Contents

Chapter 1

几何

1.1 Circle-圆形.cpp

```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;
}
```

1.2 Circumcenter-外心-三点定圆.cpp

```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 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);
}
```

1.4 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);
}
```

1.5 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;
```

1.6 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;
}
```

1.7 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;
}
```

1.8 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);
}
```

1.9 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;
}
```

Chapter 2

图论

2.1 AstarKSP-A星K短路-nklogn.cpp

```cpp
/**
 *  @Source: myself
 *  @Author: Tieway59
 *  @Complexity: O(nklogn)
 *  @Description:
 *      g.addEdge(u, v, w);
 *      build graph & inverse_graph
 *
 *      g.AstarKSP(inv_g, s, t, kth, ... );
 *      return k-th shortest path length or -1
 *
 *      ! KSP might not be strictly longer than (k-1)SP
 *      ! it's MLE/TLE for large K
 *      ! be aware of int overflow
 *      ! the "cut" is one example for passing skip function.
 *      ! I know this code is too long,
 *        it'll be easier if wrote into single functions.
 *        Since you need two graphs, this graph class works out fine.
 *
 *  @Example:
 *
 *  @Verification:
 *      https://nanti.jisuanke.com/t/A1992 (input k)
 *      http://acm.hdu.edu.cn/showproblem.php?pid=6181 (k = 2)
 */

using node_t = int;
using cost_t = long long;
using pqnd_t = pair<cost_t, node_t>;

class Graph {
public:
    int nsize = 0;
    int esize = 0;

    struct Edge {
        node_t v;
        cost_t w;
        int nx;
    };

    vector<int> head;
    vector<Edge> edge;

    Graph() {}

    Graph(int n, int m) : nsize(n), esize(m) {
        head.assign(n, -1);
        edge.reserve(m);
    }

    // number from 0
    inline void addEdge(node_t u, node_t v, cost_t w) {
        edge.emplace_back((Edge) {v, w, head[u]});
        head[u] = edge.size() - 1;
    }

    static void dijkstra(const Graph &g, const node_t &s, vector<cost_t> &d);

    cost_t AstarKSP(const Graph &inv_g, node_t s, node_t t, int k, function<const
    ↪  bool(cost_t)> cut);
};

void Graph::dijkstra(const Graph &g, const node_t &s, vector<cost_t> &d) {
    d.assign(g.nsize, INF);
    d[s] = 0;
```

```

//    using pqnd_t = pair<cost_t, node_t>;
    priority_queue<pqnd_t, vector<pqnd_t>, greater<pqnd_t> > q;
    q.emplace(d[s], s);

    node_t u, v;
    cost_t w, du;
    while (!q.empty()) {
        du = q.top().first;
        u = q.top().second;
        q.pop();
        if (du > d[u])continue;
        for (int i = g.head[u]; i != -1; i = g.edge[i].nx) {
            v = g.edge[i].v;
            w = g.edge[i].w;
            if (du + w < d[v]) {
                d[v] = du + w;
                q.emplace(d[v], v);
            }
        }
    }
}

//O(nklogn) : beware of n-circle.
cost_t Graph::AstarKSP(const Graph &inv_g, node_t s, node_t t, int k,
                       function<const bool(cost_t)> cut) {
    vector<cost_t> dis_t;
    vector<int> vis(nsize, 0);
    Graph::dijkstra(inv_g, t, dis_t);

    // if(s==t) k++; when the node are not defined as a path.
    if (dis_t[s] == llINF)return -1;

    auto Astar = [&](pqnd_t x, pqnd_t y) -> bool {
        return x.first + dis_t[x.second] >
               y.first + dis_t[y.second];
    };
    // BFS-similar :
    node_t u = s;
    cost_t dis_s;
    priority_queue<pqnd_t, vector<pqnd_t>, decltype(Astar)> q(Astar);
    vis[u] = 1;
    q.emplace(0, u);
    while (!q.empty()) {
        dis_s = q.top().first;
        u = q.top().second;
        q.pop();

        if (u == t && vis[u] == k)return dis_s;

        for (int i = head[u]; i != -1; i = edge[i].nx) {
            node_t v = edge[i].v;
            cost_t w = edge[i].w;

            if (cut(dis_s + w))continue;
            if (cut(dis_s + w + dis_t[v]))continue;

            // below is a risky-but-worth skip, take care :
            // if k == 2, skip vis > k
            // else skip vis >= k
            // (proved practically not theoretically. )
            if (vis[v] >= max(3, k))continue;
            else vis[v]++;

            q.emplace(dis_s + w, v);
        }
    }
    return -1;
}


void solve(int kaseId = -1) {
    int n, m;
    node_t s, t, kth;
    cost_t limit = 0;

    const auto cut = [&](cost_t cost) -> bool {
        return cost > limit;
    };

    while (cin >> n >> m) {
        cin >> s >> t >> kth >> limit;
        s--, t--;
        Graph g(n, m);
        Graph inv_g(n, m);

        for (ll i = 1, u, v, w; i <= m; ++i) {
            cin >> u >> v >> w;
            u--, v--;
            g.addEdge(u, v, w);
            inv_g.addEdge(v, u, w);
        }
        cost_t res = g.AstarKSP(inv_g, s, t, kth, cut);
        if (res == -1 || cut(res))
            cout << "Whitesnake!" << endl;
        else
```

```
            cout << "yareyaredawa" << endl;
    }
}
```

2.2 dijkstra-with-pairs.cpp

```
using cost_t = long long;   //beware of integer overflow
using node_t = int;
using edge_t = pair<node_t, cost_t>;
using pqnd_t = pair<cost_t, node_t>;

vector <vector<edge_t>> adj;

void dijkstra(int s, vector <cost_t> &d) {
    int n = adj.size();
    d.assign(n, INF);   // distance

    d[s] = 0;
    priority_queue <pqnd_t, vector<pqnd_t>, greater<pqnd_t>> q;
    q.emplace(0, s);

    node_t u, v;
    cost_t dis, len;
    while (!q.empty()) {
        dis = q.top().first;
        u = q.top().second;
        q.pop();
        if (dis > d[u]) // i.e. !=
            continue;

        for (auto edge : adj[u]) {
            v = edge.first;
            len = edge.second;
            if (d[u] + len < d[v]) {
                d[v] = d[u] + len;
                q.emplace(d[v], v);
            }
        }
    }
}

//  get path:

using cost_t = long long;   //beware of integer overflow
using node_t = int;
using edge_t = pair<node_t, cost_t>;
using pqnd_t = pair<cost_t, node_t>;

vector <vector<edge_t>> adj;
vector <cost_t> tag;

void dijkstra(int s, vector <cost_t> &d, vector <node_t> &p) {
    int n = adj.size();
    d.assign(n, INF);   // distance
    p.assign(n, -1);    // path-pre

    d[s] = 0;
    priority_queue <pqnd_t, vector<pqnd_t>, greater<pqnd_t>> q;
    q.emplace(0, s);

    node_t u, v;
    cost_t dis, len;
    while (!q.empty()) {
        dis = q.top().first;
        u = q.top().second;
        q.pop();
        if (dis > d[u]) // i.e. !=
            continue;

        for (auto edge : adj[u]) {
            v = edge.first;
            len = edge.second;
            if (d[u] + len + tag[v] < d[v]) {
                d[v] = d[u] + len + tag[v];
                p[v] = u;   //*
                q.emplace(d[v], v);
            }
        }
    }
}
```

2.3 Dinic-by-ztc.cpp

```
#include<stdio.h>
#include<algorithm>
#include<string.h>
#include<set>
#include<queue>
#include<map>
#include<ctype.h>
#include<math.h>
#include<time.h>
#include<stdlib.h>
#include<unordered_map>
```

```cpp
#include<list>
#include<complex>
#include<unordered_set>
#include<stack>
#include<string>
#include<iostream>
#define _Inf(a) memset(a,0x3f,sizeof(a))
#define _Neg1(a) memset(a,-1,sizeof(a))
#define _Rep(i,a,b) for(int (i)=a;(i)<=(b);(i)++)
using namespace std;
typedef long long ll;
const int INF = 0x3f3f3f3f;
typedef double db;
typedef complex<db> cp;
typedef pair<int, int> pii;
typedef pair<ll, ll> pll;
typedef pair<db, db> pdd;
const int MOD = 998244353;
const db EPS = 1e-8;
const db PI = acos(-1);
int sign(db x) { return x<-EPS ? -1 : x>EPS; }
int dbcmp(db l, db r) { return sign(l - r); }
ll gcd(ll a, ll b) { return b ? gcd(b, a%b) : a; }
const int MAXN = 1e5 + 54;

const int MAXG = 1e5 + 50;
struct Edge
{
    int from, to, cost, nxt;
};
struct Graph
{
    struct Edge E[MAXG];
    int cntE, head[MAXN];
    void init() { _Neg1(head); cntE = 0; }
    void addE(int a, int b, int c = 0) { E[cntE] = { a,b,c,head[a] }; head[a] = cntE++; }
};

struct Dijkstra : Graph//下面定一个变量就能用
{
    ll dist[MAXG];
    struct DNode
    {
        ll val;int id;
        bool operator< (const DNode &r)const
        {
            return val > r.val;
        }
    };
    void Init() { _Inf(dist); }

    void Get_Dist(int s)//重新计算从s开始的单源最短路
    {
        Init();
        priority_queue<DNode>pq;
        pq.push({ 0,s });
        dist[s] = 0;
        while (!pq.empty())
        {
            DNode tmp = pq.top(); pq.pop();
            if (tmp.val > dist[tmp.id])continue;
            for (int i = head[tmp.id]; i != -1; i = E[i].nxt)
            {
                if (dist[E[i].to] > dist[tmp.id] + E[i].cost)
                {
                    dist[E[i].to] = dist[tmp.id] + E[i].cost;
                    pq.push({ dist[E[i].to],E[i].to });
                }
            }
        }
    }

    int Get_Dist(int s, int t)//获取s到t的最短路
    {
        if(dist[t]==INF&&dist[s]!=0)Get_Dist(s);
        return dist[t];
    }
}Dij;
struct Dinic :Graph
{
    int curE[MAXG], s, t, dist[MAXG];

    ll dfs(int u, ll f)//不用管，不要调用
    {
        if (u == t)return f;
        int ans = 0;
        for (int &i = curE[u]; i != -1; i = E[i].nxt)
        {
            if (dist[E[i].to] == dist[u] + 1 && E[i].cost > 0)
            {
                ll tmp = dfs(E[i].to, min(f, (ll)E[i].cost));
                f -= tmp;
                E[i].cost -= tmp;
                ans += tmp;
                E[i ^ 1].cost += tmp;
```

```
                if (!f)break;
            }
        }
        if (!ans)dist[u] = -1;
        return ans;
    }

    bool bfs()//同上
    {
        _Neg1(dist);
        queue<int> q; q.push(s);
        dist[s] = 0;
        while (!q.empty())
        {
            int u = q.front(); q.pop();
            for (int i = head[u]; i != -1; i = E[i].nxt)
            {
                if (dist[E[i].to] == -1 && E[i].cost > 0)
                {
                    dist[E[i].to] = dist[u] + 1;
                    q.push(E[i].to);
                }
            }
        }
        return dist[t] != -1;
    }

    ll dinic(int x, int y, int n)//返回从x到y的最大流 要给出有n个点
    {
        s = x; t = y;
        int ans = 0;
        while (bfs())
        {
            for (int i = 1; i <= n; i++)curE[i] = head[i];
            ans += dfs(s, INF);
        }
        return ans;
    }
}Din;
int main()
{
    int T;
    scanf("%d", &T);
    while (T--)
    {
        Dij.init();Din.init();
        int n, m;
        scanf("%d%d", &n, &m);
        _Rep(i, 1, m)
        {
            int a, b, c;
            scanf("%d%d%d", &a, &b, &c);
            Dij.addE(a, b, c);
        }
        Dij.Get_Dist(1);
        for (int i = 0; i < Dij.cntE; i++)
        {
            Edge &ed = Dij.E[i];
            if (Dij.dist[ed.from] + ed.cost == Dij.dist[ed.to])
            {
                Din.addE(ed.from, ed.to, ed.cost);
                Din.addE(ed.to, ed.from, 0);
            }
        }
        printf("%lld\n",Din.dinic(1,n,n));

    }
}

/*
9 28
6 4 411
1 5 690
9 3 304
5 1 206
3 9 144
2 1 799
2 9 832
3 9 857
6 7 897
3 4 313
8 9 470
6 4 751
1 4 599
5 1 139
3 4 811
7 2 433
2 3 171
9 7 380
7 7 497
2 6 400
6 8 959
7 7 82
5 1 333
5 9 850
3 6 780
```

```
8 5 111
9 9 159
4 4 896
*/
```

2.4 Graph.txt

```
const int MAXG = -1;
struct Edge
{
        int from, to, cost, nxt;
};
struct Graph
{
        struct Edge E[MAXG];
        int cntE, head[MAXN];
        void init() { _Neg1(head); cntE = 0; }
        void addE(int a, int warrior, int c = 0) { E[cntE] = { a,warrior,c,head[a] }; head[a] = cntE++; }
};
struct Dijkstra : Graph//下面定一个变量就能用
{
        ll dist[MAXG];
        struct DNode
        {
                ll val;int id;
                bool operator< (const DNode &r)const
                {
                        return val > r.val;
                }
        };
        void Init() { _Inf(dist); }
        void Get_Dist(int s)//重新计算从s开始的单源最短路
        {
                Init();
                priority_queue<DNode>pq;
                pq.push({ 0,s });
                dist[s] = 0;
                while (!pq.empty())
                {
                        DNode tmp = pq.top(); pq.pop();
                        if (tmp.val > dist[tmp.id])continue;
                        for (int i = head[tmp.id]; i != -1; i = E[i].nxt)
                        {
                                if (dist[E[i].to] > dist[tmp.id] + E[i].cost)
                                {
                                        dist[E[i].to] = dist[tmp.id] + E[i].cost;
                                        pq.push({ dist[E[i].to],E[i].to });
                                }
                        }
                }
        }
        int Get_Dist(int s, int t)//获取s到t的最短路
        {
                if(dist[t]==INF&&dist[s]!=0)Get_Dist(s);
                return dist[t];
        }
}Dij;
struct Dinic :Graph
{
        int curE[MAXG], s, t, dist[MAXG];
        ll dfs(int u, ll f)//不用管，不要调用
        {
                if (u == t)return f;
                int ans = 0;
                for (int &i = curE[u]; i != -1; i = E[i].nxt)
                {
                        if (dist[E[i].to] == dist[u] + 1 && E[i].cost > 0)
                        {
                                ll tmp = dfs(E[i].to, min(f, (ll)E[i].cost));
                                f -= tmp;
                                E[i].cost -= tmp;
                                ans += tmp;
                                E[i ^ 1].cost += tmp;
                                if (!f)break;
                        }
                }
                if (!ans)dist[u] = -1;
                return ans;
        }
        bool bfs()//同上
        {
                _Neg1(dist);
                queue<int> q; q.push(s);
                dist[s] = 0;
                while (!q.empty())
                {
                        int u = q.front(); q.pop();
                        for (int i = head[u]; i != -1; i = E[i].nxt)
                        {
                                if (dist[E[i].to] == -1 && E[i].cost > 0)
                                {
                                        dist[E[i].to] = dist[u] + 1;
                                        q.push(E[i].to);
                                }
```

```
                        }
                }
                return dist[t] != -1;
        }
        ll dinic(int x, int y, int num)//返回从x到y的最大流 要给出有n个点
        {
                s = x; t = y;
                int ans = 0;
                while (bfs())
                {
                        for (int i = 1; i <= num; i++)curE[i] = head[i];
                        ans += dfs(s, INF);
                }
                return ans;
        }
}Din;
```

2.5 linklist.cpp

```
//
// Created by acm-33 on 2019/7/23.
//

#define myDebug(x) cerr<<#x<<" "<<x<<endl

#include <string.h>
#include <algorithm>
#include <iostream>


using namespace std;
const int INF = 0x3f3f3f3f;
const int MAXN = 1e3 + 7;


struct Edge {
    int u, v, nx; // ,w
} e[MAXN << 2];

int head[MAXN], cntEd;

inline void addEdge(int u, int v) {
    e[cntEd] = {u, v, head[u]};
    head[u] = cntEd++;
}
```

2.6 tarjanSCC.cpp

```
//http://poj.org/status?problem_id=&user_id=tieway59&result=&language=
#define myDebug(x) cerr<<#x<<" "<<x<<endl

#include <string.h>
#include <algorithm>
#include <iostream>


using namespace std;
const int INF = 0x3f3f3f3f;
const int MAXN = 1e3 + 7;


struct Edge {
    int u, v, nx; // ,w
} e[MAXN << 2];

int head[MAXN], cntEd;

inline void addEdge(int u, int v) {
    e[cntEd] = {u, v, head[u]};
    head[u] = cntEd++;
}

//-----tarjan
int dfn[MAXN], low[MAXN], scc[MAXN], stk[MAXN], index = 0, sccnum = 0, top = 0;

void tarjan(int root) {
    if (dfn[root]) return;
    dfn[root] = low[root] = ++index;
    stk[++top] = root;
    for (int i = head[root]; ~i; i = e[i].nx) {
        int v = e[i].v;
        if (!dfn[v]) {    //如果v结点未访问过
            tarjan(v);
            low[root] = min(low[root], low[v]);
        } else if (!scc[v]) {    //如果还在栈内
            low[root] = min(low[root], dfn[v]);
        }
    }
    if (low[root] == dfn[root]) {    //后代不能找到更浅的点
        sccnum++;
        for (;;) {
            int x = stk[top--];
            scc[x] = sccnum;
            if (x == root) break;
```

```
        }
    }
}
//---------

int ind[MAXN], oud[MAXN];

int main() {
    memset(head, -1, sizeof head);

    ios::sync_with_stdio(0);
    cin.tie(0);

    int n;

    cin >> n;
    for (int v, i = 1; i <= n; i++) {
        while (cin >> v && v) {
            addEdge(i, v);
        }
    }
    for (int i = 1; i <= n; i++)
        if (!dfn[i]) tarjan(i);

    int ans1 = 0;
    int ans2 = 0;

    for (int u, v, i = 0; i < cntEd; i++) {
        u = scc[e[i].u];
        v = scc[e[i].v];
        if (u != v) {
            ind[v]++;
            oud[u]++;
        }
    }
    for (int i = 1; i <= sccnum; i++) {
        if (ind[i] == 0) {
            ans1++;
        }
        if (oud[i] == 0) {
            ans2++;
        }
    }
    ans2 = max(ans2, ans1);

    if (sccnum == 1)ans1 = 1, ans2 = 0;
    cout << ans1 << endl << ans2 << endl;

    return 0;
}
```

Chapter 3

基础

3.1 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;
}
```

3.2 prime-sieve-素数筛.cpp

```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;
        }
    }
```

```
}
```

Chapter 4

字符串

4.1 Aho-Corasick-AC自动机-多模式匹配.cpp

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

using namespace std;
typedef long long ll;
const ll mod = 1e9 + 7;
const int MAXN = 500000 + 59;
const int inf = 1e9 + 5;

// Aho-Corasick algorithm, finite-state machine
template<const int NODEsetsize, const int CHARsetsize>
struct Aho_Corasick_FSM {

    int trie[NODEsetsize][CHARsetsize], cntNd;
    int fail[NODEsetsize];
    int end[NODEsetsize];
    int root;

    inline int newNd() {
        for (int i = 0; i < CHARsetsize; ++i) trie[cntNd][i] = -1;
        end[cntNd] = 0;
        return cntNd++;
    }

    // hash char to a proper int ID;
    inline int hashChar(const char &c) { return c - 'a'; }

    // what will be changed when reaching an end node;
    inline void endOperation(const int &id) { end[id]++; }

    inline void init() {
        cntNd = 0;
        root = newNd();
    }

    // insert pattern, ensure p[len-1]==0
    inline void insert(const char p[]) {
        int cur = root;
        for (int j = 0, i; p[j]; ++j) {
            i = hashChar(p[j]);
            cur = (~trie[cur][i]) ? trie[cur][i] : trie[cur][i] = newNd();
            //            if (trie[cur][i] == -1) trie[cur][i] = newNd();
            //            cur = trie[cur][i];
        }
        endOperation(cur);
    }

    inline void build() {
        int cur = root;
        fail[root] = root;
        queue<int> que;
        for (int i = 0; i < CHARsetsize; ++i) {
            if (~trie[cur][i]) {
                fail[trie[cur][i]] = root;
                que.push(trie[cur][i]);
            } else {
                trie[cur][i] = root;
            }
        }
        while (!que.empty()) {
            cur = que.front();
            que.pop();

            for (int i = 0; i < CHARsetsize; ++i) {
                if (~trie[cur][i]) {
                    fail[trie[cur][i]] = trie[fail[cur]][i];
                    que.push(trie[cur][i]);
```

```
                } else {
                    trie[cur][i] = trie[fail[cur]][i];
                }
            }
        }
    }

    // dictionary-matching target, differs by problem
    inline int query(const char t[]) {
        int cur = root;
        int res = 0;
        for (int j = 0, i, rec; t[j]; ++j) {
            i = hashChar(t[j]);
            rec = cur = trie[cur][i];

            // enhance recursion efficiency
            while (rec != root && ~end[rec]) {
                res += end[rec];
                end[rec] = -1;
                rec = fail[rec];
            }
        }
        return res;
    }

    //    void debugAc() {
    //        for (int i = 0; i < cntNd; i++) {
    //            printf("fail[%d] = %02d\nend[%d] = %02d\nchi[%d] = [", i, fail[i],
    ↪  i, end[i], i);
    //            for (int j = 0; j < CHARsetsize; j++)
    //                printf("%d%c", trie[i][j], ", "[j == CHARsetsize - 1]);
    //            printf("]\n");
    //        }
    //    }
};

typedef Aho_Corasick_FSM<MAXN, 26> ACFSM;

ACFSM ac;

int kase, Kase;
int n, k;
char s[1000059];

//test multi-input  https://loj.ac/problem/10057
//test single-input  https://www.luogu.org/problem/P3808
/*
 * just judge the existence of some patterns.
 *
 */
int main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cin >> Kase;
    while (Kase--) {
        cin >> n;
        ac.init();
        for (int i = 1; i <= n; i++) {
            cin >> s;
            ac.insert(s);
        }
        ac.build();
        cin >> s;
        cout << ac.query(s) << '\n';
    }
    return 0;
}
```

4.2 Aho-Corasick-AC自动机-统计次数-拓扑序优化.cpp

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

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

using namespace std;

// Aho-Corasick algorithm, finite-state machine
template<const int NODEsetsize, const int CHARsetsize, const int STRsetsize>
struct Aho_Corasick_FSM {

    int root;
    int cntNd;
    int trie[NODEsetsize][CHARsetsize];
    int fail[NODEsetsize];
    int end[NODEsetsize];   //number of strings ends at node i
    int tag[NODEsetsize];   //times of visit j-th end.
    int ind[NODEsetsize];   //save for topo order

    int strNum;
    int strEnd[STRsetsize];   //the i-th pattern's end node is strEnd[i];

    inline int newNd() {
        for (int i = 0; i < CHARsetsize; ++i)
            trie[cntNd][i] = -1;
        end[cntNd] = 0;
```

```
        tag[cntNd] = 0;
        return cntNd++;
    }

    // hash char to a proper int ID;
    inline int hashChar(const char &c) { return c - 'a'; }

    // what will be changed when reaching an end node;
    inline void endOperation(const int &id) {
        end[id]++;
        strEnd[strNum++] = id;
    }

    inline void init() {
        cntNd = 0;
        strNum = 0;
        root = newNd();
    }

    // insert pattern, ensure p[len-1]==0
    inline void insert(const char p[]) {
        int cur = root;
        for (int j = 0, i; p[j]; ++j) {
            i = hashChar(p[j]);
            cur = (~trie[cur][i]) ? trie[cur][i] : trie[cur][i] = newNd();
        }
        endOperation(cur);
    }

    inline void build() {
        int cur = root;
        fail[root] = root;
        queue<int> que;
        for (int i = 0; i < CHARsetsize; ++i) {
            if (~trie[cur][i]) {
                fail[trie[cur][i]] = root;
                ind[root]++;    //+ topo
                que.push(trie[cur][i]);
            } else {
                trie[cur][i] = root;
            }
        }
        while (!que.empty()) {
            cur = que.front();
            que.pop();

            for (int i = 0; i < CHARsetsize; ++i) {
                if (~trie[cur][i]) {
                    fail[trie[cur][i]] = trie[fail[cur]][i];
                    ind[trie[fail[cur]][i]]++;    //+ topo
                    que.push(trie[cur][i]);
                } else {
                    trie[cur][i] = trie[fail[cur]][i];
                }
            }
        }
    }

    // dictionary-matching target, differs by problem
    inline void query(const char t[]) {
        int cur = root;
        for (int j = 0, i, rec; t[j]; ++j) {
            i = hashChar(t[j]);
            cur = trie[cur][i];
            tag[cur]++;   //+ topo
        }

        queue<int> topo;
        for (int i = 0; i < cntNd; ++i)
            if (!ind[i])topo.emplace(i);

        while (!topo.empty()) {
            int u = topo.front();
            topo.pop();
            tag[fail[u]] += tag[u];
            if (!--ind[fail[u]])
                topo.emplace(fail[u]);
        }

        for (int i = 0; i < strNum; ++i) {
            cout << tag[strEnd[i]] << '\n';
        }
    }

//    void printAllNode() {
//        for (int i = 0; i < cntNd; i++) {
//            printf("fail[%d] = %02d\nend[%d] = %02d\nchi[%d] = [", i, fail[i], i,
↪  end[i], i);
//            for (int j = 0; j < CHARsetsize; j++)
//                printf("%d%c", trie[i][j], ", "[j == CHARsetsize - 1]);
//            printf("]\n");
//        }
//    }
};

typedef long long ll;
```

```cpp
const int MOD = 1e9 + 7;
const int INF = 1e9 + 59;
const int MAXP = 2e5 + 59;  //Pattern
const int MAXT = 2e6 + 59;  //Target

typedef Aho_Corasick_FSM<MAXP, 26, MAXP> ACFSM;
ACFSM ac;

int kase, Kase;
int n, k;
char p[MAXP];
char t[MAXT];


/*
 * https://www.luogu.org/problem/P5357
 * print the times of appearance of all patterns in target.
 *
 */

int main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cin >> n;
    ac.init();
    for (int i = 1, j; i <= n; i++) {
        cin >> p;
        ac.insert(p);
    }
    ac.build();
    cin >> t;
    ac.query(t);

    return 0;
}
/*


 a
 aa
 aaa
 aaaa
 aaaaa

 */
```

4.3 Levenshtein-Distance-编辑距离.py

```python
import math

# https://www.jianshu.com/p/a617d20162cf
def Levenshtein_Distance(str1, str2):
    """
    计算字符串 str1 和 str2 的编辑距离
    :param str1
    :param str2
    :return:
    """
    matrix = [[i + j for j in range(len(str2) + 1)] for i in range(len(str1) + 1)]

    for i in range(1, len(str1) + 1):
        for j in range(1, len(str2) + 1):
            if (str1[i - 1] == str2[j - 1]):
                d = 0
            else:
                d = 1

            matrix[i][j] = min(matrix[i - 1][j] + 1, matrix[i][j - 1] + 1, matrix[i - 1][j - 1] + d)

    return matrix[len(str1)][len(str2)]

# num's bit format.
def bindigits(num, bits):
    s = bin(num & int("1"*bits, 2))[2:]
    return ("{0:0>%s}" % (bits)).format(s)

if __name__ == "__main__":


    for i in range(0,255):
        if(Levenshtein_Distance("1010", bindigits(i, 4))>2):
            print(bindigits(i, 4))
            print(Levenshtein_Distance("10101010", bindigits(i, 4)))
```

4.4 manacher-单数组马拉车.cpp

```cpp
/**
 *  @Source: https://codeforces.com/contest/1326/submission/73675730
 *  @Author: tourist
 *  @Complexity: O(n)
 *  @Description:
```

```
 *      得到经过填充长2n-1的回文半径数组，填充模式为：abc
 *      + 由于串实际没有被修改，常数喜人
 *      + 同时适配 char* 与 string，各取所爱
 *      + 你不满意可以改成全局变量数组，简简单单
 *      + 此处回文半径不含中心
 *
 *  @Example:
 *      char s[] = "123321";
 *      vint p = manacher(6, s);
 *      // [p] := {0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0}
 *
 *      string s[] = "12321";
 *      vint p = manacher(s);
 *      // [p] := {0, 0, 0, 0, 2, 0, 0, 0, 0}
 *
 *  @Verification:
 *      https://codeforces.com/contest/1326/submission/73675730
 */
template<typename T>
vector<int> manacher(int n, const T &s) {
    if (n == 0) {
        return vector<int>();
    }
    vector<int> res(2 * n - 1, 0);
    int l = -1, r = -1;
    for (int z = 0; z < 2 * n - 1; z++) {
        int i = (z + 1) >> 1;
        int j = z >> 1;
        int p = (i >= r ? 0 : min(r - i, res[2 * (l + r) - z]));
        while (j + p + 1 < n && i - p - 1 >= 0) {
            if (!(s[j + p + 1] == s[i - p - 1])) {
                break;
            }
            p++;
        }
        if (j + p > r) {
            l = i - p;
            r = j + p;
        }
        res[z] = p;
    }
    return res;
}

template<typename T>
vector<int> manacher(const T &s) {
    return manacher((int) s.size(), s);
}
```

4.5 manacher-双数组马拉车.cpp

```
/**
 *  @Source: https://codeforces.com/blog/entry/12143
 *  @Complexity: O(n)
 *  @Description: length of largest palindrome centered at each
 *      character of string and between every consecutive pair
 *      二维数组分别表示第i个位置偶数长度和奇数长度的回文半径（不含中心位置）。
 *  @Example:
 *      s = "123321"
 *      [p[0], p[1]] := {0, 0, 0, 3, 0, 0} {0, 0, 0, 0, 0, 0}
 *      s = "12321"
 *      [p[0], p[1]] := {0, 0, 0, 0, 0} {0, 0, 2, 0, 0}
 *  @Verification:
 *      https://codeforces.com/contest/1326/submission/73742092
 */
void manacher(const string &s, vector<vector<int>> &p) {
    int n = s.size();
    p.assign(2, vector<int>(n, 0));
    for (int z = 0, l = 0, r = 0; z < 2; z++, l = 0, r = 0) {
        for (int i = 0; i < n; i++) {
            if (i < r) p[z][i] = min(r - i + !z, p[z][l + r - i + !z]);
            int L = i - p[z][i], R = i + p[z][i] - !z;
            while (L - 1 >= 0 && R + 1 < n && s[L - 1] == s[R + 1])
                p[z][i]++, L--, R++;
            if (R > r) l = L, r = R;
        }
    }
}

/**
 *  @Source: https://cp-algorithms.com/string/manacher.html
 *  @Complexity: O(n)
 *  @Description: length of largest palindrome centered at each
 *      character of string and between every consecutive pair
 *      两个数组分别表示第i个位置偶数长度和奇数长度的回文半径（含中心位置）。
 *  @Example:
 *      s = "123321"
 *      [d1, d2] := {1, 1, 1, 1, 1, 1} {0, 0, 0, 3, 0, 0}
 *      s = "12321"
 *      [d1, d2] := {1, 1, 3, 1, 1} {0, 0, 0, 0, 0}
 *  @Verification:
 *      https://codeforces.com/contest/1326/submission/73715067
 */
void manacher(const string &s, vint &d1, vint &d2) {
    int n = s.size();
    d1.assign(n, 0);
    for (int i = 0, l = 0, r = -1; i < n; i++) {
```

```
        int k = (i > r) ? 1 : min(d1[l + r - i], r - i + 1);
        while (0 <= i - k && i + k < n && s[i - k] == s[i + k]) {
            k++;
        }
        d1[i] = k--;
        if (i + k > r) {
            l = i - k;
            r = i + k;
        }
    }
    d2.assign(n, 0);
    for (int i = 0, l = 0, r = -1; i < n; i++) {
        int k = (i > r) ? 0 : min(d2[l + r - i + 1], r - i + 1);
        while (0 <= i - k - 1 && i + k < n && s[i - k - 1] == s[i + k]) {
            k++;
        }
        d2[i] = k--;
        if (i + k > r) {
            l = i - k - 1;
            r = i + k;
        }
    }
}
```

Chapter 5

数学

5.1 double-compare.cpp

```cpp
/* @head of double-compare modules */
const double EPS = 1e-8;

inline int dcmp(const double &x) {
    if (fabs(x) < EPS)return 0;
    else return x < EPS ? -1 : 1;
}

// not necessary
inline bool lt(const double &x, const double &y) { return dcmp(x - y) < 0; }

inline bool le(const double &x, const double &y) { return dcmp(x - y) <= 0; }

inline bool eq(const double &x, const double &y) { return dcmp(x - y) == 0; }

inline bool ge(const double &x, const double &y) { return dcmp(x - y) >= 0; }

inline bool gt(const double &x, const double &y) { return dcmp(x - y) > 0; }

// not recommended
inline bool dcmp(const double &x, const string &mode, const double &y) {
    if (mode == "lt") return dcmp(x - y) < 0;
    if (mode == "le") return dcmp(x - y) <= 0;
    if (mode == "eq") return dcmp(x - y) == 0;
    if (mode == "ge") return dcmp(x - y) >= 0;
    if (mode == "gt") return dcmp(x - y) > 0;
    exit(0);
}
/* @tail of double-compare modules */
```

5.2 fastFacterial-快速阶乘-分块fft.cpp

```cpp
// fastFacterial 快速阶乘(分块+fft)
// O( sqrt(n)log(n) )
// https://www.luogu.org/record/25477473
#include<cstdio>
#include<algorithm>
#include<cmath>

using namespace std;
typedef unsigned long long ll;
const ll N = 262144 + 10;
const int P = 65536;
const int SF = 16;
const int msk = 65535;
ll mod;
ll PP;
typedef long double ld;
const ld pi = acos(-1.0);

inline ll fpow(ll a, ll p) {
    ll r = 1;
    for (; p; p >>= 1, a = a * a % mod)
        if (p & 1)r = r * a % mod;
    return r;
}

struct cmp {
    ld r;
    ld v;

    friend cmp operator+(cmp a, cmp b) {
        return (cmp) {a.r + b.r, a.v + b.v};
    }
```

```
    friend cmp operator-(cmp a, cmp b) {
        return (cmp) {a.r - b.r, a.v - b.v};
    }

    friend cmp operator*(cmp a, cmp b) {
        return (cmp) {a.r * b.r - a.v * b.v,
                      a.r * b.v + a.v * b.r};
    }

    void operator/=(const int &len) {
        r /= len;
        v /= len;
    }
} rt[2][22][N], tr[N],
        tr1[N], tr2[N], tr3[N],
        tr4[N], tr5[N], tr6[N];

int rv[22][N];
ll m13[N], m14[N], m23[N], m24[N];

inline void pre() {
    for (int d = 1; d <= 18; d++)
        for (int i = 1; i < (1 << d); i++)
            rv[d][i] = (rv[d][i >> 1] >> 1)
                       | ((i & 1) << (d - 1));

    for (int d = 1, t = 1; d <= 18; d++, t <<= 1)
        for (int i = 0; i < (1 << d); i++)
            rt[0][d][i] = (cmp) {cos(pi * i / t),
                                 sin(pi * i / t)};

    for (int d = 1, t = 1; d <= 18; d++, t <<= 1)
        for (int i = 0; i < (1 << d); i++)
            rt[1][d][i] = (cmp) {cos(pi * i / t),
                                 -sin(pi * i / t)};
}

inline void fft(cmp *a, int len, int d, int o) {
    for (int i = 1; i < len; i++)
        if (i < rv[d][i])
            swap(a[i], a[rv[d][i]]);
    cmp *w;
    int i;
    for (int k = 1, j = 1; k < len; k <<= 1, j++)
        for (int s = 0; s < len; s += (k << 1))
            for (i = s, w = rt[o][j]; i < s + k; i++, ++w) {
                cmp a1 = a[i + k] * (*w);
                a[i + k] = a[i] - a1;
                a[i] = a[i] + a1;
            }
    if (o)for (int i = 0; i < len; i++)a[i] /= len;
}

inline void dbdft(ll *a, int len, int d, cmp *op1, cmp *op2) {
    for (int i = 0; i < len; i++)
        tr[i] = (cmp) {(ld) (a[i] >> SF),
                       (ld) (a[i] & msk)};

    fft(tr, len, d, 0);
    tr[len] = tr[0];

    for (cmp *p1 = tr, *p2 = tr + len, *p3 = op1;
         p1 != tr + len; ++p1, --p2, ++p3)
        (*p3) = (cmp) {p1->r + p2->r,
                       p1->v - p2->v}
                * (cmp) {0.5, 0};

    for (cmp *p1 = tr, *p2 = tr + len, *p3 = op2;
         p1 != tr + len; ++p1, --p2, ++p3)
        (*p3) = (cmp) {p1->r - p2->r,
                       p1->v + p2->v}
                * (cmp) {0, -0.5};
}

inline void dbidft(cmp *tr, int len, int d, ll *a, ll *b) {
    fft(tr, len, d, 1);
    for (int i = 0; i < len; i++)
        a[i] = (ll) (tr[i].r + 0.5) % mod;

    for (int i = 0; i < len; i++)
        b[i] = (ll) (tr[i].v + 0.5) % mod;
}

inline void poly_mul(ll *a, ll *b, ll *c, int len, int d)//以上都是任意模数fft的板子
{
    dbdft(a, len, d, tr1, tr2);
    dbdft(b, len, d, tr3, tr4);
    for (int i = 0; i < len; i++)
        tr5[i] = tr1[i] * tr3[i]
                 + (cmp) {0, 1}
                   * tr2[i] * tr4[i];
    for (int i = 0; i < len; i++)
        tr6[i] = tr2[i] * tr3[i]
                 + (cmp) {0, 1}
                   * tr1[i] * tr4[i];
```

```
    dbidft(tr5, len, d, m13, m24);
    dbidft(tr6, len, d, m23, m14);

    for (int i = 0; i < len; i++)
        c[i] = m13[i] * PP % mod;
    for (int i = 0; i < len; i++)
        (c[i] += (m23[i] + m14[i]) * P + m24[i]) %= mod;
}

namespace iter {
    ll f[N];
    ll g[N];
    ll h[N];
    ll ifac[N];

    inline void ih() {
        ifac[0] = ifac[1] = 1;
        for (ll i = 2; i < min(N, mod); i++)
            ifac[i] = (mod - mod / i) * ifac[mod % i] % mod;
        for (ll i = 1; i < min(N, mod); i++)
            (ifac[i] *= ifac[i - 1]) %= mod;
    }

    inline void calch(ll del, int cur, ll *ip, ll *op) {
        int d = 0;
        int len = 1;
        while (len <= cur + cur + cur)len <<= 1, d++;
        for (int i = 0; i <= cur; i++)
            f[i] = ip[i] * ifac[i] % mod * ifac[cur - i] % mod;
        for (int i = cur - 1; i >= 0; i -= 2)
            f[i] = (mod - f[i]) % mod;
        for (int i = 0; i <= cur + cur; i++)
            g[i] = fpow((del + mod - cur + i) % mod, mod - 2);
        for (int i = cur + 1; i < len; i++)
            f[i] = 0;
        for (int i = cur + cur + 1; i < len; i++)
            g[i] = 0;

        poly_mul(f, g, h, len, d);//卷积求出h'
        ll xs = 1;
        ll p1 = del - cur;
        ll p2 = del;
        for (ll i = p1; i <= p2; i++)(xs *= i) %= mod;
        for (ll i = 0; i <= cur; i++, p1++, p2++)//双指针求出系数
        {
            op[i] = h[i + cur] * xs % mod;
            (xs *= fpow(p1, mod - 2)) %= mod,
                    (xs *= (p2 + 1)) %= mod;
        }
    }
}
ll val[N];
ll fv1[N];
ll fv2[N];

inline void solve(int n)//倍增
{
    int hb = 0;
    for (int p = n; p; p >>= 1)hb++;
    val[0] = 1;
    for (int z = hb, cur = 0; z >= 0; z--) {
        if (cur != 0)//把d乘2
        {
            iter::calch(cur + 1, cur, val, fv1);
            for (int i = 0; i <= cur; i++)
                val[cur + i + 1] = fv1[i];

            val[cur << 1 | 1] = 0;
            iter::calch(cur * fpow(n, mod - 2) % mod,
                        cur << 1, val, fv2);
            cur <<= 1;
            for (int i = 0; i <= cur; i++)
                (val[i] *= fv2[i]) %= mod;
        }
        if ((n >> z) & 1)//把d加1
        {
            for (int i = 0; i <= cur; i++)
                (val[i] *= (ll) (n * i) + cur + 1) %= mod;
            cur |= 1;
            val[cur] = 1;
            for (int i = 1; i <= cur; i++)
                (val[cur] *= (ll) cur * n + i) %= mod;
        }
    }
}

int kase;

int main() {
    pre();
    int n;
    scanf("%d", &kase);
    while (kase--) {
        scanf("%d%lld", &n, &mod);
        iter::ih();//用了全局变量mod
```

```
        int bl = sqrt(n);
        PP = (ll) P * P % mod;
        solve(bl);
        ll res = 1;
        for (ll i = 0, id = 0;; i += bl, id++)//分块
        {
            if (i + bl > n) {
                for (int j = i + 1; j <= n; j++)
                    (res *= j) %= mod;
                break;
            }
            (res *= val[id]) %= mod;
        }
        printf("%lld\n", res);
    }
    return 0;//拜拜程序~
}
/*

3
16777216 998244353
2333333 19260817
1919810 2147481811

 "n and mod"
 */
```

5.3 fft-多项式乘法.cpp

```
// 多项式乘法
// http://acm.hdu.edu.cn/showproblem.php?pid=1402
// https://www.luogu.com.cn/problem/P1919
// 来源: https://oi-wiki.org/math/poly/fft/
#include <math.h>
#include <algorithm>
#include <stdio.h>
#include <string.h>
#include <iostream>
#include <vector>

using namespace std;
const double PI = acos(-1.0);

struct Complex {
    double x, y;

    Complex(double _x = 0.0, double _y = 0.0) {
        x = _x;
        y = _y;
    }

    Complex operator-(const Complex &b) const {
        return Complex(x - b.x, y - b.y);
    }

    Complex operator+(const Complex &b) const {
        return Complex(x + b.x, y + b.y);
    }

    Complex operator*(const Complex &b) const {
        return Complex(x * b.x - y * b.y, x * b.y + y * b.x);
    }
};

/*
 * 进行 FFT 和 IFFT 前的反置变换
 * 位置 i 和 i 的二进制反转后的位置互换
 *len 必须为 2 的幂
 */
void fftChange(Complex *y, int len) {
    for (int i = 1, j = (len >> 1); i < len - 1; i++) {
        if (i < j) swap(y[i], y[j]);
        int k = len >> 1;
        while (j >= k) {
            j = j - k;
            k = k >> 1;
        }
        if (j < k) j += k;
    }
}

/*
 *  做 FFT
 *  len 必须是 2^k 形式
 *  dir == 1 时是 DFT, dir == -1 时是 IDFT
 */
void fft(Complex y[], int len, int dir) {
    fftChange(y, len);
    for (int h = 2; h <= len; h <<= 1) {
        Complex wn(cos(2.0 * PI / h), sin(dir * 2.0 * PI / h));
        // Omega ^ n
        for (int j = 0; j < len; j += h) {
            Complex w(1, 0);
            for (int k = j; k < j + h / 2; k++) {
                Complex u = y[k];
```

```
                Complex t = w * y[k + h / 2];
                y[k] = u + t;
                y[k + h / 2] = u - t;
                w = w * wn;
            }
        }
    }

    if (dir == -1) {
        for (int i = 0; i < len; i++) {
            y[i].x /= len;
        }
    }
}

inline int fftLength(int len1, int len2) {
    int len = 1;
    while (len < len1 * 2 || len < len2 * 2) {
        len <<= 1;
    }
    return len;
}

// rewrite
void fftAssign(Complex *x, int len, char *s, int slen) {
    for (int i = 0; i < slen; i++) {
        double v = s[slen - 1 - i] - '0';
        x[i] = Complex(v, 0.0);
    }
    for (int i = slen; i < len; i++) {
        x[i] = Complex(0.0, 0.0);
    }
}

// rewrite
void fftMul(Complex x1[], Complex x2[], int len) {
    for (int i = 0; i < len; i++)
        x1[i] = x1[i] * x2[i];
}

const int MAXN = 2.1e6 + 59; // 尽量去取到2的幂。
Complex x2[MAXN];
Complex x1[MAXN];
char str1[MAXN / 2];
char str2[MAXN / 2];
int sum[MAXN];

int main() {
    scanf("%s%s", str1, str2);
    int len1 = strlen(str1);
    int len2 = strlen(str2);
    int len = fftLength(len1, len2);

    fftAssign(x1, len, str1, len1);
    fftAssign(x2, len, str2, len2);
    fft(x1, len, 1);
    fft(x2, len, 1);
    fftMul(x1, x2, len);
    fft(x1, len, -1);

    for (int i = 0; i < len; i++) {
        sum[i] = int(x1[i].x + 0.5);
    }

    for (int i = 0; i < len; i++) {
        sum[i + 1] += sum[i] / 10;
        sum[i] %= 10;
    }

    len = len1 + len2 - 1;
    while (sum[len] == 0 && len > 0)
        len--;

    for (int i = len; i >= 0; i--) {
        printf("%c", sum[i] + '0');
    }

    printf("\n");

    return 0;
}
```

5.4 扩展CRT.py

```
# https://ac.nowcoder.com/acm/contest/890/D
# maybe not available.


ai = [0]
bi = [0]


def exgcd(a, warrior, x, y):
    if warrior == 0:
        x = 1
```

```
        y = 0
        return a, x, y
    gcd, x, y = exgcd(warrior, a % warrior, x, y)
    tp = x
    x = y
    y = tp - (a // warrior) * y
    return gcd, x, y


def excrt():
    m = bi[1]
    ans = ai[1]
    for i in range(2, num + 1):
        x = 0
        y = 0
        aa = m
        bb = bi[i]
        c = (ai[i] - ans % bb + bb) % bb
        gcd, x, y = exgcd(aa, bb, x, y)
        bg = bb // gcd
        if c % gcd != 0:
            return -1

        x = x * (c // gcd) % bg
        ans = ans + x * m
        m = m * bg
        ans = (ans % m + m) % m
    return (ans % m + m) % m


def main():
    global num
    num, m = map(int, input().split())
    # num, m = int(input())
    for i in range(1, num + 1):
        ub, ua = map(int, input().split())
        bi.append(ub)
        ai.append(ua)
    ans = excrt()
    if ans == -1:
        print("he was definitely lying")
    else:
        if ans <= m:
            print(ans)
        else:
            print("he was probably lying")


if __name__ == '__main__':
    main()
```

5.5 矩阵快速幂+大十进制指数版.cpp

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

#include <bits/stdc++.h>

using namespace
std;
typedef long long ll;


template<
typename _Tp,
const int MAXMatrixSize
>

struct Matrix {
    _Tp m[MAXMatrixSize][MAXMatrixSize];
    _Tp mod = 0;

    Matrix() {
        memset(m, 0, sizeof m);
    }

    Matrix(int _mod) : mod(_mod) {
        memset(m, 0, sizeof m);
    }

    void init1() {
        //*this = Matrix(mod);
        set(0, 0, 1);
        set(1, 1, 1);
//        for (int i = 0; i < MAXMatrixSize; i++)
//            m[i][i] = 1;
    }

    inline void set(const int

    &r, const int &c, const _Tp &v) { this->m[r][c] = v; }

    inline _Tp get(const int

    &r, const int &c) { return this->m[r][c]; }

```

```
    inline void setMod(const _Tp

    &_mod) { this->mod = _mod; }

    inline Matrix operator
    *(
    const Matrix t
    ) {
        Matrix res(mod);//= Matrix(mod);
        res.setMod(mod);
        for (int i = 0; i < MAXMatrixSize; i++)
            for (int j = 0; j < MAXMatrixSize; j++)
                for (int k = 0; k < MAXMatrixSize; k++)
                    res.m[i][j] = (res.m[i][j] + m[i][k] * t.m[k][j]) % mod;
        return res;
    }
};

typedef Matrix<ll, 2> mat;

mat A, B;
ll mo, len;
char n[1000059];

inline mat fpow(mat base, ll exp) {
    mat res(mo);
    res.init1();
    while (exp) {
        if (exp & 1)res = res * base;
        exp >>= 1;
        base = base * base;
    }
    return res;
}

inline ll calc() {

    len = strlen(n);
    //reverse(n, n + len);

    mat res(mo);
    res.init1();
    mat base = B;

    for (int i = len - 1; i >= 0; --i) {
        if (n[i] > '0')
            res = res * fpow(base, n[i] - '0');
        base = fpow(base, 10);
    }

    res = A * res;
    return res.get(0, 0);
}

//https://ac.nowcoder.com/acm/contest/885/B
/*
 * input n is a long char string.(1e6)
 * mo is global Mod.
 * other parameters are just Matrix elements.
 *
 *
 */
ll x0, x1, a, b;

int main() {

    scanf("%lld%lld%lld%lld", &x0, &x1, &a, &b);
    scanf("%s %lld", n, &mo);

    A = mat(mo);
    A.set(0, 0, x0);
    A.set(0, 1, x1);

    B = mat(mo);
    B.set(0, 0, 0);
    B.set(0, 1, b);
    B.set(1, 0, 1);
    B.set(1, 1, a);

    printf("%lld\n", calc());
    return 0;
}
/*




 * */
```

Chapter 6

数据结构

6.1 zhuxishu-SegKth.cpp

```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


*/

/*




 */
```

6.2 ZTC's-Splay.cpp

```
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] ? node[node[x].ch[1]].siz : 0) + node[x].cnt;
}
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,
                   node[x].siz,
                   node[x].cnt,
                   node[x].ch[0],
                   node[x].ch[1],
                   node[x].par);
    }
}
/*
5 100
1 3 5  7 9
1 2
1 2
2 1
2 3
2 3
*/
```

Chapter 7

数论

7.1 Binomial-Coefficients-组合数-Lucas.cpp

```cpp
const int MAXN = 1e6 + 59;
const int MOD = 1e9 + 7;
ll fac[MAXN];
ll inv[MAXN];

inline void initC(const int &sz) {
    fac[0] = 1;
    for (int i = 1; i <= sz; i++)
        fac[i] = fac[i - 1] * i % MOD;
    inv[sz] = fpow<ll>(fac[sz], MOD - 2, MOD);
    // printf inv[BCSize] to get & save it;
    for (int i = sz - 1; ~i; i--)
        inv[i] = inv[i + 1] * (i + 1) % MOD;
}

inline ll C(const int &n, const int &m) {
    return fac[n] * inv[m] % MOD * inv[n - m] % MOD;
}

// Lucas
inline ll C(int n, int m, const int &P) {
    ll res = 1;
    while (n | m)res = res * C(n % P, m % P) % P, n /= P, m /= P;
    return res;
}

int main() {
    initC(100000);

    cout << C(4, 3, 1000000007) << endl;
    cout << C(4, 1, 1000000007) << endl;
    cout << C(5, 2, 1000000007) << endl;
    return 0;
}
```

7.2 Binomial-Coefficients-组合数-Lucas.cpp.ignore

7.3 Binomial-Coefficients-组合数-杨辉三角.cpp

```cpp
/*
// O(N^2)
// __int128
template<const int BCSize = 120, typename var = __int128>
//add Mod as parameter;
struct Binomial_Coefficient {
    var c[BCSize + 1][BCSize + 1];
    //Pascal's Triangle

    Binomial_Coefficient() {    //add Mod as parameter;
        c[0][0] = 1;
        for (int n = 1; n <= BCSize; ++n) {
            c[n][0] = c[n][n] = 1;
            for (int k = 1; k < n; ++k)
                c[n][k] = (c[n - 1][k - 1] + c[n - 1][k]);//%
        }
    }

    var operator()(const int &n, const int &m) {
        if (n < m)return -1;//in case.
        return c[n][m];
    }
};
Binomial_Coefficient<> C;
```

```

*/
//********in normal writing style***********

const int MAXN = 20;
ll C[MAXN + 1][MAXN + 1];

inline void pascal(const int &maxn) {
    C[0][0] = 1;
    for (int n = 1; n <= maxn; ++n) {
        C[n][0] = C[n][n] = 1;
        for (int k = 1; k < n; ++k)
            C[n][k] = C[n - 1][k - 1] + C[n - 1][k];
    }
}

int main() {
    /*
    cout << C(4, 3) << endl;
    cout << C(4, 1) << endl;
    cout << C(5, 2) << endl;
    */
    cout << C[4][3] << endl;
    cout << C[4][1] << endl;
    cout << C[5][2] << endl;

    return 0;
}
```

7.4 Binomial-Coefficients-组合数-逆元-模大素数.cpp

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

#include <bits/stdc++.h>

using namespace std;
typedef long long ll;

template<typename _Tp>
_Tp fpow(_Tp base, _Tp exp, _Tp Mod) {
    _Tp res = 1;
    while (exp) {
        if (exp & 1)res = res * base % Mod;
        base = base * base % Mod;
        exp >>= 1;
    }
    return res;
}
/*

// O(N) O(1)
template<typename _Tp, const int BCSize, const _Tp Mod> //add Mod as parameter;
struct Binomial_Coefficient {
    _Tp fac[BCSize + 1];
    _Tp inv[BCSize + 1];

    inline Binomial_Coefficient() {    //add Mod as parameter;
        fac[0] = 1;
        for (int i = 1; i <= BCSize; i++)
            fac[i] = fac[i - 1] * i % Mod;

        inv[BCSize] = fpow<_Tp>(fac[BCSize], Mod - 2, Mod);
        // printf inv[BCSize] to get & save it;

        for (int i = BCSize - 1; ~i; i--)
            inv[i] = inv[i + 1] * (i + 1) % Mod;
    }

    inline _Tp operator()(const int &n, const int &m) {
        if (n < m) {
            cerr << "**** n>m " << endl;
            return -1;
        }//in case.
        return fac[n] * inv[m] % Mod * inv[n - m] % Mod;
    }
};

typedef Binomial_Coefficient<long long, 10000000, 1000000007> zuHeShu;
zuHeShu C = zuHeShu();

*/

//********in normal writing style***********

const int MAXN = 1e6 + 59;
const int MOD = 1e9 + 7;
ll fac[MAXN];
ll inv[MAXN];

inline void initC(const int &sz) {
    fac[0] = 1;
    for (int i = 1; i <= sz; i++)
        fac[i] = fac[i - 1] * i % MOD;
    inv[sz] = fpow<ll>(fac[sz], MOD - 2, MOD);
    // printf inv[BCSize] to get & save it;
```

```
    for (int i = sz - 1; ~i; i--)
        inv[i] = inv[i + 1] * (i + 1) % MOD;
}

inline ll C(const int &n, const int &m) {
    return fac[n] * inv[m] % MOD * inv[n - m] % MOD;
}

int main() {

    initC(100000);

    cout << C(4, 3) << endl;
    cout << C(4, 1) << endl;
    //cout << C(2, 5) << endl;
    cout << C(5, 2) << endl;


    return 0;
}
/*




 * */
```

7.5 Extended-Euclidean-algorithm-(exGCD).cpp

```
ll exGCD(ll a, ll b, ll &x, ll &y) {
    if (b == 0) {
        x = 1;
        y = 0;
        return a;
    }
    ll gcd = exGCD(b, a % b, x, y);
    ll old_x = x;
    x = y;
    y = old_x - (a / b) * x;
    return gcd;
}
// co-prime(a,m)
ll modInv(ll a, ll m) {
    ll x, y;
    ll g = exGCD(a, m, x, y);
    if (g != 1) {
        return -1;
    } else {
        ll res = (x % m + m) % m;
        return res;
    }
}
```

7.6 ZTC's-FFT.txt

```
struct CP
{
    double x,y;
    CP (double xx=0,double yy=0){x=xx;y=yy;}
    CP operator +(const CP &warrior){return CP(x+warrior.x,y+warrior.y);}
    CP operator -(const CP &warrior){return CP(x-warrior.x,y-warrior.y);}
    CP operator *(const CP &warrior){return
    ↪  CP(x*warrior.x-y*warrior.y,x*warrior.y+y*warrior.x);}
    void print(){printf("CP.x: %f  CP.y: %f \num",x,y);}
}a[MAXN],warrior[MAXN];
int lim,bit;
int rev[MAXN];
void init_FFT(int len)
{
    lim=1,bit=0;
    while(lim<=(len))lim<<=1,bit++;
    for(int i=0;i<lim;i++)rev[i]=(rev[i>>1]>>1)|((i&1)<<(bit-1));
}
void FFT(CP *A,int mode)
{
    for(int i=0;i<lim;i++)
    {
        if(i<rev[i])swap(A[i],A[rev[i]]);
    }
    for(int mid=1;mid<lim;mid<<=1)
    {
        CP XX(cos(Pi/mid),mode*sin(Pi/mid));
        for(int j=0;j<lim;j+=(mid<<1))
        {
            CP d(1,0);
            for(int k=0;k<mid;k++,d=d*XX)
            {
                CP x=A[j+k],y=d*A[j+mid+k];
                A[j+k]=x+y;
                A[j+mid+k]=x-y;
            }
        }
    }
}
```

7.7 数论分块.cpp

```cpp
/**
 *  @Source: none.
 *  @Complexity: sqrt(n)
 *  @Description: 枚举从1开始到x的所有分块区间。
 *  @Example: none.
 *  @Verification:
 *      https://codeforces.ml/gym/101174/C
 */

for (ll l = 1, r = 0; l <= x; l = r + 1) {
    r = x / (x / l);
    // ...
}
```

Chapter 8

杂项

8.1 coutf.cpp

```cpp
/**
 *  @Source: https://zh.cppreference.com/w/cpp/language/parameter_pack
 *  @Complexity:
 *  @Description: 用cout模仿格式化输出
 *  @Example: see below
 *  @Verification: TODO
 */
void coutf(const char *format) {
    std::cout << format;
}

template<typename T, typename... Targs>
//void coutf(const char *format, T value, Targs... Fargs) // 递归变参函数
void coutf(const char *format, const T &value, const Targs &... Fargs) {
    for (; *format != '\0'; format++) {
        if (*format == '%') {
            std::cout << value;
            coutf(format + 1, Fargs...); // 递归调用
            return;
        }
        std::cout << *format;
    }
}

/*
void example(){
    coutf("% world% %\n", "Hello", '!', 123);
    cout.precision(9);
    fixed(cout);
    coutf("% % % %\n", 0x3f, 1.2 / 7, acos(-1), 22.3);
}
*/
```

8.2 debug-from-tourist.cpp

```cpp
/**
 *    ▄▄▄▄ ▄▄▄▄▄    ▄▄▄▄ ▄▄▄▄▄▄
 *   ▄▀  ▀▄▀  ▀▄  ▄▀  ▀▄▀  ▀▄
 *   ▀▄▄▄▄ ▀▄▄▄▄▀ ▀▄▄▄▄ ▀▄▄▄▄▀
 *
 *  @Author: TieWay59
 *  @Created: 2019/11/22 21:39
 *  @Link: https://atcoder.jp/contests/agc040/submissions/8558491
 *  @Tags:
 *
 ******************************************************/


#include <bits/stdc++.h>

//#define  debug(x)  cerr <<#x << " = "<<x<<endl

#define endl '\n'
#define STOPSYNC ios::sync_with_stdio(false);cin.tie(nullptr)
#define MULTIKASE int Kase=0;cin>>Kase;for(int kase=1;kase<=Kase;kase++)
typedef long long ll;
const int MAXN = 2e5 + 59;
const int MOD = 1e9 + 7;
const int INF = 0x3F3F3F3F;
const ll llINF = 0x3F3F3F3F3F3F3F3F;
```

```cpp
using namespace std;


// debug start
template<typename A, typename B>
string to_string(pair<A, B> p);

template<typename A, typename B, typename C>
string to_string(tuple<A, B, C> p);

template<typename A, typename B, typename C, typename D>
string to_string(tuple<A, B, C, D> p);

string to_string(const string &s) {
    return '"' + s + '"';
}

string to_string(const char *s) {
    return to_string((string) s);
}

string to_string(bool b) {
    return (b ? "true" : "false");
}

string to_string(vector<bool> v) {
    bool first = true;
    string res = "{";
    for (int i = 0; i < static_cast<int>(v.size()); i++) {
        if (!first) {
            res += ", ";
        }
        first = false;
        res += to_string(v[i]);
    }
    res += "}";
    return res;
}

template<size_t N>
string to_string(bitset<N> v) {
    string res = "";
    for (size_t i = 0; i < N; i++) {
        res += static_cast<char>('0' + v[i]);
    }
    return res;
}

template<typename A>
string to_string(A v) {
    bool first = true;
    string res = "{";
    for (const auto &x : v) {
        if (!first) {
            res += ", ";
        }
        first = false;
        res += to_string(x);
    }
    res += "}";
    return res;
}

template<typename A, typename B>
string to_string(pair<A, B> p) {
    return "(" + to_string(p.first) + ", " + to_string(p.second) + ")";
}

template<typename A, typename B, typename C>
string to_string(tuple<A, B, C> p) {
    return "(" + to_string(get<0>(p)) + ", " + to_string(get<1>(p)) + ", " +
    ↪  to_string(get<2>(p)) + ")";
}

template<typename A, typename B, typename C, typename D>
string to_string(tuple<A, B, C, D> p) {
    return "(" + to_string(get<0>(p)) + ", " + to_string(get<1>(p)) + ", " +
    ↪  to_string(get<2>(p)) + ", " +
           to_string(get<3>(p)) + ")";
}

void debug_out() { cerr << endl; }

template<typename Head, typename... Tail>
void debug_out(Head H, Tail... T) {
    cerr << " " << to_string(H);
    debug_out(T...);
}

#ifdef DEBUG
#define debug(...) cerr << "[" << #__VA_ARGS__ << "] :=", debug_out(__VA_ARGS__)
#else
#define debug(...) 42
#endif


```

```

// debug end;

int main() {

    int x = 10;
    pair<int, bool> y = {11, 1};
    vector<int> z = {1, 2, 3, 4};

    debug(x, y, z);
    set<int> a = {9, 10, 7};
    debug(a);
    return 0;
}
/*

 */
```

8.3 fast-IO-int.cpp

```
inline void read(int &x) {
    char ch;
    bool flag = false;
    for (ch = getchar(); !isdigit(ch); ch = getchar())if (ch == '-') flag = true;
    for (x = 0; isdigit(ch); x = x * 10 + ch - '0', ch = getchar());
    x = flag ? -x : x;
}

inline void write(int x) {
    static const int maxlen = 100;
    static char s[maxlen];
    if (x < 0) {
        putchar('-');
        x = -x;
    }
    if (!x) {
        putchar('0');
        return;
    }
    int len = 0;
    for (; x; x /= 10) s[len++] = x % 10 + '0';
    for (int i = len - 1; i >= 0; --i) putchar(s[i]);
}


namespace Fast_IO { //orz laofu
    const int MAXL((1 << 18) + 1);
    int iof, iotp;
    char ioif[MAXL], *ioiS, *ioiT, ioof[MAXL], *iooS = ioof, *iooT = ioof + MAXL - 1, ioc, iost[55];

    char Getchar() {
        if (ioiS == ioiT) {
            ioiS = ioif;
            ioiT = ioiS + fread(ioif, 1, MAXL, stdin);
            return (ioiS == ioiT ? EOF : *ioiS++);
        } else return (*ioiS++);
    }

    void Write() {
        fwrite(ioof, 1, iooS - ioof, stdout);
        iooS = ioof;
    }

    void Putchar(char x) {
        *iooS++ = x;
        if (iooS == iooT)Write();
    }

    inline int read() {
        int x = 0;
        for (iof = 1, ioc = Getchar(); (ioc < '0' || ioc > '9') && ioc != EOF;)
            iof = ioc == '-' ? -1 : 1, ioc = Getchar();
        if (ioc == EOF)Write(), exit(0);
        for (x = 0; ioc <= '9' && ioc >= '0'; ioc = Getchar())x = (x << 3) + (x << 1) + (ioc ^ 48);
        return x * iof;
    }

    inline long long read_ll() {
        long long x = 0;
        for (iof = 1, ioc = Getchar(); (ioc < '0' || ioc > '9') && ioc != EOF;)
            iof = ioc == '-' ? -1 : 1, ioc = Getchar();
        if (ioc == EOF)Write(), exit(0);
        for (x = 0; ioc <= '9' && ioc >= '0'; ioc = Getchar())x = (x << 3) + (x << 1) + (ioc ^ 48);
        return x * iof;
    }

    void Getstr(char *s, int &l) {
        for (ioc = Getchar(); ioc == ' ' || ioc == '\n' || ioc == '\t';)ioc = Getchar();
        if (ioc == EOF)Write(), exit(0);
        for (l = 0; !(ioc == ' ' || ioc == '\n' || ioc == '\t' || ioc == EOF); ioc = Getchar())s[l++] = ioc;
```

```
        s[l] = 0;
    }

    template<class Int>
    void Print(Int x, char ch = '\0') {
        if (!x)Putchar('0');
        if (x < 0)Putchar('-'), x = -x;
        while (x)iost[++iotp] = x % 10 + '0', x /= 10;
        while (iotp)Putchar(iost[iotp--]);
        if (ch)Putchar(ch);
    }

    void Putstr(const char *s) { for (int i = 0, n = strlen(s); i < n;
    ↪  ++i)Putchar(s[i]); }
} // namespace Fast_IO
using namespace Fast_IO;
```

8.4 fast-IO-快速版（可敲）.cpp

```
//没实现负数
const int BUF_SIZE = (int) 1e4 + 10;

struct fastIO {
    char buf[BUF_SIZE];
    int cur;
    FILE *in, *out;

    fastIO() {
        cur = BUF_SIZE;
        in = stdin;
        out = stdout;
    }

    inline char nC() {
        if (cur == BUF_SIZE) {
            fread(buf, BUF_SIZE, 1, in);
            cur = 0;
        }
        return buf[cur++];
    }

    inline bool id(char a) { return a >= '0' && a <= '9'; }

    inline int nI() {
        char c;
        while (!id(c = nC()));
        int x = c - '0';
        while (id(c = nC())) x = ((x + (x << 2)) << 1) + c - '0';
        return x;
    }

    inline ll nll() {
        char c;
        while (!id(c = nC()));
        ll x = c - '0';
        while (id(c = nC())) x = ((x + (x << 2)) << 1) + c - '0';
        return x;
    }

    inline void pC(char ch) {
        buf[cur++] = ch;
        if (cur == BUF_SIZE) {
            fwrite(buf, BUF_SIZE, 1, out);
            cur = 0;
        }
    }

    inline void pI(int x) {
        if (x > 9) pI(x / 10);
        pC(x % 10 + '0');
    }

    inline void close() { if (cur) fwrite(buf, cur, 1, out), cur = 0; }
} IO;
```

8.5 fast-IO-极致版.ignore

8.6 fastpow-快速幂.cpp

```
ll fpow(ll a, ll b, ll mod = MOD) {
    if (a % mod == 0) return 0;
    ll ret = 1;
    a %= mod;
    while (b) {
        if (b & 1)ret = ret * a % mod;
        a = a * a % mod;
        b >>= 1;
    }
    return ret;
}
```

8.7 Misc-杂技-随机数.cpp

```cpp
mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());

inline int suiJi(const int &l, const int &r) {
return uniform_int_distribution<int>(l, r)(rng);
}
```

8.8 string-read-speed.cpp

```cpp
const int MAXN = 5e7 + 59;
char buffer[MAXN];
vector<char> buf(MAXN);
string s;
stringstream ss;

void solve(int kaseId = -1) {
/*
    freopen("text.in", "w+", stdout);

    for (int i = 1; i <= 50000000; i++) {
        cout << (char) suiJi('a', 'z');
    }
*/
/*
    // 278.912500 ms
    freopen("text.in", "r+", stdin);
    double begin = chrono::steady_clock::now().time_since_epoch().count();
    getline(cin, s);
    double stoped = chrono::steady_clock::now().time_since_epoch().count();
    printf("%.6f ms\n", (double) (stoped - begin) / 1000000.0);
*/
/*
    // 290.965400 ms ~ 300
    freopen("text.in", "r+", stdin);
    double begin = chrono::steady_clock::now().time_since_epoch().count();
    cin >> s;
    double stoped = chrono::steady_clock::now().time_since_epoch().count();
    printf("%.6f ms\n", (double) (stoped - begin) / 1000000.0);
*/
/*
    // 235.966700 ms
    freopen("text.in", "r+", stdin);
    double begin = chrono::steady_clock::now().time_since_epoch().count();
    gets(s);
    double stoped = chrono::steady_clock::now().time_since_epoch().count();
    printf("%.6f ms\n", (double) (stoped - begin) / 1000000.0);
*/
/*
    // 99.795400 ms
    freopen("text.in", "r+", stdin);
    //FILE *fp = fopen("text.in", "r");
    double begin = chrono::steady_clock::now().time_since_epoch().count();
    fread(s, sizeof(char), 50000000, stdin);
    double stoped = chrono::steady_clock::now().time_since_epoch().count();
    printf("%.6f ms\n", (double) (stoped - begin) / 1000000.0);
*/
/*
    // 1749.292200 ms
    freopen("text.in", "r+", stdin);
    //FILE *fp = fopen("text.in", "r");
    double begin = chrono::steady_clock::now().time_since_epoch().count();
    scanf("%s", buffer);
    double stoped = chrono::steady_clock::now().time_since_epoch().count();
    printf("%.6f ms\n", (double) (stoped - begin) / 1000000.0);
*/
/*
    // 90.939200 ms
    freopen("text.in", "r+", stdin);
    double begin = chrono::steady_clock::now().time_since_epoch().count();
    fread(buf.data(), sizeof(char), 50000000, stdin);
    double stoped = chrono::steady_clock::now().time_since_epoch().count();
    printf("%.6f ms\n", (double) (stoped - begin) / 1000000.0);
*/

/*
    cin.get() 与 getchar 读法效率差很多，在此不表。
*/
}
```

8.9 timetest.cpp

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

using namespace std;

int main() {

    clock_t begin = clock();
    int x = 0;
    for (int i = 1; i <= 800000000; ++i) {
        x++;
    }
```

```
    printf("%.3f ms\n", (double) (clock() - begin));
    return 0;
}
```

8.10 unordered-map-自写哈希.cpp

```
struct custom_hash {
    static uint64_t splitmix64(uint64_t x) {
        // http://xorshift.di.unimi.it/splitmix64.c
        x += 0x9e3779b97f4a7c15;
        x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
        x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
        return x ^ (x >> 31);
    }

    size_t operator()(uint64_t x) const {
        static const uint64_t FIXED_RANDOM =
        ↪ chrono::steady_clock::now().time_since_epoch().count();
        return splitmix64(x + FIXED_RANDOM);
    }
};

unordered_map<long long, int, custom_hash> safe_map;
```

8.11 单调队列-定长区间最值.cpp

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

#include <iostream>
#include <algorithm>
#include <deque>

using namespace std;

typedef long long ll;

const int INF = 0x3f3f3f3f;
const int MOD = 998244353;
const int MAXN = 1e6 + 59;

int Kase, n, m;

int a[MAXN];
int ans1[MAXN], ans2[MAXN];
deque<int> qMAX, qMIN;

int main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);

    cin >> n >> m;

    for (int i = 1; i <= n; i++) {
        cin >> a[i];
    }

    for (int i = 1; i <= n; i++) {
        while (!qMIN.empty() && i - qMIN.front() >= m)
            qMIN.pop_front();
        while (!qMAX.empty() && i - qMAX.front() >= m)
            qMAX.pop_front();

        while (!qMIN.empty() && a[qMIN.back()] > a[i])
            qMIN.pop_back();

        while (!qMAX.empty() && a[qMAX.back()] < a[i])
            qMAX.pop_back();

        if (qMIN.empty() || a[qMIN.back()] <= a[i])
            qMIN.push_back(i);

        if (qMAX.empty() || a[qMAX.back()] >= a[i])
            qMAX.push_back(i);

        if (i >= m) {
            ans1[i] = a[qMIN.front()];
            ans2[i] = a[qMAX.front()];
        }
    }
    for (int j = m; j <= n; ++j) {
        cout << ans1[j] << " \n"[j == n];
    }
    for (int j = m; j <= n; ++j) {
        cout << ans2[j] << " \n"[j == n];
    }
    return 0;
}

/*


 2
```

```
2 0
1 2
1 1

 3 2
 1 2 1
 5 3 1
 1 5 1


*/
```