1.ddp
2.猫树
3.dancing link x
4.alpha-beta 剪枝
5.矩阵树定理
6.计算几何
7.李超线段树

1.ddp

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

using namespace std;

typedef long long LL;

const int maxn = 500010;
const int INF = 0x3f3f3f3f;

int Begin[maxn], Next[maxn], To[maxn], e, n, m;
int size[maxn], son[maxn], top[maxn], fa[maxn], dis[maxn], p[maxn], id[maxn],
    End[maxn];
// p[i]表示i树剖后的编号，id[p[i]] = i
int cnt, tot, a[maxn], f[maxn][2];

struct matrix {
  int g[2][2];
  matrix() { memset(g, 0, sizeof(g)); }
  matrix operator*(const matrix &b) const  // 重载矩阵乘
  {
    matrix c;
    for (int i = 0; i <= 1; i++)
      for (int j = 0; j <= 1; j++)
        for (int k = 0; k <= 1; k++)
          c.g[i][j] = max(c.g[i][j], g[i][k] + b.g[k][j]);
    return c;
  }
} Tree[maxn], g[maxn];  // Tree[]是建出来的线段树，g[]是维护的每个点的矩阵

inline void PushUp(int root) {
  Tree[root] = Tree[root << 1] * Tree[root << 1 | 1];
}

inline void Build(int root, int l, int r) {
  if (l == r) {
    Tree[root] = g[id[l]];
    return;
  }
  int Mid = l + r >> 1;
  Build(root << 1, l, Mid);
  Build(root << 1 | 1, Mid + 1, r);
  PushUp(root);
}

inline matrix Query(int root, int l, int r, int L, int R) {
  if (L <= l && r <= R) return Tree[root];
  int Mid = l + r >> 1;
  if (R <= Mid) return Query(root << 1, l, Mid, L, R);
  if (Mid < L) return Query(root << 1 | 1, Mid + 1, r, L, R);
  return Query(root << 1, l, Mid, L, R) *
         Query(root << 1 | 1, Mid + 1, r, L, R);
  // 注意查询操作的书写
```

```
}

inline void Modify(int root, int l, int r, int pos) {
  if (l == r) {
    Tree[root] = g[id[l]];
    return;
  }
  int Mid = l + r >> 1;
  if (pos <= Mid)
    Modify(root << 1, l, Mid, pos);
  else
    Modify(root << 1 | 1, Mid + 1, r, pos);
  PushUp(root);
}

inline void Update(int x, int val) {
  g[x].g[1][0] += val - a[x];
  a[x] = val;
  // 首先修改x的g矩阵
  while (x) {
    matrix last = Query(1, 1, n, p[top[x]], End[top[x]]);
    // 查询top[x]的原本g矩阵
    Modify(1, 1, n,
           p[x]);  // 进行修改(x点的g矩阵已经进行修改但线段树上的未进行修改)
    matrix now = Query(1, 1, n, p[top[x]], End[top[x]]);
    // 查询top[x]的新g矩阵
    x = fa[top[x]];
    g[x].g[0][0] +=
        max(now.g[0][0], now.g[1][0]) - max(last.g[0][0], last.g[1][0]);
    g[x].g[0][1] = g[x].g[0][0];
    g[x].g[1][0] += now.g[0][0] - last.g[0][0];
    // 根据变化量修改fa[top[x]]的g矩阵
  }
}

inline void add(int u, int v) {
  To[++e] = v;
  Next[e] = Begin[u];
  Begin[u] = e;
}

inline void DFS1(int u) {
  size[u] = 1;
  int Max = 0;
  f[u][1] = a[u];
  for (int i = Begin[u]; i; i = Next[i]) {
    int v = To[i];
    if (v == fa[u]) continue;
    dis[v] = dis[u] + 1;
    fa[v] = u;
    DFS1(v);
    size[u] += size[v];
    if (size[v] > Max) {
      Max = size[v];
      son[u] = v;
    }
    f[u][1] += f[v][0];
    f[u][0] += max(f[v][0], f[v][1]);
```

```
    // DFS1过程中同时求出f[i][0/1]
  }
}

inline void DFS2(int u, int t) {
  top[u] = t;
  p[u] = ++cnt;
  id[cnt] = u;
  End[t] = cnt;
  g[u].g[1][0] = a[u];
  g[u].g[1][1] = -INF;
  if (!son[u]) return;
  DFS2(son[u], t);
  for (int i = Begin[u]; i; i = Next[i]) {
    int v = To[i];
    if (v == fa[u] || v == son[u]) continue;
    DFS2(v, v);
    g[u].g[0][0] += max(f[v][0], f[v][1]);
    g[u].g[1][0] += f[v][0];
    // g矩阵根据f[i][0/1]求出
  }
  g[u].g[0][1] = g[u].g[0][0];
}

int main() {
  scanf("%d%d", &n, &m);
  for (int i = 1; i <= n; i++) scanf("%d", &a[i]);
  for (int i = 1; i <= n - 1; i++) {
    int u, v;
    scanf("%d%d", &u, &v);
    add(u, v);
    add(v, u);
  }
  dis[1] = 1;
  DFS1(1);
  DFS2(1, 1);
  Build(1, 1, n);
  for (int i = 1; i <= m; i++) {
    int x, val;
    scanf("%d%d", &x, &val);
    Update(x, val);
    matrix ans = Query(1, 1, n, 1, End[1]);  // 查询1所在重链的矩阵乘
    printf("%d\n", max(ans.g[0][0], ans.g[1][0]));
  }
  return 0;
}
```

2.猫树

例：给出一个区间，每次询问一个区间 [l,r][l,r] 的所有子区间的区间最小值之和。

```
#include <cstdio>
#include <algorithm>
#include <cstring>
#include <cassert>
#include <cmath>
#include <iostream>
using namespace std;
#define RG register
#define IL __inline__ __attribute__((always_inline))
#define For(i, a, b) for(RG int i = a, ___u = b; i <= ___u; ++i)
#define Dor(i, a, b) for(RG int i = b, ___d = a; i >= ___d; --i)
#define Rep(i, a, b) for(RG int i = a, ___u = b; i != ___u; ++i)
#define dmin(a, b) ((a) < (b) ? (a) : (b))
#define dmax(a, b) ((a) > (b) ? (a) : (b))
#define cmin(a, b) ((a) > (b) ? (a) = (b) : 1)
#define cmax(a, b) ((a) < (b) ? (a) = (b) : 1)
#define ddel(a, b) ((a) > (b) ? (a) - (b) : (b) - (a))
#define dabs(i) ((i) > 0 ? (i) : -(i))
typedef long long ll;
typedef unsigned uint;
typedef unsigned long long ull;
typedef long double ld;

#include <queue>
#include <vector>
const int MaxN = 131072;
    int a[MaxN], N;
    // 对单调栈中的数进行归并，并确定每一位对应的单调栈中元素
    struct Node
    {
        int m;
        struct Data
        {
            int pos;            // 在左右单调栈中的排名
            ll sum, ans;        // 答案的和
            ll pre;                // 到中点这一段的和
        } *pre;
        IL ll query(RG int l, RG int r)
        {
            RG int pl = pre[l].pos, pr = pre[r].pos;
            return pre[l].pre + pre[r].pre + ((pl < pr)
                ? pre[l].ans + (m - l)       * (pre[r].sum - (pl == m - l
 ? 0 : pre[l + pl - 1].sum))
                : pre[r].ans + (r - m + 1) * (pre[l].sum - (pr == r - m + 1 ? 0
: pre[r - pr + 1].sum)));
        }
    } T[262144];
    void build(RG const int i, RG const int l, RG const int r)
    {
        if(l + 1 == r || N <= l) return;
```

```
        RG const int m = T[i].m = (l + r) >> 1;
        build(i << 1, l, m);
        build(i << 1 | 1, m, r);

        RG Node::Data *f = T[i].pre = (new Node::Data[r - l]) - l;

        static int stack[MaxN], b[MaxN];
        RG int top, cur, V; RG ll pre, ran, sum;
        pre = sum = top = ran = 0; cur = 2000000000; *stack = m;
        for(RG int i = m - 1; i >= l; --i)
        {
            V = a[i];
            for(; top && a[stack[top]] >= V; --top)
                ran -= (ll) a[stack[top]] * (stack[top - 1] - stack[top]);
            ran += (ll) V * (stack[top] - i);
            stack[++top] = i;
            cmin(cur, V);
            f[i].sum = (sum += (b[i] = cur));
            f[i].pre = (pre += ran);
        }
        pre = sum = top = ran = 0; cur = 2000000000; *stack = m - 1;
        for(RG int i = m; i <= r - 1; ++i)
        {
            V = a[i];
            for(; top && a[stack[top]] >= V; --top)
                ran -= (ll) a[stack[top]] * (stack[top] - stack[top - 1]);
            ran += (ll) V * (i - stack[top]);
            stack[++top] = i;
            cmin(cur, V);
            f[i].sum = (sum += b[i] = cur);
            f[i].pre = (pre += ran);
        }
        sum = 0;
        RG int pl = m - 1, pr = m, N = 0, tmp;
        while(l <= pl || pr < r)
            (pr >= r || (l <= pl && b[pl] > b[pr]))
                ? (f[pl].pos = ++N, (tmp = (N - (m - pl)))         ? sum += (ll)
b[pl] * tmp : 0, f[pl--].ans = sum)
                : (f[pr].pos = ++N, (tmp = (N - (pr - m + 1)))     ? sum += (ll)
b[pr] * tmp : 0, f[pr++].ans = sum);
    }

    IL void main()
    {
        RG int (*F)() = read<int>;
        RG int n = N = F(), m = F(), D = 1;
        while(D < n) D <<= 1;
        Rep(i, 0, n) a[i] = F();
        build(1, 0, D);
        RG int l, r, v, d;
        static int pre[1024];
        Rep(i, 1, 1024) pre[i] = pre[i >> 1] + 1;
        while(m--)
        {
            l = F() - 1, r = F() - 1;
            if(l == r) Print(a[l]);
            else
            {
```

```
                l += D, r += D;
                v = l ^ r;
                d = (v < 1024 ? pre[v] : 10 + pre[v >> 10]);
                Print(T[l >> d].query(l - D, r - D));
            }
        }
        io::flush();
    }
```

3.dancing link x

```
#include <bits/stdc++.h>
const int N = 1e6 + 10;
int ans[10][10], stk[N];
inline int read() {
  register int x = 0, f = 0, ch;
  while (!isdigit(ch = getchar())) f |= ch == '-';
  while (isdigit(ch)) x = (x << 1) + (x << 3) + (ch ^ 48), ch = getchar();
  return f ? -x : x;
}  //快读
struct DLX {
  static const int MAXSIZE = 1e5 + 10;
  int n, m, tot, first[MAXSIZE + 10], siz[MAXSIZE + 10];
  int L[MAXSIZE + 10], R[MAXSIZE + 10], U[MAXSIZE + 10], D[MAXSIZE + 10];
  int col[MAXSIZE + 10], row[MAXSIZE + 10];
  void build(const int &r, const int &c) {  //进行build操作
    n = r, m = c;
    for (register int i = 0; i <= c; ++i) {
      L[i] = i - 1, R[i] = i + 1;
      U[i] = D[i] = i;
    }
    L[0] = c, R[c] = 0, tot = c;
    memset(first, 0, sizeof(first));
    memset(siz, 0, sizeof(siz));
  }
  void insert(const int &r, const int &c) {  //进行insert操作
    col[++tot] = c, row[tot] = r, ++siz[c];
    D[tot] = D[c], U[D[c]] = tot, U[tot] = c, D[c] = tot;
    if (!first[r])
      first[r] = L[tot] = R[tot] = tot;
    else {
      R[tot] = R[first[r]], L[R[first[r]]] = tot;
      L[tot] = first[r], R[first[r]] = tot;
    }
  }
  void remove(const int &c) {  //进行remove操作
    register int i, j;
    L[R[c]] = L[c], R[L[c]] = R[c];
    for (i = D[c]; i != c; i = D[i])
      for (j = R[i]; j != i; j = R[j])
        U[D[j]] = U[j], D[U[j]] = D[j], --siz[col[j]];
  }
  void recover(const int &c) {  //进行recover操作
    register int i, j;
    for (i = U[c]; i != c; i = U[i])
      for (j = L[i]; j != i; j = L[j]) U[D[j]] = D[U[j]] = j, ++siz[col[j]];
    L[R[c]] = R[L[c]] = c;
  }
  bool dance(int dep) {  // dance
    register int i, j, c = R[0];
    if (!R[0]) {
      for (i = 1; i < dep; ++i) {
        int x = (stk[i] - 1) / 9 / 9 + 1;
        int y = (stk[i] - 1) / 9 % 9 + 1;
        int v = (stk[i] - 1) % 9 + 1;
```

```
        ans[x][y] = v;
      }
      return 1;
    }
    for (i = R[0]; i != 0; i = R[i])
      if (siz[i] < siz[c]) c = i;
    remove(c);
    for (i = D[c]; i != c; i = D[i]) {
      stk[dep] = row[i];
      for (j = R[i]; j != i; j = R[j]) remove(col[j]);
      if (dance(dep + 1)) return 1;
      for (j = L[i]; j != i; j = L[j]) recover(col[j]);
    }
    recover(c);
    return 0;
  }
} solver;
int GetId(int row, int col, int num) {
  return (row - 1) * 9 * 9 + (col - 1) * 9 + num;
}
void Insert(int row, int col, int num) {
  int dx = (row - 1) / 3 + 1;
  int dy = (col - 1) / 3 + 1;
  int room = (dx - 1) * 3 + dy;
  int id = GetId(row, col, num);
  int f1 = (row - 1) * 9 + num;             // task 1
  int f2 = 81 + (col - 1) * 9 + num;        // task 2
  int f3 = 81 * 2 + (room - 1) * 9 + num;   // task 3
  int f4 = 81 * 3 + (row - 1) * 9 + col;    // task 4
  solver.insert(id, f1);
  solver.insert(id, f2);
  solver.insert(id, f3);
  solver.insert(id, f4);
}
int main() {
  solver.build(729, 324);
  for (register int i = 1; i <= 9; ++i)
    for (register int j = 1; j <= 9; ++j) {
      ans[i][j] = read();
      for (register int v = 1; v <= 9; ++v) {
        if (ans[i][j] && ans[i][j] != v) continue;
        Insert(i, j, v);
      }
    }
  solver.dance(1);
  for (register int i = 1; i <= 9; ++i, putchar('\n'))
    for (register int j = 1; j <= 9; ++j, putchar(' ')) printf("%d", ans[i][j]);
  return 0;
}
```

4.alpha-beta 剪枝

```
int alpha_beta(int u, int alph, int beta, bool is_max) {
  if (!son_num[u]) return val[u];
  if (is_max) {
    for (int i = 0; i < son_num[u]; ++i) {
      int d = son[u][i];
      alph = max(alph, alpha_beta(d, alph, beta, is_max ^ 1));
      if (alph >= beta) break;
    }
    return alph;
  } else {
    for (int i = 0; i < son_num[u]; ++i) {
      int d = son[u][i];
      beta = min(beta, alpha_beta(d, alph, beta, is_max ^ 1));
      if (alph >= beta) break;
    }
    return beta;
  }
}
```

5.矩阵树定理

- 无向图

度数矩阵D：若存在边(x，y，z)，则 D[x] [x] + = z ; D[y] [y] + = z ;

邻接矩阵A：若存在边(x，y，z)，则 A[x] [y] + = z ; A[y] [x] + = z ;

基尔霍夫矩阵 K = D - A

删去任意一行和任意一列，求矩阵行列式即可

- 有向图

度数矩阵D：若存在边(x，y，z)，则外向树中 D[y] [y] + = z; 内向树中 D[x] [x] + = z;

邻接矩阵A：若存在边(x，y，z)，则内向树和外向树中均为 A[x] [y] + = z ;

删去指定根所在的行和列，求矩阵行列式即可

这里只放求行列式的代码

```
#include<bits/stdc++.h>
#define INL inline
#define ll long long
using namespace std;
const int N=605;
int n,a[N][N],MOD;
INL int sol(){
    int res=1,w=1;
    for(int i=1;i<=n;i++){
        for(int j=i+1;j<=n;++j){
            while(a[i][i]){
                int div=a[j][i]/a[i][i];
                for(int k=i;k<=n;++k){
                    a[j][k]=(a[j][k]-1ll*div*a[i][k]%MOD+MOD)%MOD;
                }
                swap(a[i],a[j]);w=-w;
            }//对第 i 行和第 j 行做辗转相减。
            swap(a[i],a[j]);w=-w;
        }
    }
    for(int i=1;i<=n;i++)res=1ll*a[i][i]*res%MOD;
    res=1ll*w*res;
    return (res+MOD)%MOD;//经 典 模 加 模
}
int main(){
    n=read(),MOD=read();
    for(int i=1;i<=n;i++)
        for(int j=1;j<=n;j++)
            a[i][j]=read();
    int ans=sol();printf("%d\n",ans);
    return 0;
}
```

6.计算几何

```
#include <bits/stdc++.h>
#define Clr(a,b) memset(a,b,sizeof(a));
#define Fill(a,b) generate(a.begin(), a.end(), [](){return b;})
#define Count(a,b) count_if(a.begin(), a.end(), [](int a##_param){return
a##_param == b;})
#define mp(a,b) make_pair(a,b)
#define Rep(i,a,b) for(int i = a, i##_end = b; i <= i##_end; i ++)
#define Cnt(i,a,b) for(int i = a, i##_end = b; i >= i##_end; i --)

#define PREC 2

using namespace std;

typedef long long ll;
typedef unsigned long long ull;
typedef pair<int,int> pii;
template<typename T> T gcd(T a, T b) {return b?gcd(b,a%b):a;}

template<typename T> T sqr(T a) {return a*a;}



typedef double GeometryType;

struct vec {
    typedef GeometryType T;
    T x, y;
    vec () {}
    vec (T x, T y): x(x), y(y) {}
    vec operator + (const vec& a) const {
        return vec(x + a.x, y + a.y);
    }
    vec operator - (const vec& a) const {
        return vec(x - a.x, y - a.y);
    }
    vec operator * (const T& a) const {
        return vec(a*x, a*y);
    }
    friend vec operator * (const T& a, const vec& b) {
        return vec(a*b.x, a*b.y);
    }
    vec operator / (const T& a) const {
        return vec(x/a, y/a);
    }
    T operator * (const vec& a) const {
        return x*a.x + y*a.y;
    }
    T operator / (const vec& a) const {
        return x*a.y - y*a.x;
    }
    T sqr() const {
        return (*this)*(*this);
    }
    T mag() const {
```

```
        return sqrt(this->sqr());
    }
    friend istream& operator >> (istream& is, vec& v) {
        return is >> v.x >> v.y;
    }
    friend ostream& operator << (ostream& os, const vec& v) {
        return os << "(" << v.x << "," << v.y << ")";
    }
    #define COMP_COORD
    //#define COMP_ANGLE
    #ifdef COMP_COORD
    bool operator < (const vec& b) const {
        if(x == b.x) return y < b.y;
        return x < b.x;
    }
    bool operator == (const vec& b) const {
        return x == b.x && y == b.y;
    }
    #endif //COMP_COORD

    #ifdef COMP_ANGLE
    float quad() const {
        if(x == 0 && y == 0) return 0;
        if(y == 0){
            if(x > 0) return 0.5;
            else return 2.5;
        }
        if(x == 0) {
            if(y > 0) return 1.5;
            else return 3.5;
        }
        if(x > 0) {
            if(y > 0) return 1;
            else return 4;
        } else {
            if(y > 0) return 2;
            else return 3;
        }
    }
    bool cmp(const vec& i) const {
        float a = this->quad(), b = i.quad();
        if(a != b) return a < b;
        return this->cross(i) > 0;
    }
    bool same(const vec& i) const {
        return this->quad() == i.quad() && this->cross(i) == 0;
    }
    bool operator < (const vec& i) const {
        return this->cmp(i);
    }
    bool operator > (const vec& i) const {
        return i < (*this);
    }
    bool operator == (const vec& i) const {
        return same(i);
    }
    #endif // COMP_ANGLE
};
```

```
struct seg {
    typedef GeometryType T;
    vec p, v;
    seg() {}
    seg(vec p1, vec p2): p(p1), v(p2-p1) {}
    T mag() {
        return v.mag();
    }
    T dist(vec t) const {
        return abs((t-p)/v)/t.mag();
    }
    T dist2 (vec t) const {
        return sqr((t-p)/v)/t.sqr();
    }
    bool parallel(const seg& b) const {
        return ((this->v)/(b.v)) == 0;
    }
    vec inter(const seg& b) const {
        return (this->p)+(this->v)*((b.v/(this->p - b.p))/(this->v/b.v));
    }
    vec operator * (const seg& b) const {
        return this->inter(b);
    }
    bool between(const seg& b) const {
        T a = v/(b.p - p), z = v/((b.p + b.v) - p);
        if(a == 0 || z == 0) return true;
        return (a<0) ^ (z<0);
    }
    bool operator / (const seg& b) const{
        return this->between(b) && b.between(*this);
    }
};

struct poly : vector<vec> {
    typedef GeometryType T;

    void add(vec a) {
        this->push_back(a);
    }
    template<typename... T> void add(vec a, T... args) {
        this->push_back(a);
        add(args...);
    }

    poly() {}
    template<typename... T> poly(T... args) {
        add(args...);
    }

    T area() {
        T r = 0;
        Rep(i,0,this->size()-2) {
            r += (*this)[i]/(*this)[i+1];
        }
        r += ((*this)[this->size()-1]/(*this)[0]);
        return abs(r)/2;
    }
```

```
    #ifdef COMP_COORD
    poly convex() {
        sort(this->begin(), this->end());
        poly r;
        Rep(i,0,this->size()-1) {
            while(r.size() >= 2 &&
                ((*this)[i]-r[r.size()-2])/(r[r.size()-1]-r[r.size()-2])>=0)
                r.pop_back();
            r.add((*this)[i]);
        }
        int m = r.size();
        Cnt(i,this->size()-2,0) {
            while(r.size() >= (size_t)(1 + m) &&
                ((*this)[i]-r[r.size()-2])/(r[r.size()-1]-r[r.size()-2])>=0)
                r.pop_back();
            r.add((*this)[i]);
        }
        r.pop_back();
        return r;
    }
    #endif //COMP_COORD

    #ifdef COMP_ANGLE

    #endif
};

struct dyconvex {
private:
    set<vec> s;
public:
    dyconvex(){}
    void init(vec* p)
    /**
    only allow 3 points in the array p[]:p[0]~p[2]
    */
    {
        for(int i = 0;i < 3;i ++)
            s.insert(p[i]);
        int cnt = 0;
        for(auto i = s.begin();i!=s.end();i ++)
            p[cnt++] = *i;
        if( (p[2]-p[0])/(p[1]-p[0]) >= 0 )
            s.erase( s.find(p[1]) );
    }
    bool query(vec p) {
        auto it = s.lower_bound(p);
        if(it==s.end())return false;
        if(it!=s.begin()) {
            auto ft = it;ft--;
            vec v1 = (*it)-(*ft),v2 = p-(*ft);
            if(v1/v2>=0)return true;
            else return false;
        }
        else {
            if((*it)==p)return true;
            return false;
        }
```

```
    }
    void insert(vec p)
    {
        s.insert(p);
        auto it = s.find(p);
        auto ft = it, bk = it;
        if(ft!=s.begin())
        {
            ft--;
            while(ft!=s.begin())
            {
                auto ff = ft;ff--;
                vec v1 = *it-*ff,v2 = *ft-*ff;
                if(v1/v2>=0)
                    s.erase(ft);
                else break;
                ft = ff;
            }
        }
        if(bk!=s.end())
        {
            bk++;
            while(bk!=s.end())
            {
                auto bb = bk;bb++;
                if(bb==s.end())break;
                vec v1 = *bb-*it,v2 = *bk-*it;
                if(v1/v2>=0)
                    s.erase(bk);
                else break;
                bk = bb;
            }
        }
    }
};

poly HalfPlane(vector<seg> v) {
    poly r;
    return r;
}

signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    cout << fixed;
    cout << setprecision(PREC);

    // Sample
    vec a(0,0), b(0,1), c(1,0), d(1,1);

    poly p(a,b,c,d);
    p = p.convex();
    cout << p.area() << endl;

    cin >> a;
    cout << a;
```

```
    return 0;
}
```

7.李超线段树

> 你要资磁动态维护一个平面直角坐标系，资磁在中间插入一条线段，资磁询问与x=x0这条直线相交的所有线段中，交点的y轴坐标的最大(小)值。

我们要维护的东西是这个：维护这个区间内的所有直线中，从上往下能够看到的最长的那个线段，也就是没有被其他直线覆盖长度最大的段。

考虑怎么插入一条直线，假设它当前处理到了某个区间：

- 如果这个区间没有记录最长的线段，那么直接把这个区间记录的线段修改为这条线段，然后返回。
- 如果当前线段在这个区间内已经被这个区间内的最长线段为覆盖，那么直接gg，返回就好了。
- 反过来，如果完全覆盖了之前记录的线段，那么直接赋值、返回。
- 否则和已经记录的直线有交，判断哪根线段覆盖的区域较长，把这个区间记录的值给修改一下，然后把短的那一半丢下去递归。

这样子复杂度是啥呢？显然维护复杂度看起来不太对的就是最后一项，但是不难证明每次递归下去直线长度都至少要减少一半，所以这个东西的复杂度就是一个log的。
至于询问？那就是单点询问啦，在线段树上一直走到这个单点为之，把路径上所有记录的线段拿出来取一个max就好啦。

```
#include<iostream>
#include<cstdio>
#include<cmath>
using namespace std;
#define MAX 100100
#define lson (now<<1)
#define rson (now<<1|1)
inline int read()
{
    int x=0;bool t=false;char ch=getchar();
    while((ch<'0'||ch>'9')&&ch!='-')ch=getchar();
    if(ch=='-')t=true,ch=getchar();
    while(ch<='9'&&ch>='0')x=x*10+ch-48,ch=getchar();
    return t?-x:x;
}
int N=100000;
struct Node
{
    bool fl;int id;double k,b;
    void upd(int _id,double _k,double _b){id=_id,k=_k;b=_b;}
}t[MAX<<2];
double K[MAX],B[MAX];
void Modify(int now,int l,int r,int id,double k,double b)
{
    if(!t[now].fl){t[now].fl=true;t[now].upd(id,k,b);return;}
    int mid=(l+r)>>1;
    double l1=l*t[now].k+t[now].b,r1=r*t[now].k+t[now].b;
    double l2=l*k+b,r2=r*k+b;
    if(l1>=l2&&r1>=r2)return;
    if(l2>l1&&r2>r1){t[now].upd(id,k,b);return;}
    double x=(t[now].b-b)/(k-t[now].k);
    if(x<=mid)
    {

if(l1>l2)Modify(lson,l,mid,t[now].id,t[now].k,t[now].b),t[now].upd(id,k,b);
```

```
        else Modify(lson,l,mid,id,k,b);
    }
    else
    {
        if(l1>l2)Modify(rson,mid+1,r,id,k,b);
        else
Modify(rson,mid+1,r,t[now].id,t[now].k,t[now].b),t[now].upd(id,k,b);
    }
}
void Modify(int now,int l,int r,int L,int R,int id,double k,double b)
{
    if(L<=l&&r<=R){Modify(now,l,r,id,k,b);return;}
    int mid=(l+r)>>1;
    if(L<=mid)Modify(lson,l,mid,L,R,id,k,b);
    if(R>mid)Modify(rson,mid+1,r,L,R,id,k,b);
}
void Cmax(int &a,int b,int x)
{
    double ya=K[a]*x+B[a];
    double yb=K[b]*x+B[b];
    if(ya<yb||(fabs(ya-yb)<1e-7&&a>b))a=b;
}
int Query(int now,int l,int r,int x)
{
    if(l==r)return t[now].id;
    int mid=(l+r)>>1,ret=t[now].id;
    if(x<=mid)Cmax(ret,Query(lson,l,mid,x),x);
    else Cmax(ret,Query(rson,mid+1,r,x),x);
    return ret;
}
int Q,ans,tot;
int main()
{
    Q=read();
    while(Q--)
    {
        int opt=read();
        if(!opt)
        {
            int x=((read()+ans-1)%39989+1);
            ans=Query(1,1,N,x);
            printf("%d\n",ans);
        }
        else
        {
            int x0=(read()+ans-1)%39989+1,y0=(read()+ans-1)%1000000000+1;
            int x1=(read()+ans-1)%39989+1,y1=(read()+ans-1)%1000000000+1;
            if(x0>x1)swap(x0,x1),swap(y0,y1);
            ++tot;K[tot]=1.0*(y0-y1)/(x0-x1);B[tot]=y0-K[tot]*x0;
            Modify(1,1,N,x0,x1,tot,K[tot],B[tot]);
        }
    }
    return 0;
}
```