National Research University Higher School of Economics

Elderly Passion Fruit

Alexandr Nekrasov, Iurii Pustovalov, Igor Markelov

November 19, 2022

Contest (1)

template.cpp

36 lines

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

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(a) (a).begin(), (a).end()
#define vin(a) \
  for (auto& i : a) cin >> i

mt19937 rnd(chrono::steady_clock::now().time_since_epoch().count());

template <typename T1, typename T2>
inline void chkmin(T1& x, const T2& y) {
  if (y < x) {
    x = y;
  }
}

template <typename T1, typename T2>
inline void chkmax(T1& x, const T2& y) {
  if (x < y) {
    x = y;
  }
}

signed main() {
  cin.tie(0)->sync_with_stdio(0);
  cout.precision(20), cout.setf(ios::fixed);

  return 0;
}
```

genfolders.sh

6 lines

```
for f in {a..z}
do
    mkdir $f
    cp template.cpp $f/$f.cpp
    touch $f/in
done
```

hash.sh

3 lines

```
# 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
```

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).

e44914, 13 lines

```
#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: $\mathcal{O}(\log(n))$

cf36cc, 29 lines

```
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>

typedef __gnu_pbds::tree<int, __gnu_pbds::null_type, std::less<int>,
                         __gnu_pbds::rb_tree_tag,
                         __gnu_pbds::tree_order_statistics_node_update>
    oset;

#include <iostream>

int main() {
  oset X;
  X.insert(1);
  X.insert(2);
  X.insert(4);
  X.insert(8);
  X.insert(16);

  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
}
```

Strings (3)

Manacher.cpp

Description: Manacher algorithm
Time: $\mathcal{O}(n)$

a6ddfb, 23 lines

```
vector<int> manacherOdd(string s) {
  int n = s.size();
  vector<int> d1(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: $\mathcal{O}(n\log(n))$

3caefc, 45 lines

```
vector<int> buildSuffixArray(string& s) {
  // Remove, if you want to sort cyclic shifts
  s += "$";
  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;
  }
  return a;
}
```

Lcp.cpp

Description: lcp array
Time: $\mathcal{O}(n)$

fa8216, 25 lines

```
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;
  }
  // for suffixes!!!
  s.pop_back();
  a.erase(a.begin());
  lcp.erase(lcp.begin());
  return lcp;
}
```

Eertree.cpp

Description: Creates Eertree of string str
Time: $\mathcal{O}(n)$

7924c8, 37 lines

```
struct eertree {
  int len[MAXN], suffLink[MAXN];
  int to[MAXN][26];
  int numV, v;
  void addLetter(int n, string& str) {
    while (str[n - len[v] - 1] != str[n]) v = suffLink[v];
    int u = suffLink[v];
    while (str[n - len[u] - 1] != str[n]) u = suffLink[u];
    int u_ = to[u][str[n] - 'a'];
    int v_ = to[v][str[n] - 'a'];
    if (v_ == -1) {
      v_ = to[v][str[n] - 'a'] = numV;
      len[numV++] = len[v] + 2;
      suffLink[v_] = u_;
    }
    v = v_;
  }
  void init() {
    len[0] = -1;
    len[1] = 0;
    suffLink[1] = 0;
    suffLink[0] = 0;
    numV = 2;
    for (int i = 0; i < 26; ++i) {
      to[0][i] = numV++;
      suffLink[numV - 1] = 1;
      len[numV - 1] = 1;
    }
    v = 0;
  }
  void init(int sz) {
    for (int i = 0; i < sz; ++i) {
      len[i] = suffLink[i] = 0;
      for (int j = 0; j < 26; ++j) to[i][j] = -1;
    }
  }
};
```

SuffixAutomaton.cpp

Description: Build suffix automaton.
Time: $\mathcal{O}(n)$

662a10, 45 lines

```
struct state {
  int len, link;
  map<char, int> next;
};

const int MAXLEN = 100000;
state st[MAXLEN * 2];
int sz, last;

void sa_init() {
  sz = last = 0;
  st[0].len = 0;
  st[0].link = -1;
  ++sz;
  /*
  // if you want to build an automaton for different strings:
  for (int i=0; i<MAXLEN*2; ++i)
          st[i].next.clear();
  */
}

void sa_extend(char c) {
  int cur = sz++;
  st[cur].len = st[last].len + 1;
  int p;
  for (p = last; p != -1 && !st[p].next.count(c); p = st[p].link)
    st[p].next[c] = cur;
  if (p == -1)
    st[cur].link = 0;
  else {
    int q = st[p].next[c];
    if (st[p].len + 1 == st[q].len)
      st[cur].link = q;
    else {
      int clone = sz++;
      st[clone].len = st[p].len + 1;
      st[clone].next = st[q].next;
      st[clone].link = st[q].link;
      for (; p != -1 && st[p].next[c] == q; p = st[p].link)
        st[p].next[c] = clone;
      st[q].link = st[cur].link = clone;
    }
  }
  last = cur;
}
```

Graph (4)

Hungarian.cpp

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

5afee5, 41 lines

```
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: $\mathcal{O}(n^3)$

23839d, 118 lines

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

3d8a3f, 142 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;
    }
  }
}
```

Pushrelabel.cpp

Description: Maxflow
Time: $\mathcal{O}\left(n^2 m\right)$

1dbe57, 87 lines

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

typedef long long ll;

struct MaxFlow {
  static const ll INF = 1e18 + 228;  // maybe int?
  struct edge {
    int to, rev;
    ll cap;  // maybe int?
  };

  int n;
  vector<vector<edge>> g;
  vector<ll> ex;  // maybe int?
  vector<int> q;

  ll flow(int t) {  // maybe int?
    while (true) {
      vector<int> dist(n, n);
      dist[t] = 0;
      int l = 0;
      int r = 1;
      q[0] = t;
```

```
      while (l != r) {
        int v = q[l++];
        for (auto e : g[v]) {
          if (g[e.to][e.rev].cap > 0 && dist[e.to] > dist[v] + 1) {
            dist[e.to] = dist[v] + 1;
            q[r++] = e.to;
          }
        }
      }
      ll was = ex[t];
      for (int ind = r - 1; ind >= 0; ind--) {
        int v = q[ind];
        if (ex[v] == 0)
          continue;
        for (auto &e : g[v]) {
          if (dist[e.to] + 1 == dist[v] && e.cap > 0) {
            auto f = min(ex[v], e.cap);
            e.cap -= f;
            ex[e.to] += f;
            ex[v] -= f;
            g[e.to][e.rev].cap += f;
          }
        }
      }
      if (was == ex[t]) {
        break;
      }
    }
    return ex[t];
  }
  MaxFlow(int n) : n(n) {
    g.resize(n);
    ex.resize(n);
    q.resize(n);
  }
  ll run(int s, int t) {  // maybe int?
    ex[s] = INF;
    return flow(t);
  }
  void add_edge(int a, int b, int c, int cr = 0) {
    int sza = g[a].size();
    int szb = g[b].size();
    g[a].push_back({b, szb, c});
    g[b].push_back({a, sza, cr});
  }
};

int main() {
  int n;
  cin >> n;
  MaxFlow mf(n);
  int s = 0, t = n - 1;
  int m;
  cin >> m;
  for (int i = 0; i < m; i++) {
    int a, b, c;
    cin >> a >> b >> c;
    a--;
    b--;
    mf.add_edge(a, b, c);
  }
  cout << mf.run(s, t) << endl;
}
```

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;
      }
    }
  }
}
```

Geometry (5)

Point.cpp

Description: struct Point

cbfa4e, 46 lines

```
struct Point {
  ld x = 0, y = 0;
  Point() = default;
  Point(ld _x, ld _y) : x(_x), y(_y) {
  }
  Point ort() const {
    return Point(-y, x);
  }
  int half() const {
    return sign(y) == 1 || (sign(y) == 0 && sign(x) >= 0);
  }
  bool operator<(const Point& other) const {
    if (sign(y - other.y) != 0) {
      return y < other.y;
    } else if (sign(x - other.x) != 0) {
      return x < other.x;
    } else {
      return false;
    }
  }
  Point turn(ld sin, ld cos) const {
    return Point(x * cos - y * sin, x * sin + y * cos);
  }
  Point turn(ld phi) const {
    return turn(sin(phi), cos(phi));
  }
};

#define Vec Point

ld getAngle(Vec& lhs, Vec& rhs) {
  return atan2(lhs ^ rhs, lhs * rhs);
}

bool cmpHalf(const Vec& lhs, const Vec& rhs) {
  if (lhs.half() != rhs.half()) {
    return lhs.half();
  } else {
    int sgn = sign(lhs ^ rhs);
    if (!sgn) {
      return lhs.len2() < rhs.len2();
    } else {
      return sgn == 1;
    }
  }
}
```

Line.cpp

Description: struct Line

02e3a0, 30 lines

```
struct Line {
  ld a = 0, b = 0, c = 0;
  Line() = default;
  void norm() {
    // for half planes
    ld d = Vec(a, b).len();
    assert(sign(d) > 0);
    a /= d;
    b /= d;
    c /= d;
  }
  Line(ld _a, ld _b, ld _c) : a(_a), b(_b), c(_c) {
    norm();
  }
  Line(Point x, Point y)
      : a(y.y - x.y), b(x.x - y.x), c(x.y * y.x - x.x * y.y) {
    norm();
  }
  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;
  }
};
```

Intersections.cpp

Description: Geometry intersections

a7a42d, 78 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 isCrossed(Point& a, Point& b, Point& c, 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;
  Vec v1, v2, v3;
  v1 = b - a;
  v2 = c - a;
  v3 = d - a;
  if (sign(v1 ^ v2) * sign(v1 ^ v3) == 1) return false;
  v1 = d - c;
  v2 = a - c;
  v3 = b - c;
  if (sign(v1 ^ v2) * sign(v1 ^ v3) == 1) return false;
  return true;
}
```

```
bool cross(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 = Point(dx / d, dy / d);
  return true;
}

int cross(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 && o1 == o2) {
    return 3;
  }
  ld len = (o1 - o2).len();
  if (sign(len - r1 - r2) == 1 || sign(r1 - len - r2) == 1) {
    return 0;
  }
  ld d = (sq(r1) - sq(r2) + sq(len)) / 2 / len;
  Vec v = (o2 - o1).norm();
  Point a = o1 + v * d;
  if (sign(len - r1 - r2) == 0 || sign(len + r2 - r1) == 0) {
    I1 = a;
    return 1;
  }
  v = v.ort() * sqrt(sq(r1) - sq(d));
  I1 = a + v;
  I2 = a - v;
  return 2;
}

int cross(Point& o, ld r, Line& l, Point& I1, Point& I2) {
  ld len = dist(l, o);
  int sgn = sign(len - r);
  if (sgn == 1) {
    return 0;
  }
  Vec v = Vec(l.a, l.b).norm() * len;
  if (sign(l.eval(o + v)) != 0) {
    v = Point() - v;
  }
  Point a = o + v;
  if (sgn == 0) {
    I1 = a;
    return 1;
  }
  v = Vec(-l.b, l.a).norm() * sqrt(sq(r) - sq(len));
  I1 = a + v;
  I2 = a - v;
  return 2;
}
```

Tangents.cpp

Description: Tangents to circles.

649ac8, 41 lines

```
int tangents(Point& o, ld r, Point& p, Point& I1, Point& I2) {
  ld len = (o - p).len();
  int sgn = sign(len - r);
  if (sgn == -1) {
    return 0;
  } else if (sgn == 0) {
    I1 = p;
    return 1;
  } else {
    ld x = sq(r) / len;
    Vec v = (p - o).norm() * x;
    Point a = o + v;
    v = (p - o).norm().ort() * 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);
}

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;
}
```

Polygon.cpp

Description: Polygon functions

48483d, 68 lines

```
ld area(vector<Point>& p) {
  ld ans = 0;
  int n = p.size();
  for (int i = 0; i < n; ++i) {
    ans += p[i] ^ p[i + 1 < n ? i + 1 : 0];
  }
  return abs(ans) / 2;
}

ld perimeter(vector<Point>& p) {
  ld ans = 0;
```

```cpp
  int n = p.size();
  for (int i = 0; i < n; ++i) {
    ans += (p[i] - p[i + 1 < n ? i + 1 : 0]).len();
  }
  return ans;
}

bool isCounterclockwise(vector<Point>& p) {
  int n = p.size();
  int pos = min_element(all(p)) - p.begin();
  return sign((p[pos + 1 < n ? pos + 1 : 0] - p[pos]) ^
              (p[pos - 1 >= 0 ? pos - 1 : n - 1] - p[pos])) == 1;
}

bool isConvex(vector<Point>& p) {
  int n = p.size();
  int sgn = 0;
  for (int i = 0; i < n; ++i) {
    int cur_sgn = sign((p[i - 1 >= 0 ? i - 1 : n - 1] - p[i]) ^
                       (p[i + 1 < n ? i + 1 : 0] - p[i]));
    if (sgn && sgn != cur_sgn) {
      return false;
    }
    sgn = cur_sgn;
  }
  return true;
}

vector<Point> convexHull(vector<Point> p) {
  if (p.empty()) {
    return {};
  }
  int n = p.size();
  int pos = min_element(all(p)) - p.begin();
  swap(p[0], p[pos]);
  for (int i = 1; i < n; ++i) p[i] = p[i] - p[0];
  sort(p.begin() + 1, p.end(), [&](Point& lhs, Point& rhs) -> bool {
    int sgn = sign(lhs ^ rhs);
    if (!sgn) {
      return lhs.len2() < rhs.len2();
    }
    return sgn == 1;
  });
  for (int i = 1; i < n; ++i) p[i] = p[i] + p[0];
  int top = 0;
  for (int i = 0; i < n; ++i) {
    while (top >= 2) {
      Vec v1 = p[top - 1] - p[top - 2];
      Vec v2 = p[i] - p[top - 1];
      if (sign(v1 ^ v2) == 1) break;
      --top;
    }
    p[top++] = p[i];
  }
  p.resize(top);
  return p;
}
```

IsInPolygon.cpp

Description: Is in polygon functions

c97da7, 64 lines

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

bool isIn(vector<Point>& p, Point& a) {
  int n = p.size();
  // depends on limitations
  Point b = a + Point(1e9, 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 true;
    }
    cnt += isCrossed(x, y, a, b);
  }
  return 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 false;
  if (sign((p[n - 1] - p[0]) ^ (a - p[0])) > 0) return false;
```

```
  int pos = lower_bound(p.begin() + 2, p.end(), a,
                        [&](Point lhs, Point rhs) -> bool {
                          return sign((lhs - p[0]) ^ (rhs - 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: $\mathcal{O}(n)$

3a9573, 21 lines

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

TangentsAlex.cpp

Description: Find both tangets to the convex polygon.
(Zakaldovany algos mozhet sgonyat za pivom tak zhe).
Time: $\mathcal{O}(\log(n))$

b2b424, 15 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 = l; 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: $\mathcal{O}(n)$ (expected)

4cb75f, 33 lines

```
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({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 = getPoint(lines[i]);
    for (int j = i + 1; j < n; ++j) {
      if (lines[i] == lines[j]) continue;
      Point p, pj = getPoint(lines[j]);
      if (!cross(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: $\mathcal{O}(n\log(n))$

2a2340, 64 lines

```
Vec getPoint(Line l) {
  return Vec(-l.b, l.a);
}

bool bad(Line a, Line b, Line c) {
  Point x;
  assert(cross(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 (a.y >= 0 && b.y < 0) return 1;
    if (a.y < 0 && b.y >= 0) return 0;
```

```
    if (a.y == 0 && b.y == 0) return a.x > 0 && b.x < 0;
    return (a ^ b) > 0;
  });

  vector<pair<Line, int> > st;
  for (int it = 0; it < 2; it++) {
    for (int i = 0; i < (int)lines.size(); i++) {
      bool flag = false;
      while (!st.empty()) {
        if ((getPoint(st.back().first) - getPoint(lines[i])).len() < 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 I;
      assert(cross(st[j].first, st[j + 1].first, I) == 1);
      ans.push_back(I);
    }
    break;
  }
  return ans;
}
```

Math (6)

BerlekampMassey.cpp

Description: Find the shortest linear-feedback shift register

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

505033, 32 lines

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

GoncharFedor.cpp

Description: Calculating number of points $x, y \geq 0, Ax + By \leq C$

Time: $\mathcal{O}\left(\log(C)\right)$

0ef10e, 9 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;
}
```

PrimalityTest.cpp

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

af473a, 27 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 % mod;
      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, 21 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: $\mathcal{O}\left(n^{\frac{1}{4}}\right)$

f0d7c6, 49 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.pbc(i);
        n /= i;
      }
    }
    if (n != 1) res.pbc(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.pbc(j);
        return v1;
      }
    }
  }
  return {n};
}
```

NTT.cpp

Description: Calculating FFT modulo MOD
Time: $\mathcal{O}(n \log(n))$

8b7830, 55 lines

```cpp
// DONT FORGET TO CALL initNTT() AND CHECK MAXLOG
const int MOD = 998244353, MAXLOG = 20;
const int N = (1 << MAXLOG), MAXN = (1 << MAXLOG) + 228;
int rev[MAXN], w[MAXN], n, m, a[MAXN], b[MAXN], fans[MAXN];
void initNTT() {
  int g = 2;
  for (;; g++) {
    int y = g;
    for (int i = 0; i < MAXLOG - 1; ++i) {
      y = mul(y, y);
    }
    if (y == MOD - 1) {
      break;
    }
  }
  w[0] = 1;
  for (int i = 1; i < N; ++i) {
    w[i] = mul(w[i - 1], g);
  }
  rev[0] = 0;
  for (int i = 1; i < N; ++i) {
    rev[i] = (rev[i >> 1] >> 1) ^ ((i & 1) << (MAXLOG - 1));
  }
}
void NTT(int n, int LOG, int* a) {
  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 << 1) {
      for (int i = 0; i < len; ++i) {
        int x = a[st + i],
            y = mul(a[st + len + i], w[i << (MAXLOG - 1 - lvl)]);
        a[st + i] = add(x, y);
        a[st + i + len] = sub(x, y);
      }
    }
  }
}
void mul() {
  int LOG = __lg(2 * max(n, m) - 1) + 1;
  int sz = 1 << LOG;
  fill(a + n, a + sz, 0);
  fill(b + m, b + sz, 0);
  NTT(sz, LOG, a), NTT(sz, LOG, b);
  for (int i = 0; i < sz; ++i) a[i] = mul(a[i], b[i]);
  NTT(sz, LOG, a);
  int inv_sz = inv(sz);
  for (int i = 0; i < sz; ++i) fans[i] = mul(a[i], inv_sz);
  reverse(fans + 1, fans + sz);
}
// DONT FORGET TO CALL initNTT() AND CHECK MAXLOG
```

FFT.cpp

Description: Calculating product of two polynomials
Time: $\mathcal{O}(n \log(n))$

ed80ec, 44 lines

```cpp
// DONT FORGET TO INITFFT() AND CHECK MAXLOG
const ld PI = acos(-1);
using cd = complex<long double>;
const int MAXLOG = 20, N = (1 << MAXLOG), MAXN = (1 << MAXLOG) + 228;
int rev[MAXN], n, m, fans[MAXN];
cd w[MAXN], a[MAXN], b[MAXN];
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, int LOG, cd* a) {
  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 << 1) {
      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;
      }
    }
  }
}
void mul() {
  int LOG = __lg(2 * max(n, m) - 1) + 1;
  int sz = 1 << LOG;
  fill(a + n, a + sz, 0);
  fill(b + m, b + sz, 0);
  FFT(sz, LOG, a), FFT(sz, LOG, b);
  for (int i = 0; i < sz; i++) a[i] *= b[i];
  FFT(sz, LOG, a);
  for (int i = 0; i < sz; i++) fans[i] = (int)(a[i].real() / sz + 0.5);
  reverse(fans + 1, fans + sz);
}
// DONT FORGET TO INITFFT() AND CHECK MAXLOG
```

AndConvolution.cpp

Description: Calculating and-convolution modulo smth
Time: $\mathcal{O}(n\log(n))$

5dedf4, 22 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)]);
      }
    }
  }
} // https://judge.yosupo.jp/problem/bitwise_and_convolution
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;
}
```

6.1 Fun things

$ClassesCount = \dfrac{1}{|G|}\sum_{\pi\in G} I(\pi)$

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

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

$S(n,k) = S(n-1,k-1) + kS(n-1,k)$

$S(n,k) = \sum_{j=0}^{n-1}\binom{n-1}{j}S(j,k-1)$

$S(n,k) = \frac{1}{k!}\sum_{j=0}^{k}(-1)^{k+j}\binom{k}{j}j^n$

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

$\int_a^b f(x)dx \approx \dfrac{b-a}{6}(f(a) + 4f(\dfrac{a+b}{2}) + f(b))$

$x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)}, O(loglog)$

$G(n) = n \oplus (n >> 1)$

$g(n) = \sum_{d|n} f(d) \Rightarrow f(n) = \sum_{d|n} g(d)\mu(\frac{n}{d})$

$\sum_{d|n}\mu(d) = [n=1], \mu(1) = 1, \mu(p) = -1, \mu(p^k) = 0$

$\sin(a\pm b) = \sin a\cos b \pm \sin b\cos a$

$\cos(a\pm b) = \cos a\cos b \mp \sin a\sin b$

$\operatorname{tg}(a\pm b) = \dfrac{\operatorname{tg} a \pm \operatorname{tg} b}{1 \mp \operatorname{tg} a \operatorname{tg} b}$

$\operatorname{ctg}(a\pm b) = \dfrac{\operatorname{ctg} a \operatorname{ctg} b \mp 1}{\operatorname{ctg} b \pm \operatorname{ctg} a}$

$\sin\dfrac{a}{2} = \pm\sqrt{\dfrac{1-\cos a}{2}}$

$\cos\dfrac{a}{2} = \pm\sqrt{\dfrac{1+\cos a}{2}}$

$\operatorname{tg}\dfrac{a}{2} = \dfrac{\sin a}{1-\cos a} = \dfrac{1-cosa}{sina}$

$\sin a\sin b = \dfrac{\cos(a-b) - \cos(a+b)}{2}$

$\sin a\cos b = \dfrac{\sin(a-b) + \sin(a+b)}{2}$

$\cos a\cos b = \dfrac{\cos(a-b) + \cos(a+b)}{2}$

Table of Basic Integrals (7)

Basic Forms

$$\int x^n dx = \frac{1}{n+1}x^{n+1}, \; n \neq -1 \tag{7.1}$$

$$\int \frac{1}{x} dx = \ln|x| \tag{7.2}$$

$$\int u dv = uv - \int v du \tag{7.3}$$

$$\int \frac{1}{ax+b} dx = \frac{1}{a}\ln|ax+b| \tag{7.4}$$

Integrals of Rational Functions

$$\int \frac{1}{(x+a)^2} dx = -\frac{1}{x+a} \tag{7.5}$$

$$\int (x+a)^n dx = \frac{(x+a)^{n+1}}{n+1}, n \neq -1 \tag{7.6}$$

$$\int x(x+a)^n dx = \frac{(x+a)^{n+1}((n+1)x-a)}{(n+1)(n+2)} \tag{7.7}$$

$$\int \frac{1}{1+x^2} dx = \tan^{-1} x \tag{7.8}$$

$$\int \frac{1}{a^2+x^2} dx = \frac{1}{a}\tan^{-1}\frac{x}{a} \tag{7.9}$$

$$\int \frac{x}{a^2+x^2} dx = \frac{1}{2}\ln|a^2+x^2| \tag{7.10}$$

$$\int \frac{x^2}{a^2+x^2} dx = x - a\tan^{-1}\frac{x}{a} \tag{7.11}$$

$$\int \frac{x^3}{a^2+x^2} dx = \frac{1}{2}x^2 - \frac{1}{2}a^2\ln|a^2+x^2| \tag{7.12}$$

$$\int \frac{1}{ax^2+bx+c} dx = \frac{2}{\sqrt{4ac-b^2}}\tan^{-1}\frac{2ax+b}{\sqrt{4ac-b^2}} \tag{7.13}$$

$$\int \frac{1}{(x+a)(x+b)} dx = \frac{1}{b-a}\ln\frac{a+x}{b+x}, \; a \neq b \tag{7.14}$$

$$\int \frac{x}{(x+a)^2} dx = \frac{a}{a+x} + \ln|a+x| \tag{7.15}$$

$$\int \frac{x}{ax^2+bx+c} dx = \frac{1}{2a}\ln|ax^2+bx+c| - \frac{b}{a\sqrt{4ac-b^2}}\tan^{-1}\frac{2ax+b}{\sqrt{4ac-b^2}} \tag{7.16}$$

Integrals with Roots

$$\int \sqrt{x-a}\; dx = \frac{2}{3}(x-a)^{3/2} \tag{7.17}$$

$$\int \frac{1}{\sqrt{x \pm a}}\; dx = 2\sqrt{x \pm a} \tag{7.18}$$

$$\int \frac{1}{\sqrt{a-x}}\; dx = -2\sqrt{a-x} \tag{7.19}$$

$$\int x\sqrt{x-a}\; dx = \begin{cases} \frac{2a}{3}(x-a)^{3/2} + \frac{2}{5}(x-a)^{5/2}, \text{ or} \\ \frac{2}{3}x(x-a)^{3/2} - \frac{4}{15}(x-a)^{5/2}, \text{ or} \\ \frac{2}{15}(2a+3x)(x-a)^{3/2} \end{cases} \tag{7.20}$$

$$\int \sqrt{ax+b}\; dx = \left(\frac{2b}{3a} + \frac{2x}{3}\right)\sqrt{ax+b} \tag{7.21}$$

$$\int (ax+b)^{3/2}\; dx = \frac{2}{5a}(ax+b)^{5/2} \tag{7.22}$$

$$\int \frac{x}{\sqrt{x \pm a}}\; dx = \frac{2}{3}(x \mp 2a)\sqrt{x \pm a} \tag{7.23}$$

$$\int \sqrt{\frac{x}{a-x}}\; dx = -\sqrt{x(a-x)} - a\tan^{-1}\frac{\sqrt{x(a-x)}}{x-a} \tag{7.24}$$

$$\int \sqrt{\frac{x}{a+x}}\; dx = \sqrt{x(a+x)} - a\ln\left[\sqrt{x} + \sqrt{x+a}\right] \tag{7.25}$$

$$\int x\sqrt{ax+b}\; dx = \frac{2}{15a^2}(-2b^2 + abx + 3a^2x^2)\sqrt{ax+b} \tag{7.26}$$

$$\int \sqrt{x(ax+b)}\; dx = \frac{1}{4a^{3/2}}\left[(2ax+b)\sqrt{ax(ax+b)} - b^2\ln\left|a\sqrt{x} + \sqrt{a(ax+b)}\right|\right] \tag{7.27}$$

$$\int \sqrt{x^3(ax+b)}\ dx = \left[\frac{b}{12a} - \frac{b^2}{8a^2x} + \frac{x}{3}\right]\sqrt{x^3(ax+b)} + \frac{b^3}{8a^{5/2}}\ln\left|a\sqrt{x} + \sqrt{a(ax+b)}\right| \tag{7.28}$$

$$\int \sqrt{x^2 \pm a^2}\ dx = \frac{1}{2}x\sqrt{x^2 \pm a^2} \pm \frac{1}{2}a^2 \ln\left|x + \sqrt{x^2 \pm a^2}\right| \tag{7.29}$$

$$\int \sqrt{a^2 - x^2}\ dx = \frac{1}{2}x\sqrt{a^2 - x^2} + \frac{1}{2}a^2 \tan^{-1}\frac{x}{\sqrt{a^2 - x^2}} \tag{7.30}$$

$$\int x\sqrt{x^2 \pm a^2}\ dx = \frac{1}{3}\left(x^2 \pm a^2\right)^{3/2} \tag{7.31}$$

$$\int \frac{1}{\sqrt{x^2 \pm a^2}}\ dx = \ln\left|x + \sqrt{x^2 \pm a^2}\right| \tag{7.32}$$

$$\int \frac{1}{\sqrt{a^2 - x^2}}\ dx = \sin^{-1}\frac{x}{a} \tag{7.33}$$

$$\int \frac{x}{\sqrt{x^2 \pm a^2}}\ dx = \sqrt{x^2 \pm a^2} \tag{7.34}$$

$$\int \frac{x}{\sqrt{a^2 - x^2}}\ dx = -\sqrt{a^2 - x^2} \tag{7.35}$$

$$\int \frac{x^2}{\sqrt{x^2 \pm a^2}}\ dx = \frac{1}{2}x\sqrt{x^2 \pm a^2} \mp \frac{1}{2}a^2 \ln\left|x + \sqrt{x^2 \pm a^2}\right| \tag{7.36}$$

$$\int \sqrt{ax^2 + bx + c}\ dx = \frac{b + 2ax}{4a}\sqrt{ax^2 + bx + c} + \frac{4ac - b^2}{8a^{3/2}}\ln\left|2ax + b + 2\sqrt{a(ax^2 + bx^{+}c)}\right| \tag{7.37}$$

$$\int x\sqrt{ax^2 + bx + c}\ dx = \frac{1}{48a^{5/2}}\left(2\sqrt{a}\sqrt{ax^2 + bx + c}\left(-3b^2 + 2abx + 8a(c + ax^2)\right)\right.$$
$$\left. + 3(b^3 - 4abc)\ln\left|b + 2ax + 2\sqrt{a}\sqrt{ax^2 + bx + c}\right|\right) \tag{7.38}$$

$$\int \frac{1}{\sqrt{ax^2 + bx + c}}\ dx = \frac{1}{\sqrt{a}}\ln\left|2ax + b + 2\sqrt{a(ax^2 + bx + c)}\right| \tag{7.39}$$

$$\int \frac{x}{\sqrt{ax^2 + bx + c}}\ dx = \frac{1}{a}\sqrt{ax^2 + bx + c} - \frac{b}{2a^{3/2}}\ln\left|2ax + b + 2\sqrt{a(ax^2 + bx + c)}\right| \tag{7.40}$$

$$\int \frac{dx}{(a^2 + x^2)^{3/2}} = \frac{x}{a^2\sqrt{a^2 + x^2}} \tag{7.41}$$

Integrals with Logarithms

$$\int \ln ax\ dx = x\ln ax - x \tag{7.42}$$

$$\int x\ln x\ dx = \frac{1}{2}x^2\ln x - \frac{x^2}{4} \tag{7.43}$$

$$\int x^2\ln x\ dx = \frac{1}{3}x^3\ln x - \frac{x^3}{9} \tag{7.44}$$

$$\int x^n\ln x\ dx = x^{n+1}\left(\frac{\ln x}{n+1} - \frac{1}{(n+1)^2}\right), \quad n \neq -1 \tag{7.45}$$

$$\int \frac{\ln ax}{x}\ dx = \frac{1}{2}(\ln ax)^2 \tag{7.46}$$

$$\int \frac{\ln x}{x^2}\ dx = -\frac{1}{x} - \frac{\ln x}{x} \tag{7.47}$$

$$\int \ln(ax + b)\ dx = \left(x + \frac{b}{a}\right)\ln(ax + b) - x, a \neq 0 \tag{7.48}$$

$$\int \ln(x^2 + a^2)\ dx = x\ln(x^2 + a^2) + 2a\tan^{-1}\frac{x}{a} - 2x \tag{7.49}$$

$$\int \ln(x^2 - a^2)\ dx = x\ln(x^2 - a^2) + a\ln\frac{x + a}{x - a} - 2x \tag{7.50}$$

$$\int \ln\left(ax^2 + bx + c\right)\ dx = \frac{1}{a}\sqrt{4ac - b^2}\tan^{-1}\frac{2ax + b}{\sqrt{4ac - b^2}} - 2x + \left(\frac{b}{2a} + x\right)\ln\left(ax^2 + bx + c\right) \tag{7.51}$$

$$\int x\ln(ax + b)\ dx = \frac{bx}{2a} - \frac{1}{4}x^2 + \frac{1}{2}\left(x^2 - \frac{b^2}{a^2}\right)\ln(ax + b) \tag{7.52}$$

$$\int x\ln\left(a^2 - b^2x^2\right)\ dx = -\frac{1}{2}x^2 + \frac{1}{2}\left(x^2 - \frac{a^2}{b^2}\right)\ln\left(a^2 - b^2x^2\right) \tag{7.53}$$

$$\int (\ln x)^2 \ dx = 2x - 2x \ln x + x(\ln x)^2 \tag{7.54}$$

$$\int (\ln x)^3 \ dx = -6x + x(\ln x)^3 - 3x(\ln x)^2 + 6x \ln x \tag{7.55}$$

$$\int x(\ln x)^2 \ dx = \frac{x^2}{4} + \frac{1}{2}x^2(\ln x)^2 - \frac{1}{2}x^2 \ln x \tag{7.56}$$

$$\int x^2(\ln x)^2 \ dx = \frac{2x^3}{27} + \frac{1}{3}x^3(\ln x)^2 - \frac{2}{9}x^3 \ln x \tag{7.57}$$

Integrals with Exponentials

$$\int e^{ax} \ dx = \frac{1}{a}e^{ax} \tag{7.58}$$

$$\int \sqrt{x}e^{ax} \ dx = \frac{1}{a}\sqrt{x}e^{ax} + \frac{i\sqrt{\pi}}{2a^{3/2}}\operatorname{erf}\left(i\sqrt{ax}\right), \text{ where } \operatorname{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2} dt \tag{7.59}$$

$$\int xe^x \ dx = (x-1)e^x \tag{7.60}$$

$$\int xe^{ax} \ dx = \left(\frac{x}{a} - \frac{1}{a^2}\right) e^{ax} \tag{7.61}$$

$$\int x^2 e^x \ dx = \left(x^2 - 2x + 2\right) e^x \tag{7.62}$$

$$\int x^2 e^{ax} \ dx = \left(\frac{x^2}{a} - \frac{2x}{a^2} + \frac{2}{a^3}\right) e^{ax} \tag{7.63}$$

$$\int x^3 e^x \ dx = \left(x^3 - 3x^2 + 6x - 6\right) e^x \tag{7.64}$$

$$\int x^n e^{ax} \ dx = \frac{x^n e^{ax}}{a} - \frac{n}{a}\int x^{n-1}e^{ax} \mathrm{d}x \tag{7.65}$$

$$\int x^n e^{ax} \ dx = \frac{(-1)^n}{a^{n+1}}\Gamma[1+n, -ax], \text{ where } \Gamma(a,x) = \int_x^\infty t^{a-1}e^{-t} \ \mathrm{d}t \tag{7.66}$$

$$\int e^{ax^2} \ dx = -\frac{i\sqrt{\pi}}{2\sqrt{a}}\operatorname{erf}\left(ix\sqrt{a}\right) \tag{7.67}$$

$$\int e^{-ax^2} \ dx = \frac{\sqrt{\pi}}{2\sqrt{a}}\operatorname{erf}\left(x\sqrt{a}\right) \tag{7.68}$$

$$\int xe^{-ax^2} \ dx = -\frac{1}{2a}e^{-ax^2} \tag{7.69}$$

$$\int x^2 e^{-ax^2} \ dx = \frac{1}{4}\sqrt{\frac{\pi}{a^3}}\operatorname{erf}(x\sqrt{a}) - \frac{x}{2a}e^{-ax^2} \tag{7.70}$$

Integrals with Trigonometric Functions

$$\int \sin ax \ dx = -\frac{1}{a}\cos ax \tag{7.71}$$

$$\int \sin^2 ax \ dx = \frac{x}{2} - \frac{\sin 2ax}{4a} \tag{7.72}$$

$$\int \sin^3 ax \ dx = -\frac{3\cos ax}{4a} + \frac{\cos 3ax}{12a} \tag{7.73}$$

$$\int \sin^n ax \ dx = -\frac{1}{a}\cos ax \ \ {}_2F_1\left[\frac{1}{2}, \frac{1-n}{2}, \frac{3}{2}, \cos^2 ax\right] \tag{7.74}$$

$$\int \cos ax \ dx = \frac{1}{a}\sin ax \tag{7.75}$$

$$\int \cos^2 ax \ dx = \frac{x}{2} + \frac{\sin 2ax}{4a} \tag{7.76}$$

$$\int \cos^3 ax dx = \frac{3\sin ax}{4a} + \frac{\sin 3ax}{12a} \tag{7.77}$$

$$\int \cos^p ax dx = -\frac{1}{a(1+p)}\cos^{1+p} ax \times {}_2F_1\left[\frac{1+p}{2}, \frac{1}{2}, \frac{3+p}{2}, \cos^2 ax\right] \tag{7.78}$$

$$\int \cos x \sin x \ dx = \frac{1}{2}\sin^2 x + c_1 = -\frac{1}{2}\cos^2 x + c_2 = -\frac{1}{4}\cos 2x + c_3 \tag{7.79}$$

$$\int \cos ax \sin bx \ dx = \frac{\cos[(a-b)x]}{2(a-b)} - \frac{\cos[(a+b)x]}{2(a+b)}, a \neq b \tag{7.80}$$

$$\int \sin^2 ax \cos bx \ dx = -\frac{\sin[(2a-b)x]}{4(2a-b)} + \frac{\sin bx}{2b} - \frac{\sin[(2a+b)x]}{4(2a+b)} \tag{7.81}$$

$$\int \sin^2 x \cos x \ dx = \frac{1}{3}\sin^3 x \tag{7.82}$$

$$\int \cos^2 ax \sin bx \; dx = \frac{\cos[(2a-b)x]}{4(2a-b)} - \frac{\cos bx}{2b} - \frac{\cos[(2a+b)x]}{4(2a+b)} \tag{7.83}$$

$$\int \cos^2 ax \sin ax \; dx = -\frac{1}{3a} \cos^3 ax \tag{7.84}$$

$$\int \sin^2 ax \cos^2 bx dx = \frac{x}{4} - \frac{\sin 2ax}{8a} - \frac{\sin[2(a-b)x]}{16(a-b)} + \frac{\sin 2bx}{8b} - \frac{\sin[2(a+b)x]}{16(a+b)} \tag{7.85}$$

$$\int \sin^2 ax \cos^2 ax \; dx = \frac{x}{8} - \frac{\sin 4ax}{32a} \tag{7.86}$$

$$\int \tan ax \; dx = -\frac{1}{a} \ln \cos ax \tag{7.87}$$

$$\int \tan^2 ax \; dx = -x + \frac{1}{a} \tan ax \tag{7.88}$$

$$\int \tan^n ax \; dx = \frac{\tan^{n+1} ax}{a(1+n)} \times {}_2F_1\left(\frac{n+1}{2}, 1, \frac{n+3}{2}, -\tan^2 ax\right) \tag{7.89}$$

$$\int \tan^3 ax dx = \frac{1}{a} \ln \cos ax + \frac{1}{2a} \sec^2 ax \tag{7.90}$$

$$\int \sec x \; dx = \ln|\sec x + \tan x| = 2 \tanh^{-1}\left(\tan \frac{x}{2}\right) \tag{7.91}$$

$$\int \sec^2 ax \; dx = \frac{1}{a} \tan ax \tag{7.92}$$

$$\int \sec^3 x \; dx = \frac{1}{2} \sec x \tan x + \frac{1}{2} \ln|\sec x + \tan x| \tag{7.93}$$

$$\int \sec x \tan x \; dx = \sec x \tag{7.94}$$

$$\int \sec^2 x \tan x \; dx = \frac{1}{2} \sec^2 x \tag{7.95}$$

$$\int \sec^n x \tan x \; dx = \frac{1}{n} \sec^n x, n \neq 0 \tag{7.96}$$

$$\int \csc x \; dx = \ln\left|\tan \frac{x}{2}\right| = \ln|\csc x - \cot x| + C \tag{7.97}$$

$$\int \csc^2 ax \; dx = -\frac{1}{a} \cot ax \tag{7.98}$$

$$\int \csc^3 x \; dx = -\frac{1}{2} \cot x \csc x + \frac{1}{2} \ln|\csc x - \cot x| \tag{7.99}$$

$$\int \csc^n x \cot x \; dx = -\frac{1}{n} \csc^n x, n \neq 0 \tag{7.100}$$

$$\int \sec x \csc x \; dx = \ln|\tan x| \tag{7.101}$$

Products of Trigonometric Functions and Monomials

$$\int x \cos x \; dx = \cos x + x \sin x \tag{7.102}$$

$$\int x \cos ax \; dx = \frac{1}{a^2} \cos ax + \frac{x}{a} \sin ax \tag{7.103}$$

$$\int x^2 \cos x \; dx = 2x \cos x + \left(x^2 - 2\right) \sin x \tag{7.104}$$

$$\int x^2 \cos ax \; dx = \frac{2x \cos ax}{a^2} + \frac{a^2x^2 - 2}{a^3} \sin ax \tag{7.105}$$

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

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

$$\int x \sin x \; dx = -x \cos x + \sin x \tag{7.108}$$

$$\int x \sin ax \; dx = -\frac{x \cos ax}{a} + \frac{\sin ax}{a^2} \tag{7.109}$$

$$\int x^2 \sin x \; dx = \left(2 - x^2\right) \cos x + 2x \sin x \tag{7.110}$$

$$\int x^2 \sin ax\ dx = \frac{2-a^2x^2}{a^3}\cos ax + \frac{2x\sin ax}{a^2} \tag{7.111}$$

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

$$\int x\cos^2 x\ dx = \frac{x^2}{4} + \frac{1}{8}\cos 2x + \frac{1}{4}x\sin 2x \tag{7.113}$$

$$\int x\sin^2 x\ dx = \frac{x^2}{4} - \frac{1}{8}\cos 2x - \frac{1}{4}x\sin 2x \tag{7.114}$$

$$\int x\tan^2 x\ dx = -\frac{x^2}{2} + \ln\cos x + x\tan x \tag{7.115}$$

$$\int x\sec^2 x\ dx = \ln\cos x + x\tan x \tag{7.116}$$

Products of Trigonometric Functions and Exponentials

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

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

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

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

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

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

Integrals of Hyperbolic Functions

$$\int \cosh ax\ dx = \frac{1}{a}\sinh ax \tag{7.123}$$

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

$$\int \sinh ax\ dx = \frac{1}{a}\cosh ax \tag{7.125}$$

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

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

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

$$\int \cos ax\cosh bx\ dx = \frac{1}{a^2+b^2}\left[a\sin ax\cosh bx + b\cos ax\sinh bx\right] \tag{7.129}$$

$$\int \cos ax\sinh bx\ dx = \frac{1}{a^2+b^2}\left[b\cos ax\cosh bx + a\sin ax\sinh bx\right] \tag{7.130}$$

$$\int \sin ax\cosh bx\ dx = \frac{1}{a^2+b^2}\left[-a\cos ax\cosh bx + b\sin ax\sinh bx\right] \tag{7.131}$$

$$\int \sin ax\sinh bx\ dx = \frac{1}{a^2+b^2}\left[b\cosh bx\sin ax - a\cos ax\sinh bx\right] \tag{7.132}$$

$$\int \sinh ax\cosh ax dx = \frac{1}{4a}\left[-2ax + \sinh 2ax\right] \tag{7.133}$$

$$\int \sinh ax\cosh bx\ dx = \frac{1}{b^2-a^2}\left[b\cosh bx\sinh ax - a\cosh ax\sinh bx\right] \tag{7.134}$$

Problem	Status	Comment	Iurii	Alex	Igor
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					