Notebook

pedroteosousa

Contents

1 Geometry **1**
1.1 Miscellaneous Geometry 1
1.2 Convex Hull 2

2 Data Structures **4**
2.1 Trie 4
2.2 Binary Indexed Tree 5
2.3 Lazy Segment Tree 6
2.4 Union Find 8

3 Mathematics **9**
3.1 Matrix 9
3.2 Fast Fourier Transform 10
3.3 Extended Euclidean Algorithm 12

4 Miscellaneous **13**
4.1 vim settings 13

1 Geometry

1.1 Miscellaneous Geometry

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

typedef double type;
double EPS = 1e-12;

struct point {
    type x, y;
    point(type xp = 0.0, type yp = 0.0) {
        x = xp;
        y = yp;
    }
    point(const point &p) {
        x = p.x;
        y = p.y;
    }
    point operator+ (const point &p) const { return point(x+p.x, y+p.y); }
    point operator- (const point &p) const { return point(x-p.x, y-p.y); }
    point operator* (type c) { return point(c*x, c*y); }
    point operator/ (type c) { return point(x/c, y/c); }

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

type dot(point p, point q)  { return p.x*q.x+p.y*q.y; }
type dist(point p, point q)  { return sqrt(dot(p-q,p-q)); }
type cross(point p, point q)  { return p.x*q.y-p.y*q.x; }

point projectInLine(point c, point a, point b) {
    return a + (b-a)*dot(c-a, b-a)/dot(b-a, b-a);
}

point projectInSegment(point c, point a, point b) {
    point lineP = projectInLine(c, a, b);
    type maxDist = max(dist(a, lineP), dist(b, lineP));
    if (maxDist > dist(a, b)) {
        if (dist(a, c) > dist(b, c)) return b;
        else return a;
    }
    else return lineP;
}

int main() {
    point a(0, 0), b(1, 1), c(1, 0);
    c = projectInSegment(c, a, b);
    printf("%lf %lf\n", c.x, c.y);
}
```

1.2 Convex Hull

```
// This solves problem E on codeforces gym 101484
#include <bits/stdc++.h>
using namespace std;

typedef long long type;
double EPS = 1e-12;
```

```
struct point {
    type x, y;
    point(type xp = 0, type yp = 0) {
        x = xp;
        y = yp;
    }

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

type cross(point p, point q)  { return p.x*q.y-p.y*q.x; }

double side(point a, point b, point c) {
    return cross(a, b) + cross(b, c) + cross(c, a);
}

vector<point> convex_hull(vector<point> p) {
    int n = p.size(), k = 0;
    if (n == 1) return p;
    vector<point> hull(2*n);

    sort(p.begin(), p.end());

    for(int i=0; i<n; i++) {
        // use <= when including collinear points
        while(k>=2 && (side(hull[k-2], hull[k-1], p[i]) < 0)) k--;
        hull[k++] = p[i];
    }

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

    hull.resize(k-1);
    return hull;
}

set<point> v1, v2;
int main() {
    int n, m; scanf("%d %d", &n, &m);
    vector<point> h;
    for (int i=0;i<n;i++) {
        int a, b; scanf("%d %d", &a, &b);
        point c = point(a, b);
        v1.insert(c); h.push_back(c);
    }
    for (int i=0;i<m;i++) {
        int a, b; scanf("%d %d", &a, &b);
        point c = point(a, b);
        v2.insert(c); h.push_back(c);
    }
    h = convex_hull(h);
    if (v1.find(h[0]) != v1.end()) {
        for (int i=0;i<h.size();i++)
            if (v2.find(h[i]) != v2.end()) {
                printf("NO\n"); return 0;
            }
    }
    else {
        for (int i=0;i<h.size();i++)
            if (v1.find(h[i]) != v1.end()) {
                printf("NO\n"); return 0;
            }
    }
    printf("YES\n");
}
```

2 Data Structures

2.1 Trie

```
const int A = 26;

typedef struct trie {
    struct node {
        int to[A], freq, end;
    };
    struct node t[N];
    int sz = 0;
    int offset = 'a';

    // init trie
    void init() {
        memset(t, 0, sizeof(struct node));
    }
    // insert string
    void insert(char *s, int p = 0) {
        t[p].freq++;
        if (*s == 0) {
            t[p].end++;
            return;
        }
        if (t[p].to[*s - offset] == 0)
            t[p].to[*s - offset] = ++sz;
        insert(s+1, t[p].to[*s - offset]);
    }

    // check if string is on trie
    int find(char *s, int p = 0) {
        if (*s == 0)
            return t[p].end;
        if (t[p].to[*s - offset] == 0)
            return false;
        return find(s+1, t[p].to[*s - offset]);
    }

    // count the number of strings that have this prefix
    int count(char *s, int p = 0) {
        if (*s == 0)
            return t[p].freq;
        if (t[p].to[*s - offset] == 0)
            return 0;
        return count(s+1, t[p].to[*s - offset]);
    }

    // erase a string
    int erase(char *s, int p = 0) {
        if (*s == 0 && t[p].end) {
            --t[p].end;
            return --t[p].freq;
        }
        if ((*s == 0 && t[p].end == 0) || t[p].to[*s - offset] == 0)
            return -1;
        int count = erase(s+1, t[p].to[*s - offset]);
        if (count == 0)
            t[p].to[*s - offset] = 0;
        if (count == -1)
            return -1;
        return --t[p].freq;
    }
} trie;
```

2.2 Binary Indexed Tree

```
int b[N];

int update(int p, int val, int n) {
    for(;p < n; p += p & -p) b[p] += val;
}

int getsum(int p) {
    int sum = 0;
    for(; p != 0; p -= p & -p) {
        sum += b[p];
    }
    return sum;
}
```

2.3 Lazy Segment Tree

```
// This solves HORRIBLE on SPOJ
#include <bits/stdc++.h>
using namespace std;

typedef long long lli;

const lli N = 1e5 + 5;
const lli inf = 1791791791;

/* type: 0 = min
         1 = max
         2 = sum */
template <int type> struct seg_tree {
    lli seg[4*N];
    lli lazy[4*N];

    seg_tree() {
        memset(seg, 0, sizeof(seg));
        memset(lazy, 0, sizeof(lazy));
    }

    void do_lazy(lli root, lli ll, lli rl) {
        if (type == 2)
            seg[root] += (rl-ll+1)*lazy[root];
        else
            seg[root] += lazy[root];
        if (ll != rl) {
            lazy[2*root+1] += lazy[root];
            lazy[2*root+2] += lazy[root];
        }
```

```
            lazy[root] = 0;
        }

        // sum update
        lli update(lli l, lli r, lli val, lli ll = 0, lli rl = N-1, lli root = 0) {
            do_lazy(root, ll, rl);
            if (r < ll || l > rl) return seg[root];
            if (ll >= l && rl <= r) {
                lazy[root] += val;
                do_lazy(root, ll, rl);
                return seg[root];
            }
            lli update_left = update(l, r, val, ll, (ll+rl)/2, 2*root+1);
            lli update_right = update(l, r, val, (ll+rl)/2+1, rl, 2*root+2);
            if (type == 0)
                return seg[root] = min(update_left, update_right);
            if (type == 1)
                return seg[root] = max(update_left, update_right);
            if (type == 2)
                return seg[root] = update_left + update_right;
        }

        lli query(lli l, lli r, lli ll = 0, lli rl = N-1, int root = 0) {
            do_lazy(root, ll, rl);
            if (r < ll || l > rl) {
                if (type == 0)
                    return inf;
                if (type == 1)
                    return -inf;
                if (type == 2)
                    return 0;
            }
            if (ll >= l && rl <= r) return seg[root];
            lli query_left = query(l, r, ll, (ll+rl)/2, 2*root+1);
            lli query_right = query(l, r, (ll+rl)/2+1, rl, 2*root+2);
            if (type == 0)
                return min(query_left, query_right);
            if (type == 1)
                return max(query_left, query_right);
            if (type == 2)
                return query_left + query_right;
        }
    };

    int main() {
        int t; scanf("%d", &t);
        while (t--) {
            int n, c; scanf("%d %d", &n, &c);
            seg_tree <2> t;
            while (c--) {
                int op, l, r;
                scanf("%d %d %d", &op, &l, &r);
                l--; r--;
                if (op == 0) {
                    int v; scanf("%d", &v);
                    t.update(l, r, v);
                }
                else
                    printf("%lld\n", t.query(l, r));
            }
        }
    }
```

2.4 Union Find

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

const int N = 5e5 + 5;
int p[N], w[N];

int find(int x) {
    return p[x] = (x == p[x] ? x : find(p[x]));
}

void join(int a, int b) {
    if ((a = find(a)) == (b = find(b))) return;
    if (w[a] < w[b]) swap(a, b);
    w[a] += w[b];
    p[b] = a;
}

int main() {
    int n;
    scanf("%d", &n);
    for(int i=0;i<n;i++)
        w[p[i] = i] = 1;
    return 0;
}
```

3 Mathematics

3.1 Matrix

```
// This solves problem MAIN74 on SPOJ
#include <bits/stdc++.h>
using namespace std;

const int mod = 1e9+7;

template <int n> struct matrix {
    long long mat[n][n];
    matrix () {
        memset (mat, 0, sizeof (mat));
    }
    matrix (long long temp[n][n]) {
        memcpy (mat, temp, sizeof (mat));
    }
    void identity() {
        memset (mat, 0, sizeof (mat));
        for (int i=0;i<n;i++)
            mat[i][i] = 1;
    }
    matrix<n> operator* (const matrix<n> &a) const {
        matrix<n> temp;
        for (int i=0; i<n; i++)
            for (int j=0; j<n; j++)
                for (int k=0; k<n; k++)
                    temp.mat[i][j] += mat[i][k]*a.mat[k][j];
        return temp;
    }
    matrix<n> operator% (long long m) {
        matrix<n> temp(mat);
        for (int i=0; i<n; i++)
            for (int j=0; j<n; j++)
                temp.mat[i][j] %= m;
        return temp;
    }
    matrix<n> pow(long long e, long long m) {
        matrix<n> temp;
        if (e == 0) {
            temp.identity();
            return temp%m;
        }
        if (e == 1) {
            memcpy (temp.mat, mat, sizeof (temp.mat));
            return temp%m;
        }
        temp = pow(e/2, m);
        if (e % 2 == 0)
            return (temp*temp)%m;
        else
            return (((temp*temp)%m)*pow(1, m))%m;
    }
};

int main() {
    int t;
    scanf("%d", &t);
    while (t--) {
        long long n;
        scanf("%lld", &n);
        matrix<2> m;
        long long temp[2][2] = {{1, 1}, {1, 0}};
        memcpy (m.mat, temp, sizeof (m.mat));
        m = m.pow(n+2, mod);
        if (n == 0) m.mat[0][0] = 0;
        if (n == 1) m.mat[0][0] = 2;
        printf("%lld\n", m.mat[0][0]);
    }
    return 0;
}
```

3.2 Fast Fourier Transform

```
// This solves VFMUL on SPOJ
#include <bits/stdc++.h>
using namespace std;

#define PI 3.14159265359

const int N = 3e5 + 5;
typedef complex<double> base;

// p[0]*x^0 + p[1]*x + ...
void fft(vector<base> &p, bool inverse) {
    if (p.size() == 1)
        return;
    int n = p.size();

    vector<base> a[2];
    for (int i=0; i<n; i++)
```

```
        a[i%2].push_back(p[i]);

    for (int i=0; i<2; i++)
        fft(a[i], inverse);

    double theta = (2*PI/n)*(inverse ? -1 : 1);
    base w(1), wn(cos(theta), sin(theta));
    for (int i=0; i<n/2; i++) {
        p[i] = (a[0][i] + w * a[1][i]) / (base)(inverse ? 2 : 1);
        p[i+n/2] = (a[0][i] - w * a[1][i]) / (base)(inverse ? 2 : 1);
        w *= wn;
    }
}

// c ends being a*b
void multiply(vector<int> &a, vector<int> &b, vector<int> &c) {
    vector<base> na(a.begin(), a.end()), nb(b.begin(), b.end());
    int n = 1;
    while (n < max(a.size(), b.size())) n <<= 1;
    n <<= 1;
    na.resize(n); nb.resize(n);

    fft(na, false); fft(nb, false);
    for (int i=0;i<n;i++) {
        na[i] *= nb[i];
    }
    fft(na, true);

    c.resize(n);
    for (int i=0;i<n;i++)
        c[i] = (int)(na[i].real() + 0.5);
}

int main() {
    int t; scanf("%d", &t);
    while (t--) {
        char s1[N], s2[N];
        scanf("%s %s", s1, s2);
        int n1 = strlen(s1), n2 = strlen(s2);
        vector<int> a, b, c;

        for (int i=n1-1;i>=0;i--)
            a.push_back(s1[i]-'0');
        for (int i=n2-1;i>=0;i--)
            b.push_back(s2[i]-'0');
        multiply(a, b, c);
        c.resize(2*c.size());
        for (int i=0;i<c.size()-1;i++) {
            c[i+1] += c[i]/10;
            c[i] %= 10;
        }

        int found = 0;
        for (int i=c.size()-1;i>=0; i--) {
            if (c[i] != 0) found = 1;
            if (found) printf("%c", c[i] + '0');
        }
        if (!found) printf("0");
        printf("\n");
    }
    return 0;
}
```

3.3 Extended Euclidean Algorithm

```
// This solves 10104 on UVa
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;

ll ext(ll a, ll b, ll &x, ll &y) {
    if (a == 0) {
        x = 0;
        y = 1;
        return b;
    }
    ll x1, y1;
    ll gcd = ext(b%a, a, x1, y1);

    x = y1 - (b/a)*x1;
    y = x1;

    return gcd;
}

int main() {
    ll a, b;
    while (scanf("%lld %lld", &a, &b) != EOF) {
        ll x, y;
        ll gcd = ext(a, b, x, y);
        if (a == b && x > y) swap(x, y);
        printf("%lld %lld %lld\n", x, y, gcd);
    }
    return 0;
}
```

4 Miscellaneous

4.1 vim settings

```
set ai si noet ts=4 sw=4 sta sm nu rnu
inoremap <NL> <ESC>o
nnoremap <NL> o
inoremap <C-up> <C-o>:m-2<CR>
inoremap <C-down> <C-o>:m+1<CR>
nnoremap <C-up> :m-2<CR>
nnoremap <C-down> :m+1<CR>
vnoremap <C-up> :m-2<CR>gv
vnoremap <C-down> :m'>+1<CR>gv
syntax on
colors evening
highlight Normal ctermbg=none "No background
highlight nonText ctermbg=none
```