Data Structure

BIT

```
1
    // start from 1
 2
    #include <bits/stdc++.h>
 3
    using namespace std;
    using ll = long long;
    const 11 MAXN = 100005;
    11 tree[MAXN];
 7
    11 lowbit(int x) { return (x) & (-x); };
 8
    void Update(int i, ll x) {
 9
      // increase
10
       for (int pos = i; pos <= MAXN; pos += lowbit(pos)) {</pre>
         tree[pos] += x;
11
12
13
    }
    11 PrefixQuery(int n) {
14
15
      11 \text{ ret} = 0;
       for (int pos = n; pos; pos -= lowbit(pos)) {
16
17
         ret += tree[pos];
18
       }
19
      return ret;
20
21
    11 RangeQuery(int ql, int qr) { return PrefixQuery(qr) - PrefixQuery(ql - 1); }
    int main() {
22
       int a[10] = \{-1, 4, 2, 1, 5, 6, 7, 2, 1, 4\};
23
       for (int i = 1; i \le 9; i++) {
24
25
         Update(i, a[i]);
26
27
       for (int i = 1; i \leftarrow 9; i++) {
28
         cout << PrefixQuery(i) << endl;</pre>
29
       }
30
       return 0;
31
    }
32
```

Mono Queue

```
#include <bits/stdc++.h>
// monotonic descending queue, segMax at front
using namespace std;

void getSegMax(vector<int>& v, int k, vector<int>& ans) {
    deque<int> que;
    int n = v.size();
    for (int i = 0; i + 1 < k; ++i) {</pre>
```

```
9
         while (!que.empty() && v[que.back()] <= v[i]) que.pop_back();</pre>
10
         que.push_back(i);
11
       }
12
       for (int i = k - 1; i < n; ++i) {
13
         while (!que.empty() && v[que.back()] <= v[i]) que.pop_back();</pre>
14
         que.push back(i);
         while (que.front() <= i - k) que.pop_front();</pre>
15
16
         ans.push_back(v[que.front()]);
17
       }
18
    }
    void getSegMin(vector<int>& v, int k, vector<int>& ans) {
19
20
       deque<int> que;
21
       int n = v.size();
       for (int i = 0; i + 1 < k; ++i) {
22
23
         while (!que.empty() && v[que.back()] >= v[i]) que.pop_back();
24
         que.push_back(i);
25
       for (int i = k - 1; i < n; ++i) {
26
         while (!que.empty() && v[que.back()] >= v[i]) que.pop_back();
27
28
         que.push_back(i);
29
         while (que.front() <= i - k) que.pop_front();</pre>
         ans.push_back(v[que.front()]);
30
31
       }
32
33
    int main() {
34
       vector\langle int \rangle v = {2, 3, 1, 4, 5, 6, 7, 3};
35
       vector<int> ans;
36
       getSegMin(v, 3, ans);
37
       for (auto itm: ans) {
38
         cout << itm << " ";</pre>
39
       }
40
       return 0;
41
    }
```

Segment Tree Range

```
1 #include <iostream>
    using namespace std;
    using ll = long long;
    const int MAXN = 200005;
    struct Node {
7
      // TODO modify to fit the need
8
      11 1, r;
9
      11 ans, mulv, addv;
10
      Node() {}
11
12
    Node tree[MAXN << 2];
13
    11 n, m, q, rawValues[MAXN];
14
    void MergeNode(Node &f, const Node &lc, const Node &rc) {
15
16
      // TODO VARY based on different problems
17
      f.ans = (lc.ans + rc.ans) % m;
      f.addv = 0;
18
19
      f.mulv = 1;
20
21
    void NodeAdd(int k, ll addv) {
22
```

```
23
    }
24
    void NodeMul(int k, ll mulv) {
25
26
27
    void SpreadTag(Node &f, Node &sn) {
      // TODO VARY based on different problems
28
      11 addv = f.addv, mulv = f.mulv;
29
      sn.ans = (sn.ans * mulv % m + (sn.r - sn.l + 1) % m * addv % m) % m;
30
      sn.mulv = sn.mulv * mulv % m;
31
32
      sn.addv = (sn.addv * mulv % m + addv) % m;
33
34
    void PushUp(int k) { // up a level
35
      MergeNode(tree[k], tree[k \langle\langle 1 | 1 \rangle\rangle;
36
37
    void PushDown(int k) { // push the lazy tag down a level
      if (!(tree[k].addv == 0 && tree[k].mulv == 1)) {
38
39
         SpreadTag(tree[k], tree[k << 1]);</pre>
         SpreadTag(tree[k], tree[k << 1 | 1]);</pre>
40
         // TODO reset father's lazy tag
41
42
        tree[k].addv = 0;
43
        tree[k].mulv = 1;
      }
44
45
    void BuildTree(int k, int l, int r) {
46
47
      // prepare the nodes
48
      tree[k].l = 1;
49
      tree[k].r = r;
50
      if (1 == r) {
51
         // TODO VARY based on different problems
52
         tree[k].ans = rawValues[1];
53
        tree[k].addv = 0;
54
        tree[k].mulv = 1;
55
      } else {
         int mid = 1 + (r - 1) / 2;
56
57
         BuildTree(k << 1, 1, mid);</pre>
         BuildTree(k \langle\langle 1 \mid 1, mid + 1, r \rangle\rangle;
58
59
         PushUp(k);
60
      }
61
62
    void UpdateSegMul(int k, int l, int r, ll mulv) {
63
      if (1 <= tree[k].1 && tree[k].r <= r) {
64
         // TODO VARY based on problems
65
         // record the operation for query with smaller range
66
         tree[k].ans = tree[k].ans * mulv % m;
67
         tree[k].mulv = tree[k].mulv * mulv % m;
68
        tree[k].addv = tree[k].addv * mulv % m;
69
      } else {
70
         PushDown(k);
71
         int mid = tree[k].1 + (tree[k].r - tree[k].1) / 2;
72
         if (mid >= 1) // separated update
73
           UpdateSegMul(k << 1, 1, r, mulv);</pre>
74
         if (mid < r) UpdateSegMul(k << 1 | 1, 1, r, mulv);</pre>
75
         PushUp(k);
76
77
78
    void UpdateSegAdd(int k, int l, int r, ll addv) {
79
      if (1 <= tree[k].1 && tree[k].r <= r) {
80
         // TODO VARY based on problems
         tree[k].ans = (tree[k].ans + addv * (tree[k].r - tree[k].l + 1) % m) % m;
81
82
         tree[k].addv = (tree[k].addv + addv) % m;
83
      } else {
```

```
84
         PushDown(k);
85
         int mid = tree[k].1 + (tree[k].r - tree[k].1) / 2;
         if (mid >= 1) // separated update
86
87
           UpdateSegAdd(k << 1, 1, r, addv);</pre>
88
         if (mid < r) UpdateSegAdd(k << 1 | 1, 1, r, addv);</pre>
 89
         PushUp(k);
       }
90
 91
     void UpdateDot(int k, int pos, 11 val) {
92
93
       if (tree[k].1 == tree[k].r) {
         // TODO VARY based on problems
94
95
         // tree[k].sum = val;
96
       } else {
97
         PushDown(k);
98
         int mid = tree[k].1 + (tree[k].r - tree[k].1) / 2;
99
         if (pos <= mid) // separated update</pre>
100
           UpdateDot(k << 1, pos, val);</pre>
101
         else
           UpdateDot(k << 1 | 1, pos, val);</pre>
102
103
         PushUp(k);
104
       }
105
106
     Node Query(int k, int ql, int qr) {
       if (tree[k].l >= ql && tree[k].r <= qr) return tree[k];
107
108
       // when not single, push down firstly, then do the query
109
       PushDown(k);
       int mid = tree[k].1 + (tree[k].r - tree[k].1) / 2;
110
111
       Node resL, resR, retVal;
112
       bool hasL = false, hasR = false;
113
       if (ql <= mid) {
         hasL = true;
114
115
         resL = Query(k << 1, ql, qr);</pre>
116
117
       if (mid < qr) {
118
         hasR = true;
119
         resR = Query(k \ll 1 \mid 1, q1, qr);
120
121
       if (hasL && hasR)
122
         MergeNode(retVal, resL, resR);
123
       else if (hasL)
         retVal = resL;
124
125
       else if (hasR)
126
         retVal = resR;
127
       return retVal;
128
     }
129
     int main() {
130
       ios::sync_with_stdio(false);
131
       cin >> n >> q >> m;
132
       for (int i = 1; i <= n; i++) cin >> rawValues[i];
133
       134
       BuildTree(1, 1, n);
135
       136
       int t, 1, r, v;
137
       while (q--) {
         cin >> t >> l >> r;
138
139
         if (t == 3) {
140
           cout << Query(1, 1, r).ans << "\n";</pre>
141
         } else if (t == 1) {
142
           cin >> v;
143
           UpdateSegMul(1, 1, r, v);
144
         } else if (t == 2) {
```

```
145 | cin >> v;

146 | UpdateSegAdd(1, 1, r, v);

147 | }

148 | }

149 | return 0;

150 | }
```

Union Set

```
1
    #include <iostream>
    using namespace std;
 3
    const int MAXN = 100005;
    int father[MAXN];
 5
    int trank[MAXN];
 6
 7
    void Init(int n) {
 8
      for (int i = 0; i < n; ++i) {
        father[i] = i;
 9
10
        trank[i] = 0;
11
      }
12
13
    int Find(int x) {
      if (father[x] == x) {
14
15
        return x;
      }
16
      return father[x] = Find(father[x]);
17
18
    void Unite(int x, int y) {
19
      x = Find(x);
20
      y = Find(y);
21
22
      if (x == y) {
23
        return;
24
25
      if (trank[x] < trank[y]) {</pre>
26
        father[x] = y;
27
      } else {
        father[y] = x;
28
        if (trank[x] == trank[y]) {
29
30
           trank[x]++;
31
32
      }
33
    }
34
    bool inSame(int x, int y) \{ return Find(x) == Find(y); \}
```

Geometry

```
int sgn (double x) { // sign of a double
  if (fabs(x) < eps) return 0;
  else if (x < 0) return -1;
  else return 1;
}</pre>
```

3D Sphere

```
#include <bits/stdc++.h>
 1
    using namespace std;
 3
    const double PI = acos(-1.0);
 4
    struct Sphere {
 5
      double x, y, z, r;
 6
      Sphere() {}
 7
      Sphere(double x, double y, double z, double r): x(x), y(y), z(z), r(r) {}
8
9
    double IntersectionVolume(Sphere o, Sphere t) {
10
      // basic formula: V = (3 * r - h) * h * h * PI / 3
      // calculated from spinning surface calculus
11
12
      if (o.r < t.r) swap(o, t);
13
      double dis = sqrt((o.x - t.x) * (o.x - t.x) + (o.y - t.y) * (o.y - t.y) +
                         (o.z - t.z) * (o.z - t.z));
14
15
      if (dis <= o.r - t.r) { // completely in</pre>
        return 4.0 / 3 * PI * t.r * t.r * t.r;
16
17
      } else if (dis <= o.r) { // center of the smaller sphere in bigger sphere
18
        // \cos A = (b2 + c2 - a2) / 2bc
19
        double angleb = acos((t.r * t.r + dis * dis - o.r * o.r) / (2 * t.r * dis));
20
        double anglea = PI - angleb;
        double 1 = t.r * cos(anglea);
21
22
        double H = o.r - 1 - dis;
        double h = t.r - 1;
23
        return 4.0 / 3 * PI * t.r * t.r * t.r - PI / 3 * (3 * t.r - h) * h * h +
24
25
                PI / 3 * (3 * o.r - H) * H * H;
26
      } else if (dis < o.r + t.r) { // normal intersection</pre>
27
        double angler = acos((t.r * t.r + dis * dis - o.r * o.r) / (2 * t.r * dis));
        double angleR = acos((o.r * o.r + dis * dis - t.r * t.r) / (2 * o.r * dis));
28
29
        double H = o.r - o.r * cos(angleR);
30
        double h = t.r - t.r * cos(angler);
        return PI / 3 * (3 * t.r - h) * h * h + PI / 3 * (3 * o.r - H) * H * H;
31
32
      } else {
33
        return 0;
34
35
36
    double IntersectionSurface(Sphere &o, Sphere &t) {
      // basic formula: S = 2 * PI * r * h
37
38
      if (o.r < t.r) swap(o, t);
      double dis = sqrt((o.x - t.x) * (o.x - t.x) + (o.y - t.y) * (o.y - t.y) +
39
```

```
40
                          (o.z - t.z) * (o.z - t.z));
41
       if (dis <= o.r - t.r) { // completely in</pre>
42
         return 4 * PI * t.r * t.r;
43
      } else if (dis <= o.r) { // center of the smaller sphere in bigger sphere</pre>
        double angleb = acos((t.r * t.r + dis * dis - o.r * o.r) / (2 * t.r * dis));
44
        double anglea = PI - angleb;
45
        double 1 = t.r * cos(anglea);
46
47
        double H = o.r - l - dis;
        double h = t.r - 1;
48
        return 4 * PI * t.r * t.r - 2 * PI * t.r * h + 2 * PI * o.r * H;
49
      } else if (dis < o.r + t.r) { // normal intersection</pre>
50
        double angler = acos((t.r * t.r + dis * dis - o.r * o.r) / (2 * t.r * dis));
51
52
        double angleR = acos((o.r * o.r + dis * dis - t.r * t.r) / (2 * o.r * dis));
        double H = o.r - o.r * cos(angleR);
53
54
         double h = t.r - t.r * cos(angler);
        return 2 * PI * t.r * h + 2 * PI * o.r * H;
55
56
      } else {
57
        return 0;
58
59
    }
60
    int main() {
61
      Sphere A, B;
      cin >> A.x >> A.y >> A.z >> A.r;
62
63
      cin >> B.x >> B.y >> B.z >> B.r;
64
      cout << fixed << setprecision(10) << 4*PI*(A.r*A.r+B.r*B.r) - IntersectionSurface(A, B) << endl;</pre>
65
      return 0;
66
   }
```

2D Vector

```
1
 2
      * structs of
 3
      * point, vector, segment
 4
      * and some operator overloads
     */
 6
    // whether a seg AB intersects with a circle O?
    // see the endpoints' tangent point (P, Q) angle
    // angles: AOP + BOQ < AOB <==> intersect
 9
    #include <bits/stdc++.h>
10
    using namespace std;
    using 11 = long long;
11
    11 \text{ MOD} = 1e9 + 7;
12
13
    11 QpowMod(11 bse, 11 pwr) {
       11 \text{ ret} = 1;
14
15
       while (pwr) {
         if (pwr & 1) ret = ret * bse % MOD;
16
         bse = bse * bse % MOD;
17
18
         pwr >>= 1;
19
       }
20
       return ret;
21
22
     struct Point2 {
23
       11 x, y;
24
       Point2() : x(0), y(0) {}
       Point2(11 \_x, \ 11 \_y) \ : \ x(\_x), \ y(\_y) \ \{\}
25
       11 Norm2() { return 1ll * x * x + 1ll * y * y; }
26
27
       double Norm() { return sqrt(Norm2()); }
28
       Point2 operator+(const Point2 &po) {
```

```
return Point2(x + po.x, y + po.y);
29
30
31
      Point2 operator-(const Point2 &po) {
32
        // note the direction
33
        return Point2(x - po.x, y - po.y);
34
      }
35
      bool operator==(const Point2 &po) {
36
        return x == po.x && y == po.y;
37
      }
38
    };
39
    typedef Point2 Vector2;
    struct Segment2 {
40
41
      Point2 s, e;
42
      Segment2() {}
43
      Segment2(Point2 _s, Point2 _e) : s(_s), e(_e) {}
44
45
    11 MulCross(const Point2 &p1, const Point2 &p2) {
      return p1.x * p2.y - p1.y * p2.x;
46
47
48
    11 MulDot(const Point2 &p1, const Point2 &p2) {
49
      return p1.x * p2.x + p1.y * p2.y;
50
51
    double DisPointToSeg(Point2 p, Point2 s1, Point2 s2) {
52
      Point2 v1 = p - s1, v2 = s2 - s1;
53
      if (MulDot(v2, v1) < 0 | MulDot(v2, v1) > v2.Norm2())
54
        return min(1.0 * (p - s1).Norm(), 1.0 * (p - s2).Norm());
      return abs(1.0 * MulCross(v2, v1) / v2.Norm());
55
56
57
    int Dis2PointToSeg_INT(Point2 p, Point2 s1, Point2 s2) {
58
      // square of distance between two points
      Point2 v = p - s1, u = s2 - s1;
59
60
      if (MulDot(u, v) < 0 \mid MulDot(u, v) > u.Norm2())
        return min((p - s1).Norm2(), (p - s2).Norm2()) % MOD;
61
      return ((MulCross(v, u) % MOD) * (MulCross(v, u) % MOD)) % MOD *
62
63
             QpowMod(u.Norm2() % MOD, MOD - 2) % MOD;
64
65
    int main() { return 0; }
```

3D Vector

```
#include <bits/stdc++.h>
 2
    using namespace std;
    using ll = long long;
    11 \text{ MOD} = 1e9 + 7;
 4
    struct Point3 {
 6
      11 x, y, z;
 7
      Point3(): x(0), y(0), z(0) {}
      Point3(11_x, 11_y, 11_z): x(_x), y(_y), z(_z) {}
 8
 9
      11 Norm2() { return x * x + y * y + z * z; }
10
      double Norm() { return sqrt(Norm2()); }
11
      Point3 operator+(const Point3 &po) {
12
        return Point3(x + po.x, y + po.y, z + po.z);
13
14
      Point3 operator-(const Point3 &po) {
15
        return Point3(x - po.x, y - po.y, z - po.z);
```

```
16
      bool operator==(const Point3 &po) {
17
18
       return x == po.x && y == po.y && z == po.z;
19
    };
20
21
    typedef Point3 Vector3;
    struct Segment3 {
22
23
     Point3 s, e;
24
     Segment3() {}
25
     Segment3(Point3 _s, Point3 _e): s(_s), e(_e) {}
26
   };
    11 MulDot(const Point3 &p1, const Point3 &p2) {
27
    return p1.x * p2.x + p1.y * p2.y + p1.z * p2.z;
28
29
    Point3 MulCross(const Point3 &p1, const Point3 &p2) {
30
    return Point3(p1.y * p2.z - p1.z * p2.y, p1.z * p2.x - p1.x * p2.z, p1.x * p2.y - p1.y * p2.x);
31
32
   }
33
    int main() {
34
      Point3 a{0, 0, 1}, b{1, 1, 1};
35
      Point3 c = MulCross(a, b);
36
     cout << c.Norm() << endl;</pre>
37
      return 0;
38
   }
39
```

Math

C_n^m

```
#include <stdio.h>
 2
    using 11 = long long;
    const 11 MN = 2000000;
 4
    const 11 MOD = 1000000007;
    int fac[MN + 5], inv[MN + 5];
 6
 7
    11 qpowMod(ll bse, ll pwr) {
 8
      11 \text{ ret} = 1;
 9
       while (pwr) {
         if (pwr & 1) ret = ret * bse % MOD;
10
         bse = bse * bse % MOD;
11
12
         pwr >>= 1;
       }
13
14
      return ret;
15
    void init() {
16
17
      fac[0] = 1;
       for (int i = 1; i \leftarrow MN; i++) fac[i] = 111 * fac[i - 1] * i % MOD;
18
19
      inv[MN] = qpowMod(fac[MN], MOD - 2);
      for (int i = MN - 1; i \ge 0; i--) inv[i] = 111 * <math>inv[i + 1] * (i + 1) % MOD;
20
21
    int C(int n, int m) {
22
23
     if (m > n) return 0;
       return 1ll * fac[n] * inv[m] % MOD * inv[n - m] % MOD;
24
25
26
    int main() {
27
      init();
       printf("%d\n", C(5, 3));
28
29
       return 0;
30 }
```

Euler Primers

```
#include <bits/stdc++.h>
 1
 2
    using namespace std;
 3
    using 11 = long long;
    const int MAXN = 1e6 + 5;
    const int MOD = 1e9 + 7;
    // priority_queue<11, vector<11>, greater<11>> minor_que;
    int prime[MAXN];
 9
    bool vis[MAXN];
10
    int cnt = 0;
    11 \text{ maxv} = -1;
11
12
    void EulerPrime(int n) {
13
       for (int i = 2; i <= n; ++i) {
```

```
14
        if (vis[i] == 0) {
15
           prime[cnt++] = i;
16
           vis[i] = 1;
17
18
        for (int j = 0; i * prime[j] <= n; ++j) {
19
           vis[i * prime[j]] = 1;
           if (i % prime[j] == 0) break; // key of O(n)
20
21
        }
      }
22
23
    }
24
    int main() {
25
      EulerPrime(100);
      for (int i = 0; i < cnt; ++i) printf("%d ", prime[i]);</pre>
26
27
      printf("\n");
28
      return 0;
29
```

Josephus Ring

```
// n-1 规模时留下的最后一人,与 n 规模的相差了一个偏移量 k。J_{n,k} = (J_{n-1,k} + k) mod n。 (从 0
    编号,下同,答案加一个偏移即可)
 2
    #include <cstdio>
 3
    long long josephus(int n, int k) {
 4
      if (n == 1)
 5
        return 0;
 6
 7
        return (josephus(n - 1, k) + k) \% n;
8
9
    int main(void) {
10
      long long n, k;
11
      scanf("%lld %lld", &n, &k);
12
13
      printf("%lld\n", 1 + josephus(n, k));
14
      return 0;
15
    }
    // total n, k-th out, find the m-th out, start from 1
16
17
    void solve(int casei) {
      cout << "Case #" << casei << ": ";</pre>
18
19
      long long ans = (K - 1) \% (N - M + 1);
20
      if (K == 1) {
        cout << M << endl;</pre>
21
22
        return;
23
      }
24
      for (11 i = N - M + 2; i \leftarrow N; i++) {
25
        ans = (ans + K) % i; // normal iteration
        // jump forward
26
        11 \text{ rem} = (i - ans - 1) / K;
27
        rem = min(rem, N - i); // limit the times of jump
28
29
        i += rem; // jump
30
        ans += rem * K;
31
32
      cout << ans + 1 << endl;</pre>
33
   }
```

Matrix Inverse Element

```
Inverse element of 2x2 matrix \begin{pmatrix} a & b \\ c & d \end{pmatrix} is \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}/(ad-bc).
```

Matrix Power

```
#include <bits/stdc++.h>
 1
    #define inf 0x3f3f3f3f
 2
 3
    using namespace std;
    typedef long long 11;
 5
    const int N = 205, mod = 998244353, MS = 205;
 6
    struct Mat {
 7
      11 a[MS][MS];
 8
      11 n, m;
9
      Mat(int n = 0, int m = 0) : n(n), m(m) \{ memset(a, 0, sizeof(a)); \}
      Mat operator*(const Mat& B) const {
10
11
        Mat C(n, B.m);
12
        for (int i = 1; i <= n; i++)
           for (int j = 1; j <= B.m; j++)
13
14
             for (int k = 1; k \leftarrow m; k++)
15
               C.a[i][j] = (C.a[i][j] + a[i][k] * B.a[k][j]) % mod;
16
        return C;
17
18
    };
19
    Mat gpow(Mat a, int n) {
20
     Mat ans(a.n, a.n);
21
      for (int i = 1; i \leftarrow a.n; i++) ans.a[i][i] = 1;
22
      for (; n; n >>= 1, a = a * a)
23
        if (n \& 1) ans = ans * a;
24
      return ans;
25
    }
26
    int main() {
27
     11 n;
28
      cin >> n;
29
      string s;
30
      cin >> s;
31
      11 \text{ now = stol(s)};
      Mat A(100, 100);
32
33
      A = qpow(A, n);
34
35
      Mat B(100, 100);
36
      B.a[1][1] = 1;
37
      B = B * A;
38
      cout << B.a[1][now];</pre>
39
   }
```

Quick Power

```
#include <cstdio>
// a^(-1) mod p => a^(p - 2) mod p

// n * n * (n + 1) * (n + 1) / 4 = \sum_{1}^{n} i^3

// n * (n + 1) * (2n + 1) / 6 = \sum_{1}^{n} i^2

using ll = long long;
```

```
6 | 11 MOD = 1e9+7;
7
   11 QpowMod(ll bse, ll pwr) {
8
     ll ret = 1;
     while (pwr) {
9
      if (pwr & 1) ret = ret * bse % MOD;
10
      bse = bse * bse % MOD;
11
      pwr >>= 1;
12
13
    }
14
    return ret;
15 }
   int main() {
16
    printf("%11d", QpowMod(2, 199) * 6 % MOD);
17
18
    return 0;
19 }
```

Graph

SCC kosaraju

```
#include <cstdio>
 2
    #include <stack>
    using namespace std;
    stack<int> stk;
    // adjacent matrix
    int mp[10][10];
    // reversed graph
    int mpt[10][10];
 8
 9
    int vst[10];
10
    int clr[10];
    int vn, en;
11
12
    void dfs1(int s) {
13
      if (vst[s] == 1) return;
14
       vst[s] = 1;
15
       // dfs routine
       for (int i = 1; i \leftarrow vn; ++i) {
16
         if (mp[s][i] < 0x3f3f3f3f) {</pre>
17
18
           dfs1(i);
19
         }
20
       }
21
       // push
22
       stk.push(s);
23
24
    void dfs2(int s, int cnt) {
25
       if (vst[s] == 0) return;
       clr[s] = cnt;
26
27
       vst[s] = 0;
28
       for (int i = 1; i \leftarrow vn; ++i) {
         if (mpt[s][i] < 0x3f3f3f3f) {</pre>
29
30
           dfs2(i, cnt);
31
         }
32
33
34
     void init() {
35
       for (int i = 1; i \leftarrow vn; ++i) {
36
         for (int j = 1; j \leftarrow vn; ++j) {
           mp[i][j] = mp[j][i] = 0x3f3f3f3f;
37
38
           mpt[i][j] = mpt[j][i] = 0x3f3f3f3f;
39
         mpt[i][i] = mp[i][i] = 0;
40
41
       }
42
     void SCC_kor() {
43
44
       for (int i = 1; i <= vn; ++i) {
         if (vst[i] == 0) dfs1(i);
45
46
47
       int cnt = 1;
48
       while (!stk.empty()) {
49
         int s = stk.top();
50
         stk.pop();
```

```
51
        if (vst[s] == 0) continue;
52
        dfs2(s, cnt++);
53
      }
54
      // vertexes with same value in clr[] is in one SCC
      for (int i = 1; i \le vn; ++i) {
55
56
        printf("%d ", clr[i]);
57
      printf("\n");
58
59
60
    int main() {
      scanf("%d %d", &vn, &en);
61
62
      init();
63
      for (int i = 1; i <= en; ++i) {
        int fr, to;
64
        scanf("%d %d", &fr, &to);
65
        mp[fr][to] = 1;
66
67
        mpt[to][fr] = 1;
      }
68
69
      SCC_kor();
70
      return 0;
71
   }
```

SCC tarjan

```
#include <bits/stdc++.h>
 1
 2
    using namespace std;
 3
    int n, m;
    struct node {
 4
      vector<int> nxt;
    } g[100000];
 6
 7
    int dfn[100000], low[100000], d[100000], col[100000], cnt[100000], stk[100000];
    int vis[100000];
    int top, deep, colour;
10
    void tarjan(int u) {
      dfn[u] = low[u] = ++deep;
11
      stk[top++] = u;
12
13
      vis[u] = 1;
      for (int i = 0; i < g[u].nxt.size(); i++) {</pre>
14
15
        int v = g[u].nxt[i];
        if (!vis[v]) {
16
17
          tarjan(v);
18
          low[u] = min(low[v], low[u]);
19
        } else {
20
           low[u] = min(low[v], low[u]);
21
        }
22
      }
23
      if (dfn[u] == low[u]) {
24
        int node;
25
        colour++;
26
        while (node != u) {
27
           node = stk[top - 1];
28
           top--;
29
           col[node] = colour;
30
31
32
   }
```

String

KMP

```
int nxt[100005];
 2
    char t[100005];
 3
    void getNxt() {
 4
      nxt[0] = -1;
      int k = -1, j = 0;
      while (t[j] != '\0') {
 6
 7
        if (k == -1 || t[k] == t[j]) {
          nxt[++j] = ++k;
 8
 9
        } else {
10
           k = nxt[k];
11
12
13
   }
```

Manarcher

```
// find the palindrome in O(n)
    #include <bits/stdc++.h>
    using namespace std;
    char s[100005];
    int ps = 0;
    int p[100005], ctr, maxr, mirr;
 7
    void solve() {
 8
       ctr = maxr = 0;
 9
       for (int i = 0; i < ps; ++i) {
10
         mirr = 2 * ctr - i;
         if (i < maxr) {</pre>
11
12
           p[i] = min(maxr - i, p[mirr]);
13
         } else {
           p[i] = 0;
15
         while (s[i - 1 - p[i]] == s[i + 1 + p[i]]) {
16
17
           p[i]++;
18
         }
         if (p[i] + i > maxr) {
19
20
          ctr = i;
21
           maxr = p[i] + i;
22
         }
23
24
       int maxi = 0;
25
       for (int i = 0; i < ps; ++i) {
         maxi = p[maxi] < p[i] ? i : maxi;</pre>
26
27
       printf("%d\n", p[maxi]);
28
       for (int i = maxi - p[maxi]; i \leftarrow maxi + p[maxi]; ++i) {
29
30
         if (s[i] != '#') {
```

```
31
        printf("%c", s[i]);
32
      }
33
     }
    printf("\n");
34
35
36
   int main() {
37
     int Case = 1;
38
     while (Case--) {
39
       char c = getchar();
40
       s[ps++] = '#';
       while (c != '\n') {
41
42
        s[ps++] = c;
43
        s[ps++] = '#';
44
        c = getchar();
45
      }
46
       solve();
47
     }
48
    return 0;
49 }
```

Misc

fastIO

```
namespace GTI
 1
 2
         char gc(void)
 3
 4
 5
             const int S=1<<17;</pre>
             static char buf[S],*s=buf,*t=buf;
 6
 7
             if (s==t) t=buf+fread(s=buf,1,S,stdin);
             if (s==t) return EOF;
 8
 9
             return *s++;
10
         int gti(void)
11
12
13
             int a=0,b=1,c=gc();
14
             for (;!isdigit(c);c=gc()) b^=(c=='-');
             for (;isdigit(c);c=gc()) a=a*10+c-'0';
15
16
             return b?a:-a;
17
         }
18
    };
19
```

Discretization

```
1
    namespace GTI
 2
 3
         char gc(void)
 4
 5
             const int S=1<<17;</pre>
             static char buf[S],*s=buf,*t=buf;
 6
             if (s==t) t=buf+fread(s=buf,1,S,stdin);
             if (s==t) return EOF;
 8
 9
             return *s++;
         }
10
         int gti(void)
11
12
             int a=0,b=1,c=gc();
13
             for (;!isdigit(c);c=gc()) b^=(c=='-');
14
15
             for (;isdigit(c);c=gc()) a=a*10+c-'0';
16
             return b?a:-a;
17
         }
18
    };
```

Inverse Pair Merge Sort

```
1
    using ll = long long;
 2
    11 \text{ MAXN} = 2e5 + 5;
 3
    11 n, q[MAXN], tmp[MAXN];
 4
    // [1, r]
 5
    11 merge_sort(int 1, int r) {
 6
      if (1 >= r) return 0;
 7
      11 \text{ mid} = (1 + r) >> 1;
      11 res = merge_sort(1, mid) + merge_sort(mid + 1, r);
 9
10
      11 k = 0, i = 1, j = mid + 1;
      while (i <= mid && j <= r) {
11
12
        if (q[i] <= q[j])
13
           tmp[k++] = q[i++];
14
        else {
15
          tmp[k++] = q[j++];
           res += mid - i + 1;
16
17
        }
      }
18
19
      while (i <= mid) tmp[k++] = q[i++];
      while (j \le r) tmp[k++] = q[j++];
20
      for (ll i = 1, j = 0; i <= r; i++, j++) q[i] = tmp[j];
21
22
      return res;
23
   }
```

Modui

```
/**
 1
 2
     * Modui range number of distinct values
 3
    #include <bits/stdc++.h>
 5
    using namespace std;
    #define endl "\n";
 6
 7
    #define IOS_ONLY
     ios::sync_with_stdio(false); \
 8
 9
     cin.tie(0);
10
      cout.tie(0);
11
    const int MAXN = 30005, MAXQ = 200005, MAXM = 1000005;
12
    int sq;
13
    struct Query {
14
     int ql, qr, id;
15
      bool operator<(const Query &o) const {</pre>
        // sqrt(n) partitions, assign sq with sqrt(n) first
16
        if (ql / sq != o.ql / sq) return ql < o.ql;</pre>
17
        if (ql / sq & 1) return qr < o.qr; // order by parity
18
        return qr > o.qr;
19
      }
20
21
    } Q[MAXQ];
    int A[MAXN], ans[MAXQ], Cnt[MAXM], cur, pl = 1, pr = 0, n;
22
23
   inline void add(int pos) {
24
      if (Cnt[A[pos]] == 0) cur++;
25
      Cnt[A[pos]]++;
26
    }
```

```
27
    inline void del(int pos) {
28
      Cnt[A[pos]]--;
      if (Cnt[A[pos]] == 0) cur--;
29
30
    int main() {
31
32
      IOS_ONLY
33
      cin >> n;
34
      sq = sqrt(n);
      for (int i = 1; i \leftarrow n; ++i) cin >> A[i];
35
36
      int q;
37
      cin >> q;
      for (int i = 0; i < q; ++i) { // offline query
38
39
        cin >> Q[i].ql >> Q[i].qr;
40
        Q[i].id = i;
41
42
      sort(Q, Q + q); // sort, KEY of modui
43
      for (int i = 0; i < q; ++i) {
44
        while (pl > Q[i].ql) add(--pl);
45
        while (pr < Q[i].qr) add(++pr);
46
        while (pl < Q[i].ql) del(pl++);
47
        while (pr > Q[i].qr) del(pr--);
        ans[Q[i].id] = cur; // store the rasult
48
49
50
      for (int i = 0; i < q; ++i) cout << ans[i] << endl;</pre>
51
      return 0;
   }
52
```