Team Reference
Universidad Central de las Villas : 3N1?M4

Miembros del Equipo: Alejandro Jiménez Fabián, Ruddy Guerrero Álvarez, Luis Enrique Gonzales Saborit

*** Significa:** Solución a un Problema

Contents

1. Utils

1.1. C++ Template.

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

typedef long long int ll;
typedef long double ld;
typedef pair<int,int> pii;
typedef pair<ll,ll> pll;

#define INIT ios_base::sync_with_stdio(false);\
         cin.tie(0),cout.tie()
#define endl '\n'
#define fr first
#define sc second
#define pb push_back
#define eb emplace_back
#define mp make_pair
#define lb lower_bound
#define ub upper_bound
#define ins insert
#define ers erase
#define sz(c) ((int)(c).size())
#define all(x) (x).begin(),(x).end()
#define unique(x) (x).resize(unique(all(x))-(x).begin())
#define debug(_fmt,...) fprintf(stderr,"("#__VA_ARGS__ ")␣=␣(" \
 _fmt")\n",__VA_ARGS__)

int main()
{
 #ifdef OJUDGE
  //freopen("in","r",stdin);
 #endif
 INIT;



 return 0;
}
```

1.2. Java Fast Input.

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.StringTokenizer;

public class InputReader
{
 public BufferedReader reader;
 public StringTokenizer tokenizer;

 public InputReader(InputStream stream) {
  reader = new BufferedReader(new InputStreamReader(stream), 32768);
  tokenizer = null;
 }

 public String next() {
  while (tokenizer == null || !tokenizer.hasMoreTokens()) {
   try {
    tokenizer = new StringTokenizer(reader.readLine());
   } catch (IOException e) {
    throw new RuntimeException(e);
   }
  }
  return tokenizer.nextToken();
 }

 public int nextInt() {
  return Integer.parseInt(next());
 }
 public long nextLong() { return Long.parseLong(next()); }
}
```

1.3. Test Input Generator.

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

int main()
{
 srand(time(nullptr));
 int tc;
 cin >> tc;
 while(tc--)
 {
  string str;
  stringstream ss;
  ss << tc;
  ss >> str;
  str = "intro"+str;
  freopen(str.data(),"w",stdout);
  // Code goes here ...
  fflush(stdout);
 }

 return 0;
}
```

1.4. Makefile.

```
%: ; g++-7 -Wall $@.cpp -o $@ -DOJUDGE
```

1.5. Sample Input Tester.

```
#!/bin/bash
targ=$1
if (( $# == 0 ))
then targ=a
fi
rm "$targ";make "$targ"
for f in in*
do
 echo "=====␣TEST:␣$f␣====="
 ./"$targ" < "$f"
done
```

1.6. Tester against Correct Solution.

```
#!/bin/bash
targ=$1;tester=$2
if (( $# == 0 ))
then targ=a;tester=model
fi
rm "$targ";rm "$tester"
make "$targ";make "$tester"
for f in in*
do
 echo "=====␣TEST:␣$f␣====="
# echo "OUTPUT:"
 ./"$targ" < "$f" > "tmp1"
# echo "ANSWER::"
 ./"$tester" < "$f" > "tmp2"

 if cmp "tmp1" "tmp2"
 then echo OK;ok=$((ok+1))
 else echo WA;wa=$((wa+1))
 fi
done
echo "OK:␣$ok␣WA:␣$wa"
```

2. Data Structures

2.1. Kd-Tree*.

```
//testeado en http://coj.uci.cu/24h/problem.xhtml?pid=2067 (City Houses)
/* Compute for each house, the minimum distance
 * to any of the other houses (Manhattan distance)
 */
#include <algorithm>
#include <cstdio>
#include <climits>
#include <cmath>
using namespace std;
typedef long long i64;
const int MAX = 1e5 + 10;
#define sqr(x) (abs(x))
struct point {
      int x, y;
} data[MAX], d[MAX];
inline bool cmpx(const point& a, const point& b) {
      return a.x < b.x;
}

inline bool cmpy(const point& a, const point& b) {
      return a.y < b.y;
}
inline i64 dist(const point& point1, const point& point2) {
      return abs(point1.x - point2.x) + abs(point1.y - point2.y);
}

int cant[4*MAX];
void Create2DTree(int index, int left, int der, bool h = 1) {
      cant[index] = der - left + 1;
      if (der - left + 1 <= 1) return;
      int med = (left + der + 1) / 2;
      nth_element(data + left, data + med,
   data + der + 1, h ? cmpx : cmpy);
      Create2DTree(2 * index, left, med - 1, h ^ 1);
      Create2DTree(2 * index + 1, med + 1, der, h ^ 1);
}

point que;
i64 best;
void query(int index, int left, int der, int h) {
      if (cant[index] <= 0) return;

      int m = (left + der + 1) / 2;
      if (dist (data[m], que))
      best = min(best, dist(data[m], que));

      if (cant[index] == 1)return;

      if (!(h & 1)) {
            bool dir = cmpx(que, data[m]);
            if (dir) {
                  query(2 * index, left, m - 1, h + 1);
                  if (best >= sqr(que.x - data[m].x))
                        query(2 * index + 1, m + 1, der, h + 1);
            } else {
                  query(2 * index + 1, m + 1, der, h + 1);
                  if (best >= sqr(que.x - data[m].x))
                        query(2 * index, left, m - 1, h + 1);
            }
      } else {
            bool dir = cmpy(que, data[m]);
            if (dir) {
                  query(2 * index, left, m - 1, h + 1);
                  if (best >= sqr(que.y - data[m].y))
                        query(2 * index + 1, m + 1, der, h + 1);
            } else {
                  query(2 * index + 1, m + 1, der, h + 1);
                  if (best >= sqr(que.y - data[m].y))
                        query(2 * index, left, m - 1, h + 1);
            }
      }
}

int main() {
      int n, q; scanf("%d", &n);
      for (int i = 1; i <= n; i++){
            scanf("%d%d", &data[i].x, &data[i].y);
            d[i] = data[i];
   }

      Create2DTree(1, 1, n);
      q = n;
      for (int i = 1; i <= n; ++i) {
            best = LLONG_MAX;
            que = d[i];
```

```
            query(1, 1, n, 0);
            printf("%lld\n", (best));
```

2.2. Vantage Point Tree* -1.

```
/* Vantage point Tree
 * testeado en http://coj.uci.cu/24h/problem.xhtml?pid=1914
 *
 * Consider having N (1 <= N <= 10^4) points in the 2D space.
 * You will be queried with Q (1 <= Q <= 10^4) circles and you
 * need print how many of the given points are inside the circle.
 */
#include <bits/stdc++.h>

using namespace std;

typedef complex<double> point;

namespace std
{
      bool operator <(point p, point q)
      {
            if (real(p) != real(q))
                  return real(p) < real(q);
            return imag(p) < imag(q);
      }
}

struct vantage_point_tree
{
      struct node
      {
            point p;
            int can;
            double th;
            node *l, *r;
      }*root;

      vector<pair<double, point>> aux;

      vantage_point_tree(vector<point> ps)
      {
            for (int i = 0; i < ps.size(); ++i)
                  aux.push_back({ 0, ps[i] });
            root = build(0, ps.size());
      }
      node *build(int l, int r)
      {
            if (l == r)
                  return 0;
            swap(aux[l], aux[l + rand() % (r - l)]);
            point p = aux[l++].second;
            if (l == r)
                  return new node({ p });
            for (int i = l; i < r; ++i)
                  aux[i].first = norm(p - aux[i].second);
            int m = (l + r) / 2;
            nth_element(aux.begin() + l, aux.begin() + m, aux.begin() + r);
            return new node({ p, m-l, sqrt(aux[m].first), build(l, m),
          build(m, r) });
      }

      int k_nn(node *t, point p, int k)
      {
            if (!t)
                  return 0;

            double d = sqrt (norm (p - t->p));

      int a = 0;

            if (k >= d + t -> th)
         a = t -> can + 1;
       else
         if (d <= k + t -> th)
            a = k_nn (t->l, p, k) + (d <= k);

            int b = k_nn (t->r, p, k);
            return a+b;
      }

      int k_nn (point p, int k){
            return k_nn(root, p, k);
      }
};
```

```
int x, y, r;

int main(){

   int n, m;scanf ("%d%d", &n, &m);
   vector <point> v;
   for (int i = 1; i <= n; ++i){
       scanf ("%d%d", &x, &y);
       v.push_back ({x, y});
   }
   vantage_point_tree vp (v);
   for (int i = 1; i <= m; ++i){
       scanf ("%d%d%d", &x, &y, &r);
       point p1 = {x, y};
       int p = vp.k_nn (p1, r);
       printf ("%d\n", p);
   }

   return 0;
}
```

2.3. Vantage Point Tree* -2.

```
/*
   Vantage Point Tree (vp tree)

   Description:
   Vantage point tree is a metric tree.
   Each tree node has a point, radius, and two childs.
   The points of left descendants are contained in the ball B(p,r)
   and the points of right descendants are excluded from the ball.

   We can find k-nearest neighbors of a given point p efficiently
   by pruning search.

   Complexity:
   Construction: O(n log n)
   Search: O(log n)
*/

typedef complex<double> point;

namespace std
{
   bool operator <(point p, point q)
   {
      if (real(p) != real(q))
         return real(p) < real(q);
      return imag(p) < imag(q);
   }
}

struct vantage_point_tree
{
   struct node
   {
      point p;
      double th;
      node *l, *r;
   }*root;

   vector<pair<double, point>> aux;

   vantage_point_tree(vector<point> ps)
   {
      for (int i = 0; i < ps.size(); ++i)
         aux.push_back({ 0, ps[i] });
      root = build(0, ps.size());
   }

   node *build(int l, int r)
   {
      if (l == r)
         return 0;
      swap(aux[l], aux[l + rand() % (r - l)]);
      point p = aux[l++].second;
      if (l == r)
         return new node({ p });
      for (int i = l; i < r; ++i)
         aux[i].first = norm(p - aux[i].second);
      int m = (l + r) / 2;
      nth_element(aux.begin() + l, aux.begin() + m, aux.begin() + r);
      return new node({ p, sqrt(aux[m].first), build(l, m),
       build(m, r) });
   }

   priority_queue<pair<double, node*>> que;
```

```
   void k_nn(node *t, point p, int k)
   {
      if (!t)
         return;
      double d = abs(p - t->p);
      if (que.size() < k)
         que.push({ d, t });
      else if (que.top().first > d)
      {
         que.pop();
         que.push({ d, t });
      }
      if (!t->l && !t->r)
         return;
      if (d < t->th)
      {
         k_nn(t->l, p, k);
         if (t->th - d <= que.top().first)
            k_nn(t->r, p, k);
      }
      else
      {
         k_nn(t->r, p, k);
         if (d - t->th <= que.top().first)
            k_nn(t->l, p, k);
      }
   }

   vector<point> k_nn(point p, int k)
   {
      k_nn(root, p, k);
      vector<point> ans;
      for (; !que.empty(); que.pop())
         ans.push_back(que.top().second->p);
      reverse(ans.begin(), ans.end());
      return ans;
   }
};
```

2.4. Persistent Segment Tree.

```
struct node { //struct de nodo de segment tree
   int v, left, right;

   node(int v, int left, int right)
    : v(v), left(left), right(right) {}
};

//IMPORTANTE: siempre recordar annadir null al stree,
// osea hacer stree.push_back(null)
node null(0, 0, 0);
vector <node> stree;

int root[MAX]; //arreglos de raices

int build(int st, int nd) {
   if(st == nd) {
      stree.push_back(node(a[st], 0, 0));
      return (int) stree.size() - 1;
   }

   int mid = (st + nd) >> 1;

   int left = build(st, mid);
   int right = build(mid + 1, nd);

   stree.push_back(node(0, left, right)); //combino como haga falta
   return (int) stree.size() - 1;
}

int insert(int x, int st, int nd, int p, int v) {
   if(st > p || nd < p)
      return x; //si .. tengo q regresar x

   if(st == nd) {
      stree.push_back(node(v, 0, 0));
      return (int) stree.size() - 1;
   }

   int mid = (st + nd) >> 1;

   int left = insert(stree[x].left, st, mid, p, v);
   int right = insert(stree[x].right, mid + 1, nd, p, v);

   stree.push_back(node(stree[left].v + stree[right].v, left, right));
   //combino como haga falta, esto es un ejemplo
   return (int) stree.size() - 1;
```

```
}

int query(int x, int st, int nd, int p) { //query .. casi igual q un
                                // stree normal
   if(st > p || nd < p)
      return -1;

   if(st == nd)
      return stree[x].v;

   int mid = (st + nd) >> 1;

   int left = query(stree[x].left, st, mid, p);
   int right = query(stree[x].right, mid + 1, nd, p);

   return max(left, right);
}
```

2.5. Implicit indexes Cartesian Tree* -1.

```
// Implicit indexes Cartesian Tree
// testeado en https://www.spoj.com/problems/GSS6/
/*
 Given a sequence A of N (N <= 100000) integers,
 you have to apply Q (Q <= 100000) operations:
 Insert, delete, replace an element, find the
 maximum contiguous(non empty) sum in a given interval.
*/
#include <bits/stdc++.h>

using namespace std;

typedef struct item * pitem;
struct item {
 int prior;
 int pre, suf, msum;
 int value, tot;
 int cnt, rev;
 pitem l, r;

 item (){
  prior = rand ();
  cnt = 1;
  value = tot = pre = suf;
  msum = -1<<30;
  rev = 0;
  l = r = 0;
 }
};

int cnt (pitem it) {return it ? it->cnt : 0;}
void upd_cnt (pitem it) {if (it)it->cnt = cnt(it->l) + cnt(it->r) + 1;}
int tot (pitem it) {return it ? it->tot : 0;}

void upd_tot (pitem it) {
 if (it)
  it->tot = tot(it->l) + tot(it->r) + it -> value;
}

int suf (pitem it){return it ? it -> suf : 0;}
int pre (pitem it){return it ? it -> pre : 0;}
int msum (pitem it) {return it ? it->msum : -1<<30;}

void upd_msum (pitem it) {
 if (it)
  it->msum = max ({msum(it->l), msum (it->r),
   suf (it -> l) + it -> value + pre (it -> r)});
}

void upd_pre (pitem it){
 if (it)
  it -> pre = max (pre (it -> l),
   tot (it -> l) + it -> value + pre (it -> r));
}

void upd_suf (pitem it){
 if (it)
  it -> suf = max (suf (it -> r),
   tot (it -> r) + it -> value + suf (it -> l));
}

void push (pitem it) {
 if (it && it->rev) {
  it->rev = false;
  swap (it->l, it->r);
  if (it->l) it->l->rev ^= true;
  if (it->r) it->r->rev ^= true;
 }
```

```
}

void merge (pitem & t, pitem l, pitem r) {
 push (l); push (r);
 if (!l || !r) t = l ? l : r;
 else if (l->prior > r->prior) merge (l->r, l->r, r), t = l;
 else merge (r->l, l, r->l), t = r;
 upd_cnt (t); upd_tot (t);
 upd_pre (t); upd_suf (t);
 upd_msum (t);
}

void split (pitem t, pitem & l, pitem & r, int key, int add = 0) {
 if (!t) return void( l = r = 0 );
 push (t);
 int cur_key = add + cnt(t->l);
 if (key <= cur_key) split (t->l, l, t->l, key, add), r = t;
 else split (t->r, t->r, r, key, add + 1 + cnt(t->l)), l = t;
 upd_cnt (t); upd_tot (t);
 upd_pre (t); upd_suf (t);
 upd_msum (t);
}

void reverse (pitem t, int l, int r) {
 pitem t1, t2, t3;
 split (t, t1, t2, l); split (t2, t2, t3, r-l+1);
 t2->rev ^= true;
 merge (t, t1, t2); merge (t, t, t3);
}

void output (pitem t) {
 if (!t) return;
 push (t);
 output (t->l);
 printf ("%c", t->value);
 output (t->r);
}

int main(){
 srand(time(0));
 ios_base::sync_with_stdio (0);
 cin.tie (0);
 cout.tie (0);

 pitem root = 0;
 int n; cin >> n;
 for (int i = 0; i < n; ++i){
  int v; cin >> v;
  pitem x = new item ();
  x -> value = v;
  x -> msum = v;
  x -> pre = x -> suf = max (0, v);
  merge (root, root, x);
 }

 cin >> n;
 int x, y;
 pitem tem = 0, x1;
 pitem b = 0;
 while (n--){
  char s;
  cin >> s;
  switch (s){
   case 'I':
    cin >> x >> y;
    tem = 0;
    split (root, root, tem, x-1);

    x1 = new item ();
    x1 -> value = y;
    x1 -> msum = y;
    x1 -> pre = x1 -> suf = max (0, y);

    merge (root, root, x1);
    merge (root, root, tem);

    break ;
   case 'D':
    cin >> x;
    tem = 0, b = 0;
    split (root, root, tem, x-1);
    split (tem, b, tem, 1);
    merge (root, root, tem);
    break ;
   case 'R':
    cin >> x >> y;
    tem = 0, b = 0;

    x1 = new item ();
    x1 -> value = y;
    x1 -> msum = y;
    x1 -> pre = x1 -> suf = max (0, y);

    split (root, root, tem, x-1);
```

```
      merge (root, root, x1);

      split (tem, b, tem, 1);
      merge (root, root, tem);
      break ;
     case 'Q':
      cin >> x >> y;
      tem = 0, b = 0;
      split (root, root, tem, x-1);
      split (tem, tem, b, y-x+1);
      cout << msum (tem) << "\n";
      merge (root, root, tem);
      merge (root, root, b);

      break ;
    }
  }
  return 0;
}
```

2.6. Implicit indexes Cartesian Tree* -2.

```
// Implicit indexes Cartesian tree
// testeado (Robotic Sort) https://www.spoj.com/problems/CERC07S/
#include <bits/stdc++.h>
using namespace std;

typedef struct item * pitem;
struct item
{
 int prior, value, mi, can, pos, mpos, cnt;
 bool rev;
 pitem l, r;

 item (){
  prior = rand ();cnt = 1;
  value = 0;mi = INT_MAX;pos = INT_MAX;
  mpos = INT_MAX;
  can = rev = 0;
  l = r = 0;
 }
};

int cnt (pitem it) {return it ? it->cnt : 0;}
void upd_cnt (pitem it) {if (it) it->cnt = cnt(it->l) + cnt(it->r) + 1;}
int mi (pitem it){return it ? it -> mi : INT_MAX; }

int can (pitem it){
  if (!it) return 0;
  if (it -> rev == 0) return it -> can;
  else return cnt (it) - it -> can + 1;
}

int mpos (pitem it){return it ? it -> mpos : INT_MAX;}

void upd_mi (pitem it){
  if (it){
    int m1 = mi (it -> l), m2 = mi (it -> r);
    int p1 = mpos (it -> l), p2 = mpos (it -> r);
    int c1 = can (it -> l), c2 = can (it -> r);

    int v = it -> value;
    int p = it -> pos;

    int menor = min ({v, m1, m2});
    int pos = 1<<30, can = 0;

    if (m1 == menor && pos > p1){
      can = c1;
      pos = p1;
    }

    if (m2 == menor && pos > p2){
      can = c2 + 1 + cnt (it -> l);
      pos = p2;
    }

    if (v == menor && pos > p){
      can = 1 + cnt (it -> l);
      pos = p;
    }

    it -> mpos = pos;
    it -> mi = menor;
    it -> can = can;
  }
}
```

```
void push (pitem it) {
  if (it && it->rev) {
    it->rev = false;
    swap (it->l, it->r);
    if (it->l) it->l->rev ^= true;
    if (it->r) it->r->rev ^= true;
  }
}

void merge (pitem & t, pitem l, pitem r) {
  push (l); push (r);
  if (!l || !r) t = l ? l : r;
  else if (l->prior > r->prior) merge (l->r, l->r, r), t = l;
  else merge (r->l, l, r->l), t = r;
  upd_cnt (t); upd_mi (t);
}

void split (pitem t, pitem & l, pitem & r, int key, int add = 0) {
  if (!t) return void( l = r = 0 );
  push (t);
  int cur_key = add + cnt(t->l);
  if (key <= cur_key) split (t->l, l, t->l, key, add), r = t;
  else split (t->r, t->r, r, key, add + 1 + cnt(t->l)), l = t;
  upd_cnt (t); upd_mi (t);
}

void reverse (pitem t, int l, int r) {
  pitem t1, t2, t3;
  split (t, t1, t2, l); split (t2, t2, t3, r-l+1);
  t2->rev ^= true;
  merge (t, t1, t2); merge (t, t, t3);
}

void erase (pitem it){
  if (it){
    erase (it -> l);
    erase (it -> r);
    delete (it);
  }
}

void output (pitem t) {
  if (!t) return;
  push (t);
  output (t->l);
  printf ("%d␣%d␣%d␣mpos:␣%d␣mi:␣%d␣can:␣%d\n",
  t->value, t -> pos, t -> prior, t -> mpos, t -> mi, t -> can);
  output (t->r);
}

int n, x;

int main(){
  srand(time(0));
  while (scanf ("%d", &n) && n){
    pitem root = 0;
    for (int i = 0; i < n; ++i){
      scanf ("%d", &x);
      pitem tem = new item ();
      tem -> value = x;
      tem -> mi = x;
      tem -> can = 1;
      tem -> pos = i+1;
      tem -> mpos = i+1;
      merge (root, root, tem);
    }

    int cant = 0;
    while (n--){
      int p = can (root);

      printf ("%d", p+cant);
      if (n == 0) printf ("\n");
      else printf ("␣");

      pitem x = 0;
      reverse (root, 0, p-1);
      split (root, x, root, 1);

      cant++;
      erase (x);
    }
    erase (root);
  }
  return 0;
}
```

2.7. Presistent Treap*.

```
/* Treap persistente
 * testeado https://www.codechef.com/status/GENETICS
 *
 * >> cross operation - they take DNA1 and DNA2 and
 * numbers k1 and k2. Then two new DNAs are
 * created: DNA3 = DNA1[1..k1]+DNA2[k2+1..] and
 * DNA4 = DNA2[1..k2]+DNA1[k1+1..].
 * >> mutate operation - they take a DNA, number k
 * and one of the bases. Then they replace the
 * base in position k in DNA with that base.
 * >> count operation - they take DNA and numbers k1
 * and k2 (k1 <= k2). This operation should return
 * the number of A, G, T, C bases in DNA[k1..k2].
 */
#include <bits/stdc++.h>
using namespace std;

char a[300005];

typedef struct item * pitem;
struct item {

     int prior;
     int sum[5];
     int cnt;
     int id;
     pitem l, r;
     item (){
          prior = rand ();
          cnt = 1;
          id = 0;
          memset (sum, 0, sizeof (sum));
          l = r = 0;
     }
};

int f (char car){
   if (car == 'A') return 0;
   if (car == 'G') return 1;
   if (car == 'T') return 2;
   return 3;
}

int cnt (pitem it) {
 return it ? it->cnt : 0;
}

int sum (pitem it, int k) {
 return it ? it->sum[k] : 0;
}

void upd_cnt (pitem it) {
 if (it)
 it->cnt = cnt(it->l) + cnt(it->r) + 1;
}

void upd_sum (pitem it){
    if (it){
       for (int i = 0; i < 4; ++i)
           it -> sum[i] = sum (it -> l, i) + sum (it -> r, i);
       it -> sum[it -> id]++;
    }
}

void merge (pitem & t, pitem l, pitem r) {

    if (!l || !r)
       t = l ? l : r;
    else
      if (l->prior > r->prior)
         merge (l->r, l->r, r), t = l;
       else
         merge (r->l, l, r->l), t = r;
    upd_cnt (t);
    upd_sum (t);
}

pitem clone (pitem x){

     pitem y = new item ();
     if (x -> r)y -> r = x -> r;
     if (x -> l)y -> l = x -> l;

     y -> cnt = x -> cnt;
     y -> id = x -> id;
     y -> prior = x -> prior;
     for (int i = 0; i < 4; ++i)
         y -> sum[i] = x -> sum[i];
     return y;
}
```

```
void splitp (pitem t, pitem & l, pitem & r, int key, int add = 0) {

    if (!t)
       return void( l = r = 0 );

    t = clone (t);

    int cur_key = add + cnt(t->l);
    if (key <= cur_key)
       splitp (t->l, l, t->l, key, add), r = t;
     else
       splitp (t->r, t->r, r, key, add + 1 + cnt(t->l)), l = t;

    upd_cnt (t);
    upd_sum (t);
}

void print (pitem t){
    if (t){
       print (t -> l);
       if (t -> id == 0) printf ("A");
       if (t -> id == 1) printf ("G");
       if (t -> id == 2) printf ("T");
       if (t -> id == 3) printf ("C");
       print (t -> r);
    }
}

int main(){

   srand(time(0));

   int n;
   cin >> n; vector <pitem> v;
   for (int i = 0; i < n; ++i){
       cin >> a;
       int l = strlen (a);
       pitem root = 0;
       for (int j = 0; j < l; ++j){
           pitem tem = new item();
           tem -> sum[f(a[j])]++;
           tem -> id = f (a[j]);
           merge (root, root, tem);
       }

       v.push_back (root);
}

int q;
cin >> q;
while (q--){
     string s; int id1, id2, k1, k2; char car;
     cin >> s;
     if (s == "COUNT"){
        cin >> id1 >> k1 >> k2; id1--;
        pitem x = 0, y = 0;
        splitp (v[id1], v[id1], x, k1-1);
        splitp (x, x, y, k2-k1+1);

        for (int i = 0; i < 4; ++i){
            cout << x -> sum[i];
            if (i == 3) cout << "\n";
             else cout << "␣";
        }

        merge (v[id1], v[id1], x);
        merge (v[id1], v[id1], y);
     }
      else
        if (s == "MUTATE"){
           cin >> id1 >> k1 >> car; id1--;

           pitem x = 0, y = 0;
           splitp (v[id1], v[id1], x, k1-1);
           splitp (x, x, y, 1);

           x -> id = f (car);

           merge (v[id1], v[id1], x);
           merge (v[id1], v[id1], y);
        }
        else{
           cin >> id1 >> id2 >> k1 >> k2; id1--; id2--;

           pitem x1 = 0, y1 = 0, x2 = 0, y2 = 0, root1 = 0,
            root2 = 0;

           splitp (v[id1], x1, y1, k1);
           splitp (v[id2], x2, y2, k2);

           merge (root1, x1, y2);
           merge (root2, x2, y1);
```

```
            // print (x1); printf("\n"); print (y1);
            // print (x2); printf("\n"); print (y2);

             v.push_back (root1);
             v.push_back (root2);
          }

          //for (auto it : v){print (it); printf ("\n");}
          // printf ("--------------\n");
   }

   return 0;
```

2.8. Treap Pintar*.

```
/*
S i j c: change the characters in the range [i, j] to the value c
R i j: reverse the substring starting from position i to position j
C i j c: count the number of characters with value c in
      the range [i, j]
testeado en COJ problem 2418
*/
#include <bits/stdc++.h>
using namespace std;

typedef struct item * pitem;
struct item {
     int prior, value, cnt, c[27];
     bool mar;
     bool rev;
     pitem l, r;

     item (){
          prior = rand ();
          value = cnt = 0;
          rev = mar = 0;
          l = r = 0;
          memset (c, 0, sizeof (c));
     }
};

int cnt (pitem it) {
   return it ? it->cnt : 0;
}

void upd_cnt (pitem it) {
    if (it)
       it->cnt = cnt(it->l) + cnt(it->r) + 1;
}

int c (pitem it, int p){
   return it ? it -> c[p] : 0;
}

void upd_c (pitem it){
    if (it){

       for (int i = 0; i < 27; ++i)
           it -> c[i] = c (it -> l, i) + c (it -> r, i);
       it -> c[it -> value] += 1;
    }
}

void push (pitem it){
    if (it && it->rev){
       it->rev = false;
       swap (it->l, it->r);
       if (it->l) it->l->rev ^= true;
       if (it->r) it->r->rev ^= true;
    }

    if (it && it -> mar){
       it -> mar = false;

       if (it->l) {
          it->l->value = it->value;
          it->l->mar = true;
}
/*
2
CTCGC
TGCGG
5
MUTATE 1 2 A
COUNT 2 2 4
MUTATE 2 1 G
CROSS 2 1 1 5
COUNT 4 3 6
*/
```

```
            memset (it->l->c, 0, sizeof (it->l->c));
            upd_cnt (it->l);
            it->l->c[it->l->value] = it->l->cnt;
        }

        if (it->r){
            it->r->value = it->value;
            it->r->mar = true;

            memset (it->r->c, 0, sizeof (it->r->c));
            upd_cnt (it->r);
            it->r->c[it->r->value] = it->r->cnt;
        }
    }
}

void merge (pitem & t, pitem l, pitem r) {
    push (l);
    push (r);
    if (!l || !r)
        t = l ? l : r;
    else if (l->prior > r->prior)
           merge (l->r, l->r, r), t = l;
         else
           merge (r->l, l, r->l), t = r;
    upd_cnt (t);
    upd_c (t);
}

void split (pitem t, pitem & l, pitem & r, int key, int add = 0) {
    if (!t)
    return void( l = r = 0 );
    push (t);
    int cur_key = add + cnt(t->l);
    if (key <= cur_key)
        split (t->l, l, t->l, key, add), r = t;
   else
        split (t->r, t->r, r, key, add + 1 + cnt(t->l)), l = t;
    upd_cnt (t);
    upd_c (t);
}

void reverse (pitem t, int l, int r) {
    pitem t1, t2, t3;
    split (t, t1, t2, l-1);
    split (t2, t2, t3, r-l+1);
    t2->rev ^= true;
    merge (t, t1, t2);
    merge (t, t, t3);
}

void output (pitem t) {
    if (!t) return;
    push (t);
    output (t->l);
    printf ("%d ", t->value);
    output (t->r);
}

const int MAX = 1e5 + 15;

char a[MAX], car[5];
int p1, p2, q;

pitem root = 0;

int f (char x){
   return x-'a';
}

int main(){

  // srand (time (0));

   scanf ("%s", a);
   int l = strlen (a);
   for (int i = 0; i < l; ++i){
       pitem x = new item();
       x -> value = f (a[i]);
       merge (root, root, x);
   }

   scanf ("%d", &q);
   while (q--){
        scanf ("%s", car);
        pitem x = 0, y = 0;
        if (car[0] == 'C'){
           scanf ("%d %d %s", &p1, &p2, car);
           int id = f (car[0]);
           split (root, root, x, p1-1);
           split (x, x, y, p2-p1+1);

           printf ("%d\n", c (x, id));
```

```
            merge (root, root, x);
            merge (root, root, y);
         }
         else
            if (car[0] == 'R'){
               scanf ("%d %d", &p1, &p2);
               reverse (root, p1, p2);
            }
             else{
               scanf ("%d%d%s", &p1, &p2, car);

               split (root, root, x, p1-1);
               split (x, x, y, p2-p1+1);
               x -> value = f (car[0]);
               x -> mar = true;

               merge (root, root, x);
               merge (root, root, y);
             }

         //output(root);printf ("\n----------\n");
      }

      return 0;
}
```

2.9. Treap (Explicit)*.

```
// Treap (Explicit)
// testeado https://www.spoj.com/problems/ORDERSET/
#include <bits/stdc++.h>
using namespace std;

typedef struct node{
  int val,prior,size;
  struct node *l,*r;
  node(int val,int prior,int size=1)
    : val(val), prior(prior), size(size), l(nullptr), r(nullptr){}
}node;
typedef node* pnode;

pnode root=nullptr;

int sz(pnode t){return t?t->size:0;}
void upd_sz(pnode t){if(t) t->size = sz(t->l)+1+sz(t->r);}

void split(pnode t,pnode &l,pnode &r,int key){
  if(!t) l=r=NULL;
  else if(t->val<=key) split(t->r,t->r,r,key),l=t; //elem=key is in l
  else split(t->l,l,t->l,key),r=t;
  upd_sz(t);
}
void merge(pnode &t,pnode l,pnode r){
  if(!l || !r) t=l?l:r;
  else if(l->prior > r->prior)merge(l->r,l->r,r),t=l;
  else merge(r->l,l,r->l),t=r;
  upd_sz(t);
}
void insert(pnode &t,pnode it){
  if(!t) t=it;
  else if(it->prior>t->prior) split(t,it->l,it->r,it->val),t=it;
  else insert(t->val<=it->val?t->r:t->l,it);
  upd_sz(t);
}
void erase(pnode &t,int key){
  if(!t)return;
  else if(t->val==key){pnode temp=t;merge(t,t->l,t->r);free(temp);}
  else erase(t->val<key?t->r:t->l,key);
  upd_sz(t);
}
pnode init(int val){
  pnode ret = (pnode)malloc(sizeof(node));
  ret->val=val;ret->size=1;ret->prior=rand();ret->l=ret->r=NULL;
  return ret;
}

/* Cuenta cantidad de elementos con valor <= key
 */
int count(pnode &r, int key)
{
  //~ pnode x=nullptr;
  //~ split(root, root, x, key-1);
  //~ int rs = sz(root);
  //~ merge(root,root,x);
  //~ return rs;
  if(r == nullptr) return 0;
```

```
 if(r -> val <= key) return sz(r->l) + 1 + count(r->r,key);
 return count(r->l,key);
}
/* Encuentra el kth elemento
 */
int kth(pnode &r, int k, int des=0)
{
 int ndes = des + sz(r->l) + 1;
 if(ndes == k) return r->val;
 if(ndes < k) return kth(r->r, k,ndes);
 else return kth(r->l, k, des);
}

int main()
{
 ios_base::sync_with_stdio(false);
 cin.tie(0);cout.tie(0);

 srand(time(nullptr));

 int q;cin >> q;
 while(q--)
 {
  char op;
  int x;
  cin >> op >> x;
  switch(op)
  {
   case 'I':
    erase(root, x);
    insert(root,init(x));
    break;
   case 'D':erase(root,x);break;
   case 'C':cout << count(root,x-1) << '\n';break;
   case 'K':
    if(sz(root) < x) cout << "invalid\n";
    else cout << kth(root,x) << '\n';
    break;
  }
 }

 return 0;
}
```

2.10. Trie.

```
// testeado en CODF476E
struct trie {
 bool mk;
 trie* next[0x100];
 trie() { fill(next, next+0x100, nullptr);mk=false;}
};

trie* wadd(string& s, trie* r)
{
 for(char c: s)
 {
  if(r->next[c] == nullptr)
   r->next[c] = new trie;
  r = r->next[c];
 }
 r->mk=true;
 return r;
}

trie* root=new trie;
```

2.11. Trie*.

```
//¯ 476E - codeforces
/* Replace each string with its non-empty
 * prefix, s.t. the new strings are still
 * unique and minimizes the total length.
 *
 * Note: solution uses a smaller to large
 * technique.
 */
#pragma comment(linker, "/stack:200000000")
#pragma GCC optimize("Ofast")
#pragma GCC optimize("unroll-loops")
#pragma GCC target("sse,sse2,sse3,ssse3,sse4,popcnt,abm,mmx,avx,tune=native")

#include <bits/stdc++.h>
```

```cpp
using namespace std;

typedef long long int ll;
typedef long double ld;
typedef pair<int,int> pii;
typedef pair<ll,ll> pll;

#define INIT ios_base::sync_with_stdio(false);\
         cin.tie(0),cout.tie();
#define endl '\n'
#define fr first
#define sc second
#define pb push_back
#define eb emplace_back
#define mp make_pair
#define ins insert
#define ers erase
#define sz(c) (c).size()
#define all(x) (x).begin(),(x).end()
#define unique(x) (x).resize(unique(all(x))-(x).begin())

struct trie {
 bool mk;
 trie* next[0x100];
 trie() { fill(next, next+0x100, nullptr);mk=false;}
};

trie* wadd(string& s, trie* r)
{
 for(char c: s)
 {
  if(r->next[c] == nullptr)
    r->next[c] = new trie;
  r = r->next[c];
 }
 r->mk=true;
 return r;
}
trie* root=new trie;

vector<multiset<int>> ss;
int js(int u,int v)
{
 if(u==v)return u;
 if(sz(ss[u]) < sz(ss[v]))
  swap(u,v);
 ss[u].ins(all(ss[v]));
 return u;
}
int dfs(trie* r,int len=0)
{
 int my=ss.size();
 ss.pb(multiset<int>());

 for(char c='a';c<='z';++c)
 {
  if(r->next[c]==nullptr)continue;
  int nn=dfs(r->next[c],len+1);
  my = js(my,nn);
 }

 if(!r->mk)
 {
  if(!ss[my].empty())
  {
   auto tmp = ss[my].end();--tmp;
   ss[my].ers(tmp);
   ss[my].ins(len);
  }
 }else ss[my].ins(len);

 return my;
}

int main()
{
 INIT
 int n;cin>>n;
 for(int i=1;i<=n;++i)
 {
  string s;
  cin>>s;
  wadd(s,root);
 }

 root->mk=true;
 int id = dfs(root);

 int ans=0;
 for(int x: ss[id])
  ans+=x;

 cout<<ans<<endl;
 return 0;
}
```

2.12. Trie MinMax Xor.

```
// testeado en http://codeforces.com/contest/948/problem/D

const int N = 3e5+7; // #of words
const int L = 32; // length of word
const int S = 2; // size of alphabet
int t[N*L][S], c[N*L], sz=0;

struct trie_xor_min_max
{
 #define MIN_XOR // comment for max
 void add(int v)
 {
  int u=0;
  for(int i=30;~i;--i)
  {
   ++c[u];
   int b = (v>>i) & 1;
   if(!t[u][b])
    t[u][b] = ++sz;
   u=t[u][b];
  }
  ++c[u];
 }
 void rem(int v)
 {
  int u=0;
  for(int i=30;~i;--i)
  {
   --c[u];
   int b = (v>>i) & 1;
   u=t[u][b];
  }
  --c[u];
 }
 pii qry(int v)
 {
  int u=0,xorn=0,val=0;
  for(int i=30;~i;--i)
  {
   int b = (v>>i) & 1;
   #ifdef MIN_XOR
   if(t[u][b] && c[t[u][b]])
   {
    //~ cout<<" same "<<i<<' '<<b<<endl;
    val|=b*(1<<i);
    u=t[u][b];
   }else
   {
    //~ cout<<" diff "<<i<<' '<<(b^1)<<endl;
    val|=(b^1)*(1<<i);
    xorn|=(1<<i);
    u=t[u][b^1];
   }
   #endif
   #ifndef MIN_XOR
   if(t[u][b^1] && c[t[u][b^1]])
   {
    val|=(b^1)*(1<<i);
    xorn|=(1<<i);
    u=t[u][b^1];
   }else
   {
    val|=b*(1<<i);
    u=t[u][b];
   }
   #endif
  }
  return mp(xorn,val);
 }
};
```

2.13. Trie MinMax Xor*.

```
//~ 470D - codeforces
/* Minimize Xor
 * For given A and P, find the lexicographically
 * smallest message O, for which there exists a
 * permutation pi such that (Oi xor pi(Pi) = Ai) for every i.
 */
#pragma comment(linker, "/stack:200000000")
#pragma GCC optimize("Ofast")
#pragma GCC optimize("unroll-loops")
#pragma GCC \
```

```
target("sse,sse2,sse3,ssse3,sse4,popcnt,abm,mmx,avx,tune=native")

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

typedef long long int ll;
typedef long double ld;
typedef pair<int,int> pii;
typedef pair<ll,ll> pll;

#define INIT ios_base::sync_with_stdio(false);\
          cin.tie(0),cout.tie();
#define endl '\n'
#define fr first
#define sc second
#define pb push_back
#define eb emplace_back
#define mp make_pair
#define ins insert
#define ers erase
#define all(x) (x).begin(),(x).end()
#define unique(x) (x).resize(unique(all(x))-(x).begin())

const int N = 3e5+7; // #of words
const int L = 32; // length of word
const int S = 2; // size of alphabet
int t[N*L][S], c[N*L], sz=0;

struct xorm
{
 #define MIN_XOR // comment for max
 void add(int v)
 {
  int u=0;
  for(int i=30;~i;--i)
  {
   ++c[u];
   int b = (v>>i) & 1;
   if(!t[u][b])
    t[u][b] = ++sz;
   u=t[u][b];
  }
  ++c[u];
 }
 void rem(int v)
 {
  int u=0;
  for(int i=30;~i;--i)
  {
   --c[u];
   int b = (v>>i) & 1;
   u=t[u][b];
  }
  --c[u];
 }
 pii qry(int v)
 {
  int u=0,xorn=0,val=0;
  for(int i=30;~i;--i)
  {
   int b = (v>>i) & 1;
   #ifdef MIN_XOR
   if(t[u][b] && c[t[u][b]])
   {
    //~ cout<<" same "<<i<<' '<<b<<endl;
    val|=b*(1<<i);
    u=t[u][b];
   }else
   {
    //~ cout<<" diff "<<i<<' '<<(b^1)<<endl;
    val|=(b^1)*(1<<i);
    xorn|=(1<<i);
    u=t[u][b^1];
   }
   #endif
   #ifndef MIN_XOR
   if(t[u][b^1] && c[t[u][b^1]])
   {
    val|=(b^1)*(1<<i);
    xorn|=(1<<i);
    u=t[u][b^1];
   }else
   {
    val|=b*(1<<i);
    u=t[u][b];
   }
   #endif
  }
  return mp(xorn,val);
 }
}X;

int n;
int a[N],p[N];
```

```
int main()
{
 INIT
 cin>>n;
 for(int i=1;i<=n;++i)
 {
  cin>>a[i];
 }

 for(int i=1;i<=n;++i)
 {
  int p;cin>>p;
  X.add(p);
 }
 //~ while(true)
 //~ {
  //~ int p;cin>>p;
  //~ if(p==-1)break;
  //~ cout<<X.qry(p).fr<<endl;
 //~ }

 for(int i=1;i<=n;++i)
 {
  pii best=X.qry(a[i]);
  cout<<best.fr<<' ';
  X.rem(best.sc);
 }

 return 0;
}
```

2.14. Union Find (+rollback).

```
//~ testado en http://codeforces.com/contest/892/problem/E
struct UF
{
	int n;
	vector<int> root, r;
	vector<PII> ops;
	UF(int n)
	:root(n+1),r(n+1,1)
	{
		for(int i=1;i<=n;++i)root[i]=i;
	}

	int fs(int u)
	{
		while(root[u]!=u)u=root[u];
		return u;
	}
	bool js(int u,int v)
	{
		u=fs(u),v=fs(v);
		if(u==v)return false;
		if(r[u]<r[v])
			swap(u,v);

		ops.emplace_back(u,v);

		r[u]+=r[v];
		root[v]=u;
		return true;
	}
	void rb(int steps)
	{ // rollback technique
		while(steps--)
		{
			PII cur=ops.back();

			r[cur.fr]-=r[cur.sc];
			root[cur.sc]=cur.sc;

			ops.pop_back();
		}
	}
};
```

2.15. Union Find (+rollback)*.

```
//¯ 446E - codeforces (Union-Find + Rollback)
/* You are given a graph G. You are given
 * some queries, each query contains a set
 * of edges of graph G, and you should
 * determine whether there is a MST
 * containing all these edges or not.
 */
#include <bits/stdc++.h>
using namespace std;

#define fr first
#define sc second

struct UF
{
	int n;
	vector<int> root, r;
	vector<pair<int&,int>> ops;
	UF(int n)
	:root(n+1),r(n+1,1)
	{
		for(int i=1;i<=n;++i)root[i]=i;
	}

	int fs(int u)
	{
		while(root[u]!=u)u=root[u];
		return u;
	}
	void modify(int &a,int b)
	{
		ops.emplace_back(a,a);
		a=b;
	}
	bool js(int u,int v)
	{
		u=fs(u),v=fs(v);
		if(u==v)return false;
		if(r[u]<r[v])
			swap(u,v);

//  ops.emplace_back(u,v);

		modify(root[v],u);
		modify(r[u],r[u]+r[v]);
/*
		r[u]+=r[v];
		root[v]=u;
*/
		return true;
	}
	void rb(int bot)
	{ // rollback technique
		while(ops.size() > bot)
		{
			ops.back().fr = ops.back().sc;
			ops.pop_back();
/*
			PII cur=ops.back();

			r[cur.fr]-=r[cur.sc];
			root[cur.sc]=cur.sc;

			ops.pop_back();
*/
		}
	}
};

const int N = 5e5+7;

struct query
{
	int u,v,w,id;
	query(int u,int v,int w,int id)
	:u(u),v(v),w(w),id(id){}
	bool operator<(const query &o)const
	{
		if(w!=o.w)return w<o.w;
		return id<o.id;
	}
};

int n,m;
vector<query> qe;

int main()
{
	scanf("%d%d", &n, &m);
	for(int i=1;i<=m;++i)
```

```
	{
		int u,v,w;
		scanf("%d%d%d", &u, &v, &w);
		qe.emplace_back(query(u,v,w,N));
	}

	int q;cin >> q;
	for(int j=1;j<=q;++j)
	{
		int k; scanf("%d", &k);
		for(int i=1;i<=k;++i)
		{
			int p;scanf("%d", &p);
			qe.emplace_back(query(qe[p-1].u,qe[p-1].v,qe[p-1].w,j));
		}
	}

	sort(qe.begin(),qe.end());
/*
	for(auto x: qe)
		cerr << x.w << ' ' << x.id << ' ' << x.u << ' ' << x.v << endl;
*/
	UF uf(n);
	vector<int> ans(q+1,1);
	for(int i=0,j;i<qe.size();i=j)
	{
		j=i;
		int cq=qe[j].id;

//  cerr << " A " << j << endl;

		if(cq == N)
		{
			do
			{
//  cerr << " bad " << j << '\n';
				uf.js(qe[j].u,qe[j].v);
				++j;
			}while(j < qe.size() && qe[j].w == qe[j-1].w
	&& qe[j].id == qe[j-1].id);
		}else
		{
			int sback=uf.ops.size();
			bool flag=true;
			do
			{
				if(!flag){++j;continue;}
//  cerr<<" good " << j << '\n';
				if(!uf.js(qe[j].u,qe[j].v))
				{
					ans[qe[j].id] &= 0;
					flag=false;
					++j;
					continue;
				}
				++j;
			}while(j < qe.size() && qe[j].w == qe[j-1].w
	&& qe[j].id == qe[j-1].id);
			uf.rb(sback);
		}
	}

	for(int i=1;i<=q;++i)
		if(ans[i]) printf("YES\n");
		else printf("NO\n");

	return 0;
}
```

2.16. **Hashed String.**

```
// testeado en http://matcomgrader.com/problem/9532/contando-personas/
typedef unsigned long long ull;
#define BASE 29
const ull Mod=1e9+7;
ull powe[2050];
void init()
{
 powe[0]=1;
 for(int i=1;i<=1010;i++)powe[i]=(powe[i-1]*BASE)%Mod;
}
int val(char x){return x-'a'+1;}
struct HashedString
{
 string str;
 vector<ull> hash;
 HashedString(){}
 HashedString(string &str)
  :str(str), hash(str.size())
```

```
 {
  hash[0] = val(str[0]);
  for(int i=1;i<(int)hash.size();++i)
   hash[i] = (hash[i-1]*BASE%Mod + val(str[i]))%Mod;
 }
 ull hashCode(){return hash.back();}
 ull get_hash(int i,int j)
 {
  if(i > 0) return (hash[j]+Mod - hash[i-1]*powe[j-i+1]%Mod)%Mod;
  return hash[j];
 }
};
```

2.17. HLD+SegmentTreeMin*.

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

typedef long long int ll;
typedef long double ld;
typedef pair<int,int> pii;
typedef pair<ll,ll> pll;

#define INIT ios_base::sync_with_stdio(false);\
         cin.tie(0),cout.tie()
#define endl '\n'
#define fr first
#define sc second
#define pb push_back
#define eb emplace_back
#define mp make_pair
#define lb lower_bound
#define ub upper_bound
#define ins insert
#define ers erase
#define sz(c) ((int)(c).size())
#define all(x) (x).begin(),(x).end()
#define unique(x) (x).resize(unique(all(x))-(x).begin())
#define debug(_fmt,...) \
fprintf(stderr,"("#__VA_ARGS__ ")␣=␣(" _fmt")\n",__VA_ARGS__)

template<typename T>
void printVector(vector<T> &v)
{
 for(T &x: v)cerr << x << '␣';
 cerr << endl;
}
// testeado en http://www.codeforces.com/contest/1023/problem/F
struct SegmentTreeMin
{
 int n,nolazy;
 vector<int> data, tree, lazy;
 SegmentTreeMin(vector<int>& data, int nolazy)
  : n(sz(data)), nolazy(nolazy), data(data), tree(4*(n+1)),
     lazy(4*(n+1), nolazy){}

 #define lf(i) ((i) << 1)
 #define rg(i) (((i) << 1) | 1)

 void build(int i=1,int lo=1,int hi=-1)
 {
  if(hi==-1)hi=n;
  lazy[i] = nolazy;
  if(lo == hi){tree[i] = data[lo-1];return;}
  int m = (lo+hi) >> 1;build(lf(i), lo, m);build(rg(i), m+1,hi);
  tree[i] = min(tree[lf(i)], tree[rg(i)]);
 }

 void prop(int i,int lo,int hi)
 {
  tree[i] = min(lazy[i],tree[i]);
  if(lo != hi)
   lazy[lf(i)] = min(lazy[lf(i)], lazy[i]),
   lazy[rg(i)] = min(lazy[rg(i)], lazy[i]);
  lazy[i] = nolazy;
 }

 void update(int x,int y,int val,int i=1,int lo=1,int hi=-1)
 {
  if(hi==-1)hi=n;
  prop(i,lo,hi);
  if(x > hi || y < lo) return;
  if(x <= lo && hi <= y){lazy[i] = min(val, lazy[i]);return;}
  int m=(lo+hi) >> 1;
  update(x,y,val,lf(i),lo,m);update(x,y,val,rg(i),m+1,hi);
  tree[i] = min(tree[lf(i)], tree[rg(i)]);
 }
```

```
  int query(int x,int y,int i=1,int lo=1,int hi=-1)
  {
   if(hi==-1)hi=n;
   prop(i,lo,hi);
   if(x > hi || y < lo) return nolazy;
   if(x <= lo && hi <= y)return tree[i];
   int m=(lo+hi) >> 1;
   return min( query(x,y,lf(i),lo,m),query(x,y,rg(i),m+1,hi) );
  }
};

// testeado en http://www.codeforces.com/contest/1023/problem/F
typedef vector<vector<int>> graph;
struct HLD
{
 int n;
 graph g; // 0 - indexed
 vector<vector<int>> paths;
 vector<int> par,nxt,depth,ppos,pid;
 vector<SegmentTreeMin> ST;

 HLD(graph &g, vector<int> &d)
  : n(sz(g)),g(g),par(n),nxt(n),depth(n),ppos(n),pid(n)
 {
  dfs(); // si g es un bosque, correr dfs en cada arbol
  for(int i=0;i<n;++i)
   if(par[i]==-1||nxt[par[i]]!=i)
   {
    paths.eb();
    vector<int> data;
    for(int j=i;~j;j=nxt[j])
    {
     paths.back().pb(j);
     ppos[j] = sz(paths.back()) - 1;
     pid[j] = sz(paths) - 1;
     data.pb(d[j]);
    }
    ST.eb(data,2e9);
    ST.back().build();
   }
 }

 int dfs(int u=0,int p=-1)
 {
  par[u] = p;
  int sz=1,mxsz=-1;
  nxt[u]=-1;
  for(int v: g[u])
  {
   if(v == p) continue;
   depth[v] = depth[u]+1;
   int tmp = dfs(v,u);
   if(tmp > mxsz)
   {
    nxt[u] = v;
    mxsz = tmp;
   }
   sz+=tmp;
  }
  return sz;
 }

 inline int root(int u)
 {return paths[pid[u]].front();}
 void update(int u,int v,int val)
 {
  while(pid[u] != pid[v])
  {
   if(depth[root(u)] < depth[root(v)])swap(u,v);
   ST[pid[u]].update(1,ppos[u]+1,val);
   u=par[root(u)];
  }
  if(depth[u] < depth[v])swap(u,v);
  v=nxt[v];
  ST[pid[u]].update(ppos[v]+1,ppos[u]+1,val);
 }

 int query(int u,int v)
 {
  int ret = 2e9;
  if(u == v) return ret; // no such path from u to v
  while(pid[u] != pid[v])
  {
   if(depth[root(u)] < depth[root(v)])swap(u,v);
   ret=min(ret, ST[pid[u]].query(1,ppos[u]+1));
   u=par[root(u)];
  }
  if(depth[u] < depth[v])swap(u,v);
  v=nxt[v];
  ret = min(ret, ST[pid[u]].query(ppos[v]+1,ppos[u]+1));
  return ret;
 }
};
```

```
void rec(vector<vector<pii>> &g, vector<int> &d, int u=0,int p=-1)
{
 for(pii &to: g[u])
 {
  if(to.sc == p) continue;
  d[to.sc] = to.fr;
  rec(g,d,to.sc,u);
 }
}

int main()
{
 #ifdef OJUDGE
  //~ freopen("in","r",stdin);
 #endif
 //~ INIT;

 int n;cin >> n;
 graph g(n);
 vector<vector<pii>> g1(n);
 for(int i=1;i<n;++i)
 {
  int u,v,c;
  cin >> u >> v >> c;
  --u,--v;
  g[u].pb(v);
  g[v].pb(u);

  g1[u].eb(c,v);
  g1[v].eb(c,u);
 }
 vector<int> d(n,2e9);
 rec(g1,d);

 HLD H(g,d);

 while(1)
 {
  int op;
  cin >> op;
  if(op == -1) break;
  if(op == 1)
  { // update
   int u,v,c;
   cin >> u >> v >> c;
   --u,--v;
   H.update(u,v,c);
  }else
  {
   int u,v;
   cin >> u >> v;
   --u,--v;
   cout << H.query(u,v) << endl;
  }
 }

 return 0;
}
/*
5
1 2 10
1 3 10
2 4 2
4 5 10

5
1 2 3
2 5 4
2 3 2
3 4 1
*/
```

2.18. **Order Statistics.**

```
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;

typedef tree<
pair<int, int>,
null_type,
less<pair<int, int>>,
rb_tree_tag,
tree_order_statistics_node_update> ordered_set;

ordered_set X; //multiset, usar con pair, primer valor
          // es el valor real, el segundo es un contador cualquiera
```

```
//los siguientes ejemplos es usandolo sin el par
X.insert(1);
X.insert(2);
X.insert(4);
X.insert(8);
X.insert(16);

//find_by_order regresa iterador al kth elemento (contando desde 0)
//order_of_key regresa cantidad de elementos menores estrictos que el
//dado estas operaciones son el O(log n) tambien soporta las
//operaciones de set normal como lower/upper bound, insert,
//erase, find, etc

//ejemplos

cout<<*X.find_by_order(1)<<endl; // 2
cout<<*X.find_by_order(2)<<endl; // 4
cout<<*X.find_by_order(4)<<endl; // 16
cout<<(end(X)==X.find_by_order(6))<<endl; // true

cout<<X.order_of_key(-5)<<endl; // 0
cout<<X.order_of_key(1)<<endl; // 0
cout<<X.order_of_key(3)<<endl; // 2
cout<<X.order_of_key(4)<<endl; // 2
cout<<X.order_of_key(400)<<endl; // 5
```

2.19. RMQ*.

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

int n, p2, p1, q;
/*
RANGE Min-Max QUERING
testeado en https://www.spoj.com/problems/RMQSQ/
*/
const int MAX = 1e5 + 5;//tamanno maximo

namespace RMQ{

     int mat[MAX][20];
     int l[MAX];
     int n;

     void build (vector <int> &v){

         n = (int)v.size();
         for (int i = 1; i <= n; ++i)
             l[i] = log2 (i);

         for (int i = 0; i < n; ++i)
             mat[i+1][0] = v[i];

         int p = n, a;
         for (int i = 1; i <= l[n]; ++i){
             a = 1 << (i-1);
             p -= a;
             for (int j = 1; j <= p; ++j)
                 mat[j][i] = min(mat[j][i-1], mat[j+a][i-1]);
             }
         }

         int find (int p1 , int p2){
            if (p1 > p2) swap (p1, p2);
            int c = l[p2-p1];
            return min (mat[p1][c], mat[p2-(1<<c)+1][c]);
         }
};

int main(){
   scanf("%d", &n);
   vector <int> v;
   for(int i = 1; i <= n; i++){
      int x;scanf("%d", &x);
      v.push_back (x);
   }

   RMQ::build(v);

   scanf ("%d", &q);
   while (q--){
        scanf ("%d %d", &p1, &p2);p1++, p2++;
        printf ("%d\n", RMQ::find (p1, p2));
   }

   return 0;
}
```

3. String

3.1. KMP (Generic).

```
// testeado coj - 2440, 2250,
// https://codeforces.com/contest/1017/problem/E
#include <bits/stdc++.h>
using namespace std;

#define pb push_back
#define eb emplace_back
#define sz(x) ((int)(x).size())
#define all(x) (x).begin(),(x).end()

template<class C>
struct KMP
{
 C p;
 vector<int> fail; // fail[j] = mayor i tq S[0..i] == S[j-i..j]
              // o -1 si no existe tal prefijo
 KMP(C &p):p(p),fail(sz(p))
 {
  for(int i=0,j=-2;i<sz(p);fail[i] = ++j, ++i)
   while(j>-2&&p[j+1]!=p[i])if(j==-1)j=-2;else j=fail[j];
 }
 vector<int> all_match(C &text, bool overlap=true)
 {
  vector<int> res;
  for(int i=0,j=-1;i<sz(text);++i)
  {
   while(j>-2&&p[j+1]!=text[i])if(j==-1)j=-2;else j=fail[j];
   if(++j==sz(p)-1)res.pb(i-j),j=overlap?fail[j]:-1;
  }
  return res;
 }
 bool match(C &text)
 {
  //~ return sz(all_match(text)) > 0;
  for(int i=0,j=-1;i<sz(text);++i)
  {
   while(j>-2&&p[j+1]!=text[i])if(j==-1)j=-2;else j=fail[j];
   if(++j==sz(p)-1)return true;
  }
  return false;
 }
};

int main()
{
 vector<int> p;
 int n,ol;cin >> n;
 while(n--){int x;cin >> x;p.pb(x);}
 KMP<vector<int>> kmp(p);

 while(1)
 {
  cin >> n >> ol;
  p.clear();
  while(n--){int x;cin >> x;p.pb(x);}
  auto ans = kmp.all_match(p,ol);
  for(int i: ans) cout << i << '␣';
  cout << endl;
 }
 return 0;
}
```

3.2. Aho-Corasick (punteros).

```
// Aho Corasick

struct node{
   int pos;
   node* fail;
   node* link;
   node* next[26];
   node(){
      pos = -1;fail = link = NULL;
      for (int i = 0; i < 26; i++) next[ i ] = NULL;
   }
};
node* root = new node();

void insert(char* patt, int idx){
   node* curr=root;
   for (int j=0;patt[j];j++){
      if (curr->next[patt[j] - 'a'] == NULL)
```

```
         curr->next[patt[j] - 'a'] = new node();
      curr = curr->next[patt[j] - 'a'];
   }
   curr->pos = idx;
}

void aho_corasick(){
   queue<node*> Q;
   for (int i = 0;i<26;i++)
      if ( root->next[i] ){
         root->next[i]->fail = root;
         Q.push( root->next[i] );
      }else root->next[i] = root;
   while ( !Q.empty() ){
      node* t = Q.front(); Q.pop();
      for (int i = 0; i < 26; i++)
         if ( t->next[i] ){
            Q.push( t->next[i] );
            node* r = t->fail;
            while ( !r->next[i] ) r = r->fail;
            t->next[i]->fail = r->next[i];
            if ( r->next[i]->pos != -1 )
              t->next[i]->link = r->next[i];
            else /////multiple matches in the same node/////
               t->next[i]->link = r->next[i]->link;
         }
   }
}

void match(char text[]){

   n = strlen( text );
   node* state = root;
   for (int i = 0; i < n; i++){
      while (state->next[ text[i]-'a' ] == NULL)
         state = state->fail;
      state = state->next[ text[i]-'a' ];

      ////////////////////
      if (state->pos != -1)
      cout<< state->pos<<"␣found␣at␣"<< i << endl;

      for (node* r = state->link; r != NULL; r = r->link)
         cout<< r->pos<<"␣found␣in␣position␣"<< i << endl;
      ////////////////////
   }
}
```

3.3. Aho-Corasick (sin punteros).

```
struct aho_corasick{
 int num;
 int pos[MAX];
 int fail[MAX];
 int link[MAX];
 int next[MAX][26];
 aho_corasick(){
    clean();
 }
 void insert(char* patt, int idx){
  int curr = 0;
  for (int j=0; patt[j]; j++){
   int a = patt[j] - 'a';
   if (next[curr][a] == 0){
    next[curr][a] = num++;
    for(int i = 0; i < 26; i++)next[num - 1][i] = 0;
    pos[num - 1] = fail[num - 1] = 0;
    link[num - 1] = -1;
   }
    curr = next[curr][a];
  }
  pos[curr] = idx;
 }
 void construct(){
  queue<int> Q;
  for (int i = 0;i < 26;i++)
   if ( next[0][i] ){
    fail[next[0][i]] = 0;
    Q.push( next[0][i] );
   }else next[0][i] = 0;
  while ( !Q.empty() ){
   int t = Q.front(); Q.pop();
   for (int i = 0; i < 26; i++){
    int v = next[t][i];
    if ( v ) {
     Q.push( v );
     int u = fail[t];
     while ( u && !next[u][i] ) u = fail[u];
```

```
      fail[ v ] = next[u][i];
      if ( pos[next[u][i]] != -1 ) link[next[t][i]] = next[u][i];
      else link[next[t][i]] = link[next[u][i]];
     }
    }
   }
  }
```

3.4. Aho-Corasick* -1.

```
#include <bits/stdc++.h>

using namespace std;


const int MAX = 3e3 + 5;

int n, l[MAX];
bool m[MAX];
char a[MAX];

vector <string> v;
#include <bits/stdc++.h>
using namespace std;

typedef long long int ll;
typedef long double ld;
typedef pair<int,int> pii;
typedef pair<ll,ll> pll;

#define INIT ios_base::sync_with_stdio(false);\
         cin.tie(0),cout.tie()
#define endl '\n'
#define fr first
#define sc second
#define pb push_back
#define eb emplace_back
#define mp make_pair
#define lb lower_bound
#define ub upper_bound
#define ins insert
#define ers erase
#define sz(c) ((int)(c).size())
#define all(x) (x).begin(),(x).end()
#define unique(x) (x).resize(unique(all(x))-(x).begin())
#define debug(_fmt,...) \
fprintf(stderr,"("#__VA_ARGS__ ") = (" _fmt")\n",__VA_ARGS__)

const int S = 26;
namespace AhoCorasick
{
 struct node
 {
  typedef struct node* pnode;
  vector<pnode> next;
  pnode fail;
  int output;
  pnode sigt;
  node() : next(S,nullptr), fail(nullptr),
   output(-1), sigt(nullptr){}
 };
 typedef struct node* pnode;
 pnode root;
 vector<string> K; // keywords

 void enter(string &a,int i)
 {
  l[i] = (int)a.size();
  pnode s = root;
  int j=0;
  while(j < sz(a) && s->next[a[j]-'a'] != nullptr)
   s=s->next[a[j++]-'a'];
  for(;j<sz(a);++j)
  {
   s->next[a[j]-'a'] = new node;
   s = s->next[a[j]-'a'];
  }
  s->output = i;
 }

 void build_failure()
 {
  void clean(){
   num = 1;
   for(int i = 0; i < 26; i++)next[num - 1][i] = 0;
   pos[num - 1] = fail[num - 1] = 0;
   link[num - 1] = -1;
  }
};
```

```
  queue<pnode> q;
  for(int a=0;a<S;++a)
  {
   if(root->next[a] != root)
   {
    root->next[a]->fail = root;
    q.push(root->next[a]);
   }
  }
  while(sz(q))
  {
   pnode r = q.front();q.pop();
   for(int a=0;a<S;++a)
    if(r->next[a] != nullptr)
    {
     pnode s = r->next[a];
     q.push(s);

     pnode state = r->fail;
     while(state->next[a]==nullptr)state = state->fail;
     s->fail = state->next[a];

     if(~s->fail->output) s->sigt = s->fail;
     else if(s->fail->sigt != nullptr) s->sigt = s->fail->sigt;
    }else r->next[a] = r->fail->next[a];
  }
}

void match(string &a)
{
  pnode s = root;
  for(int i=0;i<sz(a);++i)
  {
   s = s->next[a[i]-'a'];
   if(~s->output && m[i-l[s->output]+1]
    && l[s->output] != sz(a)){m[i+1] = 1; continue;}
   pnode cur = s->sigt;
   while(cur!=nullptr)
   {
    if(~cur->output && m[i-l[cur->output]+1]
     && l[cur->output] != sz(a)){m[i+1] = 1; break;}
    cur = cur->sigt;
   }
  }
}

//~ AhoCorasick(){}
```

```
 void init(vector<string> &v)
 {
  K = v;
  root = new node;
  for(int i=0;i<sz(K);++i)
   enter(K[i], i);
  for(int a=0;a<S;++a)
   if(root->next[a] == nullptr) root->next[a]=root;
  build_failure();
 }
}

set <string> ss;

int main(){

   scanf ("%d", &n);int sol = n;
   for (int i = 1; i <= n; ++i){
       scanf ("%s", a);
       string s = (string) a;
       if (ss.find (s) != ss.end()){
          sol--;
          continue;
       }

       ss.insert (s);

       v.push_back (s);
   }

   AhoCorasick::init (v);
   sort (v.begin(), v.end());
   for (int i = 0; i < (int)v.size(); i++){
       string s = v[i]; //printf ("%s\n", s.c_str());
       memset (m, 0, sizeof (m));
       m[0] = 1;
       AhoCorasick::match (s);
       if (m[(int)s.size()]) sol--;
      // for (int j = 0; j <= (int)s.size(); ++j)
      // printf ("%d", m[j]);printf ("\n");
   }

   printf ("%d\n", sol);

   return 0;
}
```

3.5. Aho-Corasick* -2.

```
//~ Aho-Corasick
// el menor texto en orden lexicografico que
// tiene L letras y con cantidad total de
// ocurrencias igual a K
#include <bits/stdc++.h>
using namespace std;

typedef long long int ll;
typedef long double ld;
typedef pair<int,int> pii;
typedef pair<ll,ll> pll;

#define INIT ios_base::sync_with_stdio(false);\
          cin.tie(0),cout.tie()
#define endl '\n'
#define fr first
#define sc second
#define pb push_back
#define eb emplace_back
#define mp make_pair
#define lb lower_bound
#define ub upper_bound
#define ins insert
#define ers erase
#define sz(c) ((int)(c).size())
#define all(x) (x).begin(),(x).end()
#define unique(x) (x).resize(unique(all(x))-(x).begin())
#define debug(_fmt,...) \
fprintf(stderr,"("#__VA_ARGS__ ")␣=␣(" _fmt")\n",__VA_ARGS__)

const int S = 26;
const int MAX = 107*20;

namespace aho_corasick{
 int num;
 int mk[MAX];
 int fail[MAX];
 int next[MAX][S];
 char memo[MAX][MAX][17];
 void clean(){
  num = 1;
  for(int i = 0; i < S; i++)next[num - 1][i] = 0;
  memset(memo, -1, sizeof memo);
 }
 void aho_corasick(){
  clean();
 }
 void insert(char* patt){
  int curr = 0;
  for (int j=0; patt[j]; j++){
   int a = patt[j] - 'a';
   if (next[curr][a] == 0){
    for(int i=0;i<S;++i)
    {
     next[curr][i] = num++;
    }
   }
   curr = next[curr][a];
  }
  mk[curr] = 1;
 }
 void construct(){
  queue<int> Q;
  for (int i = 0;i < S;i++)
   if ( next[0][i] ){
    fail[next[0][i]] = 0;
    Q.push( next[0][i] );
   }else next[0][i] = 0;
  while ( !Q.empty() ){
   int t = Q.front(); Q.pop();
   for (int i = 0; i < S; i++){
    int v = next[t][i];
    if ( v ) {
     Q.push( v );
     int u = fail[t];
     while ( u && !next[u][i] ) u = fail[u];
     fail[ v ] = next[u][i];
     mk[v] += mk[fail[v]];
    }
   }
  }
 }


string ans;
char can(int s,int l, int k)
{
 //~ printf("%d %d %d\n", s, l, k);
 if(~memo[s][l][k]) return memo[s][l][k];
```

```
   char &res = memo[s][l][k];
   res=0;
   if((l <= 0 && k) || k < 0)
     return res = 0;
   if(!l && !k)
     return res = 1;

   for(int i=0;i<S;++i)
   {
     int ns = s;
     while (next[ns][i] == 0)
        ns = fail[ns];
     ns = next[ns][i];
     ans += (char) i + 'a';
     int tem = can(ns,l-1,k-mk[ns]);
     if(tem){
       res = 1;
       break;
     }
     ans.pop_back();
   }
   return res;
  }
}

char str[11];
```

```
int main()
{
  #ifdef OJUDGE
    //~ freopen("in","r",stdin);
  #endif

  aho_corasick::clean();

  int n,l,k;
  scanf("%d%d%d", &n, &l, &k);
  for(int i=1;i<=n;++i)
  {
    scanf("%s", str);
    aho_corasick::insert(str);
  }

  aho_corasick::construct();

  if(!aho_corasick::can(0,l,k)) return !(cout <<"Imposible\n");
  cout << aho_corasick::ans << endl;

  return 0;
}
//~ c
```

3.6. Aho-Corasick* -3.

```
//~ Aho-Corasick
// el menor texto en orden lexicografico que tiene L
// letras y no contiene ninguna palabra como substring
#include <bits/stdc++.h>
using namespace std;

const int MAX= 600*10*13+15, S = 26;

namespace aho_corasick{
  int num;
  int bad[MAX];
  int fa[MAX];
  int fail[MAX];
  int next[MAX][S];
  string ans;
  void clean(){
    num = 1;
    for(int i = 0; i < S; i++)next[num - 1][i] = 0;
  }
  void aho_corasick(){
     clean();
  }
  void insert(char* patt){
    int curr = 0;
    for (int j=0; patt[j]; j++){
      int a = patt[j] - 'a';
      if (next[curr][a] == 0){
        for(int i=0;i<S;++i)
        {
          next[curr][i] = num++;
        }
      }
      curr = next[curr][a];
    }
```

```
  bad[curr] = 1;
 }

 void construct(){
  queue<int> Q;
  for (int i = 0;i < S;i++)
   if (next[0][i] ){
    fail[next[0][i]] = 0;
    Q.push( next[0][i] );
   }else next[0][i] = 0;

  while ( !Q.empty() ){
   int t = Q.front(); Q.pop();
   for (int i = 0; i < S; i++){
    int v = next[t][i];
    if ( v ) {
     Q.push( v );
     int u = fail[t];
     while (u && !next[u][i] ) u = fail[u];
     fail[ v ] = next[u][i];
     bad[v] |= bad[next[u][i]];
    }
   }
  }
 }
 void printBad()
 {
  for(int i=0;i<num;++i)
   printf("%d␣%d\n", i, bad[i]);
 }
 void printFail()
 {
  for(int i=0;i<num;++i)
   printf("fail␣%d␣%d\n", i, fail[i]);
 }
 void solve(int s,int p, int k)
 {
  if(p == k)return;
  for(int i=0;i<S;++i)
  {
   int ns = s;
   while (next[ns][i] == 0)
      ns = fail[ns];
   ns = next[ns][i];
   if(!bad[ns] && ns)
   {
    ans+=(char)i+'a';
    solve(ns,p+1,k);
    break;
   }
  }
 }
};

char str[15];
int main()
{
 int n,k;
 scanf("%d%d", &n, &k);
 aho_corasick::aho_corasick();
 for(int i=1;i<=n;++i)
 {
  scanf("%s", str);
  aho_corasick::insert(str);
 }
// aho_corasick::dfs();
 aho_corasick::construct();
 //~ aho_corasick::printBad();
 //~ aho_corasick::printFail();

 aho_corasick::solve(0,0,k);
 if((int)aho_corasick::ans.size() != k)return !(printf("Imposible\n"));
 printf("%s\n", aho_corasick::ans.c_str());

 return 0;
}
```

3.7. Suffix Array.

```
#include <bits/stdc++.h>
using namespace std;
#define ifor(i,st,ed) for(int i=(st);i<=(ed);++i)
const int N = 5e5+7;
namespace sa
{
 char s[N];
 int n, _sa[N], _b[N], top[N], _tmp[N];
 int LCP[N], *SA = _sa, *B = _b, *tmp = _tmp;
 void blcp()
```

```
{
  for(int i = 0, k = 0; i < n; ++i)
  {
    if(B[i] == n - 1)continue;
    for(int j = SA[B[i] + 1]; i + k < n &&
    j + k < n && s[i+k] == s[j + k] && s[i+k] != '$'; k++);
    LCP[B[i]+1] = k;
    if(k) k--;
  }
}
void bsa()
{
  //memset 0 -> _sa, _b, _tmp, top, LCP
  s[n] = '\0', n ++;
  int na = (n < 256 ? 256 : n);
  for(int i = 0; i < n ; i++)top[B[i] = s[i]]++;
  for(int i = 1; i < na; i++)top[i] += top[i - 1];
  for(int i = 0; i < n ; i++)SA[--top[B[i]]] = i;
  for(int ok = 1,j = 0;ok < n && j < n-1;ok <<= 1)
  {
    for(int i = 0; i < n; i++)
    {
      j = SA[i] - ok;
      if (j < 0)j += n;
      tmp[top[B[j]]++] = j;
    }
    SA[tmp[top[0] = 0]] = j = 0;
    for(int i = 1; i < n; i++)
    {
      if(B[tmp[i]] != B[tmp[i - 1]] ||
        B[tmp[i]+ok] != B[tmp[i-1] + ok])
        top[++j] = i;
      SA[tmp[i]] = j;
    }
    swap(B, SA), swap(SA, tmp);
  }
  blcp();
  n --, s[n] = '\0';
}
int t[N][20],Log2[N];
void brmq()
{
  ifor(i,1,n)t[i][0]=LCP[i];
  int x = -1;
  ifor(i,1,N)
  {
    if(!(i&(i-1)))++x;
    Log2[i]=x;
  }
  for(int k=1;(1<<k)<=n;++k)
    for(int i=1;i+(1<<(k-1))<=n;++i)
      t[i][k] = min(t[i][k-1],t[i+(1<<(k-1))][k-1]);
}
int qlcp(int i,int j)
{
  if(i>=j)return INT_MAX;
  int d=Log2[j-i];++i;
  return min(t[i][d],t[j-(1<<d)+1][d]);
}
void check()
{
  ifor(i,1,n)
  {
    ifor(j,0,Log2[n])
      cerr << t[i][j] << '␣';
    cerr << endl;
  }
}
void check_SA()
{
  cerr << "===Suffix␣Array===\n";
  ifor(i,0,n)
  {
    cerr << i << '␣';
    ifor(j,SA[i],n)cerr << s[j];
    cerr << '␣' << LCP[i] << endl;
  }
  cerr << "==================\n";
}
}
int main()
{
  return 0;
}
```

3.8. **Suffix Array* -1.**

```
// coj - 3931
/*
```

```
* For each name, in the same order given in the input,
* print a line containing the shortest pattern that is
* able to distinguish that contact name from the other
* names. If there are multiple patterns, print the
* lexicographically smallest. If there is no pattern,
* print the word 'IMPOSSIBLE'
*/
#include <bits/stdc++.h>

using namespace std;

typedef int I;
typedef pair<int,int> PII;

#define endl '\n'
#define fr first
#define sc second
#define mp make_pair
#define ifor(i,st,ed) for(I i=(st);i<=(ed);++i)
#define dfor(i,st,ed) for(I i=(st);i>=(ed);--i)

const I MAX = 1e6+7;
const I LEN = 1e6+7;
char s[LEN];
int n, _sa[LEN], _b[LEN], top[LEN], _tmp[LEN];
int LCP[LEN], *SA = _sa, *B = _b, *tmp = _tmp;
void build_lcp (){
   for(int i = 0, k = 0; i < n; ++i){
      if(B[i] == n - 1)
         continue;
      for(int j = SA[B[i] + 1]; i + k < n &&
                        j + k < n && s[i+k] == s[j + k]; k++);
      LCP[B[i]] = k;
      if( k ) k--;
   }
}
void build_sa (){
      //memset 0 -> _sa, _b, _tmp, top, LCP
   s[n] = '\0', n ++;
   int na = (n < 256 ? 256 : n);
   for (int i = 0; i < n ; i++)
      top[B[i] = s[i]]++;
   for (int i = 1; i < na; i++)
      top[i] += top[i - 1];
   for (int i = 0; i < n ; i++)
      SA[--top[B[i]]] = i;
   for (int ok = 1,j = 0;ok < n && j < n-1;ok <<= 1){
      for (int i = 0; i < n; i++){
         j = SA[i] - ok;
         if (j < 0)
            j += n;
         tmp[top[B[j]]++] = j;
      }
      SA[tmp[top[0] = 0]] = j = 0;
      for (int i = 1; i < n; i++){
         if (B[tmp[i]] != B[tmp[i - 1]] ||
                        B[tmp[i]+ok] != B[tmp[i-1] + ok])
            top[++j] = i;
         SA[tmp[i]] = j;
      }
      swap(B, SA), swap(SA, tmp);
   }
   build_lcp();
   n --, s[n] = '\0';
}

char tmp0[MAX];
I id[MAX],x0[MAX],l[MAX],lamda[MAX];
PII ans[MAX]; // <length, pos>

int main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(0), cout.tie(0);

  I tot; cin >> tot;
  ifor(i,1,tot)
  {
    cin >> tmp0;
    I len = strlen(tmp0);
    l[i]=len;
    x0[i] = n;
    ifor(j,0,len-1)
    {
      id[n] = i;
      s[n++] = tmp0[j];
    }
    s[n++]='$';
  }

  build_sa();
  fill(ans,ans+n+1,mp(INT_MAX,INT_MAX));

  // algorithm goes here
```

```
I st=0;
I *high=LCP;
ifor(i,0,n-1)SA[i]=SA[i+1];

while(st<n)
{
 I mn=INT_MAX;
 while(st<n-1)
 {
  mn = min(mn,high[st]);
  lamda[st] = mn;
  if(id[SA[st]]!=id[SA[st+1]])
  {
   ++st;
   break;
  }
  ++st;
 }
 if(st==n-1)
 {
  if(id[SA[st-1]]!=id[SA[st]])
   lamda[st]=high[st];
  else lamda[st] = min(mn,high[st]);
  ++st;
 }
}
//cerr << "Part 1 Complete\n";
st=n-1;high[n]=0;
while(~st)
{
 I mn=INT_MAX;
 while(st)
 {
  mn=min(mn,high[st+1]);
  lamda[st]=max(lamda[st],mn);
  if(id[SA[st]]!=id[SA[st-1]])
  {
   --st;
   break;
  }
  --st;
 }
 if(!st)
 {
  if(id[SA[st]]!=id[SA[st+1]])
   lamda[st] = high[st+1];
  else lamda[st]=max(lamda[st],min(mn,high[st+1]));
  --st;
 }
}

//cerr << "Part 2 Complete\n";
ifor(i,0,n-1)
{
 if(s[SA[i]]=='$')continue;
 I xid=id[SA[i]];
 I sln=x0[xid]+l[xid]-SA[i];
 I tmp=lamda[i];
 if(tmp>=sln)continue;
 if(ans[xid].fr>tmp+1)
 {
  ans[xid].fr=tmp+1;
  ans[xid].sc=SA[i];
 }
}

ifor(i,1,tot)
{
 if(ans[i].fr==INT_MAX)
 {
  cout << "IMPOSSIBLE\n";
  continue;
 }
 for(int j=0, x=ans[i].sc;j<ans[i].fr;++j,++x)
  cout << s[x];
 cout << endl;
}
//cerr << "Part 3 Complete\n";
return 0;
}
```

3.9. Suffix Array* -2.

```
/* Suffix Array - CHSTR (codechef)
For each test case, output Q lines,
each line should contain one integer
- amount of ways to choose exactly
Ki equal strings from the list L.
 Example:
```

```
 >> Input:
 1
 5 4
 ababa
 2
 1
 3
 4
 >> Output:
 7
 15
 1
 0
>> Explanation
L = {"a", "b", "a", "b", "a", "ab", "ba",
 "ab", "ba", "aba", "bab", "aba", "abab",
 "baba", "ababa"}.
k1 = 2: There are seven ways to choose two
 equal strings ("a", "a"), ("a", "a"),
 ("a", "a"), ("b", "b"), ("ab", "ab"),
 ("ba", "ba"), ("aba", "aba").
k2 = 1: We can choose any string from L (15 ways).
k3 = 3: There is one way to choose three equal
 strings - ("a", "a", "a").
k4 = 4: There are no four equal strings in L .
*/
#include <bits/stdc++.h>
using namespace std;

typedef int I;
typedef long long int LL;
typedef double D;
typedef long double LD;
typedef pair<I,I> PII;
typedef pair<D,D> PDD;
typedef pair<LL,LL> PLL;
typedef pair<LD,LD> PLD;
typedef complex<D> CPX;

typedef vector<I> VI;
typedef vector<LL> VLL;
typedef vector<D> VD;
typedef vector<LD> VLD;
typedef vector<PII> VPII;
typedef vector<PLL> VPLL;
typedef set<I> SI;
typedef set<LL> SLL;
typedef set<D> SD;
typedef set<LD> SLD;
typedef set<PII> SPII;
typedef set<PLL> SPLL;

#define endl '\n'
#define fr first
#define sc second
#define lb lower_bound
#define ub upper_bound
#define fd find
#define ins insert
#define ers erase
#define ifor(i,st,ed) for(I i=(st);i<=(ed);++i)
#define dfor(i,st,ed) for(I i=(st);i>=(ed);--i)
#define efor(it,x) for(auto it:(x))
#define mp make_pair
#define mt make_tuple
#define pb push_back
#define eb emplace_back
#define cout(p) cout<<fixed<<setprecision(p)
#define sum(x,st,ed) ((x)[ed]-(st>0?(x)[st-1]:0))
#define all(x) (x).begin(),(x).end()
#define sz(x) ((I)(x).size())

const int N = 5e3+7;
const LL MOD = 1e9+7;

namespace sa
{
  char s[N];
  int n, _sa[N], _b[N], top[N], _tmp[N];
  int LCP[N], *SA = _sa, *B = _b, *tmp = _tmp;
  void blcp()
  {
     for(int i = 0, k = 0; i < n; ++i)
     {
        if(B[i] == n - 1)continue;
        for(int j = SA[B[i] + 1]; i + k < n &&
        j + k < n && s[i+k] == s[j + k] && s[i+k]!='$'; k++);
        LCP[B[i]+1] = k;
        if(k) k--;
     }
  }
  void bsa()
  {
    //memset 0 -> _sa, _b, _tmp, top, LCP
```

```
    s[n] = '\0', n ++;
    int na = (n < 256 ? 256 : n);
    for(int i = 0; i < n ; i++)top[B[i] = s[i]]++;
    for(int i = 1; i < na; i++)top[i] += top[i - 1];
    for(int i = 0; i < n ; i++)SA[--top[B[i]]] = i;
    for(int ok = 1,j = 0;ok < n && j < n-1;ok <<= 1)
    {
       for(int i = 0; i < n; i++)
       {
          j = SA[i] - ok;
          if (j < 0)j += n;
          tmp[top[B[j]]++] = j;
       }
       SA[tmp[top[0] = 0]] = j = 0;
       for(int i = 1; i < n; i++)
       {
          if(B[tmp[i]] != B[tmp[i - 1]] ||
             B[tmp[i]+ok] != B[tmp[i-1] + ok])
             top[++j] = i;
          SA[tmp[i]] = j;
       }
       swap(B, SA), swap(SA, tmp);
    }
    blcp();
    n --, s[n] = '\0';
}
I t[N][20],Log2[N];
void brmq()
{
  ifor(i,1,n)t[i][0]=LCP[i];
  I x = -1;
  ifor(i,1,N)
  {
    if(!(i&(i-1)))++x;
    Log2[i]=x;
  }
  for(I k=1;(1<<k)<=n;++k)
    for(I i=1;i+(1<<(k-1))<=n;++i)
      t[i][k] = min(t[i][k-1],t[i+(1<<(k-1))][k-1]);
}
I qlcp(I i,I j)
{
  if(i>=j)return INT_MAX;
  I d=Log2[j-i];++i;
  return min(t[i][d],t[j-(1<<d)+1][d]);
}
void check()
  {
    ifor(i,1,n)
    {
      ifor(j,0,Log2[n])
        cerr << t[i][j] << ' ';
      cerr << endl;
    }
  }
}

LL C[N][N], P[N];

int bs(int lo,int hi,int l,int k)
{
  if(hi-lo<=1)
  {
    int f = sa::qlcp(k,hi);
    if(f>=l)return hi;
    return lo;
  }
  int m = (lo+hi)>>1;
  int f = sa::qlcp(k,m);
  if(f>=l)
    return bs(m,hi,l,k);
  return bs(lo,m-1,l,k);
}

void init()
{
  for(int i=1;i<=sa::n;++i)
    for(int j=sa::LCP[i]+1;j<=sa::n-sa::SA[i];++j)
    {
      int up = bs(i,sa::n,j,i);
      ++P[up-i+1];
    }
}

LL ans[N];

inline void clean()
{//memset 0 -> _sa, _b, _tmp, top, LCP
  memset(sa::_sa,0,sizeof sa::_sa);
  memset(sa::_b,0,sizeof sa::_b);
  memset(sa::_tmp,0,sizeof sa::_tmp);
  memset(sa::top,0,sizeof sa::top);
  memset(sa::LCP,0,sizeof sa::LCP);
  memset(P,0,sizeof P);
```

```
}

int main()
{
 ios_base::sync_with_stdio(false);
 cin.tie(0),cout.tie(0);

 for(int i=0;i<N;++i)
 {
  C[i][0]=C[i][i]=1;
  for(int j=1;j<i;++j)
   C[i][j] = (C[i-1][j-1]+C[i-1][j])%MOD;
 }
 int t;cin >> t;
 while(t--)
 {
  int q;
  cin >> sa::n >> q;

  for(int i=0;i<sa::n;++i)
   cin >> sa::s[i];

  sa::bsa(); // O(N1g^2N)
  sa::brmq(); // O(N1gN)

  init(); // computes P[i] - O(N^21gN)
  memset(ans,-1,sizeof ans);
  while(q--)
  {
   int k; cin >> k;
   if(k>sa::n)
   {
    cout << "0\n";
    continue;
   }
   if(~ans[k])
   {
    cout << ans[k] << endl;
    continue;
   }
   ans[k]=0;
   for(int i=k;i<=sa::n;++i)
    ans[k] = (ans[k]+P[i]*C[i][k]%MOD)%MOD;
   cout << ans[k] << endl;
  }

  if(t)clean();
 }

 return 0;
}
```

3.10. **Suffix Array* -3.**

```
/* Suffix Arrays
If LCP of Xth and Yth strings is L then how many
times L occures in the set as a substring of any string?
 Example
 Input:
 4
 ababa
 aba
 abcd
 cbab
 3
 1 2
 2 3
 1 4
 Output:
 3
 5
 0
*/
#include <bits/stdc++.h>
using namespace std;

typedef int I;
typedef pair<I,I> PII;

#define endl '\n'
#define fr first
#define sc second
#define pb push_back
#define eb emplace_back
#define mp make_pair
#define mt make_tuple
#define ins insert
#define ers erase
```

```
#define lb lower_bound
#define ub upper_bound
#define fd find
#define sz(v) ((I)(v).size())
#define ifor(i,st,ed) for(I i=(st);i<=(ed);++i)
#define dfor(i,st,ed) for(I i=(st);i>=(ed);--i)
#define efor(it,x) for(auto it:(x))
#define all(x) (x).begin(),(x).end()

const int N = 6e5+7;

namespace sa
{
 char s[N];
 int n, _sa[N], _b[N], top[N], _tmp[N];
 int LCP[N], *SA = _sa, *B = _b, *tmp = _tmp;
 void blcp()
 {
    for(int i = 0, k = 0; i < n; ++i)
    {
       if(B[i] == n - 1)continue;
       for(int j = SA[B[i] + 1]; i + k < n &&
       j + k < n && s[i+k] == s[j + k] && s[i+k]!='$'; k++);
       LCP[B[i]+1] = k;
       if(k) k--;
    }
 }
 void bsa()
 {
  //memset 0 -> _sa, _b, _tmp, top, LCP
    s[n] = '\0', n ++;
    int na = (n < 256 ? 256 : n);
    for(int i = 0; i < n ; i++)top[B[i] = s[i]]++;
    for(int i = 1; i < na; i++)top[i] += top[i - 1];
    for(int i = 0; i < n ; i++)SA[--top[B[i]]] = i;
    for(int ok = 1,j = 0;ok < n && j < n-1;ok <<= 1)
    {
       for(int i = 0; i < n; i++)
       {
          j = SA[i] - ok;
          if (j < 0)j += n;
          tmp[top[B[j]]++] = j;
       }
       SA[tmp[top[0] = 0]] = j = 0;
       for(int i = 1; i < n; i++)
       {
          if(B[tmp[i]] != B[tmp[i - 1]] ||
             B[tmp[i]+ok] != B[tmp[i-1] + ok])
             top[++j] = i;
          SA[tmp[i]] = j;
       }
       swap(B, SA), swap(SA, tmp);
    }
    blcp();
    n --, s[n] = '\0';
 }
 I t[N][20],Log2[N];
 void brmq()
 {
  ifor(i,1,n)t[i][0]=LCP[i];
  I x = -1;
  ifor(i,1,N)
  {
   if(!(i&(i-1)))++x;
   Log2[i]=x;
  }
  for(I k=1;(1<<k)<=n;++k)
   for(I i=1;i+(1<<(k-1))<=n;++i)
    t[i][k] = min(t[i][k-1],t[i+(1<<(k-1))][k-1]);
 }
 I qlcp(I i,I j)
 {
  if(i>=j)return INT_MAX;
  I d=Log2[j-i];++i;
  return min(t[i][d],t[j-(1<<d)+1][d]);
 }
 void check()
 {
  ifor(i,1,n)
  {
   ifor(j,0,Log2[n])
    cerr << t[i][j] << '␣';
   cerr << endl;
  }
 }
}

int slen[N],pos[N],x[N];

int bs1(int lo,int hi,const int &fx,const int &lcp)
{
 if(hi-lo<=1)
 {
  int fl=sa::qlcp(lo,fx);
```

```
   if(f1>=lcp)return lo;
   return hi;
  }
  int m=(lo+hi)>>1;
  int f=sa::qlcp(m,fx);
  if(f>=lcp)
   return bs1(lo,m,fx,lcp);
  return bs1(m+1,hi,fx,lcp);
}
int bs2(int lo,int hi,const int &fx,const int &lcp)
{
  if(hi-lo<=1)
  {
   int f1=sa::qlcp(fx,hi);
   if(f1>=lcp)return hi;
   return lo;
  }
  int m=(lo+hi)>>1;
  int f=sa::qlcp(fx,m);
  if(f>=lcp)
   return bs2(m,hi,fx,lcp);
  return bs2(lo,m-1,fx,lcp);
}

int main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(0),cout.tie(0);

  int n; cin >> n;sa::n=0;
  string tmp;
  ifor(i,1,n)
  {
   cin >> tmp;
   slen[i]=sz(tmp);
   pos[i]=sa::n;
   efor(c,tmp)
   {
    sa::s[sa::n++]=c;
   }
   sa::s[sa::n++]='$';
  }
```

3.11. **Suffix Array* -4.**

```
/* Suffix Array - SSTORY (codechef)
 * Encuentra la subcadena comun mas larga

  sa::bsa();sa::brmq();

  // * cerr << "===Suffix Array===\n";
  // * ifor(i,0,sa::n)
  // * {
   // * cerr << i << ' ';
   // * ifor(j,sa::SA[i],sa::n)cerr << sa::s[j];
   // * cerr << ' ' << sa::LCP[i];
   // * cerr << endl;
  // * }
  // * cerr << "==================\n";

  ifor(i,1,sa::n)
   x[sa::SA[i]] = i;
  ifor(i,1,n)
   pos[i]=x[pos[i]];

  int q;cin >> q;
  while(q--)
  {
   int u,v; cin >> u >> v;int tmp=slen[u];
   u = pos[u];v = pos[v];
   if(u>v)swap(u,v);
   int lcp=min(sa::qlcp(u,v),tmp);

   if(!lcp)
   {
    cout << "0\n";
    continue;
   }

   int lo = bs1(1,u,u,lcp);
   int hi = bs2(v,sa::n,v,lcp);

   //cerr << lo << ' ' << hi << endl;
   cout << hi-lo+1 << endl;
  }

  return 0;
}
```

```
 * entre dos cadenas, en caso de empate
 * la que primero aparezca en la segunda
 * va primero cadena.
 * Example:
 * >> Input:
 * adsyufsfdsfdsf
 * fdyusgfdfyu
 * >> Output:
 * fd
 * 2
 */
#include <bits/stdc++.h>
using namespace std;

typedef int I;
typedef pair<I,I> PII;

#define endl '\n'
#define fr first
#define sc second
#define pb push_back
#define eb emplace_back
#define mp make_pair
#define mt make_tuple
#define ins insert
#define ers erase
#define lb lower_bound
#define ub upper_bound
#define fd find
#define sz(v) ((I)(v).size())
#define ifor(i,st,ed) for(I i=(st);i<=(ed);++i)
#define dfor(i,st,ed) for(I i=(st);i>=(ed);--i)
#define efor(it,x) for(auto it:(x))
#define all(x) (x).begin(),(x).end()

const int N = 5e5+7;

namespace sa
{
  char s[N];
  int n, _sa[N], _b[N], top[N], _tmp[N];
  int LCP[N], *SA = _sa, *B = _b, *tmp = _tmp;
  void blcp()
  {
     for(int i = 0, k = 0; i < n; ++i)
     {
        if(B[i] == n - 1)continue;
        for(int j = SA[B[i] + 1]; i + k < n &&
        j + k < n && s[i+k] == s[j + k] && s[i+k] != '$'; k++);
        LCP[B[i]+1] = k;
        if(k) k--;
     }
  }
  void bsa()
  {
    //memset 0 -> _sa, _b, _tmp, top, LCP
    s[n] = '\0', n ++;
    int na = (n < 256 ? 256 : n);
    for(int i = 0; i < n ; i++)top[B[i] = s[i]]++;
    for(int i = 1; i < na; i++)top[i] += top[i - 1];
    for(int i = 0; i < n ; i++)SA[--top[B[i]]] = i;
    for(int ok = 1,j = 0;ok < n && j < n-1;ok <<= 1)
    {
       for(int i = 0; i < n; i++)
       {
          j = SA[i] - ok;
          if (j < 0)j += n;
          tmp[top[B[j]]++] = j;
       }
       SA[tmp[top[0] = 0]] = j = 0;
       for(int i = 1; i < n; i++)
       {
          if(B[tmp[i]] != B[tmp[i - 1]] ||
             B[tmp[i]+ok] != B[tmp[i-1] + ok])
             top[++j] = i;
          SA[tmp[i]] = j;
       }
       swap(B, SA), swap(SA, tmp);
    }
    blcp();
    n --, s[n] = '\0';
  }
  I t[N][20],Log2[N];
  void brmq()
  {
    ifor(i,1,n)t[i][0]=LCP[i];
    I x = -1;
    ifor(i,1,N)
    {
      if(!(i&(i-1)))++x;
      Log2[i]=x;
    }
    for(I k=1;(1<<k)<=n;++k)
      for(I i=1;i+(1<<(k-1))<=n;++i)
```

```
      t[i][k] = min(t[i][k-1],t[i+(1<<(k-1))][k-1]);
  }
  I qlcp(I i,I j)
  {
   if(i>=j)return INT_MAX;
   I d=Log2[j-i];++i;
   return min(t[i][d],t[j-(1<<d)+1][d]);
  }
  void check()
  {
   ifor(i,1,n)
   {
    ifor(j,0,Log2[n])
     cerr << t[i][j] << '␣';
    cerr << endl;
   }
  }
  void check_SA()
  {
   cerr << "===Suffix␣Array===\n";
   ifor(i,0,n)
   {
    cerr << i << '␣';
    ifor(j,SA[i],n)cerr << s[j];
    cerr << '␣' << LCP[i];
    cerr << endl;
   }
   cerr << "==================\n";
  }
}

int id[N];

void check_SA()
  {
   cerr << "===Suffix␣Array===\n";
   ifor(i,0,sa::n)
   {
    cerr << i << '␣';
    ifor(j,sa::SA[i],sa::n)cerr << sa::s[j];
    cerr << '␣' << sa::LCP[i] << '␣' << id[sa::SA[i]];
    cerr << endl;
   }
   cerr << "==================\n";
  }

int main()
{
 ios_base::sync_with_stdio(false);
 cin.tie(0),cout.tie(0);

 string tmp;
 cin >> tmp;
 efor(c,tmp)
 {
  id[sa::n] = 1;
  sa::s[sa::n++]=c;
 }
 sa::s[sa::n++]='$';
 cin >> tmp;
 efor(c,tmp)
 {
  id[sa::n] = 2;
  sa::s[sa::n++]=c;
 }
 sa::s[sa::n++]='$';

 sa::bsa();
 sa::brmq();
 //check_SA();

 vector<int> pos[2];
 ifor(i,3,sa::n)
  pos[id[sa::SA[i]]-1].pb(i);

 PII ans=mp(-1,-1);

 auto upd_ans=[&](int x,int len)
 {
  if(ans.fr < len)
  {
   ans.fr = len;
   ans.sc = sa::SA[x];
  }else if(ans.fr == len)
   if(ans.sc > sa::SA[x])
    ans.sc = sa::SA[x];
 };

 efor(it,pos[1])
 {
  auto up=ub(all(pos[0]),it);
  if(up == pos[0].begin())
  {
   int lcp=sa::qlcp(it,*up);
```

```
  upd_ans(it,lcp);
 }else if(up == pos[0].end())
 {
  up--;
  int lcp=sa::qlcp(*up,it);
  upd_ans(it,lcp);
 }else
 {
  int lcp=sa::qlcp(it,*up);
  upd_ans(it,lcp);
  up--;
  lcp=sa::qlcp(*up,it);
  upd_ans(it,lcp);
 }
 }

 if(!ans.fr)
 {
  cout << "0\n";
  return 0;
 }
 for(int i=ans.sc,j=1;j<=ans.fr;++i,++j)
  cout << sa::s[i];
 cout << endl << ans.fr << endl;

 return 0;
}
```

4. Convolution

4.1. Mobius All.

```
/* BITWISE CONVOLUTIONS
 * Outline (in order of appearance):
 * >> (f*g)[s]=Sum(f[a]*f[b] : [a|b=s and a&b=0]) -> aka SUBSET
 * >> (f*g)[s]=Sum(f[a]*f[b] : [a|b=s]) -> aka OR
 * >> (f*g)[s]=Sum(f[a]*f[b] : [a&b=0])
 * >> (f*g)[s]=Sum(f[a]*f[b] : [a|b=s and a&b!=0])
 * >> (f*g)[s]=Sum(f[a]*g[b] : [a&b=s]) -> aka AND
 * >> (f*g)[s]=Sum(f[a]*g[b] : [a^b=s]) -> aka XOR
 * Testeado en 458G - codeforces (algunas)
 * */
namespace Mobius
{
 typedef long long int LL;

 const int B=17;
 const int N=1<<B;
 const LL invN=(LL)742744451;
 const LL M=(LL)1e9+7;

 int count[N];
 void init_count() // call this first
 {for(int i=0;i<N;++i)count[i]=__builtin_popcount(i);}
 void add(LL &x,LL y)
 {
  x+=y;
  if(x>=M)x-=M;
  else if(x<0)x+=M;
 }
 void mobius_transform(LL A[],LL inv=1ll)
 {
  for(int i=0;i<B;++i)
   for(int j=0;j<N;++j)
    if(j>>i&1)
     add(A[j],inv*A[j^1<<i]);
 }

 void ranked_mobius(LL A[],LL ans[][N])
 {
  for(int k=0;k<=B;++k)
  {
   LL* a=ans[k];
   for(int i=0;i<N;++i)
    if(count[i]==k)a[i]=A[i];
    else a[i]=0;
   mobius_transform(a);
  }
 }
 void inverse_ranked_mobius(LL A[][N],LL ans[])
 { // this could be improve to O(N)
  for(int k=0;k<=B;++k)
  {
   LL* a=A[k];
   mobius_transform(a,-1);
   for(int i=0;i<N;++i)
    if(count[i]==k)ans[i]=a[i];
  }
 }

 // BITWISE CONVOLUTIONS
 LL frm[B+1][N],grm[B+1][N],fg[B+1][N];
 // Partitioning product:
 // (f*g)[s]=Sum(f[a]*f[b] : [a|b=s and a&b=0])
 void subset_convolution(LL f[],LL g[],LL ans[])
 {// Testeado en 458G
  memset(fg,0,sizeof fg);
  ranked_mobius(f,frm);
  ranked_mobius(g,grm);
  for(int k=0;k<=B;++k)
   for(int X=0;X<N;++X)
    for(int r=0;r<=k;++r)
     add(fg[k][X],frm[r][X]*grm[k-r][X]%M);
  inverse_ranked_mobius(fg,ans);
 }
 // Covering product:
 // (f*g)[s]=Sum(f[a]*f[b] : [a|b=s])
 LL tmp1[N],tmp2[N];
 void or_convolution(LL f[],LL g[],LL ans[])
 {
  memcpy(tmp1,f,N* sizeof(LL));
  memcpy(tmp2,g,N* sizeof(LL));
  mobius_transform(tmp1);
  mobius_transform(tmp2);
  for(int i=0;i<N;++i)
   ans[i]=tmp1[i]*tmp2[i]%M;
  mobius_transform(ans,-1);
 }
```

```
// Packing product:
// (f*g)[s]=Sum(f[a]*f[b] : [a&b=0])
void packing_product(LL f[],LL g[],LL ans[])
{
 subset_convolution(f,g,ans);
 mobius_transform(ans);
}
// Intersecting covering product:
// (f*g)[s]=Sum(f[a]*f[b] : [a|b=s and a&b!=0])
void intersecting_covering_product(LL f[],LL g[],LL ans[],int l=0)
{
 or_convolution(f,g,tmp1);
 subset_convolution(f,g,tmp2);
 for(int i=0;i<N;++i)
 {
  ans[i]=tmp1[i];
  add(ans[i],-tmp2[i]);
 }
}
// Intersecting product (AND):
// (f*g)[s]=Sum(f[a]*g[b] : [a&b=s])
void transform_and(LL A[],LL inv=1)
{
 for(int i=0;i<B;++i)
  for(int j=0;j<N;++j)
   if(!(j>>i&1))
    add(A[j],inv*A[j^1<<i]);
}
void and_convolution(LL f[],LL g[],LL ans[])
{ // Testeado en 458G
 memcpy(tmp1,f,N* sizeof(LL));
 memcpy(tmp2,g,N* sizeof(LL));
 transform_and(tmp1);
 transform_and(tmp2);
 for(int i=0;i<N;++i)
  ans[i]=tmp1[i]*tmp2[i]%M;
 transform_and(ans,-1);
}
// Excluding product (XOR):
// (f*g)[s]=Sum(f[a]*g[b] : [a^b=s])
void transform_xor(LL x[],bool inv=0)
{
 for(int mid=1,i=2;i<=N;mid=i,i<<=1)
  for(int j=0,lo=0;lo<N;j+1==mid?(lo+=i,j=0):++j)
  {
   LL tw1=x[lo+j],tw2=x[lo+j+mid];
   x[lo+j]=tw1+tw2;
   if(x[lo+j]>=M)x[lo+j]=x[lo+j]-M;
   x[lo+j+mid]=tw1-tw2;
   if(x[lo+j+mid]<0)x[lo+j+mid]=x[lo+j+mid]+M;
  }
 if(inv)for(int i=0;i<N;++i)x[i]=x[i]*invN%M;
}
void xor_convolution(LL f[],LL g[],LL ans[])
{ // Testeado en 458G
 memcpy(tmp1,f,N* sizeof(LL));
 memcpy(tmp2,g,N* sizeof(LL));
 transform_xor(tmp1);
 transform_xor(tmp2);
 for(int i=0;i<N;++i)
  ans[i]=tmp1[i]*tmp2[i]%M;
 transform_xor(ans,1);
}
}
```

4.2. Transformadas Xor, And.

```
/* Transformada Xor.
 * Transformada And.
 * Para transformada inversa llamar con inv=true.
 * Testeado en 458G - codeforces.
 */
const int B=17;
const int N=1<<B;
const LL M=(int)1e9+7;
const LL inv2=(int)5e8+4;
void transform_xor(LL x[],bool inv=0)
{
 for(int mid=1,i=2;i<=N;mid=i,i<<=1)
  for(int j=0,lo=0;lo<N;j+1==mid?(lo+=i,j=0):++j)
  {
   LL tw1=x[lo+j],tw2=x[lo+j+mid];
   if(inv)
   {
    x[lo+j]=(tw1+tw2)%M*inv2%M;
    x[lo+j+mid]=(tw1-tw2+M)%M*inv2%M;
   }
   else
   {
```

```
      x[lo+j]=tw1+tw2;
      if(x[lo+j]>=M)x[lo+j]-=M;
      x[lo+j+mid]=tw1-tw2;
      if(x[lo+j+mid]<0)x[lo+j+mid]+=M;
     }
   }
}
void transform_and(LL p[],bool inv=0)
{
  for(int len=1;(len<<1)<=N;len<<=1)
    for(int i=0;i<N;i+=(len<<1))
      for(int j=0;j<len;++j)
      {
       LL tw1=p[i+j];
       LL tw2=p[i+len+j];
       if(!inv)
       {
         p[i+j]=tw2;
         p[i+len+j]=tw1+tw2;
         if(p[i+len+j]>=M)p[i+len+j]-=M;
       }
       else
       {
         p[i+j]=tw2-tw1;
         if(p[i+j]<0)p[i+j]+=M;
         p[i+len+j]=tw1;
       }
      }
}
```

4.3. Mobius All*.

```
//¯ 458G - codeforces.
/* You are given an array S of n non-negative integers.
 * A 5-tuple of integers (a, b, c, d, e) is said
 * to be valid if it satisfies the following conditions:
 * -> 1 <= a, b, c, d, e <= n
 * -> (Sa | Sb) & Sc & (Sd ^ Se) = 2^i for some integer i
 * -> Sa & Sb = 0
 * Find the sum of f(Sa|Sb) * f(Sc) * f(Sd^Se) over all
 * valid 5-tuples (a, b, c, d, e), where f(i) is the i-th
 * Fibonnaci number (f(0) = 0, f(1) = 1, f(i) = f(i-1) + f(i-2))
 *
 * 1. se almacena en 'cnt' las frecuencias para cada indice de
 * fibonnaci en la entrada.
 * 2. se calcula la subset_convolution de 'cnt' con el mismo
 * y se almacena en 'ab'.
 * 3. se caclula la xor_convolution y se almacena en 'Xor'.
 * >> En cnt[i] se tiene la frecuencia del indice i de los fibonnaci.
 * >> En ab[i] se tiene la frecuencia del con que se obtiene el
 * incide i como union de dos indices a,b (i.e. a|b=i y a&b=0).
 * >> En Xor[i] se tiene la frecuencia del con que se obtiene el
 * incide i como xor de dos indices a,b (i.e. a^b=i).
 * 4. se multiplican esas frecuencias por los respectivos fibonnaci
 * (i.e.
 * for(int i=0;i<N;++i):
 * Xor[i]=Xor[i]*fib[i]
 * ab[i]=ab[i]*fib[i]
 * cnt[i]=cnt[i]*fib[i] ).
 * 5. se calcula la and_convolution de 'Xor', 'ab' y 'cnt' y se
 * almacena en 'And'.
 * 6. para dar respuesta se suman los indices con potencia de dos.
 */
#pragma comment(linker, "/stack:200000000")
#pragma GCC optimize("Ofast")
//#pragma GCC optimize("unroll-loops")
#pragma GCC \
target("sse,sse2,sse3,ssse3,sse4,popcnt,abm,mmx,avx,tune=native")

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

typedef long long int LL;
#define endl '\n'

const int B=17;
const int N=1<<B;
const int M=(int)1e9+7;
const LL invN=(LL)742744451;
typedef int I;
namespace Mobius
{
 int count[N];
 void init_count()
 {for(int i=0;i<N;++i)count[i]=__builtin_popcount(i);}
 void add(I &x,I y)
 {
   x+=y;
```

```
  if(1ll*x>=M)x-=M;
  else if(x<0)x+=M;
}
void mobius_transform(I A[],I inv=1ll)
{
  for(int i=0;i<B;++i)
    for(int j=0;j<N;++j)
      if(j>>i&1)
        add(A[j],inv*A[j^1<<i]);
}

void ranked_mobius(I A[],I ans[][N])
{
  for(int k=0;k<=B;++k)
  {
    I* a=ans[k];
    for(int i=0;i<N;++i)
      if(count[i]==k)a[i]=A[i];
      else a[i]=0;
    mobius_transform(a);
  }
}
void inverse_ranked_mobius(I A[][N],I ans[])
{
  for(int k=0;k<=B;++k)
  {
    I* a=A[k];
    mobius_transform(a,-1);
    for(int i=0;i<N;++i)
      if(count[i]==k)ans[i]=a[i];
  }
}
I frm[B+1][N],grm[B+1][N],fg[B+1][N];
void subset_convolution(I f[],I g[],I ans[])
{
  // Partitioning product:
  // (f*g)[s]=Sum(f[a]*f[b] : [a|b=s and a&b=0])
  memset(fg,0,sizeof fg);
  ranked_mobius(f,frm);
  ranked_mobius(g,grm);
  for(int k=0;k<=B;++k)
    for(int X=0;X<N;++X)
      for(int r=0;r<=k;++r)
        add(fg[k][X],1ll*frm[r][X]*grm[k-r][X]%M);
  inverse_ranked_mobius(fg,ans);
}
I tmp1[N],tmp2[N],tmp3[N];
```

```
  void transform_and(I A[],I inv=1)
  {
    for(int i=0;i<B;++i)
      for(int j=0;j<N;++j)
        if(!(j>>i&1))
          add(A[j],inv*A[j|1<<i]);
  }
  void and_convolution(I f[],I g[],I h[],I ans[])
  {
    // Intersection product:
    // (f*g)[s]=Sum(f[a]*f[b] : [a&b=s])
    transform_and(f);
    transform_and(g);
    transform_and(h);
    for(int i=0;i<N;++i)
      ans[i]=1ll*f[i]*g[i]%M*h[i]%M;
    transform_and(ans,-1);
  }
};
using namespace Mobius;
I cnt[N],fib[N],ab[N],Xor[N],And[N];
void transform_xor(I x[],bool inv=0)
{
  for(int mid=1,i=2;i<=N;mid=i,i<<=1)
    for(int j=0,lo=0;lo<N;j+1==mid?(lo+=i,j=0):++j)
    {
      I tw1=x[lo+j],tw2=x[lo+j+mid];
      x[lo+j]=tw1+tw2;
      if(1ll*x[lo+j]>=M)x[lo+j]-=M;
      x[lo+j+mid]=tw1-tw2;
      if(x[lo+j+mid]<0)x[lo+j+mid]+=M;
    }
  if(inv)
    for(int i=0;i<N;++i)
    {
      x[i]=(1ll*x[i]*invN)%M;
    }
}
void solve()
{
  fib[0]=0,fib[1]=1;
  for(int i=2;i<N;++i)
  {
    fib[i] = fib[i - 1] + fib[i - 2];
    if(1ll*fib[i]>=M)fib[i]=1ll*fib[i]-M;
  }
```

```
int n;scanf("%d", &n);
for(int i=1;i<=n;++i)
{
  int x;scanf("%d", &x);
  ++cnt[x];
}

// Subset Convolution (fast O((2^B)*B^2))
init_count();
subset_convolution(cnt,cnt,ab);
/*for(int i=0;i<16;++i)
  out<<ab[i]<<" \n"[i==15];*/

// Xor Convolution
memcpy(Xor,cnt,sizeof cnt);
transform_xor(Xor);
for(int i=0;i<N;++i)
  Xor[i]=1ll*Xor[i]*Xor[i]%M;
 transform_xor(Xor,1);

 for(int i=0;i<N;++i)
  Xor[i]=1ll*Xor[i]*fib[i]%M,
  ab[i]=1ll*ab[i]*fib[i]%M,
  cnt[i]=1ll*cnt[i]*fib[i]%M;

 // And Convolution
 and_convolution(Xor,ab,cnt,And);
 I ans=0;
 for(int i=1;i<N;i<<=1)ans=(ans+And[i])%M;
 printf("%d\n", ans);
}

int main()
{return solve(),0;}
```

5. Geometry

5.1. Convex Hull (hashed).

```
// testeado coj - 1554, 3358,
// https://codeforces.com/contest/1017/problem/E
#include <bits/stdc++.h>
using namespace std;

typedef long long int ll;
struct R2vec
{
 int x,y;
 R2vec(){}
 R2vec(int x,int y):x(x),y(y){}
 inline R2vec operator+ (const R2vec &w){return R2vec(x+w.x,y+w.y);}
 inline R2vec operator- (const R2vec &w){return R2vec(x-w.x,y-w.y);}
 inline R2vec operator- (){return R2vec(-x,-y);}
 inline ll operator* (const R2vec &w){return (ll)x*w.x + (ll)y*w.y;} // dot
 inline ll operator% (const R2vec &w){return (ll)x*w.y - (ll)y*w.x;} // cross
 inline ll operator~ (){return (ll)x*x + (ll)y*y;} // norm^2
 inline bool operator< (const R2vec &w)const{return x!=w.x?x<w.x:y<w.y;}
 inline bool operator== (const R2vec &w){return x==w.x&&y==w.y;}
 inline bool operator!= (const R2vec &w){return x!=w.x||y!=w.y;}
};
inline ostream &operator<< (ostream &out,const R2vec &w)
{return out<<'['<<w.x<<','<<w.y<<']';}

typedef pair<ll,ll> pll;
#define all(x) (x).begin(),(x).end()
#define sz(x) ((int)(x).size())
#define pb push_back
#define eb emplace_back

struct ConvexHull
{
 vector<R2vec> hull;
 ConvexHull(){}
 ConvexHull(vector<R2vec> cloud)
 {
  sort(all(cloud)); vector<R2vec> up,down;
  for(auto &w: cloud)
  {
   while(sz(up) > 1 && (up.back()-up[sz(up)-2])%(w-up.back())>=0)
    up.pop_back();
   up.pb(w);
   while(sz(down) > 1 && (down.back()-down[sz(down)-2])%(w-down.back())<=0)
    down.pop_back();
   down.pb(w);
  }
  hull = up;
  for(int i=sz(down)-2;i;--i)hull.pb(down[i]);
  //~ cout << "UPPER:\n";for(auto &w: up)cout << w;cout << endl;
  //~ cout << "LOWER:\n";for(auto &w: down)cout << w;cout << endl;
  //~ cout << "Convex Hull\n";for(auto &w: hull)cout << w;cout << endl;
 }
 #define NEXT(i) (i+1<sz(hull)?i+1:0)
 #define PREV(i) (i?i-1:sz(hull)-1)
 vector<pll> hash() // util para comparar dos convex hulls
 { // testeada en https://codeforces.com/contest/1017/problem/E
  vector<pll> res(sz(hull));
  for(int i=0;i<sz(hull);++i)
   res[i]={~(hull[i]-hull[PREV(i)]),
      (hull[PREV(i)]-hull[i])*(hull[NEXT(i)]-hull[i])};
  return res;
 }
};

int main()
{
 vector<R2vec> p;
 int n;cin >> n;
 while(n--)
 {
  int x,y;
  cin >> x >> y;
  p.pb(R2vec(x,y));
 }
 ConvexHull CH(p);
 return 0;
}
```

5.2. Convex Hull.

```
struct pt {
      int x, y, id;

      pt(int x_ = 0, int y_ = 0, int id_ = 0) {
            x = x_, y = y_, id = id_;
      }

      bool operator < (const pt &p) const {
            if(y != p.y)
                  return y < p.y;
            return x < p.x;
      }

      pt operator - (const pt &p) const {
            return pt(x - p.x, y - p.y, id);
      }
};

int cross(pt a, pt b) {
      return a.x * b.y - a.y * b.x;
}

vector <pt> p;

vector <pt> convexHull(vector <pt> &p) {
      sort(p.begin(), p.end());

      int n = p.size();
      vector <pt> h(2 * n);

      int k = 0;
      for(int i = 0; i < n; i++) {
            while(k >= 2 && cross(h[k - 1] - h[k - 2], p[i] - h[k - 2]) <= 0)
                  k--;
            h[k++] = p[i];
      }

      for(int i = n - 2, t = k + 1; i >= 0; i--) {
            while(k >= t && cross(h[k - 1] - h[k - 2], p[i] - h[k - 2]) <= 0)
                  k--;
            h[k++] = p[i];
      }

      h.resize(k - 1);
      return h;
}
```

5.3. AntipodalPairOfPoints.

```
//recibe como parametro el poligono convexo
vector <pair <pt, pt> > antipodal(vector <pt> &r) {
      vector <pair <pt, pt> > ans;
      int k = 1;
      int m = r.size();

      while(k < m - 1 && llabs(cross(r[k + 1] - r[m - 1],
   r[0] - r[m - 1])) > llabs(cross(r[k] - r[m - 1], r[0] - r[m - 1])))
            k++;

      int j = k;

      for(int i = 0; i <= k && j < m; i++) {
            ans.push_back({r[i], r[j]});

            while(j < m - 1 && llabs(cross(r[j + 1] - r[i],
     r[i + 1] - r[i])) > llabs(cross(r[j] - r[i], r[i + 1] - r[i]))) {
                  j++;
                  ans.push_back({r[i], r[j]});
            }
      }

      return ans;
}
```

5.4. ClosestPairOfPoints.

```
//Closest Pair Of Points O(nlogn)
```

```
struct pt {
      int x, y;

      pt() {}

      pt(int x, int y) : x(x), y(y) {}
} p[100005];

bool operator < (const pt &a, const pt &b) { //comparador del multiset
      if(a.y != b.y)
            return a.y < b.y;
      return a.x < b.x;
}

bool cmp(const pt &a, const pt &b) { //comparador para arreglo p
      if(a.x != b.x)
            return a.x < b.x;
      return a.y < b.y;
}

int n;
multiset <pt> s;

int dist(const pt &a, const pt &b) { //metrica especifica del problema
      return max(abs(a.x - b.x), abs(a.y - b.y));
}

int main() {
      scanf("%d", &n);
      for(int i = 1; i <= n; i++)
            scanf("%d%d", &p[i].x, &p[i].y);

      sort(p + 1, p + n + 1, cmp);

      s.clear(); //limpiar s, importante

      int ans = 1e9;
      for(int i = 1, j = 1; i <= n; i++) {
            while(p[i].x - p[j].x > ans) {
                  s.erase(s.find(p[j]));
                  j++;
            }

            auto l = s.lower_bound(pt(-1e9, p[i].y - ans - 1));
            auto u = s.upper_bound(pt(1e9, p[i].y + ans + 1));

            for(; l != u; l++)
                  ans = min(ans, dist(p[i], *l));

            s.insert(p[i]);
      }

      printf("%d\n", ans);
      return 0;
}
```

5.5. Geometría Programing Contest.

```
#include <bits/stdc++.h>

using namespace std;

const double EPS = 1e-9;
const double PI = acos (-1.0);

//0D Objects: Points
struct point{
     double x, y;

     point () {x = y = 0.0;}
     point (double _x, double _y) : x(_x), y(_y) {}

     bool operator < (const point p)const{//sobrecargar el operador menor que
        if (fabs (x - p.x) > EPS) //util para ordenar
           return x < p.x;
        return y < p.y;
     }

     bool operator == (const point p)const{
        return (fabs (x - p.x) < EPS && fabs (y - p.y) < EPS);
     }
};

double dist (point p1, point p2){ //distancia euclidiana
              //hypot(dx, dy) returns sqrt(dx * dx + dy * dy)
     return hypot (p1.x - p2.x, p1.y - p2.y);
}
```

```
//ratate p by theta degrees CCW origin (0, 0)
point rotate (point p, double theta){
     double rad = theta * PI / 180.0;
     return point (p.x * cos (rad) - p.y * sin (rad),
                p.x * sin (rad) + p.y * cos (rad));
}

//1D Objects line
//b = 1 for no vertical line and b = 0 for vertical line
struct line{
     double a, b, c; //a way to represent line a*x + b*x + c = 0
};

//the answer is stored in the third parameter (pass by reference )
void pointsToLine (point p1, point p2, line &l){
    if (fabs (p1.x - p2.x) < EPS){ //vertical line is find
       l.a = 1.0; l.b = 0.0; l.c = -p1.x; //default values
    }
    else{
       l.a = -(double)(p1.y - p2.y) / (p1.x - p2.x);
       l.b = 1.0; //Important: we fix the value of b to 1.0
       l.c = -(double)(l.a * p1.x) - p1.y;
    }
}

bool areParallel (line l1, line l2){ //check coefficients a & b
    return (fabs (l1.a - l2.a) < EPS) && (fabs (l1.b - l2.b) < EPS);
}

bool areSame (line l1, line l2){ //also check coefficient c
    return areParallel (l1, l2) && (fabs (l1.c - l2.c) < EPS);
}

//returns true (+ intersection point) if two lines are intersect
bool areIntesect (line l1, line l2, point &p){
    if (areParallel (l1, l2)) return false; //no intersection
    //solve system of 2 linear algebraic equations with 2 unknows
    p.x = (l2.b * l1.c - l1.b * l2.c) / (l2.a * l1.b - l1.a * l2.b);
    //special case: test for vertical lini to avoid division by zero
    if (fabs (l1.b) > EPS) p.y = -(l1.a * p.x + l1.c);
     else p.y = -(l2.a * p.x + l2.c);
}

struct vec{
     double x, y;
     vec (double _x, double _y) : x(_x), y(_y) {}
};

vec toVec(point a, point b){ //convert 2 point to vector a->b
   return vec (b.x - a.x, b.y - a.y);
}

vec scale (vec v, double s){ //nonnegative s = [<1 .. 1 .. >1]
   return vec (v.x * s, v.y * s); //shorter.same.longer
}

point translate (point p, vec v){ //translate p according to v
     return point (p.x + v.x, p.y + v.y);
}

double dot (vec a, vec b){
     return a.x * b.x + a.y * b.y;
}

double norm_sq (vec v){
     return v.x * v.x + v.y * v.y;
}

//returns the distance from p to the line defined by
//two points a and b (a and b must be different)
//the closest point is stored in the 4th parameter (byref)
double distToLine (point p, point a, point b, point &c){
     //formula c = a + u * ab
     vec ap = toVec (a, p), ab = toVec (a, b);
     double u = dot (ap, ab) / norm_sq (ab);
     c = translate (a, scale (ab, u)); //translate a to c
     return dist (p, c); //Euclidean distance between p and c
}

//return the distance from p to the line segment ab defined by
//two points a and b (still OK if a == b)
//the closest point is stored in the 4th parameter (byref)
double distToLineSegment (point p, point a, point b, point &c){
     vec ap = toVec (a, p), ab = toVec (a, b);
     double u = dot (ap, ab) / norm_sq (ab);
     if (u < 0.0){ //closer to a
        c = point (a.x, a.y);
        return dist (p, a);//Euclidean distance between p and a
     }
     if (u > 1.0){ //closer to b
        c = point (b.x, b.y);
        return dist (p, b);//Euclidean distance between p and b
     }
     return distToLine (p, a, b, c);//run distToLine as aboe
```

```
}

double angle (point a, point o, point b){//returns angle aob in rad
     vec oa = toVec (o, a), ob = toVec (o, b);
     return acos (dot (oa, ob) / sqrt (norm_sq (oa) * norm_sq (ob)));
}

double cross (vec a, vec b){
     return a.x * b.y - a.y * b.x;
}

//note: to accept collinear points, we have to change the ' > 0'
//return true if point r is on the left side of line pq
bool ccw (point p, point q, point r){
    return cross (toVec (p, q), toVec (p, r)) > 0;
}
```

5.6. Stanford.

```
#include <iostream>
#include <vector>
#include <cmath>
#include <cassert>

using namespace std;

double INF = 1e100;
double EPS = 1e-12;

struct PT {
 double x, y;
 PT() {}
 PT(double x, double y) : x(x), y(y) {}
 PT(const PT &p) : x(p.x), y(p.y) {}
 PT operator + (const PT &p) const { return PT(x+p.x, y+p.y); }
 PT operator - (const PT &p) const { return PT(x-p.x, y-p.y); }
 PT operator * (double c) const { return PT(x*c, y*c ); }
 PT operator / (double c) const { return PT(x/c, y/c ); }
};

double dot(PT p, PT q) { return p.x*q.x+p.y*q.y; }
double dist2(PT p, PT q) { return dot(p-q,p-q); }
double cross(PT p, PT q) { return p.x*q.y-p.y*q.x; }
ostream &operator<<(ostream &os, const PT &p) {
 os << "(" << p.x << "," << p.y << ")";
}
```

```
//return true if point r is on the same line as the line pq
bool collinear (point p, point q, point r){
    return fabs (cross (toVec (p, q), toVec (p, r))) < EPS;
}

int main(){

   point o(0, 0), a (0, -1), b (1 , 0);
   printf ("%f\n", angle (a, o, b) / PI * 180.0);
   //p1 = rotate (p1, 180.0);
   //printf ("%f %f\n", p1.x, p1.y);

   return 0;
}
```

```
// rotate a point CCW or CW around the origin
PT RotateCCW90(PT p) { return PT(-p.y,p.x); }
PT RotateCW90(PT p) { return PT(p.y,-p.x); }
PT RotateCCW(PT p, double t) {
 return PT(p.x*cos(t)-p.y*sin(t), p.x*sin(t)+p.y*cos(t));
}

// project point c onto line through a and b
// assuming a != b
PT ProjectPointLine(PT a, PT b, PT c) {
 return a + (b-a)*dot(c-a, b-a)/dot(b-a, b-a);
}

// project point c onto line segment through a and b
PT ProjectPointSegment(PT a, PT b, PT c) {
 double r = dot(b-a,b-a);
 if (fabs(r) < EPS) return a;
 r = dot(c-a, b-a)/r;
 if (r < 0) return a;
 if (r > 1) return b;
 return a + (b-a)*r;
}

// compute distance from c to segment between a and b
double DistancePointSegment(PT a, PT b, PT c) {
 return sqrt(dist2(c, ProjectPointSegment(a, b, c)));
```

```
}

// compute distance between point (x,y,z) and plane ax+by+cz=d
double DistancePointPlane(double x, double y, double z,
                          double a, double b, double c, double d)
{
  return fabs(a*x+b*y+c*z-d)/sqrt(a*a+b*b+c*c);
}

// determine if lines from a to b and c to d are parallel or collinear
bool LinesParallel(PT a, PT b, PT c, PT d) {
  return fabs(cross(b-a, c-d)) < EPS;
}

bool LinesCollinear(PT a, PT b, PT c, PT d) {
  return LinesParallel(a, b, c, d)
     && fabs(cross(a-b, a-c)) < EPS
     && fabs(cross(c-d, c-a)) < EPS;
}

// determine if line segment from a to b intersects with
// line segment from c to d
bool SegmentsIntersect(PT a, PT b, PT c, PT d) {
  if (LinesCollinear(a, b, c, d)) {
    if (dist2(a, c) < EPS || dist2(a, d) < EPS ||
      dist2(b, c) < EPS || dist2(b, d) < EPS) return true;
    if (dot(c-a, c-b) > 0 && dot(d-a, d-b) > 0 && dot(c-b, d-b) > 0)
      return false;
    return true;
  }
  if (cross(d-a, b-a) * cross(c-a, b-a) > 0) return false;
  if (cross(a-c, d-c) * cross(b-c, d-c) > 0) return false;
  return true;
}

// compute intersection of line passing through a and b
// with line passing through c and d, assuming that unique
// intersection exists; for segment intersection, check if
// segments intersect first
PT ComputeLineIntersection(PT a, PT b, PT c, PT d) {
  b=b-a; d=c-d; c=c-a;
  assert(dot(b, b) > EPS && dot(d, d) > EPS);
  return a + b*cross(c, d)/cross(b, d);
}

// compute center of circle given three points
PT ComputeCircleCenter(PT a, PT b, PT c) {
  b=(a+b)/2;
  c=(a+c)/2;
  return ComputeLineIntersection(b, b+RotateCW90(a-b), c, c+RotateCW90(a-c));
}

// determine if point is in a possibly non-convex polygon (by William
// Randolph Franklin); returns 1 for strictly interior points, 0 for
// strictly exterior points, and 0 or 1 for the remaining points.
// Note that it is possible to convert this into an *exact* test using
// integer arithmetic by taking care of the division appropriately
// (making sure to deal with signs properly) and then by writing exact
// tests for checking point on polygon boundary
bool PointInPolygon(const vector<PT> &p, PT q) {
  bool c = 0;
  for (int i = 0; i < p.size(); i++){
    int j = (i+1)%p.size();
    if ((p[i].y <= q.y && q.y < p[j].y ||
      p[j].y <= q.y && q.y < p[i].y) &&
      q.x < p[i].x + (p[j].x - p[i].x) * (q.y - p[i].y) / (p[j].y - p[i].y))
      c = !c;
  }
  return c;
}

// determine if point is on the boundary of a polygon
bool PointOnPolygon(const vector<PT> &p, PT q) {
  for (int i = 0; i < p.size(); i++)
    if (dist2(ProjectPointSegment(p[i], p[(i+1)%p.size()], q), q) < EPS)
      return true;
    return false;
}

// compute intersection of line through points a and b with
// circle centered at c with radius r > 0
vector<PT> CircleLineIntersection(PT a, PT b, PT c, double r) {
  vector<PT> ret;
  b = b-a;
  a = a-c;
  double A = dot(b, b);
  double B = dot(a, b);
  double C = dot(a, a) - r*r;
  double D = B*B - A*C;
  if (D < -EPS) return ret;
  ret.push_back(c+a+b*(-B+sqrt(D+EPS))/A);
  if (D > EPS)
    ret.push_back(c+a+b*(-B-sqrt(D))/A);
  return ret;
```

```
}

// compute intersection of circle centered at a with radius r
// with circle centered at b with radius R
vector<PT> CircleCircleIntersection(PT a, PT b, double r, double R) {
 vector<PT> ret;
 double d = sqrt(dist2(a, b));
 if (d > r+R || d+min(r, R) < max(r, R)) return ret;
 double x = (d*d-R*R+r*r)/(2*d);
 double y = sqrt(r*r-x*x);
 PT v = (b-a)/d;
 ret.push_back(a+v*x + RotateCCW90(v)*y);
 if (y > 0)
  ret.push_back(a+v*x - RotateCCW90(v)*y);
 return ret;
}

// This code computes the area or centroid of a (possibly nonconvex)
// polygon, assuming that the coordinates are listed in a clockwise or
// counterclockwise fashion. Note that the centroid is often known as
// the "center of gravity" or "center of mass".
double ComputeSignedArea(const vector<PT> &p) {
 double area = 0;
 for(int i = 0; i < p.size(); i++) {
  int j = (i+1) % p.size();
  area += p[i].x*p[j].y - p[j].x*p[i].y;
 }
 return area / 2.0;
}

double ComputeArea(const vector<PT> &p) {
 return fabs(ComputeSignedArea(p));
}

PT ComputeCentroid(const vector<PT> &p) {
 PT c(0,0);
 double scale = 6.0 * ComputeSignedArea(p);
 for (int i = 0; i < p.size(); i++){
  int j = (i+1) % p.size();
  c = c + (p[i]+p[j])*(p[i].x*p[j].y - p[j].x*p[i].y);
 }
 return c / scale;
}

// tests whether or not a given polygon (in CW or CCW order) is simple
bool IsSimple(const vector<PT> &p) {
 for (int i = 0; i < p.size(); i++) {
  for (int k = i+1; k < p.size(); k++) {
   int j = (i+1) % p.size();
   int l = (k+1) % p.size();
   if (i == l || j == k) continue;
   if (SegmentsIntersect(p[i], p[j], p[k], p[l]))
    return false;
  }
 }
 return true;
}

int main() {

 // expected: (-5,2)
 cerr << RotateCCW90(PT(2,5)) << endl;

 // expected: (5,-2)
 cerr << RotateCW90(PT(2,5)) << endl;

 // expected: (-5,2)
 cerr << RotateCCW(PT(2,5),M_PI/2) << endl;

 // expected: (5,2)
 cerr << ProjectPointLine(PT(-5,-2), PT(10,4), PT(3,7)) << endl;

 // expected: (5,2) (7.5,3) (2.5,1)
 cerr << ProjectPointSegment(PT(-5,-2), PT(10,4), PT(3,7)) << "␣"
     << ProjectPointSegment(PT(7.5,3), PT(10,4), PT(3,7)) << "␣"
     << ProjectPointSegment(PT(-5,-2), PT(2.5,1), PT(3,7)) << endl;

 // expected: 6.78903
 cerr << DistancePointPlane(4,-4,3,2,-2,5,-8) << endl;

 // expected: 1 0 1
 cerr << LinesParallel(PT(1,1), PT(3,5), PT(2,1), PT(4,5)) << "␣"
     << LinesParallel(PT(1,1), PT(3,5), PT(2,0), PT(4,5)) << "␣"
     << LinesParallel(PT(1,1), PT(3,5), PT(5,9), PT(7,13)) << endl;

 // expected: 0 0 1
 cerr << LinesCollinear(PT(1,1), PT(3,5), PT(2,1), PT(4,5)) << "␣"
     << LinesCollinear(PT(1,1), PT(3,5), PT(2,0), PT(4,5)) << "␣"
     << LinesCollinear(PT(1,1), PT(3,5), PT(5,9), PT(7,13)) << endl;

 // expected: 1 1 1 0
 cerr << SegmentsIntersect(PT(0,0), PT(2,4), PT(3,1), PT(-1,3)) << "␣"
     << SegmentsIntersect(PT(0,0), PT(2,4), PT(4,3), PT(0,5)) << "␣"
     << SegmentsIntersect(PT(0,0), PT(2,4), PT(2,-1), PT(-2,1)) << "␣"
```

```
    << SegmentsIntersect(PT(0,0), PT(2,4), PT(5,5), PT(1,7)) << endl;

 // expected: (1,2)
 cerr << ComputeLineIntersection(PT(0,0), PT(2,4), PT(3,1), PT(-1,3)) << endl;

 // expected: (1,1)
 cerr << ComputeCircleCenter(PT(-3,4), PT(6,1), PT(4,5)) << endl;

 vector<PT> v;
 v.push_back(PT(0,0));
 v.push_back(PT(5,0));
 v.push_back(PT(5,5));
 v.push_back(PT(0,5));

 // expected: 1 1 1 0 0
 cerr << PointInPolygon(v, PT(2,2)) << "␣"
    << PointInPolygon(v, PT(2,0)) << "␣"
    << PointInPolygon(v, PT(0,2)) << "␣"
    << PointInPolygon(v, PT(5,2)) << "␣"
    << PointInPolygon(v, PT(2,5)) << endl;

 // expected: 0 1 1 1 1
 cerr << PointOnPolygon(v, PT(2,2)) << "␣"
    << PointOnPolygon(v, PT(2,0)) << "␣"
    << PointOnPolygon(v, PT(0,2)) << "␣"
    << PointOnPolygon(v, PT(5,2)) << "␣"
    << PointOnPolygon(v, PT(2,5)) << endl;

 // expected: (1,6)
 // (5,4) (4,5)
 // blank line
 // (4,5) (5,4)
 // blank line
 // (4,5) (5,4)
 vector<PT> u = CircleLineIntersection(PT(0,6), PT(2,6), PT(1,1), 5);
 for (int i = 0; i < u.size(); i++) cerr << u[i] << "␣"; cerr << endl;
 u = CircleLineIntersection(PT(0,9), PT(9,0), PT(1,1), 5);
 for (int i = 0; i < u.size(); i++) cerr << u[i] << "␣"; cerr << endl;
 u = CircleCircleIntersection(PT(1,1), PT(10,10), 5, 5);
 for (int i = 0; i < u.size(); i++) cerr << u[i] << "␣"; cerr << endl;
 u = CircleCircleIntersection(PT(1,1), PT(8,8), 5, 5);
 for (int i = 0; i < u.size(); i++) cerr << u[i] << "␣"; cerr << endl;
 u = CircleCircleIntersection(PT(1,1), PT(4.5,4.5), 10, sqrt(2.0)/2.0);
 for (int i = 0; i < u.size(); i++) cerr << u[i] << "␣"; cerr << endl;
 u = CircleCircleIntersection(PT(1,1), PT(4.5,4.5), 5, sqrt(2.0)/2.0);
 for (int i = 0; i < u.size(); i++) cerr << u[i] << "␣"; cerr << endl;

 // area should be 5.0
 // centroid should be (1.1666666, 1.166666)
 PT pa[] = { PT(0,0), PT(5,0), PT(1,1), PT(0,5) };
 vector<PT> p(pa, pa+4);
 PT c = ComputeCentroid(p);
 cerr << "Area:␣" << ComputeArea(p) << endl;
 cerr << "Centroid:␣" << c << endl;

 return 0;
}
```

5.7. Pair of Intersecting Line Segments.

```
/*
e-maxx
O(N log N)
*/
#include <bits/stdc++.h>

using namespace std;

const double EPS = 1E-9;

struct pt {
      double x, y;
};

struct seg {
      pt p, q;
      int id;

      double get_y (double x) const {
            if (abs (p.x - q.x) < EPS) return p.y;
            return p.y + (q.y - p.y) * (x - p.x) / (q.x - p.x);
      }
};


inline bool intersect1d (double l1, double r1, double l2, double r2) {
      if (l1 > r1) swap (l1, r1);
      if (l2 > r2) swap (l2, r2);
```

```
      return max (l1, l2) <= min (r1, r2) + EPS;
}

inline int vec (const pt & a, const pt & b, const pt & c) {
      double s = (b.x - a.x) * (c.y - a.y) - (b.y - a.y) * (c.x - a.x);
      return abs(s)<EPS ? 0 : s>0 ? +1 : -1;
}

bool intersect (const seg & a, const seg & b) {
      return intersect1d (a.p.x, a.q.x, b.p.x, b.q.x)
            && intersect1d (a.p.y, a.q.y, b.p.y, b.q.y)
            && vec (a.p, a.q, b.p) * vec (a.p, a.q, b.q) <= 0
            && vec (b.p, b.q, a.p) * vec (b.p, b.q, a.q) <= 0;
}


bool operator< (const seg & a, const seg & b) {
      double x = max (min (a.p.x, a.q.x), min (b.p.x, b.q.x));
      return a.get_y(x) < b.get_y(x) - EPS;
}


struct event {
      double x;
      int tp, id;

      event() { }
      event (double x, int tp, int id)
            : x(x), tp(tp), id(id)
      { }

      bool operator< (const event & e) const {
            if (abs (x - e.x) > EPS) return x < e.x;
            return tp > e.tp;
      }
};

set<seg> s;
vector < set<seg>::iterator > where;

inline set<seg>::iterator prev (set<seg>::iterator it) {
      return it == s.begin() ? s.end() : --it;
}

inline set<seg>::iterator next (set<seg>::iterator it) {
      return ++it;
}

pair<int,int> solve (const vector<seg> & a) {
      int n = (int) a.size();
      vector<event> e;
      for (int i=0; i<n; ++i) {
            e.push_back (event (min (a[i].p.x, a[i].q.x), +1, i));
            e.push_back (event (max (a[i].p.x, a[i].q.x), -1, i));
      }
      sort (e.begin(), e.end());

      s.clear();
      where.resize (a.size());
      for (size_t i=0; i<e.size(); ++i){
            int id = e[i].id;
            if (e[i].tp == +1) {
                  set<seg>::iterator
                        nxt = s.lower_bound (a[id]),
                        prv = prev (nxt);
                  if (nxt != s.end() && intersect (*nxt, a[id]))
                        return make_pair (nxt->id, id);
                  if (prv != s.end() && intersect (*prv, a[id]))
                        return make_pair (prv->id, id);
                  where[id] = s.insert (nxt, a[id]);
            }
            else{
                  set<seg>::iterator
                        nxt = next (where[id]),
                        prv = prev (where[id]);
                  if (nxt != s.end() && prv != s.end() && intersect (*nxt, *prv))
                        return make_pair (prv->id, nxt->id);
                  s.erase (where[id]);
            }
      }

      return make_pair (-1, -1);
}

int main(){



   return 0;
}
```

5.8. Two Circles Common Tagents.

```
#include <bits/stdc++.h>

using namespace std;
struct pt{
      double x, y;

      pt operator- (pt p){
            pt res = { x-p.x, y-p.y };
            return res;
      }
};

struct circle : pt{
      double r;
};

struct line{
      double a, b, c;
};

const double EPS = 1E-9;

double sqr (double a) {
      return a * a;
}

void tangents (pt c, double r1, double r2, vector<line> & ans){
       double r = r2 - r1;
       double z = sqr(c.x) + sqr(c.y);
       double d = z - sqr(r);
       if (d < -EPS) return;
       d = sqrt (abs (d));
       line l;
       l.a = (c.x * r + c.y * d) / z;
       l.b = (c.y * r - c.x * d) / z;
       l.c = r1;
       ans.push_back (l);
}

vector<line> tangents (circle a, circle b){
       vector<line> ans;
       for (int i=-1; i<=1; i+=2)
              for (int j=-1; j<=1; j+=2)
                     tangents (b-a, a.r*i, b.r*j, ans);
       for (size_t i=0; i<ans.size(); ++i)
        ans[i].c -= ans[i].a * a.x + ans[i].b * a.y;

       return ans;
}

int main()
{
   cout << "Hello world!" << endl;
   return 0;
}
```

5.9. Area of Union of Triangles.

```
/*
e-maxx
*/
#include <bits/stdc++.h>

using namespace std;

struct segment{
 int x1, y1, x2, y2;
};

struct point{
 double x, y;
};

struct item{
 double y1, y2;
 int triangle_id;
};

struct line{
 int a, b, c;
};
```

```
const double EPS = 1E-7;

void intersect (segment s1, segment s2, vector<point> & res){

    line l1 = { s1.y1-s1.y2, s1.x2-s1.x1, l1.a*s1.x1+l1.b*s1.y1 },
    l2 = { s2.y1-s2.y2, s2.x2-s2.x1, l2.a*s2.x1+l2.b*s2.y1 };
    double det1 = l1.a * l2.b - l1.b * l2.a;

    if (abs (det1) < EPS) return;

    point p = { (l1.c * 1.0 * l2.b - l1.b * 1.0 * l2.c) / det1,
             (l1.a * 1.0 * l2.c - l1.c * 1.0 * l2.a) / det1 };

    if (p.x >= s1.x1-EPS && p.x <= s1.x2+EPS && p.x >= s2.x1-EPS && p.x <= s2.x2+EPS)
       res.push_back (p);
}

double segment_y (segment s, double x){
     return s.y1 + (s.y2 - s.y1) * (x - s.x1) / (s.x2 - s.x1);
}

bool eq (double a, double b) {
    return abs (a-b) < EPS;
}

vector<item> c;

bool cmp_y1_y2 (int i, int j){
    const item & a = c[i];
    const item & b = c[j];
    return a.y1 < b.y1-EPS || abs (a.y1-b.y1) < EPS && a.y2 < b.y2-EPS;
}

int main() {

   int n;
   cin >> n;
   vector<segment> a (n*3);
   for (int i=0; i<n; ++i){
       int x1, y1, x2, y2, x3, y3;
       scanf ("%d%d%d%d%d%d", &x1,&y1,&x2,&y2,&x3,&y3);
       segment s1 = { x1,y1,x2,y2 };
       segment s2 = { x1,y1,x3,y3 };
       segment s3 = { x2,y2,x3,y3 };
       a[i*3] = s1;
       a[i*3+1] = s2;
       a[i*3+2] = s3;
   }

   for (size_t i=0; i<a.size(); ++i)
       if (a[i].x1 > a[i].x2)
          swap (a[i].x1, a[i].x2), swap (a[i].y1, a[i].y2);

   vector<point> b;
   b.reserve (n*n*3);
   for (size_t i=0; i<a.size(); ++i)
       for (size_t j=i+1; j<a.size(); ++j)
           intersect (a[i], a[j], b);

   vector<double> xs (b.size());
   for (size_t i=0; i<b.size(); ++i)
       xs[i] = b[i].x;

   sort (xs.begin(), xs.end());
   xs.erase (unique (xs.begin(), xs.end(), &eq), xs.end());

   double res = 0;
   vector<char> used (n);
   vector<int> cc (n*3);
   c.resize (n*3);
   for (size_t i=0; i+1<xs.size(); ++i){
       double x1 = xs[i], x2 = xs[i+1];
       size_t csz = 0;
       for (size_t j=0; j<a.size(); ++j)
           if (a[j].x1 != a[j].x2)
              if (a[j].x1 <= x1+EPS && a[j].x2 >= x2-EPS){
                 item it = { segment_y (a[j], x1), segment_y (a[j], x2), (int)j/3 };
                 cc[csz] = (int)csz;
                 c[csz++] = it;
              }

       sort (cc.begin(), cc.begin()+csz, &cmp_y1_y2);
       double add_res = 0;
       for (size_t j=0; j<csz; ) {
           item lower = c[cc[j++]];
           used[lower.triangle_id] = true;
           int cnt = 1;
           while (cnt && j<csz) {
                char & cur = used[c[cc[j++]].triangle_id];
                cur = !cur;
                if (cur)
                   cnt++;
                else --cnt;
           }
```

```
            item upper = c[cc[j-1]];
            add_res += upper.y1 - lower.y1 + upper.y2 - lower.y2;
        }

        res += add_res * (x2 - x1) / 2;
    }

    cout.precision (8);
    cout << fixed << res;
}
```

5.10. Inscribed Circle of Convex Polygon.

```
/*
e-maxx
complexity of this algorithm is O(n log n)
Note. It is assumed that the input polygon is strictly convex, i.e., no three points are collinear.
*/
#include <bits/stdc++.h>

using namespace std;

const double EPS = 1E-9;
const double PI = acos (-1);

struct pt {
      double x, y;
      pt() { }
      pt (double x, double y) : x(x), y(y) { }
      pt operator- (const pt & p) const {
            return pt (x-p.x, y-p.y);
      }
};

double dist (const pt & a, const pt & b) {
      return sqrt ((a.x-b.x)*(a.x-b.x) + (a.y-b.y)*(a.y-b.y));
}

double get_ang (const pt & a, const pt & b) {
      double ang = abs (atan2 (a.y, a.x) - atan2 (b.y, b.x));
      return min (ang, 2*PI-ang);
}

struct line {
      double a, b, c;
      line (const pt & p, const pt & q) {
            a = p.y - q.y;
            b = q.x - p.x;
            c = - a * p.x - b * p.y;
            double z = sqrt (a*a + b*b);
            a/=z, b/=z, c/=z;
      }
};

double det (double a, double b, double c, double d) {
      return a * d - b * c;
}

pt intersect (const line & n, const line & m) {
      double zn = det (n.a, n.b, m.a, m.b);
      return pt (
            - det (n.c, n.b, m.c, m.b) / zn,
            - det (n.a, n.c, m.a, m.c) / zn
      );
}

bool parallel (const line & n, const line & m) {
      return abs (det (n.a, n.b, m.a, m.b)) < EPS;
}

double get_h (const pt & p1, const pt & p2,
      const pt & l1, const pt & l2, const pt & r1, const pt & r2)
{
      pt q1 = intersect (line (p1, p2), line (l1, l2));
      pt q2 = intersect (line (p1, p2), line (r1, r2));
      double l = dist (q1, q2);
      double alpha = get_ang (l2 - l1, p2 - p1) / 2;
      double beta = get_ang (r2 - r1, p1 - p2) / 2;
      return l * sin(alpha) * sin(beta) / sin(alpha+beta);
}

struct cmp {
      bool operator() (const pair<double,int> & a, const pair<double,int> & b) con
            if (abs (a.first - b.first) > EPS)
                  return a.first < b.first;
            return a.second < b.second;
      }
};
```

```
int main() {
      int n;
      cin >> n;
      vector<pt> p(n);
      for (int i = 0; i < n; ++i){
       pt p1;
       cin >> p1.x >> p1.y;
       p[i] = p1;
      }

      vector<int> next (n), prev (n);
      for (int i=0; i<n; ++i) {
            next[i] = (i + 1) % n;
            prev[i] = (i - 1 + n) % n;
      }

      set < pair<double,int>, cmp > q;
      vector<double> h (n);
      for (int i=0; i<n; ++i) {
            h[i] = get_h (
                  p[i], p[next[i]],
                  p[i], p[prev[i]],
                  p[next[i]], p[next[next[i]]]
            );
            q.insert (make_pair (h[i], i));
      }

      double last_time;
      while (q.size() > 2) {
            last_time = q.begin()->first;
            int i = q.begin()->second;
            q.erase (q.begin());

            next[prev[i]] = next[i];
            prev[next[i]] = prev[i];
            int nxt = next[i], nxt1 = (nxt+1)%n,
                  prv = prev[i], prv1 = (prv+1)%n;
            if (parallel (line (p[nxt], p[nxt1]), line (p[prv], p[prv1])))
                  break;

            q.erase (make_pair (h[nxt], nxt));
            q.erase (make_pair (h[prv], prv));

            h[nxt] = get_h (
                  p[nxt], p[nxt1],
                  p[prv1], p[prv],
                  p[next[nxt]], p[(next[nxt]+1)%n]
            );
            h[prv] = get_h (
                  p[prv], p[prv1],
                  p[(prev[prv]+1)%n], p[prev[prv]],
                  p[nxt], p[nxt1]
            );

            q.insert (make_pair (h[nxt], nxt));
            q.insert (make_pair (h[prv], prv));
      }

      cout << last_time << endl;
}
```

5.11. Inscribed Circle of Convex Polygon (using ternary search).

```
/*
e-maxx
Time O(N log2 C).
the required accuracy is of the order of 10^-C2
*/
#include <bits/stdc++.h>

using namespace std;

const double EPS = 1E-9;
const double INF = INT_MAX;
int steps = 60;

struct pt{
      double x, y;
};

struct line{
      double a, b, c;
};

double dist (double x, double y, line & l){
      return abs (x * l.a + y * l.b + l.c);
}

double radius (double x, double y, vector<line> & l){
```

```
    int n = (int) l.size();
    double res = INF;
    for (int i=0; i<n; ++i)
        res = min (res, dist (x, y, l[i]));
    return res;
}

double y_radius (double x, vector<pt> & a, vector<line> & l){
    int n = (int) a.size();
    double ly = INF, ry = -INF;
    for (int i=0; i<n; ++i){
        int x1 = a[i].x, x2 = a[(i+1)%n].x, y1 = a[i].y, y2 = a[(i+1)%n].y;
        if (x1 == x2) continue;
        if (x1 > x2)
           swap (x1, x2), swap (y1, y2);
        if (x1 <= x+EPS && x-EPS <= x2){
           double y = y1 + (x - x1) * (y2 - y1) / (x2 - x1);
           ly = min (ly, y);
           ry = max (ry, y);
        }
    }

    for (int sy=0; sy<steps; ++sy){
        double diff = (ry - ly) / 3;
        double y1 = ly + diff, y2 = ry - diff;
        double f1 = radius (x, y1, l), f2 = radius (x, y2, l);
        if (f1 < f2)
           ly = y1;
         else
           ry = y2;
    }

    return radius (x, ly, l);
}

int main() {

   int n;
   scanf ("%d", &n);
   vector<pt> a (n);
   for (int i = 0; i < n; ++i){
       pt p1;
       scanf ("%lf %lf", &p1.x, &p1.y);
       a[i] = p1;
   }


   vector<line> l (n);
   for (int i=0; i<n; ++i){
       l[i].a = a[i].y - a[(i+1)%n].y;
       l[i].b = a[(i+1)%n].x - a[i].x;
       double sq = sqrt (l[i].a*l[i].a + l[i].b*l[i].b);
       l[i].a /= sq, l[i].b /= sq;
       l[i].c = - (l[i].a * a[i].x + l[i].b * a[i].y);
   }

   double lx = INF, rx = -INF;
   for (int i=0; i<n; ++i){
       lx = min (lx, a[i].x);
       rx = max (rx, a[i].x);
   }

   for (int sx=0; sx<steps; sx++) {
       double diff = (rx - lx) / 3;
       double x1 = lx + diff, x2 = rx - diff;
       double f1 = y_radius (x1, a, l), f2 = y_radius (x2, a, l);
       if (f1 < f2)
          lx = x1;
        else
          rx = x2;
   }

   double ans = y_radius (lx, a, l);
   printf ("%.7lf", ans);
}
```

5.12. Test Points Belonging to Convex Polygon.

```
/*
e-maxx:
This implementation assumes that the polygon has no repeated vertices,
and the area of the polygon is non-zero.
the vertices are given in traversal order of counterclockwise (otherwise you just need to sw
the boundary of a polygon are included
Pre-prosesing O(N)
Query O(log N)
*/
#include<bits/stdc++.h>
```

```
using namespace std;

struct pt{
     int x, y;
};

struct ang{
     int a, b;
};

bool operator < (const ang & p, const ang & q){
    if (p.b == 0 && q.b == 0)
       return p.a < q.a;
    return p.a * 1ll * q.b < p.b * 1ll * q.a;
}

long long sq (pt & a, pt & b, pt & c){
    return a.x*1ll*(b.y-c.y) + b.x*1ll*(c.y-a.y) + c.x*1ll*(a.y-b.y);
}

int main() {

   int n;
   cin >> n;
   vector<pt> p (n);
   int zero_id = 0;
   for (int i=0; i<n; ++i){
       scanf ("%d%d", &p[i].x, &p[i].y);
       if (p[i].x < p[zero_id].x || p[i].x == p[zero_id].x && p[i].y < p[zero_id].y)
          zero_id = i;
   }

   pt zero = p[zero_id];
   rotate (p.begin(), p.begin()+zero_id, p.end());
   p.erase (p.begin());
   --n;

   vector<ang> a (n);
   for (int i=0; i<n; ++i) {
       a[i].a = p[i].y - zero.y;
       a[i].b = p[i].x - zero.x;
       if (a[i].a == 0)
          a[i].b = a[i].b < 0 ? -1 : 1;
   }

   int m;
   cin >> m;
   while (m--) {
        pt q; // the next request
        cin >> q.x >> q.y;
        bool in = false;
        if (q.x >= zero.x)
           if (q.x == zero.x && q.y == zero.y)
              in = true;
           else {
             ang my = { q.y - zero.y, q.x-zero.x };
             if (my.a == 0)
                my.b = my.b < 0 ? -1 : 1;

             vector<ang>::iterator it = upper_bound (a.begin(), a.end(), my);
             if (it == a.end() && my.a == a[n-1].a && my.b == a[n-1].b)
                it = a.end()-1;
             if (it != a.end() && it != a.begin()) {
                int p1 = int (it - a.begin());
                if (sq (p[p1], p[p1-1], q) <= 0)
                   in = true;
             }
           }
        puts (in ? "INSIDE" : "OUTSIDE");
   }
}
```

5.13. + Geometry.

```
struct pt {
      int64 x, y;

      pt(int64 x, int64 y) : x(x), y(y) {}

      pt operator - (const pt &p) const {
            return pt(x - p.x, y - p.y);
      }

      pt operator + (const pt &p) const {
            return pt(x + p.x, y + p.y);
      }

      pt operator * (const double &p) const {
```

```
            return pt(x * p, y * p);
      }

      pt operator / (const double &p) const {
            return pt(x / p, y / p);
      }
};

int64 cross(pt a, pt b) {
      return a.x * b.y - a.y * b.x;
}

int64 dot(pt a, pt b) {
      return a.x * b.x + a.y * b.y;
}

//this gets the cuadrant of a pt it is useful to compare angles
int getCuad(const pt &p) {
      if(p.x >= 0 && p.y >= 0)
            return 1;

      if(p.x <= 0 && p.y >= 0)
            return 2;

      if(p.x <= 0 && p.y <= 0)
            return 3;

      return 4;
}

//is a - p0 less/equal/greater than b - p0 in angle
int cmp(const pt &a, const pt &b) {
      int ca = getCuad(a - p0);
      int cb = getCuad(b - p0);

      if(ca != cb)
            return ca - cb;

      int64 c = cross(a - p0, b - p0);

      if(c > 0) //a - p0 is ccw from b (so its angle is lesser)
            return -1;

      if(c < 0)
            return 1;

      return 0; //equal angle
}

//are pt p in triangle (a, b, c), can be on border also
bool inTriangle(pt a, pt b, pt c, pt p) {
      assert(cross(b - a, c - a) > 0); //checks that a, b and c are in ccw
      return cross(p - a, b - a) <= 0 && cross(p - b, c - b) <= 0
   && cross(p - c, a - c) <= 0;
}

//Lines
typedef long long type;

struct line {
      type a, b, c;

      line(type a_ = 0, type b_ = 0, type c_ = 0) {
            a = a_, b = b_, c = c_;
            fix();
      }

      line(point p1, point p2) { //dos ptos
            a = p1.y - p2.y; //comp y del vector director
            b = p2.x - p1.x; //comp x del vector director
            c = -a * p1.x - b * p1.y; //Ax + By + C = 0 (C is neg)
            fix();
      }

      void fix() { //normalizar
            type g = __gcd(labs(a), __gcd(labs(b), labs(c)));

            if(g == 0)
                  return;

            a /= g, b /= g, c /= g;

            if(a < 0) {
                  a *= -1;
                  b *= -1;
                  c *= -1;
            }

            else if(a == 0) {
                  if(b < 0) {
                        b *= -1;
                        c *= -1;
                  }
```

```
            else if(b == 0)
                  c *= -1;
      }
}

bool operator < (const line &l) const { //ordenar
      if(a != l.a)
            return a < l.a;

      if(b != l.b)
            return b < l.b;

      return c < l.c;
}

bool operator == (const line &l) const {
      return a == l.a && b == l.b && c == l.c;
}
```

```
      bool operator != (const line &l) const {
            return !(*this == l);
      }
};

pt inter(line l1, line l2) {
      type det = l1.a * l2.b - l2.a * l1.b;

      if(det == 0) {
            //rectas paralelas
            //hacer algo
      }

      double x = (0.0 + l1.b * l2.c - l2.b * l1.c) / (0.0 + det);
      double y = (0.0 + l2.a * l1.c - l1.a * l2.c) / (0.0 + det);

  //tener en cuenta que coordenadas de pt tienen que ser double
      return pt(x, y);
}
```

6. Graphs

6.1. Centroid Decomposition.

```
vector<int> g[N];
void eadd(int u,int v)
{
 g[u].pb(v);
 g[v].pb(u);
}
int sz[N],mc[N],p[N],q[N];
bool mk[N];
int gc(int st)
{
 int b=0,e=0;
 q[e++]=st, mc[st]=0, p[st]=-1, sz[st]=1;
 while(b<e)
 {
  int u=q[b++];
  for(auto v: g[u])
  {
   if(p[u]==v||mk[v])continue;
   p[v]=u;
   mc[v]=0;
   sz[v]=1;
   q[e++]=v;
  }
 }
 for(int i=e-1;~i;--i)
 {
  int u=q[i];
  int bc=max(e-sz[u],mc[u]);
  if(2*bc<=e)
  {
   st=u;
   break;
  }
  sz[p[u]]+=sz[u];
  mc[p[u]]=max(mc[p[u]], sz[u]);
 }
 return st;
}
void solve(int u=1)
{
 // divide
 u = gc(u);

 mk[u]=true;

 // conquer

 for(auto v: g[u])
 {
  if(mk[v])continue;
  solve(v);
 }
}
```

6.2. Strongly Connected Components.

```
/* Strong Connected Components
 * Outline:
 * >> Kepp tracks of what vertex belongs
 * to whom SCC, and vice-versa.
 * >> Components are stored in reverse
 * topological order.
 * */
int deg[N],dt[N],cmp[N],t=0,cc=0;
bool mk[N];
vector<int> g[N], comp[N];
inline void eadd(int u,int v)
{g[u].pb(v);}
stack<int> stk;
int scc(int u)
{
 int lw=dt[u]=++t;
 stk.push(u);
 mk[u]=true;
 for(int v: g[u])
  if(!dt[v])
  {
   int nl=scc(v);
```

```
    lw=min(lw,nl);
   }else if(mk[v])
    lw=min(lw,dt[v]);
  if(dt[u]==lw)
  {
   ++cc;
   while(stk.top()!=u)
   {
    cmp[stk.top()]=cc;
    comp[cc].pb(stk.top());
    mk[stk.top()]=false;
    stk.pop();
   }
   cmp[stk.top()]=cc;
   comp[cc].pb(stk.top());
   mk[stk.top()]=false;
   stk.pop();
  }
  return lw;
}
```

6.3. Smaller to Large Technique* (Trees).

```
/* Smaller to Large. O(N(logN)^2)
 *
 * http://codeforces.com/contest/1042/problem/F
 * ---------- F. Leaf Sets ------------------
 * Let's call some set of leaves beautiful if
 * the maximum distance between any pair of
 * leaves in it is less or equal to k. You want
 * to split all leaves into non-intersecting
 * beautiful sets. What is the minimal number
 * of sets in such a split?
 *
 * Solution: Greedely compact for each vertex
 * the solution for it's subtree. Always taking
 * into account that we don't need each vertex
 * of a set of leafs we only need the deepest.
 */
#include <bits/stdc++.h>
using namespace std;


typedef long long int ll;
typedef long double ld;
typedef pair<int,int> pii;
typedef pair<ll,ll> pll;


#define INIT ios_base::sync_with_stdio(false);\
          cin.tie(0),cout.tie()
#define endl '\n'
#define fr first
#define sc second
#define pb push_back
#define eb emplace_back
#define mp make_pair
#define lb lower_bound
#define ub upper_bound
#define ins insert
#define ers erase
#define sz(c) ((int)(c).size())
#define all(x) (x).begin(),(x).end()
#define unique(x) (x).resize(unique(all(x))-(x).begin())
#define debug(_fmt,...) \
fprintf(stderr,"("#__VA_ARGS__ ")␣=␣(" _fmt")\n",__VA_ARGS__)


const int N = 1e6+7;


int n,k;
vector<int> g[N];
int deg[N];
void eadd(int u,int v)
{
 g[u].eb(v);
 g[v].eb(u);
 ++deg[u];
 ++deg[v];
}


// smaller to large tehnique
vector<multiset<int>> ss;
int js(int x,int y)
{
 if(x == y) return x;
 if(sz(ss[x]) < sz(ss[y])) swap (x, y);
```

```
  ss[x].ins(all(ss[y]));
  return x;
}
int dfs(int u, int d=0, int p=0)
{
  int my = sz(ss);
  ss.eb();
  if(deg[u] == 1)
  {
    ss[my].ins(d);
    return my;
  }
  for(int v: g[u])
    if(v != p) my = js(my, dfs(v,d+1,u));


  vector<int> del;
  auto it = ss[my].begin();it++;
  auto jt = ss[my].begin();
  while(it != ss[my].end())
  {
    if((*jt-d) + (*it-d) <= k) del.pb(*jt);
    else break;
    jt++,it++;
  }
  for(int i: del) ss[my].ers(ss[my].find(i));


  return my;
}
// end of technique


int main()
{
  scanf("%d%d", &n, &k);
  for(int i=1;i<n;++i)
  {
    int u,v;
    scanf("%d%d", &u, &v);
    eadd(u,v);
  }
  int rt=-1;
  for(int i=1;i<=n;++i)
    if(deg[i] > 1){rt = dfs(i);break;}
  cout << sz(ss[rt]) << endl;

  return 0;
}
//~ f
```

6.4. **Dijkstra.**

```
struct Edge
{
  int u,v;
  ll w;
  Edge(){}
  Edge(int u,int v,int w): u(u), v(v), w(w){}
  int nxt(int x){return x==u?v:u;}
};
const ll inf = numeric_limits<ll>::max();
/* funcion dijkstra:
 * g: grafo, dist: distancias minimas, st: nodo inicial
 */
void dijkstra(vector<vector<Edge*>> &g, vector<ll> &dist, int st)
{
  dist.resize(sz(g), inf);
  priority_queue<pair<ll,int>, vector<pair<ll,int>>, greater<pair<ll,int>>> pq;
  dist[st] = 0;
  pq.emplace(0, st);
  while(sz(pq))
  {
    int u = pq.top().sc;
    ll ct = pq.top().fr;
    pq.pop();
    if(dist[u] != ct) continue;
    dist[u] = ct;
    for(Edge* &to: g[u])
    {
      int v = to->nxt(u);
      if(dist[v] > dist[u] + to->w)
      {
        dist[v] = dist[u] + to->w;
        pq.emplace(dist[v], v);
      }
    }
  }
}
```

```
// modo de empleo, m - tot. aristas, n - tot. nodos
vector<Edge> G(m);
vector<vector<Edge*>> g(n+1);
for(Edge &e: G) g[e.u].eb(&e), g[e.v].eb(&e);
```

6.5. Lowest Common Ancestor (lca).

```
int pp[N][19], lv[N];
ll cost[N][19];
// construir el lca del arbol T
void blca(vector<vector<pair<ll,int>>> &T)
{
 queue<int> q;
 q.push(1);
 while(sz(q))
 {
  int u = q.front();
  q.pop();

  for(auto &to: T[u])
  {
   if(to.sc == pp[u][0])continue;
   lv[to.sc] = lv[u] + 1;
   cost[to.sc][0] = to.fr;
   pp[to.sc][0] = u;
   q.push(to.sc);
  }
 }


 for(int i=1;i<19;++i)
  for(int j=1;j<sz(T);++j)
   pp[j][i] = pp[pp[j][i-1]][i-1],
   cost[j][i] = cost[j][i-1] + cost[pp[j][i-1]][i-1];
}
// distancia u -- v en el arbol
ll distT(int u,int v)
{
 ll res=0;
 if(lv[u] < lv[v]) swap(u,v);
 for(int i=18;~i;--i)
  if(lv[pp[u][i]] >= lv[v])
  {
   res += cost[u][i];
   u = pp[u][i];
  }
 if(u == v) return res;
 assert(lv[u] == lv[v]);
 for(int i=18;~i;--i)
  if(pp[u][i] != pp[v][i])
  {
   res += cost[u][i];
   res += cost[v][i];
   u = pp[u][i];
   v = pp[v][i];
  }
 res += cost[u][0];
 res += cost[v][0];
 return res;
}
```