Templates

Shanghai Jiaotong University

Metis

Member:
Sishan Long
Yutong Xie
Jingyi Cai

Coach:
Yunqi Li
Xueyuan Zhao

Contents

1 数学 **3**
- 1.1 FFT . . . 3
- 1.2 高斯消元 . . . 5
- 1.3 中国剩余定理 . . . 8
- 1.4 Polya 寻找等价类 . . . 9
- 1.5 拉格朗日插值 . . . 10
- 1.6 欧拉公式 . . . 10
- 1.7 求行列式的值 . . . 10
- 1.8 莫比乌斯 . . . 11
- 1.9 Cayley 公式与森林计数 . . . 11

2 数据结构 **12**
- 2.1 KD Tree . . . 12
- 2.2 Splay by xyt . . . 15
- 2.3 主席树 by cjy . . . 18
- 2.4 主席树 by xyt . . . 19
- 2.5 树分治 by xyt . . . 20
- 2.6 树链剖分 by cjy . . . 22
- 2.7 树链剖分 by xyt . . . 23
- 2.8 点分治 by xyt . . . 25
- 2.9 LCT by xyt . . . 27

3 计算几何 **30**
- 3.1 向量旋转 . . . 30
- 3.2 至少被 i 个圆覆盖的面积 . . . 30
- 3.3 计算几何杂 . . . 33
- 3.4 三维变换 . . . 34

4 字符串 **36**
- 4.1 AC-Automachine by cjy . . . 36
- 4.2 AC-Automachine by xyt . . . 37
- 4.3 后缀数组 . . . 38
- 4.4 扩展 KMP . . . 39
- 4.5 回文树 . . . 40
- 4.6 SAM by lss . . . 42
- 4.7 SAM by xyt . . . 43

5 图论 **44**
- 5.1 图论相关 . . . 44
- 5.2 SteinerTree by cjy . . . 45
- 5.3 LCA by xyt . . . 46
- 5.4 KM . . . 47
- 5.5 KM 三次方 . . . 48
- 5.6 Dinic by cjy . . . 50
- 5.7 网络流 by xyt . . . 51
- 5.8 最大密度子图 . . . 53
- 5.9 强联通分量 . . . 57

5.10 边双联通分量 57
5.11 点双联通分量加构造森林块 58
5.12 K 短路 59

6 其他 **61**
6.1 Dancing Links(精确覆盖及重复覆盖) 61
6.2 序列莫队 65
6.3 模拟退火 66
6.4 Java 68
6.5 博弈论相关 70

7 Tips **70**

1 数学

1.1 FFT

```
const int maxn = 1e6 + 5;
typedef complex<long double> cpb;
int N;cpb a[maxn], aa[maxn], b[maxn], bb[maxn], c[maxn], cc[maxn];
void fft(cpb x[], cpb xx[], int n, int step, int type){
        if(n == 1){
                xx[0] = x[0];
                return;
        }
        int m = n >> 1;
        fft(x, xx, m, step << 1, type);
        fft(x + step, xx + m, m, step << 1, type);
        cpb w = exp(cpb(0., PI * type / m));
        cpb t = 1.;
        for(int i = 0; i < m; ++i){
                cpb t0 = xx[i];
                cpb t1 = xx[i+m];
                xx[i] = t0 + t * t1;
                xx[i+m] = t0 - t * t1;
                t *= w;
        }
}
int main(){
        int n, x;
        scanf("%d", &n);
        for(int i = 0; i < n; ++i) scanf("%d", &x), a[i] = cpb(x, 0.);
        for(int i = 0; i < n; ++i) scanf("%d", &x), b[i] = cpb(x, 0.);
        for(N = 1; N < n + n; N <<= 1);
        fft(a, aa, N, 1, 1);
        fft(b, bb, N, 1, 1);
        for(int i = 0; i < N; ++i) cc[i] = aa[i] * bb[i];
        fft(cc, c, N, 1, -1);
        for(int i = 0; i < N; ++i) c[i] = c[i].real() / N;
}
// 复数 递归
typedef complex<double> cpb;
void fft(cpb x[], cpb xx[], int n, int step, int type){ // step 表示步长 代
   码后面举个例子说明一下好了
    if(n == 1){xx[0] = x[0]; return;}
    int m = n >> 1;
    fft(x, xx, m, step << 1, type); // A[0]
    fft(x + step, xx + m, m, step << 1, type); // A[1]
    cpb w = exp(cpb(0, type * pi / m)); // 求原根 pi / m 其实就是 2 * pi / n
    cpb t = 1;
    for(int i = 0; i < m; ++i){
```

```
        cpb t0 = xx[i]; // 这个里面是A[0]的内容
        cpb t1 = xx[i+m]; // 这个里面是A[1]的内容
        xx[i] = t0 + t * t1;
        xx[i+m] = t0 - t * t1;
        t *= w;
    }
}
int main(){
    // main函数我就乱写了 >w<
    a[].get();
    b[].get();
    A = a.length();
    B = b.length();
    for(N = 1; N < A + B; N <<= 1);
    fft(a, aa, N, 1, 1);
    fft(b, bb, N, 1, 1);
    for(int i = 0; i < N; ++i) cc[i] = aa[i] * bb[i];
    fft(cc, c, N, 1, -1);
    for(int i = 0; i < N; ++i) c[i] /= N;
    c[].print();
    return 0;
}
// 原根 蝶型
const int p = 7340033;
const int g = 3;
int powmod(int x, int y){
    // 我又要乱写啦>w< (那为了压行我可都删掉啦!
}
void fft(int xx[], int n, int type){
    // 这里在对二进制位对称的位置进行交换
    for(int i = 0; i < n; ++i){ // i枚举每一个下表
        int j = 0; // j为n位二进制下i的对称
        for(int k = i, m = n - 1; m != 0; j = (j << 1) | (k & 1), k >>= 1, m
            >>= 1);
        if(i < j) swap(xx[i], xx[j]); // 为了防止换了之后又换回来于是只在 i
            < j 时交换
    }
    // for代替递归
    for(int m = 1; m < n; m <<= 1){ // m为当前讨论区间长度的一半
        int w = powmod(g, (1LL * type * (p - 1) / (m << 1) + p - 1) % (p -
            1));
        for(int j = 0; j < n; j += (m << 1)){ // j为当前讨论区间起始位
            // 啊这些都和递归一样了
            int t = 1;
            for(int i = 0; i < m; ++i){
                int t0 = xx[i+j];
                int t1 = 1LL * xx[i+j+m] * t % p;
                xx[i+j] = (t0 + t1) % p;
```

```
                xx[i+j+m] = (t0 - t1 + p) % p;
                t = 1LL * t * w % p;
            }
        }
    }
}
int main(){
    // 继续乱写>w<
    a[].get();
    b[].get();
    A = a.length();
    B = b.length();
    for(N = 1; N < A + B; N <<= 1);
    fft(a, N, 1);
    fft(b, N, 1);
    for(int i = 0; i < N; ++i) c[i] = 1LL * a[i] * b[i] % p;
    fft(c, N, -1);
    int inv_N = powmod(N, p - 2);
    for(int i = 0; i < N; ++i) c[i] = 1LL * c[i] * inv_N % p;
    c[].print();
    return 0;
}
```

1.2 高斯消元

```
int n, r, t;
const int pp=10007;
int e[333][333];
int fa[333];
struct Point{
    int x, y;
    int num;
    Point() {}
    Point(int x, int y, int num = -1) : x(x), y(y), num(num) {}
};
Point p[333];
int dist2(const Point &p) {
    return p.x * p.x + p.y * p.y;
}
Point operator + (const Point &a, const Point &b) {
    return Point(a.x + b.x, a.y + b.y);
}

Point operator - (const Point &a, const Point &b) {
    return Point(a.x - b.x, a.y - b.y);
}
int dot(Point a, Point b) {
```

```
    return a.x * b.x + a.y * b.y;
}
int cross(Point a, Point b) {
    return a.x * b.y - a.y * b.x;
}
int find(int x) {
    if (fa[x] == x) return x;
    else {
        fa[x] = find(fa[x]);
        return fa[x];
    }
}
void addedge(int x, int y) {
    e[x][x]++;
    e[x][y] = -1;
    int fax=find(fa[x]);
    int fay=find(fa[y]);
    if (fax != fay) fa[fax] = fay;
}
int P(int x, int k) {
    if (k == 0) return 0;
    if (k == 1) return x;
    int ret = P(x, k / 2);
    ret = ret *ret % pp;
    if (k & 1) ret = ret * x % pp;
    return ret;
}

void Guass() {
    --n;
    int ans = 1;
    for (int i = 1; i <= n; i++) {
        int pos = i; int mx = 0;
        for (int j = i; j <= n; j++)
            if (abs(e[j][i])>mx) {
                mx = abs(e[j][i]);
                pos = j;
            }
        if (pos != i) {
            for (int j = 1; j <= n; j++) {
                swap(e[i][j], e[pos][j]);
            }
            ans *= -1;
        }
        int inv = P(e[i][i], pp - 2);
        for (int j = i+1;j <= n; j++) {
            int t = inv * e[j][i] % pp;
            for (int k = i; k <= n; k++)
```

```
                e[j][k] = (e[j][k] - t*e[i][k]) % pp;
        }
    }
    for (int i = 1; i <= n; i++)
        ans = ans * e[i][i] % pp;
    if (ans < 0) ans += pp;
    cout << ans << endl;
}
void doit(int k) {
    Point a[333];
    int m = 0;
    for (int i = 1;i <= n; i++)
        if (i != k && dist2(p[i] - p[k]) <= r*r) {
            bool flag = 1;
            for (int j = 1;j <= n ; j++)
                if (j != k && j !=i) {
                    if (cross(p[j] - p[k],p[i] - p[k]) ==0 && dot(p[j] - p[k
                       ],p[i] - p[k]) >0 && dist2(p[j] - p[k]) < dist2(p[i]
                       - p[k])) {
                        flag = 0;
                        break;
                    }
                }
            if (flag) addedge(k, i);
        }
}
void solve() {
    cin >> n >> r;
    for (int i = 1; i <= n; i++) {
        scanf("%d%d", &p[i].x, &p[i].y);
    }
    for (int i = 1;i <=n; i++) fa[i] = i;
    memset(e, 0, sizeof(e));
    for (int i = 1;i <= n; i++)
        doit(i);
    for (int i = 2;i <= n; i++)
        if (find(i) != find(i-1)) {
            puts("-1");
            return;
        }
    Guass();
}
int main() {
    cin >> t;
    for (int i = 1; i <= t; i++) solve();
    return 0;
}
```

1.3 中国剩余定理

```
long long extended_Euclid(long long a, long long b, long long &x, long long
    &y) { //return gcd(a, b)
        if (b == 0) {
                x = 1;
                y = 0;
                return a;
        }
        else {
                long long tmp = extended_Euclid(b, a % b, x, y);
                long long t = x;
                x = y;
                y = t - a / b * y;
                return tmp;
        }
}
long long China_Remainder(long long a[], long long b[], int n, long long &
    cir) { //a[]存放两两互质的除数  b[]存放余数
        long long x, y, ans;
        ans = 0; cir = 1;
        for (int i = 1; i <= n; i++) cir *= a[i];
        for (int i = 1; i <= n; i++) {
                long long tmp = cir / a[i];
                extended_Euclid(a[i], tmp, x, y);
                ans = (ans + y * tmp * b[i]) % cir; //可能会爆long long  用
                    快速乘法
        }
        return (cir + ans % cir) % cir;
}
bool merge(long long &a1, long long &b1, long long a2, long long b2) { //num
    = b1(mod a1), num = b2(mod a2)
        long long x, y;
        long long d = extended_Euclid(a1, a2, x, y);
        long long c = b2 - b1;
        if (c % d) return false;
        long long p = a2 / d;
        x = (c / d * x % p + p) % p;
        b1 += a1 * x;
        a1 *= a2 / d;
        return true;
}
long long China_Remainder2(long long a[], long long b[], int n) { //a[]存放
    除数(不一定两两互质)  b[]存放余数
        long long x, y, ans, cir;
        cir = a[1]; ans = b[1];
        for (int i = 2; i <= n; i++) {
                if (!merge(cir, ans, a[i], b[i])) return -1;
```

```
        }
        return (cir + ans % cir) % cir;
}
```

1.4 Polya 寻找等价类

```
/*
Polya定理:
设G={1, 2, 3..... n}是X={a1, a2, a3.....an}上一个置换群, 用m中颜色对X中的元素进行涂色,
那么不同的涂色方案数为: 1/|G|*(m^C(1)+m^C(2)+m^C(3)+...+m^C(k)). 其中C(k)为置换 k的循环节的个数。
*/
int f[101];
long long mul[101];
bool vis[101];
int pos[101];
int n, m, k;
long long ans = 0, K;
int a[301], b[301];
int getfa(int x) { return !f[x] ? x : (f[x] = getfa(f[x])); }
int g[301][301];
long long check()
{
        int cnt = 0;
        for (int i = 1; i <= n; i ++) vis[i] = false;
        for (int i = 1; i <= n; i ++)
                if (!vis[i])
                {
                        for (int j = i; vis[j] == false; j = pos[j])
                                vis[j] = true;
                        ++ cnt;
                }
        for (int i = 1; i <= n; i ++)
                for (int j = 1; j <= n; j ++)
                        if (g[i][j] != g[pos[i]][pos[j]]) return 0;
        return mul[cnt];
}
void dfs(int x)
{
        if (x == n + 1)
        {
                long long tmp = check();
                if (tmp) ++ K;
                ans += tmp;
                return ;
        }
```

```
        for (int i = 1; i <= n; i ++)
                if (!vis[i])
                {
                        vis[i] = true;
                        pos[x] = i;
                        dfs(x + 1);
                        vis[i] = false;
                }
}
int main( )
{
        scanf("%d %d %d", &n, &m, &k);
        mul[0] = 1;
        for (int i = 1; i <= n; i ++) mul[i] = mul[i - 1] * k;
        for (int i = 1; i <= m; i ++)
                scanf("%d %d", &a[i], &b[i]), g[a[i]][b[i]] ++, g[b[i]][a[i
                   ]] ++;
        dfs(1);
        cout << ans / K << endl;
        return 0;
}
```

1.5 拉格朗日插值

$$p_j(x) = \prod_{i \in I_j} \frac{x - x_i}{x_j - x_i}$$

$$L_n(x) = \sum_{j=1}^{n} y_j p_j(x)$$

1.6 欧拉公式

```
V - E + F = C + 1 C为联通块数量
V - E + F = 2 - 2G G is the number of genus of surface
```

1.7 求行列式的值

行列式有很多性质，第 a 行 $*k$ 加到第 b 行上去，行列式的值不变。
三角行列式的值等于对角线元素之积。
第 a 行与第 b 行互换，行列式的值取反。
常数 * 行列式，可以把常数乘到某一行里去。
注意：全是整数并取模的话当然需要求逆元

1.8 莫比乌斯

$$\sum_{d|n}\mu(d)=[n==1]$$

$$\mu(m)=\begin{cases}(-1)^r & m=p_1p_2...p_r\\ 0 & p^2|n\end{cases}$$

某个 Mobius 推倒：

$$\sum_{i=1}^{n}\sum_{j=1}^{m}lcm(i,j)$$
$$=\sum_{d=1}^{n}\sum_{i=1}^{n}\sum_{j=1}^{m}[gcd(i,j)==d]\frac{ij}{d}$$
$$=\sum_{d=1}^{n}\sum_{i=1}^{n/d}\sum_{j=1}^{m/d}[gcd(i,j)==1]ijd$$
$$=\sum_{d=1}^{n}d\sum_{i=1}^{n/d}\sum_{j=1}^{m/d}i*j\sum_{d'|i,d'|j}\mu(d')$$
$$=\sum_{d=1}^{n}\sum_{d'=1}^{n/d}\sum_{i=1}^{n/dd'}\sum_{j=1}^{m/dd'}dijd'^2\mu(d')$$
令 $D=dd'$ $s(x,y)=\frac{xy(x+1)(y+1)}{4}$
$$=\sum_{D=1}^{n}s(\frac{n}{D},\frac{m}{D})D\sum_{d'|D}d'\mu(d')$$

$$\mu(n)=\begin{cases}1 & 若n=1\\ (-1)^k & 若n无平方数因子，且n=p_1p_2\dots p_k\\ 0 & 若n有大于1的平方数因数\end{cases}$$

$$\sum_{d|n}\mu(d)=\begin{cases}1 & 若n=1\\ 0 & 其他情况\end{cases}$$

$$g(n)=\sum_{d|n}f(d)\Leftrightarrow f(n)=\sum_{d|n}\mu(d)g(\frac{n}{d}),g(x)=\sum_{n=1}^{[x]}f(\frac{x}{n})\Leftrightarrow f(x)=\sum_{n=1}^{[x]}\mu(n)g(\frac{x}{n})$$

1.9 Cayley 公式与森林计数

Cayley 公式是说，一个完全图 K_n 有 n^{n-2} 棵生成树，换句话说 n 个节点的带标号的无根树有 n^{n-2} 个。

令 $g[i]$ 表示点数为 i 的森林个数，$f[i]$ 表示点数为 i 的生成树计数 $f[i] = i^{i-2}$) 那么便有

$$g[i] = \sum (g[i-j] \times cnr[i-1][j-1] \times f[j])$$

$$g[i] = \sum \frac{g[i-j] \times fac[i-1] \times f[j]}{fac[j-1] \times fac[i-j]} = fac[i-1] \times \sum (\frac{f[j]}{fac[j-1]} \times \frac{g[i-j]}{fac[i-j]})$$

2 数据结构

2.1 KD Tree

```
long long norm(const long long &x) {
    //    For manhattan distance
    return std::abs(x);
    //    For euclid distance
    return x * x;
}
struct Point {
    int x, y, id;
    const int& operator [] (int index) const {
        if (index == 0) {
            return x;
        } else {
            return y;
        }
    }
    friend long long dist(const Point &a, const Point &b) {
        long long result = 0;
        for (int i = 0; i < 2; ++i) {
            result += norm(a[i] - b[i]);
        }
        return result;
    }
} point[N];
struct Rectangle {
    int min[2], max[2];
    Rectangle() {
        min[0] = min[1] = INT_MAX;
        max[0] = max[1] = INT_MIN;
    }
    void add(const Point &p) {
        for (int i = 0; i < 2; ++i) {
            min[i] = std::min(min[i], p[i]);
            max[i] = std::max(max[i], p[i]);
        }
    }
    long long dist(const Point &p) {
        long long result = 0;
```

```
        for (int i = 0; i < 2; ++i) {
            //    For minimum distance
            result += norm(std::min(std::max(p[i], min[i]), max[i]) - p[i]);
            //    For maximum distance
            result += std::max(norm(max[i] - p[i]), norm(min[i] - p[i]));
        }
        return result;
    }
};
struct Node {
    Point seperator;
    Rectangle rectangle;
    int child[2];
    void reset(const Point &p) {
        seperator = p;
        rectangle = Rectangle();
        rectangle.add(p);
        child[0] = child[1] = 0;
    }
} tree[N << 1];
int size, pivot;
bool compare(const Point &a, const Point &b) {
    if (a[pivot] != b[pivot]) {
        return a[pivot] < b[pivot];
    }
    return a.id < b.id;
}
int build(int l, int r, int type = 1) {
    pivot = type;
    if (l >= r) {
        return 0;
    }
    int x = ++size;
    int mid = l + r >> 1;
    std::nth_element(point + l, point + mid, point + r, compare);
    tree[x].reset(point[mid]);
    for (int i = l; i < r; ++i) {
        tree[x].rectangle.add(point[i]);
    }
    tree[x].child[0] = build(l, mid, type ^ 1);
    tree[x].child[1] = build(mid + 1, r, type ^ 1);
    return x;
}
int insert(int x, const Point &p, int type = 1) {
    pivot = type;
    if (x == 0) {
        tree[++size].reset(p);
        return size;
```

```
    }
    tree[x].rectangle.add(p);
    if (compare(p, tree[x].seperator)) {
        tree[x].child[0] = insert(tree[x].child[0], p, type ^ 1);
    } else {
        tree[x].child[1] = insert(tree[x].child[1], p, type ^ 1);
    }
    return x;
}
//    For minimum distance
void query(int x, const Point &p, std::pair<long long, int> &answer, int
   type = 1) {
    pivot = type;
    if (x == 0 || tree[x].rectangle.dist(p) > answer.first) {
        return;
    }
    answer = std::min(answer,
             std::make_pair(dist(tree[x].seperator, p), tree[x].seperator.id
                ));
    if (compare(p, tree[x].seperator)) {
        query(tree[x].child[0], p, answer, type ^ 1);
        query(tree[x].child[1], p, answer, type ^ 1);
    } else {
        query(tree[x].child[1], p, answer, type ^ 1);
        query(tree[x].child[0], p, answer, type ^ 1);
    }
}
std::priority_queue<std::pair<long long, int> > answer;
void query(int x, const Point &p, int k, int type = 1) {
    pivot = type;
    if (x == 0 ||
        (int)answer.size() == k && tree[x].rectangle.dist(p) > answer.top().
           first) {
        return;
    }
    answer.push(std::make_pair(dist(tree[x].seperator, p), tree[x].seperator
       .id));
    if ((int)answer.size() > k) {
        answer.pop();
    }
    if (compare(p, tree[x].seperator)) {
        query(tree[x].child[0], p, k, type ^ 1);
        query(tree[x].child[1], p, k, type ^ 1);
    } else {
        query(tree[x].child[1], p, k, type ^ 1);
        query(tree[x].child[0], p, k, type ^ 1);
    }
}
```

2.2 Splay by xyt

```
struct Splay{
        int tot, rt;
        struct Node{
                int lson, rson, fath, sz;
                int data;
                bool lazy;
        };
        Node nd[MAXN];
        void reverse(int i){
                if(!i) return;
                swap(nd[i].lson, nd[i].rson);
                nd[i].lazy = true;
        }
        void push_down(int i){
                if(!i || !nd[i].lazy) return;
                reverse(nd[i].lson);
                reverse(nd[i].rson);
                nd[i].lazy = false;
        }
        void zig(int i){
                int j = nd[i].fath;
                int k = nd[j].fath;
                if(k && j == nd[k].lson) nd[k].lson = i;
                else if(k) nd[k].rson = i;
                nd[i].fath = k;
                nd[j].fath = i;
                nd[nd[i].rson].fath = j;
                nd[j].lson = nd[i].rson;
                nd[i].rson = j;
                nd[i].sz = nd[j].sz;
                nd[j].sz = nd[nd[j].lson].sz + nd[nd[j].rson].sz + 1;
        }
        void zag(int i){
                int j = nd[i].fath;
                int k = nd[j].fath;
                if(k && j == nd[k].lson) nd[k].lson = i;
                else if(k) nd[k].rson = i;
                nd[i].fath = k;
                nd[j].fath = i;
                nd[nd[i].lson].fath = j;
                nd[j].rson = nd[i].lson;
                nd[i].lson = j;
                nd[i].sz = nd[j].sz;
                nd[j].sz = nd[nd[j].lson].sz + nd[nd[j].rson].sz + 1;
        }
        void down_path(int i){
```

```
        if(nd[i].fath) down_path(nd[i].fath);
        push_down(i);
}
void splay(int i){
        down_path(i);
        while(nd[i].fath){
                int j = nd[i].fath;
                if(nd[j].fath == 0){
                        if(i == nd[j].lson) zig(i);
                        else zag(i);
                }else{
    int k = nd[j].fath;
    if(j == nd[k].lson){
        if(i == nd[j].lson) zig(j), zig(i);
        else zag(i), zig(i);
    }else{
        if(i == nd[j].rson) zag(j), zag(i);
        else zig(i), zag(i);
    }
                }
        }
        rt = i;
}
int insert(int stat){ // 插入信息
        int i = rt;
        ++tot;
        nd[tot].data = stat;
        nd[tot].sz = 1;
        if(!nd[i].sz){
                nd[tot].fath = 0;
                rt = tot;
                return tot;
        }
        while(i){
                ++nd[i].sz;
                if(stat < nd[i].data){
    if(nd[i].lson) i = nd[i].lson;
    else{
    nd[i].lson = tot;
    break;
    }
                }else{
    if(nd[i].rson) i = nd[i].rson;
    else{
    nd[i].rson = tot;
    break;
    }
                }
```

```
        }
        nd[tot].fath = i;
        splay(tot);
        return tot;
}
void delet(int i){ // 删除信息
        if(!i) return;
        splay(i);
        int ls = nd[i].lson;
        int rs = nd[i].rson;
        nd[ls].fath = nd[rs].fath = 0;
        nd[i].lson = nd[i].rson = 0;
        if(ls == 0){
                rt = rs;
                nd[rs].fath = 0;
        }else{
                rt = ls;
                while(nd[ls].rson) ls = nd[ls].rson;
                splay(ls);
                nd[ls].fath = 0;
                nd[rs].fath = ls;
                nd[ls].rson = rs;
        }
        nd[rt].sz += nd[nd[rt].rson].sz;
}
int get_rank(int i){ // 查询节点编号为 i 的 rank
        splay(i);
        return nd[nd[i].rson].sz + 1;
}
int find(int stat){ // 查询信息为 stat 的节点编号
        int i = rt;
        while(i){
                if(stat < nd[i].data) i = nd[i].lson;
                else if(stat > nd[i].data) i = nd[i].rson;
                        else return i;
        }
        return i;
}
int get_kth_max(int k){ // 查询第 k 大 返回其节点编号
        int i = rt;
        while(i){
                if(k <= nd[nd[i].rson].sz) i = nd[i].rson;
                else if(k > nd[nd[i].rson].sz + 1) k -= nd[nd[i].
                   rson].sz + 1, i = nd[i].lson;
    else return i;
        }
        return i;
}
```

```
}sp;
```

2.3 主席树 by cjy

```
const int N = 100005;
struct Tree {
        int l, r, L, R, x;
} h[(int)3e6];
int n, m, q, a[N], b[N];
int root[N], tot;
void Build(int x, int l, int r) {
        h[x].l = l; h[x].r = r; h[x].x = 0;
        if (l < r) {
                int m = (l + r) / 2;
                h[x].L = ++tot; Build(tot, l, m);
                h[x].R = ++tot; Build(tot, m + 1, r);
        }
}
void build(int x, int y, int num) {
        int l = h[y].l, r = h[y].r;
        h[x].l = l; h[x].r = r;
        h[x].x = h[y].x + 1; //--------------------
        if (l < r) {
                int m = (l + r) / 2;
                if (num <= m) {
                        h[x].L = ++tot; build(tot, h[y].L, num);
                        h[x].R = h[y].R;
                }
                else {
                        h[x].L = h[y].L;
                        h[x].R = ++tot; build(tot, h[y].R, num);
                }
                h[x].x = h[h[x].L].x + h[h[x].R].x;
        }
}
int find(int x, int y, int k) {
        if (h[x].l == h[x].r) return h[x].l;
        int t = h[h[y].L].x - h[h[x].L].x;
        if (t >= k) return find(h[x].L, h[y].L, k);
        else return find(h[x].R, h[y].R, k - t);
}
int main() {
        scanf("%d%d", &n, &q);
        for (int i = 1; i <= n; i++) {
                scanf("%d", &a[i]);
                b[i] = a[i];
        }
```

```
        sort(b + 1, b + n + 1);
        m = unique(b + 1, b + n + 1) - (b + 1);
        for (int i = 1; i <= n; i++)
                a[i] = lower_bound(b + 1, b + m + 1, a[i]) - (b + 1) + 1;
        root[0] = tot = 1;
        Build(1, 1, m); //------------------
        for (int i = 1; i <= n; i++) {
                root[i] = ++tot;
                build(tot, root[i - 1], a[i]);
        }
        for(int i = 1; i <= q; i++) {
                int l, r, k;
                scanf("%d%d%d", &l, &r, &k);
                printf("%d\n", b[find(root[l - 1], root[r], k)]);
        }
        return 0;
}
```

2.4 主席树 by xyt

```
const int maxn = 1e5 + 5;
const int inf = 1e9 + 1;
struct segtree{
        int tot, rt[maxn];
        struct node{
                int lson, rson, size;
        }nd[maxn*40];
        void insert(int &i, int left, int rght, int x){
                int j = ++tot;
                int mid = (left + rght) >> 1;
                nd[j] = nd[i];
                nd[j].size++;
                i = j;
                if(left == rght) return;
                if(x <= mid) insert(nd[j].lson, left, mid, x);
                else insert(nd[j].rson, mid + 1, rght, x);
    }
        int query(int i, int j, int left, int rght, int k){
                if(left == rght) return left;
                int mid = (left + rght) >> 1;
                if(nd[nd[j].lson].size - nd[nd[i].lson].size >= k) return
                   query(nd[i].lson, nd[j].lson, left, mid, k);
                else return query(nd[i].rson, nd[j].rson, mid + 1, rght, k -
                    (nd[nd[j].lson].size - nd[nd[i].lson].size));
        }
}st;
int n, m;
```

```
int a[maxn], b[maxn], rnk[maxn], mp[maxn];
bool cmp(int i, int j){return a[i] < a[j];}
int main(){
        scanf("%d%d", &n, &m);
        for(int i = 1; i <= n; ++i) scanf("%d", &a[i]);
        for(int i = 1; i <= n; ++i) rnk[i] = i;
        sort(rnk + 1, rnk + 1 + n, cmp);
        a[0] = inf;
        for(int i = 1, j = 0; i <= n; ++i){
                int k = rnk[i], kk = rnk[i-1];
                if(a[k] != a[kk]) b[k] = ++j;
                else b[k] = j;
                mp[b[k]] = a[k];
        }
        for(int i = 1; i <= n; ++i) st.insert(st.rt[i] = st.rt[i-1], 1, n, b
           [i]);
        for(int i = 1; i <= m; ++i){
                int x, y, k;
                scanf("%d%d%d", &x, &y, &k);
                printf("%d\n", mp[st.query(st.rt[x-1], st.rt[y], 1, n, k)]);
        }
        return 0;
}
```

2.5 树分治 by xyt

```
/*询问树上有多少对pair距离不超过k
  每次找重心 经过一些容斥
  求经过重心与不经过重心pair数*/
const int maxn = 1e4 + 5;
vector<pii> mp[maxn];
void add_edge(int u, int v, int d){
        mp[u].push_back(make_pair(v, d));
        mp[v].push_back(make_pair(u, d));
}
int n, ans, limit, gra, min_maxx;
int sz[maxn];
bool flag[maxn];
vector<int> vec;
void get_gra(int u, int fa, int nowsize){
        sz[u] = 1;
        int maxx = 0;
        for(int l = 0; l < mp[u].size(); ++l){
                int v = mp[u][l].first;
                if(v == fa || flag[v]) continue;
                get_gra(v, u, nowsize);
                sz[u] += sz[v];
```

```
                maxx = max(maxx, sz[v]);
        }
        maxx = max(maxx, nowsize - sz[u]);
        if(maxx < min_maxx) min_maxx = maxx, gra = u;
}
void get_dist(int u, int fa, int d){
        vec.push_back(d);
        for(int l = 0; l < mp[u].size(); ++l){
                int v = mp[u][l].first;
                if(v == fa || flag[v]) continue;
                get_dist(v, u, d + mp[u][l].second);
        }
}
int calc(int u, int delta){
        int rtn = 0;
        vec.clear();
        get_dist(u, 0, 0);
        sort(vec.begin(), vec.end());
        int m = vec.size();
        for(int i = 0, j = m - 1; i < j; ++i){
                while(i < j && vec[i] + vec[j] + delta > limit) --j;
                rtn += j - i;
        }
        return rtn;
}
void devide(int u, int nowsize){
        min_maxx = maxn;
        get_gra(u, 0, nowsize);
        flag[u=gra] = true;
        ans += calc(u, 0);
        for(int l = 0; l < mp[u].size(); ++l){
                int v = mp[u][l].first;
                if(flag[v]) continue;
                ans -= calc(v, mp[u][l].second * 2);
                devide(v, sz[v] > sz[u] ? nowsize - sz[u] : sz[v]);
        }
}
void init(){
        ans = 0;
        for(int i = 1; i <= n; ++i) mp[i].clear();
        memset(flag, 0, sizeof flag);
}
void work(){

        init();
        for(int i = 1; i < n; ++i){
                int u, v, d;
                scanf("%d%d%d", &u, &v, &d);
```

```
                add_edge(u, v, d);
        }
        devide(1, n);
        printf("%d\n", ans);
}
int main(){
        while(true){
                scanf("%d%d", &n, &limit);
                if(n == 0) break;
                work();
        }
        return 0;
}
```

2.6 树链剖分 by cjy

```
const int N = 800005;
int n, m, Max, b[N], edge_pos[N], path[N];
int tot, id[N * 2], nxt[N * 2], lst[N], val[N * 2];
int fa[N], siz[N], dep[N], hvy[N], top[N], pos[N];
struct Tree {
        int l, r;
        int mn, mx, sgn;
} h[N * 4];
void Add(int x, int y, int z) {
        id[++tot] = y; nxt[tot] = lst[x]; lst[x] = tot; val[tot] = z;
}
void dfs1(int x, int Fa) {
        fa[x] = Fa;
        siz[x] = 1;
        dep[x] = dep[Fa] + 1;
        int max_size = 0;
        for (int i = lst[x]; i; i = nxt[i]) {
                int y = id[i];
                if (y != Fa) {
                        path[y] = i; //-----------------
                        dfs1(y, x);
                        if (siz[y] > max_size) {
                                max_size = siz[y];
                                hvy[x] = y;
                        }
                        siz[x] += siz[y];
                }
        }
}
void dfs2(int x, int Top) {
        top[x] = Top;
```

```
        pos[x] = ++m;
        b[m] = val[path[x]]; //b[m] = val[x];
        edge_pos[path[x] / 2] = m; //when change only one edge's value
        if (hvy[x]) dfs2(hvy[x], Top); //heavy son need to be visited first
        for (int i = lst[x]; i; i = nxt[i]) {
                int y = id[i];
                if (y == fa[x] || y == hvy[x]) continue;
                dfs2(y, y);
        }
}
void work(int x, int y) {
        int X = top[x], Y = top[y];
        if (X == Y) {
                if (dep[x] < dep[y]) Negate(1, pos[x] + 1, pos[y]);
                else if (dep[x] > dep[y]) Negate(1, pos[y] + 1, pos[x]);
                //if (dep[x] <= dep[y]) Negate(1, pos[x], pos[y]);
                //else Negate(1, pos[y], pos[x]);
                return ;
        }
        if (dep[X] >= dep[Y]) {
                Negate(1, pos[X], pos[x]);
                work(fa[X], y);
        }
        else {
                Negate(1, pos[Y], pos[y]);
                work(x, fa[Y]);
        }
}
int main() {
        tot = 1; memset(lst, 0, sizeof(lst));
        memset(hvy, 0, sizeof(hvy));
        (Add_edge)
        dep[0] = 0; dfs1(1, 0); //the root is 1
        m = 0; dfs2(1, 1);
        build(1, 1, n);
        Change(1, edge_pos[x], y); //change one edge's valve directly in
           Tree
        work(x, y); //change value of a chain
        return 0;
}
```

2.7 树链剖分 by xyt

```
struct qtree{
        int tot;
        struct node{
                int hson, top, size, dpth, papa, newid;
```

```
}nd[maxn];

void find(int u, int fa, int d){
        nd[u].hson = 0;
        nd[u].size = 1;
        nd[u].papa = fa;
        nd[u].dpth = d;
        int max_size = 0;
        for(int l = 0; l < mp[u].size(); ++l){
                int v = mp[u][l].first;
                if(v == fa) continue;
                f[mp[u][l].second.second] = v;
                find(v, u, d + 1);
                nd[u].size += nd[v].size;
                if(max_size < nd[v].size){
                        max_size = nd[v].size;
                        nd[u].hson = v;
                }
        }
}
void connect(int u, int t){
        nd[u].top = t;
        nd[u].newid = ++tot;
        if(nd[u].hson != 0) connect(nd[u].hson, t);
        for(int l = 0; l < mp[u].size(); ++l){
                int v = mp[u][l].first;
                if(v == nd[u].papa || v == nd[u].hson) continue;
                connect(v, v);
        }
}
int query(int u, int v){
        int rtn = -inf;
        while(nd[u].top != nd[v].top){
                if(nd[nd[u].top].dpth < nd[nd[v].top].dpth) swap(u,
                   v);
                rtn = max(rtn, st.query(1, 1, n, nd[nd[u].top].newid
                   , nd[u].newid));
                u = nd[nd[u].top].papa;
        }
        if(nd[u].dpth > nd[v].dpth) swap(u, v);
        rtn = max(rtn, st.query(1, 1, n, nd[u].newid , nd[v].newid))
           ;
        return rtn;
}
void modify(int u, int v){
        while(nd[u].top != nd[v].top){
                if(nd[nd[u].top].dpth < nd[nd[v].top].dpth) swap(u,
                   v);
```

```
                        st.modify(1, 1, n, nd[nd[u].top].newid, nd[u].newid)
                            ;
                        u = nd[nd[u].top].papa;
                }
                if(nd[u].dpth > nd[v].dpth) swap(u, v);
                st.modify(1, 1, n, nd[u].newid + 1, nd[v].newid);
        }
        void clear(){
                tot = 0;
                nd[0].hson = nd[0].top = nd[0].size = nd[0].dpth = nd[0].
                    papa = nd[0].newid = 0;
                for(int i = 1; i <= n; ++i) nd[i] = nd[0];
        }
}qt;
```

2.8 点分治 by xyt

```
// POJ 1741
// 询问一棵树中有多少对点距离不超过 k
typedef pair<int, int> pii;
const int maxn = 1e4 + 5;
vector<pii> mp[maxn];
void add_edge(int u, int v, int d){
        mp[u].push_back(make_pair(v, d));
        mp[v].push_back(make_pair(u, d));
}
int n, ans, limit, gra, min_maxx;
int sz[maxn];
bool flag[maxn];
vector<int> vec;
void get_gra(int u, int fa, int nowsize){
        sz[u] = 1;
        int maxx = 0;
        for(int l = 0; l < mp[u].size(); ++l){
                int v = mp[u][l].first;
                if(v == fa || flag[v]) continue;
                get_gra(v, u, nowsize);
                sz[u] += sz[v];
                maxx = max(maxx, sz[v]);
        }
        maxx = max(maxx, nowsize - sz[u]);
        if(maxx < min_maxx) min_maxx = maxx, gra = u;
}
void get_dist(int u, int fa, int d){
        vec.push_back(d);
        for(int l = 0; l < mp[u].size(); ++l){
                int v = mp[u][l].first;
```

```
                if(v == fa || flag[v]) continue;
                get_dist(v, u, d + mp[u][l].second);
        }
}
int calc(int u, int delta){
        int rtn = 0;
        vec.clear();
        get_dist(u, 0, 0);
        sort(vec.begin(), vec.end());
        int m = vec.size();
        for(int i = 0, j = m - 1; i < j; ++i){
                while(i < j && vec[i] + vec[j] + delta > limit) --j;
                rtn += j - i;
        }
        return rtn;
}
void devide(int u, int nowsize){
        min_maxx = maxn;
        get_gra(u, 0, nowsize);
        flag[u=gra] = true;
        ans += calc(u, 0); // 加上经过重心的答案
        for(int l = 0; l < mp[u].size(); ++l){ // 容斥掉同一棵子树中经过重心
            的答案
                int v = mp[u][l].first;
                if(flag[v]) continue;
                ans -= calc(v, mp[u][l].second * 2);
                devide(v, sz[v] > sz[u] ? nowsize - sz[u] : sz[v]);
        }
}
void init(){
        ans = 0;
        for(int i = 1; i <= n; ++i) mp[i].clear();
        memset(flag, 0, sizeof flag);
}
void work(){

        init();
        for(int i = 1; i < n; ++i){
                int u, v, d;
                scanf("%d%d%d", &u, &v, &d);
                add_edge(u, v, d);
        }
        devide(1, n);
        printf("%d\n", ans);
}
int main(){
        while(true){
                scanf("%d%d", &n, &limit);
```

```
                if(n == 0) break;
                work();
        }
        return 0;
}
```

2.9 LCT by xyt

```
// 这个有些地方有点问题... // 标注部分
const int MAXN = 2e5 + 5;
int n, m;
struct Lct{
        struct Node{
                int sum;
                int lson, rson, fath, ance;
                bool lazy;
        };
        Node nd[MAXN];
        void push_up(int i){
                nd[i].sum = nd[nd[i].lson].sum + nd[nd[i].rson].sum + 1;
        }
        void reverse(int i){ //
                if(!i) return;
                swap(nd[i].lson, nd[i].rson);
                nd[i].lazy = true;
        }
        void push_down(int i){ //
                if(!i || !nd[i].lazy) return;
                reverse(nd[i].lson);
                reverse(nd[i].rson);
                nd[i].lazy = false;
        }
        void zig(int i){
                int j = nd[i].fath;
                int k = nd[j].fath;
                if(k && j == nd[k].lson) nd[k].lson = i;
                else if(k) nd[k].rson = i;
                nd[i].fath = k;
                nd[j].fath = i;
                nd[nd[i].rson].fath = j;
                nd[j].lson = nd[i].rson;
                nd[i].rson = j;
                nd[i].ance = nd[j].ance;
                push_up(j);
                push_up(i);
        }
        void zag(int i){
```

```
        int j = nd[i].fath;
        int k = nd[j].fath;
        if(k && j == nd[k].lson) nd[k].lson = i;
        else if(k) nd[k].rson = i;
        nd[i].fath = k;
        nd[j].fath = i;
        nd[nd[i].lson].fath = j;
        nd[j].rson = nd[i].lson;
        nd[i].lson = j;
        nd[i].ance = nd[j].ance;
        push_up(j);
        push_up(i);
}
void down_path(int i){ //
        if(nd[i].fath) down_path(nd[i].fath);
        push_down(i);
}
void splay(int i){
        down_path(i);
        while(nd[i].fath){
                int j = nd[i].fath;
                if(nd[j].fath == 0){
                        if(i == nd[j].lson) zig(i);
                        else zag(i);
                }else{
                        int k = nd[j].fath;
                        if(j == nd[k].lson){
                                if(i == nd[j].lson) zig(j), zig(i);
                                else zag(i), zig(i);
                        }else{
                                if(i == nd[j].rson) zag(j), zag(i);
                                else zig(i), zag(i);
                        }
                }
        }
}

void access(int i){
        int j = 0;
        while(i){
                splay(i);
                if(nd[i].rson){
                        nd[nd[i].rson].ance = i;
                        nd[nd[i].rson].fath = 0;
                }
                nd[i].rson = j;
                nd[j].fath = i;
                push_up(i);
```

```
                        j = i;
                        i = nd[i].ance;
                }
        }
        void set_root(int i){ //
                access(i);
                splay(i);
                reverse(i);
        }
        int find_root(int i){ //
                access(i);
                splay(i);
                while(nd[i].lson) i = nd[i].lson;
                splay(i);
                return i;
        }
        void link(int i, int j){ //
                set_root(i);
                nd[i].ance = j;
                access(i);
        }
        void cut(int i){ //
                access(i);
                splay(i);
                nd[nd[i].lson].ance = nd[i].ance;
                nd[nd[i].lson].fath = 0;
                nd[i].lson = 0;
                nd[i].ance = 0;
        }
};
Lct lct;
void query(){
        int pos;
        scanf("%d", &pos);
        ++pos;
        lct.access(pos);
        lct.splay(pos);
        printf("%d\n", lct.nd[pos].sum - 1);
}
void modify(){
        int pos, fath;
        scanf("%d%d", &pos, &fath);
        ++pos, fath += pos;
        if(fath > n) fath = n + 1;
        lct.splay(pos);
        if(lct.nd[pos].lson){
                lct.nd[lct.nd[pos].lson].ance = lct.nd[pos].ance;
                lct.nd[lct.nd[pos].lson].fath = 0;
```

```
                lct.nd[pos].lson = 0;
        }
        lct.nd[pos].ance = fath;
}
int main(){
        scanf("%d", &n);
        for(int i = 1; i <= n; ++i){
                int k;
                scanf("%d", &k);
                k += i;
                if(k > n) k = n + 1;
                lct.nd[i].ance = k;
        }
        for(int i = 1; i <= n + 1; ++i) lct.nd[i].sum = 1;
        scanf("%d", &m);
        for(int i = 1; i <= m; ++i){
                int k;
                scanf("%d", &k);
                if(k == 1) query();
                else modify();
        }
        return 0;
}
```

3 计算几何

3.1 向量旋转

```
void rotate(double theta){
        double coss = cos(theta), sinn = sin(theta);
        double tx = x * coss - y * sinn;
        double ty = x * sinn + y * coss;
        x = tx, y = ty;
}
```

3.2 至少被 i 个圆覆盖的面积

时间复杂度: n^2logn

```
const double pi=acos(-1);
const double eps=1e-12;
double sqr(double x){
        return x*x;
}
double sign(double x){
        return (x>eps)-(x<-eps);
}
double ans[2333];
```

```
int n;
struct P{
        double x,y;
        P(){}
        P(double x,double y):x(x),y(y){}
        void scan(){scanf("%lf%lf",&x,&y);}
        double sqrlen(){return (sqr(x)+sqr(y));}
        double len(){return sqrt(sqr(x)+sqr(y));}
        P zoom(double d){
                double l=d/len();
                return P(l*x,l*y);
        }
        P rev(){
                return P(y,-x);
        }
}dvd,a[233];
P centre[233];
double atan2(P x){
        return atan2(x.y,x.x);
}
P operator+(P a,P b){
        return P(a.x+b.x,a.y+b.y);
}
P operator-(P a,P b){
        return P(a.x-b.x,a.y-b.y);
}
double operator*(P a,P b){
        return a.x*b.y-a.y*b.x;
}
P operator*(double a,P b){
        return P(a*b.x,a*b.y);
}
P operator/(P a,double b){
        return P(a.x/b,a.y/b);
}
struct circle{
        double r;P o;
        circle(){}
        void scan(){
                o.scan();
                //scanf("%lf",&r);
        }
}cir[2333];
struct arc{
        double theta;
        int delta;
        P p;
        arc(){}
```

```
        arc(double theta,P p,int d):theta(theta),p(p),delta(d){}
}vec[4444];
int nV;
bool operator<(arc a,arc b){
        return a.theta+eps<b.theta;
}
int cnt;
void psh(double t1,P p1,double t2,P p2){
        if(t2+eps<t1)
                cnt++;
        vec[nV++]=arc(t1,p1,1);
        vec[nV++]=arc(t2,p2,-1);
}
void combine(int d,double area,P o){
        if(sign(area)==0)return;
        centre[d]=1/(ans[d]+area)*(ans[d]*centre[d]+area*o);
        ans[d]+=area;
}
bool equal(double x,double y){
        return x+eps>y and y+eps>x;
}
bool equal(P a,P b){
        return equal(a.x,b.x) and equal(a.y,b.y);
}
bool equal(circle a,circle b){
        return equal(a.o,b.o) and equal(a.r,b.r);
}
P p[4];
double cub(double x){return x*x*x;}

int main(){
    n = 0;
    cin>>n;
    for(int i = 0; i < n; ++i) cir[i].o.scan(), cin>>cir[i].r;
    for(int i = 0; i <= n; ++i) ans[i] = 0.0;
    for(int i = 0; i <= n; ++i) centre[i] = P(0, 0);
    for(int i=0;i<n;i++){
    dvd=cir[i].o-P(cir[i].r,0);
    nV=0;
    vec[nV++]=arc(-pi,dvd,1);
    cnt=0;
    for(int j=0;j<n;j++)if(i!=j){
        double d=(cir[j].o-cir[i].o).sqrlen();
            if(d<sqr(cir[j].r-cir[i].r)+eps){
                if(cir[i].r+i*eps<cir[j].r+j*eps)
                    psh(-pi,dvd,pi,dvd);
            }else if(d+eps<sqr(cir[j].r+cir[i].r)){
                double lambda=0.5*(1+(sqr(cir[i].r)-sqr(cir[j].r))/d);
```

```
                P cp=cir[i].o+lambda*(cir[j].o-cir[i].o);
                P nor((cir[j].o-cir[i].o).rev().zoom(sqrt(sqr(cir[i].r)-(cp-
                   cir[i].o).sqrlen())));
                P frm(cp+nor);
                P to(cp-nor);
                psh(atan2(frm-cir[i].o),frm,atan2(to-cir[i].o),to);
            }
        }
        sort(vec+1,vec+nV);
        vec[nV++]=arc(pi,dvd,-1);
        for(int j=0;j+1<nV;j++){
            cnt+=vec[j].delta;
            double theta=vec[j+1].theta-vec[j].theta;
            double area=sqr(cir[i].r)*theta*0.5;
            combine(cnt,area,cir[i].o+1.0/area/3*cub(cir[i].r)*P(sin(vec[j
               +1].theta)-sin(vec[j].theta),cos(vec[j].theta)-cos(vec[j+1].
               theta)));
            combine(cnt,-sqr(cir[i].r)*sin(theta)*0.5,1./3*(cir[i].o+vec[j].
               p+vec[j+1].p));
            combine(cnt,vec[j].p*vec[j+1].p*0.5,1.0/3*(vec[j].p+vec[j+1].p))
               ;
        }
    }
    printf("Case %d: ", Case);
    printf("%.3f\n\n",ans[1] );//ans[i]: 至少被i个圆覆盖的面积
        return 0;
}
```

3.3 计算几何杂

```
bool pit_on_seg(pit a, pit b, pit c){ // 点在线段上
        if(dcmp(det(b - a, c - a)) != 0) return false;
        if(dcmp(dot(a - b, a - c)) > 0) return false;
        return true;
}
bool pit_in_polygon(pit q){ // 点在多边形内
        int cnt = 0;
        for(int i = 1; i <= n; ++i){
                pit p1 = p[i];
                pit p2 = p[suc[i]];
                if(pit_on_seg(q, p1, p2)) return true;
                int k = dcmp(det(p2 - p1, q - p1));
                int d1 = dcmp(p1.y - q.y);
                int d2 = dcmp(p2.y - q.y);
                if(k > 0 && d1 <= 0 && d2 > 0) ++cnt;
                if(k < 0 && d2 <= 0 && d1 > 0) --cnt;
        }
```

```
        if(cnt != 0) return true;
        else return false;
}
bool seg_in_polygon(pit a, pit b){ // 线段在多边形内 撒点
        vec v = b - a;
        for(int t = 1; t <= 1000; ++t){
                pit c = a + v * (1.00 * (rand() % 10000) / 10000);
                if(pit_in_polygon(c)) continue;
                else return false;
        }
        return true;
}
```

3.4 三维变换

```
struct Matrix{
    double a[4][4];
    int n,m;
    Matrix(int n = 4):n(n),m(n){
                for(int i = 0; i < n; ++i)
                a[i][i] = 1;
        }
    Matrix(int n, int m):n(n),m(m){}
    Matrix(Point A){
        n = 4;
        m = 1;
        a[0][0] = A.x;
        a[1][0] = A.y;
        a[2][0] = A.z;
        a[3][0] = 1;
    }
//+-略
    Matrix operator *(const Matrix &b)const{
        Matrix ans(n,b.m);
        for (int i = 0; i < n; ++i)
        for (int j = 0; j < b.m; ++j)
        {
            ans.a[i][j] = 0;
            for (int k = 0; k < m; ++k)
            ans.a[i][j] += a[i][k] * b.a[k][j];
        }
        return ans;
    }
    Matrix operator * (double k)const{
        Matrix ans(n,m);
        for (int i = 0; i < n; ++i)
        for (int j = 0; j < m; ++j)
```

```
        ans.a[i][j] = a[i][j] * k;
        return ans;
    }
};
Matrix cur(4), I(4);

Point get(int i){//以下三个是变换矩阵，get是使用方法
    Matrix ori(p[i]);
    ori = cur * ori;
    return Point(ori.a[0][0],ori.a[1][0],ori.a[2][0]);
}

void trans(){//平移
    int l,r;
    Point vec;
    vec.read();
    cur = I;
    cur.a[0][3] = vec.x;
    cur.a[1][3] = vec.y;
    cur.a[2][3] = vec.z;
}
void scale(){//以base为原点放大k倍
    Point base;
    base.read();
    scanf("%lf",&k);
    cur = I;
    cur.a[0][0] = cur.a[1][1] = cur.a[2][2] = k;
    cur.a[0][3] = (1.0 - k) * base.x;
    cur.a[1][3] = (1.0 - k) * base.y;
    cur.a[2][3] = (1.0 - k) * base.z;
}
void rotate(){//绕以base为起点vec为方向向量的轴逆时针旋转theta
    Point base,vec;
    base.read();
    vec.read();
    double theta;
    scanf("%lf",&theta);
    if (dcmp(vec.x)==0&&dcmp(vec.y)==0&&dcmp(vec.z)==0)return;
    double C = cos(theta), S = sin(theta);
    vec = vec / len(vec);
    Matrix T1,T2;
    T1 = T2 = I;
    T1.a[0][3] = base.x;
    T1.a[1][3] = base.y;
    T1.a[2][3] = base.z;
    T2.a[0][3] = -base.x;
    T2.a[1][3] = -base.y;
    T2.a[2][3] = -base.z;
```

```
    cur = I;
    cur.a[0][0] = sqr(vec.x) * (1 - C) + C;
    cur.a[0][1] = vec.x * vec.y * (1-C) - vec.z * S;
    cur.a[0][2] = vec.x * vec.z * (1-C) + vec.y * S;
    cur.a[1][0] = vec.x * vec.y * (1-C) + vec.z * S;
    cur.a[1][1] = sqr(vec.y) * (1-C) + C;
    cur.a[1][2] = vec.y * vec.z * (1-C) - vec.x * S;
    cur.a[2][0] = vec.x * vec.z * (1-C) - vec.y * S;
    cur.a[2][1] = vec.y * vec.z * (1-C) + vec.x * S;
    cur.a[2][2] = vec.z * vec.z * (1-C) + C;
    cur = T1 * cur * T2;
}
```

4 字符串

4.1 AC-Automachine by cjy

```
#define N 1500
int next[N][10], flag[N], fail[N], a[N];
int m, ans, root;
int newnode(){
        m++;
        for (int i = 1; i <= 4; i++)
                next[m][i] = -1;
        flag[m] = 1;
        return m;
}
void init(){
        m = -1;
        root = newnode();
}
void insert(char s[]){
        int len = strlen(s+1);
        int now = root;
        for (int i = 1; i <= len; i++){
                int t = id(s[i]);
                if (next[now][t] == -1)
                        next[now][t] = newnode();
                now = next[now][t];
        }
        flag[now] = 0;
}
void build(){
        queue<int> Q;
        fail[root] = root;
        for (int i = 1; i <= 4; i++)
                if (next[root][i] == -1)
```

```
                        next[root][i] = root;
                else{
                        fail[next[root][i]] = root;
                        flag[next[root][i]] &= flag[root];
                        Q.push(next[root][i]);
                }
        while (!Q.empty()){
                int now = Q.front();
                Q.pop();
                for (int i = 1; i <= 4; i++)
                        if (next[now][i] == -1)
                                next[now][i] = next[fail[now]][i];
                        else{
                                fail[next[now][i]] = next[fail[now]][i];
                                flag[next[now][i]] &= flag[next[fail[now]][i
                                   ]];
                                Q.push(next[now][i]);
                        }
        }
}
char s[1005];
int main(){
        int n;
        int cases = 0;
        while(scanf("%d", &n), n){
                init();
                for (int i = 1; i <= n; i++){
                        scanf("%s", s+1);
                        insert(s);
                }
                build();
        }
        return 0;
}
```

4.2 AC-Automachine by xyt

```
struct trie{
    int size, indx[maxs][26], word[maxs], fail[maxs];
    bool jump[maxs];
    int idx(char ff){return ff - 'a';}
    void insert(char s[]){
        int u = 0;
        for(int i = 0; s[i]; ++i){
            int k = idx(s[i]);
            if(!indx[u][k]) indx[u][k] = ++size;
            u = indx[u][k];
```

```
        }
        word[u] = 1;
        jump[u] = true;
    }
    void get_fail(){
        queue<int> que;
        int head = 0, tail = 0;
        que.push(0);
        while(!que.empty()){
            int u = que.front();
            que.pop();
            for(int k = 0; k < 26; ++k){
                if(!indx[u][k]) continue;
                int v = indx[u][k];
                int p = fail[u];
                while(p && !indx[p][k]) p = fail[p];
                if(indx[p][k] && indx[p][k] != v) p = indx[p][k];
                fail[v] = p;
                jump[v] |= jump[p];
                que.push(v);
            }
        }
    }
    int query(char s[]){
        int rtn = 0, p = 0;
        int flag[maxs];
        memcpy(flag, word, sizeof flag);
        for(int i = 0; s[i]; ++i){
            int k = idx(s[i]);
            while(p && !indx[p][k]) p = fail[p];
            p = indx[p][k];
            int v = p;
            while(jump[v]){
                rtn += flag[v];
                flag[v] = 0;
                v = fail[v];
            }
        }
        return rtn;
    }

} dict;
```

4.3 后缀数组

```
//sa[i] 表示排第 i 位的后缀是谁 rk[i] 表示后缀 i 排第几位
//h[i] 为 suffix(sa[i-1]) 和 suffix(sa[i]) 的最长公共前缀
```

```
//开数组要*2
inline void getsa(int j){
        memset(sum,0,sizeof(sum));
    for(int i=1;i<=n;++i) ++sum[rk[i+j]];
    for(int i=1;i<=n;++i) sum[i]+=sum[i-1];
    for(int i=n;i>0;--i) tsa[sum[rk[i+j]]--]=i;
    memset(sum,0,sizeof(sum));
    for(int i=1;i<=n;++i) ++sum[rk[i]];
    for(int i=1;i<=n;++i) sum[i]+=sum[i-1];
    for(int i=n;i>0;--i) sa[sum[rk[tsa[i]]]--]=tsa[i];
}
int main(){
        scanf("%s",s+1); n=strlen(s+1);
    for(int i=1;i<=n;++i) ++sum[s[i]];
    for(int i=1;i<=MC;++i) sum[i]+=sum[i-1];
        for(int i=n;i>0;--i) sa[sum[s[i]]--]=i;
    rk[sa[1]]=1;
    for(int i=2,p=1;i<=n;++i){
        if(s[sa[i]]!=s[sa[i-1]]) ++p;
        rk[sa[i]]=p;
    }
    for(int j=1;j<=n;j<<=1){
        getsa(j);
        trk[sa[1]]=1;
        for(int i=2,p=1;i<=n;++i){
            if(rk[sa[i]]!=rk[sa[i-1]] || rk[sa[i]+j]!=rk[sa[i-1]+j]) ++p;
            trk[sa[i]]=p;
        }
        for(int i=1;i<=n;++i) rk[i]=trk[i];
        }
        for(int i=1;i<=n;++i) printf("%d ",sa[i]); printf("\n");
    for(int i=1,j=0;i<=n;++i){
        if(rk[i]==1) continue;
        while(i+j<=n && sa[rk[i]-1]+j<=n && s[i+j]==s[sa[rk[i]-1]+j]) ++j;
        h[rk[i]]=j;
        if(j>0) --j;
        }
        for(int i=1;i<=n;++i) printf("%d ",h[i]); printf("\n");
    return 0;
}
```

4.4 扩展 KMP

```
// (1-base) next[i] = lcp(text[1..n], text[i..n]), text[1..next[i]] = text[i
   ..(i + next[i] - 1)]
void build(char *pattern) {
        int len = strlen(pattern + 1);
```

```
        int j = 1, k = 2;
        for (; j + 1 <= len && pattern[j] == pattern[j + 1]; j++);
        next[1] = len;
        next[2] = j - 1;
        for (int i = 3; i <= len; i++) {
                int far = k + next[k] - 1;
                if (next[i - k + 1] < far - i + 1) {
                        next[i] = next[i - k + 1];
                }
                else {
                        j = max(far - i + 1, 0);
                        for (; i + j <= len && pattern[1 + j] == pattern[i +
                            j]; j++);
                        next[i] = j;
                        k = i;
                }
        }
}
void solve(char *text, char *pattern) {
        int len = strlen(text + 1);
        int lenp = strlen(pattern + 1);
        int j = 1, k = 1;
        for (; j <= len && j <= lenp && pattern[j] == text[j]; j++);
        extend[1] = j - 1;
        for (int i = 2; i <= len; i++) {
                int far = k + extend[k] - 1;
                if (next[i - k + 1] < far - i + 1) {
                        extend[i] = next[i - k + 1];
                }
                else {
                        j = max(far - i + 1, 0);
                        for (; i + j <= len && 1 + j <= lenp && pattern[1 +
                           j] == text[i + j]; j++);
                        extend[i] = j;
                        k = i;
                }
        }
}
```

4.5 回文树

```
const int N = 400010;
int ch[N][26], fail[N], len[N], tot, cnt1[N], cnt2[N];
char s[200010];

void ready(){
        len[0] = 0;  len[1] = -1;
```

```
        fail[0] = 1; fail[1] = -1;
}
void Insert(char *s, int *cnt){
        int now = 1, l = strlen(s), x, y, tmp;
        for(int i = 0; i < l; i ++){
                x = s[i] - 'a';
                while(s[i] != s[i - len[now] - 1]) now = fail[now];
                if (!ch[now][x]){
                        ch[now][x] = ++ tot;
                        len[tot] = len[now] + 2;
                }
                y = ch[now][x];
                tmp = fail[now];
                if (tmp == -1) fail[y] = 0;
                else{
                        while(s[i] != s[i - len[tmp] - 1]) tmp = fail[tmp];
                        fail[y] = ch[tmp][x];
                }
                now = y;
                cnt[now] ++;
        }
}
int main(){
        int T, tests = 0;
        scanf("%d", &T);
        while(tests < T){
                for(int i = 0; i <= tot; i ++){
                        for(int j = 0; j < 26; j ++) ch[i][j] = 0;
                        len[i] = cnt1[i] = cnt2[i] = 0;
                }
                tot = 1;
                ready();
                scanf("%s", s);
                Insert(s, cnt1);
                scanf("%s", s);
                Insert(s, cnt2);
                for(int i = tot; i >= 2; i --)
                        cnt1[fail[i]] += cnt1[i],
                        cnt2[fail[i]] += cnt2[i];
                LL ans = 0;
                for(int i = 2; i <= tot; i ++) ans += 1LL * cnt1[i] * cnt2[i
                   ];
                printf("Case #%d: %lld\n",++tests, ans);
        }
        return 0;
}
```

4.6 SAM by lss

```
const int L = 600005;//n * 2 开大一点，只开n会挂
struct Node
{
        Node *nx[26], *fail;
        int l, num;
};
Node *root, *last, sam[L], *b[L];
int sum[L], f[L];
int cnt;
char s[L];
int l;
void add(int x)
{
        ++cnt;
        Node *p = &sam[cnt];
        Node *pp = last;
        p->l = pp->l + 1;
        last = p;
        for(; pp && !pp->nx[x]; pp = pp->fail) pp->nx[x] = p;
        if(!pp) p->fail = root;
        else{
                if(pp->l + 1 == pp->nx[x]->l) p->fail = pp->nx[x];
                else{
                        ++cnt;
                        Node *r = &sam[cnt], *q = pp->nx[x];
                        *r = *q;
                        r->l = pp->l + 1;
                        q->fail = p->fail = r;
                        for(; pp && pp->nx[x] == q; pp = pp->fail) pp->nx[x]
                            = r;
                }
        }
}
int main()
{
        scanf("%s", s);
        l = strlen(s);
        root = last = &sam[0];
        for(int i = 0; i < l; ++i) add(s[i] - 'a');
        for(int i = 0; i <= cnt; ++i) ++sum[sam[i].l];
        for(int i = 1; i <= l; ++i) sum[i] += sum[i - 1];
        for(int i = 0; i <= cnt; ++i) b[--sum[sam[i].l]] = &sam[i];
        Node *now = root;
        for(int i = 0; i < l; ++i){
                now = now->nx[s[i] - 'a'];
                ++now->num;
```

```
        }
        for(int i = cnt; i > 0; --i){
                int len = b[i]->l;
                //cerr<<"num="<<b[i]->num<<endl;
                f[len] = max(f[len], b[i]->num);
                //cerr<<b[i]->num<<" "<<b[i]->fail->num<<" ..."<<endl;
                b[i]->fail->num += b[i]->num;
                //cerr<<b[i]->num<<" "<<b[i]->fail->num<<" ..."<<endl;
        }
        for(int i = l - 1; i >= 1; --i) f[i] = max(f[i], f[i + 1]);
        for(int i = 1; i <= l; ++i) printf("%d\n", f[i]);
        return 0;
}
```

4.7 SAM by xyt

```
const int maxn = 351000;
struct sam{
        int tot, lst;
        struct node{
                int indx[26], fa, lnth, rts;
                void init(){
                        fa = -1;
                        lnth = rts = 0;
                        memset(indx, -1, sizeof indx);
                }
        }nd[maxn];
        void init(){
                tot = lst = 0;
                nd[tot].init();
        }
        int newnode(){
                nd[++tot].init();
                return tot;
        }
        void insert(char ch){
                int c = ch - 'a';
                int newp = newnode(), p = lst;
                nd[newp].lnth = nd[p].lnth + 1;
                while(p != -1 && nd[p].indx[c] == -1){
                        nd[p].indx[c] = newp;
                        p = nd[p].fa;
                }
                if(p == -1) nd[newp].fa = 0;
                else{
                        int q = nd[p].indx[c];
                        if(nd[p].lnth + 1 == nd[q].lnth) nd[newp].fa = q;
```

```
                    else{
            int newq = newnode();
            nd[newq] = nd[q];
            nd[newq].lnth = nd[p].lnth + 1;
            nd[q].fa = nd[newp].fa = newq;
            while(p != -1 && nd[p].indx[c] == q){
                nd[p].indx[c] = newq;
                p = nd[p].fa;
            }
                    }
                }
                lst = newp;
        }

}dict;
bool cmp(int i, int j){
        return dict.nd[i].lnth > dict.nd[j].lnth;
}
int n, ans[maxn], rk[maxn];
char str[maxn];
void work(){
        dict.init();
        n = strlen(str);
        for(int i = 0; i < n; ++i) dict.insert(str[i]);
        for(int i = 1; i <= dict.tot; ++i) rk[i] = i;
        sort(rk + 1, rk + 1 + dict.tot, cmp);
        for(int i = 0, p = 0; i < n; ++i)
                dict.nd[p=dict.nd[p].indx[str[i]-'a']].rts = 1;
        for(int i = 1; i <= dict.tot; ++i){
                int p = rk[i];
                ans[dict.nd[p].lnth] = max(ans[dict.nd[p].lnth], dict.nd[p].
                   rts);
                dict.nd[dict.nd[p].fa].rts += dict.nd[p].rts;
        }
        for(int i = n; i >= 1; --i) ans[i-1] = max(ans[i-1], ans[i]);
        for(int i = 1; i <= n; ++i) printf("%d\n", ans[i]);
}
```

5 图论

5.1 图论相关

1. 差分约束系统
(1) 以 x[i] - x[j] <= c 为约束条件，j -> i : c，求最短路得到的是 x[i] <= x[s] 的最大解，存在负权回路无解
(2) 以 x[i] - x[j] >= c 为约束条件，j -> i : c，求最长路得到的时 x[i] >= x[s] 的最小解，存在正权回路无解 // 若有 x[i] = x[j] 则 i <-0-> j
2. 最大闭合权子图

s 向正权点连边，负权点向 t 连边，边权为点权绝对值，再按原图连边，边权为 INF
3. 最大密度子图：$\max\frac{|E'|}{|V'|}$
（1）猜测答案 g 若最大流大于 EPS 则 g 合法
（2）s -> v: INF, u -> t : INF + g - deg[u], u -> v : 1.00
4. 2-SAT
利用对称性建图，若 u 与 u' 在同一强连通分量中，则无解，若有解输出方案，拓扑排序后自底向上（从 ind = 0 到 otd = 0）选择删除
5. 最小割
（1）二分图最小点权覆盖集：s -> u : w[u], u -> v : INF, v -> t : w[v]

5.2 SteinerTree by cjy

```
const int N = 100005;
const int M = 200005;
const int P = 8;
const int inf = 0x3f3f3f3f;
int n, m, p, status, idx[P], f[1 << P][N];
//int top, h[N];
priority_queue<pair<int, int> > q;
bool vis[N];
int tot, lst[N], nxt[M], id[M], len[M];
void Add(int x, int y, int z) {
        id[++tot] = y; nxt[tot] =lst[x]; lst[x] = tot; len[tot] = z;
}
void dijkstra(int dis[]) {
        while(!q.empty()) {
                int x = q.top().second; q.pop();
                if (vis[x]) continue;
                vis[x] = 1;
                for (int i = lst[x]; i; i = nxt[i]) {
                        int y = id[i];
                        if (dis[x] + len[i] < dis[y]) {
                                dis[y] = dis[x] + len[i];
                                if (!vis[y]) q.push(make_pair(-dis[y], y));
                        }
                }
        }
}
void Steiner_Tree() {
        for (int i = 1; i < status; i++) {
                //top = 0;
                while (!q.empty()) q.pop();
                memset(vis, 0, sizeof(vis));
                for (int j = 1; j <= n; j++) {
                        for (int k = i & (i - 1); k; (--k) &= i)
                                f[i][j] = min(f[i][j], f[k][j] + f[i ^ k][j
                                   ]);
```

```
                        if (f[i][j] != inf) {
                                //h[++top] = j, vis[j] = 1;
                                q.push(make_pair(-f[i][j], j));
                        }
                }
                //SPFA(f[i]);
                dijkstra(f[i]);
        }
}
int main() {
        while (scanf("%d%d%d", &n, &m, &p) == 3) {
                status = 1 << p;
                tot = 0; memset(lst, 0, sizeof(lst));
                /*求最小生成森林
                  每棵生成树中至少选择一个点，点权为代价
                  新开一个空白关键点作为源
          for (int i = 1; i <= n; i++) {
           scanf("%d", &val[i]);
           Add(0, i, val[i]); Add(i, 0, val[i]);
          }*/
                for (int i = 1; i <= m; i++) {
                        int x, y, z;
                        scanf("%d%d%d", &x, &y, &z);
                        Add(x, y, z); Add(y, x, z);
                }
                for (int i = 1; i <= p; i++) scanf("%d", &idx[i]);
                memset(f, 0x3f, sizeof(f));
                for (int i = 1; i <= n; i++) f[0][i] = 0;
                for (int i = 1; i <= p; i++)
                        f[1 << (i - 1)][idx[i]] = 0;
                Steiner_Tree();
                int ans = inf;
                for (int i = 1; i <= n; i++) ans = min(ans, f[status - 1][i
                   ]);
                printf("%d\n", ans);
        }
        return 0;
}
```

5.3 LCA by xyt

```
int maxbit, dpth[maxn], ance[maxn][maxb];
void dfs(int u, int fath){
    dpth[u] = dpth[fath] + 1; ance[u][0] = fath;
    for(int i = 1; i <= maxbit; ++i) ance[u][i] = ance[ance[u][i-1]][i-1];
    for(int l = last[u]; l; l = next[l]){
        int v = dstn[l];
```

```
        if(v == fath) continue;
        dfs(v, u);
    }
}
int lca(int u, int v){
    if(dpth[u] < dpth[v]) swap(u, v);
    int p = dpth[u] - dpth[v];
    for(int i = 0; i <= maxbit; ++i)
        if(p & (1 << i)) u = ance[u][i];
    if(u == v) return u;
    for(int i = maxbit; i >= 0; --i){
        if(ance[u][i] == ance[v][i]) continue;
        u = ance[u][i]; v = ance[v][i];
    }
    return ance[u][0];
}
```

5.4 KM

```
int weight[M][M], lx[M], ly[M];
bool sx[M], sy[M];
int match[M];
bool search_path(int u){
        sx[u] = true;
        for (int v = 0; v < n; v++){
                if (!sy[v] && lx[u] + ly[v] == weight[u][v]){
                        sy[v] = true;
                        if (match[v] == -1 || search_path(match[v])){
                                match[v] = u;
                                return true;
                        }
                }
        }
        return false;
}

int KM()
{
        for (int i = 0; i < n; i++){
                lx[i] = ly[i] = 0;
                for (int j = 0; j < n; j++)
                        if (weight[i][j] > lx[i])
                                lx[i] = weight[i][j];
        }
        memset(match, -1, sizeof(match));
        for (int u = 0; u < n; u++){
                while (1){
```

```
                        memset(sx, 0, sizeof(sx));
                        memset(sy, 0, sizeof(sy));
                        if (search_path(u)) break;

                        int inc = len * len;
                        for (int i = 0; i < n; i++)
            if (sx[i])
            for (int j = 0; j < n; j++)
            if (!sy[j] && ((lx[i] + ly[j] - weight[i][j]) < inc))
                inc = lx[i] + ly[j] - weight[i][j];
                        for (int i = 0; i < n; i++){
                                if (sx[i]) lx[i] -= inc;
                                if (sy[i]) ly[i] += inc;
                        }
                }
        }
        int sum = 0;
        for (int i = 0; i < n; i++)
    if (match[i] >= 0) sum += weight[match[i]][i];
        return sum;
}

int main()
{
        memset(weight, 0, sizeof(weight));
        for (int i = 1; i <= len; i++)
                weight[a[i]][b[i]]++;
        cout<<KM()<<endl;
        return 0;
}
```

5.5 KM 三次方

```
const int N=1010;
const int INF = 1e9;
int n;
struct KM{
int w[N][N];
int lx[N], ly[N], match[N],way[N], slack[N];
bool used[N];
void initialization(){
    for(int i = 1; i <= n; i++){
        match[i] = 0;
        lx[i] = 0;
        ly[i] = 0;
        way[i] = 0;
    }
```

```
}
void hungary(int x){//for i(1 -> n) : hungary(i);
    match[0] = x;
    int j0 = 0;
    for(int j = 0; j <= n; j++){
        slack[j] = INF;
        used[j] = false;
    }
    do{
        used[j0] = true;
        int i0 = match[j0], delta = INF, j1;
        for(int j = 1; j <= n; j++){
            if(used[j] == false){
                int cur = -w[i0][j] - lx[i0] - ly[j];
                if(cur < slack[j]){
                    slack[j] = cur;
                    way[j] = j0;
                }
                if(slack[j] < delta){
                    delta = slack[j];
                    j1 = j;
                }
            }
        }
        for(int j = 0; j <= n; j++){
            if(used[j]){
                lx[match[j]] += delta;
                ly[j] -= delta;
            }
            else slack[j] -= delta;
        }
        j0 = j1;
    }while (match[j0] != 0);

    do{
        int j1 = way[j0];
        match[j0] = match[j1];
        j0 = j1;
    }while(j0);
}
    int get_ans(){//maximum ans
    int sum = 0;
    for(int i = 1; i<= n; i++)
        if(match[i] > 0) sum += -w[match[i]][i];
    return sum;
}
}KM_solver;
```

5.6 Dinic by cjy

```
const int N = 20000;
const int inf = 100000;
int tot, id[N], nxt[N], lst[N], cap[N];
int d[N];
queue<int> Q;
void Add(int x, int y, int z) {
        id[++tot] = y; nxt[tot] = lst[x]; lst[x] = tot; cap[tot] = z;
        id[++tot] = x; nxt[tot] = lst[y]; lst[y] = tot; cap[tot] = 0;
}
bool bfs() {
        while (!Q.empty()) Q.pop();
        Q.push(S);
        memset(d, 0, sizeof(d)); d[S] = 1;
        while (!Q.empty()) {
                int x = Q.front(); Q.pop();
                for (int i = lst[x]; i; i = nxt[i]) {
                        int y = id[i];
                        if (cap[i] && !d[y]) {
                                d[y] = d[x] + 1;
                                if (y == T) return true;
                                Q.push(y);
                        }
                }
        }
        return false;
}
int find(int x, int flow) {
        if (x == T) return flow;
        int res = 0;
        for (int i = lst[x]; i; i = nxt[i]) {
                int y = id[i];
                if (cap[i] && d[y] == d[x] + 1) {
                        int now = find(y, min(flow - res, cap[i]));
                        res += now;
                        cap[i] -= now, cap[i ^ 1] += now;
                }
        }
        if (!res) d[x] = -1;
        return res;
}
int dinic() {
        int ans = 0;
        while (bfs())
                ans += find(S, inf);
        return ans;
}
```

```
int main() {
        tot = 1; memset(lst, 0, sizeof(lst));

        printf("%d\n", dinic());
        return 0;
}
```

5.7 网络流 by xyt

```
// sap
struct edge{
    int v, r, flow;
    edge(int v, int flow, int r) : v(v), flow(flow), r(r) {}
};
vector<edge> mp[maxn];
void add_edge(int u, int v, int flow){
    mp[u].push_back(edge(v, flow, mp[v].size()));
    mp[v].push_back(edge(u, 0, mp[u].size() - 1));
}
int maxflow, disq[maxn], dist[maxn];
int sap(int u, int nowflow){
    if(nowflow == 0 || u == T) return nowflow;
    int tempflow, deltaflow = 0;
    for(int l = 0; l < mp[u].size(); ++l){
        int v = mp[u][l].v;
        if(mp[u][l].flow > 0 && dist[u] == dist[v] + 1){
            tempflow = sap(v, min(nowflow - deltaflow, mp[u][l].flow));
            mp[u][l].flow -= tempflow;
            mp[v][mp[u][l].r].flow += tempflow;
            deltaflow += tempflow;
            if(deltaflow == nowflow || dist[S] >= T) return deltaflow;
        }
    }
    disq[dist[u]]--;
    if(disq[dist[u]] == 0) dist[S] = T;
    dist[u]++;
    disq[dist[u]]++;
    return deltaflow;
}
int main(){
    while(dist[S] < T) maxflow += sap(S, inf);
}
// 费用流
struct edge{
    int v, r, cost, flow;
    edge(int v, int flow, int cost, int r) : v(v), flow(flow), cost(cost), r
       (r) {}
```

```
};
vector<edge> mp[maxn];
void add_edge(int u, int v, int flow, int cost){
    mp[u].push_back(edge(v, flow, cost, mp[v].size()));
    mp[v].push_back(edge(u, 0, -cost, mp[u].size() - 1));
}
int S, T, maxflow, mincost;
int dist[maxn], pth[maxn], lnk[maxn];
bool inq[maxn];
queue<int> que;
bool find_path(){
    for(int i = 1; i <= T; ++i) dist[i] = inf;
    dist[S] = 0;
    que.push(S);
    while(!que.empty()){
        int u = que.front();
        que.pop();
        inq[u] = false;
        for(int l = 0; l < mp[u].size(); ++l){
            int v = mp[u][l].v;
            if(mp[u][l].flow > 0 && dist[v] > dist[u] + mp[u][l].cost){
                dist[v] = dist[u] + mp[u][l].cost;
                pth[v] = u;
                lnk[v] = l;
                if(!inq[v]){
                    inq[v] = true;
                    que.push(v);
                }
            }
        }
    }
    if(dist[T] < inf) return true;
    else return false;
}
void adjust(){
    int deltaflow = inf, deltacost = 0;
    for(int v = T; v != S; v = pth[v]){
        deltaflow = min(deltaflow, mp[pth[v]][lnk[v]].flow);
        deltacost += mp[pth[v]][lnk[v]].cost;
    }
    maxflow += deltaflow;
    mincost += deltaflow * deltacost;
    for(int v = T; v != S; v = pth[v]){
        mp[pth[v]][lnk[v]].flow -= deltaflow;
        mp[mp[pth[v]][lnk[v]].v][mp[pth[v]][lnk[v]].r].flow += deltaflow;
    }
}
int main(){while(find_path()) adjust();}
```

5.8 最大密度子图

```
#define rep(i, l, r) for(int i = l; i <= r; ++i)
using namespace std;
const double INF = 1e7;
const int maxn = 2e4 + 10;
const int maxm = 2e5 + 10;
int h[maxn], head[maxn], nex[maxm], to[maxm], last[maxn], lastedge[maxn],
   tot, s, t;
int gap[maxn], cur[maxn], d[maxn], n;
double maxflow;
double flow[maxm];
void clear(){
        rep(i, 1, tot) head[i] = 0;
        tot = 1;
        maxflow = 0;
}
void add(int l, int r, double f){
        nex[++tot] = head[l]; head[l] = tot; to[tot] = r; flow[tot] = f;
        nex[++tot] = head[r]; head[r] = tot; to[tot] = l; flow[tot] = 0;
}
void augment(){
        double f = INF;
        for(int pos = t; pos != s; pos = last[pos])
                if(flow[lastedge[pos]] < f) f = flow[lastedge[pos]];
        for(int pos = t; pos != s; pos = last[pos])
                flow[lastedge[pos]] -= f, flow[lastedge[pos] ^ 1] += f;
        maxflow += f;
}
void isap(){
        int pos = s;
        rep(i, 1, t) gap[i] = 0;
        gap[0] = t;
        rep(i, 1, t) cur[i] = head[i], d[i] = 0;
        while(d[s] < t){
                if(pos == t){augment(); pos = s;}
                bool flag = 0;
                for(int u = cur[pos]; u; u = nex[u])
                        if(flow[u] > 0 && d[to[u]] + 1 == d[pos]){
                                flag = 1;
                                cur[pos] = u;
                                last[to[u]] = pos;
                                lastedge[to[u]] = u;
                                pos = to[u];
                                break;
                        }
                if(!flag){
                        int f = t - 1;
```

```
                        for(int u = head[pos]; u; u = nex[u])
                                if(flow[u] > 0  && d[to[u]] < f) f = d[to[u
                                    ]];
                        if(!--gap[d[pos]]) break;
                        gap[d[pos] = f + 1] ++;
                        cur[pos] = head[pos];
                        if(pos != s) pos = last[pos];
                }
        }
}
bool check(double v){
        clear();
        int sum = n;
        rep(i, 2, n)
                rep(j, 1, i - 1)
                        if(h[j] > h[i]) {
                add(++sum, i, INF);
                                add(sum, j, INF);
                        }
        s = sum + 1;
        t = s + 1;
        rep(i, n + 1, sum) add(s, i, 1.0);
        rep(i, 1, n) add(i, t, v);
        isap();
        return sum - n - maxflow > 1e-10;
}
int main(){
        int T;
        cin >> T;
        rep(cas, 1, T){
                printf("Case #%d: ", cas);
                cin >> n;
                rep(i, 1, n) scanf("%d", &h[i]);
                double l = 0, r = n, ans = 0;
                while(r - l > 1e-9){
                        double mid = (l + r) / 2;
                        if(check(mid)){
                                if(mid > ans) ans = mid;
                                l = mid;
                        }
                        else r = mid;
                }
                printf("%.8f\n", ans);
        }
        return 0;
}

const int maxn = 1e2 + 5;
```

```
const double eps = 1e-10;
const double d = 1e2;
const double inf = 1e9;
struct edge{
        int r, v;
        double flow;
        edge(int v, int r, double flow) : v(v), r(r), flow(flow) {}
};
vector<edge> mp[maxn];
void add_edge(int u, int v, double flow){
        mp[u].push_back(edge(v, mp[v].size(), flow));
        mp[v].push_back(edge(u, mp[u].size() - 1, 0.00));
}
int n, m, S, T, a[maxn], deg[maxn];
int dist[maxn], disq[maxn];
double sap(int u, double nowflow){
        if(u == T || nowflow < eps) return nowflow;
        double tempflow, deltaflow = 0.00;
        for(int l = 0; l < mp[u].size(); ++l){
                int v = mp[u][l].v;
                double flow = mp[u][l].flow;
                if(flow > eps && dist[u] == dist[v] + 1){
                        tempflow = sap(v, min(flow, nowflow - deltaflow));
                        mp[u][l].flow -= tempflow;
                        mp[v][mp[u][l].r].flow += tempflow;
                        deltaflow += tempflow;
                        if(deltaflow == nowflow || dist[S] == T) return
                           deltaflow;
                }
        }
        disq[dist[u]]--;
        if(!disq[dist[u]]) dist[S] = T;
        dist[u]++;
        disq[dist[u]]++;
        return deltaflow;
}
double value(){
        double maxflow = 0.00;
        while(dist[S] <= T) maxflow += sap(S, inf);
        return -0.50 * (maxflow - d * n);
}
void build(double g){
        g *= 2.00;
        for(int i = 1; i <= n; ++i) add_edge(S, i, d); // s -> v : INF
        for(int i = 1; i <= n; ++i) add_edge(i, T, d + g - deg[i]); // u ->
           t : INF + g - deg[u] 其中 deg[u] 为点 u 的度数（双向边）
        for(int i = 1; i <= n; ++i)
                for(int j = 1; j < i; ++j){
```

```
                        if(a[i] >= a[j]) continue;
                        add_edge(i, j, 1.00); // u -> v : 1.00
                        add_edge(j, i, 1.00);
                }
}
void clear(){
        memset(dist, 0, sizeof dist);
        memset(disq, 0, sizeof disq);
        for(int i = 1; i <= T; ++i) mp[i].clear();
}
double binary(double left, double rght){ // 猜測答案 g [1 / n, m / 1]
        int step = 0;
        while(left + eps < rght && step <= 50){
                ++step;
                double mid = (left + rght) / 2;
                clear();
                build(mid);
                double h = value();
                if(h > eps) left = mid;
                else rght = mid;
        }
        return left;
}
void work(){
        m = 0;
        scanf("%d", &n);
        S = n + 1, T = n + 2;
        for(int i = 1; i <= n; ++i) scanf("%d", &a[i]);
        for(int i = 1; i <= n; ++i) deg[i] = 0;
        for(int i = 1; i <= n; ++i)
                for(int j = 1; j < i; ++j){
                        if(a[i] >= a[j]) continue;
                        ++m;
                        ++deg[i];
                        ++deg[j];
                }
        printf("%.12f\n", binary(0.00, m));
}
int main(){
        int case_number;
        scanf("%d", &case_number);
        for(int cs = 1; cs <= case_number; ++cs){
                printf("Case #%d: ", cs);
                work();
        }
        return 0;
}
```

5.9 强联通分量

```
int cnt, top, scc;
int bel[maxn], dfn[maxn], low[maxn], stck[maxn];
bool inst[maxn];
void tarjan(int u){
        dfn[u] = low[u] = ++cnt;
        stck[++top] = u;
        inst[u] = true;
        for(int l = 0; l < mp[u].size(); ++l){
                int v = mp[u][l];
                if(!dfn[v]){
                        tarjan(v);
                        low[u] = min(low[u], low[v]);
                } else if(inst[v]) low[u] = min(low[u], dfn[v]);
        }
        if(dfn[u] == low[u]){
                ++scc;
                int v;
                do{
                        v = stck[top--];
                        bel[v] = scc;
                        inst[v] = false;
                } while(v != u);
        }
}
```

5.10 边双联通分量

```
int scc, top, cnt;
int dfn[maxn], low[maxn], stck[maxn], bel[maxn];
bool inst[maxn];
void tarjan(int u){

        dfn[u] = low[u] = ++cnt;
        stck[++top] = u;
        inst[u] = true;

        for(int l = 0; l < mp[u].size(); ++l){
                int v = mp[u][l].v;
                if(mp[u][l].flag) continue;
                mp[u][l].flag = mp[v][mp[u][l].r].flag = true;
                if(!dfn[v]){
                        tarjan(v);
                        low[u] = min(low[u], low[v]);
                } else if(inst[v]) low[u] = min(low[u], dfn[v]);
        }
```

```
        if(dfn[u] == low[u]){
                ++scc;
                int v;
                do{
                        v = stck[top--];
                        bel[v] = scc;
                        inst[v] = false;
                        ++sz[scc];
                } while(v != u);
        }
}
```

5.11 点双联通分量加构造森林块

```
//Point Biconnected Component
bool mark[M << 1];
int part;
int ind, dfn[N], low[N], st[M << 1], top, root[N];
void tarjan(int x, int cur)
{
        dfn[x] = low[x] = ++ind;
        for(int i = hd[x]; i; i = nx[i])
        {
    if(mark[i]) continue;
    mark[i] = mark[i ^ 1] = 1;
    st[++top] = i;

    int v = th[i];
    if(dfn[v])
    {
        low[x] = min(low[x], dfn[v]);
        continue;
    }
    tarjan(v, cur);
    low[x] = min(low[x], low[v]);

    if(low[v] >= dfn[x])
    {
        ++part;
        int k;
        do
        {
            k = st[top--];
            root[th[k]] = cur;//联通块里点双联通分量标号最小值
            root[th[k ^ 1]] = cur;
            addtree(part, th[k]);
            addtree(th[k], part);//part为点双联通分量的标号
```

```
            addtree(part, th[k ^ 1]);
            addtree(th[k ^ 1], part);
        }while(th[k ^ 1] != x);
    }
    }
}
bool vis[N << 1];
long long val[N << 1], son[N << 1];
void dfs(int x)
{
        vis[x] = 1;
        val[x] = (x <= n ? w[x] : 1ll);
        son[x] = 0ll;
        for(int i = thd[x]; i; i = tnx[i])
    if(!vis[tth[i]])
    {
        int v = tth[i];
        dfs(v);
        (val[x] *= val[v]) %= MOD;

        if(x <= n) (son[x] += val[v]) %= MOD;
    }
}
```

5.12 K 短路

```
// POJ 2449
/*****************************************
K短路 用dijsktra+A*启发式搜索
当点v第K次出堆的时候，这时候求得的路径是k短路。
A*算法有一个启发式函数f(p)=g(p)+h(p), 即评估函数=当前值+当前位置到终点的最短
    距离
g(p):当前从s到p点所走的路径长度，h(p)就是点p到目的点t的最短距离。
f(p)就是当前路径从s走到p在从p到t的所走距离。
步骤:
1>求出h(p)。将有向边反向，求出目的点t到所有点的最短距离，用dijkstra算法
2>将原点s加入优先队列中
3>优先队列取出f(p)最小的一个点p
如果p==t,并且出来的次数恰好是k次，那么算法结束
否则，如果p出来的次数多余k次，就不用再进入队列中
否则遍历p相邻的边，加入优先队列中
注意:如果s==t，那么求得k短路应该变成k++;
*****************************************/
#define MAXN 1005
#define MAXM 200100
struct Node{
       int v,c,nxt;
```

```
}Edge[MAXM];
int head[MAXN], tail[MAXN], h[MAXN];
struct Statement{
       int v,d,h;
       bool operator <( Statement a )const
       {    return a.d+a.h<d+h;   }
};
void addEdge( int u,int v,int c,int e ){
     Edge[e<<1].v=v; Edge[e<<1].c=c; Edge[e<<1].nxt=head[u]; head[u]=e<<1;
     Edge[e<<1|1].v=u; Edge[e<<1|1].c=c; Edge[e<<1|1].nxt=tail[v]; tail[v]=e
        <<1|1;
}

void Dijstra( int n,int s,int t ){
     bool vis[MAXN];
     memset( vis,0,sizeof(vis) );
     memset( h,0x7F,sizeof(h) );
     h[t]=0;
     for( int i=1;i<=n;i++ ){
          int min=0x7FFF;
          int k=-1;
          for( int j=1;j<=n;j++ ){
               if( vis[j]==false && min>h[j] )
                   min=h[j],k=j;
          }
          if( k==-1 )break;
          vis[k]=true;
          for( int temp=tail[k];temp!=-1;temp=Edge[temp].nxt ){
               int v=Edge[temp].v;
               if( h[v]>h[k]+Edge[temp].c )
                   h[v]=h[k]+Edge[temp].c;
          }
     }
}
int Astar_Kth( int n,int s,int t,int K ){
    Statement cur,nxt;
    //priority_queue<Q>q;
    priority_queue<Statement>FstQ;
    int cnt[MAXN];
    memset( cnt,0,sizeof(cnt) );
    cur.v=s; cur.d=0; cur.h=h[s];
    FstQ.push(cur);
    while( !FstQ.empty() ){
           cur=FstQ.top();
           FstQ.pop();
           cnt[cur.v]++;
           if( cnt[cur.v]>K ) continue;
           if( cnt[t]==K )return cur.d;
```

```
            for( int temp=head[cur.v];temp!=-1;temp=Edge[temp].nxt ){
                int v=Edge[temp].v;
                nxt.d=cur.d+Edge[temp].c;
                nxt.v=v;
                nxt.h=h[v];
                FstQ.push(nxt);
            }
    }
    return -1;
}
int main()
{
    int n,m;
    while( scanf( "%d %d",&n,&m )!=EOF ){
            int u,v,c;
            memset( head,0xFF,sizeof(head) );
            memset( tail,0xFF,sizeof(tail) );
            for( int i=0;i<m;i++ ){
                scanf( "%d %d %d",&u,&v,&c );
                addEdge( u,v,c,i );
            }
            int s,t,k;
            scanf( "%d %d %d",&s,&t,&k );
            if( s==t ) k++;
            Dijstra( n,s,t );
            printf( "%d\n",Astar_Kth( n,s,t,k ) );
    }
    return 0;
}
```

6 其他

6.1 Dancing Links(精确覆盖及重复覆盖)

```
// HUST 1017
// 给定一个 n 行 m 列的 0/1 矩阵，选择某些行使得每一列都恰有一个 1
const int MAXN = 1e3 + 5;
const int MAXM = MAXN * MAXN;
const int INF = 1e9;
int ans;
int chosen[MAXM];
struct DancingLinks{
        int row, col, tot;
        int up[MAXM], dn[MAXM], lf[MAXM], rg[MAXM];
        int hd[MAXM], sz[MAXM];
        int posr[MAXM], posc[MAXM];
```

```
void init(int _n, int _m){
        row = _n, col = _m;
        for(int i = 0; i <= col; ++i){
                sz[i] = 0;
                up[i] = dn[i] = i;
                lf[i] = i - 1;
                rg[i] = i + 1;
        }
        rg[col] = 0;
        lf[0] = col;
        tot = col;
        for(int i = 1; i <= row; ++i) hd[i] = -1;
}
void lnk(int r, int c){
        ++tot;
        ++sz[c];
        dn[tot] = dn[c];
        up[tot] = c;
        up[dn[c]] = tot;
        dn[c] = tot;
        posr[tot] = r;
        posc[tot] = c;
        if(hd[r] < 0) hd[r] = lf[tot] = rg[tot] = tot;
        else{
                lf[tot] = hd[r];
                rg[tot] = rg[hd[r]];
                lf[rg[hd[r]]] = tot;
                rg[hd[r]] = tot;
        }
}
void remove(int c){ // 删除列时删除能覆盖其的行
        rg[lf[c]] = rg[c];
        lf[rg[c]] = lf[c];
        for(int i = dn[c]; i != c; i = dn[i])
                for(int j = rg[i]; j != i; j = rg[j]){
                        dn[up[j]] = dn[j];
                        up[dn[j]] = up[j];
                        --sz[posc[j]];
                }
}
void resume(int c){
        rg[lf[c]] = c;
        lf[rg[c]] = c;
        for(int i = dn[c]; i != c; i = dn[i])
                for(int j = rg[i]; j != i; j = rg[j]){
                        up[dn[j]] = j;
                        dn[up[j]] = j;
                        ++sz[posc[j]];
```

```
                }
        }
        bool dance(int dpth){
                if(rg[0] == 0){
                        printf("%d", dpth);
                        for(int i = 0; i < dpth; ++i) printf(" %d", chosen[i
                            ]);
                        puts("");
                        return true;
                }
                int c = rg[0];
                for(int i = rg[0]; i; i = rg[i]) if(sz[i] < sz[c]) c = i;
                remove(c); // 当前消去第c列
                for(int i = dn[c]; i != c; i = dn[i]){ // 第c列是由第i行覆盖
                    的
                        chosen[dpth] = posr[i];
                        for(int j = rg[i]; j != i; j = rg[j]) remove(posc[j
                            ]); // 删除第i行能覆盖的其余列 因为它们只能被覆盖
                            一次
                        if(dance(dpth + 1)) return true;
                        for(int j = lf[i]; j != i; j = lf[j]) resume(posc[j
                            ]);
                }
                resume(c);
                return false;
        }
};
DancingLinks dlx;
int n, m;
void work(){
        dlx.init(n, m);
        for(int i = 1; i <= n; ++i){
                int k, j;
                scanf("%d", &k);
                while(k--){
                        scanf("%d", &j);
                        dlx.lnk(i, j);
                }
        }
        if(!dlx.dance(0)) puts("NO");
}
// 重复覆盖
// 给定一个 n 行 m 列的 0/1 矩阵, 选择某些行使得每一列至少有一个 1
struct DancingLinks{
        int row, col, tot;
        int up[MAXM], dn[MAXM], lf[MAXM], rg[MAXM];
        int head[MAXM], sz[MAXM];
        void init(int _n, int _m){
```

```
        row = _n, col = _m;
        for(int i = 0; i <= col; ++i){
                sz[i] = 0;
                up[i] = dn[i] = i;
                lf[i] = i - 1;
                rg[i] = i + 1;
        }
        rg[col] = 0;
        lf[0] = col;
        tot = col;
        for(int i = 1; i <= row; ++i) head[i] = -1;
}
void lnk(int r, int c){
        ++tot;
        ++sz[c];
        dn[tot] = dn[c];
        up[dn[c]] = tot;
        up[tot] = c;
        dn[c] = tot;
        if(head[r] < 0) head[r] = lf[tot] = rg[tot] = tot;
        else{
                rg[tot] = rg[head[r]];
                lf[rg[head[r]]] = tot;
                lf[tot] = head[r];
                rg[head[r]] = tot;
        }
}
void remove(int c){ // 删除列时不删除行 因为列可被重复覆盖
        for(int i = dn[c]; i != c; i = dn[i]){
                rg[lf[i]] = rg[i];
                lf[rg[i]] = lf[i];
        }
}
void resume(int c){
        for(int i = up[c]; i != c; i = up[i]){
                rg[lf[i]] = i;
                lf[rg[i]] = i;
        }
}
void dance(int d){
        if(ans <= d) return;
        if(rg[0] == 0){
                ans = min(ans, d);
                return;
        }
        int c = rg[0];
        for(int i = rg[0]; i != 0; i = rg[i]) if(sz[i] < sz[c]) c =
           i;
```

```
                for(int i = dn[c]; i != c; i = dn[i]){ // 枚举c列是被哪行覆
                    盖
                        remove(i);
                        for(int j = rg[i]; j != i; j = rg[j]) remove(j); //
                            删除可被i行覆盖的列 因为不需要再考虑它们的覆盖问
                            题
                        dance(d + 1);
                        for(int j = lf[i]; j != i; j = lf[j]) resume(j);
                        resume(i);
                }
        }
};
DancingLinks dlx;
```

6.2 序列莫队

```
const int maxn = 50005;
const int maxb = 233;
int n, m, cnt[maxn], a[maxn];
long long answ[maxn], ans;
int bk, sz, bel[maxn];
int lf[maxn], rh[maxn], rnk[maxn];
bool cmp(int i, int j){
        if(bel[lf[i]] != bel[lf[j]]) return bel[lf[i]] < bel[lf[j]];
        else return bel[rh[i]] < bel[rh[j]];
}
void widden(int i){ans += cnt[a[i]]++;}
void shorten(int i){ans -= --cnt[a[i]];}
long long gcd(long long a, long long b){
        if(b == 0) return a;
        else return gcd(b, a % b);
}
int main(){
        scanf("%d%d", &n, &m);
        bk = sqrt(n); sz = n / bk;
        while(bk * sz < n) ++bk;
        for(int b = 1, i = 1; b <= bk; ++b)
                for(; i <= b * sz && i <= n; ++i) bel[i] = b;
        for(int i = 1; i <= n; ++i) scanf("%d", &a[i]);
        for(int i = 1; i <= m; ++i) scanf("%d%d", &lf[i], &rh[i]);
        for(int i = 1; i <= m; ++i) rnk[i] = i;
        sort(rnk + 1, rnk + 1 + m, cmp);
        lf[0] = rh[0] = 1; widden(1);
        for(int i = 1; i <= m; ++i){
                int k = rnk[i], kk = rnk[i-1];
                for(int j = lf[k]; j < lf[kk]; ++j) widden(j);
                for(int j = rh[k]; j > rh[kk]; --j) widden(j);
```

```
                for(int j = lf[kk]; j < lf[k]; ++j) shorten(j);
                for(int j = rh[kk]; j > rh[k]; --j) shorten(j);
                answ[k] = ans;
        }
        for(int i = 1; i <= m; ++i){
                if(answ[i] == 0){
                        puts("0/1");
                        continue;
                }
                int lnth = rh[i] - lf[i] + 1;
                long long t = 1LL * lnth * (lnth - 1) / 2;
                long long g = gcd(answ[i], t);
                printf("%lld/%lld\n", answ[i] / g, t / g);
        }
        return 0;
}
```

6.3 模拟退火

```
int n;
double A,B;
struct Point{
    double x,y;
    Point(){}
    Point(double x,double y):x(x),y(y){}
    void modify(){
        x = max(x,0.0);
        x = min(x,A);
        y = max(y,0.0);
        y = min(y,B);
    }
}p[1000000];
double sqr(double x){
    return x * x;
}
double Sqrt(double x){
    if(x < eps) return 0;
    return sqrt(x);
}
Point operator + (const Point &a,const Point &b){
    return Point(a.x + b.x, a.y + b.y);
}
Point operator - (const Point &a,const Point &b){
    return Point(a.x - b.x, a.y - b.y);
}
Point operator * (const Point &a,const double &k){
    return Point(a.x * k, a.y * k);
```

```
}
Point operator / (const Point &a,const double &k){
    return Point(a.x / k, a.y / k);
}
double det (const Point &a,const Point &b){
    return a.x * b.y - a.y * b.x;
}
double dist(const Point &a, const Point &b){
    return Sqrt(sqr(a.x - b.x)+sqr(a.y - b.y));
}
double work(const Point &x){
    double ans = 1e9;
    for(int i=1;i<=n;i++)
        ans = min(ans,dist(x,p[i]));
    return ans;
}
int main(){
    srand(time(NULL));
    int numcase;
    cin>>numcase;
    while (numcase--){
        scanf("%lf%lf%d",&A,&B,&n);
        for(int i=1;i<=n;i++){
            scanf("%lf%lf",&p[i].x,&p[i].y);
        }
        double total_ans = 0;
        Point total_aaa;
        for(int ii = 1;ii<=total/n;ii++){
            double ans = 0;
            Point aaa;
            Point p;
            p.x = (rand() % 10000) * A / 10000;
            p.y = (rand() % 10000) * B / 10000;
            double step = 2 * max(A,B);
            for(double T = 1e6;T > 1e-2;T = T * 0.98){
                double thi = (rand() % 10000) * pi2 / 10000;
                Point now = p + Point(cos(thi), sin(thi)) * step * (rand() %
                    10000)/10000;
                now.modify();
                double now_ans = work(now);
                double delta = now_ans -ans;
                if(delta > 0) {
                    p = now;
                    ans = now_ans;
                    aaa = now;
                }
                else{
                    if((rand() % 10000) / 10000.0 > exp(delta / T)) p = now;
```

```
                }
                step = max(step * 0.9,1e-3);
            }
            if(ans > total_ans) total_ans = ans, total_aaa = aaa;
        }
        printf("The safest point is (%.1f, %.1f).\n",total_aaa.x,total_aaa.y
           );
    }
}
```

6.4 Java

```
import java.io.*;
import java.util.*;
import java.math.*;
public class Main{
        public static BigInteger n,m;
        public static Map<BigInteger,Integer> M = new HashMap();
//      public static BigInteger dfs(BigInteger x){
//              if(M.get(x)!=null)return M.get(x);
//              if(x.mod(BigInteger.valueOf(2))==1){
//              }else{
//              }
//              M.put();
//      }
    static int NNN = 1000000;
    static BigInteger N;
    static BigInteger M;
    static BigInteger One = new BigInteger("1");
    static BigInteger Two = new BigInteger("2");
    static BigInteger Zero = new BigInteger("0");
    static BigInteger[] queue = new BigInteger[NNN];
    static BigInteger[] num_step = new BigInteger[NNN];
        public static void main(String []arg){
                Scanner cin = new Scanner(System.in);
                while(true){
        int p = cin.nextInt();
        n = cin.nextBigInteger();
        m = cin.nextBigInteger();
        n.multiply(m);
        M.clear();
        if(n.compareTo(BigInteger.ZERO)==0)break;
        if(n.compareTo(m)<=0){
                                System.out.println(m.subtract(n));
                                continue;
                        }
        BigInteger[] QB = new BigInteger[5000*20];
```

```
        Integer[] QD = new Integer[5000*20];
        int head=0,tail=0;
        QB[tail]=n;
        QD[tail]=0;
        tail++;
        BigInteger ans = n.subtract(m).abs();
        while(head<tail){
            BigInteger now = QB[head],nxt;
            int dep = QD[head];
            //System.out.println("now is "+now+" dep is "+dep);
            if(ans.compareTo(BigInteger.valueOf(dep).add(m.subtract(now).abs
               ()))>0)
                ans=BigInteger.valueOf(dep).add(m.subtract(now).abs());
            head++;
            if(now.mod(BigInteger.valueOf(2)).compareTo(BigInteger.ONE)!=0){
                nxt=now.divide(BigInteger.valueOf(2));
                if(M.get(nxt)==null){
                    QB[tail]=nxt;
                    QD[tail]=dep+1;
                    tail++;
                    M.put(nxt,1);
                }
            }else{
                nxt=now.subtract(BigInteger.ONE);
                if(M.get(nxt)==null&&nxt.compareTo(BigInteger.ZERO)!=0){
                    QB[tail]=nxt;
                    QD[tail]=dep+1;
                    tail++;
                    M.put(nxt,1);
                }
                nxt=now.add(BigInteger.ONE);
                if(M.get(nxt)==null){
                    QB[tail]=nxt;
                    QD[tail]=dep+1;
                    tail++;
                    M.put(nxt,1);
                }
            }
        }
        System.out.println(ans);
    }
}
}
```

还有这样的hashset用法：

```
static Collection c = new HashSet();
if(c.contains(p) == false)
```

6.5 博弈论相关

1. Anti-SG: 规则与 Nim 基本相同，取最后一个的输。先手必胜当且仅当：(1) 所有堆的石子数都为 1 且游戏的 SG 值为 0；(2) 有些堆的石子数大于 1 且游戏的 SG 值不为 0。

2. SJ 定理: 对于任意一个 Anti-SG 游戏，如果我们规定当局面中，所有的单一游戏的 SG 值为 0 时，游戏结束，则先手必胜当且仅当：(1) 游戏的 SG 函数不为 0 且游戏中某个单一游戏的 SG 函数大于 1；(2) 游戏的 SG 函数为 0 且游戏中没有单一游戏的 SG 函数大于 1。

3. Multi-SG 游戏: 可以将一堆石子分成多堆.

4. Every-SG 游戏: 每一个可以移动的棋子都要移动. 对于我们可以赢的单一游戏，我们一定要拿到这一场游戏的胜利. 只需要考虑如何让我们必胜的游戏尽可能长的玩下去，对手相反。于是就来一个 DP，step[v] = 0；(v 为终止状态) step[v] = maxstep[u] + 1；(sg[v]>0,sg[u]=0) step[v] = minstep[u] + 1；(sg[v]==0)

5. 翻硬币游戏: N 枚硬币排成一排，有的正面朝上，有的反面朝上。游戏者根据某些约束翻硬币 (如：每次只能翻一或两枚，或者每次只能翻连续的几枚), 但他所翻动的硬币中，最右边的必须是从正面翻到反面。谁不能翻谁输。结论：局面的 SG 值为局面中每个正面朝上的棋子单一存在时的 SG 值的异或和。可用数学归纳法证明。

6. 无向树删边游戏: 规则如下: 给出一个有 N 个点的树，有一个点作为树的根节点。游戏者轮流从树中删去边，删去一条边后，不与根节点相连的部分将被移走。谁无路可走谁输。结论: 叶子节点的 SG 值为 0；中间节点的 SG 值为它的所有子节点的 SG 值加 1 后的异或和。是用数学归纳法证明。

7. Christmas Game(PKU3710): 题目大意：有 N 个局部联通的图。Harry 和 Sally 轮流从图中删边，删去一条边后，不与根节点相连的部分将被移走。Sally 为先手。图是通过从基础树中加一些边得到的。所有形成的环保证不共用边，且只与基础树有一个公共点。谁无路可走谁输。环的处理成为了解题的关键。性质：（1）对于长度为奇数的环，去掉其中任意一个边之后，剩下的两个链长度同奇偶，抑或之后的 SG 值不可能为奇数，所以它的 SG 值为 1；
(2) 对于长度为偶数的环，去掉其中任意一个边之后，剩下的两个链长度异奇偶，抑或之后的 SG 值不可能为 0，所以它的 SG 值为 0；所以我们可以去掉所有的偶环，将所有的奇环变为长短为 1 的链。这样的话，我们已经将这道题改造成了上一节的模型。

8. 无向图的删边游戏: 我们将 Christmas Game 这道题进行一步拓展——去掉对环的限制条件，这个模型应该怎样处理? 无向图的删边游戏：一个无向联通图，有一个点作为图的根。游戏者轮流从图中删去边，删去一条边后，不与根节点相连的部分将被移走。谁无路可走谁输。结论：对无向图做如下改动：将图中的任意一个偶环缩成一个新点，任意一个奇环缩成一个新点加一个新边；所有连到原先环上的边全部改为与新点相连。这样的改动不会影响图的 SG 值。

9. Staircase nim: 楼梯从地面由下向上编号为 0 到 n。游戏者在每次操作时可以将楼梯 j(1<=j<=n) 上的任意多但至少一个硬币移动到楼梯 j-1 上。将最后一枚硬币移至地上的人获胜。结论：设该游戏 Sg 函数为奇数格棋子数的 Xor 和 S。如果 S=0，则先手必败，否则必胜。

7 Tips

判斜率 (x/gcd, y/gcd) 直接丢 map 里 unique
无方案和答案% MOD 为 0 是有区别的
打标记使用时间戳
a = 10 * a + 1 可以用矩乘加速
数组要开 1e5 [+ 5]!

pow(a, b) 会调用 c++ 自带函数
强联通、双联通要考虑一个孤立点
MOD 的时候：(a - b + MOD) % MOD (a + b * c % MOD) % MOD
stack 里有时存的边，这种时候大小不要开错了
选择性段错误: 没 return 没赋初值
凸包排序后数组顺序会改变，不可以在这之后求重心
位运算优先级小于 ==
(int)x != round(x)
hash 字符串：t = t * 27(!) + s[i] - 'A' + 1
有些 dfs 里用到的数组开全局会跪
$n = 1e4$ 时明摆着要 n^2 bitset 压位
博弈题做法：1. 由最终态 BFS（类似构了一颗树）2. 打表找 sg 函数规律
没辙时想 dp 和网络流
启发式合并 $nlogn$
$\frac{n}{1} + \frac{n}{2} + ... = nlogn$
$fact[0] = 1$