Contents

1 基础 **1**

1.1 fastpower.cpp 1

1.2 prime sieve 素数筛.cpp 2

1.3 几何 3

1.3.1 Circle 圆形.cpp 3

1.3.2 Polygon 多边形.cpp 4

1.3.3 Points-Vector 点与向量.cpp 5

1.3.4 Circumcenter 外心 三点定圆.cpp 8

1.3.5 MinCircleCover 最小圆覆盖.cpp 9

1.3.6 ConvexHull 凸包.cpp 13

1.3.7 Line-Segment 直线与线段.cpp 19

1.3.8 Hull 下凸包求函数最值.cpp 21

1.3.9 ClosestPoints 最近点对.cpp 24

2 数据结构 **29**

2.1 ZTC's Splay.txt 29

2.2 zhuxishu_SegKth.cpp 33

1 基础

1.1 fastpower.cpp

```cpp
//
// Created by acm-33 on 2019/9/19.
//


template<typename var= long long>
var fpow(var a, var b, var m) {
    var ret = 1;
    while (b) {
        if (b & 1)ret = ret * a % m;
        a = a * a % m;
        b >>= 1;
    }
    return ret;
}

long long fpow(long long a, long long b, long long m) {
    long long ret = 1;
    while (b) {
        if (b & 1)ret = ret * a % m;
        a = a * a % m;
        b >>= 1;
    }
    return ret;
}
```

1.2 prime sieve 素数筛.cpp

```
//单纯求素数，本地 60ms+
const int MAXN = -1;//10000005
int prime[MAXN], pnum;
bool is_composite[MAXN];

void sieve(const int &n) {
    // 1 is exception
    for (int i = 2; i < n; ++i) {
        if (!is_composite[i]) prime[++pnum] = i;
        for (int j = 1; j <= pnum && i * prime[j] < n; ++j) {
            is_composite[i * prime[j]] = true;
            if (i % prime[j] == 0) break;
        }
    }
}

//求素数和最小素因子，本地 90ms+
const int MAXN = -1;//10000005
int prime[MAXN], pnum;
int min_composite[MAXN];

void sieve(const int &n) {
    // 1 is exception
    for (int i = 2; i < n; ++i) {
        if (!min_composite[i]) {

            prime[++pnum] = i;
            min_composite[i] = i;
        }
        for (int j = 1; j <= pnum
            && prime[j] <= min_composite[i]
            && i * prime[j] < n; ++j) {
            min_composite[i * prime[j]] = prime[j];
//              if (i % prime[j] == 0) break;
        }
    }
}
```

1.3 几何

1.3.1 Circle 圆形.cpp

```
/**
 *  @Source: team
 *  @Author: Artiprocher(Zhongjie Duan) -> tieway59
 *  @Description:
 *      圆形计算相关。
 *  @Example:
 *
 *  @Verification:
 *
 */

struct Circle {
    Point c;
    double r;

    Point point(double a)//基于圆心角求圆上一点坐标
    {
        return Point(c.x + cos(a) * r, c.y + sin(a) * r);
    }
};

double Angle(Vector v1) {
    if (v1.y >= 0)return Angle(v1, Vector(1.0, 0.0));
    else return 2 * pi - Angle(v1, Vector(1.0, 0.0));
}

int GetCC(Circle C1, Circle C2)//求两圆交点
```

```
{
    double d = Length(C1.c - C2.c);
    if (dcmp(d) == 0) {
        if (dcmp(C1.r - C2.r) == 0)return -1;//重合
        else return 0;
    }
    if (dcmp(C1.r + C2.r - d) < 0)return 0;
    if (dcmp(fabs(C1.r - C2.r) - d) > 0)return 0;
    double a = Angle(C2.c - C1.c);
    double da = acos((C1.r * C1.r + d * d - C2.r * C2.r) / (2 * C1.r *
     ↪  d));
    Point p1 = C1.point(a - da), p2 = C1.point(a + da);
    if (p1 == p2)return 1;
    else return 2;
}
```

1.3.2 Polygon 多边形.cpp

```
/**
 *  @Source: team
 *  @Author: Artiprocher(Zhongjie Duan) -> tieway59
 *  @Description:
 *      多边形相关的计算。
 *  @Example:
 *
 *  @Verification:
 *
 */

Point P[1005]; // P[] 为多边形的所有顶点，下标为 0~n-1
int n;         // n 为多边形边数
// 求多边形面积（叉积和计算法）
double PolygonArea() {
    double sum = 0;
    Point O = Point(0, 0);
    for (int i = 0; i < n; i++)
        sum += Cross(P[i] - O, P[(i + 1) % n] - O);
    if (sum < 0)sum = -sum;
    return sum / 2;
}

// STL: 求多边形面积（叉积和计算法）
double PolygonArea(const vector <Point> &P) {
    int n = P.size();
    // assert(n > 2);
    double sum = 0;
    Point O = Point(0, 0);
    for (int i = 0; i < n; i++)
        sum += Cross(P[i] - O, P[(i + 1) % n] - O);
    if (sum < 0) sum = -sum;
    return sum / 2;
}

/* 模板说明: P[] 为多边形的所有顶点, 下标为 0~n-1, n 为多边形边数 */
//判断点是否在凸多边形内（角度和判别法）
Point P[1005];
int n;

bool InsidePolygon(Point A) {
    double alpha = 0;
    for (int i = 0; i < n; i++)
        alpha += fabs(Angle(P[i] - A, P[(i + 1) % n] - A));
    return dcmp(alpha - 2 * pi) == 0;
}
```

1.3.3 Points-Vector 点与向量.cpp

```
/**
 *  @Source: team
 *  @Author: Artiprocher(Zhongjie Duan) -> tieway59
 *  @Description:
 *        点与向量相关的多种计算。
 *  @Example:
 *
 *  @Verification:
 *
 */


//#include <bits/stdc++.h>
//using namespace std;
const double EPS = 1e-6;//eps 用于控制精度
const double Pi = acos(-1.0);//pi

//精度三态函数 (>0,<0,=0)
inline int dcmp(double x) {
    if (fabs(x) < EPS)return 0;
    else if (x > 0)return 1;
    return -1;
}

//点或向量 (iostream 选择性抄写)
struct Point {
    double x, y;

    Point() {}

    Point(double x, double y) : x(x), y(y) {}

    friend ostream &operator<<(ostream &ut, Point &r) { return ut <<
     ↪  r.x << " " << r.y; }

    friend istream &operator>>(istream &in, Point &r) { return in >>
     ↪  r.x >> r.y; }
};

typedef Point Vector;

inline Vector operator+(Vector a, Vector b) {
    return Vector(a.x + b.x, a.y + b.y);
}

inline Vector operator-(Vector a, Vector b) {
    return Vector(a.x - b.x, a.y - b.y);
```

```
}

//向量数乘
inline Vector operator*(Vector a, double p) {
    return Vector(a.x * p, a.y * p);
}

//向量数除
inline Vector operator/(Vector a, double p) {
    return Vector(a.x / p, a.y / p);
}

inline bool operator==(const Point &a, const Point &b) {
    return dcmp(a.x - b.x) == 0 && dcmp(a.y - b.y) == 0;
}

//内积
inline double Dot(Vector a, Vector b) {
    return a.x * b.x + a.y * b.y;
}

//外积
inline double Cross(Vector a, Vector b) {
    return a.x * b.y - a.y * b.x;
}

//模
inline double Length(Vector a) {
    return sqrt(Dot(a, a));
}

//夹角，弧度制
inline double Angle(Vector a, Vector b) {
    return acos(Dot(a, b) / Length(a) / Length(b));
}

//逆时针旋转
inline Vector Rotate(Vector a, double rad) {
    return Vector(a.x * cos(rad) - a.y * sin(rad), a.x * sin(rad) +
     ↪  a.y * cos(rad));
}

//两点间距离
inline double Distance(Point a, Point b) {
    return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
}

//三角形面积
inline double Area(Point a, Point b, Point c) {
    return fabs(Cross(b - a, c - a) / 2);
}
```

1.3.4 Circumcenter 外心 三点定圆.cpp

```
/**
 *  @Source: blog.csdn.net/liyuanbhu/article/details/52891868
 *  @Author: tieway59
 *  @Description:
 *      注意排除三点共线。
 *      if (dcmp(Cross(pi, pj)) == 0) continue;
 *
 *  @Example:
 *      circumcenter(Point(0, 1), Point(1, 1), Point(1, 0));
 *      // 0.5 0.5
 *
 *  @Verification:
 *      https://ac.nowcoder.com/acm/contest/5667/B
 *      (solution)
 ↪  ac.nowcoder.com/acm/contest/view-submission?submissionId=44337916
 *
 */

template<typename tp>
inline tp pow2(const tp &x) {
    return x * x;
}
```

```
inline Point circumcenter(Point p1, Point p2, Point p3) {
    double a = p1.x - p2.x;
    double b = p1.y - p2.y;
    double c = p1.x - p3.x;
    double d = p1.y - p3.y;
    double e = (pow2(p1.x) - pow2(p2.x) +
                pow2(p1.y) - pow2(p2.y)) / 2;
    double f = (pow2(p1.x) - pow2(p3.x) +
                pow2(p1.y) - pow2(p3.y)) / 2;
    return Point((d * e - b * f) /
                 (a * d - b * c),
                 (a * f - c * e) /
                 (a * d - b * c));
}
```

1.3.5 MinCircleCover 最小圆覆盖.cpp

```
/**
 * @Source: https://www.luogu.com.cn/problem/solution/P1742
 * @Author: snowbody -> tieway59
 * @Description:
 *     时间复杂度 O(N)
 *     为了减少中途过度开根，距离都是先按照平方计算的。
 *
 * @Example:
 *     vector<Point> p(n);
 *     for (auto &pi : p) cin >> pi;
 *     Circle circle;
 *     MinCircleCover(p, circle);
 *
 *     6
 *     8.0 9.0
 *     4.0 7.5
 *     1.0 2.0
 *     5.1 8.7
 *     9.0 2.0
 *     4.5 1.0
 *      // r = 5.0000000000 (5.0000000000,5.0000000000)
 *
 *  @Verification:
 *      https://www.luogu.com.cn/problem/P1742
 */



//点或向量 (iostream 选择性抄写)
struct Point {
    double x, y;

    Point() {}

    Point(double x, double y) : x(x), y(y) {}

    friend ostream &operator<<(ostream &ut, Point &r) { return ut <<
     ↪  r.x << " " << r.y; }

    friend istream &operator>>(istream &in, Point &r) { return in >>
     ↪  r.x >> r.y; }
};

typedef Point Vector;

inline Vector operator+(Vector a, Vector b) {
    return Vector(a.x + b.x, a.y + b.y);
}

inline Vector operator-(Vector a, Vector b) {
    return Vector(a.x - b.x, a.y - b.y);
}

//向量数乘
inline Vector operator*(Vector a, double p) {
    return Vector(a.x * p, a.y * p);
}
```

```
//向量数除
inline Vector operator/(Vector a, double p) {
    return Vector(a.x / p, a.y / p);
}

//两点间距离
inline double Distance(Point a, Point b) {
    return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
}

inline double Distance2(Point a, Point b) {
    return ((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
}

struct Circle {
    Point c;
    double r;

    Point point(double a)//基于圆心角求圆上一点坐标
    {
        return Point(c.x + cos(a) * r, c.y + sin(a) * r);
    }
};

template<typename tp>
inline tp pow2(const tp &x) {
    return x * x;
}

inline Point circumcenter(Point p1, Point p2, Point p3) {
    double a = p1.x - p2.x;
    double b = p1.y - p2.y;
    double c = p1.x - p3.x;
    double d = p1.y - p3.y;
    double e = (pow2(p1.x) - pow2(p2.x) +
                pow2(p1.y) - pow2(p2.y)) / 2;
    double f = (pow2(p1.x) - pow2(p3.x) +
                pow2(p1.y) - pow2(p3.y)) / 2;
    return Point((d * e - b * f) /
                 (a * d - b * c),
                 (a * f - c * e) /
                 (a * d - b * c));
}


void MinCircleCover(vector <Point> &p, Circle &res) {
    int n = p.size();
    random_shuffle(p.begin(), p.end());
    // avoid *sqrt* too much killing your precision.
    for (int i = 0; i < n; i++) {
        if (Distance2(p[i], res.c) <= res.r) continue;
        res.c = p[i];
        res.r = 0;
        for (int j = 0; j < i; j++) {
            if (Distance2(p[j], res.c) <= res.r)continue;
            res.c = (p[i] + p[j]) / 2;
            res.r = Distance2(p[j], res.c);
            for (int k = 0; k < j; k++) {
                if (Distance2(p[k], res.c) <= res.r)continue;
                res.c = circumcenter(p[i], p[j], p[k]);
                res.r = Distance2(p[k], res.c);
            }
        }
    }
    res.r = sqrt(res.r);
}

void solve(int kaseId = -1) {
    int n;
    cin >> n;
    vector <Point> p(n);
    for (auto &pi : p) cin >> pi;
    Circle circle;
    MinCircleCover(p, circle);
    cout << fixed << setprecision(10) << circle.r << endl;
    cout << circle.c.x << " " << circle.c.y << endl;
```

```
}
```

1.3.6 ConvexHull 凸包.cpp

```
/**
 *  @Source: Graham_s_scan
 *  @Author: Artiprocher(Zhongjie Duan) -> tieway59
 *  @Description:
 *      n       点数
 *      P[]     点数组 index0
 *      top     栈顶, 凸包顶点数
 *      H[]     凸包的顶点 index0
 *      小心重复的凸包顶点, 也会加入凸包。
 *      H[] 逆时针顺序
 *      数组形式，理论上常数会小?
 *
 *  @Example:
 *      4
 *      4 8
 *      4 12
 *      5 9.3 (exclude)
 *      7 8
 *
 *  @Verification:
 *      https://www.luogu.com.cn/record/35363811
 *
 */
int n, top;
const int PSIZE = 100005;
Point P[PSIZE], H[PSIZE];

bool cmp(Point A, Point B) {
    double ans = Cross(A - P[0], B - P[0]);
    if (dcmp(ans) == 0)
        return dcmp(Distance(P[0], A) - Distance(P[0], B)) < 0;
    else
        return ans > 0;
}
```

```
//Graham 凸包扫描算法
void Graham() {
    for (int i = 1; i < n; i++)//寻找起点
        if (P[i].y < P[0].y || (dcmp(P[i].y - P[0].y) == 0 && P[i].x <
         ↪  P[0].x))
            swap(P[i], P[0]);
    sort(P + 1, P + n, cmp);//极角排序，中心为起点
    H[0] = P[0];
    H[1] = P[1];
    top = 2;
    for (int i = 2; i < n; i++) {
        while (top >= 2 && Cross(H[top - 1] - H[top - 2], P[i] - H[top
         ↪  - 2]) < 0)
            top--;
        H[top++] = P[i];
    }
}
```

```
/**
 * @Source: Graham_s_scan
 * @Author: Artiprocher(Zhongjie Duan) -> tieway59
 * @Description:
 *      小心重复的凸包顶点， 也会加入凸包。
 *      H[] 逆时针顺序
 *      数组形式，理论上常数会小？
 *
 * @Example:
 *      4
 *      4 8
 *      4 12
 *      5 9.3 (exclude)
 *      7 8
 *
 * @Verification:
 *      https://www.luogu.com.cn/record/35363811
 *
 */

// HEAD begin
const double EPS = 1e-6;

struct Point//点或向量
{
    double x, y;

    Point() {}

    Point(double x, double y) : x(x), y(y) {}

    friend ostream &operator<<(ostream &ut, Point &r) { return ut <<
     ↪  r.x << " " << r.y; }

    friend istream &operator>>(istream &in, Point &r) { return in >>
     ↪  r.x >> r.y; }
};

typedef Point Vector;

inline double Distance(Point a, Point b) {
    return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
}

inline Vector operator+(Vector a, Vector b) {
    return Vector(a.x + b.x, a.y + b.y);
}

inline Vector operator-(Vector a, Vector b) {
    return Vector(a.x - b.x, a.y - b.y);
}

//外积
inline double Cross(Vector a, Vector b) {
    return a.x * b.y - a.y * b.x;
}
```

```
//精度三态函数 (>0,<0,=0)
inline int dcmp(double x) {
    if (fabs(x) < EPS)return 0;
    else if (x > 0)return 1;
    return -1;
}

// HEAD end
void ConvexHull(vector <Point> &P, vector <Point> &H) {
    int n = int(P.size());
    for (int i = 1; i < n; i++)//寻找起点
        if (P[i].y < P[0].y || (dcmp(P[i].y - P[0].y) == 0 && P[i].x <
         ↪  P[0].x))
            swap(P[i], P[0]);

    //极角排序，中心为起点
    sort(P.begin() + 1, P.end(), [&P](Point A, Point B) {
        double ans = Cross(A - P[0], B - P[0]);
        if (dcmp(ans) == 0)
            return dcmp(Distance(P[0], A) - Distance(P[0], B)) < 0;
        else
            return ans > 0;
    });

    H.assign(n + n, {});
    H[0] = P[0];
    H[1] = P[1];
    int top = 2;
    for (int i = 2; i < n; i++) {
        while (top >= 2 && Cross(H[top - 1] - H[top - 2], P[i] - H[top
         ↪  - 2]) < 0)
            top--;
        H[top++] = P[i];
    }
    H.resize(top);
}
```

```
/**
 *  @Source: Andrew_s_monotone_chain
 *  @Author: Artiprocher(Zhongjie Duan) -> tieway59
 *  @Description:
 *      Andrew_s_monotone_chain
 *      从左下角开始逆时针排列，去除凸包边上的点。
 *      求出来的凸包是逆时针的。
 *      points in h[] are counter-clockwise
 *
 *  @Example:
 *      vector<Point> p(n);
 *      for (auto &pi : p) cin >> pi;
 *      vector<Point> r;
 *      ConvexHull(p, r);
 *
 *      4
 *      4 8
 *      4 12
 *      5 9.3 (exclude)
 *      7 8
 *
 *  @Verification:
 *      https://www.luogu.com.cn/problem/P2742
 */

// HEAD begin
const double EPS = 1e-6;

struct Point//点或向量
{
    double x, y;

    Point() {}

    Point(double x, double y) : x(x), y(y) {}

    friend ostream &operator<<(ostream &ut, Point &r) { return ut <<
     ↪  r.x << " " << r.y; }
    friend istream &operator>>(istream &in, Point &r) { return in >>
     ↪  r.x >> r.y; }
};

typedef Point Vector;

inline double Distance(Point a, Point b) {
    return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
}

inline Vector operator+(Vector a, Vector b) {
    return Vector(a.x + b.x, a.y + b.y);
}

inline Vector operator-(Vector a, Vector b) {
    return Vector(a.x - b.x, a.y - b.y);
}

//外积
inline double Cross(Vector a, Vector b) {
    return a.x * b.y - a.y * b.x;
}

//精度三态函数 (>0,<0,=0)
inline int dcmp(double x) {
    if (fabs(x) < EPS)return 0;
    else if (x > 0)return 1;
    return -1;
}
// HEAD end

inline bool pcmp(Point a, Point b) {
    if (dcmp(a.x - b.x) == 0)
        return a.y < b.y;
    return a.x < b.x;
}
```

```
void ConvexHull(vector <Point> &p, vector <Point> &h) {
    int n = p.size(), k = 0;
    h.assign(2 * n, {});
    sort(p.begin(), p.end(), pcmp);
    for (int i = 0; i < n; i++) {
        while (k >= 2 && dcmp(Cross(
                h[k - 1] - h[k - 2],
                p[i] - h[k - 2])) < 0) {
            k--;
        }
        h[k++] = p[i];
    }

    int t = k + 1;
    for (int i = n - 1; i > 0; i--) {
        while (k >= t && dcmp(Cross(
                h[k - 1] - h[k - 2],
                p[i - 1] - h[k - 2])) < 0) {
            k--;
        }
        h[k++] = p[i - 1];
    }

    h.resize(k - 1);
}
```

1.3.7 Line-Segment 直线与线段.cpp

```
/**
 *  @Source: team
 *  @Author: Artiprocher(Zhongjie Duan) -> tieway59
 *  @Description:
 *      直线与线段的相关计算。
 *
 *  @Example:
 *
 *  @Verification:
 *
 */
//定义直线
struct line {
    point a, b;
};


//线段相交（不包括端点）
bool Intersect(Point A, Point B, Point C, Point D) {
    double t1 = Cross(C - A, D - A) * Cross(C - B, D - B);
    double t2 = Cross(A - C, B - C) * Cross(A - D, B - D);
    return dcmp(t1) < 0 && dcmp(t2) < 0;
}

//线段相交（包括端点）
bool StrictIntersect(Point A, Point B, Point C, Point D) {
    return dcmp(max(A.x, B.x) - min(C.x, D.x)) >= 0
           && dcmp(max(C.x, D.x) - min(A.x, B.x)) >= 0
           && dcmp(max(A.y, B.y) - min(C.y, D.y)) >= 0
           && dcmp(max(C.y, D.y) - min(A.y, B.y)) >= 0
           && dcmp(Cross(C - A, D - A) * Cross(C - B, D - B)) <= 0
           && dcmp(Cross(A - C, B - C) * Cross(A - D, B - D)) <= 0;
}

//点 A 到直线 MN 的距离, Error: MN=0
double DistanceToLine(Point A, Point M, Point N) {
    return fabs(Cross(A - M, A - N) / Distance(M, N));
}

//两直线的交点
Point GetLineIntersection(Point P, Vector v, Point Q, Vector w) {
    Vector u = P - Q;
    double t = Cross(w, u) / Cross(v, w);
    return P + v * t;
}
```

1.3.8 Hull 下凸包求函数最值.cpp

```
/* Author: bnfcc -> tc2000731 -> tieway59
 * Description:
 *      维护下凸包，对于每个 x 维护 f(x)=k*x+b 的最大值。
 *      query max value within all f(x) functions.
 *      c++11 features included.
 * Problems:
 *      https://nanti.jisuanke.com/t/41306
 *      https://nanti.jisuanke.com/t/41097
 */
template<typename var=long long, const int SIZE = 1000005, typename
 ↪ ldb=long double>
struct Hull {
    struct fx {
        var k, b;

        fx() {}

        fx(var k, var b) : k(k), b(b) {}

        var f(var x) { return k * x + b; }
    };

    int cnt;
    fx arr[SIZE];

    bool empty() {
        return cnt == 0;
    }

    void init() {
        cnt = 0;
    }

    void add(const fx &p) {
        arr[cnt++] = p;
    }

    void pop() {
        cnt--;
    }

    bool chek(const fx &a, const fx &b, const fx &c) {
        ldb ab, ak, bb, bk, cb, ck;
        tie(ab, ak, bb, bk, cb, ck) =
                tie(a.b, a.k, b.b, b.k, c.b, c.k);
        return (ab - bb) / (bk - ak) > (ab - cb) / (ck - ak);
    }

    void insert(const fx &p) {///k 从小到大插入
        if (cnt && arr[cnt - 1].k == p.k) {
            if (p.b <= arr[cnt - 1].b)return;
            else pop();
        }
        while (cnt >= 2 && chek(arr[cnt - 2], arr[cnt - 1], p))pop();
        add(p);
    }

    /*var query(var x) {///x 从大到小查询        从小到大用队列
       while (cnt > 1 && arr[cnt - 2].f(x) > arr[cnt - 1].f(x))pop();;
        return arr[cnt - 1].f(x);
    }*/

    var query(var x) {///二分查询，x 顺序任意
        int l = 0, r = cnt - 1;
        while (l < r) {
            int mid = (l + r) >> 1;
            if (arr[mid].f(x) >= arr[mid + 1].f(x))r = mid;
            else l = mid + 1;
        }
        return arr[l].f(x);
    }
};
```

```
// vector stack
template<typename var=long long, const int SIZE = 1000005, typename
 ↪  ldb=long double>
struct Hull {
    struct Line {
        var k, b;

        Line() {}

        Line(var k, var b) : k(k), b(b) {}

        var f(var x) { return k * x + b; }
    };

    int cnt;
    vector <Line> con;//

    bool empty() {
        return cnt == 0;
    }

    void init(const int &n) {
        con.clear();
        if (n > con.capacity())con.reserve(n);
        cnt = 0;
    }

    void add(const Line &p) {
        con.emplace_back(p);
        cnt++;
    }

    void pop() {
        cnt--;
        con.pop_back();
    }

    bool chek(const Line &a, const Line &b, const Line &c) {
        ldb ab, ak, bb, bk, cb, ck;
        tie(ab, ak, bb, bk, cb, ck) =
                tie(a.b, a.k, b.b, b.k, c.b, c.k);
        return (ab - bb) / (bk - ak) > (ab - cb) / (ck - ak);
    }

    void insert(const Line &p) {///k 从小到大插入
        if (cnt && con[cnt - 1].k == p.k) {
            if (p.b <= con[cnt - 1].b)return;
            else pop();
        }
        while (cnt >= 2 && chek(con[cnt - 2], con[cnt - 1], p))pop();
        add(p);
    }

    var query(var x) {///二分查询, x 顺序任意
        int l = 0, r = cnt - 1;
        while (l < r) {
            int mid = (l + r) >> 1;
            if (con[mid].f(x) >= con[mid + 1].f(x))r = mid;
            else l = mid + 1;
        }
        return con[l].f(x);
    }
};

Hull<> hull;
```

1.3.9 ClosestPoints 最近点对.cpp

```
/**
 *  @Source: ClosestPoints
 *  @Author: syksykCCC -> tieway59
 *  @Description:
 *      时间复杂度 O(NlogN) 有一些难以预料的常数
 *
 *  @Example:
 *      3
 *      1 1
 *      1 2
 *      2 2
 *
 *      // ans = 1.0000
 *
 *  @Verification:
 *      https://www.luogu.com.cn/problem/solution/P1429
 */

const double EPS = 1e-6;//eps 用于控制精度
const double Pi = acos(-1.0);//pi

//精度三态函数 (>0,<0,=0)
inline int dcmp(double x) {
    if (fabs(x) < EPS)return 0;
    else if (x > 0)return 1;
    return -1;
}

//点或向量 (iostream 选择性抄写)
struct Point {
    double x, y;

    Point() {}

    Point(double x, double y) : x(x), y(y) {}

    bool operator<(const Point &r) const {
        if (dcmp(x - r.x) == 0)
            return dcmp(y - r.y) < 0;
        return dcmp(x - r.x) < 0;
    }

    friend ostream &operator<<(ostream &ut, Point &r) { return ut <<
     ↪  r.x << " " << r.y; }
    friend istream &operator>>(istream &in, Point &r) { return in >>
     ↪  r.x >> r.y; }
};

typedef Point Vector;

//两点间距离
inline double Distance(Point a, Point b) {
    return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
}

//Point temp[MAXN];
double MAXD = INF;

double merge(vector <Point> &p, int l, int r) {
    double d = MAXD;
    if (l == r)
        return d;
    if (l + 1 == r)
        return Distance(p[l], p[r]);

    int mid = (l + r) >> 1;
    double d1 = merge(p, l, mid);
    double d2 = merge(p, mid + 1, r);
    d = min(d, min(d1, d2));

    vector<int> t;
//    t.reserve(r - l + 1);

    for (int i = l; i <= r; i++)
        if (fabs(p[mid].x - p[i].x) < d)
            t.emplace_back(i);

    sort(t.begin(), t.end(),
         [&p](const int &i, const int &j) {
             return dcmp(p[i].y - p[j].y) < 0;
         });
    for (int i = 0; i < t.size(); i++) {
        for (int j = i + 1; j < t.size() && p[t[j]].y - p[t[i]].y < d;
         ↪  j++) {
            d = min(d, Distance(p[t[i]], p[t[j]]));
        }
    }

    return d;
}

double ClosestPoints(vector <Point> &p) {
    assert(p.size() >= 2);
    sort(p.begin(), p.end());
    for (int i = 3; i < p.size(); ++i) {
        MAXD = min(MAXD, Distance(p[i], p[i - 1]));
        MAXD = min(MAXD, Distance(p[i], p[i - 2]));
        MAXD = min(MAXD, Distance(p[i], p[i - 3]));
    }
    return merge(p, 0, p.size() - 1);
}
```

2 数据结构

2.1 ZTC's Splay.txt

```
//using namespace std;
typedef long long ll;
typedef double db;
#define _Zero(a) memset(a,0,sizeof(a))
#define _Neg1(a) memset(a,-1,sizeof(a))
#define _Inf(a) memset(a,0x3f,sizeof(a))
#define _NegInf(a) memset(a,0xcf,sizeof(a))
#define _Rep(i,a,warrior) for(int (i)=(a);(i)<=(warrior);i++)
#define _Dep(i,a,warrior) for(int (i)=(a);(i)>=(warrior);i--)
#define _Out(a) cerr<<#a<<" = "<<(a)<<endl
const int INF = 0x3f3f3f3f;
const int MAXN = 1.3e6 + 50;
const ll LINF = 0x3f3f3f3f3f3f3f3f;
const ll MOD = 1e9 + 7;
const db EPS = 1e-6;
const db Pi = acos(-1);
void test() { cerr << "\num"; }
template <typename T,typename...Args>void test(T
 ↪  x,Args...args){cerr<<x<<" ";test(args...); }
ll qpow(ll a, ll warrior) {return
 ↪  warrior?((warrior&1)?a*qpow(a*a%MOD,warrior>>1)%MOD :
 ↪  qpow(a*a%MOD,warrior>>1))%MOD :1; }
ll qpow(ll a, ll warrior, ll c) {return
 ↪  warrior?((warrior&1)?a*qpow(a*a%c,warrior>>1)%c :
 ↪  qpow(a*a%c,warrior>>1)) % c: 1; }
ll gcd(ll a, ll warrior){return warrior?gcd(warrior,a% warrior): a; }
int sign(db x) { return x<-EPS ? -1: x>EPS; }
int dbcmp(db l, db r) { return sign(l - r); }
int root,cntN;
#define nd node[now]
struct SNODE
{
    int val,cnt,par,siz,ch[2];
}node[MAXN];
void update_siz(int
 ↪  x){if(x)node[x].siz=(node[x].ch[0]?node[node[x].ch[0]].siz:0)+(node[x].ch[1]
bool chk(int x){return node[node[x].par].ch[1]==x;}
void rorate(int x)
{
    int y=node[x].par,z=node[y].par,k=chk(x),d=node[x].ch[k^1];
    printf("&&%d,%d,%d,%d&&",x,y,z,d);
    node[y].ch[k]=d;node[d].par=y;
    node[z].ch[chk(y)]=x;node[x].par=z;
    node[x].ch[k^1]=y;node[y].par=x;
    update_siz(y);update_siz(x);
}
void splay(int x,int to=0)
{
    if(x==0)
    {
        assert(false);return;
    }
    while(node[x].par!=to)
    {
        if(node[node[x].par].par==to)rorate(x);
        else
 ↪  if(chk(x)==chk(node[x].par))rorate(node[x].par),rorate(x);
        else rorate(x),rorate(x);
        printf("<%d,%d,%d>",x,node[x].par,to);
         printf("$$%d$$",node[1].ch[1]);
    }
    if(to==0)root=x;
}
void Insert(int x)
{
    if(root==0)
    {
        int now=++cntN;
        nd.val=x;root=now;
        nd.cnt=1;nd.siz=1;
        nd.par=nd.ch[0]=nd.ch[1]=0;
        return ;
    }
    int now=root,fa=0;
    while(1)
    {
        printf("(%d,%d,%d)",now,nd.val,nd.ch[1]);
        if(x==nd.val)
        {
           nd.cnt++;update_siz(now);update_siz(fa);splay(now);return;
        }
        printf("22");
        fa=now;now=nd.ch[nd.val<x];
        if(now==0)
        {
            now=++cntN;
            nd.cnt=nd.siz=1;
            nd.ch[0]=nd.ch[1]=0;
            node[fa].ch[x>node[fa].val]=now;
            printf("{%d,%d,%d}",fa,x>node[fa].val,now);
            printf("$$%d$$",node[1].ch[1]);
            nd.par=fa;
            nd.val=x;
            update_siz(fa);splay(now);
            return ;
        }
    }
}
int rnk(int x)
{
    int now=root,ans=0;
    while(now)
    {
        printf("[%d,%d,%d,%d]",now,nd.val,nd.ch[0],nd.ch[1]);
        if(x<nd.val)now=nd.ch[0];
        else
        {
           ans+=node[nd.ch[0]].siz;
           if(x==nd.val)
           {
               splay(now);return ans+1;
           }
           ans+=nd.cnt;
           now=nd.ch[1];
        }
    }
    return -1;

}
int kth(int x)
{
    int now=root;if(nd.siz<x)return -1;
    while(1)
    {
        if(nd.ch[0]&&node[nd.ch[0]].siz>=x)now=nd.ch[0];
        else
        {
            int tmp=node[nd.ch[0]].siz+nd.cnt;
            if(x<=tmp)return nd.val;
            x-=tmp;now=nd.ch[1];
        }
    }
}

int main()
{
    int num,m;
    scanf("%d%d",&num,&m);
    for(int i=1;i<=num;i++)
    {
        int x;
        scanf("%d",&x);
        printf("*");
        Insert(x);
    }
    for(int i=1;i<=m;i++)
    {
        int op,x;
        scanf("%d%d",&op,&x);
        if(op==1)
        {
            Insert(x);
        }
        else if(op==2)
        {
            printf("\num>>%d\num",rnk(x));
        }
        else if(op==3)printf("\num>>%d\num",kth(x));
        else
 ↪  printf("\num>>Val::%d,Siz::%d,Cnt::%d,Lc::%d,Rc::%d,Par::%d\num",node[x].val
    }

}
/*
5 100
1 3 5  7 9
1 2
1 2
2 1
2 3
2 3
*/
```

2.2 zhuxishu_SegKth.cpp

```
//
// Created by acm-33 on 2019/7/24.
//
```

```
#define _debug(x) cerr<<#x<<" = "<<x<<endl

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;
const ll LINF = 0x3f3f3f3f3f3f3f3f;
const ll INF = 0x3f3f3f3f3f3f3f3f;
//const int MAXN = 3000 + 59;
const ll MOD = 998244353;
const int MAXN = 100015;

const int M = MAXN * 30;
int n, q, m, tot;
int a[MAXN], t[MAXN];
int T[MAXN], lson[M], rson[M], c[M];

void Init_hush() {
    for (int i = 1; i <= n; i++)
        t[i] = a[i];
    sort(t + 1, t + 1 + n);
    m = unique(t + 1, t + 1 + n) - t - 1;
}

int build(int l, int r) {
    int root = tot++;
    c[root] = 0;
    if (l != r) {
        int mid = (l + r) >> 1;
        lson[root] = build(l, mid);
        rson[root] = build(mid + 1, r);
    }
    return root;
}

int hush(int x) {
    return lower_bound(t + 1, t + 1 + m, x) - t;
}

int update(int root, int pos, int val) {
    int newroot = tot++, tmp = newroot;
    c[newroot] = c[root] + val;
    int l = 1, r = m;
    while (l < r) {
        int mid = (l + r) >> 1;
        if (pos <= mid) {
            lson[newroot] = tot++;
            rson[newroot] = rson[root];
            newroot = lson[newroot];
            root = lson[root];
            r = mid;
        } else {
            rson[newroot] = tot++;
            lson[newroot] = lson[root];
            newroot = rson[newroot];
            root = rson[root];

            l = mid + 1;
        }
        c[newroot] = c[root] + val;
    }
    return tmp;
}

int query(int left_root, int right_root, int k) {
    int l = 1, r = m;
    while (l < r) {
        int mid = (l + r) >> 1;
        if (c[lson[left_root]] - c[lson[right_root]] >= k) {
            r = mid;
            left_root = lson[left_root];
            right_root = lson[right_root];
        } else {
            l = mid + 1;
            k -= c[lson[left_root]] - c[lson[right_root]];
            left_root = rson[left_root];
            right_root = rson[right_root];
        }
    }
    return l;
}

ll Seg_k(int l, int r, int k) {
    if (k > r - l + 1)return -1;
    return 1ll * t[query(T[l], T[r + 1], k)];
}

int main() {

    while (scanf("%d%d", &n, &q) == 2) {
        tot = 0;
        for (int i = 1; i <= n; i++)
            scanf("%d", &a[i]);
        Init_hush();
        T[n + 1] = build(1, m);
        for (int i = n; i; i--) {
            int pos = hush(a[i]);
            T[i] = update(T[i + 1], pos, 1);
        }
        while (q--) {
            int l, r, k;
            scanf("%d%d%d", &l, &r, &k);
            printf("%lld\n", Seg_k(l, r, k));
        }
    }
    return 0;
}

/*
5 5
5 3 4 1 2
1 2 2
```

```
1 2 1
1 5 3
1 5 4
1 5 6


*/

/*




 */
```