XCPC Code Library

XCPC's Bizarre Adventure

2022 年 10 月 19 日

目录

1 数据结构

1.1 可并堆

```cpp
struct Heap {
    LL val[MAXN], mult[MAXN], plus[MAXN];
    int lc[MAXN], rc[MAXN];
    void Mult(int t, LL dt) { if (t) val[t] *= dt, plus[t] *= dt, mult[t] *= dt; }
    void Plus(int t, LL dt) { if (t) val[t] += dt, plus[t] += dt; }
    void pushdown(int t) {
        if (mult[t] != 1) Mult(lc[t], mult[t]), Mult(rc[t], mult[t]), mult[t] = 1;
        if (plus[t]) Plus(lc[t], plus[t]), Plus(rc[t], plus[t]), plus[t] = 0;
    }
    int merge(int u, int v) {
        if (!u || !v) return u ^ v;
        if (val[u] > val[v]) swap(u, v);
        pushdown(u), rc[u] = merge(rc[u], v), swap(lc[u], rc[u]);
        return u;
    }
    int pop(int u) {
        pushdown(u);
        int t = merge(lc[u], rc[u]);
        lc[u] = rc[u] = 0;
        return t;
    }
}heap;
```

1.2 Splay

```cpp
int sz[MAXN], va[MAXN], ch[MAXN][2], flag[MAXN], n, m, cnt, rt;//flag为翻转标记
void maintain(int o) { sz[o] = sz[ch[o][0]] + sz[ch[o][1]] + 1; }
void pushdown(int x) {
    if (flag[x]) {
        flag[x] = 0; swap(ch[x][0], ch[x][1]);
        flag[ch[x][0]] ^= 1; flag[ch[x][1]] ^= 1;
    }
}
int build(int n) {
    if (!n) return 0;
    int lc = build(n >> 1);
    int now = ++ cnt;
    va[now] = now - 1;
    ch[now][0] = lc;
    ch[now][1] = build(n - (n >> 1) - 1);
    maintain(now);
    return now;
}
void Init() {
    n = read(); m = read();
    rt = build(n + 1);
}
int cmp(int x, int k) {
    if (k == sz[ch[x][0]] + 1) return -1;
    return k > sz[ch[x][0]];
}
void rotate(int &o, int d) {
    int k = ch[o][d ^ 1];
    ch[o][d ^ 1] = ch[k][d]; ch[k][d] = o;
    maintain(o); maintain(k); o = k;
}
void splay(int &o, int k) {
    pushdown(o);
    int d = cmp(o, k);
    if (d == -1) return ;
    if (d) k -= sz[ch[o][0]] + 1;
    int p = ch[o][d];
    pushdown(p);
    int d2 = cmp(p, k);
    if (d2 >= 0) {
        int k2 = d2 ? k - sz[ch[p][0]] - 1 : k;
        splay(ch[p][d2], k2);
        if (d == d2) rotate(o, d ^ 1); else rotate(ch[o][d], d);
    }
    rotate(o, d ^ 1);
}
int merge(int x, int y) {
    splay(x, sz[x]);
    ch[x][1] = y, maintain(x);
    return x;
}
void split(int o, int k, int &l, int &r) {
    splay(o, k), l = o, r = ch[o][1];
    ch[l][1] = 0, maintain(l);
}

void Solve() {
    while (m --) {
        int l, r, le, ri, md, o;
        l = read(); r = read();
        split(rt, l, le, o);
        split(o, r - l + 1, md, ri);
        flag[md] ^= 1;
        rt = merge(merge(le, md), ri);
    }
}
```

1.3 LCT

```
struct LCT {
   int fa[MAXN], ch[MAXN][2], rev[MAXN], xsum[MAXN];
   bool isrt(int x) { return x != ch[fa[x]][0] && x != ch[fa[x]][1]; }
   bool dir(int o) { return o != ch[fa[o]][0]; }
   void maintain(int o) { xsum[o] = xsum[ch[o][1]] ^ xsum[ch[o][0]] ^ w[o]; }
   void pushdown(int o) {
      if (rev[o]) rev[o] = 0; rev[ch[o][0]] ^= 1; rev[ch[o][1]] ^= 1; swap(ch[o][0], ch[o][1]);
   }
   void rotate(int o) {
      int f = fa[o], gf = fa[f], d = dir(o) ^ 1;
      fa[ch[o][d]] = f;
      ch[f][d ^ 1] = ch[o][d];
      fa[o] = gf;
      if (!isrt(f)) ch[gf][dir(f)] = o;
      fa[f] = o; ch[o][d] = f;
      maintain(f); maintain(o);
   }
   int sta[MAXN],top;
   void splay(int x) {
      sta[top = 1] = x;
      for (int t = x; !isrt(t); t = fa[t]) sta[++top] = fa[t];
      while (top) pushdown(sta[top --]);
      for ( ; !isrt(x); rotate(x)) if(!isrt(fa[x])) rotate(dir(fa[x]) == dir(x) ? fa[x] : x);
   }
   void access(int o) { for (int t = 0; o; t = o, o = fa[o]) splay(o), ch[o][1] = t, maintain(o); }
   void makeroot(int x) { access(x); splay(x); rev[x] ^= 1; }
   void link(int x, int y) { makeroot(x); fa[x] = y; }
   void cut(int x, int y) {
      makeroot(x); access(y); splay(y);
      if (ch[y][0] == x) ch[y][0] = 0, fa[x] = 0, maintain(y);
   }
   int findroot(int x) {
      access(x); splay(x);
      while (ch[x][0]) x = ch[x][0];
      return x;
   }
}lct;
void Init() {
   n = read(); m = read();
   For(i, 1, n) w[i] = lct.xsum[i] = read();
}
void Solve() {
   while (m --) {
      int op, x, y;
      op = read(); x = read(); y = read();
      if (!op) {
         lct.makeroot(x); lct.access(y); lct.splay(y);
         printf("%d\n", lct.xsum[y]);
      }
      else if (op == 1) { if (lct.findroot(x) != lct.findroot(y)) lct.link(x, y); }
      else if (op == 2) lct.cut(x, y);
      else { lct.access(x); lct.splay(x); w[x] = y; lct.maintain(x); }
   }
}
```

1.4 支配树

```
//洛谷模板题
void Dfs(int u) {
   id[dfn[u] = ++clk] = u;
   for (int v : G[u])
      if (!dfn[v])
         fa[v] = u, Dfs(v);
}
int find(int x) {
   if (f[x] == x) return x;
   int res = find(f[x]);
   if (dfn[sdom[ran[f[x]]]] < dfn[sdom[ran[x]]])
      ran[x] = ran[f[x]];
   return f[x] = res;
}
int main() {
   scanf("%d%d", &n, &m);
   for (int i = 1; i <= m; i++) {
      int u, v;
      scanf("%d%d", &u, &v);
      G[u].push_back(v);
      H[v].push_back(u);
   }
   Dfs(1);
   for (int i = 1; i <= n; i++)
      sdom[i] = f[i] = ran[i] = i;
   for (int i = clk; i > 1; i--) {
      int tmp = id[i];
      for (int v: H[tmp]) {
         if (!dfn[v])
            continue;
         find(v);
         if (dfn[sdom[ran[v]]] < dfn[sdom[tmp]])
            sdom[tmp] = sdom[ran[v]];
      }
      f[tmp] = fa[tmp];
      tr[sdom[tmp]].push_back(tmp);
      tmp = fa[tmp];
      for (int v: tr[tmp]) {
         find(v);
         if (tmp == sdom[ran[v]])
            idom[v] = tmp;
         else
            idom[v] = ran[v];
      }
      tr[tmp].clear();
   }
   for (int i = 2; i <= clk; i++) {
      int tmp = id[i];
      if (idom[tmp] ^ sdom[tmp])
         idom[tmp] = idom[idom[tmp]];
   }
   for (int i = clk; i > 1; i--)
      ans[idom[id[i]]] += ++ans[id[i]];
   ans[1]++;
   for (int i = 1; i <= n; i++)
      printf("%d ", ans[i]);
```

```
    return 0;
}
```

2 数论

2.1 Miller-Rabin & Pollard-Rho（含快速乘）

```
LL mult(LL a,LL b,LL p){
    LL d = (LL)floor(a * (LD)b / p + 0.5);
    LL ret = a * b - d * p;
    if (ret < 0) ret += p;
    return ret;
}
class MillerRabin {
    private:
        #define Pcnt 12
        const int P[Pcnt]={2,3,5,7,11,13,17,19,61,2333,4567,24251};

        LL fpm(LL x,LL y,LL X) {
            LL t=1;while(y) y&1&&(t=mult(t,x,X)),x=mult(x,x,X),y>>=1;
            return t;
        }
        int Check(LL x,int p) {
            if(!(x%p)||fpm(p%x,x-1,x)^1) return 0;
            LL k=x-1,t;
            while(!(k&1)) {
                if((t=fpm(p%x,k>>=1,x))^1&&t^(x-1)) return 0;
                if(!(t^(x-1))) return 1;
            }
            return 1;
        }
    public:
        int isP(LL x) {
            if(x<2) return false;
            for(int i=0;i^Pcnt;++i) {if(!(x^P[i])) return true;if(!Check(x,P[i])) return false;}
            return true;
        }
};
class PollardRho {
    private:
        #define Rand(x) (1LL*rand()*rand()%(x)+1)
        LL ans;
        MillerRabin MR;
        LL gcd(LL x,LL y) {return y?gcd(y,x%y):x;}
        LL Work(LL x,int y) {
            int t=0,k=1;
            LL v0=Rand(x-1),v=v0,d,s=1;
            for(;;) {
                if(v=(mult(v,v,x)+y)%x,s=mult(s,abs(v-v0),x),!(v^v0)||!s) return x;
                if(++t==k) {
                    if((d=gcd(s,x))^1) return d;
                    v0=v,k<<=1;
                }
            }
        }
        void Resolve(LL x,int t) {
            if (!(x^1)||x<=ans) return;
            if(MR.isP(x)) {
                if (ans < x) ans = x;
                return;
            }
            LL y=x;
            while((y=Work(x,t--))==x);
            while(!(x%y))x/=y;
            Resolve(x,t),Resolve(y,t);
        }
    public:
        PollardRho() {srand(1926);}
        LL GetMax(LL x) {return ans=0,Resolve(x,302627441),ans;}
}P;
```

2.2 二次剩余

```
struct field2{
    int x, y, a, p;
    field2():x(0), y(0), a(0), p(0){}
    field2(int x,int y,int a,int p):x(x),y(y),a(a),p(p){}
    field2 operator * (const field2 &f)const{
        int retx=(1ll * x * f.x + 1ll * y * f.y % p * a) % p;
        int rety=(1ll * x * f.y + 1ll * y * f.x) % p;
        return field2(retx, rety, a, p);
    }
    field2 fpm(int exp) const {
        field2 ret(1, 0, a, p), aux = *this;
        for ( ; exp > 0; exp >>= 1){
            if (exp & 1){
                ret = ret * aux;
            }
            aux = aux * aux;
        }
        return ret;
    }
};
std::vector <int> remain2(int a, int p){
    if (!a || p == 2) return {a};
    if (fpm(a, p - 1 >> 1, p) != 1) return {};
    if (p == 3) return {1, 2};
    while (true){
        field2 f(randint(p-1) + 1, randint(p - 1) + 1, a, p);
        f = f.fpm(p - 1 >> 1);
        if (f.x) continue;
        int ret = fpm(f.y, p - 2, p);
        return {min(ret, p - ret), max(ret, p - ret)};
    }
}
```

2.3 扩展欧几里得

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

2.4 欧拉函数

- 若 p 为素数，则 $\varphi(p) = p - 1$

若 $i \bmod p = 0$, 那么 $\varphi(i\times p) = p\times\varphi(i)$

若 $i \bmod p$ 不等于 0，那么 $\varphi(i\times p) = (p-1)\times\varphi(i)$

- 欧拉函数是积性函数，即当 a，b 互质时，$\varphi(a\times b) = \varphi(a)\times\varphi(b)$
- n 为奇数时，$\varphi(2\times a) = \varphi(a)$（原因:2n 为偶数，偶数和偶数一定不互质，所以只有 2n 与小于它的奇数互素的情况，则恰好就等于 n 的欧拉函数值）
- p 为素数时，$\varphi(p^a) = p^a - p^{a-1}$（原因: 一共有 p^a 个数，由于 p 为质数，所以与 p^a 不互素即包含质因子 p 的数的个数为 $(p^a)/p = p^{a-1}$，总数减去不互素的数即为 $\varphi(p^a) = p^a - p^{a-1}$
- 设 $p_1 \dots p_k$ 为 n 的质因数分解，则 $\varphi(x) = x(1-\frac{1}{p_1})(1-\frac{1}{p_2})\cdots(1-\frac{1}{p_k})$
- $\sum_{d|n}\varphi(d) = n$ （找到所有的 $gcd(i,n) = j$，发现满足 $gcd(t,n) = \frac{n}{d}(d|n)$ 的 t 有 $\varphi(d)$，然后发现可以不重复不遗漏地覆盖到所有 $gcd(i,n) = j$）
- 若 $n > 2$，那么 $\varphi(n)$ 是偶数
- 欧拉定理：若 $(a,n) = 1$, 则 $a^{\varphi(n)} \equiv 1 \pmod n$

  由欧拉定理得出另一个结论：设 m 是正整数，$(a,m) = 1$，则：$x \equiv ba^{\varphi(m)-1} \pmod m$ 是同余方程 $ax \equiv b \pmod m$ 的解
- 扩展欧拉定理：$a^x \equiv a^{x \bmod \varphi(p) + \varphi(p)[x>\varphi(p)]} \pmod p$

2.5 莫比乌斯反演

$$F(n) = \sum_{d|n} f(d) \Rightarrow f(n) = \sum_{d|n}\mu(d)F(\frac{n}{d})$$

$$F(n) = \sum_{n|d} f(d) \Rightarrow f(n) = \sum_{n|d}\mu(\frac{d}{n})F(d)$$

2.6 杜教筛

2.6.1 μ

求 $M(n) = \sum_{i=1}^{n}\mu(i)$

因为有性质 $\sum_{d|n}\mu(d) = [n=1]$，所以有：

$$1 = \sum_{i=1}^{n}\sum_{d|i}\mu(d) = \sum_{t=1}^{n}\sum_{d=1}^{\lfloor\frac{n}{t}\rfloor}\mu(d) = \sum_{i=1}^{n}M(\lfloor\frac{n}{i}\rfloor)$$

所以：$M(n) = 1 - \sum_{i=2}^{n}M(\lfloor\frac{n}{i}\rfloor)$，整除分块即可。

```
const int MAXN = 1000005, MOD = 1000007;
int mu[MAXN], Sum_mu[MAXN], prime[MAXN >> 1], cnt, np
    [MAXN], beg[MOD], nex[MOD], n, e;
LL n1, n2, v[MOD], w[MOD];
void add(int uu, LL vv, LL ww) { v[++ e] = vv, w[e] =
     ww, nex[e] = beg[uu], beg[uu] = e; }
void Get_mu() {
   mu[1] = 1;
   For(i, 2, n) {
      if (!np[i]) prime[++ cnt] = i, mu[i] = -1;
      for (int j = 1; j <= cnt && prime[j] * i <= n;
          ++ j) {
         np[i * prime[j]] = 1;
         if (!(i % prime[j])) {
            mu[i * prime[j]] = 0;
            break;
         } else mu[i * prime[j]] = -mu[i];
      }
   }
   For(i, 1, n) Sum_mu[i] = Sum_mu[i - 1] + mu[i];
}
LL Calc(LL x) {
   int tmp = x % MOD;
   if (x <= n) return Sum_mu[x];
   for (int i = beg[tmp]; i; i = nex[i]) if (v[i] ==
       x) return w[i];
   LL Ans = 1;
   for (LL l = 2, r; l <= x; l = r + 1)
      r = x / (x / l), Ans -= (r - l + 1) * 1ll *
          Calc(x / l);
   add(tmp, x, Ans);
   return Ans;
}
int main() {
   scanf("%lld%lld", &n1, &n2), n = (int)ceil(sqrt(n2
       * 1.0)) * 10;
   Get_mu();
   printf("%lld\n", Calc(n2) - Calc(n1 - 1));
   return 0;
}
```

2.6.2 φ

求 $S(n) = \sum_{i=1}^{n}\varphi(i)$

性质：$\sum_{d|n}\varphi(d) = n$

$$\sum_{i=1}^{n} i = \sum_{i=1}^{n}\sum_{d|i}\varphi(d) = \sum_{t=1}^{n}\sum_{d=1}^{\lfloor\frac{n}{t}\rfloor}\varphi(d) = \sum_{i=1}^{n}S(\lfloor\frac{n}{i}\rfloor)$$

所以：$S(n) = \sum_{i=1}^{n} i - \sum_{i=2}^{n}S(\lfloor\frac{n}{i}\rfloor)$

2.7 CRT 及扩展

2.7.1 CRT

$m_1, m_2, \dots$ 两两互质，$M = \prod m_i$

对于同余方程组：

$$\begin{cases} x \equiv c_1 \pmod{m_1} \\ x \equiv c_2 \pmod{m_2} \\ \dots \end{cases}$$

在模 M 意义下有唯一解。

令 $M_i = M/m_i$，则解 $x_0 \equiv \sum c_i \times M_i \times M_i^{-1} \pmod M$（$M_i^{-1}$ 指模 m_i 意义下的逆元，若 m_i 不是质数就只能用扩欧而不能用费马小定理求逆元）

2.7.2 exCRT

将同余方程写成不定方程的形式：

$x = c_1 + m_1 \times y_1$, $x = c_2 + m_2 \times y_2$

考虑合并以上两个方程。

易得：$c_1 + m_1 \times y_1 = c_2 + m_2 \times y_2$

移项得：$m_1 \times y_1 - m_2 \times y_2 = c_2 - c_1$

于是就可以用扩欧解决这个方程，求出 y_1 的最小正整数解并带入 $x_0 = c_1 + m_1 \times y_1$

然后就可以将两个方程合并为：$x \equiv x_0 \pmod{\mathrm{lca}(m_1, m_2)}$

2.8 Lucas 定理及扩展

2.8.1 Lucas 定理

$$\binom{n}{m} \bmod p = \binom{n \bmod p}{m \bmod p} \times \binom{\lfloor n/p \rfloor}{\lfloor m/p \rfloor} \bmod p$$

2.8.2 exLucas

对于 $C_n^m \bmod p$，我们可以令 $p=\prod_{i=1}^{q} p_i^{k_i}$，列出方程组：

$ans \equiv c_1 \pmod{p_1^{k_1}}$
$ans \equiv c_2 \pmod{p_2^{k_2}}$
$\cdots$
$ans \equiv c_q \pmod{p_q^{k_q}}$

由于 $p_1^{k_1}\cdots p_q^{k_q}$ 两两互质，所以可以直接用最基础的中国剩余定理合并。

接下来的问题是如何求出 $c_1 \cdots c_q$ 即 $C_n^m \bmod p_i^{k_i}$

我们要先分别求出 $n! \bmod p_i^{k_i}$，$m! \bmod p_i^{k_i}$，$(n-m)! \bmod p_i^{k_i}$ 的值，发现形式是差不多的，所以我们现在只研究 $n! \bmod p_i^{k_i}$

举这个例子：

假设 $n=22$，$p_i=3$，$k_i=2$

那么 $n!=1\times 2\times \cdots \times 22$

然后将其中是 3 的倍数的数提出来：

$=(1\times 2\times 4\times 5\times 7\times 8\times 10\times 11\times 13\times 14\times 16\times 17\times 19\times 20\times 22)\times 3^6\times(1\times 2\times 3\times 4\times 5\times 6\times 7)$

然后发现这个式子可以分成三部分：

1、$p_i^{k_i}$，这个可以直接快速幂

2、对于阶乘，我们可以递归求解

3、关键是怎么求第一部分的除去了 3 的倍数的数列的积

解决方法：

考虑将 $1\ldots n$ 分段，每 $p_i^{k_i}$ 个数为一段，并去除可以被三整除的数，可以发现一个性质：

$$(1\times 2\times 4\times 5\times 7\times 8)\equiv(10\times 11\times 13\times 14\times 16\times 17) \pmod{p_i^{k_i}}$$

然后对于剩下的数，一定不超过 $p_i^{k_i}$ 个，直接暴力求解即可。

另外，还有一个问题在计算除以 $m! \pmod{p_i^{k_i}}$，$(n-m)! \pmod{p_i^{k_i}}$ 时，当然需要乘以其关于模数的乘法逆元，但是如果它们不与模数互质，就无法直接求出逆元了。所以我们需要先将数中质因子 p_i 除去，求出逆元后再乘上来。

（注：计算 $n!$ 中质因子 p_i 的个数公式为：$x=\sum_{j=1}^{\infty}\lfloor\frac{n}{p_i^j}\rfloor$）

```
LL fac(LL n, LL p, LL pk) {
   if (!n) return 1;
   LL res = 1;
   for (LL i = 2; i <= pk; ++ i)
      if (i % p) (res *= i) %= pk;
   res = fpm(res, n / pk, pk);
   for (LL i = 2; i <= n % pk; ++ i)
      if (i % p) (res *= i) %= pk;
   return res * fac(n / p, p, pk) % pk;
}
LL inv(LL n, LL Mod) {
   static LL x, y, t;
   gcd(n, Mod, x, y);
   t = ((x % Mod) + Mod) % Mod;
   return t;
}
LL C(LL n, LL m, LL p, LL k, LL pk) {
   if (n < m) return 0;
   LL t1 = fac(n, p, pk), t2 = fac(m, p, pk), t3 = fac(n - m, p, pk), cnt = 0;
   for (LL i = n; i; i /= p) cnt += i / p;
   for (LL i = m; i; i /= p) cnt -= i / p;
   for (LL i = n - m; i; i /= p) cnt -= i / p;
   return t1 * inv(t2, pk) % pk * inv(t3, pk) % pk * fpm(p, cnt, pk) % pk;
}
LL CRT(LL c, LL m) { return c * inv(p / m, m) % p * (p / m) % p; }
LL exLucas(LL n, LL m) {
   LL Ans = 0, tmp = p;
   for (int i = 2; i * i <= tmp; ++ i)
      if (!(tmp % i)) {
         LL cnt = 0, prod = 1;
         while (!(tmp % i)) tmp /= i, prod *= i, ++cnt;
         (Ans += CRT(C(n, m, i, cnt, prod), prod)) %= p;
      }
   if (tmp > 1) (Ans += CRT(C(n, m, tmp, 1, tmp), tmp)) %= p;
   return Ans;
}
```

2.9 原根

对 $\varphi(p)$ 进行质因数分解，若恒有 $g^{\varphi(p)/p_i} \not\equiv 1 \pmod p$，则 g 为 p 的原根。

3 组合数学

3.1 二项式反演

$$f(n)=\sum_{k=p}^{n}\binom{n}{k}g(k)$$

$$g(n)=\sum_{k=p}^{n}(-1)^{n-k}\binom{n}{k}f(k)$$

3.2 斯特林数

3.2.1 第一类斯特林数

$$\begin{bmatrix}n\\m\end{bmatrix}=\begin{bmatrix}n-1\\m-1\end{bmatrix}+(n-1)\begin{bmatrix}n-1\\m\end{bmatrix}$$

3.2.2 第二类斯特林数

递推：$\left\{{n \atop m}\right\}=\left\{{n-1 \atop m-1}\right\}+m\left\{{n-1 \atop m}\right\}$

容斥：

$$\begin{Bmatrix}n\\m\end{Bmatrix}=\frac{1}{m!}\sum_{k=0}^{m}(-1)^k\binom{m}{k}(m-k)^n$$

关于容斥的理解：枚举空盒子的个数，其它的随便乱放，由于盒子是相同的，所以要除以 $m!$。

整理得到：

$$\begin{Bmatrix}n\\m\end{Bmatrix}=\sum_{k=0}^{m}(-1)^k\times\frac{1}{k!}\times\frac{(m-k)^n}{(m-k)!}$$

可以用 NTT 求解所有的 $\left\{{n \atop i}\right\}$。

重要性质：

$$n^k=\sum_{i=0}^{k}\begin{Bmatrix}k\\i\end{Bmatrix}\binom{n}{i}i!$$

理解：左边是将 k 个球放在 n 个盒子里；右边枚举非空盒子的个数，从 n 个盒子中选出 i 个，将 k 个球放在这 i 个盒子里，由于盒子是不同的，所有要乘 $i!$。这个式子还能写成：

$$n^k=\sum_{i=1}^{k}\begin{Bmatrix}k\\i\end{Bmatrix}n^{\underline{i}}$$

第二类斯特林数的展开式：

$$\begin{Bmatrix}n\\m\end{Bmatrix}m!=\sum_{k=0}^{m}(-1)^k\binom{m}{k}(m-k)^n$$

理解：左边是将 n 个数分成 m 个集合且集合有序的方案数；右边 k 枚举至少多少个集合是空集，然后在 m 个集合中选 k 个成为空集，n 个数乱放在剩下的集合中。

3.2.3 斯特林数反演

$$f(n)=\sum_{i=1}^{n}\begin{Bmatrix}n\\i\end{Bmatrix}g(i)$$

$$g(n)=\sum_{i=1}^{n}(-1)^{n-i}\begin{bmatrix}n\\i\end{bmatrix}f(i)$$

3.3 其它

3.3.1 Matrix-Tree 定理

G 的度数矩阵 D_G 是一个 $n\times n$ 的矩阵，当 $i\neq j$ 时，$D_{i,j}=0$；$D_{i,i}$ 的值为节点度数。

G 的邻接矩阵 A_G 也是一个 $n\times n$ 的矩阵，当 i,j 直接相连时，$A_{i,j}=1$、否则为 0。

我们定义 Kirchhoff 矩阵（也叫拉普拉斯算子）为 $C_G=D_G-A_G$，则 Matrix-Tree 定理可描述为：图 G 的所有不同生成树的个数等于其 Kirchhoff 矩阵 C_G 任何一个 $n-1$ 阶主子式的行列式的绝对值。（所谓 $n-1$ 阶主子式，即对于 r（$1\le r\le n$），将 C_G 的第 r 行、第 r 列同时去掉后得到的新矩阵）

3.3.2 Best 定理

对于一个有向图，其欧拉回路的个数等于以起点为根的树形图的个数乘以每个点度数（入度必须等于出度）减 1 的阶乘。

至于树形图个数，仍然可以用 Kirchhoff 矩阵计算：度数矩阵改为入度、$n-1$ 阶主子式只能去掉根的那一阶。

3.3.3 错排公式

f(x) = x(f(x-1)+f(x-2)) 初始化：$f(0)=1,f(1)=0,f(2)=1$

3.3.4 皮克定理

3.3.5 Catalan 数

$C_0=1$

$C_n=\sum_{i=0}^{n-1}C_iC_{n-i-1}=\binom{2n}{n}\frac{1}{n+1}=\frac{(2n)!}{(n+1)!n!}=\binom{2n}{n}-\binom{2n}{n-1}=\frac{4n-2}{n+1}C_{n-1}=\prod_{k=2}^{n}\frac{n+k}{k}$

4 计算几何

```cpp
#define PI 3.14159265358979323846 26
const double EPS = 1e-8;
using namespace std;

#define Vector Point

#define ChongHe 0
#define NeiHan 1
#define NeiQie 2
#define XiangJiao 3
#define WaiQie 4
#define XiangLi 5

int dcmp(double x) { return fabs(x) < EPS ? 0 : (x < 0 ? -1 : 1); }

struct Point {
    double x, y;

    Point(const Point& rhs): x(rhs.x), y(rhs.y) { } //拷贝构造函数
    Point(double x = 0.0, double y = 0.0): x(x), y(y) { } //构造函数

    friend istream& operator >> (istream& in, Point& P) { return in >> P.x >> P.y; }
    friend ostream& operator << (ostream& out, const Point& P) { return out << P.x << ' ' << P.y; }

    friend Vector operator + (const Vector& A, const Vector& B) { return Vector(A.x+B.x, A.y+B.y); }
    friend Vector operator - (const Point& A, const Point& B) { return Vector(A.x-B.x, A.y-B.y); }
    friend Vector operator * (const Vector& A, const double& p) { return Vector(A.x*p, A.y*p); }
    friend Vector operator / (const Vector& A, const double& p) { return Vector(A.x/p, A.y/p); }
    friend bool operator == (const Point& A, const Point& B) { return dcmp(A.x-B.x) == 0 && dcmp(A.y-B.y) == 0; }
    friend bool operator < (const Point& A, const Point& B) { return A.x < B.x || (A.x == B.x && A.y < B.y); }

    void in(void) { scanf("%lf%lf", &x, &y); }
    void out(void) { printf("%lf %lf", x, y); }
};

struct Line {
    Point P; //直线上一点
    Vector dir; //方向向量(半平面交中该向量左侧表示相应的半平面)
    double ang; //极角，即从x正半轴旋转到向量dir所需要的角（弧度）

    Line() { } //构造函数
    Line(const Line& L): P(L.P), dir(L.dir), ang(L.ang) { }
    Line(const Point& P, const Vector& dir): P(P), dir(dir) { ang = atan2(dir.y, dir.x); }

    bool operator < (const Line& L) const { //极角排序
        return ang < L.ang;
    }

    Point point(double t) { return P + dir*t; }
};

typedef vector<Point> Polygon;

struct Circle {
    Point c; //圆心
    double r; //半径

    Circle() { }
    Circle(const Circle& rhs): c(rhs.c), r(rhs.r) { }
    Circle(const Point& c, const double& r): c(c), r(r) { }

    Point point(double ang) const { return Point(c.x +
```

```cpp
        cos(ang)*r, c.y + sin(ang)*r); } //圆心角所对应的点
    double area(void) const { return PI * r * r; }
};

double Dot(const Vector& A, const Vector& B) { return A.x*B.x + A.y*B.y; } //点积
double Length(const Vector& A){ return sqrt(Dot(A, A)); }
double Angle(const Vector& A, const Vector& B) { return acos(Dot(A, B)/Length(A)/Length(B)); } //向量夹角
double Cross(const Vector& A, const Vector& B) { return A.x*B.y - A.y*B.x; } //叉积
double Area(const Point& A, const Point& B, const Point& C) { return fabs(Cross(B-A, C-A)); }

//三边构成三角形的判定
bool check_length(double a, double b, double c) {
    return dcmp(a+b-c) > 0 && dcmp(fabs(a-b)-c) < 0;
}
bool isTriangle(double a, double b, double c) {
    return check_length(a, b, c) && check_length(a, c, b) && check_length(b, c, a);
}

//平行四边形的判定（保证四边形顶点按顺序给出）
bool isParallelogram(Polygon p) {
    if (dcmp(Length(p[0]-p[1]) - Length(p[2]-p[3])) || dcmp(Length(p[0]-p[3]) - Length(p[2]-p[1]))) return false;
    Line a = Line(p[0], p[1]-p[0]);
    Line b = Line(p[1], p[2]-p[1]);
    Line c = Line(p[3], p[2]-p[3]);
    Line d = Line(p[0], p[3]-p[0]);
    return dcmp(a.ang - c.ang) == 0 && dcmp(b.ang - d.ang) == 0;
}

//梯形的判定
bool isTrapezium(Polygon p) {
    Line a = Line(p[0], p[1]-p[0]);
    Line b = Line(p[1], p[2]-p[1]);
    Line c = Line(p[3], p[2]-p[3]);
    Line d = Line(p[0], p[3]-p[0]);
    return (dcmp(a.ang - c.ang) == 0 && dcmp(b.ang - d.ang)) || (dcmp(a.ang - c.ang) && dcmp(b.ang - d.ang) == 0);
}

//菱形的判定
bool isRhombus(Polygon p) {
    if (!isParallelogram(p)) return false;
    return dcmp(Length(p[1]-p[0]) - Length(p[2]-p[1])) == 0;
}

//矩形的判定
bool isRectangle(Polygon p) {
    if (!isParallelogram(p)) return false;
    return dcmp(Length(p[2]-p[0]) - Length(p[3]-p[1])) == 0;
}

//正方形的判定
bool isSquare(Polygon p) {
    return isRectangle(p) && isRhombus(p);
}

//三点共线的判定
bool isCollinear(Point A, Point B, Point C) {
    return dcmp(Cross(B-A, C-B)) == 0;
}

//向量绕起点旋转
Vector Rotate(const Vector& A, const double& rad) {
    return Vector(A.x*cos(rad)-A.y*sin(rad), A.x*sin(rad)+A.y*cos(rad));
}

//向量的单位法线(调用前请确保A 不是零向量)
Vector Normal(const Vector& A) {
    double len = Length(A);
    return Vector(-A.y / len, A.x / len);
}

//两直线交点(用前确保两直线有唯一交点，当且仅当Cross(A.dir, B.dir)非0)
Point GetLineIntersection(const Line& A, const Line& B) {
    Vector u = A.P - B.P;
    double t = Cross(B.dir, u) / Cross(A.dir, B.dir);
    return A.P + A.dir*t;
}

//点到直线距离
double DistanceToLine(const Point& P, const Line& L) {
    Vector v1 = L.dir, v2 = P - L.P;
    return fabs(Cross(v1, v2)) / Length(v1);
}

//点到线段距离
double DistanceToSegment(const Point& P, const Point& A, const Point& B) {
    if (A == B) return Length(P - A);
    Vector v1 = B - A, v2 = P - A, v3 = P - B;
    if (dcmp(Dot(v1, v2)) < 0) return Length(v2);
    if (dcmp(Dot(v1, v3)) > 0) return Length(v3);
    return fabs(Cross(v1, v2)) / Length(v1);
}

//点在直线上的投影
Point GetLineProjection(const Point& P, const Line& L) { return L.P + L.dir*(Dot(L.dir, P - L.P)/Dot(L.dir, L.dir)); }

//点在线段上的判定
bool isOnSegment(const Point& P, const Point& A, const Point& B) {
    //若允许点与端点重合，可关闭下面的注释
    //if (P == A || P == B) return true;
    // return dcmp(Cross(A-P, B-P)) == 0 && dcmp(Dot(A-P, B-P)) < 0;
    return dcmp(Length(P-A) + Length(B-P) - Length(A-B)) == 0;
}

//线段相交判定
bool SegmentProperIntersection(const Point& a1, const Point& a2, const Point& b1, const Point& b2) {
```

```cpp
    //若允许在端点处相交，可适当关闭下面的注释
    //if (isOnSegment(a1, b1, b2) || isOnSegment(a2, b1, b2) || isOnSegment(b1, a1, a2) || isOnSegment(b2, a1, a2)) return true;
    double c1 = Cross(a2-a1, b1-a1), c2 = Cross(a2-a1, b2-a1);
    double c3 = Cross(b2-b1, a1-b1), c4 = Cross(b2-b1, a2-b1);
    return dcmp(c1)*dcmp(c2) < 0 && dcmp(c3)*dcmp(c4) < 0;
}

//多边形的有向面积
double PolygonArea(Polygon po) {
    int n = po.size();
    double area = 0.0;
    for(int i = 1; i < n-1; i++) {
        area += Cross(po[i]-po[0], po[i+1]-po[0]);
    }
    return area * 0.5;
}

//点在多边形内的判定(多边形顶点需按逆时针排列)
bool isInPolygon(const Point& p, const Polygon& poly) {
    int n = poly.size();
    for(int i = 0; i < n; i++) {
        //若允许点在多边形边上，可关闭下行注释
        // if (isOnSegment(p, poly[(i+1)%n], poly[i])) return true;
        if (Cross(poly[(i+1)%n]-poly[i], p-poly[i]) < 0) return false;
    }
    return true;
}

//过定点作圆的切线
int getTangents(const Point& P, const Circle& C, std::vector<Line>& L) {
    Vector u = C.c - P;
    double dis = Length(u);
    if (dcmp(dis - C.r) < 0) return 0;
    if (dcmp(dis - C.r) == 0) {
        L.push_back(Line(P, Rotate(u, PI / 2.0)));
        return 1;
    }
    double ang = asin(C.r / dis);
    L.push_back(Line(P, Rotate(u, ang)));
    L.push_back(Line(P, Rotate(u, -ang)));
    return 2;
}

//直线和圆的交点
int GetLineCircleIntersection(Line& L, const Circle& C, vector<Point>& sol) {
    double t1, t2;
    double a = L.dir.x, b = L.P.x - C.c.x, c = L.dir.y, d = L.P.y - C.c.y;
    double e = a*a + c*c, f = 2.0*(a*b + c*d), g = b*b + d*d - C.r*C.r;
    double delta = f*f - 4*e*g; //判别式
    if (dcmp(delta) < 0) return 0; //相离
    if (dcmp(delta) == 0) { //相切
     t1 = t2 = -f / (2 * e);
     sol.push_back(L.point(t1));
     return 1;
    }
    t1 = (-f - sqrt(delta)) / (2.0 * e); sol.push_back(L.point(t1)); // 相交
    t2 = (-f + sqrt(delta)) / (2.0 * e); sol.push_back(L.point(t2));
    return 2;
}

//两圆位置关系判定
int GetCircleLocationRelation(const Circle& A, const Circle& B) {
    double d = Length(A.c-B.c);
    double sum = A.r + B.r;
    double sub = fabs(A.r - B.r);
    if (dcmp(d) == 0) return dcmp(sub) != 0;
    if (dcmp(d - sum) > 0) return XiangLi;
    if (dcmp(d - sum) == 0) return WaiQie;
    if (dcmp(d - sub) > 0 && dcmp(d - sum) < 0) return INTERSECTING;
    if (dcmp(d - sub) == 0) return NeiQie;
    if (dcmp(d - sub) < 0) return NeiHan;
}

//两圆相交的面积
double GetCircleIntersectionArea(const Circle& A, const Circle& B) {
    int rel = GetCircleLocationRelation(A, B);
    if (rel < INTERSECTING) return min(A.area(), B.area());
    if (rel > INTERSECTING) return 0;
    double dis = Length(A.c - B.c);
    double ang1 = acos((A.r*A.r + dis*dis - B.r*B.r) / (2.0*A.r*dis));
    double ang2 = acos((B.r*B.r + dis*dis - A.r*A.r) / (2.0*B.r*dis));
    return ang1*A.r*A.r + ang2*B.r*B.r - A.r*dis*sin(ang1);
}

//凸包(Andrew算法)
//如果不希望在凸包的边上有输入点，把两个 <= 改成 <
//如果不介意点集被修改，可以改成传递引用
Polygon ConvexHull(vector<Point> p) {
    //预处理，删除重复点
    sort(p.begin(), p.end());
    p.erase(unique(p.begin(), p.end()), p.end());
    int n = p.size(), m = 0;
    Polygon res(n+1);
    for(int i = 0; i < n; i++) {
        while(m > 1 && Cross(res[m-1]-res[m-2], p[i]-res[m-2]) <= 0) m--;
        res[m++] = p[i];
    }
    int k = m;
    for(int i = n-2; i >= 0; i--) {
        while(m > k && Cross(res[m-1]-res[m-2], p[i]-res[m-2]) <= 0) m--;
        res[m++] = p[i];
    }
    m -= n > 1;
    res.resize(m);
    return res;
}
```

```cpp
//点P在有向直线L左边的判定(线上不算)
bool isOnLeft(const Line& L, const Point& P) {
    return Cross(L.dir, P-L.P) > 0;
}

//半平面交主过程
//如果不介意点集被修改，可以改成传递引用
Polygon HalfPlaneIntersection(vector<Line> L) {
    int n = L.size();
    int head, rear; //双端队列的第一个元素和最后一个元素的下标
    vector<Point> p(n); //p[i]为q[i]和q[i+1]的交点
    vector<Line> q(n); //双端队列
    Polygon ans;

    sort(L.begin(), L.end()); //按极角排序
    q[head=rear=0] = L[0]; //双端队列初始化为只有一个半平面L[0]
    for(int i = 1; i < n; i++) {
        while(head < rear && !isOnLeft(L[i], p[rear-1])) rear--;
        while(head < rear && !isOnLeft(L[i], p[head])) head++;
        q[++rear] = L[i];
        if (fabs(Cross(q[rear].dir, q[rear-1].dir)) < EPS) { //两向量平行且同向，取内侧的一个
            rear--;
            if (isOnLeft(q[rear], L[i].P)) q[rear] = L[i];
        }
        if (head < rear) p[rear-1] = GetLineIntersection(q[rear-1], q[rear]);
    }
    while(head < rear && !isOnLeft(q[head], p[rear-1])) rear--; //删除无用平面
    if (rear - head <= 1) return ans; //空集
    p[rear] = GetLineIntersection(q[rear], q[head]); //计算首尾两个半平面的交点

    for(int i = head; i <= rear; i++) { //从deque复制到输出中
        ans.push_back(p[i]);
    }
    return ans;
}
```

5 图论

5.1 点双

```cpp
void dfs(int u, int fa) {
    int chs = 0;
    dfn[u] = low[u] = ++ tim;
    for (int i = beg[u]; i; i = nex[i]) if (v[i] != fa)
    {
        tmp = mp(u, v[i]);
        if (!dfn[v[i]]) {
            stk.push(tmp), ++ chs;
            dfs(v[i], u), chkmin(low[u], low[v[i]]);
            if (low[v[i]] >= dfn[u])
            {
                iscut[u] = 1;
                ++ bccs, bcc[bccs].clear();
                for ( ; ; ) {
                    tmp = stk.top(), stk.pop();
                    if (co[tmp.x] != bccs) co[tmp.x] = bccs, bcc[bccs].pb(tmp.x);
                    if (co[tmp.y] != bccs) co[tmp.y] = bccs, bcc[bccs].pb(tmp.y);
                    if (u == tmp.x && v[i] == tmp.y)
                        break;
                }
            }
        } else if (dfn[v[i]] < dfn[u])
            stk.push(tmp), chkmin(low[u], dfn[v[i]]);
    }
    if (!fa && chs == 1) iscut[u] = 0;
}
```

5.2 边双

```cpp
int dfn[MAXN], low[MAXN], clk, stk[MAXN], top, co[MAXN], cnt;
void pop(){ int u=stk[top--]; co[u]=cnt; }
void Tarjan(int u, int pa) {
    dfn[u] = low[u] = ++ clk, stk[++ top] = u;
    for (int i = beg[u]; i; i = nex[i]) if ((i >> 1) != pa) {
        if (!dfn[v[i]]) Tarjan(v[i], i >> 1), low[u] = min(low[u], low[v[i]]);
        else low[u] = min(low[u], low[v[i]]);
    }
    if (dfn[u] == low[u]) for (++ cnt; co[stk[top]] = cnt, stk[top --] != u; );
}
```

5.3 虚树

```cpp
bool cmp(const int& a, const int& b) { return dfn[a] < dfn[b]; }
//每次建树前记得清零
For(i, 1, tot) iskey[s[i] = read()] = 1;
if (!iskey[1]) s[++ tot] = 1;
sort(s + 1, s + 1 + tot, cmp);
stk[top = 1] = 1, e_ = 0;
for (int i = 2; i <= tot; ++ i)
{
    int u = s[i], lca = LCA(u, stk[top]);
    if (lca != stk[top])
    {
        while (top > 1 && dep[stk[top - 1]] >= dep[lca])
            add_(stk[top - 1], stk[top]), -- top;
        if (stk[top] != lca) add_(lca, stk[top]), stk[top] = lca;
    }
    stk[++ top] = u;
}
Fordown(i, top, 2) add_(stk[i - 1], stk[i]);
```

5.4 仙人掌圆方树

```cpp
//[BZOJ2125]求仙人掌上的最短路
```

```cpp
void add1(int uu, int vv, int ww) { v1[++ e1] = vv, w1[e1] = ww, nex1[e1] = beg1[uu], beg1[uu] = e1; }
void add2(int uu, int vv, LL ww) { v2[++ e2] = vv, w2[e2] = ww, nex2[e2] = beg2[uu], beg2[uu] = e2; }
void Build(int u, int anc, LL dsum) {
   static LL d; ++ tot;
   for (int t = u; t != fa1[anc]; t = fa1[t])
      cir[t] = dsum, d = min(dis1[t] - dis1[anc], dsum - dis1[t] + dis1[anc]), add2(tot, t, d), add2(t, tot, d);
}
void DFS(int u, int fe) {
   dfn[u] = low[u] = ++ clk;
   for (int i = beg1[u]; i; i = nex1[i]) if ((i >> 1) != fe) {
      if (!dfn[v1[i]]) dis1[v1[i]] = dis1[u] + w1[i], fa1[v1[i]] = u, DFS(v1[i], i >> 1), chkmin(low[u], low[v1[i]]);
      else chkmin(low[u], dfn[v1[i]]);
      if (dfn[u] < low[v1[i]]) add2(u, v1[i], w1[i]), add2(v1[i], u, w1[i]);
   }
   for (int i = beg1[u]; i; i = nex1[i])
      if (fa1[v1[i]] != u && dfn[v1[i]] > dfn[u])
         Build(v1[i], u, w1[i] + dis1[v1[i]] - dis1[u]);
}
void DFS(int u) {
   for (int i = beg2[u]; i; i = nex2[i]) if (v2[i] != fa2[0][u])
      fa2[0][v2[i]] = u, dis2[v2[i]] = dis2[u] + w2[i], dep[v2[i]] = dep[u] + 1, DFS(v2[i]);
}
int LCA(int u, int v) {
   if (dep[u] < dep[v]) swap(u, v);
   Fordown(i, 14, 0) if (dep[v] + (1 << i) <= dep[u]) u = fa2[i][u];
   if (u == v) return u;
   Fordown(i, 14, 0) if (fa2[i][u] != fa2[i][v]) u = fa2[i][u], v = fa2[i][v];
   return fa2[0][u];
}
int climb(int u, int anc) {
   Fordown(i, 14, 0) if (dep[fa2[i][u]] > dep[anc]) u = fa2[i][u];
   return u;
}
LL dist(int u, int v) {
   static LL t; t = llabs(dis1[u] - dis1[v]);
   assert(cir[u] == cir[v]);
   return min(t, cir[u] - t);
}
int main() {
   static int m, q, uu, vv, ww, lca, ua, va;
   tot = n = read(), m = read(), q = read();
   while (m --) uu = read(), vv = read(), ww = read(), add1(uu, vv, ww), add1(vv, uu, ww);
   DFS(1, 0), DFS(1);
   For(j, 1, 14) For(i, 1, tot) fa2[j][i] = fa2[j - 1][fa2[j - 1][i]];
   while (q --) {
      uu = read(), vv = read(), lca = LCA(uu, vv);
      if (lca <= n) printf("%lld\n", dis2[uu] + dis2[vv] - (dis2[lca] << 1));
      else ua = climb(uu, lca), va = climb(vv, lca), printf("%lld\n", dis2[uu] + dis2[vv] - dis2[ua] - dis2[va] + dist(ua, va));
   }
   return 0;
}
```

5.5 一般图圆方树

```cpp
//[APIO2018]铁人两项
int n, e = 1, beg1[maxn], beg2[maxn << 1], nex[maxm << 1], v[maxm << 1], sz[maxn << 2], dfn[maxn], low[maxn], clk, w[maxn << 1], tot, all, stk[maxn], top, cnt;
LL Ans;
void add1(int uu, int vv) { v[++ e] = vv, nex[e] = beg1[uu], beg1[uu] = e; }
void add2(int uu, int vv) { v[++ e] = vv, nex[e] = beg2[uu], beg2[uu] = e; }
void DFS(int u, int fe) {
   dfn[u] = low[u] = ++ clk, stk[++ top] = u, w[u] = -1;
   for (int i = beg1[u]; i; i = nex[i]) if ((i >> 1) != fe) {
      if (!dfn[v[i]]) {
         DFS(v[i], i >> 1), chkmin(low[u], low[v[i]]);
         if (low[v[i]] >= dfn[u]) {
            add2(u, ++ tot), cnt = 1;
            do add2(tot, stk[top]), ++ cnt; while (stk[top --] != v[i]);
            w[tot] = cnt;
         }
      } else chkmin(low[u], dfn[v[i]]);
   }
}
void getsz(int u) {
   sz[u] = u <= n;
   for (int i = beg2[u]; i; i = nex[i])
      getsz(v[i]), sz[u] += sz[v[i]];
}
void DP(int u) {
   int pre = u <= n;
   for (int i = beg2[u]; i; i = nex[i])
      DP(v[i]), Ans += (LL)pre * w[u] * sz[v[i]], pre += sz[v[i]];
   Ans += (LL)sz[u] * (all - sz[u]) * w[u];
}
int main() {
   static int m, uu, vv;
   tot = n = read(), m = read();
   while (m --) uu = read(), vv = read(), add1(uu, vv), add1(vv, uu);
   For(i, 1, n) if (!dfn[i]) DFS(i, 0), getsz(i), all = sz[i], DP(i);
   printf("%lld\n", Ans << 1);
   return 0;
}
```

5.6 网络流

```cpp
namespace MF {
```

```
    int e = 1, f[MAXM << 1], v[MAXM << 1], beg[MAXN],
         nex[MAXM << 1], S, T;
    void add(int uu, int vv, int ff) {
        v[++ e] = vv, f[e] = ff, nex[e] = beg[uu], beg
             [uu] = e;
        v[++ e] = uu, f[e] = 0, nex[e] = beg[vv], beg[
             vv] = e;
    }
    void init() {
        S = n + 1, T = n + 2;
        //add edges...
    }
    int lev[MAXN], beg1[MAXN];
    bool BFS() {
        static queue<int> q;
        memset(lev, -1, sizeof lev);
        while (!q.empty()) q.pop();
        for (lev[S] = 0, q.push(S); !q.empty(); q.pop
             ()) {
            int u = q.front();
            for (int i = beg[u]; i; i = nex[i])
                if (f[i] && lev[v[i]] == -1) {
                    lev[v[i]] = lev[u] + 1, q.push(v[i])
                         ;
                }
        }
        return lev[T] != -1;
    }
    int DFS(int u, int flow) {
        if (u == T) return flow;
        int res = flow;
        for (int &i = beg1[u]; i; i = nex[i]) {
            if (lev[v[i]] == lev[u] + 1 && f[i]) {
                int t = DFS(v[i], min(res, f[i]));
                f[i] -= t, f[i ^ 1] += t;
                if (!(res -= t)) return flow;
            }
        }
        return flow - res;
    }
    int main() {
        int FLOW = 0;
        while (BFS()) memcpy(beg1, beg, sizeof beg),
             FLOW += DFS(S, 2);
        return FLOW;
    }
}
```

5.7 费用流

```
//记得反向流是负边权
int BFS() {
    static deque<int> q;
    For(i, 1, n) dis[i] = INF; Set(vis, 0);
    dis[s] = 0; vis[s] = 1; q.pb(s);
    while (!q.empty()) {
        int u = q.front(); q.pop_front(); vis[u] = 0;
        for (int i = beg[u]; i; i = nex[i])
            if (f[i] && chkmin(dis[v[i]], dis[u] + w[i
                 ]))
                if (!vis[v[i]]) {
                    vis[v[i]] = 1;
                    if (!q.empty() && dis[q.front()] >
                         dis[v[i]]) q.pf(v[i]); else q.pb
                              (v[i]);
                }
    }
    return dis[t] != INF;
}
int DFS(int u, int flow) {
    if (u == t) return flow;
    vis[u] = 1;
    int res = flow, tmp;
    for (int i = beg[u]; i; i = nex[i]) {
        if (vis[v[i]] || !f[i] || dis[v[i]] != dis[u]
             + w[i]) continue;
        tmp = DFS(v[i], min(f[i], res));
        f[i] -= tmp; f[i ^ 1] += tmp; Cost += tmp * w[
             i];
        if (!(res -= tmp)) return flow;
    }
    return flow - res;
}
void MCMF() {
    while (BFS()) {
        vis[t] = 1;
        while (vis[t]) Set(vis, 0), Flow += DFS(s, INF
             );
    }
}
```

5.8 匈牙利算法

```
int DFS(int u){
    For(i,1,m)if(G[u][i]&&!vis[i]){
        vis[i]=1;
        if(!mat[i]||DFS(mat[i])){mat[i]=u;return 1;}
    }
    return 0;
}
For(i,1,n){
    memset(vis,0,sizeof(vis));
    if(DFS(i))++ans;
}
```

5.9 带花树

```
int n, m, v[maxm << 1], e, nex[maxm << 1], beg[maxn],
     clk, fa[maxn], pre[maxn], mat[maxn], Ans, tim[
     maxn], vis[maxn];
queue<int> q;
void add(int uu, int vv) { v[++ e] = vv, nex[e] = beg
     [uu], beg[uu] = e; }
int find(int x) { return fa[x] == x ? x : fa[x] =
     find(fa[x]); }
int LCA(int u, int v) {
    for (++ clk;; swap(u, v)) if (u) {
        u = find(u);
        if (tim[u] == clk) return u;
        tim[u] = clk, u = pre[mat[u]];
    }
}
void blossom(int u, int v, int lca) {
    while (find(u) != lca) {
        pre[u] = v, v = mat[u];
        if (vis[v] == 2) vis[v] = 1, q.push(v);
```

```
        if (find(u) == u) fa[u] = lca;
        if (find(v) == v) fa[v] = lca;
        u = pre[v];
    }
}
int BFS(int s) {
    For(i, 1, n) fa[i] = i;
    Set(vis, 0), Set(pre, 0);
    while (!q.empty()) q.pop();
    q.push(s), vis[s] = 1;
    while (!q.empty()) {
        int u = q.front();
        q.pop();
        for (int i = beg[u]; i; i = nex[i]) {
            if (find(u) == find(v[i]) || vis[v[i]] == 2) continue;
            if (!vis[v[i]]) {
                vis[v[i]] = 2, pre[v[i]] = u;
                if (!mat[v[i]]) {
                    for (int t = v[i], las; t; t = las)
                        las = mat[pre[t]], mat[t] = pre[t], mat[pre[t]] = t;
                    return 1;
                }
                vis[mat[v[i]]] = 1, q.push(mat[v[i]]);
            } else {
                int lca = LCA(u, v[i]);
                blossom(u, v[i], lca), blossom(v[i], u, lca);
            }
        }
    }
    return 0;
}
For(i, 1, n) if (!mat[i]) Ans += BFS(i);
```

6 字符串

6.1 KMP

```
void getNext() {
    nex[0] = 0;
    For(i, 1, lent - 1) {
        int j = nex[i - 1] - 1;
        while (~j && T[j + 1] != T[i]) j = nex[j] - 1;
        if (T[j + 1] == T[i]) nex[i] = j + 2;
        else nex[i] = 0;
    }
}
void getPos() {
    int j = -1;
    Rep(i, lens) {
        while (~j && T[j + 1] != S[i]) j = nex[j] - 1;
        if (T[j + 1] == S[i]) {
            ++ j;
            if (j == lent - 1)
                printf("%d\n", i - lent + 2), j = nex[j] - 1;
        }
    }
}
```

6.2 AC 自动机

```
struct Trie {
    int ids, ch[MAXN][26], fail[MAXN], cnt[MAXN], dep[MAXN];
    vector<int> id[MAXN];
    Trie() { ids = 1; }
    void insert(char *str, int nid) {
        int len = strlen(str), u = 1;
        Rep(i, len) {
            int c = str[i] - 97;
            if (ch[u][c]) u = ch[u][c];
            else u = ch[u][c] = ++ ids;
        }
        id[u].PB(nid);
    }
    void init() {
        static queue<int> q;
        Rep(i, 26) if (ch[1][i]) {
            fail[ch[1][i]] = 1;
            dep[ch[1][i]] = 1;
            q.push(ch[1][i]);
        }
        for (; !q.empty(); q.pop()) {
            int u = q.front();
            Rep(i, 26) {
                int v = ch[u][i];
                if (!v) continue;
                fail[v] = 1, dep[v] = 1;
                for (int w = fail[u]; w; w = fail[w])
                    if (ch[w][i]) {
                        fail[v] = ch[w][i];
                        dep[v] = dep[fail[v]] + 1;
                        break;
                    }
                q.push(v);
            }
        }
    }
    void query(char *str) {
        int len = strlen(str), u = 1;
        static int ans[MAXN], bkt[MAXN], p[MAXN];
        Rep(i, len) {
            int c = str[i] - 97;
            while (u > 1 && !ch[u][c])
                u = fail[u];
            if (ch[u][c]) u = ch[u][c];
            else u = 1;
            ++ cnt[u];
        }
        For(i, 1, ids) ++ bkt[dep[i]];
        For(i, 1, ids) bkt[i] += bkt[i - 1];
        For(i, 1, ids) p[bkt[dep[i]] --] = i;
        Fordown(i, ids, 1) {
            int u = p[i];
            cnt[fail[u]] += cnt[u];
            Rep(j, SZ(id[u]))
                ans[id[u][j]] = cnt[u];
        }
        For(i, 1, n)
            printf("%d\n", ans[i]);
    }
}trie;
```

6.3 SA

```
namespace SA {
   int rk[MAXN << 1], tp[MAXN << 1], sa[MAXN], height[MAXN], m;
   void rsort(int n) {
      static int c[MAXN];
      For(i, 1, m) c[i] = 0;
      For(i, 1, n) ++ c[rk[i]];
      For(i, 1, m) c[i] += c[i - 1];
      Fordown(i, n, 1) sa[c[rk[tp[i]]] --] = tp[i];
   }
   void init(char *s, int n) {
      m = 26;
      For(i, 1, n) rk[i] = s[i] - 96, tp[i] = i;
      rsort(n);
      for (int k = 1; ; k <<= 1) {
         int p = 0;
         For(i, n - k + 1, n) tp[++ p] = i;
         For(i, 1, n) if (sa[i] > k) tp[++ p] = sa[i] - k;
         rsort(n), swap(tp, rk);
         rk[sa[1]] = m = 1;
         For(i, 2, n)
            rk[sa[i]] = tp[sa[i]] == tp[sa[i - 1]] && tp[sa[i] + k] == tp[sa[i - 1] + k] ? m : ++ m;
         if (m == n) break;
      }

      for (int i = 1, j, k = 0; i <= n; height[rk[i ++]] = k)
         for (k = k ? k - 1 : 0, j = sa[rk[i] - 1]; s[j + k] == s[i + k]; ++ k);

      For(i, 1, n) printf("%d ", sa[i]);
      putchar('\n');
      For(i, 2, n) printf("%d ", height[i]);
      putchar('\n');
   }
}
```

6.4 SAM

```
int tot = 1, las = 1, fa[MAXN << 1], ch[MAXN << 1][26], sz[MAXN << 1], len[MAXN << 1], p[MAXN << 1], bkt[MAXN << 1];
void extend(int c) {
   int np = ++ tot, p = las;
   len[las = np] = len[p] + 1, sz[np] = 1;
   while (p && !ch[p][c]) ch[p][c] = np, p = fa[p];
   if (!p) fa[np] = 1;
   else {
      int q = ch[p][c];
      if (len[q] == len[p] + 1) fa[np] = q;
      else {
         int nq = ++ tot;
         Cpy(ch[nq], ch[q]), fa[nq] = fa[q], len[nq] = len[p] + 1;
         fa[q] = fa[np] = nq;
         while (p && ch[p][c] == q) ch[p][c] = nq, p = fa[p];
      }
   }
}
For(i, 1, tot) ++ bkt[len[i]];
For(i, 1, tot) bkt[i] += bkt[i - 1];
For(i, 1, tot) p[bkt[len[i]] --] = i;
```

6.5 Manacher

```
n = read(), scanf("%s", s_ + 1);
s[++ len] = '#';
For(i, 1, n) s[++ len] = '$', s[++ len] = s_[i];
s[++ len] = '$', s[++ len] = '!';
For(i, 1, len) {
   if (s[i] != '$') continue;
   p[i] = i <= mx ? min(mx - i, p[(id << 1) - i]) : 0;
   while (s[i - p[i] - 1] == s[i + p[i] + 1]) ++ p[i];
   if (chkmax(mx, i + p[i])) id = i;
   if (!(p[i] & 1)) Ans += p[i] >> 1;
}
```

6.6 PAM

```
//[HDU5421] 双端插入PAM，输出回文串个数和本质不同回文串个数
void init() {
   fa[1] = fa[0] = 1, Set(ch, 0), len[tot = 1] = -1, ans = 0, l = (r = 1e5) + 1, suf = pre = 0, Set(s, 0);
}
void extend(int i, int &las, int ty) {
   int p = las, c = (s[i] = getchar()) - 97;
   while (s[i] != s[i - len[p] * ty - ty]) p = fa[p];
   if (!ch[p][c]) {
      int np = ++ tot, k = fa[p];
      while (s[i] != s[i - len[k] * ty - ty]) k = fa[k];
      len[np] = len[p] + 2, dep[np] = dep[fa[np] = ch[k][c]] + 1, ch[p][c] = np;
   }
   ans += dep[las = ch[p][c]];
   if (len[las] == r - l + 1) pre = suf = las;
}
int main() {
   static int T, opt;
   while (~scanf("%d", &T)) {
      init();
      while (T --) {
         opt = read();
         if (opt < 3) opt == 1 ? extend(--l, pre, -1) : extend(++ r, suf, 1);
         else opt == 3 ? printf("%d\n", tot - 1) : printf("%lld\n", ans);
      }
   }
   return 0;
}
```

7 多项式

7.1 多项式全家桶

```cpp
const int MAXN = 1 << 19, MOD = 998244353, g0 = 3;
int ig0;
int pw[MAXN], pw_[MAXN];
int fac[MAXN], ifac[MAXN];
int fpm(int a, int b = MOD - 2) {
    int ans = 1;
    for (; b; b >>= 1, a = (LL)a * a % MOD)
        if (b & 1)
            ans = (LL)ans * a % MOD;
    return ans;
}
int ad(int x, int y) { return (x += y) >= MOD ? x - MOD : x; }
void inc(int &x, int y) { if ((x += y) >= MOD) x -= MOD; }
int times2(int x) { return (x += x) >= MOD ? x - MOD : x; }
int Init(int n) {
    int pt, N;
    for (pt = 0, N = 1; N <= n; N <<= 1, ++ pt);
    ig0 = fpm(g0, MOD - 2);
    For(i, 1, pt + 1)
    pw[1 << i] = fpm(g0, (MOD - 1) / (1 << i));
    pw_[1 << i] = fpm(ig0, (MOD - 1) / (1 << i));
    fac[0] = 1;
    For(i, 1, N - 1) fac[i] = (LL)fac[i - 1] * i % MOD;
    ifac[N - 1] = fpm(fac[N - 1]);
    Fordown(i, N - 1, 1) ifac[i - 1] = (LL)ifac[i] * i % MOD;
    return N;
}
void NTT(int *a, int n, int ty) {
    static int rev[MAXN];
    static int W[MAXN];
    int pt = __builtin_ctz(n);
    Rep(i, n) if (i < (rev[i] = ((rev[i >> 1] >> 1) | ((i & 1) << (pt - 1))))) swap(a[i], a[rev[i]]);
    for (int i = 2, i2 = 1; i <= n; i2 = i, i <<= 1) {
        W[0] = 1, W[1] = ty > 0 ? pw[i] : pw_[i];
        For(j, 2, i2 - 1) W[j] = (LL)W[j - 1] * W[1] % MOD;
        for (int j = 0; j < n; j += i) {
            Rep(k, i2) {
                int x = a[j + k], y = (LL)a[j + k + i2] * W[k] % MOD;
                a[j + k] = ad(x, y), a[j + k + i2] = ad(x, MOD - y);
            }
        }
    }
    if (ty < 1) {
        int inv = fpm(n);
        Rep(i, n) a[i] = (LL)a[i] * inv % MOD;
    }
}
void Mult(int *f, int *g, int n, int *h) {
    static int f_[MAXN], g_[MAXN];
    Rep(i, n) f_[i] = f[i], g_[i] = g[i];
    For(i, n, n * 2 - 1) f_[i] = g_[i] = 0;
    NTT(f_, n << 1, 1), NTT(g_, n << 1, 1);
    Rep(i, n << 1) h[i] = (LL)f_[i] * g_[i] % MOD;
    NTT(h, n << 1, -1);
}
void Mult(int *f1, int *f2, int *f3, int n, int *h) {
    static int f1_[MAXN], f2_[MAXN], f3_[MAXN];
    Rep(i, n) f1_[i] = f1[i], f2_[i] = f2[i], f3_[i] = f3[i];
    For(i, n, n * 2 - 1) f1_[i] = f2_[i] = f3_[i] = 0;
    NTT(f1_, n << 1, 1), NTT(f2_, n << 1, 1), NTT(f3_, n << 1, 1);
    Rep(i, n << 1) h[i] = (LL)f1_[i] * f2_[i] % MOD * f3_[i] % MOD;
    NTT(h, n << 1, -1);
}
namespace Inv {
static int f[MAXN];
void Inv_(int *g, int n) {
    static int h[MAXN];
    if (n == 1) {
        g[0] = fpm(f[0]);
        return;
    }
    Inv_(g, n >> 1);
    Mult(g, g, f, n, h);
    Rep(i, n) g[i] = ad(ad(g[i], g[i]), MOD - h[i]);
}
void Inv(int *A, int n, int *ans) {
    Rep(i, n) f[i] = A[i], ans[i] = 0;
    Inv_(ans, n);
}
}
void Int(int *f, int n, int *g) {
    Fordown(i, n - 1, 1) g[i] = (LL)f[i - 1] * fpm(i) % MOD;
    g[0] = 0;
}
void Der(int *f, int n, int *g) {
    For(i, 1, n - 1) g[i - 1] = (LL)f[i] * i % MOD;
    g[n - 1] = 0;
}
void Ln(int *f, int n, int *g) {
    static int h[MAXN];
    Der(f, n, h), Inv:: Inv(f, n, g);
    Mult(h, g, n, g), Int(g, n, g);
}
namespace Exp {
static int G[MAXN];
void Exp_(int *F, int n) {
    static int H[MAXN];
    if (n == 1) {
        F[0] = 1;
        return;
    }
    Exp_(F, n >> 1);
    Ln(F, n, H);
    Rep(i, n) H[i] = ad(G[i], MOD - H[i]);
    H[0] = ad(H[0], 1);
    Mult(H, F, n, F);
}
void Exp(int *g, int n, int *ans) {
    Rep(i, n) G[i] = g[i], ans[i] = 0;
    Exp_(ans, n);
}
}
void Pow(int *f, int n, int k, int *g) {
    static int h[MAXN];
    Ln(f, n, h);
    Rep(i, n) h[i] = (LL)h[i] * k % MOD;
```

```
    Exp:: Exp(h, n, g);
}
namespace Sqrt {
static int A[MAXN], B[MAXN], a[MAXN];
void Sqrt_(int *b, int n) {
    if (n == 1) {
        b[0] = sqrt(a[0]);
        return;
    }
    Sqrt_(b, n >> 1);
    Rep(i, n) A[i] = b[i];
    Mult(A, A, n, A);
    Rep(i, n) A[i] = ad(A[i], a[i]), B[i] = ad(b[i], b[i]);
    Inv:: Inv(B, n, B);
    Mult(A, B, n, b);
}
void Sqrt(int *x, int *y, int n) {
    Rep(i, n) a[i] = x[i], y[i] = 0;
    Sqrt_(y, n);
}
}
int N = Init(131071);
```

7.2 牛顿迭代

问题：已知 G，求 F 使得 $G(F(x)) = 0$。已知 F_0 满足 $G(F_0(x)) \equiv 0 \pmod{x^t}$，则存在：

$$F(x) \equiv F_0(x) - \frac{G(F_0(x))}{G'(F_0(x))} \pmod{x^{2t}}$$

其中 $G'(F(x)) = \frac{dG}{dF}$

7.3 MTT

```
LL MOD;
namespace FFT {
    struct Z {
        LD r, i;
        Z (const LD &r0 = 0, const LD &i0 = 0) : r(r0), i(i0) {}
        Z operator + (const Z& t) const {return Z(r+t.r, i+t.i);}
        Z operator - (const Z& t) const {return Z(r-t.r, i-t.i);}
        Z operator * (const Z& t) const {return Z(r*t.r-i*t.i, r*t.i+i*t.r);}
        Z conj() const {return Z(r, -i);}
        void operator /= (const LD& t) {r /= t, i /= t;}
    };

    int n, bit, rev[MAXN];
    void init(int x) {
        n = 1, bit = 0;
        while(n <= x) n <<= 1, bit++;
        for(int i=1; i<n; i++) rev[i] = (rev[i>>1]>>1) | ((i&1)<<(bit-1));
    }
    void dft(Z *x, int f) {
        for(int i=0; i<n; i++)
            if(i < rev[i])
                swap(x[i], x[rev[i]]);
        for(int w=1; w<n; w<<=1)
        {
            for(int i=0; i<n; i+=(w<<1))
            {
                for(int j=0; j<w; j++)
                {
                    Z a = x[i+j], b = x[i+j+w] * Z(cos(PI/w*j), f*sin(PI/w*j));;
                    x[i+j] = a + b;
                    x[i+j+w] = a - b;
                }
            }
        }
        if(f == -1) for(int i=0; i<n; i++) x[i] /= n;
    }

    Z Xq[MAXN], Yq[MAXN], xlyl[MAXN], xlyh[MAXN], xhyl[MAXN], xhyh[MAXN];

    void mult(LL *x, LL *y, LL *ret) {
        for(int i=0; i<n; i++)
            Xq[i] = Z(x[i]>>15, x[i]&((1<<15)-1)),
                Yq[i] = Z(y[i]>>15, y[i]&((1<<15)-1));
        dft(Xq, +1), dft(Yq, +1);
        for(int i=0; i<n; i++)
        {
            int j = (n-i) & (n-1);
            Z xh = (Xq[i]+Xq[j].conj()) * Z(0.5, 0);
            Z xl = (Xq[i]-Xq[j].conj()) * Z(0, -0.5);
            Z yh = (Yq[i]+Yq[j].conj()) * Z(0.5, 0);
            Z yl = (Yq[i]-Yq[j].conj()) * Z(0, -0.5);
            xhyh[j] = xh*yh, xhyl[j] = xh*yl, xlyh[j] = xl*yh, xlyl[j] = xl*yl;
        }
        for(int i=0; i<n; i++)
            Xq[i] = xhyh[i] + xhyl[i] * Z(0, 1),
                Yq[i] = xlyh[i] + xlyl[i] * Z(0, 1);
        dft(Xq, +1), dft(Yq, +1);
        for(int i=0; i<n; i++)
        {
            LL xhyh = LL(Xq[i].r/n + 0.5) % MOD;
            LL xhyl = LL(Xq[i].i/n + 0.5) % MOD;
            LL xlyh = LL(Yq[i].r/n + 0.5) % MOD;
            LL xlyl = LL(Yq[i].i/n + 0.5) % MOD;
            ret[i] = ((xhyh<<30) + (xhyl<<15) + (xlyh<<15) + (xlyl)) % MOD;
        }
    }
}
//先init，后mult使用即可
```

7.4 FWT

```
void FWTor(int *a, int ty) {
    for (int i = 2; i <= N; i <<= 1)
        for (int j = 0; j < N; j += i)
            Rep(k, i >> 1)
                if (ty) a[j + (i >> 1) + k] = ad(a[j + (i >> 1) + k], a[j + k]);
                else a[j + (i >> 1) + k] = ad(a[j + (i >> 1) + k], Mod - a[j + k]);
}
void FWTand(int *a, int ty) {
    for (int i = 2; i <= N; i <<= 1)
        for (int j = 0; j < N; j += i)
```

```cpp
        Rep(k, i >> 1)
            if (ty) a[j + k] = ad(a[j + k], a[j + (i
                >> 1) + k]);
            else a[j + k] = ad(a[j + k], Mod - a[j +
                (i >> 1) + k]);
}
void FWTxor(int *a, int ty) {
    for (int i = 2; i <= N; i <<= 1)
        for (int j = 0; j < N; j += i)
            Rep(k, i >> 1) {
                int x = a[j + k], y = a[j + k + (i >> 1)
                    ];
                a[j + k] = ad(x, y), a[j + k + (i >> 1)]
                    = ad(x, Mod - y);
                if (!ty) a[j + k] = a[j + k] * inv2 %
                    Mod, a[j + k + (i >> 1)] = a[j + k +
                    (i >> 1)] * inv2 % Mod;
            }
}
```

7.5 FMT

```cpp
void FMTor(int *a, int n, int ty) {
    for (int i = 2, p = 1; i <= n; p = i, i <<= 1)
        for (int j = 0; j < n; j += i) Rep(k, p)
            inc(a[j + k + p], (LL)(ty == 1 ? 1 : MOD -
                1) * a[j + k] % MOD);
}
void FMTand(int *a, int n, int ty) {
    for (int i = 2, p = 1; i <= n; p = i, i <<= 1)
        for (int j = 0; j < n; j += i) Rep(k, p)
            inc(a[j + k], (LL)(ty == 1 ? 1 : MOD - 1) *
                a[j + k + p] % MOD);
}
```

8 其它算法

8.1 模拟退火

```cpp
//BZOJ3680
const double eps = 1e-15;
const int maxn = 1005, Times = 15;
int n, w[maxn], x[maxn], y[maxn];
double randdec(double T) { return ((rand() + rand())
    - RAND_MAX) * 1. / RAND_MAX * T * 1e-2; }
double calc(double nx, double ny) {
    double sx = 0, sy = 0, len, dx, dy;
    For(i, 1, n) {
        dx = x[i] - nx, dy = y[i] - ny, len = sqrt(dx
            * dx + dy * dy);
        if (fabs(len) < eps) continue;
        sx += w[i] * dx / len, sy += w[i] * dy / len;
    }
    return sqrt(sx * sx + sy * sy);
}
int main() {
    static double xba, yba, ansx, ansy, tba, ans;
    n = read();
    For(i, 1, n) xba += x[i] = read(), yba += y[i] =
        read(), w[i] = read();
    ansx = xba /= n, ansy = yba /= n, tba = ans = calc
        (xba, yba);
    for (int Case = Times; Case --; ) {
        double nowx = xba, nowy = yba, now = tba, res,
            newx, newy;
        for (double T = 1e6; T >= eps; T *= 0.99) {
            newx = nowx + randdec(T), newy = nowy +
                randdec(T), res = calc(newx, newy);
            if (res < ans) ans = res, ansx = newx, ansy
                = newy;
            if (res < now || exp((now - res) / T) *
                RAND_MAX < rand())
                nowx = newx, nowy = newy, now = res;
        }
    }
    printf("%.3lf %.3lf\n", ansx, ansy);
    return 0;
}
```

9 一些有用的定理和结论

- 皮克定理：$2S = 2a + b - 2$，a 为内部点数，b 为边界点数，S 为面积。
- 欧拉公式：$F + V = E + C + 1$，C 表示连通块个数

10 其它代码

10.1 pb_ds 的 hash_table

```cpp
#include<ext/pb_ds/hash_policy.hpp>
#include<ext/pb_ds/assoc_container.hpp>
gp_hash_table<int,bool> h1;
cc_hash_table<int,bool> h2;
```

10.2 __builtin

```cpp
__builtin_ffs(x)//返回x中最后一个为1的位是从后向前的第几
    位
__builtin_popcount(x)//x中1的个数。
__builtin_ctz(x)//x末尾0的个数。x=0时结果未定义。
__builtin_clz(x)//x前导0的个数。x=0时结果未定义。
//上面的宏中x都是unsigned int型的，如果传入signed或者是
    char型，会被强制转换成unsigned int。
__builtin_parity(x)//x中1的奇偶性
```

10.3 std::set

```cpp
set_union(eg1.begin(),eg1.end(),eg2.begin(),eg2.end()
    ,insert_iterator<set<int> >(eg3,eg3.begin()));
set_intersection(eg1.begin(),eg1.end(),eg2.begin(),
    eg2.end(),insert_iterator<set<int> >(eg3,eg3.
    begin()));
set_difference(eg1.begin(),eg1.end(),eg2.begin(),eg2.
    end(),insert_iterator<set<int> >(eg3,eg3.begin())
    ); //差
set_symmetric_difference(eg1.begin(),eg1.end(),eg2.
    begin(),eg2.end(),insert_iterator<set<int> >(eg3,
    eg3.begin())); //对称差
```

另外，insert() 的返回值为 pair<set<TYPE>::iterator, bool>

10.4 std::bitset

```
a ^ b //Xor
a & b //And
a | b //Or
bs.any() //是否存在1
bs.none() //是否都为0
bs.count() //1的个数
b.size() //二进制位的个数
b[pos] //第 pos 位二进制数
b.test(pos) //第 pos 位是否为 1
b.set() //全设为 1
b.set(pos) //将 pos 处设为 1
b.reset() //全设为 0
b.reset(pos) //将 pos 处设为 0
b.flip() //全部取反
b.flip(pos) //将 pos 处取反
b.to_ulong() //返回一个 unsigned long 值
b._Find_first() //返回第一个1的位置
b._Find_next(x) //返回x之后下一个1的位置
```

10.5 priority_queue 的重载运算符

```
struct cmp {
    bool operator()(int x, int y) { return pos[x] >
        pos[y]; }
};
priority_queue<int, vector<int>, cmp> q;
```

10.6 对拍

10.6.1 Windows

```
@echo off
set /a i=1
:loop
echo Case %i%:
set /a i=i+1
gen.exe
a.exe
bf.exe
fc a.out a.ans
if not errorlevel 1 goto loop
pause
```

10.6.2 Linux

```
#!/bin/bash
for i in $(seq 1 100000);do
    ./gen
    ./a
    ./a1
    if diff a.out a1.out; then
        echo $i "AC"
    else
        echo $i "WA"
        exit 0
    fi
done
```

10.7 编译选项

```
-fsanitize=address,undefined
```