Содержание

1	final/template/vimrc.txt	1
2	final/template/template.cpp	1
3	Practice round	2
4	final/template/fastIO.cpp	2
5	final/template/hashTable.cpp	2
6	final/template/optimizations.cpp	2
7	final/template/useful.cpp	2
8	final/template/Template.java	3
9	final/numeric/fft.cpp	3
10	final/numeric/fftint.cpp	4
11	final/numeric/blackbox.cpp	4
12	final/numeric/crt.cpp	5
13	final/numeric/mulMod.cpp	5
14	final/numeric/modReverse.cpp	5
15	final/numeric/pollard.cpp	5
16	final/numeric/poly.cpp	5
17	final/numeric/simplex.cpp	6
18	final/numeric/sumLine.cpp	7
19	final/numeric/berlekamp.cpp	7
20	final/numeric/integrate.cpp	7
21	final/geom/commonTangents.cpp	8
22	final/geom/halfplaneIntersection.cpp	8
23	final/geom/minDisc.cpp	8
24	final/geom/convexHull3D-N2.cpp	9
25	final/geom/polygonArcCut.cpp	9
26	final/geom/polygonTangent.cpp	9
27	final/strings/eertree.cpp	9
28	final/strings/sufAutomaton.cpp	10
29	final/strings/duval.cpp	10
30	final/graphs/centroid.cpp	11
31	final/graphs/dominatorTree.cpp	11
32	final/graphs/generalMatching.cpp	11
33	final/graphs/heavyLight.cpp	12
34	final/graphs/hungary.cpp	12
35	final/graphs/minCostNegCycle.cpp	13
36	final/graphs/retro.cpp	13
37	final/graphs/smith.cpp	14
38	final/graphs/mincut.cpp	14
39	final/graphs/twoChineseFast.cpp	14
40	final/graphs/linkcut.cpp	15
41	final/graphs/chordaltree.cpp	16
42	final/graphs/minimization.cpp	16
43	final/graphs/matroidIntersection.cpp	17

final/template/vimrc.txt

```
map <F9> :wall! <CR> :!g++ -Wall -Wextra -Wshadow -Wno-unused-result -o %:r % -std=c++14 -DHOME -D_GLIBCXX_DEBUG -fsanitize=address <CR>
map <F7> :wall! <CR> :!g++ -Wall -Wextra -Wshadow -Wno-unused-result -o %:r % -std=c++14 -DHOME -O2 <CR>
map <F8> :wall! <CR> :!ulimit -s 500000 && ./%:r <CR>
map <F10> :wall! <CR> :!g++ -Wall -Wextra -Wshadow -Wno-unused-result -o %:r % -std=c++14 -DHOME -D_GLIBCXX_DEBUG -fsanitize=address -g && gdb ./%:r <CR>

inoremap {<CR> {<CR>}<ESC>O
map <c-a> ggVG

set nu
set rnu
syntax on

map <c-t> :tabnew <CR>
map <c-l> :tabn <CR>
map <c-h> :tabp <CR>

set cin
set sw=4
set so=99
set bs=2
set et
set sts=4
```

final/template/template.cpp

```cpp
// team : SPb ITMO University 1
#include <bits/stdc++.h>

#define F first
#define S second
#define X first
#define Y second
#define pb push_back
#define sz(a) (int)(a).size()
#define all(a) (a).begin(),a.end()
#define pw(x) (1LL<<(x))

#define db(x) cerr << #x << " = " << x << endl
#define db2(x, y) cerr << "(" << #x << ", " << #y << ") = (" << x << ", " << y << ")\n";
#define db3(x, y, z) cerr << "(" << #x << ", " << #y << ", " << #z << ") = (" << x << ", " << y << ", " << z << ")\n"
#define dbv(a) cerr << #a << " = "; for (auto xxxx: a) cerr << xxxx << " "; cerr << endl

using namespace std;

typedef long long ll;
typedef double dbl;
const int INF = 1.01e9;



int main() {
#define TASK ""
#ifdef HOME
  assert(freopen(TASK".in", "r", stdin));
#endif




#ifdef HOME
  cerr << "time: " << clock() * 1.0 / CLOCKS_PER_SEC << endl;
#endif
  return 0;
}
```

Practice round

1. Посабмитить задачи каждому человеку
2. Печать
3. IDE для джавы
4. Сравнить скорость локального компьютера и сервера
5. Проверить __int128
6. Проверить прагмы (например на битсетах)
7. Узнать максимально возможный размер отправляемого кода

final/template/fastIO.cpp

```
#include <cstdio>
#include <algorithm>

/** Interface */

inline int readInt();
inline int readUInt();
inline bool isEof();

/** Read */

static const int buf_size = 100000;
static char buf[buf_size];
static int buf_len = 0, pos = 0;

inline bool isEof() {
  if (pos == buf_len) {
    pos = 0, buf_len = fread(buf, 1, buf_size, stdin);
    if (pos == buf_len) return 1;
  }
  return 0;
}

inline int getChar() { return isEof() ? -1 : buf[pos++]; }

inline int readChar() {
  int c = getChar();
  while (c != -1 && c <= 32) c = getChar();
  return c;
}

inline int readUInt() {
  int c = readChar(), x = 0;
  while ('0' <= c && c <= '9') x = x * 10 + c - '0', c = getChar();
  return x;
}

inline int readInt() {
  int s = 1, c = readChar();
  int 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;
}


// 10M int [0..1e9)
// cin 3.02
// scanf 1.2
// cin sync_with_stdio(false) 0.71
// fastRead getchar 0.53
// fastRead fread 0.15
```

final/template/hashTable.cpp

```
template <const int max_size, class HashType, class Data, const Data default_value>
struct hashTable {
  HashType hash[max_size];
  Data f[max_size];
  int size;

  int position(HashType H) const {
    int i = H % max_size;
    while (hash[i] && hash[i] != H)
      if (++i == max_size)
        i = 0;
    return i;
  }

  Data & operator [] (HashType H) {
    assert(H != 0);
    int i = position(H);
    if (!hash[i]) {
      hash[i] = H;
      f[i] = default_value;
      size++;
    }
    return f[i];
  }
};

hashTable<13, int, int, 0> h;
```

final/template/optimizations.cpp

```
inline void fasterLLDivMod(unsigned long long x, unsigned y, unsigned &out_d, unsigned &out_m) {
  unsigned xh = (unsigned)(x >> 32), xl = (unsigned)x, d, m;
#ifdef __GNUC__
  asm(
    "divl %4; \n\t"
    : "=a" (d), "=d" (m)
    : "d" (xh), "a" (xl), "r" (y)
  );
#else
  __asm {
    mov edx, dword ptr[xh];
    mov eax, dword ptr[xl];
    div dword ptr[y];
    mov dword ptr[d], eax;
    mov dword ptr[m], edx;
  };
#endif
  out_d = d; out_m = m;
}

// have no idea what sse flags are really cool; list of some of them
// -- very good with bitsets
#pragma GCC optimize("O3")
#pragma GCC target("sse,sse2,sse3,ssse3,sse4,popcnt,abm,mmx")
```

final/template/useful.cpp

```
#include "ext/pb_ds/assoc_container.hpp"
using namespace __gnu_pbds;

template <typename T> using ordered_set = tree<T, null_type, less<T>, rb_tree_tag, tree_order_statistics_node_update>;
template <typename K, typename V> using ordered_map = tree<K, V, less<K>, rb_tree_tag, tree_order_statistics_node_update>;

// HOW TO USE ::
// -- order_of_key(10) returns the number of elements in set/map strictly less than 10
// -- *find_by_order(10) returns 10-th smallest element in set/map (0-based)

bitset<N> a;
for (int i = a._Find_first(); i != a.size(); i = a._Find_next(i)) {
```

```
  cout << i << endl;
}
```

final/template/Template.java

```java
import java.util.*;
import java.io.*;

public class Template {
  FastScanner in;
  PrintWriter out;

  public void solve() throws IOException {
    int n = in.nextInt();
    out.println(n);
  }

  public void run() {
    try {
      in = new FastScanner();
      out = new PrintWriter(System.out);

      solve();

      out.close();
    } catch (IOException e) {
      e.printStackTrace();
    }
  }

  class FastScanner {
    BufferedReader br;
    StringTokenizer st;

    FastScanner() {
      br = new BufferedReader(new InputStreamReader(↩
System.in));
    }

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

    int nextInt() {
      return Integer.parseInt(next());
    }
  }

  public static void main(String[] arg) {
    new Template().run();
  }
}
```

final/numeric/fft.cpp

```cpp
namespace fft
{
  const int maxBase = 21;
  const int maxN = 1 << maxBase;

  struct num
  {
    dbl x, y;
    num(){}
    num(dbl xx, dbl yy): x(xx), y(yy) {}
    num(dbl alp): x(cos(alp)), y(sin(alp)) {}
  };

  inline num operator + (num a, num b) { return num(↩
    a.x + b.x, a.y + b.y); }
  inline num operator - (num a, num b) { return num(↩
    a.x - b.x, a.y - b.y); }
  inline num operator * (num a, num b) { return num(↩
    a.x * b.x - a.y * b.y, a.x * b.y + a.y * b.x); ↩
    }
  inline num conj(num a) { return num(a.x, -a.y); }

  const dbl PI = acos(-1);

  num root[maxN];
  int rev[maxN];
  bool rootsPrepared = false;

  void prepRoots()
  {
    if (rootsPrepared) return;
    rootsPrepared = true;
    root[1] = num(1, 0);
    for (int k = 1; k < maxBase; ++k)
    {
      num x(2 * PI / pw(k + 1));
      for (int i = pw(k - 1); i < pw(k); ++i)
      {
        root[2 * i] = root[i];
        root[2 * i + 1] = root[i] * x;
      }
    }
  }

  int base, N;

  int lastRevN = -1;
  void prepRev()
  {
    if (lastRevN == N) return;
    lastRevN = N;
    forn(i, N) rev[i] = (rev[i >> 1] >> 1) + ((i & ↩
    1) << (base - 1));
  }

  void fft(num *a, num *f)
  {
    forn(i, N) f[i] = a[rev[i]];
    for (int k = 1; k < N; k <<= 1) for (int i = 0; ↩
    i < N; i += 2 * k) forn(j, k)
    {
      num z = f[i + j + k] * root[j + k];
      f[i + j + k] = f[i + j] - z;
      f[i + j] = f[i + j] + z;
    }
  }

  num a[maxN], b[maxN], f[maxN], g[maxN];
  ll A[maxN], B[maxN], C[maxN];

  void _multMod(int mod)
  {
    forn(i, N)
    {
      int x = A[i] % mod;
      a[i] = num(x & (pw(15) - 1), x >> 15);
    }
    forn(i, N)
    {
      int x = B[i] % mod;
      b[i] = num(x & (pw(15) - 1), x >> 15);
    }
    fft(a, f);
    fft(b, g);

    forn(i, N)
    {
      int j = (N - i) & (N - 1);
```

```
      num a1 = (f[i] + conj(f[j])) * num(0.5, 0);
      num a2 = (f[i] - conj(f[j])) * num(0, -0.5);
      num b1 = (g[i] + conj(g[j])) * num(0.5 / N, 0);
      num b2 = (g[i] - conj(g[j])) * num(0, -0.5 / N);
      a[j] = a1 * b1 + a2 * b2 * num(0, 1);
      b[j] = a1 * b2 + a2 * b1;
    }

    fft(a, f);
    fft(b, g);

    forn(i, N)
    {
      ll aa = f[i].x + 0.5;
      ll bb = g[i].x + 0.5;
      ll cc = f[i].y + 0.5;
      C[i] = (aa + bb % mod * pw(15) + cc % mod * pw(30)) % mod;
    }
  }

  void prepAB(int n1, int n2)
  {
    base = 1;
    N = 2;
    while (N < n1 + n2) base++, N <<= 1;

    for (int i = n1; i < N; ++i) A[i] = 0;
    for (int i = n2; i < N; ++i) B[i] = 0;

    prepRoots();
    prepRev();
  }

  void mult(int n1, int n2)
  {
    prepAB(n1, n2);
    forn(i, N) a[i] = num(A[i], B[i]);
    fft(a, f);
    forn(i, N)
    {
      int j = (N - i) & (N - 1);
      a[i] = (f[j] * f[j] - conj(f[i] * f[i])) * num(0, -0.25 / N);
    }
    fft(a, f);
    forn(i, N) C[i] = (ll)round(f[i].x);
  }


  void multMod(int n1, int n2, int mod)
  {
    prepAB(n1, n2);
    _multMod(mod);
  }

  int D[maxN];

  void multLL(int n1, int n2)
  {
    prepAB(n1, n2);

    int mod1 = 1.5e9;
    int mod2 = mod1 + 1;

    _multMod(mod1);

    forn(i, N) D[i] = C[i];

    _multMod(mod2);

    forn(i, N)
    {
      C[i] = D[i] + (C[i] - D[i] + (ll)mod2) * (ll)mod1 % mod2 * mod1;
    }
  }
  // HOW TO USE ::
  // -- set correct maxBase
  // -- use mult(n1, n2), multMod(n1, n2, mod) and multLL(n1, n2)
  // -- input : A[], B[]
  // -- output : C[]
}
```

final/numeric/fftint.cpp

```
namespace fft {
  const int MOD = 998244353;
  const int base = 20;
  const int N = 1 << base;
  const int ROOT = 646;

  int root[N];
  int rev[N];

  ll inv(ll a, ll m = MOD) {
    if (a == 0) return 0;
    return ((1 - inv(m % a, a) * m) / a + m) % m;
  }

  void init() {
    for (int i = 0; i < N; i++) rev[i] = (rev[i >> 1] >> 1) + ((i & 1) << (base - 1));
    int NN = N >> 1;
    int z = 1;
    for (int i = 0; i < NN; i++) {
      root[i + NN] = z;
      z = z * (ll)ROOT % MOD;
    }
    for (int i = NN - 1; i > 0; --i) root[i] = root[2 * i];
  }

  void fft(int *a, int *f) {
    for (int i = 0; i < N; i++) f[i] = a[rev[i]];
    for (int k = 1; k < N; k <<= 1) {
      for (int i = 0; i < N; i += 2 * k) {
        for (int j = 0; j < k; j++) {
          int z = f[i + j + k] * (ll)root[j + k] % MOD;
          f[i + j + k] = (f[i + j] - z + MOD) % MOD;
          f[i + j] = (f[i + j] + z) % MOD;
        }
      }
    }
  }

  int A[N], B[N], C[N];
  int F[N], G[N];

  void _mult(int eq) {
    fft(A, F);
    if (eq)
      for (int i = 0; i < N; i++)
        G[i] = F[i];
    else fft(B, G);
    int invN = inv(N);
    for (int i = 0; i < N; i++) A[i] = F[i] * (ll)G[i] % MOD * invN % MOD;
    reverse(A + 1, A + N);
    fft(A, C);
  }

  void mult(int n1, int n2, int eq = 0) {
    for (int i = n1; i < N; ++i) A[i] = 0;
    for (int i = n2; i < N; ++i) B[i] = 0;

    _mult(eq);

    //forn(i, n1 + n2) C[i] = 0;
    //forn(i, n1) forn(j, n2) C[i + j] = (C[i + j] + A[i] * (ll)B[j]) % mod;
  }
}
```

final/numeric/blackbox.cpp

```
namespace blackbox
{
  int A[N];
  int B[N];
  int C[N];

  int magic(int k, int x)
  {
    B[k] = x;
    C[k] = (C[k] + A[0] * (ll)B[k]) % mod;
    int z = 1;
    if (k == N - 1) return C[k];
    while ((k & (z - 1)) == (z - 1))
    {
      //mult B[k - z + 1 ... k] x A[z .. 2 * z - 1]
      forn(i, z) fft::A[i] = A[z + i];
```

```cpp
      forn(i, z) fft::B[i] = B[k - z + 1 + i];
      fft::multMod(z, z, mod);
      forn(i, 2 * z - 1) C[k + 1 + i] = (C[k + 1 + i] + fft::C[i]) % mod;
      z <<= 1;
    }
    return C[k];
  }
  // A -- constant array
  // magic(k, x):: B[k] = x, returns C[k]
  // !! WARNING !! better to set N twice the size needed
}
```

final/numeric/crt.cpp

```cpp
int CRT(int a1, int m1, int a2, int m2) {
  return (a1 - a2 % m1 + m1) * (ll)rev(m2, m1) % m1 * m2 + a2;
}
```

final/numeric/mulMod.cpp

```cpp
ll mul( ll a, ll b, ll m ) { // works for MOD 8e18
  ll k = (ll)((long double)a * b / m);
  ll r = a * b - m * k;
  if (r < 0) r += m;
  if (r >= m) r -= m;
  return r;
}
```

final/numeric/modReverse.cpp

```cpp
int rev(int x, int m) {
  if (x == 1) return 1;
  return (1 - rev(m % x, x) * (ll)m) / x + m;
}
```

final/numeric/pollard.cpp

```cpp
namespace pollard
{
  using math::p;

  vector<pair<ll, int>> getFactors(ll N)
  {
    vector<ll> primes;

    const int MX = 1e5;
    const ll MX2 = MX * (ll)MX;

    assert(MX <= math::maxP && math::pc > 0);

    function<void(ll)> go = [&go, &primes](ll n)
    {
      for (ll x : primes) while (n % x == 0) n /= x;
      if (n == 1) return;
      if (n > MX2)
      {
        auto F = [&](ll x) {
          ll k = ((long double)x * x) / n;
          ll r = (x * x - k * n + 3) % n;
          return r < 0 ? r + n : r;
        };
        ll x = mt19937_64()() % n, y = x;
        const int C = 3 * pow(n, 0.25);

        ll val = 1;
        forn(it, C)
        {
          x = F(x), y = F(F(y));
          if (x == y) continue;
          ll delta = abs(x - y);
          ll k = ((long double)val * delta) / n;
          val = (val * delta - k * n) % n;
          if (val < 0) val += n;
          if (val == 0)
          {
            ll g = __gcd(delta, n);
            go(g), go(n / g);
            return;
          }
          if ((it & 255) == 0)
          {
            ll g = __gcd(val, n);
            if (g != 1)
            {
              go(g), go(n / g);
              return;
            }
          }
        }
      }
      primes.pb(n);
    };

    ll n = N;

    for (int i = 0; i < math::pc && p[i] < MX; ++i) if (n % p[i] == 0)
    {
      primes.pb(p[i]);
      while (n % p[i] == 0) n /= p[i];
    }

    go(n);

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

    vector<pair<ll, int>> res;
    for (ll x : primes)
    {
      int cnt = 0;
      while (N % x == 0)
      {
        cnt++;
        N /= x;
      }
      res.push_back({x, cnt});
    }
    return res;
  }
}
```

final/numeric/poly.cpp

```cpp
struct poly
{
  vi v;
  poly() {}
  poly(vi vv)
  {
    v = vv;
  }
  int size()
  {
    return (int)v.size();
  }
  poly cut(int maxLen)
  {
    if (maxLen < sz(v)) v.resize(maxLen);
    return *this;
  }
  poly norm()
  {
    while (sz(v) > 1 && v.back() == 0) v.pop_back();
    return *this;
  }
  inline int& operator [] (int i)
  {
    return v[i];
  }
  void out(string name="")
  {
    stringstream ss;
    if (sz(name)) ss << name << "=";
```

```cpp
    int fst = 1;
    forn(i, sz(v)) if (v[i])
    {
      int x = v[i];
      int sgn = 1;
      if (x > mod / 2) x = mod-x, sgn = -1;
      if (sgn == -1) ss << "-";
      else if (!fst) ss << "+";
      fst = 0;
      if (!i || x != 1)
      {
        ss << x;
        if (i > 0) ss << "*x";
        if (i > 1) ss << "^" << i;
      }
      else
      {
        ss << "x";
        if (i > 1) ss << "^" << i;
      }
    }
    if (fst) ss <<"0";
    string s;
    ss >> s;
    eprintf("%s\n", s.data());
  }
};

poly operator + (poly A, poly B)
{
  poly C;
  C.v = vi(max(sz(A), sz(B)));
  forn(i, sz(C))
  {
    if (i < sz(A)) C[i] = (C[i] + A[i]) % mod;
    if (i < sz(B)) C[i] = (C[i] + B[i]) % mod;
  }
  return C.norm();
}

poly operator - (poly A, poly B)
{
  poly C;
  C.v = vi(max(sz(A), sz(B)));
  forn(i, sz(C))
  {
    if (i < sz(A)) C[i] = (C[i] + A[i]) % mod;
    if (i < sz(B)) C[i] = (C[i] + mod - B[i]) % mod;
  }
  return C.norm();
}

poly operator * (poly A, poly B)
{
  poly C;
  C.v = vi(sz(A) + sz(B) - 1);

  forn(i, sz(A)) fft::A[i] = A[i];
  forn(i, sz(B)) fft::B[i] = B[i];
  fft::multMod(sz(A), sz(B), mod);
  forn(i, sz(C)) C[i] = fft::C[i];
  return C.norm();
}

poly inv(poly A, int n) // returns A^-1 mod x^n
{
  assert(sz(A) && A[0] != 0);
  A.cut(n);

  auto cutPoly = [](poly &from, int l, int r)
  {
    poly R;
    R.v.resize(r - l);
    for (int i = l; i < r; ++i)
    {
      if (i < sz(from)) R[i - l] = from[i];
    }
    return R;
  };

  function<int(int, int)> rev = [&rev](int x, int m)->int
  {
    if (x == 1) return 1;
    return (1 - rev(m % x, x) * (ll)m) / x + m;
  };

  poly R({rev(A[0], mod)});
  for (int k = 1; k < n; k <<= 1)
  {
    poly A0 = cutPoly(A, 0, k);
    poly A1 = cutPoly(A, k, 2 * k);
    poly H = A0 * R;
    H = cutPoly(H, k, 2 * k);
    poly R1 = (((A1 * R).cut(k) + H) * (poly({0}) - R)).cut(k);
    R.v.resize(2 * k);
    forn(i, k) R[i + k] = R1[i];
  }
  return R.cut(n).norm();
}

pair<poly, poly> divide(poly A, poly B)
{
  if (sz(A) < sz(B)) return {poly({0}), A};

  auto rev = [](poly f)
  {
    reverse(all(f.v));
    return f;
  };

  poly q = rev((inv(rev(B), sz(A) - sz(B) + 1) * rev(A)).cut(sz(A) - sz(B) + 1));
  poly r = A - B * q;

  return {q, r};
}
```

final/numeric/simplex.cpp

```cpp
vector<double> simplex(vector<vector<double> > a) {
  int n = a.size() - 1;
  int m = a[0].size() - 1;
  vector<int> left(n + 1), up(m + 1);
  iota(up.begin(), up.end(), 0);
  iota(left.begin(), left.end(), m);
  auto pivot = [&](int x, int y) {
    swap(left[x], up[y]);
    double k = a[x][y];
    a[x][y] = 1;
    vector<int> vct;
    for (int j = 0; j <= m; j++) {
      a[x][j] /= k;
      if (!eq(a[x][j], 0)) vct.push_back(j);
    }
    for (int i = 0; i <= n; i++) {
      if (eq(a[i][y], 0) || i == x) continue;
      k = a[i][y];
      a[i][y] = 0;
      for (int j : vct) a[i][j] -= k * a[x][j];
    }
  };
  while (1) {
    int x = -1;
    for (int i = 1; i <= n; i++) if (ls(a[i][0], 0) && (x == -1 || a[i][0] < a[x][0])) x = i;
    if (x == -1) break;
    int y = -1;
    for (int j = 1; j <= m; j++) if (ls(a[x][j], 0) && (y == -1 || a[x][j] < a[x][y])) y = j;
    if (y == -1) assert(0); // infeasible
    pivot(x, y);
  }
  while (1) {
    int y = -1;
    for (int j = 1; j <= m; j++) if (ls(0, a[0][j]) && (y == -1 || a[0][j] > a[0][y])) y = j;
    if (y == -1) break;
    int x = -1;
    for (int i = 1; i <= n; i++) if (ls(0, a[i][y]) && (x == -1 || a[i][0] / a[i][y] < a[x][0] / a[x][y])) x = i;
    if (x == -1) assert(0); // unbounded
    pivot(x, y);
  }
  vector<double> ans(m + 1);
  for (int i = 1; i <= n; i++) if (left[i] <= m) ans[left[i]] = a[i][0];
  ans[0] = -a[0][0];
  return ans;
}
// j=1..m: x[j]>=0
// i=1..n: sum(j=1..m) A[i][j]*x[j] <= A[i][0]
// max sum(j=1..m) A[0][j]*x[j]
// res[0] is answer
// res[1..m] is certificate
```

final/numeric/sumLine.cpp

```
// sum(i=0..n-1) (a+b*i) div m
ll solve(ll n, ll a, ll b, ll m) {
  if (b == 0) return n * (a / m);
  if (a >= m) return n * (a / m) + solve(n, a % m, b, m);
  if (b >= m) return n * (n - 1) / 2 * (b / m) + solve(n, a, b % m, m);
  return solve((a + b * n) / m, (a + b * n) % m, m, b);
}
```

final/numeric/berlekamp.cpp

```
vector<int> berlekamp(vector<int> s) {
  int l = 0;
  vector<int> la(1, 1);
  vector<int> b(1, 1);
  for (int r = 1; r <= (int)s.size(); r++) {
    int delta = 0;
    for (int j = 0; j <= l; j++) {
      delta = (delta + 1LL * s[r - 1 - j] * la[j]) % MOD;
    }
    b.insert(b.begin(), 0);
    if (delta != 0) {
      vector<int> t(max(la.size(), b.size()));
      for (int i = 0; i < (int)t.size(); i++) {
        if (i < (int)la.size()) t[i] = (t[i] + la[i]) % MOD;
        if (i < (int)b.size()) t[i] = (t[i] - 1LL * delta * b[i] % MOD + MOD) % MOD;
      }
      if (2 * l <= r - 1) {
        b = la;
        int od = inv(delta);
        for (int &x : b) x = 1LL * x * od % MOD;
        l = r - l;
      }
      la = t;
    }
  }
  assert((int)la.size() == l + 1);
  assert(l * 2 + 30 < (int)s.size());
  reverse(la.begin(), la.end());
  return la;
}

vector<int> mul(vector<int> a, vector<int> b) {
  vector<int> c(a.size() + b.size() - 1);
  for (int i = 0; i < (int)a.size(); i++) {
    for (int j = 0; j < (int)b.size(); j++) {
      c[i + j] = (c[i + j] + 1LL * a[i] * b[j]) % MOD;
    }
  }
  vector<int> res(c.size());
  for (int i = 0; i < (int)res.size(); i++) res[i] = c[i] % MOD;
  return res;
}

vector<int> mod(vector<int> a, vector<int> b) {
  if (a.size() < b.size()) a.resize(b.size() - 1);

  int o = inv(b.back());
  for (int i = (int)a.size() - 1; i >= (int)b.size() - 1; i--) {
    if (a[i] == 0) continue;
    int coef = 1LL * o * (MOD - a[i]) % MOD;
    for (int j = 0; j < (int)b.size(); j++) {
      a[i - (int)b.size() + 1 + j] = (a[i - (int)b.size() + 1 + j] + 1LL * coef * b[j]) % MOD;
    }
  }
  while (a.size() >= b.size()) {
    assert(a.back() == 0);
    a.pop_back();
  }
  return a;
}

vector<int> bin(int n, vector<int> p) {
  vector<int> res(1, 1);
  vector<int> a(2); a[1] = 1;
  while (n) {
    if (n & 1) res = mod(mul(res, a), p);
    a = mod(mul(a, a), p);
    n >>= 1;
  }
  return res;
}

int f(vector<int> t, int m) {
  vector<int> v = berlekamp(t);
  vector<int> o = bin(m - 1, v);
  int res = 0;
  for (int i = 0; i < (int)o.size(); i++) res = (res + 1LL * o[i] * t[i]) % MOD;
  return res;
}
```

final/numeric/integrate.cpp

```
function<dbl(dbl, dbl, function<dbl(dbl)>)> f = [&](dbl L, dbl R, function<dbl(dbl)> g) {
  const int ITERS = 1000000;
  dbl ans = 0;
  dbl step = (R - L) * 1.0 / ITERS;
  for (int it = 0; it < ITERS; it++) {
    double xl = L + step * it;
    double xr = L + step * (it + 1);
    dbl x1 = (xl + xr) / 2;
    dbl x0 = x1 - (x1 - xl) * sqrt(3.0 / 5);
    dbl x2 = x1 + (x1 - xl) * sqrt(3.0 / 5);
    ans += (5 * g(x0) + 8 * g(x1) + 5 * g(x2)) / 18 * step;
  }
  return ans;
};
```

final/geom/commonTangents.cpp

```cpp


vector<Line> commonTangents(pt A, dbl rA, pt B, dbl rB) {
  vector<Line> res;
  pt C = B - A;
  dbl z = C.len2();
  for (int i = -1; i <= 1; i += 2) {
    for (int j = -1; j <= 1; j += 2) {
      dbl r = rB * j - rA * i;
      dbl d = z - r * r;
      if (ls(d, 0)) continue;
      d = sqrt(max(0.01, d));
      pt magic = pt(r, d) / z;
      pt v(magic % C, magic * C);
      dbl CC = (rA * i - v % A) / v.len2();
      pt O = v * -CC;
      res.pb(Line(O, O + v.rotate()));
    }
  }
  return res;
}

// HOW TO USE ::
// --    *D*-----------------*F*
// --    *...* -          - *...*
// --   *.....* -      - *.....*
// --   *.......* -  - *.......*
// --   *...A...*  --  *...B...*
// --   *.......* -  - *.......*
// --   *.....* -      - *.....*
// --    *...* -          - *...*
// --    *C*-----------------*E*
// -- res = {CE, CF, DE, DF}
```

final/geom/halfplaneIntersection.cpp

```cpp
int getPart(pt v) {
  return less(0, v.y) || (equal(0, v.y) && less(v.x, 0));
}

int cmpV(pt a, pt b) {
  int partA = getPart(a);
  int partB = getPart(b);
  if (partA < partB) return -1;
  if (partA > partB) return 1;
  if (equal(0, a * b)) return 0;
  if (0 < a * b) return -1;
  return 1;
}

double planeInt(vector<Line> l) {
  int n = l.size();
  sort(all(l), [](Line a, Line b) {
      int r = cmpV(a.v, b.v);
      if (r != 0) return r < 0;
      return a.0 % a.v.rotate() < b.0 % a.v.rotate();
    });

  int cur = 0;
  for (int i = 0; i < n; ) {
    int j = i;
    for (; i < n && cmpV(l[j].v, l[i].v) == 0 && cmpV(l[i].v, l[j].v) == 0; i++);
    l[cur++] = l[i - 1];
  }
  n = cur;

  for (int i = 0; i < n; i++)
    l[i].id = i;

  int flagUp = 0;
  int flagDown = 0;
  for (int i = 0; i < n; i++) {
    int part = getPart(l[i].v);
    if (part == 1) flagUp = 1;
    if (part == 0) flagDown = 1;
  }
  if (!flagUp || !flagDown) return -1;

  for (int i = 0; i < n; i++) {
    pt v = l[i].v;
    pt u = l[(i + 1) % n].v;
    if (equal(0, v * u) && less(v % u, 0)) {
      pt dir = l[i].v.rotate();
      if (lessE(l[(i + 1) % n].0 % dir, l[i].0 % dir)) return 0;
      return -1;
    }
    if (less(v * u, 0))
      return -1;
  }

  cur = 0;
  vector<Line> st(n * 2);
  for (int tt = 0; tt < 2; tt++) {
    for (int i = 0; i < n; i++) {
      for (; cur >= 2; cur--) {
        pt G = st[cur - 1] * l[i];
        if (!lessE(st[cur - 2].v * (G - st[cur - 2].0), 0))
          break;
      }
      st[cur++] = l[i];
      if (cur >= 2 && lessE(st[cur - 2].v * st[cur - 1].v, 0)) return 0;
    }
  }
  vector < int > use(n, -1);
  int left = -1, right = -1;
  for (int i = 0; i < cur; i++) {
    if (use[st[i].id] == -1) {
      use[st[i].id] = i;
    }
    else {
      left = use[st[i].id];
      right = i;
      break;
    }
  }
  vector < Line > tmp;
  for (int i = left; i < right; i++)
    tmp.pb(st[i]);
  vector < pt > res;
  for (int i = 0; i < (int)tmp.size(); i++)
    res.pb(tmp[i] * tmp[(i + 1) % tmp.size()]);
  double area = 0;
  for (int i = 0; i < (int)res.size(); i++)
    area += res[i] * res[(i + 1) % res.size()];
  return area / 2;
}
```

final/geom/minDisc.cpp

```cpp

pair<pt, dbl> minDisc(vector<pt> p) {
  int n = p.size();
  pt O = pt(0, 0);
  dbl R = 0;
  random_shuffle(all(p));
  for (int i = 0; i < n; i++) {
    if (ls(R, (O - p[i]).len())) {
      O = p[i];
      R = 0;
      for (int j = 0; j < i; j++) {
        if (ls(R, (O - p[j]).len())) {
          O = (p[i] + p[j]) / 2;
          R = (p[i] - p[j]).len() / 2;
          for (int k = 0; k < j; k++) {
            if (ls(R, (O - p[k]).len())) {
              Line l1((p[i] + p[j]) / 2, (p[i] + p[j]) / 2 + (p[i] - p[j]).rotate());
              Line l2((p[k] + p[j]) / 2, (p[k] + p[j]) / 2 + (p[k] - p[j]).rotate());
              O = l1 * l2;
              R = (p[i] - O).len();
            }
          }
        }
      }
    }
  }
  return {O, R};
}
```

final/geom/convexHull3D-N2.cpp

```
struct Plane {
  pt O, v;
  vector<int> id;
};

vector<Plane> convexHull3(vector<pt> p) {
  vector<Plane> res;
  int n = p.size();
  for (int i = 0; i < n; i++)
    p[i].id = i;
  for (int i = 0; i < 4; i++) {
    vector<pt> tmp;
    for (int j = 0; j < 4; j++)
      if (i != j)
        tmp.pb(p[j]);
    res.pb({tmp[0], (tmp[1] - tmp[0]) * (tmp[2] - tmp[0]), {tmp[0].id, tmp[1].id, tmp[2].id}});
    if ((p[i] - res.back().O) % res.back().v > 0) {
      res.back().v = res.back().v * -1;
      swap(res.back().id[0], res.back().id[1]);
    }
  }
  vector<vector<int>> use(n, vector<int>(n, 0));
  int tmr = 0;
  for (int i = 4; i < n; i++) {
    int cur = 0;
    tmr++;
    vector<pair<int,int>> curEdge;
    for (int j = 0; j < sz(res); j++) {
      if ((p[i] - res[j].O) % res[j].v > 0) {
        for (int t = 0; t < 3; t++) {
          int v = res[j].id[t];
          int u = res[j].id[(t + 1) % 3];
          use[v][u] = tmr;
          curEdge.pb({v, u});
        }
      }
      else {
        res[cur++] = res[j];
      }
    }
    res.resize(cur);
    for (auto x: curEdge) {
      if (use[x.S][x.F] == tmr) continue;
      res.pb({p[i], (p[x.F] - p[i]) * (p[x.S] - p[i]), {x.F, x.S, i}});
    }
  }
  return res;
}

// plane in 3d
//(A, v) * (B, u) -> (O, n)

pt n = v * u;
pt m = v * n;
double t = (B - A) % u / (u % m);
pt O = A - m * t;
```

final/geom/polygonArcCut.cpp

```
struct Meta {
  int type; // 0 - seg, 1 - circle
  pt O;
  dbl R;
};

const Meta SEG = {0, pt(0, 0), 0};

vector<pair<pt, Meta>> cut(vector<pair<pt, Meta>> p, Line l) {
  vector<pair<pt,Meta>> res;
  int n = p.size();
  for (int i = 0; i < n; i++) {
    pt A = p[i].F;
    pt B = p[(i + 1) % n].F;
    if (le(0, l.v * (A - l.O))) {
      if (eq(0, l.v * (A - l.O)) && p[i].S.type == 1 && ls(0, l.v % (p[i].S.O - A)))
        res.pb({A, SEG});
      else
        res.pb(p[i]);
    }
    if (p[i].S.type == 0) {
      if (sign(l.v * (A - l.O)) * sign(l.v * (B - l.O)) == -1) {
        pt FF = Line(A, B) * l;
        res.pb(make_pair(FF, SEG));
      }
    }
    else {
      pt E, F;
      if (intCL(p[i].S.O, p[i].S.R, l, E, F)) {
        if (onArc(p[i].S.O, A, E, B))
          res.pb({E, SEG});
        if (onArc(p[i].S.O, A, F, B))
          res.pb({F, p[i].S});
      }
    }
  }
  return res;
}
```

final/geom/polygonTangent.cpp

```
pt tangent(vector<pt>& p, pt O, int cof) {
  int step = 1;
  for (; step < (int)p.size(); step *= 2);
  int pos = 0;
  int n = p.size();
  for (; step > 0; step /= 2) {
    int best = pos;
    for (int dx = -1; dx <= 1; dx += 2) {
      int id = ((pos + step * dx) % n + n) % n;
      if ((p[id] - O) * (p[best] - O) * cof > 0)
        best = id;
    }
    pos = best;
  }
  return p[pos];
}
```

final/strings/eertree.cpp

```
namespace eertree {
  const int INF = 1e9;
  const int N = 5e6 + 10;
  char _s[N];
  char *s = _s + 1;
  int to[N][2];
  int suf[N], len[N];
  int sz, last;

  const int odd = 1, even = 2, blank = 3;

  void go(int &u, int pos) {
    while (u != blank && s[pos - len[u] - 1] != s[pos]) {
      u = suf[u];
    }
  }

  int add(int pos) {
    go(last, pos);
    int u = suf[last];
    go(u, pos);
    int c = s[pos] - 'a';
    int res = 0;
    if (!to[last][c]) {
      res = 1;
      to[last][c] = sz;
      len[sz] = len[last] + 2;
      suf[sz] = to[u][c];
      sz++;
    }
    last = to[last][c];
    return res;
  }

  void init() {
    to[blank][0] = to[blank][1] = even;
```

```
    len[blank] = suf[blank] = INF;
    len[even] = 0, suf[even] = odd;
    len[odd] = -1, suf[odd] = blank;
    last = even;
    sz = 4;
  }
}
```

final/strings/sufAutomaton.cpp

```
namespace SA {
  const int MAXN = 1 << 18;
  const int SIGMA = 26;

  int sz, last;
  int nxt[MAXN][SIGMA];
  int link[MAXN], len[MAXN], pos[MAXN];

  void init() {
    memset(nxt, -1, sizeof(nxt));
    memset(link, -1, sizeof(link));
    memset(len, 0, sizeof(len));
    last = 0;
    sz = 1;
  }

  void add(int c) {
    int cur = sz++;
    len[cur] = len[last] + 1;
    pos[cur] = len[cur];
    int p = last;
    last = cur;
    for (; p != -1 && nxt[p][c] == -1; p = link[p]) nxt[p][c] = cur;
    if (p == -1) {
      link[cur] = 0;
      return;
    }
    int q = nxt[p][c];
    if (len[p] + 1 == len[q]) {
      link[cur] = q;
      return;
    }
    int clone = sz++;
    memcpy(nxt[clone], nxt[q], sizeof(nxt[q]));
    len[clone] = len[p] + 1;
    pos[clone] = pos[q];
    link[clone] = link[q];
    link[q] = link[cur] = clone;
    for (; p != -1 && nxt[p][c] == q; p = link[p]) nxt[p][c] = clone;
  }

  int n;
  string s;
  int l[MAXN], r[MAXN];
  int e[MAXN][SIGMA];

  void getSufTree(string _s) {
    memset(e, -1, sizeof(e));
    s = _s;
    n = s.length();
    reverse(s.begin(), s.end());
    init();
    for (int i = 0; i < n; i++) add(s[i] - 'a');
    reverse(s.begin(), s.end());
    for (int i = 1; i < sz; i++) {
      int j = link[i];
      l[i] = n - pos[i] + len[j];
      r[i] = n - pos[i] + len[i];
      e[j][s[l[i]] - 'a'] = i;
    }
  }
}
```

final/strings/duval.cpp

```
void duval(string s) {
  int n = (int) s.length();
  int i=0;
  while (i < n) {
    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) {
      cout << s.substr(i, j-k) << ' ';
      i += j - k;
    }
  }
}
```

final/graphs/centroid.cpp

```
// original author: burunduk1, rewritten by me (enot110)
// !!! warning !!! this code is not tested well
const int N = 1e5, K = 17;

int pivot, level[N], parent[N];
vector<int> v[N];

int get_pivot( int x, int xx, int n ) {
  int size = 1;
  for (int y : v[x])
  {
    if (y != xx && level[y] == -1) size += get_pivot(y, x, n);
  }
  if (pivot == -1 && (size * 2 >= n || xx == -1)) pivot = x;
  return size;
}

void build( int x, int xx, int dep, int size ) {
  assert(dep < K);
  pivot = -1;
  get_pivot(x, -1, size);
  x = pivot;
  level[x] = dep, parent[x] = xx;
  for (int y : v[x]) if (level[y] == -1)
  {
    build(y, x, dep + 1, size / 2);
  }
}
```

final/graphs/dominatorTree.cpp

```
namespace domtree {
  const int K = 18;
  const int N = 1 << K;

  int n, root;
  vector<int> e[N], g[N];
  int sdom[N], dom[N];
  int p[N][K], h[N], pr[N];
  int in[N], out[N], tmr, rev[N];

  void init(int _n, int _root) {
    n = _n;
    root = _root;
    tmr = 0;
    for (int i = 0; i < n; i++) {
      e[i].clear();
      g[i].clear();
      in[i] = -1;
    }
  }

  void addEdge(int u, int v) {
    e[u].push_back(v);
    g[v].push_back(u);
  }

  void dfs(int v) {
    in[v] = tmr++;
    for (int to : e[v]) {
      if (in[to] != -1) continue;
      pr[to] = v;
      dfs(to);
    }
    out[v] = tmr - 1;
  }

  int lca(int u, int v) {
    if (h[u] < h[v]) swap(u, v);
    for (int i = 0; i < K; i++) if ((h[u] - h[v]) & (1 << i)) u = p[u][i];
    if (u == v) return u;
    for (int i = K - 1; i >= 0; i--) {
      if (p[u][i] != p[v][i]) {
        u = p[u][i];
        v = p[v][i];
      }
    }
    return p[u][0];
  }

  void solve(int _n, int _root, vector<pair<int, int> > _edges) {
    init(_n, _root);
    for (auto ed : _edges) addEdge(ed.first, ed.second);

    dfs(root);
    for (int i = 0; i < n; i++) if (in[i] != -1) rev[in[i]] = i;
    segtree tr(tmr); // a[i]:=min(a[i],x) and return a[i]
    for (int i = tmr - 1; i >= 0; i--) {
      int v = rev[i];
      int cur = i;
      for (int to : g[v]) {
        if (in[to] == -1) continue;
        if (in[to] < in[v]) cur = min(cur, in[to]);
        else cur = min(cur, tr.get(in[to]));
      }
      sdom[v] = rev[cur];
      tr.upd(in[v], out[v], in[sdom[v]]);
    }
    for (int i = 0; i < tmr; i++) {
      int v = rev[i];
      if (i == 0) {
        dom[v] = v;
        h[v] = 0;
      } else {
        dom[v] = lca(sdom[v], pr[v]);
        h[v] = h[dom[v]] + 1;
      }
      p[v][0] = dom[v];
      for (int j = 1; j < K; j++) p[v][j] = p[p[v][j - 1]][j - 1];
    }
    for (int i = 0; i < n; i++) if (in[i] == -1) dom[i] = -1;
  }
}
```

final/graphs/generalMatching.cpp

```
//COPYPASTED FROM E-MAXX
namespace GeneralMatching {
  const int MAXN = 256;
  int n;
  vector<int> g[MAXN];
  int match[MAXN], p[MAXN], base[MAXN], q[MAXN];
  bool used[MAXN], blossom[MAXN];

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

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

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

    used[root] = true;
    int qh=0, qt=0;
    q[qt++] = root;
    while (qh < qt) {
```

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

  vector<pair<int, int> > solve(int _n, vector<pair<int, int> > edges) {
    n = _n;
    for (int i = 0; i < n; i++) g[i].clear();
    for (auto o : edges) {
      g[o.first].push_back(o.second);
      g[o.second].push_back(o.first);
    }
    memset (match, -1, sizeof match);
    for (int i=0; i<n; ++i) {
      if (match[i] == -1) {
        int v = find_path (i);
        while (v != -1) {
          int pv = p[v], ppv = match[pv];
          match[v] = pv, match[pv] = v;
          v = ppv;
        }
      }
    }
    vector<pair<int, int> > ans;
    for (int i = 0; i < n; i++) {
      if (match[i] > i) {
        ans.push_back(make_pair(i, match[i]));
      }
    }
    return ans;
  }
}
```

final/graphs/heavyLight.cpp

```
namespace hld {
  const int N = 1 << 17;
  int par[N], heavy[N], h[N];
  int root[N], pos[N];
  int n;
  vector<vector<int> > e;
  segtree tree;

  int dfs(int v) {
    int sz = 1, mx = 0;
    for (int to : e[v]) {
      if (to == par[v]) continue;
      par[to] = v;
      h[to] = h[v] + 1;
      int cur = dfs(to);
      if (cur > mx) heavy[v] = to, mx = cur;
      sz += cur;
    }
    return sz;
  }

  template <typename T>
  void path(int u, int v, T op) {
    for (; root[u] != root[v]; v = par[root[v]]) {
      if (h[root[u]] > h[root[v]]) swap(u, v);
      op(pos[root[v]], pos[v] + 1);
    }
    if (h[u] > h[v]) swap(u, v);
    op(pos[u], pos[v] + 1);
  }

  void init(vector<vector<int> > _e) {
    e = _e;
    n = e.size();
    tree = segtree(n);
    memset(heavy, -1, sizeof(heavy[0]) * n);
    par[0] = -1;
    h[0] = 0;
    dfs(0);
    for (int i = 0, cpos = 0; i < n; i++) {
      if (par[i] == -1 || heavy[par[i]] != i) {
        for (int j = i; j != -1; j = heavy[j]) {
          root[j] = i;
          pos[j] = cpos++;
        }
      }
    }
  }

  void add(int v, int x) {
    tree.add(pos[v], x);
  }

  int get(int u, int v) {
    int res = 0;
    path(u, v, [&](int l, int r) {
      res = max(res, tree.get(l, r));
    });
    return res;
  }
}
```

final/graphs/hungary.cpp

```
namespace hungary
{
  const int N = 210;

  int a[N][N];
  int ans[N];

  int calc(int n, int m)
  {
    ++n, ++m;
    vi u(n), v(m), p(m), prev(m);
    for (int i = 1; i < n; ++i)
    {
      p[0] = i;
      int x = 0;
      vi mn(m, inf);
      vi was(m, 0);
      while (p[x])
      {
        was[x] = 1;
        int ii = p[x], dd = inf, y = 0;
        for (int j = 1; j < m; ++j) if (!was[j])
        {
          int cur = a[ii][j] - u[ii] - v[j];
          if (cur < mn[j]) mn[j] = cur, prev[j] = x;
          if (mn[j] < dd) dd = mn[j], y = j;
        }
        forn(j, m)
        {
          if (was[j]) u[p[j]] += dd, v[j] -= dd;
          else mn[j] -= dd;
        }
        x = y;
      }
      while (x)
      {
        int y = prev[x];
        p[x] = p[y];
        x = y;
      }
    }
    for (int j = 1; j < m; ++j)
    {
      ans[p[j]] = j;
    }
    return -v[0];
```

```cpp
  }
  // HOW TO USE ::
  // -- set values to a[1..n][1..m] (n <= m)
  // -- run calc(n, m) to find MINIMUM
  // -- to restore permutation use ans[]
  // -- everything works on negative numbers
  //
  // !! i don't understand this code, it's copypasted from e-maxx (and rewrited by enot110)
}
```

final/graphs/minCostNegCycle.cpp

```cpp
struct Edge {
  int from, to, cap, flow;
  double cost;

};

struct Graph {
  int n;
  vector<Edge> edges;
  vector<vector<int> > e;

  Graph(int _n) {
    n = _n;
    e.resize(n);
  }

  void addEdge(int from, int to, int cap, double cost) {
    e[from].push_back(edges.size());
    edges.push_back({ from, to, cap, 0, cost });
    e[to].push_back(edges.size());
    edges.push_back({ to, from, 0, 0, -cost });
  }

  void maxflow() {
    while (1) {
      queue<int> q;
      vector<int> d(n, INF);
      vector<int> pr(n, -1);
      q.push(0);
      d[0] = 0;
      while (!q.empty()) {
        int v = q.front();
        q.pop();
        for (int i = 0; i < (int)e[v].size(); i++) {
          Edge cur = edges[e[v][i]];
          if (d[cur.to] > d[v] + 1 && cur.flow < cur.cap) {
            d[cur.to] = d[v] + 1;
            pr[cur.to] = e[v][i];
            q.push(cur.to);
          }
        }
      }
      if (d[n - 1] == INF) break;
      int v = n - 1;
      while (v) {
        edges[pr[v]].flow++;
        edges[pr[v] ^ 1].flow--;
        v = edges[pr[v]].from;
      }
    }
  }

  bool findcycle() {
    int iters = n;
    vector<int> changed;
    for (int i = 0; i < n; i++) changed.push_back(i);

    vector<vector<double> > d(iters + 1, vector<double>(n, INF));
    vector<vector<int> > p(iters + 1, vector<int>(n, -1));
    d[0].assign(n, 0);
    for (int it = 0; it < iters; it++) {
      d[it + 1] = d[it];
      vector<int> nchanged(n, 0);
      for (int v : changed) {
        for (int id : e[v]) {
          Edge cur = edges[id];
          if (d[it + 1][cur.to] > d[it][v] + cur.cost && cur.flow < cur.cap) {
            d[it + 1][cur.to] = d[it][v] + cur.cost;
            p[it + 1][cur.to] = id;
            nchanged[cur.to] = 1;
          }
        }
      }
      changed.clear();
      for (int i = 0; i < n; i++) if (nchanged[i]) changed.push_back(i);
    }
    if (changed.empty()) return 0;

    int bestU = 0, bestK = 1;
    double bestAns = INF;
    for (int u = 0; u < n; u++) {
      double curMax = -INF;
      for (int k = 0; k < iters; k++) {
        double curVal = (d[iters][u] - d[k][u]) / (iters - k);
        curMax = max(curMax, curVal);
      }
      if (bestAns > curMax) {
        bestAns = curMax;
        bestU = u;
      }
    }

    int v = bestU;
    int it = iters;
    vector<int> was(n, -1);
    while (was[v] == -1) {
      was[v] = it;
      v = edges[p[it][v]].from;
      it--;
    }
    int vv = v;
    it = was[v];
    double sum = 0;
    do {
      edges[p[it][v]].flow++;
      sum += edges[p[it][v]].cost;
      edges[p[it][v] ^ 1].flow--;
      v = edges[p[it][v]].from;
      it--;
    } while (v != vv);
    return 1;
  }
};
```

final/graphs/retro.cpp

```cpp
namespace retro
{
  const int N = 4e5 + 10;

  vi v[N];
  vi vrev[N];

  void add(int x, int y)
  {
    v[x].pb(y);
    vrev[y].pb(x);
  }

  const int UD = 0;
  const int WIN = 1;
  const int LOSE = 2;

  int res[N];
  int moves[N];
  int deg[N];
  int q[N], st, en;

  void calc(int n)
  {
    forn(i, n) deg[i] = sz(v[i]);
    st = en = 0;
    forn(i, n) if (!deg[i])
    {
      q[en++] = i;
      res[i] = LOSE;
    }
    while (st < en)
    {
      int x = q[st++];
      for (int y : vrev[x])
      {
```

```
        if (res[y] == UD && (res[x] == LOSE || (--deg[y] == 0 && res[x] == WIN)))
        {
          res[y] = 3 - res[x];
          moves[y] = moves[x] + 1;
          q[en++] = y;
        }
      }
    }
  }
}
```

final/graphs/smith.cpp

```
const int N = 1e5 + 10;

struct graph
{
  int n;

  vi v[N];
  vi vrev[N];

  void read()
  {
    int m;
    scanf("%d%d", &n, &m);
    forn(i, m)
    {
      int x, y;
      scanf("%d%d", &x, &y);
      --x, --y;
      v[x].pb(y);
      vrev[y].pb(x);
    }
  }

  int deg[N], cnt[N], used[N], f[N];
  int q[N], st, en;

  set<int> s[N];

  void calc()
  {
    forn(x, n) f[x] = -1, cnt[x] = 0;
    int val = 0;
    while (1)
    {
      st = en = 0;
      forn(x, n)
      {
        deg[x] = 0;
        used[x] = 0;
        for (int y : v[x]) if (f[y] == -1) deg[x]++;
      }
      forn(x, n) if (!deg[x] && f[x] == -1 && cnt[x] == val)
      {
        q[en++] = x;
        f[x] = val;
      }
      if (!en) break;
      while (st < en)
      {
        int x = q[st];
        st++;
        for (int y : vrev[x])
        {
          if (used[y] == 0 && f[y] == -1)
          {
            used[y] = 1;
            cnt[y]++;
            for (int z : vrev[y])
            {
              deg[z]--;
              if (f[z] == -1 && deg[z] == 0 && cnt[z] == val)
              {
                f[z] = val;
                q[en++] = z;
              }
            }
          }
        }
      }
      val++;
    }
    forn(x, n) eprintf("%d%c", f[x], " \n"[x + 1 == n]);
    forn(x, n) if (f[x] == -1)
    {
      for (int y : v[x]) if (f[y] != -1) s[x].insert(f[y]);
    }
  }
} g1, g2;

string get(int x, int y)
{
  int f1 = g1.f[x], f2 = g2.f[y];
  if (f1 == -1 && f2 == -1) return "draw";
  if (f1 == -1) {
    if (g1.s[x].count(f2)) return "first";
    return "draw";
  }
  if (f2 == -1) {
    if (g2.s[y].count(f1)) return "first";
    return "draw";
  }
  if (f1 ^ f2) return "first";
  return "second";
}
```

final/graphs/mincut.cpp

```
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;
      }
    }
  }
}
```

final/graphs/twoChineseFast.cpp

```
namespace twoc {
  struct Heap {
    static Heap* null;
    ll x, xadd;
    int ver, h;
    /* ANS */ int ei;
    Heap *l, *r;
    Heap(ll xx, int vv) : x(xx), xadd(0), ver(vv), h(1), l(null), r(null) {}
    Heap(const char*) : x(0), xadd(0), ver(0), h(0), l(this), r(this) {}
```

```
    void add(ll a) { x += a; xadd += a; }
    void push() {
      if (l != null) l->add(xadd);
      if (r != null) r->add(xadd);
      xadd = 0;
    }
  };
  Heap *Heap::null = new Heap("wqeqw");
  Heap* merge(Heap *l, Heap *r) {
    if (l == Heap::null) return r;
    if (r == Heap::null) return l;
    l->push(); r->push();
    if (l->x > r->x)
      swap(l, r);
    l->r = merge(l->r, r);
    if (l->l->h < l->r->h)
      swap(l->l, l->r);
    l->h = l->r->h + 1;
    return l;
  }
  Heap *pop(Heap *h) {
    h->push();
    return merge(h->l, h->r);
  }
  const int N = 666666;
  struct DSU {
    int p[N];
    void init(int nn) { iota(p, p + nn, 0); }
    int get(int x) { return p[x] == x ? x : p[x] = get(p[x]); }
    void merge(int x, int y) { p[get(y)] = get(x); }
  } dsu;
  Heap *eb[N];
  int n;
  /* ANS */ struct Edge {
  /* ANS */   int x, y;
  /* ANS */   ll c;
  /* ANS */ };
  /* ANS */ vector<Edge> edges;
  /* ANS */ int answer[N];
  void init(int nn) {
    n = nn;
    dsu.init(n);
    fill(eb, eb + n, Heap::null);
    edges.clear();
  }
  void addEdge(int x, int y, ll c) {
    Heap *h = new Heap(c, x);
    /* ANS */ h->ei = sz(edges);
    /* ANS */ edges.push_back({x, y, c});
    eb[y] = merge(eb[y], h);
  }
  ll solve(int root = 0) {
    ll ans = 0;
    static int done[N], pv[N];
    memset(done, 0, sizeof(int) * n);
    done[root] = 1;
    int tt = 1;
    /* ANS */ int cnum = 0;
    /* ANS */ static vector<ipair> eout[N];
    /* ANS */ for (int i = 0; i < n; ++i) eout[i].clear();
    for (int i = 0; i < n; ++i) {
      int v = dsu.get(i);
      if (done[v])
        continue;
      ++tt;
      while (true) {
        done[v] = tt;
        int nv = -1;
        while (eb[v] != Heap::null) {
          nv = dsu.get(eb[v]->ver);
          if (nv == v) {
            eb[v] = pop(eb[v]);
            continue;
          }
          break;
        }
        if (nv == -1)
          return LINF;
        ans += eb[v]->x;
        eb[v]->add(-eb[v]->x);
        /* ANS */ int ei = eb[v]->ei;
        /* ANS */ eout[edges[ei].x].push_back({++cnum, ei});
        if (!done[nv]) {
          pv[v] = nv;
          v = nv;
          continue;
        }
        if (done[nv] != tt)
          break;
        int v1 = nv;
        while (v1 != v) {
          eb[v] = merge(eb[v], eb[v1]);
          dsu.merge(v, v1);
          v1 = dsu.get(pv[v1]);
        }
      }
    }
    /* ANS */ memset(answer, -1, sizeof(int) * n);
    /* ANS */ answer[root] = 0;
    /* ANS */ set<ipair> es(all(eout[root]));
    /* ANS */ while (!es.empty()) {
    /* ANS */   auto it = es.begin();
    /* ANS */   int ei = it->second;
    /* ANS */   es.erase(it);
    /* ANS */   int nv = edges[ei].y;
    /* ANS */   if (answer[nv] != -1)
    /* ANS */     continue;
    /* ANS */   answer[nv] = ei;
    /* ANS */   es.insert(all(eout[nv]));
    /* ANS */ }
    /* ANS */ answer[root] = -1;
    return ans;
  }
  /* Usage: twoc::init(vertex_count);
   *    twoc::addEdge(v1, v2, cost);
   *    twoc::solve(root); - returns cost or LINF
   * twoc::answer contains index of ingoing edge for each vertex
   */
}
```

final/graphs/linkcut.cpp

```
#include <iostream>
#include <cstdio>
#include <cassert>

using namespace std;

// BEGIN ALGO

const int MAXN = 110000;

typedef struct _node{
 _node *l, *r, *p, *pp;
 int size; bool rev;
 _node();
 explicit _node(nullptr_t){
  l = r = p = pp = this;
  size = rev = 0;
 }
 void push(){
  if (rev){
   l->rev ^= 1; r->rev ^= 1;
   rev = 0; swap(l,r);
  }
 }
 void update();
}* node;
node None = new _node(nullptr);
node v2n[MAXN];
_node::_node(){
 l = r = p = pp = None;
 size = 1; rev = false;
}
void _node::update(){
 size = (this != None) + l->size + r->size;
 l->p = r->p = this;
}
void rotate(node v){
 assert(v != None && v->p != None);
 assert(!v->rev); assert(!v->p->rev);
 node u = v->p;
 if (v == u->l)
  u->l = v->r, v->r = u;
 else
  u->r = v->l, v->l = u;
 swap(u->p,v->p); swap(v->pp,u->pp);
 if (v->p != None){
  assert(v->p->l == u || v->p->r == u);
  if (v->p->r == u) v->p->r = v;
  else v->p->l = v;
 }
 u->update(); v->update();
}
void bigRotate(node v){
 assert(v->p != None);
```

```
 v->p->p->push();
 v->p->push();
 v->push();
 if (v->p->p != None){
  if ((v->p->l == v) ^ (v->p->p->r == v->p))
   rotate(v->p);
  else
   rotate(v);
 }
 rotate(v);
}
inline void Splay(node v){
 while (v->p != None) bigRotate(v);
}
inline void splitAfter(node v){
 v->push();
 Splay(v);
 v->r->p = None;
 v->r->pp = v;
 v->r = None;
 v->update();
}
void expose(int x){
 node v = v2n[x];
 splitAfter(v);
 while (v->pp != None){
  assert(v->p == None);
  splitAfter(v->pp);
  assert(v->pp->r == None);
  assert(v->pp->p == None);
  assert(!v->pp->rev);
  v->pp->r = v;
  v->pp->update();
  v = v->pp;
  v->r->pp = None;
 }
 assert(v->p == None);
 Splay(v2n[x]);
}
inline void makeRoot(int x){
 expose(x);
 assert(v2n[x]->p == None);
 assert(v2n[x]->pp == None);
 assert(v2n[x]->r == None);
 v2n[x]->rev ^= 1;
}
inline void link(int x,int y){
 makeRoot(x); v2n[x]->pp = v2n[y];
}
inline void cut(int x,int y){
 expose(x);
 Splay(v2n[y]);
 if (v2n[y]->pp != v2n[x]){
  swap(x,y);
  expose(x);
  Splay(v2n[y]);
  assert(v2n[y]->pp == v2n[x]);
 }
 v2n[y]->pp = None;
}
inline int get(int x,int y){
 if (x == y) return 0;
 makeRoot(x);
 expose(y); expose(x);
 Splay(v2n[y]);
 if (v2n[y]->pp != v2n[x]) return -1;
 return v2n[y]->size;
}
// END ALGO

_node mem[MAXN];


int main(){
 freopen("linkcut.in","r",stdin);
 freopen("linkcut.out","w",stdout);

 int n,m;
 scanf("%d %d",&n,&m);

 for (int i = 0; i < n; i++)
  v2n[i] = &mem[i];

 for (int i = 0; i < m; i++){
  int a,b;
  if (scanf(" link %d %d",&a,&b) == 2)
   link(a-1,b-1);
  else if (scanf(" cut %d %d",&a,&b) == 2)
   cut(a-1,b-1);
  else if (scanf(" get %d %d",&a,&b) == 2)
   printf("%d\n",get(a-1,b-1));
  else
   assert(false);
 }
 return 0;
}
```

final/graphs/chordaltree.cpp

```
void chordaltree(vector<vector<int>> e) {
  int n = e.size();

  vector<int> mark(n);
  set<pair<int, int> > st;
  for (int i = 0; i < n; i++) st.insert({-mark[i], i});

  vector<int> vct(n);
  vector<pair<int, int> > ted;
  vector<vector<int> > who(n);
  vector<vector<int> > verts(1);
  vector<int> cliq(n, -1);
  cliq.push_back(0);
  vector<int> last(n + 1, n);
  int prev = n + 1;
  for (int i = n - 1; i >= 0; i--) {
    int x = st.begin()->second;
    st.erase(st.begin());
    if (mark[x] <= prev) {
      vector<int> cur = who[x];
      cur.push_back(x);
      verts.push_back(cur);
      ted.push_back({cliq[last[x]], (int)verts.size() - 1});
    } else {
      verts.back().push_back(x);
    }
    for (int y : e[x]) {
      if (cliq[y] != -1) continue;
      who[y].push_back(x);
      st.erase({-mark[y], y});
      mark[y]++;
      st.insert({-mark[y], y});
      last[y] = x;
    }
    prev = mark[x];
    vct[i] = x;
    cliq[x] = (int)verts.size() - 1;
  }

  int k = verts.size();
  vector<int> pr(k);
  vector<vector<int> > g(k);
  for (auto o : ted) {
    pr[o.second] = o.first;
    g[o.first].push_back(o.second);
  }
}
```

final/graphs/minimization.cpp

```
namespace mimimi /* ^_^ */ {
  const int N = 100555;
  const int S = 3;
  int e[N][S];
  int label[N];
  vector<int> eb[N][S];
  int ans[N];
  void solve(int n) {
    for (int i = 0; i < n; ++i)
      for (int j = 0; j < S; ++j)
        eb[i][j].clear();
    for (int i = 0; i < n; ++i)
      for (int j = 0; j < S; ++j)
        eb[e[i][j]][j].push_back(i);
    vector<unordered_set<int>> classes(*max_element(label, label + n) + 1);
    for (int i = 0; i < n; ++i)
      classes[label[i]].insert(i);
    for (int i = 0; i < sz(classes); ++i)
      if (classes[i].empty()) {
        classes[i].swap(classes.back());
        classes.pop_back();
        --i;
```

```
      }
    for (int i = 0; i < sz(classes); ++i)
      for (int v : classes[i])
        ans[v] = i;
    for (int i = 0; i < sz(classes); ++i)
      for (int c = 0; c < S; ++c) {
        unordered_map<int, unordered_set<int>> involved;
        for (int v : classes[i])
          for (int nv : eb[v][c])
            involved[ans[nv]].insert(nv);
        for (auto &pp : involved) {
          int cl = pp.X;
          auto &cls = classes[cl];
          if (sz(pp.Y) == sz(cls))
            continue;
          for (int x : pp.Y)
            cls.erase(x);
          if (sz(cls) < sz(pp.Y))
            cls.swap(pp.Y);
          for (int x : pp.Y)
            ans[x] = sz(classes);
          classes.push_back(move(pp.Y));
        }
      }
  }
  /* Usage: initialize edges: e[vertex][character]
            labels: label[vertex]
      solve(n)
      ans[] - classes
  */
}
```

final/graphs/matroidIntersection.cpp

```
  // check(ctaken, 1) -- first matroid
  // check(ctaken, 2) -- second matroid
  vector<char> taken(m);
  while (1) {
    vector<vector<int>> e(m);
    for (int i = 0; i < m; i++) {
      for (int j = 0; j < m; j++) {
        if (taken[i] && !taken[j]) {
          auto ctaken = taken;
          ctaken[i] = 0;
          ctaken[j] = 1;
          if (check(ctaken, 2)) {
            e[i].push_back(j);
          }
        }
        if (!taken[i] && taken[j]) {
          auto ctaken = taken;
          ctaken[i] = 1;
          ctaken[j] = 0;
          if (check(ctaken, 1)) {
            e[i].push_back(j);
          }
        }
      }
    }
    vector<int> type(m);
    // 0 -- cant, 1 -- can in \2, 2 -- can in \1
    for (int i = 0; i < m; i++) {
      if (!taken[i]) {
        auto ctaken = taken;
        ctaken[i] = 1;
        if (check(ctaken, 2)) type[i] |= 1;
      }
      if (!taken[i]) {
        auto ctaken = taken;
        ctaken[i] = 1;
        if (check(ctaken, 1)) type[i] |= 2;
      }
    }
    vector<int> w(m);
    for (int i = 0; i < m; i++) {
      w[i] = taken[i] ? ed[i].c : -ed[i].c;
    }
    vector<pair<int, int>> d(m, {INF, 0});
    for (int i = 0; i < m; i++) {
      if (type[i] & 1) d[i] = {w[i], 0};
    }
    vector<int> pr(m, -1);
    while (1) {
      vector<pair<int, int>> nd = d;
      for (int i = 0; i < m; i++) {
        if (d[i].first == INF) continue;
        for (int to : e[i]) {
          if (nd[to] > make_pair(d[i].first + w[to], d[i].second + 1)) {
            nd[to] = make_pair(d[i].first + w[to], d[i].second + 1);
            pr[to] = i;
          }
        }
      }
      if (d == nd) break;
      d = nd;
    }
    int v = -1;
    for (int i = 0; i < m; i++) {
      if ((d[i].first < INF && (type[i] & 2)) && (v == -1 || d[i] < d[v])) v = i;
    }
    if (v == -1) break;
    while (v != -1) {
      sum += w[v];
      taken[v] ^= 1;
      v = pr[v];
    }
    ans[--cnt] = sum;
  }
```

dbl Simpson() { return (F(-1) + 4 * F(0) + F(1)) / 6; } dbl Runge2() { return (F(-sqrtl(1.0 / 3)) + F(sqrtl(1.0 / 3))) / 2; } dbl Runge3() { return (F(-sqrtl(3.0 / 5)) * 5 + F(0) * 8 + F(sqrtl(3.0 / 5)) * 5) / 18; }

Simpson и Runge2 – точны для полиномов степени <= 3 Runge3 – точен для полиномов степени <= 5

—

Явный Рунге-Кутт четвертого порядка, ошибка O(h^4)

y' = f(x, y) y_(n+1) = y_n + (k1 + 2 * k2 + 2 * k3 + k4) * h / 6

k1 = f(xn, yn) k2 = f(xn + h/2, yn + h/2 * k1) k3 = f(xn + h/2, yn + h/2 * k2) k4 = f(xn + h, yn + h * k3)

Методы Адамса-Башфорта

y_n+3 = y_n+2 + h * (23/12 * f(x_n+2,y_n+2) - 4/3 * f(x_n+1,y_n+1) + 5/12 * f(x_n,y_n)) y_n+4 = y_n+3 + h * (55/24 * f(x_n+3,y_n+3) - 59/24 * f(x_n+2,y_n+2) + 37/24 * f(x_n+1,y_n+1) - 3/8 * f(x_n,y_n)) y_n+5 = y_n+4 + h * (1901/720 * f(x_n+4,y_n+4) - 1387/360 * f(x_n+3,y_n+3) + 109/30 * f(x_n+2,y_n+2) - 637/360 * f(x_n+1,y_n+1) + 251/720 * f(x_n,y_n))

—

Извлечение корня по простому модулю (от Сережи) 3 <= p, 1 <= a < p, найти x^2 = a

1) Если a^((p - 1)/2) != 1, return -1 2) Выбрать случайный 1 <= i < p 3) T(x) = (x + i)^((p - 1)/2) mod (x^2 - a) = bx + c 4) Если b != 0 то вернуть c/b, иначе к шагу 2)

—

Иногда вместо того чтобы считать первообразный у простого числа, можно написать чекер ответа и перебирать случайный первообразный.

Иногда можно представить ответ в виде многочлена и вместо подсчета самих к-тов посчитать значения и проинтерполировать

—

Лемма Бернсайда:

Группа G действует на множество X Тогда число классов эквивалентности = (sum |f(g)| for g in G) / |G| где f(g) = число x (из X) : g(x) == x

—

Число простых быстрее O(n):

dp(n, k) – число чисел от 1 до n в которых все простые >= p[k] dp(n, 1) = n dp(n, j) = dp(n, j + 1) + dp(n / p[j], j), т. е. dp(n, j + 1) = dp(n, j) - dp(n / p[j], j)

Если p[j], p[k] > sqrt(n) то dp(n, j) + j == dp(n, k) + k

Хуяришь все оптимайзы сверху, но не считаешь глубже dp(n, k), n < K Потом фенвиком+сортировкой подсчитываешь за (K+Q)log все эти запросы Хуяришь во второй раз, но на этот раз берешь прекальканные значения

Если sqrt(n) < p[k] < n то (число простых до n)=dp(n, k) + k - 1

—

sum(k=1..n) k^2 = n(n+1)(2n+1)/6

sum(k=1..n) k^3 = n^2(n+1)^2/4

Чиселки:

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

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

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

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

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

Table of Integrals*

Basic Forms

$$\int x^n dx = \frac{1}{n+1}x^{n+1} \qquad (1)$$

$$\int \frac{1}{x}dx = \ln|x| \qquad (2)$$

$$\int udv = uv - \int vdu \qquad (3)$$

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

Integrals of Rational Functions

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

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

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

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

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

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

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

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

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

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

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

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

Integrals with Roots

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

$$\int \frac{1}{\sqrt{x\pm a}}dx = 2\sqrt{x\pm a} \qquad (18)$$

$$\int \frac{1}{\sqrt{a-x}}dx = -2\sqrt{a-x} \qquad (19)$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\int \frac{x}{\sqrt{x^2\pm a^2}}dx = \sqrt{x^2\pm a^2} \qquad (34)$$

$$\int \frac{x}{\sqrt{a^2-x^2}}dx = -\sqrt{a^2-x^2} \qquad (35)$$

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

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

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

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

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

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

Integrals with Logarithms

$$\int \ln ax\,dx = x\ln ax - x \qquad (42)$$

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

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

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

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

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

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

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

Integrals with Exponentials

$$\int e^{ax}dx = \frac{1}{a}e^{ax} \qquad (50)$$

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

$$\int xe^x dx = (x-1)e^x \qquad (52)$$

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

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

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

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

$$\int x^n e^{ax}\,\mathrm{dx} = \frac{x^n e^{ax}}{a} - \frac{n}{a}\int x^{n-1}e^{ax}\,\mathrm{dx} \qquad (57)$$

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

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

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

$$\int xe^{-ax^2}\,\mathrm{dx} = -\frac{1}{2a}e^{-ax^2} \qquad (61)$$

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

* © 2014. From http://integral-table.com, last revised June 14, 2014. This material is provided as is without warranty or representation about the accuracy, correctness or suitability of the material for any purpose, and is licensed under the Creative Commons Attribution-Noncommercial-ShareAlike 3.0 United States License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/3.0/ or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

Integrals with Trigonometric Functions

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Products of Trigonometric Functions and Monomials

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

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

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

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

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

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

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

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

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

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

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

Products of Trigonometric Functions and Exponentials

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

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

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

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

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

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

Integrals of Hyperbolic Functions

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

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

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

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

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

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

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

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

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

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

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

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