Оглавление

0.1 aho-korasik.cpp

```
1
2  struct node{
3      //vector<int> curword;
4      int pch = -1; //char by which you came there
5      node* par = nullptr;
6      node* link = nullptr;
7      map<int, node*> sons;
8      map<int, node*> go;
9      bool terminal = false;
10     bool goes_to_terminal = false;
11 };
12 const int maxnodes = 100 * 1000 + 10;
13 node nodes[maxnodes];
14 node* nodeptr = nodes+1;
15
16 void add_word(vector<int>& w){
17     node* v = nodes;
18     for (auto& c : w){
19         if (v->sons.count(c) == 0) v->sons[c] =
              nodeptr++; //if no such son, add
20         v->sons[c]->pch = c; //setting parent
              character
21         v->sons[c]->par = v; //setting parent
              pointer
22         //v->sons[c]->curword = v->curword;
23         //v->sons[c]->curword.push_back(c);
24         v = v->sons[c]; //moving to son
25     }
26     v->terminal = true;
27 }
28
29 node* getSuffLink(node* v);
30 node* getLink(node* v, int c);
31 node* getSuffLink(node* v){ //suffix link
32     if (v->link) return v->link; //if calced, return
33     if (v == nodes || v->par == nodes) return v->
          link = nodes; //if root or near root, return
          root
```

```
34      return v->link = getLink(getSuffLink(v->par), v
           ->pch); //get suff link from parent
35  }
36  node* getLink(node* v, int c){ //transfer function
37      if (v->go.count(c)) return v->go[c]; //if calced
           , return
38      if (v->sons.count(c)) return v->go[c] = v->sons[
           c]; //if there is son, return
39      if (v == nodes) return v->go[c] = nodes;
40      return v->go[c] = getLink(getSuffLink(v), c);
41  }
42
43  void set_terminals(node* v){
44      node* sl = getSuffLink(v);
45      if (sl->terminal) v->goes_to_terminal = true;
46      if (sl->goes_to_terminal) v->goes_to_terminal =
           true;
47  }
48
49  void init_corasik(vector<vector<int>>& words){
50
51      nodeptr = nodes + 1;
52      fill(nodes, nodes + maxnodes, node());
53      for (auto& it : words) add_word(it);
54
55      queue<node*> q;
56      q.push(nodes);
57      while(q.size()){
58          node* v = q.front();
59          q.pop();
60          getSuffLink(v);
61          set_terminals(v);
62          for (auto& it : v->sons)
63              q.push(it.second);
64      }
65  }
```

0.2 dinic.cpp

```
1  struct Dinic{
2      int s, t;
```

```
3       int graphsize;
4       struct edge {
5           int to, cap, f = 0;
6           edge() {}
7           edge(int to, int cap) :to(to), cap(cap) {}
8       };
9       vector<vector<int>> g;
10      vector<int> d, cnt;
11      vector<edge> E;
12      bool bfs() {
13          queue<int> q;
14          q.push(s);
15          cnt.assign(graphsize, 0);
16          d.assign(graphsize, inf);
17          d[s] = 0;
18          while (q.size()) {
19              int v = q.front();
20              if (v == t) return true;
21              q.pop();
22              for (auto e : g[v]) {
23                  if (d[E[e].to] == inf && E[e].f < E[
                        e].cap) {
24                      q.push(E[e].to);
25                      d[E[e].to] = d[v] + 1;
26                  }
27              }
28          }
29          return false;
30      }
31      int dfs(int v, int f) {
32          if (v == t) return f;
33          for (; cnt[v] < (int)g[v].size(); ++cnt[v])
                {
34              int e = g[v][cnt[v]];
35              int to = E[e].to;
36              if (d[to] != d[v] + 1 || E[e].f == E[e].
                    cap) continue;
37              int flow = dfs(to, min(f, E[e].cap - E[e
                    ].f));
38              if (flow) {
39                  E[e].f += flow;
```

```
40                  E[e ^ 1].f -= flow;
41                  return flow;
42              }
43          }
44          return 0;
45      }
46
47      int maxflow() {
48          int f = 0;
49          int ans = 0;
50          while (bfs())
51              while ((f = dfs(s, inf)))
52                  ans += f;
53          return ans;
54      }
55
56      void add_edge(int f, int to, int cap) {
57          g[f].push_back(int(E.size()));
58          E.push_back(edge(to, cap));
59          g[to].push_back(int(E.size()));
60          E.push_back(edge(f, 0));
61      }
62      Dinic(){}
63      Dinic(int s, int t, int graphsize):s(s), t(t),
           graphsize(graphsize){
64          g.resize(graphsize);
65      }
66  };
```

0.3 bigint.cpp

```
1  const int base = 1000000000;
2  const int base_digits = 9;
3  struct bigint {
4      vector<int> a;
5      int sign;
6      /*<arpa>*/
7      int size(){
8          if(a.empty())return 0;
9          int ans=(a.size()-1)*base_digits;
10         int ca=a.back();
```

```
11          while(ca)
12              ans++,ca/=10;
13          return ans;
14      }
15      bigint operator ^(const bigint &v){
16          bigint ans=1,a=*this,b=v;
17          while(!b.isZero()){
18              if(b%2)
19                  ans*=a;
20              a*=a,b/=2;
21          }
22          return ans;
23      }
24      string to_string(){
25          stringstream ss;
26          ss << *this;
27          string s;
28          ss >> s;
29          return s;
30      }
31      int sumof(){
32          string s = to_string();
33          int ans = 0;
34          for(auto c : s)  ans += c - '0';
35          return ans;
36      }
37      /*</arpa>*/
38      bigint() :
39              sign(1) {
40      }
41
42      bigint(long long v) {
43          *this = v;
44      }
45
46      bigint(const string &s) {
47          read(s);
48      }
49
50      void operator=(const bigint &v) {
51          sign = v.sign;
```

```
52          a = v.a;
53      }
54
55      void operator=(long long v) {
56          sign = 1;
57          a.clear();
58          if (v < 0)
59              sign = -1, v = -v;
60          for (; v > 0; v = v / base)
61              a.push_back(v % base);
62      }
63
64      bigint operator+(const bigint &v) const {
65          if (sign == v.sign) {
66              bigint res = v;
67
68              for (int i = 0, carry = 0; i < (int) max
                    (a.size(), v.a.size()) || carry; ++i)
                     {
69                  if (i == (int) res.a.size())
70                      res.a.push_back(0);
71                  res.a[i] += carry + (i < (int) a.
                        size() ? a[i] : 0);
72                  carry = res.a[i] >= base;
73                  if (carry)
74                      res.a[i] -= base;
75              }
76              return res;
77          }
78          return *this - (-v);
79      }
80
81      bigint operator-(const bigint &v) const {
82          if (sign == v.sign) {
83              if (abs() >= v.abs()) {
84                  bigint res = *this;
85                  for (int i = 0, carry = 0; i < (int)
                        v.a.size() || carry; ++i) {
86                      res.a[i] -= carry + (i < (int) v
                            .a.size() ? v.a[i] : 0);
87                      carry = res.a[i] < 0;
```

```
88                      if (carry)
89                          res.a[i] += base;
90                  }
91                  res.trim();
92                  return res;
93              }
94              return -(v - *this);
95          }
96          return *this + (-v);
97      }
98
99      void operator*=(int v) {
100         if (v < 0)
101             sign = -sign, v = -v;
102         for (int i = 0, carry = 0; i < (int) a.size
               () || carry; ++i) {
103             if (i == (int) a.size())
104                 a.push_back(0);
105             long long cur = a[i] * (long long) v +
                   carry;
106             carry = (int) (cur / base);
107             a[i] = (int) (cur % base);
108             //asm("divl %%ecx" : "=a"(carry), "=d"(a
                   [i]) : "A"(cur), "c"(base));
109         }
110         trim();
111     }
112
113     bigint operator*(int v) const {
114         bigint res = *this;
115         res *= v;
116         return res;
117     }
118
119     void operator*=(long long v) {
120         if (v < 0)
121             sign = -sign, v = -v;
122         if(v > base){
123             *this = *this * (v / base) * base + *
                   this * (v % base);
124             return ;
```

```
125         }
126         for (int i = 0, carry = 0; i < (int) a.size
               () || carry; ++i) {
127             if (i == (int) a.size())
128                 a.push_back(0);
129             long long cur = a[i] * (long long) v +
                   carry;
130             carry = (int) (cur / base);
131             a[i] = (int) (cur % base);
132             //asm("divl %%ecx" : "=a"(carry), "=d"(a
                   [i]) : "A"(cur), "c"(base));
133         }
134         trim();
135     }
136
137     bigint operator*(long long v) const {
138         bigint res = *this;
139         res *= v;
140         return res;
141     }
142
143     friend pair<bigint, bigint> divmod(const bigint
           &a1, const bigint &b1) {
144         int norm = base / (b1.a.back() + 1);
145         bigint a = a1.abs() * norm;
146         bigint b = b1.abs() * norm;
147         bigint q, r;
148         q.a.resize(a.a.size());
149
150         for (int i = a.a.size() - 1; i >= 0; i--) {
151             r *= base;
152             r += a.a[i];
153             int s1 = r.a.size() <= b.a.size() ? 0 :
                   r.a[b.a.size()];
154             int s2 = r.a.size() <= b.a.size() - 1 ?
                   0 : r.a[b.a.size() - 1];
155             int d = ((long long) base * s1 + s2) / b
                   .a.back();
156             r -= b * d;
157             while (r < 0)
158                 r += b, --d;
```

```
159             q.a[i] = d;
160         }
161
162         q.sign = a1.sign * b1.sign;
163         r.sign = a1.sign;
164         q.trim();
165         r.trim();
166         return make_pair(q, r / norm);
167     }
168
169     bigint operator/(const bigint &v) const {
170         return divmod(*this, v).first;
171     }
172
173     bigint operator%(const bigint &v) const {
174         return divmod(*this, v).second;
175     }
176
177     void operator/=(int v) {
178         if (v < 0)
179             sign = -sign, v = -v;
180         for (int i = (int) a.size() - 1, rem = 0; i
               >= 0; --i) {
181             long long cur = a[i] + rem * (long long)
                    base;
182             a[i] = (int) (cur / v);
183             rem = (int) (cur % v);
184         }
185         trim();
186     }
187
188     bigint operator/(int v) const {
189         bigint res = *this;
190         res /= v;
191         return res;
192     }
193
194     int operator%(int v) const {
195         if (v < 0)
196             v = -v;
197         int m = 0;
```

```
198         for (int i = a.size() - 1; i >= 0; --i)
199             m = (a[i] + m * (long long) base) % v;
200         return m * sign;
201     }
202
203     void operator+=(const bigint &v) {
204         *this = *this + v;
205     }
206     void operator-=(const bigint &v) {
207         *this = *this - v;
208     }
209     void operator*=(const bigint &v) {
210         *this = *this * v;
211     }
212     void operator/=(const bigint &v) {
213         *this = *this / v;
214     }
215
216     bool operator<(const bigint &v) const {
217         if (sign != v.sign)
218             return sign < v.sign;
219         if (a.size() != v.a.size())
220             return a.size() * sign < v.a.size() * v.
                sign;
221         for (int i = a.size() - 1; i >= 0; i--)
222             if (a[i] != v.a[i])
223                 return a[i] * sign < v.a[i] * sign;
224         return false;
225     }
226
227     bool operator>(const bigint &v) const {
228         return v < *this;
229     }
230     bool operator<=(const bigint &v) const {
231         return !(v < *this);
232     }
233     bool operator>=(const bigint &v) const {
234         return !(*this < v);
235     }
236     bool operator==(const bigint &v) const {
237         return !(*this < v) && !(v < *this);
```

```
238     }
239     bool operator!=(const bigint &v) const {
240         return *this < v || v < *this;
241     }
242
243     void trim() {
244         while (!a.empty() && !a.back())
245             a.pop_back();
246         if (a.empty())
247             sign = 1;
248     }
249
250     bool isZero() const {
251         return a.empty() || (a.size() == 1 && !a[0])
                ;
252     }
253
254     bigint operator-() const {
255         bigint res = *this;
256         res.sign = -sign;
257         return res;
258     }
259
260     bigint abs() const {
261         bigint res = *this;
262         res.sign *= res.sign;
263         return res;
264     }
265
266     long long longValue() const {
267         long long res = 0;
268         for (int i = a.size() - 1; i >= 0; i--)
269             res = res * base + a[i];
270         return res * sign;
271     }
272
273     friend bigint gcd(const bigint &a, const bigint
            &b) {
274         return b.isZero() ? a : gcd(b, a % b);
275     }
276     friend bigint lcm(const bigint &a, const bigint
```

```
        &b) {
277         return a / gcd(a, b) * b;
278     }
279
280     void read(const string &s) {
281         sign = 1;
282         a.clear();
283         int pos = 0;
284         while (pos < (int) s.size() && (s[pos] == '-
                ' || s[pos] == '+')) {
285             if (s[pos] == '-')
286                 sign = -sign;
287             ++pos;
288         }
289         for (int i = s.size() - 1; i >= pos; i -=
                base_digits) {
290             int x = 0;
291             for (int j = max(pos, i - base_digits +
                    1); j <= i; j++)
292                 x = x * 10 + s[j] - '0';
293             a.push_back(x);
294         }
295         trim();
296     }
297
298     friend istream& operator>>(istream &stream,
            bigint &v) {
299         string s;
300         stream >> s;
301         v.read(s);
302         return stream;
303     }
304
305     friend ostream& operator<<(ostream &stream,
            const bigint &v) {
306         if (v.sign == -1)
307             stream << '-';
308         stream << (v.a.empty() ? 0 : v.a.back());
309         for (int i = (int) v.a.size() - 2; i >= 0;
                --i)
310             stream << setw(base_digits) << setfill('
```

```
                0') << v.a[i];
311         return stream;
312     }
313
314     static vector<int> convert_base(const vector<int
           > &a, int old_digits, int new_digits) {
315         vector<long long> p(max(old_digits,
               new_digits) + 1);
316         p[0] = 1;
317         for (int i = 1; i < (int) p.size(); i++)
318             p[i] = p[i - 1] * 10;
319         vector<int> res;
320         long long cur = 0;
321         int cur_digits = 0;
322         for (int i = 0; i < (int) a.size(); i++) {
323             cur += a[i] * p[cur_digits];
324             cur_digits += old_digits;
325             while (cur_digits >= new_digits) {
326                 res.push_back(int(cur % p[new_digits
                       ]));
327                 cur /= p[new_digits];
328                 cur_digits -= new_digits;
329             }
330         }
331         res.push_back((int) cur);
332         while (!res.empty() && !res.back())
333             res.pop_back();
334         return res;
335     }
336
337     typedef vector<long long> vll;
338
339     static vll karatsubaMultiply(const vll &a, const
           vll &b) {
340         int n = a.size();
341         vll res(n + n);
342         if (n <= 32) {
343             for (int i = 0; i < n; i++)
344                 for (int j = 0; j < n; j++)
345                     res[i + j] += a[i] * b[j];
346             return res;
```

```
347         }
348
349         int k = n >> 1;
350         vll a1(a.begin(), a.begin() + k);
351         vll a2(a.begin() + k, a.end());
352         vll b1(b.begin(), b.begin() + k);
353         vll b2(b.begin() + k, b.end());
354
355         vll a1b1 = karatsubaMultiply(a1, b1);
356         vll a2b2 = karatsubaMultiply(a2, b2);
357
358         for (int i = 0; i < k; i++)
359             a2[i] += a1[i];
360         for (int i = 0; i < k; i++)
361             b2[i] += b1[i];
362
363         vll r = karatsubaMultiply(a2, b2);
364         for (int i = 0; i < (int) a1b1.size(); i++)
365             r[i] -= a1b1[i];
366         for (int i = 0; i < (int) a2b2.size(); i++)
367             r[i] -= a2b2[i];
368
369         for (int i = 0; i < (int) r.size(); i++)
370             res[i + k] += r[i];
371         for (int i = 0; i < (int) a1b1.size(); i++)
372             res[i] += a1b1[i];
373         for (int i = 0; i < (int) a2b2.size(); i++)
374             res[i + n] += a2b2[i];
375         return res;
376     }
377
378     bigint operator*(const bigint &v) const {
379         vector<int> a6 = convert_base(this->a,
              base_digits, 6);
380         vector<int> b6 = convert_base(v.a,
              base_digits, 6);
381         vll a(a6.begin(), a6.end());
382         vll b(b6.begin(), b6.end());
383         while (a.size() < b.size())
384             a.push_back(0);
385         while (b.size() < a.size())
```

```
386             b.push_back(0);
387         while (a.size() & (a.size() - 1))
388             a.push_back(0), b.push_back(0);
389         vll c = karatsubaMultiply(a, b);
390         bigint res;
391         res.sign = sign * v.sign;
392         for (int i = 0, carry = 0; i < (int) c.size
               (); i++) {
393             long long cur = c[i] + carry;
394             res.a.push_back((int) (cur % 1000000));
395             carry = (int) (cur / 1000000);
396         }
397         res.a = convert_base(res.a, 6, base_digits);
398         res.trim();
399         return res;
400     }
401 };
```

0.4 stdc++.cpp

```
1  // C++ includes used for precompiling -*- C++ -*-
2
3  // Copyright (C) 2003-2014 Free Software Foundation,
       Inc.
4  //
5  // This file is part of the GNU ISO C++ Library.
       This library is free
6  // software; you can redistribute it and/or modify
       it under the
7  // terms of the GNU General Public License as
       published by the
8  // Free Software Foundation; either version 3, or (
       at your option)
9  // any later version.
10
11 // This library is distributed in the hope that it
       will be useful,
12 // but WITHOUT ANY WARRANTY; without even the
       implied warranty of
13 // MERCHANTABILITY or FITNESS FOR A PARTICULAR
       PURPOSE.  See the
```

```
14 // GNU General Public License for more details.
15
16 // Under Section 7 of GPL version 3, you are granted
       additional
17 // permissions described in the GCC Runtime Library
      Exception, version
18 // 3.1, as published by the Free Software Foundation
      .
19
20 // You should have received a copy of the GNU
      General Public License and
21 // a copy of the GCC Runtime Library Exception along
       with this program;
22 // see the files COPYING3 and COPYING.RUNTIME
      respectively.  If not, see
23 // <http://www.gnu.org/licenses/>.
24
25 /** @file stdc++.h
26  *  This is an implementation file for a precompiled
       header.
27  */
28
29 // 17.4.1.2 Headers
30
31 // C
32 #ifndef _GLIBCXX_NO_ASSERT
33 #include <cassert>
34 #endif
35 #include <cctype>
36 #include <cerrno>
37 #include <cfloat>
38 #include <ciso646>
39 #include <climits>
40 #include <clocale>
41 #include <cmath>
42 #include <csetjmp>
43 #include <csignal>
44 #include <cstdarg>
45 #include <cstddef>
46 #include <cstdio>
47 #include <cstdlib>
```

```
48 #include <cstring>
49 #include <ctime>
50
51 #if __cplusplus >= 201103L
52 #include <ccomplex>
53 #include <cfenv>
54 #include <cinttypes>
55 #include <cstdalign>
56 #include <cstdbool>
57 #include <cstdint>
58 #include <ctgmath>
59 #include <cwchar>
60 #include <cwctype>
61 #endif
62
63 // C++
64 template <class T>
65 T __gcd(T a, T b) { return b? __gcd(b, a%b) : a; }
66 #include <algorithm>
67 #include <unordered_map>
68 #include <unordered_set>
69 #include <random>
70 #include <bitset>
71 #include <complex>
72 #include <deque>
73 #include <exception>
74 #include <fstream>
75 #include <functional>
76 #include <iomanip>
77 #include <ios>
78 #include <iosfwd>
79 #include <iostream>
80 #include <istream>
81 #include <iterator>
82 #include <limits>
83 #include <list>
84 #include <locale>
85 #include <map>
86 #include <memory>
87 #include <new>
88 #include <numeric>
```

```
89 #include <ostream>
90 #include <queue>
91 #include <set>
92 #include <sstream>
93 #include <stack>
94 #include <stdexcept>
95 #include <streambuf>
96 #include <string>
97 #include <typeinfo>
98 #include <utility>
99 #include <valarray>
100 #include <vector>
101
102 #if __cplusplus >= 201103L
103 #include <array>
104 #include <atomic>
105 #include <chrono>
106 #include <condition_variable>
107 #include <forward_list>
108 #include <future>
109 #include <initializer_list>
110 #include <mutex>
111 #include <random>
112 #include <ratio>
113 #include <regex>
114 #include <scoped_allocator>
115 #include <system_error>
116 #include <thread>
117 #include <tuple>
118 #include <typeindex>
119 #include <type_traits>
120 #include <unordered_map>
121 #include <unordered_set>
122 #endif
```

0.5 heavy-light-decomposition.cpp

```
1 struct stree{
2     int n = 1;
3     vector<int> t;
4     stree(){}
```

```
5       stree(vector<int>& v){
6           while(n < (int)v.size()) n *= 2;
7           t.resize(2*n, inf);
8           for (int i = 0; i < (int)v.size(); ++i) t[i+
              n] = v[i];
9           for (int i = n-1; i; --i) t[i] = min(t[i+i],
                t[i+i+1]);
10      }
11      void set(int i, int x){
12          t[i+=n] = x;
13          for (i/=2; i; i/=2) t[i] = min(t[i+i], t[i+i
              +1]);
14      }
15      int get(int l, int r){
16          int res = inf;
17          for (l+=n, r+=n; l<=r; l/=2, r/=2){
18              if (l % 2 == 1) res = min(res, t[l++]);
19              if (r % 2 == 0) res = min(res, t[r--]);
20          }
21          return res;
22      }
23  };
24
25  struct HeavyLightDecomposition{
26      vector<vector<int>> g;          //your instance
            of graph
27      vector<stree> dec;              //structures:
            trees or tables or whatever
28      vector<int> conn;               //vertex to
            which structs connect
29      vector<pair<int,int>> ind;      //indexes of
            vertices: number of struct and index inside
            struct
30      vector<int> sz;                 //sizes of
            subtrees;
31      vector<int> dep;                //depth of
            vertices;
32      static const int maxpw = 25;    //maximum depth
            of graph is 2^maxpw
33      vector<array<int, maxpw>> par;  //parents for
            lca
```

```
34
35      void dfs(int v, int p, int d){
36          //for calculating lca, sz, dep
37          par[v][0] = p;
38          dep[v] = d;
39          for (int i = 1; i < maxpw; ++i)
40              par[v][i] = par[par[v][i-1]][i-1];
41          for (auto& to : g[v]){
42              if (to == p) continue;
43              dfs(to, v, d+1);
44              sz[v] += sz[to];
45          }
46      }
47
48      void dfs_build(int v, int p, vector<int>& tmp,
           vector<int>& vals){
49          //for building decomposition itself
50          if ((int)tmp.size() == 0) conn.push_back(p);
51          ind[v] = make_pair((int)dec.size(), (int)tmp
               .size());
52          tmp.push_back(vals[v]);
53          sort(g[v].begin(), g[v].end(), [&](int a,
               int b){return sz[a] > sz[b];});
54
55          int sons = 0;
56          for (auto& to : g[v]){
57              if (to == p) continue;
58              if (sons != 0) tmp = vector<int>();
59              dfs_build(to, v, tmp, vals);
60              ++sons;
61          }
62          if (sons == 0) dec.push_back(stree(tmp));
63      }
64
65      HeavyLightDecomposition(){}
66      HeavyLightDecomposition(vector<vector<int>>& gg,
            vector<int> vals){
67          //takes graph and values in vertices
68          g = gg;
69          int n = (int)g.size();
70          sz.assign(n, 1);
```

```
71          dep.resize(n);
72          ind.resize(n);
73          par.resize(n);
74          dfs(0, 0, 0);
75          //for (auto& it : sz) cout << it << " ";
76          //cout << endl;
77          vector<int> tmp;
78          dfs_build(0, 0, tmp, vals);
79      }
80      int getpar(int v, int up){
81          //this is for lca
82          for (int curpw = maxpw - 1; curpw >= 0; --
               curpw)
83              if ((up >> curpw) & 1) v = par[v][curpw
                   ];
84          return v;
85      }
86      int LCA(int a, int b){
87          if (dep[a] < dep[b]) swap(a, b);
88          a = getpar(a, dep[a] - dep[b]);
89          if (a == b) return a;
90          for (int curpw = maxpw - 1; curpw >= 0; --
               curpw){
91              if (par[a][curpw] == par[b][curpw])
                   continue;
92              a = par[a][curpw];
93              b = par[b][curpw];
94          }
95          return par[a][0];
96      }
97
98      void set(int v, int x){
99          dec[ind[v].first].set(ind[v].second, x);
100     }
101     int get(int a, int b){
102         //be careful with ranges for every .get
               operation
103         int c = LCA(a, b);
104         int res = inf;
105         while(ind[a].first != ind[c].first){
106             res = min(res, dec[ind[a].first].get(0,
```

```
                ind[a].second));
107             a = conn[ind[a].first];
108         }
109         if (a != c) res = min(res, dec[ind[a].first
              ].get(ind[c].second + 1, ind[a].second));
110         swap(a, b);
111         while(ind[a].first != ind[c].first){
112             res = min(res, dec[ind[a].first].get(0,
                ind[a].second));
113             a = conn[ind[a].first];
114         }
115         if (a != c) res = min(res, dec[ind[a].first
              ].get(ind[c].second + 1, ind[a].second));
116
117         //if decompositon is on edges, you dont need
              next line
118         //but you still need to give vertices values
119         res = min(res, dec[ind[c].first].get(ind[c].
              second, ind[c].second));
120         return res;
121     }
122
123 };
```

0.6 geom.cpp

```
1  #include <bits/stdc++.h>
2  //#define int int64_t
3  #define all(x) (x).begin(), (x).end()
4  #define out(x) return void(cout << x << endl)
5  #define OUT(x) ((cout << x), exit(0))
6  using namespace std;
7  typedef long double db;
8  const db eps = 1e-9;
9  const db pi = acos(-1.0);
10 const int64_t INF = (int64_t)(2e18);
11 const int inf = (int)(1e9 + 7);
12 //----------------------------------------//
13
14 bool eq(db a, db b) { return abs(a - b) < eps; }
15 struct pt {
```

```
16      db x, y;
17      pt() {}
18      pt(db x, db y) :x(x), y(y) {}
19      bool operator<(pt a){return eq(x, a.x)? y < a.y
           : x < a.x;}
20      pt operator+(pt b) { return pt(x + b.x, y + b.y)
           ; }
21      pt operator-(pt b) { return pt(x - b.x, y - b.y)
           ; }
22      pt operator*(db d) { return pt(x * d, y * d); }
23      pt operator/(db d) { return pt(x / d, y / d); }
24      pt& operator+=(pt b) { return *this = *this + b;
            }
25      pt& operator-=(pt b) { return *this = *this + b;
            }
26      pt& operator*=(db d) { return *this = *this * d;
            }
27      pt& operator/=(db d) { return *this = *this / d;
            }
28  };
29  pt operator*(db d, pt p) { return pt(p.x * d, p.y *
       d); }
30  ostream& operator<<(ostream& os, pt p) { return os
       << '[' << p.x << ",␣" << p.y << ']'; }
31  istream& operator>>(istream& is, pt& p) { return is
       >> p.x >> p.y; }
32
33  bool eq(pt a, pt b) { return eq(a.x, b.x) && eq(a.y,
        b.y); }
34  db dot(pt a, pt b) { return a.x * b.x + a.y * b.y; }
35  db cross(pt a, pt b) { return a.y * b.x - a.x * b.y;
        }
36  db dist2(pt a, pt b = pt(0, 0)) { return (a.x - b.x)
        * (a.x - b.x) + (a.y - b.y) * (a.y - b.y); }
37  db angle(pt a, pt b = pt(1, 0)) { return atan2(cross
       (a, b), dot(a, b)); }
38
39  db polygon_area(vector<pt> polygon) {
40      db res = 0;
41      for (int i = 0; i < (int)polygon.size(); ++i)
42          res += cross(polygon[i], polygon[(i + 1) %
```

```
            int(polygon.size())]);
    return abs(res / 2);
}

struct line {
    db a, b, c;
    line() {}
    line(db a, db b, db c) :a(a), b(b), c(c) {}
    line(pt p1, pt p2) :a(p1.y - p2.y), b(p2.x - p1.
       x), c(-a*p1.x - b*p1.y) {}
    db at(pt p) { return a * p.x + b * p.y + c; }
};
line normal(line l) {
    db mult = 1.0 / sqrt(l.a * l.a + l.b * l.b);
    return line(l.a * mult, l.b * mult, l.c * mult);
}
line bisector_line(pt a, pt b, pt c) {
    //line of angle abc
    pt v1 = (a - b); v1 /= sqrt(dist2(v1));
    pt v2 = (c - b); v2 /= sqrt(dist2(v2));
    return line(b, b + v1 + v2);
}
db dist(pt p, line l) { return abs(l.at(p) / sqrt(
   dist2(pt(l.a, l.b)))); }
db dist(line l, pt p) { return dist(p, l); }


struct beam {
    pt st, fn;
    line l;
    beam() {}
    beam(pt p1, pt p2) :st(p1), fn(p2), l(p1, p2) {}
};
db dist(pt p, beam b) {
    if (dot(p - b.st, b.fn - b.st) < 0) return sqrt(
       dist2(p, b.st));
    return dist(p, b.l);
}
db dist(beam b, pt p) { return dist(p, b); }

struct segment {
```

```
80      pt st, fn;
81      line l;
82      segment() {}
83      segment(pt pt1, pt pt2) :st(pt1), fn(pt2), l(pt1
            , pt2) {}
84  };
85  db dist(pt p, segment s) {
86      if (dot(p - s.st, s.fn - s.st) < 0) return sqrt(
            dist2(p, s.st));
87      if (dot(p - s.fn, s.st - s.fn) < 0) return sqrt(
            dist2(p, s.fn));
88      return dist(p, s.l);
89  }
90  db dist(segment b, pt p) { return dist(p, b); }
91
92  segment bounding_rectangle(segment s){
93      segment res;
94      res.st.x = min(s.st.x, s.fn.x);
95      res.st.y = min(s.st.y, s.fn.y);
96      res.fn.x = max(s.st.x, s.fn.x);
97      res.fn.y = max(s.st.y, s.fn.y);
98      return res;
99  }
100 bool intersect(segment s1, segment s2)
101 {
102     segment br1 = bounding_rectangle(s1);
103     segment br2 = bounding_rectangle(s2);
104     if ((br1.fn.x - br2.st.x) > -eps &&
105         (br2.fn.x - br1.st.x) > -eps &&
106         (br1.fn.y - br2.st.y) > -eps &&
107         (br2.fn.y - br1.st.y) > -eps)
108     {
109         db c1 = cross(s2.st - s1.st, s1.fn - s1.st)
                * cross(s2.fn - s1.st, s1.fn - s1.st);
110         db c2 = cross(s1.st - s2.st, s2.fn - s2.st)
                * cross(s1.fn - s2.st, s2.fn - s2.st);
111         if (c1 > -eps || c2 > -eps) return false;
112         return true;
113     }
114     return false;
115 }
```

```
116 db dist(segment s1, segment s2) {
117     if (intersect(s1,s2)) return 0.0;
118     db res = dist(s1.st, s2);
119     res = min(res, dist(s1.fn, s2));
120     res = min(res, dist(s2.st, s1));
121     res = min(res, dist(s2.fn, s1));
122     return res;
123 }
124 pt cross_point(segment a, segment b) {
125     pt l = a.st;
126     pt r = a.fn;
127     for (int i = 0; i < 100; ++i) {
128         if (eq(l, r)) break;
129         pt mid1 = (2 * l + r) / 3;
130         pt mid2 = (l + 2 * r) / 3;
131         if (dist(mid1, b) < dist(mid2, b)) r = mid2;
132         else l = mid1;
133     }
134     return r;
135 }
136 pt cross_point(line a, line b){
137     pt res;
138     res.x = (a.c * b.b - b.c * a.b) / (a.a * b.b - b
        .a * a.b);
139     res.y = (a.a * b.c - b.a * a.c) / (a.a * b.b - b
        .a * a.b);
140     return res;
141 }
142
143
144
145 int32_t main() {
146     ios_base::sync_with_stdio(false);
147     cout << fixed << setprecision(10);
148     cin.tie(nullptr);
149 #ifdef _MY
150     freopen("input.txt", "r", stdin);
151     freopen("output.txt", "w", stdout);
152 #endif
153 #ifndef _MY
154     freopen("line1.in", "r", stdin);
```

```
155     freopen("line1.out", "w", stdout);
156 #endif
157
158     pt a, b;
159     cin >> a >> b;
160     line l(a, b);
161     cout << l.a << "␣" << l.b << "␣" << l.c;
162
163
164
165     return 0;
166 }
```

0.7 ordered$_set$.cpp

```
1  #include <ext/pb_ds/assoc_container.hpp>
2  #include <ext/pb_ds/tree_policy.hpp>
3
4  using namespace __gnu_pbds;
5
6  typedef tree < ll, null_type, less < ll >,
     rb_tree_tag, tree_order_statistics_node_update >
     ordered_set;
7
8
9
10 ordered_set X;
11 X.insert(1);
12 X.insert(2);
13 X.insert(4);
14 X.insert(8);
15 X.insert(16);
16
17 cout<<*X.find_by_order(1)<<endl; // 2
18 cout<<*X.find_by_order(2)<<endl; // 4
19 cout<<*X.find_by_order(4)<<endl; // 16
20 cout<<(end(X)==X.find_by_order(6))<<endl; // true
21
22 cout<<X.order_of_key(-5)<<endl;  // 0
23 cout<<X.order_of_key(1)<<endl;   // 0
24 cout<<X.order_of_key(3)<<endl;   // 2
```

```
25 cout<<X.order_of_key(4)<<endl;   // 2
26 cout<<X.order_of_key(400)<<endl; // 5
```

0.8 gauss module.cpp

```
1  namespace Gauss {
2      typedef vector<int> vi;
3      const int inf = int(1e9 + 7);
4      int binpow(int a, int p) {
5          int res = 1;
6          for (; p; p >>= 1) {
7              if (p & 1)res = res * a % inf;
8              a = a * a % inf;
9          }
10         return res;
11     }
12
13     int sub(int a, int b) {
14         int res = a - b;
15         res %= inf;
16         res += inf;
17         res %= inf;
18         return res;
19     }
20
21     vi gauss(vector<vi> a, vi b) {
22         int n = a.size();
23         for (int row = 0; row < n; ++row) {
24             bool good = true;
25             for (int col = row; col < n; ++col) {
26                 if (a[row][col] == 0) continue;
27                 swap(a[row], a[col]);
28                 swap(b[row], b[col]);
29                 break;
30             }
31
32
33             int div = binpow(a[row][row], inf - 2);
34             for (int col = 0; col < n; ++col) a[row
                   ][col] = a[row][col] * div % inf;
35             b[row] = b[row] * div % inf;
```

```
36
37              for (int currow = row + 1; currow < n;
                    ++currow) {
38                  if (a[currow][row] == 0) continue;
39                  int mult = a[currow][row];
40                  for (int col = 0; col < n; ++col) a[
                        currow][col] = sub(a[currow][col
                        ], a[row][col] * mult);
41                  b[currow] = sub(b[currow], b[row] *
                        mult);
42              }
43          }
44
45          for (int row = n - 1; row >= 0; --row) {
46              int div = binpow(a[row][row], inf - 2);
47              for (int col = 0; col < n; ++col) a[row
                    ][col] = a[row][col] * div % inf;
48              b[row] = b[row] * div % inf;
49
50              for (int currow = 0; currow < row; ++
                    currow) {
51                  int mult = a[currow][row];
52                  for (int col = 0; col < n; ++col) a[
                        currow][col] = sub(a[currow][col
                        ], a[row][col] * mult);
53                  b[currow] = sub(b[currow], b[row] *
                        mult);
54              }
55          }
56
57          for (int i = 0; i < n; ++i) {
58              b[i] *= binpow(a[i][i], inf - 2);
59              b[i] %= inf;
60              a[i][i] = 1;
61          }
62          return b;
63      }
64  }
65  using namespace Gauss;
```

0.9 li$_c$*hao.cpp*

```
1  struct line
2  {
3      int64_t k, b;
4      line(int64_t k = 0, int64_t b = INF) :k(k), b(b)
           {}
5      int64_t operator[](int64_t a) { return a * k + b
           ; }
6  };
7  struct li_chao
8  {
9      static const int64_t maxn = 100 * 1000 + 4;
10     vector<int64_t> x;
11     vector<line> tr;
12     int64_t n = 1;
13     li_chao() {}
14     li_chao(vector<int64_t>& xx) {
15         //unordered array of all xs
16         x = xx;
17         x.push_back(INF);
18         sort(all(x));
19         x.resize(unique(all(x)) - begin(x));
20         tr.assign(4 * maxn, line());
21     }
22     void set(line nl, int64_t v = 1, int64_t l = 0,
           int64_t r = maxn)
23     {
24         r = min(r, (int64_t)x.size() - 1);
25         int64_t m = (l + r) / 2;
26         bool left = nl[x[l]] < tr[v][x[l]];
27         bool mid = nl[x[m]] < tr[v][x[m]];
28         bool right = nl[x[r]] < tr[v][x[r]];
29         if (mid) swap(nl, tr[v]);
30         if (r - l == 1) return;
31         if (left != mid) set(nl, v + v, l, m);
32         if (right != mid) set(nl, v + v + 1, m, r);
33
34     }
35     int64_t get(int64_t p, int64_t v = 1, int64_t l
```

```
        = 0, int64_t r = maxn)
36      {
37          //gives minimal value
38          r = min(r, (int64_t)x.size() - 1);
39          int64_t m = (l + r) / 2;
40          if (r - l == 1) return tr[v][x[p]];
41          if (p < m) return min(tr[v][x[p]], get(p, v
               + v, l, m));
42          else return min(tr[v][x[p]], get(p, v + v +
               1, m, r));
43      }
44  };
```

0.10 minqueue.cpp

```
1   template<class T>
2   struct minqueue
3   {
4       int b = 0, e = 0;
5       deque<T> d;
6       deque<int> di;
7       T& get() { return d.front(); }
8       void push(T a)
9       {
10          while (d.size() && d.back() > a)
11          {
12              d.pop_back();
13              di.pop_back();
14          }
15          d.push_back(a);
16          di.push_back(b++);
17      }
18      void pop()
19      {
20          if (d.size() && di.front() == e)
21          {
22              d.pop_front();
23              di.pop_front();
24          }
25          ++e;
26      }
```

```
27          int size(){ return b - e; }
28 };
```

0.11 fft.cpp

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  #define pb  push_back
4  #define mp  make_pair
5  #define fs  first
6  #define sc  second
7  #define ll  long long
8  #define vi  vector<int>
9  #define vvi vector<vi >
10 #define all(x) x.begin(), x.end()
11 #define PI  acos(-1.0)
12
13 struct base {
14     double x, y;
15     base (double a = 0, double b = 0) { x = a; y = b
          ; }
16 };
17 base operator+(base& a, base& b) { return base(a.x +
       b.x, a.y + b.y); }
18 base operator-(base& a, base& b) { return base(a.x -
       b.x, a.y - b.y); }
19 base operator*(base& a, base& b) { return base(a.x *
       b.x - a.y * b.y, a.x * b.y + a.y * b.x); }
20 void operator*=(base& a, base& b) { a = a * b; }
21 void operator/=(base& a, double n) { a.x /= n; a.y
     /= n; }
22
23 int rev(int num, int lg_n) {
24     int res = 0;
25     for (int i = 0; i < lg_n; ++i)
26         if (num & (1 << i))
27             res |= 1 << (lg_n - 1 - i);
28     return res;
29 }
30
31 void fft(vector<base>& a, bool invert) {
```

```
32      int n = (int)a.size();
33      int lg_n = 0;
34      while ((1 << lg_n) < n)
35          ++lg_n;
36
37      for (int i = 0; i < n; ++i)
38          if (i < rev(i, lg_n))
39              swap(a[i], a[rev(i, lg_n)]);
40
41      for (int len = 2; len <= n; len <<= 1) {
42          double ang = 2 * PI / len * (invert ? -1 :
                1);
43          base wlen(cos(ang), sin(ang));
44          for (int i = 0; i < n; i += len) {
45              base w(1);
46              for (int j = 0; j < len / 2; ++j) {
47                  base u = a[i + j];
48                  base v = a[i + j + len / 2] * w;
49                  a[i + j] = u + v;
50                  a[i + j + len / 2] = u - v;
51                  w *= wlen;
52              }
53          }
54      }
55      if (invert)
56          for (int i = 0; i < n; ++i)
57              a[i] /= n;
58  }
59
60  int m, n;
61  vector<base> a, b, c;
62
63  void read(vector<base>& v) {
64      int k; cin >> k;
65      v.resize(k);
66      for (int i = 0; i < k; ++i)
67          cin >> v[i].x;
68  }
69
70  int main() {
71      ios_base::sync_with_stdio(false);
```

```
72      read(a);
73      read(b);
74      m = max((int)a.size(), (int)b.size());
75      n = 1;
76      while (n < m + m)
77          n = n + n;
78      while (a.size() < n) a.pb(base());
79      while (b.size() < n) b.pb(base());
80      c.resize(n);
81
82      fft(a, false);
83      fft(b, false);
84      for (int i = 0; i < n; ++i)
85          c[i] = a[i] * b[i];
86      fft(c, true);
87
88      return 0;
89  }
```

0.12 bridges.cpp

```
1   namespace FindAllBridges
2   {
3       vector<int> bridges;
4       vector<int> fup, tin;
5       int tin_global = 0;
6       void dfs(int cur, int parent, int edge_i, vector
            <vector<pair<int, int>>>& g)
7       {
8           tin[cur] = ++tin_global;
9           fup[cur] = tin_global;
10          for (int i = 0; i < g[cur].size(); i++)
11          {
12              int to, index;
13              tie(to, index) = g[cur][i];
14              if (tin[to] == 0)
15              {
16                  dfs(to, cur, index, g);
17                  fup[cur] = min(fup[cur], fup[to]);
18              }
19              else
```

```
20              {
21                  if (to != parent)
22                      fup[cur] = min(fup[cur], tin[to
                           ]);
23              }
24          }
25          if (fup[cur] == tin[cur] && edge_i != -1)
               bridges.push_back(edge_i);
26      }
27
28      vector<int> find_bridges(vector<vector<pair<int,
            int>>>& g)
29      {   //to, index_of_edge
30          tin_global = 0;
31          int n = g.size();
32          fup.assign(n, -1);
33          tin.assign(n, 0);
34          bridges.clear();
35          dfs(0, -1, -1, g);
36          return bridges;
37      }
38
39  }
```

0.13 hashes.cpp

```
1  const long long maxn = 1000004;
2  const pair<long long, long long> p = make_pair(263,
       263);
3  pair<long long, long long> invp; //k^-1 % mod == k^(
       mod - 2) % mod
4  const pair<long long, long long> mod = make_pair
       (1000000007, 1000000009);
5  long long binpow(long long a, long long p, long long
        m){
6      long long res = 1;
7      for (; p>0; p/=2){
8          if (p % 2 == 1) res = res * a % m;
9          a = a * a % m;
10     }
11     return res;
```

```
12 }
13 pair<long long, long long> pw[maxn]; //p^0, p^1, p
      ^2, ....
14 pair<long long, long long> invpw[maxn];//p^0, p^-1,
      p^-2 ...
15
16 struct myhash{
17     vector<pair<long long, long long>> h;
18     myhash(){}
19     myhash(string s){
20         for (int i = 0; i < (int)s.size(); ++i){
21             h.push_back(pair<long long, long long>()
                  );
22             h.back().first = s[i] * pw[i].first %
                  mod.first;
23             h.back().second = s[i] * pw[i].second %
                  mod.second;
24             if (i != 0) {
25                 h[i].first = (h[i].first + h[i-1].
                      first) % mod.first;
26                 h[i].second = (h[i].second + h[i-1].
                      second) % mod.second;
27             }
28         }
29     }
30     //including borders
31     pair<long long, long long> get_hash(int l, int r
          ){
32         pair<long long, long long> res = h[r];
33         if (l != 0) res.first += mod.first - h[l-1].
              first;
34         if (l != 0) res.second += mod.second - h[l
              -1].second;
35         res.first = res.first * invpw[l].first % mod
              .first;
36         res.second = res.second * invpw[l].second %
              mod.second;
37         return res;
38     }
39 };
```

0.14 debug-tourist.cpp

```
1  template <typename A, typename B>
2  string to_string(pair<A, B> p);
3
4  template <typename A, typename B, typename C>
5  string to_string(tuple<A, B, C> p);
6
7  template <typename A, typename B, typename C,
       typename D>
8  string to_string(tuple<A, B, C, D> p);
9
10 string to_string(const string& s) {
11   return '"' + s + '"';
12 }
13
14 string to_string(const char* s) {
15   return to_string((string) s);
16 }
17
18 string to_string(bool b) {
19   return (b ? "true" : "false");
20 }
21
22 string to_string(vector<bool> v) {
23   bool first = true;
24   string res = "{";
25   for (int i = 0; i < static_cast<int>(v.size()); i
         ++) {
26     if (!first) {
27       res += ", ";
28     }
29     first = false;
30     res += to_string(v[i]);
31   }
32   res += "}";
33   return res;
34 }
35
36 template <size_t N>
```

```
37 string to_string(bitset<N> v) {
38   string res = "";
39   for (size_t i = 0; i < N; i++) {
40     res += static_cast<char>('0' + v[i]);
41   }
42   return res;
43 }
44
45 template <typename A>
46 string to_string(A v) {
47   bool first = true;
48   string res = "{";
49   for (const auto &x : v) {
50     if (!first) {
51       res += ",␣";
52     }
53     first = false;
54     res += to_string(x);
55   }
56   res += "}";
57   return res;
58 }
59
60 template <typename A, typename B>
61 string to_string(pair<A, B> p) {
62   return "(" + to_string(p.first) + ",␣" + to_string
       (p.second) + ")";
63 }
64
65 template <typename A, typename B, typename C>
66 string to_string(tuple<A, B, C> p) {
67   return "(" + to_string(get<0>(p)) + ",␣" +
       to_string(get<1>(p)) + ",␣" + to_string(get<2>(
       p)) + ")";
68 }
69
70 template <typename A, typename B, typename C,
     typename D>
71 string to_string(tuple<A, B, C, D> p) {
72   return "(" + to_string(get<0>(p)) + ",␣" +
       to_string(get<1>(p)) + ",␣" + to_string(get<2>(
```

```
      p)) + ",␣" + to_string(get<3>(p)) + ")";
}

void debug_out() { cerr << endl; }

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

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

0.15 fftmodule.cpp

```

const int MOD = 998244353;
ll qp(ll a, ll b){
    while (b<0) b += MOD - 1;
    ll x = 1; a %= MOD;
    while (b){
        if (b & 1) x = x*a%MOD;
        a = a*a%MOD; b >>= 1;
    }
    return x;
}
int *w0[2333], *w1[2333];
int getK(int n)
{
    int s = 1; while (s<n) s <<= 1; return s;
}
void prep(int K)
{
    static int pool[1048576 * 4 + 3], *p = pool, i =
        1, j = 1;
    for (; j <= K; ++i, j <<= 1){
```

```
21          w0[i] = p; w1[i] = (p += j); p += j;
22          ll g = qp(3, (MOD - 1) >> i), ig = qp(g, MOD
                - 2);
23          w0[i][0] = w1[i][0] = 1;
24          for (int k = 1; k<j; ++k)
25              w0[i][k] = w0[i][k - 1] * g%MOD,
26              w1[i][k] = w1[i][k - 1] * ig%MOD;
27      }
28  }
29  void fft(int* x, int K, int v)
30  {
31      prep(K);
32      for (int i = 0, j = 0; i<K; i++){
33          if (i>j) swap(x[i], x[j]);
34          for (int l = K >> 1; (j ^= l)<l; l >>= 1);
35      }
36      for (int i = 0; i<K; i++) x[i] = (x[i] % MOD +
            MOD) % MOD;
37      for (int i = 2, c = 1; i <= K; i <<= 1, ++c)
38          for (int *w = v ? w1[c] : w0[c], j = 0; j<K;
                j += i)
39              for (int h = i >> 1, a, b, l = 0; l<h;
                    ++l){
40                  a = x[j + l]; if (a >= MOD) a -= MOD
                        ;
41                  b = (ll)x[j + h + l] * w[l] % MOD,
42                      x[j + h + l] = a - b + MOD, x[j
                            + l] = a + b;
43              }
44      for (int i = 0; i<K; i++) x[i] = (x[i] % MOD +
            MOD) % MOD;
45      if (!v) return;
46      ll rv = qp(K, MOD - 2);
47      for (int i = 0; i<K; i++) x[i] = x[i] * rv%MOD;
48  }
49  vector<int> operator * (const vector<int>& a, const
        vector<int>& b)
50  {
51      static int p[4*1048576], q[4*1048576];
52      int w = a.size() + b.size() - 1;
53      vector<int> c; c.resize(w);
```

```
54      if (b.size()<13){
55          for (int i = 0; i<(int)a.size(); ++i)
56              for (int j = 0; j<(int)b.size(); ++j)
57                  c[i + j] = (c[i + j] + (ll)a[i] * b[
                        j]) % MOD;
58          return c;
59      }
60      int K = getK(w);
61      for (int i = 0; i<K; ++i) p[i] = q[i] = 0;
62      for (int i = 0; i<(int)a.size(); ++i) p[i] = a[i
           ];
63      for (int i = 0; i<(int)b.size(); ++i) q[i] = b[i
           ];
64      fft(p, K, 0); fft(q, K, 0);
65      for (int i = 0; i<K; ++i)
66          p[i] = p[i] * (ll)q[i] % MOD;
67      fft(p, K, 1);
68      for (int i = 0; i<w; ++i) c[i] = p[i];
69      return c;
70  }
```

0.16 generator.cpp

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  namespace Generator {
5      const int maxn = (1 << 17);
6      int used[maxn]; //idea similar to segment tree
7      int usedused[maxn];
8
9      mt19937_64 rng(time(0));
10     //mt19937_64 rng(1);
11     int getnext() {
12         int res = -1;
13         while (res == -1 || res < 2) {
14             res = rng() % maxn;
15         }
16         return res;
17     }
18
```

```
19      string getbinary(int a) {
20          string res = "";
21          while (a > 1) {
22              char next = ('0' + (a & 1));
23              res.push_back(next);
24              a /= 2;
25          }
26          reverse(begin(res), end(res));
27          return res;
28      }
29
30
31      void update_used_down(int a) {
32          if (a >= maxn) return;
33          if (used[a] == 1) return;
34          used[a] = 1;
35          update_used_down(a + a);
36          update_used_down(a + a + 1);
37      }
38      void update_used_up(int a) {
39          while (a > 1) {
40              used[a] = 1;
41              a /= 2;
42          }
43      }
44      void update_used(int a) {
45          update_used_down(a);
46          update_used_up(a);
47      }
48
49      vector<string> solve(int n, int pref) {
50
51          vector<int> output;
52          fill(used, used + maxn, 0);
53          fill(usedused, usedused + maxn, 0);
54
55          n -= pref;
56
57          //generate with unique prefixes
58          for (int i = 0; i < n; ++i) {
59              int cur = -1;
```

```
60              while (cur == -1 || used[cur] == 1) {
61                  cur = getnext();
62              }
63              output.push_back(cur);
64              update_used(cur);
65              usedused[cur] = 1;
66          }
67
68          vector<int> used_idx;
69          for (int i = 0; i < maxn; ++i)
70              if (used[i]) used_idx.push_back(i);
71
72          //generate prefixes
73          for (int i = 0; i < pref; ++i) {
74              int cur = -1;
75              while (cur == -1 || usedused[cur] == 1)
                    {
76                  int idx = int(rng() % used_idx.size
                        ());
77                  cur = used_idx[idx];
78              }
79              output.push_back(cur);
80              usedused[cur] = 1;
81          }
82
83          //shuffle result
84          shuffle(begin(output), end(output), rng);
85
86          vector<string> resres;
87          for (auto& it : output) {
88              resres.push_back(getbinary(it));
89              //cout << getbinary(it) << endl;
90          }
91          return resres;
92      }
93  }
94
95
96  int checker(vector<string>& vs) {
97      int res = 0;
98      set<string> s;
```

```
99      for (auto& it : s) {
100         s.insert(it);
101     }
102     return res;
103 }
104
105 int main() {
106     ios_base::sync_with_stdio(false);
107     cout << fixed << setprecision(10);
108     cin.tie(nullptr);
109 #ifdef _MY
110     freopen("input.txt", "r", stdin);
111     freopen("output.txt", "w", stdout);
112 #endif
113
114     int n = 1000;
115     int cnt = 0;
116     for (int i = 0; i < 10; ++i){
117         string test = "test";
118         test = test + char(i / 10 + '0') + char(i %
               10 + '0') + ".in";
119         ofstream out(test.c_str());
120
121         int pref = 0;
122         //while (pref == -1 || pref > 900) pref =
               Generator::rng() % 1000;
123         auto input = Generator::solve(n, pref);
124         cout << checker(input) << endl;
125         out << input.size() << endl;
126         for (auto& it : input) out << it << endl;
127     }
128
129
130 }
```

0.17 mod$_c$omb.cpp

```
1 const int mod = 998244353;
2 int add(int a, int b) { return (a + b) % mod; }
3 int sub(int a, int b) { return (mod + a - b) % mod;
    }
```

```
4  int mult(int a, int b) { return ll(a) * b % mod; }
5  int binpow(int a, int p) {
6      int res = 1;
7      for (; p; p /= 2) {
8          if (p & 1) res = mult(res, a);
9          a = mult(a, a);
10     }
11     return res;
12 }
13 int divi(int a, int b) { return mult(a, binpow(b,
       mod - 2)); }
14
15 int fact[maxn];
16 int invf[maxn];
17 void init() {
18     fact[0] = 1;
19     for (int i = 1; i < maxn; ++i) fact[i] = mult(
           fact[i - 1], i);
20     for (int i = 0; i < maxn; ++i) invf[i] = divi(1,
           fact[i]);
21 }
22 int C(int n, int k) {
23     int res = fact[n];
24     res = mult(res, invf[k]);
25     res = mult(res, invf[n - k]);
26     return res;
27 }
28 int Crep(int n, int k) {
29     return C(n + k - 1, n);
30 }
```

0.18 extgcd.cpp

```
1  ll extgcd(ll a, ll b, ll & x, ll & y) {
2      if (a == 0) {
3          x = 0; y = 1;
4          return b;
5      }
6      ll x1, y1;
7      ll d = extgcd(b%a, a, x1, y1);
8      x = y1 - (b / a) * x1;
```

```
9       y = x1;
10      return d;
11  }
```

0.19 geom$_{g}$*ood.cpp*

```
1
2   /*
3   namespace GEOMA_INT{
4       typedef long long ll;
5       bool eq(ll a, ll b){return a == b;}
6       struct pt{
7           ll x, y;
8           pt(){}
9           pt(ll x, ll y):x(x), y(y){}
10          bool operator<(pt a){return eq(x, a.x)? y <
                a.y : x < a.x;}
11          pt operator-(){return pt(-x, -y);}
12          pt operator+(pt b){return pt(x + b.x, y + b.
                y);}
13          pt operator-(pt b){return pt(x - b.x, y - b.
                y);}
14          pt operator*(ll m){return pt(x*m, y*m);}
15          pt& operator+=(pt b){return *this = *this +
                b;}
16          pt& operator-=(pt b){return *this = *this -
                b;}
17          pt& operator*=(ll m){return *this = *this *
                m;}
18      };
19      pt operator*(ll a, pt b){return pt(a*b.x, a*b.y)
            ;}
20      ostream& operator<<(ostream& str, pt p){
21          return str << "[" << p.x << ", " << p.y <<
                "]";
22      }
23      bool eq(pt a, pt b){return eq(a.x, b.x) && eq(a.
            y, b.y);}
24      ll dot  (pt a, pt b){return a.x*b.x + a.y*b.y;}
25      ll cross(pt a, pt b){
26          ll res = 0;
```

```
27          res += a.y * b.x;
28          res -= a.x * b.y;
29          return res;
30      }
31      ll dist2(pt a, pt b = pt(0, 0)){
32          ll res = 0;
33          res += (a.x - b.x) * (a.x - b.x);
34          res += (a.y - b.y) * (a.y - b.y);
35          return res;
36      }
37
38      //given vector of points of polygon
39      //returns 2 times area of that polygon
40      ll polygon_area_x2(vector<pt> v){
41          ll res = 0;
42          for (int i = 0; i < (int)v.size(); ++i){
43              res += cross(v[i], v[(i+1)%(int)v.size()
                    ]);
44          }
45          return abs(res);
46      }
47      struct circle{
48          pt c;
49          ll r;
50          circle(){}
51          circle(pt c, ll r):c(c), r(r){}
52      };
53      bool is_inside(pt p, circle c){ return dist2(p,
            c.c) < c.r*c.r; }
54      bool on_surface(pt p, circle c){ return dist2(p,
            c.c) == c.r*c.r; }
55  };
56  using namespace GEOMA_INT;
57  */
58
59  namespace GEOMA_REAL{
60      typedef long double ld;
61      const ld eps = 1e-6;
62      const ld pi = acos(-1.0);
63      bool eq(ld a, ld b){return abs(a-b) < eps;}
64
```

```
65      struct pt{
66          ld x, y;
67          pt(){}
68          pt(ld x, ld y):x(x),y(y){}
69          bool operator<(pt a){return eq(x, a.x)? y <
               a.y : x < a.x;}
70          pt operator-(){return pt(-x, -y);}
71          pt operator+(pt b){return pt(x + b.x, y + b.
               y);}
72          pt operator-(pt b){return pt(x - b.x, y - b.
               y);}
73          pt operator*(ld m){return pt(x*m, y*m);}
74          pt operator/(ld m){assert(!eq(m, 0.0));
               return pt(x/m, y/m);}
75          pt& operator+=(pt b){return *this = *this +
               b;}
76          pt& operator-=(pt b){return *this = *this -
               b;}
77          pt& operator*=(ld m){return *this = *this *
               m;}
78          pt& operator/=(ld m){return *this = *this /
               m;}
79      };
80      pt operator*(ld a, pt b){return pt(a*b.x, a*b.y)
           ;}
81      ostream& operator<<(ostream& str, pt p){
82          return str << "[" << p.x << ",␣" << p.y << "
               ]";
83      }
84
85      bool eq(pt a, pt b){return eq(a.x, b.x) && eq(a.
           y, b.y);}
86      ld dot  (pt a, pt b){return a.x*b.x + a.y*b.y;}
87      ld cross(pt a, pt b){
88          ld res = 0;
89          res += a.y * b.x;
90          res -= a.x * b.y;
91          return res;
92      }
93      ld dist2(pt a, pt b = pt(0, 0)){
94          ld res = 0;
```

```
95          res += (a.x - b.x) * (a.x - b.x);
96          res += (a.y - b.y) * (a.y - b.y);
97          return res;
98      }
99      ld dist(pt a, pt b = pt(0, 0)){return sqrt(dist2
           (a, b));}
100
101     //given vector of points of polygon
102     //returns area of that polygon
103     ld polygon_area(vector<pt> v){
104         ld res = 0;
105         for (int i = 0; i < (int)v.size(); ++i){
106             res += cross(v[i], v[(i+1)%(int)v.size()
                   ]);
107         }
108         return abs(res) / 2;
109     }
110
111     ld angle(pt p){return atan2(p.y, p.x);}
112     pt rotate(pt p, ld a){
113         pt res;
114         res.x = cos(a)*p.x - sin(a)*p.y;
115         res.y = sin(a)*p.x + cos(a)*p.y;
116         return res;
117     }
118
119     struct circle{
120         pt c; ld r;
121         circle(){}
122         circle(pt c, ld r):c(c),r(r){}
123     };
124
125     //returns whether point is strictly inside the
           circle
126     bool is_inside(pt p, circle c){
127         ld d = dist2(p, c.c);
128         if (eq(d, c.r*c.r)) return false;
129         if (d > c.r*c.r) return false;
130         return true;
131     }
132
```

```
133
134     //returns whether point is strictly on surface
           of circle
135     bool on_surface(pt p, circle c){return eq(dist2(
           p, c.c), c.r*c.r);}
136
137
138     //returns points on surface of circle that
139     //are in lines tangent to that circle
140     //in counterclockwise order
141     pair<pt, pt> tangent_points(pt p, circle c){
142         ld hsqr = dist2(p, c.c);
143         assert(!eq(hsqr, c.r*c.r));
144         assert(hsqr > c.r*c.r);
145         ld bsqr = hsqr - c.r*c.r;
146         ld ang = acos(sqrt(bsqr / hsqr));
147         pt cur(sqrt(bsqr), 0);
148         pair<pt, pt> res;
149         res.first  = rotate(cur, angle(c.c - p) -
               ang) + p;
150         res.second = rotate(cur, angle(c.c - p) +
               ang) + p;
151         return res;
152     }
153 };
154 using namespace GEOMA_REAL;
```

0.20 double persistent stree.cpp

```
1  //#pragma comment(linker, "/stack:200000000")
2  #pragma GCC optimize("O3")
3  #include <bits/stdc++.h>
4  //#define int ll
5  #define fs first
6  #define sd second
7  #define mp make_pair
8  #define pb push_back
9  #define sz(x) int((x).size())
10 #define all(x) begin(x), end(x)
11 #define OUT(x) { cout << x; exit(0); }
12 //#define resize do_not_use_resize
```

```
13 using namespace std;
14 typedef double db;
15 typedef long long ll;
16 typedef vector<int> vi;
17 typedef pair<int, int> pii;
18 const db eps = 1e-9;
19 const db pi = acos(-1.0);
20 const db dinf = 1e250;
21 const ll INF = (ll)(2e18);
22 const int inf = (int)(1e9 + 7);
23 //------------------------------------------//
24
25 struct pt
26 {
27         int first, second, th;
28         pt() {}
29         pt(int first, int second, int th) : fs(fs),
             sd(sd), th(th) {}
30         bool operator<(pt p)
31         {
32                 if (fs != p.fs) return fs < p.fs;
33                 if (sd != p.sd) return sd < p.sd;
34                 return th < p.th;
35         }
36 };
37
38 typedef vector<pt>::iterator iterp;
39 typedef vector<pii>::iterator iter;
40
41 const int maxn = 50 * 1000 + 5;
42
43 vi x, y;
44
45 struct node
46 {
47         unsigned short val = 0;
48         node* l = 0, *r = 0;
49 };
50 struct stree
51 {
52         vector<node*> t;
```

```
53          stree() { t.pb(new node()); }
54          node*& safe_ptr(node*& ptr) { return ptr ?
               ptr : ptr = new node(); }
55          int safe_get(node* ptr) { return ptr ? ptr->
               val : 0; }
56          node* set(iter b, iter e, node* v, int tl =
               0, int tr = maxn)
57          {
58                  node* curv = new node(*safe_ptr(v));
59                  if (tl + 1 == tr)
60                  {
61                          for (auto it = b; it != e;
                               ++it) curv->val += it->sd
                               ;
62                          return curv;
63                  }
64                  int mid = (tl + tr) / 2;
65                  iter midi = lower_bound(b, e, mp(mid
                       , -inf));
66                  if (b < midi) curv->l = set(b, midi,
                        safe_ptr(v->l), tl, mid);
67                  if (e > midi) curv->r = set(midi, e,
                        safe_ptr(v->r), mid, tr);
68                  curv->val = safe_get(curv->l) +
                       safe_get(curv->r);
69                  if (curv->val == 0)
70                  {
71                          delete curv;
72                          curv = nullptr;
73                  }
74                  return curv;
75          }
76          int get(int l, int r, node* v, int tl = 0,
               int tr = maxn)
77          {
78                  if (tl == l && tr == r) return v->
                       val;
79                  int mid = (tl + tr) / 2;
80                  int res = 0;
81                  if (l < mid) res += get(l, min(r,
                       mid), safe_ptr(v->l), tl, mid);
```

```
82                  if (r > mid) res += get(max(l, mid),
                        r, safe_ptr(v->r), mid, tr);
83                  return res;
84          }
85  };
86
87  struct node2
88  {
89          stree* val;
90          node* ptr = nullptr;
91          node2* l = 0, *r = 0;
92          node2()
93          {
94                  val = new stree();
95                  ptr = val->t.back();
96          }
97          node*& safe_ptr(node*& ptr) { return ptr ?
               ptr : ptr = new node(); }
98          int get(int y1, int y2) { return val->get(y1
               , y2, safe_ptr(ptr)); }
99          void set(iter b, iter e) { ptr = val->set(b,
                e, ptr); }
100 };
101 struct stree2
102 {
103         vector<node2*> t;
104         stree2() { t.pb(new node2()); }
105         node2*& safe_ptr(node2*& ptr) { return ptr ?
                ptr : ptr = new node2(); }
106         int safe_get(node2* ptr, int y1, int y2) {
               return ptr ? ptr->get(y1, y2) : 0; }
107         node2* set(iterp b, iterp e, node2* v, int
               tl = 0, int tr = maxn)
108         {
109                 node2* curv = new node2(*v);
110
111                 vector<pii>* kek = new vector<pii>()
                       ;
112                 for (auto it = b; it != e; ++it) kek
                       ->pb({ it->sd, it->th });
113                 sort(kek->begin(), kek->end());
```

```
114
115                 if (b != e) curv->set(kek->begin(),
                        kek->end());
116                 delete kek;
117
118                 if (tl + 1 == tr) return curv;
119                 int mid = (tl + tr) / 2;
120                 iterp midi = lower_bound(b, e, pt(
                        mid, -inf, -inf));
121                 if (b != midi) curv->l = set(b, midi
                        , safe_ptr(v->l), tl, mid);
122                 if (e != midi) curv->r = set(midi, e
                        , safe_ptr(v->r), mid, tr);
123                 return curv;
124         }
125         int get(int l, int r, int y1, int y2, node2*
                v, int tl = 0, int tr = maxn)
126         {
127                 if (tl == l && tr == r) return v->
                        get(y1, y2);
128                 int mid = (tl + tr) / 2;
129                 int res = 0;
130                 if (l < mid) res += get(l, min(r,
                        mid), y1, y2, safe_ptr(v->l), tl,
                         mid);
131                 if (r > mid) res += get(max(l, mid),
                         r, y1, y2, safe_ptr(v->r), mid,
                        tr);
132                 return res;
133         }
134 };
135
136
137
138
139
140 int32_t main()
141 {
142         ios_base::sync_with_stdio(0);
143         cout << fixed << setprecision(10);
144         cin.tie(0);
```

```
145
146 #ifdef _MY
147         freopen("input.txt", "r", stdin);
148         freopen("output.txt", "w", stdout);
149 #endif
150
151         x.pb(-inf); x.pb(inf);
152         y.pb(-inf); y.pb(inf);
153
154         int n;
155         cin >> n;
156         vector<pii> pts(n);
157
158         for (auto& it : pts)
159         {
160                 cin >> it.fs >> it.sd;
161                 x.pb(it.fs);
162                 y.pb(it.sd);
163         }
164
165         sort(all(x));
166         sort(all(y));
167         sort(all(pts));
168         x.resize(unique(all(x)) - begin(x));
169         y.resize(unique(all(y)) - begin(y));
170         pts.resize(unique(all(pts)) - begin(pts));
171
172         for (auto& it : pts)
173         {
174                 it.fs = int(lower_bound(all(x), it.
                    fs) - begin(x));
175                 it.sd = int(lower_bound(all(y), it.
                    sd) - begin(y));
176         }
177
178         vector<vi> forx(sz(x));
179         vector<vi> fory(sz(y));
180         for (auto& it : pts) forx[it.fs].pb(it.sd);
181         for (auto& it : pts) fory[it.sd].pb(it.fs);
182         for (auto& it : forx)
183         {
```

```
184                 if (sz(it) == 0) continue;
185                 it.pb(sz(y) - 1);
186                 it.pb(0);
187                 sort(all(it));
188         }
189         for (auto& it : fory)
190         {
191                 if (sz(it) == 0) continue;
192                 it.pb(sz(x) - 1);
193                 it.pb(0);
194                 sort(all(it));
195         }
196
197
198         vector<vector<pii>> eventsy(sz(y));
199         vector<vector<pii>> eventsx(sz(x));
200         for (int curx = 0; curx < sz(x); ++curx)
201         {
202                 for (int i = sz(forx[curx]) - 2; i
                       >= 0; --i)
203                         eventsy[forx[curx][i]].pb({
                           curx, forx[curx][i + 1]
                           });
204         }
205         for (int cury = 0; cury < sz(y); ++cury)
206         {
207                 for (int i = sz(fory[cury]) - 2; i
                       >= 0; --i)
208                         eventsx[fory[cury][i]].pb({
                           cury, fory[cury][i + 1]
                           });
209         }
210         for (auto& it : eventsy) sort(all(it));
211
212         stree2 diffx;
213         stree2 diffy;
214         vi curxs(sz(x), 0);
215         vi curys(sz(y), 0);
216         vector<node2*> diffxstates;
217         vector<node2*> diffystates;
218         for (int cury = 0; cury < sz(y); ++cury)
```

```
219         {
220                 map<pii, int> m;
221                 for (auto& it : eventsy[cury])
222                 {
223                         if (curxs[it.fs] != 0) m[{it
                                .fs, curxs[it.fs] }]--;
224                         m[it]++;
225                 }
226                 vector<pt> kek;
227                 for (auto& it : m) if (it.sd) kek.pb
                        ({ it.fs.fs, it.fs.sd, it.sd });
228
229                 diffx.t.pb(diffx.set(all(kek), diffx
                        .t.back()));
230                 diffxstates.pb(diffx.t.back());
231         }
232         for (int curx = 0; curx < sz(x); ++curx)
233         {
234                 map<pii, int> m;
235                 for (auto& it : eventsx[curx])
236                 {
237                         if (curys[it.fs] != 0) m[{it
                                .fs, curys[it.fs] }]--;
238                         m[it]++;
239                 }
240                 vector<pt> kek;
241                 for (auto& it : m) if (it.sd) kek.pb
                        ({ it.fs.fs, it.fs.sd, it.sd });
242
243                 diffy.t.pb(diffy.set(all(kek), diffy
                        .t.back()));
244                 diffystates.pb(diffy.t.back());
245         }
246
247
248         int q;
249         cin >> q;
250         for (int i = 0; i < q; ++i)
251         {
252                 int x1, y1, x2, y2;
253                 cin >> x1 >> y1 >> x2 >> y2;
```

```
254                 x1 = int(lower_bound(all(x), x1) -
                       begin(x));
255                 y1 = int(lower_bound(all(y), y1) -
                       begin(y));
256                 x2 = int(upper_bound(all(x), x2) -
                       begin(x));
257                 y2 = int(upper_bound(all(y), y2) -
                       begin(y));
258                 cout << diffx.get(x1, x2, y1, y2,
                       diffxstates[y1 - 1]) << "␣";
259                 cout << diffy.get(y1, y2, x1, x2,
                       diffystates[x1 - 1]) << endl;
260         }
261
262
263 #ifdef _MY
264         int loop = 0;
265         while (true)
266         {
267                 for (int i = 0; i < 1000000; ++i)++
                       loop;
268                 cerr << "huh" << endl;
269         }
270         cout << loop;
271 #endif
272
273         return 0;
274 }
```

0.21 mincost$_max flow$.cpp

```
1  struct MCMF{
2      int s, t;
3      struct edge
4      {
5          int to, f = 0, cap, cost;
6          edge() {}
7          edge(int to, int cap, int cost) :to(to), cap
               (cap), cost(cost) {}
8      };
9
```

```
10      vector<vi> g;
11      vector<edge> E;
12      vi d, inq, p, pe;
13      void augument()
14      {
15          queue<int> q;
16          d.assign(t + 1, inf);
17          inq.assign(t + 1, 0);
18          p.assign(t + 1, -1);
19          pe.assign(t + 1, -1);
20          d[s] = 0;
21          q.push(s);
22          while (q.size())
23          {
24              int k = q.front();
25              q.pop();
26              inq[k] = 0;
27              for (auto& e : g[k])
28              {
29                  if (E[e].cap - E[e].f > 0)
30                  {
31                      int w = E[e].cost;
32                      int to = E[e].to;
33                      if (d[to] > d[k] + w)
34                      {
35                          p[to] = k;
36                          pe[to] = e;
37                          d[to] = d[k] + w;
38                          if (!inq[to])
39                          {
40                              q.push(to);
41                              inq[to] = 1;
42                          }
43                      }
44                  }
45              }
46          }
47
48
49      }
50
```

```
51      int check()
52      {
53          if (p[t] == -1) return -1;
54          int mx = inf;
55          int cur = t;
56          while (cur != s)
57          {
58              mx = min(mx, E[pe[cur]].cap - E[pe[cur
                    ]].f);
59              cur = p[cur];
60          }
61          cur = t;
62          while (cur != s)
63          {
64              E[pe[cur]].f += mx;
65              E[pe[cur] ^ 1].f -= mx;
66              cur = p[cur];
67          }
68          return mx * d[t];
69      }
70
71      int mincost_maxflow()
72      {
73          int res = 0;
74          while (true)
75          {
76              augument();
77              int cur = check();
78              if (cur == -1) break;
79              res += cur;
80          }
81          return res;
82      }
83
84      void add_edge(int f, int to, int cap, int cost)
85      {
86          g[f].push_back(int(E.size()));
87          E.push_back(edge(to, cap, cost));
88          g[to].push_back(int(E.size()));
89          E.push_back(edge(f, 0, -cost));
90      }
```

```
91
92      MCMF(){}
93      MCMF(int s, int t, int graphsize):s(s), t(t),
           graphsize(graphsize){
94          g.resize(graphsize);
95      }
96  };
```

0.22 core.cpp

```
1   #include "bits/stdc++.h"
2   //#define int ll
3   #define F first
4   #define S second
5   #define mp make_pair
6   #define all(x) (x).begin(), (x).end()
7   #define out(x) return void(cout << (x) << endl)
8   #define OUT(x) ((cout << (x)), exit(0))
9   using namespace std;
10  typedef long long ll;
11  typedef long double ld;
12  [[maybe_unused]]const int64_t INF = (int64_t)(2e18);
13  [[maybe_unused]]const int inf = (int)(1e9 + 7);
14  [[maybe_unused]]const int maxn = 500 * 1000 + 100;
15  //------------------------------------------//
16
17
18
19
20  int32_t main(){
21      ios_base::sync_with_stdio(false);
22      cout << fixed << setprecision(10);
23      cin.tie(nullptr);
24  #ifdef _MY
25      freopen("VimProject/input.txt", "r", stdin);
26      freopen("VimProject/output.txt", "w", stdout);
27  #endif
28
29
30      return 0;
31  }
```

0.23 stree pushes sum max.cpp

```
1  struct stree {
2      vector<int> t;
3      vector<int> p;
4      int n = 1;
5      stree() {}
6      stree(int nn) {
7          while (n < nn) n *= 2;
8          t.assign(2 * n, 0);
9          p.assign(2 * n, 0);
10     }
11     int val(int v) { return t[v] + p[v]; }
12     void push(int v) {
13         t[v] += p[v];
14         if (v < n) {
15             p[2 * v] += p[v];
16             p[2 * v + 1] += p[v];
17         }
18         p[v] = 0;
19     }
20     void upd(int v) { t[v] = max(val(2 * v), val(2 *
           v + 1)); }
21     int get(int l, int r, int v, int tl, int tr) {
22         if (tr == -1) tr = n;
23         if (tl == l && tr == r) return val(v);
24
25         push(v);
26         int res = 0;
27         int mid = (tl + tr) / 2;
28         if (l < mid) res = max(res, get(l, min(mid,
              r), 2 * v, tl, mid));
29         if (r > mid) res = max(res, get(max(l, mid),
               r, 2 * v + 1, mid, tr));
30         upd(v);
31
32         return res;
33     }
34     void set(int l, int r, int x, int v, int tl, int
           tr) {
```

```
35          if (tr == -1) tr = n;
36          if (tl == l && tr == r) return void(p[v] +=
               x);
37
38          push(v);
39          int mid = (tl + tr) / 2;
40          if (l < mid) set(l, min(mid, r), x, 2 * v,
               tl, mid);
41          if (r > mid) set(max(l, mid), r, x, 2 * v +
               1, mid, tr);
42          upd(v);
43      }
44      void set(int l, int r, int x) {
45          r = min(r, n);
46          l = max(l, 0ll);
47          set(l, r, x, 1, 0, n);
48      }
49      int get(int l, int r) {
50          r = min(r, n);
51          l = max(l, 0ll);
52          return get(l, r, 1, 0, n);
53      }
54  };
```

0.24 graph-of-blocks-and-cutpoints.cpp

```
1  #include <bits/stdc++.h>
2  //#define int int64_t
3  #define all(x) (x).begin(), (x).end()
4  #define out(x) return void(cout << x << endl)
5  #define OUT(x) ((cout << x), exit(0))
6  using namespace std;
7  typedef long double db;
8  const int64_t INF = (int64_t)(2e18);
9  const int inf = (int)(1e9 + 7);
10 //----------------------------------------//
11
12 const int maxn = 3000 * 100 + 10;
13 int curtime = 0;
14
15 int color_edge[maxn];
```

```
16 int tin[maxn];
17 int fup[maxn];
18 int added[maxn];
19 vector<pair<int,int>> g[maxn];
20 vector<int> g1[2 * maxn];
21
22 vector<int> cuts;
23 bool is_cut(int v) {
24     auto it = lower_bound(all(cuts), v);
25     if (it == cuts.end()) return false;
26     if (*it != v) return false;
27     return true;
28 }
29 void dfs_cuts(int v, int root) {
30     fup[v] = tin[v] = ++curtime;
31     int sons = 0;
32     bool is_cut = false;
33     for (auto e : g[v]) {
34         if (tin[e.first]) fup[v] = min(fup[v], tin[e
               .first]);
35         else {
36             ++sons;
37             dfs_cuts(e.first, root);
38             fup[v] = min(fup[v], fup[e.first]);
39             if (fup[e.first] >= tin[v]) is_cut = 1;
40         }
41     }
42     if (v == root && sons > 1) cuts.push_back(v);
43     if (v != root && is_cut) cuts.push_back(v);
44 }
45
46 vector<pair<int, int>> edges;
47 int used[maxn];
48 set<int> comps[maxn];
49 int newcol = -1;
50 void dfs_color(int v, int color, int par) {
51     used[v] = 1;
52     for (auto& e : g[v]) {
53         if (e.first == par) continue;
54
55         if (used[e.first] && tin[e.first] < tin[v])
```

```
56              color_edge[e.second] = color;
57          if (used[e.first]) continue;
58
59          if (fup[e.first] >= tin[v]) {
60              color_edge[e.second] = ++newcol;
61              dfs_color(e.first, newcol, v);
62          }
63          else {
64              color_edge[e.second] = color;
65              dfs_color(e.first, color, v);
66          }
67      }
68  }
69
70  set<int> res;
71  bool dfs_find(int v, int goal) {
72      used[v] = 1;
73      if (v == goal) return true;
74      bool good = false;
75      for (auto& to : g1[v]) {
76          if (used[to]) continue;
77          good |= dfs_find(to, goal);
78      }
79      if (good && v < cuts.size())
80          res.insert(cuts[v]);
81      return good;
82  }
83
84  int32_t main() {
85      ios_base::sync_with_stdio(false);
86      cout << fixed << setprecision(10);
87      cin.tie(nullptr);
88  #ifdef _MY
89      freopen("input.txt", "r", stdin);
90      freopen("output.txt", "w", stdout);
91  #endif
92
93      int n, m;
94      cin >> n >> m;
95
96      for (int i = 0; i < m; ++i) {
```

```
97          int a, b;
98          cin >> a >> b;
99          --a; --b;
100         edges.push_back({ a, b });
101         g[a].push_back({ b, i });
102         g[b].push_back({ a, i });
103     }
104
105     for (int i = 0; i < n; ++i) {
106         if (tin[i]) continue;
107         dfs_cuts(i, i);
108     }
109
110     newcol = (int)cuts.size() - 1;
111     for (int i = 0; i < n; ++i) {
112         if (used[i]) continue;
113         dfs_color(i, newcol, i);
114     }
115
116     sort(all(cuts));
117     for (int i = 0; i < m; ++i) {
118         int u = edges[i].first;
119         int v = edges[i].second;
120         added[u] = 1;
121         added[v] = 1;
122         if (is_cut(u)) {
123             int iu = int(lower_bound(all(cuts), u) -
                    begin(cuts));
124             int iv = color_edge[i];
125             g1[iu].push_back(iv);
126             g1[iv].push_back(iu);
127         }
128         swap(u, v);
129         if (is_cut(u)) {
130             int iu = int(lower_bound(all(cuts), u) -
                    begin(cuts));
131             int iv = color_edge[i];
132             g1[iu].push_back(iv);
133             g1[iv].push_back(iu);
134         }
135     }
```

```
136
137     int nn = (int)cuts.size() + newcol + 1;
138     for (int i = 0; i < nn; ++i) {
139         sort(all(g1[i]));
140         g1[i].resize(unique(all(g1[i])) - begin(g1[i
                ]));
141     }
142
143     for (int i = 0; i < m; ++i) {
144         comps[edges[i].first].insert(color_edge[i]);
145         comps[edges[i].second].insert(color_edge[i])
                ;
146     }
147     for (int i = 0; i < n; ++i) {
148         if (added[i]) continue;
149         comps[i].insert(++newcol);
150     }
151
152     int q;
153     cin >> q;
154     while (q--) {
155         int f, t;
156         cin >> f >> t;
157         --f; --t;
158         fill(used, used + nn, 0);
159         int u = *comps[f].begin();
160         int v = *comps[t].begin();
161         res.clear();
162         dfs_find(u, v);
163         res.erase(f);
164         res.erase(t);
165         cout << res.size() << endl;
166
167     }
168
169
170
171     return 0;
172 }
```

0.25 lca.cpp

```
1  struct LCA
2  {
3      int root = 0;
4      vi dep;
5      vector<vi> p;
6      void dfs(int v, int par, int curd, vector<vi>& g
           )
7      {
8          dep[v] = curd;
9          p[v][0] = par;
10
11         for (int i = 1; i < 20; ++i)
12         {
13             p[v][i] = p[p[v][i - 1]][i - 1];
14         }
15
16         for (auto& it : g[v])
17         {
18             if (it == par) continue;
19             dfs(it, v, curd + 1, g);
20         }
21     }
22     void init(vector<vi>& g)
23     {
24         dep.resize(g.size(), -1);
25         p.resize(g.size(), vi(20, -1));
26         dfs(root, root, 0, g);
27     }
28
29     int parent(int v, int h)
30     {
31         for (int i = 0; i < 20; ++i)
32         {
33             if (h & (1 << i))
34                 v = p[v][i];
35         }
36         return v;
37     }
```

```
38
39      int lca(int a, int b)
40      {
41          if (dep[a] < dep[b]) swap(a, b);
42          a = parent(a, dep[a] - dep[b]);
43          if (a == b) return a;
44
45          for (int i = 19; i >= 0; --i)
46          {
47              if (p[a][i] != p[b][i])
48              {
49                  a = p[a][i];
50                  b = p[b][i];
51              }
52          }
53          return p[a][0];
54      }
55      LCA() {}
56      LCA(vector<vi>& g, int rt = 0):root(rt) { init(g
           ); }
57  };
```

0.26 cutpoints.cpp

```
1   namespace FindAllCutPoints
2   {
3       vector<int> points;
4       vector<int> fup, tin;
5       int tin_global = 0;
6
7       void dfs(int cur, int parent, vector<vector<int
           >>& g)
8       {
9           bool is_ans = false;
10          int sons = 0;
11          tin[cur] = ++tin_global;
12          fup[cur] = tin_global;
13          for (int i = 0; i < g[cur].size(); i++)
14          {
15              int to = g[cur][i];
16              if (tin[to] == 0)
```

```
17              {
18                  dfs(to, cur, g);
19                  sons++;
20                  fup[cur] = min(fup[cur], fup[to]);
21                  if (fup[to] >= tin[cur])
22                      is_ans = true;
23              }
24              else
25              {
26                  if (to != parent)
27                  {
28                      fup[cur] = min(fup[cur], tin[to
                            ]);
29                  }
30              }
31          }
32          if ((cur != 0 && is_ans) || (cur == 0 &&
                sons > 1)) points.push_back(cur);
33      }
34
35      vector<int> find_cutpoints(vector<vector<int>>&
            g)
36      {
37          tin_global = 0;
38          int n = g.size();
39          fup.assign(n, -1);
40          tin.assign(n, 0);
41          points.clear();
42          dfs(0, -1, g);
43          return points;
44      }
45
46  }
47  using namespace FindAllCutPoints;
```

0.27 time$_c$*ounting.cpp*

```
1  //before main
2  #include <chrono>
3  chrono::time_point<chrono::steady_clock> cl;
4  double current_time() { return (chrono::steady_clock
```

```
    ::now() - cl).count() / 1e9; }
5
6
7
8  // at the beginning of main
9  cl = chrono::steady_clock::now();
10
11 // check time with
12 double cur_seconds = current_time();
```

0.28 binpow.cpp

```
1  ll binpow(ll a, ll p)
2  {
3      ll res = 1;
4      for (; p; p >>= 1)
5      {
6          if (p & 1) res *= a;
7          a *= a;
8      }
9      return res;
10 }
```

0.29 manacher.cpp

```
1  template <typename T>
2  vector<int> manacher(int n, const T &s) {
3    if (n == 0) {
4      return vector<int>();
5    }
6    vector<int> res(2 * n - 1, 0);
7    int l = -1, r = -1;
8    for (int z = 0; z < 2 * n - 1; z++) {
9      int i = (z + 1) >> 1;
10     int j = z >> 1;
11     int p = (i >= r ? 0 : min(r - i, res[2 * (l + r)
           - z]));
12     while (j + p + 1 < n && i - p - 1 >= 0) {
13       if (!(s[j + p + 1] == s[i - p - 1])) {
14         break;
15       }
```

```
16        p++;
17      }
18      if (j + p > r) {
19        l = i - p;
20        r = j + p;
21      }
22      res[z] = p;
23    }
24    return res;
25  }
26  template <typename T>
27  vector<int> manacher(const T &s) {
28    return manacher((int) s.size(), s);
29  }
```

0.30 includes.cpp

```
1  #include <fstream>
2  #include <random>
3  #include <chrono>
4  #include <iostream>
5  #include <vector>
6  #include <stack>
7  #include <set>
8  #include <map>
9  #include <queue>
10 #include <bitset>
11 #include <cmath>
12 #include <iomanip>
13 #include <sstream>
14 #include <string>
15 #include <algorithm>
16 #include <tuple>
17 #include <unordered_set>
18 #include <unordered_map>
```

0.31 newdelete.cpp

```
1  const int Mb = 230;
2  const int MAX_MEM = Mb * 1024 * 1024;
3  size_t mpos = 0;
```

```
4  char mem[MAX_MEM];
5  inline void* operator new(size_t n)
6  {
7      char * res = mem + mpos;
8      mpos += n;
9      assert(mpos <= MAX_MEM);
10     return (void*)res;
11 }
12 inline void operator delete(void*) {}
```

0.32 persistent stree ptr.cpp

```
1  namespace stree_ns
2  {
3  #ifdef _MY
4      const int maxn = 8;
5  #endif
6  #ifndef _MY
7      const int maxn = 1000 * 1000 + 5;
8  #endif
9
10     struct stree
11     {
12         //changable part
13         typedef int T;
14         static const T neutral = inf;
15         inline T func_get(T a, T b) { return min(a,
               b); }
16         inline void func_set(T& a, T& b) { a = b; }
17         //-------------
18
19         struct node
20         {
21             T val = neutral;
22             node* l = 0, *r = 0;
23         };
24
25
26         vector<node*> t;
27         stree() { t.push_back(new node()); }
28         node*& safe_ptr(node*& ptr) { return ptr ?
```

```
        ptr : ptr = new node(); }
29      int safe_get(node* ptr) { return ptr ? ptr->
          val : neutral; }
30      void build(vi& ar, node*& v, int tl = 0, int
          tr = maxn)
31      {
32          if (tl + 1 == tr) return void(v->val =
              ar[tl]);
33          int mid = (tl + tr) / 2;
34          if (tl < ar.size()) build(ar, safe_ptr(v
              ->l), tl, mid);
35          if (mid < ar.size()) build(ar, safe_ptr(
              v->r), mid, tr);
36          v->val = func_get(safe_get(v->l),
              safe_get(v->r));
37      }
38      node* set(int i, T x, node* v, int tl = 0,
          int tr = maxn)
39      {
40          node* curv = new node(*v);
41          if (i == tl && i + 1 == tr) { func_set(
              curv->val, x); return curv; }
42          int mid = (tl + tr) / 2;
43          if (i < mid) curv->l = set(i, x,
              safe_ptr(v->l), tl, mid);
44          else curv->r = set(i, x, safe_ptr(v->r),
               mid, tr);
45          curv->val = func_get(safe_get(curv->l),
              safe_get(curv->r));
46          return curv;
47      }
48      int get(int l, int r, node* v, int tl = 0,
          int tr = maxn)
49      {
50          if (tl == l && tr == r) return v->val;
51          int mid = (tl + tr) / 2;
52          int res = neutral;
53          if (l < mid) res = func_get(res, get(l,
              min(r, mid), safe_ptr(v->l), tl, mid)
              );
54          if (r > mid) res = func_get(res, get(max
```

```
                (l, mid), r, safe_ptr(v->r), mid, tr)
                );
55              return res;
56          }
57      };
58
59      int ind = 0;
60      char c[1000];
61      void print(stree::node* t, int side = 0, int
           prside = 0)
62      {
63  #ifdef _MY
64          if (!t) return;
65          c[ind] = side ^ prside ? '|' : '␣';
66          ++ind;
67          print(t->l, 0, side);
68          c[ind] = side ? '\\' : '/';
69          for (int i = 2; i <= ind; ++i) cout << '␣'
               << c[i] << '␣';
70          cout << '[' << t->val << ']' << endl;
71          print(t->r, 1, side);
72          --ind;
73  #endif
74      }
75  }
```

0.33 mincore.cpp

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4
5  int main()
6  {
7
8
9
10         return 0;
11 }
```

0.34 treap.cpp

```
1  namespace my_treap {
2      mt19937 rng(0);
3      inline bool check(int l, int r) { return rng() %
           (l + r) < l; }
4
5      struct Node {
6          Node *l, *r;
7          int val;
8          int cnt = 1;
9          Node() {}
10         Node(int val) :val(val) { l = r = 0; }
11     };
12     typedef Node* treap;
13
14     inline int cnt(treap t) { return t ? t->cnt : 0;
           }
15     inline void upd(treap& t) { if (t) t->cnt = cnt(
           t->l) + 1 + cnt(t->r); }
16     inline void push(treap& t) {}
17
18     void split(treap& tl, treap& tr, treap t, int
           key)
19     {
20         push(t);
21         if (!t) return void(tl = tr = nullptr);
22         if (cnt(t->l) < key) {
23             split(t->r, tr, t->r, key - cnt(t->l) -
                   1);
24             upd(tl = t);
25         }
26         else {
27             split(tl, t->l, t->l, key);
28             upd(tr = t);
29         }
30     }
31     void merge(treap& t, treap tl, treap tr) {
32         push(tl); push(tr);
33         if (!tl || !tr) { t = tl ? tl : tr; }
```

```
34          else if (check(cnt(tl), cnt(tr))) {
35              merge(tl->r, tl->r, tr);
36              t = tl;
37          }
38          else {
39              merge(tr->l, tl, tr->l);
40              t = tr;
41          }
42          upd(t);
43      }
44      int at(treap& t, int i) {
45          treap t1, t2, t3;
46          split(t1, t2, t, i);
47          split(t2, t3, t2, 1);
48          int res = t2->val;
49          merge(t, t1, t2);
50          merge(t, t, t3);
51          return res;
52      }
53      treap build(vi& ar) {
54          treap t = nullptr;
55          for (auto& it : ar) merge(t, t, new Node(it)
                );
56          return t;
57      }
58
59      int ind = 0;
60      char c[1000];
61      void print(Node* t, int side = 0, int prside =
            0)
62          {
63  #ifdef _MY
64                  if (!t) return;
65                  c[ind] = side ^ prside ? '|' : '␣';
66                  ++ind;
67                  print(t->l, 0, side);
68                  c[ind] = side ? '\\' : '/';
69                  for (int i = 2; i <= ind; ++i) cout
                        << '␣' << c[i] << '␣';
70                  cout << '[' << t->val << ']' << endl
                        ;
```

```
71                  print(t->r, 1, side);
72                  --ind;
73 #endif
74          }
75
76 }
77 using namespace my_treap;
```

0.35 java example.cpp

```
1  import java.util.*;
2  import java.math.*;
3  import java.io.*;
4
5  public class Main {
6      public static void main(String[] args) {
7          Scanner in = new Scanner(System.in);
8          PrintWriter out = new PrintWriter(System.out
               );
9
10         long one = Integer.MAX_VALUE / Integer.
               MAX_VALUE;
11         long zero = one - one;
12         long two = one + one;
13         long three = one + two;
14
15         long n, k;
16         n = in.nextLong();
17         k = in.nextLong();
18         long m = (n - three*k + one) / two;
19
20         m = Math.max(zero, m);
21
22         out.println(m);
23
24         out.close();
25     }
26 }
```

0.36 stree pushes ptr.cpp

```
1  //#pragma comment(linker, "/stack:200000000")
2  //#pragma GCC optimize("O3")
3  #include <bits/stdc++.h>
4  #define int ll
5  #define xx first
6  #define yy second
7  #define pb push_back
8  #define all(x) begin(x), end(x)
9  #define OUT(x) ((cout << x), exit(0))
10 using namespace std;
11 typedef double db;
12 typedef long long ll;
13 typedef vector<int> vi;
14 typedef pair<int, int> pii;
15 const db eps = 1e-9;
16 const db pi = acos(-1.0);
17 const db dinf = 1e250;
18 const ll INF = (ll)(2e18);
19 const int inf = (int)(1e9 + 7);
20 //------------------------------------------//
21
22 map<int, vector<pii>> m;
23
24 const int maxn = inf;
25
26 struct node
27 {
28     node* l = nullptr, * r = nullptr;
29     int32_t mx = 0, p = 0;
30 };
31 node*& safe_ptr(node*& ptr) { return ptr ? ptr : ptr
       = new node(); }
32 int safe_get(node*& ptr) { return ptr ? ptr->mx +
     ptr->p : 0; }
33 struct stree
34 {
35     node* t;
36     stree() { t = new node(); }
37
38
39     void push(node* v, int tl, int tr)
```

```
40      {
41          if (tl == tr) return;
42          v->mx += v->p;
43          if (tl + 1 != tr)
44          {
45              safe_ptr(v->l)->p += v->p;
46              safe_ptr(v->r)->p += v->p;
47          }
48          v->p = 0;
49      }
50
51      void set(int l, int r, int x, node* v, int tl =
           0, int tr = maxn)
52      {
53          if (tl == l && tr == r) return void(v->p +=
              x);
54
55          push(v, tl, tr);
56
57          int mid = (tl + tr) / 2;
58          if (l < mid) set(l, min(mid, r), x, safe_ptr
              (v->l), tl, mid);
59          if (r > mid) set(max(mid, l), r, x, safe_ptr
              (v->r), mid, tr);
60
61          push(safe_ptr(v->l), tl, mid);
62          push(safe_ptr(v->r), mid, tr);
63          v->mx = max(safe_get(v->l), safe_get(v->r));
64      }
65      int get(int l, int r, node* v, int tl = 0, int
           tr = maxn)
66      {
67          push(v, l, r);
68          if (l == tl && r == tr) return v->mx;
69          int res = -INF;
70          int mid = (tl + tr) / 2;
71          if (l < mid) res = max(res, get(l, min(mid,
              r), safe_ptr(v->l), tl, mid));
72          if (r > mid) res = max(res, get(max(mid, l),
               r, safe_ptr(v->r), mid, tr));
73          return res;
```

```
74      }
75
76      int get_max_ind(node* v, int tl = 0, int tr =
            maxn)
77      {
78          push(v, tl, tr);
79
80          if (tl + 1 == tr) return tl;
81          int mid = (tl + tr) / 2;
82          if (safe_get(v->l) == v->mx) return
                get_max_ind(safe_ptr(v->l), tl, mid);
83          else return get_max_ind(safe_ptr(v->r), mid,
                tr);
84      }
85
86  };
87
88  int ind = 0;
89  char c[1000];
90  void print(node* t, int side = 0, int prside = 0)
91  {
92  #ifdef _MY
93      if (!t) return;
94      c[ind] = side ^ prside ? '|' : '␣';
95      ++ind;
96      print(t->l, 0, side);
97      c[ind] = side ? '\\' : '/';
98      for (int i = 2; i <= ind; ++i) cout << '␣' << c[
            i] << '␣';
99      cout << '[' << t->mx << "," << t->p << ']' <<
            endl;
100     print(t->r, 1, side);
101     --ind;
102 #endif
103 }
104
105 int32_t main()
106 {
107     ios_base::sync_with_stdio(0);
108     cout << fixed << setprecision(10);
109     cin.tie(0);
```

```
110
111 #ifdef _MY
112     freopen("input.txt", "r", stdin);
113     freopen("output.txt", "w", stdout);
114 #endif
115
116     int n;
117     cin >> n;
118
119     set<pii> pts;
120
121     int res = 0;
122     pii resp;
123
124     m[0].pb({ 0,0 });
125     for (int i = 0; i < n; ++i)
126     {
127         int x, y, c;
128         cin >> x >> y >> c;
129         m[min(x, y)].pb({ max(x,y), c });
130         pts.insert({ x,y });
131
132     }
133
134     for (int i = 0; ; ++i)
135     {
136         if (!pts.count({ i,i }))
137         {
138             resp = { i,i };
139             break;
140         }
141     }
142
143
144
145     stree tr;
146
147
148     set<int> s;
149     for (auto& it : m)
150     {
```

```
151         s.insert(it.xx);
152         for (auto& itt : it.yy)
153             s.insert(itt.xx);
154     }
155     auto its1 = s.begin();
156     auto its2 = its1; ++its2;
157
158
159
160
161     for (; its1 != s.end(); ++its1, its2 == s.end()
            ? its2 : ++its2)
162     {
163         if (its2 != s.end())
164             tr.set(*its1, *its2, -*its1, tr.t);
165         else
166             tr.set(*its1, maxn, -*its1, tr.t);
167     }
168     //print(tr.t);
169
170     for (auto& it : m)
171     {
172         for (auto& itt : it.yy)
173             tr.set(itt.xx, maxn, itt.yy, tr.t);
174
175     }
176
177
178
179
180     auto it1 = m.begin();
181     auto it2 = it1; ++it2;
182
183     for (; it2 != m.end(); ++it1, ++it2)
184     {
185         //print(tr.t); cout << endl;
186         if (safe_get(tr.t) > res)
187         {
188             res = safe_get(tr.t);
189             resp = { it1->xx, tr.get_max_ind(tr.t)
                    };
```

```
190         }
191         //print(tr.t); cout << endl;
192         for (auto& itt : it1->yy) tr.set(itt.xx,
               maxn, -itt.yy, tr.t);
193         tr.set(it1->xx, maxn, it2->xx - it1->xx, tr.
               t);
194         tr.set(it1->xx, it2->xx, -INF, tr.t);
195     }
196
197     cout << res << endl;
198     cout << resp.xx << " " << resp.xx << " " << resp
           .yy << " " << resp.yy;
199
200
201     return 0;
202 }
```

0.37 gp$_h$ash$_t$able.cpp

```
1 #include <ext/pb_ds/assoc_container.hpp>
2 using namespace __gnu_pbds;
3 gp_hash_table<int, int> table;
```

0.38 stree pushes sum inc.cpp

```
1  struct stree {
2      vector<int> t;
3      vector<int> p;
4      int n = 1;
5      stree() {}
6      stree(int nn) {
7          while (n < nn) n *= 2;
8          t.assign(2 * n, 0);
9          p.assign(2 * n, 0);
10     }
11     void push(int v, int tl, int tr) {
12         t[v] += p[v] * (tr - tl);
13         if (v < n) {
14             p[v + v] += p[v];
15             p[v + v + 1] += p[v];
16         }
```

```
17          p[v] = 0;
18      }
19      void set(int l, int r, int x, int v = 1, int tl
           = 0, int tr = -1) {
20          if (tr == -1) tr = n;
21          if (l >= r) return;
22          push(v, tl, tr);
23          if (l == tl && r == tr) return void(p[v] +=
              x);
24          int mid = (tl + tr) / 2;
25          if (l < mid) set(l, min(r, mid), x, v + v,
              tl, mid);
26          if (r > mid) set(max(l, mid), r, x, v + v +
              1, mid, tr);
27          if (v < n) {
28              push(v + v, tl, mid);
29              push(v + v + 1, mid, tr);
30          }
31          t[v] = t[v + v] + t[v + v + 1];
32      }
33      int get(int l, int r, int v = 1, int tl = 0, int
            tr = -1) {
34          if (tr == -1) tr = n;
35          if (l >= r) return 0;
36          push(v, tl, tr);
37          if (l == tl && r == tr) return t[v];
38          int mid = (tl + tr) / 2;
39          int res = 0;
40          if (l < mid) res += get(l, min(r, mid), v +
              v, tl, mid);
41          if (r > mid) res += get(max(l, mid), r, v +
              v + 1, mid, tr);
42          return res;
43      }
44  };
```

0.39 dsu$_r$*ollback.cpp*

```
1   struct dsu {
2       vector<int> p, s, B;
3       dsu() {}
```

```
4       dsu(int n) {
5           p.resize(n);
6           s.assign(n, 1);
7           iota(all(p), 0);
8       }
9       int root(int v) { return p[v] == v ? v : root(p[
            v]); }
10      void unite(int a, int b) {
11          a = root(a);
12          b = root(b);
13          if (a == b) { B.push_back(-1); return; }
14          if (s[a] < s[b]) swap(a, b);
15          s[a] += s[b];
16          p[b] = a;
17          B.push_back(b);
18      }
19      void rollback() {
20          int b = B.back();
21          B.pop_back();
22                  if (b == -1) return;
23          s[p[b]] -= s[b];
24          p[b] = b;
25      }
26  };
```

0.40 dsu.cpp

```
1   struct dsu {
2       vi p, s;
3       dsu() {}
4       dsu(int n) {
5           p.resize(n);
6           s.assign(n, 1);
7           iota(all(p), 0);
8       }
9       int root(int v) {
10          if (p[v] != v) p[v] = root(p[v]);
11          return p[v];
12      }
13      void unite(int a, int b) {
14          a = root(a);
```

```
15          b = root(b);
16          if (a == b) return;
17          if (s[b] > s[a]) swap(a, b);
18          p[b] = a;
19          s[a] += s[b];
20      }
21
22  };
```

0.41 TeamReferenceHeader.tex

```
1  \documentclass[a4paper,12pt]{report}
2  \usepackage[T2A]{fontenc}
3  \usepackage[english,russian]{babel}     %russian
       support
4  \usepackage{indentfirst}                %first
       paragraph will be with indent
5  \usepackage{amsmath}                    %equations
6  \usepackage{listings}
7  \lstset{
8      language=C++,
9      breaklines=true,
10     numbers=left,
11     basicstyle=\ttfamily,
12 }
```

0.42 sufarray.cpp

```
1  vi sufarray(string s)
2  {
3      vi p;
4      vi c;
5
6      s += char(int(31));
7      int n = s.size();
8
9      p.resize(n);
10     c.resize(n);
11
12     c[0] = 0;
13     for (int i = 0; i < n; i++)
```

```
14      {
15          p[i] = i;
16      }
17
18      sort(all(p), [&s](int a, int b) { return s[a] <
            s[b]; });
19
20      int curclass = 0;
21      c[p[0]] = 0;
22      for (int i = 1; i < n; i++)
23      {
24          if (s[p[i]] != s[p[i - 1]]) ++curclass;
25          c[p[i]] = curclass;
26      }
27
28
29      for (int i = 2; i < n; i <<= 1)
30      {
31          vi cswap(n, inf);
32          vector<pii> next;
33          for (int j = 0; j < n; j++)
34              next.pb({ p[j], (p[j] + (i >> 1)) % n })
                    ;
35
36          sort(all(next),
37              [&c](pii a, pii b)
38              {
39                  if (c[a.first] != c[b.first]) return
                        c[a.first] < c[b.first];
40                  return c[a.second] < c[b.second];
41              });
42
43          for (int j = 0; j < next.size(); j++)
44          {
45              p[j] = next[j].first;
46          }
47
48          int curclass = 0;
49          cswap[p[0]] = 0;
50          for (int j = 1; j < n; ++j)
51          {
```

```
52              pii& a = next[j];
53              pii& b = next[j - 1];
54              if (c[a.first] != c[b.first] || c[a.
                   second] != c[b.second]) ++curclass;
55              cswap[p[j]] = curclass;
56          }
57
58          swap(c, cswap);
59
60      }
61
62      vi res(p.begin() + 1, p.end());
63      return res;
64
65  }
```

0.43 geom igor.cpp

```
1  int sgn(const ld &a) { return (a > eps ? 1 : (a < -
      eps ? -1 : 0)); }
2
3  struct pt {
4      ld x, y;
5      pt(ld x = 0, ld y = 0) : x(x), y(y) {}
6      const pt operator-(const pt &a)     const {
         return pt(x - a.x, y - a.y); }
7      const pt operator+(const pt &a)     const {
         return pt(x + a.x, y + a.y); }
8      const pt operator*(const ld &a)     const {
         return pt(x * a, y * a); }
9      const pt operator/(const ld &a)     const {
         return pt(x / a, y / a); }
10     const ld operator^(const pt &a)     const {
         return (x * a.y - y * a.x); }
11     const ld operator*(const pt &a)     const {
         return (x * a.x + y * a.y); }
12     const bool operator<(const pt &a)   const {
         return sgn(x - a.x) == -1 || (sgn(x - a.x) ==
          0 && sgn(y - a.y) == -1); }
13     const bool operator==(const pt &a)  const {
         return sgn(x - a.x) == 0 && sgn(y - a.y) ==
```

```
        0; }
14      const ld sqdist()                       const {
          return sq(x) + sq(y); }
15      const ld dist()                         const {
          return sqrtl((ld)sqdist()); }
16      const pt norm(const ld &a)              const {
          return pt(x, y) * (a / dist()); }
17      const pt rotateCW(const ld &ang)        const {
          return pt(x * cosl(ang) - y * sinl(ang),  x *
           sinl(ang) + y * cosl(ang)); }
18      const pt rotateCCW(const ld &ang)       const {
          return pt(x * cosl(ang) + y * sinl(ang), -x *
           sinl(ang) + y * cosl(ang)); }
19  };
20
21  ld ang(pt a, pt b) { return atan2l(a ^ b, a * b); }
22
23  struct line {
24      ld a, b, c;
25      line(pt p1, pt p2) {
26          a = -(p2.y - p1.y);
27          b =  (p2.x - p1.x);
28          c = -(a * p1.x + b * p1.y);
29          fix();
30      }
31      line(ld a = 1, ld b = 0, ld c = 0) : a(a), b(b),
            c(c) { fix(); }
32      void fix() {
33          ld d = sqrt(sq(a) + sq(b));
34          a /= d; b /= d; c /= d;
35          if (sgn(a) == -1 || (sgn(a) == 0 && sgn(b)
              == -1))
36              a = -a, b = -b, c = -c;
37      }
38      const ld dist(const pt &p)          const { return
            fabsl(a * p.x + b * p.y + c); }
39      const ld orientdist(const pt &p)    const { return
            a * p.x + b * p.y + c; }
40      const pt proj(const pt &p)          const { return
            p - pt(a, b) * orientdist(p); }
41      const pt symm(const pt &p)          const { return
```

```
          p - pt(a, b) * (orientdist(p) * 2.0); }
42 };
43
44 struct segment {
45     pt a, b;
46     segment(pt a = pt(), pt b = pt(1)) : a(a), b(b)
          {}
47     const bool on_segment(const pt &p) const {
          return sgn((a - b).dist() - (a - p).dist() -
          (b - p).dist()) == 0; }
48     const ld dist(const pt &p)         const { line
          l(a, b);
49         return (on_segment(l.proj(p)) ? l.dist(p) :
              min((a - p).dist(), (b - p).dist()));
50     }
51 };
52
53 struct circle {
54     pt center;
55     ld r;
56     circle(pt center = pt(), ld r = 1) : center(
          center), r(r) {}
57     const bool on_circle(const pt &p) const { return
          sgn((center - p).dist() - r) ==  0; }
58     const bool in_circle(const pt &p) const { return
          sgn((center - p).dist() - r) == -1; }
59     const bool out_crcle(const pt &p) const { return
          sgn((center - p).dist() - r) ==  1; }
60     const ld dist_out_circle(const pt &a, const pt &
          b) {
61         segment s(a, b);
62         line    l(a, b);
63         pt maybe_in = l.proj(center);
64         if (s.on_segment(maybe_in) && in_circle(
              maybe_in)) {
65             ld st1 = sqrtl((center - a).sqdist() -
                  sq(r));
66             ld st2 = sqrtl((center - b).sqdist() -
                  sq(r));
67             ld ang1 = PI / 2 - atan2l(r, st1);
68             ld ang2 = PI / 2 - atan2l(r, st2);
```

```
69              return st1 + st2 + r * (fabsl(ang(a -
                    center, b - center)) - ang1 - ang2);
70          }
71          return (a - b).dist();
72      }
73  };
74
75  struct polygon {
76      vector<pt> v;
77      polygon(vector<pt> v = vector<pt>()) : v(v) {}
78
79      int det(pt a, pt b, pt c) {
80          return sgn((c - b) ^ (b - a));
81      }
82      vector<pt> convex() {
83          sort(all(v));
84          vector<pt> cvx;
85          pt be = v.front(), en = v.back();
86          set<pt> up, dn;
87          for (int i = 0; i < v.size(); ++i) {
88              if (det(be, v[i], en) >= 0) up.insert(v[
                    i]);
89              if (det(be, v[i], en) <= 0) dn.insert(v[
                    i]);
90          }
91          for (sit it = up.begin(); it != up.end(); ++
                it) {
92              sit c = it;     if (*c == be) continue;
93              sit b = c; --b; if (*b == be) continue;
94              sit a = b; --a;
95              if (det(*a, *b, *c) <= 0) { up.erase(b);
                    --it; }
96          }
97          for (sit it = dn.begin(); it != dn.end(); ++
                it) {
98              sit c = it;     if (*c == be) continue;
99              sit b = c; --b; if (*b == be) continue;
100             sit a = b; --a;
101             if (det(*a, *b, *c) >= 0) { dn.erase(b);
                    --it; }
102         }
```

```
103         for (sit it = ++up.begin(); it != up.end();
              ++it) cvx.pb(*it);
104         for (set<pt>::reverse_iterator it = ++dn.
              rbegin(); it != dn.rend(); ++it) cvx.pb(*
              it);
105         return cvx;
106     }
107     ld perimeter() {
108         ld ans = 0;
109         for (int i = 0; i < v.size(); ++i)
110             ans += (v[i] - v[(i + 1) % v.size()]).
                  dist();
111         return ans;
112     }
113     ld square() {
114         ld ans = 0;
115         for (int i = 0; i < v.size(); ++i)
116             ans += v[i] ^ v[(i + 1) % v.size()];
117         return fabsl(ans) / 2.0;
118     }
119 };
120
121 struct intersections {
122     static int line_line(const line &a, const line &
          b, pt &p) {
123         pt v_a(a.a, a.b);
124         pt v_b(b.a, b.b);
125         if (v_a == v_b && sgn(a.c - b.c) == 0)
              return 2;
126         if (v_a == v_b && sgn(a.c - b.c) != 0)
              return 0;
127         ld d0 = a.a * b.b - a.b * b.a;
128         p = pt(a.c * b.b - a.b * b.c, a.a * b.c - a.
              c * b.a) / (-d0);
129         return 1;
130     }
131     static int line_circle(const circle &C, const
          line &L, pt &one, pt &two) {
132         pt p = L.proj(C.center);
133         if (C.on_circle(p)) { one = two = p; return
              1; }
```

```
134         if (C.out_crcle(p)) return 0;
135         if (C.in_circle(p)) {
136             pt vec = pt(-L.b, L.a) * sqrtl(sq(C.r) -
                    (C.center - p).sqdist());
137             one = p + vec;
138             two = p - vec;
139             return 2;
140         }
141         return -1;
142     }
143     static int circle_circle(const circle &a, const
           circle &b, pt &one, pt &two) {
144         line l = line(-2.0 * a.center.x + 2.0 * b.
               center.x,
145             -2.0 * a.center.y + 2.0 * b.center.y,
146             sq(a.center.x) + sq(a.center.y) -
147             sq(b.center.x) - sq(b.center.y) -
148             sq(a.r) + sq(b.r));
149         return line_circle(b, l, one, two);
150     }
151     static int segment_segment(const segment &a,
           const segment &b, pt &p) {
152         bool cross = (sgn((a.a - b.a) ^ (a.b - b.a))
                * sgn((a.a - b.b) ^ (a.b - b.b)) <= 0 &&
153             sgn((b.a - a.a) ^ (b.b - a.a)) * sgn((b.
                   a - a.b) ^ (b.b - a.b)) <= 0);
154         ld xy = max(min(a.a.y, a.b.y), min(b.a.y, b.
               b.y));
155         ld xx = max(min(a.a.x, a.b.x), min(b.a.x, b.
               b.x));
156         ld ny = min(max(a.a.y, a.b.y), max(b.a.y, b.
               b.y));
157         ld nx = min(max(a.a.x, a.b.x), max(b.a.x, b.
               b.x));
158         bool bbox = (sgn(nx - xx) >= 0 && sgn(ny -
               xy) >= 0);
159         if (cross && bbox) return line_line(line(a.a
               , a.b), line(b.a, b.b), p);
160         return 0;
161     }
162 };
```

0.44 z func.cpp

```
1  template<class t>
2  vector<int> z_func(t s) {
3      vector<int> z(s.size(), 0);
4      int l = 0;
5      int r = 0;
6      for (int i = 1; i < s.size(); i++) {
7          int count = 0;
8          if (i <= r)  count = min(z[i - l], r - i +
               1);
9          for (int j = count; i + j < s.size() && s[j]
               == s[i + j]; j++)
10             count++;
11         z[i] = count;
12         if (i + count >= r){
13             l = i;
14             r = i + count - 1;
15         }
16     }
17     return z;
18 }
```

0.45 prefix func.cpp

```
1  template<class t>
2  vector<int> p_func(t s) {
3      int n = s.size();
4      vector<int> p(n, 0);
5      for (int i = 1; i < n; i++) {
6          p[i] = p[i - 1];
7          while (p[i] > 0 && s[i] != s[p[i]])
8              p[i] = p[p[i] - 1];
9          if (s[i] == s[p[i]]) p[i]++;
10     }
11     return p;
12 }
```