ACM-ICPC Team Reference Document

Izhevsk State Technical University (Gogolev, Kismatov, Lykov)

Contents

1 Common **1**
- 1.1 template 1
- 1.2 compile 1
- 1.3 check 1
- 1.4 how To Solve Problems 2

2 Geometry **2**
- 2.1 structures 2
- 2.2 segment Intersection 2
- 2.3 semiplane Intersection 2
- 2.4 Line and circle 3
- 2.5 Two circles 3
- 2.6 Two common tangets to circles 4

3 Matrix **4**
- 3.1 Multiply 4
- 3.2 Gauss SLAU 4

4 Strings **4**
- 4.1 Suf LCP 4
- 4.2 LCP 4
- 4.3 Suf Automaton 5
- 4.4 pref Function 5
- 4.5 Z Function 5
- 4.6 Aho Corasick 5

5 Graphs **5**
- 5.1 cutpoints 5
- 5.2 bridges 6
- 5.3 min Cost Max Flow 6

6 Data Structures **6**
- 6.1 fenwick Tree 6
- 6.2 dekart Tree 6
- 6.3 Long 7

7 DP **7**
- 7.1 fft 7
- 7.2 convex Hull Trick 8

8 Comb **8**
- 8.1 basic 8

1 Common

1.1 template

```
optional, complex, string, vector, list, map, set, deque, queue,
        stack, bitset,
algorithm, sstream, iostream, iomanip, cstdio, cstdlib, ctime,
        cstring, cmath,
cstdarg, cassert, ctime, tuple, unordered_set, unordered_map,
        random, chrono
using namespace std;

#define ln ”\n”
#define pb push_back
#define mp make_pair
#define ins insert
#define sz(x) (int)x.size()

#define All(x) (x).begin(), (x).end()
#define fi(a, b) for (auto i = (a); i <= (b); i++)
#define fdi(a, b) for (auto i = (a); i >= (b); i--)
#define fx(A) for (auto &x : (A))

typedef long long ll;
typedef vector<int> vi;
typedef vector<vi> vvi;
typedef pair<int, int> pii;

template<typename T>
ostream& operator<<(ostream& os, vector<T> v) {
        fz(v) {
                os << z << ”␣”;
        }
        return os;
}

#ifdef LOCAL
        #define dbg(x) {cerr << ___LINE___ << ”\t” << #x
                << ”:␣” << (x) << ln;}
#else
        #define dbg(x) {}
#endif

#ifdef LOCAL
    #define ass(x) if (!(x)) { cerr << ___LINE___ << ”\
        tassertion␣failed:␣” << #x << ln; abort(); }
#else
    #define ass(x) assert(x)
#endif

mt19937_64 rnd(chrono::steady_clock::now().time_since_epoch
        ().count());

int main()
{
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    #ifdef LOCAL
        freopen(”input.txt”, ”r”, stdin);
        freopen(”output.txt”, ”w”, stdout);
    #endif
}
```

1.2 compile

```
g++ --std=c++20 !.! -DLOCAL -O2 -Wall -Wl,--stack
        =67108864 -o !.exe
```

1.3 check

```
:start
gen.exe
sol.exe
copy output.txt answer.txt
NAME.exe
fc output.txt answer.txt
if ERRORLEVEL 1 goto end
goto start
:end
```

1.4 how To Solve Problems

1. .
2.
3.
4. :
    (a)
    (b)
    (c)
    (d)
    (e)
    (f)
    (g)
    (h)
    (i)
    (j)
    (k)
    (l)
    (m)
    (n)
    (o)
    (p) ( )
    (q)

2 Geometry

2.1 structures

```
const ld eps = 1e-6;
struct Point {
    ld x;
    ld y;
    Point operator+(Point B) {
        return Point{x + B.x, y + B.y};
    }
    Point operator-(Point B) {
        return Point{x - B.x, y - B.y};
    }
    Point operator*(double t) {
        return Point{x * t, y * t};
    }
    bool operator==(Point B) {
        return abs(x - B.x) <= eps && abs(y - B.y)<=eps;
    }
    Point norm() {
        ld U = sqrt(x * x + y * y);
        return Point{x / U, y / U};
    }
    ld len() {
        return sqrt(x * x + y * y);
    }
};

struct Line{
  ll A;
  ll B;
  ll C;
};

struct Rec{
  Line L;
  Seg sg;
  Point V;
};

struct Seg {
    Point a;
    Point b;
    Point to_vec() {
        return Point{b.x - a.x, b.y - a.y};
    }
};
ld dot(Point p1, Point p2) {
    return p1.x * p2.x + p1.y * p2.y;
}
ld dot(Point p1, Point p2, Point p3) {
    return dot(p2 - p1, p3 - p1);
}
ld cross(Point p1, Point p2) {
    return p1.x * p2.y - p2.x * p1.y;
}
ld cross(Point p1, Point p2, Point p3) {
    return cross(p2 - p1, p3 - p1);
}
```

2.2 segment Intersection

```
optional<Point> intersect(Seg s1, Seg s2) {
    if (fabs(cross(s1.to_vec(), s2.to_vec())) < eps) {
        return {};
    }
    double t1 = cross((s2.a - s1.a), s2.to_vec()) / cross(s1.
        to_vec(), s2.to_vec());
    double t2 = cross((s1.a - s2.a), s1.to_vec()) / cross(s2.
        to_vec(), s1.to_vec());
    if (t1 < -eps || t1 > 1 + eps) {
        return {};
    }
    if (t2 < -eps || t2 > 1 + eps) {
        return {};
    }
    return s1.a + (s1.b - s1.a) * t1;
}
```

2.3 semiplane Intersection

```
ld area(Point a, Point b, Point c) {
    return (ld)1.0 * ((a.x - b.x) * (a.y - c.y) - (a.x - c.x) * (a.y -
        b.y));
}
bool inside(Point A, Point B, Point C, Point p) {
    if (area(A, B, p) == 0 || area(B, C, p) == 0 || area(A, C, p)
        == 0)
        return false;
    ld v1 = area(A, B, p);
    ld v2 = area(B, C, p);
    ld v3 = area(A, C, p);
    ld S = area(A, B, C);
    v1 = abs(v1);
    v2 = abs(v2);
    v3 = abs(v3);
    S = abs(S);
    return abs(S - v1 - v2 - v3) <= eps;
}
Point find_bis(Point A, Point B, Point C) {
    auto v = A - B;
    auto v2 = C - B;
    v.norm();
    v2.norm();
    return v + v2;
}
ld get_acos(double x) {
    return acos(max(-1., min(1., x)));
}

Line make_line(Point a, Point b) {
    ll A = round(b.y - a.y);
    ll B = round(a.x - b.x);
    ll C = round(a.y * b.x - a.x * b.y);
    if (A < 0)
    {
        A *= (-1);
        B *= (-1);
        C *= (-1);
    }
    ll g = gcd(A, gcd(abs(B), abs(C)));
    A /= g;
    B /= g;
    C /= g;
    return Line{A, B, C};
}
Point intersect(Line a, Line b) {
    ld d = a.A * b.B - b.A * a.B;
    ld d1 = -a.C * b.B + a.B * b.C;
    ld d2 = -a.A * b.C + a.C * b.A;
    ass(abs(d) > eps);
    Point res = {d1 / d, d2 / d};
```

```
    ass(on_line(res, a));
    ass(on_line(res, b));
    return {d1 / d, d2 / d};
}
Point get_ort(Point p, Point p2) {
    return {p2.y - p.y, p.x - p2.x};
}
Point get_ort(Line a) {
    return {a.A, a.B};
}
const ld pi = get_acos(-1.0);
bool good(deque<Rec> &dq) {
    vector<Point> P;
    vector<Rec> v;
    while (sz(dq)) {
        v.pb(dq.back());
        dq.pop_back();
    }
    fi(1, sz(v) - 1) {
        P.pb(intersect(v[i - 1].L, v[i].L));
    }
    P.pb(intersect(v[0].L, v[sz(v) - 1].L));
    ld res = 0;

    P.pb(P[0]);
    ld s = 0, s2 = 0;
    fi(1, sz(P) - 1) {
        s += P[i].y * P[i - 1].x;
        s2 += P[i].x * P[i - 1].y;
    }
    res = abs(s - s2);

    return res > 0 + eps;
}
bool paral(Line a, Line b) {
    return a.A * b.B - a.B * b.A == 0;
}
bool fun(ll T) {
    if (T == 0)
        return true;
    vector<Rec> vec(n + 1);
    set<tuple<ll, ll, ll>> t;
    fi(1, n) {
        Point p = a[i];
        Point p2 = a[go(i, T + 1, n)];
        vec[i] = Rec{make_line(p, p2), Seg{p, p2}, get_ort(p,
            p2)};
        if (area(p, p2, {p.x + 1'000'000 * vec[i].V.x, p.y + 1'000'
            000 * vec[i].V.y}) > 0 + eps) {
            vec[i].V = vec[i].V * -1;
        }
        // vec[i].V.norm();
        // t.ins({vec[i].L.A,vec[i].L.B,vec[i].L.C});
    }
    // ass(sz(t)==n);
    vector<Rec> top, bot;
    fi(1, n) {
        if (vec[i].V.y > 0 + eps || (abs(vec[i].V.y) <= eps &&
            vec[i].V.x > 0 + eps)) {
            bot.pb(vec[i]);
        } else {
            top.pb(vec[i]);
        }
    }
    sort(All(top), [](Rec &a, Rec &b)
        { return area(a.V, b.V, {0, 0}) <= eps; });
    sort(All(bot), [](Rec &a, Rec &b)
        { return area(a.V, b.V, {0, 0}) + eps >= 0; });
    vec = vector<Rec>(1);
    fz(bot) vec.pb(z);
    reverse(All(top));
    fz(top) vec.pb(z);
    deque<Rec> dq;
    dbg(vec);
    fi(1, n) {
        while (sz(dq) >= 2) {
            auto rec = dq.back();
            dq.pop_back();
            auto rec2 = dq.back();
            auto P = intersect(rec.L, rec2.L);
            dbg(P);
            if (area(vec[i].sg.a, vec[i].sg.b, P) <= eps) {
                dq.push_back(rec);
                break;
            } else {
                continue;
            }
        }
        while (sz(dq) >= 2) {
            auto rec = dq.front();
            dq.pop_front();
            auto rec2 = dq.front();
            Point P = intersect(rec.L, rec2.L);
            dbg(P);
            dbg(area(vec[i].sg.a, vec[i].sg.b, P));
            if (area(vec[i].sg.a, vec[i].sg.b, P) <= eps) {
                dq.push_front(rec);
                break;
            } else {
                continue;
            }
        }
        dq.pb(vec[i]);
    }

    while (sz(dq) >= 3) {
        auto rec = dq.front();
        dq.pop_front();
        auto rec2 = dq.front();
        auto P = intersect(rec.L, rec2.L);
        dbg(P);
        if (area(dq.back().sg.a, dq.back().sg.b, P) <= eps) {
            dq.push_front(rec);
            break;
        } else {
            continue;
        }
    }

    while (sz(dq) >= 3) {
        auto rec = dq.back();
        dq.pop_back();
        auto rec2 = dq.back();
        auto P = intersect(rec.L, rec2.L);
        dbg(P);
        if (area(dq.front().sg.a, dq.front().sg.b, P) <= eps) {
            dq.push_back(rec);
            break;
        } else {
            continue;
        }
    }

    dbg(T);
    dbg(sz(dq));

    return sz(dq) > 2;
}
```

2.4 Line and circle

We say first circle center at point $(0, 0)$. Line is (A, B, C). Find $x_0 = \frac{-AC}{A^2+B^2}$

$y_0 = \frac{-BC}{A^2+B^2}$

If dist from (x_0, y_0) to $(0, 0)$ is bigger than r then answer is zero. Otherwise if it equals to r then answer is (x_0, y_0). Otherwise we have 2 solutions: $d = \sqrt{r^2 - \frac{C^2}{A^2+B^2}}$ $V = \sqrt{\frac{d^2}{A^2+B^2}}$ $a_x = x_0 + BV$ $a_y = y_0 - AV$ $b_x = x_0 - BV$ $b_y = y_0 + AV$

2.5 Two circles

We say first circle center at point $(0, 0)$. $x^2 + y^2 = r_1^2$ $(x - x_2)^2 + (y - y_2)^2 = r_2^2$

$x^2 + y^2 = r_1^2$ $x(-2x_2) + y(-2y_2) + (x_2^2 + y_2^2 + r_1^2 + r_2^2) = 0$ Now we have new task to find intersection of line and circle: $Ax + By + C = 0$, $A = -2x_2$, $B = -2y_2$, $C = x_2^2 + y_2^2 + r_1^2 - r_2^2$

2.6 Two common tangets to circles

We have to find all tangent lines to two circles. First circle is at $(0,0)$.

We say first circle center at point $(0,0)$. $x^2+y^2 = r_1^2$ $(x-x_2)^2+(y-y_2)^2 = r_2^2$

$x^2+y^2 = r_1^2$ $x(-2x_2)+y(-2y_2)+(x_2^2+y_2^2+r_1^2+r_2^2) = 0$ Now we have new task to find intersection of line and circle: $Ax+By+C = 0$, $A = -2x_2$, $B = -2y_2$, $C = x_2^2+y_2^2+r_1^2-r_2^2$

3 Matrix

3.1 Multiply

```
Matrix mult_mtr(Matrix A, Matrix B, ll MOD) {
        vector<vll>res;
        ll n2, m2;
        n2 = A.n;
        m2 = B.m;
        res = vector<vll>(n2+1,vll(m2+1));
        fi(1, n2) {
                fj(1, m2) {
                        ll cur = 0;
                        fo(1, A.m) {
                                cur = cur + A.mtr[i][o] * B.mtr[o][j
                                        ];
                        }
                        res[i][j] = cur;
                }
        }
        return Matrix{n2,m2,res};
}
```

3.2 Gauss SLAU

```
int gauss (vector < vector<double> > a, vector<double> &
       ans) {
        int n = (int) a.size();
        int m = (int) a[0].size() - 1;
        vector<int> where (m, -1);
        for (int col=0, row=0; col<m && row<n; ++col) {
                int sel = row;
                fi(row, n - 1) {
                        if (abs (a[i][col]) > abs (a[sel][col]))
                                sel = i;
                }
                if (abs (a[sel][col]) < EPS)
                        continue;
                fi(col, m) {
                        swap (a[sel][i], a[row][i]);
                }
                where[col] = row;
                fi(0, n - 1) {
                        if (i != row) {
                                double c = a[i][col] / a[row][col];
                                fj(col, m) {
                                        a[i][j] -= a[row][j] * c;
                                }
                        }
                }
                ++row;
        }
        ans.assign (m, 0);
        fi(0, m - 1) {
                if (where[i] != -1)
                        ans[i] = a[where[i]][m] / a[where[i]][i];
        }
        fi(0, n - 1) {
                double sum = 0;
                fj(0, m - 1) {
                        sum += ans[j] * a[i][j];
                }
                if (abs (sum - a[i][m]) > EPS)
                        return 0;
        }
        fi(0, m - 1) {
                if (where[i] == -1) return INF;
                }

                return 1;
        }
```

4 Strings

4.1 Suf LCP

```
string s;
vi p, c, rc;
int x = 1;
int gett(int a, int x) {
    if (a > x) return a - x;
    return sz(s) - 1 - (x - a) % (sz(s) - 1);
}
int gett2(int a, int x) {
    if (a + x < sz(s))
        return a + x;
    return 1 + (a + x) % sz(s);
}
bool comp1(int a, int b) {
    return (s[a] < s[b]);
}
int n, S;
vvi e;
void Sort(vector<int> &p) {
    fi(1, n) {
        int t = gett(p[i], x);
        e[rc[t]].pb(t);
    }
    p.clear();
    p.pb(0);
    fi(0, n) {
        fz(e[i]) {
            p.pb(z);
        }
        e[i].clear();
    }
}
void init() {
    n = sz(s) + 1;
    s = "#" + s + (char)31;
    c = vi(n + 1);
    rc = vi(n + 1);
    fi(0, n) p.pb(i);
    sort(Allf(p), comp1);
    c[1] = 0;
    fi(2, n) {
        if (s[p[i]] == s[p[i - 1]]) c[i] = c[i - 1];
        else c[i] = c[i - 1] + 1;
    }
    fi(1, n) {
        rc[p[i]] = c[i];
    }
    e = vvi(n + 1);
    x = 1;
    while (x < n) {
        Sort(p);
        c[1] = 0;
        fi(2, n) {
            if (rc[p[i]] == rc[p[i - 1]] && rc[gett2(p[i], x)] == rc[
                gett2(p[i - 1], x)]) {
                c[i] = c[i - 1];
            } else {
                c[i] = c[i - 1] + 1;
            }
        }
        fi(1, n) {
            rc[p[i]] = c[i];
        }
        x *= 2;
    }
}
```

4.2 LCP

```
vll lcm(n + 1);
ll k = 0;
fi(1, n) {
    ll x = rev[i] - 1;
```

```
    ll cnt = max(k - 1, (ll)0);
    fj(cnt, 1e5 + 10) {
        if (s[i + j] == s[arr[x] + j]) cnt++;
        else break;
    }
    if (x == 0) cnt = 0;
    lcm[x] = cnt;
    k = cnt;
}
```

4.3 Suf Automaton

```
struct state {
        int len, link;
        map<char, int> next;
};
const int MAXLEN = 1'000'000;
state st[MAXLEN*2];
int siz, last;
void sa_init() {
        siz = last = 0;
        st[0].len = 0;
        st[0].link = -1;
        siz++;
}
void sa_extend (char c) {
        int cur = siz++;
        st[cur].len = st[last].len + 1;
        int p;
        for (p = last; p != -1 && !st[p].next.count(c); p = st[p].
              link) {
                st[p].next[c] = cur;
    }
        if (p == -1) {
                st[cur].link = 0;
        } else {
                int q = st[p].next[c];
                if (st[p].len + 1 == st[q].len) {
                        st[cur].link = q;
                } else {
                        int clone = siz++;
                        st[clone].len = st[p].len + 1;
                        st[clone].next = st[q].next;
                        st[clone].link = st[q].link;
                        for (; p != -1 && st[p].next[c] == q; p =
                              st[p].link)
                                st[p].next[c] = clone;
                        st[q].link = st[cur].link = clone;
                }
        }
        last = cur;
}
```

4.4 pref Function

```
vi prefix_function(string s) {
        int n = sz(s);
        vi pi(n);
        fi(1, n - 1) {
                int j = pi[i - 1];
                while (j > 0 && s[i] != s[j]) {
                        j = pi[j - 1];
        }
                if (s[i] == s[j]) j++;
                pi[i] = j;
        }
        return pi;
}
```

4.5 Z Function

```
vi z_function (string s) {
        int n = sz(s);
        vi z (n);
        for (int i = 1, l = 0, r = 0; i < n; i++) {
                if (i <= r) z[i] = min(r - i + 1, z[i - l]);
                while (i + z[i] < n && s[z[i]] == s[i + z[i]]) {
                        z[i]++;
        }
                if (i + z[i] - 1 > r) {
                        l = i;
                        r = i + z[i] - 1;
                }
                }
                return z;
}
```

4.6 Aho Corasick

```
struct Node {
        int leaf = -1;
        char c = '\0';
        Node* par = nullptr;
        map<char, Node*> nxt;
        Node* suf = nullptr;
        map<char, Node*> sufs;
        Node(char t, Node* p) {
                c = t;
                par = p;
        }
};
Node* root;
void add(string& s, int j) {
        Node* cur = root;
        fi(0, sz(s)- 1) {
                if(cur->nxt.count(s[i])) {
                        cur = cur->nxt[s[i]];
                } else {
                        Node* t = new Node(s[i],cur);
                        cur->nxt[s[i]] = t;
                        cur = t;
                }
        }
        cur->leaf = j;
}
Node* get_link(Node* a) {
        if(a->suf) return a->suf;
        if(a == root) {
                a->suf = root;
                return a->suf;
        }
        if(a->par == root) {
                return root;
        }
        char t = a->c;
        Node* cur = get_link(a->par);
        while(cur != root && !cur->nxt.count(t)) {
                cur = get_link(cur);
        }
        if(cur->nxt.count(t)) {
                cur = cur->nxt[t];
        }
        return a->suf = cur;
}
Node* go(Node* a,char c) {
        Node* cur = a;
        while(cur != root && !cur->nxt.count(c)) {
                cur = get_link(cur);
        }
        if(cur->nxt.count(c)) {
                cur = cur->nxt[c];
        }
        return cur;
}

string s;
ll n;
vector<string> a;
void init() {
        root = new Node('\0',nullptr);
        fi(1, n) {
                add(a[i], i);
        }
}
```

5 Graphs

5.1 cutpoints

```
void dfs(ll x, ll p = -1) {
    if (color[x]) return;
    color[x] = 1;
    _time++;
    tin[x] = _time;
    tup[x] = _time;
    ll q = 0;
    bool f = 0;
    fy(e[x]) {
        if (y == p) continue;
        if (!color[y]) {
            q++;
            dfs(y, x);
            if (tup[y] >= tin[x]) {
                f = 1;
            }
            tup[x] = min(tup[x], tup[y]);
        } else {
            tup[x] = min(tup[x], tin[y]);
        }
    }
    if (p == -1) {
        if (q > 1) ans.pb(x);
    } else {
        if (f) ans.pb(x);
    }
}
```

5.2 bridges

```
void dfs(ll x, ll p = -1) {
    if (color[x]) return;
    color[x] = 1;
    tin[x] = _time++;
    tup[x] = tin[x];
    for (auto &to : e[x]) {
        if (to == p) continue;
        if (color[to]) {
            tup[x] = min(tup[x], tin[to]);
            continue;
        } else {
            dfs(to, x);
            tup[x] = min(tup[x], tup[to]);
            if (tup[to] > tin[x]) {
                ans.pb(t[{to, x}]);
                dbg(mp(to, x));
            }
        }
    }
}
```

5.3 min Cost Max Flow

```
bool deikstr() {
    vll d(_k + 1, INF);
    set<pll> t;
    d[S] = 0;
    t.ins({d[S], S});
    vector<Edge *> p(_k + 1, NULL);
    while (sz(t)) {
        auto [_, x] = *t.begin();
        t.erase(t.begin());
        fy(e[x]) {
            if (y->f + 1 <= y->c && d[y->y] > d[x] + y->w) {
                t.erase({d[y->y], y->y});
                d[y->y] = d[x] + y->w;
                t.ins({d[y->y], y->y});
                p[y->y] = y;
            }
        }
    }
    if (d[T] == INF) return false;
    int x = T;
    while (x != S) {
        p[x]->f++;
        p[x]->rev->f--;
        x = p[x]->rev->y;
    }
    fi(1, _k) {
        fy(e[i]) {
            y->w = y->w + d[i] - d[y->y];
        }
    }
    return true;
}
```

6 Data Structures

6.1 fenwick Tree

```
// Linear
vll fen;
void add(int p, ll val) {
    for(int i = p; i <= n; i = (i | (i + 1))) {
        fen[i] += val;
    }
}
ll sum(int p) {
    ll res = 0;
    for(int i = p; i >= 0; i = (i & (i + 1)) - 1) {
        res += fen[i];
    }
    return res;
}

// Matrix
vvll fen;
void add(int x, int y, ll val) {
    for(int i = x; i <= n; i = (i | (i + 1))) {
        for(int j = y; j <= n; j = (j | (j + 1))) {
            fen[i][j] += val;
        }
    }
}
ll sum(int x, int y) {
    ll res = 0;
    for(int i = x; i >= 0; i = (i & (i + 1)) - 1) {
        for (int j = y; j >= 0; j = (j & (j + 1)) - 1) {
            res += fen[i][j];
        }
    }
    return res;
}
```

6.2 dekart Tree

```
struct Node {
    ll val;
    ll cnt;
    ll y;
    Node *l;
    Node *r;
};
void upd(Node *a) {
    a->cnt = 1;
    if (a->l)
        a->cnt += a->l->cnt;
    if (a->r)
        a->cnt += a->r->cnt;
}
Node *merge(Node *a, Node *b) {
    if (!a)
        return b;
    if (!b)
        return a;
    if (a->y > b->y) {
        a->r = merge(a->r, b);
        upd(a);
        return a;
    }
    b->l = merge(a, b->l);
    upd(b);
    return b;
}
pair<Node *, Node *> split(Node *a, ll q) {
    if (!a)
        return {0, 0};
    ll left = 0;
    if (a->l)
        left += a->l->cnt;
    if (left >= q) {
        auto [l, r] = split(a->l, q);
        a->l = r;
        upd(a);
```

```
        return {l, a};
    }
    auto [l, r] = split(a->r, q - left - 1);
    a->r = l;
    upd(a);
    return {a, r};
}
```

6.3 Long

```
struct Long {
    int base = 10000;
    int len = 4;
    vi num = {1, 0};
    int operator[](int x) const {
        if (x > num[0]) return 0;
        return num[x];
    }
    int& operator[](int x) {
        if (x > num[0]) {
            num.resize(x + 1, 0);
            num[0] = x;
        }
        return num[x];
    }
};
Long operator+(const Long& A, const Long& B) {
    Long res;

    int n = max(A[0], B[0]);
    fi(1, n) {
        res[i] = A[i] + B[i];
        if (i > 1) {
            res[i] += res[i - 1] / res.base;
            res[i - 1] %= res.base;
        }
    }
    while(res[res[0]] >= res.base) {
        int m = res[0];
        res[m + 1] = res[m] / res.base;
        res[m] %= res.base;
    }
    return res;
}
Long operator*(const Long& A, const int& B) {
    Long res;
    int n = A[0];
    fi(1, n) {
        res[i] = A[i] * B;
        if (i > 1) {
            res[i] += res[i - 1] / res.base;
            res[i - 1] %= res.base;
        }
    }
    while(res[res[0]] >= res.base) {
        int m = res[0];
        res[m + 1] = res[m] / res.base;
        res[m] %= res.base;
    }
    return res;
}
Long operator*(const Long& A, const Long& B) {
    Long res;
    fi(1, A[0]) {
        if (A[i] == 0) continue;
        fj(1, B[0]) {
            if (A[i] * B[j] == 0) continue;
            int x = i + j - 1;
            res[x] += A[i] * B[j];
            while(x > 1 && res[x - 1] >= res.base) {
                res[x] += res[x - 1] / res.base;
                res[x - 1] %= res.base;
            }
        }
        while(res[res[0]] >= res.base) {
            int m = res[0];
            res[m + 1] += res[m] / res.base;
            res[m] %= res.base;
        }
    }
    return res;
}
pair<Long, int> operator/(const Long& A, const int B) {
    Long res;
    ll x = 1;
    ll d = 0, m = 0;
    fdi(A[0], 1) {
        ll val = m * res.base + A[i];
        d = val / B;
        m = val % B;
        if (x == 1 && d == 0) continue;
        res[x] = d;
        x++;
    }
    reverse(Allf(res.num));
    return mp(res, m);
}
void print(const Long& A) {
    printf("%d", A[A[0]]);
    fdi(A[0] - 1, 1) {
        printf("%04d", A[i]);
    }
    printf(ln);
}
Long str_to_Long(const string& s) {
    int n = sz(s);
    Long res;
    int x = 1;
    for(int i = n - 1; i >= 0; i -= 4) {
        int val = 0;
        for(int j = max(0, i - 3); j <= i; j++) {
            val = val * 10 + (s[j] - '0');
        }
        res[x] = val;
        x++;
    }
    return res;
}
Long int_to_Long(int num) {
    Long res;
    int x = 1;
    while(num) {
        int val = num % res.base;
        num /= res.base;
        res[x] = val;
        x++;
    }
    return res;
}
```

7 DP

7.1 fft

```
using cd = complex<double>;
const double PI = acos(-1);
int reverse(int num, int lg_n) {
    int res = 0;
    for (int i = 0; i < lg_n; i++) {
        if (num & (1 << i))
            res |= 1 << (lg_n - 1 - i);
    }
    return res;
}
void fft(vector<cd> & a, bool invert) {
    int n = a.size();
    int lg_n = 0;
    while ((1 << lg_n) < n)
        lg_n++;

    for (int i = 0; i < n; i++) {
        if (i < reverse(i, lg_n))
            swap(a[i], a[reverse(i, lg_n)]);
    }

    for (int len = 2; len <= n; len <<= 1) {
        double ang = 2 * PI / len * (invert ? -1 : 1);
        cd wlen(cos(ang), sin(ang));
        for (int i = 0; i < n; i += len) {
            cd w(1);
            for (int j = 0; j < len / 2; j++) {
                cd u = a[i+j], v = a[i+j+len/2] * w;
                a[i+j] = u + v;
                a[i+j+len/2] = u - v;
                w *= wlen;
            }
        }
    }

    if (invert) {
```

```
        for (cd & x : a) {
            x /= n;
                }
    }
}
vector<int> multiply(vector<int>const& a, vector<int>const&
        b) {
    vector<cd> fa(a.begin(), a.end()), fb(b.begin(), b.end());
    int n = 1;
    while (n < sz(a) + sz(b) ) {
        n <<= 1;
    }
    fa.resize(n);
    fb.resize(n);

    fft(fa, false);
    fft(fb, false);
    for (int i = 0; i < n; i++)
        fa[i] *= fb[i];
    fft(fa, true);

    vector<int> result(n);
    for (int i = 0; i < n; i++)
        result[i] = round(fa[i].real());
    for( int i = 0;i < n;i++){
      result[i] = min(result[i],5);
    }
    while(result.back() == 0)result.pop_back();
    return result;
}
```

7.2 convex Hull Trick

```
struct Line {
    mutable ll k, m, p;
    bool operator<(const Line &o) const { return k < o.k; }
    bool operator<(ll x) const { return p < x; }
};

struct LineContainer : multiset<Line, less<>> {
    // (for doubles, use inf = 1/.0, div(a,b) = a/b)
    static const ll inf = INF;
    ll div(ll a, ll b) { // floored division
        return a / b - ((a ^ b) < 0 && a % b);
    }
    bool isect(iterator x, iterator y) {
        if (y == end())
            return x->p = inf, 0;
        if (x->k == y->k)
            x->p = x->m > y->m ? inf : -inf;
        else
            x->p = div(y->m - x->m, x->k - y->k);
        return x->p >= y->p;
    }
    void add(ll k, ll m) {
        auto z = insert({k, m, 0}), y = z++, x = y;
        while (isect(y, z))
            z = erase(z);
        if (x != begin() && isect(--x, y))
            isect(x, y = erase(y));
        while ((y = x) != begin() && (--x)->p >= y->p)
            isect(x, erase(y));
    }
    ll query(ll x) {
        assert(!empty());
        auto l = *lower_bound(x);
        return l.k * x + l.m;
    }
};
```

8 Comb

8.1 basic

$Cat(n,k) = \frac{1}{n+1}C(2n,n)$

$N(n,k) = \frac{1}{n}C(n,k)C(n,k-1)$

$\sum_{n=0}^{\inf}\sum_{k=1}^{n} N(n,k)z^n t^k = \frac{1+z(t-1)-\sqrt{1-2z(t+1)+z^2(t-1)^2}}{2z}$

$S(n,k) = \sum_{j=0}^{k-1}(-1)^j C(k,j) \cdot (m-j)^n$ $O(k\log n)$