



National Research University Higher School of Economics

# BasketBobr

Alexey Mikhnenko, Anton Stepanov, Alexey Vasilyev

December 14, 2025

# Contest (1)

template.cpp	42 lines
<pre>#ifndef LOCAL #define _GLIBCXX_DEBUG #endif #include &lt;bits/stdc++.h&gt; using namespace std;  using ll = long long; using ld = long double; using ull = unsigned long long;  #define pbc push_back #define mp make_pair #define all(v) begin(v),end(v) #define vin(v) for (auto &amp;el : a) cin &gt;&gt; el  mt19937 rnd(chrono::steady_clock::now().time_since_epoch().count());  template &lt;typename T1, typename T2&gt; inline void chkmin(T1 &amp;x, const T2 &amp;y) {     if (y &lt; x) {         x = y;     } }  template &lt;typename T1, typename T2&gt; inline void chkmax(T1 &amp;x, const T2 &amp;y) {     if (x &lt; y) {         x = y;     } }  void solve() {  }  signed main() {     cin.tie(0)-&gt;sync_with_stdio(0);     cout.precision(20), cout.setf(ios::fixed);     int t = 1;     // cin &gt;&gt; t;     while (t--) {         solve();     } }</pre>	

genfolders.sh	6 lines
<pre>chmod +x bld* for f in {A..Z} do     mkdir \$f     cp main.cpp bld* \$f done</pre>	

bld	1 lines
<pre>g++ -std=c++20 -g -DLOCAL -fsanitize=address,bounds,undefined -o \$1 \$1.cpp</pre>	

bldf	1 lines
<pre>g++ -std=c++20 -g -O2 -o \$1 \$1.cpp</pre>	

hacks.sh	2 lines
<pre>UBSAN_OPTIONS=print_stacktrace=1 ./main gdb rbreak regex</pre>	

hash.sh	3 lines
<pre># Hashes a file, ignoring all whitespace and comments. # Use for verifying that code was correctly typed. cpp -dD -P -fpreprocessed   tr -d '[:space:]'  md5sum  cut -c-6</pre>	

clion.cpp	2 lines
<pre>set(CMAKE_CXX_STANDARD 20) set(CMAKE_CXX_FLAGS "-DLOCAL")</pre>	
C++ (2)	
GpHashtable.cpp	
Description: Hash map with mostly the same API as unordered_map, but ~3x faster. Uses 1.5x memory. Initial capacity must be a power of 2 (if provided).	

#include <ext/pb_ds/assoc_container.hpp> #include <ext/pb_ds/tree_policy.hpp> using namespace __gnu_pbds;  const int RANDOM =     chrono::high_resolution_clock::now().time_since_epoch().count(); struct hasher {     int operator()(int x) const { return x ^ RANDOM; } };  gp_hash_table<int, int, hasher> table;	
--	--

OrderedSet.cpp	
Description: A set (not multiset!) with support for finding the n'th element, and finding the index of an element. To get a map, change null_type.	
Time: O(log(n))	
<bits/extc++.h>, <bits/stdc++.h>	
dffb260, 37 lines	
using namespace __gnu_pbds; using namespace std;  template <typename T> using ordered_set =     tree<T, null_type, less<>, rb_tree_tag,         tree_order_statistics_node_update>;  int main() {     ordered_set<int> X;     X.insert(1);     X.insert(2);     X.insert(4);     X.insert(8);     X.insert(16);      assert(*X.find_by_order(1) == 2);     assert(*X.find_by_order(2) == 4);     assert(*X.find_by_order(4) == 16);     assert(X.find_by_order(6) ==X.end());	

assert(X.order_of_key(-5) == 0); assert(X.order_of_key(1) == 0); assert(X.order_of_key(3) == 2); assert(X.order_of_key(4) == 2); assert(X.order_of_key(400) == 5); // std::cout << *X.find_by_order(1) << std::endl; // 2 // std::cout << *X.find_by_order(2) << std::endl; // 4 // std::cout << *X.find_by_order(4) << std::endl; // 16 // std::cout << (end(X) == X.find_by_order(6)) << std::endl; // true  // std::cout << X.order_of_key(-5) << std::endl; // 0 // std::cout << X.order_of_key(1) << std::endl; // 0 // std::cout << X.order_of_key(3) << std::endl; // 2 // std::cout << X.order_of_key(4) << std::endl; // 2 // std::cout << X.order_of_key(400) << std::endl; // 5 return 0;	
{	

bitset.cpp	521d1f, 2 lines
Description: bitset	
bs._Find_first() bs._Find_next(idx) - returns right after	
alloc.cpp	
Description: fastalloc	
8726b1, 11 lines	
const int MAX_MEM = 1e8;	

int mpos = 0; char mem[MAX_MEM]; inline void *operator new(size_t n) {     assert((mpos += n) <= MAX_MEM);     return (void *) (mem + mpos - n); }  void operator delete(void *) noexcept {} // must have! void operator delete(void *, size_t) noexcept {} // must have!	
---	--

fastio.cpp	
Description: fastio	
79fd14, 52 lines	
inline int readChar(); template <class T = int> inline T readInt(); template <class T> inline void writeInt(T x, char end = 0); inline void writeChar(int x); inline void writeWord(const char *s); static const int buf_size = 4096; inline int getChar() {     static char buf[buf_size];     static int len = 0, pos = 0;     if (pos == len) pos = 0, len = fread(buf, 1, buf_size, stdin);     if (pos == len) return -1;     return buf[pos++]; }  inline int readChar() {     int c = getChar();     while (c <= 32) c = getChar();     return c; }  template <class T> inline T readInt() {     int s = 1, c = readChar();     T x = 0;     if (c == '-') s = -1, c = getChar();     while ('0' <= c && c <= '9') x = x * 10 + c - '0', c = getChar();     return s == 1 ? x : -x; }  static int write_pos = 0; static char write_buf[buf_size]; inline void writeChar(int x) {     if (write_pos == buf_size)         fwrite(write_buf, 1, buf_size, stdout), write_pos = 0;     write_buf[write_pos++] = x; }  template <class T> inline void writeInt(T x, char end) {     if (x < 0) writeChar('-', x = -x);     char s[24];     int n = 0;     while (x    !n) s[n++] = '0' + x % 10, x /= 10;     while (n--) writeChar(s[n]);     if (end) writeChar(end); }  inline void writeWord(const char *s) {     while (*s) writeChar(*s++); }  struct Flusher {     ~Flusher() {         if (write_pos) fwrite(write_buf, 1, write_pos, stdout), write_pos = 0;     } } flusher;	

# Strings (3)

Manacher.cpp	
Description: Manacher algorithm	
Time: O(n)	
a6ddfb, 27 lines	
vector<int> manacherOdd(string s) {     int n = s.size();     vector<int> dl(n);	

```
int l = 0, r = -1;
for (int i = 0; i < n; ++i) {
    int k = i > r ? 1 : min(d1[l + r - i], r - i + 1);
    while (i + k < n && i - k >= 0 && s[i + k] == s[i - k])
        ++k;
    d1[i] = k;
    if (i + k - 1 > r)
        l = i - k + 1, r = i + k - 1;
}

vector<int> manacherEven(string s) {
    int n = s.size();
    vector<int> d2(n);
    l = 0, r = -1;
    for (int i = 0; i < n; ++i) {
        int k = i > r ? 0 : min(d2[l + r - i + 1], r - i + 1);
        while (i + k < n && i - k - 1 >= 0 && s[i + k] == s[i - k - 1])
            ++k;
        d2[i] = k;
        if (i + k - 1 > r)
            l = i - k, r = i + k - 1;
    }
}
```

AhoCorasick.cpp

Description: Build aho-corasick automaton.  
Time:  $\mathcal{O}(n)$  ae5fc2, 19 lines

```
int go(int v, char c);

int get_link(int v) {
    if (t[v].link == -1)
        if (v == 0 || t[v].p == 0)
            t[v].link = 0;
        else
            t[v].link = go(get_link(t[v].p), t[v].pch);
    return t[v].link;
}

int go(int v, char c) {
    if (t[v].go[c] == -1)
        if (t[v].next[c] != -1)
            t[v].go[c] = t[v].next[c];
        else
            t[v].go[c] = v == 0 ? 0 : go(get_link(v), c);
    return t[v].go[c];
}
```

SuffixArray.cpp

Description: Build suffix array  
Time:  $\mathcal{O}(n \log(n))$  5bd011, 47 lines

```
vector<int> buildSuffixArray(string &s) {
    // Remove, if you want to sort cyclic shifts
    s += (char)(1);
    int n = s.size();
    vector<int> a(n);
    iota(all(a), 0);
    stable_sort(all(a), [&](int i, int j) { return s[i] < s[j]; });
    vector<int> c(n);
    int cc = 0;
    for (int i = 0; i < n; ++i) {
        if (i == 0 || s[a[i]] != s[a[i - 1]]) {
            c[a[i]] = cc++;
        } else {
            c[a[i]] = c[a[i - 1]];
        }
    }
    for (int L = 1; L < n; L *= 2) {
        vector<int> cnt(n);
        for (auto i : c) {
            cnt[i]++;
        }
        vector<int> pref(n);
        for (int i = 1; i < n; ++i) {
            pref[i] = pref[i - 1] + cnt[i - 1];
        }
        vector<int> na(n);
```

```
for (int i = 0; i < n; ++i) {
    int pos = (a[i] - L + n) % n;
    na[pref[c[pos]]++] = pos;
}
a = na;
vector<int> nc(n);
cc = 0;
for (int i = 0; i < n; ++i) {
    if (i == 0 || c[a[i]] != c[a[i - 1]] ||
        c[(a[i] + L) % n] != c[(a[i - 1] + L) % n]) {
        nc[a[i]] = cc++;
    } else {
        nc[a[i]] = nc[a[i - 1]];
    }
}
c = nc;
}
a.erase(a.begin());
s.pop_back();
return a;
}
```

Lcp.cpp

Description: lcp array  
Time:  $\mathcal{O}(n)$  1cc27c, 43 lines

```
vector<int> perm;
vector<int> buildLCP(string &s, vector<int> &a) {
    int n = s.size();
    vector<int> ra(n);
    for (int i = 0; i < n; ++i) {
        ra[a[i]] = i;
    }
    vector<int> lcp(n - 1);
    int cur = 0;
    for (int i = 0; i < n; ++i) {
        cur--;
        chkmax(cur, 0);
        if (ra[i] == n - 1) {
            cur = 0;
            continue;
        }
        int j = a[ra[i] + 1];
        while (s[i + cur] == s[j + cur]) cur++;
        lcp[ra[i]] = cur;
    }
    perm.resize(a.size());
    for (int i = 0; i < a.size(); ++i) perm[a[i]] = i;
    return lcp;
}

int cntr[MAXN];
int spt[MAXN][lggl];
void build(vector<int> &a) {
    for (int i = 0; i < a.size(); ++i) {
        spt[i][0] = a[i];
    }
    for (int i = 2; i < MAXN; ++i) cntr[i] = cntr[i / 2] + 1;
    for (int h = 1; (1 << (h - 1)) < a.size(); ++h) {
        for (int i = 0; i + (1 << (h - 1)) < a.size(); ++i) {
            spt[i][h] = min(spt[i][h - 1], spt[i + (1 << (h - 1))][h - 1]);
        }
    }
}

int getLCP(int l, int r) {
    l = perm[l], r = perm[r];
    if (l > r) swap(l, r);
    int xx = cntr[r - l];
    return min(spt[l][xx], spt[r - (1 << xx)][xx]);
}
```

Eertree.cpp

Description: Palindrome Tree  
Time:  $\mathcal{O}(n)$  6e64b6, 49 lines

```
struct palindromic_tree {
    int new_node() {
        tree.push_back(node());
        return static_cast<int>(tree.size()) - 1;
    }
```

```
}

int find_suffix(int v) {
    int n = str.size();
    while (tree[v].length == n - 1 || str.back() != str[n - 2 - tree[v].length]) {
        v = tree[v].suflink;
    }
    return v;
}

struct node {
    int length = 0, suflink = -1, to[ALPHABET];
    node() { memset(to, -1, sizeof(to)); }
};

int even, odd, last;
vector<node> tree;
vector<int> str;

palindromic_tree() {
    odd = new_node();
    even = new_node();
    tree[even].suflink = tree[odd].suflink = odd;
    tree[odd].length = -1;
    last = even;
}

void add(int symbol) {
    str.push_back(symbol);
    last = find_suffix(last);
    if (tree[last].to[symbol] == -1) {
        int v = new_node();
        tree[v].length = tree[last].length + 2;
        int u = find_suffix(tree[last].suflink);
        if (tree[u].to[symbol] != -1) {
            tree[v].suflink = tree[u].to[symbol];
        }
        else {
            tree[v].suflink = even;
        }
        tree[last].to[symbol] = v;
    }
    last = tree[last].to[symbol];
}
};
```

SuffixAutomaton.cpp

Description: Build suffix automaton.  
Time:  $\mathcal{O}(n)$  340cda, 53 lines

```
const int alpha = 26;

struct state {
    int len, link;
    array<int, alpha> next;
};

state st[MAXLEN * 2];
int sz, last;

void sa_init() {
    sz = last = 0;
    st[0].len = 0;
    st[0].link = -1;
    st[0].next.fill(-1);
    ++sz;
}

int sa_cut(int v, int c) {
    assert(st[v].next[c] != -1);
    int to = st[v].next[c];
    if (st[to].len == st[v].len + 1) {
        return to;
    }
    int clone = sz++;
    st[clone].len = st[v].len + 1;
    st[clone].next = st[to].next;
    st[clone].link = st[to].link;
    for (; v != -1 && st[v].next[c] == to; v = st[v].link)
```

```
        st[v].next[c] = clone;
        st[to].link = clone;
        return clone;
    }

    void sa_extend(int c) {
        if (st[last].next[c] != -1) {
            int to = sa_cut(last, c);
            last = to;
            return;
        }
        int cur = sz++;
        st[cur].len = st[last].len + 1;
        st[cur].next.fill(-1);
        int v;
        for (v = last; v != -1 && st[v].next[c] == -1; v = st[v].link)
            st[v].next[c] = cur;
        if (v == -1)
            st[cur].link = 0;
        else {
            int to = st[v].next[c];
            st[cur].link = sa_cut(v, c);
        }
        last = cur;
    }
}
```

PrefixZ.cpp  
Description: Calculates Prefix,Z-functions  
Time:  $\mathcal{O}(n)$

```
vector<int> pf(string s) {
    int k = 0;
    vector<int> p(s.size());
    for (int i = 1; i < s.size(); ++i) {
        while (k && s[i] != s[k])
            k = p[k - 1];
        k += (s[i] == s[k]);
        p[i] = k;
    }
    return p;
}

vector<int> zf(string s) {
    int n = s.size();
    vector<int> z(n, 0);
    for (int i = 1, l = 0, r = 0; i < n; ++i) {
        if (i <= r)
            z[i] = min(r - i + 1, z[i - 1]);
        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;
    }
    z[0] = n;
    return z;
}
```

MinShift.cpp  
Description: Calculates min-cyclic-shift of s, Duval decomposition  
Time:  $\mathcal{O}(n)$

```
string minshift(string s) {
    int i = 0, ans = 0;
    s += s; // Remove for lyndon decomposition
    int n = s.size();
    while (i < n / 2) { // (i < n) lyndon
        ans = i;
        int j = i + 1, k = i;
        while (j < n && s[k] <= s[j]) {
            if (s[k] < s[j])
                k = i;
            else
                ++k;
            ++j;
        }
        while (i <= k) {
            // s.substr(i, j-k) - simple string
            i += j - k;
        }
    }
}
```

```
        return s.substr(ans, n / 2);
    }
}
```

SA-IS.cpp  
Description: Build suffix array  
Time:  $\mathcal{O}(n)$

```
void induced_sort(vector<int> &vec, int LIM, vector<int> &sa, vector<bool>
    > &sl,
                    vector<int> &fx) {
    vector<int> l(LIM), r(LIM);
    for (int c : vec) {
        if (c + 1 < LIM) {
            ++l[c + 1];
        }
        ++r[c];
    }
    partial_sum(all(l), l.begin());
    partial_sum(all(r), r.begin());
    fill(all(sa), -1);
    for (int i = fx.size() - 1; i >= 0; --i) {
        sa[--r[vec[fx[i]]]] = fx[i];
    }
    for (int i : sa) {
        if (i >= 1 && sl[i - 1]) {
            sa[l[vec[i - 1]]++] = i - 1;
        }
    }
    fill(all(r), 0);
    for (int c : vec) ++r[c];
    partial_sum(all(r), r.begin());
    for (int k = sa.size() - 1, i = sa[k]; k >= 1; --k, i = sa[k])
        if (i >= 1 && !sl[i - 1]) sa[--r[vec[i - 1]]] = i - 1;
}

vector<int> SA-IS(vector<int> &vec, int LIM) {
    const int n = vec.size();
    vector<int> sa(n), fx;
    vector<bool> sl(n);
    sl[n - 1] = false;
    for (int i = n - 2; i >= 0; --i) {
        sl[i] = (vec[i] > vec[i + 1] || (vec[i] == vec[i + 1] && sl[i + 1]));
        if (sl[i] && !sl[i + 1]) {
            fx.pb(i + 1);
        }
    }
    reverse(all(fx));
    induced_sort(vec, LIM, sa, sl, fx);
    vector<int> nfx(fx.size()), lmv(fx.size());
    for (int i = 0, k = 0; i < n; ++i) {
        if (!sl[sa[i]] && sa[i] >= 1 && sl[sa[i] - 1]) {
            nfx[k++] = sa[i];
        }
    }
    int cur = 0;
    sa[n - 1] = cur;
    for (int k = 1; k < nfx.size(); ++k) {
        int i = nfx[k - 1], j = nfx[k];
        if (vec[i] != vec[j]) {
            sa[j] = ++cur;
            continue;
        }
        bool flag = false;
        for (int a = i + 1, b = j + 1;; ++a, ++b) {
            if (vec[a] != vec[b]) {
                flag = true;
                break;
            }
            if ((!sl[a] && sl[a - 1]) || (!sl[b] && sl[b - 1])) {
                flag = !((!sl[a] && sl[a - 1]) && (!sl[b] && sl[b - 1]));
                break;
            }
        }
        sa[j] = (flag ? ++cur : cur);
    }
    for (int i = 0; i < fx.size(); ++i) {
        lmv[i] = sa[fx[i]];
    }
    if (cur + 1 < (int)fx.size()) {
```

```
        auto lms = SA-IS(lmv, cur + 1);
        for (int i = 0; i < fx.size(); ++i) {
            nfx[i] = fx[lms[i]];
        }
    }
    induced_sort(vec, LIM, sa, sl, nfx);
    return sa;
}

template <typename T>
vector<int> suffix_array(T &s, const int LIM = 128) {
    vector<int> vec(s.size() + 1);
    copy(all(s), begin(vec));
    vec.back() = (char)(1);
    auto ret = SA-IS(vec, LIM);
    ret.erase(ret.begin());
    return ret;
}
```

Graph (4)

Hungarian.cpp  
Description: Hungarian algorithm  
Time:  $\mathcal{O}(n^3)$

```
int n, m;
vector<vector<int>>> a;
vector<int> u(n + 1), v(m + 1), p(m + 1), way(m + 1);
for (int i = 1; i <= n; ++i) {
    p[0] = i;
    int j0 = 0;
    vector<int> minv(m + 1, INF);
    vector<char> used(m + 1, false);
    do {
        used[j0] = true;
        int i0 = p[j0], delta = INF, j1;
        for (int j = 1; j <= m; ++j)
            if (!used[j]) {
                int cur = a[i0][j] - u[i0] - v[j];
                if (cur < minv[j])
                    minv[j] = cur, way[j] = j0;
                if (minv[j] < delta)
                    delta = minv[j], j1 = j;
            }
        for (int j = 0; j <= m; ++j)
            if (used[j])
                u[p[j]] += delta, v[j] -= delta;
            else
                minv[j] -= delta;
        j0 = j1;
    } while (p[j0] != 0);
    do {
        int j1 = way[j0];
        p[j0] = p[j1];
        j0 = j1;
    } while (j0);
}
```

```
// matching
vector<int> ans(n + 1);
for (int j = 1; j <= m; ++j) {
    ans[p[j]] = j;
}
```

```
// cost
int cost = -v[0];
```

BlossomShrinking.cpp  
Description: Maximum matching in general graph  
Time:  $\mathcal{O}(n^3)$

```
struct Edge {
    int u, v;
};
const int N = 510;
int n, m;
vector<int> g[N];
```

```
vector<Edge> perfectMatching;
int match[N], par[N], base[N];
bool used[N], blossom[N], lcaUsed[N];
int lca(int u, int v) {
    fill(lcaUsed, lcaUsed + n, false);
    while (u != -1) {
        u = base[u];
        lcaUsed[u] = true;
        if (match[u] == -1)
            break;
        u = par[match[u]];
    }
    while (v != -1) {
        v = base[v];
        if (lcaUsed[v])
            return v;
        v = par[match[v]];
    }
    assert(false);
    return -1;
}

void markPath(int v, int myBase, int children) {
    while (base[v] != myBase) {
        blossom[v] = blossom[match[v]] = true;
        par[v] = children;
        children = match[v];
        v = par[match[v]];
    }
}

int findPath(int root) {
    iota(base, base + n, 0);
    fill(par, par + n, -1);
    fill(used, used + n, false);
    queue<int> q;
    q.push(root);
    used[root] = true;
    while (!q.empty()) {
        int v = q.front();
        q.pop();
        for (auto to : g[v]) {
            if (match[v] == to)
                continue;
            if (base[v] == base[to])
                continue;
            if (to == root || (match[to] != -1 && par[match[to]] != -1))
                {
                    fill(blossom, blossom + n, false);
                    int myBase = lca(to, v);
                    markPath(v, myBase, to);
                    markPath(to, myBase, v);
                    for (int u = 0; u < n; ++u) {
                        if (!blossom[base[u]])
                            continue;
                        base[u] = myBase;
                        if (used[u])
                            continue;
                        used[u] = true;
                        q.push(u);
                    }
                }
            else if (par[to] == -1) {
                par[to] = v;
                if (match[to] == -1)
                    return to;
            }
            used[match[to]] = true;
            q.push(match[to]);
        }
    }
    return -1;
}

void blossomShrinking() {
    fill(match, match + n, -1);
    for (int v = 0; v < n; ++v) {
        if (match[v] != -1)
            continue;
        int nxt = findPath(v);
        while (nxt != -1) {
            int parV = par[nxt];
```

```
            int parParV = match[parV];
            match[nxt] = parV;
            match[parV] = nxt;
            nxt = parParV;
        }
    }
    for (int v = 0; v < n; ++v) {
        if (match[v] != -1 && v < match[v]) {
            perfectMatching.push_back({v, match[v]});
        }
    }
}

signed main() {
    cin >> n;
    int u, v;
    set<pair<int, int>> edges;
    while (cin >> u >> v) {
        --u;
        --v;
        if (u > v)
            swap(u, v);
        if (edges.count({u, v}))
            continue;
        edges.insert({u, v});
        g[u].push_back(v);
        g[v].push_back(u);
    }
    blossomShrinking();
    cout << perfectMatching.size() * 2 << '\n';
    for (auto i : perfectMatching) {
        cout << i.u + 1 << " " << i.v + 1 << "\n";
    }
    return 0;
}

Lct.cpp
Description: link-cut tree
Time:  $O(n \log(n))$ 
3d8a3f, 136 lines

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

const int MAXN = 1e5 + 228;

struct node {
    node *ch[2];
    node *p;
    bool rev;
    int sz;

    node() {
        ch[0] = ch[1] = p = NULL;
        rev = false;
        sz = 1;
    }
};

int getsz(node *n) { return (n == NULL) ? 0 : n->sz; }

void pull(node *n) { n->sz = getsz(n->ch[0]) + getsz(n->ch[1]) + 1; }

void push(node *n) {
    if (n->rev) {
        if (n->ch[0]) {
            n->ch[0]->rev ^= 1;
        }
        if (n->ch[1]) {
            n->ch[1]->rev ^= 1;
        }
        swap(n->ch[0], n->ch[1]);
        n->rev = 0;
    }
}

bool isRoot(node *n) {
    return n->p == NULL || (n->p->ch[0] != n && n->p->ch[1] != n);
}

int chnum(node *n) { return n->p->ch[1] == n; }
```

```
void attach(node *n, node *p, int num) {
    if (n != NULL)
        n->p = p;
    if (p != NULL)
        p->ch[num] = n;
}

void rotate(node *n) {
    int num = chnum(n);
    node *p = n->p;
    node *b = n->ch[1 - num];
    n->p = p->p;
    if (!isRoot(p)) {
        p->p->ch[chnum(p)] = n;
    }
    attach(p, n, 1 - num);
    attach(b, p, num);
    pull(p);
    pull(n);
}

node *qq[MAXN];

void splay(node *n) {
    node *nn = n;
    int top = 0;
    qq[top++] = nn;
    while (!isRoot(nn)) {
        nn = nn->p;
        qq[top++] = nn;
    }
    while (top) {
        push(qq[--top]);
    }
    while (!isRoot(n)) {
        if (!isRoot(n->p)) {
            if (chnum(n) == chnum(n->p)) {
                rotate(n->p);
            } else {
                rotate(n);
            }
        }
        rotate(n);
    }
}

void expose(node *n) {
    splay(n);
    n->ch[1] = NULL;
    pull(n);
    while (n->p != NULL) {
        splay(n->p);
        attach(n, n->p, 1);
        pull(n->p);
        splay(n);
    }
}

void makeRoot(node *n) {
    expose(n);
    n->rev ^= 1;
}

node *nodes[MAXN];

int main() {
    int n;
    cin >> n;
    for (int i = 0; i <= n; i++) {
        nodes[i] = new node();
    }
    int q;
    cin >> q;
    while (q--) {
        string s;
        cin >> s;
        int u, v;
        cin >> u >> v;
```

```
        makeRoot(nodes[u]);
        makeRoot(nodes[v]);
        if (s == "get") {
            if (isRoot(nodes[u]) && u != v) {
                cout << "-1" << endl;
            } else {
                cout << getsz(nodes[v]) - 1 << endl;
            }
        } else if (s == "link") {
            nodes[v]->p = nodes[u];
        } else {
            push(nodes[v]);
            nodes[v]->ch[1] = NULL;
            nodes[u]->p = NULL;
        }
    }
}
```

MaxFlow.cpp  
Description: Dinic  
Time:  $O(n^2m)$

1c1bc8, 72 lines

```
struct MaxFlow {
    const int inf = 1e9 + 20;
    struct edge {
        int a, b, cap;
    };
    int n;
    vector<edge> e;
    vector<vector<int>> g;
    MaxFlow() {}
    int s, t;
    vector<int> d, ptr;
    void init(int n_, int s_, int t_) {
        s = s_, t = t_, n = n_;
        g.resize(n);
        ptr.resize(n);
    }
    void addedge(int a, int b, int cap) {
        g[a].pb(e.size());
        e.pb({a, b, cap});
        g[b].pb(e.size());
        e.pb({b, a, 0});
    }
    bool bfs() {
        d.assign(n, inf);
        d[s] = 0;
        queue<int> q;
        q.push(s);
        while (q.size()) {
            int v = q.front();
            q.pop();
            for (int i : g[v]) {
                if (e[i].cap > 0) {
                    int b = e[i].b;
                    if (d[b] > d[v] + 1) {
                        d[b] = d[v] + 1;
                        q.push(b);
                    }
                }
            }
        }
    }
    int dfs(int v, int flow) {
        if (v == t) return flow;
        if (!flow) return 0;
        int sum = 0;
        for (; ptr[v] < g[v].size(); ++ptr[v]) {
            int b = e[g[v][ptr[v]]].b;
            int cap = e[g[v][ptr[v]]].cap;
            if (cap <= 0) continue;
            if (d[b] != d[v] + 1) continue;
            int x = dfs(b, min(flow, cap));
            int id = g[v][ptr[v]];
            e[id].cap -= x;
            e[id ^ 1].cap += x;
            flow -= x;
        }
    }
    return d[t] != inf;
}
```

```
        sum += x;
    }
    return sum;
}
int dinic() {
    int ans = 0;
    while (1) {
        if (!bfs()) break;
        ptr.assign(n, 0);
        int x = dfs(s, inf);
        if (!x) break;
        ans += x;
    }
    return ans;
}
};
```

MCMF.cpp  
Description: Min cost  
Time:  $O(?)$

32340a, 61 lines

```
struct MCMF {
    struct edge {
        int a, b, cap, cost;
    };
    vector<edge> e;
    vector<vector<int>> g;
    int s, t;
    int n;
    void init(int N, int S, int T) {
        s = S, t = T, n = N;
        g.resize(N);
        e.clear();
    }
    void addedge(int a, int b, int cap, int cost) {
        g[a].pb(e.size());
        e.pb({a, b, cap, cost});
        g[b].pb(e.size());
        e.pb({b, a, 0, -cost});
    }
    int getcost(int k) {
        int flow = 0;
        int cost = 0;
        while (flow < k) {
            vector<int> d(n, INF);
            vector<int> pr(n);
            d[s] = 0;
            queue<int> q;
            q.push(s);
            while (q.size()) {
                int v = q.front();
                q.pop();
                for (int i : g[v]) {
                    int u = e[i].b;
                    if (e[i].cap && d[u] > d[v] + e[i].cost) {
                        d[u] = d[v] + e[i].cost;
                        q.push(u);
                        pr[u] = i;
                    }
                }
            }
            if (d[t] == INF) return INF;
            int gf = k - flow;
            int v = t;
            while (v != s) {
                int id = pr[v];
                chkmin(gf, e[id].cap);
                v = e[id].a;
            }
            v = t;
            while (v != s) {
                int id = pr[v];
                e[id].cap -= gf;
                e[id ^ 1].cap += gf;
                cost += e[id].cost * gf;
                v = e[id].a;
            }
            flow += gf;
        }
    }
}
```

```
        return cost;
    }
};

MCMFfast.cpp
Description: Min cost with potentials
Time:  $O(?)$ 
363228, 86 lines

struct MCMF {
    struct edge {
        int a, b, cap, cost;
    };
    vector<edge> e;
    vector<vector<int>> g;
    vector<ll> po;
    int s, t;
    int n;
    void init(int N, int S, int T) {
        s = S, t = T, n = N;
        g.resize(N);
        e.clear();
    }
    void addedge(int a, int b, int cap, int cost) {
        g[a].pb(e.size());
        e.pb({a, b, cap, cost});
        g[b].pb(e.size());
        e.pb({b, a, 0, -cost});
    }
    void calc_p() {
        po.assign(n, INF);
        vector<int> inq(n);
        queue<int> q;
        q.push(s);
        po[s] = 0;
        while (!q.empty()) {
            int v = q.front();
            q.pop();
            inq[v] = 0;
            for (auto i : g[v]) {
                if (po[e[i].b] > po[v] + e[i].cost && e[i].cap) {
                    po[e[i].b] = po[v] + e[i].cost;
                    if (!inq[e[i].b]) q.push(e[i].b);
                    inq[e[i].b] = 1;
                }
            }
        }
    }
    ll getcost(int k) {
        calc_p();
        int flow = 0;
        ll cost = 0;
        while (flow < k) {
            vector<ll> d(n, INF);
            vector<int> pr(n);
            d[s] = 0;
            set<pair<ll, int>> q;
            q.insert(mp(0ll, s));
            while (q.size()) {
                int v = q.begin()->second;
                q.erase(q.begin());
                for (int i : g[v]) {
                    int u = e[i].b;
                    if (e[i].cap && d[u] > d[v] + e[i].cost + po[v] - po[e[i].b]) {
                        q.erase(mp(d[u], u));
                        d[u] = d[v] + e[i].cost + po[v] - po[e[i].b];
                        q.insert(mp(d[u], u));
                        pr[u] = i;
                    }
                }
            }
            if (d[t] == INF) return INF;
            for (int i = 0; i < n; ++i) {
                if (d[i] != INF) po[i] += d[i];
            }
            int gf = k - flow;
            int v = t;
            while (v != s) {
                int id = pr[v];
                chkmin(gf, e[id].cap);
            }
        }
    }
}
```

```
        v = e[id].a;
    }
    v = t;
    while (v != s) {
        int id = pr[v];
        e[id].cap -= gf;
        e[id ^ 1].cap += gf;
        cost += 1ll * e[id].cost * gf;
        v = e[id].a;
    }
    flow += gf;
}
return cost;
}
};
```

GlobalMincut.cpp

Description: Global min cut

Time:  $\mathcal{O}(n^3)$  7b8a6b, 35 lines

```
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;
        }
    }
}
```

WeightedMatching.cpp

Description: Max weighted matching

Time:  $\mathcal{O}(N^3)$  or so c3f149, 193 lines

```
#define Dist(e) (lab[e.u] + lab[e.v] - g[e.u][e.v].w * 2)
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;
    while(1) {
        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)
            chkmin(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);
        }
    }
    ll answer = 0;
    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);
}
```

### DominatorTree.cpp

Description: Dominator tree

Time: ? e82004, 52 lines

```
struct DominatorTree {
    vector<basic_string<int>> g, rg, bucket;
    basic_string<int> arr, par, rev, sdom, dom, dsu, label;
    int n, t;

    DominatorTree(int n) : g(n), rg(n), bucket(n), arr(n, -1), par(n, -1),
        rev(n, -1),
        sdom(n, -1), dom(n, -1), dsu(n, 0), label(n, 0), n(n), t(0) {}
    void add_edge(int u, int v) {
        g[u] += v;
    }
    void dfs(int u) {
        arr[u] = t;
        rev[t] = u;
        label[t] = sdom[t] = dsu[t] = t;
        t++;
        for (int w : g[u]) {
            if (arr[w] == -1) {
                dfs(w);
                par[arr[w]] = arr[u];
            }
            rg[arr[w]] += arr[u];
        }
    }
    int find(int u, int x=0) {
        if (u == dsu[u]) return x ? -1 : u;
        int v = find(dsu[u], x + 1);
        if (v < 0) return u;
        if (sdom[label[dsu[u]]] < sdom[label[u]])
            label[u] = label[dsu[u]];
        dsu[u] = v;
        return x ? v : label[u];
    }
    vector<int> run(int root) {
        dfs(root);
        iota(dom.begin(), dom.end(), 0);
        for (int i = t - 1; i >= 0; --i) {
            for (int w : rg[i]) sdom[i] = min(sdom[i], sdom[find(w)]);
            if (i) bucket[sdom[i]] += i;
            for (int w : bucket[i]) {
                int v = find(w);
                if (sdom[v] == sdom[w]) dom[w] = sdom[w];
                else dom[w] = v;
            }
            if (i > 1) dsu[i] = par[i];
        }
    }
}
```

```
        for (int i = 1; i < t; i++) if (dom[i] != sdom[i]) dom[i] = dom[
            dom[i]];
        vector<int> outside_dom(n, -1);
        for (int i = 1; i < t; i++) outside_dom[rev[i]] = rev[dom[i]];
        // -1 if vertex is not reachable
        return outside_dom;
    }
};
```

### OrientedSpanningTree.cpp

Description: Oriented Spanning Tree

Time:  $\mathcal{O}(n \log n)$ ? 3d7a73, 96 lines

```
struct RollbackUF {
    vector<int> p, sz;
    vector<int> changes;
    RollbackUF(int n) {
        p.resize(n);
        changes.reserve(n);
        sz.resize(n, 1);
        for (int i = 0; i < n; ++i) p[i] = i;
    }
    int time() {
        return changes.size();
    }
    int find(int v) {
        if (v == p[v]) return v;
        return find(p[v]);
    }
    bool join(int a, int b) {
        a = find(a);
        b = find(b);
        if (a == b) return false;
        if (sz[a] > sz[b]) swap(a, b);
        changes.push_back(a);
        sz[b] += sz[a];
        p[a] = b;
        return true;
    }
    void rollback(int t) {
        while (changes.size() > t) {
            int v = changes.back();
            sz[p[v]] -= sz[v];
            p[v] = v;
            changes.pop_back();
        }
    }
};
struct Edge { int a, b; ll w; };
struct Node {
    Edge key;
    Node *l, *r;
    ll delta;
    void prop() {
        key.w += delta;
        if (l) l->delta += delta;
        if (r) r->delta += delta;
        delta = 0;
    }
    Edge top() { prop(); return key; }
};
Node *merge(Node *a, Node *b) {
    if (!a || !b) return a ? b;
    a->prop(), b->prop();
    if (a->key.w > b->key.w) swap(a, b);
    swap(a->l, (a->r = merge(b, a->r)));
    return a;
}
void pop(Node& a) { a->prop(); a = merge(a->l, a->r); }
pair<ll, vi> dmst(int n, int r, vector<Edge>& g) {
    RollbackUF uf(n);
    vector<Node*> heap(n);
    for (Edge e : g) heap[e.b] = merge(heap[e.b], new Node(e));
    ll res = 0;
    vi seen(n, -1), path(n), par(n);
    seen[r] = r;
    vector<Edge> Q(n), in(n, {-1, -1}), comp;
    deque<tuple<int, int, vector<Edge>>> cys;
    for (int s = 0; s < n; ++s) {
```

```
        int u = s, qi = 0, w;
        while (seen[u] < 0) {
            if (!heap[u]) return {-1, {}};
            Edge e = heap[u]->top();
            heap[u]->delta -= e.w, pop(heap[u]);
            Q[qi] = e, path[qi++] = u, seen[u] = s;
            res += e.w, u = uf.find(e.a);
            if (seen[u] == s) {
                Node* cyc = 0;
                int end = qi, time = uf.time();
                do {
                    cyc = merge(cyc, heap[w = path[--qi]]);
                } while (uf.join(u, w));
                u = uf.find(u), heap[u] = cyc, seen[u] = -1;
                cys.push_front({u, time, {Q[qi], &Q[end]}});
            }
        }
        for (int i = 0; i < qi; ++i) {
            in[uf.find(Q[i].b)] = Q[i];
        }
    }
    for (auto& [u, t, comp] : cys) { // restore so l ( optional )
        uf.rollback(t);
        Edge inEdge = in[u];
        for (auto& e : comp) in[uf.find(e.b)] = e;
        in[uf.find(inEdge.b)] = inEdge;
    }
    for (int i = 0; i < n; ++i) par[i] = in[i].a;
    return {res, par};
}
```

### MatroidIntersection.cpp

Description: matroid interstion

Time: ? d2387f, 71 lines

```
template<typename T, typename A, typename B>
vector<T> matroid_intersection(const std::vector<T> &ground_set, const A
    &matroid1, const B &matroid2) {
    //weighted - minimize (weight, cnt edges) with dijkstra
    int n = ground_set.size();
    vector<char> in_set(n), inm1(n), inm2(n);
    vector<bool> used(n);
    vi par(n), left, right;
    while (true) {
        A m1 = matroid1;
        B m2 = matroid2;
        left.clear(); right.clear();
        for (int i = 0; i < n; i++)
            if (in_set[i]) {
                m1.add(ground_set[i]);
                m2.add(ground_set[i]);
                left.push_back(i);
            } else {
                right.push_back(i);
            }
        fill(all(inm1), 0); fill(all(inm2), 0);
        bool found = false;
        for (int i : right) {
            inm1[i] = m1.independed_with(ground_set[i]);
            inm2[i] = m2.independed_with(ground_set[i]);
            if (inm1[i] && inm2[i]) {
                in_set[i] = 1;
                found = true;
                break;
            }
        }
        if (found) continue;
        fill(all(used), false); fill(all(par), -1);
        queue<int> que;
        for (int i : right) if (inm1[i]) {
            used[i] = true;
            que.push(i);
        }
        while (!que.empty() && !found) {
            int v = que.front();
            que.pop();
            if (in_set[v]) {
                A m = matroid1;
                for (int i : left) if (i != v) m.add(ground_set[i]);
```



```
for (int u : right)
    if (!used[u] && m.independed_with(ground_set[u])) {
        par[u] = v;
        used[u] = true;
        que.push(u);
        if (inm2[u]) {
            found = true;
            for (; u != -1; u = par[u]) in_set[u] ^= 1;
            break;
        }
    }
} else {
    B m = m2;
    m.add_extra(ground_set[v]);
    for (auto u : left)
        if (!used[u] && m.independed_without(ground_set[u]))
            {
                par[u] = v;
                used[u] = true;
                que.push(u);
            }
    }
}
if (!found) break;
}
vector<T> res;
for (int i = 0; i < n; i++) if (in_set[i]) res.push_back(ground_set[i]);
return res;
}
```

MinMeanCycle.cpp

Description:

MINIMUM MEAN CYCLE ALGORITHM

Input:

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

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  $\max_{\substack{0 \leq k \leq n-1 \\ F_k(x) < \infty}} \frac{F_n(x) - F_k(x)}{n - k}$  is minimum.

⑥

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

d41d8c, 1 lines

Geometry (5)

Point.cpp

Description: struct Point

80dfd5, 80 lines

```
const ld EPS = 1e-7;

ld sq(ld x) {
    return x * x;
}

int sign(ld x) {
    if (x < -EPS) {
        return -1;
    }
    if (x > EPS) {
```

```
return 1;
}
return 0;
}

#define vec point
struct point { // % - cross, * - dot
    ld x, y;
    auto operator<=>(const point&) const = default;
};
ld operator*(const point &a, const point &b) {
    return a.x * b.x + a.y * b.y;
}
ld operator%(const point &a, const point &b) {
    return a.x * b.y - a.y * b.x;
}
point operator-(const point &a, const point &b) {
    return {a.x - b.x, a.y - b.y};
}
point operator+(const point &a, const point &b) {
    return {a.x + b.x, a.y + b.y};
}
point operator*(const point &a, ld b) {
    return {a.x * b, a.y * b};
}
point operator/(const point &a, ld b) {
    return {a.x / b, a.y / b};
}
bool operator<(const point &a, const point &b) {
    if (sign(a.y - b.y) != 0) {
        return a.y < b.y;
    } else if (sign(a.x - b.x) != 0) {
        return a.x < b.x;
    }
    return 0;
}
ld len2(const point &a) {
    return sq(a.x) + sq(a.y);
}
ld len(const point &a) {
    return sqrt(len2(a));
}
point norm(point a) {
    return a / len(a);
}
int half(point a) {
    return (sign(a.y) == -1 || (sign(a.y) == 0 && a.x < 0));
}
point ort(point a) {
    return {-a.y, a.x};
}
point turn(point a, ld ang) {
    return {a.x * cos(ang) - a.y * sin(ang), a.x * sin(ang) + a.y * cos(ang)};
}
ld getAngle(point &a, point &b) {
    return atan2(a % b, a * b);
}
bool cmpHalf(const point &a, const point &b) {
    if (half(a) != half(b)) {
        return half(b);
    } else {
        int sgn = sign(a % b);
        if (!sgn) {
            return len2(a) < len2(b);
        } else {
            return sgn == 1;
        }
    }
}
}
```

Line.cpp

Description: struct Line

887306, 26 lines

```
struct line {
    ld a, b, c;
    void norm() {
        // for half planes
        ld d = len({a, b});
```

```
assert(sign(d) > 0);
a /= d;
b /= d;
c /= d;
}
ld eval(point p) const { return a * p.x + b * p.y + c; }
bool isIn(point p) const { return sign(eval(p)) >= 0; }
bool operator==(const line &other) const {
    return sign(a * other.b - b * other.a) == 0 &&
        sign(a * other.c - c * other.a) == 0 &&
        sign(b * other.c - c * other.b) == 0;
}
};
line getln(point a, point b) {
    line res;
    res.a = a.y - b.y;
    res.b = b.x - a.x;
    res.c = -(res.a * a.x + res.b * a.y);
    res.norm();
    return res;
}
```

Intersections.cpp

Description: Geometry intersections

45d7d9, 75 lines

```
bool isCrossed(ld lx, ld rx, ld ly, ld ry) {
    if (lx > rx)
        swap(lx, rx);
    if (ly > ry)
        swap(ly, ry);
    return sign(min(rx, ry) - max(lx, ly)) >= 0;
}

// if two segments [a, b] and [c, d] has AT LEAST one common point -> true
bool intersects(const point &a, const point &b, const point &c, const point &d) {
    if (!isCrossed(a.x, b.x, c.x, d.x))
        return false;
    if (!isCrossed(a.y, b.y, c.y, d.y))
        return false;
    if (sign((b - a) % (c - a)) * sign((b - a) % (d - a)) == 1) return 0;
    if (sign((d - c) % (a - c)) * sign((d - c) % (b - c)) == 1) return 0;
    return 1;
}

//intersecting lines
bool intersect(line l, line m, point &I) {
    ld d = l.b * m.a - m.b * l.a;
    if (sign(d) == 0) {
        return false;
    }
    ld dx = m.b * l.c - m.c * l.b;
    ld dy = m.c * l.a - l.c * m.a;
    I = {dx / d, dy / d};
    return true;
}

//intersecting circles
int intersect(point o1, ld r1, point o2, ld r2, point &i1, point &i2) {
    if (r1 < r2) {
        swap(o1, o2);
        swap(r1, r2);
    }
    if (sign(r1 - r2) == 0 && len2(o2 - o1) < EPS) {
        return 3;
    }
    ld ln = len(o1 - o2);
    if (sign(ln - r1 - r2) == 1 || sign(r1 - ln - r2) == 1) {
        return 0;
    }
    ld d = (sq(r1) - sq(r2) + sq(ln)) / 2 / ln;
    vec v = norm(o2 - o1);
    point a = o1 + v * d;
    if (sign(ln - r1 - r2) == 0 || sign(ln + r2 - r1) == 0) {
        i1 = a;
        return 1;
    }
    v = ort(v) * sqrt(sq(r1) - sq(d));
    i1 = a + v;
    i2 = a - v;
```

```
        return 2;
    }
    //intersecting line and circle, line should be normed
int intersect(point o, ld r, line l, point &i1, point &i2) {
    ld len = abs(l.eval(o));
    int sgn = sign(len - r);
    if (sgn == 1) {
        return 0;
    }
    vec v = norm(vec{l.a, l.b}) * len;
    if (sign(l.eval(o + v)) != 0) {
        v = vec{0, 0} - v;
    }
    point a = o + v;
    if (sgn == 0) {
        i1 = a;
        return 1;
    }
    v = norm({-l.b, l.a}) * sqrt(sq(r) - sq(len));
    i1 = a + v;
    i2 = a - v;
    return 2;
}
```

Tangents.cpp

Description: Tangents to circles.

c73373, 43 lines

```
// tangents from point to circle
int tangents(point &o, ld r, point &p, point &i1, point &i2) {
    ld ln = len(o - p);
    int sgn = sign(ln - r);
    if (sgn == -1) {
        return 0;
    } else if (sgn == 0) {
        i1 = p;
        return 1;
    } else {
        ld x = sq(r) / ln;
        vec v = norm(p - o) * x;
        point a = o + v;
        v = ort(norm(p - o)) * sqrt(sq(r) - sq(x));
        i1 = a + v;
        i2 = a - v;
        return 2;
    }
}

void _tangents(point c, ld r1, ld r2, vector<line> &ans) {
    ld r = r2 - r1;
    ld z = sq(c.x) + sq(c.y);
    ld d = z - sq(r);
    if (sign(d) == -1)
        return;
    d = sqrt(abs(d));
    line l;
    l.a = (c.x * r + c.y * d) / z;
    l.b = (c.y * r - c.x * d) / z;
    l.c = r1;
    ans.push_back(l);
}

// tangents between two circles
vector<line> tangents(point o1, ld r1, point o2, ld r2) {
    vector<line> ans;
    for (int i = -1; i <= 1; i += 2)
        for (int j = -1; j <= 1; j += 2)
            _tangents(o2 - o1, r1 * i, r2 * j, ans);
    for (int i = 0; i < (int)ans.size(); ++i)
        ans[i].c -= ans[i].a * o1.x + ans[i].b * o1.y;
    return ans;
}
```

Hull.cpp

Description: Polygon functions

fc1928, 16 lines

```
vector<point> hull(vector<point> p, bool need_all=false) {
    sort(all(p));
    p.erase(unique(all(p)), end(p));
    int n = p.size(), k = 0;
    if (n <= 2) return p;
```

```
    vector<point> ch(2 * n);
    ld th = need_all ? -EPS : +EPS; // 0 : 1 if int
    for (int i = 0; i < n; ch[k++] = p[i++]) {
        while (k >= 2 && (ch[k - 1] - ch[k - 2]) % (p[i] - ch[k - 1]) < th)
            --k;
    }
    for (int i = n - 2, t = k + 1; i >= 0; ch[k++] = p[i--]) {
        while (k >= t && (ch[k - 1] - ch[k - 2]) % (p[i] - ch[k - 1]) < th)
            --k;
    }
    ch.resize(k - 1);
    return ch;
}
```

IsInPolygon.cpp

Description: Is in polygon functions

f17b31, 65 lines

```
bool isOnSegment(point &a, point &b, point &x) {
    if (sign(len2(a - b)) == 0) {
        return sign(len(a - x)) == 0;
    }
    return sign((b - a) % (x - a)) == 0 && sign((b - x) * (a - x)) <= 0;
    // optional (slower, but works better if there are some precision
    // problems) return sign((b - a).len() - (x - a).len() - (x - b).len
    // ())
    // == 0;
}

int isIn(vector<point> &p, point &a) {
    int n = p.size();
    // depends on limitations(2*MAXC + 228)
    point b = a + point(2e9 + 228, 1);
    int cnt = 0;
    for (int i = 0; i < n; ++i) {
        point x = p[i];
        point y = p[i + 1 < n ? i + 1 : 0];
        if (isOnSegment(x, y, a)) {
            // depends on the problem statement
            return 1;
        }
        cnt += intersects(x, y, a, b);
    }
    return 2 * (cnt % 2 == 1);
    /*optional (atan2 is VERY SLOW)!
    ld ans = 0;
    int n = p.size();
    for (int i = 0; i < n; ++i) {
        Point x = p[i];
        Point y = p[i + 1 < n ? i + 1 : 0];
        if (isOnSegment(x, y, a)) {
            // depends on the problem statement
            return true;
        }
        x = x - a;
        y = y - a;
        ans += atan2(x ^ y, x * y);
    }
    return abs(ans) > 1;*/
}
```

```
bool isInTriangle(point &a, point &b, point &c, point &x) {
    return sign((b - a) % (x - a)) >= 0 && sign((c - b) % (x - b)) >= 0
        &&
        sign((a - c) % (x - c)) >= 0;
}

// points should be in the counterclockwise order
bool isInConvex(vector<point> &p, point &a) {
    int n = p.size();
    assert(n >= 3);
    // assert(isConvex(p));
    // assert(isCounterclockwise(p));
    if (sign((p[1] - p[0]) % (a - p[0])) < 0)
        return 0;
    if (sign((p[n - 1] - p[0]) % (a - p[0])) > 0)
        return 0;
    int pos = lower_bound(p.begin() + 2, p.end(), a,
        [&](point a, point b) -> bool {
            return sign((a - p[0]) % (b - p[0])) > 0;
```

```
        }) -
        p.begin());
    assert(pos > 1 && pos < n);
    return isInTriangle(p[0], p[pos - 1], p[pos], a);
}
```

Diameter.cpp

Description: Rotating calipers.

Time:  $O(n)$

0f341c, 21 lines

```
ld diameter(vector<point> p) {
    p = hull(p);
    int n = p.size();
    if (n <= 1) {
        return 0;
    }
    if (n == 2) {
        return len(p[0] - p[1]);
    }
    ld ans = 0;
    int i = 0, j = 1;
    while (i < n) {
        while (sign((p[(i + 1) % n] - p[i]) % (p[(j + 1) % n] - p[j])) >=
            0) {
            chkmax(ans, len(p[i] - p[j]));
            j = (j + 1) % n;
        }
        chkmax(ans, len(p[i] - p[j]));
        ++i;
    }
    return ans;
}
```

TangentsAlex.cpp

Description: Find both tangets to the convex polygon.  
(Zakaldovany algos mozhet sgonyat za pivom tak zhe).

Time:  $O(\log(n))$

2eeea8, 17 lines

```
pair<int, int> tangents_alex(vector<point> &p, point &a) {
    int n = p.size();
    int l = __lg(n);
    auto findWithSign = [&](int val) {
        int i = 0;
        for (int k = 1; k >= 0; --k) {
            int i1 = (i - (1 << k) + n) % n;
            int i2 = (i + (1 << k)) % n;
            if (sign((p[i1] - a) % (p[i] - a)) == val)
                i = i1;
            if (sign((p[i2] - a) % (p[i] - a)) == val)
                i = i2;
        }
        return i;
    };
    return {findWithSign(1), findWithSign(-1)};
}
```

IsHpiEmpty.cpp

Description: Determines is half plane intersectinos.

Time:  $O(n)$  (expected)

3b5e69, 42 lines

```
// all lines must be normed!!!!, sign > 0
bool isHpiEmpty(vector<line> lines) {
    // return hpi(lines).empty();
    // overflow/precision problems?
    shuffle(all(lines), rnd);
    const ld C = 1e9;
    point ans(C, C);
    vector<point> box = {{-C, -C}, {C, -C}, {C, C}, {-C, C}};
    for (int i = 0; i < 4; ++i)
        lines.push_back(getIn(box[i], box[(i + 1) % 4]));
    int n = lines.size();
    for (int i = n - 4; i >= 0; --i) {
        if (lines[i].isIn(ans))
            continue;
        point up(0, C + 1), down(0, -C - 1), pi = {lines[i].b, -lines[i].
            a};
        for (int j = i + 1; j < n; ++j) {
            if (lines[i] == lines[j])
                continue;
```

```
point p, pj = {lines[j].b, -lines[j].a};
if (!intersect(lines[i], lines[j], p)) {
    if (sign(pi * pj) != -1)
        continue;
    if (sign(lines[i].c + lines[j].c) *
        (!sign(pi.y) ? sign(pi.x) : -1) ==
            1)
        return true;
} else {
    if ((!sign(pi.y) ? sign(pi.x) : sign(pi.y)) * (sign(pi %
        pj)) ==
            1)
        chkmin(up, p);
    else
        chkmax(down, p);
}
}
if ((ans = up) < down)
    return true;
}
// for (int i = 0; i < n; ++i) {
//     assert(lines[i].eval(ans) < EPS);
// }
return false;
}
```

HalfPlaneIntersection.cpp  
Description: Find the intersection of the half planes.  
Time:  $\mathcal{O}(n \log(n))$

fdf28f, 62 lines

```
vec getPoint(line l) { return {-l.b, l.a}; }

bool bad(line a, line b, line c) {
    point x;
    assert(intersect(b, c, x) == 1);
    return a.eval(x) < 0;
}

// Do not forget about the bounding box
vector<point> hpi(vector<line> lines) {
    sort(all(lines), [](line al, line bl) -> bool {
        point a = getPoint(al);
        point b = getPoint(bl);
        if (half(a) != half(b)) {
            return half(a) < half(b);
        }
        return a % b > 0;
    });

    vector<pair<line, int>> st;
    for (int it = 0; it < 2; it++) {
        for (int i = 0; i < (int)lines.size(); i++) {
            bool flag = false;
            while (!st.empty()) {
                if (len(getPoint(st.back().first) - getPoint(lines[i])) <
                    EPS) {
                    if (lines[i].c >= st.back().first.c) {
                        flag = true;
                        break;
                    } else {
                        st.pop_back();
                    }
                } else if (getPoint(st.back().first) % getPoint(lines[i])
                    < EPS / 2) {
                    return {};
                } else if (st.size() >= 2 &&
                    bad(st[st.size() - 2].first, st[st.size() -
                        1].first,
                        lines[i])) {
                    st.pop_back();
                } else {
                    break;
                }
            }
            if (!flag)
                st.push_back({lines[i], i});
        }
    }
}
```

```
vector<int> en(lines.size(), -1);
vector<point> ans;
for (int i = 0; i < (int)st.size(); i++) {
    if (en[st[i].second] == -1) {
        en[st[i].second] = i;
        continue;
    }
    for (int j = en[st[i].second]; j < i; j++) {
        point l;
        assert(intersect(st[j].first, st[j + 1].first, l) == 1);
        ans.push_back(l);
    }
    break;
}
return ans;
}
```

CHT.cpp  
Description: CHT for minimum, k is decreasing, works for equal slopes

30baab, 34 lines

```
struct line {
    int k, b;
    int eval(int x) {
        return k * x + b;
    }
};
struct part {
    line a;
    ld x;
};
ld intersection(line a, line b) {
    return (ld) (a.b - b.b) / (b.k - a.k);
}
struct ConvexHullMin {
    vector<part> st;
    void add(line a) {
        if (!st.empty() && st.back().a.k == a.k) {
            if (st.back().a.b > a.b) st.pop_back();
            else return;
        }
        while (st.size() > 1 && intersection(st[st.size() - 2].a, a) <=
            st[st.size() - 2].x) st.pop_back();
        if (!st.empty()) st.back().x = intersection(st.back().a, a);
        st.push_back({a, INF});
    }
    int get_val(int x) {
        int l = -1, r = (int)st.size() - 1;
        while (r - l > 1) {
            int m = (l + r) / 2;
            if (st[m].x < x) l = m;
            else r = m;
        }
        return st[r].a.eval(x);
    }
};
```

DynamicCHT.cpp  
Description: Dynamic CHT for maximum

8a0777, 30 lines

```
struct Line {
    mutable ll k, m, p;
    bool operator<(const Line& o) const {
        return Q ? p < o.p : k < o.k;
    }
};
struct LineContainer : multiset<Line> {
    const ll inf = LLONG_MAX;
    ll div(ll a, ll b) {
        return a / b - ((a ^ b) < 0 && a % b);
    }
    bool isect(iterator x, iterator y) {
        if (y == end()) { x->p = inf; return false; }
        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());
    Q = 1; auto l = *lower_bound({0,0,x}); Q = 0;
    return l.k * x + l.m;
}
};
```

MinPlusConv.cpp  
Description: Min-Plusconv, A is convex down  
Time:  $\mathcal{O}(n \log n)$

5d63d9, 28 lines

```
// Assumptions: 'a' is convex, 'opt' has size 'n+m-1'
// 'opt[k]' will be equal to 'arg min(a[k-i] + b[i])'
template<typename T>
void convex_arbitrary_min_plus_conv(T *a, int n, T *b, int m, int *opt) {
    auto rec = [&](auto &&self, int lx, int rx, int ly, int ry) -> void {
        if (lx > rx) return;
        int mx = (lx + rx) >> 1;
        opt[mx] = ly;
        for (int i = ly; i <= ry; ++i)
            if (mx >= i && (mx - opt[mx] >= n || a[mx - opt[mx]] + b[opt[
                mx]] > a[mx - i] + b[i]))
                opt[mx] = i;
        self(self, lx, mx - 1, ly, opt[mx]);
        self(self, mx + 1, rx, opt[mx], ry);
    };
    rec(rec, 0, n + m - 2, 0, m - 1);
}
```

```
// Assumptions: 'a' is convex down
template<typename T>
std::vector<T> convex_arbitrary_min_plus_conv(const std::vector<T> &a,
    const std::vector<T> &b) {
    int n = a.size(), m = b.size();
    int *opt = (int*) malloc(sizeof(int) * (n + m - 1));
    convex_arbitrary_min_plus_conv(a.data(), n, b.data(), m, opt);
    std::vector<T> ans(n + m - 1);
    for (int i = 0; i < n + m - 1; ++i) ans[i] = a[i - opt[i]] + b[opt[i]
        ];
    free(opt);
    return ans;
}
```

Kinetic.cpp  
Description: kinetic segment tree  
Time:  $\mathcal{O}(hz)$

49b24c, 127 lines

```
//vnutrennii functions — poluintervali, vneshnie — otrezki. ishet min
priamuy
struct line {
    ll k,b,temp;
    ll eval() const {
        return k * temp + b;
    }
    ll melting_point(const line& other) const {
        ll val1 = eval();
        ll val2 = other.eval();
        assert(val1 <= val2);
        if (other.k >= k) {
            return INF;
        }
        ll delta_val = val2 - val1;
        ll delta_k = k - other.k;
        assert(delta_val >= 0 && delta_k > 0);
        return (delta_val + delta_k - 1) / delta_k;
    }
};
struct kinetic_segtree {
    struct node {
        ll lazy_b = 0, lazy_temp = 0, melt = INF;
        line best;
        node(line best = line()) : best(best) {}
    };
    int n;
    vector<node> tree;
```

```
void update(int v) {
    if (make_pair(tree[v << 1].best.eval(), tree[v << 1].best.k) <
        make_pair(tree[v << 1 | 1].best.eval(), tree[v << 1 | 1].
            best.k)) {
        tree[v].best = tree[v << 1].best;
        tree[v].melt = tree[v].best.melting_point(tree[v << 1 | 1].
            best);
    } else {
        tree[v].best = tree[v << 1 | 1].best;
        tree[v].melt = tree[v].best.melting_point(tree[v << 1].best);
    }

    tree[v].melt = min({tree[v].melt, tree[v << 1].melt, tree[v << 1
        | 1].melt});
    assert(tree[v].melt > 0);
}

void apply(int v, int vl, int vr, ll delta_b, ll delta_temp) {
    tree[v].lazy_b += delta_b;
    tree[v].lazy_temp += delta_temp;

    tree[v].best.b += delta_b;
    tree[v].best.temp += delta_temp;

    tree[v].melt -= delta_temp;
    if (tree[v].melt <= 0) {
        push(v, vl, vr);
        update(v);
    }
}

void push(int v, int vl, int vr) {
    int vm = (vl + vr) / 2;
    apply(v << 1, vl, vm, tree[v].lazy_b, tree[v].lazy_temp);
    apply(v << 1 | 1, vm, vr, tree[v].lazy_b, tree[v].lazy_temp);
    tree[v].lazy_b = 0;
    tree[v].lazy_temp = 0;
}

void build(int v, int vl, int vr, const vector<line> &lines) {
    if (vr - vl == 1) {
        tree[v] = node(lines[vl]);
        return;
    }
    int vm = (vl + vr) / 2;
    build(v << 1, vl, vm, lines);
    build(v << 1 | 1, vm, vr, lines);
    update(v);
}

void add(int v, int vl, int vr, int l, int r, ll delta_b, ll
    delta_temp) {
    if (r <= vl || vr <= l) {
        return;
    }
    if (l <= vl && vr <= r) {
        apply(v, vl, vr, delta_b, delta_temp);
        return;
    }
    push(v, vl, vr);
    int vm = (vl + vr) / 2;
    add(v << 1, vl, vm, l, r, delta_b, delta_temp);
    add(v << 1 | 1, vm, vr, l, r, delta_b, delta_temp);
    update(v);
}

void change_line(int v, int vl, int vr, int pos, const line &new_line
    ) {
    if (vr - vl == 1) {
        tree[v].best = new_line;
        return;
    }
    push(v, vl, vr);
    int vm = (vl + vr) / 2;
    if (pos < vm) {
        change_line(v << 1, vl, vm, pos, new_line);
    } else {
        change_line(v << 1 | 1, vm, vr, pos, new_line);
    }
    update(v);
}

ll query(int v, int vl, int vr, int l, int r) {
    if (r <= vl || vr <= l) {
        return INF;
    }
```

```
    }
    if (l <= vl && vr <= r) {
        return tree[v].best.eval();
    }
    push(v, vl, vr);
    int vm = (vl + vr) / 2;
    return min(query(v << 1, vl, vm, l, r), query(v << 1 | 1, vm, vr,
        l, r));
}

kinetic_segtree(const vector<line> &lines) : n(lines.size()), tree(4
    * n) {
    build(1, 0, n, lines);
}

kinetic_segtree(int n) : n(n), tree(4 * n) {
    vector<line> lines(n, {0, INF, 0});
    build(1, 0, n, lines);
}

void add(int l, int r, ll delta_b, ll delta_temp) {
    assert(delta_temp >= 0);
    add(1, 0, n, l, r + 1, delta_b, delta_temp);
}

void change_line(int pos, const line &new_line) {
    assert(0 <= pos && pos < n);
    change_line(1, 0, n, pos, new_line);
}

ll query(int l, int r) {
    return query(1, 0, n, l, r + 1);
}
}
```

GoldenSearch.cpp

Description: Golden Search

31d45b, 14 lines

```
double gss(double a, double b, double (*f)(double)) {
    double r = (sqrt(5) - 1) / 2, eps = 1e-7;
    double x1 = b - r * (b - a), x2 = a + r * (b - a);
    double f1 = f(x1), f2 = f(x2);
    while (b - a > eps)
        if (f1 < f2) { // change to > to find maximum
            b = x2; x2 = x1; f2 = f1;
            x1 = b - r * (b - a); f1 = f(x1);
        } else {
            a = x1; x1 = x2; f1 = f2;
            x2 = a + r * (b - a); f2 = f(x2);
        }
    return a;
}
```

3dBasic.cpp

Description: Basic 3d geom things

467773, 79 lines

```
const int inf = int(1e9) + int(1e5);
const ll inf1 = ll(2e18) + ll(1e10);
const ld eps = 1e-9;
bool ze(ld x) {
    return fabs1(x) < eps;
}

struct pt {
    ld x, y, z;
    pt operator+(const pt &p) const {
        return pt{x + p.x, y + p.y, z + p.z};
    }
    pt operator-(const pt &p) const {
        return pt{x - p.x, y - p.y, z - p.z};
    }
    ld operator*(const pt &p) const {
        return x * p.x + y * p.y + z * p.z;
    }
    pt operator*(ld a) const {
        return pt{x * a, y * a, z * a};
    }
    pt operator%(const pt &p) const {
        return pt{y * p.z - z * p.y, z * p.x - x * p.z, x * p.y - y * p.x};
    }
    ld abs() const {
        return sqrt1(*this * *this);
    }
    ld abs2() const {
        return *this * *this;
    }
}
```

```
    }
    pt norm() const {
        ld d = abs();
        return pt{x / d, y / d, z / d};
    }
};
// BEGINCODE
struct Plane {
    pt v;
    ld c;
    Plane(pt a, pt b, pt c) {
        v = ((b - a) % (c - a)).norm();
        this->c = a * v;
    }
    ld dist(pt p) {
        return p * v - c;
    }
};

pt projection(pt p, pt a, pt b) {
    pt v = b - a;
    if (ze(v.abs2())) {
        // stub : bad line
        return a;
    }
    return a + v * ((p - a) * v) / (v * v));
}

pair<pt, pt> planesIntersection(Plane a, Plane b) {
    pt dir = a.v % b.v;
    if (ze(dir.abs2())) {
        // stub : parallel planes
        return {pt{1e18, 1e18, 1e18}, pt{1e18, 1e18, 1e18}};
    }
    ld s = a.v * b.v;
    pt v3 = b.v - a.v * s;
    pt h = a.v * a.c + v3 * ((b.c - a.c * s) / (v3 * v3));
    return {h, h + dir};
}

pair<pt, pt> commonPerpendicular(pt a, pt b, pt c, pt d) {
    pt v = (b - a) % (d - c);
    ld S = v.abs();
    if (ze(S)) {
        // stub : parallel lines
        return {pt{1e18, 1e18, 1e18}, pt{1e18, 1e18, 1e18}};
    }
    v = v.norm();
    pt sh = v * (v * c - v * a);
    pt a2 = a + sh;
    ld s1 = ((c - a2) % (d - a2)) * v;
    pt p = a + (b - a) * (s1 / S);
    return {p, p + sh};
}
```

NDHull.cpp

Description: Hull in arbitrary number of dimensions

Time:  $O(N * Dim * Hull)$

cf8067, 77 lines

```
const int DIM = 4;
typedef array<ll, DIM> pt;
pt operator-(const pt &a, const pt &b) {
    pt res;
    forn(i, DIM) res[i] = a[i] - b[i];
    return res;
}

typedef array<pt, DIM - 1> Edge;
typedef array<pt, DIM> Face;
vector<Face> faces;
ll det(pt *a) {
    int p[DIM];
    iota(p, p + DIM, 0);
    ll res = 0;
    do {
        ll x = 1;
        forn(i, DIM) {
            forn(j, i) if (p[j] > p[i]) x *= -1;
            x *= a[i][p[i]];
        }
        res += x;
    } while (next_permutation(p, p + DIM));
    return res;
}
```

```

    for (int k = 0; k < j; ++k) {
        if (len(o - ps[k]) > r * EPS) {
            o = circumcenter(ps[i], ps[j], ps[k]);
            r = len(o - ps[i]);
        }
    }
}

}

}

}

return {o, r};
}

```

**Time:**  $\mathcal{O}(n \log n)$  8c39c9, 17 lines

<b>Description:</b> dealing with planar graphs	b0fad2, 170 lines
--	-------------------

```
// g.size() == pts.size() + 1, so that there is one new outer face
// all previously outer faces will have g[v].size() == 0
vector<vector<int>> build_faces_graph(const vector<vec> &pts, const
    vector<vector<int>> &faces) {
    vector<int> realface(faces.size());
    iota(all(realface), 0);
    vector<vec> qq;
    vector<int> ind;
    for (int i = 0; i < (int) faces.size(); ++i) {
        vector<vec> face;
        for (int j : faces[i]) {
            face.pb(pts[j]);
        }
        ld a = get_area(face);
        if (a < 0) {
```

```
pair<vec, ld> mindisk(vector<vec> ps) {
    shuffle(all(ps), rnd);
    vec o = ps[0];
    ld r = 0, EPS = 1 + 1e-8;
    for (int i = 0; i < ps.size(); ++i) {
        if (len(o - ps[i]) > r * EPS) {
            o = ps[i], r = 0;
            for (int j = 0; j < i; ++j) {
                if (len(o - ps[j]) > r * EPS) {
                    o = (ps[i] + ps[j]) / 2;
                    r = len(o - ps[i]);
                }
            }
        }
    }
    return pair<vec, ld>(o, r);
}
```

```

        // if only one outter face, then realface[i] = faces.size();
        otherwise following code
        vec v = *min_element(all(face));
        v.x -= 10 * EPS;
        qq.pbc(v);
        ind.pbc(i);
        // realface[i] = find_face(pts, faces, v);
        // assert(realface[i] != i);
    }
}

if (l) { // slow, but easy to write
    for (int i = 0; i < (int)qq.size(); ++i) {
        int j = find_face(pts, faces, qq[i]);
        assert(j != ind[i]);
        realface[ind[i]] = j;
    }
} else {
    vector<int> res = point_location(pts, faces, qq);
    for (int i = 0; i < (int)qq.size(); ++i) {
        int j = res[i];
        assert(j != ind[i]);
        realface[ind[i]] = j;
    }
}

map<pair<int, int>, int> edge2face;
for (int i = 0; i < (int) faces.size(); ++i) {
    for (int j = 0; j < (int) faces[i].size(); ++j) {
        int a = faces[i][j];
        int b = faces[i][(j + 1) % faces[i].size()];
        edge2face[{a, b}] = realface[i];
    }
}

vector<vector<int>> g(faces.size() + 1);
for (auto [pp, c] : edge2face) {
    g[c].pbc(edge2face[{pp.second, pp.first}]);
}

for (auto &el : g) {
    sort(all(el));
    el.erase(unique(all(el)), el.end());
}

return g;
}

vector<vector<int>> get_faces(const vector<vec> &pts, const vector<vector<int>> &g) {
    int n = g.size();
    vector<vector<pair<int, int>>> g2(n);
    int cur_edge = 0;
    for (int i = 0; i < n; ++i) {
        for (int j : g[i]) {
            if (i < j) {
                g2[j].pbc({i, cur_edge});
                g2[i].pbc({j, cur_edge ^ 1});
                cur_edge += 2;
            }
        }
    }

    vector<int> ind(cur_edge), used(cur_edge);
    for (int i = 0; i < n; ++i) {
        sort(all(g2[i]), [&](auto a, auto b) {
            auto va = pts[a.first] - pts[i];
            auto vb = pts[b.first] - pts[i];
            return mp(half(va), (ld)0) < mp(half(vb), va % vb);
        });
        for (int j = 0; j < (int) g2[i].size(); ++j) {
            ind[g2[i][j].second] = j;
        }
    }

    vector<vector<int>> faces;
    for (int i = 0; i < n; ++i) {
        for (int ei = 0; ei < (int)g[i].size(); ++ei) {
            if (used[g2[i][ei].second]) continue;
            vector<int> face;
            int v = i;
            int e = g2[v][ei].second;
            while (!used[e]) {
                used[e] = 1;
                face.pbc(v);
                int u = g2[v][ind[e]].first;

```

```

        int newe = g2[u][(ind[e ^ 1] - 1 + g2[u].size()) % g2[u].size()];
        v = u;
        e = newe;
    }
    faces.push_back(face);
}

return faces;
}

pair<vector<vec>, vector<vector<int>>> build_graph(vector<pair<vec, vec>> segs) {
    vector<vec> p;
    vector<vector<int>> g;
    map<pair<ll, ll>, int> id;
    auto getid = [&](vec v) {
        auto r = mp(ll(round(v.x * 1'000'000'000 + EPS * sign(v.x))), ll(round(v.y * 1'000'000'000 + EPS * sign(v.y))));
        if (!id.count(r)) {
            g.pbc({});
            int i = id.size();
            id[r] = i;
            p.pbc(v);
            return i;
        }
        return id[r];
    };

    for (int i = 0; i < (int)segs.size(); ++i) {
        vector<int> cur = {getid(segs[i].first), getid(segs[i].second)};
        for (int j = 0; j < (int)segs.size(); ++j) {
            if (i != j) {
                if (intersects(segs[i].first, segs[i].second, segs[j].first, segs[j].second)) {
                    vec res;
                    if (intersect(getln(segs[i].first, segs[i].second), getln(segs[j].first, segs[j].second), res)) {
                        cur.pbc(getid(res));
                    } else {
                        if (isOnSegment(segs[i].first, segs[i].second, segs[j].first))
                            cur.pbc(getid(segs[j].first));
                        if (isOnSegment(segs[i].first, segs[i].second, segs[j].second))
                            cur.pbc(getid(segs[j].second));
                    }
                }
            }
        }
        sort(all(cur), [&](int i, int j) { return p[i] < p[j]; });
        cur.erase(unique(all(cur)), cur.end());
        for (int j = 1; j < (int)cur.size(); ++j) {
            g[cur[j]].pbc(cur[j - 1]);
            g[cur[j - 1]].pbc(cur[j]);
        }
    }

    for (auto &el : g) {
        sort(all(el));
        el.erase(unique(all(el)), el.end());
    }

    return {p, g};
}

```

## PointLocation.cpp

Description: Point location xd

573c9d, 276 lines

const vec arb = {(int)1e9 + 228, (int)1e9 + 228}; // ne soupadaet s drygimi tochkami

```

bool ge(const ll& a, const ll& b) { return a >= b; }
bool le(const ll& a, const ll& b) { return a <= b; }
bool eq(const ll& a, const ll& b) { return a == b; }
bool gt(const ll& a, const ll& b) { return a > b; }
bool lt(const ll& a, const ll& b) { return a < b; }

```

```

ll vec::dot(const vec &a) const {
    return *this * a;
}

ll vec::cross(const vec &a) const {

```

```

    return *this % a;
}

ll vec::dot(const vec &a, const vec &b) const {
    return (a - *this) * (b - *this);
}

ll vec::cross(const vec &a, const vec &b) const {
    return (a - *this) % (b - *this);
}

struct Edge {
    vec l, r;
    auto operator<=>(const Edge &) const = default;
};

bool edge_cmp(const Edge& edge1, const Edge& edge2) {
    const vec a = edge1.l, b = edge1.r;
    const vec c = edge2.l, d = edge2.r;
    int val = sign(a.cross(b, c)) + sign(a.cross(b, d));
    if (val != 0)
        return val > 0;
    val = sign(c.cross(d, a)) + sign(c.cross(d, b));
    return val < 0;
}

enum EventType { DEL = 2, ADD = 3, GET = 1, VERT = 0 };

struct Event {
    EventType type;
    int pos;
    bool operator<(const Event& event) const { return type < event.type; }
};

vector<Edge> sweepline(vector<Edge> planar, vector<vec> queries) {
    using vec_type = decltype(vec::x);

    // collect all x-coordinates
    auto s = set<vec_type, std::function<bool(const vec_type&, const vec_type&)>>(lt);
    for (vec p : queries)
        s.insert(p.x);
    for (auto e : planar) {
        s.insert(e.l.x);
        s.insert(e.r.x);
    }

    // map all x-coordinates to ids
    int cid = 0;
    auto id = map<vec_type, int, std::function<bool(const vec_type&, const vec_type&)>>(lt);
    for (auto x : s)
        id[x] = cid++;

    // create events
    auto t = set<Edge, decltype(*edge_cmp)>(edge_cmp);
    auto vert_cmp = [] (const pair<vec_type, int>& l, const pair<vec_type, int>& r) {
        if (!eq(l.first, r.first))
            return lt(l.first, r.first);
        return l.second < r.second;
    };
    auto vert = set<pair<vec_type, int>, decltype(vert_cmp)>(vert_cmp);
    vector<vector<Event>> events(cid);
    for (int i = 0; i < (int)queries.size(); ++i) {
        int x = id[queries[i].x];
        events[x].push_back(Event{GET, i});
    }

    for (int i = 0; i < (int)planar.size(); ++i) {
        int lx = id[planar[i].l.x], rx = id[planar[i].r.x];
        if (lx > rx) {
            swap(lx, rx);
            swap(planar[i].l, planar[i].r);
        }
        if (lx == rx) {

```

```

        events[lx].push_back(Event{VERT, i});
    } else {
        events[lx].push_back(Event{ADD, i});
        events[rx].push_back(Event{DEL, i});
    }
}

// perform sweep line algorithm
vector<Edge> ans(queries.size(), {arb, arb});
for (int x = 0; x < cid; x++) {
    sort(events[x].begin(), events[x].end());
    vert.clear();
    for (Event event : events[x]) {
        if (event.type == DEL) {
            t.erase(planar[event.pos]);
        }
        if (event.type == VERT) {
            vert.insert(make_pair(
                min(planar[event.pos].l.y, planar[event.pos].r.y),
                event.pos));
        }
        if (event.type == ADD) {
            t.insert(planar[event.pos]);
        }
        if (event.type == GET) {
            auto jt = vert.upper_bound(
                make_pair(queries[event.pos].y, planar.size()));
            if (jt != vert.begin()) {
                --jt;
                int i = jt->second;
                if (ge(max(planar[i].l.y, planar[i].r.y),
                    queries[event.pos].y)) {
                    ans[event.pos] = planar[i];
                    continue;
                }
            }
            Edge e;
            e.l = e.r = queries[event.pos];
            auto it = t.upper_bound(e);
            if (it != t.begin()) {
                ans[event.pos] = *(--it);
            }
        }
    }
}

for (Event event : events[x]) {
    if (event.type != GET)
        continue;
    if (ans[event.pos].l != arb &&
        eq(ans[event.pos].l.x, ans[event.pos].r.x))
        continue;

    Edge e;
    e.l = e.r = queries[event.pos];
    auto it = t.upper_bound(e);
    if (it == t.begin())
        e = {arb, arb};
    else
        e = *(--it);
    if (ans[event.pos].l == arb) {
        ans[event.pos] = e;
        continue;
    }
    if (e.l == arb)
        continue;
    if (e == ans[event.pos])
        continue;
    if (id[ans[event.pos].r.x] == x) {
        if (id[e.l.x] == x) {
            if (gt(e.l.y, ans[event.pos].r.y))
                ans[event.pos] = e;
        }
    } else {
        ans[event.pos] = e;
    }
}
}
return ans;
}

```

```

struct DCEL {
    struct Edge {
        vec origin;
        int nxt;
        int twin;
        int face;
    };
    vector<Edge> body;
};

// outer face is -1, returns (l,i) is point is strictly inside face i,
// and (0,1) if point lies on the edge i
vector<pair<int, int>> point_location(DCEL planar, vector<vec> queries)
{
    vector<pair<int, int>> ans(queries.size());
    vector<Edge> planar2;
    map<Edge, int> pos;
    map<Edge, int> added_on;
    int n = planar.body.size();
    for (int i = 0; i < n; i++) {
        if (planar.body[i].face > planar.body[planar.body[i].twin].face)
            continue;
        Edge e;
        e.l = planar.body[i].origin;
        e.r = planar.body[planar.body[i].twin].origin;
        if (e.r.x < e.l.x) swap(e.l, e.r);
        added_on[e] = i;
        pos[e] =
            lt(planar.body[i].origin.x, planar.body[planar.body[i].twin].
                origin.x)
                ? planar.body[i].face
                : planar.body[planar.body[i].twin].face;
        planar2.push_back(e);
    }
    auto res = sweepline(planar2, queries);
    for (int i = 0; i < (int)queries.size(); i++) {
        if (res[i].l == arb) {
            ans[i] = make_pair(1, -1);
            continue;
        }
        vec p = queries[i];
        vec l = res[i].l, r = res[i].r;
        if (eq(p.cross(l, r), 0) && le(p.dot(l, r), 0)) {
            ans[i] = make_pair(0, added_on[res[i]]);
            continue;
        }
        ans[i] = make_pair(1, pos[res[i]]);
    }
    return ans;
}

DCEL buildDCEL(const vector<vec> &pts, const vector<vector<int>> &g) {
    int n = g.size();
    vector<vector<pair<int, int>>> g2(n);
    int cur_edge = 0;
    for (int i = 0; i < n; ++i) {
        for (int j : g[i]) {
            if (i < j) {
                g2[j].pbc({i, cur_edge});
                g2[i].pbc({j, cur_edge ^ 1});
                cur_edge += 2;
            }
        }
    }
    vector<int> ind(cur_edge), used(cur_edge);
    for (int i = 0; i < n; ++i) {
        sort(all(g2[i]), [&](auto a, auto b) {
            auto va = pts[a.first] - pts[i];
            auto vb = pts[b.first] - pts[i];
            return mp(half(va), 0LL) < mp(half(vb), va % vb);
        });
        for (int j = 0; j < (int) g2[i].size(); ++j) {
            ind[g2[i][j].second] = j;
        }
    }
    using Edge = DCEL::Edge;
    vector<Edge> edges(cur_edge);
    for (int i = 0; i < cur_edge; ++i) {
        edges[i].twin = i ^ 1;
    }
}

```

```

int cur_face = 0;
for (int i = 0; i < n; ++i) {
    for (int ei = 0; ei < (int)g[i].size(); ++ei) {
        if (used[g2[i][ei].second]) continue;
        vector<vec> face;
        vector<int> inds;
        int v = i;
        int e = g2[v][ei].second;
        while (!used[e]) {
            edges[e].origin = pts[v];
            edges[e].face = cur_face;
            inds.pbc(e);
            used[e] = 1;
            face.pbc(pts[v]);
            int u = g2[v][ind[e]].first;
            int newe = g2[u][(ind[e ^ 1] - 1 + g2[u].size()) % g2[u].
                size()].second;
            edges[e].nxt = newe;
            v = u;
            e = newe;
        }
        if (sign(get_area(face)) <= 0) {
            for (int i : inds) {
                edges[i].face = -1;
            }
        } else {
            ++cur_face;
        }
    }
}
return {edges};
}

```

## Svg.cpp

Description: geometry visualizer

e9032a, 36 lines

```

struct SVG {
    FILE *out;
    ld sc = 50;

    void open() {
        out = fopen("image.svg", "w");
        fprintf(out, "<svg xmlns='http://www.w3.org/2000/svg' viewBox
            ='-1000 -1000 2000 2000'>\n");
    }

    void line(vec a, vec b) {
        a = a * sc, b = b * sc;
        fprintf(out, "<line x1='%Lf' y1='%Lf' x2='%Lf' y2='%Lf' stroke='
            black'/>\n", a.x, -a.y, b.x, -b.y);
    }

    void circle(vec a, ld r = -1, string col = "red") {
        r = (r == -1 ? 10 : sc * r);
        a = a * sc;
        fprintf(out, "<circle cx='%Lf' cy='%Lf' r='%Lf' fill='%s'/>\n", a
            .x, -a.y, r, col.c_str());
    }

    void text(vec a, string s) {
        a = a * sc;
        fprintf(out, "<text x='%Lf' y='%Lf' font-size='10px'>%s</text>\n"
            , a.x, -a.y, s.c_str());
    }

    void close() {
        fprintf(out, "</svg>\n");
        fclose(out);
        out = 0;
    }

    ~SVG() {
        if (out)
            close();
    }
} svg;

```

Delauney.cpp

**Description:** Fast Delaunay triangulation. Each circumcircle contains none of the input points. There must be no duplicate points. If all points are on a line, no triangles will be returned. Should work for doubles as well, though there may be precision issues in 'circ'. Returns triangles in order {t[0][0], t[0][1], t[0][2], t[1][0], ...}, all counter-clockwise.

**Time:**  $\mathcal{O}(n \log n)$

```
9e818a, 97 lines

typedef vec P;
typedef struct Quad* Q;
// using lll = _int128_t; // (can be ll if coords are < 2e4)
P arb(LLONG_MAX,LLONG_MAX); // not equal to any other point
#define rep(i,a,b) for (int i=a;i<b;++i)

lll vec::cross(const vec &b) const {
    return *this % b;
}

lll vec::cross(const vec &b, const vec &c) const {
    return (b - *this) % (c - *this);
}

struct Quad {
    Q rot, o; P p = arb; bool mark;
    P& F() { return r()->p; }
    Q& r() { return rot->rot; }
    Q prev() { return rot->o->rot; }
    Q next() { return r()->prev(); }
} *H;

bool circ(P p, P a, P b, P c) { // is p in the circumcircle?
    lll p2 = len2(p), A = len2(a)-p2,
        B = len2(b)-p2, C = len2(c)-p2;
    return p.cross(a,b)*C + p.cross(b,c)*A + p.cross(c,a)*B > 0;
}

Q makeEdge(P orig, P dest) {
    Q r = H ? H : new Quad(new Quad(new Quad{new Quad{0}}));
    H = r->o; r->r()->r() = r;
    rep(i,0,4) r = r->rot, r->p = arb, r->o = i & 1 ? r : r->r();
    r->p = orig; r->F() = dest;
    return r;
}

void splice(Q a, Q b) {
    swap(a->o->rot->o, b->o->rot->o); swap(a->o, b->o);
}

Q connect(Q a, Q b) {
    Q q = makeEdge(a->F(), b->p);
    splice(q, a->next());
    splice(q->r(), b);
    return q;
}

pair<Q,Q> rec(const vector<P>& s) {
    if (s.size() <= 3) {
        Q a = makeEdge(s[0], s[1]), b = makeEdge(s[1], s.back());
        if (s.size() == 2) return { a, a->r() };
        splice(a->r(), b);
        auto side = s[0].cross(s[1], s[2]);
        Q c = side ? connect(b, a) : 0;
        return {side < 0 ? c->r() : a, side < 0 ? c : b->r() };
    }

#define H(e) e->F(), e->p
#define valid(e) (e->F().cross(H(base)) > 0)
    Q A, B, ra, rb;
    int half = s.size() / 2;
    tie(ra, A) = rec({all(s) - half});
    tie(B, rb) = rec({s.size() - half + all(s)});
    while ((B->p.cross(H(A)) < 0 && (A = A->next())) ||
        (A->p.cross(H(B)) > 0 && (B = B->r()->o)));
    Q base = connect(B->r(), A);
    if (A->p == ra->p) ra = base->r();
    if (B->p == rb->p) rb = base;

#define DEL(e, init, dir) Q e = init->dir; if (valid(e)) \
    while (circ(e->dir->F(), H(base), e->F())) { \
        Q t = e->dir; \
        splice(e, e->prev()); \
        splice(e->r(), e->r()->prev()); \
    }
```

```
e->o = H; H = e; e = t; \
}
}
for (;;) {
    DEL(LC, base->r(), o); DEL(RC, base, prev());
    if (!valid(LC) && !valid(RC)) break;
    if (!valid(LC) || (valid(RC) && circ(H(RC), H(LC))))
        base = connect(RC, base->r());
    else
        base = connect(base->r(), LC->r());
}
return { ra, rb };
}

vector<P> triangulate(vector<P> pts) {
    sort(all(pts)); assert(unique(all(pts)) == pts.end());
    if (pts.size() < 2) return {};
    Q e = rec(pts).first;
    vector<Q> q = {e};
    int qi = 0;
    while (e->o->F().cross(e->F(), e->p) < 0) e = e->o;
#define ADD { Q c = e; do { c->mark = 1; pts.push_back(c->p); \
    q.push_back(c->r()); c = c->next(); } while (c != e); }
    ADD; pts.clear();
    while (qi < q.size()) if (!(e = q[qi++]->mark) ADD;
    return pts;
}
```

SegmentInPolygon.cpp

**Description:** length of longest segment inside polygon

```
<bits/stdc++.h>
509dfa, 40 lines

#define ll long long
using namespace std;
const int N=210;
int n,w;
double s,res,ans;
double isp(point x1,point y1,point x2,point y2){
    return 1.0*( (x2-x1)*(y2-y1))/((y2-x2)*(y1-x1));
}
double calc(point a,point b){
    v.clear(),w=s=res=0;
    for(int i=1;i<=n;i++){
        int x=sgn((b-a)|(p[i-1]-a)),y=sgn((b-a)|(p[i]-a));
        if(x==y) continue;
        v.push_back({isp(a,b,p[i-1],p[i]),(x<y?-1):(x&y?2:1)});
    }
    sort(v.begin(),v.end());
    for(int i=0;i<(int)v.size();i++){
        if(w) s+=v[i].first-v[i-1].first;
        else res=max(res,s),s=0;
        w+=v[i].second;
    }
    return max(res,s)*((b-a).len());
}

signed main(){
    scanf("%d",&n);
    for(int i=1;i<=n;i++) scanf("%d%d",&p[i].x,&p[i].y);
    p[0]=p[n];
    for(int i=1;i<=n;i++)
        for(int j=i+1;j<=n;j++) ans=max(ans,calc(p[i],p[j]));
    printf("%.9lf\n",ans);
    return 0;
}
```

Math (6)

BerlekampMassey.cpp

**Description:** Find the shortest linear-feedback shift register

**Time:**  $\mathcal{O}(n^2)$

```
vector<int> berlekamp(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 + (ll) 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) * powmod(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-2} returns {2, 1}

// kth element of p/q as fps
int getkfps(vector<int> p, vector<int> q, ll k) {
    assert(q[0] != 0);
    while (k) {
        auto f = q;
        for (int i = 1; i < (int)f.size(); i += 2) {
            f[i] = sub(0, f[i]);
        }
        auto p2 = conv(p, f);
        auto q2 = conv(q, f);
        p.clear(), q.clear();
        for (int i = k % 2; i < (int)p2.size(); i += 2) {
            p.pbc(p2[i]);
        }
        for (int i = 0; i < (int)q2.size(); i += 2) {
            q.pbc(q2[i]);
        }
        k >= 1;
        return mul(p[0], inv(q[0]));
    }

    // vals - initials values of recurrence, c - result of belekamp on vals
    int getk(const vector<int> &vals, vector<int> c, ll k) {
        int d = c.size();
        c.insert(c.begin(), MOD-1);
        while (c.back() == 0) {
            c.pop_back();
        }
        for (auto &el : c) {
            el = sub(0, el);
        }
        vector<int> p(d);
        copy(vals.begin(), vals.begin() + d, p.begin());
        p = conv(p, c);
        p.resize(d);
        return getkfps(p, c, k);
    }
}
```

```
vector<int> getmod(vector<int> a, vector<int> md) {
    for (int i = a.size() - 1; i + 1 >= md.size(); --i) {
```



```
int v = mul(a[i], inv(md.back()));
for (int j = 0; j < md.size(); ++j) {
    a[i - md.size() + 1 + j] = sub(a[i - md.size() + 1 + j], mul(
        md[j], v));
}
a.pop_back();
}
return a;
}
```

GoncharFedor.cpp

Description: Calculating number of points  $x, y \geq 0, Ax + By \leq C$   
Time:  $\mathcal{O}(\log(C))$  0ef10e, 11 lines

```
ll solve_triangle(ll A, ll B, ll C) { // x,y >=0, Ax+By<=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 solve_triangle(B - k * A, A, C - A * (k * p + d + 1)) +
        (p + 1) * (d + 1) + k * p * (p + 1) / 2;
}
```

CRT.cpp

Description: CRT for arbitrary modulus 28309e, 25 lines

```
int extgcd(int a, int b, int &x, int &y) { // define int ll
    if (a == 0) {
        x = 0, y = 1;
        return b;
    }
    int x1, y1;
    int g = extgcd(b % a, a, x1, y1);
    x = y1 - x1 * (b / a);
    y = x1;
    return g;
}
int lcm(int a, int b) { return a / __gcd(a, b) * b; }
int crt(int mod1, int mod2, int rem1, int rem2) {
    int r = (rem2 - (rem1 % mod2) + mod2) % mod2;
    int x, y;
    int g = extgcd(mod1, mod2, x, y);
    if (r % g) return -1;
    x %= mod2;
    if (x < 0) x += mod2;
    int ans = (x * (r / g)) % mod2;
    ans = ans * mod1 + rem1;
    assert(ans % mod1 == rem1);
    assert(ans % mod2 == rem2);
    return ans % lcm(mod1, mod2);
}
```

Fastmod.cpp

Description: Fast multiplication by modulo(in [0;2b)) 38ea39, 7 lines

```
struct FastMod {
    ull b, m;
    FastMod(ull b) : b(b), m(-1ULL / b) {}
    ull reduce(ull a) { // a % b + (0 or b)
        return a - (ull)((__uint128_t(m) * a) >> 64) * b;
    }
};
```

ModularSqrt.cpp

Description: Calculating sqrt modulo smth  
Time:  $\mathcal{O}(\log^2)$  19a793, 23 lines

```
ll sqrt(ll a, ll p) {
    a %= p;
    if (a < 0) a += p;
    if (a == 0) return 0;
    assert(modpow(a, (p - 1) / 2, p) == 1); // e lse no so lution
    if (p % 4 == 3) return modpow(a, (p + 1) / 4, p);
    // a^(n+3)/8 or 2^(n+3)/8 * 2^(n=1)/4 works i f p % 8 == 5
    ll s = p - 1, n = 2;
    int r = 0, m;
```

```
while (s % 2 == 0) ++r, s /= 2;
while (modpow(n, (p - 1) / 2, p) != p - 1) ++n;
ll x = modpow(a, (s + 1) / 2, p);
ll b = modpow(a, s, p), g = modpow(n, s, p);
for (; r = m) {
    ll t = b;
    for (m = 0; m < r && t != 1; ++m) t = t * t % p;
    if (m == 0) return x;
    ll gs = modpow(g, 1LL << (r - m - 1), p);
    g = gs * gs % p;
    x = x * gs % p;
    b = b * g % p;
}
}
```

DiscreteLog.cpp

Description: Discrete log  
Time:  $\mathcal{O}(\sqrt{n})$  1cc247, 9 lines

```
ll modLog(ll a, ll b, ll m) {
    ll n = (ll)sqrt(m) + 1, e = 1, f = 1, j = 1;
    unordered_map<ll, ll> A;
    while (j <= n && (e = f = e * a % m) != b % m) A[e * b % m] = j++;
    if (e == b % m) return j;
    if (__gcd(m, e) == __gcd(m, b))
        for (int i = 2; i < n + 2; ++i)
            if (A.count(e = e * f % m)) return n * i - A[e];
    return -1;
}
```

PrimalityTest.cpp

Description: Checking primality of p  
Time:  $\mathcal{O}(\log(C))$  ad2714, 32 lines

```
const int iters = 8; // can change
bool isprime(ll p) {
    if (p == 1 || p == 4)
        return 0;
    if (p == 2 || p == 3)
        return 1;
    for (int it = 0; it < iters; ++it) {
        ll a = rnd() % (p - 2) + 2;
        ll nw = p - 1;
        while (nw % 2 == 0)
            nw /= 2;
        ll x = binpow(a, nw, p); // int128
        if (x == 1)
            continue;
        ll last = x;
        nw *= 2;
        while (nw <= p - 1) {
            x = (__int128_t)x * x % p;
            if (x == 1) {
                if (last != p - 1) {
                    return 0;
                }
                break;
            }
            last = x;
            nw *= 2;
        }
        if (x != 1)
            return 0;
    }
    return 1;
}
```

XorConvolution.cpp

Description: Calculating xor-convolution of 2 vectors modulo smth  
Time:  $\mathcal{O}(n \log(n))$  454afd, 23 lines

```
void fwht(vector<int> &a) {
    int n = a.size();
    for (int l = 1; l < n; l <= 1) {
        for (int i = 0; i < n; i += 2 * l) {
            for (int j = 0; j < l; ++j) {
                int u = a[i + j], v = a[i + j + l];
                a[i + j] = add(u, v), a[i + j + l] = sub(u, v);
            }
        }
    }
}
```

```

    }
} // https://judge.yosupo.jp/problem/bitwise_xor_convolution
vector<int> xorconvo(vector<int> a, vector<int> b) {
    int n = 1;
    while (n < max(a.size(), b.size()))
        n *= 2;
    a.resize(n), b.resize(n);
    fwht(a), fwht(b);
    int in = inv(n);
    for (int i = 0; i < n; ++i)
        a[i] = mul(a[i], mul(b[i], in));
    fwht(a);
    return a;
}
```

Factorization.cpp

Description: Factorizing a number real quick  
Time:  $\mathcal{O}(n^{\frac{1}{4}})$  f0d7c6, 51 lines

```
ll gcd(ll a, ll b) {
    while (b)
        a %= b, swap(a, b);
    return a;
}

ll f(ll a, ll n) { return ((__int128_t)a * a % n + 1) % n; }

vector<ll> factorize(ll n) {
    if (n <= 1e6) { // can add primality check for speed?
        vector<ll> res;
        for (ll i = 2; i * i <= n; ++i) {
            while (n % i == 0) {
                res.pb(i);
                n /= i;
            }
        }
        if (n != 1)
            res.pb(n);
        return res;
    }
    ll x = rnd() % (n - 1) + 1;
    ll y = x;
    ll tries = 10 * sqrt(sqrt(n));
    const int C = 60;
    for (ll i = 0; i < tries; i += C) {
        ll xs = x;
        ll ys = y;
        ll m = 1;
        for (int k = 0; k < C; ++k) {
            x = f(x, n);
            y = f(f(y, n), n);
            m = (__int128_t)m * abs(x - y) % n;
        }
        if (gcd(n, m) == 1)
            continue;
        x = xs, y = ys;
        for (int k = 0; k < C; ++k) {
            x = f(x, n);
            y = f(f(y, n), n);
            ll res = gcd(n, abs(x - y));
            if (res != 1 && res != n) {
                vector<ll> v1 = factorize(res), v2 = factorize(n / res);
                for (auto j : v2)
                    v1.pb(j);
                return v1;
            }
        }
    }
    return {n};
}
```

PrimeCount.cpp

Description: counting number of primes below N  
Time:  $\mathcal{O}(N^{\frac{2}{3}})$  a8507c, 53 lines

```
ll prime_pi(const ll N) {
    if (N <= 1) return 0;
    if (N == 2) return 1;
```

```

const int v = sqrt(N);
int s = (v + 1) / 2;
vector<int> smalls(s);
for (int i = 1; i < s; i++) smalls[i] = i;
vector<int> roughs(s);
for (int i = 0; i < s; i++) roughs[i] = 2 * i + 1;
vector<ll> larges(s);
for (int i = 0; i < s; i++) larges[i] = (N / (2 * i + 1) - 1) / 2;
vector<bool> skip(v + 1);
const auto divide = [](ll n, ll d) -> int { return n / d; };
const auto half = [](int n) -> int { return (n - 1) >> 1; };
int pc = 0;
for (int p = 3; p <= v; p += 2)
    if (!skip[p]) {
        int q = p * p;
        if ((ll)q * q > N) break;
        skip[p] = true;
        for (int i = q; i <= v; i += 2 * p) skip[i] = true;
        int ns = 0;
        for (int k = 0; k < s; k++) {
            int i = roughs[k];
            if (skip[i]) continue;
            ll d = (ll)i * p;
            larges[ns] = larges[k] -
                (d <= v ? larges[smalls[d >> 1] - pc]
                 : smalls[half(divide(N, d))]) +
                pc;
            roughs[ns++] = i;
        }
        s = ns;
        for (int i = half(v), j = ((v / p) - 1) | 1; j >= p; j -= 2) {
            int c = smalls[j >> 1] - pc;
            for (int e = (j * p) >> 1; i >= e; i--) smalls[i] -= c;
        }
        pc++;
    }
larges[0] += (ll)(s + 2 * (pc - 1)) * (s - 1) / 2;
for (int k = 1; k < s; k++) larges[0] -= larges[k];
for (int l = 1; l < s; l++) {
    ll q = roughs[l];
    ll M = N / q;
    int e = smalls[half(M / q)] - pc;
    if (e < l + 1) break;
    ll t = 0;
    for (int k = l + 1; k <= e; k++)
        t += smalls[half(divide(M, roughs[k]))];
    larges[0] += t - (ll)(e - 1) * (pc + l - 1);
}
return larges[0] + 1;
}

```

## NTT.cpp

Description: Fast FFT!

Time:  $\mathcal{O}(n \log(n))$

e7ea21, 272 lines

// Don't use Ofast, potential slow down by 2x!  
 // Write mint first!

```

int maxn, maxk;
vector<int> rvi;
vector<mint> wpws;

```

```

void build_fft(int _maxk) {
    maxk = _maxk;
    maxn = (1 << maxk);
    rvi.resize(maxn);
    rvi[0] = 0;
    for (int i = 1; i < maxn; i += 1) {
        rvi[i] = (rvi[i >> 1] >> 1);
        if (i & 1) {
            rvi[i] |= (1 << (maxk - 1));
        }
    }
    mint w = mint(3).pow((mod - 1) / maxn);
    mint pw = 1;
    wpws.resize(maxn);
    rep(i, maxn) {
        wpws[rvi[i]] = pw;
        pw *= w;
    }
}

```

```

    }
}

void fft(vector<mint>& a, int k) {
    int n = (1 << k);
    for (int ln = n / 2; ln >= 1; ln /= 2) {
        int ln2 = ln * 2;
        for (int i = 0; i < n; i += ln2) {
            auto w = wpws[i / ln];
            for (int j = i; j < i + ln; j += 1) {
                auto u = a[j];
                auto v = a[j + ln] * w;
                a[j] = u + v;
                a[j + ln] = u - v;
            }
        }
    }
    rep(i, n) {
        int mrv = (rvi[i] >> (maxk - k));
        if (mrv < i) {
            swap(a[i], a[mrv]);
        }
    }
}

void inv_fft(vector<mint>& a, int k) {
    fft(a, k);
    int n = (1 << k);
    mint invn = mint(n).inv();
    rep(i, n) {
        a[i] *= invn;
    }
    reverse(a.begin() + 1, a.end());
}

vector<mint> mul(vector<mint> a, vector<mint> b) {
    if (a.empty() or b.empty()) {
        return {};
    }
    auto ca = a;
    auto cb = b;
    int lna = len(a);
    int lnb = len(b);
    int k = __lg(lna + lnb - 1);
    if (lna + lnb - 1 == (1 << k) + 1) {
        auto c = mul(vector<mint>(a.begin(), a.end() - 1), b);
        c.resize(lna + lnb - 1);
        rep(j, lnb) {
            c[lna - 1 + j] += a[lna - 1] * b[j];
        }
        return c;
    }
    if (lna + lnb - 1 > (1 << k)) {
        k += 1;
    }
    int n = (1 << k);
    a.resize(n);
    b.resize(n);
    fft(a, k);
    fft(b, k);
    rep(i, n) {
        a[i] *= b[i];
    }
    inv_fft(a, k);
    a.resize(lna + lnb - 1);
    return a;
}

vector<mint> operator+(vector<mint> a, vector<mint> b) {
    a.resize(max(a.size(), b.size()));
    for (int i = 0; i < (int)b.size(); ++i) {
        a[i] += b[i];
    }
    return a;
}

vector<mint> operator-(vector<mint> a, vector<mint> b) {
    a.resize(max(a.size(), b.size()));
    for (int i = 0; i < (int)b.size(); ++i) {

```

```

        a[i] -= b[i];
    }
    return a;
}

vector<mint> inv(const vector<mint>& a, int need) {
    vector<mint> b = { a[0].inv() };
    while ((int)b.size() < need) {
        vector<mint> a1 = a;
        int m = b.size();
        a1.resize(min((int)a1.size(), 2 * m));
        b = mul(b, vector<mint>(2) - mul(a1, b));
        b.resize(2 * m);
    }
    b.resize(need);
    return b;
}

vector<mint> mul2(vector<mint> a, vector<mint> b) {
    int lna = len(a);
    int lnb = len(b);
    int k = 0;
    while ((1 << k) < lna) {
        ++k;
    }
    int n = (1 << k);
    a.resize(n);
    reverse(all(b));
    b.resize(n);
    fft(a, k);
    fft(b, k);
    rep(i, n) a[i] *= b[i];
    inv_fft(a, k);
    vector<mint> c(lna - lnb + 1);
    rep(i, len(c)) {
        c[i] = a[lnb - 1 + i];
    }
    return c;
}

vector<mint> multipoint(vector<mint> a, vector<mint> x) {
    int n = x.size();
    int m = len(a);
    vector<vector<mint>> tree(2 * n);
    for (int i = 0; i < n; ++i) {
        tree[i + n] = { 1, 0 - x[i] };
    }
    for (int i = n - 1; i; --i) {
        tree[i] = mul(tree[2 * i], tree[2 * i + 1]);
    }
    auto tinv = inv(tree[1], m);
    a.resize(n + m - 1);
    auto c = mul2(a, tinv);
    tree[1] = c;
    for (int i = 1; i < n; i += 1) {
        auto x = tree[i + 1];
        auto y = tree[i + i + 1];
        tree[i + i] = mul2(tree[i], y);
        tree[i + i + 1] = mul2(tree[i], x);
    }
    vector<mint> res(n);
    for (int i = 0; i < n; ++i) {
        res[i] = tree[i + n][0];
    }
    return res;
}

vector<mint> div(vector<mint> a, vector<mint> b) {
    int n = a.size() - 1;
    int m = b.size() - 1;
    if (n < m) return { 0 };
    reverse(all(a));
    reverse(all(b));
    a.resize(n - m + 1);
    b.resize(n - m + 1);
    vector<mint> c = inv(b, b.size());
    vector<mint> q = mul(a, c);
    q.resize(n - m + 1);
    reverse(all(q));
}

```

```
    return q;
}

vector<mint> mod_poly(vector<mint> a, vector<mint> b) {
    auto res = a - mul(b, div(a, b));
    res.resize(len(b) - 1);
    return res;
}

vector<mint> deriv(vector<mint> a) {
    for (int i = 1; i < (int)a.size(); ++i) {
        a[i - 1] = a[i] * i;
    }
    a.back() = 0;
    if (a.size() > 1) {
        a.pop_back();
    }
    return a;
}

vector<mint> integ(vector<mint> a) {
    a.push_back(0);
    for (int i = (int)a.size() - 1; i; --i) {
        a[i] = a[i - 1] * mint(i).inv();
    }
    a[0] = 0;
    return a;
}

vector<mint> log(vector<mint> a, int n) {
    auto res = integ(mul(deriv(a), inv(a, n)));
    res.resize(n);
    return res;
}

vector<mint> exp(vector<mint> a, int need) {
    vector<mint> b = { 1 };
    while ((int)b.size() < need) {
        vector<mint> a1 = a;
        int m = b.size();
        a1.resize(min((int)a1.size(), 2 * m));
        a1[0] += 1;
        b = mul(b, a1 - log(b, 2 * m));
        b.resize(2 * m);
    }
    b.resize(need);
    return b;
}

vector<mint> gf_projection(vector<mint> f) { // ensure that f[0]=0
    int lnf = len(f);
    int n = 1;
    while (n < len(f)) n *= 2;
    vector<mint> g(n);
    g[n - lnf] = 1;
    f.resize(n);
    rep(i, n) f[i] = 0 - f[i];
    int m = 1;
    while (n > 1) {
        f.resize(4 * n * m);
        f[2 * n * m] = 1;
        g.resize(4 * n * m);
        fft(f);
        fft(g);
        auto q = f;
        rotate(q.begin(), q.begin() + 2 * n * m, q.end());
        vector<mint> gf(4 * n * m), ff(4 * n * m);
        rep(i, 4 * n * m) {
            gf[i] = g[i] * q[i];
            ff[i] = f[i] * q[i];
        }
        inv_fft(gf);
        inv_fft(ff);
        ff[0] -= 1;
        f.assign(2 * n * m, 0);
        g.assign(2 * n * m, 0);
        rep(i, n / 2) rep(j, 2 * m) {
            f[j * n + i] = ff[j * (2 * n) + 2 * i];
            g[j * n + i] = gf[j * (2 * n) + 2 * i + 1];
        }
    }
}
```

```
    }
    n /= 2; m *= 2;
}
vector<mint> res(m);
rep(i, m) {
    res[i] = g[2 * i];
}
reverse(all(res));
res.resize(lnf);
return res;
}

FFT.cpp
Description: Calculating product of two polynomials
Time:  $O(n \log(n))$ 
3adba5, 67 lines

const ld PI = acos(-1);
using cd = complex<ld>;
const int MAXLOG = 19, N = (1 << MAXLOG), MAXN = (1 << MAXLOG) + 228;
int rev[MAXN];
cd w[MAXN];
bool fftInit = false;

void initFFT() {
    for (int i = 0; i < N; ++i) {
        w[i] = cd(cos(2 * PI * i / N), sin(2 * PI * i / N));
    }
    rev[0] = 0;
    for (int i = 1; i < N; ++i) {
        rev[i] = (rev[i >> 1] >> 1) ^ ((i & 1) << (MAXLOG - 1));
    }
}

void FFT(int n, vector<cd>& a, bool rv = false) {
    if (!fftInit) {
        initFFT();
        fftInit = 1;
    }
    int LOG = ceil(log2(n));
    for (int i = 0; i < n; ++i) {
        if (i < (rev[i] >> (MAXLOG - LOG))) {
            swap(a[i], a[(rev[i] >> (MAXLOG - LOG))]);
        }
    }
    for (int lvl = 0; lvl < LOG; ++lvl) {
        int len = 1 << lvl;
        for (int st = 0; st < n; st += len * 2) {
            for (int i = 0; i < len; ++i) {
                cd x = a[st + i], y = a[st + len + i] * w[i << (MAXLOG - 1 - lvl)];
                a[st + i] = x + y;
                a[st + i + len] = x - y;
            }
        }
    }
    if (rv) {
        reverse(a.begin() + 1, a.end());
        for (auto& i : a) i /= n;
    }
}

vector<ll> mul(vector<ll> a, vector<ll> b) {
    int xd = max(a.size(), b.size()) * 2;
    int cur = 1;
    while (cur < xd) {
        cur *= 2;
    }
    a.resize(cur);
    b.resize(cur);
    vector<cd> ma(cur), mb(cur);
    for (int i = 0; i < cur; ++i) {
        ma[i] += a[i];
        mb[i] += b[i];
    }
    FFT(cur, ma);
    FFT(cur, mb);
    for (int i = 0; i < cur; ++i) ma[i] *= mb[i];
    FFT(cur, ma, true);
    vector<ll> ans(cur);
```

```
    for (int i = 0; i < cur; ++i) {
        ans[i] = (ll)(ma[i].real() + 0.5);
    }
    return ans;
}

AndConvolution.cpp
Description: Calculating and-convolution modulo smth
Time:  $O(n \log(n))$ 
5dedf4, 24 lines

void conv(vector<int> &a, bool x) {
    int n = a.size();
    for (int j = 0; (1 << j) < n; ++j) {
        for (int i = 0; i < n; ++i) {
            if (!(i & (1 << j))) {
                if (x)
                    a[i] = add(a[i], a[i | (1 << j)]);
                else
                    a[i] = sub(a[i], a[i | (1 << j)]);
            }
        }
    }
}

vector<int> andcon(vector<int> a, vector<int> b) {
    int n = 1;
    while (n < max(a.size(), b.size()))
        n *= 2;
    a.resize(n), b.resize(n);
    conv(a, 1), conv(b, 1);
    for (int i = 0; i < n; ++i)
        a[i] = mul(a[i], b[i]);
    conv(a, 0);
    return a;
}

SubsetConvolution.cpp
Description: subset convolution
Time:  $O(2^n * n^2)$  (500 ms n = 20 with pragms)
a47122, 39 lines

void transform(int n, int N, vector<int>& b, const vector<int>& a,
               const vector<int>& pc, bool rev) {
    if (!rev) {
        b.assign(N << n, 0);
        for (int i = 0; i < (int)a.size(); ++i) b[pc[i] + i * N] = a[i];
    }
    for (int w = 1; w <= (1 << n); ++w) {
        for (int d = 0; !(w & (1 << d)); ++d) {
            int W = N * (w - (1 << d)), dd = N << d;
            for (int i = N * (w - (2 << d)); i < W; ++i) {
                if (!rev) b[i + dd] = add(b[i + dd], b[i]);
                else b[i + dd] = sub(b[i + dd], b[i]);
            }
        }
    }
}

vector<int> SubsetConvolution(const vector<int>& a, const vector<int>& b) {
    {
        int n = 0;
        while ((1 << n) < max(a.size(), b.size())) n++;
        int N = n + 1;
        vector<int> pc(1 << n, 0);
        for (int i = 1; i < (1 << n); ++i) pc[i] = pc[i - (i & -i)] + 1;
        vector<int> bufA, bufB;
        transform(n, N, bufA, a, pc, false);
        transform(n, N, bufB, b, pc, false);
        for (int i = 0; i < (1 << n); ++i) {
            int I = i * N;
            vector<int> Q(N);
            for (int ja = 0; ja <= pc[i]; ++ja) {
                for (int jb = pc[i] - ja, x = min(n - ja, pc[i]); jb <= x; ++jb) {
                    Q[ja + jb] = add(Q[ja + jb], mul(bufA[ja + I], bufB[jb + I]));
                }
            }
            copy(Q.begin(), Q.end(), bufA.begin() + I);
        }
        transform(n, N, bufA, a, pc, true);
        vector<int> res(1 << n);
```

<pre>for(int i = 0; i&lt;(1&lt;n); ++i) res[i] = bufA[pc[i] + i*N]; return res; }</pre>	
<b>Simplex.cpp</b> <b>Description:</b> Simplex <b>Time:</b> exponential XD(ok for 200-300 variables/bounds)	4dda3e, 99 lines

```
/* solver for linear programs of the form
maximize c^T x, subject to A x <= b, x >= 0
outputs target function for optimal solution and
the solution by reference
if unbounded above : returns inf, if infeasible : returns -inf
create Simplex.Steep<ld> LP(A, b, c), then call LP. Solve(x)
*/
template <typename DOUBLE>
struct Simplex_Steep {
    using VD = vector<DOUBLE>;
    using VVD = vector<VD>;
    using VI = vector<int>;
    DOUBLE EPS = 1e-12;
    int m, n;
    VI B, N;
    VVD D;
    Simplex_Steep(const VVD &A, const VD &b, const VD &c)
        : m(b.size()), n(c.size()), B(m), N(n + 1), D(m + 2, VD(n + 2)) {
        for (int i = 0; i < m; i++)
            for (int j = 0; j < n; j++) D[i][j] = A[i][j];
        for (int i = 0; i < m; i++) {
            B[i] = n + i;
            D[i][n] = -1;
            D[i][n + 1] = b[i];
        }
        for (int j = 0; j < n; j++) {
            N[j] = j;
            D[m][j] = -c[j];
        }
        N[n] = -1;
        D[m + 1][n] = 1;
    }
    void Pivot(int r, int s) {
        for (int i = 0; i < m + 2; i++)
            if (i != r)
                for (int j = 0; j < n + 2; j++)
                    if (j != s) D[i][j] -= D[r][j] * D[i][s] / D[r][s];
        for (int j = 0; j < n + 2; j++)
            if (j != s) D[r][j] /= D[r][s];
        for (int i = 0; i < m + 2; i++)
            if (i != r) D[i][s] /= -D[r][s];
        D[r][s] = 1.0 / D[r][s];
        swap(B[r], N[s]);
    }
    bool Simplex(int phase) {
        int x = m + (int) (phase == 1);
        while (true) {
            int s = -1;
            DOUBLE c_val = -1;
            for (int j = 0; j <= n; j++) {
                if (phase == 2 && N[j] == -1) continue;
                DOUBLE norm_sq = 0;
                for (int k = 0; k <= m; k++) norm_sq += D[k][j] * D[k][j];
                norm_sq = max(norm_sq, EPS);
                DOUBLE c_val_j = D[x][j] / sqrtl(norm_sq);
                if (s == -1 || c_val_j < c_val ||
                    (c_val == c_val_j && N[j] < N[s])) {
                    s = j;
                    c_val = c_val_j;
                }
            }
            if (D[x][s] >= -EPS) return true;
            int r = -1;
            for (int i = 0; i < m; i++) {
                if (D[i][s] <= EPS) continue;
                if (r == -1 || D[i][n + 1] / D[i][s] < D[r][n + 1] / D[r][s] ||
                    (D[i][n + 1] / D[i][s] == D[r][n + 1] / D[r][s] &&
                     B[i] < B[r]))
                    r = i;
            }
        }
    }
};
```

```

    }
    if (r == -1) return false;
    Pivot(r, s);
}
}
DOUBLE Solve(VD &x) {
    int r = 0;
    for (int i = 1; i < m; i++)
        if (D[i][n + 1] < D[r][n + 1]) r = i;
    if (D[r][n + 1] <= -EPS) {
        Pivot(r, n);
        if (!Simplex(1) || D[m + 1][n + 1] < -EPS)
            return numeric_limits<DOUBLE>::infinity();
        for (int i = 0; i < m; i++)
            if (B[i] == -1) {
                int s = -1;
                for (int j = 0; j <= n; j++)
                    if (s == -1 || D[i][j] < D[i][s] ||
                        (D[i][j] == D[i][s] && N[j] < N[s]))
                        s = j;
                Pivot(i, s);
            }
        }
    if (!Simplex(2)) return numeric_limits<DOUBLE>::infinity();
    x = VD(n);
    for (int i = 0; i < m; i++)
        if (B[i] < n) x[B[i]] = D[i][n + 1];
    return D[m][n + 1];
}
};
```

<b>DeterminantLd.cpp</b> <b>Description:</b> Determinant in ld	1a6123, 18 lines
---	------------------

```
double det(vector<vector<double>>& a) {
    int n = sz(a);
    double res = 1;
    for (int i = 0; i < n; ++i) {
        int b = i;
        for (int j = i + 1; j < n; ++j)
            if (fabs(a[j][i]) > fabs(a[b][i])) b = j;
        if (i != b) swap(a[i], a[b]), res *= -1;
        res *= a[i][i];
        if (res == 0) return 0;
        for (int j = i + 1; j < n; ++j) {
            double v = a[j][i] / a[i][i];
            if (v != 0)
                for (int k = i + 1; k < n; ++k) a[j][k] -= v * a[i][k];
        }
    }
    return res;
}
```

<b>DeterminantInt.cpp</b> <b>Description:</b> Determinant in ints	c2ab5a, 19 lines
--	------------------

```
const ll mod = 12345;
ll det(vector<vector<ll>>& a) {
    int n = sz(a);
    ll ans = 1;
    for (int i = 0; i < n; ++i) {
        for (int j = i + 1; j < n; ++j) {
            while (a[j][i] != 0) { // gcd step
                ll t = a[i][i] / a[j][i];
                if (t)
                    for (int k = i; k < n; ++k)
                        a[i][k] = (a[i][k] - a[j][k] * t) % mod;
                swap(a[i], a[j]);
                ans *= -1;
            }
        }
        ans = ans * a[i][i] % mod;
        if (!ans) return 0;
    }
    return (ans + mod) % mod;
}
```

<b>TridiagSLE.cpp</b> <b>Description:</b> Tridiagonal SLE solver(didnt test yet) <b>Time:</b> $O(N)$	532e1d, 16 lines
--	------------------

```
vector<ld> trisle(vector<ld> a, vector<ld> b, vector<ld> c) {
    // a[i] * x[i - 1] + c[i] * x[i] + b[i] * x[i + 1] == f[i]
    int n = a.size(); // a[0] == 0, b[n - 1] == 0
    alpha[1] = -(ld)b[0] / c[0];
    beta[1] = (ld)f[0] / c[0];
    for (int i = 1; i < n - 1; i++) {
        ld zn = (ld)a[i] * alpha[i] + c[i];
        alpha[i + 1] = -(ld)b[i] / zn;
        beta[i + 1] = (f[i] - (ld)a[i] * beta[i]) / zn;
    }
    x[n - 1] = (f[n - 1] - a[n - 1] * beta[n - 1]) /
        (a[n - 1] * alpha[n - 1] + c[n - 1]);
    for (int i = n - 2; i >= 0; i--)
        x[i] = alpha[i + 1] * x[i + 1] + beta[i + 1];
    return x;
}
```

<b>SolveLinear.cpp</b> <b>Description:</b> Solving linear systems <b>Time:</b> $O(n^3)$	44c9ab, 35 lines
---	------------------

```
typedef vector<double> vd;
const double eps = 1e-12; // rep(i,a,b) = for(int i=a;i<b;++i)
int solveLinear(vector<vd>& A, vd& b, vd& x) {
    int n = sz(A), m = sz(x), rank = 0, br, bc;
    if (n) assert(sz(A[0]) == m);
    vi col(m);
    iota(all(col), 0);
    rep(i, 0, n) {
        double v, bv = 0;
        rep(r, i, n) rep(c, i, m) if ((v = fabs(A[r][c])) > bv) br = r, bc = c, bv = v;
        if (bv <= eps) {
            rep(j, i, n) if (fabs(b[j]) > eps) return -1;
            break;
        }
        swap(A[i], A[br]);
        swap(b[i], b[br]);
        swap(col[i], col[bc]);
        rep(j, 0, n) swap(A[j][i], A[j][bc]);
        bv = 1 / A[i][i];
        rep(j, i + 1, n) {
            double fac = A[j][i] * bv;
            b[j] -= fac * b[i];
            rep(k, i + 1, m) A[j][k] -= fac * A[i][k];
        }
        rank++;
    }
    x.assign(m, 0);
    for (int i = rank; i--;) {
        b[i] /= A[i][i];
        x[col[i]] = b[i];
        rep(j, 0, i) b[j] -= A[j][i] * b[i];
    }
    return rank; // (multiple solutions if rank < m)
}
```

<b>PolyInter.cpp</b> <b>Description:</b> Interpolating polynomials <b>Time:</b> $O(n^2)$	4edad5, 14 lines
--	------------------

```
typedef vector<double> vd;
vd interpolate(vd x, vd y, int n) {
    vd res(n), temp(n);
    for (int k = 0; k < n - 1; ++k)
        for (int i = k + 1; i < n; ++i) y[i] = (y[i] - y[k]) / (x[i] - x[k]);
    double last = 0;
    temp[0] = 1;
    for (int k = 0; k < n; ++k)
        for (int i = 0; i < n; ++i) {
            res[i] += y[k] * temp[i];
            swap(last, temp[i]);
            temp[i] -= last * x[k];
        }
}
```

```
    }
    return res;
}

CharPoly.cpp
Description: det(a - xI)
666c0e, 37 lines

vector<int> CharacteristicPolynomial(vector<vector<int>>> a) {
    int n = a.size();
    for(int j = 0; j < n - 2; j++) {
        for(int i = j + 1; i < n; i++) {
            if(a[i][j] != 0) {
                swap(a[j + 1], a[i]);
                for(int k = 0; k < n; k++) swap(a[k][j + 1], a[k][i]);
                break;
            }
        }
        if(a[j + 1][j] != 0) {
            int flex = inv(a[j + 1][j]);
            for(int i = j + 2; i < n; i++) {
                if(a[i][j] == 0) continue;
                int coe = mul(flex, a[i][j]);
                for(int l = j; l < n; l++) a[i][l] = sub(a[i][l], mul(coe, a[j + 1][l]));
                for(int k = 0; k < n; k++) a[k][j + 1] = add(a[k][j + 1], mul(coe, a[k][i]));
            }
        }
        vector<vector<int>>> p(n + 1);
        p[0] = {1};
        for(int i = 1; i <= n; i++) {
            p[i].resize(i + 1);
            for(int j = 0; j < i; j++) {
                p[i][j + 1] = sub(p[i][j + 1], p[i - 1][j]);
                p[i][j] = add(p[i][j], mul(p[i - 1][j], a[i - 1][i - 1]));
            }
            int x = 1;
            for(int m = 1; m < i; m++) {
                x = mul(x, sub(0, a[i - m][i - m - 1]));
                int coe = mul(x, a[i - m - 1][i - 1]);
                for(int j = 0; j < i - m; j++) p[i][j] = add(p[i][j], mul(coe, p[i - m - 1][j]));
            }
        }
        return p[n];
    }
}
```

```
FloorSum.cpp
Description: finds  $\sum_{x=0}^{n-1} [(kx + b) / m]$ . Require  $k \geq 0, b \geq 0, m \geq 0, n \geq 0$ 
932f6c, 41 lines

template<typename T>
T floor_sum(T k, T b, T m, T n) {
    if (k == 0) {
        return (b / m) * n;
    }
    if (k >= m || b >= m) {
        return n * (n - 1) / 2 * (k / m) + n * (b / m) + floor_sum(k % m, b % m, m, n);
    }
    T ymax = (k * (n - 1) + b) / m;
    return n * ymax - floor_sum(m, m + k - b - 1, k, ymax);
}
```

```
WaysCount.cpp
Description: Find number of right-up paths from (0, 0) to (x, y), not touching lines y=x+l and y=x+r Time: O((x + y)/(r - l))
57f1b0, 12 lines

mint flex(ll x, ll y, ll l, ll r) {
    if (l >= 0 or r <= 0) {
        return 0;
    }
    ll n = x + y;
    mint res = 0;
    for (ll k = -(n / (r - l)); k <= n / (r - l); k += 1) {
        res += cnk(n, x + k * (r - l));
        res -= cnk(n, y - r + k * (r - l));
    }
    return res;
}
```

### 6.1 Fun things

$$ClassesCount = \frac{1}{|G|} \sum_{\pi \in G} I(\pi)$$
$$ClassesCount = \frac{1}{|G|} \sum_{\pi \in G} k^{C(\pi)}$$

Stirling 2kind - count of partitions of n objects into k nonempty sets:

$$S(n, k) = S(n - 1, k - 1) + kS(n - 1, k)$$
$$S(n, k) = \sum_{j=0}^{n-1} \binom{n-1}{j} S(j, k - 1)$$
$$S(n, k) = \frac{1}{k!} \sum_{j=0}^k (-1)^{k+j} \binom{k}{j} j^n$$
$$n! \approx \sqrt{2n\pi} (\frac{n}{e})^n$$

$\binom{n}{k} \equiv \prod_i \binom{n_i}{k_i}$ ,  $n_i, k_i$  - digits of  $n, k$  in p-adic system

$$\int_a^b f(x) dx \approx \frac{b-a}{6} (f(a) + 4f(\frac{a+b}{2}) + f(b))$$
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, O(loglog)$$
$$G(n) = n \oplus (n \ggg 1)$$
$$g(n) = \sum_{d|n} f(d) \Rightarrow f(n) = \sum_{d|n} g(d) \mu(\frac{n}{d})$$
$$\sum_{d|n} \mu(d) = [n = 1], \mu(1) = 1, \mu(p) = -1, \mu(p^k) = 0$$
$$\sin(a \pm b) = \sin a \cos b \pm \sin b \cos a$$
$$\cos(a \pm b) = \cos a \cos b \mp \sin a \sin b$$
$$\operatorname{tg}(a \pm b) = \frac{\operatorname{tg} a \pm \operatorname{tg} b}{1 \mp \operatorname{tg} a \operatorname{tg} b}$$
$$\operatorname{ctg}(a \pm b) = \frac{\operatorname{ctg} a \operatorname{ctg} b \mp 1}{\operatorname{ctg} b \pm \operatorname{ctg} a}$$
$$\sin \frac{a}{2} = \pm \sqrt{\frac{1 - \cos a}{2}}$$
$$\cos \frac{a}{2} = \pm \sqrt{\frac{1 + \cos a}{2}}$$
$$\operatorname{tg} \frac{a}{2} = \frac{\sin a}{1 - \cos a} = \frac{1 - \cos a}{\sin a}$$
$$\sin \alpha = \frac{2 \operatorname{tg} \frac{\alpha}{2}}{1 + \operatorname{tg}^2 \frac{\alpha}{2}}$$
$$\cos \alpha = \frac{1 - \operatorname{tg}^2 \frac{\alpha}{2}}{1 + \operatorname{tg}^2 \frac{\alpha}{2}}$$
$$\operatorname{tg} \alpha = \frac{2 \operatorname{tg} \frac{\alpha}{2}}{1 - \operatorname{tg}^2 \frac{\alpha}{2}}$$
$$\sin^2 \alpha = \frac{1 - \cos 2\alpha}{2}$$
$$\sin^3 \alpha = \frac{3 \sin \alpha - \sin 3\alpha}{4}$$
$$\cos^2 \alpha = \frac{1 + \cos 2\alpha}{2}$$
$$\cos^3 \alpha = \frac{3 \cos \alpha + \cos 3\alpha}{4}$$
$$\sin a \sin b = \frac{\cos(a - b) - \cos(a + b)}{2}$$
$$\sin a \cos b = \frac{\sin(a - b) + \sin(a + b)}{2}$$
$$\cos a \cos b = \frac{\cos(a - b) + \cos(a + b)}{2}$$

1 jan 2000 - saturday, 1 jan 1900 - monday, 14 apr 1961 - friday

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

Fibonacci: 45:1134903170. 46:1836311903(max int), 91: 4660046610375530309

Highly composite numbers:  
≤ 1000 :  $d(840) = 32, \leq 10^4 : d(9240) = 64, \leq 10^5 : d(83160) = 128, \leq 10^6 : d(720720) = 240, \leq 10^7 : d(8648640) = 448, \leq 10^8 : d(91891800) = 768, \leq 10^9 : d(931170240) = 1344, \leq 10^{11} : d(97772875200) = 4032, \leq 10^{15} : d(866421317361600) = 26880, \leq 10^{18} : d(897612484786617600) = 103680$

BEST Theorem:  
 $ec(G) = \#SpanningTrees(G) \cdot \prod_{v \in V} (deg(v) - 1)!$

Erdos: Graph exists  
 $\Leftrightarrow d_1 \geq .. \geq d_n, \forall k \sum_{i=1}^k d_i \leq k(k - 1) + \sum_{i=k+1}^n min(d_i, k)$

Pick:  $Area = Interior + \frac{Bounds}{2} - 1$   
Euler:  $V - E + F = 1 + C$   
Kirchhoff: put degree on diagonal, −1 for each edge, cut out first row + column, calc det - result is #SpanningTrees  
Tree Hash: for vertex  $v$  calculate  $\prod_i (c_i + d_{h_i})$ , where  $c_i$  - hash of ith child,  $d_{h_i}$  - random number associated to depth of current child  
Get position of Gray Code g: int n = 0; for (; g; g>>= 1) n xor= g; return n;

Table of Basic Integrals (7)

Basic Forms

∫ x^n dx = 1/(n+1) x^{n+1}, n ≠ -1 (7.1)

∫ 1/x dx = ln |x| (7.2)

∫ u dv = uv - ∫ v du (7.3)

∫ 1/(ax+b) dx = 1/a ln |ax+b| (7.4)

Integrals of Rational Functions

∫ 1/(x+a)^2 dx = -1/(x+a) (7.5)

∫ (x+a)^n dx = (x+a)^{n+1}/(n+1), n ≠ -1 (7.6)

∫ x(x+a)^n dx = (x+a)^{n+1}((n+1)x-a)/((n+1)(n+2)) (7.7)

∫ 1/(1+x^2) dx = tan^-1 x (7.8)

∫ 1/(a^2+x^2) dx = 1/a tan^-1 x/a (7.9)

∫ x/(a^2+x^2) dx = 1/2 ln |a^2+x^2| (7.10)

∫ x^2/(a^2+x^2) dx = x - a tan^-1 x/a (7.11)

∫ x^3/(a^2+x^2) dx = 1/2 x^2 - 1/2 a^2 ln |a^2+x^2| (7.12)

∫ 1/(ax^2+bx+c) dx = 2/(√(4ac-b^2)) tan^-1 (2ax+b)/√(4ac-b^2) (7.13)

∫ 1/((x+a)(x+b)) dx = 1/(b-a) ln (a+x)/(b+x), a ≠ b (7.14)

∫ x/((x+a)^2) dx = a/(a+x) + ln |a+x| (7.15)

∫ x/(ax^2+bx+c) dx = 1/2a ln |ax^2+bx+c| - b/(a√(4ac-b^2)) tan^-1 (2ax+b)/√(4ac-b^2) (7.16)

Integrals with Roots

∫ √(x-a) dx = 2/3 (x-a)^{3/2} (7.17)

∫ 1/√(x±a) dx = 2√(x±a) (7.18)

∫ 1/√(a-x) dx = -2√(a-x) (7.19)

∫ x√(x-a) dx = { 2a/3 (x-a)^{3/2} + 2/5 (x-a)^{5/2}, or 2/3 x(x-a)^{3/2} - 4/15 (x-a)^{5/2}, or 2/15 (2a+3x)(x-a)^{3/2} } (7.20)

∫ √(ax+b) dx = (2b/3a + 2x/3) √(ax+b) (7.21)

∫ (ax+b)^{3/2} dx = 2/5a (ax+b)^{5/2} (7.22)

∫ x/√(x±a) dx = 2/3 (x ∓ 2a) √(x±a) (7.23)

∫ √(x/(a-x)) dx = -√(x(a-x)) - a tan^-1 √(x(a-x))/(x-a) (7.24)

∫ √(x/(a+x)) dx = √(x(a+x)) - a ln [√x + √(x+a)] (7.25)

∫ x√(ax+b) dx = 2/15a^2 (-2b^2+abx+3a^2x^2)√(ax+b) (7.26)

∫ √(x(ax+b)) dx = 1/(4a^{3/2}) [(2ax+b)√(ax(ax+b)) - b^2 ln |a√x + √(a(ax+b))|] (7.27)

∫ √(x^3(ax+b)) dx = [b/12a - b^2/(8a^2x) + x/3] √(x^3(ax+b)) + b^3/(8a^{5/2}) ln |a√x + √(a(ax+b))| (7.28)

∫ √(x^2 ± a^2) dx = 1/2 x√(x^2 ± a^2) ± 1/2 a^2 ln |x + √(x^2 ± a^2)| (7.29)

∫ √(a^2 - x^2) dx = 1/2 x√(a^2 - x^2) + 1/2 a^2 tan^-1 x/√(a^2 - x^2) (7.30)

∫ x√(x^2 ± a^2) dx = 1/3 (x^2 ± a^2)^{3/2} (7.31)

∫ 1/√(x^2 ± a^2) dx = ln |x + √(x^2 ± a^2)| (7.32)

∫ 1/√(a^2 - x^2) dx = sin^-1 x/a (7.33)

∫ x/√(x^2 ± a^2) dx = √(x^2 ± a^2) (7.34)

∫ x/√(a^2 - x^2) dx = -√(a^2 - x^2) (7.35)

∫ x^2/√(x^2 ± a^2) dx = 1/2 x√(x^2 ± a^2) ± 1/2 a^2 ln |x + √(x^2 ± a^2)| (7.36)

∫ √(ax^2+bx+c) dx = (b+2ax)/(4a) √(ax^2+bx+c) + 4ac-b^2/(8a^{3/2}) ln |2ax+b+2√(a(ax^2+bx+c))| (7.37)

∫ x√(ax^2+bx+c) dx = 1/(48a^{5/2}) (2√a√(ax^2+bx+c) (-3b^2+2abx+8a(c+ax^2)) + 3(b^3-4abc) ln |b+2ax+2√a√(ax^2+bx+c)|) (7.38)

∫ 1/√(ax^2+bx+c) dx = 1/√a ln |2ax+b+2√(a(ax^2+bx+c))| (7.39)

∫ x/√(ax^2+bx+c) dx = 1/a √(ax^2+bx+c) - b/(2a^{3/2}) ln |2ax+b+2√(a(ax^2+bx+c))| (7.40)

∫ dx/((a^2+x^2)^{3/2}) = x/(a^2√(a^2+x^2)) (7.41)

Integrals with Logarithms

∫ ln ax dx = x ln ax - x (7.42)

∫ x ln x dx = 1/2 x^2 ln x - x^2/4 (7.43)

∫ x^2 ln x dx = 1/3 x^3 ln x - x^3/9 (7.44)

∫ x^n ln x dx = x^{n+1} (ln x/(n+1) - 1/((n+1)^2)), n ≠ -1 (7.45)

∫ ln x/x dx = 1/2 (ln ax)^2 (7.46)

∫ ln x/x^2 dx = -1/x - ln x/x (7.47)

∫ ln(ax+b) dx = (x+b/a) ln(ax+b) - x, a ≠ 0 (7.48)

∫ ln(x^2+a^2) dx = x ln(x^2+a^2) + 2a tan^-1 x/a - 2x (7.49)

∫ ln(x^2-a^2) dx = x ln(x^2-a^2) + a ln x/(x-a) - 2x (7.50)

∫ ln(ax^2+bx+c) dx = 1/a √(4ac-b^2) tan^-1 (2ax+b)/√(4ac-b^2) - 2x + (b/2a+x) ln(ax^2+bx+c) (7.51)

Integrals with Exponentials

Integrals with Trigonometric Functions

∫ x ln(ax + b) dx = (bx)/(2a) - 1/4 x^2 + 1/2 (x^2 - (b^2)/(a^2)) ln(ax + b) (7.52)

∫ x ln(a^2 - b^2 x^2) dx = -1/2 x^2 + 1/2 (x^2 - (a^2)/(b^2)) ln(a^2 - b^2 x^2) (7.53)

∫ (ln x)^2 dx = 2x - 2x ln x + x(ln x)^2 (7.54)

∫ (ln x)^3 dx = -6x + x(ln x)^3 - 3x(ln x)^2 + 6x ln x (7.55)

∫ x(ln x)^2 dx = (x^2)/4 + 1/2 x^2 (ln x)^2 - 1/2 x^2 ln x (7.56)

∫ x^2 (ln x)^2 dx = (2x^3)/(27) + 1/3 x^3 (ln x)^2 - 2/9 x^3 ln x (7.57)

∫ e^{ax} dx = 1/a e^{ax} (7.58)

∫ √x e^{ax} dx = 1/a √x e^{ax} + (i√π)/(2a^{3/2}) erf(i√ax), where erf(x) = 2/√π ∫\_0^x e^{-t^2} dt (7.59)

∫ x e^x dx = (x - 1)e^x (7.60)

∫ x e^{ax} dx = (x/a - 1/a^2) e^{ax} (7.61)

∫ x^2 e^x dx = (x^2 - 2x + 2) e^x (7.62)

∫ x^2 e^{ax} dx = (x^2/a - 2x/a^2 + 2/a^3) e^{ax} (7.63)

∫ x^3 e^x dx = (x^3 - 3x^2 + 6x - 6) e^x (7.64)

∫ x^n e^{ax} dx = (x^n e^{ax})/a - n/a ∫ x^{n-1} e^{ax} dx (7.65)

∫ x^n e^{ax} dx = ((-1)^n)/(a^{n+1}) Γ[1 + n, -ax], where Γ(a, x) = ∫\_x^∞ t^{a-1} e^{-t} dt (7.66)

∫ e^{ax^2} dx = -(i√π)/(2√a) erf(ix√a) (7.67)

∫ e^{-ax^2} dx = (√π)/(2√a) erf(x√a) (7.68)

∫ x e^{-ax^2} dx = -1/(2a) e^{-ax^2} (7.69)

∫ x^2 e^{-ax^2} dx = 1/4 √(π/a^3) erf(x√a) - x/(2a) e^{-ax^2} (7.70)

∫ sin ax dx = -1/a cos ax (7.71)

∫ sin^2 ax dx = x/2 - (sin 2ax)/(4a) (7.72)

∫ sin^3 ax dx = -(3 cos ax)/(4a) + (cos 3ax)/(12a) (7.73)

∫ sin^n ax dx = -1/a cos ax {}\_2F\_1[1/2, (1-n)/2, 3/2, cos^2 ax] (7.74)

∫ cos ax dx = 1/a sin ax (7.75)

∫ cos^2 ax dx = x/2 + (sin 2ax)/(4a) (7.76)

∫ cos^3 ax dx = (3 sin ax)/(4a) + (sin 3ax)/(12a) (7.77)

∫ cos^p ax dx = -1/(a(1+p)) cos^{1+p} ax × {}\_2F\_1[1/2+p, 1/2, 3/2+p, cos^2 ax] (7.78)

∫ cos x sin x dx = 1/2 sin^2 x + c\_1 = -1/2 cos^2 x + c\_2 = -1/4 cos 2x + c\_3 (7.79)

∫ cos ax sin bx dx = (cos[(a-b)x])/2(a-b) - (cos[(a+b)x])/2(a+b), a ≠ b (7.80)

∫ sin^2 ax cos bx dx = -(sin[(2a-b)x])/4(2a-b) + (sin bx)/(2b) - (sin[(2a+b)x])/4(2a+b) (7.81)

∫ sin^2 x cos x dx = 1/3 sin^3 x (7.82)

∫ cos^2 ax sin bx dx = (cos[(2a-b)x])/4(2a-b) - (cos bx)/(2b) - (cos[(2a+b)x])/4(2a+b) (7.83)

∫ cos^2 ax sin ax dx = -1/3a cos^3 ax (7.84)

∫ sin^2 ax cos^2 bxdx = x/4 - (sin 2ax)/(8a) - (sin[2(a-b)x])/16(a-b) + (sin 2bx)/(8b) - (sin[2(a+b)x])/16(a+b) (7.85)

∫ sin^2 ax cos^2 ax dx = x/8 - (sin 4ax)/(32a) (7.86)

∫ tan ax dx = -1/a ln cos ax (7.87)

∫ tan^2 ax dx = -x + 1/a tan ax (7.88)

∫ tan^n ax dx = (tan^{n+1} ax)/(a(1+n)) × {}\_2F\_1((n+1)/2, 1, (n+3)/2, -tan^2 ax) (7.89)

∫ tan^3 ax dx = 1/a ln cos ax + 1/(2a) sec^2 ax (7.90)

∫ sec ax dx = ln |sec ax + tan ax| = 2 tanh^{-1} (tan (x/2)) (7.91)

∫ sec^2 ax dx = 1/a tan ax (7.92)

∫ sec^3 x dx = 1/2 sec x tan x + 1/2 ln |sec x + tan x| (7.93)

∫ sec x tan x dx = sec x (7.94)

∫ sec^2 x tan x dx = 1/2 sec^2 x (7.95)

∫ sec^n x tan x dx = 1/n sec^n x, n ≠ 0 (7.96)

∫ csc x dx = ln |tan (x/2)| = ln |csc x - cot x| + C (7.97)

∫ csc^2 ax dx = -1/a cot ax (7.98)

∫ csc^3 x dx = -1/2 cot x csc x + 1/2 ln |csc x - cot x| (7.99)

∫ csc^n x cot x dx = -1/n csc^n x, n ≠ 0 (7.100)

∫ sec x csc x dx = ln |tan x| (7.101)

## Products of Trigonometric Functions and Monomials

$$\int x \cos x \, dx = \cos x + x \sin x \quad (7.102)$$

$$\int x \cos ax \, dx = \frac{1}{a^2} \cos ax + \frac{x}{a} \sin ax \quad (7.103)$$

$$\int x^2 \cos x \, dx = 2x \cos x + (x^2 - 2) \sin x \quad (7.104)$$

$$\int x^2 \cos ax \, dx = \frac{2x \cos ax}{a^2} + \frac{a^2 x^2 - 2}{a^3} \sin ax \quad (7.105)$$

$$\int x^n \cos x \, dx = -\frac{1}{2}(i)^{n+1} [\Gamma(n+1, -ix) + (-1)^n \Gamma(n+1, ix)] \quad (7.106)$$

$$\int x^n \cos ax \, dx = \frac{1}{2}(ia)^{1-n} [(-1)^n \Gamma(n+1, -iax) - \Gamma(n+1, ixa)] \quad (7.107)$$

$$\int x \sin x \, dx = -x \cos x + \sin x \quad (7.108)$$

$$\int x \sin ax \, dx = -\frac{x \cos ax}{a} + \frac{\sin ax}{a^2} \quad (7.109)$$

$$\int x^2 \sin x \, dx = (2 - x^2) \cos x + 2x \sin x \quad (7.110)$$

$$\int x^2 \sin ax \, dx = \frac{2 - a^2 x^2}{a^3} \cos ax + \frac{2x \sin ax}{a^2} \quad (7.111)$$

$$\int x^n \sin x \, dx = -\frac{1}{2}(i)^n [\Gamma(n+1, -ix) - (-1)^n \Gamma(n+1, -ix)] \quad (7.112)$$

$$\int x \cos^2 x \, dx = \frac{x^2}{4} + \frac{1}{8} \cos 2x + \frac{1}{4} x \sin 2x \quad (7.113)$$

$$\int x \sin^2 x \, dx = \frac{x^2}{4} - \frac{1}{8} \cos 2x - \frac{1}{4} x \sin 2x \quad (7.114)$$

$$\int x \tan^2 x \, dx = -\frac{x^2}{2} + \ln \cos x + x \tan x \quad (7.115)$$

$$\int x \sec^2 x \, dx = \ln \cos x + x \tan x \quad (7.116)$$

## Products of Trigonometric Functions and Exponentials

$$\int e^x \sin x \, dx = \frac{1}{2} e^x (\sin x - \cos x) \quad (7.117)$$

$$\int e^{bx} \sin ax \, dx = \frac{1}{a^2 + b^2} e^{bx} (b \sin ax - a \cos ax) \quad (7.118)$$

$$\int e^x \cos x \, dx = \frac{1}{2} e^x (\sin x + \cos x) \quad (7.119)$$

$$\int e^{bx} \cos ax \, dx = \frac{1}{a^2 + b^2} e^{bx} (a \sin ax + b \cos ax) \quad (7.120)$$

$$\int x e^x \sin x \, dx = \frac{1}{2} e^x (\cos x - x \cos x + x \sin x) \quad (7.121)$$

$$\int x e^x \cos x \, dx = \frac{1}{2} e^x (x \cos x - \sin x + x \sin x) \quad (7.122)$$

## Integrals of Hyperbolic Functions

$$\int \cosh ax \, dx = \frac{1}{a} \sinh ax \quad (7.123)$$

$$\int e^{ax} \cosh bx \, dx = \begin{cases} \frac{e^{ax}}{a^2 - b^2} [a \cosh bx - b \sinh bx] & a \neq b \\ \frac{e^{2ax}}{4a} + \frac{x}{2} & a = b \end{cases} \quad (7.124)$$

$$\int \sinh ax \, dx = \frac{1}{a} \cosh ax \quad (7.125)$$

$$\int e^{ax} \sinh bx \, dx = \begin{cases} \frac{e^{ax}}{a^2 - b^2} [-b \cosh bx + a \sinh bx] & a \neq b \\ \frac{e^{2ax}}{4a} - \frac{x}{2} & a = b \end{cases} \quad (7.126)$$

$$\int \tanh ax \, dx = \frac{1}{a} \ln \cosh ax \quad (7.127)$$

$$\int e^{ax} \tanh bx \, dx = \begin{cases} \frac{e^{(a+2b)x}}{(a+2b)} {}_2F_1 \left[ 1 + \frac{a}{2b}, 1, 2 + \frac{a}{2b}, -e^{2bx} \right] \\ \quad - \frac{1}{a} e^{ax} {}_2F_1 \left[ 1, \frac{a}{2b}, 1 + \frac{a}{2b}, -e^{2bx} \right] & a \neq b \\ \frac{e^{ax} - 2 \tan^{-1}[e^{ax}]}{a} & a = b \end{cases} \quad (7.128)$$

$$\int \cos ax \cosh bx \, dx = \frac{1}{a^2 + b^2} [a \sin ax \cosh bx + b \cos ax \sinh bx] \quad (7.129)$$

$$\int \cos ax \sinh bx \, dx = \frac{1}{a^2 + b^2} [b \cos ax \cosh bx + a \sin ax \sinh bx] \quad (7.130)$$

$$\int \sin ax \cosh bx \, dx = \frac{1}{a^2 + b^2} [-a \cos ax \cosh bx + b \sin ax \sinh bx] \quad (7.131)$$

$$\int \sin ax \sinh bx \, dx = \frac{1}{a^2 + b^2} [b \cosh bx \sin ax - a \cos ax \sinh bx] \quad (7.132)$$

$$\int \sinh ax \cosh ax \, dx = \frac{1}{4a} [-2ax + \sinh 2ax] \quad (7.133)$$

$$\int \sinh ax \cosh bx \, dx = \frac{1}{b^2 - a^2} [b \cosh bx \sinh ax - a \cosh ax \sinh bx] \quad (7.134)$$



Problem	Status	Comment	pandapythoner	mangooste	allvik
A - 1					
B - 2					
C - 3					
D - 4					
E - 5					
F - 6					
G - 7					
H - 8					
I - 9					
J - 10					
K - 11					
L - 12					
M - 13					
N - 14					
O - 15					