& s, const vec& c, int64 mod) : init(s), trans(c), mod(
            mod), m(s.size()) {}
210     LinearRecurrence(const vec& s, int64 mod, bool is_prime = true) : mod(mod)
211     {
212         vec A;
213         if (is_prime)
214             A = BerlekampMassey(s, mod);
215         else
216             A = ReedsSloane(s, mod);
217         if (A.empty()) A = {0};
218         m = A.size() - 1;
219         trans.resize(m);
220         for (int i = 0; i < m; ++i)
221         {
222             trans[i] = (mod - A[i + 1]) % mod;
223         }
224         std::reverse(trans.begin(), trans.end());
225         init = {s.begin(), s.begin() + m};
226     }
227     int64 calc(int64 n)
228     {
229         if (mod == 1) return 0;
230         if (n < m) return init[n];
231         vec v(m), u(m << 1);
232         int msk = !!n;
233         for (int64 m = n; m > 1; m >>= 1) msk <<= 1;
234         v[0] = 1 % mod;
235         for (int x = 0; msk; msk >>= 1, x <<= 1)
236         {
237             std::fill_n(u.begin(), m * 2, 0);
238             x |= !!(n & msk);
239             if (x < m)
240                 u[x] = 1 % mod;
241             else
242             { // can be optimized by fft/ntt
243                 for (int i = 0; i < m; ++i)
244                 {
245                     for (int j = 0, t = i + (x & 1); j < m; ++j, ++t)
```

```
246                     {
247                         u[t] = (u[t] + v[i] * v[j]) % mod;
248                     }
249                 }
250                 for (int i = m * 2 - 1; i >= m; --i)
251                 {
252                     for (int j = 0, t = i - m; j < m; ++j, ++t)
253                     {
254                         u[t] = (u[t] + trans[j] * u[i]) % mod;
255                     }
256                 }
257             }
258             v = {u.begin(), u.begin() + m};
259         }
260         int64 ret = 0;
261         for (int i = 0; i < m; ++i)
262         {
263             ret = (ret + v[i] * init[i]) % mod;
264         }
265         return ret;
266     }
267 
268     vec init, trans;
269     int64 mod;
270     int m;
271 };
```

1.4 组合数学

1.4.1 Lucas

```
1  const int maxn = 1e6 + 10;
2  
3  ll fac[maxn], inv[maxn], facinv[maxn];
4  
5  void init() {
6      fac[0] = inv[0] = facinv[0] = 1;
7      fac[1] = inv[1] = facinv[1] = 1;
8      for (int i = 2; i < maxn; i++) {
9          fac[i] = fac[i - 1] * i % mod;
10         inv[i] = mod - mod / i * inv[mod % i] % mod;
11         facinv[i] = facinv[i - 1] * inv[i] % mod;
12     }
13 }
14 
15 ll C(int n, int k) {
16     if (k > n || k < 0) return 0;
17     return fac[n] * facinv[k] % mod * facinv[n - k] % mod;
18 }
19 
20 ll lucas(ll n, ll m) {
21     ll res = 1;
22     while (n && m) {
23         res = res * C(n % mod, m % mod) % mod;
24         n /= mod;
25         m /= mod;
26     }
27     return res;
28 }
```

1.4.2 exLucas

```
typedef long long ll;

ll p, n, m;

ll exgcd(ll a, ll b, ll &x, ll &y)
{
    if(!b) { x = 1; y = 0; return a; }
    ll res = exgcd(b, a % b, x, y), t;
    t = x, x = y, y = t - a / b * y;
    return res;
}

ll qp(ll a, ll n, ll mod)
{
    ll ans = 1, base = a;
    for(; n; n >>= 1, (base *= base) %= mod) if(n & 1) (ans *= base) %= mod;
    return ans;
}

ll fac(ll n, ll a, ll b)
{
    if(!n) return 1;
    ll res = 1;
    for(ll i = 2; i <= b; i ++)
        if(i % a) (res *= i) %= b;
    res = qp(res, n / b, b);
    for(ll i = 2; i <= n % b; i ++)
        if(i % a) (res *= i) %= b;
    return res * fac(n / a, a, b) % b;
}

ll inv(ll n, ll mod)
{
    ll x, y;
    exgcd(n, mod, x, y);
    return (x += mod) > mod ? x - mod : x;
}

ll CRT(ll b, ll mod) { return b * inv(p / mod, mod) % p * (p / mod) % p; }

ll C(ll n, ll m, ll a, ll b)
{
    ll up = fac(n, a, b), d1 = fac(m, a, b), d2 = fac(n - m, a, b);
    ll k = 0;
    for(ll i = n; i; i /= a) k += i / a;
    for(ll i = m; i; i /= a) k -= i / a;
    for(ll i = n - m; i; i /= a) k -= i / a;
    return up * inv(d1, b) % b * inv(d2, b) % b * qp(a, k, b) % b;
}

ll exlucas(ll n, ll m)
{
    ll res = 0, tmp = p, b;
    int lim = sqrt(p) + 5;
    for(int i = 2; i <= lim; i ++) if(tmp % i == 0)
    {
        b = 1;
```

```
58          while(tmp % i == 0) tmp /= i, b *= i;
59          (res += CRT(C(n, m, i, b), b)) %= p;
60      }
61      if(tmp > 1) (res += CRT(C(n, m, tmp, tmp), tmp)) %= p;
62      return res;
63  }
```

1.4.3 递推组合数

```
1   // ---
2   // $0 \leq m \leq n \leq 1000$
3   // ---
4   const int maxn = 1010;
5   ll C[maxn][maxn];
6   void init() {
7       C[0][0] = 1;
8       for (int i = 1; i < maxn; i++)
9       {
10          C[i][0] = 1;
11          for (int j = 1; j <= i; j++) C[i][j] = (C[i - 1][j - 1] + C[i - 1][j]) % mod;
12      }
13  }
14
15  // ---
16  // $0 \leq m \leq n \leq 10^5$, 模$p$为素数
17  // ---
18  const int maxn = 100010;
19  ll f[maxn];
20  ll inv[maxn]; // 阶乘的逆元
21  void CalFact() {
22      f[0] = 1;
23      for (int i = 1; i < maxn; i++) f[i] = (f[i - 1] * i) % p;
24      inv[maxn - 1] = qp(f[maxn - 1], p - 2);
25      for (int i = maxn - 2; ~i; i--) inv[i] = inv[i + 1] * (i + 1) % p;
26  }
27  ll C(int n, int m) { return f[n] * inv[m] % p * inv[n - m] % p; }
```

1.4.4 小模数组合数

```
1   // ---
2   // $p$小$n$,$m$大
3   // ---
4
5   const int NICO = 100000+10;
6   const int MOD  = 99991;
7   ll f[NICO];
8
9   ll Lucas(ll a,ll k)
10  {
11      ll res = 1;
12      while(a && k)
13      {
14          ll a1 = a % MOD;
15          ll b1 = k % MOD;
16          if(a1 < b1) return 0;
17          res = res*f[a1]*qp(f[b1]*f[a1-b1]%MOD,MOD-2)%MOD;
18          a /= MOD;
19          k /= MOD;
```

```
    }
    return res;
}

void init()
{
    f[0] = 1;
    for(int i=1;i<=MOD;i++)
    {
        f[i] = f[i-1]*i%MOD;
    }
}

int main()
{
    init();
    cout << Lucas(5,2) << endl;
}
```

1.4.5 大模数组合数

```
// ---
// $n$,$m$小$p$大
// ---
map<int, ll> m;

const int MOD = 1e9+7;
void fun(int n, int k) {
    for (int i = 2; i <= sqrt(n * 1.0); i++) {
        while (n % i == 0) {
            n /= i;
            m[i] += k;
        }
    }
    if (n > 1) {
        m[n] += k;
    }
}

ll C(ll a, ll b) {
    if (a < b || a < 0 || b < 0)
        return 0;
    m.clear();
    ll ret = 1;
    b = min(a - b, b);
    for (int i = 0; i < b; i++) {
        fun(a - i, 1);
    }
    for (int i = b; i >= 1; i--) {
        fun(i, -1);
    }
    for (__typeof(m.begin()) it = m.begin(); it != m.end(); it++) {
        if ((*it).second != 0) {
            ret *= qp((*it).first, (*it).second);
            ret %= MOD;
        }
    }
    return ret;
```

```
}

int main(int argc,char *argv[])
{
    ll a, b;
    while (scanf("%lld%lld", &a, &b) != EOF) {
        printf("%lld\n", C(a, b));
    }
    return 0;
}
```

1.5 多项式

1.5.1 FFT

```
// maxn 至少是大于m+n的2次方数
// m+n 2e5 maxn (1<<18)+50
// m+n 4e5 maxn (1<<19)+50
// m+n 2e6 maxn (1<<21)+50

// 普通fft
#include <bits/stdc++.h>

using namespace std;
typedef long long ll;

struct Complex {
    double r, i;

    Complex(double r, double i) : r(r), i(i) {}
    Complex() {}
    inline Complex operator+(const Complex &rhs) const { return Complex(r + rhs.r, i + rhs.i); }
    inline Complex operator-(const Complex &rhs) const { return Complex(r - rhs.r, i - rhs.i); }
    inline Complex operator*(const Complex &rhs) const { return Complex(r * rhs.r - i * rhs.i, r * rhs.i + i * rhs.r); }
    inline void operator/=(const double &x) { r /= x, i /= x; }
    inline void operator*=(const Complex &rhs) { *this = Complex(r * rhs.r - i * rhs.i, r * rhs.i + i * rhs.r); }
    inline void operator+=(const Complex &rhs) { r += rhs.r, i += rhs.i; }
};

const int maxn = 4e6 + 6;
#define PI 3.14159265354
int pos[maxn];

void init(const int &n) {
    for (int i = 0, j = 0; i < n; ++i) {
        pos[i] = j;
        for (int l = n >> 1; (j ^= l) < l; l >>= 1);
    }
}

void transform(Complex *a, const int &n, bool inverse) {
    for (int i = 0; i < n; ++i) if (i > pos[i]) std::swap(a[i], a[pos[i]]);
    for (int l = 2; l <= n; l <<= 1) {
        int m = l / 2;
```

```
40          Complex omega = {cos(2 * PI / l), inverse ? -sin(2 * PI / l) : sin(2 * PI / l)
                };
41          for (Complex *p = a; p != a + n; p += l) {
42              Complex x = {1, 0};
43              for (int i = 0; i < m; ++i, x *= omega) {
44                  Complex t = x * p[m + i];
45                  p[m + i] = p[i] - t;
46                  p[i] += t;
47              }
48          }
49      }
50  }
51
52  void dft(Complex *a, const int &n) { transform(a, n, 0); }
53
54  void idft(Complex *a, const int &n) {
55      transform(a, n, 1);
56      for (int i = 0; i < n; ++i) a[i] /= n;
57  }
58
59  Complex A[maxn], B[maxn], C[maxn];
60
61  void FFT(int n, int m) {// len(A),len(B)
62      int cnt = 1;
63      while (cnt <= (n + m)) cnt <<= 1;
64      init(cnt);
65      dft(A, cnt);
66      dft(B, cnt);
67      for (int i = 0; i < cnt; i++) C[i] = A[i] * B[i];
68      idft(C, cnt);
69      for (int i = 0; i <= n + m; i++) C[i].r = ll(C[i].r + 0.01);
70  }
71
72  int main() {
73      int n, m, tem;
74      cin >> n >> m;
75      for (int i = 0; i <= n; i++) scanf("%d", &tem), A[i].r = tem;
76      for (int i = 0; i <= m; i++) scanf("%d", &tem), B[i].r = tem;
77      FFT(n, m);
78      for (int i = 0; i <= n + m; i++) printf("%lld ", ll(C[i].r));
79      cout << "\n";
80  }
81
82  // 两次变换
83  #define ld double
84  const ld PI = acosl(-1);
85  struct cplx {
86      ld a, b;
87      cplx(ld a = 0, ld b = 0) : a(a), b(b) {}
88      const cplx operator+(const cplx &c) const { return cplx(a + c.a, b + c.b); }
89      const cplx operator-(const cplx &c) const { return cplx(a - c.a, b - c.b); }
90      const cplx operator*(const cplx &c) const { return cplx(a * c.a - b * c.b, a * c.b
            + b * c.a); }
91      const cplx operator/(const ld &x) const { return cplx(a / x, b / x); }
92      const cplx conj() const { return cplx(a, -b); }
93  };
94  int rev[maxn];
95  cplx w[maxn];
96  cplx f[maxn];
```

```
97  void prepare(int &n) {
98      int sz = __builtin_ctz(n);
99      for (int i = 1; i < n; ++i) rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (sz - 1));
100     w[0] = 0, w[1] = 1, sz = 1;
101     while (1 << sz < n) {
102         cplx w_n = cplx(cosl(2 * PI / (1 << (sz + 1))), sinl(2 * PI / (1 << (sz + 1))))
                ;
103         for (int i = 1 << (sz - 1); i < (1 << sz); ++i) {
104             w[i << 1] = w[i], w[i << 1 | 1] = w[i] * w_n;
105         }
106         ++sz;
107     }
108 }
109 void fft(cplx *a, int n) {
110     for (int i = 1; i < n - 1; ++i) if (i < rev[i]) swap(a[i], a[rev[i]]);
111     for (int h = 1; h < n; h <<= 1) {
112         for (int s = 0; s < n; s += h << 1) {
113             for (int i = 0; i < h; ++i) {
114                 cplx &u = a[s + i], &v = a[s + i + h], t = v * w[h + i];
115                 v = u - t, u = u + t;
116             }
117         }
118     }
119 }
120 template<class T>
121 vector<T> multiply(const vector<T> &a, const vector<T> &b) {
122     int n = a.size(), m = b.size(), sz = 1;
123     while (sz < n + m - 1) sz <<= 1;
124     prepare(sz);
125     for (int i = 0; i < sz; ++i) f[i] = cplx(i < n ? a[i] : 0, i < m ? b[i] : 0);
126     fft(f, sz);
127     for (int i = 0; i <= (sz >> 1); ++i) {
128         int j = (sz - i) & (sz - 1);
129         cplx x = (f[i] * f[i] - (f[j] * f[j]).conj()) * cplx(0, -0.25);
130         f[j] = x, f[i] = x.conj();
131     }
132     fft(f, sz);
133     vector<T> c(n + m - 1);
134     for (int i = 0; i < n + m - 1; ++i) {
135         c[i] = ((T) (f[i].a / sz + 0.3));
136     }
137     return c;
138 }
139 #undef ld
```

1.5.2 NTT

```
1  const static int N = 4e6 + 6;
2  typedef long long ll;
3  ll pos[N];
4  const ll mod = 998244353, root = 3;
5  inline ll fastpow(ll a, ll b) {
6      ll ans = 1;
7      for (;b;a=a*a%mod,b>>=1) if(b&1)ans=ans*a%mod;
8      return ans;
9  }
10 inline void exgcd(ll a,ll b,ll &g,ll &x,ll &y) {
11     if (!b) g=a,x=1,y=0;
```

```
12      else exgcd(b,a%b,g,y,x),y-=x*(a/b);
13  }
14  inline ll inv(ll a) {
15     ll g,x,y;
16      exgcd(a,mod,g,x,y);
17      return (x%mod+mod)%mod;
18  }
19  void init(const int &n) {
20      for (int i = 0,j=0; i < n; ++i) {
21          pos[i]=j;for (int l = n >> 1; (j ^= l) < l; l >>= 1);
22      }
23  }
24  void transform(ll *a, const int &n, bool inverse) {
25      for (int i=0; i<n;++i) if(i>pos[i]) swap(a[i],a[pos[i]]);
26      for (int l=2; l<=n;l<<=1) {
27          int m=l/2;ll omega=fastpow(inverse?inv(root):root,(mod-1)/l);
28          for (ll *p=a;p!=a+n;p+=l) {
29              ll x=1;
30              for (int i=0;i<m;++i,x=x*omega%mod) {
31                  ll t=x*p[m+i]%mod;
32                  p[m+i]=(p[i]-t+mod)%mod;(p[i]+=t)%=mod;
33              }
34          }
35      }
36  }
37  void dft(ll *a, const int &n) {
38      transform(a,n,0);
39  }
40  void idft(ll *a, const int &n) {
41      const ll INV=inv(n);
42      transform(a,n,1);
43      for (int i=0;i<n;i++) a[i]=a[i]*INV % mod;
44  }
45  const int maxn=4e6+6;
46  ll a[maxn],b[maxn],c[maxn],pic[maxn],pec[maxn],plc[maxn],ppc[maxn];
47  int INV[maxn];
48  void poly_inv(int n,ll *a,ll *b) {
49      if(n==1) {b[0]=inv(a[0]);return;}
50      poly_inv((n+1)/2,a,b);
51      int cnt=1;while(cnt<=n*2) cnt<<=1;init(cnt);
52      copy(a,a+n,pic);fill(pic+n,pic+cnt,0);fill(b+n,b+cnt,0);dft(pic,cnt);dft(b,cnt);
53      for(int i=0;i<cnt;i++) (b[i]*=(2ll-pic[i]*b[i])%mod)%=mod;
54      for(int i=0;i<cnt;i++) b[i]=(b[i]+mod)%mod;
55      idft(b,cnt);fill(b+n,b+cnt,0);
56  }
57  void poly_ln(int n,ll *a,ll *b) { //G'=F'/F
58      poly_inv(n,a,b);
59      int cnt=1;while(cnt<=n*2-3) cnt<<=1;init(cnt);
60      for(int i=0;i<n-1;i++) plc[i]=a[i+1]*(i+1)%mod;
61      fill(plc+n-1,plc+cnt,0);fill(b+n,b+cnt,0);dft(plc,cnt);dft(b,cnt);
62      for(int i=0;i<cnt;i++) b[i]=plc[i]*b[i]%mod;
63      idft(b,cnt);for(int i=n-1;i>=1;i--) b[i]=b[i-1]*INV[i]%mod;b[0]=0;
64      fill(b+n,b+cnt,0);
65  }
66  void poly_exp(int n,ll *a,ll *b) {
67      if(n==1) {b[0]=1;return;}
68      poly_exp((n+1)/2,a,b);poly_ln(n,b,pec);
69      int cnt=1;while(cnt<=n*2-2) cnt<<=1;init(cnt);
70      for(int i=0;i<n;i++) pec[i]=(-pec[i]+a[i])%mod;pec[0]+=1;
```

```
    dft(pec,cnt);dft(b,cnt);
    for(int i=0;i<cnt;i++) b[i]=(b[i]*pec[i])%mod;
    idft(b,cnt);fill(b+n,b+cnt,0);
}
void poly_pow(int n,ll *a,ll *b,ll k) {
    poly_ln(n,a,ppc);
    for(int i=0;i<n;i++) ppc[i]*=k;
    poly_exp(n,ppc,b);
}
char buff[maxn];
int main(){
    ios::sync_with_stdio(false);
    cin.tie(0);cout.tie(0);
    int n;cin>>n>>buff;
    INV[1]=1;for(int i=2;i<n;++i) INV[i]=(mod-mod/i)*INV[mod%i]%mod;
    ll k=0;
    for(int i=0;buff[i];i++) k=(k*10+buff[i]-'0')%mod;
    for(int i=0;i<n;i++) cin>>a[i];
    poly_pow(n,a,b,k);
    for(int i=0;i<n;i++) cout<<b[i]<<' ';
}
```

1.5.3 MTT

```
const int maxn = 262144;
int mod = 1e9+7;

namespace MTT {
    struct comp {
        double x, y;
        comp(double x_ = 0, double y_ = 0): x(x_), y(y_) {}
    };
    inline comp operator + (const comp& a, const comp& b) { return comp(a.x + b.x, a.y + b.y); }
    inline comp operator - (const comp& a, const comp& b) { return comp(a.x - b.x, a.y - b.y); }
    inline comp operator * (const comp& a, const comp& b) { return comp(a.x * b.x - a.y * b.y, a.x * b.y + a.y * b.x); }
    inline comp conj(const comp& a) { return comp(a.x, -a.y); }

    const double PI = acosl(-1);
    comp w[maxn + 5];
    int bitrev[maxn + 5];
    int N, L;

    void fft(vector<comp>& a) {
        for (int i = 0; i < N; ++i) if (i < bitrev[i]) swap(a[i], a[bitrev[i]]);
        for (int i = 2, lyc = N >> 1; i <= N; i <<= 1, lyc >>= 1) {
            for (int j = 0; j < N; j += i) {
                int pl = j, pr = j + (i >> 1), p = 0;
                for (int k = 0; k < (i >> 1); ++k) {
                    comp tmp = a[pr] * w[p];
                    a[pr] = a[pl] - tmp, a[pl] = a[pl] + tmp;
                    ++ pl, ++ pr, p += lyc;
                }
            }
        }
    }
```

```
32
33      inline void fft_prepare() {
34          for (int i = 0; i < N; ++i) bitrev[i] = bitrev[i >> 1] >> 1 | ((i & 1) << (L -
                1));
35          for (int i = 0; i < N; ++i) w[i] = comp(cos(2 * PI * i / N), sin(2 * PI * i / N
                ));
36      }
37
38      // if max ans is n+m, n=n+m+1
39      void init(int n) {
40          L = 0;
41          while ((1 << L) < n) ++ L;
42          N = 1 << L;
43          fft_prepare();
44      }
45
46      inline vector<int> conv(vector<int> x, vector<int> y) {
47          vector<comp> a(N + 5), b(N + 5);
48          vector<comp> dfta(N + 5), dftb(N + 5), dftc(N + 5), dftd(N + 5);
49          for (int i = 0; i < (int)x.size(); ++i) {
50              if ((x[i] %= mod) < 0) x[i] += mod;
51              a[i] = comp(x[i] & 32767, x[i] >> 15);
52          }
53          for (int i = 0; i < (int)y.size(); ++i) {
54              if ((y[i] %= mod) < 0) y[i] += mod;
55              b[i] = comp(y[i] & 32767, y[i] >> 15);
56          }
57          fft(a), fft(b);
58          for (int i = 0; i < N; ++i) {
59              int j = (N - i) & (N - 1);
60              comp da = (a[i] + conj(a[j])) * comp(0.5, 0);
61              comp db = (a[i] - conj(a[j])) * comp(0, -0.5);
62              comp dc = (b[i] + conj(b[j])) * comp(0.5, 0);
63              comp dd = (b[i] - conj(b[j])) * comp(0, -0.5);
64              dfta[j] = da * dc;
65              dftb[j] = da * dd;
66              dftc[j] = db * dc;
67              dftd[j] = db * dd;
68          }
69          for (int i = 0; i < N; ++i) a[i] = dfta[i] + dftb[i] * comp(0, 1);
70          for (int i = 0; i < N; ++i) b[i] = dftc[i] + dftd[i] * comp(0, 1);
71          fft(a), fft(b);
72          vector<int> z(N);
73          for (int i = 0; i < N; ++i) {
74              int da = (ll)(a[i].x / N + 0.5) % mod;
75              int db = (ll)(a[i].y / N + 0.5) % mod;
76              int dc = (ll)(b[i].x / N + 0.5) % mod;
77              int dd = (ll)(b[i].y / N + 0.5) % mod;
78              z[i] = (da + ((ll)(db + dc) << 15) + ((ll)dd << 30)) % mod;
79              if (z[i] < 0) z[i] += mod;
80          }
81          while (!z.empty() && z.back() == 0) z.pop_back();
82          return z;
83      }
84  }
85
86  int main(int argc, char* argv[]) {
87      int n, m;
88      R(n, m);
```

```
    VI a(n + 5), b(m + 5);
    MTT::init(n + m + 1);
    for (int i = 0; i <= n; ++i) R(a[i]);
    for (int i = 0; i <= m; ++i) R(b[i]);
    auto res = MTT::conv(a, b);
    while (res.size() < (n + m + 1)) res.push_back(0);
    W(res);
    return 0;
}
```

1.5.4 FWT

```
//FWT 完后需要先模一遍
template<typename T>
void fwt(ll a[], int n, T f) {
    for (int d = 1; d < n; d *= 2)
        for (int i = 0, t = d * 2; i < n; i += t)
            for(int j = 0; j < d; j ++)
                f(a[i + j], a[i + j + d]);
}

void AND(ll& a, ll& b) { a += b; }
void OR(ll& a, ll& b) { b += a; }
void XOR (ll& a, ll& b) {
    ll x = a, y = b;
    a = (x + y) % mod;
    b = (x - y + mod) % mod;
}
void rAND(ll& a, ll& b) { a -= b; }
void rOR(ll& a, ll& b) { b -= a; }
void rXOR(ll& a, ll& b) {
    static ll INV2 = (mod + 1) / 2;
    ll x = a, y = b;
    a = (x + y) * INV2 % mod;
    b = (x - y + mod) * INV2 % mod;
}

//FWT 子集卷积 i与j=0, i或j为k
a[popcount(x)][x] = A[x]
b[popcount(x)][x] = B[x]
fwt(a[i]) fwt(b[i])
c[i + j][x] += a[i][x] * b[j][x]
rfwt(c[i])
ans[x] = c[popcount(x)][x]
```

1.5.5 杜教筛

```
const int maxn=3e6+5;

unordered_map<int,ll> Mmu;
unordered_map<int,ll> Mphi;
ll mu[maxn],phi[maxn],prim[maxn],pcnt,mn[maxn];
void get_T() {
    phi[1]=mu[1]=1;
    for(int i=2;i<maxn;i++) {
        if(!mn[i]) {
            mn[i]=i;prim[++pcnt]=i;
            phi[i]=i-1;mu[i]=-1;
```

```
        }
        for(int j=1;j<=pcnt&&i*prim[j]<maxn;j++) {
            mn[i*prim[j]]=prim[j];
            if(i%prim[j]==0) {
                phi[i*prim[j]]=phi[i]*prim[j];
                break;
            }
            mu[i*prim[j]]=-mu[i];
            phi[i*prim[j]]=phi[i]*(prim[j]-1);
        }
    }
    for(int i=1;i<maxn;i++)
        mu[i]+=mu[i-1],phi[i]+=phi[i-1];
}
ll Smu(int n) {
    if(n==1) return 1;
    if(n<maxn) return mu[n];
    if(Mmu.find(n)!=Mmu.end()) return Mmu[n];
    ll ans=0;
    for(int l=2,r;l<=n;l=r+1) {
        r=min(n,n/(n/l));
        ans+=Smu(n/l)*(r-l+1);
    }
    return Mmu[n]=1-ans;
}
ll Sphi(int n) {
    if(n==1) return 1;
    if(n<maxn) return phi[n];
    if(Mphi.find(n)!=Mphi.end()) return Mphi[n];
    ll ans=0;
    for(int l=2,r;l<=n;l=r+1) {
        r=min(n,n/(n/l));
        ans+=Sphi(n/l)*(r-l+1);
    }
    return Mphi[n]=1ll*n*(n+1)/2-ans;
}
int main() {
    get_T();
    int T;cin>>T;while(T--) {
        int n;cin>>n;
        cout<<Sphi(n)<<' '<<Smu(n)<<endl;
    }
    return 0;
}
```

1.5.6 min25

```
const int maxn=1e6+5;
int primer[maxn],pcnt;
bool insp[maxn];
ll fsum1[maxn],fsum2[maxn];
const int mod=1e9+7,INV6=166666668;
void sieve() {
    for(int i=2;i<maxn;i++) {
        if(!insp[i]) {
            primer[++pcnt]=i;
            fsum1[pcnt]=(fsum1[pcnt-1]+i)%mod;
            fsum2[pcnt]=(fsum2[pcnt-1]+1ll*i*i)%mod;
```

```
12          }
13          for(int j=1;j<=pcnt&&primer[j]*i<maxn;j++) {
14              insp[i*primer[j]]=1;
15              if(i%primer[j]==0) break;
16          }
17      }
18  }
19
20  // Let g(j,m) be the sum of f'(i) for all i which is not greater
21  // than m and is prime or the min pri-factor of i > p[j]
22  // We got that g(j,m)=g(j-1,m)-f'(p[j])*( g(j-1,m/p[j])-\sum_{i=1}^{j-1}f'(p[i]) )
23  // g(0,m) = \sum_{i=2}^m f'(j)
24  // Here f'(j) is a function whose value equals f(j) when j is prime
25  // and f'(j) is an acompletely multiplicative function
26  // We can use DP to calc g(j,{n/1,n/2,...,n/n}) (sqrt(n) situation in total)
27  // So g(j,m) equals to the sum of f(i) for all primer i which not greater than m
28  //  when p[j]*p[j] > m
29  int id1[maxn],id2[maxn],tot=0;
30  ll val[maxn],n,g1[maxn],g2[maxn];
31  //index of n/1,n/2....n/n
32  inline int getid(ll m) {
33      if(m<=n/m) return id1[m];
34      return id2[n/m];
35  }
36  void init() {
37      for(ll l=1,r;l<=n;l=r+1) {
38          r=n/(n/l);ll w=n/l;
39          val[++tot]=w;
40          if(w<=n/w) id1[w]=tot;
41          else id2[n/w]=tot;
42          w%=mod;
43          g1[tot]=w*(w+1)/2%mod;
44          g2[tot]=w*(w+1)%mod*(2ll*w+1)%mod*INV6%mod;
45          g1[tot]=(g1[tot]-1+mod)%mod;
46          g2[tot]=(g2[tot]-1+mod)%mod;
47      }
48      // Start DP
49      for(int i=1;1ll*primer[i]*primer[i]<=n;i++) {
50          for(int j=1;j<=tot&&1ll*primer[i]*primer[i]<=val[j];j++) {
51              int k=getid(val[j]/primer[i]);
52              g1[j]=(g1[j]-1ll*primer[i]*(g1[k]-fsum1[i-1]+mod)%mod+mod)%mod;
53              g2[j]=(g2[j]-1ll*primer[i]*primer[i]%mod*(g2[k]-fsum2[i-1]+mod)%mod+mod)%
                    mod;
54          }
55      }
56  }
57  ll getS(ll m,int k) {
58      if(m<primer[k]) return 0;
59      ll res=(g2[getid(m)]-g1[getid(m)]-fsum2[k-1]+fsum1[k-1])%mod;
60      for(int i=k;1ll*primer[i]*primer[i]<=m;i++) {
61          for(ll pie=primer[i],pie1=pie*primer[i];pie1<=m;pie=pie1,pie1*=primer[i]){
62              ll t1=pie%mod,t2=pie1%mod;
63              res=(res+t1*(t1-1)%mod*getS(m/pie,i+1)%mod+t2*(t2-1)%mod)%mod;
64          }
65      }
66      return (res+mod)%mod;
67  }
68  int main() {
69      sieve();cin>>n;
```

```
70      init();
71      // for(int i=1;i<=10;i++) cout<<g1[i]<<endl;
72      // cout<<g1[1]<<endl;
73      cout<<(getS(n,1)+1)%mod<<endl;
74      return 0;
75  }
```

1.6 Others

1.6.1 公式

1. 约数定理：若 $n=\prod_{i=1}^{k}p_i^{a_i}$，则
   (a) 约数个数 $f(n)=\prod_{i=1}^{k}(a_i+1)$
   (b) 约数和 $g(n)=\prod_{i=1}^{k}(\sum_{j=0}^{a_i}p_i^j)$
2. 小于 n 且互素的数之和为 $n\varphi(n)/2$
3. 若 $\gcd(n,i)=1$，则 $\gcd(n,n-i)=1(1\le i\le n)$
4. 错排公式：$D(n)=(n-1)(D(n-2)+D(n-1))=\sum_{i=2}^{n}\frac{(-1)^k n!}{k!}=[\frac{n!}{e}+0.5]$
5. 部分错排公式：n + m 个数中 m 个数必须错排求排列数
   (a) 1 dp[i] = n*dp[i−1]+(i−1)*(dp[i−1]+dp[i−2]);
   (b) 2 dp[0] = n!;
   (c) 3 dp[1] = n*n!;
   (d) dp[m] 为所求解
6. 海伦公式：$S=\sqrt{p(p-a)(p-b)(p-c)}$，其中 $p=\frac{(a+b+c)}{2}$
7. 求 $C(n,k)$ 中素因子 P 的个数：把 n 转化为 P 进制，并记它每个位上的和为 $S1$ 把 $n-k$，k 做同样的处理，得到 $S2$，$S3$ 则答案为：$\frac{S2+S3-S1}{P-1}$
8. 威尔逊定理：$p\ is\ prime\ \Rightarrow (p-1)!\equiv -1\pmod p$
9. 欧拉定理：$\gcd(a,n)=1\Rightarrow a^{\varphi(n)}\equiv 1\pmod n$
10. 欧拉定理推广：$\gcd(n,p)=1\Rightarrow a^n\equiv a^{n\%\varphi(p)}\pmod p$
11. 模的幂公式：$a^n\pmod m=\begin{cases}a^n \bmod m & n<\varphi(m)\\ a^{n\%\varphi(m)+\varphi(m)}\bmod m & n\ge\varphi(m)\end{cases}$
12. 素数定理：对于不大于 n 的素数个数 $\pi(n)$，$\lim\limits_{n\to\infty}\pi(n)=\frac{n}{\ln n}$
13. 位数公式：正整数 x 的位数 $N=\log_{10}(n)+1$
14. 斯特灵公式 $n!\approx\sqrt{2\pi n}(\frac{n}{e})^n$
15. 设 $a>1,m,n>0$, 则 $\gcd(a^m-1,a^n-1)=a^{\gcd(m,n)}-1$
16. 设 $a>b,\gcd(a,b)=1$, 则 $\gcd(a^m-b^m,a^n-b^n)=a^{\gcd(m,n)}-b^{\gcd(m,n)}$

$$G=\gcd(C_n^1,C_n^2,...,C_n^{n-1})=\begin{cases}n, & n\text{ is prime}\\ 1, & n\text{ has multy prime factors}\\ p, & n\text{ has single prime factor }p\end{cases}$$

$\gcd(Fib(m),Fib(n))=Fib(\gcd(m,n))$

17. 求和公式：
   (a) $\sum k=\frac{n(n+1)}{2}$
   (b) $\sum 2k-1=n^2$
   (c) $\sum k^2=\frac{n(n+1)(2n+1)}{6}$
   (d) $\sum(2k-1)^2=\frac{n(4n^2-1)}{3}$

(e) $\sum k^3 = (\frac{n(n+1)}{2})^2$

(f) $\sum (2k-1)^3 = n^2(2n^2-1)$

(g) $\sum k^4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30}$

(h) $\sum k^5 = \frac{n^2(n+1)^2(2n^2+2n-1)}{12}$

(i) $\sum k(k+1) = \frac{n(n+1)(n+2)}{3}$

(j) $\sum k(k+1)(k+2) = \frac{n(n+1)(n+2)(n+3)}{4}$

(k) $\sum k(k+1)(k+2)(k+3) = \frac{n(n+1)(n+2)(n+3)(n+4)}{5}$

18. 若 $\gcd(m,n)=1$, 则:

(a) 最大不能组合的数为 $m*n-m-n$

(b) 不能组合数个数 $N = \frac{(m-1)(n-1)}{2}$

19. $(n+1)lcm(C_n^0, C_n^1, ..., C_n^{n-1}, C_n^n) = lcm(1,2,...,n+1)$

20. 若 p 为素数，则 $(x+y+...+w)^p \equiv x^p + y^p + ... + w^p \pmod p$

21. 卡特兰数：1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012
$h(0)=h(1)=1, h(n) = \frac{(4n-2)h(n-1)}{n+1} = \frac{C_{2n}^n}{n+1} = C_{2n}^n - C_{2n}^{n-1}$

22. 伯努利数：$B_n = -\frac{1}{n+1}\sum_{i=0}^{n-1} C_{n+1}^i B_i$

$$\sum_{i=1}^{n} i^k = \frac{1}{k+1}\sum_{i=1}^{k+1} C_{k+1}^i B_{k+1-i}(n+1)^i$$

23. 二项式反演：

$$f_n = \sum_{i=0}^{n}(-1)^i \binom{n}{i} g_i \Leftrightarrow g_n = \sum_{i=0}^{n}(-1)^i \binom{n}{i} f_i$$

$$f_n = \sum_{i=0}^{n}\binom{n}{i} g_i \Leftrightarrow g_n = \sum_{i=0}^{n}(-1)^{n-i} \binom{n}{i} f_i$$

24. 莫比乌斯反演:

(a) 令 $f(d) = \sum_{i=1}^{n}\sum_{j=1}^{m}[gcd(i,j)=d]$

(b) $F(n) = \sum_{n|d} f(d) = \lfloor \frac{N}{n} \rfloor \lfloor \frac{M}{n} \rfloor$

(c) 有 $f(n) = \sum_{n|d} \mu(\lfloor \frac{d}{n} \rfloor) F(d)$

(d) $\phi(n) = \sum_{d|n} d * \mu(n/d)$

25. 2 的 n 次方，在 pow 时可以精确输出最大 2^1023，pow(2,1023)

26. FFT 常用素数

$r\ 2^k+1$	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	3
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

1.7 高数

1.7.1 拉格朗日插值

```
const int maxn = 1e5 + 10;
int x[maxn], y[maxn];
int qp(int a, int n) {
    ll ans = 1, base = a;
    for(; n; (base *= base) %= mod, n >>= 1) if(n & 1) (ans *= base) %= mod;
    return ans;
}

int lagrange(int n, int *x, int *y, int xi) {
    int ans = 0;
    for(int i = 0; i <= n; i ++) {
        int s1 = 1, s2 = 1;
        for(int j = 0; j <= n; j ++) if(i != j) {
```

```
14              s1 = 1ll * s1 * (xi - x[j]) % mod;
15              s2 = 1ll * s2 * (x[i] - x[j]) % mod;
16          }
17          ans = (1ll * ans + 1ll * y[i] * s1 % mod * qp(s2, mod - 2) % mod) % mod;
18      }
19      return (ans + mod) % mod;
20  }
```

1.7.2 拉格朗日插值 (连续取值)

```
1   int x[maxn], y[maxn];
2   int s1[maxn], s2[maxn], ifac[maxn];
3
4   //如果x的取值是连续一段, 可以做到O(n)求解
5   int lagrange(int n, int *x, int *y, int xi) {
6       int ans = 0;
7       s1[0] = (xi - x[0]) % mod, s2[n + 1] = 1;
8       for(int i = 1; i <= n; i ++) s1[i] = 1ll * s1[i - 1] * (xi - x[i]) % mod;
9       for(int i = n; i >= 0; i --) s2[i] = 1ll * s2[i + 1] * (xi - x[i]) % mod;
10      ifac[0] = ifac[1] = 1;
11      for(int i = 2; i <= n; i ++) ifac[i] = -1ll * mod / i * ifac[mod % i] % mod;
12      for(int i = 2; i <= n; i ++) ifac[i] = 1ll * ifac[i] * ifac[i - 1] % mod;
13      for(int i = 0; i <= n; i ++)
14          (ans += 1ll * y[i] * (i == 0 ? 1 : s1[i - 1]) % mod * s2[i + 1] % mod * ifac[i]
                % mod * (((n - i) & 1) ? -1 : 1) * ifac[n - i] % mod) %= mod;
15      return (ans + mod) % mod;
16  }
```

1.7.3 辛普森积分

```
1   typedef double ld;
2   ld a, b, c, d;
3   inline ld f(ld x) { return (c * x + d) / (a * x + b); }
4   inline ld simpson(ld l, ld r, ld fl, ld fr, ld fm) { return (r - l) * (fl + fr + 4 * fm
        ) / 6; }
5
6   inline ld asr(ld l, ld r, ld eps, ld fl, ld fr, ld fm) {
7       ld flm = f((l + (l + r) / 2) / 2), frm = f((r + (l + r) / 2) / 2);
8       ld L = simpson(l, (l + r) / 2, fl, fm, flm), R = simpson((l + r) / 2, r, fm, fr,
            frm), A = simpson(l, r, fl, fr, fm);
9       if (fabs(L + R - A) <= 15 * eps) return L + R + (L + R - A) / 15;
10      return asr(l, (l + r) / 2, eps / 2, fl, fm, flm) + asr((l + r) / 2, r, eps / 2, fm,
             fr, frm);
11  }
12
13  inline ld cal(ld l, ld r) { return asr(l, r, 1e-10, f(l), f(r), f((l + r) / 2)); }
```

2 Graph Theory

2.1 路径

2.1.1 Dijkstra

```
const int maxn = 1e5 + 10;
const int inf = 0x3f3f3f3f;

int head[maxn], dis[maxn], cnt, n;

struct Edge { int nex,to,w; }edge[20*maxn];

void add(int u,int v,int w)
{
    edge[++cnt].nex=head[u];
    edge[cnt].w=w;
    edge[cnt].to=v;
    head[u]=cnt;
}

void dijkstra(int s)
{
    priority_queue<pair<int, int>, vector<pair<int, int> >, greater<pair<int, int> > >
        que;
    memset(dis, 0x3f, sizeof dis);
    que.push({0, s}); dis[s] = 0;
    while(!que.empty())
    {
        auto f = que.top(); que.pop();
        int u = f.second, d = f.first;
        if(d != dis[u]) continue;
        for(int i = head[u]; ~i; i = edge[i].nex)
        {
            int v = edge[i].to, w = edge[i].w;
            if(dis[u] + w < dis[v])
            {
                dis[v] = dis[u] + w;
                que.push({dis[v], v});
            }
        }
    }
}
```

2.1.2 Euler Path

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

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

```
    }
}

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

2.1.3 K shortest Path(Astar)

```
// 有向图 k 短路
const int N = 1010, M = 10010, inf = 1e9+50;
int n, m;
int g[N], h[N], v[M<<1], w[M<<1], nxt[M<<1], ed, d[N], vis[N], ans[N];
typedef pair<int, int> P;
priority_queue<P, vector<P>, greater<P> > Q;
void add(int x, int y, int z) {
    v[++ed] = x, w[ed] = z, nxt[ed] = g[y], g[y] = ed;
    v[++ed] = y, w[ed] = z, nxt[ed] = h[x], h[x] = ed;
}

int KthShortest(int S, int T, int k) {
    int x;
    for (int i = 1; i <= k; ++i) ans[i] = -1;
    for (int i = 1; i <= n; ++i) d[i] = inf;
    Q.push(P(d[T] = 0, T));
    while (!Q.empty()) {
        P t = Q.top(); Q.pop();
        if (d[t.second] < t.first) continue;
        for (int i = g[x = t.second]; i; i = nxt[i]) {
            if (d[x] + w[i] < d[v[i]]) Q.push(P(d[v[i]] = d[x] + w[i], v[i]));
        }
    }
    if (d[S] < inf) Q.push(P(d[S], S));
    while (!Q.empty()) {
        P t = Q.top(); Q.pop(); vis[x = t.second] ++;
        if (x == T && vis[T] <= k) ans[vis[T]] = t.first;
        if (vis[T] > k) break;
        if (vis[x] <= k) for (int i = h[x]; i; i = nxt[i]) {
            Q.push(P(t.first - d[x] + d[v[i]] + w[i], v[i]));
        }
    }
    return ans[k];
}
```

2.1.4 K shortest Path(可持久化可并堆)

```
#include <bits/stdc++.h>
#include<ext/pb_ds/priority_queue.hpp>
```

```
3
4  using namespace std;
5
6  const int N = '';
7  const int M = '';
8  const int logM = 20;
9  const int inf = 0x3f3f3f3f;
10
11 int n, m, k, S, T;
12
13 struct Edge{ int nex, to, w; };
14
15 struct Graph
16 {
17     int head[N], cnt;
18     Edge edge[M];
19     void init(int n) { for(int i = 0; i <= n; i ++) head[i] = 0; cnt = 0; }
20     void addedge(int u, int v, int val) { edge[++ cnt].nex = head[u], edge[cnt].to = v,
           edge[cnt].w = val, head[u] = cnt; }
21 }g, rg;
22
23 int dis[N];
24
25 void dijkstra()
26 {
27     priority_queue<pair<int, int>, vector<pair<int, int> >, greater<pair<int, int> > >
          que;
28     memset(dis, inf, sizeof dis);
29     que.push({0, T}); dis[T] = 0;
30     const int *head = rg.head; const Edge *edge = rg.edge;
31     while(!que.empty())
32     {
33         auto f = que.top(); que.pop();
34         int u = f.second, d = f.first;
35         if(d != dis[u]) continue;
36         for(int i = head[u]; i; i = edge[i].nex)
37         {
38             int v = edge[i].to, w = edge[i].w;
39             if(dis[u] + w < dis[v]) { dis[v] = dis[u] + w; que.push({dis[v], v}); }
40         }
41     }
42 }
43
44 bool tree_edge[M], vis[N];
45 int fa[N], st[N], top;
46
47 void dfs(int u)
48 {
49     vis[u] = true;
50     st[++ top] = u;
51     for(int i = rg.head[u]; i; i = rg.edge[i].nex)
52     {
53         int v = rg.edge[i].to;
54         if(!vis[v] && dis[v] == dis[u] + rg.edge[i].w)
55         {
56             fa[v] = u;
57             tree_edge[i] = true;
58             dfs(v);
59         }
```

```
60      }
61  }
62
63  namespace LT
64  {
65      int son[M * logM][2];
66      int ht[M * logM], val[M * logM], id[M * logM];
67      int tot;
68
69      int newnode(int _val, int _id, int _dis = 0)
70      {
71          int now = ++ tot;
72          val[now] = _val, id[now] = _id;
73          ht[now] = _dis, son[now][0] = son[now][1] = 0;
74          return now;
75      }
76
77      int _copy(int ori)
78      {
79          int now = ++tot;
80          val[now] = val[ori], id[now] = id[ori];
81          ht[now] = ht[ori], son[now][0] = son[ori][0], son[now][1] = son[ori][1];
82          return now;
83      }
84
85      int merge(int a, int b)
86      {
87          if(!a || !b) return a | b;
88          if(val[a] > val[b]) swap(a, b);
89          int now = _copy(a);
90          son[now][1] = merge(son[now][1], b);
91          if(ht[son[now][0]] < ht[son[now][1]]) swap(son[now][0], son[now][1]);
92          ht[now] = ht[son[now][1]] + 1;
93          return now;
94      }
95
96      void insert(int &rt, int val, int id) { rt = merge(newnode(val, id), rt); }
97  }
98
99  int rt[M];
100
101 void build_heap()
102 {
103     for(int i = 1; i <= top; i ++)
104     {
105         int u = st[i];
106         rt[u] = rt[fa[u]];
107         for(int i = g.head[u]; i; i = g.edge[i].nex)
108         {
109             int v = g.edge[i].to;
110             if(!tree_edge[i] && dis[v] != inf) LT::insert(rt[u], dis[v] - dis[u] + g.
                    edge[i].w, v);
111         }
112     }
113 }
114
115 int solve(int k)
116 {
117     if(k == 1) return dis[S];
```

```
    __gnu_pbds::priority_queue<pair<int, int>, greater<pair<int, int> > > que;
    que.push({dis[S] + LT::val[rt[S]], rt[S]});
    while(!que.empty())
    {
        pair<int, int> f = que.top(); que.pop();
        if((--k) == 1) return f.first;
        int v = f.first, u = f.second;
        int lc = LT::son[u][0], rc = LT::son[u][1], o = LT::id[u];
        if(rt[o]) que.push({v + LT::val[rt[o]], rt[o]});
        if(lc) que.push({v + LT::val[lc] - LT::val[u], lc});
        if(rc) que.push({v + LT::val[rc] - LT::val[u], rc});
    }
    return -1;
}

void init()
{
    g.init(n), rg.init(n);
    memset(rt, 0, sizeof rt);
    memset(tree_edge, 0, sizeof tree_edge);
    top = LT::tot = 0;
}

void getans()
{
    //input S-T
    init();
    dijkstra();
    dfs(T);
    build_heap();
    cout << solve(k);
}
```

2.2 生成树

2.2.1 Matrix Tree

```
const int N = 305;
const int mod = 1e9 + 7;

int n, m, a[N][N];

int Gauss(int n) {
    int ans = 1;
    for (int i = 1; i <= n; i++) {
        for (int k = i + 1; k <= n; k++) {
            while (a[k][i]) {
                int d = a[i][i] / a[k][i];
                for (int j = i; j <= n; j++) {
                    a[i][j] = (a[i][j] - 1LL * d * a[k][j] % mod + mod) % mod;
                }
                std::swap(a[i], a[k]);
                ans = -ans;
            }
        }
        ans = 1LL * ans * a[i][i] % mod;
    }
    return (ans % mod + mod) % mod;
}
```

```
23
24 int main() {
25     scanf("%d%d", &n, &m);
26     for (int i = 1; i <= m; i++) {
27         int u, v;
28         scanf("%d%d", &u, &v);
29         a[u][v]--, a[v][u]--;
30         a[u][u]++, a[v][v]++;
31     }
32     printf("%d\n", Gauss(n - 1));
33     return 0;
34 }
```

2.2.2 Steiner Tree

```
1  /*BZOJ:4774
2  无向图G从1-n进行编号，选择一些边，使对于1<=i<=d,i号点和n-i+1号点连通，最小化选出的所有
       边权值和。
3  1. 枚举子树形态 $dp[S][i] = min(dp[s]+dp[S \ xor \  s])$
4  2. 按照边进行松弛 $dp[S][i] = min(dp[S][j]+w[j][i])$
5  其中$S$为选取的子集，$s$ 和$S\ xor\ s$为$S$的状态划分。第二类转移方程可以通过跑一次最短
       路进行松弛。
6  本题需要再做一次子集dp，因为不成对的点可能不连通。
7  */
8  #include <bits/stdc++.h>
9
10 using namespace std;
11
12 const int maxn = 1e4 + 10;
13 const int inf = 0x3f3f3f3f;
14
15 int head[maxn], cnt;
16 struct Edge {int nex, to, w; }edge[maxn<<1];
17
18 void add(int u, int v, int w)
19 {
20     edge[cnt].nex = head[u];
21     edge[cnt].to = v;
22     edge[cnt].w = w;
23     head[u] = cnt++;
24 }
25
26 int f[1<<10][maxn], ans[20];
27 bool in[maxn];
28
29 queue<int> que;
30
31 void spfa(int S)
32 {
33     while(!que.empty())
34     {
35         int u = que.front(); que.pop();
36         in[u] = false;
37         for(int i = head[u]; ~i; i = edge[i].nex)
38         {
39             int v = edge[i].to;
40             if(f[S][v] > f[S][u] + edge[i].w)
41             {
```

```
                f[S][v] = f[S][u] + edge[i].w;
                if(!in[v]) que.push(v), in[v] = true;
            }
        }
    }
}

int Steiner_Tree(int n, int d)
{
    memset(f, 0x3f, sizeof f);
    for(int i = 1; i <= d; i++)
        f[1 << (i - 1)][i] = f[1 << (d + i - 1)][n - i + 1] = 0;
    int lim = 1<<(d<<1);
    for(int S = 1; S < lim; S++)
    {
        for(int i = 1; i <= n; i++)
        {
            for(int s = (S - 1) & S; s; s = (s - 1) & S)
                f[S][i] = min(f[S][i], f[s][i] + f[S ^ s][i]);
            if(f[S][i] != inf) que.push(i), in[i] = true;
        }
        spfa(S);
    }
    lim = 1<<d;
    memset(ans, 0x3f, sizeof ans);
    for(int S = 1; S < lim; S++)
        for(int i = 1; i <= n; i++)
            ans[S] = min(ans[S], f[S^(S<<d)][i]);
    for(int S = 1; S < lim; S++)
        for(int s = (S - 1) & S; s; s = (s - 1) & S)
            ans[S] = min(ans[S], ans[s] + ans[S ^ s]);
    return ans[lim - 1] == inf ? -1 : ans[lim - 1];
}

int main()
{
    int n, m, d, u, v, w;
    scanf("%d%d%d", &n, &m, &d);
    memset(head, 0xff, sizeof head);
    while(m--)
    {
        scanf("%d%d%d", &u, &v, &w);
        add(u, v, w);
        add(v, u, w);
    }
    printf("%d\n", Steiner_Tree(n, d));
    return 0;
}
```

2.2.3 最小树形图

```
const int INF = 0x3f3f3f3f;
const int maxn = 10000;
const int maxm = 10000;

struct Edge{int u,v,cost; } edge[maxm];

int pre[maxn], id[maxn], vis[maxn], in[maxn];
```

```
8
9  int zhuliu(int root, int n, int m)
10 {
11     int res=0, u, v;
12     for(;;)
13     {
14         for(int i=0; i<n; i++) in[i] = INF;
15         for(int i=0; i<m; i++) if(edge[i].u != edge[i].v && edge[i].cost < in[edge[i].v
               ])
16         {
17             pre[edge[i].v] = edge[i].u;
18             in[edge[i].v] = edge[i].cost;
19         }
20         for(int i=0; i<n; i++) if(i != root && in[i] ==INF) return -1;
21         int tn=0;
22         memset(id, 0xff, sizeof id);
23         memset(vis, 0xff, sizeof vis);
24         in[root] = 0;
25         for(int i=0; i<n;i++)
26         {
27             res += in[i];
28             v = i;
29             while( vis[v] != i && id[v] == -1 && v!= root) vis[v] = i, v = pre[v];
30             if(v != root && id[v] == -1)
31             {
32                 for(int u = pre[v]; u != v; u = pre[u]) id[u] = tn;
33                 id[v] = tn++;
34             }
35         }
36         if(tn == 0) break;
37         for(int i=0; i<n; i++) if(id[i] == -1) id[i] = tn++;
38         for(int i=0; i<m; )
39         {
40             v = edge[i].v;
41             edge[i].u = id[edge[i].u];
42             edge[i].v = id[edge[i].v];
43             if(edge[i].u != edge[i].v) edge[i++].cost -= in[v];
44             else swap(edge[i], edge[--m]);
45         }
46         n = tn;
47         root = id[root];
48     }
49     return res;
50 }
```

2.3 连通性

2.3.1 割点

```
1  const int maxn = 1e4 + 10;
2
3  vector<int> edge[maxn];
4  int n, dfn[maxn], low[maxn], cnt = 0;
5  bool vis[maxn], cut[maxn];
6
7  void Tarjan(int u, int fa)
8  {
9      dfn[u] = low[u] = ++cnt;
10     vis[u] = true;
```

```
11      int children = 0;
12      for (int i = 0; i < edge[u].size(); i++)
13      {
14          int v = edge[u][i];
15          if (v != fa && vis[v])
16              low[u] = min(low[u], dfn[v]);
17          else if (!vis[v])
18          {
19              Tarjan(v, u);
20              children++;
21              low[u] = min(low[u], low[v]);
22              if (fa == -1 && children > 1) //若u是根节点且子节点数大于1
23                  cut[u] = true;    //u是割点
24              else if (fa != -1 && low[v] >= dfn[u])    //若u不是根节点且v不能访问到u的父
                    节点
25                  cut[u] = true;    //u是割点
26          }
27      }
28  }
```

2.3.2 桥

```
1   const int maxn = 1e4 + 10;
2
3   vector<int> edge[maxn];
4   int n, dfn[maxn], low[maxn], father[maxn], cnt = 0;
5   bool bridge[maxn][maxn];
6
7   void Tarjan(int u, int fa)
8   {
9       dfn[u] = low[u] = ++cnt;
10      for (int i = 0; i < edge[u].size(); i++)
11      {
12          int v = edge[u][i];
13          if (!dfn[v])    //未访问节点v
14          {
15              Tarjan(v, u);
16              low[u] = min(low[u], low[v]);
17              if (low[v] > dfn[u]) //节点v到达祖先必须经过(u,v)
18                  bridge[u][v] = bridge[v][u] = true;    //(u,v)是桥
19          }
20          else if (fa != v)    //u的父节点不是v, (u,v)不存在重边
21              low[u] = min(low[u], dfn[v]);
22      }
23  }
```

2.3.3 强连通分量

```
1   const int maxn=1000+10;
2
3   vector<int> edge[maxn];
4
5   int dfn[maxn], low[maxn];
6   int stack[maxn], index, tot;
7   int belong[maxn], inde[maxn], outde[maxn], scc;
8   bool vis[maxn];
9
10  void add(int u, int v)
```

```
{
    edge[u].push_back(v);
    edge[v].push_back(u);
}

void Tarjan(int u)
{
    dfn[u] = low[u] = ++tot;
    stack[++index] = u;
    vis[u] = true;
    int v;
    for(int i = 0;i < edge[u].size(); i++)
    {
        v=edge[u][i];
        if(!dfn[v])
        {
            Tarjan(v);
            low[u] = min(low[v], low[u]);
        }
        else if(vis[v]) low[u] = min(low[v], dfn[u]);
    }
    if(dfn[u] == low[u])
    {
        scc++;
        do
        {
            v = stack[index--];
            vis[v] = false;
            belong[v] = scc;
        }while(v != u);
    }
}
```

2.3.4 点双联通分量

```
const int maxn = 10000 + 10;

struct Edge{ int u, v; };
vector<int> G[maxn], bcc[maxn];

int dfn[maxn], low[maxn], bccno[maxn], idx, bcc_cnt, bridge;
bool iscut[maxn];

stack<Edge> st;

void dfs(int u, int pre)
{
    dfn[u] = low[u] = ++idx;
    int child = 0;
    for(auto v : G[u])
    {
        if(v == pre) continue;
        if(!dfn[v])
        {
            child ++;
            st.push({u, v});
            dfs(v, u);
            low[u] = min(low[u], low[v]);
```

```
24              if(low[v] >= dfn[u])
25              {
26                  iscut[u] = true;
27                  bcc[++bcc_cnt].clear();
28                  Edge x;
29                  do
30                  {
31                      x = st.top(); st.pop();
32                      if(bccno[x.u] != bcc_cnt) { bcc[bcc_cnt].push_back(x.u); bccno[x.u]
                            = bcc_cnt; }
33                      if(bccno[x.v] != bcc_cnt) { bcc[bcc_cnt].push_back(x.v); bccno[x.v]
                            = bcc_cnt; }
34                  } while(x.u != u || x.v != v);
35              }
36              if(low[v] > dfn[u]) ++ bridge;
37          }
38          else if(dfn[v] < dfn[u])
39          {
40              st.push({u, v});
41              low[u] = min(low[u], dfn[v]);
42          }
43      }
44      if(pre < 0 && child == 1) iscut[u] = 0;
45  }
46
47  void find_bcc(int n)
48  {
49      memset(dfn, 0, sizeof dfn);
50      memset(iscut, 0, sizeof iscut);
51      memset(bccno, 0, sizeof bccno);
52      for(int i = 1; i <= bcc_cnt; i ++) bcc[i].clear();
53      idx = bcc_cnt = bridge = 0;
54      for(int i = 0; i < n; i ++) if(!dfn[i]) dfs(i, -1);
55  }
```

2.3.5 边双联通分量

```
1   const int maxn = 10000 + 10;
2
3   int low[maxn], dfn[maxn], head[maxn], cnt, idx;
4   int cutEdge[maxn << 2];
5   struct  Edge { int nex, v; }edge[maxn << 2];
6
7   void add(int u, int v) { edge[cnt].nex = head[u], edge[cnt].v = v, head[u] = cnt ++; }
8
9   void dfs(int u, int pre)
10  {
11      low[u] = dfn[u] = ++idx;
12      for(int i = head[u]; ~i; i = edge[i].nex)
13      {
14          int v = edge[i].v;
15          if(v == pre) continue;
16          if(!dfn[v])
17          {
18              dfs(v, u);
19              low[u] = min(low[u], low[v]);
20              if(low[v] > dfn[u]) cutEdge[i] = cutEdge[i ^ 1] = 1;
21          }
```

```
22          else if(dfn[v] < dfn[u]) low[u] = min(low[u], dfn[v]);
23      }
24  }
25
26  int n, m;
27  vector<int> edge[maxn];
28  int dfn[maxn], low[maxn], idx;
29  int st[maxn], stsz;
30  // 标号特性: 子节点标号大于父节点
31  int inWhichGroup[maxn], groupNow, groupRt[maxn];
32  int fa[maxn];
33
34  void dfs(int u, int fa) {
35      ::fa[u] = fa;
36      dfn[u] = low[u] = ++idx, st[++stsz] = u;
37      int firstToVisFa = 1;
38      for (auto v : edge[u]) {
39          if (v != fa || !firstToVisFa) {
40              if (!dfn[v]) {
41                  dfs(v, u);
42                  low[u] = min(low[u], low[v]);
43              } else low[u] = min(low[u], dfn[v]);
44          } else firstToVisFa = 0;
45      }
46      if (dfn[u] == low[u]) {
47          ++groupNow;
48          groupRt[groupNow] = u;
49          do {
50              inWhichGroup[st[stsz]] = groupNow;
51          } while (st[stsz --] != u);
52      }
53  }
```

2.4 图匹配

1. 二分图中的最大匹配数 = 最小点覆盖数
2. 最小路径覆盖 = 最小路径覆盖= | G | - 最大匹配数
3. 二分图最大独立集 = 顶点数-最小点覆盖
4. 二分图的最大团 = 补图的最大独立集

2.4.1 Hungary Algorithm

```
1   const int maxn = 150;
2
3   int n;
4   int edge[maxn][maxn];
5   int linker[maxn];
6   bool vis[maxn];
7
8   bool path(int u)
9   {
10      for (int v = 1; v <= n; v++)
11      {
12          if (edge[u][v] && !vis[v])
13          {
14              vis[v] = true;
15              if (linker[v] == -1 || path(linker[v]))
```

```
16              {
17                  linker[v] = u;
18                  return true;
19              }
20          }
21      }
22      return false;
23  }
24
25  int hungary()
26  {
27      int res = 0;
28      memset(linker, 0xff, sizeof(linker));
29      for (int i = 1; i <= n; i++)
30      {
31          memset(vis, false, sizeof(vis));
32          res += path(i);
33      }
34      return res;
35  }
```

2.4.2 Hopcroft-karp Algorithm

```
1   //复杂度O(n^0.5*m)
2   struct Hopcroft {
3   #define maxn 100005
4   #define maxm 100005
5   #define INF 0x3f3f3f3f
6       struct Edge { int v, next; } edge[maxm];
7       int nx, cnt, dis;
8       int first[maxn];
9       int xlink[maxn], ylink[maxn];
10      int dx[maxn], dy[maxn];
11      int vis[maxn];
12
13      void init(int n) {
14          cnt = 0;
15          for (int i = 0; i <= n; ++i) first[i] = ylink[i] = xlink[i] = -1;
16          nx = n + 1;
17      }
18
19      void add_edge(int u, int v) {
20          edge[cnt].v = v, edge[cnt].next = first[u], first[u] = cnt++;
21      }
22
23      int bfs() {
24          queue<int> q;
25          dis = INF;
26          for (int i = 0; i < nx; ++i) dx[i] = dy[i] = -1;
27          for (int i = 0; i < nx; i++) {
28              if (xlink[i] == -1) {
29                  q.push(i);
30                  dx[i] = 0;
31              }
32          }
33          while (!q.empty()) {
34              int u = q.front();
35              q.pop();
```

```
            if (dx[u] > dis) break;
            for (int e = first[u]; e != -1; e = edge[e].next) {
                int v = edge[e].v;
                if (dy[v] == -1) {
                    dy[v] = dx[u] + 1;
                    if (ylink[v] == -1) dis = dy[v];
                    else {
                        dx[ylink[v]] = dy[v] + 1;
                        q.push(ylink[v]);
                    }
                }
            }
        }
        return dis != INF;
    }

    int find(int u) {
        for (int e = first[u]; e != -1; e = edge[e].next) {
            int v = edge[e].v;
            if (!vis[v] && dy[v] == dx[u] + 1) {
                vis[v] = 1;
                if (ylink[v] != -1 && dy[v] == dis) continue;
                if (ylink[v] == -1 || find(ylink[v])) {
                    xlink[u] = v, ylink[v] = u;
                    return 1;
                }
            }
        }
        return 0;
    }

    int maxmatch() {
        int ans = 0;
        for (int i = 0; i < nx; ++i) vis[i] = 0;
        while (bfs()) {
            for (int i = 0; i < nx; ++i) vis[i] = 0;
            for (int i = 0; i < nx; i++)
                if (xlink[i] == -1)
                    ans += find(i);
        }
        return ans;
    }

#undef maxn
#undef maxm
} solve;
```

2.4.3 二分图多重匹配

```
const int maxn = 1e2 + 5;//左边最大点数
const int maxm = 1e2 + 5;//右边最大点数
int graph[maxn][maxm], vis[maxm];//图G和增广路访问标记
int match[maxm][maxn];//左边元素与右边元素第n次匹配
int nx, ny, m;//左边点数, 右边点数,边数
int vol[maxm];//右边点多重匹配可容纳值
int cnt[maxm];//右边点已匹配值

bool find_path(int u)//找增广路
```

```
{
    for (int i = 0; i < ny; i++)//注意，这里节点是从0开始编号，题目有时是从1开始编号
    {
        if (graph[u][i] && !vis[i])//不在增广路
        {
            vis[i] = 1;//放进增广路
            if (cnt[i] < vol[i])//如果当前已匹配数量小于可容纳量，则直接匹配
            {
                match[i][cnt[i]++] = u;
                return true;
            }
            for (int j = 0; j < cnt[i]; j++)
            {
                if (find_path(match[i][j]))//如果先前已匹配右边的点能另外找到增广路，则此点仍可匹配
                {
                    match[i][j] = u;
                    return true;
                }
            }
        }
    }
    return false;
}

int max_match()//计算多重匹配的最大匹配数
{
    int res = 0;
    memset(match, -1, sizeof(match));
    memset(cnt, 0, sizeof(cnt));
    for (int i = 0; i < nx; i++)
    {
        memset(vis, 0, sizeof(vis));
        if (find_path(i)) res++;
    }
    return res;
}

bool all_match()//判断左边的点是否都与右边的点匹配了
{
    memset(cnt, 0, sizeof(cnt));
    for (int i = 0; i < nx; i++)
    {
        memset(vis, 0, sizeof(vis));
        if (!find_path(i)) return false;
    }
    return true;
}
```

2.4.4 二分图最大权匹配 (KM 算法)

```
int n;
int cost[maxn][maxn];
int lx[maxn], ly[maxn], match[maxn], slack[maxn];
int Prev[maxn];
bool vy[maxn];
const int INF = 0X3f3f3f3f;

```

```
void augment(int root) {
    fill(vy + 1, vy + n + 1, false);
    fill(slack + 1, slack + n + 1, INF);
    int py;
    match[py = 0] = root;
    do {
        vy[py] = true;
        int x = match[py], yy;
        int delta = INF;
        for (int y = 1; y <= n; y++) {
            if (!vy[y]) {
                if (lx[x] + ly[y] - cost[x][y] < slack[y])
                    slack[y] = lx[x] + ly[y] - cost[x][y], Prev[y] = py;
                if (slack[y] < delta) delta = slack[y], yy = y;
            }
        }
        for (int y = 0; y <= n; y++) {
            if (vy[y])
                lx[match[y]] -= delta, ly[y] += delta;
            else
                slack[y] -= delta;
        }
        py = yy;
    } while (match[py] != -1);
    do {
        int pre = Prev[py];
        match[py] = match[pre], py = pre;
    } while (py);
}

int KM() {
    for (int i = 1; i <= n; i++) {
        lx[i] = ly[i] = 0;
        match[i] = -1;
        for (int j = 1; j <= n; j++) lx[i] = max(lx[i], cost[i][j]);
    }
    int answer = 0;
    for (int root = 1; root <= n; root++) augment(root);
    for (int i = 1; i <= n; i++) answer += lx[i], answer += ly[i];
    return answer;
}
```

2.4.5 一般图匹配带花树

```
//一般图匹配，带花树算法
const int maxn = 1000 + 10;

struct Edmond
{
    vector<int> edge[maxn];
    queue<int> que;

    int n, pre[maxn], type[maxn], link[maxn], nex[maxn], vis[maxn];

    void init(int n)
    {
        this->n = n;
        for (int i = 0; i <= n; i++) edge[i].clear();
```

```
15          memset(link, 0, sizeof(link));
16      }
17
18      void add(int u, int v)
19      {
20          edge[u].push_back(v);
21          edge[v].push_back(u);
22      }
23
24      int Find(int x)
25      {
26          return x == pre[x] ? x : pre[x] = Find(pre[x]);
27      }
28
29      void combine(int x, int lca)    //如果找到奇环, 对当前点x和找到的
30      {
31          while (x != lca)
32          {
33              int u = link[x], v = nex[u];
34              if (Find(v) != lca) nex[v] = u;
35              if (type[u] == 1) type[u] = 2, que.push(u);
36              pre[Find(x)] = Find(u);
37              pre[Find(u)] = Find(v);
38              x = v;
39          }
40      }
41
42      void contrack(int x, int y)
43      {
44          int lca = x;
45          memset(vis, 0, sizeof(vis));
46          for (int i = x; i; i = nex[link[i]])
47          {
48              i = Find(i);
49              vis[i] = 1;
50          }
51          for (int i = y; i; i = nex[link[i]])
52          {
53              i = Find(i);
54              if (vis[i])
55              {
56                  lca = i;
57                  break;
58              }
59          }
60          if (lca != Find(x)) nex[x] = y;
61          if (lca != Find(y)) nex[y] = x;
62          combine(x, lca);
63          combine(y, lca);
64      }
65
66      void bfs(int s)
67      {
68          memset(type, 0, sizeof(type));
69          memset(nex, 0, sizeof(nex));
70          for (int i = 1; i <= n; i++) pre[i] = i;
71          while (!que.empty()) que.pop();
72          que.push(s);
73          type[s] = 2;
```

```
74          while (!que.empty())
75          {
76              int x = que.front();
77              que.pop();
78              for (int i = 0; i < edge[x].size(); i++)
79              {
80                  int y = edge[x][i];
81                  if (Find(x) == Find(y) || link[x] == y || type[y] == 1) continue;
82                  if (type[y] == 2) contrack(x, y);
83                  else if (link[y])
84                  {
85                      nex[y] = x;
86                      type[y] = 1;
87                      type[link[y]] = 2;
88                      que.push(link[y]);
89                  } else
90                  {
91                      nex[y] = x;
92                      int pos = y, u = nex[pos], v = link[u];
93                      while (pos)
94                      {
95                          link[pos] = u;
96                          link[u] = pos;
97                          pos = v;
98                          u = nex[pos];
99                          v = link[u];
100                     }
101                     return;
102                 }
103             }
104         }
105     }
106
107     int maxmatch()
108     {
109         for (int i = 1; i <= n; i++) if (!link[i]) bfs(i);
110         int ans = 0;
111         for (int i = 1; i <= n; i++) if (link[i]) ans++;
112         return ans / 2;
113     }
114 }ans;
```

2.5 网络流

2.5.1 Dinic

```
1  const int MAX_V = 1000 + 10;
2  const int INF = 0x3f3f3f3f;
3
4  //用于表示边的结构体（终点，流量，反向边）
5  struct edge{int to, cap, rev;};
6
7  vector<edge> G[MAX_V];  //图的邻接表表示
8  int level[MAX_V];   //顶点到源点的距离标号
9  int iter[MAX_V];    //当前弧
10
11 void add(int from, int to, int cap)
12 {
13     G[from].push_back((edge){to, cap, (int)G[to].size()});
```

```
14      G[to].push_back((edge){from, 0, (int)G[from].size() - 1});
15  }
16
17  //计算从源点出发的距离标号
18  void bfs(int s)
19  {
20      memset(level, -1, sizeof(level));
21      queue<int> que;
22      level[s] = 0;
23      que.push(s);
24      while(!que.empty())
25      {
26          int v = que.front(); que.pop();
27          for(int i = 0; i < G[v].size(); i++)
28          {
29              edge &e = G[v][i];
30              if(e.cap > 0 && level[e.to] < 0)
31              {
32                  level[e.to] = level[v] + 1;
33                  que.push(e.to);
34              }
35          }
36      }
37  }
38
39  //通过DFS寻找增广路
40  int dfs(int v, int t, int f)
41  {
42      if(v == t) return f;
43      for(int &i = iter[v]; i<G[v].size(); i++)
44      {
45          edge &e = G[v][i];
46          if(e.cap > 0 && level[v] < level[e.to])
47          {
48              int d = dfs(e.to, t, min(f, e.cap));
49              if(d > 0)
50              {
51                  e.cap -= d;
52                  G[e.to][e.rev].cap += d;
53                  return d;
54              }
55          }
56      }
57      return 0;
58  }
59
60  //求解从s到t的最大流
61  int max_flow(int s, int t)
62  {
63      int flow = 0;
64      for(;;)
65      {
66          bfs(s);
67          if(level[t] < 0) return flow;
68          memset(iter, 0, sizeof(iter));
69          int f;
70          while((f = dfs(s,t,INF)) > 0) flow += f;
71      }
72  }
```

2.5.2 ISAP

```
struct Edge {
  int from, to, cap, flow;
  Edge(int u, int v, int c, int f) : from(u), to(v), cap(c), flow(f) {}
};

bool operator<(const Edge& a, const Edge& b) {
  return a.from < b.from || (a.from == b.from && a.to < b.to);
}

struct ISAP {
  int n, m, s, t;
  vector<Edge> edges;
  vector<int> G[maxn];
  bool vis[maxn];
  int d[maxn];
  int cur[maxn];
  int p[maxn];
  int num[maxn];

  void AddEdge(int from, int to, int cap) {
    edges.push_back(Edge(from, to, cap, 0));
    edges.push_back(Edge(to, from, 0, 0));
    m = edges.size();
    G[from].push_back(m - 2);
    G[to].push_back(m - 1);
  }

  bool BFS() {
    memset(vis, 0, sizeof(vis));
    queue<int> Q;
    Q.push(t);
    vis[t] = 1;
    d[t] = 0;
    while (!Q.empty()) {
      int x = Q.front();
      Q.pop();
      for (int i = 0; i < G[x].size(); i++) {
        Edge& e = edges[G[x][i] ^ 1];
        if (!vis[e.from] && e.cap > e.flow) {
          vis[e.from] = 1;
          d[e.from] = d[x] + 1;
          Q.push(e.from);
        }
      }
    }
    return vis[s];
  }

  void init(int n) {
    this->n = n;
    for (int i = 0; i < n; i++) G[i].clear();
    edges.clear();
  }

  int Augment() {
    int x = t, a = INF;
    while (x != s) {
```

```
58        Edge& e = edges[p[x]];
59        a = min(a, e.cap - e.flow);
60        x = edges[p[x]].from;
61      }
62      x = t;
63      while (x != s) {
64        edges[p[x]].flow += a;
65        edges[p[x] ^ 1].flow -= a;
66        x = edges[p[x]].from;
67      }
68      return a;
69    }
70
71    int Maxflow(int s, int t) {
72      this->s = s;
73      this->t = t;
74      int flow = 0;
75      BFS();
76      memset(num, 0, sizeof(num));
77      for (int i = 0; i < n; i++) num[d[i]]++;
78      int x = s;
79      memset(cur, 0, sizeof(cur));
80      while (d[s] < n) {
81        if (x == t) {
82          flow += Augment();
83          x = s;
84        }
85        int ok = 0;
86        for (int i = cur[x]; i < G[x].size(); i++) {
87          Edge& e = edges[G[x][i]];
88          if (e.cap > e.flow && d[x] == d[e.to] + 1) {
89            ok = 1;
90            p[e.to] = G[x][i];
91            cur[x] = i;
92            x = e.to;
93            break;
94          }
95        }
96        if (!ok) {
97          int m = n - 1;
98          for (int i = 0; i < G[x].size(); i++) {
99            Edge& e = edges[G[x][i]];
100           if (e.cap > e.flow) m = min(m, d[e.to]);
101         }
102         if (--num[d[x]] == 0) break;
103         num[d[x] = m + 1]++;
104         cur[x] = 0;
105         if (x != s) x = edges[p[x]].from;
106       }
107     }
108     return flow;
109   }
110 };
```

2.5.3 MCMF

```
1  const int maxn = 10000 + 10;
2  const int inf = 0x3f3f3f3f;
```

```
3
4  struct Edge { int from, to, cap, flow, cost; };
5
6  struct MCMF
7  {
8      int n, m;
9      vector<Edge> edges;
10     vector<int> G[maxn];
11     bool inq[maxn];
12     int dis[maxn], path[maxn], a[maxn];
13
14     void init(int n)
15     {
16         this->n = n;
17         for(int i = 0;i <= n;i ++)
18             G[i].clear();
19         edges.clear();
20     }
21
22     void addEdge(int from, int to, int cap, int cost)
23     {
24         edges.push_back(Edge{from, to, cap, 0, cost});
25         edges.push_back(Edge{to, from, 0, 0, -cost});
26         m = edges.size();
27         G[from].push_back(m - 2);
28         G[to].push_back(m - 1);
29     }
30
31     bool Bellman_Ford(int s, int t, int& flow, int& cost)
32     {
33         for(int i = 0; i<= n; i++) dis[i] = inf;
34         memset(inq, 0, sizeof inq);
35         dis[s]=0, inq[s]=true, path[s]=0, a[s]=inf;
36         queue<int> Q;
37         Q.push(s);
38         while(!Q.empty())
39         {
40             int u = Q.front(); Q.pop();
41             inq[u] = false;
42             for(int i = 0; i < G[u].size(); i++)
43             {
44                 Edge& e = edges[G[u][i]];
45                 if(e.cap > e.flow && dis[e.to] > dis[u] + e.cost)
46                 {
47                     dis[e.to] = dis[u] + e.cost;
48                     path[e.to] = G[u][i];
49                     a[e.to] = min(a[u], e.cap - e.flow);
50                     if(!inq[e.to])
51                     {
52                         Q.push(e.to);
53                         inq[e.to] = true;
54                     }
55                 }
56             }
57         }
58         if(dis[t] == inf) return false;     //求最小费用最大流
59         //if(1ll * dis[t] * a[t] > 0) return false; 求可行流最小费用, 因此当费用增量大
               于0时不继续增加流量
60         flow += a[t];
```

```
61          cost += dis[t] * a[t];
62          for(int u = t; u != s; u = edges[path[u]].from)
63          {
64              edges[path[u]].flow += a[t];
65              edges[path[u] ^ 1].flow -= a[t];
66          }
67          return true;
68      }
69
70      int mincostMaxFlow(int s, int t)
71      {
72          int flow = 0, cost = 0;
73          while(Bellman_Ford(s, t, flow, cost));
74          return cost;
75      }
76  };
```

2.5.4 Trick

建模技巧

二分图带权最大独立集。给出一个二分图，每个结点上有一个正权值。要求选出一些点，使得这些点之间没有边相连，且权值和最大。

解： 在二分图的基础上添加源点 S 和汇点 T，然后从 S 向所有 X 集合中的点连一条边，所有 Y 集合中的点向 T 连一条边，容量均为该点的权值。X 结点与 Y 结点之间的边的容量均为无穷大。这样，对于图中的任意一个割，将割中的边对应的结点删掉就是一个符合要求的解，权和为所有权减去割的容量。因此，只需要求出最小割，就能求出最大权和。

公平分配问题。把 m 个任务分配给 n 个处理器。其中每个任务有两个候选处理器，可以任选一个分配。要求所有处理器中，任务数最多的那个处理器所分配的任务数尽量少。不同任务的候选处理器集 $\{p_1, p_2\}$ 保证不同。

解： 本题有一个比较明显的二分图模型，即 X 结点是任务，Y 结点是处理器。二分答案 x，然后构图，首先从源点 S 出发向所有的任务结点引一条边，容量等于 1，然后从每个任务结点出发引两条边，分别到达它所能分配到的两个处理器结点，容量为 1，最后从每个处理器结点出发引一条边到汇点 T，容量为 x，表示选择该处理器的任务不能超过 x。这样网络中的每个单位流量都是从 S 流到一个任务结点，再到处理器结点，最后到汇点 T。只有当网络中的总流量等于 m 时才意味着所有任务都选择了一个处理器。这样，我们通过 $O(\log m)$ 次最大流便算出了答案。

区间 k 覆盖问题。数轴上有一些带权值的左闭右开区间。选出权和尽量大的一些区间，使得任意一个数最多被 k 个区间覆盖。

解： 本题可以用最小费用流解决，构图方法是把每个数作为一个结点，然后对于权值为 w 的区间 $[u, v)$ 加边 $u \to v$，容量为 1，费用为 $-w$。再对所有相邻的点加边 $i \to i+1$，容量为 k，费用为 0。最后，求最左点到最右点的最小费用最大流即可，其中每个流量对应一组互不相交的区间。如果数值范围太大，可以先进行离散化。

最大闭合子图。给定带权图 G（权值可正可负），求一个权和最大的点集，使得起点在该点集中的任意弧，终点也在该点集中。

解： 新增附加源 s 和附加汇 t，从 s 向所有正权点引一条边，容量为权值；从所有负权点向汇点引一条边，容量为权值的相反数。求出最小割以后，$S - \{s\}$ 就是最大闭合子图。

最大密度子图。给出一个无向图，找一个点集，使得这些点之间的边数除以点数的值（称为子图的密度）最大。

解： 如果两个端点都选了，就必然要选边，这就是一种推导。如果把每个点和每条边都看成新图中的结点，可以把问题转化为最大闭合子图。

无源汇有上下界可行流： 附加源 S 和汇 T；对于边 (u, v, min, max)，记 $d[u] -= min, d[v] += max$，并添加弧 $(u, v, max - min)$；对于流量不平衡的点 u，设多余流量为 W，如果 $W > 0$，添加弧 $S -> u : W$，否则若 $W < 0$，添加弧 $u -> T : -W$，求改造后的网络 $S - T$ 最大流即可，当且仅当所有附加弧满载时原图有可行流。

有源汇有上下界可行流：建 $t->s$，容量为 inf，然后和无源汇相同。

有源汇有上下界最大/最小流：与上面相同，跑完可行流 $S->T$ 后去掉边 $t->s$，最大流为加 $s->t$，最小流为 $G[s][t].cap - max_flow(t,s)$。

2.5.5 Stoer Wagner

```
#define INF 100000000
bool vis[maxn], com[maxn];
int mp[maxn][maxn], w[maxn], s, t;

int maxadj(int n, int v) {
    int CUT = 0;
    memset(vis, 0, sizeof vis);
    memset(w, 0, sizeof w);
    for (int i = 0; i < n; ++i) {
        int num = 0, mx = -INF;
        for (int j = 0; j < v; ++j) {
            if (!com[j] && !vis[j] && w[j] > mx) {
                mx = w[j];
                num = j;
            }
        }
        vis[num] = 1;
        s = t;
        t = num;
        CUT = w[t];
        for (int j = 0; j < v; ++j) {
            if (!com[j] && !vis[j]) w[j] += mp[num][j];
        }
    }
    return CUT;
}

int stoer(int v) {
    int mincut = INF;
    int n = v;
    memset(com, 0, sizeof com);
    for (int i = 0; i < v - 1; ++i) {
        int cut;
        s = 0, t = 0;
        cut = maxadj(n, v);
        n --;
        if (cut < mincut) mincut = cut;
        com[t] = 1;
        for (int j = 0; j < v; ++j) {
            if (!com[j]) {
                mp[j][s] += mp[j][t];
                mp[s][j] += mp[t][j];
            }
        }
    }
    return mincut;
}
```

2.5.6 ZKW 费用流

```
struct MCMF
```

```
2  {
3      int last[maxn], dis[maxn], cnt, ans;
4      int s, t;
5      bool vis[maxn];
6      struct edge { int from, to, cap, w, op, nex; } e[500000 + 10];
7
8      void init(int S, int T)
9      {
10         s = S, t = T;
11         cnt = 0, ans = 0;
12         memset(vis, 0, sizeof vis);
13         memset(dis, 0, sizeof dis);
14         memset(last, 0, sizeof last);
15     }
16
17     void add(int u, int v, int cap, int cost)
18     {
19         e[++ cnt] = { u, v, cap, cost, cnt + 1, last[u] };
20         last[u] = cnt;
21         e[++ cnt] = { v, u, 0, -cost, cnt - 1, last[v] };
22         last[v] = cnt;
23     }
24
25     int dfs(int x, int maxf)
26     {
27         if(x == t || maxf == 0) return maxf;
28         int ret = 0;
29         vis[x] = 1;
30         for(int i = last[x]; i; i = e[i].nex)
31             if(e[i].cap && dis[e[i].to] + e[i].w == dis[x] && !vis[e[i].to])
32             {
33                 int f = dfs(e[i].to, min(e[i].cap, maxf - ret));
34                 ans += f * e[i].w;
35                 e[i].cap -= f;
36                 e[e[i].op].cap += f;
37                 ret += f;
38                 if(ret == maxf) break;
39             }
40         return ret;
41     }
42
43     bool change()
44     {
45         int mn = inf;
46         for(int i = 0; i <= t; i ++)
47             if(vis[i])
48                 for(int j = last[i]; j; j = e[j].nex)
49                     if(!vis[e[j].to] && e[j].cap) mn = min(mn, -dis[i] + e[j].w + dis[e
                           [j].to]);
50         if(mn == inf) return false;
51         for(int i = 0; i <= t; i ++) if(vis[i]) dis[i] += mn;
52         return true;
53     }
54
55     void zkw()
56     {
57         do
58         {
59             for(int i = 0; i <= t; i ++) vis[i] = 0;
```

```
60              while(dfs(s, inf)) for(int i = 0; i <= t; i ++) vis[i] = 0;
61          }
62          while(change());
63      }
64  }ans;
```

2.6 Others

2.6.1 拓扑排序

```
1   const int maxn = 1e5 + 10;
2
3   vector<int> edge[maxn];
4   int indegree[maxn];
5
6   void add(int u, int v)
7   {
8       edge[u].push_back(v);
9       indegree[v]++;
10  }
11
12  void Toposort(int n)
13  {
14      queue<int> que;
15      for (int i = 1; i <= n; i++)
16          if (!indegree[i]) que.push(i);    //将图中没有前驱，即入度为0的点加入队列
17      while (!que.empty())
18      {
19          int u = que.front();
20          que.pop();
21          indegree[u] = -1;    //从图中删去此顶点
22          for (int i = 0; i < edge[u].size(); i++)
23          {
24              int v = edge[u][i];
25              indegree[v]--;    //删去图中以u为尾的弧
26              if (!indegree[v]) que.push(v);    //将新增的当前入度为0的点压入队列中
27          }
28      }
29  }
```

2.6.2 2-SAT

```
1   /*2-SAT连边含义：选A必选B
2       点$x_i$表示选，$x_i'$表示不选
3       1.必选$x_i$，等价于$x_i=1$：$x_i'→x_i$
4       2.必不选$x_i$，等价于$x_i=0$，$x_i→x_i'$
5       3.$x_i$与$x_j$中至少选择一个，等价于$x_iORx_j=1$，连边$x_i'→x_j$,$x_j'→x_i$
6       4.$x_i$与$x_j$不都选，等价于$x_iANDx_j=0$，连边$x_i→x_j',x_j→x_i'$
7       5.$x_i$与$x_j$情况相同，等价于$x_iXORx_j=0$，连边$x_i→x_j$,$x_i'→x_j'$,$x_j→x_i$,
            $x_j'→x_i'$
8       6.$x_i$与$x_j$情况相反，等价于$x_iXORx_j=1$，连边$x_i→x_j'$,$x_i'→x_j$,$x_j→x_i'$,
            $x_j'→x_i$
9   */
10
11  const int maxn = 2e6 + 10;
12
13  namespace twosat {
14      int n;
```

```
15      int low[maxn], dfn[maxn], color[maxn], cnt, scc_cnt;
16      bool instack[maxn];
17
18      vector<int> g[maxn];
19      stack<int> st;
20
21      void init(int _n) {
22          n = _n;
23          cnt = scc_cnt = 0;
24          for (int i = 0; i <= n * 2; ++i) {
25              dfn[i] = 0;
26              g[i].clear();
27          }
28      }
29
30      void Tarjan(int u) {
31          low[u] = dfn[u] = ++cnt;
32          st.push(u);
33          instack[u] = true;
34          for (const auto &v : g[u]) {
35              if (!dfn[v]) Tarjan(v), low[u] = min(low[u], low[v]);
36              else if (instack[v]) low[u] = min(low[u], dfn[v]);
37          }
38          if (low[u] == dfn[u]) {
39              ++scc_cnt;
40              do {
41                  color[u] = scc_cnt;
42                  u = st.top();
43                  st.pop();
44                  instack[u] = false;
45              } while (low[u] != dfn[u]);
46          }
47      }
48
49      inline void add(int a, int b) { g[a].push_back(b); }
50
51      inline void AND(int a, int b, int c) {
52          if (c == 1) add(a, a + n), add(b, b + n);
53          else add(a, b + n), add(b, a + n);
54      }
55
56      inline void OR(int a, int b, int c) {
57          if (c == 0) add(a + n, a), add(b + n, b);
58          else add(a + n, b), add(b + n, a);
59      }
60
61      inline void XOR(int a, int b, int c) {
62          if (c == 0) add(a, b), add(a + n, b + n), add(b, a), add(b + n, a + n);
63          else add(a, b + n), add(a + n, b), add(b, a + n), add(b + n, a);
64      }
65
66      bool TWO_SAT() {
67          for (int i = 1; i <= (n << 1); i++) if (!dfn[i]) Tarjan(i);
68          for (int i = 1; i <= n; i++)
69              if (color[i] == color[i + n]) return false;
70          for (int i = 1; i <= n; i++)
71              printf("%d ", color[i] > color[i + n]);
72          return true;
73      }
```

```
}
```

2.6.3 差分约束系统

```
//以$x_i-x_j y$为约束条件，建图求最短路后得到的是最大解。所有的解都不大于且尽可能逼近$dis[x0]$
//最短路对应最大解，最长路对应最小解

const int maxn = 1000 + 10;
const int inf = 0x3f3f3f3f;

struct Edge
{
    int nex, to, w;
} edge[10 * maxn];

int head[maxn], cnt, dis[maxn], n;
bool vis[maxn];

void init()
{
    cnt = 0;
    memset(head, 0xff, sizeof head);
}

void add(int u, int v, int w)
{
    edge[cnt].nex = head[u];
    edge[cnt].to = v;
    edge[cnt].w = w;
    head[u] = ++cnt;
}

void spfa(int u)
{
    int u, v, w;
    for (int i = 1; i <= n; i++) dis[i] = inf, vis[i] = false;
    dis[u] = 0;
    queue<int> que;
    que.push(u);
    vis[u] = true;
    while (!que.empty())
    {
        u = que.front();
        que.pop();
        vis[u] = false;
        for (int i = head[u]; ~i; i = edge[i].nex)
        {
            v = edge[i].v, w = edge[i].w;
            if (dis[u] + w < dis[v])
            {
                dis[v] = dis[u] + w;
                if (!vis[v])
                {
                    que.push(v);
                    vis[v] = true;
                }
            }
```

```
54          }
55      }
56  }
```

2.6.4 支配树

```
1   const int N = 2e5 + 10;
2
3   int n, m;
4
5   struct G
6   {
7       vector<int> edge[N];
8       inline void add(int u, int v) { edge[u].push_back(v); }
9   }a, b, c, d;
10
11  int dfn[N], id[N], fa[N], cnt;
12
13  void dfs(int u)
14  {
15      dfn[u] = ++ cnt; id[cnt] = u;
16      int len = a.edge[u].size();
17      for(auto v : a.edge[u]) if(!dfn[v]) { fa[v] = u; dfs(v); }
18  }
19
20  int semi[N], idom[N], belong[N], val[N];
21
22  int find(int x)
23  {
24      if(x == belong[x]) return x;
25      int tmp = find(belong[x]);
26      if(dfn[semi[val[belong[x]]]] < dfn[semi[val[x]]]) val[x] = val[belong[x]];
27      return belong[x] = tmp;
28  }
29
30  void tarjan()
31  {
32      for(int i = cnt; i > 1; i --)
33      {
34          int u = id[i];
35          for(auto v : b.edge[u])
36          {
37              if(!dfn[v]) continue;
38              find(v);
39              if(dfn[semi[val[v]]] < dfn[semi[u]]) semi[u] = semi[val[v]];
40          }
41          c.add(semi[u], u);
42          belong[u] = fa[u];
43          u = fa[u];
44          for(auto v : c.edge[u])
45          {
46              find(v);
47              if(semi[val[v]] == u) idom[v] = u;
48              else idom[v] = val[v];
49          }
50      }
51      for(int i = 2; i <= cnt; i ++)
52      {
```

```
53          int u = id[i];
54          if(idom[u] != semi[u]) idom[u] = idom[idom[u]];
55      }
56  }
57
58  int ans[N];
59
60  void dfs_ans(int u)
61  {
62      ans[u] = 1;
63      for(auto v : d.edge[u]) dfs_ans(v), ans[u] += ans[v];
64  }
65
66  void solve()
67  {
68      int u, v;
69      scanf("%d%d", &n, &m);
70      while(m --)
71      {
72          scanf("%d%d", &u, &v);
73          a.add(u, v);
74          b.add(v, u);
75      }
76      for(int i = 1; i <= n; i ++) semi[i] = belong[i] = val[i] = i;
77      dfs(1);
78      tarjan();
79      for(int i = 2; i <= n; i ++) d.add(idom[i], i);
80      dfs_ans(1);
81      for(int i = 1; i <= n; i ++) printf("%d ", ans[i]);
82  }
```

2.6.5 Stable Matching Problem

```
1   const int maxn = 1000 + 10;
2
3   int pre[maxn][maxn], order[maxn][maxn], nex[maxn];
4   int hus[maxn], wife[maxn];
5   queue<int> que;
6
7   void engage(int man, int woman) {
8       int m = hus[woman];
9       if (m) wife[m] = 0, q.push(m);
10      wife[man] = woman;
11      hus[woman] = man;
12  }
13
14  int solve() {
15      for (int i = 1; i <= n; i++) {
16          for (int j = 1; j <= n; j++)
17              scanf("%d", &pre[i][j]);
18          nex[i] = 1;
19          wife[i] = 0;
20          que.push(i);
21      }
22      for (int i = 1; i <= n; i++) {
23          for (int j = 1; j <= n; j++) {
24              int x;
25              scanf("%d", &x);
```

```
26              order[i][x] = j;
27          }
28          hus[i] = 0;
29      }
30
31      while (!que.empty()) {
32          int man = que.front();
33          que.pop();
34          int woman = pre[man][nex[man]++];
35          if (!hus[woman]) engage(man, woman);
36          else if (order[woman][man] < order[woman][hus[woman]]) engage(man, woman);
37          else que.push(man);
38      }
39  }
```

2.6.6 一般图最大团

```
1   #define u64 unsigned long long
2   #define i64 long long
3
4   const u64 BITCOUNT = sizeof(u64) * 8;
5
6   u64 count_trailing_zeroes(u64 a) {
7       if (a == 0ull)
8           return BITCOUNT;
9       return __builtin_ctzll(a);
10  }
11  u64 disable_bit(u64 a, u64 bit) { return a & (~(1ull << bit)); }
12  u64 popcount(u64 a) { return __builtin_popcountll(a); }
13
14  map<u64, u64> max_clique_cache;
15
16  u64 max_clique(u64 mask, vector<u64> const &graph_matrix) { // 最大独立集传补图即可
17      if (max_clique_cache.find(mask) != max_clique_cache.end())
18          return max_clique_cache[mask];
19      u64 a = count_trailing_zeroes(mask);
20      if (a == BITCOUNT) return 0;
21      u64 res1 = max_clique(disable_bit(mask, a), graph_matrix);
22      u64 res2 = max_clique(mask & disable_bit(graph_matrix[a], a), graph_matrix) | (1ull
            << a);
23      u64 res = popcount(res1) > popcount(res2) ? res1 : res2;
24      max_clique_cache[mask] = res;
25      return res;
26  }
27
28  int main() {
29      vector<u64> M;
30      int n;
31      while (scanf("%d", &n) != EOF) {
32          if (n == 0) break;
33          M.clear();
34          M.resize(n);
35          max_clique_cache.clear();
36          for (int i = 0; i < n; ++i) {
37              for (int j = 0; j < n; ++j) {
38                  int x; scanf("%d", &x);
39                  if (x == 1) M[i] |= 1ll << j;
40              }
```

```
41          }
42          printf("%d\n", popcount(max_clique((1ll << n) - 1, M)));
43      }
44  }
```

3 DataStructrue

3.1 SegmentTreeDS

3.1.1 SGTB

```
int n, a[maxn];

struct node {
    int p, t, se;

    node() {}

    node(int p, int t, int se) : p(p), t(t), se(se) {}

    inline friend node combineMax(node a, node b) {
        node c;
        if (a.p < b.p) {
            c.p = b.p;
            c.t = b.t;
            c.se = max(a.p, b.se);
        } else if (a.p > b.p) {
            c.p = a.p;
            c.t = a.t;
            c.se = max(a.se, b.p);
        } else {
            c.p = a.p;
            c.t = a.t + b.t;
            c.se = max(a.se, b.se);
        }
        return c;
    }

    inline friend node combineMin(node a, node b) {
        node c;
        if (a.p > b.p) {
            c.p = b.p;
            c.t = b.t;
            c.se = min(a.p, b.se);
        } else if (a.p < b.p) {
            c.p = a.p;
            c.t = a.t;
            c.se = min(a.se, b.p);
        } else {
            c.p = a.p;
            c.t = a.t + b.t;
            c.se = min(a.se, b.se);
        }
        return c;
    }

    inline friend node operator+(node a, int p) {
        return node(a.p + p, a.t, a.se + p);
    }
};

struct SGTB {
#define inf 1e9
    node mx[maxn << 2], mi[maxn << 2];
```

```
54      int tag[maxn << 2];
55      ll s[maxn << 2];
56  # define ls (x<<1)
57  # define rs (x<<1|1)
58
59      inline void up(int x) {
60          mx[x] = combineMax(mx[ls], mx[rs]);
61          mi[x] = combineMin(mi[ls], mi[rs]);
62          s[x] = s[ls] + s[rs];
63      }
64
65      // a = max(a, t)
66      inline void pushmax(int x, int l, int r, int p) {
67          s[x] += 1ll * mi[x].t * (p - mi[x].p);
68          mi[x].p = p;
69          mx[x].p = max(mx[x].p, p);
70          if (mi[x].p == mx[x].p) {
71              mi[x].se = inf, mx[x].se = -inf;
72              mi[x].t = mx[x].t = r - l + 1;
73              s[x] = 1ll * mi[x].p * (r - l + 1);
74          } else mx[x].se = max(mx[x].se, p);
75      }
76
77      // a = min(a, t)
78      inline void pushmin(int x, int l, int r, int p) {
79          s[x] += 1ll * mx[x].t * (p - mx[x].p);
80          mx[x].p = p;
81          mi[x].p = min(mi[x].p, p);
82          if (mi[x].p == mx[x].p) {
83              mi[x].se = inf, mx[x].se = -inf;
84              mi[x].t = mx[x].t = r - l + 1;
85              s[x] = 1ll * mi[x].p * (r - l + 1);
86          } else mi[x].se = min(mi[x].se, p);
87      }
88
89      inline void pushtag(int x, int l, int r, int p) {
90          tag[x] += p;
91          s[x] += 1ll * (r - l + 1) * p;
92          mx[x] = mx[x] + p, mi[x] = mi[x] + p;
93      }
94
95      inline void down(int x, int l, int r) {
96          int mid = l + r >> 1;
97          if (tag[x]) {
98              pushtag(ls, l, mid, tag[x]);
99              pushtag(rs, mid + 1, r, tag[x]);
100             tag[x] = 0;
101         }
102         if (mx[ls].p > mx[x].p && mx[ls].se < mx[x].p) pushmin(ls, l, mid, mx[x].p);
103         if (mx[rs].p > mx[x].p && mx[rs].se < mx[x].p) pushmin(rs, mid + 1, r, mx[x].p)
                ;
104         if (mi[ls].p < mi[x].p && mi[ls].se > mi[x].p) pushmax(ls, l, mid, mi[x].p);
105         if (mi[rs].p < mi[x].p && mi[rs].se > mi[x].p) pushmax(rs, mid + 1, r, mi[x].p)
                ;
106     }
107
108     inline void build(int x, int l, int r) {
109         tag[x] = 0;
110         if (l == r) {
```

```
111             mx[x].p = mi[x].p = s[x] = a[l], mx[x].t = mi[x].t = 1, mx[x].se = -inf, mi
                    [x].se = inf;
112             return;
113         }
114         int mid = l + r >> 1;
115         build(ls, l, mid);
116         build(rs, mid + 1, r);
117         up(x);
118     }
119
120     // add p to [L, R]
121     inline void edt(int x, int l, int r, int L, int R, int p) {
122         if (L <= l && r <= R) {
123             pushtag(x, l, r, p);
124             return;
125         }
126         down(x, l, r);
127         int mid = l + r >> 1;
128         if (L <= mid) edt(ls, l, mid, L, R, p);
129         if (R > mid) edt(rs, mid + 1, r, L, R, p);
130         up(x);
131     }
132
133     // ai=min(ai, p) in [L, R]
134     inline void edtmin(int x, int l, int r, int L, int R, int p) {
135         if (mx[x].p <= p) return;
136         if (L <= l && r <= R && mx[x].se < p) {
137             pushmin(x, l, r, p);
138             return;
139         }
140         down(x, l, r);
141         int mid = l + r >> 1;
142         if (L <= mid) edtmin(ls, l, mid, L, R, p);
143         if (R > mid) edtmin(rs, mid + 1, r, L, R, p);
144         up(x);
145     }
146
147     // ai=max(ai, p) in [L, R]
148     inline void edtmax(int x, int l, int r, int L, int R, int p) {
149         if (mi[x].p >= p) return;
150         if (L <= l && r <= R && mi[x].se > p) {
151             pushmax(x, l, r, p);
152             return;
153         }
154         down(x, l, r);
155         int mid = l + r >> 1;
156         if (L <= mid) edtmax(ls, l, mid, L, R, p);
157         if (R > mid) edtmax(rs, mid + 1, r, L, R, p);
158         up(x);
159     }
160
161     inline int gmax(int x, int l, int r, int L, int R) {
162         if (L <= l && r <= R) return mx[x].p;
163         down(x, l, r);
164         int mid = l + r >> 1, ret = -inf;
165         if (L <= mid) ret = max(ret, gmax(ls, l, mid, L, R));
166         if (R > mid) ret = max(ret, gmax(rs, mid + 1, r, L, R));
167         return ret;
168     }
```

```
169
170     inline int gmin(int x, int l, int r, int L, int R) {
171         if (L <= l && r <= R) return mi[x].p;
172         down(x, l, r);
173         int mid = l + r >> 1, ret = inf;
174         if (L <= mid) ret = min(ret, gmin(ls, l, mid, L, R));
175         if (R > mid) ret = min(ret, gmin(rs, mid + 1, r, L, R));
176         return ret;
177     }
178
179     inline ll gsum(int x, int l, int r, int L, int R) {
180         if (L <= l && r <= R) return s[x];
181         down(x, l, r);
182         int mid = l + r >> 1;
183         ll ret = 0;
184         if (L <= mid) ret += gsum(ls, l, mid, L, R);
185         if (R > mid) ret += gsum(rs, mid + 1, r, L, R);
186         return ret;
187     }
188
189     inline void debug(int x, int l, int r) {
190         printf("%d %d %d [%d %d %d] [%d %d %d] %lld\n", x, l, r, mx[x].p, mx[x].t, mx[x
                ].se, mi[x].p, mi[x].t, mi[x].se,
191                s[x]);
192         if (l == r) return;
193         int mid = l + r >> 1;
194         debug(ls, l, mid);
195         debug(rs, mid + 1, r);
196     }
197
198 # undef ls
199 # undef rs
200 # undef inf
201 } T;
```

3.1.2 离散化区间

```
1  // 原题1e5个区间有2e5个端点，离散化出来4e5个区间
2  // 然后线段树需要4e5*4=16e5的大小
3  // 注意三个数组要开离散化数量的四倍，如果不需要sz可以不用这个数组。
4  int val[maxn << 4];
5  int lpos[maxn << 2], rpos[maxn << 2], tot, sz[maxn << 2];
6  vector<int> xpos;
7  sort(xpos.begin(), xpos.end());
8  xpos.erase(unique(xpos.begin(), xpos.end()), xpos.end());
9  tot = 1;
10 lpos[1] = rpos[1] = xpos[0];
11 sz[1] = 1;
12 for (int i = 1; i < xpos.size(); ++i) {
13     if (xpos[i] - xpos[i - 1] != 1) {
14         lpos[++tot] = xpos[i - 1] + 1;
15         rpos[tot] = xpos[i] - 1;
16         sz[tot] = rpos[tot] - lpos[tot] + 1;
17     }
18     ++tot;
19     lpos[tot] = rpos[tot] = xpos[i];
20     sz[tot] = 1;
21 }
```

```
22 le = lower_bound(lpos + 1, lpos + 1 + tot, p[i].x) - lpos;
23 re = upper_bound(rpos + 1, rpos + 1 + tot, p[i].y) - rpos - 1;
```

3.1.3 动态区间最大子段和

```
1  namespace ST {
2      struct node{
3          ll ans,ls,rs,sum;
4      }xx[maxn << 2];
5      inline void pushdown(int x){
6          xx[x].sum=xx[x<<1].sum+xx[x<<1|1].sum;
7          xx[x].ls=max(xx[x<<1].ls,xx[x<<1].sum+xx[x<<1|1].ls);
8          xx[x].rs=max(xx[x<<1|1].rs,xx[x<<1|1].sum+xx[x<<1].rs);
9          xx[x].ans=max(xx[x<<1].ans,max(xx[x<<1|1].ans,xx[x<<1].rs+xx[x<<1|1].ls));
10         return;
11     }
12     inline void build(int k,int l,int r){
13         if(l==r){
14             xx[k].ls=xx[k].rs=xx[k].ans=xx[k].sum=0;
15             return;
16         }
17         int mid=l+r>>1;
18         build(k<<1,l,mid),build(k<<1|1,mid+1,r);
19         pushdown(k);
20         return;
21     }
22     inline void change(int k,int l,int r,int x,int y,int w){ // 1, 1, n
23         if(x<=l&&r<=y){
24             xx[k].ls += w;
25             xx[k].rs += w;
26             xx[k].ans += w;
27             xx[k].sum += w;
28 //            xx[k].ls=xx[k].rs=xx[k].ans=xx[k].sum=w;
29             return;
30         }
31         int mid=l+r>>1;
32         if(x<=mid) change(k<<1,l,mid,x,y,w);
33         if(mid<y) change(k<<1|1,mid+1,r,x,y,w);
34         pushdown(k);
35         return;
36     }
37     inline node query(int k,int l,int r,int x,int y){
38         if(x<=l&&r<=y) {
39             return xx[k];
40         }
41         int mid=l+r>>1;
42         if(x<=mid&&!(mid<y)) return query(k<<1,l,mid,x,y);
43         else if(!(x<=mid)&&mid<y) return query(k<<1|1,mid+1,r,x,y);
44         else{
45             node st,t1=query(k<<1,l,mid,x,y),t2=query(k<<1|1,mid+1,r,x,y);
46             st.sum=t1.sum+t2.sum;
47             st.ls=max(t1.ls,t1.sum+t2.ls);
48             st.rs=max(t2.rs,t2.sum+t1.rs);
49             st.ans=max(t1.ans,max(t2.ans,t1.rs+t2.ls));
50             return st;
51         }
52     }
53 }
```

3.1.4 动态开点权值线段树

```
int root[100005];
int ls[1800000], rs[1800000], sum[1800000];
int sz = 0;

void insert(int &k, int l, int r, int val){
    if (!k) k = ++sz;
    if (l == r) {
        sum[k] = 1;
        return;
    }
    int mid = (l + r) >> 1;
    if (val <= mid) insert(ls[k], l, mid, val);
    else insert(rs[k], mid + 1, r, val);
    sum[k] = sum[ls[k]] + sum[rs[k]];
}

int query(int k, int l, int r, int rank) {
    if (l == r) return l;
    int mid = (l + r) >> 1;
    if (sum[ls[k]] >= rank) return query(ls[k], l, mid, rank);
    else return query(rs[k], mid + 1, r, rank - sum[ls[k]]);
}
int merge(int x, int y)
{
    if (!x) return y;
    if (!y) return x;
    ls[x] = merge(ls[x], ls[y]);
    rs[x] = merge(rs[x], rs[y]);
    sum[x] = sum[ls[x]] + sum[rs[x]];
    return x;
}
insert(root[i], 1, n, a[i]);
query(root[p], 1, n, x);
```

3.1.5 扫描线

```
// 范用型扫描线, del储存上界+1, add储存下界, 先del后add即可
struct node {
    int lpos, rpos, linepos;
    bool operator < (const node& oth) const {
        return linepos < oth.linepos;
    }
};
vector<node> add, del;
int delpos = 0;
int res = 0;
for (int addpos = 0; addpos < add.size(); ++addpos) {
    while (delpos < del.size() && del[delpos].linepos <= add[addpos].linepos) {
        up(del[delpos].lpos, del[delpos].rpos, -1);
        delpos ++;
    }
    up(add[addpos].lpos, add[addpos].rpos, 1);
    res = max(res, val[1]);
}

// 求面积并
```

```
#define maxn 222
#define tmp (st<<1)
#define mid ((l+r)>>1)
#define lson l,mid,tmp
#define rson mid+1,r,tmp|1
using namespace std;
int cnt[maxn<<2];
double sum[maxn<<2];
double x[maxn];
struct Seg{
    double h,l,r;
    int s;
    Seg(){}
    Seg(double a,double b,double c,int d):l(a),r(b),h(c),s(d){}
    bool operator<(const Seg &cmp)const{
        return h<cmp.h;
    }
}ss[maxn];
void push_up(int st,int l,int r){
    if(cnt[st])sum[st]=x[r+1]-x[l];
    else if(l==r)sum[st]=0;
    else sum[st]=sum[tmp]+sum[tmp|1];
}
void update(int L,int R,int c,int l,int r,int st){
    if(L<=l&&r<=R){
        cnt[st]+=c;
        push_up(st,l,r);
        return ;
    }
    if(L<=mid)update(L,R,c,lson);
    if(R>mid)update(L,R,c,rson);
    push_up(st,l,r);
}
int main(){
    int n,tot=1,m;
    while(scanf("%d",&n)&&n){
        double a,b,c,d;
        m=0;
        while(n--){
            scanf("%lf%lf%lf%lf",&a,&b,&c,&d);
            x[m]=a;
            ss[m++]=Seg(a,c,b,1);
            x[m]=c;
            ss[m++]=Seg(a,c,d,-1);
        }
        sort(x,x+m);
        sort(ss,ss+m);
        double ans=0;
        for(int i=0;i<m;++i){
            int l=lower_bound(x,x+m,ss[i].l)-x;
            int r=lower_bound(x,x+m,ss[i].r)-x-1;
            update(l,r,ss[i].s,0,m-1,1);
            ans+=sum[1]*(ss[i+1].h-ss[i].h);
        }
        printf("Test case #%dnTotal explored area: %.2lfnn",tot++,ans);
    }
    return 0;
}

```

```cpp
// 面积交
#include<bits/stdc++.h>
#define maxn 100005
#define lson l,mid,rt<<1
#define rson mid+1,r,rt<<1|1
#define pb push_back
using namespace std;

double tree[maxn<<2],tree2[maxn<<2];
int lazy[maxn<<2];
vector<double>ve;

struct seg{
    double l,r,h;
    int flag;
    seg(){}
    seg(double _l,double _r,double _h,int _flag){l=_l,r=_r,h=_h,flag=_flag;}
    bool operator<(const seg &b)const{return h<b.h;}
}s[maxn];

void push_up(int l,int r,int rt){
    if(lazy[rt]) tree[rt]=ve[r]-ve[l-1];
    else if(l==r) tree[rt]=0;
    else tree[rt]=tree[rt<<1]+tree[rt<<1|1];
}

void push_up2(int l,int r,int rt){
    if(lazy[rt]>1) tree2[rt]=ve[r]-ve[l-1];
    else if(l==r) tree2[rt]=0;
    else if(lazy[rt]==1)tree2[rt]=tree[rt<<1]+tree[rt<<1|1];
    else tree2[rt]=tree2[rt<<1]+tree2[rt<<1|1];
}

void build(int l,int r,int rt){
    tree[rt]=0,lazy[rt]=0;
    if(l==r) return;
    int mid=l+r>>1;
    build(lson);
    build(rson);
}

void add(int L,int R,int v,int l,int r,int rt){
    if(L<=l&&R>=r){
        lazy[rt]+=v;
        push_up(l,r,rt);
        push_up2(l,r,rt);
        return;
    }
    int mid=l+r>>1;
    if(L<=mid) add(L,R,v,lson);
    if(R>mid) add(L,R,v,rson);
    push_up(l,r,rt);
    push_up2(l,r,rt);
}

int getid(double x){ return lower_bound(ve.begin(),ve.end(),x)-ve.begin()+1;}

int main(){
    int n;
```

```
    int Case=1;
    int T;
    scanf("%d",&T);
    while(T--){
        scanf("%d",&n);
        ve.clear();
        int tot=0;
        double x1,y1,x2,y2;
        for(int i=1;i<=n;i++){
            scanf("%lf %lf %lf %lf",&x1,&y1,&x2,&y2);
            ve.pb(x1),ve.pb(x2);
            s[++tot]=seg(x1,x2,y1,1);
            s[++tot]=seg(x1,x2,y2,-1);
        }
        sort(ve.begin(),ve.end());
        ve.erase(unique(ve.begin(),ve.end()),ve.end());
        sort(s+1,s+tot+1);
        int N=ve.size();
        build(1,N,1);
        double ans=0;
        for(int i=1;i<tot;i++){
            int L=getid(s[i].l);
            int R=getid(s[i].r)-1;
            add(L,R,s[i].flag,1,N,1);
            ans+=tree2[1]*(s[i+1].h-s[i].h);
        }
        printf("%.2f\n",ans);
    }
}

// 求周长并
#include<bits/stdc++.h>
#define maxn 100005
#define lson l,mid,rt<<1
#define rson mid+1,r,rt<<1|1
#define pb push_back
using namespace std;

int tree[maxn<<2];
int lazy[maxn<<2];
vector<int>ve[2];
int k;

struct seg{
    int l,r,h;
    int flag;
    seg(){}
    seg(int _l,int _r,int _h,int _flag){l=_l,r=_r,h=_h,flag=_flag;}
    bool operator<(const seg &b)const{return h<b.h;}
}s[maxn];

void push_up(int l,int r,int rt){
    if(lazy[rt]) tree[rt]=ve[k][r]-ve[k][l-1];
    else if(l==r) tree[rt]=0;
    else tree[rt]=tree[rt<<1]+tree[rt<<1|1];
}

void build(int l,int r,int rt){
    tree[rt]=0,lazy[rt]=0;
```

```
198     if(l==r) return;
199     int mid=l+r>>1;
200     build(lson);
201     build(rson);
202 }
203
204 void add(int L,int R,int v,int l,int r,int rt){
205     if(L<=l&&R>=r){
206         lazy[rt]+=v;
207         push_up(l,r,rt);
208         return;
209     }
210     int mid=l+r>>1;
211     if(L<=mid) add(L,R,v,lson);
212     if(R>mid) add(L,R,v,rson);
213     push_up(l,r,rt);
214 }
215
216 int getid(int x){return lower_bound(ve[k].begin(),ve[k].end(),x)-ve[k].begin()+1;}
217
218 int main(){
219     int n;
220     while(~scanf("%d",&n)){
221         ve[0].clear();
222         ve[1].clear();
223         int x1,y1,x2,y2;
224         for(int i=1;i<=n;i++){
225             scanf("%d %d %d %d",&x1,&y1,&x2,&y2);
226             ve[0].pb(x1),ve[0].pb(x2);
227             ve[1].pb(y1),ve[1].pb(y2);
228             s[i]=seg(x1,x2,y1,1);
229             s[i+n]=seg(x1,x2,y2,-1);
230             s[i+n+n]=seg(y1,y2,x1,1);
231             s[i+n+n+n]=seg(y1,y2,x2,-1);
232         }
233         int ans=0;
234         int pos=1;
235         for(k=0;k<2;k++){
236             sort(ve[k].begin(),ve[k].end());
237             ve[k].erase(unique(ve[k].begin(),ve[k].end()),ve[k].end());
238             sort(s+pos,s+pos+n+n);
239             int N=ve[k].size();
240             build(1,N,1);
241             int pre=0;
242             for(int i=pos;i<pos+n+n;i++){
243                 int L=getid(s[i].l);
244                 int R=getid(s[i].r)-1;
245                 add(L,R,s[i].flag,1,N,1);
246                 ans+=abs(tree[1]-pre);
247                 pre=tree[1];
248             }
249             pos+=n+n;
250         }
251         printf("%d\n",ans);
252     }
253 }
```

3.2 HLD

3.2.1 HLD

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;
/*
node 计算点权， path 下放后计算边权， edge 根据边的编号计算边权
work 中没有build需手动写
sz[]数组，以x为根的子树节点个数
top[]数组，当前节点的所在链的顶端节点
son[]数组，重儿子
deep[]数组，当前节点的深度
fa[]数组，当前节点的父亲
idx[]数组，树中每个节点剖分后的新编号
rnk[]数组，idx的逆，表示线段上中当前位置表示哪个节点
*/

const int maxn = 1e5+5;

int sz[maxn], top[maxn], son[maxn], deep[maxn], fa[maxn], idx[maxn], rnk[maxn];
int tot;
int n, le, re;
ll k;

struct HLD {
#define type int

    struct edge {
        int a, b;
        type v;

        edge(int _a, int _b, type _v = 0) : a(_a), b(_b), v(_v) {}
    };

    struct node {
        int to;
        type w;

        node() {}

        node(int _to, type _w) : to(_to), w(_w) {}
    };

    vector<int> mp[maxn];
    vector<edge> e;

    void init(int _n) {
        n = _n;
        for (int i = 0; i <= n; i++) mp[i].clear();
        e.clear();
        e.push_back(edge(0, 0));
    }

    void add_edge(int a, int b, type v = 0) {
//        e.push_back(edge(a,b,v));
        mp[a].push_back(b);
        mp[b].push_back(a);
```

```
56      }
57
58      void dfs1(int x, int pre, int h) {
59          int i, to;
60          deep[x] = h;
61          fa[x] = pre;
62          sz[x] = 1;
63          for (i = 0; i < (int) (mp[x].size()); i++) {
64              to = mp[x][i];
65              if (to == pre) continue;
66              dfs1(to, x, h + 1);
67              sz[x] += sz[to];
68              if (son[x] == -1 || sz[to] > sz[son[x]]) son[x] = to;
69          }
70      }
71
72      void dfs2(int x, int tp) {
73          int i, to;
74          top[x] = tp;
75          idx[x] = ++tot;
76          rnk[idx[x]] = x;
77          if (son[x] == -1) return;
78          dfs2(son[x], tp);
79          for (i = 0; i < (int) (mp[x].size()); i++) {
80              to = mp[x][i];
81              if (to != son[x] && to != fa[x]) dfs2(to, to);
82          }
83      }
84
85      void work(int _rt = 1) {
86          memset(son, -1, sizeof son);
87          tot = 0;
88          dfs1(_rt, 0, 0);
89          dfs2(_rt, _rt);
90      }
91
92      int LCA(int x, int y) {
93          while (top[x] != top[y]) {
94              if (deep[top[x]] < deep[top[y]]) swap(x, y);
95              x = fa[top[x]];
96          }
97          if (deep[x] > deep[y]) swap(x, y);
98          return x;
99      }
100
101     void modify_node(int x, int y, type val) {
102         while (top[x] != top[y]) {
103             if (deep[top[x]] < deep[top[y]]) swap(x, y);
104             le = idx[top[x]], re = idx[x];
105             k = val;
106             update(1, 1, n);
107             x = fa[top[x]];
108         }
109         if (deep[x] > deep[y]) swap(x, y);
110         le = idx[x], re = idx[y];
111         k = val;
112         update(1, 1, n);
113     }
114
```

```
115     type query_node(int x, int y) {
116         type res = 0;
117         while (top[x] != top[y]) {
118             if (deep[top[x]] < deep[top[y]]) swap(x, y);
119             le = idx[top[x]], re = idx[x];
120             res += query(1, 1, n);
121             x = fa[top[x]];
122         }
123         if (deep[x] > deep[y]) swap(x, y);
124         le = idx[x], re = idx[y];
125         res += query(1, 1, n);
126         return res;
127     }
128 
129     //path
130 //    void init_path()
131 //    {
132 //        v[idx[rt]]=0;
133 //        for(int i=1;i<n;i++)
134 //        {
135 //            if(deep[e[i].a]<deep[e[i].b]) swap(e[i].a,e[i].b);
136 //            a[idx[e[i].a]]=e[i].v;
137 //        }
138 //        build(n);
139 //    }
140     void modify_edge(int id, type val) {
141         if (deep[e[id].a] > deep[e[id].b]) {
142             le = idx[e[id].a], re = idx[e[id].a];
143             k = val;
144             update(1, 1, n);
145         } else {
146             le = idx[e[id].b], re = idx[e[id].b];
147             k = val;
148             update(1, 1, n);
149         }
150     }
151 
152     void modify_path(int x, int y, type val) {
153         while (top[x] != top[y]) {
154             if (deep[top[x]] < deep[top[y]]) swap(x, y);
155             le = idx[top[x]], re = idx[x];
156             k = val;
157             update(1, 1, n);
158             x = fa[top[x]];
159         }
160         if (deep[x] > deep[y]) swap(x, y);
161         if (x != y) {
162             le = idx[x] + 1, re = idx[y];
163             k = val;
164             update(1, 1, n);
165         }
166     }
167 
168     type query_path(int x, int y) {
169         type res = 0;
170         while (top[x] != top[y]) {
171             if (deep[top[x]] < deep[top[y]]) swap(x, y);
172             le = idx[top[x]], re = idx[x];
173             res += query(1, 1, n);
```

```
            x = fa[top[x]];
        }
        if (deep[x] > deep[y]) swap(x, y);
        if (x != y) {
            le = idx[x] + 1, re = idx[y];
            res += query(1, 1, n);
        }
        return res;
    }

#undef type
} hld;
```

3.3 RMQ

3.3.1 RMQbyIndex

```
int pmax(int x, int y) { return a[x] > a[y] ? x : y; }
int pmin(int x, int y) { return a[x] < a[y] ? x : y; }
void init() {
    for (int i = 1; i <= n; i++) {
        maxx[i][0] = minn[i][0] = i;
    }
    for (int j = 1; 1 << (j - 1) <= n; j++) {
        for (int i = 1; i + (1 << j) - 1 <= n; i++) {
            int t = 1 << (j - 1);
            maxx[i][j] = pmax(maxx[i][j - 1], maxx[i + t][j - 1]);
            minn[i][j] = pmin(minn[i][j - 1], minn[i + t][j - 1]);
        }
    }
}
int query(int l, int r) {
    int j = 0;
    while ((1 << (j + 1)) <= r - l + 1) j++;
    int i = r - (1 << j) + 1;
//    return pmax(maxx[l][j], maxx[i][j]);
    return pmin(minn[l][j], minn[i][j]);
}
```

3.3.2 RMQinNM

```
//二维RMQ
int v[302][302];
int maxx[302][302][9][9], minn[302][302][9][9];

void RMQ(int n, int m) {
    int i, j, ii, jj;
    for (i = 1; i <= n; i++) {
        for (j = 1; j <= m; j++) {
            maxx[i][j][0][0] = minn[i][j][0][0] = v[i][j];
        }
    }
    for (ii = 0; (1 << ii) <= n; ii++) {
        for (jj = 0; (1 << jj) <= m; jj++) {
            if (ii + jj) {
                for (i = 1; i + (1 << ii) - 1 <= n; i++) {
                    for (j = 1; j + (1 << jj) - 1 <= m; j++) {
                        if (ii) {
```

```
18                              minn[i][j][ii][jj] = min(minn[i][j][ii - 1][jj], minn[i +
                                    (1 << (ii - 1))][j][ii - 1][jj]);
19                              maxx[i][j][ii][jj] = max(maxx[i][j][ii - 1][jj], maxx[i +
                                    (1 << (ii - 1))][j][ii - 1][jj]);
20                          } else {
21                              minn[i][j][ii][jj] = min(minn[i][j][ii][jj - 1], minn[i][j
                                    + (1 << (jj - 1))][ii][jj - 1]);
22                              maxx[i][j][ii][jj] = max(maxx[i][j][ii][jj - 1], maxx[i][j
                                    + (1 << (jj - 1))][ii][jj - 1]);
23                          }
24                      }
25                  }
26              }
27          }
28      }
29  }
30
31  int query(int x1, int y1, int x2, int y2) {
32      int k1 = 0;
33      while ((1 << (k1 + 1)) <= x2 - x1 + 1) k1++;
34      int k2 = 0;
35      while ((1 << (k2 + 1)) <= y2 - y1 + 1) k2++;
36      x2 = x2 - (1 << k1) + 1;
37      y2 = y2 - (1 << k2) + 1;
38      return max(max(maxx[x1][y1][k1][k2], maxx[x1][y2][k1][k2]), max(maxx[x2][y1][k1][k2
            ], maxx[x2][y2][k1][k2]));
39  //  return min(min(minn[x1][y1][k1][k2],minn[x1][y2][k1][k2]),min(minn[x2][y1][k1][k2],
        minn[x2][y2][k1][k2]));
40  }
```

3.4 MO

3.4.1 分块

```
1  // 非预处理数组版
2  inline int belong(int x) { return (x - 1) / block + 1; }
3  inline int lpos(int x) { return 1 + (x - 1) * block; }
4  inline int rpos(int x) { return min(n, x * block); }
5  int sz = (n - 1) / block + 1;
6
7  // 预处理版, maxn大于1e6已经不可能处理了
8  const int maxb = 1005;
9  int n, m;
10 int belong[maxn], lpos[maxb], rpos[maxb];
11 int val[maxn], lazy[maxb];
12 int block;
13
14 scanf("%d", &n);
15 block = sqrt(n);
16 for (int i = 1; i <= n; ++i) {
17     scanf("%d", &val[i]);
18     belong[i] = (i - 1) / block + 1;
19 }
20 int sz = (n - 1) / block + 1;
21 for (int i = 1; i <= sz; ++i) {
22     lpos[i] = 1 + (i - 1) * block;
23     rpos[i] = i * block;
24 }
25 rpos[sz] = n;
```

3.4.2 带修莫队

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;
const int maxn = 50005;

struct MO {
    int l, r, id, oppre;
}q[maxn];

int n, m, col[maxn], block, belong[maxn], colpre[maxn];
int changepos[maxn], changepre[maxn], changenow[maxn];
int vis[maxn * 20];
int ans;
int res[maxn];
bool cmp(const MO& a, const MO& b) {
    if (belong[a.l] != belong[b.l]) return a.l < b.l;
    if (belong[a.r] != belong[b.r]) return a.r < b.r;
    return a.oppre < b.oppre;
}
void add(int x) {}

void del(int x) {}

void unmodify(int pos, int now) {
    if (q[pos].l <= changepos[now] && changepos[now] <= q[pos].r) {
        del(changenow[now]);
        add(changepre[now]);
    }
    col[changepos[now]] = changepre[now];
}

void modify(int pos, int now) {
    if (q[pos].l <= changepos[now] && changepos[now] <= q[pos].r) {
        del(changepre[now]);
        add(changenow[now]);
    }
    col[changepos[now]] = changenow[now];
}

int main() {
    scanf("%d%d", &n, &m);
    block = pow(n, 0.66666);
    for (int i = 1; i <= n; ++i) {
        scanf("%d", &col[i]);
        colpre[i] = col[i];
        belong[i] = i / block + 1;
    }
    char s[2];
    int t = 0, t2 = 0;
    for (int i = 1; i <= m; ++i) {
        scanf("%s", s);
        if (s[0] == 'Q') {
            ++t;
            scanf("%d%d", &q[t].l, &q[t].r);
            q[t].oppre = t2;
            q[t].id = t;
        } else {
```

```
58              ++t2;
59              scanf("%d%d", &changepos[t2], &changenow[t2]);
60              changepre[t2] = colpre[changepos[t2]];
61              colpre[changepos[t2]] = changenow[t2];
62          }
63      }
64      sort(q + 1, q + 1 + t, cmp);
65      int l = 1, r = 0, now = 0;
66      for (int i = 1; i <= t; ++i) {
67          while(r < q[i].r) add(col[++r]);
68          while(r > q[i].r) del(col[r--]);
69          while(l < q[i].l) del(col[l++]);
70          while(l > q[i].l) add(col[--l]);
71          while (now < q[i].oppre) modify(i, ++now);
72          while (now > q[i].oppre) unmodify(i, now--);
73          res[q[i].id] = ans;
74      }
75      for (int i = 1; i <= t; ++i) printf("%d\n", res[i]);
76      return 0;
77  }
```

3.4.3 序列莫队

```
1   // const int maxn = 50005;
2
3   struct MO {
4       int l, r, id;
5   }q[maxn];
6
7   int n, m, col[maxn], block, belong[maxn];
8   int vis[maxn * 10];
9   ll res[maxn], ans;
10  bool cmp(const MO& a, const MO& b) { return belong[a.l] == belong[b.l] ? a.r < b.r : a.
        l < b.l; }
11  void add(int x) {
12      vis[x] ++;
13      ans += 1ll * x * (vis[x] * vis[x] - (vis[x] - 1) * (vis[x] - 1));
14  }
15
16  void del(int x) {
17      vis[x] --;
18      ans -= 1ll * x * ((vis[x] + 1) * (vis[x] + 1) - vis[x] * vis[x]);
19  }
20
21  int main() {
22      scanf("%d%d", &n, &m);
23      block = sqrt(n);
24      for (int i = 1; i <= n; ++i) {
25          scanf("%d", &col[i]);
26          belong[i] = i / block + 1;
27      }
28      for (int i = 1; i <= m; ++i) {
29          scanf("%d%d", &q[i].l, &q[i].r);
30          q[i].id = i;
31      }
32      sort(q + 1, q + 1 + m, cmp);
33      int l = 1, r = 0;
34      for (int i = 1; i <= m; ++i) {
```

```
35          while(r < q[i].r) add(col[++r]);
36          while(r > q[i].r) del(col[r--]);
37          while(l < q[i].l) del(col[l++]);
38          while(l > q[i].l) add(col[--l]);
39          res[q[i].id] = ans;
40      }
41      for (int i = 1; i <= m; ++i) printf("%lld\n", res[i]);
42      return 0;
43  }
```

3.4.4 弹飞绵羊

```
1   /*
2   每个装置设定初始弹力系数ki，当绵羊达到第i个装置时，它会往后弹ki步，
3   达到第i+ki个装置，若不存在第i+ki个装置，则绵羊被弹飞。
4   绵羊想知道当它从第i个装置起步时，被弹几次后会被弹飞。
5   为了使得游戏更有趣，Lostmonkey可以修改某个弹力装置的弹力系数，任何时候弹力系数均为正整
        数。
6   */
7
8   int n, m;
9   int belong[maxn], lpos[maxn], rpos[maxn];
10  int val[maxn], nxt[maxn], k[maxn], lst[maxn];
11  int block;
12
13  void update(int pos) {
14      int llim = lpos[belong[pos]], rlim = rpos[belong[pos]];
15      for (int i = pos; i >= llim; --i) {
16          if (val[i] + i > rlim) {
17              k[i] = 1;
18              nxt[i] = val[i] + i;
19              if (val[i] + i > n) lst[i] = i;
20              else lst[i] = lst[nxt[i]];
21          } else {
22              k[i] = 1 + k[val[i] + i];
23              nxt[i] = nxt[val[i] + i];
24              lst[i] = lst[val[i] + i];
25          }
26      }
27  }
28
29  void init() {
30      for (int i = n; i >= 1; --i) {
31          int rlim = rpos[belong[i]];
32          if (val[i] + i > rlim) {
33              k[i] = 1;
34              nxt[i] = val[i] + i;
35              if (val[i] + i > n) lst[i] = i;
36              else lst[i] = lst[nxt[i]];
37          } else {
38              k[i] = 1 + k[val[i] + i];
39              nxt[i] = nxt[val[i] + i];
40              lst[i] = lst[val[i] + i];
41          }
42      }
43  }
44
45  int query(int pos) {
```

```
46      int res = 0;
47      while (pos <= n) {
48          res += k[pos];
49          if (nxt[pos] > n) printf("%d ", lst[pos]);
50          pos = nxt[pos];
51      }
52      return res;
53  }
54
55  int main(int argc, char* argv[]) {
56      scanf("%d%d", &n, &m);
57      block = sqrt(n) * 1.6 + 1;
58      for (int i = 1; i <= n; ++i) {
59          scanf("%d", &val[i]);
60          belong[i] = (i - 1) / block + 1;
61      }
62      int sz = (n - 1) / block + 1;
63      for (int i = 1; i <= sz; ++i) {
64          lpos[i] = 1 + (i - 1) * block;
65          rpos[i] = i * block;
66      }
67      rpos[sz] = n;
68      init();
69      while (m--) {
70          int op;
71          scanf("%d", &op);
72          if (op == 1) {
73              int pos;
74              scanf("%d", &pos);
75              printf("%d\n", query(pos));
76          } else {
77              int pos, kl;
78              scanf("%d%d", &pos, &kl);
79              val[pos] = kl;
80              update(pos);
81          }
82      }
83      return 0;
84  }
```

3.4.5 树莫队

```
1   // rnk保存欧拉序
2   int sz[maxn], top[maxn], son[maxn], deep[maxn], fa[maxn], idx[maxn], ed[maxn], rnk[maxn
        *2];
3   int tot, n, m;
4   vector<int> edge[maxn];
5   int val[maxn];
6   vector<int> xpos;
7
8   inline void dfs1(int u, int pre, int h) {
9       deep[u] = h;
10      fa[u] = pre;
11      sz[u] = 1;
12      for (auto to : edge[u]) {
13          if (to == pre) continue;
14          dfs1(to, u, h + 1);
15          sz[u] += sz[to];
```

```
16          if (son[u] == 0 || sz[to] > sz[son[u]]) son[u] = to;
17      }
18  }
19
20  inline void dfs2(int u, int tp) {
21      top[u] = tp;
22      idx[u] = ++tot, rnk[tot] = u;
23      if (son[u] == 0) {
24          ed[u] = ++tot, rnk[tot] = u;
25          return;
26      }
27      dfs2(son[u], tp);
28      for (auto to : edge[u]) {
29          if (to != son[u] && to != fa[u]) dfs2(to, to);
30      }
31      ed[u] = ++tot, rnk[tot] = u;
32  }
33
34  inline int LCA(int x, int y) {
35      while (top[x] != top[y]) {
36          if (deep[top[x]] < deep[top[y]]) swap(x, y);
37          x = fa[top[x]];
38      }
39      if (deep[x] > deep[y]) swap(x, y);
40      return x;
41  }
42
43  int belong[maxn*2], block;
44  int res[maxn], ans;
45  // 每个点是否访问 (欧拉序去重)
46  int vis[maxn];
47  // 标记数组
48  int pre[maxn];
49  struct MO {
50      int l, r, id, lca;
51      bool operator < (const MO& oth) const {
52          return belong[l] == belong[oth.l] ? r < oth.r : belong[l] < belong[oth.l];
53      }
54  }q[maxm];
55
56  inline void add(int x) {
57      pre[x] ++;
58      if (pre[x] == 1) ans ++;
59  }
60
61  inline void del(int x) {
62      pre[x] --;
63      if (pre[x] == 0) ans --;
64  }
65
66  inline void deal(int x) {
67      vis[x] ? del(val[x]) : add(val[x]);
68      vis[x] = !vis[x];
69  }
70
71  int main(int argc, char* argv[]) {
72      scanf("%d%d", &n, &m);
73      block = sqrt(n);
74      xpos.resize(n + 1);
```

```
75      for (int i = 1; i <= n; ++i) {
76          scanf("%d", &val[i]);
77          xpos[i] = val[i];
78      }
79      sort(xpos.begin(), xpos.end());
80      xpos.erase(unique(xpos.begin(), xpos.end()), xpos.end());
81      for (int i = 1; i <= n; ++i) val[i] = lower_bound(xpos.begin(), xpos.end(), val[i])
            - xpos.begin();
82      // 欧拉序长度为n两倍所以分块要分两倍大小
83      for (int i = 1; i <= n * 2; ++i) {
84          belong[i] = (i - 1) / block + 1;
85      }
86      for (int i = 1, u, v; i < n; ++i) {
87          scanf("%d%d", &u, &v);
88          edge[u].push_back(v);
89          edge[v].push_back(u);
90      }
91      // 树剖预处理lca
92      dfs1(1, 0, 0);
93      dfs2(1, 1);
94      for (int i = 1, x, y; i <= m; ++i) {
95          scanf("%d%d", &x, &y);
96          if (idx[x] > idx[y]) swap(x, y);
97          int _lca = LCA(x, y);
98          q[i].id = i;
99          if (_lca == x) q[i].l = idx[x], q[i].r = idx[y], q[i].lca = 0;
100         else q[i].l = ed[x], q[i].r = idx[y], q[i].lca = _lca;
101 //        cerr << q[i].l << " " << q[i].r << " " << q[i].id << " " << q[i].lca << endl;
102     }
103     sort(q + 1, q + 1 + m);
104     int l = 1, r = 0;
105     for (int i = 1; i <= m; ++i) {
106         while(r < q[i].r) deal(rnk[++r]);
107         while(r > q[i].r) deal(rnk[r--]);
108         while(l < q[i].l) deal(rnk[l++]);
109         while(l > q[i].l) deal(rnk[--l]);
110         if (q[i].lca) deal(q[i].lca);
111         res[q[i].id] = ans;
112         if (q[i].lca) deal(q[i].lca);
113     }
114     for (int i = 1; i <= m; ++i) {
115         printf("%d\n", res[i]);
116     }
117     return 0;
118 }
```

3.5 VirtualTree

3.5.1 VirtualTree

```
1  const int pow2 = 19;
2  int n;
3  vector<int> adj0[maxn], adj1[maxn];
4  int st[maxn << 1][pow2 + 1], dep[maxn], euler[maxn], euler_clock;
5  int stk[maxn], fa0[maxn];
6  vector<int> cache;
7  void link0(int u, int v) { adj0[u].emplace_back(v); adj0[v].emplace_back(u); }
8  void link1(int u, int v) { adj1[u].emplace_back(v), cache.push_back(u); }
9  void clearAll() {
```

```
    for (int i = 1; i <= n; ++i) {
        adj0[i].clear();
        adj1[i].clear();
    }
    euler_clock = 0;
}
void clearVT() { for (auto i : cache) adj1[i].clear(); cache.clear(); }
void dfs0(int u, int p) {
    fa0[u] = p;
    dep[u] = dep[p] + 1;
    st[++euler_clock][0] = u;
    euler[u] = euler_clock;
    for (const auto& v : adj0[u]) if (v != p) {
            dfs0(v, u);
            st[++euler_clock][0] = u;
        }
}
inline bool cmp(int u, int v) {return dep[u] < dep[v];}
inline int upper(int u, int v) {return cmp(u, v) ? u : v;}
void lca_init() {
    for (int i = 0; i != 31 - __builtin_clz(euler_clock); ++i)
        for (int j = 1; j + (1 << (i + 1)) <= euler_clock; ++j)
            st[j][i + 1] = upper(st[j][i], st[j + (1 << i)][i]);
}
inline int lca(int u, int v) {
    if (u == v) return u;
    u = euler[u];
    v = euler[v];
    if (u > v) swap(u, v);
    int temp = 31 - __builtin_clz(++v - u);
    return upper(st[u][temp], st[v - (1 << temp)][temp]);
}
// build 后 stk[1] 是该树的根节点, 且为有根树
void build(vector<int>& key) {
    sort(key.begin(), key.end(), [&] (int u, int v) { return euler[u] < euler[v]; });
    key.erase(unique(key.begin(), key.end()), key.end());
    int top = 0;
    for (auto u : key) {
        if (!top) {
            stk[++top] = u;
            continue;
        }
        int p = lca(u, stk[top]);
        while (euler[p] < euler[stk[top]]) {
            if (euler[p] >= euler[stk[top - 1]]) {
                link1(p, stk[top]);
                if (stk[--top] != p) stk[++top] = p;
                break;
            }
            link1(stk[top - 1], stk[top]);
            --top;
        }
        stk[++top] = u;
    }
    while (top > 1) {
        link1(stk[top - 1], stk[top]);
        --top;
    }
}
```

```

int f[maxn];
int res;
int vis[maxn];
void dfs1(int u) {
    fa1[u] = p;
    len[u] = dep[u] - dep[p];
    for (auto v : adj1[u]) {
        dfs1(v);
    }
}

int main(int argc, char* argv[]) {
    scanf("%d", &n);
    for (int i = 1, u, v; i < n; ++i) {
        scanf("%d%d", &u, &v);
        link0(u, v);
    }
    dfs0(1, 0);
    lca_init();
    int m; scanf("%d", &m);
    for (int i = 0; i < m; ++i) {
        int sz; scanf("%d", &sz);
        vector<int> key(sz);
        for (int j = 0; j < sz; ++j) {
            scanf("%d", &key[j]);
            vis[key[j]] = 1;
        }
        build(key);
        res = 0;
        dfs1(stk[1]);
        printf("%d\n", res);
        for (int j = 0; j < sz; ++j) vis[key[j]] = 0;
        clearVT();
    }
    return 0;
}
```

3.6 PersistentDS

3.6.1 主席树区间 k 大

```
// const int maxn = 100005;
int n, m;
int a[maxn];
int root[maxn];
int cnt = 0;
vector<int> b;
struct node {
    int l, r, val;
}p[maxn * 40];

void update(int l, int r, int pre, int &now, int pos) {
    p[now = ++cnt] = p[pre];
    p[now].val++;
    if (l == r) {
        return;
    }
    int mid = l + r >> 1;
```

```
18      if (pos <= mid) update(l, mid, p[pre].l, p[now].l, pos);
19      else update(mid + 1, r, p[pre].r, p[now].r, pos);
20  }
21
22  /*
23  void build(int pre, int &now, int pos) {
24      now = pre;
25      for (auto i : a[pos]) {
26          update(1, n, pre, now, i);
27      }
28  }
29  */
30
31  int query(int l, int r, int x, int y, int k) {
32      if (l == r) return b[l - 1];
33      int mid = l + r >> 1;
34      int temp = p[p[y].l].val - p[p[x].l].val;
35      if (k <= temp) return query(l, mid, p[x].l, p[y].l, k);
36      return query(mid + 1, r, p[x].r, p[y].r, k - temp);
37  }
38
39  int main(int argc,char *argv[])
40  {
41      while (scanf("%d%d", &n, &m) != EOF) {
42          b.clear();
43          cnt = 0;
44          for (int i = 1; i <= n; ++i) scanf("%d", &a[i]), b.push_back(a[i]);
45          sort(b.begin(), b.end());
46          b.erase(unique(b.begin(), b.end()), b.end());
47          for (int i = 1; i <= n; ++i) {
48              update(1, b.size(), root[i - 1], root[i], lower_bound(b.begin(), b.end(), a
                    [i]) - b.begin() + 1);
49          }
50          int L, R, k;
51          while (m--) {
52              scanf("%d%d%d", &L, &R, &k);
53              printf("%d\n", query(1, b.size(), root[L - 1], root[R], k));
54          }
55      }
56      return 0;
57  }
```

3.6.2 动态森林

```
1   const int maxn = 10010;
2   struct Splay {
3       int ch[maxn][2], fa[maxn], tag[maxn];
4       void clear(int x) { ch[x][0] = ch[x][1] = fa[x] = tag[x] = 0; }
5       int getch(int x) { return ch[fa[x]][1] == x; }
6       int isroot(int x) { return ch[fa[x]][0] != x && ch[fa[x]][1] != x; }
7       void pushdown(int x) {
8           if (tag[x]) {
9               if (ch[x][0]) swap(ch[ch[x][0]][0], ch[ch[x][0]][1]), tag[ch[x][0]] ^= 1;
10              if (ch[x][1]) swap(ch[ch[x][1]][0], ch[ch[x][1]][1]), tag[ch[x][1]] ^= 1;
11              tag[x] = 0;
12          }
13      }
14      void update(int x) {
```

```
15          if (!isroot(x)) update(fa[x]);
16          pushdown(x);
17      }
18      void rotate(int x) {
19          int y = fa[x], z = fa[y], chx = getch(x), chy = getch(y);
20          fa[x] = z;
21          if (!isroot(y)) ch[z][chy] = x;
22          ch[y][chx] = ch[x][chx ^ 1];
23          fa[ch[x][chx ^ 1]] = y;
24          ch[x][chx ^ 1] = y;
25          fa[y] = x;
26      }
27      void splay(int x) {
28          update(x);
29          for (int f = fa[x]; f = fa[x], !isroot(x); rotate(x))
30              if (!isroot(f)) rotate(getch(x) == getch(f) ? f : x);
31      }
32      void access(int x) {
33          for (int f = 0; x; f = x, x = fa[x]) splay(x), ch[x][1] = f;
34      }
35      void makeroot(int x) {
36          access(x);
37          splay(x);
38          swap(ch[x][0], ch[x][1]);
39          tag[x] ^= 1;
40      }
41      int find(int x) {
42          access(x);
43          splay(x);
44          while (ch[x][0]) x = ch[x][0];
45          splay(x);
46          return x;
47      }
48  } st;
49
50  // 动态森林
51  namespace LCT {
52      void clear(int n) { for (int i = 0; i <= n; ++i) st.clear(i); }
53      bool isConnect(int u, int v) { return st.find(u) == st.find(v); }
54      bool add(int u, int v) {
55          if (isConnect(u, v)) return false;
56          st.makeroot(u), st.fa[u] = v;
57          return true;
58      }
59      // 无法判断是否存在直接相连的边
60      void del(int u, int v) {
61          st.makeroot(u);
62          st.access(v);
63          st.splay(v);
64          if (st.ch[v][0] == u && !st.ch[u][1]) st.ch[v][0] = st.fa[u] = 0;
65      }
66  }
67  using namespace LCT;
```

3.7 Tree

3.7.1 LCA

```
1  // const int maxn = 1e5 + 10;
```

```

// 普通倍增lca
int n, dep[maxn], fa[maxn][30];
vector<int> edge[maxn];

void dfs(int u, int pre) {
    dep[u] = dep[pre] + 1, fa[u][0] = pre;
    for(int i = 1; (1 << i) <= n; i ++)
        fa[u][i] = fa[fa[u][i - 1]][i - 1];
    for(auto v : edge[u]) if(v != pre) dfs(v, u);
}

int LCA(int u, int v) {
    if(dep[u] < dep[v]) swap(u, v);
    int d = dep[u] - dep[v];
    for(int i = 0; (1 << i) <= d; i ++)
        if((1 << i) & d) u = fa[u][i];
    if(u == v) return u;
    for(int i = 20; i >= 0; i --)
        if(fa[u][i] != fa[v][i])
            u = fa[u][i], v = fa[v][i];
    return fa[u][0];
}
```

3.7.2 点分治

```
int n, k;

// 清零 head 和 tot
const int maxm = maxn * 2;
int ver[maxm], Next[maxm], head[maxn], edge[maxm];
int tot;
void addEdge(int u, int v, int w){
    ver[++tot]=v;
    Next[tot]=head[u];
    head[u]=tot;
    edge[tot]=w;
}

int sz[maxn], vis[maxn];
int rt, mxsz, has;

void getrt(int u, int pre) {
    sz[u] = 1;
    int mxnow = 0;
    for (int i = head[u]; i; i = Next[i]) {
        int v = ver[i];
        if (v == pre || vis[v]) continue;
        getrt(v, u);
        sz[u] += sz[v];
        mxnow = max(mxnow, sz[v]);
    }
    mxnow = max(mxnow, has - sz[u]);
    if (mxnow < mxsz) {
        mxsz = mxnow, rt = u;
    }
}

```

```
33 int dl[maxn], r;
34 int val[maxn];
35
36 void getdis(int u, int pre) {
37     dl[r++] = val[u];
38     for (int i = head[u]; i; i = Next[i]) {
39         int v = ver[i];
40         if (v == pre || vis[v]) continue;
41         val[v] = val[u] + edge[i];
42         getdis(v, u);
43     }
44 }
45
46 ll cal(int u, int pre) {
47     r = 0;
48     val[u] = pre;
49     getdis(u, 0);
50     ll sum = 0;
51     sort(dl, dl + r);
52     r --;
53     int l = 0;
54     while (l < r) {
55         if (dl[l] + dl[r] > k) r --;
56         else sum += r - l, l ++;
57     }
58     return sum;
59 }
60
61 ll res = 0;
62 void dfs(int u) {
63     res += cal(u, 0);
64     vis[u] = 1;
65     for (int i = head[u]; i; i = Next[i]) {
66         int v = ver[i];
67         if (vis[v]) continue;
68         res -= cal(v, edge[i]);
69         has = sz[v];
70         mxsz = 0x3f3f3f3f;
71         getrt(v, 0);
72         dfs(rt);
73     }
74 }
75
76 int main(int argc, char* argv[]) {
77     while (scanf("%d%d", &n, &k) != EOF && (n || k)) {
78         tot = 0; memset(head, 0, sizeof head);
79         memset(vis, 0, sizeof vis);
80         res = 0;
81         for (int i = 1, u, v, w; i < n; ++i) {
82             scanf("%d%d%d", &u, &v, &w);
83             addEdge(u, v, w);
84             addEdge(v, u, w);
85         }
86         mxsz = 0x3f3f3f3f;
87         has = n;
88         getrt(1, 0);
89         dfs(rt);
90         printf("%lld\n", res);
91     }
```

```
92      return 0;
93  }
```

3.8 Splay

```
1   /*
2   1. 插入x数
3   2. 删除x数(若有多个相同的数, 因只删除一个)
4   3. 查询x数的排名(若有多个相同的数, 因输出最小的排名)
5   4. 查询排名为x的数
6   5. 求x的前驱(前驱定义为小于x, 且最大的数)
7   6. 求x的后继(后继定义为大于x, 且最小的数)
8   */
9
10  const int N = 1e5 + 7;
11
12  struct Splay {
13      int ch[N][2], fa[N], val[N], cnt[N], size[N], tol, root;
14      inline bool chk(int x) {
15          return ch[fa[x]][1] == x;
16      }
17      inline void pushup(int x) {
18          size[x] = size[ch[x][0]] + size[ch[x][1]] + cnt[x];
19      }
20      void rotate(int x) {
21          int y = fa[x], z = fa[y], k = chk(x), w = ch[x][k ^ 1];
22          ch[y][k] = w; fa[w] = y;
23          ch[z][chk(y)] = x; fa[x] = z;
24          ch[x][k ^ 1] = y; fa[y] = x;
25          pushup(y); pushup(x);
26      }
27      void splay(int x, int goal = 0) {
28          while (fa[x] != goal) {
29              int y = fa[x], z = fa[y];
30              if (z != goal) {
31                  if (chk(x) == chk(y)) rotate(y);
32                  else rotate(x);
33              }
34              rotate(x);
35          }
36          if (!goal) root = x;
37      }
38      void insert(int x) {
39          int cur = root, p = 0;
40          while (cur && val[cur] != x) {
41              p = cur;
42              cur = ch[cur][x > val[cur]];
43          }
44          if (cur) {
45              cnt[cur]++;
46          } else {
47              cur = ++tol;
48              if (p) ch[p][x > val[p]] = cur;
49              ch[cur][0] = ch[cur][1] = 0;
50              fa[cur] = p; val[cur] = x;
51              cnt[cur] = size[cur] = 1;
52          }
53          splay(cur);
```

```
54      }
55      void find(int x) {
56          int cur = root;
57          while (ch[cur][x > val[cur]] && x != val[cur])
58              cur = ch[cur][x > val[cur]];
59          splay(cur);
60      }
61      int kth(int k) {
62          int cur = root;
63          while (1) {
64              if (ch[cur][0] && k <= size[ch[cur][0]])
65                  cur = ch[cur][0];
66              else if (k > size[ch[cur][0]] + cnt[cur])
67                  k -= size[ch[cur][0]] + cnt[cur], cur = ch[cur][1];
68              else
69                  break;
70          }
71          return cur;
72      }
73      int pre(int x) {
74          find(x);
75          if (val[root] < x) return root;
76          int cur = ch[root][0];
77          while (ch[cur][1]) cur = ch[cur][1];
78          return cur;
79      }
80      int succ(int x) {
81          find(x);
82          if (val[root] > x) return root;
83          int cur = ch[root][1];
84          while (ch[cur][0]) cur = ch[cur][0];
85          return cur;
86      }
87      void del(int x) {
88          int last = pre(x), nxt = succ(x);
89          splay(last); splay(nxt, last);
90          int del = ch[nxt][0];
91          if (cnt[del] > 1)
92              cnt[del]--, splay(del);
93          else
94              ch[nxt][0] = 0;
95      }
96      int getrk(int x) {
97          find(x);
98          return size[ch[root][0]];
99      }
100 } splay;
101
102 int n;
103
104 int main() {
105     //freopen("in.txt", "r", stdin);
106     splay.insert(0x3f3f3f3f);
107     splay.insert(0xcfcfcfcf);
108     read(n);
109     while (n--) {
110         int opt, x;
111         read(opt, x);
112         if (opt == 1) splay.insert(x);
```

```
113         else if (opt == 2) splay.del(x);
114         else if (opt == 3) print(splay.getrk(x));
115         else if (opt == 4) print(splay.val[splay.kth(x + 1)]);
116         else if (opt == 5) print(splay.val[splay.pre(x)]);
117         else print(splay.val[splay.succ(x)]);
118     }
119     flush();
120     return 0;
121 }
```

3.9 Others

3.9.1 BITinNM

```
1  struct Fenwick_Tree {
2  #define type int
3      type bit[maxn][maxn];
4      int n, m;
5      void init(int _n, int _m) {
6          n = _n;
7          m = _m;
8          mem(bit, 0);
9      }
10     int lowbit(int x) { return x & (-x); }
11     void update(int x, int y, type v) {
12         int i, j;
13         for (i = x; i <= n; i += lowbit(i)) {
14             for (j = y; j <= m; j += lowbit(j)) {
15                 bit[i][j] += v;
16             }
17         }
18     }
19     type get(int x, int y) {
20         type i, j, res = 0;
21         for (i = x; i > 0; i -= lowbit(i)) {
22             for (j = y; j > 0; j -= lowbit(j)) {
23                 res += bit[i][j];
24             }
25         }
26         return res;
27     }
28     type query(int x1, int x2, int y1, int y2) {
29         x1--;
30         y1--;
31         return get(x2, y2) - get(x1, y2) - get(x2, y1) + get(x1, y1);
32     }
33 #undef type
34 } tr;
35
36 // 二维区间前缀和写法（非树状数组）
37 inline void range_add(int xa, int ya, int xb, int yb) { add(xa, ya, 1), add(xa, yb + 1,
       -1), add(xb + 1, ya, -1), add(xb + 1, yb + 1, 1); }
38 inline ll range_ask(int xa, int ya, int xb, int yb){ return ask(xb, yb) - ask(xb, ya -
      1) - ask(xa - 1, yb) + ask(xa - 1, ya - 1); }
39 inline void build() {
40     // 预处理出每个点的单点值
41     for (int i = 1; i < n + 5; ++i) {
42         for (int j = 1; j < m + 5; ++j) {
43             st[i][j] += st[i - 1][j] + st[i][j - 1] - st[i - 1][j - 1];
```

```
        }
    }
    // 再求一次处理出每个点的前缀和
    for (int i = 1; i < n + 5; ++i) {
        for (int j = 1; j < m + 5; ++j) {
            if (st[i][j] > 1) st[i][j] = 1;
            st[i][j] += st[i - 1][j] + st[i][j - 1] - st[i - 1][j - 1];
        }
    }
}

// 二维树状数组区间加与求和
ll t1[maxn][maxn], t2[maxn][maxn], t3[maxn][maxn], t4[maxn][maxn];
void add(ll x, ll y, ll z){
    for(int X = x; X <= n; X += X & -X)
        for(int Y = y; Y <= m; Y += Y & -Y){
            t1[X][Y] += z;
            t2[X][Y] += z * x;
            t3[X][Y] += z * y;
            t4[X][Y] += z * x * y;
        }
}
ll ask(ll x, ll y){
    ll res = 0;
    for(int i = x; i; i -= i & -i)
        for(int j = y; j; j -= j & -j)
            res += (x + 1) * (y + 1) * t1[i][j]
                - (y + 1) * t2[i][j]
                - (x + 1) * t3[i][j]
                + t4[i][j];
    return res;
}

// 区间加, 询问单点: 直接维护前缀差分数组, 求单点=普通求前缀和
```

3.9.2 静态区间 k 大划分树

```
// const int maxn = 100010;
int tree[20][maxn];
// 读入sorted并排序, 赋值给tree的第0层
int sorted[maxn];
int toleft[20][maxn];
// 保存左子树的和
// ll sum[20][maxn];

// 1, n, 0
void build(int l, int r, int dep) {
    if (l == r) return;
    // sum[dep][0] = 0;
    toleft[dep][0] = 0;
    int mid = l + r >> 1;
    int same = mid - l + 1;
    for (int i = l; i <= r; ++i) {
        if (tree[dep][i] < sorted[mid]) same--;
    }
    int lpos = l, rpos = mid + 1;
    for (int i = l; i <= r; ++i) {
        // sum[dep][i] = sum[dep][i - 1];
```

```
        if (tree[dep][i] < sorted[mid]) {
            // sum[dep][i] += tree[dep][i];
            tree[dep + 1][lpos++] = tree[dep][i];
        }
        else if (tree[dep][i] == sorted[mid] && same > 0) {
            // sum[dep][i] += tree[dep][i];
            tree[dep + 1][lpos++] = tree[dep][i];
            same --;
        } else tree[dep + 1][rpos ++] = tree[dep][i];
        toleft[dep][i] = toleft[dep][l - 1] + lpos - l;
    }
    build(l, mid, dep + 1);
    build(mid + 1, r, dep + 1);
}

//(1~k-1)的数的和, 注意每次查询前初始化
// ll ress = 0;

// L = 1, R = n,  dep = 0, l,r是查询区间
int query(int L, int R, int l, int r, int dep, int k) {
    if (l == r) return tree[dep][l];
    int mid = (L + R) >> 1;
    int cnt = toleft[dep][r] - toleft[dep][l - 1];
    if (cnt >= k) {
        int newl = L + toleft[dep][l - 1] - toleft[dep][L - 1];
        int newr = newl + cnt - 1;
        return query(L, mid, newl, newr, dep + 1, k);
    } else {
        int newr = r + toleft[dep][R] - toleft[dep][r];
        int newl = newr - (r - l - cnt);
        // ress += sum[dep][r] - sum[dep][l - 1];
        return query(mid + 1, R, newl, newr, dep + 1, k - cnt);
    }
}


scan(n), scan(m);
for (int i = 1; i <= n; ++i) {
    scan(sorted[i]);
    tree[0][i] = sorted[i];
}
sort(sorted + 1, sorted + 1 + n);
build(1, n, 0);
int l, r, k;
while (m--) {
    scan(l), scan(r), scan(k);
    printf("%d\n", query(1, n, l, r, 0, k));
}
```

3.9.3 二叉堆

```
template<class T = int>
struct BinaryHeap {
    vector<T> Heap;
    int sz;
    BinaryHeap(int s = 0) { Heap.resize(s), sz = 0; }
    void init(int s) { Heap.resize(s), sz = 0; }
    bool less(T a, T b) { return a > b; }
```

```
8       void up(int p) {
9           while (p > 1) {
10              if (less(Heap[p / 2], Heap[p])) {
11                  swap(Heap[p], Heap[p / 2]);
12                  p /= 2;
13              } else break;
14          }
15      }
16      void down(int p) {
17          int s = p * 2;
18          while (s <= sz) {
19              if (s < sz && less(Heap[s], Heap[s + 1])) s++;
20              if (less(Heap[p], Heap[s])) {
21                  swap(Heap[s], Heap[p]);
22                  p = s, s = p * 2;
23              } else break;
24          }
25      }
26      void insert(int val) {
27          Heap[++sz] = val;
28          up(sz);
29      }
30      void removeTop() {
31          Heap[1] = Heap[sz--];
32          down(1);
33      }
34      void remove(int pos) {
35          Heap[pos] = Heap[sz--];
36          up(pos), down(pos);
37      }
38      int getTop() {
39         assert(sz > 0);
40         return Heap[1];
41      }
42  };
```

4 String

4.1 KMP

4.1.1 KMP

```
// nxt[0]表示失配到完全不匹配
int nxt[maxm];

void getNext(char *s, int len) {
    int i = 0, j = -1;
    nxt[i] = j;
    while (i < len) {
        if (j == -1 || s[i] == s[j]) nxt[++i] = ++j;
        else j = nxt[j];
    }
}

// a为原串, b为模式串, 下标从0开始, 找第一个出现模式串的位置 (起点为1) , 找不到返回-1
int KMP(char *a, char *b, int n, int m) {
    getNext(b, m);
    int i = 0, j = 0;
    while (i < n && j < m) {
        if (j == -1 || a[i] == b[j]) ++i, ++j;
        else j = nxt[j];
    }
    return j == m ? i - m + 1 : -1;
}
```

4.1.2 exKMP

```
const int maxn = 1e5 + 10;
int nex[maxn], extend[maxn];

//预处理计算Next数组
void getNext(char *str)
{
    int i = 0, j, po, len = strlen(str);
    nex[0] = len;     //初始化nex[0]
    while (str[i] == str[i + 1] && i + 1 < len) i++;   //计算nex[1]
    nex[1] = i;
    po = 1;   //初始化po的位置
    for (int i = 2; i < len; i++)
    {
        if (nex[i - po] + i < nex[po] + po)  //第一种情况, 可以直接得到nex[i]的值
            nex[i] = nex[i - po];
        else    //第二种情况, 要继续匹配才能得到nex[i]的值
        {
            j = nex[po] + po - i;
            if (j < 0) j = 0;    //如果i>po+nex[po],则要从头开始匹配
            while (i + j < len && str[j] == str[j + i]) j++;
            nex[i] = j;
            po = i;   //更新po的位置
        }
    }
}

void EXKMP(char *s1, char *s2)
{
```

```
    int i = 0, j, po, len = strlen(s1), l2 = strlen(s2);
    getNext(s2);
    while (s1[i] == s2[i] && i < l2 && i < len) i++;
    extend[0] = i;
    po = 0;
    for (int i = 1; i < len; i++)
    {
        if (nex[i - po] + i < extend[po] + po)
            extend[i] = nex[i - po];
        else
        {
            j = extend[po] + po - i;
            if (j < 0) j = 0;
            while (i + j < len && j < l2 && s1[j + i] == s2[j]) j++;
            extend[i] = j;
            po = i;
        }
    }
}
```

4.2 Trie

4.2.1 Trie

```
const int maxn = 2e6 + 10;

int trie[maxn][30], tot;
bool flag[maxn];

void insert_ch(char *str)
{
    int len = strlen(str);
    int root = 0;
    for (int i = 0; i < len; i++)
    {
        int id = str[i] - 'a';
        if (!trie[root][id]) trie[root][id] = ++tot;
        root = trie[root][id];
    }
    flag[root] = true;
}

bool find_ch(char *str)
{
    int len = strlen(str);
    int root = 0;
    for (int i = 0; i < len; i++)
    {
        int id = str[i] - 'a';
        if (!trie[root][id]) return false;
        root = trie[root][id];
    }
    return true;
}
```

4.2.2 Persistence Trie

```
const int maxn = 1e5 + 10;

int a[maxn], rt[maxn], n;

struct Trie
{
    int tot;
    int child[maxn * 32][2], sum[maxn *32];
    int insert(int x, int val)
    {
        int tmp, y;
        tmp = y= ++tot;
        for(int i = 30; i >= 0; --i)
        {
            child[y][0] = child[x][0];
            child[y][1] = child[x][1];
            sum[y] = sum[x] + 1;
            int t = val >> i & 1;
            x = child[x][t];
            child[y][t] = ++tot;
            y = child[y][t];
        }
        sum[y] = sum[x] + 1;
        return tmp;
    }
    int query(int l, int r, int val)
    {
        int tmp = 0;
        for(int i =30; i >= 0; --i)
        {
            int t = val >> i & 1;
            if(sum[child[r][t^1]] - sum[child[l][t^1]]) tmp += (1<<i), r = child[r][t
                ^1], l = child[l][t ^ 1];
            else r = child[r][t], l = child[l][t];
        }
        return tmp;
    }
}trie;
```

4.2.3 01Trie

```
struct Trie {
    int tree[maxn*20][2], tot;
    int flag[maxn*20];

    void insert_ch(int x) {
        int root = 0;
        flag[0]++;
        for (int i = 30; i >= 0; --i) {
            int id = (x >> i) & 1;
            if (!tree[root][id]) {
                tree[root][id] = ++tot;
                tree[tree[root][id]][0] = tree[tree[root][id]][1] = 0;
                flag[tree[root][id]] = flag[tree[tree[root][id]][0]] = flag[tree[tree[
                    root][id]][1]] = 0;
            }
            root = tree[root][id];
            flag[root]++;
```

```
17          }
18      }
19
20      void del(int x) {
21          int root = 0;
22          flag[0]--;
23          for (int i = 30; i >= 0; --i) {
24              int id = (x >> i) & 1;
25              assert(tree[root][id]);
26              if (flag[tree[root][id]] == 1) {
27                  flag[tree[root][id]] = 0;
28                  tree[root][id] = 0;
29                  return;
30              }
31              root = tree[root][id];
32              flag[root]--;
33          }
34      }
35
36      int find_ch(int x, int flag = 0) { // flag 0 最小异或值, 1 最大异或值
37          int root = 0;
38          int res = 0;
39          for (int i = 30; i >= 0; --i) {
40              int id = ((x >> i) & 1);
41              if (flag) id = !id;
42              if (tree[root][id]) {
43                  root = tree[root][id];
44                  res = res << 1 | id;
45              } else {
46                  root = tree[root][!id];
47                  res = res << 1 | (!id);
48              }
49          }
50          return res;
51      }
52
53      void init() {
54          tree[0][0] = tree[0][1] = 0;
55          tot = 0;
56      }
57  };
```

4.3 Manachar

4.3.1 Manacher

```
1   const int maxn = 1e5 + 10;
2
3   char s[maxn];
4
5   char tmp[maxn << 1];
6   int Len[maxn << 1];
7
8   int init(char *str)
9   {
10      int len = strlen(str);
11      tmp[0] = '@';
12      for (int i = 1; i <= 2 * len; i += 2)
13      {
```

```
14          tmp[i] = '#';
15          tmp[i + 1] = str[i / 2];
16      }
17      tmp[2 * len + 1] = '#';
18      tmp[2 * len + 2] = '$';
19      tmp[2 * len + 3] = 0;
20      return 2 * len + 1;
21  }
22
23  int manacher(char *str)
24  {
25      int mx = 0, ans = 0, pos = 0;
26      int len = init(str);
27      for (int i = 1; i <= len; i++)
28      {
29          if (mx > i) Len[i] = min(mx - i, Len[2 * pos - i]);
30          else Len[i] = 1;
31          while (tmp[i - Len[i]] == tmp[i + Len[i]]) Len[i]++;
32          if (Len[i] + i > mx) mx = Len[i] + i, pos = i;
33      }
34  }
```

4.4 Aho-Corasick Automation

4.4.1 AC Automation

```
1   class AC_automation
2   {
3   public:
4       int trie[maxn][26], cnt;
5       int tag[maxn];
6       int fail[maxn], num[maxn], res[maxn], in[maxn], Map[maxn];
7
8       void init()
9       {
10          memset(trie, 0, sizeof trie);
11          memset(tag, 0, sizeof tag);
12          memset(fail, 0, sizeof fail);
13          cnt = 0;
14      }
15
16      void insert(char *str, int id)
17      {
18          int root = 0;
19          for (int i = 0; str[i]; i++)
20          {
21              int id = str[i] - 'a';
22              if (!trie[root][id]) trie[root][id] = ++cnt;
23              root = trie[root][id];
24          }
25          if(!tag[root]) tag[root] = id;
26          Map[id] = tag[root];
27      }
28
29      void build()
30      {
31          queue<int> que;
32          for (int i = 0; i < 26; i++) if (trie[0][i]) que.push(trie[0][i]);
33          while (!que.empty())
```

```
34          {
35              int k = que.front();
36              que.pop();
37              for (int i = 0; i < 26; i++)
38              {
39                  if (trie[k][i])
40                  {
41                      fail[trie[k][i]] = trie[fail[k]][i];
42                      que.push(trie[k][i]);
43                      in[fail[trie[k][i]]] ++;
44                  } else trie[k][i] = trie[fail[k]][i];
45              }
46          }
47      }
48
49      void toposort()
50      {
51          queue<int> que;
52          for(int i = 1; i <= cnt; i ++) if(in[i] == 0) que.push(i);
53          while(!que.empty())
54          {
55              int u = que.front(); que.pop();
56              res[tag[u]] = num[u];
57              int v = fail[u]; in[v] --;
58              num[v] += num[u];
59              if(in[v] == 0) que.push(v);
60          }
61      }
62
63      void query(char *str, int n)
64      {
65          int u = 0, len = strlen(s);
66          for(int i = 0; i < len; i ++)
67              u = trie[u][str[i] - 'a'], num[u] ++;
68          toposort();
69          for(int i = 1; i <= n; i ++) printf("%d\n", res[Map[i]]);
70      }
71  } AC;
```

4.5 Suffix Array

4.5.1 Suffix Array

```
1   char s[maxn];
2   int sa[maxn], t[maxn], t2[maxn], c[maxn], n;
3
4   //build_sa(n + 1, 130), sa, height下标从1开始,rk下标从0开始
5   void build_sa(int n, int m)
6   {
7       int *x = t, *y = t2;
8       for(int i = 0; i < m; i++) c[i] = 0;
9       for(int i = 0; i < n; i++) c[x[i] = s[i]]++;
10      for(int i = 1; i < m; i++) c[i] += c[i - 1];
11      for(int i = n - 1; i >= 0; i--) sa[--c[x[i]]] = i;
12      for(int k = 1; k <= n; k <<= 1)
13      {
14          int p = 0;
15          for(int i = n - k; i < n; i++) y[p++] = i;
16          for(int i = 0; i < n; i++) if(sa[i] >= k) y[p++] = sa[i] - k;
```

```
17          for(int i = 0; i < m; i++) c[i] = 0;
18          for(int i = 0; i < n; i++) c[x[y[i]]]++;
19          for(int i = 0; i < m; i++) c[i] += c[i - 1];
20          for(int i = n - 1; i >= 0; i--) sa[--c[x[y[i]]]] = y[i];
21          swap(x, y);
22          p = 1; x[sa[0]] = 0;
23          for(int i = 1; i < n; i++)
24              x[sa[i]] = y[sa[i - 1]] == y[sa[i]] && y[sa[i - 1] + k] == y[sa[i] + k] ? p
                    - 1 : p++;
25          if(p >= n) break;
26          m = p;
27      }
28  }
29
30  int rk[maxn], height[maxn];
31
32  void getHeight()
33  {
34      for(int i = 1; i <= n; i++) rk[sa[i]] = i;
35      for(int i = 0, k = 0; i < n; i++)
36      {
37          if(k) k--;
38          int j = sa[rk[i] - 1];
39          while(s[i + k] == s[j + k]) k++;
40          height[rk[i]] = k;
41      }
42  }
43
44  int dp[maxn][20];
45
46  void RMQ()
47  {
48      for(int i = 1; i <= n; i ++) dp[i][0] = height[i];
49      for(int j = 1; (1 << j) < maxn; j ++)
50          for(int i = 1; i + (1 << j) - 1 <= n; i ++)
51              dp[i][j] = min(dp[i][j - 1], dp[i + (1 << (j - 1))][j - 1]);
52  }
53
54  int query(int l, int r)
55  {
56      int k = 0;
57      while((1 << (k + 1)) <= r - l + 1) k ++;
58      return min(dp[l][k], dp[r - (1 << k) + 1][k]);
59  }
60
61  int lcp(int x, int y)
62  {
63      x = rk[x], y = rk[y];
64      if(x > y) swap(x, y);
65      return query(x + 1, y);
66  }
```

4.5.2 SA badcw

```
1   namespace SA {
2       const int maxn = 2e5 + 10;
3       int t1[maxn], t2[maxn], c[maxn];
4       int Rank[maxn], height[maxn];
```

```
5       int RMQ[maxn];
6       int mm[maxn];
7       int sa[maxn];
8       int best[25][maxn];
9       bool cmp(int *r, int a, int b, int l) {
10          return r[a] == r[b] && r[a + l] == r[b + l];
11      }
12      void da(char str[], int sa[], int Rank[], int height[], int n, int m) {
13          n++;
14          int i, j, p, *x = t1, *y = t2;
15          for (i = 0; i < m; i++)c[i] = 0;
16          for (i = 0; i < n; i++)c[x[i] = str[i]]++;
17          for (i = 1; i < m; i++)c[i] += c[i - 1];
18          for (i = n - 1; i >= 0; i--)sa[--c[x[i]]] = i;
19          for (j = 1; j <= n; j <<= 1) {
20              p = 0;
21              for (i = n - j; i < n; i++)y[p++] = i;
22              for (i = 0; i < n; i++)if (sa[i] >= j)y[p++] = sa[i] - j;
23              for (i = 0; i < m; i++)c[i] = 0;
24              for (i = 0; i < n; i++)c[x[y[i]]]++;
25              for (i = 1; i < m; i++)c[i] += c[i - 1];
26              for (i = n - 1; i >= 0; i--)sa[--c[x[y[i]]]] = y[i];
27              swap(x, y);
28              p = 1;
29              x[sa[0]] = 0;
30              for (i = 1; i < n; i++)
31                  x[sa[i]] = cmp(y, sa[i - 1], sa[i], j) ? p - 1 : p++;
32              if (p >= n)break;
33              m = p;
34          }
35          int k = 0;
36          n--;
37          for (i = 0; i <= n; i++)Rank[sa[i]] = i;
38          for (i = 0; i < n; i++) {
39              if (k)k--;
40              j = sa[Rank[i] - 1];
41              while (str[i + k] == str[j + k])k++;
42              height[Rank[i]] = k;
43          }
44      }
45      void initRMQ(int n) {
46          for (int i = 1; i <= n; i++)
47              RMQ[i] = height[i];
48          mm[0] = -1;
49          for (int i = 1; i <= n; i++)
50              mm[i] = ((i & (i - 1)) == 0) ? mm[i - 1] + 1 : mm[i - 1];
51          for (int i = 1; i <= n; i++)best[0][i] = i;
52          for (int i = 1; i <= mm[n]; i++)
53              for (int j = 1; j + (1 << i) - 1 <= n; j++) {
54                  int a = best[i - 1][j];
55                  int b = best[i - 1][j + (1 << (i - 1))];
56                  if (RMQ[a] < RMQ[b])best[i][j] = a;
57                  else best[i][j] = b;
58              }
59      }
60      int askRMQ(int a, int b) {
61          int t;
62          t = mm[b - a + 1];
63          b -= (1 << t) - 1;
```

```
64          a = best[t][a];
65          b = best[t][b];
66          return RMQ[a] < RMQ[b] ? a : b;
67      }
68      int lcp(int a, int b) {
69          a = Rank[a];
70          b = Rank[b];
71          if (a > b) swap(a, b);
72          //cout << askRMQ(a + 1, b) << endl;
73          return height[askRMQ(a + 1, b)];
74      }
75      void preprocess(char *str, int n, int m) {
76          da(str, sa, Rank, height, n, m);
77          initRMQ(n);
78      }
79  }
```

4.6 PalindromicTree

4.6.1 PalindromicTree

```
1   const int maxn = 2e6+6;
2   const int N = 26;
3   const int mod = 51123987;
4
5   struct Palindromic_Tree {
6   //    vector<pair<int, int> > next[maxn];
7       int next[maxn][N];//next指针, next指针和字典树类似, 指向的串为当前串两端加上同一个
            字符构成
8       int fail[maxn]{};//fail指针, 失配后跳转到fail指针指向的节点
9       int cnt[maxn]{}; //表示节点i表示的本质不同的串的个数 (建树时求出的不是完全的, 最后
            count()函数跑一遍以后才是正确的)
10      int num[maxn]{}; //表示以节点i表示的最长回文串的最右端点为回文串结尾的回文串个数
11      int len[maxn]{};//len[i]表示节点i表示的回文串的长度 (一个节点表示一个回文串)
12      int S[maxn]{};//存放添加的字符
13      int last{};//指向新添加一个字母后所形成的最长回文串表示的节点。
14      int n{};//表示添加的字符个数。
15      int p{};//表示添加的节点个数。
16      //0向前加, 1向后加字符
17      //int last[2];
18      //int lpos, rpos;
19
20      int newnode(int l) {//新建节点
21  //        next[p].clear();
22          for (int i = 0; i < N; ++i) next[p][i] = 0;
23          cnt[p] = 0;
24          num[p] = 0;
25          len[p] = l;
26          return p++;
27      }
28
29      void init() {//初始化
30          n = last = p = 0;
31          newnode(0);
32          newnode(-1);
33          S[n] = -1;//开头放一个字符集中没有的字符, 减少特判
34          fail[0] = 1;
35          // lpos 为字符串最大长度
36          // last[0] = last[1] = 0;
```

```
37          // lpos = 100000, rpos = lpos - 1;
38          // S[lpos - 1] = S[rpos + 1] = -1;
39      }
40
41      int get_fail(int x) {//和KMP一样，失配后找一个尽量最长的
42          // op 0 向前， 1 向后
43          // if (op == 0) while (S[lpos + len[x] + 1] != S[lpos]) x = fail[x];
44          // else while(S[rpos - len[x] - 1] != S[rpos]) x = fail[x];
45          while (S[n - len[x] - 1] != S[n]) x = fail[x];
46          return x;
47      }
48
49  //    int find(int u, int c) {
50  //        vector<pair<int, int> > & x = next[u];
51  //        int sz = x.size();
52  //        for(int i = 0; i < sz; ++i) {
53  //            if(x[i].first == c) return x[i].second;
54  //        }
55  //        return 0;
56  //    }
57
58      int add(int c) {
59          // 注意清空左右字符
60          // if (op == 0) S[--lpos] = c, S[lpos - 1] = -1;
61          // else S[++rpos] = c, S[rpos + 1] = -1;
62          S[++n] = c;
63          int cur = get_fail(last);//通过上一个回文串找这个回文串的匹配位置
64  //        int x = find(cur, c);
65  //        if (!x) {
66          if (!next[cur][c]) {//如果这个回文串没有出现过，说明出现了一个新的本质不同的回
                文串
67              int now = newnode(len[cur] + 2);//新建节点
68  //            x = now;
69  //            fail[now] = find(get_fail(fail[cur]), c);
70  //            next[cur].emplace_back(make_pair(c, now));
71              fail[now] = next[get_fail(fail[cur])][c];//和AC自动机一样建立fail指针，以便
                    失配后跳转
72              next[cur][c] = now;
73              num[now] = num[fail[now]] + 1;
74          }
75  //        last = x;
76          // 修改最终长度
77          // if (len[last[op]] == rpos - lpos + 1) last[op ^ 1] = last[op];
78          last = next[cur][c];
79          cnt[last]++;
80          return num[last];
81      }
82
83      void count() {
84          for (int i = p - 1; i >= 0; --i) cnt[fail[i]] += cnt[i];
85          //父亲累加儿子的cnt，因为如果fail[v]=u，则u一定是v的子回文串!
86      }
87  } solve;
88
89  char s[maxn];
90
91  // 求相交回文串数量
92  ll a[maxn], b[maxn];
93  int main() {
```

```
94      solve.init();
95      int n;
96      scanf("%d", &n);
97      scanf("%s", s);
98      for (int i = 0; i < n; ++i) a[i] = solve.add(s[i] - 'a');
99      solve.init();
100     for (int i = n - 1; i >= 0; --i) b[i] = (b[i + 1] + solve.add(s[i] - 'a')) % mod;
101     ll res = (b[0] * (b[0] - 1) / 2) % mod;
102     for (int i = 0; i < n; ++i) res = ((res - (a[i] * b[i + 1]) + mod) % mod) % mod;
103     printf("%lld\n", res);
104     return 0;
105 }
```

4.7 Hash

4.7.1 hash

```
1  // hash常用素数
2  // 61, 83, 113, 151, 211
3  // 91815541, 38734667, 68861641
4  // 917120411, 687840301, 386910137, 515880193
5  // 1222827239, 1610612741
6
7  typedef unsigned long long ull;
8  struct mhash {
9      // 自然溢出无模数 805306457
10     ull base[maxn];
11     ull hash_index[maxn];
12     ull seed; //31, 131
13     void inithash(ull seedt = 31) {
14         base[0] = 1;
15         seed = seedt;
16         for (int i = 1; i < maxn; ++i) base[i] = base[i - 1] * seed;
17     }
18     void H(char *p, int n) { // from 1 to n
19         hash_index[0] = 0;
20         for (int i = 1; i <= n; ++i) hash_index[i] = hash_index[i - 1] * seed + p[i] -
               'a';
21     }
22     ull gethash(int s, int e) {
23         return hash_index[e] - hash_index[s - 1] * base[e - s + 1];
24     }
25 };
26
27 // 26个素数, 解决加法hash
28 int prime[] = {34183,13513,152993,13591,19687,350869,111187,766091,769297,
29                633469,752273,298651,617191,880421,136067,
30                1408397,726899,458921,2133701,2599847,2730947,4696343,10267237,
31                18941059,34078909,69208409};
```

4.7.2 doubleHash

```
1  namespace Hash{
2
3      template<class __A,class __B>
4      class Hash{
5      private:
6          static const int size=2000000;
```

```
7           __B *hash; __A *O; int sz;
8       public:
9           Hash(int hash_size=size){ sz=hash_size;
10              hash=(__B *)malloc(sizeof(__B)*sz);
11              O=(__A *)malloc(sizeof(__A)*sz);
12              memset(O,0xff,sizeof(__A)*sz);
13          }~Hash(){free(O);free(hash);}
14          __B &operator [](const __A &_O){
15              int loc=_O%sz;
16              while(~O[loc]&&O[loc]!=_O){
17                  ++loc;
18                  if(loc>sz)loc=0;
19              }if(!~O[loc])O[loc]=_O;
20              return hash[loc];
21          }
22          void clear(){memset(O,0xff,sizeof(__A)*sz);}
23      };
24
25      struct StringDoubleHashResult{
26          int32_t *H1,*H2,c_len,len;
27          StringDoubleHashResult(int32_t sz=0){
28              len=sz; c_len=0; //cur_len;
29              if(len<=0){
30                  H1=H2=0;
31                  return;
32              }
33              H1=(int32_t *)malloc(sizeof(int32_t)*sz);
34              H2=(int32_t *)malloc(sizeof(int32_t)*sz);
35          }
36          ~StringDoubleHashResult(){}
37          void clear(){free(H1);free(H2);len=0;H1=H2=0;}
38          void resize(int new_len){
39              int32_t *T1=(int32_t *)malloc(sizeof(int32_t)*new_len);
40              int32_t *T2=(int32_t *)malloc(sizeof(int32_t)*new_len);
41              for(int i=0;i<c_len;++i)T1[i]=H1[i],T2[i]=H2[i];
42              free(H1);free(H2); H1=T1; H2=T2; len=new_len;
43          }
44          void erase(int ers_len){//erase suffix
45              c_len-=ers_len;
46              if(c_len<0)c_len=0;
47          }
48          //erase prefix not better than reculc
49      };
50
51      namespace hash_random{
52          const int mod_tot=5;
53          const int mod[]={1000000009,1000000007,998244353,917120411,515880193};
54      };
55
56      class StringDoubleHash{
57      private:
58          static const int enable_random=1;
59          int32_t sz,HA1,HA2;
60          long long B,C;
61          int32_t *H1,*H2;
62      public:
63          StringDoubleHash(int32_t SZ=2e6+5,int32_t ha1=-1,int32_t ha2=-1,int32_t b=-1,
                int32_t c=-1){
64              sz=SZ;
```

```
65              if(enable_random){
66                  std::mt19937 rnd(time(0)+19990630);
67                  int z1= rnd() % hash_random::mod_tot;
68                  int z2= (z1 +rnd()%(hash_random::mod_tot - 1) + 1) % hash_random::
                        mod_tot;
69                  if(ha1<0)ha1=hash_random::mod[z1];
70                  if(ha2<0)ha2=hash_random::mod[z2];
71                  if(b<0)b=rnd()%114514+23333;
72                  if(c<0)c=rnd()%1919810+23333;
73              } else {
74                  if(ha1<0)ha1=1e9+7;
75                  if(ha2<0)ha2=1e9+9;
76                  if(b<0)b=114514;
77                  if(c<0)c=1919810;
78              }
79              HA1=ha1; HA2=ha2; B=b; C=c;
80              //cerr<<HA1<<" "<<HA2<<" "<<B<<" "<<C<<endl;
81              H1=(int32_t *)malloc(sizeof(int32_t)*sz);
82              H2=(int32_t *)malloc(sizeof(int32_t)*sz);
83              init_hash_val();
84          }
85          ~StringDoubleHash(){free(H1);free(H2);}
86          void init_hash_val(){
87              H1[0]=H2[0]=1;
88              for(int32_t i=1;i<sz;++i){
89                  H1[i]=(H1[i-1]*B)%HA1;
90                  H2[i]=(H2[i-1]*B)%HA2;
91              }
92          }
93          template <class _Tp>
94          StringDoubleHashResult culc_hash(const _Tp &s,int32_t len,int32_t tot_len=-1){
95              if(tot_len<0)tot_len=len;
96              StringDoubleHashResult R(tot_len);
97              if(len<=0)return R;
98              R.H1[0]=(s[0]+C)%HA1;
99              R.H2[0]=(s[0]+C)%HA2;
100             for(int32_t i=1;i<len;++i){
101                 R.H1[i]=(R.H1[i-1]*B+s[i]+C)%HA1;
102                 R.H2[i]=(R.H2[i-1]*B+s[i]+C)%HA2;
103             }
104             R.c_len=len;
105             return R;
106         }
107         // s is the char* first, len is the append length
108         template <class _Tp>
109         void append(StringDoubleHashResult &R,const _Tp &s,int32_t len){
110             if(len<=0)return;
111             int t_len=R.len;
112             while(R.c_len+len>t_len)t_len<<=1;
113             if(t_len>R.len)R.resize(t_len);
114             for(int32_t i=R.c_len;i<R.c_len+len;++i){
115                 if(i==0){
116                     R.H1[i]=(s[i-R.c_len]+C)%HA1;
117                     R.H2[i]=(s[i-R.c_len]+C)%HA2;
118                 } else {
119                     R.H1[i]=(R.H1[i-1]*B+s[i-R.c_len]+C)%HA1;
120                     R.H2[i]=(R.H2[i-1]*B+s[i-R.c_len]+C)%HA2;
121                 }
122             }
```

```
123             R.c_len+=len;
124         }
125         void append(StringDoubleHashResult &R, char s){
126             int t_len=R.len;
127             while(R.c_len+1>t_len)t_len<<=1;
128             if(t_len>R.len)R.resize(t_len);
129             for(int32_t i=R.c_len;i<R.c_len+1;++i){
130                 if(i==0){
131                     R.H1[i]=(s+C)%HA1;
132                     R.H2[i]=(s+C)%HA2;
133                 } else {
134                     R.H1[i]=(R.H1[i-1]*B+s+C)%HA1;
135                     R.H2[i]=(R.H2[i-1]*B+s+C)%HA2;
136                 }
137             }
138             R.c_len+=1;
139         }
140         //return hash [l,r)
141         ll gethash(const StringDoubleHashResult &R, int32_t l,int32_t r){
142             if(l>r||l<0||r-->R.c_len)return -1;//fail
143             ll v1=l>0?R.H1[l-1]*(long long)H1[r-l+1]%HA1:0;
144             ll v2=l>0?R.H2[l-1]*(long long)H2[r-l+1]%HA2:0;
145             v1=R.H1[r]-v1; v2=R.H2[r]-v2;
146             if(v1<0)v1+=HA1; if(v2<0)v2+=HA2;
147             return v1<<32|v2;
148         }
149         //merge two hashes as one(s1+s2), but need s2's length
150         ll merge_hash(const long long &hs1,const long long &hs2,int lenr){
151             int32_t m1=hs1>>32,m2=hs1&0xffffffffLL;
152             int32_t m3=hs2>>32,m4=hs2&0xffffffffLL;
153             m1=m1*(long long)H1[lenr]%HA1+m3;
154             if(m1>=HA1)m1-=HA1;
155             m2=m2*(long long)H2[lenr]%HA2+m4;
156             if(m2>=HA2)m2-=HA2;
157             return (long long)m1<<32|m2;
158         }
159     };
160 };
```

4.7.3 二维 hash

```
1  #define ull unsigned long long
2  const int maxn = 1005;
3  ull hs[maxn][maxn];
4  char a[maxn][maxn];
5  int n, m;
6  ull base1 = 131, base2 = 13331;
7  ull pwb1[maxn] = {1}, pwb2[maxn] = {1};
8
9  void init() {
10     for (int i = 1; i < maxn; ++i) {
11         pwb1[i] = pwb1[i - 1] * base1;
12         pwb2[i] = pwb2[i - 1] * base2;
13     }
14 }
15
16 void Hash() {
17     for(int i=1;i<=n;i++)
```

```
        for(int j=1;j<=m;j++)
            hs[i][j]=hs[i][j-1]*base1+a[i][j] - 'a';
    for(int i=1;i<=n;i++)
        for(int j=1;j<=m;j++)
            hs[i][j]+=hs[i-1][j]*base2;
}

// 右下角(i,j), 行列长度n,m
ull getHs(int i, int j, int lenn, int lenm) {
    return hs[i][j] - hs[i - lenn][j] * pwb2[lenn] -
            hs[i][j - lenm] * pwb1[lenm] +
            hs[i - lenn][j - lenm] * pwb2[lenn] * pwb1[lenm];
}
```

4.7.4 树 hash 同构

```
// n=1e5的话base开2e6+9, 可以输出看到top不比n小即可
const int base = 2e6+9;
// vis大小要开到素数大小, turn表示当前树的编号, p是预处理数组
int vis[base + 1], top, turn, p[base + 1];
// 程序开头调用一次
void init() {
    top = 0;
    for (int i = 2; i <= base; ++i) {
        if (!vis[i]) {
            p[++top] = i;
        }
        for (int j = 1; j <= top && i * p[j] <= base; ++j) {
            vis[i * p[j]] = 1;
            if (i % p[j] == 0) break;
        }
    }
    assert(top >= maxn);
}

vector<int> edge[maxn];
// h[x]表示x这棵子树的hash值, g[x]表示以x为根的hash值
int h[maxn], g[maxn], sz[maxn];

struct TreeHash {
    int n;
    // 如果树比较多, 在类内部开edge可能会炸内存, 可以改到外面做前向星
    // 除了hs是答案其他都可以改到外部, 只有edge需要清零
    // vector<int> edge[maxn];
    // int h[maxn], g[maxn], sz[maxn];
    vector<int> hs;

    void init(int n_ = 0) {
        n = n_;
        hs.clear();
    }

    void dfs1(int u, int pre) {
        sz[u] = 1;
        h[u] = 1;
        for (auto v : edge[u]) {
            if (v == pre) continue;
            dfs1(v, u);
```

```
43              h[u] = (h[u] + 1ll * h[v] * p[sz[v]] % mod) % mod;
44              sz[u] += sz[v];
45          }
46      }
47
48      void dfs2(int u, int pre, int V, int needres = 1) {
49          g[u] = (h[u] + 1ll * V * p[n - sz[u]] % mod) % mod;
50          if (needres) hs.push_back(g[u]);
51          for (auto v : edge[u]) {
52              if (v == pre) continue;
53              dfs2(v, u, (g[u] - 1ll * h[v] * p[sz[v]] % mod + mod) % mod);
54          }
55      }
56
57      void work(int needres = 1) {
58          // 无根树选一个不存在的点当pre即可, 当多棵无根树判重时需要sort
59          dfs1(1, 0);
60          dfs2(1, 0, 0, needres);
61          sort(hs.begin(), hs.end());
62      }
63  };
64
65  // 获取删掉某叶子节点后以与该叶子节点相邻点开头的hash值
66  // int res = (hs[edge[i][0]] - 2 + mod) % mod;
```

4.8 Suffix Automation

4.8.1 SAM

```
1   const int maxn = 2e4 + 10;
2
3   struct SuffixAutomation
4   {
5       int last, cnt;
6       int ch[maxn << 1][26], fa[maxn << 1], len[maxn << 1], pos[maxn << 1];
7       int sz[maxn << 1], a[maxn << 1], c[maxn << 1];
8
9       void init()
10      {
11          last = cnt = 1;
12          memset(ch[1], 0, sizeof ch[1]);
13          fa[1] = len[1] = 0;
14          a[1] = c[1] = 0;
15      }
16
17      int inline newnode(int idx)
18      {
19          ++cnt;
20          memset(ch[cnt], 0, sizeof ch[cnt]);
21          fa[cnt] = len[cnt] = sz[cnt] = a[cnt] = c[cnt] = 0;
22          pos[cnt] = idx;
23          return cnt;
24      }
25
26      void ins(int c)
27      {
28          int p = last , np = newnode(pos[last] + 1);
29          last = np, len[np] = len[p] + 1;
30          for(; p && !ch[p][c]; p = fa[p]) ch[p][c] = np;
```

```
31          if(!p) fa[np] = 1;
32          else
33          {
34              int q = ch[p][c];
35              if(len[p] + 1 == len[q]) fa[np] = q;
36              else
37              {
38                  int nq = newnode(pos[p] + 1);
39                  len[nq] = len[p] + 1;
40                  memcpy(ch[nq], ch[q], sizeof ch[q]);
41                  fa[nq] = fa[q], fa[q] = fa[np] = nq;
42                  for(; ch[p][c] == q; p = fa[p]) ch[p][c] = nq;
43              }
44          }
45          sz[np] = 1;
46      }
47
48      int solve(int n)
49      {
50          /*求两个串的LCS:
51              对一个字符串建立SAM, 记录一个当前匹配的长度Len和当前节点v, 枚举另一个字符串
                    的每个字符;
52              如果p有字符v的转移边出边, 则使Len加一, 并使p转移到出边指向的节点上;
53              否则不断向父节点上跳, 直到当前节点有字符p的转移出边, 或者跳到根节点;
54          */
55          int p = 1, ans = 0, now_len = 0;
56          for(int i = 0; s2[i]; i ++)
57          {
58              if(ch[p][s2[i] - 'a']) p = ch[p][s2[i] - 'a'], now_len ++;
59              else
60              {
61                  for(;p && !ch[p][s2[i] -'a'] ; p = fa[p]) ;
62                  if(p == 0) now_len = 0, p = 1;
63                  else now_len = len[p] + 1, p = ch[p][s2[i] - 'a'];
64              }
65              ans = max(now_len, ans);
66          }
67      }
68
69      void Toposort()
70      {
71          long long ans = 0;
72          for(int i = 1; i <= cnt; i ++) c[len[i]] ++;
73          for(int i = 1; i <= cnt; i ++) c[i] += c[i - 1];
74          for(int i = 1; i <= cnt; i ++) a[c[len[i]] --] = i;
75          for(int i = cnt; i; i --) sz[fa[a[i]]] += sz[a[i]];
76      }
77  }sam;
```

4.9 Others

4.9.1 最小表示法

```
1  // 0起始
2  int Gao(char a[], int len) {
3      int i = 0, j = 1, k = 0;
4      while (i < len && j < len && k < len) {
5          int cmp = a[(j + k) % len] - a[(i + k) % len];
6          if (cmp == 0) k++;
```

```
7           else {
8               if (cmp > 0) j += k + 1;
9               else i += k + 1;
10              if (i == j) j ++;
11              k = 0;
12          }
13      }
14      return min(i, j);
15  }
```

4.9.2 Lyndon

```
1   void duval(char s[])
2   {
3       int n = strlen(s), i = 0;
4       vector<int> res; //Lyndon串的开头
5       while(i < n)
6       {
7           int j = i, k = i + 1;
8           while(j < n && s[j] <= s[k])
9           {
10              if(s[j] < s[k]) j = i;
11              else j ++;
12              k ++;
13          }
14          while(i <= j)
15          {
16              res.push_back(i);
17              i += k - j;
18          }
19      }
20  }
```

5 dp

5.1 BitDP

5.1.1 数位 dp 计和

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;
const int mod = 998244353;
pair<ll, ll> dp[20][1<<10];
bool vis[20][1<<10];
int k;
int t[20];
ll base[20];

pair<ll, ll> dfs(int pos, int state, bool limit, bool lead) {
    if (pos == -1) return __builtin_popcount(state) <= k ? make_pair(1, 0) : make_pair
        (0, 0);
    if (!limit && !lead && vis[pos][state]) return dp[pos][state];
    int up = limit ? t[pos] : 9;
    pair<ll, ll> res = {0, 0};
    for (int i = 0; i <= up; ++i) {
        int n_s = state;
        if (lead && i == 0) n_s = 0;
        else n_s = state | (1 << i);
        auto tmp = dfs(pos - 1, n_s, limit && i == t[pos], lead && i == 0);
        ll pre = 1ll * i * base[pos] % mod;
        (res.first += tmp.first) %= mod;
        (res.second += tmp.second + pre * tmp.first) %= mod;
    }
    if (!limit && !lead) dp[pos][state] = res, vis[pos][state] = 1;
    return res;
}

ll solve(ll x) {
    int pos = 0;
    do {
        t[pos ++] = x % 10;
    } while (x /= 10);
    return dfs(pos - 1, 0, true, true).second;
}

int main(int argc,char *argv[])
{
    base[0] = 1;
    for (int i = 1; i < 20; ++i) base[i] = base[i - 1] * 10;
    ll l, r;
    scanf("%lld%lld%d", &l, &r, &k);
    printf("%lld\n", (solve(r) - solve(l - 1) + mod) % mod);
    return 0;
}
```

5.1.2 两个数数位 dp

```
// 二进制数位dp, 求a $\in$ 1~x 和 b $\in$ 1~y, 满足 $a & b > c || a ^ b < c$的对数
ll dp[maxn][2][2][2][2];
int a[maxn], b[maxn], c[maxn];

```

```
5
6  void cal(int *xt, ll x) {
7      int has = 0;
8      while (x) {
9          xt[has++] = x % 2;
10         x /= 2;
11     }
12 }
13
14 ll dfs(int pos, int o1, int o2, int lim1, int lim2) {
15     if (pos < 0) return 1;
16     ll &t = dp[pos][o1][o2][lim1][lim2];
17     if (t != -1) return t;
18     int up1 = o1 ? a[pos] : 1;
19     int up2 = o2 ? b[pos] : 1;
20     ll res = 0;
21     for (int i = 0; i <= up1; ++i) {
22         for (int j = 0; j <= up2; ++j) {
23             int t1 = i & j;
24             int t2 = i ^ j;
25             if (lim1 && t1 > c[pos]) continue;
26             if (lim2 && t2 < c[pos]) continue;
27             res += dfs(pos - 1, o1 && i == up1, o2 && j == up2, lim1 && t1 == c[pos],
                   lim2 && t2 == c[pos]);
28         }
29     }
30     return t = res;
31 }
32
33 ll solve(ll x, ll y, ll z) {
34     memset(dp, -1ll, sizeof dp);
35     for (int i = 0; i < 33; ++i) a[i] = b[i] = c[i] = 0;
36     cal(a, x);
37     cal(b, y);
38     cal(c, z);
39     return dfs(32, 1, 1, 1, 1);
40 }
41
42 int main(int argc, char *argv[]) {
43     int T;
44     scanf("%d", &T);
45     ll x, y, z;
46     for (int kase = 1; kase <= T; ++kase) {
47         scanf("%lld%lld%lld", &x, &y, &z);
48         ll res = solve(x, y, z);
49         res -= max(0ll, y - z + 1);
50         res -= max(0ll, x - z + 1);
51         printf("%lld\n", x * y - res);
52     }
53     return 0;
54 }
```

5.2 Subsequence

5.2.1 MaxSum

```
1  // 传入序列a和长度n, 返回最大子序列和
2  int MaxSeqSum(int a[], int n)
3  {
```

```
    int rt = 0, cur = 0;
    for (int i = 0; i < n; i++)
        cur += a[i], rt = max(cur, rt), cur = max(0, cur);
    return rt;
}
```

5.2.2 LIS

```
// 简单写法(下标从0开始,只返回长度)
int dp[N];
int LIS(int a[], int n)
{
    memset(dp, 0x3f, sizeof(dp));
    for (int i = 0; i < n; i++) *lower_bound(dp, dp + n, a[i]) = a[i];
    return lower_bound(dp, dp + n, INF) - dp;
}

// 小常数nlogn求序列用树状数组维护dp即可
// dp[i] = max(dp[j]) + 1 (j < i && a[j] < a[i])
```

5.2.3 LongestCommonIncrease

```
// 序列下标从1开始
int LCIS(int a[], int b[], int n, int m)
{
    memset(dp, 0, sizeof(dp));
    for (int i = 1; i <= n; i++)
    {
        int ma = 0;
        for (int j = 1; j <= m; j++)
        {
            dp[i][j] = dp[i - 1][j];
            if (a[i] > b[j]) ma = max(ma, dp[i - 1][j]);
            if (a[i] == b[j]) dp[i][j] = ma + 1;
        }
    }
    return *max_element(dp[n] + 1, dp[n] + 1 + m);
}
```

5.2.4 LCS

```
#include <stdio.h>
#include <string.h>

#define M 30005
#define SIZE 128
#define WORDMAX 3200
#define BIT 32

char s1[M], s2[M];
int nword;
unsigned int str[SIZE][WORDMAX];
unsigned int tmp1[WORDMAX], tmp2[WORDMAX];

void pre(int len)
{
    int i, j;
```

```
17      memset(str, 0, sizeof(str));
18      for(i = 0; i < len; i ++)
19          str[s1[i]][i / BIT] |= 1 << (i % BIT);
20  }
21
22  void cal(unsigned int *a, unsigned int *b, char ch)
23  {
24      int i, bottom = 1, top;
25      unsigned int x, y;
26      for(i = 0; i < nword; i ++)
27      {
28          y = a[i];
29          x = y | str[ch][i];
30          top = (y >> (BIT - 1)) & 1;
31          y = (y << 1) | bottom;
32          if(x < y) top = 1;
33          b[i] = x & ((x - y) ^ x);
34          bottom = top;
35      }
36  }
37
38  int bitcnt(unsigned int *a)
39  {
40      int i, j, res = 0, t;
41      unsigned int b[5] = {0x55555555, 0x33333333, 0x0f0f0f0f, 0x00ff00ff, 0x0000ffff}, x
            ;
42      for(i = 0; i < nword; i ++)
43      {
44          x = a[i];
45          t = 1;
46          for(j = 0; j < 5; j ++, t <<= 1)
47              x = (x & b[j]) + ((x >> t) & b[j]);
48          res += x;
49      }
50      return res;
51  }
52
53  void process()
54  {
55      int i, j, len1, len2;
56      unsigned int *a, *b, *t;
57      len1 = strlen(s1);
58      len2 = strlen(s2);
59      nword = (len1 + BIT - 1) / BIT;
60      pre(len1);
61      memset(tmp1, 0, sizeof(tmp1));
62      a = &tmp1[0];
63      b = &tmp2[0];
64      for(i = 0; i < len2; i ++)
65      {
66          cal(a, b, s2[i]);
67          t = a; a = b; b = t;
68      }
69      printf("%d\n", bitcnt(a));
70  }
71
72  int main()
73  {
74      while(scanf("%s%s", s1, s2) != EOF)
```

```
75          process();
76      return 0;
77  }
```

5.3 Others

问题 设 $f(i)=\min(y[k]-s[i]\times x[k]),k\in[1,i-1]$, 现在要求出所有 $f(i),i\in[1,n]$

考虑两个决策 j 和 k，如果 j 比 k 优，则

$$y[j]-s[i]\times x[j]<y[k]-s[i]\times x[k]$$

化简得:

$$\frac{y_j-y_k}{x_j-x_k}<s_i$$

不等式左边是个斜率，我们把它设为 $\text{slope}(j,k)$

我们可以维护一个单调递增的队列，为什么呢?

因为如果 $\text{slope}(q[i-1],q[i])>\text{slope}(q[i],q[i+1])$，那么当前者成立时，后者必定成立。即 $q[i]$ 决策优于 $q[i-1]$ 决策时，$q[i+1]$ 必然优于 $q[i]$，因此 $q[i]$ 就没有存在的必要了。所以我们要维护递增的队列。

那么每次的决策点 i，都要满足

$$\begin{cases}\text{slope}(q[i-1],q[i])<s[i]\\ \text{slope}(q[i],q[i+1])\geq s[i]\end{cases}$$

一般情况去二分这个 i 即可。

如果 $s[i]$ 是单调不降的，那么对于决策 j 和 $k(j<k)$ 来说，如果决策 k 优于决策 j，那么对于 $i\in[k+1,n]$，都存在决策 k 优于决策 j，因此决策 j 就可以舍弃了。这样的话我们可以用单调队列进行优化，可以少个 log。

单调队列滑动窗口最大值

```
1  // k为滑动窗口的大小，数列下标从1开始，d为序列长度+1
2  deque<int> q;
3  for (int i = 0, j = 0; i + k <= d; i++)
4  {
5      while (j < i + k)
6      {
7          while (!q.empty() && a[q.back()] < a[j]) q.pop_back();
8          q.push_back(j++);
9      }
10     while (q.front() < i) q.pop_front();
11     // a[q.front()]为当前滑动窗口的最大值
12 }
```

5.3.1 矩阵快速幂

```
1  const int sz = 5;
2  struct Matrix {
3      ll a[sz][sz];
4      Matrix() { memset(a, 0, sizeof a); }
5      void pr() {
6          printf("*\n");
7          for (int i = 0; i < sz; ++i) {
8              for (int j = 0; j < sz; ++j) {
9                  printf("%lld ", a[i][j]);
10             }
11             printf("\n");
12         }
13     }
14     void tr() {
15         for (int i = 0; i < sz; ++i) {
```

```
            for (int j = i + 1; j < sz; ++j) {
                swap(a[i][j], a[j][i]);
            }
        }
    }
} res, t1;

void init() {
    ;
}

Matrix mul(Matrix a, Matrix b) {
    Matrix res;
    // assert(a.m == b.n)
    for (int i = 0; i < sz; i++) // a.n
        for (int j = 0; j < sz; j++) // b.m
            for (int k = 0; k < sz; k++) // a.m, b.n
                (res.a[i][j] += a.a[i][k] * b.a[k][j] % mod) %= mod;
    return res;
}

Matrix Pow(ll n) {
    init();
    //for(int i = 0; i < cur; i++) res.a[i][i] = 1;
    while (n > 0) {
        if (n & 1) res = mul(res, t1);
        t1 = mul(t1, t1);
        n >>= 1;
    }
    return res;
}
```

5.3.2 单调栈

```
// 求左边第一个比a[i]小的和右边最后一个不比a[i]小的位置
for (int i = 1; i <= n; i++) {
    while (top && a[sta[top - 1]] >= a[i]) top--;
    la[i] = (top == 0) ? 1 : sta[top - 1] + 1;
    sta[top++] = i;
}
top = 0;
for (int i = n; i >= 1; i--) {
    while (top && a[sta[top - 1]] >= a[i]) top--;
    ra[i] = (top == 0) ? n : sta[top - 1] - 1;
    sta[top++] = i;
}
```

5.3.3 单调队列

```
// 循环序列的最大子段和
int a[maxn];
int pre[maxn * 2];
int qu[maxn * 2];
int n, resl, resr, res, k;

int main(int argc, char* argv[]) {
    int T;
    scanf("%d", &T);
```

```
    for (int kase = 1; kase <= T; ++kase) {
        scanf("%d%d", &n, &k);
        for (int i = 1; i <= n; ++i) {
            scanf("%d", &a[i]);
            pre[i] = pre[i - 1] + a[i];
        }
        for (int i = n + 1; i <= 2 * n; ++i) {
            pre[i] = pre[i - 1] + a[i - n];
        }
        res = -0x3f3f3f3f;
        resl = resr = -1;
        int l = 1, r = 0;
        for (int i = 1; i <= 2 * n; ++i) {
            while (l <= r && pre[qu[r]] >= pre[i - 1]) r--;
            qu[++r] = i - 1;
            while (l <= r && qu[l] < i - k) l++;
            int tmp = pre[i] - pre[qu[l]];
            if (tmp > res) {
                res = tmp;
                resl = qu[l] + 1;
                resr = i;
            }
        }
        if (resl > n) resl -= n;
        if (resr > n) resr -= n;
        printf("%d %d %d\n", res, resl, resr);
    }
    return 0;
}
```

6 Geometry

6.1 geo

```
#define mp make_pair
#define fi first
#define se second
#define pb push_back
typedef double db;
const db eps=1e-6;
const db pi=acos(-1);
int sign(db k){
    if (k>eps) return 1; else if (k<-eps) return -1; return 0;
}
int cmp(db k1,db k2){return sign(k1-k2);}
int inmid(db k1,db k2,db k3){return sign(k1-k3)*sign(k2-k3)<=0;}// k3 在 [k1,k2] 内
struct point{
    db x,y;
    point operator + (const point &k1) const{return (point){k1.x+x,k1.y+y};}
    point operator - (const point &k1) const{return (point){x-k1.x,y-k1.y};}
    point operator * (db k1) const{return (point){x*k1,y*k1};}
    point operator / (db k1) const{return (point){x/k1,y/k1};}
    int operator == (const point &k1) const{return cmp(x,k1.x)==0&&cmp(y,k1.y)==0;}
    // 逆时针旋转
    point turn(db k1){return (point){x*cos(k1)-y*sin(k1),x*sin(k1)+y*cos(k1)};}
    point turn90(){return (point){-y,x};}
    bool operator < (const point k1) const{
        int a=cmp(x,k1.x);
        if (a==-1) return 1; else if (a==1) return 0; else return cmp(y,k1.y)==-1;
    }
    db abs(){return sqrt(x*x+y*y);}
    db abs2(){return x*x+y*y;}
    db dis(point k1){return ((*this)-k1).abs();}
    point unit(){db w=abs(); return (point){x/w,y/w};}
    void scan(){double k1,k2; scanf("%lf%lf",&k1,&k2); x=k1; y=k2;}
    void print(){printf("%.11lf %.11lf\n",x,y);}
    db getw(){return atan2(y,x);}
    point getdel(){if (sign(x)==-1||(sign(x)==0&&sign(y)==-1)) return (*this)*(-1);
        else return (*this);}
    int getP() const{return sign(y)==1||(sign(y)==0&&sign(x)==-1);}
};
int inmid(point k1,point k2,point k3){return inmid(k1.x,k2.x,k3.x)&&inmid(k1.y,k2.y,k3.
    y);}
db cross(point k1,point k2){return k1.x*k2.y-k1.y*k2.x;}
db dot(point k1,point k2){return k1.x*k2.x+k1.y*k2.y;}
db rad(point k1,point k2){return atan2(cross(k1,k2),dot(k1,k2));}
// -pi -> pi
int compareangle (point k1,point k2){
    return k1.getP()<k2.getP()||(k1.getP()==k2.getP()&&sign(cross(k1,k2))>0);
}
point proj(point k1,point k2,point q){ // q 到直线 k1,k2 的投影
    point k=k2-k1; return k1+k*(dot(q-k1,k)/k.abs2());
}
point reflect(point k1,point k2,point q){return proj(k1,k2,q)*2-q;}
int clockwise(point k1,point k2,point k3){// k1 k2 k3 逆时针 1 顺时针 -1 否则 0
    return sign(cross(k2-k1,k3-k1));
}
int checkLL(point k1,point k2,point k3,point k4){// 求直线 (L) 线段 (S)k1,k2 和 k3,k4
    的交点
```

```
53      return cmp(cross(k3-k1,k4-k1),cross(k3-k2,k4-k2))!=0;
54  }
55  point getLL(point k1,point k2,point k3,point k4){
56      db w1=cross(k1-k3,k4-k3),w2=cross(k4-k3,k2-k3); return (k1*w2+k2*w1)/(w1+w2);
57  }
58  int intersect(db l1,db r1,db l2,db r2){
59      if (l1>r1) swap(l1,r1); if (l2>r2) swap(l2,r2); return cmp(r1,l2)!=-1&&cmp(r2,l1)
            !=-1;
60  }
61  int checkSS(point k1,point k2,point k3,point k4){
62      return intersect(k1.x,k2.x,k3.x,k4.x)&&intersect(k1.y,k2.y,k3.y,k4.y)&&
63      sign(cross(k3-k1,k4-k1))*sign(cross(k3-k2,k4-k2))<=0&&
64      sign(cross(k1-k3,k2-k3))*sign(cross(k1-k4,k2-k4))<=0;
65  }
66  db disSP(point k1,point k2,point q){
67      point k3=proj(k1,k2,q);
68      if (inmid(k1,k2,k3)) return q.dis(k3); else return min(q.dis(k1),q.dis(k2));
69  }
70  db disSS(point k1,point k2,point k3,point k4){
71      if (checkSS(k1,k2,k3,k4)) return 0;
72      else return min(min(disSP(k1,k2,k3),disSP(k1,k2,k4)),min(disSP(k3,k4,k1),disSP(k3,
            k4,k2)));
73  }
74  int onS(point k1,point k2,point q){return inmid(k1,k2,q)&&sign(cross(k1-q,k2-k1))==0;}
75  struct circle{
76      point o; db r;
77      void scan(){o.scan(); scanf("%lf",&r);}
78      int inside(point k){return cmp(r,o.dis(k));}
79  };
80  struct line{
81      // p[0]->p[1]
82      point p[2];
83      line(point k1,point k2){p[0]=k1; p[1]=k2;}
84      point& operator [] (int k){return p[k];}
85      int include(point k){return sign(cross(p[1]-p[0],k-p[0]))>0;}
86      point dir(){return p[1]-p[0];}
87      line push(){ // 向外 ( 左手边 ) 平移 eps
88          const db eps = 1e-6;
89          point delta=(p[1]-p[0]).turn90().unit()*eps;
90          return {p[0]-delta,p[1]-delta};
91      }
92  };
93  point getLL(line k1,line k2){return getLL(k1[0],k1[1],k2[0],k2[1]);}
94  int parallel(line k1,line k2){return sign(cross(k1.dir(),k2.dir()))==0;}
95  int sameDir(line k1,line k2){return parallel(k1,k2)&&sign(dot(k1.dir(),k2.dir()))==1;}
96  int operator < (line k1,line k2){
97      if (sameDir(k1,k2)) return k2.include(k1[0]);
98      return compareangle(k1.dir(),k2.dir());
99  }
100 int checkpos(line k1,line k2,line k3){return k3.include(getLL(k1,k2));}
101 vector<point> halfPlaneIS(vector<line> &l) {// 求半平面交 , 半平面是逆时针方向 , 输出按
        照逆时针, 点数小于等于2表示无法构成半平面交
102     sort(l.begin(), l.end());
103     deque<line> q;
104     for (int i = 0; i < (int)l.size(); ++i) {
105         if (i && sameDir(l[i], l[i - 1])) continue;
106         while (q.size() > 1 && !checkpos(q[q.size() - 2], q[q.size() - 1], l[i])) q.
                pop_back();
107         while (q.size() > 1 && !checkpos(q[1], q[0], l[i])) q.pop_front();
```

```
108         q.push_back(l[i]);
109     }
110     while (q.size() > 2 && !checkpos(q[q.size() - 2], q[q.size() - 1], q[0])) q.
            pop_back();
111     while (q.size() > 2 && !checkpos(q[1], q[0], q[q.size() - 1])) q.pop_front();
112     vector<point> ret;
113     for (int i = 0; i < (int)q.size(); ++i) ret.push_back(getLL(q[i], q[(i + 1) % q.
            size()]));
114     return ret;
115 }
116 // vector<line> getHL(vector<line> &L){ // 求半平面交 , 半平面是逆时针方向 , 输出按照逆
        时针
117 //     sort(L.begin(),L.end()); deque<line> q;
118 //     for (int i=0;i<(int)L.size();i++){
119 //         if (i&&sameDir(L[i],L[i-1])) continue;
120 //         while (q.size()>1&&!checkpos(q[q.size()-2],q[q.size()-1],L[i])) q.pop_back()
        ;
121 //         while (q.size()>1&&!checkpos(q[1],q[0],L[i])) q.pop_front();
122 //         q.push_back(L[i]);
123 //     }
124 //     while (q.size()>2&&!checkpos(q[q.size()-2],q[q.size()-1],q[0])) q.pop_back();
125 //     while (q.size()>2&&!checkpos(q[1],q[0],q[q.size()-1])) q.pop_front();
126 //     vector<line>ans; for (int i=0;i<q.size();i++) ans.push_back(q[i]);
127 //     return ans;
128 // }
129 db closepoint(vector<point>&A,int l,int r){ // 最近点对 , 先要按照 x 坐标排序
130     if (r-l<=5){
131         db ans=1e20;
132         for (int i=l;i<=r;i++) for (int j=i+1;j<=r;j++) ans=min(ans,A[i].dis(A[j]));
133         return ans;
134     }
135     int mid=l+r>>1; db ans=min(closepoint(A,l,mid),closepoint(A,mid+1,r));
136     vector<point>B; for (int i=l;i<=r;i++) if (abs(A[i].x-A[mid].x)<=ans) B.push_back(A
            [i]);
137     sort(B.begin(),B.end(),[](point k1,point k2){return k1.y<k2.y;});
138     for (int i=0;i<B.size();i++) for (int j=i+1;j<B.size()&&B[j].y-B[i].y<ans;j++) ans=
            min(ans,B[i].dis(B[j]));
139     return ans;
140 }
141 int checkposCC(circle k1,circle k2){// 返回两个圆的公切线数量
142     if (cmp(k1.r,k2.r)==-1) swap(k1,k2);
143     db dis=k1.o.dis(k2.o);  int w1=cmp(dis,k1.r+k2.r),w2=cmp(dis,k1.r-k2.r);
144     if (w1>0) return 4; else if (w1==0) return 3; else if (w2>0) return 2;
145     else if (w2==0) return 1; else return 0;
146 }
147 vector<point> getCL(circle k1,point k2,point k3){ // 沿着 k2->k3 方向给出 , 相切给出两
        个
148     point k=proj(k2,k3,k1.o); db d=k1.r*k1.r-(k-k1.o).abs2();
149     if (sign(d)==-1) return {};
150     point del=(k3-k2).unit()*sqrt(max((db)0.0,d)); return {k-del,k+del};
151 }
152 vector<point> getCC(circle k1,circle k2){// 沿圆 k1 逆时针给出 , 相切给出两个
153     int pd=checkposCC(k1,k2); if (pd==0||pd==4) return {};
154     db a=(k2.o-k1.o).abs2(),cosA=(k1.r*k1.r+a-k2.r*k2.r)/(2*k1.r*sqrt(max(a,(db)0.0)));
155     db b=k1.r*cosA,c=sqrt(max((db)0.0,k1.r*k1.r-b*b));
156     point k=(k2.o-k1.o).unit(),m=k1.o+k*b,del=k.turn90()*c;
157     return {m-del,m+del};
158 }
159 vector<point> TangentCP(circle k1,point k2){// 沿圆 k1 逆时针给出
```

```
160     db a=(k2-k1.o).abs(),b=k1.r*k1.r/a,c=sqrt(max((db)0.0,k1.r*k1.r-b*b));
161     point k=(k2-k1.o).unit(),m=k1.o+k*b,del=k.turn90()*c;
162     return {m-del,m+del};
163 }
164 vector<line> TangentoutCC(circle k1,circle k2){
165     int pd=checkposCC(k1,k2); if (pd==0) return {};
166     if (pd==1){point k=getCC(k1,k2)[0]; return {(line){k,k}};}
167     if (cmp(k1.r,k2.r)==0){
168         point del=(k2.o-k1.o).unit().turn90().getdel();
169         return {(line){k1.o-del*k1.r,k2.o-del*k2.r},(line){k1.o+del*k1.r,k2.o+del*k2.r
                }};
170     } else {
171         point p=(k2.o*k1.r-k1.o*k2.r)/(k1.r-k2.r);
172         vector<point>A=TangentCP(k1,p),B=TangentCP(k2,p);
173         vector<line>ans; for (int i=0;i<A.size();i++) ans.push_back((line){A[i],B[i]});
174         return ans;
175     }
176 }
177 vector<line> TangentinCC(circle k1,circle k2){
178     int pd=checkposCC(k1,k2); if (pd<=2) return {};
179     if (pd==3){point k=getCC(k1,k2)[0]; return {(line){k,k}};}
180     point p=(k2.o*k1.r+k1.o*k2.r)/(k1.r+k2.r);
181     vector<point>A=TangentCP(k1,p),B=TangentCP(k2,p);
182     vector<line>ans; for (int i=0;i<A.size();i++) ans.push_back((line){A[i],B[i]});
183     return ans;
184 }
185 vector<line> TangentCC(circle k1,circle k2){
186     int flag=0; if (k1.r<k2.r) swap(k1,k2),flag=1;
187     vector<line>A=TangentoutCC(k1,k2),B=TangentinCC(k1,k2);
188     for (line k:B) A.push_back(k);
189     if (flag) for (line &k:A) swap(k[0],k[1]);
190     return A;
191 }
192 db getarea(circle k1,point k2,point k3){
193     // 圆 k1 与三角形 k2 k3 k1.o 的有向面积交
194     point k=k1.o; k1.o=k1.o-k; k2=k2-k; k3=k3-k;
195     int pd1=k1.inside(k2),pd2=k1.inside(k3);
196     vector<point>A=getCL(k1,k2,k3);
197     if (pd1>=0){
198         if (pd2>=0) return cross(k2,k3)/2;
199         return k1.r*k1.r*rad(A[1],k3)/2+cross(k2,A[1])/2;
200     } else if (pd2>=0){
201         return k1.r*k1.r*rad(k2,A[0])/2+cross(A[0],k3)/2;
202     }else {
203         int pd=cmp(k1.r,disSP(k2,k3,k1.o));
204         if (pd<=0) return k1.r*k1.r*rad(k2,k3)/2;
205         return cross(A[0],A[1])/2+k1.r*k1.r*(rad(k2,A[0])+rad(A[1],k3))/2;
206     }
207 }
208 circle getcircle(point k1,point k2,point k3){
209     db a1=k2.x-k1.x,b1=k2.y-k1.y,c1=(a1*a1+b1*b1)/2;
210     db a2=k3.x-k1.x,b2=k3.y-k1.y,c2=(a2*a2+b2*b2)/2;
211     db d=a1*b2-a2*b1;
212     point o=(point){k1.x+(c1*b2-c2*b1)/d,k1.y+(a1*c2-a2*c1)/d};
213     return (circle){o,k1.dis(o)};
214 }
215 circle getScircle(vector<point> A){
216     random_shuffle(A.begin(),A.end());
217     circle ans=(circle){A[0],0};
```

```
    for (int i=1;i<A.size();i++)
        if (ans.inside(A[i])==-1){
            ans=(circle){A[i],0};
            for (int j=0;j<i;j++)
                if (ans.inside(A[j])==-1){
                    ans.o=(A[i]+A[j])/2; ans.r=ans.o.dis(A[i]);
                    for (int k=0;k<j;k++)
                        if (ans.inside(A[k])==-1)
                            ans=getcircle(A[i],A[j],A[k]);
                }
        }
    return ans;
}
db area(vector<point> A){ // 多边形用 vector<point> 表示 , 逆时针
    db ans=0;
    for (int i=0;i<A.size();i++) ans+=cross(A[i],A[(i+1)%A.size()]);
    return ans/2;
}
int checkconvex(vector<point>A){
    int n=A.size(); A.push_back(A[0]); A.push_back(A[1]);
    for (int i=0;i<n;i++) if (sign(cross(A[i+1]-A[i],A[i+2]-A[i]))==-1) return 0;
    return 1;
}
int contain(vector<point>A,point q){ // 2 内部 1 边界 0 外部
    int pd=0; A.push_back(A[0]);
    for (int i=1;i<A.size();i++){
        point u=A[i-1],v=A[i];
        if (onS(u,v,q)) return 1; if (cmp(u.y,v.y)>0) swap(u,v);
        if (cmp(u.y,q.y)>=0||cmp(v.y,q.y)<0) continue;
        if (sign(cross(u-v,q-v))<0) pd^=1;
    }
    return pd<<1;
}
vector<point> ConvexHull(vector<point>A,int flag=1){ // flag=0 不严格 flag=1 严格
    int n=A.size(); vector<point>ans(n*2);
    sort(A.begin(),A.end()); int now=-1;
    for (int i=0;i<A.size();i++){
        while (now>0&&sign(cross(ans[now]-ans[now-1],A[i]-ans[now-1]))<flag) now--;
        ans[++now]=A[i];
    } int pre=now;
    for (int i=n-2;i>=0;i--){
        while (now>pre&&sign(cross(ans[now]-ans[now-1],A[i]-ans[now-1]))<flag) now--;
        ans[++now]=A[i];
    } ans.resize(now); return ans;
}
db convexDiameter(vector<point>A){
    int now=0,n=A.size(); db ans=0;
    for (int i=0;i<A.size();i++){
        now=max(now,i);
        while (1){
            db k1=A[i].dis(A[now%n]),k2=A[i].dis(A[(now+1)%n]);
            ans=max(ans,max(k1,k2)); if (k2>k1) now++; else break;
        }
    }
    return ans;
}
vector<point> convexcut(vector<point>A,point k1,point k2){
    // 保留 k1,k2,p 逆时针的所有点
    int n=A.size(); A.push_back(A[0]); vector<point>ans;
```

```
277     for (int i=0;i<n;i++){
278         int w1=clockwise(k1,k2,A[i]),w2=clockwise(k1,k2,A[i+1]);
279         if (w1>=0) ans.push_back(A[i]);
280         if (w1*w2<0) ans.push_back(getLL(k1,k2,A[i],A[i+1]));
281     }
282     return ans;
283 }
284 int checkPoS(vector<point>A,point k1,point k2){
285     // 多边形 A 和直线 ( 线段 )k1->k2 严格相交 , 注释部分为线段
286     struct ins{
287         point m,u,v;
288         int operator < (const ins& k) const {return m<k.m;}
289     }; vector<ins>B;
290     //if (contain(A,k1)==2||contain(A,k2)==2) return 1;
291     vector<point>poly=A; A.push_back(A[0]);
292     for (int i=1;i<A.size();i++) if (checkLL(A[i-1],A[i],k1,k2)){
293         point m=getLL(A[i-1],A[i],k1,k2);
294         if (inmid(A[i-1],A[i],m)/*&&inmid(k1,k2,m)*/) B.push_back((ins){m,A[i-1],A[i]})
                ;
295     }
296     if (B.size()==0) return 0; sort(B.begin(),B.end());
297     int now=1; while (now<B.size()&&B[now].m==B[0].m) now++;
298     if (now==B.size()) return 0;
299     int flag=contain(poly,(B[0].m+B[now].m)/2);
300     if (flag==2) return 1;
301     point d=B[now].m-B[0].m;
302     for (int i=now;i<B.size();i++){
303         if (!(B[i].m==B[i-1].m)&&flag==2) return 1;
304         int tag=sign(cross(B[i].v-B[i].u,B[i].m+d-B[i].u));
305         if (B[i].m==B[i].u||B[i].m==B[i].v) flag+=tag; else flag+=tag*2;
306     }
307     //return 0;
308     return flag==2;
309 }
310 int checkinp(point r,point l,point m){
311     if (compareangle(l,r)){return compareangle(l,m)&&compareangle(m,r);}
312     return compareangle(l,m)||compareangle(m,r);
313 }
314 int checkPosFast(vector<point>A,point k1,point k2){ // 快速检查线段是否和多边形严格相交
315     if (contain(A,k1)==2||contain(A,k2)==2) return 1; if (k1==k2) return 0;
316     A.push_back(A[0]); A.push_back(A[1]);
317     for (int i=1;i+1<A.size();i++)
318         if (checkLL(A[i-1],A[i],k1,k2)){
319             point now=getLL(A[i-1],A[i],k1,k2);
320             if (inmid(A[i-1],A[i],now)==0||inmid(k1,k2,now)==0) continue;
321             if (now==A[i]){
322                 if (A[i]==k2) continue;
323                 point pre=A[i-1],ne=A[i+1];
324                 if (checkinp(pre-now,ne-now,k2-now)) return 1;
325             } else if (now==k1){
326                 if (k1==A[i-1]||k1==A[i]) continue;
327                 if (checkinp(A[i-1]-k1,A[i]-k1,k2-k1)) return 1;
328             } else if (now==k2||now==A[i-1]) continue;
329             else return 1;
330         }
331     return 0;
332 }
333 // 拆分凸包成上下凸壳 凸包尽量都随机旋转一个角度来避免出现相同横坐标
334 // 尽量特判只有一个点的情况 凸包逆时针
```

```
335 void getUDP(vector<point>A,vector<point>&U,vector<point>&D){
336     db l=1e100,r=-1e100;
337     for (int i=0;i<A.size();i++) l=min(l,A[i].x),r=max(r,A[i].x);
338     int wherel,wherer;
339     for (int i=0;i<A.size();i++) if (cmp(A[i].x,l)==0) wherel=i;
340     for (int i=A.size();i;i--) if (cmp(A[i-1].x,r)==0) wherer=i-1;
341     U.clear(); D.clear(); int now=wherel;
342     while (1){D.push_back(A[now]); if (now==wherer) break; now++; if (now>=A.size())
            now=0;}
343     now=wherel;
344     while (1){U.push_back(A[now]); if (now==wherer) break; now--; if (now<0) now=A.size
            ()-1;}
345 }
346 // 需要保证凸包点数大于等于 3,2 内部 ,1 边界 ,0 外部
347 int containCoP(const vector<point>&U,const vector<point>&D,point k){
348     db lx=U[0].x,rx=U[U.size()-1].x;
349     if (k==U[0]||k==U[U.size()-1]) return 1;
350     if (cmp(k.x,lx)==-1||cmp(k.x,rx)==1) return 0;
351     int where1=lower_bound(U.begin(),U.end(),(point){k.x,-1e100})-U.begin();
352     int where2=lower_bound(D.begin(),D.end(),(point){k.x,-1e100})-D.begin();
353     int w1=clockwise(U[where1-1],U[where1],k),w2=clockwise(D[where2-1],D[where2],k);
354     if (w1==1||w2==-1) return 0; else if (w1==0||w2==0) return 1; return 2;
355 }
356 // d 是方向 , 输出上方切点和下方切点
357 pair<point,point> getTangentCow(const vector<point> &U,const vector<point> &D,point d){
358     if (sign(d.x)<0||(sign(d.x)==0&&sign(d.y)<0)) d=d*(-1);
359     point whereU,whereD;
360     if (sign(d.x)==0) return mp(U[0],U[U.size()-1]);
361     int l=0,r=U.size()-1,ans=0;
362     while (l<r){int mid=l+r>>1; if (sign(cross(U[mid+1]-U[mid],d))<=0) l=mid+1,ans=mid
            +1; else r=mid;}
363     whereU=U[ans]; l=0,r=D.size()-1,ans=0;
364     while (l<r){int mid=l+r>>1; if (sign(cross(D[mid+1]-D[mid],d))>=0) l=mid+1,ans=mid
            +1; else r=mid;}
365     whereD=D[ans]; return mp(whereU,whereD);
366 }
367 // 先检查 contain, 逆时针给出
368 pair<point,point> getTangentCoP(const vector<point>&U,const vector<point>&D,point k){
369     db lx=U[0].x,rx=U[U.size()-1].x;
370     if (k.x<lx){
371         int l=0,r=U.size()-1,ans=U.size()-1;
372         while (l<r){int mid=l+r>>1; if (clockwise(k,U[mid],U[mid+1])==1) l=mid+1; else
                ans=mid,r=mid;}
373         point w1=U[ans]; l=0,r=D.size()-1,ans=D.size()-1;
374         while (l<r){int mid=l+r>>1; if (clockwise(k,D[mid],D[mid+1])==-1) l=mid+1; else
                 ans=mid,r=mid;}
375         point w2=D[ans]; return mp(w1,w2);
376     } else if (k.x>rx){
377         int l=1,r=U.size(),ans=0;
378         while (l<r){int mid=l+r>>1; if (clockwise(k,U[mid],U[mid-1])==-1) r=mid; else
                ans=mid,l=mid+1;}
379         point w1=U[ans]; l=1,r=D.size(),ans=0;
380         while (l<r){int mid=l+r>>1; if (clockwise(k,D[mid],D[mid-1])==1) r=mid; else
                ans=mid,l=mid+1;}
381         point w2=D[ans]; return mp(w2,w1);
382     } else {
383         int where1=lower_bound(U.begin(),U.end(),(point){k.x,-1e100})-U.begin();
384         int where2=lower_bound(D.begin(),D.end(),(point){k.x,-1e100})-D.begin();
385         if ((k.x==lx&&k.y>U[0].y)||(where1&&clockwise(U[where1-1],U[where1],k)==1)){
```

```
386             int l=1,r=where1+1,ans=0;
387             while (l<r){int mid=l+r>>1; if (clockwise(k,U[mid],U[mid-1])==1) ans=mid,l=
                    mid+1; else r=mid;}
388             point w1=U[ans]; l=where1,r=U.size()-1,ans=U.size()-1;
389             while (l<r){int mid=l+r>>1; if (clockwise(k,U[mid],U[mid+1])==1) l=mid+1;
                    else ans=mid,r=mid;}
390             point w2=U[ans]; return mp(w2,w1);
391         } else {
392             int l=1,r=where2+1,ans=0;
393             while (l<r){int mid=l+r>>1; if (clockwise(k,D[mid],D[mid-1])==-1) ans=mid,l
                    =mid+1; else r=mid;}
394             point w1=D[ans]; l=where2,r=D.size()-1,ans=D.size()-1;
395             while (l<r){int mid=l+r>>1; if (clockwise(k,D[mid],D[mid+1])==-1) l=mid+1;
                    else ans=mid,r=mid;}
396             point w2=D[ans]; return mp(w1,w2);
397         }
398     }
399 }
400 struct P3{
401     db x,y,z;
402     P3 operator + (P3 k1){return (P3){x+k1.x,y+k1.y,z+k1.z};}
403     P3 operator - (P3 k1){return (P3){x-k1.x,y-k1.y,z-k1.z};}
404     P3 operator * (db k1){return (P3){x*k1,y*k1,z*k1};}
405     P3 operator / (db k1){return (P3){x/k1,y/k1,z/k1};}
406     db abs2(){return x*x+y*y+z*z;}
407     db abs(){return sqrt(x*x+y*y+z*z);}
408     P3 unit(){return (*this)/abs();}
409     int operator < (const P3 k1) const{
410         if (cmp(x,k1.x)!=0) return x<k1.x;
411         if (cmp(y,k1.y)!=0) return y<k1.y;
412         return cmp(z,k1.z)==-1;
413     }
414     int operator == (const P3 k1){
415         return cmp(x,k1.x)==0&&cmp(y,k1.y)==0&&cmp(z,k1.z)==0;
416     }
417     void scan(){
418         double k1,k2,k3; scanf("%lf%lf%lf",&k1,&k2,&k3);
419         x=k1; y=k2; z=k3;
420     }
421 };
422 P3 cross(P3 k1,P3 k2){return (P3){k1.y*k2.z-k1.z*k2.y,k1.z*k2.x-k1.x*k2.z,k1.x*k2.y-k1.
        y*k2.x};}
423 db dot(P3 k1,P3 k2){return k1.x*k2.x+k1.y*k2.y+k1.z*k2.z;}
424 //p=(3,4,5),l=(13,19,21),theta=85 ans=(2.83,4.62,1.77)
425 P3 turn3D(db k1,P3 l,P3 p){
426     l=l.unit(); P3 ans; db c=cos(k1),s=sin(k1);
427     ans.x=p.x*(l.x*l.x*(1-c)+c)+p.y*(l.x*l.y*(1-c)-l.z*s)+p.z*(l.x*l.z*(1-c)+l.y*s);
428     ans.y=p.x*(l.x*l.y*(1-c)+l.z*s)+p.y*(l.y*l.y*(1-c)+c)+p.z*(l.y*l.z*(1-c)-l.x*s);
429     ans.z=p.x*(l.x*l.z*(1-c)-l.y*s)+p.y*(l.y*l.z*(1-c)+l.x*s)+p.z*(l.x*l.x*(1-c)+c);
430     return ans;
431 }
432 typedef vector<P3> VP;
433 typedef vector<VP> VVP;
434 db Acos(db x){return acos(max(-(db)1,min(x,(db)1)));}
435 // 球面距离 , 圆心原点 , 半径 1
436 db Odist(P3 a,P3 b){db r=Acos(dot(a,b)); return r;}
437 db r; P3 rnd;
438 vector<db> solve(db a,db b,db c){
439     db r=sqrt(a*a+b*b),th=atan2(b,a);
```

```
440     if (cmp(c,-r)==-1) return {0};
441     else if (cmp(r,c)<=0) return {1};
442     else {
443         db tr=pi-Acos(c/r); return {th+pi-tr,th+pi+tr};
444     }
445 }
446 vector<db> jiao(P3 a,P3 b){
447     // dot(rd+x*cos(t)+y*sin(t),b) >= cos(r)
448     if (cmp(Odist(a,b),2*r)>0) return {0};
449     P3 rd=a*cos(r),z=a.unit(),y=cross(z,rnd).unit(),x=cross(y,z).unit();
450     vector<db> ret = solve(-(dot(x,b)*sin(r)),-(dot(y,b)*sin(r)),-(cos(r)-dot(rd,b)));
451     return ret;
452 }
453 db norm(db x,db l=0,db r=2*pi){ // change x into [l,r)
454     while (cmp(x,l)==-1) x+=(r-l); while (cmp(x,r)>=0) x-=(r-l);
455     return x;
456 }
457 db disLP(P3 k1,P3 k2,P3 q){
458     return (cross(k2-k1,q-k1)).abs()/(k2-k1).abs();
459 }
460 db disLL(P3 k1,P3 k2,P3 k3,P3 k4){
461     P3 dir=cross(k2-k1,k4-k3); if (sign(dir.abs())==0) return disLP(k1,k2,k3);
462     return fabs(dot(dir.unit(),k1-k2));
463 }
464 VP getFL(P3 p,P3 dir,P3 k1,P3 k2){
465     db a=dot(k2-p,dir),b=dot(k1-p,dir),d=a-b;
466     if (sign(fabs(d))==0) return {};
467     return {(k1*a-k2*b)/d};
468 }
469 VP getFF(P3 p1,P3 dir1,P3 p2,P3 dir2){// 返回一条线
470     P3 e=cross(dir1,dir2),v=cross(dir1,e);
471     db d=dot(dir2,v); if (sign(abs(d))==0) return {};
472     P3 q=p1+v*dot(dir2,p2-p1)/d; return {q,q+e};
473 }
474 // 3D Covex Hull Template
475 db getV(P3 k1,P3 k2,P3 k3,P3 k4){ // get the Volume
476     return dot(cross(k2-k1,k3-k1),k4-k1);
477 }
478 db rand_db(){return 1.0*rand()/RAND_MAX;}
479 VP convexHull2D(VP A,P3 dir){
480     P3 x={(db)rand(),(db)rand(),(db)rand()}; x=x.unit();
481     x=cross(x,dir).unit(); P3 y=cross(x,dir).unit();
482     P3 vec=dir.unit()*dot(A[0],dir);
483     vector<point>B;
484     for (int i=0;i<A.size();i++) B.push_back((point){dot(A[i],x),dot(A[i],y)});
485     B=ConvexHull(B); A.clear();
486     for (int i=0;i<B.size();i++) A.push_back(x*B[i].x+y*B[i].y+vec);
487     return A;
488 }
489 namespace CH3{
490     VVP ret; set<pair<int,int> >e;
491     int n; VP p,q;
492     void wrap(int a,int b){
493         if (e.find({a,b})==e.end()){
494             int c=-1;
495             for (int i=0;i<n;i++) if (i!=a&&i!=b){
496                 if (c==-1||sign(getV(q[c],q[a],q[b],q[i]))>0) c=i;
497             }
498             if (c!=-1){
```

```
499                 ret.push_back({p[a],p[b],p[c]});
500                 e.insert({a,b}); e.insert({b,c}); e.insert({c,a});
501                 wrap(c,b); wrap(a,c);
502             }
503         }
504     }
505     VVP ConvexHull3D(VP _p){
506         p=q=_p; n=p.size();
507         ret.clear(); e.clear();
508         for (auto &i:q) i=i+(P3){rand_db()*1e-4,rand_db()*1e-4,rand_db()*1e-4};
509         for (int i=1;i<n;i++) if (q[i].x<q[0].x) swap(p[0],p[i]),swap(q[0],q[i]);
510         for (int i=2;i<n;i++) if ((q[i].x-q[0].x)*(q[1].y-q[0].y)>(q[i].y-q[0].y)*(q
                [1].x-q[0].x)) swap(q[1],q[i]),swap(p[1],p[i]);
511         wrap(0,1);
512         return ret;
513     }
514 }
515 VVP reduceCH(VVP A){
516     VVP ret; map<P3,VP> M;
517     for (VP nowF:A){
518         P3 dir=cross(nowF[1]-nowF[0],nowF[2]-nowF[0]).unit();
519         for (P3 k1:nowF) M[dir].pb(k1);
520     }
521     for (pair<P3,VP> nowF:M) ret.pb(convexHull2D(nowF.se,nowF.fi));
522     return ret;
523 }
524 //  把一个面变成 ( 点 , 法向量 ) 的形式
525 pair<P3,P3> getF(VP F){
526     return mp(F[0],cross(F[1]-F[0],F[2]-F[0]).unit());
527 }
528 // 3D Cut 保留 dot(dir,x-p)>=0 的部分
529 VVP ConvexCut3D(VVP A,P3 p,P3 dir){
530     VVP ret; VP sec;
531     for (VP nowF: A){
532         int n=nowF.size(); VP ans; int dif=0;
533         for (int i=0;i<n;i++){
534             int d1=sign(dot(dir,nowF[i]-p));
535             int d2=sign(dot(dir,nowF[(i+1)%n]-p));
536             if (d1>=0) ans.pb(nowF[i]);
537             if (d1*d2<0){
538                 P3 q=getFL(p,dir,nowF[i],nowF[(i+1)%n])[0];
539                 ans.push_back(q); sec.push_back(q);
540             }
541             if (d1==0) sec.push_back(nowF[i]); else dif=1;
542             dif|=(sign(dot(dir,cross(nowF[(i+1)%n]-nowF[i],nowF[(i+1)%n]-nowF[i])))
                    ==-1);
543         }
544         if (ans.size()>0&&dif) ret.push_back(ans);
545     }
546     if (sec.size()>0) ret.push_back(convexHull2D(sec,dir));
547     return ret;
548 }
549 db vol(VVP A){
550     if (A.size()==0) return 0; P3 p=A[0][0]; db ans=0;
551     for (VP nowF:A)
552         for (int i=2;i<nowF.size();i++)
553             ans+=abs(getV(p,nowF[0],nowF[i-1],nowF[i]));
554     return ans/6;
555 }
```

```
556 VVP init(db INF) {
557     VVP pss(6,VP(4));
558     pss[0][0] = pss[1][0] = pss[2][0] = {-INF, -INF, -INF};
559     pss[0][3] = pss[1][1] = pss[5][2] = {-INF, -INF, INF};
560     pss[0][1] = pss[2][3] = pss[4][2] = {-INF, INF, -INF};
561     pss[0][2] = pss[5][3] = pss[4][1] = {-INF, INF, INF};
562     pss[1][3] = pss[2][1] = pss[3][2] = {INF, -INF, -INF};
563     pss[1][2] = pss[5][1] = pss[3][3] = {INF, -INF, INF};
564     pss[2][2] = pss[4][3] = pss[3][1] = {INF, INF, -INF};
565     pss[5][0] = pss[4][0] = pss[3][0] = {INF, INF, INF};
566     return pss;
567 }
```

7 Others

7.1 mint 类

```
const int mod = 998244353;

struct mint {
  int n;
  mint(int n_ = 0) : n(n_) {}
};

mint operator+(mint a, mint b) { return (a.n += b.n) >= mod ? a.n - mod : a.n; }
mint operator-(mint a, mint b) { return (a.n -= b.n) < 0 ? a.n + mod : a.n; }
mint operator*(mint a, mint b) { return 1LL * a.n * b.n % mod; }
mint &operator+=(mint &a, mint b) { return a = a + b; }
mint &operator-=(mint &a, mint b) { return a = a - b; }
mint &operator*=(mint &a, mint b) { return a = a * b; }
ostream &operator<<(ostream &o, mint a) { return o << a.n; }
```

7.2 不重叠区间贪心

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

const int maxn = 5e5+5;
pair<int, int> a[maxn];
int main() {
    int n;
    cin >> n;
    for (int i = 1; i <= n; ++i) {
        cin >> a[i].second >> a[i].first;
    }
    sort(a + 1, a + 1 + n);
    int res = 1;
    int tmp = a[1].first;
//    printf("%d %d\n", a[1].second, a[1].first);
    for (int i = 2; i <= n; ++i) {
        if (a[i].second > tmp) {
            res ++;
//            printf("%d %d\n", a[i].second, a[i].first);
            tmp = a[i].first;
        }
    }
    printf("%d\n", res);
    return 0;
}
```

7.3 BigInt 类

```
const double PI = acos(-1.0);
struct Complex{
    double x,y;
    Complex(double _x = 0.0,double _y = 0.0){
        x = _x;
        y = _y;
    }
```

```
8       Complex operator-(const Complex &b)const{
9           return Complex(x - b.x,y - b.y);
10      }
11      Complex operator+(const Complex &b)const{
12          return Complex(x + b.x,y + b.y);
13      }
14      Complex operator*(const Complex &b)const{
15          return Complex(x*b.x - y*b.y,x*b.y + y*b.x);
16      }
17  };
18  void change(Complex y[],int len){
19      int i,j,k;
20      for(int i = 1,j = len/2;i<len-1;i++){
21          if(i < j)    swap(y[i],y[j]);
22          k = len/2;
23          while(j >= k){
24              j = j - k;
25              k = k/2;
26          }
27          if(j < k)    j+=k;
28      }
29  }
30  void fft(Complex y[],int len,int on){
31      change(y,len);
32      for(int h = 2;h <= len;h<<=1){
33          Complex wn(cos(on*2*PI/h),sin(on*2*PI/h));
34          for(int j = 0;j < len;j += h){
35              Complex w(1,0);
36              for(int k = j;k < j + h/2;k++){
37                  Complex u = y[k];
38                  Complex t = w*y[k + h/2];
39                  y[k] = u + t;
40                  y[k + h/2] = u - t;
41                  w = w*wn;
42              }
43          }
44      }
45      if(on == -1){
46          for(int i = 0;i < len;i++){
47              y[i].x /= len;
48          }
49      }
50  }
51  class BigInt
52  {
53  #define Value(x, nega) ((nega) ? -(x) : (x))
54  #define At(vec, index) ((index) < vec.size() ? vec[(index)] : 0)
55      static int absComp(const BigInt &lhs, const BigInt &rhs)
56      {
57          if (lhs.size() != rhs.size())
58              return lhs.size() < rhs.size() ? -1 : 1;
59          for (int i = lhs.size() - 1; i >= 0; --i)
60              if (lhs[i] != rhs[i])
61                  return lhs[i] < rhs[i] ? -1 : 1;
62          return 0;
63      }
64      using Long = long long;
65      const static int Exp = 9;
66      const static Long Mod = 1000000000;
```

```
67      mutable std::vector<Long> val;
68      mutable bool nega = false;
69      void trim() const
70      {
71          while (val.size() && val.back() == 0)
72              val.pop_back();
73          if (val.empty())
74              nega = false;
75      }
76      int size() const { return val.size(); }
77      Long &operator[](int index) const { return val[index]; }
78      Long &back() const { return val.back(); }
79      BigInt(int size, bool nega) : val(size), nega(nega) {}
80      BigInt(const std::vector<Long> &val, bool nega) : val(val), nega(nega) {}
81
82  public:
83      friend std::ostream &operator<<(std::ostream &os, const BigInt &n)
84      {
85          if (n.size())
86          {
87              if (n.nega)
88                  putchar('-');
89              for (int i = n.size() - 1; i >= 0; --i)
90              {
91                  if (i == n.size() - 1)
92                      printf("%lld", n[i]);
93                  else
94                      printf("%0*lld", n.Exp, n[i]);
95              }
96          }
97          else
98              putchar('0');
99          return os;
100     }
101     friend BigInt operator+(const BigInt &lhs, const BigInt &rhs)
102     {
103         BigInt ret(lhs);
104         return ret += rhs;
105     }
106     friend BigInt operator-(const BigInt &lhs, const BigInt &rhs)
107     {
108         BigInt ret(lhs);
109         return ret -= rhs;
110     }
111     BigInt(Long x = 0)
112     {
113         if (x < 0)
114             x = -x, nega = true;
115         while (x >= Mod)
116             val.push_back(x % Mod), x /= Mod;
117         if (x)
118             val.push_back(x);
119     }
120     BigInt(const char *s)
121     {
122         int bound = 0, pos;
123         if (s[0] == '-')
124             nega = true, bound = 1;
125         Long cur = 0, pow = 1;
```

```
126         for (pos = strlen(s) - 1; pos >= Exp + bound - 1; pos -= Exp, val.push_back(cur
                ), cur = 0, pow = 1)
127             for (int i = pos; i > pos - Exp; --i)
128                 cur += (s[i] - '0') * pow, pow *= 10;
129         for (cur = 0, pow = 1; pos >= bound; --pos)
130             cur += (s[pos] - '0') * pow, pow *= 10;
131         if (cur)
132             val.push_back(cur);
133     }
134     BigInt &operator=(const char *s){
135         BigInt n(s);
136         *this = n;
137         return n;
138     }
139     BigInt &operator=(const Long x){
140         BigInt n(x);
141         *this = n;
142         return n;
143     }
144     friend std::istream &operator>>(std::istream &is, BigInt &n){
145         string s;
146         is >> s;
147         n=(char*)s.data();
148         return is;
149     }
150     BigInt &operator+=(const BigInt &rhs)
151     {
152         const int cap = std::max(size(), rhs.size()) + 1;
153         val.resize(cap);
154         int carry = 0;
155         for (int i = 0; i < cap - 1; ++i)
156         {
157             val[i] = Value(val[i], nega) + Value(At(rhs, i), rhs.nega) + carry, carry =
                     0;
158             if (val[i] >= Mod)
159                 val[i] -= Mod, carry = 1;
160             else if (val[i] < 0)
161                 val[i] += Mod, carry = -1;
162         }
163         if ((val.back() = carry) == -1) //assert(val.back() == 1 or 0 or -1)
164         {
165             nega = true, val.pop_back();
166             bool tailZero = true;
167             for (int i = 0; i < cap - 1; ++i)
168             {
169                 if (tailZero && val[i])
170                     val[i] = Mod - val[i], tailZero = false;
171                 else
172                     val[i] = Mod - 1 - val[i];
173             }
174         }
175         trim();
176         return *this;
177     }
178     friend BigInt operator-(const BigInt &rhs)
179     {
180         BigInt ret(rhs);
181         ret.nega ^= 1;
182         return ret;
```

```
183     }
184     BigInt &operator-=(const BigInt &rhs)
185     {
186         rhs.nega ^= 1;
187         *this += rhs;
188         rhs.nega ^= 1;
189         return *this;
190     }
191     friend BigInt operator*(const BigInt &lhs, const BigInt &rhs)
192     {
193         int len=1;
194         BigInt ll=lhs,rr=rhs;
195         ll.nega = lhs.nega ^ rhs.nega;
196         while(len<2*lhs.size()||len<2*rhs.size())len<<=1;
197         ll.val.resize(len),rr.val.resize(len);
198         Complex x1[len],x2[len];
199         for(int i=0;i<len;i++){
200             Complex nx(ll[i],0.0),ny(rr[i],0.0);
201             x1[i]=nx;
202             x2[i]=ny;
203         }
204         fft(x1,len,1);
205         fft(x2,len,1);
206         for(int i = 0 ; i < len; i++)
207             x1[i] = x1[i] * x2[i];
208         fft( x1 , len , -1 );
209         for(int i = 0 ; i < len; i++)
210             ll[i] = int( x1[i].x + 0.5 );
211         for(int i = 0 ; i < len; i++){
212             ll[i+1]+=ll[i]/Mod;
213             ll[i]%=Mod;
214         }
215         ll.trim();
216         return ll;
217     }
218     friend BigInt operator*(const BigInt &lhs, const Long &x){
219         BigInt ret=lhs;
220         bool negat = ( x < 0 );
221         Long xx = (negat) ? -x : x;
222         ret.nega ^= negat;
223         ret.val.push_back(0);
224         ret.val.push_back(0);
225         for(int i = 0; i < ret.size(); i++)
226             ret[i]*=xx;
227         for(int i = 0; i < ret.size(); i++){
228             ret[i+1]+=ret[i]/Mod;
229             ret[i] %= Mod;
230         }
231         ret.trim();
232         return ret;
233     }
234     BigInt &operator*=(const BigInt &rhs) { return *this = *this * rhs; }
235     BigInt &operator*=(const Long &x) { return *this = *this * x; }
236     friend BigInt operator/(const BigInt &lhs, const BigInt &rhs)
237     {
238         static std::vector<BigInt> powTwo{BigInt(1)};
239         static std::vector<BigInt> estimate;
240         estimate.clear();
241         if (absComp(lhs, rhs) < 0)
```

```
242             return BigInt();
243         BigInt cur = rhs;
244         int cmp;
245         while ((cmp = absComp(cur, lhs)) <= 0)
246         {
247             estimate.push_back(cur), cur += cur;
248             if (estimate.size() >= powTwo.size())
249                 powTwo.push_back(powTwo.back() + powTwo.back());
250         }
251         if (cmp == 0)
252             return BigInt(powTwo.back().val, lhs.nega ^ rhs.nega);
253         BigInt ret = powTwo[estimate.size() - 1];
254         cur = estimate[estimate.size() - 1];
255         for (int i = estimate.size() - 1; i >= 0 && cmp != 0; --i)
256             if ((cmp = absComp(cur + estimate[i], lhs)) <= 0)
257                 cur += estimate[i], ret += powTwo[i];
258         ret.nega = lhs.nega ^ rhs.nega;
259         return ret;
260     }
261     friend BigInt operator/(const BigInt &num,const Long &x){
262         bool negat = ( x < 0 );
263         Long xx = (negat) ? -x : x;
264         BigInt ret;
265         Long k = 0;
266         ret.val.resize( num.size() );
267         ret.nega = (num.nega ^ negat);
268         for(int i = num.size() - 1 ;i >= 0; i--){
269             ret[i] = ( k * Mod + num[i]) / xx;
270             k = ( k * Mod + num[i]) % xx;
271         }
272         ret.trim();
273         return ret;
274     }
275     bool operator==(const BigInt &rhs) const
276     {
277         return nega == rhs.nega && val == rhs.val;
278     }
279     bool operator!=(const BigInt &rhs) const { return nega != rhs.nega || val != rhs.
        val; }
280     bool operator>=(const BigInt &rhs) const { return !(*this < rhs); }
281     bool operator>(const BigInt &rhs) const { return !(*this <= rhs); }
282     bool operator<=(const BigInt &rhs) const
283     {
284         if (nega && !rhs.nega)
285             return true;
286         if (!nega && rhs.nega)
287             return false;
288         int cmp = absComp(*this, rhs);
289         return nega ? cmp >= 0 : cmp <= 0;
290     }
291     bool operator<(const BigInt &rhs) const
292     {
293         if (nega && !rhs.nega)
294             return true;
295         if (!nega && rhs.nega)
296             return false;
297         return (absComp(*this, rhs) < 0) ^ nega;
298     }
299     void swap(const BigInt &rhs) const
```

```
300     {
301         std::swap(val, rhs.val);
302         std::swap(nega, rhs.nega);
303     }
304 };
305 BigInt ba,bb;
306 int main(){
307     cin>>ba>>bb;
308     cout << ba + bb << '\n';//和
309     cout << ba - bb << '\n';//差
310     cout << ba * bb << '\n';//积
311     BigInt d;
312     cout << (d = ba / bb) << '\n';//商
313     cout << ba - d * bb << '\n';//余
314     return 0;
315 }
```

7.4 date

```
1  string dayOfWeek[] = {"Mo", "Tu", "We", "Th", "Fr", "Sa", "Su"};
2  // converts Gregorian date to integer (Julian day number)
3  int DateToInt (int m, int d, int y){
4      return
5          1461 * (y + 4800 + (m - 14) / 12) / 4 +
6          367 * (m - 2 - (m - 14) / 12 * 12) / 12 -
7          3 * ((y + 4900 + (m - 14) / 12) / 100) / 4 +
8          d - 32075;
9  }
10
11 // converts integer (Julian day number) to Gregorian date: month/day/year
12 void IntToDate (int jd, int &m, int &d, int &y){
13     int x, n, i, j;
14     x = jd + 68569;
15     n = 4 * x / 146097;
16     x -= (146097 * n + 3) / 4;
17     i = (4000 * (x + 1)) / 1461001;
18     x -= 1461 * i / 4 - 31;
19     j = 80 * x / 2447;
20     d = x - 2447 * j / 80;
21     x = j / 11;
22     m = j + 2 - 12 * x;
23     y = 100 * (n - 49) + i + x;
24 }
25 // converts integer (Julian day number) to day of week
26 string IntToDay (int jd){
27     return dayOfWeek[jd % 7];
28 }
```

7.5 Frac 类

```
1  struct Frac {
2      ll a, b;
3      void getJian() {
4          ll gcd = abs(__gcd(a, b));
5          a /= gcd;
6          b /= gcd;
7          if (b < 0) {
```

```
            a = -a;
            b = -b;
        }
    }
    Frac(ll a_ = 1, ll b_ = 1) {
        a = a_;
        b = b_;
        getJian();
    }
    Frac operator + (const Frac& oth) {
        ll bt = b * oth.b;
        ll at = a * oth.b + oth.a * b;
        return Frac(at, bt);
    }
    Frac operator * (const Frac& oth) {
        a *= oth.a;
        b *= oth.b;
        getJian();
        return *this;
    }
    bool operator < (const Frac& oth) const {
        return a * oth.b < b * oth.a;
    }
    bool operator == (const Frac& oth) const {
        return a * oth.b == b * oth.a;
    }
    bool operator <= (const Frac& oth) const {
        return a * oth.b <= b * oth.a;
    }
};
```

7.6 模拟退火 (最小圆覆盖)

```
const int maxn = 1e5 + 10;
const double eps = 1e-8;
const double delta = 0.98;
const double inf = 1e18;

struct Point { double x, y; } p[maxn];

double dis(Point A, Point B) { return sqrt((A.x - B.x) * (A.x - B.x) + (A.y - B.y) * (A.y - B.y)); };

double Simulate_Annea(int n)
{
    Point S;
    S.x = S.y = 0;
    double t = 1000;
    double res = inf;
    while(t > eps)
    {
        int k = 0;
        for(int i = 0; i < n; i ++) if(dis(S, p[i]) > dis(S, p[k])) k = i;
        double d = dis(S, p[k]);
        res = min(res, d);
        S.x += (p[k].x - S.x) / d * t;
        S.y += (p[k].y - S.y) / d * t;
        t *= delta;
```

```
25      }
26      return res;
27  }
28
29  int main()
30  {
31      int n;
32      scanf("%d", &n);
33      for(int i = 0; i < n; i ++) scanf("%lf%lf", &p[i].x, &p[i].y);
34      printf("%.3f\n", Simulate_Annea(n));
35      return 0;
36  }
```

7.7 string 类

```
1   const int maxn = 1005;
2   struct String{
3       int nex[maxn];
4       char x[maxn];
5       int len;
6       int getLength() {
7           return len;
8       }
9       void getNext() {
10          int n = len, i = 0, j = -1;
11          nex[0] = -1;
12          while (i < n) {
13              if (j == -1 || x[i] == x[j]) nex[++i] = ++j;
14              else j = -1;
15          }
16      }
17      void input() {
18          scanf("%s", x);
19          len = strlen(x);
20      }
21      void inputAndCal() {
22          scanf("%s", x);
23          len = strlen(x);
24          getNext();
25      }
26      void show() {
27          printf("%s\n", x);
28      }
29      bool operator < (const String&oth) const {
30          return strcmp(x, oth.x) < 0;
31      }
32      char operator [] (const int a) const {
33          return x[a];
34      }
35      bool substring(String b) {//b is the substring of a
36          int m = len, n = b.getLength();
37          int i = 0, j = 0;
38          while (i < m && j < n) {
39              if (j == -1 || x[i] == b[j]) ++i, ++j;
40              else j = b.nex[j];
41              if (j == n) return true;
42          }
43          return false;
```

```
44      }
45  };
```

7.8 前缀异或和

```
1   ll xor_sum(ll n) {
2       ll t=n&3;
3       if (t&1) return t/2ull^1;
4       return t/2ull^n;
5   }
```

7.9 约瑟夫环第 k 个

```
1   ll kth(ll n, ll m, ll k) { // n个人, m间隔, 第k个出列的人
2       if (m == 1) return k;
3       ll res = (m - 1) % (n - k + 1);
4       for (ll i = n - k + 2, stp = 0; i <= n; i += stp, res += stp * m) {
5           if (res + m >= i) {
6               res = (res + m) % i;
7               i++;
8               stp = 0;
9           } else {
10              stp = (i - res - 2) / (m - 1);
11              if (i + stp > n) {
12                  res += (n - (i - 1)) * m;
13                  break;
14              }
15          }
16      }
17      return res + 1;
18  }
19
20  ll dieInXturn(int n, int k, int x) { // n个人, m间隔, 第k个人出列时间
21      ll tmp = 0;
22      while (n) {
23          x = (x + n) % n;
24          if (k > n)x += (k - x - 1 + n - 1) / n * n;
25          if ((x + 1) % k == 0) {
26              tmp += (x + 1) / k;
27              break;
28          } else {
29              if (k > n) {
30                  tmp += x / k;
31                  ll ttmp = x;
32                  x = x - (x / n + 1) * (x / k) + (x + n) / n * n - k;
33                  n -= ttmp / k;
34
35              } else {
36                  tmp += n / k;
37                  x = x - x / k;
38                  x += n - n / k * k;
39                  n -= n / k;
40              }
41          }
42      }
43      return tmp;
44  }
```

7.10 二分

```
// a为二分数组, x为需要查找的数, 返回最左端和最右端
pair<int, int> F(vector<int> a, int x) {
    int l = 0, r = a.size() - 1;
    int lres = -1;
    while (l <= r) {
        int mid = l + r >> 1;
        int tt = a[mid];
        if (tt >= x) {
            r = mid - 1;
        } else if (tt < x) {
            l = mid + 1;
        }
    }
    if (l >= a.size() || a[l] != x) return make_pair(-1, -1);
    lres = l;
    l = 0, r = a.size() - 1;
    while (l <= r) {
        int mid = l + r >> 1;
        int tt = a[mid];
        if (tt > x) {
            r = mid - 1;
        } else if (tt <= x) {
            l = mid + 1;
        }
    }
    return make_pair(lres, r);
}
```

7.11 猛男 IO 挂

```
const int LEN = 100000;
struct fastio {
    int it, len;
    char s[LEN + 5];
    fastio() {
        it = len = 0;
    }
    char get() {
        if (it < len) return s[it++];
        it = 0, len = fread(s, 1, LEN, stdin);
        return len ? s[it++] : EOF;
    }
    bool notend() {
        char c;
        for (c = get(); c == ' ' || c == '\n'; c = get());
        if (it) it--;
        return c != EOF;
    }
    void put(char c) {
        if (it == LEN) fwrite(s, 1, LEN, stdout), it = 0;
        s[it++] = c;
    }
    void flush() {
        fwrite(s, 1, it, stdout);
    }
} buff, bufo;
```

```
inline int getint() {
    char c;
    int res = 0, sig = 1;
    for (c = buff.get(); c < '0' || c > '9'; c = buff.get()) if (c == '-') sig = -1;
    for (; c >= '0' && c <= '9'; c = buff.get()) res = res * 10 + (c - '0');
    return sig * res;
}
inline ll getll() {
    char c;
    ll res = 0, sig = 1;
    for (c = buff.get(); c < '0' || c > '9'; c = buff.get()) if (c == '-') sig = -1;
    for (; c >= '0' && c <= '9'; c = buff.get()) res = res * 10 + (c - '0');
    return sig * res;
}
inline void putint(int x, char suf) {
    if (!x) bufo.put('0');
    else {
        if (x < 0) bufo.put('-'), x = -x;
        int k = 0;
        char s[15];
        while (x) {
            s[++k] = x % 10 + '0';
            x /= 10;
        }
        for (; k; k--) bufo.put(s[k]);
    }
    bufo.put(suf);
}
inline void putll(ll x, char suf) {
    if (!x) bufo.put('0');
    else {
        if (x < 0) bufo.put('-'), x = -x;
        int k = 0;
        char s[25];
        while (x) {
            s[++k] = x % 10 + '0';
            x /= 10;
        }
        for (; k; k--) bufo.put(s[k]);
    }
    bufo.put(suf);
}
inline char get_char() {
    char c;
    for (c = buff.get(); c == ' ' || c == '\n'; c = buff.get());
    return c;
}
```

7.12 贪心结论

```
// n个区间，挪到使得某个点被所有区间覆盖需要的最少步数时，选择的点是所有区间端点的中位数（mid~mid+1答案都是一样的）

// 2 * n 的格子填数，使得列上的最大和最小的填充方法：最大配最小，次大配次小，以此类推

// n个数，重排后使得相同位置上数不同的最大值为：如果max<=(sum - max)则为sum，如果max>(sum-max)则为2*(sum-max)

```

```
// 不重叠区间贪心
pair<int, int> a[maxn];
int main() {
    int n;
    cin >> n;
    for (int i = 1; i <= n; ++i) {
        cin >> a[i].second >> a[i].first;
    }
    sort(a + 1, a + 1 + n);
    int res = 1;
    int tmp = a[1].first;
//    printf("%d %d\n", a[1].second, a[1].first);
    for (int i = 2; i <= n; ++i) {
        if (a[i].second > tmp) {
            res ++;
//            printf("%d %d\n", a[i].second, a[i].first);
            tmp = a[i].first;
        }
    }
    printf("%d\n", res);
    return 0;
}
```

7.13 builtin

```
__builtin_popcount(unsigned int n) // 1的个数
__builtin_parity(unsigned int n) // 奇数个1返回1, 偶数个返回0
__builtin_ctz(unsigned int n) // 判断n的二进制末尾后面0的个数
__builtin_clz(unsigned int n) //返回前导0的个数
```

7.14 n 以内 k 因子的个数

```
// 返回1~n中k因子的个数
ll dig(ll n, ll k) {
    if (k == 1) return n;
    ll res = 0;
    while (n > 0) {
        res += n / k;
        n /= k;
    }
    return res;
}
```

7.15 每个点左右两边最长不重子序列

```
int r = 1;
for (int i = 1; i <= n; ++i) {
    while (r <= n && !vis[a[r]]) vis[a[r++]] = 1;
    vis[a[i]] = 0;
    R[i] = r - 1;
}
int l = n;
for (int i = n; i >= 1; --i) {
    while (l >= 1 && !vis[a[l]]) vis[a[l--]] = 1;
    vis[a[i]] = 0;
    L[i] = l + 1;
}
```