1 Environment

1.1 Linux

US keymap: `setxkbmap us`
Left Shift: `xmodmap -e 'keycode 94=Shift_L'`
Disable Caps Lock: `setxkbmap -option caps:none`
Enable Caps Lock: `setxkbmap -option`

1.2 .vimrc

```
set␣nocp
filetype␣plugin␣indent␣on
syn␣on
set␣ts=4␣sw=4␣ai␣si␣ar␣aw␣nu␣is␣mouse=a
let␣c_no_curly_error=1
colorscheme␣peachpuff

ino␣{<CR>␣{<CR>}<ESC>O

let␣tmp=":set␣makeprg=g++\\␣-DBZ\\␣-Wall\\␣-W\\␣-Wno-sign-compare
↪  \\␣%\\␣-o\\␣%<\\␣"
nm␣<C-b>␣:exe␣tmp␣.␣"-O2"␣<CR>:make␣<CR>:cwindow␣<CR>
nm␣<F9>␣:exe␣tmp␣.␣"-D_GLIBCXX_DEBUG\\␣-fsanitize=undefined
↪  \\␣-O2"␣<CR>:make␣<CR>:cwindow␣<CR>
nm␣<C-F9>␣:exe␣tmp␣.␣"-D_GLIBCXX_DEBUG\\␣-fsanitize=undefined
↪  \\␣-g"␣<CR>:make␣<CR>:cwindow␣<CR>

nm␣<F5>␣:!./%<␣<CR>
nm␣<C-F5>␣:!gdb␣--tui␣%<␣<CR>

nm␣<F6>␣:!time␣./%<␣<␣%<.in␣<CR>
nm␣<C-F6>␣:!gdb␣--tui␣-ex␣'set␣args␣<␣%<.in'␣%<<CR>

au␣BufNewFile␣*.cpp␣0r␣~/.vim/tmp.cpp

vm␣<C-c>␣"+y
nm␣<C-a>␣ggVG

nm␣<C-v>␣"+gP
im␣<C-v>␣<ESC>"+gPi
nm␣<C-p>␣:!printfile␣%␣<CR>
nm␣<C-S-F12>␣:!submit␣%␣&&␣printfile␣%␣<CR>
```

1.3 C++ template

```cpp
#ifndef BZ
#pragma GCC optimize "-O3"
#endif
#include <bits/stdc++.h>

#define ALL(v) (v).begin(), (v).end()

using ll = long long;
using ld = long double;
using ull = uint64_t;

using namespace std;

int main() {
    ios_base::sync_with_stdio(0), cin.tie(0), cout.tie(0);
    cout.setf(ios::fixed), cout.precision(20);
    return 0;
}

//C-b <F5> <F6>
//<F9> <F5> <F6>
//<C-F9> <C-F5> <C-F6>
//<C-a>
//<C-c>
//<C-v> in command mode
//<C-v> in insert mode
```

Contents

1.4 Table

PROBLEM	ICPC 2019				
	READ	SOLVED	WRITTEN	OPENED	COMMENT
A					
B					
C					
D					
E					
F					
G					
H					
I					
J					
K					
L					
M					
N					

2 Math

2.1 Integrals

$$\int \frac{dx}{a^2+x^2} = \frac{1}{a}\arctan\frac{x}{a} \qquad \int \frac{dx}{a^2-x^2} = \frac{1}{2a}\ln\left|\frac{a+x}{a-x}\right|$$

$$\int \frac{x\,dx}{a^2\pm x^2} = \pm\frac{1}{2}\ln|a^2\pm x^2| \qquad \int \frac{dx}{\sqrt{a^2-x^2}} = \arcsin\frac{x}{a}$$

$$\int \sqrt{a^2-x^2}\,dx = \frac{x}{2}\sqrt{a^2-x^2}+\frac{a^2}{2}\arcsin\frac{x}{a} \qquad \int \frac{x\,dx}{\sqrt{a^2\pm x^2}} = \pm\sqrt{a^2\pm x^2}$$

$$\int \sqrt{x^2\pm a^2} = \frac{x}{2}\sqrt{x^2\pm a^2}\pm\frac{a^2}{2}\ln|x+\sqrt{x^2\pm a^2}| \qquad \int \frac{dx}{\sqrt{x^2\pm a^2}} = \ln|x+\sqrt{x^2\pm a^2}|$$

3 Tricks

3.1 GCC

```
//bitset
_Find_first(); _Find_next(prev);
//PBDS
//#include <ext/pb_ds/assoc_container.hpp>
#include <bits/extc++.h>
using namespace __gnu_pbds;

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

find_by_order(k);
order_of_key(x);
//Первая возвращает итератор на k-ый по величине элемент (отсчёт с нуля),
↪  вторая - возвращает количество элементов в множестве, строго меньших,
↪  чем наш элемент.

//hasttable
gp_hash_table<int, int> table;

//SSE pragma

#pragma GCC target("sse,sse2,sse3,ssse3,sse4,avx,avx2")
#pragma GCC target("sse,sse2,sse3,ssse3,sse4,popcnt,abm,mmx")

//measure time
ld sum = 0;
auto start = std::chrono::system_clock::now();
auto end = std::chrono::system_clock::now();
sum += std::chrono::duration<double>(end - start).count();
```

3.2 Mult

```
ll mul(ll a, ll b, ll m) {
  ll q = (ll)((ld)a * (ld)b / (ld)m);
  ll r = a * b - q * m;
  return (r + 5 * m) % m;
}
```

4 Popular Algos

4.1 Hungary

```
//finds minimal solution
void hungary(int n, int m) {
    using T = int;
    vector<T> u(n + 1), v(m + 1);
    vector<int> p(m + 1), way(m + 1);
    for (int i = 1; i <= n; ++i) {
        p[0] = i;
        int j0 = 0;
        vector<T> minv (m + 1, INF);
        vector<char> used (m + 1, 0);
        do {
            used[j0] = 1;
            int i0 = p[j0], j1 = 0;
            T d = INF;
```

```
            for (int j = 1; j <= m; ++j)
                if (!used[j]) {
                    T cur = a[i0][j] - u[i0] - v[j];
                    if (cur < minv[j])
                        minv[j] = cur, way[j] = j0;
                    if (minv[j] < d)
                        d = minv[j], j1 = j;
                }
            for (int j = 0; j <= m; ++j)
                if (used[j])
                    u[p[j]] += d, v[j] -= d;
                else
                    minv[j] -= d;
            j0 = j1;
        } while (p[j0] != 0);
        do {
            int j1 = way[j0];
            p[j0] = p[j1];
            j0 = j1;
        } while (j0);
    }

    vector<int> ans (n + 1);
    for (int j = 1; j <= m; ++j)
        ans[p[j]] = j;
    T cost = -v[0];
}
```

4.2 CHT

```
struct Line {
    ll k, b;
    mutable const Line *nx;
    bool operator<(ll x) const {
        if (!nx) return 0;
        return b - nx->b < (nx->k - k) * x;
    }
    bool operator<(const Line& rhs) const {
        return k < rhs.k;
    }
};
// will maintain upper hull for maximum
struct HullDynamic : multiset<Line, less<>> {
    bool bad(iterator y) {
        auto z = next(y);
        if (y == begin()) {
            if (z == end()) return 0;
            return y->k == z->k && y->b <= z->b;
        }
        auto x = prev(y);
        if (z == end()) return y->k == x->k && y->b <= x->b;
        return (x->b - y->b)*(z->k - y->k) >= (y->b - z->b)*(y->k - x->k);
        //ll dv(ll a, ll b) { assert(b > 0); return a / b + (a % b >= 0)}
        //return dv(x->b - y->b, y->k - x->k) >= dv(y->b - z->b, z->k -
        ↪  y->k);
    }
    void insert_line(ll k, ll b) {
        auto y = insert({k, b, 0});
        if (bad(y)) { erase(y); return; }
        auto z = next(y);
        while (z != end() && bad(z)) z = erase(z);
        if (z != end()) y->nx = &*z;
        while (y != begin() && bad(z = prev(y))) erase(z);
        if (y != begin()) z->nx = &*y;
    }
    ll eval(ll x) {
        auto l = *lower_bound(x);
        return l.k * x + l.b;
    }
};
```

5 Geometry

5.1 HPI

```
//getPoint - возвращает направляющий вектор полуплоскости (l:
↪  getPoint(l)^p>=0)
//Считаем, что все прямые нормированы
//Считаем, что есть bounding box
//Возвращает вектор точек пересечения в ccw порядке
```

```
sort(lines.begin(), lines.end(), [] (line al, line bl) -> bool {
            point a = getPoint(al);
            point b = getPoint(bl);
            if (a.y >= 0 && b.y < 0)
                return 1;
            if (a.y < 0 && b.y >= 0)
                return 0;
            if (a.y == 0 && b.y == 0)
                return a.x > 0 && b.x < 0;
            return a * b > 0;
        });
vector<pair<line, int>> st;
for (int it = 0; it < 2; ++it)
    for (int i = 0; i < lines.size(); ++i) {
        int fl = 0;
        while (!st.empty()) {
            if ((getPoint(st.back().first) - getPoint(lines[i])).len() <=
            ↪   eps) {
                if (st.back().first.c >= lines[i].c) {
                    fl = 1;
                    break;
                }
                else
                    st.pop_back();
            }
            else if (getPoint(st.back().first) * getPoint(lines[i]) <= eps
            ↪   / 2)
                return {}; //don't intersect
            else if (st.size() >= 2 &&
                    st[st.size() - 2].first.get(cross(st.back().first,
                    ↪   lines[i])) < 0) {
                st.pop_back();
            }
            else
                break;
        }
        if (!fl)
            st.push_back({lines[i], i});
    }
fill(en, en + lines.size(), -1);
vector<point> p;
for (int i = 0; i < st.size(); ++i) {
    int x = st[i].second;
    if (en[x] == -1)
        en[x] = i;
    else {
        for (int j = en[x]; j < i; ++j)
            p.push_back(cross(st[j].first, st[j + 1].first));
        break;
    }
}
return p;
```

5.2 Radical Axis

$2(q.x - p.x)x + 2(q.y - p.y)y = rp^2 - rq^2 + (q\hat{\ }q) - (p\hat{\ }p)$

5.3 Hull3d

```
//call dfs(i,j), i,j is any edge on convex hull,  will visit all faces
void dfs(int i, int j) {
    if (vis[i][j]) return;
    vis[i][j] = true;

    int k = 0;
    while (k == i || k == j) k++;
    for (int l = 0; l < n; l++) {
        //side returns which side pts[l] lies on plane defined by pts i,
        ↪   j, k
        if (l != i && l != j && side(pts[i], pts[j], pts[k], pts[l]) > 0)
            k = l;
    }
    // points i,j,k form a face on convex hull.
    dfs(k, j);
    dfs(i, k);
}
```

5.4 Another Hull3d

1. Найти тетраэдр

2. Добавлять точки по одной
3. Удалить треугольники, которые под нами
4. Добавить грани к оставшимся ребрам

5.5 Simplex

```
// Chinese simplex

//WARNING: MAXN and MAXM are too small.
const ld eps = 1e-10;
struct Simplex{
    ld a[MAXN][MAXM], b[MAXN], c[MAXM], d[MAXN][MAXM];
    ld x[MAXM];
    int ix[MAXN + MAXM]; // !!! array all indexed from 0
    // max{cx} subject to {Ax<=b,x>=0}
    // n: constraints, m: vars !!!
    // x[] is the optimal solution vector
    // usage :
    // value = simplex(a, b, c, N, M);
    Simplex() {
        memset(this, 0, sizeof(*this));
    }
    pair<ld,bool> simplex(int n, int m){
        ++m;
        int r = n, s = m - 1;
        memset(d, 0, sizeof(d));
        for (int i = 0; i < n + m; ++i) ix[i] = i;
        for (int i = 0; i < n; ++i) {
            for (int j = 0; j < m - 1; ++j) d[i][j] = -a[i][j];
            d[i][m - 1] = 1;
            d[i][m] = b[i];
            if (d[r][m] > d[i][m]) r = i;
        }
        for (int j = 0; j < m - 1; ++j) d[n][j] = c[j];
        d[n + 1][m - 1] = -1;
        for (ld dd;; ) {
            if (r < n) {
                swap(ix[s], ix[r + m]);
                d[r][s] = 1.0 / d[r][s];
                for (int j = 0; j <= m; ++j)
                    if (j != s) d[r][j] *= -d[r][s];
                for (int i = 0; i <= n + 1; ++i) if (i != r) {
                    for (int j = 0; j <= m; ++j) if (j != s)
                        d[i][j] += d[r][j] * d[i][s];
                    d[i][s] *= d[r][s];
                }
            }
            r = -1; s = -1;
            for (int j = 0; j < m; ++j)
                if (s < 0 || ix[s] > ix[j]) {
                    if (d[n + 1][j] > eps ||
                            (d[n + 1][j] > -eps && d[n][j] > eps))
                        s = j;
                }
            if (s < 0) break;
            for (int i = 0; i < n; ++i) if (d[i][s] < -eps) {
                if (r < 0 ||
                        (dd = d[r][m] / d[r][s] - d[i][m] / d[i][s]) <
                        ↪  -eps ||
                        (dd < eps && ix[r + m] > ix[i + m]))
                    r = i;
            }
            if (r < 0) return { 0 , false }; // not bounded
        }
        if (d[n + 1][m] < -eps) return { 0 , false }; // not executable
        ld ans = 0;
        for(int i=0; i<m; i++) x[i] = 0;
        for (int i = m; i < n + m; ++i) { // the missing enumerated x[i] =
        ↪  0
            if (ix[i] < m - 1){
                ans += d[i - m][m] * c[ix[i]];
                x[ix[i]] = d[i-m][m];
            }
        }
        return { ans , true };
    }
};

// magic simplex by Ivan Belonogov
```

```
struct Simplex {
    // maximize C^{t}x suspect to Ax <= b
    ll a[MAX_M][MAX_N];
    ll b[MAX_M];
    ll c[MAX_N];
    ll v;
    ll n, m;
    ll left[MAX_M];
    ll up[MAX_N];
    ll res[MAX_N];
    int pos[MAX_N];

       Simplex() {
        memset(this, 0, sizeof(*this));
    }

    void pivot(ll x, ll y) {
        swap(left[x], up[y]);
        ll k = a[x][y];
        assert(abs(k) == 1);
        a[x][y] = 1;
        b[x] /= k;
        int cur = 0;
        for (int i = 0; i < n; i++) {
            a[x][i] = a[x][i] / k;
            if (a[x][i] != 0)
                pos[cur++] = i;
        }

        for (int i = 0; i < m; i++) {
            if (i == x || a[i][y] == 0) continue;
            ll cof = a[i][y];
            b[i] -= cof * b[x];
            a[i][y] = 0;
            for (int j = 0; j < cur; j++)
                a[i][pos[j]] -= cof * a[x][pos[j]];
        }
        ll cof = c[y];
        v += cof * b[x];
        c[y] = 0;
        for (int i = 0; i < cur; i++) {
            c[pos[i]] -= cof * a[x][pos[i]];
        }
    }

    void solve(int nn, int mm) {
        n = nn, m = mm;
        for (int i = 0; i < n; i++)
            up[i] = i;
        for (int i = 0; i < m; i++)
            left[i] = i + n;

        int c1 = 0;
        while (1) {
            int x = -1;
            for (int i = 0; i < m; i++)
                if (b[i] < 0 && (x == -1 || b[i] < b[x])) {
                    x = i;
                }
            if (x == -1) break;

            int y = -1;
            for (int j = 0; j < n; j++)
                if (a[x][j] < 0) {
                    y = j;
                    break;
                }
            if (y == -1) {
                assert(false); // no solution
            }
            c1++;
            pivot(x, y);
        }
        while (1) {
            int y = -1;
            for (int i = 0; i < n; i++)
                if (c[i] > 0  && (y == -1 || (c[i] > c[y]))) {
                    y = i;
                }
```

```
            if (y == -1) break;

            int x = -1;
            for (int i = 0; i < m; i++) {
                if (a[i][y] > 0) {
                    if (x == -1 || (b[i] / a[i][y] < b[x] / a[x][y])) {
                        x = i;
                    }
                }
            }
            if (y == -1) {
                assert(false); // infinite solution
            }
            pivot(x, y);
        }

        memset(res, 0, sizeof(res));

        for (int i = 0; i < m; i++) {
            if (left[i] < n) {
                res[left[i]] = b[i];
            }
        }

        // res is an answer, v = C^{T}res
    }
};
```

6 Strings

6.1 Eertree

```
void addLetter(int n)
{
        while (str[n - len[v] - 1] != str[n] )
                v = suffLink[v];
        int u = suffLink[v];
        while (str[n - len[u] - 1] != str[n] )
                u = suffLink[u];
        int u_ = to[u][str[n] - 'a'];
        int v_ = to[v][str[n] - 'a'];
        if (v_ == -1)
        {
                v_ = to[v][str[n] - 'a'] = numV;
                len[numV++] = len[v] + 2;
                suffLink[v_] = u_;
        }
        v = v_;
        cnt[v]++;
}

void init()
{
        memset(to, -1, sizeof to);
        str[0] = '#';
        len[0] = -1;
        len[1] = 0;
        len[2] = len[3] = 1;
        suffLink[1] = 0;
        suffLink[0] = 0;
        suffLink[2] = 1;
        suffLink[3] = 1;
        to[0][0] = 2;
        to[0][1] = 3;
        numV = 4;
}
```

6.2 SufArray

```
void buildsa() {
    s[n] = 0;
    ++n;
    int sz = max(AL, n + 1) + 2;
    memset(cnt, 0, sizeof(cnt[0]) * sz);
    for (int i = 0; i < n; ++i)
        c[i] = s[i], ++cnt[c[i] + 1];
    for (int i = 1; i < sz; ++i)
        cnt[i] += cnt[i - 1];
    for (int i = 0; i < n; ++i)
        p[cnt[c[i]]++] = i;
```

```
for (int k = 0; (1 << k) < n; ++k) {
    for (int i = 0; i < n; ++i)
        p[i] = nm(p[i] - (1 << k), n);
    memset(cnt, 0, sizeof(cnt[0]) * sz);
    for (int i = 0; i < n; ++i)
        ++cnt[c[i] + 1];
    for (int i = 1; i < sz; ++i)
        cnt[i] += cnt[i - 1];
    for (int i = 0; i < n; ++i)
        p2[cnt[c[p[i]]]++] = p[i];
    memcpy(p, p2, sizeof(p[0]) * n);
    c2[p[0]] = 0;
    int now = 0;
    for (int i = 1; i < n; ++i) {
        if (c[p[i]] == c[p[i - 1]] &&
                c[nm(p[i] + (1 << k), n)] == c[nm(p[i - 1] + (1 << k),
                ↪  n)])
            c2[p[i]] = now;
        else
            c2[p[i]] = ++now;
    }
    memcpy(c, c2, sizeof(c[0]) * n);
    if (now == n - 1)
        break;
}
int lst = 0;
for (int i = 0; i < n; ++i) {
    int x = c[i];
    lst = max(0, lst - 1);
    if (x == n - 1)
        lst = 0;
    else {
        int y = p[x + 1];
        while (i + lst < n && y + lst < n && s[i + lst] == s[y + lst])
            ++lst;
        lcp[x] = lst;
    }
}
--n;
for (int i = 0; i < n; ++i)
        p[i] = p[i + 1], --c[i], lcp[i] = lcp[i + 1];
}
```

6.3 SufAuto

```
const int AL = 26;
const int SYM = 'a';

struct SA {
    vector<int> sf;
    vector<array<int, AL>> go;
    vector<int> len;
    int newn() {
        sf.push_back(-1);
        go.emplace_back(array<int, AL>());
        go.back().fill(-1);
        len.push_back(0);
        return sf.size() - 1;
    }
    SA(string s) {
        int cur = newn();
        for (int i = 0; i < s.size(); ++i) {
            int c = s[i] - SYM;
            int nw = newn();
            len[nw] = i + 1;
            while (cur != -1 && go[cur][c] == -1)
                go[cur][c] = nw, cur = sf[cur];
            if (cur == -1)
                sf[nw] = 0;
            else {
                int q = go[cur][c];
                if (len[q] == len[cur] + 1)
                    sf[nw] = q;
                else {
                    int cl = len.size();
                    len.push_back(len[cur] + 1);
                    go.push_back(go[q]);
                    sf.push_back(sf[q]);
                    sf[q] = sf[nw] = cl;
                    while (cur != -1 && go[cur][c] == q)
```

```
                    go[cur][c] = cl, cur = sf[cur];
                }
            }
            cur = nw;
        }
    }
};
```

6.4 SufTree

```
const int inf = 1e9;
char s[maxn];
map<int, int> to[maxn];
int len[maxn] = {inf}, fps[maxn], link[maxn];
int node, pos;
int sz = 1, n = 0;

int make_node(int _pos, int _len) {
    fps[sz] = _pos;
    len [sz] = _len;
    return sz++;
}
void go_edge() {
    while(pos > len[to[node][s[n - pos]]]) {
        node = to[node][s[n - pos]];
        pos -= len[node];
    }
}
void add_letter(int c) {
    s[n++] = c;
    pos++;
    int last = 0;
    while(pos > 0) {
        go_edge();
        int edge = s[n - pos];
        int &v = to[node][edge];
        int t = s[fps[v] + pos - 1];
        if(v == 0) {
            v = make_node(n - pos, inf);
            link[last] = node;
            last = 0;
        }
        else if(t == c) {
            link[last] = node;
            return;
        }
        else {
            int u = make_node(fps[v], pos - 1);
            to[u][c] = make_node(n - 1, inf);
            to[u][t] = v;
            fps[v] += pos - 1;
            len [v] -= pos - 1;
            v = u;
            link[last] = u;
            last = u;
        }
        if(node == 0)
            pos--;
        else
            node = link[node];
    }
}
//fps - first occurrence in string
//len - length of edge (inf for last edges
//link - suf link for vertices
```

6.5 Tandem

```
int ds[MAXN];

void run(int l, int r) {
    if (l + 1 == r)
        return;
    int m = (l + r) >> 1;
    run(l, m), run(m, r);
    s1 = s.substr(m, r - m) + "#" + s.substr(l, m - l);
    s2 = s1;
    reverse(ALL(s2));
    zf(s1, z1);
    zf(s2, z2);
```

```
    for (int i = l; i <= r; ++i)
        ds[i] = 0;
    //BEGIN PRIMITIVE TANDEMS
    for (int i = 1; i < r - m; ++i) {
        if (ds[m + i])
            continue;
        int go = z1[i] / i;
        for (int j = 1; j <= go; ++j)
            ds[m + (j + 1) * i] = 1;
    }
    for (int i = 1; i < m - l; ++i) {
        if (ds[m - i])
            continue;
        int go = z2[i] / i;
        for (int j = 1; j <= go; ++j)
            ds[m - i * (j + 1)] = 1;
    }
    //END PRIMITIVE TANDEMS
    for (int len = 1; len <= m - l; ++len) {
        int x = m - len;
        if (ds[x])
            continue;
        int xl = max(x - len + 1, x - z2[len]);
        int xr = min(x, x - len + z1[r - m + 1 + x - l]);
        for (int j = xl; j <= xr; ++j)
            vv[len].push_back(j);
    }
    for (int len = 1; len <= r - m; ++len) {
        int x = m + len;
        if (ds[x])
            continue;
        int xl = max(x + 1, x + len - z2[m - l + 1 + r - x]) - 2 * len;
        int xr = min(x + len - 1, x + z1[len]) - 2 * len;
        for (int j = xl; j <= xr; ++j)
            vv[len].push_back(j)
    }
}
```

7 Number Theory

7.1 Numbers

$10^6+3, 10^7+19, 10^8+7, 10^9+7, 10^{10}+19, 10^{11}+3, 10^{12}+39, 10^{13}+37, 10^{14}+31, 10^{15}+37, 10^{16}+61, 10^{17}+3, 10^{18}+3$

$10^6-17, 10^7-9, 10^8-11, 10^9-63, 10^{10}-33, 10^{11}-23, 10^{12}-11, 10^{13}-29, 10^{14}-27, 10^{15}-11, 10^{16}-63, 10^{17}-3, 10^{18}-11, 1234567891$

$998244353 = 2^{22} * 2 * 7 * 17 + 1, 662^{2^{22}} \equiv 1$

Fibonacci: $45 : 1134903170$ $46 : 1836311903$ $47 : 2971215073$ $91 : 4660046610375530309$ $92 : 7540113804746346429$ $93 : 12200160415121876738$

Highly composite numbers $20 : d(12) = 6$ $50 : d(48) = 10$ $100 : d(60) = 12$ $1000 : d(840) = 32$ $10^4 : d(9240) = 64$ $10^5 : d(83160) = 128$ $10^6 : d(720720) = 240$ $10^7 : d(8648640) = 448$ $10^8 : d(91891800) = 768$ $10^9 : d(931170240) = 1344$ $10^{11} : d(97772875200) = 4032$ $10^{12} : d(963761198400) = 6720$ $10^{15} : d(866421317361600) = 26880$ $10^{18} : d(897612484786617600) = 103680$

Bell numbers: $0 : 1, 1 : 1, 2 : 2, 3 : 5, 4 : 15, 5 : 52, 6 : 203, 7 : 877, 8 : 4140, 9 : 21147, 10 : 115975, 11 : 678570, 12 : 4213597, 13 : 27644437, 14 : 190899322, 15 : 1382958545, 16 : 10480142147, 17 : 82864869804, 18 : 682076806159, 19 : 5832742205057, 20 : 51724158235372, 21 : 474869816156751, 22 : 4506715738447323, 23 : 44152005855084346$

Catalan numbers: $0 : 1, 1 : 1, 2 : 2, 3 : 5, 4 : 14, 5 : 42, 6 : 132, 7 : 429, 8 : 1430, 9 : 4862, 10 : 16796, 11 : 58786, 12 : 208012, 13 : 742900, 14 : 2674440, 15 : 9694845, 16 : 35357670, 17 : 129644790, 18 : 477638700, 19 : 1767263190, 20 : 6564120420, 21 : 24466267020, 22 : 91482563640, 23 : 343059613650, 24 : 1289904147324, 25 : 4861946401452$

Partitions numbers: $0 : 1, 1 : 1, 2 : 2, 3 : 3, 4 : 5, 5 : 7, 6 : 11, 7 : 15, 8 : 22, 9 : 30, 10 : 42, 20 : 627, 30 : 5604, 40 : 37338, 50 : 204226, 60 : 966467, 70 : 4087968, 80 : 15796476, 90 : 56634173, 100 : 190569292$

7.2 Prefix Inverse O(n)

```
r[i] = 1;
for (int i = 2; i < MOD; ++i)
    r[i] = (MOD - (MOD / i) * r[MOD % i] % MOD) % MOD;
```

7.3 Gonchar Fedor

```
// Counts x, y >= 0 such that Ax + By <= C.
ll count_triangle(ll A, ll B, ll C) {
    if (C < 0) return 0;
    if (A > B) swap(A, B);
```

```
    ll p = C / B;
    ll k = B / A;
    ll d = (C - p * B) / A;
    return count_triangle(B - k * A, A, C - A * (k * p + d + 1)) + (p + 1)
    ↪  * (d + 1) + k * p * (p + 1) / 2;
}
```

7.4 Primes < n

Введём обозначение p_j - j-е простое число. Пусть $dp_{n,j}$ — это количество k таких, что $1 \leq k \leq n$ и все простые делители k не меньше p_j (заметим, что число 1 будет учтено во всех $dp_{n,j}$, поскольку оно не имеет простых делителей). $dp_{n,j}$ удовлетворяет реккурентному соотношению:
$dp_{n,1} = n$ (поскольку $p_1 = 2$).
$dp_{n,j} = dp_{n,j} + 1 + dp_{\left[\frac{n}{p_j}\right],j}$, следовательно $dp_{n,j+1} = dp_{n,j} - dp_{\left[\frac{n}{p_j}\right],j}$.
Пусть p_k — это наименьшее простое большее $\sqrt{n}$. Тогда $\pi(n) = dp_{n,k} + k - 1$ (по определению первое слагаемое учитывает все простые не меньшие k).
Если вычислять реккурентность $dp_{n,k}$ напрямую, то все достижимые состояния будут иметь вид $dp_{\left[\frac{n}{i}\right],j}$. Также можно заметить, что если p_j и p_k оба больше $\sqrt{n}$, то $dp_{n,j}+j = dp_{n,k}+k$. Поэтому, для всех $\left[\frac{n}{i}\right]$ нам достаточно хранить только $\approx \pi(\sqrt{n/i})$ значений $dp_{\left[\frac{n}{i}\right],j}$.
Вместо прямого подсчёта всех состояний динамики, будем осуществлять двухшаговый процесс:
Выберем K.
Запустим рекурсивное вычисление $dp_{n,k}$. Если при этом в какой-то момент мы захотим посчитать значение для состояние $n < K$, запомним запрос "посчитать количество чисел не больше n с простыми делителями не меньше k".
Посчитаем ответы на все запросы в off-line: вычислим решето для чисел до K, отсортируем все числа по наименьшему простому делителю. Теперь мы можем ответить на все запросы, используя структуру данных на прибавление в точке и взятие суммы на отрезке (например, дерево Фенвика). Запомним все ответы глобально.
Снова запустим рекурсивное вычисление $dp_{n,j}$. Но в этот раз если мы захотим посчитать значение для состояния $n < K$, мы можем использовать уже вычисленное значение.
Эффективность этой идеи сильно зависит от величины Q — количество запросов, которые нам придётся обработать.
Предпосчёт ответов для Q запросов может быть выполнен за время $(K+Q)\log(K+Q)$ и это наиболее ёмкая часть вычислений. $K \approx \left(\frac{n}{\log n}\right)^{2/3}$.

8 Numerical

8.1 Runge-Kutta

$y_{n+1} = y_n + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right),$
$t_{n+1} = t_n + h$
$k_1 = h\ f(t_n, y_n),$
$k_2 = h\ f\left(t_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right),$
$k_3 = h\ f\left(t_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right),$
$k_4 = h\ f\left(t_n + h, y_n + k_3\right).$

8.2 Gaussian Quadrature

```
const ld C1 = sqrt(3.0 / 7 - 2.0 * sqrt(6.0 / 5) / 7.0);
const ld C2 = sqrt(3.0 / 7 + 2.0 * sqrt(6.0 / 5) / 7.0);
const ld D1 = (18.0 + sqrt(30)) / 36;
const ld D2 = (18.0 - sqrt(30)) / 36;
ld h = (rb - lb) / C / 2;
for (int i = 0; i < C; ++i) {
    ld x = (i + 1) * h * 2 + lb;
    ans += h * (D2 * (f(x - h * C2) + f(x + h * C2)) + D1 * (f(x - h * C1)
    ↪  + f(x + h * C1)));
}
```

9 Other

9.1 Walsh-Hadamard

```
void fft1(int *a, int n) {
    for (int bl = 1; bl < n; bl *= 2) {
        for (int i = 0; i < n; i += 2 * bl) {
            for (int j = i; j < i + bl; ++j) {
                int u = a[j];
                int v = a[j + bl];
                a[j] = (u + v) % MOD;
                a[j + bl] = (u - v + MOD) % MOD;
            }
```

```
        }
    }
}

/*
 * Inverse "fft"
 * B2 - inverse of 2 modulo MOD
 */

void fft2(int *a, int n) {
    for (int bl = n / 2; bl >= 1; bl /= 2) {
        for (int i = 0; i < n; i += 2 * bl) {
            for (int j = i; j < i + bl; ++j) {
                int u = a[j];
                int v = a[j + bl];
                a[j] = (ll)(u + v) * B2 % MOD;
                a[j + bl] = (ll)(u - v + MOD) * B2 % MOD;
            }
        }
    }
}
```

9.2 Global MinCut

```
const int MAXN = 500;
int n, g[MAXN][MAXN];
int best_cost = 1000000000;
vector<int> best_cut;

void mincut() {
    vector<int> v[MAXN];
    for (int i=0; i<n; ++i)
        v[i].assign (1, i);
    int w[MAXN];
    bool exist[MAXN], in_a[MAXN];
    memset (exist, true, sizeof exist);
    for (int ph=0; ph<n-1; ++ph) {
        memset (in_a, false, sizeof in_a);
        memset (w, 0, sizeof w);
        for (int it=0, prev; it<n-ph; ++it) {
            int sel = -1;
            for (int i=0; i<n; ++i)
                if (exist[i] && !in_a[i] && (sel == -1 || w[i] > w[sel]))
                    sel = i;
            if (it == n-ph-1) {
                if (w[sel] < best_cost)
                    best_cost = w[sel],  best_cut = v[sel];
                v[prev].insert (v[prev].end(), v[sel].begin(),
                ↪  v[sel].end());
                for (int i=0; i<n; ++i)
                    g[prev][i] = g[i][prev] += g[sel][i];
                exist[sel] = false;
            }
            else {
                in_a[sel] = true;
                for (int i=0; i<n; ++i)
                    w[i] += g[sel][i];
                prev = sel;
            }
        }
    }
}
```

9.3 Matroids

9.3.1 Обычное

Given a set $X \in \mathcal{F}_1 \cap \mathcal{F}_2$, we define a directed auxiliary graph G_X by

$$\begin{aligned} A_X^{(1)} &:= \{ (x, y) : y \in E \setminus X,\ x \in C_1(X, y) \setminus \{y\} \}, \\ A_X^{(2)} &:= \{ (y, x) : y \in E \setminus X,\ x \in C_2(X, y) \setminus \{y\} \}, \\ G_X &:= (E, A_X^{(1)} \cup A_X^{(2)}). \end{aligned}$$

We set

$$\begin{aligned} S_X &:= \{y \in E \setminus X : X \cup \{y\} \in \mathcal{F}_1\}, \\ T_X &:= \{y \in E \setminus X : X \cup \{y\} \in \mathcal{F}_2\} \end{aligned}$$

EDMONDS' MATROID INTERSECTION ALGORITHM

Input: Two matroids $(E, \mathcal{F}_1)$ and $(E, \mathcal{F}_2)$, given by independence oracles.

Output: A set $X \in \mathcal{F}_1 \cap \mathcal{F}_2$ of maximum cardinality.

① Set $X := \emptyset$.

② **For** each $y \in E \setminus X$ and $i \in \{1, 2\}$ **do**: Compute
$C_i(X, y) := \{x \in X \cup \{y\} : X \cup \{y\} \notin \mathcal{F}_i,\ (X \cup \{y\}) \setminus \{x\} \in \mathcal{F}_i\}$.

③ Compute S_X, T_X, and G_X as defined above.

④ Apply BFS to find a shortest S_X-T_X-path P in G_X.
If none exists **then stop**.

⑤ Set $X := X \triangle V(P)$ and **go to** ②.

9.3.2 Взвешенное

WEIGHTED MATROID INTERSECTION ALGORITHM

Input: Two matroids $(E, \mathcal{F}_1)$ and $(E, \mathcal{F}_2)$, given by independence oracles. Weights $c : E \to \mathbb{R}$.

Output: A set $X \in \mathcal{F}_1 \cap \mathcal{F}_2$ of maximum weight.

① Set $k := 0$ and $X_0 := \emptyset$. Set $c_1(e) := c(e)$ and $c_2(e) := 0$ for all $e \in E$.

② **For** each $y \in E \setminus X_k$ and $i \in \{1, 2\}$ **do**: Compute
$C_i(X_k, y) := \{x \in X_k \cup \{y\} : X_k \cup \{y\} \notin \mathcal{F}_i,\ (X_k \cup \{y\}) \setminus \{x\} \in \mathcal{F}_i\}$.

③ Compute

$$
\begin{aligned}
A^{(1)} &:= \{(x, y) : y \in E \setminus X_k,\ x \in C_1(X_k, y) \setminus \{y\}\},\\
A^{(2)} &:= \{(y, x) : y \in E \setminus X_k,\ x \in C_2(X_k, y) \setminus \{y\}\},\\
S &:= \{y \in E \setminus X_k : X_k \cup \{y\} \in \mathcal{F}_1\},\\
T &:= \{y \in E \setminus X_k : X_k \cup \{y\} \in \mathcal{F}_2\}.
\end{aligned}
$$

④ Compute

$$
\begin{aligned}
m_1 &:= \max\{c_1(y) : y \in S\},\\
m_2 &:= \max\{c_2(y) : y \in T\},\\
\bar{S} &:= \{y \in S : c_1(y) = m_1\},\\
\bar{T} &:= \{y \in T : c_2(y) = m_2\},\\
\bar{A}^{(1)} &:= \{(x, y) \in A^{(1)} : c_1(x) = c_1(y)\},\\
\bar{A}^{(2)} &:= \{(y, x) \in A^{(2)} : c_2(x) = c_2(y)\},\\
\bar{G} &:= (E, \bar{A}^{(1)} \cup \bar{A}^{(2)}).
\end{aligned}
$$

⑤ Apply BFS to compute the set R of vertices reachable from $\bar{S}$ in $\bar{G}$.

⑥ **If** $R \cap \bar{T} \neq \emptyset$ **then**: Find an $\bar{S}$-$\bar{T}$-path P in $\bar{G}$ with a minimum number of edges, set $X_{k+1} := X_k \triangle V(P)$ and $k := k + 1$ and **go to** ②.

⑦ Compute

$$
\begin{aligned}
\varepsilon_1 &:= \min\{c_1(x) - c_1(y) : (x, y) \in \delta^+_{A^{(1)}}(R)\},\\
\varepsilon_2 &:= \min\{c_2(x) - c_2(y) : (y, x) \in \delta^+_{A^{(2)}}(R)\},\\
\varepsilon_3 &:= \min\{m_1 - c_1(y) : y \in S \setminus R\},\\
\varepsilon_4 &:= \min\{m_2 - c_2(y) : y \in T \cap R\},\\
\varepsilon &:= \min\{\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4\}
\end{aligned}
$$

(where $\min \emptyset := \infty$).

⑧ **If** $\varepsilon < \infty$ **then**:
Set $c_1(x) := c_1(x) - \varepsilon$ and $c_2(x) := c_2(x) + \varepsilon$ for all $x \in R$. **Go to** ④.
If $\varepsilon = \infty$ **then**:
Among $X_0, X_1, \dots, X_k$, let X be the one with maximum weight. **Stop**.

9.4 Min Mean Cycle

Minimum Mean Cycle Algorithm

Input: A digraph G, weights $c : E(G) \to \mathbb{R}$.

Output: A circuit C with minimum mean weight or the information that G is acyclic.

① Add a vertex s and edges (s, x) with $c((s, x)) := 0$ for all $x \in V(G)$ to G.

② Set $n := |V(G)|$, $F_0(s) := 0$, and $F_0(x) := \infty$ for all $x \in V(G) \setminus \{s\}$.

③ **For** $k := 1$ **to** n **do**:
  For all $x \in V(G)$ **do**:
    Set $F_k(x) := \infty$.
    For all $(w, x) \in \delta^-(x)$ **do**:
      If $F_{k-1}(w) + c((w, x)) < F_k(x)$ **then**:
        Set $F_k(x) := F_{k-1}(w) + c((w, x))$ and $p_k(x) := w$.

④ **If** $F_n(x) = \infty$ for all $x \in V(G)$ **then stop** (G is acyclic).

⑤ Let x be a vertex for which $\max_{\substack{0 \le k \le n-1 \\ F_k(x) < \infty}} \frac{F_n(x) - F_k(x)}{n - k}$ is minimum.

⑥ Let C be any circuit in the edge progression given by
$p_n(x), p_{n-1}(p_n(x)), p_{n-2}(p_{n-1}(p_n(x))), \ldots$

9.5 General Matching

```
const int MAXN = ...;

int n;
vector<int> g[MAXN];
int match[MAXN], p[MAXN], base[MAXN], q[MAXN];
bool used[MAXN], blossom[MAXN];

int lca (int a, int b) {
    bool used[MAXN] = { 0 };
    for (;;) {
        a = base[a];
        used[a] = true;
        if (match[a] == -1)  break;
        a = p[match[a]];
    }
    for (;;) {
        b = base[b];
        if (used[b])  return b;
        b = p[match[b]];
    }
}

void mark_path (int v, int b, int children) {
    while (base[v] != b) {
        blossom[base[v]] = blossom[base[match[v]]] = true;
        p[v] = children;
        children = match[v];
        v = p[match[v]];
    }
}

int find_path (int root) {
    memset (used, 0, sizeof used);
    memset (p, -1, sizeof p);
    for (int i=0; i<n; ++i)
        base[i] = i;

    used[root] = true;
    int qh=0, qt=0;
    q[qt++] = root;
    while (qh < qt) {
        int v = q[qh++];
        for (size_t i=0; i<g[v].size(); ++i) {
            int to = g[v][i];
            if (base[v] == base[to] || match[v] == to)  continue;
            if (to == root || match[to] != -1 && p[match[to]] != -1) {
                int curbase = lca (v, to);
                memset (blossom, 0, sizeof blossom);
                mark_path (v, curbase, to);
                mark_path (to, curbase, v);
                for (int i=0; i<n; ++i)
                    if (blossom[base[i]]) {
                        base[i] = curbase;
                        if (!used[i]) {
```

```
                        used[i] = true;
                        q[qt++] = i;
                    }
                }
            }
            else if (p[to] == -1) {
                p[to] = v;
                if (match[to] == -1)
                    return to;
                to = match[to];
                used[to] = true;
                q[qt++] = to;
            }
        }
    }
    return -1;
}

int main() {
    //inittialize graph

    memset (match, -1, sizeof match);
    for (int i=0; i<n; ++i)
        if (match[i] == -1) {
            int v = find_path (i);
            while (v != -1) {
                int pv = p[v],  ppv = match[pv];
                match[v] = pv,  match[pv] = v;
                v = ppv;
            }
        }
}
```

9.6 Java

```
class FastReader {
    BufferedReader br;
    StringTokenizer tk;

    FastReader(InputStreamReader ir) {
        br = new BufferedReader(ir);
    }

    String next() {
        while ((tk == null) || (!tk.hasMoreElements())) {
            tk = new StringTokenizer(nextLine());
        }
        return tk.nextToken();
    }
    String nextLine() {
        try {
            return br.readLine();
        } catch (IOException e) {
            e.printStackTrace();
            return null;
        }
    }

}
```

9.7 Link-Cut Tree

```
const int N=1501000;
struct node {
  node *s[2],*f,*minv;
  int val,d,id;
  bool rev;
  bool isr() { return !f||(f->s[0]!=this && f->s[1]!=this);}
  bool dir() { return f->s[1]==this;}
  void setc(node *c,int d) { s[d]=c;if (c) c->f=this;}
  void push() {
    if (rev) { swap(s[0],s[1]); rep(i,0,2) if (s[i]) s[i]->rev^=1;} rev=0;
  }
  void upd() {
    minv=this; val=d;
    rep(i,0,2) if (s[i]&&s[i]->val>val) val=s[i]->val,minv=s[i]->minv;

  }
}pool[N],*cur;
stack<node*> sta;
```

```
void rot(node *x) {
  node *p=x->f;bool d=x->dir();
  if (!p->isr()) p->f->setc(x,p->dir()); else x->f=p->f;
  p->setc(x->s[!d],d);x->setc(p,!d);
  p->upd();
}
void splay(node *x) {
  node *q=x;
  while (1) { sta.push(q);if (q->isr()) break; q=q->f; }
  while (!sta.empty()) sta.top()->push(),sta.pop();
  while (!x->isr()) {
    if (x->f->isr()) rot(x);
    else if (x->isr()==x->f->isr()) rot(x->f),rot(x);
    else rot(x),rot(x);
  }
  x->upd();
}
node *expose(node *x) {
  node *q=NULL;
  for (;x;x=x->f) splay(x),x->s[1]=q,(q=x)->upd();
  return q;
}
void evert(node *x) { expose(x); splay(x); x->rev^=1; x->push();}
void expose(node *x,node *y) { evert(x); expose(y); splay(x);}
void link(node *x,node *y) { evert(x); evert(y); x->setc(y,1);}
void cut(node *x,node *y) { expose(x,y); x->s[1]=y->f=NULL;}
```

9.8 Minimum directed spanning tree

Given a weighted directed graph with a fixed root vertex r, find a set of edges with minimum total weight that makes all vertices reachable from r.

$O(nm)$ solution

1. For each vertex $v \neq r$, consider the weights of all incoming edges: $w_1, w_2, \ldots, w_k$. Subtract $\min(w_1, w_2, \ldots, w_k)$ from each of $w_1, w_2, \ldots, w_k$. We'll denote this operation as *nullifying* vertex v.
2. If every node is *zero-reachable* from r (reachable by zero-weight edges), break.
3. Compress SCCs consisting of zero-weight edges (or just compress any zero-weight cycle) and repeat.

$O(m \log^\alpha n)$ solution

1. Pick any vertex v that is not zero-reachable from r. Let $v_0 = v$.
2. For each i, nullify v_i and let v_{i+1} be any vertex such that edge (v_{i+1}, v_i) has weight 0.
3. If at some point v_i is zero-reachable from r, return to step 1 and pick another v (if any).
4. If $v_i = v_j$ for $j < i$, we have found a zero-weight cycle. Compress this cycle into a new vertex u.
   Set $v_j \leftarrow u$, set $i \leftarrow j$, and go to step 2.

How to perform these steps efficiently?

- For each vertex v, store all edges incoming into v in a data structure (DS) that allows extracting the smallest element, changing all elements of DS by the same constant, and merging two DS into a single one. Options:
  - `std::set` with a global shift (insert elements from small to large $\rightarrow$ merge in $O(\log^2 n)$ amortized)
  - *treap* with implicit keys, minimum on subtrees and lazy propagation ($O(\log n)$)
  - leftist heap, pairing heap, randomized meldable heap... ($O(\log n)$)
- Keep all vertices in a DSU (disjoint set union). Moving from v_i to v_{i+1}, unite their sets in DSU. Now v is zero-reachable from r iff it is in the same set.

9.9 Dominator tree

Given a directed graph with a fixed root vertex r, vertex u *dominates* vertex v if every path from r to v contains u. Build a tree rooted at r such that u dominates v in the graph iff u is an ancestor of v in the tree.

Vertex $u = idom(v)$ is an *immediate dominator* of vertex v iff u is the parent of v in the dominator tree.

For a directed acyclic graph

1. For each vertex v in order of topological sorting of the graph, we will find $idom(v)$.
2. Consider all edges incoming into v: (u_1, v), (u_2, v), ..., (u_k, v).
3. Every dominator of v must be dominating $u_1, u_2, \ldots, u_k$ at the same time.

4. *idom*(v) is the least common ancestor (LCA) of $u_1, u_2, \ldots, u_k$ in the dominator tree.
5. LCA of two vertices can be found with binary lifting in $O(\log n)$. Jumps of sizes 2^k from v must be calculated after finding *idom*(v). The overall complexity is $O(m \log n)$.

For a general directed graph

1. Perform a depth-first search (DFS) on the graph. Renumerate the vertices in order of first visit.
2. Graph edges can be directed top-to-bottom, bottom-to-top, right-to-left, but not left-to-right in the DFS tree.
3. *Semidominator sdom*(v) of vertex v is the smallest vertex u such that there exists a path $u = v_0, v_1, \ldots, v_k = v$ such that $v_i > v$ for $1 \le i < k$.
4. *sdom*(v) and *idom*(v) are ancestors of v in the DFS tree.
5. Consider the following graph: the edges of the DFS tree plus edges $(sdom(v), v)$ for all $v \neq r$. The dominator tree for this graph is the same as for the original graph. But this graph is acyclic, and we already know how to build the dominator tree for acyclic graphs quickly.

Observations required for finding semidominators:

1. Consider all edges incoming into vertex v: (u_1, v), (u_2, v), …, (u_k, v).
2. If u_i is an ancestor of v in the DFS tree, u_i is a candidate for *sdom*(v).
3. Otherwise, consider all vertices w that are ancestors of u but not of v: *sdom*(w) is a candidate for *sdom*(v).
4. *sdom*(v) is the smallest numbered vertex out of all candidates.

The actual algorithm for finding semidominators in $O(m \log n)$:

1. Place all vertices in a disjoint set union (DSU). Every set in the DSU is a rooted tree. For each vertex v, store the minimum of *sdom* on the path from v to its parent in the DSU set.
2. Process vertices in decreasing order. After processing vertex v, link it to its DFS tree parent in the DSU. Note that rank heuristic can not be applied, but just path compression is enough for $O(\log n)$ per query.
3. When processing vertex v, for each edge (u_i, v) such that u_i is not an ancestor of v, compress the path from u_i to its root in the DSU. Now the minimum of *sdom* we are storing for u_i is the candidate for *sdom*(v).

9.10 Tree hash

Хеш дерева с k поддеревьями, имеющих хеши c_i и глубину h вычисляется как $\prod_{i=1}^{k}(c_i + d_h)$ где d_i — это случайные величины, независимо выбранные для каждой глубины.

9.11 Weighted Matching

```cpp
#include <bits/stdc++.h>
#define DIST(e) (lab[e.u]+lab[e.v]-g[e.u][e.v].w*2)
using namespace std;
typedef long long ll;
const int N=1023,INF=1e9;
struct Edge {
    int u,v,w;
} g[N][N];
int
↪  n,m,n_x,lab[N],match[N],slack[N],st[N],pa[N],flower_from[N][N],S[N],vis[N];
vector<int> flower[N];
deque<int> q;
void update_slack(int u,int x) {
    if(!slack[x]||DIST(g[u][x])<DIST(g[slack[x]][x]))slack[x]=u;
}
void set_slack(int x) {
    slack[x]=0;
    for(int u=1; u<=n; ++u)
        if(g[u][x].w>0&&st[u]!=x&&S[st[u]]==0)update_slack(u,x);
}
void q_push(int x) {
    if(x<=n)return q.push_back(x);
    for(int i=0; i<flower[x].size(); i++)q_push(flower[x][i]);
}
void set_st(int x,int b) {
    st[x]=b;
    if(x<=n)return;
    for(int i=0; i<flower[x].size(); ++i)set_st(flower[x][i],b);
}
int get_pr(int b,int xr) {
    int pr=find(flower[b].begin(),flower[b].end(),xr)-flower[b].begin();
    if(pr%2==1) {
        reverse(flower[b].begin()+1,flower[b].end());
```

```
        return (int)flower[b].size()-pr;
    }
    else return pr;
}
void set_match(int u,int v) {
    match[u]=g[u][v].v;
    if(u<=n)return;
    Edge e=g[u][v];
    int xr=flower_from[u][e.u],pr=get_pr(u,xr);
    for(int i=0; i<pr; ++i)set_match(flower[u][i],flower[u][i^1]);
    set_match(xr,v);
    rotate(flower[u].begin(),flower[u].begin()+pr,flower[u].end());
}
void augment(int u,int v) {
    int xnv=st[match[u]];
    set_match(u,v);
    if(!xnv)return;
    set_match(xnv,st[pa[xnv]]);
    augment(st[pa[xnv]],xnv);
}
int get_lca(int u,int v) {
    static int t=0;
    for(++t; u||v; swap(u,v)) {
        if(u==0)continue;
        if(vis[u]==t)return u;
        vis[u]=t;
        u=st[match[u]];
        if(u)u=st[pa[u]];
    }
    return 0;
}
void add_blossom(int u,int lca,int v) {
    int b=n+1;
    while(b<=n_x&&st[b])++b;
    if(b>n_x)++n_x;
    lab[b]=0,S[b]=0;
    match[b]=match[lca];
    flower[b].clear();
    flower[b].push_back(lca);
    for(int x=u,y; x!=lca; x=st[pa[y]])
        flower[b].push_back(x),flower[b].push_back(y=st[match[x]]),q_push(y);
    reverse(flower[b].begin()+1,flower[b].end());
    for(int x=v,y; x!=lca; x=st[pa[y]])
        flower[b].push_back(x),flower[b].push_back(y=st[match[x]]),q_push(y);
    set_st(b,b);
    for(int x=1; x<=n_x; ++x)g[b][x].w=g[x][b].w=0;
    for(int x=1; x<=n; ++x)flower_from[b][x]=0;
    for(int i=0; i<flower[b].size(); ++i) {
        int xs=flower[b][i];
        for(int x=1; x<=n_x; ++x)
            if(g[b][x].w==0||DIST(g[xs][x])<DIST(g[b][x]))
                g[b][x]=g[xs][x],g[x][b]=g[x][xs];
        for(int x=1; x<=n; ++x)
            if(flower_from[xs][x])flower_from[b][x]=xs;
    }
    set_slack(b);
}
void expand_blossom(int b) {
    for(int i=0; i<flower[b].size(); ++i)
        set_st(flower[b][i],flower[b][i]);
    int xr=flower_from[b][g[b][pa[b]].u],pr=get_pr(b,xr);
    for(int i=0; i<pr; i+=2) {
        int xs=flower[b][i],xns=flower[b][i+1];
        pa[xs]=g[xns][xs].u;
        S[xs]=1,S[xns]=0;
        slack[xs]=0,set_slack(xns);
        q_push(xns);
    }
    S[xr]=1,pa[xr]=pa[b];
    for(int i=pr+1; i<flower[b].size(); ++i) {
        int xs=flower[b][i];
        S[xs]=-1,set_slack(xs);
    }
    st[b]=0;
}
bool on_found_Edge(const Edge &e) {
    int u=st[e.u],v=st[e.v];
    if(S[v]==-1) {
        pa[v]=e.u,S[v]=1;
        int nu=st[match[v]];
```

```
        slack[v]=slack[nu]=0;
        S[nu]=0,q_push(nu);
    }
    else if(S[v]==0) {
        int lca=get_lca(u,v);
        if(!lca)return augment(u,v),augment(v,u),1;
        else add_blossom(u,lca,v);
    }
    return 0;
}
bool matching() {
    fill(S,S+n_x+1,-1),fill(slack,slack+n_x+1,0);
    q.clear();
    for(int x=1; x<=n_x; ++x)
        if(st[x]==x&&!match[x])pa[x]=0,S[x]=0,q_push(x);
    if(q.empty())return 0;
    for(;;) {
        while(q.size()) {
            int u=q.front();
            q.pop_front();
            if(S[st[u]]==1)continue;
            for(int v=1; v<=n; ++v)
                if(g[u][v].w>0&&st[u]!=st[v]) {
                    if(DIST(g[u][v])==0) {
                        if(on_found_Edge(g[u][v]))return 1;
                    }
                    else update_slack(u,st[v]);
                }
        }
        int d=INF;
        for(int b=n+1; b<=n_x; ++b)
            if(st[b]==b&&S[b]==1)d=min(d,lab[b]/2);
        for(int x=1; x<=n_x; ++x)
            if(st[x]==x&&slack[x]) {
                if(S[x]==-1)d=min(d,DIST(g[slack[x]][x]));
                else if(S[x]==0)d=min(d,DIST(g[slack[x]][x])/2);
            }
        for(int u=1; u<=n; ++u) {
            if(S[st[u]]==0) {
                if(lab[u]<=d)return 0;
                lab[u]-=d;
            }
            else if(S[st[u]]==1)lab[u]+=d;
        }
        for(int b=n+1; b<=n_x; ++b)
            if(st[b]==b) {
                if(S[st[b]]==0)lab[b]+=d*2;
                else if(S[st[b]]==1)lab[b]-=d*2;
            }
        q.clear();
        for(int x=1; x<=n_x; ++x)
            if(st[x]==x&&slack[x]&&st[slack[x]]!=x&&DIST(g[slack[x]][x])==0)
                if(on_found_Edge(g[slack[x]][x]))return 1;
        for(int b=n+1; b<=n_x; ++b)
            if(st[b]==b&&S[b]==1&&lab[b]==0)expand_blossom(b);
    }
    return 0;
}
pair<ll,int> weight_blossom() {
    fill(match,match+n+1,0);
    n_x=n;
    int n_matches=0;
    ll tot_weight=0;
    for(int u=0; u<=n; ++u)st[u]=u,flower[u].clear();
    int w_max=0;
    for(int u=1; u<=n; ++u)
        for(int v=1; v<=n; ++v) {
            flower_from[u][v]=(u==v?u:0);
            w_max=max(w_max,g[u][v].w);
        }
    for(int u=1; u<=n; ++u)lab[u]=w_max;
    while(matching())++n_matches;
    for(int u=1; u<=n; ++u)
        if(match[u]&&match[u]<u)
            tot_weight+=g[u][match[u]].w;
    return make_pair(tot_weight,n_matches);
}
int main() {
    cin>>n>>m;
    for(int u=1; u<=n; ++u)
```

```
        for(int v=1; v<=n; ++v)
            g[u][v]=Edge {u,v,0};
    for(int i=0,u,v,w; i<m; ++i) {
        cin>>u>>v>>w;
        g[u][v].w=g[v][u].w=w;
    }
    cout<<weight_blossom().first<<'\n';
    for(int u=1; u<=n; ++u)cout<<match[u]<<' ';
}
```

9.12 Berlekamp-Massey

```
vector<int> berlekamp_massey(vector<int> x){
    vector<int> ls, cur;
    int lf = 0, d = 0;
    for(int i = 0; i < x.size(); ++i){
        ll t = 0;
        for(int j = 0; j < cur.size(); ++j){
            t = (t + 1ll * x[i-j-1] * cur[j]) % MOD;
        }
        if((t - x[i]) % MOD == 0) continue;
        if(cur.empty()){
            cur.resize(i+1);
            lf = i;
            d = (t - x[i]) % MOD;
            continue;
        }
        ll k = -(x[i] - t) * pw(d, MOD - 2) % MOD;
        vector<int> c(i-lf-1);
        c.push_back(k);
        for(auto &j : ls) c.push_back(-j * k % MOD);
        if(c.size() < cur.size()) c.resize(cur.size());
        for(int j = 0; j < cur.size(); ++j){
            c[j] = (c[j] + cur[j]) % MOD;
        }
        if(i-lf+(int)ls.size()>=(int)cur.size()){
            tie(ls, lf, d) = make_tuple(cur, i, (t - x[i]) % MOD);
        }
        cur = c;
    }
    for(auto &i: cur) i = (i % MOD + MOD) % MOD;
    return cur;
}

//for a_i = 2 * a_i-1 + a_i-1 returns {2, 1}
```

Конечные цепные дроби

Пусть a_0 — целое, $a_1, \ldots, a_n$ — натуральные числа. Определим две последовательности

$$p_{-2} = 0, \quad p_{-1} = 1, \quad p_k = a_k p_{k-1} + p_{k-2}, \quad k = 0, \ldots, n,$$
$$q_{-2} = 1, \quad q_{-1} = 0, \quad q_k = a_k q_{k-1} + q_{k-2}, \quad k = 0, \ldots, n.$$

Напоминание: для любого $k = 0, \ldots, n$ справедливы соотношения

а) $p_k/q_k = [a_0; a_1, \ldots, a_k]$;
б) $p_k q_{k-1} - p_{k-1} q_k = (-1)^{k+1}$;
в) $p_k q_{k-2} - p_{k-2} q_k = (-1)^k a_k$;
г) $(p_k, q_k) = 1$.
д) $\frac{p_k}{q_k} - \frac{p_{k-1}}{q_{k-1}} = \frac{(-1)^{k+1}}{q_k q_{k-1}}$; ж) $\frac{p_0}{q_0} < \frac{p_2}{q_2} < \frac{p_4}{q_4} < \ldots \leqslant \frac{p_n}{q_n} \leqslant \ldots < \frac{p_5}{q_5} < \frac{p_3}{q_3} < \frac{p_1}{q_1}$;
е) $\frac{p_k}{q_k} - \frac{p_{k-2}}{q_{k-2}} = \frac{(-1)^k a_k}{q_k q_{k-2}}$; з) $q_k > q_{k-1}$, $q_k > 2q_{k-2}$.

Числа p_k/q_k называются *подходящими дробями* цепной дроби $[a_0; a_1, \ldots, a_n]$.

9.13 Integrals

Table of Integrals*

Basic Forms

$$\int x^n dx = \frac{1}{n+1}x^{n+1} \tag{1}$$

$$\int \frac{1}{x}dx = \ln|x| \tag{2}$$

$$\int u dv = uv - \int v du \tag{3}$$

$$\int \frac{1}{ax+b}dx = \frac{1}{a}\ln|ax+b| \tag{4}$$

Integrals of Rational Functions

$$\int \frac{1}{(x+a)^2}dx = -\frac{1}{x+a} \tag{5}$$

$$\int (x+a)^n dx = \frac{(x+a)^{n+1}}{n+1}, n \neq -1 \tag{6}$$

$$\int x(x+a)^n dx = \frac{(x+a)^{n+1}((n+1)x-a)}{(n+1)(n+2)} \tag{7}$$

$$\int \frac{1}{1+x^2}dx = \tan^{-1}x \tag{8}$$

$$\int \frac{1}{a^2+x^2}dx = \frac{1}{a}\tan^{-1}\frac{x}{a} \tag{9}$$

$$\int \frac{x}{a^2+x^2}dx = \frac{1}{2}\ln|a^2+x^2| \tag{10}$$

$$\int \frac{x^2}{a^2+x^2}dx = x - a\tan^{-1}\frac{x}{a} \tag{11}$$

$$\int \frac{x^3}{a^2+x^2}dx = \frac{1}{2}x^2 - \frac{1}{2}a^2\ln|a^2+x^2| \tag{12}$$

$$\int \frac{1}{ax^2+bx+c}dx = \frac{2}{\sqrt{4ac-b^2}}\tan^{-1}\frac{2ax+b}{\sqrt{4ac-b^2}} \tag{13}$$

$$\int \frac{1}{(x+a)(x+b)}dx = \frac{1}{b-a}\ln\frac{a+x}{b+x},\ a \neq b \tag{14}$$

$$\int \frac{x}{(x+a)^2}dx = \frac{a}{a+x} + \ln|a+x| \tag{15}$$

$$\int \frac{x}{ax^2+bx+c}dx = \frac{1}{2a}\ln|ax^2+bx+c| - \frac{b}{a\sqrt{4ac-b^2}}\tan^{-1}\frac{2ax+b}{\sqrt{4ac-b^2}} \tag{16}$$

Integrals with Roots

$$\int \sqrt{x-a}\,dx = \frac{2}{3}(x-a)^{3/2} \tag{17}$$

$$\int \frac{1}{\sqrt{x\pm a}}dx = 2\sqrt{x\pm a} \tag{18}$$

$$\int \frac{1}{\sqrt{a-x}}dx = -2\sqrt{a-x} \tag{19}$$

$$\int x\sqrt{x-a}\,dx = \frac{2}{3}a(x-a)^{3/2} + \frac{2}{5}(x-a)^{5/2} \tag{20}$$

$$\int \sqrt{ax+b}\,dx = \left(\frac{2b}{3a} + \frac{2x}{3}\right)\sqrt{ax+b} \tag{21}$$

$$\int (ax+b)^{3/2}dx = \frac{2}{5a}(ax+b)^{5/2} \tag{22}$$

$$\int \frac{x}{\sqrt{x\pm a}}dx = \frac{2}{3}(x \mp 2a)\sqrt{x\pm a} \tag{23}$$

$$\int \sqrt{\frac{x}{a-x}}dx = -\sqrt{x(a-x)} - a\tan^{-1}\frac{\sqrt{x(a-x)}}{x-a} \tag{24}$$

$$\int \sqrt{\frac{x}{a+x}}dx = \sqrt{x(a+x)} - a\ln\left[\sqrt{x}+\sqrt{x+a}\right] \tag{25}$$

$$\int x\sqrt{ax+b}\,dx = \frac{2}{15a^2}(-2b^2+abx+3a^2x^2)\sqrt{ax+b} \tag{26}$$

$$\int \sqrt{x(ax+b)}\,dx = \frac{1}{4a^{3/2}}\left[(2ax+b)\sqrt{ax(ax+b)} - b^2\ln\left|a\sqrt{x}+\sqrt{a(ax+b)}\right|\right] \tag{27}$$

$$\int \sqrt{x^3(ax+b)}\,dx = \left[\frac{b}{12a} - \frac{b^2}{8a^2x} + \frac{x}{3}\right]\sqrt{x^3(ax+b)} + \frac{b^3}{8a^{5/2}}\ln\left|a\sqrt{x}+\sqrt{a(ax+b)}\right| \tag{28}$$

$$\int \sqrt{x^2\pm a^2}\,dx = \frac{1}{2}x\sqrt{x^2\pm a^2} \pm \frac{1}{2}a^2\ln\left|x+\sqrt{x^2\pm a^2}\right| \tag{29}$$

$$\int \sqrt{a^2-x^2}\,dx = \frac{1}{2}x\sqrt{a^2-x^2} + \frac{1}{2}a^2\tan^{-1}\frac{x}{\sqrt{a^2-x^2}} \tag{30}$$

$$\int x\sqrt{x^2\pm a^2}\,dx = \frac{1}{3}\left(x^2\pm a^2\right)^{3/2} \tag{31}$$

$$\int \frac{1}{\sqrt{x^2\pm a^2}}dx = \ln\left|x+\sqrt{x^2\pm a^2}\right| \tag{32}$$

$$\int \frac{1}{\sqrt{a^2-x^2}}dx = \sin^{-1}\frac{x}{a} \tag{33}$$

$$\int \frac{x}{\sqrt{x^2\pm a^2}}dx = \sqrt{x^2\pm a^2} \tag{34}$$

$$\int \frac{x}{\sqrt{a^2-x^2}}dx = -\sqrt{a^2-x^2} \tag{35}$$

$$\int \frac{x^2}{\sqrt{x^2\pm a^2}}dx = \frac{1}{2}x\sqrt{x^2\pm a^2} \mp \frac{1}{2}a^2\ln\left|x+\sqrt{x^2\pm a^2}\right| \tag{36}$$

$$\int \sqrt{ax^2+bx+c}\,dx = \frac{b+2ax}{4a}\sqrt{ax^2+bx+c} + \frac{4ac-b^2}{8a^{3/2}}\ln\left|2ax+b+2\sqrt{a(ax^2+bx+c)}\right| \tag{37}$$

$$\int x\sqrt{ax^2+bx+c} = \frac{1}{48a^{5/2}}\Big(2\sqrt{a}\sqrt{ax^2+bx+c} \times \left(-3b^2+2abx+8a(c+ax^2)\right) + 3(b^3-4abc)\ln\left|b+2ax+2\sqrt{a}\sqrt{ax^2+bx+c}\right|\Big) \tag{38}$$

$$\int \frac{1}{\sqrt{ax^2+bx+c}}dx = \frac{1}{\sqrt{a}}\ln\left|2ax+b+2\sqrt{a(ax^2+bx+c)}\right| \tag{39}$$

$$\int \frac{x}{\sqrt{ax^2+bx+c}}dx = \frac{1}{a}\sqrt{ax^2+bx+c} - \frac{b}{2a^{3/2}}\ln\left|2ax+b+2\sqrt{a(ax^2+bx+c)}\right| \tag{40}$$

$$\int \frac{dx}{(a^2+x^2)^{3/2}} = \frac{x}{a^2\sqrt{a^2+x^2}} \tag{41}$$

Integrals with Logarithms

$$\int \ln ax\,dx = x\ln ax - x \tag{42}$$

$$\int \frac{\ln ax}{x}dx = \frac{1}{2}(\ln ax)^2 \tag{43}$$

$$\int \ln(ax+b)dx = \left(x+\frac{b}{a}\right)\ln(ax+b) - x, a \neq 0 \tag{44}$$

$$\int \ln(x^2+a^2)\,\mathrm{d}x = x\ln(x^2+a^2) + 2a\tan^{-1}\frac{x}{a} - 2x \tag{45}$$

$$\int \ln(x^2-a^2)\,\mathrm{d}x = x\ln(x^2-a^2) + a\ln\frac{x+a}{x-a} - 2x \tag{46}$$

$$\int \ln\left(ax^2+bx+c\right)dx = \frac{1}{a}\sqrt{4ac-b^2}\tan^{-1}\frac{2ax+b}{\sqrt{4ac-b^2}} - 2x + \left(\frac{b}{2a}+x\right)\ln\left(ax^2+bx+c\right) \tag{47}$$

$$\int x\ln(ax+b)dx = \frac{bx}{2a} - \frac{1}{4}x^2 + \frac{1}{2}\left(x^2-\frac{b^2}{a^2}\right)\ln(ax+b) \tag{48}$$

$$\int x\ln\left(a^2-b^2x^2\right)dx = -\frac{1}{2}x^2 + \frac{1}{2}\left(x^2-\frac{a^2}{b^2}\right)\ln\left(a^2-b^2x^2\right) \tag{49}$$

Integrals with Exponentials

$$\int e^{ax}dx = \frac{1}{a}e^{ax} \tag{50}$$

$$\int \sqrt{x}e^{ax}dx = \frac{1}{a}\sqrt{x}e^{ax} + \frac{i\sqrt{\pi}}{2a^{3/2}}\operatorname{erf}\left(i\sqrt{ax}\right), \quad \text{where } \operatorname{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt \tag{51}$$

$$\int xe^x dx = (x-1)e^x \tag{52}$$

$$\int xe^{ax}dx = \left(\frac{x}{a} - \frac{1}{a^2}\right)e^{ax} \tag{53}$$

$$\int x^2e^x dx = \left(x^2-2x+2\right)e^x \tag{54}$$

$$\int x^2e^{ax}dx = \left(\frac{x^2}{a} - \frac{2x}{a^2} + \frac{2}{a^3}\right)e^{ax} \tag{55}$$

$$\int x^3e^x dx = \left(x^3-3x^2+6x-6\right)e^x \tag{56}$$

$$\int x^ne^{ax}\,\mathrm{d}x = \frac{x^ne^{ax}}{a} - \frac{n}{a}\int x^{n-1}e^{ax}\mathrm{d}x \tag{57}$$

$$\int x^ne^{ax}\,\mathrm{d}x = \frac{(-1)^n}{a^{n+1}}\Gamma[1+n,-ax], \quad \text{where } \Gamma(a,x) = \int_x^\infty t^{a-1}e^{-t}\,\mathrm{d}t \tag{58}$$

$$\int e^{ax^2}\,\mathrm{d}x = -\frac{i\sqrt{\pi}}{2\sqrt{a}}\operatorname{erf}\left(ix\sqrt{a}\right) \tag{59}$$

$$\int e^{-ax^2}\,\mathrm{d}x = \frac{\sqrt{\pi}}{2\sqrt{a}}\operatorname{erf}\left(x\sqrt{a}\right) \tag{60}$$

$$\int xe^{-ax^2}\,\mathrm{d}x = -\frac{1}{2a}e^{-ax^2} \tag{61}$$

$$\int x^2e^{-ax^2}\,\mathrm{d}x = \frac{1}{4}\sqrt{\frac{\pi}{a^3}}\operatorname{erf}(x\sqrt{a}) - \frac{x}{2a}e^{-ax^2} \tag{62}$$

Integrals with Trigonometric Functions

$$\int \sin ax dx = -\frac{1}{a}\cos ax \tag{63}$$

$$\int \sin^2 ax dx = \frac{x}{2} - \frac{\sin 2ax}{4a} \tag{64}$$

$$\int \sin^n ax dx = -\frac{1}{a}\cos ax \;\; {}_2F_1\left[\frac{1}{2}, \frac{1-n}{2}, \frac{3}{2}, \cos^2 ax\right] \tag{65}$$

$$\int \sin^3 ax dx = -\frac{3\cos ax}{4a} + \frac{\cos 3ax}{12a} \tag{66}$$

$$\int \cos ax dx = \frac{1}{a}\sin ax \tag{67}$$

$$\int \cos^2 ax dx = \frac{x}{2} + \frac{\sin 2ax}{4a} \tag{68}$$

$$\int \cos^p ax dx = -\frac{1}{a(1+p)}\cos^{1+p} ax \times {}_2F_1\left[\frac{1+p}{2}, \frac{1}{2}, \frac{3+p}{2}, \cos^2 ax\right] \tag{69}$$

$$\int \cos^3 ax dx = \frac{3\sin ax}{4a} + \frac{\sin 3ax}{12a} \tag{70}$$

$$\int \cos ax \sin bx dx = \frac{\cos[(a-b)x]}{2(a-b)} - \frac{\cos[(a+b)x]}{2(a+b)}, a \neq b \tag{71}$$

$$\int \sin^2 ax \cos bx dx = -\frac{\sin[(2a-b)x]}{4(2a-b)} + \frac{\sin bx}{2b} - \frac{\sin[(2a+b)x]}{4(2a+b)} \tag{72}$$

$$\int \sin^2 x \cos x dx = \frac{1}{3}\sin^3 x \tag{73}$$

$$\int \cos^2 ax \sin bx dx = \frac{\cos[(2a-b)x]}{4(2a-b)} - \frac{\cos bx}{2b} - \frac{\cos[(2a+b)x]}{4(2a+b)} \tag{74}$$

$$\int \cos^2 ax \sin ax dx = -\frac{1}{3a}\cos^3 ax \tag{75}$$

$$\int \sin^2 ax \cos^2 bx dx = \frac{x}{4} - \frac{\sin 2ax}{8a} - \frac{\sin[2(a-b)x]}{16(a-b)} + \frac{\sin 2bx}{8b} - \frac{\sin[2(a+b)x]}{16(a+b)} \tag{76}$$

$$\int \sin^2 ax \cos^2 ax dx = \frac{x}{8} - \frac{\sin 4ax}{32a} \tag{77}$$

$$\int \tan ax dx = -\frac{1}{a}\ln\cos ax \tag{78}$$

$$\int \tan^2 ax dx = -x + \frac{1}{a}\tan ax \tag{79}$$

$$\int \tan^n ax dx = \frac{\tan^{n+1} ax}{a(1+n)} \times {}_2F_1\left(\frac{n+1}{2}, 1, \frac{n+3}{2}, -\tan^2 ax\right) \tag{80}$$

$$\int \tan^3 ax dx = \frac{1}{a}\ln\cos ax + \frac{1}{2a}\sec^2 ax \tag{81}$$

$$\int \sec x dx = \ln|\sec x + \tan x| = 2\tanh^{-1}\left(\tan\frac{x}{2}\right) \tag{82}$$

$$\int \sec^2 ax dx = \frac{1}{a}\tan ax \tag{83}$$

$$\int \sec^3 x \,\mathrm{d}x = \frac{1}{2}\sec x\tan x + \frac{1}{2}\ln|\sec x + \tan x| \tag{84}$$

$$\int \sec x \tan x dx = \sec x \tag{85}$$

$$\int \sec^2 x \tan x dx = \frac{1}{2}\sec^2 x \tag{86}$$

$$\int \sec^n x \tan x dx = \frac{1}{n}\sec^n x, n \neq 0 \tag{87}$$

$$\int \csc x dx = \ln\left|\tan\frac{x}{2}\right| = \ln|\csc x - \cot x| + C \tag{88}$$

$$\int \csc^2 ax dx = -\frac{1}{a}\cot ax \tag{89}$$

$$\int \csc^3 x dx = -\frac{1}{2}\cot x\csc x + \frac{1}{2}\ln|\csc x - \cot x| \tag{90}$$

$$\int \csc^n x \cot x dx = -\frac{1}{n}\csc^n x, n \neq 0 \tag{91}$$

$$\int \sec x \csc x dx = \ln|\tan x| \tag{92}$$

Products of Trigonometric Functions and Monomials

$$\int x\cos x dx = \cos x + x\sin x \tag{93}$$

$$\int x\cos ax dx = \frac{1}{a^2}\cos ax + \frac{x}{a}\sin ax \tag{94}$$

$$\int x^2\cos x dx = 2x\cos x + \left(x^2 - 2\right)\sin x \tag{95}$$

$$\int x^2\cos ax dx = \frac{2x\cos ax}{a^2} + \frac{a^2x^2 - 2}{a^3}\sin ax \tag{96}$$

$$\int x^n cosx dx = -\frac{1}{2}(i)^{n+1}\left[\Gamma(n+1, -ix) + (-1)^n\Gamma(n+1, ix)\right] \tag{97}$$

$$\int x^n cosax dx = \frac{1}{2}(ia)^{1-n}\left[(-1)^n\Gamma(n+1, -iax) - \Gamma(n+1, ixa)\right] \tag{98}$$

$$\int x\sin x dx = -x\cos x + \sin x \tag{99}$$

$$\int x\sin ax dx = -\frac{x\cos ax}{a} + \frac{\sin ax}{a^2} \tag{100}$$

$$\int x^2\sin x dx = \left(2 - x^2\right)\cos x + 2x\sin x \tag{101}$$

$$\int x^2\sin ax dx = \frac{2 - a^2x^2}{a^3}\cos ax + \frac{2x\sin ax}{a^2} \tag{102}$$

$$\int x^n\sin x dx = -\frac{1}{2}(i)^n\left[\Gamma(n+1, -ix) - (-1)^n\Gamma(n+1, -ix)\right] \tag{103}$$

Products of Trigonometric Functions and Exponentials

$$\int e^x\sin x dx = \frac{1}{2}e^x(\sin x - \cos x) \tag{104}$$

$$\int e^{bx}\sin ax dx = \frac{1}{a^2+b^2}e^{bx}(b\sin ax - a\cos ax) \tag{105}$$

$$\int e^x\cos x dx = \frac{1}{2}e^x(\sin x + \cos x) \tag{106}$$

$$\int e^{bx}\cos ax dx = \frac{1}{a^2+b^2}e^{bx}(a\sin ax + b\cos ax) \tag{107}$$

$$\int xe^x\sin x dx = \frac{1}{2}e^x(\cos x - x\cos x + x\sin x) \tag{108}$$

$$\int xe^x\cos x dx = \frac{1}{2}e^x(x\cos x - \sin x + x\sin x) \tag{109}$$

Integrals of Hyperbolic Functions

$$\int \cosh ax dx = \frac{1}{a}\sinh ax \tag{110}$$

$$\int e^{ax}\cosh bx dx = \begin{cases} \dfrac{e^{ax}}{a^2-b^2}[a\cosh bx - b\sinh bx] & a \neq b \\ \dfrac{e^{2ax}}{4a} + \dfrac{x}{2} & a = b \end{cases} \tag{111}$$

$$\int \sinh ax dx = \frac{1}{a}\cosh ax \tag{112}$$

$$\int e^{ax}\sinh bx dx = \begin{cases} \dfrac{e^{ax}}{a^2-b^2}[-b\cosh bx + a\sinh bx] & a \neq b \\ \dfrac{e^{2ax}}{4a} - \dfrac{x}{2} & a = b \end{cases} \tag{113}$$

$$\int e^{ax}\tanh bx dx = \begin{cases} \dfrac{e^{(a+2b)x}}{(a+2b)}{}_2F_1\left[1+\dfrac{a}{2b}, 1, 2+\dfrac{a}{2b}, -e^{2bx}\right] - \dfrac{1}{a}e^{ax}{}_2F_1\left[\dfrac{a}{2b}, 1, 1E, -e^{2bx}\right] & a \neq b \\ \dfrac{e^{ax} - 2\tan^{-1}[e^{ax}]}{a} & a = b \end{cases} \tag{114}$$

$$\int \tanh ax \, dx = \frac{1}{a}\ln\cosh ax \tag{115}$$

$$\int \cos ax\cosh bx dx = \frac{1}{a^2+b^2}\left[a\sin ax\cosh bx + b\cos ax\sinh bx\right] \tag{116}$$

$$\int \cos ax\sinh bx dx = \frac{1}{a^2+b^2}\left[b\cos ax\cosh bx + a\sin ax\sinh bx\right] \tag{117}$$

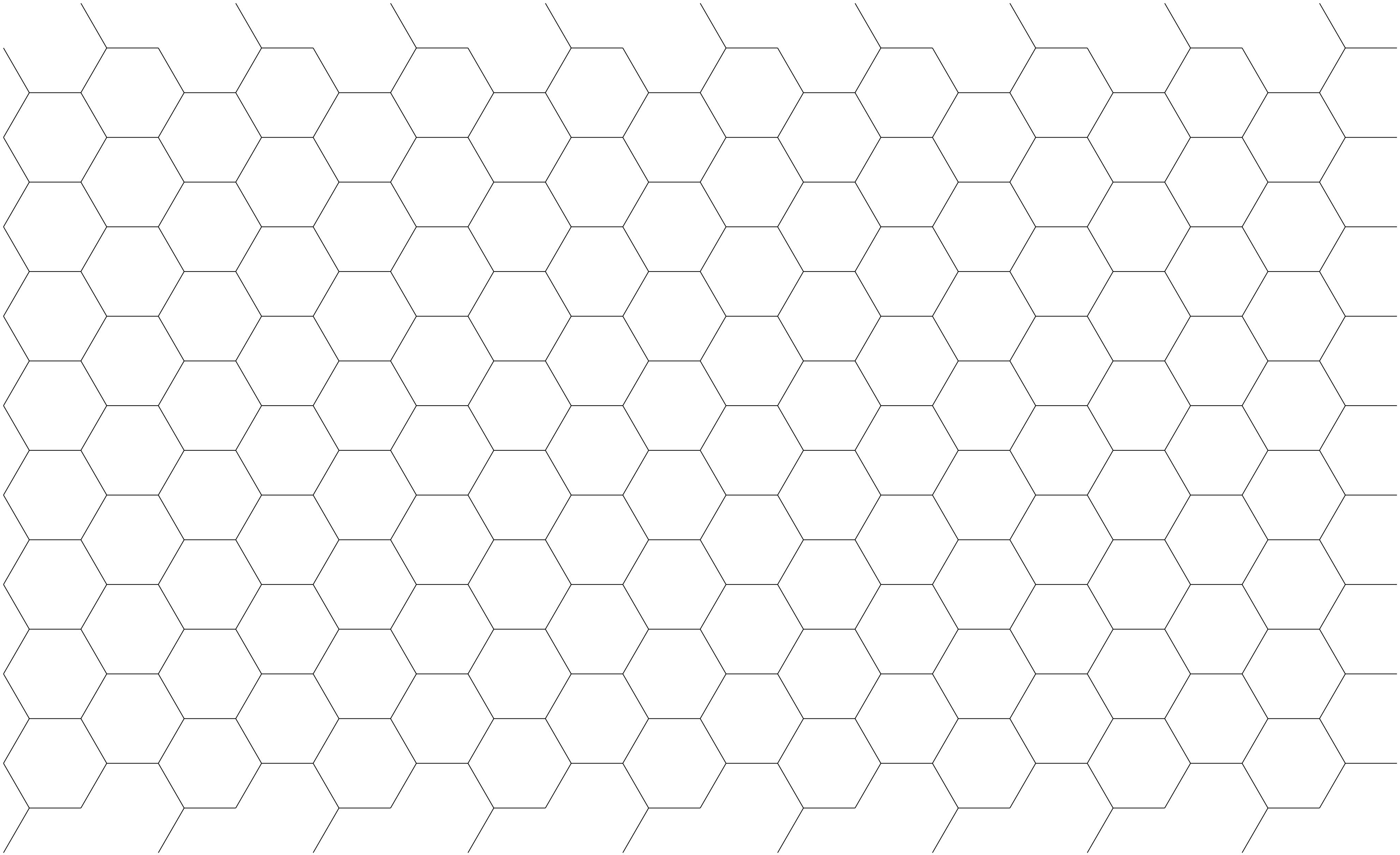
$$\int \sin ax\cosh bx dx = \frac{1}{a^2+b^2}\left[-a\cos ax\cosh bx + b\sin ax\sinh bx\right] \tag{118}$$

$$\int \sin ax\sinh bx dx = \frac{1}{a^2+b^2}\left[b\cosh bx\sin ax - a\cos ax\sinh bx\right] \tag{119}$$

$$\int \sinh ax\cosh ax dx = \frac{1}{4a}\left[-2ax + \sinh 2ax\right] \tag{120}$$

$$\int \sinh ax\cosh bx dx = \frac{1}{b^2-a^2}\left[b\cosh bx\sinh ax - a\cosh ax\sinh bx\right] \tag{121}$$

9.14 Hexagon



Быстрое преобразование Фурье, многочлены и ряды Трактуем дискретное преобразование Фурье как вычисление $\sum\limits_{t=0}^{k-1} a_t x^t$ в корнях из единицы $\left\{e^{i\frac{2\pi}{k}t}\right\}_{t=0}^{k-1}$. С ним можно производить следующие операции:

1. Умножение многочленов. Вычисляем, умножаем покомпонентно, возвращаемся к коэффициентам.

2. Вычисление обратного ряда. Для формальных рядов вида $P = a_0 + xA$ существует обратный ряд $P^{-1}P = 1$. Пусть мы умеем вычислять первые t его коэффициентов (для $t = 1$ имеем a_0^{-1}). Обозначим соответствующий многочлен как P_t. Имеем $P_tP = 1 \mod z^t$, т.е. $P_tP = 1 + z^tQ(x)$. Пусть $P_{2t} = P_t + z^tR(x)$, тогда $P_{2t}P = P_tP + z^tRP = 1 + z^t(Q + RP) = 1 \mod z^{2t}$, откуда $R = -QP_t \mod z^t$. Таким образом, $P_{2t} = P_t + (1 - P_tP)P_t = P_t(2 - P_tP)$, что позволяет нам вычислять P_t за $O(t \log t)$.

3. Деление многочленов с остатком. Есть $A(x)$ степени m и $B(x)$ степени $n < m$. Нужно найти $D(x)$, $R(x)$ такие что $A(x) = D(x) \cdot B(x) + R(x)$ и степень $R(x)$ меньше n. Запишем это в матричном виде.

$$\begin{pmatrix} a_m \\ a_{m-1} \\ \dots \\ \dots \\ a_1 \\ a_0 \end{pmatrix} = \begin{pmatrix} b_n & 0 & 0 & \dots & 0 & 0 \\ b_{n-1} & b_n & 0 & \dots & 0 & 0 \\ b_{n-2} & b_{n-1} & b_n & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & b_0 & b_1 \\ 0 & 0 & 0 & \dots & 0 & b_0 \end{pmatrix} \begin{pmatrix} d_{m-n} \\ d_{m-n-1} \\ \dots \\ \dots \\ d_1 \\ d_0 \end{pmatrix} + \begin{pmatrix} 0 \\ \dots \\ 0 \\ r_{n-1} \\ \dots \\ r_0 \end{pmatrix}$$

В уравнениях на первые $m - n + 1$ координат не входят $\{r_k\}$, поэтому, рассматривая систему

$$\begin{pmatrix} b_n & 0 & \dots & 0 \\ b_{n-1} & b_n & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ b_{2n-m} & b_{2n-m+1} & \dots & b_n \end{pmatrix} \begin{pmatrix} d_{m-n} \\ d_{m-n-1} \\ \dots \\ d_0 \end{pmatrix} = \begin{pmatrix} a_m \\ a_{m-1} \\ \dots \\ a_n \end{pmatrix}$$

можно найти $\{d_k\}$. Развернув последовательности $b'_k = b_{n-k}$, $d'_k = d_{m-n-k}$, $a'_k = a_{m-k}$, получаем

$$\begin{pmatrix} b'_0 & 0 & \dots & 0 \\ b'_1 & b'_0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ b'_{m-n} & b'_{m-n-1} & \dots & b'_0 \end{pmatrix} \begin{pmatrix} d'_0 \\ d'_1 \\ \dots \\ d'_{m-n} \end{pmatrix} = \begin{pmatrix} a'_0 \\ a'_1 \\ \dots \\ a'_{m-n} \end{pmatrix}$$

Перепишем в виде $A^r = B^rD^r \mod z^{m-n+1}$, где $A^r = \sum\limits_{t=0}^{m-n} a'_t x^t$, $B^r = \sum\limits_{t=0}^{m-n} b'_t x^t$, $D^r = \sum\limits_{t=0}^{m-n} d'_t x^t$.
Отсюда $D^r = A^r(B^r)^{-1} \mod z^{m-n+1}$, что можно вычислить по предыдущему пункту.

Быстрее, но менее наглядно: пусть $x = z^{-1}$ и $A^r = z^mA$, $B^r = z^nB$, $D^r = z^{m-n}D$, $R^r = z^{n-1}R$ – многочлены с развёрнутыми коэффициентами. Тогда $A^r = D^rB^r + z^{m-n+1}R^r = D^rB^r \mod z^{m-n+1}$.

4. Вычисление $\ln P$. $(\ln P)' = \frac{P'}{P}$, что вычисляется с помощью нахождения обратного ряда.

5. Многоточечное вычисление. Нужно вычислить $P(x_i)$ для $\{x_i\}_{i=1}^n$. Учитывая $P(x_i) = P \mod (x - x_i)$, вычислим $P \mod \prod\limits_{i=1}^{n/2-1}(x - x_i)$ и $P \mod \prod\limits_{i=n/2}^{n}(x - x_i)$ и запустимся рекурсивно. Получим $O(n \log^2 n)$.

6. Интерполяция. Дан набор точек $\{(x_i, y_i)\}_{i=0}^{n-1}$, нужно найти $P : P(x_i) = y_i$. Пусть мы нашли полином P_1 для первых $n/2$ точек. Тогда $P = P_1 + P_2 \prod\limits_{i=0}^{n/2-1}(x - x_i) = P_1 + P_2Q$. Нахождение P_2 сведём к интерполяции и многоточечному вычислению: $P_2(x_i) = \frac{y_i - P_1(x_i)}{Q(x_i)}$ для $i > n/2$. Получим $O(n \log^3 n)$.