南京大学 ACM-ICPC 集训队代码模版库

Contents

1 General

1.1 Code library checksum

```
ab14 #!/usr/bin/python3
c502 import re, sys, hashlib
427e
f7db for line in sys.stdin.read().strip().split("\n") :
ddf5     print(hashlib.md5(re.sub(r'\s|//.*', '', line).encode('utf8')).hexdigest()
           [-4:], line)
```

1.2 Makefile

```
dab2 .PHONY : run
427e
5f25 run : $(t)
427e
207e $(t) : $(t).cpp
155d   g++ --std=c++14 -O2 -D__LOCAL_DEBUG__ -fsanitize=undefined -fsanitize=address
         -ggdb -pipe -o $@ $<
```

1.3 .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,<,>,[,]
```

1.4 Template

```
#include <bits/stdc++.h>                                                   302f
using namespace std;                                                       421c
                                                                           427e
#ifdef __LOCAL_DEBUG__                                                     426f
# define _debug(fmt, ...) fprintf(stderr, "[%s]␣" fmt "\n", \              3341
    __func__, ##__VA_ARGS__)                                               611f
#else                                                                      a8cb
# 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 2-SAT

```
0f42  const int MAXN = 100005;
03a9  struct twoSAT{
5c83      int n;
8f72      vector<int> G[MAXN*2];
d060      bool mark[MAXN*2];
b42d      int S[MAXN*2], c;
427e
d34f      void init(int n){
b985          this->n = n;
f9ec          for (int i=0; i<n*2; i++) G[i].clear();
0609          memset(mark, 0, sizeof(mark));
95cf      }
427e
3bd5      bool dfs(int x){
bd70          if (mark[x^1]) return false;
c96a          if (mark[x]) return true;
fd23          mark[x] = true;
4bea          S[c++] = x;
1ce6          for (int i=0; i<G[x].size(); i++)
d942              if (!dfs(G[x][i])) return false;
3361          return true;
95cf      }
427e
5894      void add_clause(int x, bool xval, int y, bool yval){
6afe          x = x * 2 + xval;
e680          y = y * 2 + yval;
81cc          G[x^1].push_back(y);
6835          G[y^1].push_back(x);
95cf      }
427e
d0cb      bool solve() {
7c39          for (int i=0; i<n*2; i+=2){
e63f              if (!mark[i] && !mark[i+1]){
88fb                  c = 0;
f4b9                  if (!dfs(i)){
3f03                      while (c > 0) mark[S[--c]] = false;
86c5                      if (!dfs(i+1)) return false;
95cf                  }
95cf              }
95cf          }
3361          return true;
95cf      }
427e
5f0a      inline bool value(unsigned i){return mark[2*i+1];}
329b  };
```

2.2 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);
3dd4   fail[0] = fail[1] = 0;
d8a8   for (int i = 1; i < len; i++) {
147f     int j = fail[i];
3c79     while (j && p[i] != p[j]) j = fail[j];
4643     fail[i + 1] = p[i] == p[j] ? j + 1 : 0;
95cf   }
95cf }
427e
c464 inline void found(int pos) {
427e   // ! add codes for having found at pos
95cf }
427e
1932 void match(const char* t, const char* p) {  // must be called after construct
8482   int n = strlen(t);
8fd0   int j = 0;
be8e   rep(i, n) {
4e19     while (j && p[j] != t[i]) j = fail[j];
b5d5     if (p[j] == t[i]) j++;
f024     if (j == len) found(i - len + 1);
95cf   }
95cf }
```

3.2 Manacher algorithm

```
81d4 struct Manacher {
cd09   int Len;
9255   vector<int> lc;
b301   string s;
427e
ec07   void work() {
c033     lc[1] = 1;
6bef     int k = 1;
                                                                  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
  void init(const char *tt) {                                     bfd5
    int len = strlen(tt);                                         aaaf
    s.resize(len * 2 + 10);                                       f701
    lc.resize(len * 2 + 10);                                      7045
    s[0] = '*';                                                   8e13
    s[1] = '#';                                                   ae54
    for (int i = 0; i < len; i++) {                               1321
      s[i * 2 + 2] = tt[i];                                       e995
      s[i * 2 + 1] = '#';                                         69fd
    }                                                             95cf
    s[len * 2 + 1] = '#';                                         43fd
    s[len * 2 + 2] = '\0';                                        75d1
    Len = len * 2 + 2;                                            61f7
    work();                                                       3e7a
  }                                                               95cf
                                                                  427e
  pair<int, int> maxpal(int l, int r) {                           b194
    int center = l + r + 1;                                       901a
    int rad = lc[center] / 2;                                     ffb2
    int rmid = (l + r + 1) / 2;                                   ab54
    int rl = rmid - rad, rr = rmid + rad - 1;                     17e4
    if ((r ^ l) & 1) {                                            3908
    } else rr++;                                                  69f3
    return {max(l, rl), min(r, rr)};                              69dc
  }                                                               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     }
cc78     while (!q.empty()) {
31f0       int r = q.front();
15dd       q.pop();
ce3c       rep(c, CHARN) {
ab59         int u = tr[r][c];
0ef5         if (!u) {
9d58           tr[r][c] = tr[fail[r]][c];
b333           continue;
95cf         }
3e14         q.push(u);
b3ff         int v = fail[r];
d2ea         while (v && !tr[v][c]) v = fail[v];
c275         fail[u] = tr[v][c];
654c         last[u] = tag[fail[u]] ? fail[u] : last[fail[u]];
95cf       }
95cf     }
95cf   }
427e
7752   void found(int pos, int j) {
043e     if (j) {
427e       // ! add codes for having found word with tag[j]
4a96       found(pos, last[j]);
95cf     }
95cf   }
427e
9785   void find(const char* text) {  // must be called after construct()
80a4     int p = 0, c, len = strlen(text);
9c94     rep(i, len) {
b3db       c = id(text[i]);
f119       p = tr[p][c];
```

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

3.5 Rolling hash

PLEASE call `init_hash()` in `int main()`!

Usage:

`build(str)`	Construct the hasher with given string.
`operator()(l, r)`	Get hash value of substring $[l, r)$.

```
1e42  const LL mod = 1006658951440146419, g = 967;
9f60  const int MAXN = 200005;
0291  LL pg[MAXN];
427e
6832  inline LL mul(LL x, LL y) {
c919      return __int128_t(x) * y % mod;
95cf  }
427e
599a  void init_hash() {   // must be called in `int main()`
286f      pg[0] = 1;
d00f      for (int i = 1; i < MAXN; i++)
4aa9          pg[i] = pg[i - 1] * g % mod;
95cf  }
427e
7e62  struct hasher {
534a      LL val[MAXN];
427e
4554      void build(const char *str) {   // assume lower-case letter only
f937          for (int i = 0; str[i]; i++)
9645              val[i+1] = (mul(val[i], g) + str[i]) % mod;
95cf      }
427e
19f8      LL operator() (int l, int r) { // [l, r)
```

```
        return (val[r] - mul(val[l], pg[r - l]) + mod) % mod;    9986
    }                                                             95cf
} ha;                                                             b179
```

4 Linear Algebra

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

```
8b44 const int MAXD = 30;
03a6 struct linearbasis {
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();
  vector<int> irow(n), icol(n), ipiv(n); a25e
  LL det = 1; 2976
 427e
  rep (i, n) { be8e
    int pj = -1, pk = -1; d2b5
    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) {
  LL e = MOD - 2, r = 1;
  while (e) {
    if (e & 1) r = r * b % MOD;
    b = b * b % MOD;
    e >>= 1;
  }
  return r;
}

struct Poly {
  vector<int> a;

  Poly() { a.clear(); }

  Poly(vector<int> &a) : a(a) {}

  int length() const { return a.size(); }

  Poly move(int d) {
    vector<int> na(d, 0);
    na.insert(na.end(), a.begin(), a.end());
    return Poly(na);
  }

  int calc(vector<int> &d, int pos) {
    int ret = 0;
    for (int i = 0; i < (int)a.size(); ++i) {
      if ((ret += (long long)d[pos - i] * a[i] % MOD) >= MOD) {
        ret -= MOD;
      }
    }
    return ret;
  }

  Poly operator - (const Poly &b) {
    vector<int> na(max(this->length(), b.length()));
    for (int i = 0; i < (int)na.size(); ++i) {
      int aa = i < this->length() ? this->a[i] : 0,
          bb = i < b.length() ? b.a[i] : 0;
      na[i] = (aa + MOD - bb) % MOD;
    }
    return Poly(na);
  }
};

Poly operator * (const int &c, const Poly &p) {
  vector<int> na(p.length());
  for (int i = 0; i < (int)na.size(); ++i) {
    na[i] = (long long)c * p.a[i] % MOD;
  }
  return na;
}

vector<int> solve(vector<int> a) {
  int n = a.size();
  Poly s, b;
  s.a.push_back(1), b.a.push_back(1);
  for (int i = 1, j = 0, ld = a[0]; i < n; ++i) {
    int d = s.calc(a, i);
    if (d) {
      if ((s.length() - 1) * 2 <= i) {
        Poly ob = b;
        b = s;
        s = s - (long long)d * inverse(ld) % MOD * ob.move(i - j);
        j = i;
        ld = d;
      } else {
        s = s - (long long)d * inverse(ld) % MOD * b.move(i - j);
      }
    }
  }
  // Caution: s.a might be shorter than expected
  return s.a;
}
```

4.5 Fast Walsh-Hadamard transform

```
void fwt(int* a, int n){
    for (int d = 1; d < n; d <<= 1)
        for (int i = 0; i < n; i += d << 1)
            rep (j, d){
                int x = a[i+j], y = a[i+j+d];
                // a[i+j] = x+y, a[i+j+d] = x-y;    // xor
                // a[i+j] = x+y;                    // and
                // a[i+j+d] = x+y;                  // or
95cf             }
95cf }
427e
4db1 void ifwt(int* a, int n){
5595     for (int d = 1; d < n; d <<= 1)
05f2         for (int i = 0; i < n; i += d << 1)
b833             rep (j, d){
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 Fast fourier transform

```
4e09 const int NMAX = 1<<20;
427e
3fbf typedef complex<double> cplx;
427e
abd1 const double PI = 2*acos(0.0);
12af struct FFT{
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
```

```
    void dft(cplx* a, cplx* w){                                   b941
        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 (cplx* p = a; p != a + N; p += l)                 b3cf
                rep (k, m){                                       c24f
                    cplx t = w[N/l*k] * p[k+m];                   fe06
                    p[k+m] = p[k] - t; p[k] += t;                 ecbf
                }                                                 95cf
        }                                                         95cf
    }                                                             95cf
                                                                  427e
    void fft(cplx* a){dft(a, omega);}                             617b
    void ifft(cplx* a){                                           a123
        dft(a, oinv);                                             3b2f
        rep (i, N) a[i] /= N;                                     57fc
    }                                                             95cf
                                                                  427e
    void conv(cplx* a, cplx* b){                                  bdc0
        fft(a); fft(b);                                           6497
        rep (i, N) a[i] *= b[i];                                  12a5
        ifft(a);                                                  f84e
    }                                                             95cf
};                                                                329b
```

4.7 Number theoretic transform

```
const int NMAX = 1<<21;                                           4ab9
                                                                  427e
// 998244353 = 7*17*2^23+1, G = 3                                 427e
const int P = 1004535809, G = 3; // = 479*2^21+1                  fb9a
                                                                  427e
struct NTT{                                                       87ab
    int rev[NMAX];                                                c47c
    LL omega[NMAX], oinv[NMAX];                                   0eda
    int g, g_inv; // g: g_n = G^((P-1)/n)                         81af
    int K, N;                                                     9827
                                                                  427e
    LL powmod(LL b, LL e){                                        2a2c
        LL r = 1;                                                 95a2
        while (e){                                                3e90
            if (e&1) r = r * b % P;                               6624
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){
7ba3            rev[i] = (rev[i>>1]>>1) | ((i&1)<<(K-1));
ad4f            if (i){
8d8b                omega[i] = omega[i-1] * g % P;
9e14                oinv[i] = oinv[i-1] * g_inv % P;
95cf            }
95cf        }
95cf    }
427e
9668    void _ntt(LL* a, LL* 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;
7a1d            for (LL* p = a; p != a + N; p += l)
c24f                rep (k, m){
0ad3                    LL t = w[N/l*k] * p[k+m] % P;
6209                    p[k+m] = (p[k] - t + P) % P;
fa1b                    p[k] = (p[k] + t) % P;
95cf                }
95cf        }
95cf    }
427e
92ea    void ntt(LL* a){_ntt(a, omega);}
5daf    void intt(LL* a){
1f2a        LL inv = powmod(N, P-2);
9910        _ntt(a, oinv);
a873        rep (i, N) a[i] = a[i] * inv % P;
95cf    }
427e
3a5b    void conv(LL* a, LL* b){
ad16        ntt(a); ntt(b);
e49e        rep (i, N) a[i] = a[i] * b[i] % P;
5748        intt(a);
```

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

5 Number Theory

5.1 Sieve of Euler

```
namespace sieve {                                   b62e
  constexpr int MAXN = 10000007;                    6589
  bool p[MAXN]; // true if not prime                e982
  int prime[MAXN], sz;                              6ae8
  int pval[MAXN], pcnt[MAXN];                       cbf7
  int f[MAXN];                                      6030
                                                    427e
  void exec(int N = MAXN) {                         76f6
    p[0] = p[1] = 1;                                9628
                                                    427e
    pval[1] = 1;                                    8a8a
    pcnt[1] = 0;                                    bdda
    f[1] = 1;                                       c6b9
                                                    427e
    for (int i = 2; i < N; i++) {                   a643
      if (!p[i]) {                                  01d6
        prime[sz++] = i;                            b2b2
        for (LL j = i; j < N; j *= i) {             37d9
          int b = j / i;                            758c
          pval[j] = i * pval[b];                    81fd
          pcnt[j] = pcnt[b] + 1;                    e0f3
          f[j] = ______; // f[j] = f(i^pcnt[j])     a96c
        }                                           95cf
      }                                             95cf
      for (int j = 0; i * prime[j] < N; j++) {      34c0
        int x = i * prime[j]; p[x] = 1;             f87a
        if (i % prime[j] == 0) {                    20cc
          pval[x] = pval[i] * prime[j];             9985
          pcnt[x] = pcnt[i] + 1;                    3f93
        } else {                                    8e2e
          pval[x] = prime[j];                       cc91
          pcnt[x] = 1;                              6322
        }                                           95cf
        if (x != pval[x]) {                         6191
d614            f[x] = f[x / pval[x]] * f[pval[x]]
95cf          }
5f51          if (i % prime[j] == 0) break;
95cf        }
95cf      }
95cf    }
95cf  }
```

5.2 Miller-Rabin primality test

The array a[] (excluding senitel, i.e. LLONG_MAX) should be

{2}	when $n < 2,047$.
{2, 7, 61}	when $n < 4,759,123,141$ (2^{32}) .
{2, 3, 5, 7, 11}	when $n < 2.1 \times 10^{12}$.
{2, 325, 9375, 28178, 450775, 9780504, 1795265022}	when $n < 2^{64}$.

```
f16f  bool test(LL n){
59f2      if (n < 3) return n==2;
427e      // ! The array a[] should be modified if the range of x changes.
3f11      const LL a[] = {2LL, 7LL, 61LL, LLONG_MAX};
c320      LL r = 0, d = n-1, x;
f410      while (~d & 1) d >>= 1, r++;
2975      for (int i=0; a[i] < n; i++){
ece1          x = powmod(a[i], d, n); // ! powmod must use for 64bit mulmod
7f99          if (x == 1 || x == n-1) goto next;
e257          rep (i, r) {
d7ff              x = mulmod(x, x, n);
8d2e              if (x == n-1) goto next;
95cf          }
438e          return false;
d490  next:;
95cf      }
3361      return true;
95cf  }
```

5.3 Pollard's rho algorithm

```
2e6b  ULL gcd(ULL a, ULL b) {return b ? gcd(b, a % b) : a;}
427e
```

```
ULL PollardRho(ULL n){                                54a5
    ULL c, x, y, d = n;                               45eb
    if (~n&1) return 2;                               d3e5
    while (d == n){                                   3c69
        x = y = 2;                                    0964
        d = 1;                                        4753
        c = rand() % (n - 1) + 1;                     5952
        while (d == 1){                               9e5b
            x = (mulmod(x, x, n) + c) % n;            33d5
            y = (mulmod(y, y, n) + c) % n;            e1bf
            y = (mulmod(y, y, n) + c) % n;            e1bf
            d = gcd(x>y ? x-y : y-x, n);              a313
        }                                             95cf
    }                                                 95cf
    return d;                                         5d89
}                                                     95cf
```

6 Graph Theory

6.1 Strongly connected component

```
const int MAXV = 100005;                              837c
                                                      427e
struct graph{                                         2ea0
    vector<int> adj[MAXV];                            88e3
    stack<int> s;                                     9cad
    int V; // number of vertices                      3d02
    int pre[MAXV], lnk[MAXV], scc[MAXV];              8b6c
    int time, sccn;                                   27ee
                                                      427e
    void add_edge(int u, int v){                      bfab
        adj[u].push_back(v);                          c71a
    }                                                 95cf
                                                      427e
    void dfs(int u){                                  d714
        pre[u] = lnk[u] = ++time;                     7e41
        s.push(u);                                    80f6
        for (int v : adj[u]){                         18f6
            if (!pre[v]){                             173e
                dfs(v);                               5f3c
                lnk[u] = min(lnk[u], lnk[v]);         002c
6068             } else if (!scc[v]){
d5df                 lnk[u] = min(lnk[u], pre[v]);
95cf             }
95cf         }
8de2         if (lnk[u] == pre[u]){
660f             sccn++;
3c9e             int x;
a69f             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 };
```

6.2 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
    void add_edge(int u, int v) {                                        bfab
        adj[u].push_back(v);                                             c71a
        adj[v].push_back(u);                                             a717
    }                                                                    95cf
                                                                         427e
    int dfs(int u, int fa) {                                             7d3c
        int lowu = pre[u] = ++dfs_clock;                                 9fe6
        int child = 0;                                                   ec14
        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
```

```
17be        void find_bcc(int n) {
8c2f            memset(pre, 0, sizeof pre);
e2d2            memset(iscut, 0, sizeof iscut);
40d3            memset(bccno, -1, sizeof bccno);
fae2            dfs_clock = bcc_cnt = 0;
5c63            rep (i, n) if (!pre[i]) dfs(i, -1);
95cf        }
329b    };
```

6.3 Minimum spanning arborescence (Chu-Liu)

All vertices are 1-based.

Usage:

`getans(n, root, edges)`	Compute the total size of MSA rooted at `root`.

Time Complexity: $O(|V||E|)$

```
bcf8    struct edge {
54f1        int u, v;
309c        LL w;
329b    };
427e
f5a4    const int MAXN = 10005;
7124    LL in[MAXN];
1c1d    int pre[MAXN], vis[MAXN], id[MAXN];
427e
5a43    LL getans(int n, int rt, vector<edge>& edges) {
f7ff        LL ans = 0;
8abb        int cnt = 0;
a147        while (1) {
641a            Rep (i, n) in[i] = LLONG_MAX, id[i] = vis[i] = 0;
0705            for (auto e : edges) {
073a                if (e.u != e.v and e.w < in[e.v]) {
c1df                    pre[e.v] = e.u;
5fbc                    in[e.v] = e.w;
95cf                }
95cf            }
3fdb            in[rt] = 0;
34d7            Rep (i, n) {
3c97                if (in[i] == LLONG_MAX) return -1;
cf57                ans += in[i];
a763                int u;
4b0e                for (u = i; u != rt && vis[u] != i && !id[u]; u = pre[u])
                    vis[u] = i;                                 88a2
                if (u != rt && !id[u]) {                        4b22
                    id[u] = ++cnt;                              b66e
                    for (int v = pre[u]; v != u; v = pre[v])    0443
                        id[v] = cnt;                            5c22
                }                                               95cf
            }                                                   95cf
            if (!cnt) return ans;                               91e9
            Rep (i, n) if (!id[i]) id[i] = ++cnt;               5e22
            for (auto& e : edges) {                             7400
                LL laz = in[e.v];                               7750
                e.u = id[e.u];                                  97ae
                e.v = id[e.v];                                  fae6
                if (e.u != e.v) e.w -= laz;                     bdd2
            }                                                   95cf
            n = cnt; rt = id[rt]; cnt = 0;                      6cc4
        }                                                       95cf
    }                                                           95cf
```

6.4 Maximum flow (Dinic)

Usage:

`add_edge(u, v, c)`	Add an edge from u to v with capacity c.
`max_flow(s, t)`	Compute maximum flow from s to t.

Time Complexity: For general graph, $O(V^2E)$; for network with unit capacity, $O(\min\{V^{2/3}, \sqrt{E}\}E)$; for bipartite network, $O(\sqrt{V}E)$.

```
struct edge{                                                    bcf8
    int from, to;                                               60e2
    LL cap, flow;                                               5e6d
};                                                              329b
                                                                427e
const int MAXN = 1005;                                          e2cd
struct Dinic {                                                  9062
    int n, m, s, t;                                             4dbf
    vector<edge> edges;                                         9f0c
    vector<int> G[MAXN];                                        b891
    bool vis[MAXN];                                             bbb6
    int d[MAXN];                                                b40a
    int cur[MAXN];                                              ddec
                                                                427e
    void add_edge(int from, int to, LL cap) {                   5973
```

```
        edges.push_back(edge{from, to, cap, 0});
        edges.push_back(edge{to, from, 0, 0});
        m = edges.size();
        G[from].push_back(m-2);
        G[to].push_back(m-1);
    }

    bool bfs() {
        memset(vis, 0, sizeof(vis));
        queue<int> q;
        q.push(s);
        vis[s] = 1;
        d[s] = 0;
        while (!q.empty()) {
            int x = q.front(); q.pop();
            for (int i = 0; i < G[x].size(); i++) {
                edge& e = edges[G[x][i]];
                if (!vis[e.to] && e.cap > e.flow) {
                    vis[e.to] = 1;
                    d[e.to] = d[x] + 1;
                    q.push(e.to);
                }
            }
        }
        return vis[t];
    }

    LL dfs(int x, LL a) {
        if (x == t || a == 0) return a;
        LL flow = 0, f;
        for (int& i = cur[x]; i < G[x].size(); i++) {
            edge& e = edges[G[x][i]];
            if(d[x] + 1 == d[e.to] && (f = dfs(e.to, min(a, e.cap-e.flow))) > 0) {
                e.flow += f;
                edges[G[x][i]^1].flow -= f;
                flow += f;
                a -= f;
                if(a == 0) break;
            }
        }
        return flow;
    }

    LL max_flow(int s, int t) {
        this->s = s; this->t = t;
        LL flow = 0;
        while (bfs()) {
            memset(cur, 0, sizeof(cur));
            flow += dfs(s, LLONG_MAX);
        }
        return flow;
    }

    vector<int> min_cut() { // call this after maxflow
        vector<int> ans;
        for (int i = 0; i < edges.size(); i++) {
            edge& e = edges[i];
            if(vis[e.from] && !vis[e.to] && e.cap > 0) ans.push_back(i);
        }
        return ans;
    }
};
```

6.5 Maximum cardinality bipartite matching (Hungarian)

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

#define rep(i, n) for (int i = 0; i < (n); i++)
#define Rep(i, n) for (int i = 1; i <= (n); i++)
#define range(x) (x).begin(), (x).end()
typedef long long LL;

struct Hungarian{
    int nx, ny;
    vector<int> mx, my;
    vector<vector<int> > e;
    vector<bool> mark;

    void init(int nx, int ny){
        this->nx = nx;
        this->ny = ny;
        mx.resize(nx); my.resize(ny);
        e.clear(); e.resize(nx);
        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  };
```

6.6 Minimum cost maximum flow

```
bcf8  struct edge{
60e2      int from, to;
d698      int cap, flow;
32cc      LL cost;
329b  };
427e
cc3e  const LL INF = LLONG_MAX / 2;
2aa8  const int MAXN = 5005;
c6cb  struct MCMF {
9ceb      int s, t, n, m;
9f0c      vector<edge> edges;
b891      vector<int> G[MAXN];
f74f      bool inq[MAXN]; // queue
8f67      LL d[MAXN];     // distance
9524      int p[MAXN];    // previous
b330      int a[MAXN];    // improvement
427e
f7f2      void add_edge(int from, int to, int cap, LL cost) {
24f0          edges.push_back(edge{from, to, cap, 0, cost});
95f0          edges.push_back(edge{to, from, 0, 0, -cost});
fe77          m = edges.size();
dff5          G[from].push_back(m-2);
8f2d          G[to].push_back(m-1);
95cf      }
427e
3c52      bool spfa(){
93d2          queue<int> q;
8494          fill(d, d + MAXN, INF); d[s] = 0;
fd48          memset(inq, 0, sizeof(inq));
5e7c          q.push(s); inq[s] = true;
2dae          p[s] = 0; a[s] = INT_MAX;
cc78          while (!q.empty()){
b0aa              int u = q.front(); q.pop(); inq[u] = false;
ddff              rep (i, G[u].size()){
c234                  edge& e = edges[G[u][i]];
3601                  if (e.cap > e.flow && d[e.to] > d[u] + e.cost){
55bc                      d[e.to] = d[u] + e.cost;
0bea                      p[e.to] = G[u][i];
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  };
```

6.7 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  
```

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

6.8 Heavy-light decomposition

```
const int MAXN = 100005;                                                0f42
vector<int> adj[MAXN];                                                  0b32
int sz[MAXN], top[MAXN], fa[MAXN], son[MAXN], depth[MAXN], id[MAXN];    42f2
                                                                        427e
void dfs1(int x, int dep, int par){                                     be5c
    depth[x] = dep;                                                     7489
    sz[x] = 1;                                                          2ee7
    fa[x] = par;                                                        adb4
    int maxn = 0, s = 0;                                                b79d
    for (int c: adj[x]){                                                c861
        if (c == par) continue;                                         fe45
        dfs1(c, dep + 1, x);                                            fd2f
        sz[x] += sz[c];                                                 b790
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      }
6083      if (depth[u] > depth[v]) swap(u, v);
427e      // id[u] to id[v]
95cf  }
```

7 Data Structures

7.1 Segment tree

```
3942  LL p;
1ebb  const int MAXN = 4 * 100006;
451a  struct segtree {
27be    int l[MAXN], m[MAXN], r[MAXN];
4510    LL val[MAXN], tadd[MAXN], tmul[MAXN];
427e
ac35  #define lson (o<<1)
1294  #define rson (o<<1|1)
427e
1344    void pull(int o) {
bbe9      val[o] = (val[lson] + val[rson]) % p;
95cf    }
427e
e4bc    void push_add(int o, LL x) {
5dd6      val[o] = (val[o] + x * (r[o] - l[o])) % p;
6eff      tadd[o] = (tadd[o] + x) % p;
95cf    }
427e
d658    void push_mul(int o, LL x) {
b82c      val[o] = val[o] * x % p;
aa86      tadd[o] = tadd[o] * x % p;
649f      tmul[o] = tmul[o] * x % p;
95cf    }
427e
b149    void push(int o) {
3159      if (l[o] == m[o]) return;
0a90      if (tmul[o] != 1) {
0f4a        push_mul(lson, tmul[o]);
045e        push_mul(rson, tmul[o]);
ac0a        tmul[o] = 1;
95cf      }
1b82      if (tadd[o]) {
9547        push_add(lson, tadd[o]);
0e73        push_add(rson, tadd[o]);
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 {
f7ff       LL ans = 0;
c4b0       push(o);
c5f8       if (m[o] > ll) ans += query(lson, ll, rr);
ef81       if (m[o] < rr) ans += query(rson, ll, rr);
a420       return ans % p;
95cf     }
95cf   }
4d99 } seg;
```

7.2 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
    if (w) fa[w] = y;                                              fa6f
    fa[y] = x; fa[x] = z;                                          3540
    pull(y);                                                       72ef
  }                                                                95cf
                                                                   427e
  void pushall(int x) {                                            bc1b
    if (isroot(x)) pushall(fa[x]);                                 a316
    push(x);                                                       a97b
  }                                                                95cf
                                                                   427e
  void splay(int x) {                                              f69c
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]) {
a97b        push(x);
b83a        x = ch[x][0];
95cf      }
8fec      splay(x);
d074      return x;
95cf    }
427e
70d3    void Link(int u, int v) {  // assume unconnected before
b8a5      chroot(u);
2448      fa[u] = v;
95cf    }
427e
c2f4    void Cut(int u, int v) {  // assume connected before
e8ce      split(u, v);
fd95      fa[u] = ch[v][0] = 0;
743b      pull(v);
95cf    }
427e
6ca2    int Query(int u, int v) {
e8ce      split(u, v);
a5ba      return sum[v];
95cf    }
427e
eaba    void Update(int u, int x) {
46ce      splay(u);
1d62      val[u] = x;
95cf    }
329b  };
```

7.3 Balanced binary search tree from `pb_ds`

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

7.4 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;
181e      int mid = (l + r) / 2;
cb5a      if (rt->left->value - lt->left->value < k) {
8edb        k -= rt->left->value - lt->left->value;
2412        return Query(lt->right, rt->right, mid, r, k);
8e2e      } else {
0119        return Query(lt->left, rt->left, l, mid, k);
95cf      }
95cf    }
427e
c9ad    static int query(node* lt, node *rt, int k) {
9e27      return Query(lt, rt, 0, n, k);
95cf    }
427e
b19c    node *Inc(int l, int r, int pos) const {
5794      node* a = new node(*this);
ce96      if (r > l + 1) {
181e        int mid = (l + r) / 2;
203d        if (pos < mid)
f44a          a->left = left->Inc(l, mid, pos);
649a        else
1024          a->right = right->Inc(mid, r, pos);
95cf      }
2b3e      a->value++;
5ffd      return a;
95cf    }
427e
e80f    node *inc(int index) {
c246      return Inc(0, n, index);
95cf    }
865a  } nodes[8000000];
427e
99ce  int node::n, node::pos;
1987  inline void* node::operator new(size_t size) {
bb3c    return nodes + (pos++);
95cf  }
```

7.5 Sparse table, range extremum query

The array is 0-based and the range is closed.

```
db63  const int MAXN = 100007;
b330  int a[MAXN];
69ae  int st[MAXN][32 - __builtin_clz(MAXN)];
427e
8041  inline int ext(int x, int y){return x>y?x:y;} // ! max
427e
d34f  void init(int n){
ce01      int l = 31 - __builtin_clz(n);
cf75      rep (i, n) st[i][0] = a[i];
b811      rep (j, l)
6937          rep (i, 1+n-(1<<j))
082a              st[i][j+1] = ext(st[i][j], st[i+(1<<j)][j]);
95cf  }
427e
c863  int rmq(int l, int r){
92f5      int k = 31 - __builtin_clz(r-l+1);
baa2      return ext(st[l][k], st[r-(1<<k)+1][k]);
95cf  }
```

8 Geometrics

8.1 2D geometric template

```
302f #include <bits/stdc++.h>
421c using namespace std;
427e
4553 typedef int T;
c0ae typedef struct pt {
7a9d     T x, y;
ffaa     T operator , (pt a) { return x*a.x + y*a.y; } // inner product
3ec7     T operator * (pt a) { return x*a.y - y*a.x; } // outer product
221a     pt operator + (pt a) { return {x+a.x, y+a.y}; }
8b34     pt operator - (pt a) { return {x-a.x, y-a.y}; }
427e
368b     pt operator * (T k) { return {x*k, y*k}; }
90f4     pt operator - () { return {-x, -y};}
ba8c } vec;
427e
0ea6 typedef pair<pt, pt> seg;
427e
8d6e bool ptOnSeg(pt& p, seg& s){
ce77     vec v1 = s.first - p, v2 = s.second - p;
de97     return (v1, v2) <= 0 && v1 * v2 == 0;
95cf }
427e
427e // 0 not on segment
427e // 1 on segment except vertices
427e // 2 on vertices
8421 int ptOnSeg2(pt& p, seg& s){
ce77     vec v1 = s.first - p, v2 = s.second - p;
70ca     T ip = (v1, v2);
8b14     if (v1 * v2 != 0 || ip > 0) return 0;
0847     return (v1, v2) ? 1 : 2;
95cf }
427e
427e // if two orthogonal rectangles do not touch, return true
72bb inline bool nIntRectRect(seg a, seg b){
f9ac     return min(a.first.x, a.second.x) > max(b.first.x, b.second.x) ||
f486            min(a.first.y, a.second.y) > max(b.first.y, b.second.y) ||
39ce            min(b.first.x, b.second.x) > max(a.first.x, a.second.x) ||
80c7            min(b.first.y, b.second.y) > max(a.first.y, a.second.y);
95cf }
427e
427e // >0 in order
427e // <0 out of order
427e // =0 not standard
7538 inline double rotOrder(vec a, vec b, vec c){return double(a*b)*(b*c);}
427e
31ed inline bool intersect(seg a, seg b){
427e     // ! if (nIntRectRect(a, b)) return false; // if commented, assume that a
           and b are non-collinear
cb52     return rotOrder(b.first-a.first, a.second-a.first, b.second-a.first) >= 0 &&
059e            rotOrder(a.first-b.first, b.second-b.first, a.second-b.first) >= 0;
95cf }
427e
427e // 0 not insersect
427e // 1 standard intersection
427e // 2 vertex-line intersection
427e // 3 vertex-vertex intersection
427e // 4 collinear and have common point(s)
4d19 int intersect2(seg& a, seg& b){
5dc4     if (nIntRectRect(a, b)) return 0;
42c0     vec va = a.second - a.first, vb = b.second - b.first;
2096     double j1 = rotOrder(b.first-a.first, va, b.second-a.first),
72fe            j2 = rotOrder(a.first-b.first, vb, a.second-b.first);
5ac6     if (j1 < 0 || j2 < 0) return 0;
9400     if (j1 != 0 && j2 != 0) return 1;
83db     if (j1 == 0 && j2 == 0){
6b0c         if (va * vb == 0) return 4; else return 3;
fb17     } else return 2;
95cf }
427e
2c68 template <typename Tp = T>
5894 inline pt getIntersection(pt P, vec v, pt Q, vec w){
6850     static_assert(is_same<Tp, double>::value, "must␣be␣double!");
7c9a     return P + v * (w*(P-Q)/(v*w));
95cf }
427e
427e // -1 outside the polygon
427e // 0  on the border of the polygon
427e // 1  inside the polygon
cbdd int ptOnPoly(pt p, pt* poly, int n){
5fb4     int wn = 0;
1294     for (int i = 0; i < n; i++) {
427e
3cae         T k, d1 = poly[i].y - p.y, d2 = poly[(i+1)%n].y - p.y;
b957         if (k = (poly[(i+1)%n] - poly[i])*(p - poly[i])){
8c40             if (k > 0 && d1 <= 0 && d2 > 0) wn++;
3c4d             if (k < 0 && d2 <= 0 && d1 > 0) wn--;
aad3         } else return 0;
95cf     }
0a5f     return wn ? 1 : -1;
95cf }
427e
d4a3 istream& operator >> (istream& lhs, pt& rhs){
fa86     lhs >> rhs.x >> rhs.y;
331a     return lhs;
95cf }
427e
07ae istream& operator >> (istream& lhs, seg& rhs){
5cab     lhs >> rhs.first >> rhs.second;
331a     return lhs;
95cf }
```

9 Appendices

9.1 Primes

9.1.1 First primes

p	$g(p)$	p	$g(p)$	p	$g(p)$	p	$g(p)$	p	$g(p)$
2	1	3	2	5	2	7	3	11	2
13	2	17	3	19	2	23	5	29	2
31	3	37	2	41	6	43	3	47	5
53	2	59	2	61	2	67	2	71	7
73	5	79	3	83	2	89	3	97	5
101	2	103	5	107	2	109	6	113	3
127	3	131	2	137	3	139	2	149	2
151	6	157	5	163	2	167	5	173	2
179	2	181	2	191	19	193	5	197	2
199	3	211	2	223	3	227	2	229	6

9.1.2 Arbitrary length primes

$\lg p$	p	$g(p)$	p	$g(p)$
3	967	5	1031	14
4	9859	2	10273	10
5	96331	10	102931	3
6	958543	6	1031137	5
7	9594539	2	10169651	2
8	96243449	3	103211039	7
9	980483981	2	1042484357	2
10	9858935453	2	10261276009	7
11	95748666809	3	101759940101	2
12	950781833849	3	1012797784423	5
13	9739822952371	7	10037217092377	7
14	96181051140397	5	104974966380359	11
15	981030138360889	13	1029038416465403	2
16	9655206098080843	3	10116299875820773	2
17	97687777921994419	3	101506415998163437	2

9.1.3 $\sim 1 \times 10^9$

p	$g(p)$	p	$g(p)$	p	$g(p)$
954854573	3	967607731	2	973215833	3
975831713	3	978949117	2	980766497	3
983879921	3	985918807	3	986608921	29
991136977	5	991752599	13	997137961	11
1003911991	3	1009775293	2	1012423549	6
1021000537	5	1023976897	7	1024153643	2
1037027287	3	1038812881	11	1044754639	3
1045125617	3	1047411427	3	1047753349	6

9.1.4 $\sim 1 \times 10^{18}$

p	$g(p)$	p	$g(p)$
951970612352230049	3	963284339889659609	3
967495386904694119	3	969751761517096213	2
983238274281901499	2	984647442475101409	23
989286107138674069	11	1002507954383424641	3
1006658951440146419	2	1020152326159075903	3
1034876265966119449	7	1042753851435034019	2
1043609016597371563	2	1045571042176595707	2
1048364250160580293	2	1049495624119026949	2

9.2 Pell's equation

$x^2 - ny^2 = 1$, where n is a positive nonsquare integer.
Let (x_0, y_0) be the smallest positive solution of the equation, then the k-th solution is:

$$\begin{pmatrix} x_k \\ y_k \end{pmatrix} = \begin{pmatrix} x_0 & ny_0 \\ y_0 & x_0 \end{pmatrix}^k \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$$

Some smallest solutions to Pell's equation:

n	2	3	5	6	7	8	10	11	12	13	14	15	17	18	19	20
x	3	2	9	5	8	3	19	10	7	649	15	4	33	17	170	9
y	2	1	4	2	3	1	6	3	2	180	4	1	8	4	39	2

9.3 Burnside's lemma and Polya's enumeration theorem

The Burnside's lemma says that

$$|X/G| = \frac{1}{|G|} \sum_{g \in G} |X^g|$$

where G is a group acting on X, X^g is the set of elements in X that are fixed by g, i.e. $X^g = \{x \in X : gx = x\}$.
The unweighted version of Pólya enumeration theorem says that

$$|Y^X/G| = \frac{1}{|G|} \sum_{g \in G} m^{c_g}$$

where $m = |X|$ is the number of colors, c_g is the number of the cycles of permutation g.

9.4 Lagrange interpolation formula

Given sample points $(x_1, y_1), \cdots, (x_k, y_k)$, the interpolation polynomial is

$$L(x) = \sum_{j=1}^{k} y_j l_j(x)$$

where

$$l_j(x) = \prod_{0 \le m \le k, m \ne j} \frac{x - x_m}{x_j - x_m}$$