南京大学 ACM-ICPC 集训队代码模版库

Contents

1 General

1.1 Code library checksum

```
c502  import re, sys, hashlib
427e
b41f  def digest_line(s):
d74e      return hashlib.md5(re.sub(r'\s|//.*', '', s)).hexdigest()[−4:]
427e
f7db  for line in sys.stdin.read().strip().split("\n"):
f335      print digest_line(line), line
```

1.2 .vimrc

```
914c  set nocompatible
733d  syntax on
6bbc  colorscheme slate
7db5  set number
b0e3  set cursorline
061b  set shiftwidth=2
8011  set softtabstop=2
a66d  set tabstop=2
d23a  set expandtab
5245  set magic
740c  set smartindent
bee8  set backspace=indent,eol,start
815d  set cmdheight=1
0a40  set laststatus=2
e458  set statusline=\ %<%F[%1*%M%*%n%R%H]%=\ %y\ %0(%{&fileformat}\ %{&encoding}\ %c
        :%l/%L%)\
1c67  set whichwrap=b,s,<,>,[,]
```

1.3 Template

```
302f  #include <bits/stdc++.h>
421c  using namespace std;
427e
426f  #ifdef __LOCAL_DEBUG__
3507  # define _debug(fmt, ...) fprintf(stderr, "\033[94m%s:␣" fmt "\n\033[0m", \
611f      __func__, ##__VA_ARGS__)
a8cb  #else
```

```
# define _debug(...) ((void) 0)                                 e6b5
#endif                                                          1937
#define rep(i, n) for (int i=0; i<(n); i++)                     0d6c
#define Rep(i, n) for (int i=1; i<=(n); i++)                    cfe3
#define range(x) (x).begin(), (x).end()                         8843
typedef long long LL;                                           5cad
typedef unsigned long long ULL;                                 b773
                                                                427e
template <unsigned p>                                           5120
struct Zp{                                                      87b8
    unsigned x;                                                 7797
    Zp(unsigned x):x(x){}                                       ff67
    operator unsigned(){return x;}                              22e3
    Zp operator ^ (ULL e) {                                     fecc
        Zp b=x, r=1;                                            4fce
        while (e) {                                             3e90
            if (e&1) r=r*b;                                     5421
            b=b*b;                                              2059
            e>>=1;                                              16fc
        }                                                       95cf
        return r;                                               547e
    }                                                           95cf
    Zp operator + (Zp rhs) {return (x+rhs)%p;}                  a2f5
    Zp operator − (Zp rhs) {return (x+p−rhs)%p;}                664b
    Zp operator * (Zp rhs) {return x*rhs%p;}                    3ec4
    Zp operator / (Zp rhs) {return Zp(x)*(rhs^(p−2));}          7cfd
};                                                              329b
                                                                427e
typedef Zp<1000000007> zp;                                      370f
                                                                427e
zp operator"" _ (ULL n){return n;}                              0795
```

2 Miscellaneous Algorithms

2.1 Fast fourier transform

```
const int NMAX = 1<<20;                                         4e09
                                                                427e
typedef complex<double> cplx;                                   3fbf
                                                                427e
const double PI = 2*acos(0.0);                                  abd1
struct FFT{                                                     12af
c47c     int rev[NMAX];
27d7     cplx omega[NMAX], oinv[NMAX];
9827     int K, N;
427e
1442     FFT(int k){
e209         K = k; N = 1 << k;
b393         rep (i, N){
7ba3             rev[i] = (rev[i>>1]>>1) | ((i&1)<<(K-1));
1908             omega[i] = polar(1.0, 2.0 * PI / N * i);
a166             oinv[i] = conj(omega[i]);
95cf         }
95cf     }
427e
b941     void dft(cplx* a, cplx* w){
a215         rep (i, N) if (i < rev[i]) swap(a[i], a[rev[i]]);
ac6e         for (int l = 2; l <= N; l *= 2){
2969             int m = l/2;
b3cf             for (cplx* p = a; p != a + N; p += l)
c24f                 rep (k, m){
fe06                     cplx t = w[N/l*k] * p[k+m];
ecbf                     p[k+m] = p[k] - t; p[k] += t;
95cf                 }
95cf         }
95cf     }
427e
617b     void fft(cplx* a){dft(a, omega);}
a123     void ifft(cplx* a){
3b2f         dft(a, oinv);
57fc         rep (i, N) a[i] /= N;
95cf     }
427e
bdc0     void conv(cplx* a, cplx* b){
6497         fft(a); fft(b);
12a5         rep (i, N) a[i] *= b[i];
f84e         ifft(a);
95cf     }
329b };
```

2.2 2-SAT

```
0f42 const int MAXN = 100005;
03a9 struct twoSAT{
5c83     int n;
```

```
    vector<int> G[MAXN*2];                                   8f72
    bool mark[MAXN*2];                                       d060
    int S[MAXN*2], c;                                        b42d
                                                             427e
    void init(int n){                                        d34f
        this->n = n;                                         b985
        for (int i=0; i<n*2; i++) G[i].clear();              f9ec
        memset(mark, 0, sizeof(mark));                       0609
    }                                                        95cf
                                                             427e
    bool dfs(int x){                                         3bd5
        if (mark[x^1]) return false;                         bd70
        if (mark[x]) return true;                            c96a
        mark[x] = true;                                      fd23
        S[c++] = x;                                          4bea
        for (int i=0; i<G[x].size(); i++)                    1ce6
            if (!dfs(G[x][i])) return false;                 d942
        return true;                                         3361
    }                                                        95cf
                                                             427e
    void add_clause(int x, bool xval, int y, bool yval){     5894
        x = x * 2 + xval;                                    6afe
        y = y * 2 + yval;                                    e680
        G[x^1].push_back(y);                                 81cc
        G[y^1].push_back(x);                                 6835
    }                                                        95cf
                                                             427e
    bool solve() {                                           d0cb
        for (int i=0; i<n*2; i+=2){                          7c39
            if (!mark[i] && !mark[i+1]){                     e63f
                c = 0;                                       88fb
                if (!dfs(i)){                                f4b9
                    while (c > 0) mark[S[--c]] = false;      3f03
                    if (!dfs(i+1)) return false;             86c5
                }                                            95cf
            }                                                95cf
        }                                                    95cf
        return true;                                         3361
    }                                                        95cf
                                                             427e
    inline bool value(unsigned i){return mark[2*i+1];}       5f0a
};                                                           329b
```

2.3 Knuth's optimization

```
5c83 int n;
d77c int dp[256][256], dc[256][256];
427e
b7ec template <typename T>
0bc7 void compute(T cost) {
0423   for (int i = 0; i <= n; i++) {
8f5e     dp[i][i] = 0;
9488     dc[i][i] = i;
95cf   }
be8e   rep (i, n) {
95b5     dp[i][i+1] = 0;
aa0f     dc[i][i+1] = i;
95cf   }
ec08   for (int len = 2; len <= n; len++) {
88b8     for (int i = 0; i + len <= n; i++) {
d3da       int j = i + len;
9824       int lbnd = dc[i][j-1], rbnd = dc[i+1][j];
a24a       dp[i][j] = INT_MAX / 2;
f933       int c = cost(i, j);
90d2       for (int k = lbnd; k <= rbnd; k++) {
9bd0         int res = dp[i][k] + dp[k][j] + c;
26b5         if (res < dp[i][j]) {
e6af           dp[i][j] = res;
9c88           dc[i][j] = k;
95cf         }
95cf       }
95cf     }
95cf   }
329b };
```

3 String

3.1 Knuth-Morris-Pratt algorithm

```
2836 const int SIZE = 10005;
9847 int fail[SIZE];
57b7 int len;
427e
182f void construct(const char* p) {
aaa1   len = strlen(p);
  fail[0] = fail[1] = 0;                                                            3dd4
  for (int i = 1; i < len; i++) {                                                   d8a8
    int j = fail[i];                                                                147f
    while (j && p[i] != p[j]) j = fail[j];                                          3c79
    fail[i + 1] = p[i] == p[j] ? j + 1 : 0;                                         4643
  }                                                                                 95cf
}                                                                                   95cf
                                                                                    427e
inline void found(int pos) {                                                        c464
  // ! add codes for having found at pos                                            427e
}                                                                                   95cf
                                                                                    427e
void match(const char* t, const char* p) {  // must be called after construct      1932
  int n = strlen(t);                                                                8482
  int j = 0;                                                                        8fd0
  rep(i, n) {                                                                       be8e
    while (j && p[j] != t[i]) j = fail[j];                                          4e19
    if (p[j] == t[i]) j++;                                                          b5d5
    if (j == len) found(i - len + 1);                                               f024
  }                                                                                 95cf
}                                                                                   95cf
```

3.2 Manacher algorithm

```
struct Manacher {                                                                   81d4
  int Len;                                                                          cd09
  vector<int> lc;                                                                   9255
  string s;                                                                         b301
                                                                                    427e
  void work() {                                                                     ec07
    lc[1] = 1;                                                                      c033
    int k = 1;                                                                      6bef
                                                                                    427e
    for (int i = 2; i <= Len; i++) {                                                491f
      int p = k + lc[k] - 1;                                                        7957
      if (i <= p) {                                                                 5e04
        lc[i] = min(lc[2 * k - i], p - i + 1);                                      24a1
      } else {                                                                      8e2e
        lc[i] = 1;                                                                  e0e5
      }                                                                             95cf
      while (s[i + lc[i]] == s[i - lc[i]]) lc[i]++;                                 74ff
      if (i + lc[i] > k + lc[k]) k = i;                                             2b9a
    }                                                                               95cf
95cf     }
427e
bfd5     void init(const char *tt) {
aaaf       int len = strlen(tt);
f701       s.resize(len * 2 + 10);
7045       lc.resize(len * 2 + 10);
8e13       s[0] = '*';
ae54       s[1] = '#';
1321       for (int i = 0; i < len; i++) {
e995         s[i * 2 + 2] = tt[i];
69fd         s[i * 2 + 1] = '#';
95cf       }
43fd       s[len * 2 + 1] = '#';
75d1       s[len * 2 + 2] = '\0';
61f7       Len = len * 2 + 2;
3e7a       work();
95cf     }
427e
b194     pair<int, int> maxpal(int l, int r) {
901a       int center = l + r + 1;
ffb2       int rad = lc[center] / 2;
ab54       int rmid = (l + r + 1) / 2;
17e4       int rl = rmid - rad, rr = rmid + rad - 1;
3908       if ((r ^ l) & 1) {
69f3       } else rr++;
69dc       return {max(l, rl), min(r, rr)};
95cf     }
329b   };
```

3.3 Aho-corasick automaton

```
a1ad   struct AC : Trie {
9143     int fail[MAXN];
daca     int last[MAXN];
427e
8690     void construct() {
93d2       queue<int> q;
a7a6       fail[0] = 0;
ce3c       rep(c, CHARN) {
b1c6         if (int u = tr[0][c]) {
a506           fail[u] = 0;
3e14           q.push(u);
f689           last[u] = 0;
```

```
        }                                                       95cf
      }                                                         95cf
      while (!q.empty()) {                                      cc78
        int r = q.front();                                      31f0
        q.pop();                                                15dd
        rep(c, CHARN) {                                         ce3c
          int u = tr[r][c];                                     ab59
          if (!u) {                                             0ef5
            tr[r][c] = tr[fail[r]][c];                          9d58
            continue;                                           b333
          }                                                     95cf
          q.push(u);                                            3e14
          int v = fail[r];                                      b3ff
          while (v && !tr[v][c]) v = fail[v];                   d2ea
          fail[u] = tr[v][c];                                   c275
          last[u] = tag[fail[u]] ? fail[u] : last[fail[u]];     654c
        }                                                       95cf
      }                                                         95cf
    }                                                           95cf
                                                                427e
    void found(int pos, int j) {                                7752
      if (j) {                                                  043e
        // ! add codes for having found word with tag[j]        427e
        found(pos, last[j]);                                    4a96
      }                                                         95cf
    }                                                           95cf
                                                                427e
    void find(const char* text) {  // must be called after construct()   9785
      int p = 0, c, len = strlen(text);                         80a4
      rep(i, len) {                                             9c94
        c = id(text[i]);                                        b3db
        p = tr[p][c];                                           f119
        if (tag[p])                                             f08e
          found(i, p);                                          389b
        else if (last[p])                                       1e67
          found(i, last[p]);                                    299e
      }                                                         95cf
    }                                                           95cf
  };                                                            329b
```

3.4 Trie

```
  const int MAXN = 12000;                                       e6f1
```

```
dd87 const int CHARN = 26;
427e
8ff5 inline int id(char c) { return c - 'a'; }
427e
a281 struct Trie {
5c83   int n;
f4f5   int tr[MAXN][CHARN];  // Trie tree, 0 denotes fail
35a5   int tag[MAXN];
427e
4fee   Trie() {
3ccc     memset(tr[0], 0, sizeof(tr[0]));
4d52     tag[0] = 0;
46bf     n = 1;
95cf   }
427e
427e   // tag should not be 0
30b0   void add(const char* s, int t) {
d50a     int p = 0, c, len = strlen(s);
9c94     rep(i, len) {
3140       c = id(s[i]);
d6c8       if (!tr[p][c]) {
26dd         memset(tr[n], 0, sizeof(tr[n]));
2e5c         tag[n] = 0;
73bb         tr[p][c] = n++;
95cf       }
f119       p = tr[p][c];
95cf     }
35ef     tag[p] = t;
95cf   }
427e
427e   // returns 0 if not found
427e   // AC automaton does not need this function
216c   int search(const char* s) {
d50a     int p = 0, c, len = strlen(s);
9c94     rep(i, len) {
3140       c = id(s[i]);
f339       if (!tr[p][c]) return 0;
f119       p = tr[p][c];
95cf     }
840e     return tag[p];
95cf   }
329b };
```

4 Math

4.1 Matrix powermod

```
const int MAXN = 105;                                          44b4
const LL modular = 1000000007;                                 92df
int n; // order of matrices                                    5c83
                                                               427e
struct matrix{                                                 8864
    LL m[MAXN][MAXN];                                          3180
                                                               427e
    void operator *=(matrix& a){                               43c5
        static LL t[MAXN][MAXN];                               e735
        Rep (i, n){                                            34d7
            Rep (j, n){                                        4c11
                t[i][j] = 0;                                   ee1e
                Rep (k, n){                                    c4a7
                    t[i][j] += (m[i][k] * a.m[k][j]) % modular; fcaf
                    t[i][j] %= modular;                        199e
                }                                              95cf
            }                                                  95cf
        }                                                      95cf
        memcpy(m, t, sizeof(t));                               dad4
    }                                                          95cf
};                                                             329b
                                                               427e
matrix r;                                                      63d8
void m_powmod(matrix& b, LL e){                                3ec2
    memset(r.m, 0, sizeof(r.m));                               83f0
    Rep(i, n)                                                  a7c3
        r.m[i][i] = 1;                                         de64
    while (e){                                                 3e90
        if (e & 1) r *= b;                                     5a0e
        b *= b;                                                35c5
        e >>= 1;                                               16fc
    }                                                          95cf
}                                                              95cf
```

4.2 Linear basis

```
const int MAXD = 30;                                           8b44
struct linearbasis {                                           03a6
3558      ULL b[MAXD] = {};
427e
842f      bool insert(ll v) {
9b2b          for (int j = MAXD - 1; j >= 0; j--) {
de36              if (!(v & (1ll << j))) continue;
ee78              if (b[j]) v ^= b[j]
037f              else {
7836                  for (int k = 0; k < j; k++)
f0b4                      if (v & (1ll << k)) v ^= b[k];
b0aa                  for (int k = j + 1; k < MAXD; k++)
46c9                      if (b[k] & (1ll << j)) b[k] ^= v;
8295                  b[j] = v;
3361                  return true;
95cf              }
95cf          }
438e          return false;
95cf      }
329b  };
```

4.3 Gauss elimination over finite field

```
b784  const LL p = 1000000007;
427e
2a2c  LL powmod(LL b, LL e) {
95a2    LL r = 1;
3e90    while (e) {
1783      if (e & 1) r = r * b % p;
5549      b = b * b % p;
16fc      e >>= 1;
95cf    }
547e    return r;
95cf  }
427e
c130  typedef vector<LL> VLL;
42ac  typedef vector<VLL> VVLL;
427e
2c62  LL gauss(VVLL &a, VVLL &b) {
561b    const int n = a.size(), m = b[0].size();
a25e    vector<int> irow(n), icol(n), ipiv(n);
2976    LL det = 1;
427e
be8e    rep (i, n) {
d2b5      int pj = -1, pk = -1;
```

```
      rep (j, n) if (!ipiv[j])                                                6b4a
        rep (k, n) if (!ipiv[k])                                              e582
          if (pj == -1 || a[j][k] > a[pj][pk]) {                              6112
            pj = j;                                                           a905
            pk = k;                                                           657b
          }                                                                   95cf
      if (a[pj][pk] == 0) return 0;                                           d480
      ipiv[pk]++;                                                             0305
      swap(a[pj], a[pk]);                                                     8dad
      swap(b[pj], b[pk]);                                                     aad8
      if (pj != pk) det = (p - det) % p;                                      be4d
      irow[i] = pj;                                                           d080
      icol[i] = pk;                                                           f156
                                                                              427e
      LL c = powmod(a[pk][pk], p - 2);                                        4ecd
      det = det * a[pk][pk] % p;                                              865b
      a[pk][pk] = 1;                                                          c36a
      rep (j, n) a[pk][j] = a[pk][j] * c % p;                                 dd36
      rep (j, m) b[pk][j] = b[pk][j] * c % p;                                 1b23
      rep (j, n) if (j != pk) {                                               f8f3
        c = a[j][pk];                                                         e97f
        a[j][pk] = 0;                                                         c449
        rep (k, n) a[j][k] = (a[j][k] + p - a[pk][k] * c % p) % p;            820b
        rep (k, m) b[j][k] = (b[j][k] + p - b[pk][k] * c % p) % p;            f039
      }                                                                       95cf
    }                                                                         95cf
                                                                              427e
    for (int j = n - 1; j >= 0; j--) if (irow[j] != icol[j]) {                37e1
      for (int k = 0; k < n; k++) swap(a[k][irow[j]], a[k][icol[j]]);         50dc
    }                                                                         95cf
    return det;                                                               f27f
  }                                                                           95cf
```

4.4 Berlekamp-Massey algorithm

```
const LL MOD = 1000000007;                                                    2b86
                                                                              427e
LL inverse(LL b) {                                                            391d
  LL e = MOD - 2, r = 1;                                                      32d3
  while (e) {                                                                 3e90
    if (e & 1) r = r * b % MOD;                                               9a62
    b = b * b % MOD;                                                          29ea
    e >>= 1;                                                                  16fc
```

```
95cf      }
547e      return r;
95cf    }
427e
32a6    struct Poly {
afe0      vector<int> a;
427e
9794      Poly() { a.clear(); }
427e
de81      Poly(vector<int> &a) : a(a) {}
427e
8087      int length() const { return a.size(); }
427e
16de      Poly move(int d) {
b31d        vector<int> na(d, 0);
f915        na.insert(na.end(), a.begin(), a.end());
cecf        return Poly(na);
95cf      }
427e
fa1a      int calc(vector<int> &d, int pos) {
5b57        int ret = 0;
501c        for (int i = 0; i < (int)a.size(); ++i) {
5de5          if ((ret += (long long)d[pos - i] * a[i] % MOD) >= MOD) {
3041            ret -= MOD;
95cf          }
95cf        }
ee0f        return ret;
95cf      }
427e
c856      Poly operator - (const Poly &b) {
bd55        vector<int> na(max(this->length(), b.length()));
d1a7        for (int i = 0; i < (int)na.size(); ++i) {
3507          int aa = i < this->length() ? this->a[i] : 0,
2bee              bb = i < b.length() ? b.a[i] : 0;
9526          na[i] = (aa + MOD - bb) % MOD;
95cf        }
cecf        return Poly(na);
95cf      }
329b    };
427e
5473    Poly operator * (const int &c, const Poly &p) {
72de      vector<int> na(p.length());
d1a7      for (int i = 0; i < (int)na.size(); ++i) {
bf0c        na[i] = (long long)c * p.a[i] % MOD;
```

```
  }                                                                     95cf
  return na;                                                            aaab
}                                                                       95cf
                                                                        427e
vector<int> solve(vector<int> a) {                                      afff
  int n = a.size();                                                     9f23
  Poly s, b;                                                            58d0
  s.a.push_back(1), b.a.push_back(1);                                   4e8f
  for (int i = 1, j = 0, ld = a[0]; i < n; ++i) {                       c2aa
    int d = s.calc(a, i);                                               4158
    if (d) {                                                            d503
      if ((s.length() - 1) * 2 <= i) {                                  c29d
        Poly ob = b;                                                    db9d
        b = s;                                                          6bce
        s = s - (long long)d * inverse(ld) % MOD * ob.move(i - j);      1d0e
        j = i;                                                          0889
        ld = d;                                                         64f1
      } else {                                                          8e2e
        s = s - (long long)d * inverse(ld) % MOD * b.move(i - j);       714e
      }                                                                 95cf
    }                                                                   95cf
  }                                                                     95cf
  // Caution: s.a might be shorter than expected                        427e
  return s.a;                                                           e235
}                                                                       95cf
```

4.5 Fast Walsh-Hadamard transform

```
void fwt(int* a, int n){                                                061e
    for (int d = 1; d < n; d <<= 1)                                     5595
        for (int i = 0; i < n; i += d << 1)                             05f2
            rep (j, d){                                                 b833
                int x = a[i+j], y = a[i+j+d];                           7796
                // a[i+j] = x+y, a[i+j+d] = x-y;    // xor              427e
                // a[i+j] = x+y;                    // and              427e
                // a[i+j+d] = x+y;                  // or               427e
            }                                                           95cf
}                                                                       95cf
                                                                        427e
void ifwt(int* a, int n){                                               4db1
    for (int d = 1; d < n; d <<= 1)                                     5595
        for (int i = 0; i < n; i += d << 1)                             05f2
            rep (j, d){                                                 b833
7796                  int x = a[i+j], y = a[i+j+d];
427e                  // a[i+j] = (x+y)/2, a[i+j+d] = (x-y)/2;    // xor
427e                  // a[i+j] = x-y;                           // and
427e                  // a[i+j+d] = y-x;                         // or
95cf              }
95cf  }
427e
2ab6  void conv(int* a, int* b, int n){
950a      fwt(a, n);
e427      fwt(b, n);
8a42      rep(i, n) a[i] *= b[i];
430f      ifwt(a, n);
95cf  }
```

4.6 Number theoretic transform

```
4ab9  const int NMAX = 1<<21;
427e
427e  // 998244353 = 7*17*2^23+1, G = 3
fb9a  const int P = 1004535809, G = 3; // = 479*2^21+1
427e
87ab  struct NTT{
c47c      int rev[NMAX];
0eda      LL omega[NMAX], oinv[NMAX];
81af      int g, g_inv; // g: g_n = G^((P-1)/n)
9827      int K, N;
427e
2a2c      LL powmod(LL b, LL e){
95a2          LL r = 1;
3e90          while (e){
6624              if (e&1) r = r * b % P;
489e              b = b * b % P;
16fc              e >>= 1;
95cf          }
547e          return r;
95cf      }
427e
f420      NTT(int k){
e209          K = k; N = 1 << k;
7652          g = powmod(G, (P-1)/N);
4b3a          g_inv = powmod(g, N-1);
e04f          omega[0] = oinv[0] = 1;
b393          rep (i, N){
```

```
            rev[i] = (rev[i>>1]>>1) | ((i&1)<<(K-1));              7ba3
            if (i){                                                ad4f
                omega[i] = omega[i-1] * g % P;                     8d8b
                oinv[i] = oinv[i-1] * g_inv % P;                   9e14
            }                                                      95cf
        }                                                          95cf
    }                                                              95cf
                                                                   427e
    void _ntt(LL* a, LL* w){                                       9668
        rep (i, N) if (i < rev[i]) swap(a[i], a[rev[i]]);          a215
        for (int l = 2; l <= N; l *= 2){                           ac6e
            int m = l/2;                                           2969
            for (LL* p = a; p != a + N; p += l)                    7a1d
                rep (k, m){                                        c24f
                    LL t = w[N/l*k] * p[k+m] % P;                  0ad3
                    p[k+m] = (p[k] - t + P) % P;                   6209
                    p[k] = (p[k] + t) % P;                         fa1b
                }                                                  95cf
        }                                                          95cf
    }                                                              95cf
                                                                   427e
    void ntt(LL* a){_ntt(a, omega);}                               92ea
    void intt(LL* a){                                              5daf
        LL inv = powmod(N, P-2);                                   1f2a
        _ntt(a, oinv);                                             9910
        rep (i, N) a[i] = a[i] * inv % P;                          a873
    }                                                              95cf
                                                                   427e
    void conv(LL* a, LL* b){                                       3a5b
        ntt(a); ntt(b);                                            ad16
        rep (i, N) a[i] = a[i] * b[i] % P;                         e49e
        intt(a);                                                   5748
    }                                                              95cf
};                                                                 329b
```

4.7 Sieve of Euler

```
const int MAXX = 1e7+5;                                            cfc3
bool p[MAXX];                                                      5861
int prime[MAXX], sz;                                               73ae
                                                                   427e
void sieve(){                                                      9bc6
    p[0] = p[1] = 1;                                               9628
    for (int i = 2; i < MAXX; i++){
        if (!p[i]) prime[sz++] = i;
        for (int j = 0; j < sz && i*prime[j] < MAXX; j++){
            p[i*prime[j]] = 1;
            if (i % prime[j] == 0) break;
        }
    }
}
```

4.8 Miler-Rabin primality test

```
bool test(LL n){
    if (n < 3) return n==2;
    // ! The array a[] should be modified if the range of x changes.
    const LL a[] = {2LL, 7LL, 61LL, LLONG_MAX};
    LL r = 0, d = n-1, x;
    while (~d & 1) d >>= 1, r++;
    for (int i=0; a[i] < n; i++){
        x = powmod(a[i], d, n);
        if (x == 1 || x == n-1) goto next;
        rep (i, r) {
            x = mulmod(x, x, n);
            if (x == n-1) goto next;
        }
        return false;
next:;
    }
    return true;
}
```

4.9 Pollard's rho algorithm

```
ULL gcd(ULL a, ULL b) {return b ? gcd(b, a % b) : a;}

ULL PollardRho(ULL n){
    ULL c, x, y, d = n;
    if (~n&1) return 2;
    while (d == n){
        x = y = 2;
        d = 1;
        c = rand() % (n - 1) + 1;
        while (d == 1){
            x = (mulmod(x, x, n) + c) % n;
            y = (mulmod(y, y, n) + c) % n;
            y = (mulmod(y, y, n) + c) % n;
            d = gcd(x>y ? x-y : y-x, n);
        }
    }
    return d;
}
```

5 Graph Theory

5.1 Strongly connected component

```
const int MAXV = 100005;

struct graph{
    vector<int> adj[MAXV];
    stack<int> s;
    int V; // number of vertices
    int pre[MAXV], lnk[MAXV], scc[MAXV];
    int time, sccn;

    void add_edge(int u, int v){
        adj[u].push_back(v);
    }

    void dfs(int u){
        pre[u] = lnk[u] = ++time;
        s.push(u);
        for (int v : adj[u]){
            if (!pre[v]){
                dfs(v);
                lnk[u] = min(lnk[u], lnk[v]);
            } else if (!scc[v]){
                lnk[u] = min(lnk[u], pre[v]);
            }
        }
        if (lnk[u] == pre[u]){
            sccn++;
            int x;
            do {
```

```
3834                x = s.top(); s.pop();
b0e9                scc[x] = sccn;
6757            } while (x != u);
95cf        }
95cf    }
427e
4c88    void find_scc(){
f4a2        time = sccn = 0;
8de7        memset(scc, 0, sizeof scc);
8c2f        memset(pre, 0, sizeof pre);
6901        Rep (i, V){
56d1            if (!pre[i]) dfs(i);
95cf        }
95cf    }
427e
27ce    vector<int> adjc[MAXV];
364d    void contract(){
1a1e        Rep (i, V)
21a2            rep (j, adj[i].size()){
b730                if (scc[i] != scc[adj[i][j]])
b46e                    adjc[scc[i]].push_back(scc[adj[i][j]]);
95cf            }
95cf    }
329b };
```

5.2 Vertex biconnected component

```
0f42 const int MAXN = 100005;
2ea0 struct graph {
33ae     int pre[MAXN], iscut[MAXN], bccno[MAXN], dfs_clock, bcc_cnt;
848f     vector<int> adj[MAXN], bcc[MAXN];
6b06     set<pair<int, int>> bcce[MAXN];
427e
76f7     stack<pair<int, int>> s;
427e
bfab     void add_edge(int u, int v) {
c71a         adj[u].push_back(v);
a717         adj[v].push_back(u);
95cf     }
427e
7d3c     int dfs(int u, int fa) {
9fe6         int lowu = pre[u] = ++dfs_clock;
ec14         int child = 0;
```

```
        for (int v : adj[u]) {                                          18f6
            if (!pre[v]) {                                              173e
                s.push({u, v});                                         e7f8
                child++;                                                fdcf
                int lowv = dfs(v, u);                                   f851
                lowu = min(lowu, lowv);                                 189c
                if (lowv >= pre[u]) {                                   b687
                    iscut[u] = 1;                                       6323
                    bcc[bcc_cnt].clear();                               57eb
                    bcce[bcc_cnt].clear();                              90b8
                    while (1) {                                         a147
                        int xu, xv;                                     a6a3
                        tie(xu, xv) = s.top(); s.pop();                 a0c3
                        bcce[bcc_cnt].insert({min(xu, xv), max(xu, xv)}); 0ef5
                        if (bccno[xu] != bcc_cnt) {                     3db2
                            bcc[bcc_cnt].push_back(xu);                 e0db
                            bccno[xu] = bcc_cnt;                        d27f
                        }                                               95cf
                        if (bccno[xv] != bcc_cnt) {                     f357
                            bcc[bcc_cnt].push_back(xv);                 752b
                            bccno[xv] = bcc_cnt;                        57c9
                        }                                               95cf
                        if (xu == u && xv == v) break;                  7096
                    }                                                   95cf
                    bcc_cnt++;                                          03f5
                }                                                       95cf
            } else if (pre[v] < pre[u] && v != fa) {                    7470
                s.push({u, v});                                         e7f8
                lowu = min(lowu, pre[v]);                               f115
            }                                                           95cf
        }                                                               95cf
        if (fa < 0 && child == 1) iscut[u] = 0;                         e104
        return lowu;                                                    1160
    }                                                                   95cf
                                                                        427e
    void find_bcc(int n) {                                              17be
        memset(pre, 0, sizeof pre);                                     8c2f
        memset(iscut, 0, sizeof iscut);                                 e2d2
        memset(bccno, -1, sizeof bccno);                                40d3
        dfs_clock = bcc_cnt = 0;                                        fae2
        rep (i, n) if (!pre[i]) dfs(i, -1);                             5c63
    }                                                                   95cf
};                                                                      329b
```

5.3 Maximum flow (Dinic)

```
bcf8 struct edge{
60e2     int from, to;
5e6d     LL cap, flow;
329b };
427e
e2cd const int MAXN = 1005;
9062 struct Dinic {
4dbf     int n, m, s, t;
9f0c     vector<edge> edges;
b891     vector<int> G[MAXN];
bbb6     bool vis[MAXN];
b40a     int d[MAXN];
ddec     int cur[MAXN];
427e
5973     void add_edge(int from, int to, LL cap) {
7b55         edges.push_back(edge{from, to, cap, 0});
1db7         edges.push_back(edge{to, from, 0, 0});
fe77         m = edges.size();
dff5         G[from].push_back(m-2);
8f2d         G[to].push_back(m-1);
95cf     }
427e
1836     bool bfs() {
3b73         memset(vis, 0, sizeof(vis));
93d2         queue<int> q;
5d13         q.push(s);
2cd2         vis[s] = 1;
721d         d[s] = 0;
cc78         while (!q.empty()) {
66ba             int x = q.front(); q.pop();
3b61             for (int i = 0; i < G[x].size(); i++) {
b510                 edge& e = edges[G[x][i]];
bba9                 if (!vis[e.to] && e.cap > e.flow) {
cd72                     vis[e.to] = 1;
cf26                     d[e.to] = d[x] + 1;
ca93                     q.push(e.to);
95cf                 }
95cf             }
95cf         }
b23b         return vis[t];
95cf     }
427e
```

```
    LL dfs(int x, LL a) {                                                     9252
        if (x == t || a == 0) return a;                                       6904
        LL flow = 0, f;                                                       8bf9
        for (int& i = cur[x]; i < G[x].size(); i++) {                         f515
            edge& e = edges[G[x][i]];                                         b510
            if(d[x] + 1 == d[e.to] && (f = dfs(e.to, min(a, e.cap-e.flow))) > 0)   2374
              {
                e.flow += f;                                                  1cce
                edges[G[x][i]^1].flow -= f;                                   e16d
                flow += f;                                                    a74d
                a -= f;                                                       23e5
                if(a == 0) break;                                             97ed
            }                                                                 95cf
        }                                                                     95cf
        return flow;                                                          84fb
    }                                                                         95cf
                                                                              427e
    LL max_flow(int s, int t) {                                               5bf2
        this->s = s; this->t = t;                                             590d
        LL flow = 0;                                                          62e2
        while (bfs()) {                                                       ed58
            memset(cur, 0, sizeof(cur));                                      f326
            flow += dfs(s, LLONG_MAX);                                        fb3a
        }                                                                     95cf
        return flow;                                                          84fb
    }                                                                         95cf
                                                                              427e
    vector<int> min_cut() { // call this after maxflow                        c72e
        vector<int> ans;                                                      1df9
        for (int i = 0; i < edges.size(); i++) {                              df9a
            edge& e = edges[i];                                               56d8
            if(vis[e.from] && !vis[e.to] && e.cap > 0) ans.push_back(i);      46a2
        }                                                                     95cf
        return ans;                                                           4206
    }                                                                         95cf
};                                                                            329b
```

5.4 Maximum cardinality bipartite matching (Hungarian)

```
#include <bits/stdc++.h>                                                      302f
using namespace std;                                                          421c
                                                                              427e
#define rep(i, n) for (int i = 0; i < (n); i++)                               0d6c
```

```
cfe3 #define Rep(i, n) for (int i = 1; i <= (n); i++)
8843 #define range(x) (x).begin(), (x).end()
5cad typedef long long LL;
427e
84ee struct Hungarian{
fbf6     int nx, ny;
9ec6     vector<int> mx, my;
9d4c     vector<vector<int> > e;
edec     vector<bool> mark;
427e
8324     void init(int nx, int ny){
c1d1         this->nx = nx;
f9c1         this->ny = ny;
ac92         mx.resize(nx); my.resize(ny);
3f11         e.clear(); e.resize(nx);
1023         mark.resize(nx);
95cf     }
427e
4589     inline void add(int a, int b){
486c         e[a].push_back(b);
95cf     }
427e
0c2b     bool augment(int i){
207c         if (!mark[i]) {
dae4             mark[i] = true;
6a1e             for (int j : e[i]){
0892                 if (my[j] == -1 || augment(my[j])){
9ca3                     mx[i] = j; my[j] = i;
3361                     return true;
95cf                 }
95cf             }
95cf         }
438e         return false;
95cf     }
427e
3fac     int match(){
5b57         int ret = 0;
b0f1         fill(range(mx), -1);
b957         fill(range(my), -1);
4ed1         rep (i, nx){
13a5             fill(range(mark), false);
cc89             if (augment(i)) ret++;
95cf         }
ee0f         return ret;
```

```
    }                                                                95cf
};                                                                   329b
```

5.5 Minimum cost maximum flow

```
struct edge{                                                         bcf8
    int from, to;                                                    60e2
    int cap, flow;                                                   d698
    LL cost;                                                         32cc
};                                                                   329b
                                                                     427e
const LL INF = LLONG_MAX / 2;                                        cc3e
const int MAXN = 5005;                                               2aa8
struct MCMF {                                                        c6cb
    int s, t, n, m;                                                  9ceb
    vector<edge> edges;                                              9f0c
    vector<int> G[MAXN];                                             b891
    bool inq[MAXN]; // queue                                         f74f
    LL d[MAXN];     // distance                                      8f67
    int p[MAXN];    // previous                                      9524
    int a[MAXN];    // improvement                                   b330
                                                                     427e
    void add_edge(int from, int to, int cap, LL cost) {              f7f2
        edges.push_back(edge{from, to, cap, 0, cost});               24f0
        edges.push_back(edge{to, from, 0, 0, -cost});                95f0
        m = edges.size();                                            fe77
        G[from].push_back(m-2);                                      dff5
        G[to].push_back(m-1);                                        8f2d
    }                                                                95cf
                                                                     427e
    bool spfa(){                                                     3c52
        queue<int> q;                                                93d2
        fill(d, d + MAXN, INF); d[s] = 0;                            8494
        memset(inq, 0, sizeof(inq));                                 fd48
        q.push(s); inq[s] = true;                                    5e7c
        p[s] = 0; a[s] = INT_MAX;                                    2dae
        while (!q.empty()){                                          cc78
            int u = q.front(); q.pop(); inq[u] = false;              b0aa
            rep (i, G[u].size()){                                    ddff
                edge& e = edges[G[u][i]];                            c234
                if (e.cap > e.flow && d[e.to] > d[u] + e.cost){      3601
                    d[e.to] = d[u] + e.cost;                         55bc
                    p[e.to] = G[u][i];                               0bea
8249                     a[e.to] = min(a[u], e.cap - e.flow);
e5d3                     if (!inq[e.to]) q.push(e.to), inq[e.to] = true;
95cf                 }
95cf             }
95cf         }
6d7c         return d[t] != INF;
95cf     }
427e 
71a4     void augment(){
06f1         int u = t;
b19d         while (u != s){
db09             edges[p[u]].flow += a[t];
25a9             edges[p[u]^1].flow -= a[t];
e6c9             u = edges[p[u]].from;
95cf         }
95cf     }
427e 
6e20 #ifdef GIVEN_FLOW
5972     bool min_cost(int s, int t, int f, LL& cost) {
590d         this->s = s; this->t = t;
21d4         int flow = 0;
23cb         cost = 0;
22dc         while (spfa()) {
bcdb             augment();
a671             if (flow + a[t] >= f){
9c87                 cost += (f - flow) * a[t]; flow = f;
3361                 return true;
8e2e             } else {
2a83                 flow += a[t]; cost += a[t] * d[t];
95cf             }
95cf         }
438e         return false;
95cf     }
a8cb #else
f9a9     int min_cost(int s, int t, LL& cost) {
590d         this->s = s; this->t = t;
21d4         int flow = 0;
23cb         cost = 0;
22dc         while (spfa()) {
bcdb             augment();
2a83             flow += a[t]; cost += a[t] * d[t];
95cf         }
84fb         return flow;
95cf     }
1937 #endif
329b };
```

5.6 Global minimum cut (Stoer-Wagner)

```
f9d7 typedef vector<LL> VI;
045e typedef vector<VI> VVI;
427e 
f012 pair<LL, VI> stoer(VVI &w) {
66f7     int n = w.size();
4d98     VI used(n), c, bestc;
329d     LL bestw = -1;
427e 
cd21     for (int ph = n - 1; ph >= 0; ph--) {
ec6e         VI wt = w[0], added = used;
f20e         int prev, last = 0;
4b32         rep (i, ph) {
8bfc             prev = last;
0706             last = -1;
4942             for (int j = 1; j < n; j++)
c4b9                 if (!added[j] && (last == -1 || wt[j] > wt[last]))
887d                     last = j;
71bc             if (i == ph - 1) {
9cfa                 rep (j, n) w[prev][j] += w[last][j];
1f25                 rep (j, n) w[j][prev] = w[prev][j];
5613                 used[last] = true;
8e11                 c.push_back(last);
bb8e                 if (bestw == -1 || wt[last] < bestw) {
bab6                     bestc = c;
372e                     bestw = wt[last];
95cf                 }
8e2e             } else {
caeb                 rep (j, n) wt[j] += w[last][j];
8b92                 added[last] = true;
95cf             }
95cf         }
95cf     }
038c     return {bestw, bestc};
95cf }
```

5.7 Heavy-light decomposition

```
0f42 const int MAXN = 100005;
0b32 vector<int> adj[MAXN];
42f2 int sz[MAXN], top[MAXN], fa[MAXN], son[MAXN], depth[MAXN], id[MAXN];
427e
be5c void dfs1(int x, int dep, int par){
7489     depth[x] = dep;
2ee7     sz[x] = 1;
adb4     fa[x] = par;
b79d     int maxn = 0, s = 0;
c861     for (int c: adj[x]){
fe45         if (c == par) continue;
fd2f         dfs1(c, dep + 1, x);
b790         sz[x] += sz[c];
f0f1         if (sz[c] > maxn){
c749             maxn = sz[c];
fe19             s = c;
95cf         }
95cf     }
0e08     son[x] = s;
95cf }
427e
ba54 int cid = 0;
3644 void dfs2(int x, int t){
8d96     top[x] = t;
d314     id[x] = ++cid;
c4a1     if (son[x]) dfs2(son[x], t);
c861     for (int c: adj[x]){
9881         if (c == fa[x]) continue;
5518         if (c == son[x]) continue;
13f9         else dfs2(c, c);
95cf     }
95cf }
427e
0f04 void decomp(int root){
9fa4     dfs1(root, 1, 0);
1c88     dfs2(root, root);
95cf }
427e
2c98 void query(int u, int v){
03a1     while (top[u] != top[v]){
45ec         if (depth[top[u]] < depth[top[v]]) swap(u, v);
427e         // id[top[u]] to id[u]
005b         u = fa[top[u]];
95cf     }
```

```
    if (depth[u] > depth[v]) swap(u, v);          6083
    // id[u] to id[v]                             427e
}                                                 95cf
```

6 Data Structures

6.1 Segment tree

```
LL p;                                             3942
const int MAXN = 4 * 100006;                      1ebb
struct segtree {                                  451a
  int l[MAXN], m[MAXN], r[MAXN];                  27be
  LL val[MAXN], tadd[MAXN], tmul[MAXN];           4510
                                                  427e
#define lson (o<<1)                               ac35
#define rson (o<<1|1)                             1294
                                                  427e
  void pull(int o) {                              1344
    val[o] = (val[lson] + val[rson]) % p;         bbe9
  }                                               95cf
                                                  427e
  void push_add(int o, LL x) {                    e4bc
    val[o] = (val[o] + x * (r[o] - l[o])) % p;    5dd6
    tadd[o] = (tadd[o] + x) % p;                  6eff
  }                                               95cf
                                                  427e
  void push_mul(int o, LL x) {                    d658
    val[o] = val[o] * x % p;                      b82c
    tadd[o] = tadd[o] * x % p;                    aa86
    tmul[o] = tmul[o] * x % p;                    649f
  }                                               95cf
                                                  427e
  void push(int o) {                              b149
    if (l[o] == m[o]) return;                     3159
    if (tmul[o] != 1) {                           0a90
      push_mul(lson, tmul[o]);                    0f4a
      push_mul(rson, tmul[o]);                    045e
      tmul[o] = 1;                                ac0a
    }                                             95cf
    if (tadd[o]) {                                1b82
      push_add(lson, tadd[o]);                    9547
      push_add(rson, tadd[o]);                    0e73
```

```
6234        tadd[o] = 0;
95cf      }
95cf    }
427e
471c    void build(int o, int ll, int rr) {
0e87      int mm = (ll + rr) / 2;
9d27      l[o] = ll; r[o] = rr; m[o] = mm;
ac0a      tmul[o] = 1;
5c92      if (ll == mm) {
001f        scanf("%lld", val + o);
e5b6        val[o] %= p;
8e2e      } else {
7293        build(lson, ll, mm);
5e67        build(rson, mm, rr);
ba26        pull(o);
95cf      }
95cf    }
427e
4406    void add(int o, int ll, int rr, LL x) {
3c16      if (ll <= l[o] && r[o] <= rr) {
db32        push_add(o, x);
8e2e      } else {
c4b0        push(o);
4305        if (m[o] > ll) add(lson, ll, rr, x);
d5a6        if (m[o] < rr) add(rson, ll, rr, x);
ba26        pull(o);
95cf      }
95cf    }
427e
48cd    void mul(int o, int ll, int rr, LL x) {
3c16      if (ll <= l[o] && r[o] <= rr) {
e7d0        push_mul(o, x);
8e2e      } else {
c4b0        push(o);
d1ba        if (ll < m[o]) mul(lson, ll, rr, x);
67f3        if (m[o] < rr) mul(rson, ll, rr, x);
ba26        pull(o);
95cf      }
95cf    }
427e
0f62    LL query(int o, int ll, int rr) {
3c16      if (ll <= l[o] && r[o] <= rr) {
6dfe        return val[o];
8e2e      } else {
```

```
      LL ans = 0;                                                      f7ff
      push(o);                                                         c4b0
      if (m[o] > ll) ans += query(lson, ll, rr);                       c5f8
      if (m[o] < rr) ans += query(rson, ll, rr);                       ef81
      return ans % p;                                                  a420
    }                                                                  95cf
  }                                                                    95cf
} seg;                                                                 4d99
```

6.2 Link/cut tree

```
// about 0.13s per 100k ops @Luogu.org                                 427e
                                                                       427e
namespace LCT {                                                        ed4d
  const int MAXN = 300005;                                             5ece
  int fa[MAXN], ch[MAXN][2], val[MAXN], sum[MAXN];                     6a6d
  bool rev[MAXN];                                                      c6e1
                                                                       427e
  bool isroot(int x) {                                                 7839
    return ch[fa[x]][0] == x || ch[fa[x]][1] == x;                     45a9
  }                                                                    95cf
                                                                       427e
  void pull(int x) {                                                   3bf9
    sum[x] = val[x] ^ sum[ch[x][0]] ^ sum[ch[x][1]];                   6664
  }                                                                    95cf
                                                                       427e
  void reverse(int x) {                                                3698
    swap(ch[x][0], ch[x][1]);                                          7850
    rev[x] ^= 1;                                                       52c6
  }                                                                    95cf
                                                                       427e
  void push(int x) {                                                   1a53
    if (rev[x]) {                                                      8f1f
      if (ch[x][0]) reverse(ch[x][0]);                                 ebf3
      if (ch[x][1]) reverse(ch[x][1]);                                 6eb0
      rev[x] = 0;                                                      8fc1
    }                                                                  95cf
  }                                                                    95cf
                                                                       427e
  void rotate(int x) {                                                 425f
    int y = fa[x], z = fa[y], k = ch[y][1] == x, w = ch[x][!k];        51af
    if (isroot(y)) ch[z][ch[z][1] == y] = x;                           e1fe
    ch[x][!k] = y; ch[y][k] = w;                                       af46
```

```
fa6f      if (w) fa[w] = y;
3540      fa[y] = x; fa[x] = z;
72ef      pull(y);
95cf    }
427e
bc1b    void pushall(int x) {
a316      if (isroot(x)) pushall(fa[x]);
a97b      push(x);
95cf    }
427e
f69c    void splay(int x) {
d095      int y = x, z = 0;
8ab3      pushall(y);
f244      while (isroot(x)) {
ceef        y = fa[x]; z = fa[y];
4449        if (isroot(y)) rotate((ch[y][0] == x) ^ (ch[z][0] == y) ? x : y);
cf90        rotate(x);
95cf      }
78a0      pull(x);
95cf    }
427e
6229    void access(int x) {
1548      int z = x;
ba78      for (int y = 0; x; x = fa[y = x]) {
8fec        splay(x);
b05d        ch[x][1] = y;
78a0        pull(x);
95cf      }
7afd      splay(z);
95cf    }
427e
502e    void chroot(int x) {
766a      access(x);
cb0d      reverse(x);
95cf    }
427e
471a    void split(int x, int y) {
3015      chroot(x);
29b5      access(y);
95cf    }
427e
d87a    int Root(int x) {
766a      access(x);
874d      while (ch[x][0]) {
```

```
      push(x);                                            a97b
      x = ch[x][0];                                       b83a
    }                                                     95cf
    splay(x);                                             8fec
    return x;                                             d074
  }                                                       95cf
                                                          427e
  void Link(int u, int v) {  // assume unconnected before 70d3
    chroot(u);                                            b8a5
    fa[u] = v;                                            2448
  }                                                       95cf
                                                          427e
  void Cut(int u, int v) {  // assume connected before    c2f4
    split(u, v);                                          e8ce
    fa[u] = ch[v][0] = 0;                                 fd95
    pull(v);                                              743b
  }                                                       95cf
                                                          427e
  int Query(int u, int v) {                               6ca2
    split(u, v);                                          e8ce
    return sum[v];                                        a5ba
  }                                                       95cf
                                                          427e
  void Update(int u, int x) {                             eaba
    splay(u);                                             46ce
    val[u] = x;                                           1d62
  }                                                       95cf
};                                                        329b
```

6.3 Balanced binary search tree from `pb_ds`

```
#include <ext/pb_ds/assoc_container.hpp>                                          0475
using namespace __gnu_pbds;                                                       332d
                                                                                  427e
tree<int, null_type, less<int>, rb_tree_tag, tree_order_statistics_node_update>   43a7
  rkt;
// null_tree_node_update                                                          427e
                                                                                  427e
// SAMPLE USAGE                                                                   427e
rkt.insert(x);          // insert element                                         190e
rkt.erase(x);           // erase element                                          05d4
rkt.order_of_key(x);    // obtain the number of elements less than x              add5
rkt.find_by_order(i);   // iterator to i–th (numbered from 0) smallest element    b064
```

```
c103 rkt.lower_bound(x);
4ff4 rkt.upper_bound(x);
b19b rkt.join(rkt2);          // merge tree (only if their ranges do not intersect)
cb47 rkt.split(x, rkt2);      // split all elements greater than x to rkt2
```

6.4 Persistent segment tree, range k-th query

```
f1a7 struct node {
2ff6   static int n, pos;
427e 
7cec   int value;
70e2   node *left, *right;
427e 
20b0   void* operator new(size_t size);
427e 
3dc0   static node* Build(int l, int r) {
b6c5     node* a = new node;
ce96     if (r > l + 1) {
181e       int mid = (l + r) / 2;
3ba2       a->left = Build(l, mid);
8aaf       a->right = Build(mid, r);
8e2e     } else {
bfc4       a->value = 0;
95cf     }
5ffd     return a;
95cf   }
427e 
5a45   static node* init(int size) {
2c46     n = size;
7ee3     pos = 0;
be52     return Build(0, n);
95cf   }
427e 
93c0   static int Query(node* lt, node *rt, int l, int r, int k) {
d30c     if (r == l + 1) return l;
    int mid = (l + r) / 2;                                          181e
    if (rt->left->value - lt->left->value < k) {                    cb5a
      k -= rt->left->value - lt->left->value;                       8edb
      return Query(lt->right, rt->right, mid, r, k);                2412
    } else {                                                        8e2e
      return Query(lt->left, rt->left, l, mid, k);                  0119
    }                                                               95cf
  }                                                                 95cf
                                                                    427e
  static int query(node* lt, node *rt, int k) {                     c9ad
    return Query(lt, rt, 0, n, k);                                  9e27
  }                                                                 95cf
                                                                    427e
  node *Inc(int l, int r, int pos) const {                          b19c
    node* a = new node(*this);                                      5794
    if (r > l + 1) {                                                ce96
      int mid = (l + r) / 2;                                        181e
      if (pos < mid)                                                203d
        a->left = left->Inc(l, mid, pos);                           f44a
      else                                                          649a
        a->right = right->Inc(mid, r, pos);                         1024
    }                                                               95cf
    a->value++;                                                     2b3e
    return a;                                                       5ffd
  }                                                                 95cf
                                                                    427e
  node *inc(int index) {                                            e80f
    return Inc(0, n, index);                                        c246
  }                                                                 95cf
} nodes[8000000];                                                   865a
                                                                    427e
int node::n, node::pos;                                             99ce
inline void* node::operator new(size_t size) {                      1987
  return nodes + (pos++);                                           bb3c
}                                                                   95cf
```