

Contents

1	几何	2
1.1	Circle-圆形.cpp	2
1.2	Circumcenter-外心-三点定圆.cpp	2
1.3	ClosestPoints-最近点对.cpp	2
1.4	ConvexHull-凸包.cpp	4
1.5	Hull-下凸包求函数最值.cpp	6
1.6	Line-Segment-直线与线段.cpp	8
1.7	MinCircleCover-最小圆覆盖.cpp	8
1.8	Points-Vector-点与向量.cpp	9
1.9	Polygon-多边形.cpp	10
2	图论	11
2.1	AstarKSP-A星K短路-nklogn.cpp	11
2.2	dijkstra-with-pairs.cpp	13
2.3	Dinic-by-ztc.cpp	14
2.4	Graph.txt	16
2.5	linklist.cpp	17
2.6	tarjanSCC.cpp	17
3	基础	18
3.1	fastpower.cpp	18
3.2	prime-sieve-素数筛.cpp	19
4	字符串	19
4.1	Aho-Corasick-AC自动机-多模式匹配.cpp	19
4.2	Aho-Corasick-AC自动机-统计次数-拓扑序优化.cpp	20
4.3	Levenshtein-Distance-编辑距离.py	22
4.4	manacher-单数组马拉车.cpp	23
4.5	manacher-双数组马拉车.cpp	23
5	数学	24
5.1	double-compare.cpp	24
5.2	fastFactorial-快速阶乘-分块fft.cpp	24
5.3	fft-多项式乘法.cpp	27
5.4	扩展CRT.py	28
5.5	矩阵快速幂+大十进制指数版.cpp	29
6	数据结构	30
6.1	zhuxishu-SegKth.cpp	31
6.2	ZTC's-Splay.cpp	32

7	数论	34
7.1	Binomial-Coefficients-组合数-Lucas.cpp	34
7.2	Binomial-Coefficients-组合数-Lucas.cpp.ignore	34
7.3	Binomial-Coefficients-组合数-杨辉三角.cpp	34
7.4	Binomial-Coefficients-组合数-逆元-模大素数.cpp	35
7.5	Extended-Euclidean-algorithm-(exGCD).cpp	36
7.6	ZTC's-FFT.txt	36
7.7	数论分块.cpp	36
8	杂项	37
8.1	coutf.cpp	37
8.2	debug-from-tourist.cpp	37
8.3	fast-IO-int.cpp	39
8.4	fast-IO-快速版(可敲).cpp	40
8.5	fast-IO-极致版.ignore	40
8.6	fastpow-快速幂.cpp	40
8.7	Misc-杂技-随机数.cpp	40
8.8	string-read-speed.cpp	40
8.9	timetest.cpp	41
8.10	unordered-map-自写哈希.cpp	41
8.11	单调队列-定长区间最值.cpp	42

1 几何

1.1 Circle-圆形.cpp

```
1 /**
2  * @Source: team
3  * @Author: Artiprocher(Zhongjie Duan) -> tieway59
4  * @Description:
5  * 圆形计算相关。
6  * @Example:
7  *
8  * @Verification:
9  *
10 */
11
12 struct Circle {
13     Point c;
14     double r;
15
16     Point point(double a)//基于圆心角求圆上一点坐标
17     {
18         return Point(c.x + cos(a) * r, c.y + sin(a) * r);
19     }
20 };
21
22 double Angle(Vector v1) {
23     if (v1.y >= 0) return Angle(v1, Vector(1.0, 0.0));
24     else return 2 * pi - Angle(v1, Vector(1.0, 0.0));
25 }
26
27 int GetCC(Circle C1, Circle C2)//求两圆交点
28 {
29     double d = Length(C1.c - C2.c);
30     if (dcmp(d) == 0) {
31         if (dcmp(C1.r - C2.r) == 0) return -1;//重合
32         else return 0;
33     }
34     if (dcmp(C1.r + C2.r - d) < 0) return 0;
35     if (dcmp(fabs(C1.r - C2.r) - d) > 0) return 0;
36     double a = Angle(C2.c - C1.c);
37     double da = acos((C1.r * C1.r + d * d - C2.r * C2.r) / (2 * C1.r * d));
38     Point p1 = C1.point(a - da), p2 = C1.point(a + da);
39     if (p1 == p2) return 1;
40     else return 2;
41 }
```

1.2 Circumcenter-外心-三点定圆.cpp

```
1 /**
2  * @Source: blog.csdn.net/liyuanbhu/article/details/52891868
3  * @Author: tieway59
4  * @Description:
5  * 注意排除三点共线。
6  * if (dcmp(Cross(pi, pj)) == 0) continue;
7  *
8  * @Example:
9  * circumcenter(Point(0, 1), Point(1, 1), Point(1, 0));
10 * // 0.5 0.5
11 *
12 * @Verification:
13 * https://ac.nowcoder.com/acm/contest/5667/B
14 * (solution) ac.nowcoder.com/acm/contest/view-submission?submissionId=44337916
15 *
16 */
17
18 template<typename tp>
19 inline tp pow2(const tp &x) {
20     return x * x;
21 }
22
23 inline Point circumcenter(Point p1, Point p2, Point p3) {
24     double a = p1.x - p2.x;
25     double b = p1.y - p2.y;
26     double c = p1.x - p3.x;
27     double d = p1.y - p3.y;
28     double e = (pow2(p1.x) - pow2(p2.x) +
29                 pow2(p1.y) - pow2(p2.y)) / 2;
30     double f = (pow2(p1.x) - pow2(p3.x) +
31                 pow2(p1.y) - pow2(p3.y)) / 2;
32     return Point((d * e - b * f) /
33                 (a * d - b * c),
34                 (a * f - c * e) /
35                 (a * d - b * c));
36 }
```

1.3 ClosestPoints-最近点对.cpp

```
1 /**
2  * @Source: ClosestPoints
3  * @Author: syksykCCC -> tieway59
4  * @Description:
5  * 时间复杂度  $O(N\log N)$  有一些难以预料的常数
6  *
7  * @Example:
```

```

8  *      3
9  *      1 1
10 *      1 2
11 *      2 2
12 *
13 *      // ans = 1.0000
14 *
15 * @Verification:
16 *      https://www.luogu.com.cn/problem/solution/P1429
17 */
18
19 const double EPS = 1e-6; // eps用于控制精度
20 const double Pi = acos(-1.0); // pi
21
22 // 精度三态函数(>0, <0, =0)
23 inline int dcmp(double x) {
24     if (fabs(x) < EPS) return 0;
25     else if (x > 0) return 1;
26     return -1;
27 }
28
29 // 点或向量 (iostream选择性抄写)
30 struct Point {
31     double x, y;
32
33     Point() {}
34
35     Point(double x, double y) : x(x), y(y) {}
36
37     bool operator<(const Point &r) const {
38         if (dcmp(x - r.x) == 0)
39             return dcmp(y - r.y) < 0;
40         return dcmp(x - r.x) < 0;
41     }
42
43     friend ostream &operator<<(ostream &ut, Point &r) { return ut << r.x << " " <<
44         ↪ r.y; }
45
46     friend istream &operator>>(istream &in, Point &r) { return in >> r.x >> r.y; }
47 };
48
49 typedef Point Vector;
50
51 // 两点间距离
52 inline double Distance(Point a, Point b) {
53     return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
54 }

```

```

55 // Point temp[MAXN];
56 double MAXD = INF;
57
58 double merge(vector<Point> &p, int l, int r) {
59     double d = MAXD;
60     if (l == r)
61         return d;
62     if (l + 1 == r)
63         return Distance(p[l], p[r]);
64
65     int mid = (l + r) >> 1;
66     double d1 = merge(p, l, mid);
67     double d2 = merge(p, mid + 1, r);
68     d = min(d, min(d1, d2));
69
70     vector<int> t;
71     // t.reserve(r - l + 1);
72
73     for (int i = l; i <= r; i++)
74         if (fabs(p[mid].x - p[i].x) < d)
75             t.emplace_back(i);
76
77     sort(t.begin(), t.end(),
78         [&p](const int &i, const int &j) {
79             return dcmp(p[i].y - p[j].y) < 0;
80         });
81
82     for (int i = 0; i < t.size(); i++) {
83         for (int j = i + 1; j < t.size() && p[t[j]].y - p[t[i]].y < d; j++) {
84             d = min(d, Distance(p[t[i]], p[t[j]]));
85         }
86     }
87
88     return d;
89 }
90
91 double ClosestPoints(vector<Point> &p) {
92     assert(p.size() >= 2);
93     sort(p.begin(), p.end());
94     for (int i = 3; i < p.size(); ++i) {
95         MAXD = min(MAXD, Distance(p[i], p[i - 1]));
96         MAXD = min(MAXD, Distance(p[i], p[i - 2]));
97         MAXD = min(MAXD, Distance(p[i], p[i - 3]));
98     }
99     return merge(p, 0, p.size() - 1);
100 }

```

1.4 ConvexHull-凸包.cpp

```
1  /**
2   * @Source: Graham_s_scan
3   * @Author: Artiprocher(Zhongjie Duan) -> tieway59
4   * @Description:
5   *     n      点数
6   *     P[]    点数组 index0
7   *     top    栈顶, 凸包顶点数
8   *     H[]    凸包的顶点 index0
9   *     小心重复的凸包顶点, 也会加入凸包。
10  *     H[]逆时针顺序
11  *     数组形式, 理论上常数会小?
12  *
13  * @Example:
14  *     4
15  *     4 8
16  *     4 12
17  *     5 9.3 (exclude)
18  *     7 8
19  *
20  * @Verification:
21  *     https://www.luogu.com.cn/record/35363811
22  */
23
24 int n, top;
25 const int PSIZE = 100005;
26 Point P[PSIZE], H[PSIZE];
27
28 bool cmp(Point A, Point B) {
29     double ans = Cross(A - P[0], B - P[0]);
30     if (dcmp(ans) == 0)
31         return dcmp(Distance(P[0], A) - Distance(P[0], B)) < 0;
32     else
33         return ans > 0;
34 }
35
36 //Graham凸包扫描算法
37 void Graham() {
38     for (int i = 1; i < n; i++) //寻找起点
39         if (P[i].y < P[0].y || (dcmp(P[i].y - P[0].y) == 0 && P[i].x < P[0].x))
40             swap(P[i], P[0]);
41     sort(P + 1, P + n, cmp); //极角排序, 中心为起点
42     H[0] = P[0];
43     H[1] = P[1];
44     top = 2;
45     for (int i = 2; i < n; i++) {
46         while (top >= 2 && Cross(H[top - 1] - H[top - 2], P[i] - H[top - 2]) < 0)
47             top--;
```

```
48         H[top++] = P[i];
49     }
50 }
51
52 /**
53  * @Source: Graham_s_scan
54  * @Author: Artiprocher(Zhongjie Duan) -> tieway59
55  * @Description:
56  *     小心重复的凸包顶点, 也会加入凸包。
57  *     H[]逆时针顺序
58  *     数组形式, 理论上常数会小?
59  *
60  * @Example:
61  *     4
62  *     4 8
63  *     4 12
64  *     5 9.3 (exclude)
65  *     7 8
66  *
67  * @Verification:
68  *     https://www.luogu.com.cn/record/35363811
69  *
70  */
71
72 // HEAD begin
73 const double EPS = 1e-6;
74
75 struct Point //点或向量
76 {
77     double x, y;
78
79     Point() {}
80
81     Point(double x, double y) : x(x), y(y) {}
82
83     friend ostream &operator<<(ostream &ut, Point &r) { return ut << r.x << " " <<
84         ↪ r.y; }
85
86     friend istream &operator>>(istream &in, Point &r) { return in >> r.x >> r.y; }
87 };
88
89 typedef Point Vector;
90
91 inline double Distance(Point a, Point b) {
92     return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
93 }
94
```

```

95 inline Vector operator+(Vector a, Vector b) {
96     return Vector(a.x + b.x, a.y + b.y);
97 }
98
99 inline Vector operator-(Vector a, Vector b) {
100     return Vector(a.x - b.x, a.y - b.y);
101 }
102
103 //外积
104 inline double Cross(Vector a, Vector b) {
105     return a.x * b.y - a.y * b.x;
106 }
107
108 //精度三态函数(>0,<0,=0)
109 inline int dcmp(double x) {
110     if (fabs(x) < EPS) return 0;
111     else if (x > 0) return 1;
112     return -1;
113 }
114
115 // HEAD end
116 void ConvexHull(vector<Point> &P, vector<Point> &H) {
117     int n = int(P.size());
118     for (int i = 1; i < n; i++) //寻找起点
119         if (P[i].y < P[0].y || (dcmp(P[i].y - P[0].y) == 0 && P[i].x < P[0].x))
120             swap(P[i], P[0]);
121
122     //极角排序, 中心为起点
123     sort(P.begin() + 1, P.end(), [&P](Point A, Point B) {
124         double ans = Cross(A - P[0], B - P[0]);
125         if (dcmp(ans) == 0)
126             return dcmp(Distance(P[0], A) - Distance(P[0], B)) < 0;
127         else
128             return ans > 0;
129     });
130
131     H.assign(n + n, {});
132     H[0] = P[0];
133     H[1] = P[1];
134     int top = 2;
135     for (int i = 2; i < n; i++) {
136         while (top >= 2 && Cross(H[top - 1] - H[top - 2], P[i] - H[top - 2]) < 0)
137             top--;
138         H[top++] = P[i];
139     }
140     H.resize(top);
141 }
142

```

```

143 /**
144  * @Source: Andrew_s_monotone_chain
145  * @Author: Artiprocher(Zhongjie Duan) -> tieway59
146  * @Description:
147  *     Andrew_s_monotone_chain
148  *     从左下角开始逆时针排列, 去除凸包边上的点。
149  *     求出来的凸包是逆时针的。
150  *     points in h[] are counter-clockwise
151  *
152  * @Example:
153  *     vector<Point> p(n);
154  *     for (auto &pi : p) cin >> pi;
155  *     vector<Point> r;
156  *     ConvexHull(p, r);
157  *
158  *     4
159  *     4 8
160  *     4 12
161  *     5 9.3 (exclude)
162  *     7 8
163  *
164  * @Verification:
165  *     https://www.luogu.com.cn/problem/P2742
166  */
167
168 // HEAD begin
169 const double EPS = 1e-6;
170
171 struct Point//点或向量
172 {
173     double x, y;
174
175     Point() {}
176
177     Point(double x, double y) : x(x), y(y) {}
178
179     friend ostream &operator<<(ostream &ut, Point &r) { return ut << r.x << " " <<
180         ↪ r.y; }
181
182     friend istream &operator>>(istream &in, Point &r) { return in >> r.x >> r.y; }
183 };
184
185 typedef Point Vector;
186
187 inline double Distance(Point a, Point b) {
188     return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
189 }
190

```

```

190 inline Vector operator+(Vector a, Vector b) {
191     return Vector(a.x + b.x, a.y + b.y);
192 }
193
194 inline Vector operator-(Vector a, Vector b) {
195     return Vector(a.x - b.x, a.y - b.y);
196 }
197
198 //外积
199 inline double Cross(Vector a, Vector b) {
200     return a.x * b.y - a.y * b.x;
201 }
202
203 //精度三态函数(>0,<0,=0)
204 inline int dcmp(double x) {
205     if (fabs(x) < EPS) return 0;
206     else if (x > 0) return 1;
207     return -1;
208 }
209 // HEAD end
210
211 inline bool pcmp(Point a, Point b) {
212     if (dcmp(a.x - b.x) == 0)
213         return a.y < b.y;
214     return a.x < b.x;
215 }
216
217 void ConvexHull(vector <Point> &p, vector <Point> &h) {
218     int n = p.size(), k = 0;
219     h.assign(2 * n, {});
220     sort(p.begin(), p.end(), pcmp);
221     for (int i = 0; i < n; i++) {
222         while (k >= 2 && dcmp(Cross(
223             h[k - 1] - h[k - 2],
224             p[i] - h[k - 2])) < 0) {
225             k--;
226         }
227         h[k++] = p[i];
228     }
229
230     int t = k + 1;
231     for (int i = n - 1; i > 0; i--) {
232         while (k >= t && dcmp(Cross(
233             h[k - 1] - h[k - 2],
234             p[i - 1] - h[k - 2])) < 0) {
235             k--;
236         }
237         h[k++] = p[i - 1];

```

```

238     }
239
240     h.resize(k - 1);
241 }

```

1.5 Hull-下凸包求函数最值.cpp

```

1  /* Author: bnfcc -> tc2000731 -> tieaway59
2  * Description:
3  *     维护下凸包, 对于每个x维护f(x)=k*x+b的最大值。
4  *     query max value within all f(x) functions.
5  *     c++11 features included.
6  * Problems:
7  *     https://nanti.jisuanke.com/t/41306
8  *     https://nanti.jisuanke.com/t/41097
9  */
10 template<typename var=long long, const int SIZE = 1000005, typename ldb=long double>
11 struct Hull {
12     struct fx {
13         var k, b;
14
15         fx() {}
16
17         fx(var k, var b) : k(k), b(b) {}
18
19         var f(var x) { return k * x + b; }
20     };
21
22     int cnt;
23     fx arr[SIZE];
24
25     bool empty() {
26         return cnt == 0;
27     }
28
29     void init() {
30         cnt = 0;
31     }
32
33     void add(const fx &p) {
34         arr[cnt++] = p;
35     }
36
37     void pop() {
38         cnt--;
39     }
40
41     bool chek(const fx &a, const fx &b, const fx &c) {

```

```

42     ldb ab, ak, bb, bk, cb, ck;
43     tie(ab, ak, bb, bk, cb, ck) =
44         tie(a.b, a.k, b.b, b.k, c.b, c.k);
45     return (ab - bb) / (bk - ak) > (ab - cb) / (ck - ak);
46 }
47
48 void insert(const fx &p) {///k从小到大插入
49     if (cnt && arr[cnt - 1].k == p.k) {
50         if (p.b <= arr[cnt - 1].b) return;
51         else pop();
52     }
53     while (cnt >= 2 && chek(arr[cnt - 2], arr[cnt - 1], p)) pop();
54     add(p);
55 }
56
57 /*var query(var x) {///x从大到小查询 从小到大用队列
58     while (cnt > 1 && arr[cnt - 2].f(x) > arr[cnt - 1].f(x)) pop();;
59     return arr[cnt - 1].f(x);
60 }*/
61
62 var query(var x) {///二分查询, x顺序任意
63     int l = 0, r = cnt - 1;
64     while (l < r) {
65         int mid = (l + r) >> 1;
66         if (arr[mid].f(x) >= arr[mid + 1].f(x)) r = mid;
67         else l = mid + 1;
68     }
69     return arr[l].f(x);
70 }
71 };
72
73 // vector stack
74 template<typename var=long long, const int SIZE = 1000005, typename ldb=long double>
75 struct Hull {
76     struct Line {
77         var k, b;
78
79         Line() {}
80
81         Line(var k, var b) : k(k), b(b) {}
82
83         var f(var x) { return k * x + b; }
84     };
85
86     int cnt;
87     vector<Line> con;
88
89     bool empty() {

```

```

90         return cnt == 0;
91     }
92
93     void init(const int &n) {
94         con.clear();
95         if (n > con.capacity()) con.reserve(n);
96         cnt = 0;
97     }
98
99     void add(const Line &p) {
100         con.emplace_back(p);
101         cnt++;
102     }
103
104     void pop() {
105         cnt--;
106         con.pop_back();
107     }
108
109     bool chek(const Line &a, const Line &b, const Line &c) {
110         ldb ab, ak, bb, bk, cb, ck;
111         tie(ab, ak, bb, bk, cb, ck) =
112             tie(a.b, a.k, b.b, b.k, c.b, c.k);
113         return (ab - bb) / (bk - ak) > (ab - cb) / (ck - ak);
114     }
115
116     void insert(const Line &p) {///k从小到大插入
117         if (cnt && con[cnt - 1].k == p.k) {
118             if (p.b <= con[cnt - 1].b) return;
119             else pop();
120         }
121         while (cnt >= 2 && chek(con[cnt - 2], con[cnt - 1], p)) pop();
122         add(p);
123     }
124
125     var query(var x) {///二分查询, x顺序任意
126         int l = 0, r = cnt - 1;
127         while (l < r) {
128             int mid = (l + r) >> 1;
129             if (con[mid].f(x) >= con[mid + 1].f(x)) r = mid;
130             else l = mid + 1;
131         }
132         return con[l].f(x);
133     }
134 };
135
136 Hull<> hull;

```

1.6 Line-Segment-直线与线段.cpp

```
1  /**
2   * @Source: team
3   * @Author: Artiprocher(Zhongjie Duan) -> tieway59
4   * @Description:
5   *     直线与线段的相关计算。
6   *
7   * @Example:
8   *
9   * @Verification:
10  */
11 //定义直线
12 struct line {
13     point a, b;
14 };
15
16
17 //线段相交 (不包括端点)
18 bool Intersect(Point A, Point B, Point C, Point D) {
19     double t1 = Cross(C - A, D - A) * Cross(C - B, D - B);
20     double t2 = Cross(A - C, B - C) * Cross(A - D, B - D);
21     return dcmp(t1) < 0 && dcmp(t2) < 0;
22 }
23
24 //线段相交 (包括端点)
25 bool StrictIntersect(Point A, Point B, Point C, Point D) {
26     return dcmp(max(A.x, B.x) - min(C.x, D.x)) >= 0
27         && dcmp(max(C.x, D.x) - min(A.x, B.x)) >= 0
28         && dcmp(max(A.y, B.y) - min(C.y, D.y)) >= 0
29         && dcmp(max(C.y, D.y) - min(A.y, B.y)) >= 0
30         && dcmp(Cross(C - A, D - A) * Cross(C - B, D - B)) <= 0
31         && dcmp(Cross(A - C, B - C) * Cross(A - D, B - D)) <= 0;
32 }
33
34 //点A到直线MN的距离, Error: MN=0
35 double DistanceToLine(Point A, Point M, Point N) {
36     return fabs(Cross(A - M, A - N) / Distance(M, N));
37 }
38
39 //两直线的交点
40 Point GetLineIntersection(Point P, Vector v, Point Q, Vector w) {
41     Vector u = P - Q;
42     double t = Cross(w, u) / Cross(v, w);
43     return P + v * t;
44 }
45 }
```

1.7 MinCircleCover-最小圆覆盖.cpp

```
1  /**
2   * @Source: https://www.luogu.com.cn/problem/solution/P1742
3   * @Author: snowbody -> tieway59
4   * @Description:
5   *     时间复杂度  $O(N)$ 
6   *     为了减少中途过度开根, 距离都是先按照平方计算的。
7   *
8   * @Example:
9   *     vector<Point> p(n);
10    *     for (auto &pi : p) cin >> pi;
11    *     Circle circle;
12    *     MinCircleCover(p, circle);
13    *
14    *     6
15    *     8.0 9.0
16    *     4.0 7.5
17    *     1.0 2.0
18    *     5.1 8.7
19    *     9.0 2.0
20    *     4.5 1.0
21    *     // r = 5.0000000000 (5.0000000000, 5.0000000000)
22    *
23    * @Verification:
24    *     https://www.luogu.com.cn/problem/P1742
25    */
26
27 //点或向量 (iostream选择性抄写)
28 struct Point {
29     double x, y;
30
31     Point() {}
32
33     Point(double x, double y) : x(x), y(y) {}
34
35     friend ostream &operator<<(ostream &ut, Point &r) { return ut << r.x << " " <<
36         ↪ r.y; }
37
38     friend istream &operator>>(istream &in, Point &r) { return in >> r.x >> r.y; }
39 };
40
41 typedef Point Vector;
42
43 inline Vector operator+(Vector a, Vector b) {
44     return Vector(a.x + b.x, a.y + b.y);
45 }
46 }
```



```

47 inline Vector operator-(Vector a, Vector b) {
48     return Vector(a.x - b.x, a.y - b.y);
49 }
50
51 //向量数乘
52 inline Vector operator*(Vector a, double p) {
53     return Vector(a.x * p, a.y * p);
54 }
55
56 //向量数除
57 inline Vector operator/(Vector a, double p) {
58     return Vector(a.x / p, a.y / p);
59 }
60
61 //两点间距离
62 inline double Distance(Point a, Point b) {
63     return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
64 }
65
66 inline double Distance2(Point a, Point b) {
67     return ((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
68 }
69
70 struct Circle {
71     Point c;
72     double r;
73
74     Point point(double a)//基于圆心角求圆上一点坐标
75     {
76         return Point(c.x + cos(a) * r, c.y + sin(a) * r);
77     }
78 };
79
80 template<typename tp>
81 inline tp pow2(const tp &x) {
82     return x * x;
83 }
84
85 inline Point circumcenter(Point p1, Point p2, Point p3) {
86     double a = p1.x - p2.x;
87     double b = p1.y - p2.y;
88     double c = p1.x - p3.x;
89     double d = p1.y - p3.y;
90     double e = (pow2(p1.x) - pow2(p2.x) +
91                 pow2(p1.y) - pow2(p2.y)) / 2;
92     double f = (pow2(p1.x) - pow2(p3.x) +
93                 pow2(p1.y) - pow2(p3.y)) / 2;

```

```

95     return Point((d * e - b * f) /
96                  (a * d - b * c),
97                  (a * f - c * e) /
98                  (a * d - b * c));
99 }
100
101 void MinCircleCover(vector<Point> &p, Circle &res) {
102     int n = p.size();
103     random_shuffle(p.begin(), p.end());
104     // avoid *sqrt* too much killing your precision.
105     for (int i = 0; i < n; i++) {
106         if (Distance2(p[i], res.c) <= res.r) continue;
107         res.c = p[i];
108         res.r = 0;
109         for (int j = 0; j < i; j++) {
110             if (Distance2(p[j], res.c) <= res.r) continue;
111             res.c = (p[i] + p[j]) / 2;
112             res.r = Distance2(p[j], res.c);
113             for (int k = 0; k < j; k++) {
114                 if (Distance2(p[k], res.c) <= res.r) continue;
115                 res.c = circumcenter(p[i], p[j], p[k]);
116                 res.r = Distance2(p[k], res.c);
117             }
118         }
119     }
120     res.r = sqrt(res.r);
121 }
122
123 void solve(int kaseId = -1) {
124     int n;
125     cin >> n;
126     vector<Point> p(n);
127     for (auto &pi : p) cin >> pi;
128     Circle circle;
129     MinCircleCover(p, circle);
130     cout << fixed << setprecision(10) << circle.r << endl;
131     cout << circle.c.x << " " << circle.c.y << endl;
132 }
133

```

1.8 Points-Vector-点与向量.cpp

```

1 /**
2  * @Source: team
3  * @Author: Artiprocher(Zhongjie Duan) -> tieway59
4  * @Description:
5  * 点与向量相关的多种计算。
6  * @Example:

```

```

7  *
8  * @Verification:
9  *
10 */
11
12
13 //include <bits/stdc++.h>
14 //using namespace std;
15 const double EPS = 1e-6;//eps用于控制精度
16 const double Pi = acos(-1.0);//pi
17
18 //精度三态函数(>0,<0,=0)
19 inline int dcmp(double x) {
20     if (fabs(x) < EPS)return 0;
21     else if (x > 0)return 1;
22     return -1;
23 }
24
25 //点或向量 (iostream选择性抄写)
26 struct Point {
27     double x, y;
28
29     Point() {}
30
31     Point(double x, double y) : x(x), y(y) {}
32
33     friend ostream &operator<<(ostream &ut, Point &r) { return ut << r.x << " " <<
34         ↪ r.y; }
35
36     friend istream &operator>>(istream &in, Point &r) { return in >> r.x >> r.y; }
37 };
38
39 typedef Point Vector;
40
41 inline Vector operator+(Vector a, Vector b) {
42     return Vector(a.x + b.x, a.y + b.y);
43 }
44
45 inline Vector operator-(Vector a, Vector b) {
46     return Vector(a.x - b.x, a.y - b.y);
47 }
48
49 //向量数乘
50 inline Vector operator*(Vector a, double p) {
51     return Vector(a.x * p, a.y * p);
52 }
53
54 //向量数除

```

```

54 inline Vector operator/(Vector a, double p) {
55     return Vector(a.x / p, a.y / p);
56 }
57
58 inline bool operator==(const Point &a, const Point &b) {
59     return dcmp(a.x - b.x) == 0 && dcmp(a.y - b.y) == 0;
60 }
61
62 //内积
63 inline double Dot(Vector a, Vector b) {
64     return a.x * b.x + a.y * b.y;
65 }
66
67 //外积
68 inline double Cross(Vector a, Vector b) {
69     return a.x * b.y - a.y * b.x;
70 }
71
72 //模
73 inline double Length(Vector a) {
74     return sqrt(Dot(a, a));
75 }
76
77 //夹角,弧度制
78 inline double Angle(Vector a, Vector b) {
79     return acos(Dot(a, b) / Length(a) / Length(b));
80 }
81
82 //逆时针旋转
83 inline Vector Rotate(Vector a, double rad) {
84     return Vector(a.x * cos(rad) - a.y * sin(rad), a.x * sin(rad) + a.y * cos(rad));
85 }
86
87 //两点间距离
88 inline double Distance(Point a, Point b) {
89     return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
90 }
91
92 //三角形面积
93 inline double Area(Point a, Point b, Point c) {
94     return fabs(Cross(b - a, c - a) / 2);
95 }

```

1.9 Polygon-多边形.cpp

```

1 /**
2  * @Source: team
3  * @Author: Artiprocher(Zhongjie Duan) -> tieway59

```

```

4  * @Description:
5  * 多边形相关的计算。
6  * @Example:
7  *
8  * @Verification:
9  *
10 */
11
12 Point P[1005]; // P[]为多边形的所有顶点, 下标为0~n-1
13 int n; // n为多边形边数
14 // 求多边形面积 (叉积和算法)
15 double PolygonArea() {
16     double sum = 0;
17     Point O = Point(0, 0);
18     for (int i = 0; i < n; i++)
19         sum += Cross(P[i] - O, P[(i + 1) % n] - O);
20     if (sum < 0) sum = -sum;
21     return sum / 2;
22 }
23
24 // STL: 求多边形面积 (叉积和算法)
25 double PolygonArea(const vector<Point> &P) {
26     int n = P.size();
27     // assert(n > 2);
28     double sum = 0;
29     Point O = Point(0, 0);
30     for (int i = 0; i < n; i++)
31         sum += Cross(P[i] - O, P[(i + 1) % n] - O);
32     if (sum < 0) sum = -sum;
33     return sum / 2;
34 }
35
36 /*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*/
37 //判断点是否在凸多边形内 (角度和判别法)
38 Point P[1005];
39 int n;
40
41 bool InsidePolygon(Point A) {
42     double alpha = 0;
43     for (int i = 0; i < n; i++)
44         alpha += fabs(Angle(P[i] - A, P[(i + 1) % n] - A));
45     return dcmp(alpha - 2 * pi) == 0;
46 }

```

2 图论

tieway59

2.1 AstarKSP-A星K短路-nklogn.cpp

```

1 /**
2  * @Source: myself
3  * @Author: Tieway59
4  * @Complexity: O(nklogn)
5  * @Description:
6  *     g.addEdge(u, v, w);
7  *     build graph & inverse_graph
8  *
9  *     g.AstarKSP(inv_g, s, t, kth, ... );
10 *     return k-th shortest path length or -1
11 *
12 *     ! KSP might not be strictly longer than (k-1)SP
13 *     ! it's MLE/TLE for large K
14 *     ! be aware of int overflow
15 *     ! the "cut" is one example for passing skip function.
16 *     ! I know this code is too long,
17 *     ! it'll be easier if wrote into single functions.
18 *     Since you need two graphs, this graph class works out fine.
19 *
20 * @Example:
21 *
22 * @Verification:
23 *     https://nanti.jisuanke.com/t/A1992 (input k)
24 *     http://acm.hdu.edu.cn/showproblem.php?pid=6181 (k = 2)
25 */
26
27 using node_t = int;
28 using cost_t = long long;
29 using pqnd_t = pair<cost_t, node_t>;
30
31 class Graph {
32 public:
33     int nsize = 0;
34     int esize = 0;
35
36     struct Edge {
37         node_t v;
38         cost_t w;
39         int nx;
40     };
41
42     vector<int> head;
43     vector<Edge> edge;
44
45     Graph() {}
46
47     Graph(int n, int m) : nsize(n), esize(m) {

```

```

48     head.assign(n, -1);
49     edge.reserve(m);
50 }
51
52 // number from 0
53 inline void addEdge(node_t u, node_t v, cost_t w) {
54     edge.emplace_back((Edge) {v, w, head[u]});
55     head[u] = edge.size() - 1;
56 }
57
58 static void dijkstra(const Graph &g, const node_t &s, vector<cost_t> &d);
59
60 cost_t AstarKSP(const Graph &inv_g, node_t s, node_t t, int k, function<const
↵ bool(cost_t)> cut);
61 };
62
63 void Graph::dijkstra(const Graph &g, const node_t &s, vector<cost_t> &d) {
64     d.assign(g.nsize, INF);
65     d[s] = 0;
66
67 // using pqnd_t = pair<cost_t, node_t>;
68 priority_queue<pqnd_t, vector<pqnd_t>, greater<pqnd_t> > q;
69 q.emplace(d[s], s);
70
71 node_t u, v;
72 cost_t w, du;
73 while (!q.empty()) {
74     du = q.top().first;
75     u = q.top().second;
76     q.pop();
77     if (du > d[u])continue;
78     for (int i = g.head[u]; i != -1; i = g.edge[i].nx) {
79         v = g.edge[i].v;
80         w = g.edge[i].w;
81         if (du + w < d[v]) {
82             d[v] = du + w;
83             q.emplace(d[v], v);
84         }
85     }
86 }
87 }
88
89 //O(nklogn) : beware of n-circle.
90 cost_t Graph::AstarKSP(const Graph &inv_g, node_t s, node_t t, int k,
91     function<const bool(cost_t)> cut) {
92     vector<cost_t> dis_t;
93     vector<int> vis(nsize, 0);
94     Graph::dijkstra(inv_g, t, dis_t);

```

```

95
96 // if(s==t) k++; when the node are not defined as a path.
97 if (dis_t[s] == 11INF)return -1;
98
99 auto Astar = [&](pqnd_t x, pqnd_t y) -> bool {
100     return x.first + dis_t[x.second] >
101         y.first + dis_t[y.second];
102 };
103 // BFS-similar :
104 node_t u = s;
105 cost_t dis_s;
106 priority_queue<pqnd_t, vector<pqnd_t>, decltype(Astar)> q(Astar);
107 vis[u] = 1;
108 q.emplace(0, u);
109 while (!q.empty()) {
110     dis_s = q.top().first;
111     u = q.top().second;
112     q.pop();
113
114     if (u == t && vis[u] == k)return dis_s;
115
116     for (int i = head[u]; i != -1; i = edge[i].nx) {
117         node_t v = edge[i].v;
118         cost_t w = edge[i].w;
119
120         if (cut(dis_s + w))continue;
121         if (cut(dis_s + w + dis_t[v]))continue;
122
123         // below is a risky-but-worth skip, take care :
124         // if k == 2, skip vis > k
125         // else skip vis >= k
126         // (proved practically not theoretically. )
127         if (vis[v] >= max(3, k))continue;
128         else vis[v]++;
129
130         q.emplace(dis_s + w, v);
131     }
132 }
133 return -1;
134 }
135
136
137 void solve(int kaseId = -1) {
138     int n, m;
139     node_t s, t, kth;
140     cost_t limit = 0;
141
142     const auto cut = [&](cost_t cost) -> bool {

```

```

143     return cost > limit;
144 };
145
146 while (cin >> n >> m) {
147     cin >> s >> t >> kth >> limit;
148     s--, t--;
149     Graph g(n, m);
150     Graph inv_g(n, m);
151
152     for (ll i = 1, u, v, w; i <= m; ++i) {
153         cin >> u >> v >> w;
154         u--, v--;
155         g.addEdge(u, v, w);
156         inv_g.addEdge(v, u, w);
157     }
158     cost_t res = g.AstarKSP(inv_g, s, t, kth, cut);
159     if (res == -1 || cut(res))
160         cout << "Whitesnake!" << endl;
161     else
162         cout << "yareyaredawa" << endl;
163 }
164 }

```

2.2 dijkstra-with-pairs.cpp

```

1 using cost_t = long long;    //beware of integer overflow
2 using node_t = int;
3 using edge_t = pair<node_t, cost_t>;
4 using pqnd_t = pair<cost_t, node_t>;
5
6 vector <vector<edge_t>> adj;
7
8 void dijkstra(int s, vector <cost_t> &d) {
9     int n = adj.size();
10    d.assign(n, INF);    // distance
11
12    d[s] = 0;
13    priority_queue <pqnd_t, vector<pqnd_t>, greater<pqnd_t>> q;
14    q.emplace(0, s);
15
16    node_t u, v;
17    cost_t dis, len;
18    while (!q.empty()) {
19        dis = q.top().first;
20        u = q.top().second;
21        q.pop();
22        if (dis > d[u]) // i.e. !=
23            continue;

```

```

24
25        for (auto edge : adj[u]) {
26            v = edge.first;
27            len = edge.second;
28            if (d[u] + len < d[v]) {
29                d[v] = d[u] + len;
30                q.emplace(d[v], v);
31            }
32        }
33    }
34 }
35
36 // get path:
37
38 using cost_t = long long;    //beware of integer overflow
39 using node_t = int;
40 using edge_t = pair<node_t, cost_t>;
41 using pqnd_t = pair<cost_t, node_t>;
42
43 vector <vector<edge_t>> adj;
44 vector <cost_t> tag;
45
46 void dijkstra(int s, vector <cost_t> &d, vector <node_t> &p) {
47     int n = adj.size();
48     d.assign(n, INF);    // distance
49     p.assign(n, -1);    // path-pre
50
51     d[s] = 0;
52     priority_queue <pqnd_t, vector<pqnd_t>, greater<pqnd_t>> q;
53     q.emplace(0, s);
54
55     node_t u, v;
56     cost_t dis, len;
57     while (!q.empty()) {
58         dis = q.top().first;
59         u = q.top().second;
60         q.pop();
61         if (dis > d[u]) // i.e. !=
62             continue;
63
64         for (auto edge : adj[u]) {
65             v = edge.first;
66             len = edge.second;
67             if (d[u] + len + tag[v] < d[v]) {
68                 d[v] = d[u] + len + tag[v];
69                 p[v] = u;    /*
70                 q.emplace(d[v], v);
71             }

```

```

72     }
73 }
74 }

```

2.3 Dinic-by-ztc.cpp

```

1  #include<stdio.h>
2  #include<algorithm>
3  #include<string.h>
4  #include<set>
5  #include<queue>
6  #include<map>
7  #include<ctype.h>
8  #include<math.h>
9  #include<time.h>
10 #include<stdlib.h>
11 #include<unordered_map>
12 #include<list>
13 #include<complex>
14 #include<unordered_set>
15 #include<stack>
16 #include<string>
17 #include<iostream>
18 #define _Inf(a) memset(a,0x3f,sizeof(a))
19 #define _Neg1(a) memset(a,-1,sizeof(a))
20 #define _Rep(i,a,b) for(int (i)=a;(i)<=(b);(i)++)
21 using namespace std;
22 typedef long long ll;
23 const int INF = 0x3f3f3f3f;
24 typedef double db;
25 typedef complex<db> cp;
26 typedef pair<int, int> pii;
27 typedef pair<ll, ll> pll;
28 typedef pair<db, db> pdd;
29 const int MOD = 998244353;
30 const db EPS = 1e-8;
31 const db PI = acos(-1);
32 int sign(db x) { return x<-EPS ? -1 : x>EPS; }
33 int dbcmp(db l, db r) { return sign(l - r); }
34 ll gcd(ll a, ll b) { return b ? gcd(b, a%b) : a; }
35 const int MAXN = 1e5 + 54;
36
37 const int MAXG = 1e5 + 50;
38 struct Edge
39 {
40     int from, to, cost, nxt;
41 };
42 struct Graph

```

```

43 {
44     struct Edge E[MAXG];
45     int cntE, head[MAXN];
46     void init() { _Neg1(head); cntE = 0; }
47     void addE(int a, int b, int c = 0) { E[cntE] = { a,b,c,head[a] }; head[a] =
48         ↪ cntE++; }
49 };
50 struct Dijkstra : Graph//下面定一个变量就能用
51 {
52     ll dist[MAXG];
53     struct DNode
54     {
55         ll val;int id;
56         bool operator< (const DNode &r)const
57         {
58             return val > r.val;
59         }
60     };
61     void Init() { _Inf(dist); }
62
63     void Get_Dist(int s)//重新计算从s开始的单源最短路
64     {
65         Init();
66         priority_queue<DNode>pq;
67         pq.push({ 0,s });
68         dist[s] = 0;
69         while (!pq.empty())
70         {
71             DNode tmp = pq.top(); pq.pop();
72             if (tmp.val > dist[tmp.id])continue;
73             for (int i = head[tmp.id]; i != -1; i = E[i].nxt)
74             {
75                 if (dist[E[i].to] > dist[tmp.id] + E[i].cost)
76                 {
77                     dist[E[i].to] = dist[tmp.id] + E[i].cost;
78                     pq.push({ dist[E[i].to],E[i].to });
79                 }
80             }
81         }
82     }
83
84     int Get_Dist(int s, int t)//获取s到t的最短路
85     {
86         if(dist[t]==INF&&dist[s]!=0)Get_Dist(s);
87         return dist[t];
88     }
89 }Dij;

```

```

90 struct Dinic :Graph
91 {
92     int curE[MAXG], s, t, dist[MAXG];
93
94     ll dfs(int u, ll f)//不用管, 不要调用
95     {
96         if (u == t)return f;
97         int ans = 0;
98         for (int &i = curE[u]; i != -1; i = E[i].nxt)
99         {
100             if (dist[E[i].to] == dist[u] + 1 && E[i].cost > 0)
101             {
102                 ll tmp = dfs(E[i].to, min(f, (ll)E[i].cost));
103                 f -= tmp;
104                 E[i].cost -= tmp;
105                 ans += tmp;
106                 E[i ^ 1].cost += tmp;
107                 if (!f)break;
108             }
109         }
110         if (!ans)dist[u] = -1;
111         return ans;
112     }
113
114     bool bfs()//同上
115     {
116         _Neg1(dist);
117         queue<int> q; q.push(s);
118         dist[s] = 0;
119         while (!q.empty())
120         {
121             int u = q.front(); q.pop();
122             for (int i = head[u]; i != -1; i = E[i].nxt)
123             {
124                 if (dist[E[i].to] == -1 && E[i].cost > 0)
125                 {
126                     dist[E[i].to] = dist[u] + 1;
127                     q.push(E[i].to);
128                 }
129             }
130         }
131         return dist[t] != -1;
132     }
133
134     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
135     {
136         s = x; t = y;
137         int ans = 0;

```

```

138         while (bfs())
139         {
140             for (int i = 1; i <= n; i++)curE[i] = head[i];
141             ans += dfs(s, INF);
142         }
143         return ans;
144     }
145 }Din;
146 int main()
147 {
148     int T;
149     scanf("%d", &T);
150     while (T--)
151     {
152         Dij.init();Din.init();
153         int n, m;
154         scanf("%d%d", &n, &m);
155         _Rep(i, 1, m)
156         {
157             int a, b, c;
158             scanf("%d%d%d", &a, &b, &c);
159             Dij.addE(a, b, c);
160         }
161         Dij.Get_Dist(1);
162         for (int i = 0; i < Dij.cntE; i++)
163         {
164             Edge &ed = Dij.E[i];
165             if (Dij.dist[ed.from] + ed.cost == Dij.dist[ed.to])
166             {
167                 Din.addE(ed.from, ed.to, ed.cost);
168                 Din.addE(ed.to, ed.from, 0);
169             }
170         }
171         printf("%lld\n",Din.dinic(1,n,n));
172     }
173 }
174
175 /*
176 9 28
177 6 4 411
178 1 5 690
179 9 3 304
180 5 1 206
181 3 9 144
182 2 1 799
183 2 9 832
184 3 9 857

```

```

186 6 7 897
187 3 4 313
188 8 9 470
189 6 4 751
190 1 4 599
191 5 1 139
192 3 4 811
193 7 2 433
194 2 3 171
195 9 7 380
196 7 7 497
197 2 6 400
198 6 8 959
199 7 7 82
200 5 1 333
201 5 9 850
202 3 6 780
203 8 5 111
204 9 9 159
205 4 4 896
206 */

```

2.4 Graph.txt

```

1 const int MAXG = -1;
2 struct Edge
3 {
4     int from, to, cost, nxt;
5 };
6 struct Graph
7 {
8     struct Edge E[MAXG];
9     int cntE, head[MAXN];
10    void init() { _Neg1(head); cntE = 0; }
11    void addE(int a, int warrior, int c = 0) { E[cntE] = { a,warrior,c,head[a] };
12        ↪ head[a] = cntE++; }
13 };
14 struct Dijkstra : Graph//下面定一个变量就能用
15 {
16     ll dist[MAXG];
17     struct DNode
18     {
19         ll val;int id;
20         bool operator< (const DNode &r)const
21         {
22             return val > r.val;
23         }
24     };
25 };

```

tieway59

```

24 void Init() { _Inf(dist); }
25 void Get_Dist(int s)//重新计算从s开始的单源最短路
26 {
27     Init();
28     priority_queue<DNode>pq;
29     pq.push({ 0,s });
30     dist[s] = 0;
31     while (!pq.empty())
32     {
33         DNode tmp = pq.top(); pq.pop();
34         if (tmp.val > dist[tmp.id])continue;
35         for (int i = head[tmp.id]; i != -1; i = E[i].nxt)
36         {
37             if (dist[E[i].to] > dist[tmp.id] + E[i].cost)
38             {
39                 dist[E[i].to] = dist[tmp.id] + E[i].cost;
40                 pq.push({ dist[E[i].to],E[i].to });
41             }
42         }
43     }
44 }
45 int Get_Dist(int s, int t)//获取s到t的最短路
46 {
47     if(dist[t]==INF&&dist[s]!=0)Get_Dist(s);
48     return dist[t];
49 }
50 }Dij;
51 struct Dinic :Graph
52 {
53     int curE[MAXG], s, t, dist[MAXG];
54     ll dfs(int u, ll f)//不用管，不要调用
55     {
56         if (u == t)return f;
57         int ans = 0;
58         for (int &i = curE[u]; i != -1; i = E[i].nxt)
59         {
60             if (dist[E[i].to] == dist[u] + 1 && E[i].cost > 0)
61             {
62                 ll tmp = dfs(E[i].to, min(f, (ll)E[i].cost));
63                 f -= tmp;
64                 E[i].cost -= tmp;
65                 ans += tmp;
66                 E[i ^ 1].cost += tmp;
67                 if (!f)break;
68             }
69         }
70         if (!ans)dist[u] = -1;
71         return ans;

```



```

72     }
73     bool bfs()//同上
74     {
75         _Neg1(dist);
76         queue<int> q; q.push(s);
77         dist[s] = 0;
78         while (!q.empty())
79         {
80             int u = q.front(); q.pop();
81             for (int i = head[u]; i != -1; i = E[i].nxt)
82             {
83                 if (dist[E[i].to] == -1 && E[i].cost > 0)
84                 {
85                     dist[E[i].to] = dist[u] + 1;
86                     q.push(E[i].to);
87                 }
88             }
89         }
90         return dist[t] != -1;
91     }
92     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
93     {
94         s = x; t = y;
95         int ans = 0;
96         while (bfs())
97         {
98             for (int i = 1; i <= num; i++) curE[i] = head[i];
99             ans += dfs(s, INF);
100         }
101         return ans;
102     }
103 }Din;

```

2.5 linklist.cpp

```

1 //
2 // Created by acm-33 on 2019/7/23.
3 //
4
5 #define myDebug(x) cerr<<#x<<" "<<x<<endl
6
7 #include <string.h>
8 #include <algorithm>
9 #include <iostream>
10
11 using namespace std;
12 const int INF = 0x3f3f3f3f;

```

tieway59

```

14 const int MAXN = 1e3 + 7;
15
16
17 struct Edge {
18     int u, v, nx; // ,w
19 } e[MAXN << 2];
20
21 int head[MAXN], cntEd;
22
23 inline void addEdge(int u, int v) {
24     e[cntEd] = {u, v, head[u]};
25     head[u] = cntEd++;
26 }

```

2.6 tarjanSCC.cpp

```

1 //http://poj.org/status?problem_id=&user_id=tieway59&result=&language=
2 #define myDebug(x) cerr<<#x<<" "<<x<<endl
3
4 #include <string.h>
5 #include <algorithm>
6 #include <iostream>
7
8
9 using namespace std;
10 const int INF = 0x3f3f3f3f;
11 const int MAXN = 1e3 + 7;
12
13
14 struct Edge {
15     int u, v, nx; // ,w
16 } e[MAXN << 2];
17
18 int head[MAXN], cntEd;
19
20 inline void addEdge(int u, int v) {
21     e[cntEd] = {u, v, head[u]};
22     head[u] = cntEd++;
23 }
24
25 //-----tarjan
26 int dfn[MAXN], low[MAXN], scc[MAXN], stk[MAXN], index = 0, sccnum = 0, top = 0;
27
28 void tarjan(int root) {
29     if (dfn[root]) return;
30     dfn[root] = low[root] = ++index;
31     stk[++top] = root;
32     for (int i = head[root]; ~i; i = e[i].nx) {

```

```

33     int v = e[i].v;
34     if (!dfn[v]) { //如果v结点未访问过
35         tarjan(v);
36         low[root] = min(low[root], low[v]);
37     } else if (!scc[v]) { //如果还在栈内
38         low[root] = min(low[root], dfn[v]);
39     }
40 }
41 if (low[root] == dfn[root]) { //后代不能找到更浅的点
42     sccnum++;
43     for (;;) {
44         int x = stk[top--];
45         scc[x] = sccnum;
46         if (x == root) break;
47     }
48 }
49 }
50 //-----
51
52 int ind[MAXN], oud[MAXN];
53
54 int main() {
55     memset(head, -1, sizeof head);
56
57     ios::sync_with_stdio(0);
58     cin.tie(0);
59
60     int n;
61
62     cin >> n;
63     for (int v, i = 1; i <= n; i++) {
64         while (cin >> v && v) {
65             addEdge(i, v);
66         }
67     }
68     for (int i = 1; i <= n; i++)
69         if (!dfn[i]) tarjan(i);
70
71     int ans1 = 0;
72     int ans2 = 0;
73
74     for (int u, v, i = 0; i < cntEd; i++) {
75         u = scc[e[i].u];
76         v = scc[e[i].v];
77         if (u != v) {
78             ind[v]++;
79             oud[u]++;
80         }

```

```

81     }
82     for (int i = 1; i <= sccnum; i++) {
83         if (ind[i] == 0) {
84             ans1++;
85         }
86         if (oud[i] == 0) {
87             ans2++;
88         }
89     }
90     ans2 = max(ans2, ans1);
91
92     if (sccnum == 1) ans1 = 1, ans2 = 0;
93     cout << ans1 << endl << ans2 << endl;
94
95     return 0;
96 }

```

3 基础

3.1 fastpower.cpp

```

1 //
2 // Created by acm-33 on 2019/9/19.
3 //
4
5
6 template<typename var= long long>
7 var fpow(var a, var b, var m) {
8     var ret = 1;
9     while (b) {
10         if (b & 1) ret = ret * a % m;
11         a = a * a % m;
12         b >>= 1;
13     }
14     return ret;
15 }
16
17 long long fpow(long long a, long long b, long long m) {
18     long long ret = 1;
19     while (b) {
20         if (b & 1) ret = ret * a % m;
21         a = a * a % m;
22         b >>= 1;
23     }
24     return ret;
25 }

```

3.2 prime-sieve-素数筛.cpp

```
1 //单纯求素数, 本地60ms+
2 const int MAXN = -1; //10000005
3 int prime[MAXN], pnum;
4 bool is_composite[MAXN];
5
6 void sieve(const int &n) {
7     // 1 is exception
8     for (int i = 2; i < n; ++i) {
9         if (!is_composite[i]) prime[++pnum] = i;
10        for (int j = 1; j <= pnum && i * prime[j] < n; ++j) {
11            is_composite[i * prime[j]] = true;
12            if (i % prime[j] == 0) break;
13        }
14    }
15 }
16
17 //求素数和最小素因子, 本地90ms+
18 const int MAXN = -1; //10000005
19 int prime[MAXN], pnum;
20 int min_composite[MAXN];
21
22 void sieve(const int &n) {
23     // 1 is exception
24     for (int i = 2; i < n; ++i) {
25         if (!min_composite[i]) {
26
27             prime[++pnum] = i;
28             min_composite[i] = i;
29         }
30         for (int j = 1; j <= pnum
31             && prime[j] <= min_composite[i]
32             && i * prime[j] < n; ++j) {
33             min_composite[i * prime[j]] = prime[j];
34             // if (i % prime[j] == 0) break;
35         }
36     }
37 }
```

4 字符串

4.1 Aho-Corasick-AC自动机-多模式匹配.cpp

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
```

tieway59

```
4 typedef long long ll;
5 const ll mod = 1e9 + 7;
6 const int MAXN = 500000 + 59;
7 const int inf = 1e9 + 5;
8
9 // Aho-Corasick algorithm, finite-state machine
10 template<const int NODEsetsize, const int CHARsetsize>
11 struct Aho_Corasick_FSM {
12
13     int trie[NODEsetsize][CHARsetsize], cntNd;
14     int fail[NODEsetsize];
15     int end[NODEsetsize];
16     int root;
17
18     inline int newNd() {
19         for (int i = 0; i < CHARsetsize; ++i) trie[cntNd][i] = -1;
20         end[cntNd] = 0;
21         return cntNd++;
22     }
23
24     // hash char to a proper int ID;
25     inline int hashChar(const char &c) { return c - 'a'; }
26
27     // what will be changed when reaching an end node;
28     inline void endOperation(const int &id) { end[id]++; }
29
30     inline void init() {
31         cntNd = 0;
32         root = newNd();
33     }
34
35     // insert pattern, ensure p[len-1]==0
36     inline void insert(const char p[]) {
37         int cur = root;
38         for (int j = 0, i; p[j]; ++j) {
39             i = hashChar(p[j]);
40             cur = (~trie[cur][i]) ? trie[cur][i] : trie[cur][i] = newNd();
41             // if (trie[cur][i] == -1) trie[cur][i] = newNd();
42             // cur = trie[cur][i];
43         }
44         endOperation(cur);
45     }
46
47     inline void build() {
48         int cur = root;
49         fail[root] = root;
50         queue<int> que;
51         for (int i = 0; i < CHARsetsize; ++i) {
```

```

52     if (~trie[cur][i]) {
53         fail[trie[cur][i]] = root;
54         que.push(trie[cur][i]);
55     } else {
56         trie[cur][i] = root;
57     }
58 }
59 while (!que.empty()) {
60     cur = que.front();
61     que.pop();
62
63     for (int i = 0; i < CHARsetsize; ++i) {
64         if (~trie[cur][i]) {
65             fail[trie[cur][i]] = trie[fail[cur]][i];
66             que.push(trie[cur][i]);
67         } else {
68             trie[cur][i] = trie[fail[cur]][i];
69         }
70     }
71 }
72 }
73
74 // dictionary-matching target, differs by problem
75 inline int query(const char t[]) {
76     int cur = root;
77     int res = 0;
78     for (int j = 0, i, rec; t[j]; ++j) {
79         i = hashChar(t[j]);
80         rec = cur = trie[cur][i];
81
82         // enhance recursion efficiency
83         while (rec != root && ~end[rec]) {
84             res += end[rec];
85             end[rec] = -1;
86             rec = fail[rec];
87         }
88     }
89     return res;
90 }
91
92 // void debugAc() {
93 //     for (int i = 0; i < cntNd; i++) {
94 //         printf("fail[%d] = %02d\nend[%d] = %02d\nchi[%d] = [", i, fail[i],
95 //             i, end[i], i);
96 //         for (int j = 0; j < CHARsetsize; j++)
97 //             printf("%d%c", trie[i][j], " "[j == CHARsetsize - 1]);
98 //         printf("]\n");
99 //     }

```

```

99     // }
100 };
101
102 typedef Aho_Corasick_FSM<MAXN, 26> ACFSM;
103
104 ACFSM ac;
105
106 int kase, Kase;
107 int n, k;
108 char s[1000059];
109
110 //test multi-input https://loj.ac/problem/10057
111 //test single-input https://www.luogu.org/problem/P3808
112 /*
113  * just judge the existence of some patterns.
114  *
115  */
116 int main() {
117     ios_base::sync_with_stdio(0);
118     cin.tie(0);
119     cin >> Kase;
120     while (Kase--) {
121         cin >> n;
122         ac.init();
123         for (int i = 1; i <= n; i++) {
124             cin >> s;
125             ac.insert(s);
126         }
127         ac.build();
128         cin >> s;
129         cout << ac.query(s) << '\n';
130     }
131     return 0;
132 }

```

4.2 Aho-Corasick-AC自动机-统计次数-拓扑序优化.cpp

```

1 #include <bits/stdc++.h>
2
3 #define _debug(x) cerr<<#x<<" = "<<x<<endl
4
5 using namespace std;
6
7 // Aho-Corasick algorithm, finite-state machine
8 template<const int NODEsetsize, const int CHARsetsize, const int STRsetsize>
9 struct Aho_Corasick_FSM {
10
11     int root;

```

```

12 int cntNd;
13 int trie[NODEsetsize][CHARsetsize];
14 int fail[NODEsetsize];
15 int end[NODEsetsize]; //number of strings ends at node i
16 int tag[NODEsetsize]; //times of visit j-th end.
17 int ind[NODEsetsize]; //save for topo order
18
19 int strNum;
20 int strEnd[STRsetsize]; //the i-th pattern's end node is strEnd[i];
21
22 inline int newNd() {
23     for (int i = 0; i < CHARsetsize; ++i)
24         trie[cntNd][i] = -1;
25     end[cntNd] = 0;
26     tag[cntNd] = 0;
27     return cntNd++;
28 }
29
30 // hash char to a proper int ID;
31 inline int hashChar(const char &c) { return c - 'a'; }
32
33 // what will be changed when reaching an end node;
34 inline void endOperation(const int &id) {
35     end[id]++;
36     strEnd[strNum++] = id;
37 }
38
39 inline void init() {
40     cntNd = 0;
41     strNum = 0;
42     root = newNd();
43 }
44
45 // insert pattern, ensure p[len-1]==0
46 inline void insert(const char p[]) {
47     int cur = root;
48     for (int j = 0, i; p[j]; ++j) {
49         i = hashChar(p[j]);
50         cur = (~trie[cur][i]) ? trie[cur][i] : trie[cur][i] = newNd();
51     }
52     endOperation(cur);
53 }
54
55 inline void build() {
56     int cur = root;
57     fail[root] = root;
58     queue<int> que;
59     for (int i = 0; i < CHARsetsize; ++i) {

```

```

60         if (~trie[cur][i]) {
61             fail[trie[cur][i]] = root;
62             ind[root]++; //+ topo
63             que.push(trie[cur][i]);
64         } else {
65             trie[cur][i] = root;
66         }
67     }
68     while (!que.empty()) {
69         cur = que.front();
70         que.pop();
71
72         for (int i = 0; i < CHARsetsize; ++i) {
73             if (~trie[cur][i]) {
74                 fail[trie[cur][i]] = trie[fail[cur]][i];
75                 ind[trie[fail[cur]][i]]++; //+ topo
76                 que.push(trie[cur][i]);
77             } else {
78                 trie[cur][i] = trie[fail[cur]][i];
79             }
80         }
81     }
82 }
83
84 // dictionary-matching target, differs by problem
85 inline void query(const char t[]) {
86     int cur = root;
87     for (int j = 0, i, rec; t[j]; ++j) {
88         i = hashChar(t[j]);
89         cur = trie[cur][i];
90         tag[cur]++; //+ topo
91     }
92
93     queue<int> topo;
94     for (int i = 0; i < cntNd; ++i)
95         if (!ind[i]) topo.emplace(i);
96
97     while (!topo.empty()) {
98         int u = topo.front();
99         topo.pop();
100         tag[fail[u]] += tag[u];
101         if (!--ind[fail[u]])
102             topo.emplace(fail[u]);
103     }
104
105     for (int i = 0; i < strNum; ++i) {
106         cout << tag[strEnd[i]] << '\n';
107     }

```

```

108     }
109
110 // void printAllNode() {
111 //     for (int i = 0; i < cntNd; i++) {
112 //         printf("fail[%d] = %02d\nend[%d] = %02d\nchi[%d] = [", i, fail[i], i,
113 //             end[i], i);
114 //         for (int j = 0; j < CHARsetsize; j++)
115 //             printf("%d%c", trie[i][j], " "[j == CHARsetsize - 1]);
116 //         printf("]\n");
117 //     }
118 };
119
120 typedef long long ll;
121 const int MOD = 1e9 + 7;
122 const int INF = 1e9 + 59;
123 const int MAXP = 2e5 + 59; //Pattern
124 const int MAXT = 2e6 + 59; //Target
125
126 typedef Aho_Corasick_FSM<MAXP, 26, MAXP> ACFSM;
127 ACFSM ac;
128
129 int kase, Kase;
130 int n, k;
131 char p[MAXP];
132 char t[MAXT];
133
134 /*
135  * https://www.luogu.org/problem/P5357
136  * print the times of appearance of all patterns in target.
137  */
138
139
140
141 int main() {
142     ios_base::sync_with_stdio(0);
143     cin.tie(0);
144     cin >> n;
145     ac.init();
146     for (int i = 1, j; i <= n; i++) {
147         cin >> p;
148         ac.insert(p);
149     }
150     ac.build();
151     cin >> t;
152     ac.query(t);
153
154     return 0;

```

```

155 }
156 /*
157
158
159 a
160 aa
161 aaa
162 aaaa
163 aaaaa
164
165 */

```

4.3 Levenshtein-Distance-编辑距离.py

```

1 import math
2
3 # https://www.jianshu.com/p/a617d20162cf
4 def Levenshtein_Distance(str1, str2):
5     """
6     计算字符串 str1 和 str2 的编辑距离
7     :param str1
8     :param str2
9     :return:
10    """
11    matrix = [[i + j for j in range(len(str2) + 1)] for i in range(len(str1) + 1)]
12
13    for i in range(1, len(str1) + 1):
14        for j in range(1, len(str2) + 1):
15            if (str1[i - 1] == str2[j - 1]):
16                d = 0
17            else:
18                d = 1
19
20            matrix[i][j] = min(matrix[i - 1][j] + 1, matrix[i][j - 1] + 1, matrix[i - 1][j - 1] + d)
21
22    return matrix[len(str1)][len(str2)]
23
24 # num's bit format.
25 def bindigits(num, bits):
26     s = bin(num & int("1"*bits, 2))[2:]
27     return ("{:0>{}s}" % (bits)).format(s)
28
29 if __name__ == "__main__":
30
31
32     for i in range(0, 255):
33         if (Levenshtein_Distance("1010", bindigits(i, 4)) > 2):

```

```

34     print(bindigits(i, 4))
35     print(Levenshtein_Distance("10101010", bindigits(i, 4)))

```

4.4 manacher-单数组马拉车.cpp

```

1  /**
2   * @Source: https://codeforces.com/contest/1326/submission/73675730
3   * @Author: tourist
4   * @Complexity: O(n)
5   * @Description:
6   *     得到经过填充长2n-1的回文半径数组, 填充模式为: abc
7   *     + 由于串实际没有被修改, 常数喜人
8   *     + 同时适配 char* 与 string, 各取所爱
9   *     + 你不满意可以改成全局变量数组, 简简单单
10    *     + 此处回文半径不含中心
11    *
12    * @Example:
13    *     char s[] = "123321";
14    *     vint p = manacher(6, s);
15    *     // [p] := {0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0}
16    *
17    *     string s[] = "12321";
18    *     vint p = manacher(s);
19    *     // [p] := {0, 0, 0, 0, 2, 0, 0, 0, 0}
20    *
21    * @Verification:
22    *     https://codeforces.com/contest/1326/submission/73675730
23    */
24 template<typename T>
25 vector<int> manacher(int n, const T &s) {
26     if (n == 0) {
27         return vector<int>();
28     }
29     vector<int> res(2 * n - 1, 0);
30     int l = -1, r = -1;
31     for (int z = 0; z < 2 * n - 1; z++) {
32         int i = (z + 1) >> 1;
33         int j = z >> 1;
34         int p = (i >= r ? 0 : min(r - i, res[2 * (1 + r) - z]));
35         while (j + p + 1 < n && i - p - 1 >= 0) {
36             if (!(s[j + p + 1] == s[i - p - 1])) {
37                 break;
38             }
39             p++;
40         }
41         if (j + p > r) {
42             l = i - p;
43             r = j + p;

```

```

44     }
45     res[z] = p;
46 }
47 return res;
48 }
49
50 template<typename T>
51 vector<int> manacher(const T &s) {
52     return manacher((int) s.size(), s);
53 }

```

4.5 manacher-双数组马拉车.cpp

```

1  /**
2   * @Source: https://codeforces.com/blog/entry/12143
3   * @Complexity: O(n)
4   * @Description: length of largest palindrome centered at each
5   *     character of string and between every consecutive pair
6   *     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)。
7   * @Example:
8   *     s = "123321"
9   *     [p[0], p[1]] := {0, 0, 0, 3, 0, 0} {0, 0, 0, 0, 0, 0}
10    *     s = "12321"
11    *     [p[0], p[1]] := {0, 0, 0, 0, 0} {0, 0, 2, 0, 0}
12    * @Verification:
13    *     https://codeforces.com/contest/1326/submission/73742092
14    */
15 void manacher(const string &s, vector<vector<int>> &p) {
16     int n = s.size();
17     p.assign(2, vector<int>(n, 0));
18     for (int z = 0, l = 0, r = 0; z < 2; z++, l = 0, r = 0) {
19         for (int i = 0; i < n; i++) {
20             if (i < r) p[z][i] = min(r - i + !z, p[z][1 + r - i + !z]);
21             int L = i - p[z][i], R = i + p[z][i] - !z;
22             while (L - 1 >= 0 && R + 1 < n && s[L - 1] == s[R + 1])
23                 p[z][i]++, L--, R++;
24             if (R > r) l = L, r = R;
25         }
26     }
27 }
28
29 /**
30 * @Source: https://cp-algorithms.com/string/manacher.html
31 * @Complexity: O(n)
32 * @Description: length of largest palindrome centered at each
33 *     character of string and between every consecutive pair
34 *     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)。
35 * @Example:

```

```

36 *      s = "123321"
37 *      [d1, d2] := {1, 1, 1, 1, 1, 1} {0, 0, 0, 3, 0, 0}
38 *      s = "12321"
39 *      [d1, d2] := {1, 1, 3, 1, 1, 1} {0, 0, 0, 0, 0, 0}
40 * @Verification:
41 *      https://codeforces.com/contest/1326/submission/73715067
42 */
43 void manacher(const string &s, vint &d1, vint &d2) {
44     int n = s.size();
45     d1.assign(n, 0);
46     for (int i = 0, l = 0, r = -1; i < n; i++) {
47         int k = (i > r) ? 1 : min(d1[l + r - i], r - i + 1);
48         while (0 <= i - k && i + k < n && s[i - k] == s[i + k]) {
49             k++;
50         }
51         d1[i] = k--;
52         if (i + k > r) {
53             l = i - k;
54             r = i + k;
55         }
56     }
57     d2.assign(n, 0);
58     for (int i = 0, l = 0, r = -1; i < n; i++) {
59         int k = (i > r) ? 0 : min(d2[l + r - i + 1], r - i + 1);
60         while (0 <= i - k - 1 && i + k < n && s[i - k - 1] == s[i + k]) {
61             k++;
62         }
63         d2[i] = k--;
64         if (i + k > r) {
65             l = i - k - 1;
66             r = i + k;
67         }
68     }
69 }

```

5 数学

5.1 double-compare.cpp

```

1 /* @head of double-compare modules */
2 const double EPS = 1e-8;
3
4 inline int dcmp(const double &x) {
5     if (fabs(x) < EPS) return 0;
6     else return x < EPS ? -1 : 1;
7 }
8

```

tieway59

```

9 // not necessary
10 inline bool lt(const double &x, const double &y) { return dcmp(x - y) < 0; }
11
12 inline bool le(const double &x, const double &y) { return dcmp(x - y) <= 0; }
13
14 inline bool eq(const double &x, const double &y) { return dcmp(x - y) == 0; }
15
16 inline bool ge(const double &x, const double &y) { return dcmp(x - y) >= 0; }
17
18 inline bool gt(const double &x, const double &y) { return dcmp(x - y) > 0; }
19
20 // not recommended
21 inline bool dcmp(const double &x, const string &mode, const double &y) {
22     if (mode == "lt") return dcmp(x - y) < 0;
23     if (mode == "le") return dcmp(x - y) <= 0;
24     if (mode == "eq") return dcmp(x - y) == 0;
25     if (mode == "ge") return dcmp(x - y) >= 0;
26     if (mode == "gt") return dcmp(x - y) > 0;
27     exit(0);
28 }
29 /* @tail of double-compare modules */

```

5.2 fastFactorial-快速阶乘-分块fft.cpp

```

1 // fastFactorial 快速阶乘(分块+fft)
2 // O( sqrt(n)log(n) )
3 // https://www.luogu.org/record/25477473
4 #include<cstdio>
5 #include<algorithm>
6 #include<cmath>
7
8 using namespace std;
9 typedef unsigned long long ll;
10 const ll N = 262144 + 10;
11 const int P = 65536;
12 const int SF = 16;
13 const int msk = 65535;
14 ll mod;
15 ll PP;
16 typedef long double ld;
17 const ld pi = acos(-1.0);
18
19 inline ll fpow(ll a, ll p) {
20     ll r = 1;
21     for (; p >= 1, a = a * a % mod)
22         if (p & 1) r = r * a % mod;
23     return r;
24 }

```



```

25
26 struct cmp {
27     ld r;
28     ld v;
29
30     friend cmp operator+(cmp a, cmp b) {
31         return (cmp) {a.r + b.r, a.v + b.v};
32     }
33
34     friend cmp operator-(cmp a, cmp b) {
35         return (cmp) {a.r - b.r, a.v - b.v};
36     }
37
38     friend cmp operator*(cmp a, cmp b) {
39         return (cmp) {a.r * b.r - a.v * b.v,
40                       a.r * b.v + a.v * b.r};
41     }
42
43     void operator/=(const int &len) {
44         r /= len;
45         v /= len;
46     }
47 } rt[2][22][N], tr[N],
48   tr1[N], tr2[N], tr3[N],
49   tr4[N], tr5[N], tr6[N];
50
51 int rv[22][N];
52 ll m13[N], m14[N], m23[N], m24[N];
53
54 inline void pre() {
55     for (int d = 1; d <= 18; d++)
56         for (int i = 1; i < (1 << d); i++)
57             rv[d][i] = (rv[d][i >> 1] >> 1)
58                 | ((i & 1) << (d - 1));
59
60     for (int d = 1, t = 1; d <= 18; d++, t <= 1)
61         for (int i = 0; i < (1 << d); i++)
62             rt[0][d][i] = (cmp) {cos(pi * i / t),
63                                   sin(pi * i / t)};
64
65     for (int d = 1, t = 1; d <= 18; d++, t <= 1)
66         for (int i = 0; i < (1 << d); i++)
67             rt[1][d][i] = (cmp) {cos(pi * i / t),
68                                   -sin(pi * i / t)};
69 }
70
71 inline void fft(cmp *a, int len, int d, int o) {
72     for (int i = 1; i < len; i++)

```

```

73         if (i < rv[d][i])
74             swap(a[i], a[rv[d][i]]);
75     cmp *w;
76     int i;
77     for (int k = 1, j = 1; k < len; k <= 1, j++)
78         for (int s = 0; s < len; s += (k << 1))
79             for (i = s, w = rt[o][j]; i < s + k; i++, ++w) {
80                 cmp a1 = a[i + k] * (*w);
81                 a[i + k] = a[i] - a1;
82                 a[i] = a[i] + a1;
83             }
84     if (o) for (int i = 0; i < len; i++) a[i] /= len;
85 }
86
87 inline void dbdft(ll *a, int len, int d, cmp *op1, cmp *op2) {
88     for (int i = 0; i < len; i++)
89         tr[i] = (cmp) {(ld) (a[i] >> SF),
90                       (ld) (a[i] & msk)};
91
92     fft(tr, len, d, 0);
93     tr[len] = tr[0];
94
95     for (cmp *p1 = tr, *p2 = tr + len, *p3 = op1;
96         p1 != tr + len; ++p1, --p2, ++p3)
97         (*p3) = (cmp) {p1->r + p2->r,
98                       p1->v - p2->v}
99             * (cmp) {0.5, 0};
100
101     for (cmp *p1 = tr, *p2 = tr + len, *p3 = op2;
102         p1 != tr + len; ++p1, --p2, ++p3)
103         (*p3) = (cmp) {p1->r - p2->r,
104                       p1->v + p2->v}
105             * (cmp) {0, -0.5};
106 }
107
108 inline void dbidft(cmp *tr, int len, int d, ll *a, ll *b) {
109     fft(tr, len, d, 1);
110     for (int i = 0; i < len; i++)
111         a[i] = (ll) (tr[i].r + 0.5) % mod;
112
113     for (int i = 0; i < len; i++)
114         b[i] = (ll) (tr[i].v + 0.5) % mod;
115 }
116
117 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
118 {
119     dbdft(a, len, d, tr1, tr2);
120     dbdft(b, len, d, tr3, tr4);

```

```

121 for (int i = 0; i < len; i++)
122     tr5[i] = tr1[i] * tr3[i]
123         + (cmp) {0, 1}
124         * tr2[i] * tr4[i];
125 for (int i = 0; i < len; i++)
126     tr6[i] = tr2[i] * tr3[i]
127         + (cmp) {0, 1}
128         * tr1[i] * tr4[i];
129
130 dbidft(tr5, len, d, m13, m24);
131 dbidft(tr6, len, d, m23, m14);
132
133 for (int i = 0; i < len; i++)
134     c[i] = m13[i] * PP % mod;
135 for (int i = 0; i < len; i++)
136     (c[i] += (m23[i] + m14[i]) * P + m24[i]) %= mod;
137 }
138
139 namespace iter {
140     ll f[N];
141     ll g[N];
142     ll h[N];
143     ll ifac[N];
144
145     inline void ih() {
146         ifac[0] = ifac[1] = 1;
147         for (ll i = 2; i < min(N, mod); i++)
148             ifac[i] = (mod - mod / i) * ifac[mod % i] % mod;
149         for (ll i = 1; i < min(N, mod); i++)
150             (ifac[i] *= ifac[i - 1]) %= mod;
151     }
152
153     inline void calch(ll del, int cur, ll *ip, ll *op) {
154         int d = 0;
155         int len = 1;
156         while (len <= cur + cur + cur) len <= 1, d++;
157         for (int i = 0; i <= cur; i++)
158             f[i] = ip[i] * ifac[i] % mod * ifac[cur - i] % mod;
159         for (int i = cur - 1; i >= 0; i -= 2)
160             f[i] = (mod - f[i]) % mod;
161         for (int i = 0; i <= cur + cur; i++)
162             g[i] = fpow((del + mod - cur + i) % mod, mod - 2);
163         for (int i = cur + 1; i < len; i++)
164             f[i] = 0;
165         for (int i = cur + cur + 1; i < len; i++)
166             g[i] = 0;
167
168         poly_mul(f, g, h, len, d); //卷积求出h'

```

```

169     ll xs = 1;
170     ll p1 = del - cur;
171     ll p2 = del;
172     for (ll i = p1; i <= p2; i++) (xs *= i) %= mod;
173     for (ll i = 0; i <= cur; i++, p1++, p2++) //双指针求出系数
174     {
175         op[i] = h[i + cur] * xs % mod;
176         (xs *= fpow(p1, mod - 2)) %= mod,
177         (xs *= (p2 + 1)) %= mod;
178     }
179 }
180
181 ll val[N];
182 ll fv1[N];
183 ll fv2[N];
184
185 inline void solve(int n) //倍增
186 {
187     int hb = 0;
188     for (int p = n; p; p >>= 1) hb++;
189     val[0] = 1;
190     for (int z = hb, cur = 0; z >= 0; z--) {
191         if (cur != 0) //把d乘2
192         {
193             iter::calch(cur + 1, cur, val, fv1);
194             for (int i = 0; i <= cur; i++)
195                 val[cur + i + 1] = fv1[i];
196
197             val[cur << 1 | 1] = 0;
198             iter::calch(cur * fpow(n, mod - 2) % mod,
199                 cur << 1, val, fv2);
200
201             cur <<= 1;
202             for (int i = 0; i <= cur; i++)
203                 (val[i] *= fv2[i]) %= mod;
204         }
205         if ((n >> z) & 1) //把d加1
206         {
207             for (int i = 0; i <= cur; i++)
208                 (val[i] *= (ll) (n * i) + cur + 1) %= mod;
209             cur |= 1;
210             val[cur] = 1;
211             for (int i = 1; i <= cur; i++)
212                 (val[cur] *= (ll) cur * n + i) %= mod;
213         }
214     }
215
216     int kase;

```

```

217 int main() {
218     pre();
219     int n;
220     scanf("%d", &kase);
221     while (kase--) {
222         scanf("%d%lld", &n, &mod);
223         iter::ih();//用了全局变量mod
224         int bl = sqrt(n);
225         PP = (11) P * P % mod;
226         solve(bl);
227         ll res = 1;
228         for (ll i = 0, id = 0;; i += bl, id++)//分块
229         {
230             if (i + bl > n) {
231                 for (int j = i + 1; j <= n; j++)
232                     (res *= j) %= mod;
233                 break;
234             }
235             (res *= val[id]) %= mod;
236         }
237         printf("%lld\n", res);
238     }
239     return 0;//拜拜程序~
240 }
241 /*
242 3
243 16777216 998244353
244 2333333 19260817
245 1919810 2147481811
246
247 "n and mod"
248 */
249
250

```

5.3 fft-多项式乘法.cpp

```

1 // 多项式乘法
2 // http://acm.hdu.edu.cn/showproblem.php?pid=1402
3 // https://www.luogu.com.cn/problem/P1919
4 // 来源: https://oi-wiki.org/math/poly/fft/
5 #include <math.h>
6 #include <algorithm>
7 #include <stdio.h>
8 #include <string.h>
9 #include <iostream>
10 #include <vector>
11

```

```

12 using namespace std;
13 const double PI = acos(-1.0);
14
15 struct Complex {
16     double x, y;
17
18     Complex(double _x = 0.0, double _y = 0.0) {
19         x = _x;
20         y = _y;
21     }
22
23     Complex operator-(const Complex &b) const {
24         return Complex(x - b.x, y - b.y);
25     }
26
27     Complex operator+(const Complex &b) const {
28         return Complex(x + b.x, y + b.y);
29     }
30
31     Complex operator*(const Complex &b) const {
32         return Complex(x * b.x - y * b.y, x * b.y + y * b.x);
33     }
34 };
35
36 /*
37 * 进行 FFT 和 IFFT 前的反置变换
38 * 位置 i 和 i 的二进制反转后的位置互换
39 * len 必须为 2 的幂
40 */
41 void fftChange(Complex *y, int len) {
42     for (int i = 1, j = (len >> 1); i < len - 1; i++) {
43         if (i < j) swap(y[i], y[j]);
44         int k = len >> 1;
45         while (j >= k) {
46             j = j - k;
47             k = k >> 1;
48         }
49         if (j < k) j += k;
50     }
51 }
52
53 /*
54 * 做 FFT
55 * len 必须是 2^k 形式
56 * dir == 1 时是 DFT, dir == -1 时是 IDFT
57 */
58 void fft(Complex y[], int len, int dir) {
59     fftChange(y, len);

```

```

60     for (int h = 2; h <= len; h <= 1) {
61         Complex wn(cos(2.0 * PI / h), sin(dir * 2.0 * PI / h));
62         // Omega ^ n
63         for (int j = 0; j < len; j += h) {
64             Complex w(1, 0);
65             for (int k = j; k < j + h / 2; k++) {
66                 Complex u = y[k];
67                 Complex t = w * y[k + h / 2];
68                 y[k] = u + t;
69                 y[k + h / 2] = u - t;
70                 w = w * wn;
71             }
72         }
73     }
74
75     if (dir == -1) {
76         for (int i = 0; i < len; i++) {
77             y[i].x /= len;
78         }
79     }
80 }
81
82 inline int fftLength(int len1, int len2) {
83     int len = 1;
84     while (len < len1 * 2 || len < len2 * 2) {
85         len <<= 1;
86     }
87     return len;
88 }
89
90 // rewrite
91 void fftAssign(Complex *x, int len, char *s, int slen) {
92     for (int i = 0; i < slen; i++) {
93         double v = s[slen - 1 - i] - '0';
94         x[i] = Complex(v, 0.0);
95     }
96     for (int i = slen; i < len; i++) {
97         x[i] = Complex(0.0, 0.0);
98     }
99 }
100
101 // rewrite
102 void fftMul(Complex x1[], Complex x2[], int len) {
103     for (int i = 0; i < len; i++)
104         x1[i] = x1[i] * x2[i];
105 }
106
107 const int MAXN = 2.1e6 + 59; // 尽量去取到2的幂。

```

```

108 Complex x2[MAXN];
109 Complex x1[MAXN];
110 char str1[MAXN / 2];
111 char str2[MAXN / 2];
112 int sum[MAXN];
113
114 int main() {
115     scanf("%s%s", str1, str2);
116     int len1 = strlen(str1);
117     int len2 = strlen(str2);
118     int len = fftLength(len1, len2);
119
120     fftAssign(x1, len, str1, len1);
121     fftAssign(x2, len, str2, len2);
122     fft(x1, len, 1);
123     fft(x2, len, 1);
124     fftMul(x1, x2, len);
125     fft(x1, len, -1);
126
127     for (int i = 0; i < len; i++) {
128         sum[i] = int(x1[i].x + 0.5);
129     }
130
131     for (int i = 0; i < len; i++) {
132         sum[i + 1] += sum[i] / 10;
133         sum[i] %= 10;
134     }
135
136     len = len1 + len2 - 1;
137     while (sum[len] == 0 && len > 0)
138         len--;
139
140     for (int i = len; i >= 0; i--) {
141         printf("%c", sum[i] + '0');
142     }
143
144     printf("\n");
145
146     return 0;
147 }

```

5.4 扩展CRT.py

```

1 # https://ac.nowcoder.com/acm/contest/890/D
2 # maybe not available.
3
4
5 ai = [0]

```

```

6 bi = [0]
7
8
9 def exgcd(a, warrior, x, y):
10     if warrior == 0:
11         x = 1
12         y = 0
13         return a, x, y
14     gcd, x, y = exgcd(warrior, a % warrior, x, y)
15     tp = x
16     x = y
17     y = tp - (a // warrior) * y
18     return gcd, x, y
19
20
21 def excrt():
22     m = bi[1]
23     ans = ai[1]
24     for i in range(2, num + 1):
25         x = 0
26         y = 0
27         aa = m
28         bb = bi[i]
29         c = (ai[i] - ans % bb + bb) % bb
30         gcd, x, y = exgcd(aa, bb, x, y)
31         bg = bb // gcd
32         if c % gcd != 0:
33             return -1
34
35         x = x * (c // gcd) % bg
36         ans = ans + x * m
37         m = m * bg
38         ans = (ans % m + m) % m
39     return (ans % m + m) % m
40
41
42 def main():
43     global num
44     num, m = map(int, input().split())
45     # num, m = int(input())
46     for i in range(1, num + 1):
47         ub, ua = map(int, input().split())
48         bi.append(ub)
49         ai.append(ua)
50     ans = excrt()
51     if ans == -1:
52         print("he was definitely lying")
53     else:

```

```

54         if ans <= m:
55             print(ans)
56         else:
57             print("he was probably lying")
58
59
60 if __name__ == '__main__':
61     main()

```

5.5 矩阵快速幂+大十进制指数版.cpp

```

1 #define _debug(x) cerr<<#x<<" = "<<x<<endl
2
3 #include <bits/stdc++.h>
4
5 using namespace
6 std;
7 typedef long long ll;
8
9
10 template<
11 typename _Tp,
12 const int MAXMatrixSize
13 >
14
15 struct Matrix {
16     _Tp m[MAXMatrixSize][MAXMatrixSize];
17     _Tp mod = 0;
18
19     Matrix() {
20         memset(m, 0, sizeof m);
21     }
22
23     Matrix(int _mod) : mod(_mod) {
24         memset(m, 0, sizeof m);
25     }
26
27     void init1() {
28         /*this = Matrix(mod);
29         set(0, 0, 1);
30         set(1, 1, 1);
31         for (int i = 0; i < MAXMatrixSize; i++)
32             m[i][i] = 1;
33     }
34
35     inline void set(const int
36
37     &r, const int &c, const _Tp &v) { this->m[r][c] = v; }

```

```

38 inline _Tp get(const int
39
40 &r, const int &c) { return this->m[r][c]; }
41
42 inline void setMod(const _Tp
43
44 &_mod) { this->mod = _mod; }
45
46 inline Matrix operator
47 *(
48 const Matrix t
49 ) {
50     Matrix res(mod);//= Matrix(mod);
51     res.setMod(mod);
52     for (int i = 0; i < MAXMatrixSize; i++)
53         for (int j = 0; j < MAXMatrixSize; j++)
54             for (int k = 0; k < MAXMatrixSize; k++)
55                 res.m[i][j] = (res.m[i][j] + m[i][k] * t.m[k][j]) % mod;
56     return res;
57 }
58 };
59
60 typedef Matrix<ll, 2> mat;
61
62 mat A, B;
63 ll mo, len;
64 char n[1000059];
65
66 inline mat fpow(mat base, ll exp) {
67     mat res(mo);
68     res.init1();
69     while (exp) {
70         if (exp & 1) res = res * base;
71         exp >>= 1;
72         base = base * base;
73     }
74     return res;
75 }
76
77 inline ll calc() {
78
79     len = strlen(n);
80     //reverse(n, n + len);
81
82     mat res(mo);
83     res.init1();
84     mat base = B;
85

```

```

86     for (int i = len - 1; i >= 0; --i) {
87         if (n[i] > '0')
88             res = res * fpow(base, n[i] - '0');
89         base = fpow(base, 10);
90     }
91
92     res = A * res;
93     return res.get(0, 0);
94 }
95
96 //https://ac.nowcoder.com/acm/contest/885/B
97 /*
98  * input n is a long char string.(1e6)
99  * mo is global Mod.
100  * other parameters are just Matrix elements.
101  *
102  */
103 ll x0, x1, a, b;
104
105 int main() {
106
107     scanf("%lld%lld%lld%lld", &x0, &x1, &a, &b);
108     scanf("%s %lld", n, &mo);
109
110     A = mat(mo);
111     A.set(0, 0, x0);
112     A.set(0, 1, x1);
113
114     B = mat(mo);
115     B.set(0, 0, 0);
116     B.set(0, 1, b);
117     B.set(1, 0, 1);
118     B.set(1, 1, a);
119
120     printf("%lld\n", calc());
121     return 0;
122 }
123
124 /*
125
126
127
128
129 * */
130

```

6 数据结构

6.1 zhuxishu-SegKth.cpp

```
1 //
2 // Created by acm-33 on 2019/7/24.
3 //
4
5 #define _debug(x) cerr<<#x<<" = "<<x<<endl
6
7 #include <bits/stdc++.h>
8
9 using namespace std;
10
11 typedef long long ll;
12 const ll LINF = 0x3f3f3f3f3f3f3f3f;
13 const ll INF = 0x3f3f3f3f3f3f3f3f;
14 //const int MAXN = 3000 + 59;
15 const ll MOD = 998244353;
16 const int MAXN = 100015;
17
18 const int M = MAXN * 30;
19 int n, q, m, tot;
20 int a[MAXN], t[MAXN];
21 int T[MAXN], lson[M], rson[M], c[M];
22
23 void Init_hush() {
24     for (int i = 1; i <= n; i++)
25         t[i] = a[i];
26     sort(t + 1, t + 1 + n);
27     m = unique(t + 1, t + 1 + n) - t - 1;
28 }
29
30 int build(int l, int r) {
31     int root = tot++;
32     c[root] = 0;
33     if (l != r) {
34         int mid = (l + r) >> 1;
35         lson[root] = build(l, mid);
36         rson[root] = build(mid + 1, r);
37     }
38     return root;
39 }
40
41 int hush(int x) {
42     return lower_bound(t + 1, t + 1 + m, x) - t;
43 }
44
45 int update(int root, int pos, int val) {
46     int newroot = tot++, tmp = newroot;
47     c[newroot] = c[root] + val;
```

```
48     int l = 1, r = m;
49     while (l < r) {
50         int mid = (l + r) >> 1;
51         if (pos <= mid) {
52             lson[newroot] = tot++;
53             rson[newroot] = rson[root];
54             newroot = lson[newroot];
55             root = lson[root];
56             r = mid;
57         } else {
58             rson[newroot] = tot++;
59             lson[newroot] = lson[root];
60             newroot = rson[newroot];
61             root = rson[root];
62
63             l = mid + 1;
64         }
65         c[newroot] = c[root] + val;
66     }
67     return tmp;
68 }
69
70 int query(int left_root, int right_root, int k) {
71     int l = 1, r = m;
72     while (l < r) {
73         int mid = (l + r) >> 1;
74         if (c[lson[left_root]] - c[lson[right_root]] >= k) {
75             r = mid;
76             left_root = lson[left_root];
77             right_root = lson[right_root];
78         } else {
79             l = mid + 1;
80             k -= c[lson[left_root]] - c[lson[right_root]];
81             left_root = rson[left_root];
82             right_root = rson[right_root];
83         }
84     }
85     return l;
86 }
87
88 ll Seg_k(int l, int r, int k) {
89     if (k > r - l + 1) return -1;
90     return 1ll * t[query(T[l], T[r + 1], k)];
91 }
92
93 int main() {
94
95     while (scanf("%d%d", &n, &q) == 2) {
```

```

96     tot = 0;
97     for (int i = 1; i <= n; i++)
98         scanf("%d", &a[i]);
99     Init_hush();
100    T[n + 1] = build(1, m);
101    for (int i = n; i; i--) {
102        int pos = hush(a[i]);
103        T[i] = update(T[i + 1], pos, 1);
104    }
105    while (q--) {
106        int l, r, k;
107        scanf("%d%d%d", &l, &r, &k);
108        printf("%lld\n", Seg_k(l, r, k));
109    }
110 }
111 return 0;
112 }
113
114 /*
115 5 5
116 5 3 4 1 2
117 1 2 2
118 1 2 1
119 1 5 3
120 1 5 4
121 1 5 6
122
123 */
124
125 /*
126
127
128
129
130
131 */

```

6.2 ZTC's-Splay.cpp

```

1  int root, cntN;
2  #define nd node[now]
3  struct SNODE
4  {
5      int val, cnt, par, siz, ch[2];
6  } node[MAXN];
7  void update_siz(int x)
8  {
9      if (x)

```

tieway59

```

18     node[x].siz = (node[x].ch[0] ? node[node[x].ch[0]].siz : 0) +
19                 (node[x].ch[1] ? node[node[x].ch[1]].siz : 0) + node[x].cnt;
20 }
21 bool chk(int x) { return node[node[x].par].ch[1] == x; }
22 void rorate(int x)
23 {
24     int y = node[x].par, z = node[y].par, k = chk(x), d = node[x].ch[k ^ 1];
25     printf("&&&d,%d,%d,%d&&", x, y, z, d);
26     node[y].ch[k] = d;
27     node[d].par = y;
28     node[z].ch[chk(y)] = x;
29     node[x].par = z;
30     node[x].ch[k ^ 1] = y;
31     node[y].par = x;
32     update_siz(y);
33     update_siz(x);
34 }
35 void splay(int x, int to = 0)
36 {
37     if (x == 0)
38     {
39         assert(false);
40         return;
41     }
42     while (node[x].par != to)
43     {
44         if (node[node[x].par].par == to)
45             rorate(x);
46         else if (chk(x) == chk(node[x].par))
47             rorate(node[x].par), rorate(x);
48         else
49             rorate(x), rorate(x);
50         printf("<%d,%d,%d>", x, node[x].par, to);
51         printf("$$$d$$", node[1].ch[1]);
52     }
53     if (to == 0)
54         root = x;
55 }
56 void Insert(int x)
57 {
58     if (root == 0)
59     {
60         int now = ++cntN;
61         nd.val = x;
62         root = now;
63         nd.cnt = 1;
64         nd.siz = 1;
65         nd.par = nd.ch[0] = nd.ch[1] = 0;

```



```

58     return;
59 }
60 int now = root, fa = 0;
61 while (1)
62 {
63     printf("(%d,%d,%d)", now, nd.val, nd.ch[1]);
64     if (x == nd.val)
65     {
66         nd.cnt++;
67         update_siz(now);
68         update_siz(fa);
69         splay(now);
70         return;
71     }
72     printf("22");
73     fa = now;
74     now = nd.ch[nd.val < x];
75     if (now == 0)
76     {
77         now = ++cntN;
78         nd.cnt = nd.siz = 1;
79         nd.ch[0] = nd.ch[1] = 0;
80         node[fa].ch[x > node[fa].val] = now;
81         printf("{%d,%d,%d}", fa, x > node[fa].val, now);
82         printf("$$$d$$", node[1].ch[1]);
83         nd.par = fa;
84         nd.val = x;
85         update_siz(fa);
86         splay(now);
87         return;
88     }
89 }
90 }
91 int rnk(int x)
92 {
93     int now = root, ans = 0;
94     while (now)
95     {
96         printf("[%d,%d,%d,%d]", now, nd.val, nd.ch[0], nd.ch[1]);
97         if (x < nd.val)
98             now = nd.ch[0];
99         else
100         {
101             ans += node[nd.ch[0]].siz;
102             if (x == nd.val)
103             {
104                 splay(now);
105                 return ans + 1;

```

```

106     }
107     ans += nd.cnt;
108     now = nd.ch[1];
109 }
110 }
111 return -1;
112 }
113 int kth(int x)
114 {
115     int now = root;
116     if (nd.siz < x)
117         return -1;
118     while (1)
119     {
120         if (nd.ch[0] && node[nd.ch[0]].siz >= x)
121             now = nd.ch[0];
122         else
123         {
124             int tmp = node[nd.ch[0]].siz + nd.cnt;
125             if (x <= tmp)
126                 return nd.val;
127             x -= tmp;
128             now = nd.ch[1];
129         }
130     }
131 }
132
133 int main()
134 {
135     int num, m;
136     scanf("%d%d", &num, &m);
137     for (int i = 1; i <= num; i++)
138     {
139         int x;
140         scanf("%d", &x);
141         printf("*");
142         Insert(x);
143     }
144     for (int i = 1; i <= m; i++)
145     {
146         int op, x;
147         scanf("%d%d", &op, &x);
148         if (op == 1)
149         {
150             Insert(x);
151         }
152         else if (op == 2)
153         {

```

```

154     printf("\num>>%d\num", rnk(x));
155 }
156 else if (op == 3)
157     printf("\num>>%d\num", kth(x));
158 else
159     printf("\num>>Val::%d,Siz::%d,Cnt::%d,Lc::%d,Rc::%d,Par::%d\num",
160           node[x].val,
161           node[x].siz,
162           node[x].cnt,
163           node[x].ch[0],
164           node[x].ch[1],
165           node[x].par);
166 }
167 }
168 /*
169 5 100
170 1 3 5 7 9
171 1 2
172 1 2
173 2 1
174 2 3
175 2 3
176 */

```

7 数论

7.1 Binomial-Coefficients-组合数-Lucas.cpp

```

1 const int MAXN = 1e6 + 59;
2 const int MOD = 1e9 + 7;
3 ll fac[MAXN];
4 ll inv[MAXN];
5
6 inline void initC(const int &sz) {
7     fac[0] = 1;
8     for (int i = 1; i <= sz; i++)
9         fac[i] = fac[i - 1] * i % MOD;
10    inv[sz] = fpow<ll>(fac[sz], MOD - 2, MOD);
11    // printf inv[BCSize] to get & save it;
12    for (int i = sz - 1; ~i; i--)
13        inv[i] = inv[i + 1] * (i + 1) % MOD;
14 }
15
16 inline ll C(const int &n, const int &m) {
17     return fac[n] * inv[m] % MOD * inv[n - m] % MOD;
18 }
19

```

tieway59

```

28 // Lucas
29 inline ll C(int n, int m, const int &P) {
30     ll res = 1;
31     while (n | m) res = res * C(n % P, m % P) % P, n /= P, m /= P;
32     return res;
33 }
34
35 int main() {
36     initC(1000000);
37
38     cout << C(4, 3, 1000000007) << endl;
39     cout << C(4, 1, 1000000007) << endl;
40     cout << C(5, 2, 1000000007) << endl;
41     return 0;
42 }

```

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

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

```

1 /*
2 // O(N^2)
3 // __int128
4 template<const int BCSIZE = 120, typename var = __int128>
5 //add Mod as parameter;
6 struct Binomial_Coefficient {
7     var c[BCSIZE + 1][BCSIZE + 1];
8     //Pascal's Triangle
9
10    Binomial_Coefficient() { //add Mod as parameter;
11        c[0][0] = 1;
12        for (int n = 1; n <= BCSIZE; ++n) {
13            c[n][0] = c[n][n] = 1;
14            for (int k = 1; k < n; ++k)
15                c[n][k] = (c[n - 1][k - 1] + c[n - 1][k]); //%
16        }
17    }
18
19    var operator()(const int &n, const int &m) {
20        if (n < m) return -1; //in case.
21        return c[n][m];
22    }
23 };
24 Binomial_Coefficient<> C;
25
26 */
27 //*****in normal writing style*****

```

```

28
29 const int MAXN = 20;
30 ll C[MAXN + 1][MAXN + 1];
31
32 inline void pascal(const int &maxn) {
33     C[0][0] = 1;
34     for (int n = 1; n <= maxn; ++n) {
35         C[n][0] = C[n][n] = 1;
36         for (int k = 1; k < n; ++k)
37             C[n][k] = C[n - 1][k - 1] + C[n - 1][k];
38     }
39 }
40
41 int main() {
42     /*
43     cout << C(4, 3) << endl;
44     cout << C(4, 1) << endl;
45     cout << C(5, 2) << endl;
46     */
47     cout << C[4][3] << endl;
48     cout << C[4][1] << endl;
49     cout << C[5][2] << endl;
50
51     return 0;
52 }

```

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

```

1 #define _debug(x) cerr<<#x<<" = "<<x<<endl
2
3 #include <bits/stdc++.h>
4
5 using namespace std;
6 typedef long long ll;
7
8 template<typename _Tp>
9 _Tp fpow(_Tp base, _Tp exp, _Tp Mod) {
10     _Tp res = 1;
11     while (exp) {
12         if (exp & 1) res = res * base % Mod;
13         base = base * base % Mod;
14         exp >>= 1;
15     }
16     return res;
17 }
18 /*
19
20 // O(N) O(1)

```

tieway59

```

21 template<typename _Tp, const int BCSIZE, const _Tp Mod> //add Mod as parameter;
22 struct Binomial_Coefficient {
23     _Tp fac[BCSIZE + 1];
24     _Tp inv[BCSIZE + 1];
25
26     inline Binomial_Coefficient() { //add Mod as parameter;
27         fac[0] = 1;
28         for (int i = 1; i <= BCSIZE; i++)
29             fac[i] = fac[i - 1] * i % Mod;
30
31         inv[BCSIZE] = fpow<_Tp>(fac[BCSIZE], Mod - 2, Mod);
32         // printf inv[BCSIZE] to get & save it;
33
34         for (int i = BCSIZE - 1; ~i; i--)
35             inv[i] = inv[i + 1] * (i + 1) % Mod;
36     }
37
38     inline _Tp operator()(const int &n, const int &m) {
39         if (n < m) {
40             cerr << "**** n>m " << endl;
41             return -1;
42         } //in case.
43         return fac[n] * inv[m] % Mod * inv[n - m] % Mod;
44     }
45 };
46
47 typedef Binomial_Coefficient<long long, 10000000, 1000000007> zuHeShu;
48 zuHeShu C = zuHeShu();
49
50 /*
51
52 //*****in normal writing style*****
53
54 const int MAXN = 1e6 + 59;
55 const int MOD = 1e9 + 7;
56 ll fac[MAXN];
57 ll inv[MAXN];
58
59 inline void initC(const int &sz) {
60     fac[0] = 1;
61     for (int i = 1; i <= sz; i++)
62         fac[i] = fac[i - 1] * i % MOD;
63     inv[sz] = fpow<ll>(fac[sz], MOD - 2, MOD);
64     // printf inv[BCSIZE] to get & save it;
65     for (int i = sz - 1; ~i; i--)
66         inv[i] = inv[i + 1] * (i + 1) % MOD;
67 }
68

```

```

69 inline ll C(const int &n, const int &m) {
70     return fac[n] * inv[m] % MOD * inv[n - m] % MOD;
71 }
72
73 int main() {
74
75     initC(100000);
76
77     cout << C(4, 3) << endl;
78     cout << C(4, 1) << endl;
79     //cout << C(2, 5) << endl;
80     cout << C(5, 2) << endl;
81
82
83     return 0;
84 }
85 /*
86
87
88
89
90 * */

```

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

```

1  ll exGCD(ll a, ll b, ll &x, ll &y) {
2      if (b == 0) {
3          x = 1;
4          y = 0;
5          return a;
6      }
7      ll gcd = exGCD(b, a % b, x, y);
8      ll old_x = x;
9      x = y;
10     y = old_x - (a / b) * x;
11     return gcd;
12 }
13 // co-prime(a,m)
14 ll modInv(ll a, ll m) {
15     ll x, y;
16     ll g = exGCD(a, m, x, y);
17     if (g != 1) {
18         return -1;
19     } else {
20         ll res = (x % m + m) % m;
21         return res;
22     }
23 }

```

7.6 ZTC's-FFT.txt

```

1 struct CP
2 {
3     double x,y;
4     CP (double xx=0,double yy=0){x=xx;y=yy;}
5     CP operator +(const CP &warrior){return CP(x+warrior.x,y+warrior.y);}
6     CP operator -(const CP &warrior){return CP(x-warrior.x,y-warrior.y);}
7     CP operator *(const CP &warrior){return
8         ↪ CP(x*warrior.x-y*warrior.y,x*warrior.y+y*warrior.x);}
9     void print(){printf("CP.x: %f  CP.y: %f \num",x,y);}
10 }a[MAXN],warrior[MAXN];
11 int lim,bit;
12 int rev[MAXN];
13 void init_FFT(int len)
14 {
15     lim=1,bit=0;
16     while(lim<=(len))lim<=1,bit++;
17     for(int i=0;i<lim;i++)rev[i]=(rev[i>>1]>>1)|((i&1)<<(bit-1));
18 }
19 void FFT(CP *A,int mode)
20 {
21     for(int i=0;i<lim;i++)
22     {
23         if(i<rev[i])swap(A[i],A[rev[i]]);
24     }
25     for(int mid=1;mid<lim;mid<=1)
26     {
27         CP XX(cos(Pi/mid),mode*sin(Pi/mid));
28         for(int j=0;j<lim;j+=(mid<<1))
29         {
30             CP d(1,0);
31             for(int k=0;k<mid;k++,d=d*XX)
32             {
33                 CP x=A[j+k],y=d*A[j+mid+k];
34                 A[j+k]=x+y;
35                 A[j+mid+k]=x-y;
36             }
37         }
38     }
39 }

```

7.7 数论分块.cpp

```

1 /**
2  * @Source: none.
3  * @Complexity: sqrt(n)
4  * @Description: 枚举从1开始到x的所有分块区间。

```

```

5  * @Example: none.
6  * @Verification:
7  *   https://codeforces.ml/gym/101174/C
8  */
9
10 for (ll l = 1, r = 0; l <= x; l = r + 1) {
11     r = x / (x / l);
12     // ...
13 }

```

8 杂项

8.1 coutf.cpp

```

1  /**
2  * @Source: https://zh.cppreference.com/w/cpp/language/parameter_pack
3  * @Complexity:
4  * @Description: 用cout模仿格式化输出
5  * @Example: see below
6  * @Verification: TODO
7  */
8  void coutf(const char *format) {
9      std::cout << format;
10 }
11
12 template<typename T, typename... Targs>
13 //void coutf(const char *format, T value, Targs... Fargs) // 递归变参函数
14 void coutf(const char *format, const T &value, const Targs &... Fargs) {
15     for (; *format != '\0'; format++) {
16         if (*format == '%') {
17             std::cout << value;
18             coutf(format + 1, Fargs...); // 递归调用
19             return;
20         }
21         std::cout << *format;
22     }
23 }
24
25 /**
26 void example(){
27     coutf("% world% %\n", "Hello", '!', 123);
28     cout.precision(9);
29     fixed(cout);
30     coutf("% % % %\n", 0x3f, 1.2 / 7, acos(-1), 22.3);
31 }
32 */

```

8.2 debug-from-tourist.cpp

```

1  /**
2  *
3  *
4  *
5  *
6  *
7  *
8  *
9  * @Author: TieWay59
10 * @Created: 2019/11/22 21:39
11 * @Link: https://atcoder.jp/contests/agc040/submissions/8558491
12 * @Tags:
13 *
14  *****/
15
16
17 #include <bits/stdc++.h>
18
19 // #define debug(x) cerr << #x << " = "<< x << endl
20
21 #define endl '\n'
22 #define STOPSYNC ios::sync_with_stdio(false); cin.tie(nullptr)
23 #define MULTIKASE int Kase=0; cin>>Kase; for(int kase=1; kase<=Kase; kase++)
24 typedef long long ll;
25 const int MAXN = 2e5 + 59;
26 const int MOD = 1e9 + 7;
27 const int INF = 0x3F3F3F3F;
28 const ll llINF = 0x3F3F3F3F3F3F3F3F;
29 using namespace std;
30
31
32 // debug start
33 template<typename A, typename B>
34 string to_string(pair<A, B> p);
35
36 template<typename A, typename B, typename C>
37 string to_string(tuple<A, B, C> p);
38
39 template<typename A, typename B, typename C, typename D>
40 string to_string(tuple<A, B, C, D> p);
41
42 string to_string(const string &s) {
43     return "'" + s + "'";
44 }
45
46 string to_string(const char *s) {

```

```

47     return to_string((string) s);
48 }
49
50 string to_string(bool b) {
51     return (b ? "true" : "false");
52 }
53
54 string to_string(vector<bool> v) {
55     bool first = true;
56     string res = "{}";
57     for (int i = 0; i < static_cast<int>(v.size()); i++) {
58         if (!first) {
59             res += ", ";
60         }
61         first = false;
62         res += to_string(v[i]);
63     }
64     res += "}";
65     return res;
66 }
67
68 template<size_t N>
69 string to_string(bitset<N> v) {
70     string res = "";
71     for (size_t i = 0; i < N; i++) {
72         res += static_cast<char>('0' + v[i]);
73     }
74     return res;
75 }
76
77 template<typename A>
78 string to_string(A v) {
79     bool first = true;
80     string res = "{}";
81     for (const auto &x : v) {
82         if (!first) {
83             res += ", ";
84         }
85         first = false;
86         res += to_string(x);
87     }
88     res += "}";
89     return res;
90 }
91
92 template<typename A, typename B>
93 string to_string(pair<A, B> p) {
94     return "(" + to_string(p.first) + ", " + to_string(p.second) + ")";

```

```

95 }
96
97 template<typename A, typename B, typename C>
98 string to_string(tuple<A, B, C> p) {
99     return "(" + to_string(get<0>(p)) + ", " + to_string(get<1>(p)) + ", " +
100         to_string(get<2>(p)) + ")";
101 }
102
103 template<typename A, typename B, typename C, typename D>
104 string to_string(tuple<A, B, C, D> p) {
105     return "(" + to_string(get<0>(p)) + ", " + to_string(get<1>(p)) + ", " +
106         to_string(get<2>(p)) + ", " +
107         to_string(get<3>(p)) + ")";
108 }
109
110 void debug_out() { cerr << endl; }
111
112 template<typename Head, typename... Tail>
113 void debug_out(Head H, Tail... T) {
114     cerr << " " << to_string(H);
115     debug_out(T...);
116 }
117
118 #ifdef DEBUG
119 #define debug(...) cerr << "[" << #__VA_ARGS__ << "]" :=", debug_out(__VA_ARGS__)
120 #else
121 #define debug(...) 42
122 #endif
123
124 // debug end;
125
126 int main() {
127
128     int x = 10;
129     pair<int, bool> y = {11, 1};
130     vector<int> z = {1, 2, 3, 4};
131
132     debug(x, y, z);
133     set<int> a = {9, 10, 7};
134     debug(a);
135     return 0;
136 }
137 /*
138
139 */

```

8.3 fast-I0-int.cpp

```
1 inline void read(int &x) {
2     char ch;
3     bool flag = false;
4     for (ch = getchar(); !isdigit(ch); ch = getchar()) if (ch == '-') flag = true;
5     for (x = 0; isdigit(ch); x = x * 10 + ch - '0', ch = getchar());
6     x = flag ? -x : x;
7 }
8
9 inline void write(int x) {
10     static const int maxlen = 100;
11     static char s[maxlen];
12     if (x < 0) {
13         putchar('-');
14         x = -x;
15     }
16     if (!x) {
17         putchar('0');
18         return;
19     }
20     int len = 0;
21     for (; x /= 10) s[len++] = x % 10 + '0';
22     for (int i = len - 1; i >= 0; --i) putchar(s[i]);
23 }
24
25
26 namespace Fast_IO { //orz laofu
27     const int MAXL((1 << 18) + 1);
28     int iof, iotp;
29     char ioif[MAXL], *ioiS, *ioiT, ioof[MAXL], *iooS = ioof, *iooT = ioof + MAXL - 1,
30         ↪ ioc, iost[55];
31
32     char Getchar() {
33         if (ioiS == ioiT) {
34             ioiS = ioif;
35             ioiT = ioiS + fread(ioif, 1, MAXL, stdin);
36             return (ioiS == ioiT ? EOF : *ioiS++);
37         } else return (*ioiS++);
38     }
39
40     void Write() {
41         fwrite(ioof, 1, iooS - ioof, stdout);
42         iooS = ioof;
43     }
44
45     void Putchar(char x) {
46         *iooS++ = x;
47         if (iooS == iooT) Write();
48     }
49 }
```

```
47 }
48
49 inline int read() {
50     int x = 0;
51     for (iof = 1, ioc = Getchar(); (ioc < '0' || ioc > '9') && ioc != EOF;)
52         iof = ioc == '-' ? -1 : 1, ioc = Getchar();
53     if (ioc == EOF) Write(), exit(0);
54     for (x = 0; ioc <= '9' && ioc >= '0'; ioc = Getchar()) x = (x << 3) + (x << 1)
55         ↪ + (ioc ^ 48);
56     return x * iof;
57 }
58
59 inline long long read_ll() {
60     long long x = 0;
61     for (iof = 1, ioc = Getchar(); (ioc < '0' || ioc > '9') && ioc != EOF;)
62         iof = ioc == '-' ? -1 : 1, ioc = Getchar();
63     if (ioc == EOF) Write(), exit(0);
64     for (x = 0; ioc <= '9' && ioc >= '0'; ioc = Getchar()) x = (x << 3) + (x << 1)
65         ↪ + (ioc ^ 48);
66     return x * iof;
67 }
68
69 void Getstr(char *s, int &l) {
70     for (ioc = Getchar(); ioc == ' ' || ioc == '\n' || ioc == '\t';) ioc =
71         ↪ Getchar();
72     if (ioc == EOF) Write(), exit(0);
73     for (l = 0; !(ioc == ' ' || ioc == '\n' || ioc == '\t' || ioc == EOF); ioc =
74         ↪ Getchar()) s[l++] = ioc;
75     s[l] = 0;
76 }
77
78 template<class Int>
79 void Print(Int x, char ch = '\0') {
80     if (!x) Putchar('0');
81     if (x < 0) Putchar('-'), x = -x;
82     while (x) iost[++iotp] = x % 10 + '0', x /= 10;
83     while (iotp) Putchar(iost[iotp--]);
84     if (ch) Putchar(ch);
85 }
86
87 void Putstr(const char *s) { for (int i = 0, n = strlen(s); i < n;
88     ↪ ++i) Putchar(s[i]); }
89 } // namespace Fast_IO
90 using namespace Fast_IO;
```

8.4 fast-I0-快速版(可敲).cpp

```
1 //没实现负数
2 const int BUF_SIZE = (int) 1e4 + 10;
3
4 struct fastIO {
5     char buf[BUF_SIZE];
6     int cur;
7     FILE *in, *out;
8
9     fastIO() {
10         cur = BUF_SIZE;
11         in = stdin;
12         out = stdout;
13     }
14
15     inline char nC() {
16         if (cur == BUF_SIZE) {
17             fread(buf, BUF_SIZE, 1, in);
18             cur = 0;
19         }
20         return buf[cur++];
21     }
22
23     inline bool id(char a) { return a >= '0' && a <= '9'; }
24
25     inline int nI() {
26         char c;
27         while (!id(c = nC()));
28         int x = c - '0';
29         while (id(c = nC())) x = ((x + (x << 2)) << 1) + c - '0';
30         return x;
31     }
32
33     inline ll nll() {
34         char c;
35         while (!id(c = nC()));
36         ll x = c - '0';
37         while (id(c = nC())) x = ((x + (x << 2)) << 1) + c - '0';
38         return x;
39     }
40
41     inline void pC(char ch) {
42         buf[cur++] = ch;
43         if (cur == BUF_SIZE) {
44             fwrite(buf, BUF_SIZE, 1, out);
45             cur = 0;
46         }
47     }
```

tieway59

```
48
49     inline void pI(int x) {
50         if (x > 9) pI(x / 10);
51         pC(x % 10 + '0');
52     }
53
54     inline void close() { if (cur) fwrite(buf, cur, 1, out), cur = 0; }
55 } IO;
```

8.5 fast-I0-极致版.ignore

8.6 fastpow-快速幂.cpp

```
1 ll fpow(ll a, ll b, ll mod = MOD) {
2     if (a % mod == 0) return 0;
3     ll ret = 1;
4     a %= mod;
5     while (b) {
6         if (b & 1) ret = ret * a % mod;
7         a = a * a % mod;
8         b >>= 1;
9     }
10    return ret;
11 }
```

8.7 Misc-杂技-随机数.cpp

```
1 mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());
2
3 inline int suiJi(const int &l, const int &r) {
4     return uniform_int_distribution<int>(l, r)(rng);
5 }
```

8.8 string-read-speed.cpp

```
1 const int MAXN = 5e7 + 59;
2 char buffer[MAXN];
3 vector<char> buf(MAXN);
4 string s;
5 stringstream ss;
6
7 void solve(int kaseId = -1) {
8     /*
9         freopen("text.in", "w+", stdout);
10     */
```



```

11     for (int i = 1; i <= 50000000; i++) {
12         cout << (char) suiJi('a', 'z');
13     }
14     */
15     /*
16     // 278.912500 ms
17     freopen("text.in", "r+", stdin);
18     double begin = chrono::steady_clock::now().time_since_epoch().count();
19     getline(cin, s);
20     double stoped = chrono::steady_clock::now().time_since_epoch().count();
21     printf("%.6f ms\n", (double) (stoped - begin) / 1000000.0);
22     */
23     /*
24     // 290.965400 ms ~ 300
25     freopen("text.in", "r+", stdin);
26     double begin = chrono::steady_clock::now().time_since_epoch().count();
27     cin >> s;
28     double stoped = chrono::steady_clock::now().time_since_epoch().count();
29     printf("%.6f ms\n", (double) (stoped - begin) / 1000000.0);
30     */
31     /*
32     // 235.966700 ms
33     freopen("text.in", "r+", stdin);
34     double begin = chrono::steady_clock::now().time_since_epoch().count();
35     gets(s);
36     double stoped = chrono::steady_clock::now().time_since_epoch().count();
37     printf("%.6f ms\n", (double) (stoped - begin) / 1000000.0);
38     */
39     /*
40     // 99.795400 ms
41     freopen("text.in", "r+", stdin);
42     //FILE *fp = fopen("text.in", "r");
43     double begin = chrono::steady_clock::now().time_since_epoch().count();
44     fread(s, sizeof(char), 50000000, stdin);
45     double stoped = chrono::steady_clock::now().time_since_epoch().count();
46     printf("%.6f ms\n", (double) (stoped - begin) / 1000000.0);
47     */
48     /*
49     // 1749.292200 ms
50     freopen("text.in", "r+", stdin);
51     //FILE *fp = fopen("text.in", "r");
52     double begin = chrono::steady_clock::now().time_since_epoch().count();
53     scanf("%s", buffer);
54     double stoped = chrono::steady_clock::now().time_since_epoch().count();
55     printf("%.6f ms\n", (double) (stoped - begin) / 1000000.0);
56     */
57     /*
58     // 90.939200 ms

```

```

59     freopen("text.in", "r+", stdin);
60     double begin = chrono::steady_clock::now().time_since_epoch().count();
61     fread(buf.data(), sizeof(char), 50000000, stdin);
62     double stoped = chrono::steady_clock::now().time_since_epoch().count();
63     printf("%.6f ms\n", (double) (stoped - begin) / 1000000.0);
64     */
65     /*
66     cin.get() 与 getchar 读法效率差很多，在此不表。
67     */
68     */
69 }

```

8.9 timetest.cpp

```

1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 int main() {
6
7     clock_t begin = clock();
8     int x = 0;
9     for (int i = 1; i <= 800000000; ++i) {
10         x++;
11     }
12     printf("%.3f ms\n", (double) (clock() - begin));
13     return 0;
14 }

```

8.10 unordered-map-自写哈希.cpp

```

1 struct custom_hash {
2     static uint64_t splitmix64(uint64_t x) {
3         // http://xorshift.di.unimi.it/splitmix64.c
4         x += 0x9e3779b97f4a7c15;
5         x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
6         x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
7         return x ^ (x >> 31);
8     }
9
10    size_t operator()(uint64_t x) const {
11        static const uint64_t FIXED_RANDOM =
12            chrono::steady_clock::now().time_since_epoch().count();
13        return splitmix64(x + FIXED_RANDOM);
14    }
15 };

```

```

15 unordered_map<long long, int, custom_hash> safe_map;
16

```

8.11 单调队列-定长区间最值.cpp

```

1  #define _debug(x) cerr<<#x<<" = "<<x<<endl
2
3  #include <iostream>
4  #include <algorithm>
5  #include <deque>
6
7  using namespace std;
8
9  typedef long long ll;
10
11 const int INF = 0x3f3f3f3f;
12 const int MOD = 998244353;
13 const int MAXN = 1e6 + 59;
14
15 int Kase, n, m;
16
17 int a[MAXN];
18 int ans1[MAXN], ans2[MAXN];
19 deque<int> qMAX, qMIN;
20
21 int main() {
22     ios_base::sync_with_stdio(0);
23     cin.tie(0);
24
25     cin >> n >> m;
26
27     for (int i = 1; i <= n; i++) {
28         cin >> a[i];
29     }
30
31     for (int i = 1; i <= n; i++) {
32         while (!qMIN.empty() && i - qMIN.front() >= m)
33             qMIN.pop_front();
34         while (!qMAX.empty() && i - qMAX.front() >= m)
35             qMAX.pop_front();
36
37         while (!qMIN.empty() && a[qMIN.back()] > a[i])
38             qMIN.pop_back();
39
40         while (!qMAX.empty() && a[qMAX.back()] < a[i])
41             qMAX.pop_back();
42
43         if (qMIN.empty() || a[qMIN.back()] <= a[i])

```

```

44         qMIN.push_back(i);
45
46         if (qMAX.empty() || a[qMAX.back()] >= a[i])
47             qMAX.push_back(i);
48
49         if (i >= m) {
50             ans1[i] = a[qMIN.front()];
51             ans2[i] = a[qMAX.front()];
52         }
53     }
54     for (int j = m; j <= n; ++j) {
55         cout << ans1[j] << " \n"[j == n];
56     }
57     for (int j = m; j <= n; ++j) {
58         cout << ans2[j] << " \n"[j == n];
59     }
60     return 0;
61 }
62
63 /*
64
65     2
66     2 0
67     1 2
68     1 1
69
70     3 2
71     1 2 1
72     5 3 1
73     1 5 1
74
75
76 */
77

```