Содержание

final/template/vimrc.txt

```
1  map <F9> :wall! <CR> :!g++ -Wall -Wextra -Wshadow -
       Wno-unused-result -o %:r % -std=c++14 -DHOME -
       D_GLIBCXX_DEBUG -fsanitize=address <CR>
2  map <F7> :wall! <CR> :!g++ -Wall -Wextra -Wshadow -
       Wno-unused-result -o %:r % -std=c++14 -DHOME -
       O2 <CR>
3  map <F8> :wall! <CR> :!ulimit -s 500000 && ./%:r <CR
       >
4  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>
5
6  inoremap {<CR> {<CR>}<ESC>O
7  map <c-a> ggVG
8
9  set nu
10 set rnu
11 syntax on
12
13 map <c-t> :tabnew <CR>
14 map <c-l> :tabn <CR>
15 map <c-h> :tabp <CR>
16
17 set cin
18 set sw=4
19 set so=99
20 set bs=2
21 set et
22 set sts=4
```

final/template/template.cpp

```cpp
1  // team : SPb ITMO University 1
2  #include <bits/stdc++.h>
3
4  #define F first
5  #define S second
6  #define X first
7  #define Y second
8  #define pb push_back
9  #define sz(a) (int)(a).size()
10 #define all(a) (a).begin(),a.end()
11 #define pw(x) (1LL<<(x))
12
13 #define db(x) cerr << #x << " = " << x << endl
14 #define db2(x, y) cerr << "(" << #x << ", " << #y <<
       ") = (" << x << ", " << y << ")\n";
15 #define db3(x, y, z) cerr << "(" << #x << ", " << #y
       << ", " << #z << ") = (" << x << ", " << y <<
       ", " << z << ")\n"
16 #define dbv(a) cerr << #a << " = "; for (auto xxxx:
       a) cerr << xxxx << " "; cerr << endl
17
18 using namespace std;
19
20 typedef long long ll;
21 typedef double dbl;
22 const int INF = 1.01e9;
23
24
25
26 int main() {
27 #define TASK ""
28 #ifdef HOME
29   assert(freopen(TASK".in", "r", stdin));
30 #endif
31
32
33
34
35 #ifdef HOME
36   cerr << "time: " << clock() * 1.0 / CLOCKS_PER_SEC
       << endl;
37 #endif
38   return 0;
39 }
```

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

```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

```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

```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

```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)) {
```

```
13      cout << i << endl;
14  }
```

final/template/Template.java

```java
1   import java.util.*;
2   import java.io.*;
3
4   public class Template {
5     FastScanner in;
6     PrintWriter out;
7
8     public void solve() throws IOException {
9       int n = in.nextInt();
10      out.println(n);
11    }
12
13    public void run() {
14      try {
15        in = new FastScanner();
16        out = new PrintWriter(System.out);
17
18        solve();
19
20        out.close();
21      } catch (IOException e) {
22        e.printStackTrace();
23      }
24    }
25
26    class FastScanner {
27      BufferedReader br;
28      StringTokenizer st;
29
30      FastScanner() {
31        br = new BufferedReader(new InputStreamReader(←
    System.in));
32      }
33
34      String next() {
35        while (st == null || !st.hasMoreTokens()) {
36          try {
37            st = new StringTokenizer(br.readLine());
38          } catch (IOException e) {
39            e.printStackTrace();
40          }
41        }
42        return st.nextToken();
43      }
44
45      int nextInt() {
46        return Integer.parseInt(next());
47      }
48    }
49
50    public static void main(String[] arg) {
51      new Template().run();
52    }
53  }
```

final/template/bitset.cpp

```cpp
1
2   const int SZ = 6;
3   const int BASE = pw(SZ);
4   const int MOD = BASE - 1;
5
6   struct Bitset {
7     typedef unsigned long long T;
8     vector<T> data;
9     int n;
10    void resize(int nn) {
11      n = nn;
12      data.resize((n + BASE - 1) / BASE);
13    }
14    void set(int pos, int val) {
15      int id = pos >> SZ;
16      int rem = pos & MOD;
17      data[id] ^= data[id] & pw(rem);
18      data[id] |= val * pw(rem);
19    }
```

```cpp
20    int get(int pos) {
21      return (data[pos >> SZ] >> (pos & MOD)) & 1;
22    }
23    // k > 0 -> (*this) << k
24    // k < 0 -> (*this) >> (-k)
25    Bitset shift (int k) {
26      Bitset res;
27      res.resize(n);
28      int s = k / BASE;
29      int rem = k % BASE;
30      if (rem < 0) {
31        rem += BASE;
32        s--;
33      }
34      int p1 = BASE - rem;
35      T mask = (p1 == 64)? -1: pw(p1) - 1;
36      for (int i = max(0, -s); i < sz(data) - max(s, ←
    0); i++) {
37        res.data[i + s] |= (data[i] & mask) << rem;
38      }
39      if (rem != 0) {
40        for (int i = max(0, -s - 1); i < sz(data) - ←
    max(s + 1, 0); i++) {
41          res.data[i + s + 1] |= (data[i] >> p1) & (pw←
    (rem) - 1);
42        }
43      }
44      int cc = data.size() * BASE - n;
45      res.data.back() <<= cc;
46      res.data.back() >>= cc;
47      return res;
48    }
49  };
```

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);
```

```cpp
         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

```
1  namespace fft {
2    const int MOD = 998244353;
3    const int base = 20;
4    const int N = 1 << base;
5    const int ROOT = 646;
6
7    int root[N];
8    int rev[N];
9
10   ll inv(ll a, ll m = MOD) {
11     if (a == 0) return 0;
12     return ((1 - inv(m % a, a) * m) / a + m) % m;
13   }
14
15   void init() {
16     for (int i = 0; i < N; i++) rev[i] = (rev[i >> ↩
     1] >> 1) + ((i & 1) << (base - 1));
17     int NN = N >> 1;
18     int z = 1;
19     for (int i = 0; i < NN; i++) {
20       root[i + NN] = z;
21       z = z * (ll)ROOT % MOD;
22     }
23     for (int i = NN - 1; i > 0; --i) root[i] = root↩
     [2 * i];
24   }
25
26   void fft(int *a, int *f) {
27     for (int i = 0; i < N; i++) f[i] = a[rev[i]];
28     for (int k = 1; k < N; k <<= 1) {
29       for (int i = 0; i < N; i += 2 * k) {
30         for (int j = 0; j < k; j++) {
31           int z = f[i + j + k] * (ll)root[j + k] % ↩
     MOD;
32           f[i + j + k] = (f[i + j] - z + MOD) % MOD;
33           f[i + j] = (f[i + j] + z) % MOD;
34         }
35       }
36     }
37   }
38
39   int A[N], B[N], C[N];
40   int F[N], G[N];
41
42   void _mult(int eq) {
43     fft(A, F);
44     if (eq)
45       for (int i = 0; i < N; i++)
46         G[i] = F[i];
47     else fft(B, G);
48     int invN = inv(N);
49     for (int i = 0; i < N; i++) A[i] = F[i] * (ll)G[↩
     i] % MOD * invN % MOD;
50     reverse(A + 1, A + N);
51     fft(A, C);
52   }
53
54   void mult(int n1, int n2, int eq = 0) {
55     for (int i = n1; i < N; ++i) A[i] = 0;
56     for (int i = n2; i < N; ++i) B[i] = 0;
57
58     _mult(eq);
59
60     //forn(i, n1 + n2) C[i] = 0;
61     //forn(i, n1) forn(j, n2) C[i + j] = (C[i + j] +↩
      A[i] * (ll)B[j]) % mod;
62   }
63 }
```

final/numeric/blackbox.cpp

```
1  namespace blackbox
2  {
3    int A[N];
4    int B[N];
5    int C[N];
6
7    int magic(int k, int x)
8    {
9      B[k] = x;
10     C[k] = (C[k] + A[0] * (ll)B[k]) % mod;
11     int z = 1;
12     if (k == N - 1) return C[k];
13     while ((k & (z - 1)) == (z - 1))
14     {
15       //mult B[k - z + 1 ... k] x A[z .. 2 * z - 1]
16       forn(i, z) fft::A[i] = A[z + i];
```

```
17       forn(i, z) fft::B[i] = B[k - z + 1 + i];
18       fft::multMod(z, z, mod);
19       forn(i, 2 * z - 1) C[k + 1 + i] = (C[k + 1 + i↩
     ] + fft::C[i]) % mod;
20       z <<= 1;
21     }
22     return C[k];
23   }
24   // A -- constant array
25   // magic(k, x):: B[k] = x, returns C[k]
26   // !! WARNING !! better to set N twice the size ↩
     needed
27 }
```

final/numeric/crt.cpp

```
1  int CRT(int a1, int m1, int a2, int m2) {
2    return (a1 - a2 % m1 + m1) * (ll)rev(m2, m1) % m1 ↩
     * m2 + a2;
3  }
```

final/numeric/mulMod.cpp

```
1  ll mul( ll a, ll b, ll m ) { // works for MOD 8e18
2    ll k = (ll)((long double)a * b / m);
3    ll r = a * b - m * k;
4    if (r < 0) r += m;
5    if (r >= m) r -= m;
6    return r;
7  }
```

final/numeric/modReverse.cpp

```
1  int rev(int x, int m) {
2    if (x == 1) return 1;
3    return (1 - rev(m % x, x) * (ll)m) / x + m;
4  }
```

final/numeric/pollard.cpp

```
1  namespace pollard
2  {
3    using math::p;
4
5    vector<pair<ll, int>> getFactors(ll N)
6    {
7      vector<ll> primes;
8
9      const int MX = 1e5;
10     const ll MX2 = MX * (ll)MX;
11
12     assert(MX <= math::maxP && math::pc > 0);
13
14     function<void(ll)> go = [&go, &primes](ll n)
15     {
16       for (ll x : primes) while (n % x == 0) n /= x;
17       if (n == 1) return;
18       if (n > MX2)
19       {
20         auto F = [&](ll x) {
21           ll k = ((long double)x * x) / n;
22           ll r = (x * x - k * n + 3) % n;
23           return r < 0 ? r + n : r;
24         };
25         ll x = mt19937_64()() % n, y = x;
26         const int C = 3 * pow(n, 0.25);
27
28         ll val = 1;
29         forn(it, C)
```

```cpp
30            {
31                x = F(x), y = F(F(y));
32                if (x == y) continue;
33                ll delta = abs(x - y);
34                ll k = ((long double)val * delta) / n;
35                val = (val * delta - k * n) % n;
36                if (val < 0) val += n;
37                if (val == 0)
38                {
39                    ll g = __gcd(delta, n);
40                    go(g), go(n / g);
41                    return;
42                }
43                if ((it & 255) == 0)
44                {
45                    ll g = __gcd(val, n);
46                    if (g != 1)
47                    {
48                        go(g), go(n / g);
49                        return;
50                    }
51                }
52            }
53        }
54        primes.pb(n);
55    };
56
57    ll n = N;
58
59    for (int i = 0; i < math::pc && p[i] < MX; ++i) ↩
       if (n % p[i] == 0)
60    {
61        primes.pb(p[i]);
62        while (n % p[i] == 0) n /= p[i];
63    }
64
65    go(n);
66
67    sort(primes.begin(), primes.end());
68
69    vector<pair<ll, int>> res;
70    for (ll x : primes)
71    {
72        int cnt = 0;
73        while (N % x == 0)
74        {
75            cnt++;
76            N /= x;
77        }
78        res.push_back({x, cnt});
79    }
80    return res;
81    }
82 }
```

```cpp
31        int fst = 1;
32        forn(i, sz(v)) if (v[i])
33        {
34            int x = v[i];
35            int sgn = 1;
36            if (x > mod / 2) x = mod-x, sgn = -1;
37            if (sgn == -1) ss << "-";
38            else if (!fst) ss << "+";
39            fst = 0;
40            if (!i || x != 1)
41            {
42                ss << x;
43                if (i > 0) ss << "*x";
44                if (i > 1) ss << "^" << i;
45            }
46            else
47            {
48                ss << "x";
49                if (i > 1) ss << "^" << i;
50            }
51        }
52        if (fst) ss <<"0";
53        string s;
54        ss >> s;
55        eprintf("%s\n", s.data());
56    }
57 };
58
59 poly operator + (poly A, poly B)
60 {
61    poly C;
62    C.v = vi(max(sz(A), sz(B)));
63    forn(i, sz(C))
64    {
65        if (i < sz(A)) C[i] = (C[i] + A[i]) % mod;
66        if (i < sz(B)) C[i] = (C[i] + B[i]) % mod;
67    }
68    return C.norm();
69 }
70
71 poly operator - (poly A, poly B)
72 {
73    poly C;
74    C.v = vi(max(sz(A), sz(B)));
75    forn(i, sz(C))
76    {
77        if (i < sz(A)) C[i] = (C[i] + A[i]) % mod;
78        if (i < sz(B)) C[i] = (C[i] + mod - B[i]) % mod;
79    }
80    return C.norm();
81 }
82
83 poly operator * (poly A, poly B)
84 {
85    poly C;
86    C.v = vi(sz(A) + sz(B) - 1);
87
88    forn(i, sz(A)) fft::A[i] = A[i];
89    forn(i, sz(B)) fft::B[i] = B[i];
90    fft::multMod(sz(A), sz(B), mod);
91    forn(i, sz(C)) C[i] = fft::C[i];
92    return C.norm();
93 }
94
95 poly inv(poly A, int n) // returns A^-1 mod x^n
96 {
97    assert(sz(A) && A[0] != 0);
98    A.cut(n);
99
100   auto cutPoly = [](poly &from, int l, int r)
101   {
102       poly R;
103       R.v.resize(r - l);
104       for (int i = l; i < r; ++i)
105       {
106           if (i < sz(from)) R[i - l] = from[i];
107       }
108       return R;
109   };
110
111   function<int(int, int)> rev = [&rev](int x, int m)↩
       ->int
112   {
113       if (x == 1) return 1;
114       return (1 - rev(m % x, x) * (ll)m) / x + m;
115   };
116
117   poly R({rev(A[0], mod)});
118   for (int k = 1; k < n; k <<= 1)
119   {
120       poly A0 = cutPoly(A, 0, k);
121       poly A1 = cutPoly(A, k, 2 * k);
122       poly H = A0 * R;
```

final/numeric/poly.cpp

```cpp
1  struct poly
2  {
3     vi v;
4     poly() {}
5     poly(vi vv)
6     {
7         v = vv;
8     }
9     int size()
10    {
11        return (int)v.size();
12    }
13    poly cut(int maxLen)
14    {
15        if (maxLen < sz(v)) v.resize(maxLen);
16        return *this;
17    }
18    poly norm()
19    {
20        while (sz(v) > 1 && v.back() == 0) v.pop_back();
21        return *this;
22    }
23    inline int& operator [] (int i)
24    {
25        return v[i];
26    }
27    void out(string name="")
28    {
29        stringstream ss;
30        if (sz(name)) ss << name << "=";
```

```
123        H = cutPoly(H, k, 2 * k);
124        poly R1 = (((A1 * R).cut(k) + H) * (poly({0}) - ←↩
           R)).cut(k);
125        R.v.resize(2 * k);
126        forn(i, k) R[i + k] = R1[i];
127    }
128    return R.cut(n).norm();
129 }
130
131 pair<poly, poly> divide(poly A, poly B)
132 {
133    if (sz(A) < sz(B)) return {poly({0}), A};
134
135    auto rev = [](poly f)
136    {
137        reverse(all(f.v));
138        return f;
139    };
140
141    poly q = rev((inv(rev(B), sz(A) - sz(B) + 1) * rev←↩
       (A)).cut(sz(A) - sz(B) + 1));
142    poly r = A - B * q;
143
144    return {q, r};
145 }
```

final/numeric/simplex.cpp

```
1  vector<double> simplex(vector<vector<double> > a) {
2     int n = a.size() - 1;
3     int m = a[0].size() - 1;
4     vector<int> left(n + 1), up(m + 1);
5     iota(up.begin(), up.end(), 0);
6     iota(left.begin(), left.end(), m);
7     auto pivot = [&](int x, int y) {
8        swap(left[x], up[y]);
9        double k = a[x][y];
10       a[x][y] = 1;
11       vector<int> vct;
12       for (int j = 0; j <= m; j++) {
13          a[x][j] /= k;
14          if (!eq(a[x][j], 0)) vct.push_back(j);
15       }
16       for (int i = 0; i <= n; i++) {
17          if (eq(a[i][y], 0) || i == x) continue;
18          k = a[i][y];
19          a[i][y] = 0;
20          for (int j : vct) a[i][j] -= k * a[x][j];
21       }
22    };
23    while (1) {
24       int x = -1;
25       for (int i = 1; i <= n; i++) if (ls(a[i][0], 0) ←↩
          && (x == -1 || a[i][0] < a[x][0])) x = i;
26       if (x == -1) break;
27       int y = -1;
28       for (int j = 1; j <= m; j++) if (ls(a[x][j], 0) ←↩
          && (y == -1 || a[x][j] < a[x][y])) y = j;
29       if (y == -1) assert(0); // infeasible
30       pivot(x, y);
31    }
32    while (1) {
33       int y = -1;
34       for (int j = 1; j <= m; j++) if (ls(0, a[0][j]) ←↩
          && (y == -1 || a[0][j] > a[0][y])) y = j;
35       if (y == -1) break;
36       int x = -1;
37       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;
38       if (x == -1) assert(0); // unbounded
39       pivot(x, y);
40    }
41    vector<double> ans(m + 1);
42    for (int i = 1; i <= n; i++) if (left[i] <= m) ans←↩
       [left[i]] = a[i][0];
43    ans[0] = -a[0][0];
44    return ans;
45 }
46 // j=1..m: x[j]>=0
47 // i=1..n: sum(j=1..m) A[i][j]*x[j] <= A[i][0]
48 // max sum(j=1..m) A[0][j]*x[j]
49 // res[0] is answer
50 // res[1..m] is certificate
```

final/numeric/sumLine.cpp

```
1  // sum(i=0..n-1) (a+b*i) div m
2  ll solve(ll n, ll a, ll b, ll m) {
3     if (b == 0) return n * (a / m);
4     if (a >= m) return n * (a / m) + solve(n, a % m, b←↩
       , m);
5     if (b >= m) return n * (n - 1) / 2 * (b / m) + ←↩
       solve(n, a, b % m, m);
6     return solve((a + b * n) / m, (a + b * n) % m, m, ←↩
       b);
7  }
```

final/numeric/berlekamp.cpp

```
1  vector<int> berlekamp(vector<int> s) {
2     int l = 0;
3     vector<int> la(1, 1);
4     vector<int> b(1, 1);
5     for (int r = 1; r <= (int)s.size(); r++) {
6        int delta = 0;
7        for (int j = 0; j <= l; j++) {
8           delta = (delta + 1LL * s[r - 1 - j] * la[j]) %←↩
          MOD;
9        }
10       b.insert(b.begin(), 0);
11       if (delta != 0) {
12          vector<int> t(max(la.size(), b.size()));
13          for (int i = 0; i < (int)t.size(); i++) {
14             if (i < (int)la.size()) t[i] = (t[i] + la[i←↩
          ]) % MOD;
15             if (i < (int)b.size()) t[i] = (t[i] - 1LL * ←↩
          delta * b[i] % MOD + MOD) % MOD;
16          }
17          if (2 * l <= r - 1) {
18             b = la;
19             int od = inv(delta);
20             for (int &x : b) x = 1LL * x * od % MOD;
21             l = r - l;
22          }
23          la = t;
24       }
25    }
26    assert((int)la.size() == l + 1);
27    assert(l * 2 + 30 < (int)s.size());
28    reverse(la.begin(), la.end());
29    return la;
30 }
31
32 vector<int> mul(vector<int> a, vector<int> b) {
33    vector<int> c(a.size() + b.size() - 1);
34    for (int i = 0; i < (int)a.size(); i++) {
35       for (int j = 0; j < (int)b.size(); j++) {
36          c[i + j] = (c[i + j] + 1LL * a[i] * b[j]) % ←↩
          MOD;
37       }
38    }
39    vector<int> res(c.size());
40    for (int i = 0; i < (int)res.size(); i++) res[i] =←↩
       c[i] % MOD;
41    return res;
42 }
43
44 vector<int> mod(vector<int> a, vector<int> b) {
45    if (a.size() < b.size()) a.resize(b.size() - 1);
46
47    int o = inv(b.back());
48    for (int i = (int)a.size() - 1; i >= (int)b.size()←↩
       - 1; i--) {
49       if (a[i] == 0) continue;
50       int coef = 1LL * o * (MOD - a[i]) % MOD;
51       for (int j = 0; j < (int)b.size(); j++) {
52          a[i - (int)b.size() + 1 + j] = (a[i - (int)b.←↩
          size() + 1 + j] + 1LL * coef * b[j]) % MOD;
53       }
54    }
55    while (a.size() >= b.size()) {
56       assert(a.back() == 0);
57       a.pop_back();
58    }
59    return a;
60 }
61
62 vector<int> bin(int n, vector<int> p) {
```

```
63     vector<int> res(1, 1);
64     vector<int> a(2); a[1] = 1;
65     while (n) {
66       if (n & 1) res = mod(mul(res, a), p);
67       a = mod(mul(a, a), p);
68       n >>= 1;
69     }
70     return res;
71   }
72
73   int f(vector<int> t, int m) {
74     vector<int> v = berlekamp(t);
75     vector<int> o = bin(m - 1, v);
76     int res = 0;
77     for (int i = 0; i < (int)o.size(); i++) res = (res
         + 1LL * o[i] * t[i]) % MOD;
78     return res;
79   }
```

final/numeric/integrate.cpp

```
1  function<dbl(dbl, dbl, function<dbl(dbl)>)> f = [&](
      dbl L, dbl R, function<dbl(dbl)> g) {
2    const int ITERS = 1000000;
3    dbl ans = 0;
4    dbl step = (R - L) * 1.0 / ITERS;
5    for (int it = 0; it < ITERS; it++) {
6      double xl = L + step * it;
7      double xr = L + step * (it + 1);
8      dbl x1 = (xl + xr) / 2;
9      dbl x0 = x1 - (x1 - xl) * sqrt(3.0 / 5);
10     dbl x2 = x1 + (x1 - xl) * sqrt(3.0 / 5);
11     ans += (5 * g(x0) + 8 * g(x1) + 5 * g(x2)) / 18
        * step;
12   }
13   return ans;
14 };
```

final/geom/commonTangents.cpp

```
1
2
3  vector<Line> commonTangents(pt A, dbl rA, pt B, dbl
      rB) {
4    vector<Line> res;
5    pt C = B - A;
6    dbl z = C.len2();
7    for (int i = -1; i <= 1; i += 2) {
8      for (int j = -1; j <= 1; j += 2) {
9        dbl r = rB * j - rA * i;
10       dbl d = z - r * r;
11       if (ls(d, 0)) continue;
12       d = sqrt(max(0.0l, d));
13       pt magic = pt(r, d) / z;
14       pt v(magic % C, magic * C);
15       dbl CC = (rA * i - v % A) / v.len2();
16       pt O = v * -CC;
17       res.pb(Line(O, O + v.rotate()));
18     }
19   }
20   return res;
21 }
22
23 // HOW TO USE ::
24 // --    *D*--------------*F*
25 // --    *...*-        -*...*
26 // --   *.....* -     - *.....*
27 // --   *.......* -  - *.......*
28 // --   *...A...*   --   *...B...*
29 // --   *.......* -  - *.......*
30 // --   *.....* -     - *.....*
31 // --    *...*-        -*...*
32 // --    *C*--------------*E*
33 // -- res = {CE, CF, DE, DF}
```

final/geom/halfplaneIntersection.cpp

```
1  int getPart(pt v) {
2    return less(0, v.y) || (equal(0, v.y) && less(v.x,
      0));
3  }
4
5  int cmpV(pt a, pt b) {
6    int partA = getPart(a);
7    int partB = getPart(b);
8    if (partA < partB) return -1;
9    if (partA > partB) return 1;
10   if (equal(0, a * b)) return 0;
11   if (0 < a * b) return -1;
12   return 1;
13 }
14
15 double planeInt(vector<Line> l) {
16   int n = l.size();
17   sort(all(l), [](Line a, Line b) {
18       int r = cmpV(a.v, b.v);
19       if (r != 0) return r < 0;
20       return a.O % a.v.rotate() < b.O % a.v.rotate()
      ;
21     });
22
23   int cur = 0;
24   for (int i = 0; i < n; ) {
25     int j = i;
26     for (; i < n && cmpV(l[j].v, l[i].v) == 0 &&
       cmpV(l[i].v, l[j].v) == 0; i++);
27     l[cur++] = l[i - 1];
28   }
29   n = cur;
30
31   for (int i = 0; i < n; i++)
32     l[i].id = i;
33
34   int flagUp = 0;
35   int flagDown = 0;
36   for (int i = 0; i < n; i++) {
37     int part = getPart(l[i].v);
38     if (part == 1) flagUp = 1;
39     if (part == 0) flagDown = 1;
40   }
41   if (!flagUp || !flagDown) return -1;
42
```

```
43     for (int i = 0; i < n; i++) {
44        pt v = l[i].v;
45        pt u = l[(i + 1) % n].v;
46        if (equal(0, v * u) && less(v % u, 0)) {
47           pt dir = l[i].v.rotate();
48           if (lessE(l[(i + 1) % n].O % dir, l[i].O % dir
        )) return 0;
49           return -1;
50        }
51        if (less(v * u, 0))
52           return -1;
53     }
54
55     cur = 0;
56     vector<Line> st(n * 2);
57     for (int tt = 0; tt < 2; tt++) {
58        for (int i = 0; i < n; i++) {
59           for (; cur >= 2; cur--) {
60              pt G = st[cur - 1] * l[i];
61              if (!lessE(st[cur - 2].v * (G - st[cur - 2].
        O), 0))
62                 break;
63           }
64           st[cur++] = l[i];
65           if (cur >= 2 && lessE(st[cur - 2].v * st[cur -
        1].v, 0)) return 0;
66        }
67     }
68     vector < int > use(n, -1);
69     int left = -1, right = -1;
70     for (int i = 0; i < cur; i++) {
71        if (use[st[i].id] == -1) {
72           use[st[i].id] = i;
73        }
74        else {
75           left = use[st[i].id];
76           right = i;
77           break;
78        }
79     }
80     vector < Line > tmp;
81     for (int i = left; i < right; i++)
82        tmp.pb(st[i]);
83     vector < pt > res;
84     for (int i = 0; i < (int)tmp.size(); i++)
85        res.pb(tmp[i] * tmp[(i + 1) % tmp.size()]);
86     double area = 0;
87     for (int i = 0; i < (int)res.size(); i++)
88        area += res[i] * res[(i + 1) % res.size()];
89     return area / 2;
90  }
```

final/geom/minDisc.cpp

```
1
2   pair<pt, dbl> minDisc(vector<pt> p) {
3      int n = p.size();
4      pt O = pt(0, 0);
5      dbl R = 0;
6      random_shuffle(all(p));
7      for (int i = 0; i < n; i++) {
8         if (ls(R, (O - p[i]).len())) {
9            O = p[i];
10           R = 0;
11           for (int j = 0; j < i; j++) {
12              if (ls(R, (O - p[j]).len())) {
13                 O = (p[i] + p[j]) / 2;
14                 R = (p[i] - p[j]).len() / 2;
15                 for (int k = 0; k < j; k++) {
16                    if (ls(R, (O - p[k]).len())) {
17                       Line l1((p[i] + p[j]) / 2, (p[i] + p[j
        ]) / 2 + (p[i] - p[j]).rotate());
18                       Line l2((p[k] + p[j]) / 2, (p[k] + p[j
        ]) / 2 + (p[k] - p[j]).rotate());
19                       O = l1 * l2;
20                       R = (p[i] - O).len();
21                    }
22                 }
23              }
24           }
25        }
26     }
27     return {O, R};
28  }
```

final/geom/convexHull3D-N2.cpp

```
1
2   struct Plane {
3      pt O, v;
4      vector<int> id;
5   };
6
7   vector<Plane> convexHull3(vector<pt> p) {
8      vector<Plane> res;
9      int n = p.size();
10     for (int i = 0; i < n; i++)
11        p[i].id = i;
12     for (int i = 0; i < 4; i++) {
13        vector<pt> tmp;
14        for (int j = 0; j < 4; j++)
15           if (i != j)
16              tmp.pb(p[j]);
17        res.pb({tmp[0], (tmp[1] - tmp[0]) * (tmp[2] -
        tmp[0]), {tmp[0].id, tmp[1].id, tmp[2].id}});
18        if ((p[i] - res.back().O) % res.back().v > 0) {
19           res.back().v = res.back().v * -1;
20           swap(res.back().id[0], res.back().id[1]);
21        }
22     }
23     vector<vector<int>> use(n, vector<int>(n, 0));
24     int tmr = 0;
25     for (int i = 4; i < n; i++) {
26        int cur = 0;
27        tmr++;
28        vector<pair<int,int>> curEdge;
29        for (int j = 0; j < sz(res); j++) {
30           if ((p[i] - res[j].O) % res[j].v > 0) {
31              for (int t = 0; t < 3; t++) {
32                 int v = res[j].id[t];
33                 int u = res[j].id[(t + 1) % 3];
34                 use[v][u] = tmr;
35                 curEdge.pb({v, u});
36              }
37           }
38           else {
39              res[cur++] = res[j];
40           }
41        }
42        res.resize(cur);
43        for (auto x: curEdge) {
44           if (use[x.S][x.F] == tmr) continue;
45           res.pb({p[i], (p[x.F] - p[i]) * (p[x.S] - p[i
        ]), {x.F, x.S, i}});
46        }
47     }
48     return res;
49  }
50
51  // plane in 3d
52  //(A, v) * (B, u) -> (O, n)
53
54  pt n = v * u;
55  pt m = v * n;
56  double t = (B - A) % u / (u % m);
57  pt O = A - m * t;
```

final/geom/polygonArcCut.cpp

```
1
2   struct Meta {
3      int type; // 0 - seg, 1 - circle
4      pt O;
5      dbl R;
6   };
7
8   const Meta SEG = {0, pt(0, 0), 0};
9
10  vector<pair<pt, Meta>> cut(vector<pair<pt, Meta>> p,
        Line l) {
11     vector<pair<pt,Meta>> res;
12     int n = p.size();
13     for (int i = 0; i < n; i++) {
14        pt A = p[i].F;
15        pt B = p[(i + 1) % n].F;
16        if (le(0, l.v * (A - l.O))) {
17           if (eq(0, l.v * (A - l.O)) && p[i].S.type == 1
        && ls(0, l.v % (p[i].S.O - A)))
18              res.pb({A, SEG});
```

```
19          else
20            res.pb(p[i]);
21        }
22        if (p[i].S.type == 0) {
23          if (sign(l.v * (A - l.O)) * sign(l.v * (B - l.↵
      O)) == -1) {
24            pt FF = Line(A, B) * l;
25            res.pb(make_pair(FF, SEG));
26          }
27        }
28        else {
29          pt E, F;
30          if (intCL(p[i].S.O, p[i].S.R, l, E, F)) {
31            if (onArc(p[i].S.O, A, E, B))
32              res.pb({E, SEG});
33            if (onArc(p[i].S.O, A, F, B))
34              res.pb({F, p[i].S});
35          }
36        }
37      }
38      return res;
39    }
```

final/geom/polygonTangent.cpp

```
1  pt tangent(vector<pt>& p, pt O, int cof) {
2      int step = 1;
3      for (; step < (int)p.size(); step *= 2);
4      int pos = 0;
5      int n = p.size();
6      for (; step > 0; step /= 2) {
7        int best = pos;
8        for (int dx = -1; dx <= 1; dx += 2) {
9          int id = ((pos + step * dx) % n + n) % n;
10         if ((p[id] - O) * (p[best] - O) * cof > 0)
11           best = id;
12       }
13       pos = best;
14     }
15     return p[pos];
16   }
```

final/geom/checkPlaneInt.cpp

```
1
2  bool checkPoint(vector<Line> l, pt& ret) {
3      random_shuffle(all(l));
4      pt A = l[0].O;
5      for (int i = 1; i < sz(l); i++) {
6        if (!le(0, l[i].v * (A - l[i].O))) {
7          dbl mn = -INF;
8          dbl mx = INF;
9          for (int j = 0; j < i; j++) {
10           if (eq(l[j].v * l[i].v, 0)) {
11             if (l[j].v % l[i].v < 0 && (l[j].O - l[i].↵
      O) % l[i].v.rotate() <= 0) {
12               return false;
13             }
14           }
15           else {
16             pt u = l[j].v.rotate();
17             dbl proj = (l[j].O - l[i].O) % u / (l[i].v↵
      % u);
18             if (l[i].v * l[j].v > 0) {
19               mx = min(mx, proj);
20             }
21             else {
22               mn = max(mn, proj);
23             }
24           }
25         }
26         if (mn <= mx) {
27           A = l[i].O + l[i].v * mn;
28         }
29         else {
30           return false;
31         }
32       }
33     }
34     ret = A;
35     return true;
```

```
36  }
```

final/geom/furthestPoints.cpp

```
1  ll furthestPoints(vector<pt> p) {
2      int n = p.size();
3      int cur = 1;
4      ll answer = 0;
5      for (int i = 0; i < n; i++) {
6        for (; (p[(i + 1) % n] - p[i]) * (p[(cur + 1) % ↵
      n] - p[cur]) > 0; cur = (cur + 1) % n);
7        answer = max(answer, (p[i] - p[cur]).len2());
8      }
9      return answer;
10   }
```

final/strings/eertree.cpp

```
1  namespace eertree {
2      const int INF = 1e9;
3      const int N = 5e6 + 10;
4      char _s[N];
5      char *s = _s + 1;
6      int to[N][2];
7      int suf[N], len[N];
8      int sz, last;
9
10     const int odd = 1, even = 2, blank = 3;
11
12     void go(int &u, int pos) {
13       while (u != blank && s[pos - len[u] - 1] != s[↵
      pos]) {
14         u = suf[u];
15       }
16     }
17
18     int add(int pos) {
19       go(last, pos);
20       int u = suf[last];
21       go(u, pos);
22       int c = s[pos] - 'a';
23       int res = 0;
24       if (!to[last][c]) {
25         res = 1;
26         to[last][c] = sz;
27         len[sz] = len[last] + 2;
28         suf[sz] = to[u][c];
29         sz++;
30       }
31       last = to[last][c];
32       return res;
33     }
34
35     void init() {
36       to[blank][0] = to[blank][1] = even;
37       len[blank] = suf[blank] = INF;
38       len[even] = 0, suf[even] = odd;
39       len[odd] = -1, suf[odd] = blank;
40       last = even;
41       sz = 4;
42     }
43  }
```

final/strings/sufAutomaton.cpp

```
1  namespace SA {
2      const int MAXN = 1 << 18;
3      const int SIGMA = 26;
4
5      int sz, last;
6      int nxt[MAXN][SIGMA];
7      int link[MAXN], len[MAXN], pos[MAXN];
8
9      void init() {
10       memset(nxt, -1, sizeof(nxt));
```

```
11      memset(link, -1, sizeof(link));
12      memset(len, 0, sizeof(len));
13      last = 0;
14      sz = 1;
15    }
16
17    void add(int c) {
18      int cur = sz++;
19      len[cur] = len[last] + 1;
20      pos[cur] = len[cur];
21      int p = last;
22      last = cur;
23      for (; p != -1 && nxt[p][c] == -1; p = link[p]) ←
            nxt[p][c] = cur;
24      if (p == -1) {
25        link[cur] = 0;
26        return;
27      }
28      int q = nxt[p][c];
29      if (len[p] + 1 == len[q]) {
30        link[cur] = q;
31        return;
32      }
33      int clone = sz++;
34      memcpy(nxt[clone], nxt[q], sizeof(nxt[q]));
35      len[clone] = len[p] + 1;
36      pos[clone] = pos[q];
37      link[clone] = link[q];
38      link[q] = link[cur] = clone;
39      for (; p != -1 && nxt[p][c] == q; p = link[p]) ←
            nxt[p][c] = clone;
40    }
41
42    int n;
43    string s;
44    int l[MAXN], r[MAXN];
45    int e[MAXN][SIGMA];
46
47    void getSufTree(string _s) {
48      memset(e, -1, sizeof(e));
49      s = _s;
50      n = s.length();
51      reverse(s.begin(), s.end());
52      init();
53      for (int i = 0; i < n; i++) add(s[i] - 'a');
54      reverse(s.begin(), s.end());
55      for (int i = 1; i < sz; i++) {
56        int j = link[i];
57        l[i] = n - pos[i] + len[j];
58        r[i] = n - pos[i] + len[i];
59        e[j][s[l[i]] - 'a'] = i;
60      }
61    }
62  }
```

```
26      cn[p[0]] = 0;
27      for (int i = 1; i < n; i++) {
28        cl += c[p[i]] != c[p[i - 1]] || c[(p[i] + len)←
            % n] != c[(p[i - 1] + len) % n];
29        cn[p[i]] = cl - 1;
30      }
31      for (int i = 0; i < n; i++) c[i] = cn[i];
32    }
33
34    for (int i = 0; i < n; i++) o[p[i]] = i;
35
36    int z = 0;
37    for (int i = 0; i < n; i++) {
38      int j = o[i];
39      if (j == n - 1) {
40        z = 0;
41      } else {
42        while (s[i + z] == s[p[j + 1] + z]) z++;
43      }
44      lcp[j] = z;
45      z -= !!z;
46    }
47  }
```

final/strings/duval.cpp

```
1  void duval(string s) {
2    int n = (int) s.length();
3    int i=0;
4    while (i < n) {
5      int j=i+1, k=i;
6      while (j < n && s[k] <= s[j]) {
7        if (s[k] < s[j])
8          k = i;
9        else
10         ++k;
11       ++j;
12     }
13     while (i <= k) {
14       cout << s.substr(i, j-k) << ' ';
15       i += j - k;
16     }
17   }
18 }
```

final/strings/sufArray.cpp

```
1  int n;
2  char s[N];
3  int p[N], pn[N], c[N], cn[N], cnt[N];
4  int o[N];
5  int lcp[N];
6
7  void build() {
8    for (int i = 0; i < 256; i++) cnt[i] = 0;
9    for (int i = 0; i < n; i++) cnt[(int)s[i]]++;
10   for (int i = 1; i < 256; i++) cnt[i] += cnt[i - ←
          1];
11   for (int i = n - 1; i >= 0; i--) p[--cnt[(int)s[i←
          ]]] = i;
12   int cl = 1;
13   c[p[0]] = 0;
14   for (int i = 1; i < n; i++) {
15     cl += s[p[i]] != s[p[i - 1]];
16     c[p[i]] = cl - 1;
17   }
18
19   for (int len = 1; len < n; len <<= 1) {
20     for (int i = 0; i < cl; i++) cnt[i] = 0;
21     for (int i = 0; i < n; i++) cnt[c[i]]++;
22     for (int i = 1; i < cl; i++) cnt[i] += cnt[i - ←
            1];
23     for (int i = 0; i < n; i++) pn[i] = (p[i] - len ←
          + n) % n;
24     for (int i = n - 1; i >= 0; i--) p[--cnt[c[pn[i←
          ]]]] = pn[i];
25     cl = 1;
```

11

final/graphs/centroid.cpp

```
1   // original author: burunduk1, rewritten by me (←
        enot110)
2   // !!! warning !!! this code is not tested well
3   const int N = 1e5, K = 17;
4
5   int pivot, level[N], parent[N];
6   vector<int> v[N];
7
8   int get_pivot( int x, int xx, int n ) {
9     int size = 1;
10    for (int y : v[x])
11    {
12      if (y != xx && level[y] == -1) size += get_pivot←
        (y, x, n);
13    }
14    if (pivot == -1 && (size * 2 >= n || xx == -1)) ←
        pivot = x;
15    return size;
16  }
17
18  void build( int x, int xx, int dep, int size ) {
19    assert(dep < K);
20    pivot = -1;
21    get_pivot(x, -1, size);
22    x = pivot;
23    level[x] = dep, parent[x] = xx;
24    for (int y : v[x]) if (level[y] == -1)
25    {
26      build(y, x, dep + 1, size / 2);
27    }
28  }
```

final/graphs/dominatorTree.cpp

```
1   namespace domtree {
2     const int K = 18;
3     const int N = 1 << K;
4
5     int n, root;
6     vector<int> e[N], g[N];
7     int sdom[N], dom[N];
8     int p[N][K], h[N], pr[N];
9     int in[N], out[N], tmr, rev[N];
10
11    void init(int _n, int _root) {
12      n = _n;
13      root = _root;
14      tmr = 0;
15      for (int i = 0; i < n; i++) {
16        e[i].clear();
17        g[i].clear();
18        in[i] = -1;
19      }
20    }
21
22    void addEdge(int u, int v) {
23      e[u].push_back(v);
24      g[v].push_back(u);
25    }
26
27    void dfs(int v) {
28      in[v] = tmr++;
29      for (int to : e[v]) {
30        if (in[to] != -1) continue;
31        pr[to] = v;
32        dfs(to);
33      }
34      out[v] = tmr - 1;
35    }
36
37    int lca(int u, int v) {
38      if (h[u] < h[v]) swap(u, v);
39      for (int i = 0; i < K; i++) if ((h[u] - h[v]) & ←
        (1 << i)) u = p[u][i];
40      if (u == v) return u;
41      for (int i = K - 1; i >= 0; i--) {
42        if (p[u][i] != p[v][i]) {
43          u = p[u][i];
44          v = p[v][i];
45        }
46      }
47      return p[u][0];
```

```
48    }
49
50    void solve(int _n, int _root, vector<pair<int, int←
        >> _edges) {
51      init(_n, _root);
52      for (auto ed : _edges) addEdge(ed.first, ed.←
        second);
53
54      dfs(root);
55      for (int i = 0; i < n; i++) if (in[i] != -1) rev←
        [in[i]] = i;
56      segtree tr(tmr); // a[i]:=min(a[i],x) and return←
        a[i]
57      for (int i = tmr - 1; i >= 0; i--) {
58        int v = rev[i];
59        int cur = i;
60        for (int to : g[v]) {
61          if (in[to] == -1) continue;
62          if (in[to] < in[v]) cur = min(cur, in[to]);
63          else cur = min(cur, tr.get(in[to]));
64        }
65        sdom[v] = rev[cur];
66        tr.upd(in[v], out[v], in[sdom[v]]);
67      }
68      for (int i = 0; i < tmr; i++) {
69        int v = rev[i];
70        if (i == 0) {
71          dom[v] = v;
72          h[v] = 0;
73        } else {
74          dom[v] = lca(sdom[v], pr[v]);
75          h[v] = h[dom[v]] + 1;
76        }
77        p[v][0] = dom[v];
78        for (int j = 1; j < K; j++) p[v][j] = p[p[v][j←
        - 1]][j - 1];
79      }
80      for (int i = 0; i < n; i++) if (in[i] == -1) dom←
        [i] = -1;
81    }
82  }
```

final/graphs/generalMatching.cpp

```
1   //COPYPASTED FROM E-MAXX
2   namespace GeneralMatching {
3     const int MAXN = 256;
4     int n;
5     vector<int> g[MAXN];
6     int match[MAXN], p[MAXN], base[MAXN], q[MAXN];
7     bool used[MAXN], blossom[MAXN];
8
9     int lca (int a, int b) {
10      bool used[MAXN] = { 0 };
11      for (;;) {
12        a = base[a];
13        used[a] = true;
14        if (match[a] == -1) break;
15        a = p[match[a]];
16      }
17      for (;;) {
18        b = base[b];
19        if (used[b]) return b;
20        b = p[match[b]];
21      }
22    }
23
24    void mark_path (int v, int b, int children) {
25      while (base[v] != b) {
26        blossom[base[v]] = blossom[base[match[v]]] = ←
        true;
27        p[v] = children;
28        children = match[v];
29        v = p[match[v]];
30      }
31    }
32
33    int find_path (int root) {
34      memset (used, 0, sizeof used);
35      memset (p, -1, sizeof p);
36      for (int i=0; i<n; ++i)
37        base[i] = i;
38
39      used[root] = true;
40      int qh=0, qt=0;
41      q[qt++] = root;
42      while (qh < qt) {
```

```
43        int v = q[qh++];
44        for (size_t i=0; i<g[v].size(); ++i) {
45          int to = g[v][i];
46          if (base[v] == base[to] || match[v] == to)
               continue;
47          if (to == root || (match[to] != -1 && p[
               match[to]] != -1)) {
48            int curbase = lca (v, to);
49            memset (blossom, 0, sizeof blossom);
50            mark_path (v, curbase, to);
51            mark_path (to, curbase, v);
52            for (int i=0; i<n; ++i)
53              if (blossom[base[i]]) {
54                base[i] = curbase;
55                if (!used[i]) {
56                  used[i] = true;
57                  q[qt++] = i;
58                }
59              }
60          }
61          else if (p[to] == -1) {
62            p[to] = v;
63            if (match[to] == -1)
64              return to;
65            to = match[to];
66            used[to] = true;
67            q[qt++] = to;
68          }
69        }
70      }
71      return -1;
72    }
73
74    vector<pair<int, int> > solve(int _n, vector<pair<
         int, int> > edges) {
75      n = _n;
76      for (int i = 0; i < n; i++) g[i].clear();
77      for (auto o : edges) {
78        g[o.first].push_back(o.second);
79        g[o.second].push_back(o.first);
80      }
81      memset (match, -1, sizeof match);
82      for (int i=0; i<n; ++i) {
83        if (match[i] == -1) {
84          int v = find_path (i);
85          while (v != -1) {
86            int pv = p[v], ppv = match[pv];
87            match[v] = pv, match[pv] = v;
88            v = ppv;
89          }
90        }
91      }
92      vector<pair<int, int> > ans;
93      for (int i = 0; i < n; i++) {
94        if (match[i] > i) {
95          ans.push_back(make_pair(i, match[i]));
96        }
97      }
98      return ans;
99    }
100 }
```

final/graphs/heavyLight.cpp

```
1  namespace hld {
2    const int N = 1 << 17;
3    int par[N], heavy[N], h[N];
4    int root[N], pos[N];
5    int n;
6    vector<vector<int> > e;
7    segtree tree;
8
9    int dfs(int v) {
10     int sz = 1, mx = 0;
11     for (int to : e[v]) {
12       if (to == par[v]) continue;
13       par[to] = v;
14       h[to] = h[v] + 1;
15       int cur = dfs(to);
16       if (cur > mx) heavy[v] = to, mx = cur;
17       sz += cur;
18     }
19     return sz;
20   }
21
22   template <typename T>
23   void path(int u, int v, T op) {
```

```
24       for (; root[u] != root[v]; v = par[root[v]]) {
25         if (h[root[u]] > h[root[v]]) swap(u, v);
26         op(pos[root[v]], pos[v] + 1);
27       }
28       if (h[u] > h[v]) swap(u, v);
29       op(pos[u], pos[v] + 1);
30     }
31
32     void init(vector<vector<int> > _e) {
33       e = _e;
34       n = e.size();
35       tree = segtree(n);
36       memset(heavy, -1, sizeof(heavy[0]) * n);
37       par[0] = -1;
38       h[0] = 0;
39       dfs(0);
40       for (int i = 0, cpos = 0; i < n; i++) {
41         if (par[i] == -1 || heavy[par[i]] != i) {
42           for (int j = i; j != -1; j = heavy[j]) {
43             root[j] = i;
44             pos[j] = cpos++;
45           }
46         }
47       }
48     }
49
50     void add(int v, int x) {
51       tree.add(pos[v], x);
52     }
53
54     int get(int u, int v) {
55       int res = 0;
56       path(u, v, [&](int l, int r) {
57         res = max(res, tree.get(l, r));
58       });
59       return res;
60     }
61 }
```

final/graphs/hungary.cpp

```
1  namespace hungary
2  {
3    const int N = 210;
4
5    int a[N][N];
6    int ans[N];
7
8    int calc(int n, int m)
9    {
10     ++n, ++m;
11     vi u(n), v(m), p(m), prev(m);
12     for (int i = 1; i < n; ++i)
13     {
14       p[0] = i;
15       int x = 0;
16       vi mn(m, inf);
17       vi was(m, 0);
18       while (p[x])
19       {
20         was[x] = 1;
21         int ii = p[x], dd = inf, y = 0;
22         for (int j = 1; j < m; ++j) if (!was[j])
23         {
24           int cur = a[ii][j] - u[ii] - v[j];
25           if (cur < mn[j]) mn[j] = cur, prev[j] = x;
26           if (mn[j] < dd) dd = mn[j], y = j;
27         }
28         forn(j, m)
29         {
30           if (was[j]) u[p[j]] += dd, v[j] -= dd;
31           else mn[j] -= dd;
32         }
33         x = y;
34       }
35       while (x)
36       {
37         int y = prev[x];
38         p[x] = p[y];
39         x = y;
40       }
41     }
42     for (int j = 1; j < m; ++j)
43     {
44       ans[p[j]] = j;
45     }
46     return -v[0];
```

```
47      }
48      // HOW TO USE ::
49      // -- set values to a[1..n][1..m] (n <= m)
50      // -- run calc(n, m) to find MINIMUM
51      // -- to restore permutation use ans[]
52      // -- everything works on negative numbers
53      //
54      // !! i don't understand this code, it's ←
        copypasted from e-maxx (and rewrited by enot110←
        )
55  }
```

final/graphs/minCostNegCycle.cpp

```
1   struct Edge {
2     int from, to, cap, flow;
3     double cost;
4
5   };
6
7   struct Graph {
8     int n;
9     vector<Edge> edges;
10    vector<vector<int> > e;
11
12    Graph(int _n) {
13      n = _n;
14      e.resize(n);
15    }
16
17    void addEdge(int from, int to, int cap, double ←
        cost) {
18      e[from].push_back(edges.size());
19      edges.push_back({ from, to, cap, 0, cost });
20      e[to].push_back(edges.size());
21      edges.push_back({ to, from, 0, 0, -cost });
22    }
23
24    void maxflow() {
25      while (1) {
26        queue<int> q;
27        vector<int> d(n, INF);
28        vector<int> pr(n, -1);
29        q.push(0);
30        d[0] = 0;
31        while (!q.empty()) {
32          int v = q.front();
33          q.pop();
34          for (int i = 0; i < (int)e[v].size(); i++) {
35            Edge cur = edges[e[v][i]];
36            if (d[cur.to] > d[v] + 1 && cur.flow < cur←
                .cap) {
37              d[cur.to] = d[v] + 1;
38              pr[cur.to] = e[v][i];
39              q.push(cur.to);
40            }
41          }
42        }
43        if (d[n - 1] == INF) break;
44        int v = n - 1;
45        while (v) {
46          edges[pr[v]].flow++;
47          edges[pr[v] ^ 1].flow--;
48          v = edges[pr[v]].from;
49        }
50      }
51    }
52
53    bool findcycle() {
54      int iters = n;
55      vector<int> changed;
56      for (int i = 0; i < n; i++) changed.push_back(i)←
          ;
57
58      vector<vector<double> > d(iters + 1, vector<←
          double>(n, INF));
59      vector<vector<int> > p(iters + 1, vector<int>(n,←
          -1));
60      d[0].assign(n, 0);
61      for (int it = 0; it < iters; it++) {
62        d[it + 1] = d[it];
63        vector<int> nchanged(n, 0);
64        for (int v : changed) {
65          for (int id : e[v]) {
66            Edge cur = edges[id];
67            if (d[it + 1][cur.to] > d[it][v] + cur.←
                cost && cur.flow < cur.cap) {
```

```
68              d[it + 1][cur.to] = d[it][v] + cur.cost;
69              p[it + 1][cur.to] = id;
70              nchanged[cur.to] = 1;
71            }
72          }
73        }
74        changed.clear();
75        for (int i = 0; i < n; i++) if (nchanged[i]) ←
            changed.push_back(i);
76      }
77      if (changed.empty()) return 0;
78
79      int bestU = 0, bestK = 1;
80      double bestAns = INF;
81      for (int u = 0; u < n; u++) {
82        double curMax = -INF;
83        for (int k = 0; k < iters; k++) {
84          double curVal = (d[iters][u] - d[k][u]) / (←
              iters - k);
85          curMax = max(curMax, curVal);
86        }
87        if (bestAns > curMax) {
88          bestAns = curMax;
89          bestU = u;
90        }
91      }
92
93      int v = bestU;
94      int it = iters;
95      vector<int> was(n, -1);
96      while (was[v] == -1) {
97        was[v] = it;
98        v = edges[p[it][v]].from;
99        it--;
100     }
101     int vv = v;
102     it = was[v];
103     double sum = 0;
104     do {
105       edges[p[it][v]].flow++;
106       sum += edges[p[it][v]].cost;
107       edges[p[it][v] ^ 1].flow--;
108       v = edges[p[it][v]].from;
109       it--;
110     } while (v != vv);
111     return 1;
112    }
113  };
```

final/graphs/retro.cpp

```
1   namespace retro
2   {
3     const int N = 4e5 + 10;
4
5     vi v[N];
6     vi vrev[N];
7
8     void add(int x, int y)
9     {
10      v[x].pb(y);
11      vrev[y].pb(x);
12    }
13
14    const int UD = 0;
15    const int WIN = 1;
16    const int LOSE = 2;
17
18    int res[N];
19    int moves[N];
20    int deg[N];
21    int q[N], st, en;
22
23    void calc(int n)
24    {
25      forn(i, n) deg[i] = sz(v[i]);
26      st = en = 0;
27      forn(i, n) if (!deg[i])
28      {
29        q[en++] = i;
30        res[i] = LOSE;
31      }
32      while (st < en)
33      {
34        int x = q[st++];
35        for (int y : vrev[x])
36        {
```

```
37        if (res[y] == UD && (res[x] == LOSE || (--↩
      deg[y] == 0 && res[x] == WIN)))
38        {
39            res[y] = 3 - res[x];
40            moves[y] = moves[x] + 1;
41            q[en++] = y;
42        }
43      }
44    }
45  }
46 }
```

final/graphs/smith.cpp

```
1  const int N = 1e5 + 10;
2
3  struct graph
4  {
5    int n;
6
7    vi v[N];
8    vi vrev[N];
9
10   void read()
11   {
12     int m;
13     scanf("%d%d", &n, &m);
14     forn(i, m)
15     {
16       int x, y;
17       scanf("%d%d", &x, &y);
18       --x, --y;
19       v[x].pb(y);
20       vrev[y].pb(x);
21     }
22   }
23
24   int deg[N], cnt[N], used[N], f[N];
25   int q[N], st, en;
26
27   set<int> s[N];
28
29   void calc()
30   {
31     forn(x, n) f[x] = -1, cnt[x] = 0;
32     int val = 0;
33     while (1)
34     {
35       st = en = 0;
36       forn(x, n)
37       {
38         deg[x] = 0;
39         used[x] = 0;
40         for (int y : v[x]) if (f[y] == -1) deg[x]++;
41       }
42       forn(x, n) if (!deg[x] && f[x] == -1 && cnt[x]↩
      == val)
43       {
44         q[en++] = x;
45         f[x] = val;
46       }
47       if (!en) break;
48       while (st < en)
49       {
50         int x = q[st];
51         st++;
52         for (int y : vrev[x])
53         {
54           if (used[y] == 0 && f[y] == -1)
55           {
56             used[y] = 1;
57             cnt[y]++;
58             for (int z : vrev[y])
59             {
60               deg[z]--;
61               if (f[z] == -1 && deg[z] == 0 && cnt[z↩
      ] == val)
62               {
63                 f[z] = val;
64                 q[en++] = z;
65               }
66             }
67           }
68         }
69       }
70       val++;
71     }
```

```
72     forn(x, n) eprintf("%d%c", f[x], " \n"[x + 1 == ↩
      n]);
73     forn(x, n) if (f[x] == -1)
74     {
75       for (int y : v[x]) if (f[y] != -1) s[x].insert↩
      (f[y]);
76     }
77   }
78 } g1, g2;
79
80 string get(int x, int y)
81 {
82   int f1 = g1.f[x], f2 = g2.f[y];
83   if (f1 == -1 && f2 == -1) return "draw";
84   if (f1 == -1) {
85     if (g1.s[x].count(f2)) return "first";
86     return "draw";
87   }
88   if (f2 == -1) {
89     if (g2.s[y].count(f1)) return "first";
90     return "draw";
91   }
92   if (f1 ^ f2) return "first";
93   return "second";
94 }
```

final/graphs/mincut.cpp

```
1  const int MAXN = 500;
2  int n, g[MAXN][MAXN];
3  int best_cost = 1000000000;
4  vector<int> best_cut;
5
6  void mincut() {
7    vector<int> v[MAXN];
8    for (int i=0; i<n; ++i)
9      v[i].assign(1, i);
10   int w[MAXN];
11   bool exist[MAXN], in_a[MAXN];
12   memset(exist, true, sizeof exist);
13   for (int ph=0; ph<n-1; ++ph) {
14     memset(in_a, false, sizeof in_a);
15     memset(w, 0, sizeof w);
16     for (int it=0, prev; it<n-ph; ++it) {
17       int sel = -1;
18       for (int i=0; i<n; ++i)
19         if (exist[i] && !in_a[i] && (sel == -1 || w[↩
            i] > w[sel]))
20           sel = i;
21       if (it == n-ph-1) {
22         if (w[sel] < best_cost)
23           best_cost = w[sel], best_cut = v[sel];
24         v[prev].insert(v[prev].end(), v[sel].begin↩
            (), v[sel].end());
25         for (int i=0; i<n; ++i)
26           g[prev][i] = g[i][prev] += g[sel][i];
27         exist[sel] = false;
28       }
29       else {
30         in_a[sel] = true;
31         for (int i=0; i<n; ++i)
32           w[i] += g[sel][i];
33         prev = sel;
34       }
35     }
36   }
37 }
```

final/graphs/twoChineseFast.cpp

```
1  namespace twoc {
2    struct Heap {
3      static Heap* null;
4      ll x, xadd;
5      int ver, h;
6      /* ANS */ int ei;
7      Heap *l, *r;
8      Heap(ll xx, int vv) : x(xx), xadd(0), ver(vv), h↩
        (1), l(null), r(null) {}
9      Heap(const char*) : x(0), xadd(0), ver(0), h(0),↩
         l(this), r(this) {}
```

```
10      void add(ll a) { x += a; xadd += a; }
11      void push() {
12        if (l != null) l->add(xadd);
13        if (r != null) r->add(xadd);
14        xadd = 0;
15      }
16    };
17    Heap *Heap::null = new Heap("wqeqw");
18    Heap* merge(Heap *l, Heap *r) {
19      if (l == Heap::null) return r;
20      if (r == Heap::null) return l;
21      l->push(); r->push();
22      if (l->x > r->x)
23        swap(l, r);
24      l->r = merge(l->r, r);
25      if (l->l->h < l->r->h)
26        swap(l->l, l->r);
27      l->h = l->r->h + 1;
28      return l;
29    }
30    Heap *pop(Heap *h) {
31      h->push();
32      return merge(h->l, h->r);
33    }
34    const int N = 666666;
35    struct DSU {
36      int p[N];
37      void init(int nn) { iota(p, p + nn, 0); }
38      int get(int x) { return p[x] == x ? x : p[x] = ↩
      get(p[x]); }
39      void merge(int x, int y) { p[get(y)] = get(x); }
40    } dsu;
41    Heap *eb[N];
42    int n;
43    /* ANS */ struct Edge {
44    /* ANS */   int x, y;
45    /* ANS */   ll c;
46    /* ANS */ };
47    /* ANS */ vector<Edge> edges;
48    /* ANS */ int answer[N];
49    void init(int nn) {
50      n = nn;
51      dsu.init(n);
52      fill(eb, eb + n, Heap::null);
53      edges.clear();
54    }
55    void addEdge(int x, int y, ll c) {
56      Heap *h = new Heap(c, x);
57      /* ANS */ h->ei = sz(edges);
58      /* ANS */ edges.push_back({x, y, c});
59      eb[y] = merge(eb[y], h);
60    }
61    ll solve(int root = 0) {
62      ll ans = 0;
63      static int done[N], pv[N];
64      memset(done, 0, sizeof(int) * n);
65      done[root] = 1;
66      int tt = 1;
67      /* ANS */ int cnum = 0;
68      /* ANS */ static vector<ipair> eout[N];
69      /* ANS */ for (int i = 0; i < n; ++i) eout[i].↩
      clear();
70      for (int i = 0; i < n; ++i) {
71        int v = dsu.get(i);
72        if (done[v])
73          continue;
74        ++tt;
75        while (true) {
76          done[v] = tt;
77          int nv = -1;
78          while (eb[v] != Heap::null) {
79            nv = dsu.get(eb[v]->ver);
80            if (nv == v) {
81              eb[v] = pop(eb[v]);
82              continue;
83            }
84            break;
85          }
86          if (nv == -1)
87            return LINF;
88          ans += eb[v]->x;
89          eb[v]->add(-eb[v]->x);
90          /* ANS */ int ei = eb[v]->ei;
91          /* ANS */ eout[edges[ei].x].push_back({++↩
      cnum, ei});
92          if (!done[nv]) {
93            pv[v] = nv;
94            v = nv;
95            continue;
96          }
97          if (done[nv] != tt)
98            break;
99          int v1 = nv;
```

```
100        while (v1 != v) {
101          eb[v] = merge(eb[v], eb[v1]);
102          dsu.merge(v, v1);
103          v1 = dsu.get(pv[v1]);
104        }
105      }
106    }
107    /* ANS */ memset(answer, -1, sizeof(int) * n);
108    /* ANS */ answer[root] = 0;
109    /* ANS */ set<ipair> es(all(eout[root]));
110    /* ANS */ while (!es.empty()) {
111    /* ANS */   auto it = es.begin();
112    /* ANS */   int ei = it->second;
113    /* ANS */   es.erase(it);
114    /* ANS */   int nv = edges[ei].y;
115    /* ANS */   if (answer[nv] != -1)
116    /* ANS */     continue;
117    /* ANS */   answer[nv] = ei;
118    /* ANS */   es.insert(all(eout[nv]));
119    /* ANS */ }
120    /* ANS */ answer[root] = -1;
121    return ans;
122  }
123  /* Usage: twoc::init(vertex_count);
124   *    twoc::addEdge(v1, v2, cost);
125   *    twoc::solve(root); - returns cost or LINF
126   * twoc::answer contains index of ingoing edge for↩
      each vertex
127   */
128 }
```

final/graphs/linkcut.cpp

```
1  #include <iostream>
2  #include <cstdio>
3  #include <cassert>
4
5  using namespace std;
6
7  // BEGIN ALGO
8
9  const int MAXN = 110000;
10
11 typedef struct _node{
12   _node *l, *r, *p, *pp;
13   int size; bool rev;
14   _node();
15   explicit _node(nullptr_t){
16     l = r = p = pp = this;
17     size = rev = 0;
18   }
19   void push(){
20     if (rev){
21       l->rev ^= 1; r->rev ^= 1;
22       rev = 0; swap(l,r);
23     }
24   }
25   void update();
26 }* node;
27 node None = new _node(nullptr);
28 node v2n[MAXN];
29 _node::_node(){
30   l = r = p = pp = None;
31   size = 1; rev = false;
32 }
33 void _node::update(){
34   size = (this != None) + l->size + r->size;
35   l->p = r->p = this;
36 }
37 void rotate(node v){
38   assert(v != None && v->p != None);
39   assert(!v->rev); assert(!v->p->rev);
40   node u = v->p;
41   if (v == u->l)
42     u->l = v->r, v->r = u;
43   else
44     u->r = v->l, v->l = u;
45   swap(u->p,v->p); swap(v->pp,u->pp);
46   if (v->p != None){
47     assert(v->p->l == u || v->p->r == u);
48     if (v->p->r == u) v->p->r = v;
49     else v->p->l = v;
50   }
51   u->update(); v->update();
52 }
53 void bigRotate(node v){
54   assert(v->p != None);
```

```
55   v->p->p->push();
56   v->p->push();
57   v->push();
58   if (v->p->p != None){
59     if ((v->p->l == v) ^ (v->p->p->r == v->p))
60       rotate(v->p);
61     else
62       rotate(v);
63   }
64   rotate(v);
65 }
66 inline void Splay(node v){
67   while (v->p != None) bigRotate(v);
68 }
69 inline void splitAfter(node v){
70   v->push();
71   Splay(v);
72   v->r->p = None;
73   v->r->pp = v;
74   v->r = None;
75   v->update();
76 }
77 void expose(int x){
78   node v = v2n[x];
79   splitAfter(v);
80   while (v->pp != None){
81     assert(v->p == None);
82     splitAfter(v->pp);
83     assert(v->pp->r == None);
84     assert(v->pp->p == None);
85     assert(!v->pp->rev);
86     v->pp->r = v;
87     v->pp->update();
88     v = v->pp;
89     v->r->pp = None;
90   }
91   assert(v->p == None);
92   Splay(v2n[x]);
93 }
94 inline void makeRoot(int x){
95   expose(x);
96   assert(v2n[x]->p == None);
97   assert(v2n[x]->pp == None);
98   assert(v2n[x]->r == None);
99   v2n[x]->rev ^= 1;
100 }
101 inline void link(int x,int y){
102   makeRoot(x); v2n[x]->pp = v2n[y];
103 }
104 inline void cut(int x,int y){
105   expose(x);
106   Splay(v2n[y]);
107   if (v2n[y]->pp != v2n[x]){
108     swap(x,y);
109     expose(x);
110     Splay(v2n[y]);
111     assert(v2n[y]->pp == v2n[x]);
112   }
113   v2n[y]->pp = None;
114 }
115 inline int get(int x,int y){
116   if (x == y) return 0;
117   makeRoot(x);
118   expose(y); expose(x);
119   Splay(v2n[y]);
120   if (v2n[y]->pp != v2n[x]) return -1;
121   return v2n[y]->size;
122 }
123 // END ALGO
124
125 _node mem[MAXN];
126
127
128 int main(){
129   freopen("linkcut.in","r",stdin);
130   freopen("linkcut.out","w",stdout);
131
132   int n,m;
133   scanf("%d %d",&n,&m);
134
135   for (int i = 0; i < n; i++)
136     v2n[i] = &mem[i];
137
138   for (int i = 0; i < m; i++){
139     int a,b;
140     if (scanf(" link %d %d",&a,&b) == 2)
141       link(a-1,b-1);
142     else if (scanf(" cut %d %d",&a,&b) == 2)
143       cut(a-1,b-1);
144     else if (scanf(" get %d %d",&a,&b) == 2)
145       printf("%d\n",get(a-1,b-1));
146     else
147       assert(false);
```

```
148   }
149   return 0;
150 }
```

final/graphs/chordaltree.cpp

```
1 void chordaltree(vector<vector<int>> e) {
2   int n = e.size();
3
4   vector<int> mark(n);
5   set<pair<int, int>> st;
6   for (int i = 0; i < n; i++) st.insert({-mark[i], i
      });
7
8   vector<int> vct(n);
9   vector<pair<int, int>> ted;
10   vector<vector<int>> who(n);
11   vector<vector<int>> verts(1);
12   vector<int> cliq(n, -1);
13   cliq.push_back(0);
14   vector<int> last(n + 1, n);
15   int prev = n + 1;
16   for (int i = n - 1; i >= 0; i--) {
17     int x = st.begin()->second;
18     st.erase(st.begin());
19     if (mark[x] <= prev) {
20       vector<int> cur = who[x];
21       cur.push_back(x);
22       verts.push_back(cur);
23       ted.push_back({cliq[last[x]], (int)verts.size
        () - 1});
24     } else {
25       verts.back().push_back(x);
26     }
27     for (int y : e[x]) {
28       if (cliq[y] != -1) continue;
29       who[y].push_back(x);
30       st.erase({-mark[y], y});
31       mark[y]++;
32       st.insert({-mark[y], y});
33       last[y] = x;
34     }
35     prev = mark[x];
36     vct[i] = x;
37     cliq[x] = (int)verts.size() - 1;
38   }
39
40   int k = verts.size();
41   vector<int> pr(k);
42   vector<vector<int>> g(k);
43   for (auto o : ted) {
44     pr[o.second] = o.first;
45     g[o.first].push_back(o.second);
46   }
47 }
```

final/graphs/minimization.cpp

```
1 namespace mimimi /* ^_^ */ {
2   const int N = 100555;
3   const int S = 3;
4   int e[N][S];
5   int label[N];
6   vector<int> eb[N][S];
7   int ans[N];
8   void solve(int n) {
9     for (int i = 0; i < n; ++i)
10       for (int j = 0; j < S; ++j)
11         eb[i][j].clear();
12     for (int i = 0; i < n; ++i)
13       for (int j = 0; j < S; ++j)
14         eb[e[i][j]][j].push_back(i);
15     vector<unordered_set<int>> classes(*max_element(
        label, label + n) + 1);
16     for (int i = 0; i < n; ++i)
17       classes[label[i]].insert(i);
18     for (int i = 0; i < sz(classes); ++i)
19       if (classes[i].empty()) {
20         classes[i].swap(classes.back());
21         classes.pop_back();
22         --i;
```

```
23          }
24      for (int i = 0; i < sz(classes); ++i)
25        for (int v : classes[i])
26          ans[v] = i;
27      for (int i = 0; i < sz(classes); ++i)
28        for (int c = 0; c < S; ++c) {
29          unordered_map<int, unordered_set<int>> ←
      involved;
30          for (int v : classes[i])
31            for (int nv : eb[v][c])
32              involved[ans[nv]].insert(nv);
33          for (auto &pp : involved) {
34            int cl = pp.X;
35            auto &cls = classes[cl];
36            if (sz(pp.Y) == sz(cls))
37              continue;
38            for (int x : pp.Y)
39              cls.erase(x);
40            if (sz(cls) < sz(pp.Y))
41              cls.swap(pp.Y);
42            for (int x : pp.Y)
43              ans[x] = sz(classes);
44            classes.push_back(move(pp.Y));
45          }
46        }
47    }
48    /* Usage: initialize edges: e[vertex][character]
49                    labels: label[vertex]
50          solve(n)
51          ans[] - classes
52    */
53  }
```

```
53        for (int to : e[i]) {
54          if (nd[to] > make_pair(d[i].first + w[to],←
      d[i].second + 1)) {
55            nd[to] = make_pair(d[i].first + w[to], d←
      [i].second + 1);
56            pr[to] = i;
57          }
58        }
59      }
60      if (d == nd) break;
61      d = nd;
62    }
63    int v = -1;
64    for (int i = 0; i < m; i++) {
65      if ((d[i].first < INF && (type[i] & 2)) && (v ←
      == -1 || d[i] < d[v])) v = i;
66    }
67    if (v == -1) break;
68    while (v != -1) {
69      sum += w[v];
70      taken[v] ^= 1;
71      v = pr[v];
72    }
73    ans[--cnt] = sum;
74  }
```

final/graphs/matroidIntersection.cpp

```
1   // check(ctaken, 1) --- first matroid
2   // check(ctaken, 2) --- second matroid
3   vector<char> taken(m);
4   while (1) {
5     vector<vector<int>> e(m);
6     for (int i = 0; i < m; i++) {
7       for (int j = 0; j < m; j++) {
8         if (taken[i] && !taken[j]) {
9           auto ctaken = taken;
10          ctaken[i] = 0;
11          ctaken[j] = 1;
12          if (check(ctaken, 2)) {
13            e[i].push_back(j);
14          }
15        }
16        if (!taken[i] && taken[j]) {
17          auto ctaken = taken;
18          ctaken[i] = 1;
19          ctaken[j] = 0;
20          if (check(ctaken, 1)) {
21            e[i].push_back(j);
22          }
23        }
24      }
25    }
26    vector<int> type(m);
27    // 0 --- cant, 1 --- can in \2, 2 --- can in \1
28    for (int i = 0; i < m; i++) {
29      if (!taken[i]) {
30        auto ctaken = taken;
31        ctaken[i] = 1;
32        if (check(ctaken, 2)) type[i] |= 1;
33      }
34      if (!taken[i]) {
35        auto ctaken = taken;
36        ctaken[i] = 1;
37        if (check(ctaken, 1)) type[i] |= 2;
38      }
39    }
40    vector<int> w(m);
41    for (int i = 0; i < m; i++) {
42      w[i] = taken[i] ? ed[i].c : -ed[i].c;
43    }
44    vector<pair<int, int>> d(m, {INF, 0});
45    for (int i = 0; i < m; i++) {
46      if (type[i] & 1) d[i] = {w[i], 0};
47    }
48    vector<int> pr(m, -1);
49    while (1) {
50      vector<pair<int, int>> nd = d;
51      for (int i = 0; i < m; i++) {
52        if (d[i].first == INF) continue;
```

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
Числелки:
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

prod (k=1..+inf) (1-x^k) = sum(q=-inf..+inf) (-1)^q x^((3q^2-q)/2)

Table of Integrals*

Basic Forms

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

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

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

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

Integrals of Rational Functions

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

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

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

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

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

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

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

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

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

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

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

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

Integrals with Roots

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Integrals with Logarithms

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

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

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

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

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

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

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

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

Integrals with Exponentials

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

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

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

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

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

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

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

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

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

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

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

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

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

Integrals with Trigonometric Functions

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Products of Trigonometric Functions and Monomials

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

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

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

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

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

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

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

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

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

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

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

Products of Trigonometric Functions and Exponentials

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

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

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

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

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

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

Integrals of Hyperbolic Functions

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

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

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

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

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

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

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

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

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

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

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

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