Содержание

1   final/template/template.cpp

```cpp
// team : SPb ITMO University
// setxkbmap us
#include <bits/stdc++.h>
#ifdef SIR
  #define err(...) fprintf(stderr, __VA_ARGS__)
#else
  #define err(...) 42
#endif

#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;

void solve() {

}

int main() {
#ifdef SIR
  freopen("input.txt", "r", stdin), freopen("output.
      txt", "w", stdout);
#endif
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  solve();
  return 0;
}
```

2   Practice round

- Посабмитить задачи каждому человеку.
- Распечатать решение.
- IDE для джавы.
- Сравнить скорость локального компьютера и сервера.
- Проверить int128.
- Проверить прагмы. Например, на bitset.

3 final/stuff/debug.cpp

```cpp
#include <bits/stdc++.h>
#define _GLIBCXX_DEBUG

using namespace std;

template <class T>
struct MyVector : vector<T> {
 MyVector() : vector<T>() { }
 MyVector( int n ) : vector<T>(n) { }
 T &operator [] ( int i )   { return vector<T>::at(i); }
 T operator [] ( int i ) const { return vector<T>::at(i); }
};

/** Если в вашем коде вместо всех       int[] и vector<int>
    использовать MyVector<int>,
   вы увидите все    range check errorы- */
MyVector<int> b(10), a;

int main() {
   MyVector<int> a(50);
   for (int i = 1; i <= 600; i++) a[i] = i;
   cout << a[500] << "\n";
}
```

4 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
```

5 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")
```

6 final/template/useful.cpp

```cpp
#include "ext/pb_ds/assoc_container.hpp"
#include <bits/extc++.h> /** keep-include */
using namespace __gnu_pbds;

gp_hash_table<ll, int> h({},{},{},{}, {1 << 16});
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;
}
```

7 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() {
```

```
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 }
```

8 final/template/bitset.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   }
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 };
```

9 final/template/treapNoRec.cpp

```
1  pnode Q[107], W[107], E[107];
2  int tp[107];
3
4  pnode merge(pnode L, pnode R) {
5    ind = 0;
6    while (1) {
7      Q[ind] = L, W[ind] = R;
8      if (!L) { E[ind++] = R; break; }
9      if (!R) { E[ind++] = L; break; }
10     if (L->prior > R->prior) {
11       L = L->R;
12       tp[ind] = 0;
13     } else {
14       R = R->L;
15       tp[ind] = 1;
16     }
17     ind++;
18   }
19   for (int i = ind - 2; i >= 0; i--) {
20     if (tp[i] == 0) {
21       Q[i]->R = E[i + 1], upd(Q[i]);
22       E[i] = Q[i];
23     } else {
24       W[i]->L = E[i + 1], upd(W[i]);
25       E[i] = W[i];
26     }
27   }
28   return E[0];
29 }
30
31 pair<pnode, pnode> split(pnode T, int key) {
32   ind = 0;
33   while (1) {
34     E[ind] = T;
35     if (!T) {
36       Q[ind] = W[ind] = NULL, ind++;
37       break;
38     }
39     if (T->key <= key) T = T->R, tp[ind] = 0;
40     else T = T->L, tp[ind] = 1;
41     ind++;
42   }
43   for (int i = ind - 2; i >= 0; i--) {
44     if (tp[i] == 0) {
45       E[i]->R = Q[i + 1], upd(E[i]);
46       Q[i] = E[i], W[i] = W[i + 1];
47     } else {
48       E[i]->L = W[i + 1], upd(E[i]);
49       Q[i] = Q[i + 1], W[i] = E[i];
50     }
51   }
52   return { Q[0], W[0] };
53 }
```

10   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[]
}
```

11   final/numeric/fst.cpp

```
 1  Transform to a basis with fast convolutions of the ←
       form
 2   * $\displaystyle c[z] = \sum\nolimits_{z = x \oplus←
       y} a[x] \cdot b[y]$,
 3   * where $\oplus$ is one of AND, OR, XOR. The size ←
       of $a$ must be a power of two.
 4
 5  void FST(vi& a, bool inv) {
 6    for (int n = sz(a), step = 1; step < n; step *= 2)←
         {
 7      for (int i = 0; i < n; i += 2 * step) rep(j,i,i+←
         step) {
 8        int &u = a[j], &v = a[j + step]; tie(u, v) =
 9          inv ? pii(v − u, u) : pii(v, u + v); // AND
10          inv ? pii(v, u − v) : pii(u + v, u); // OR
11          pii(u + v, u − v);                   // XOR
12      }
13    }
14    if (inv) trav(x, a) x /= sz(a); // XOR only
15  }
16  vi conv(vi a, vi b) {
17    FST(a, 0); FST(b, 0);
18    rep(i,0,sz(a)) a[i] *= b[i];
19    FST(a, 1); return a;
20  }
```

12   final/numeric/fftint.cpp

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

```
56      int invN = inv(N);
57      for (int i = 0; i < N; i++) A[i] = F[i] * (ll)G[←
         i] % MOD * invN % MOD;
58      reverse(A + 1, A + N);
59      fft(A, C);
60    }
61
62    void mult(int n1, int n2, int eq = 0) {
63      int n = n1 + n2, cur_base = 0;
64      while ((1 << cur_base) < n) cur_base++;
65      _init(cur_base + 1);
66
67      for (int i = n1; i < N; ++i) A[i] = 0;
68      for (int i = n2; i < N; ++i) B[i] = 0;
69
70      _mult(eq);
71
72      //forn(i, n1 + n2) C[i] = 0;
73      //forn(i, n1) forn(j, n2) C[i + j] = (C[i + j] +←
          A[i] * (ll)B[j]) % mod;
74    }
75
76    vector<int> mult(vector<int> A, vector<int> B) {
77      for (int i = 0; i < A.size(); i++) fft::A[i] = A←
         [i];
78      for (int i = 0; i < A.size(); i++) fft::B[i] = B←
         [i];
79      mult(A.size(), B.size());
80      vector<int> C(A.size() + B.size());
81      for (int i = 0; i < A.size() + B.size(); i++) C[←
         i] = fft::C[i];
82      return C;
83    }
84  }
```

13   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 }
```

14   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 }
```

15   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  }
```

16   final/numeric/extendedgcd.cpp

```
1  int gcd(int a, int b, int &x, int &y) {
2     if (a == 0) {
3        x = 0, y = 1;
4        return b;
5     }
6     int x1, y1;
7     int d = gcd(b % a, a, x1, y1);
8     x = y1 - (b / a) * x1;
9     y = x1;
10    return d;
11 }
```

17   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  }
```

18   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  }
```

19   final/numeric/pollard.cpp

```
1  namespace pollard
2  {
3     using math::p;
4
5     vector<pair<ll, int>> getFactors(ll N)  {
6        vector<ll> primes;
7
8        const int MX = 1e5;
9        const ll MX2 = MX * (ll)MX;
10
11       assert(MX <= math::maxP && math::pc > 0);
12
13       function<void(ll)> go = [&go, &primes](ll n) {
14          for (ll x : primes) while (n % x == 0) n /= x;
15          if (n == 1) return;
16          if (n > MX2) {
17             auto F = [&](ll x) {
18                ll k = ((long double)x * x) / n;
19                ll r = (x * x - k * n + 3) % n;
20                return r < 0 ? r + n : r;
21             };
22             ll x = mt19937_64()() % n, y = x;
23             const int C = 3 * pow(n, 0.25);
24
25             ll val = 1;
26             forn(it, C) {
27                x = F(x), y = F(F(y));
28                if (x == y) continue;
29                ll delta = abs(x - y);
30                ll k = ((long double)val * delta) / n;
31                val = (val * delta - k * n) % n;
32                if (val < 0) val += n;
33                if (val == 0)  {
34                   ll g = __gcd(delta, n);
35                   go(g), go(n / g);
36                   return;
37                }
38                if ((it & 255) == 0) {
39                   ll g = __gcd(val, n);
```

```
40            if (g != 1) {
41                go(g), go(n / g);
42                return;
43            }
44        }
45        }
46        }
47        primes.pb(n);
48    };
49
50    ll n = N;
51
52    for (int i = 0; i < math::pc && p[i] < MX; ++i) ↵
      if (n % p[i] == 0) {
53        primes.pb(p[i]);
54        while (n % p[i] == 0) n /= p[i];
55    }
56    go(n);
57    sort(primes.begin(), primes.end());
58
59    vector<pair<ll, int>> res;
60    for (ll x : primes) {
61        int cnt = 0;
62        while (N % x == 0) {
63            cnt++;
64            N /= x;
65        }
66        res.push_back({x, cnt});
67    }
68    return res;
69    }
70 }
```

20   final/numeric/poly.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 << "=";
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;
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 }
```

21   final/numeric/simplex.cpp

```cpp
1  typedef double T; // long double, Rational, double +↩
       mod<P>...
2  typedef vector<T> vd;
3  typedef vector<vd> vvd;
4
5  const T eps = 1e-8, inf = 1/.0;
6  #define MP make_pair
7  #define ltj(X) if(s == -1 || MP(X[j],N[j]) < MP(X[s↩
       ],N[s])) s=j
8  #define sz(X) ((X).size())
9  #define rep(i,l,r) for (int i = (l); i < (r); i++)
10
11 struct LPSolver {
12   // Description: Solves a general linear ↩
       maximization problem: maximize $c^T x$ subject ↩
       to $Ax \le b$, $x \ge 0$.
13   // A is a matrix with shape (number of ↩
       inequalities, number of variables)
14   // Returns −inf if there is no solution, inf if ↩
       there are arbitrarily good solutions, or the ↩
       maximum value of $c^T x$ otherwise.
15   // The input vector is set to an optimal $x$ (or ↩
       in the unbounded case, an arbitrary solution ↩
       fulfilling the constraints).
16
17   int m, n;
18   vector<int> N, B;
19   vvd D;
20
21   LPSolver(const vvd& A, const vd& b, const vd& c) :
22       m(sz(b)), n(sz(c)), N(n+1), B(m), D(m+2, vd(n↩
       +2)) {
23     rep(i,0,m) rep(j,0,n) D[i][j] = A[i][j];
24     rep(i,0,m) { B[i] = n+i; D[i][n] = −1; D[i][n+1]↩
       = b[i];}
25     rep(j,0,n) { N[j] = j; D[m][j] = −c[j]; }
26     N[n] = −1; D[m+1][n] = 1;
27   }
28
29   void pivot(int r, int s) {
30     T *a = D[r].data(), inv = 1 / a[s];
31     rep(i,0,m+2) if (i != r && abs(D[i][s]) > eps) {
32       T *b = D[i].data(), inv2 = b[s] * inv;
33       rep(j,0,n+2) b[j] −= a[j] * inv2;
34       b[s] = a[s] * inv2;
35     }
36     rep(j,0,n+2) if (j != s) D[r][j] *= inv;
37     rep(i,0,m+2) if (i != r) D[i][s] *= −inv;
38     D[r][s] = inv;
39     swap(B[r], N[s]);
40   }
41
42   bool simplex(int phase) {
43     int x = m + phase − 1;
44     for (int it = 0; it < 100; it++) {
45       int s = −1;
46       rep(j,0,n+1) if (N[j] != −phase) ltj(D[x]);
47       if (D[x][s] >= −eps) return true;
48       int r = −1;
49       rep(i,0,m) {
50         if (D[i][s] <= eps) continue;
51         if (r == −1 || MP(D[i][n+1] / D[i][s], B[i])
52             < MP(D[r][n+1] / D[r][s], B[r])) r =↩
       i;
53       }
54       if (r == −1) return false;
55       pivot(r, s);
56     }
57   }
58
59   T solve(vd &x) {
60     int r = 0;
61     rep(i,1,m) if (D[i][n+1] < D[r][n+1]) r = i;
62     if (D[r][n+1] < −eps) {
63       pivot(r, n);
64       if (!simplex(2) || D[m+1][n+1] < −eps) return ↩
       −inf;
65       rep(i,0,m) if (B[i] == −1) {
66         int s = 0;
67         rep(j,1,n+1) ltj(D[i]);
68         pivot(i, s);
```

```cpp
69       }
70     }
71     bool ok = simplex(1); x = vd(n);
72     rep(i,0,m) if (B[i] < n) x[B[i]] = D[i][n+1];
73     return ok ? D[m][n+1] : inf;
74   }
75 };
```

22   final/numeric/sumLine.cpp

```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 }
```

23   final/numeric/integrate.cpp

```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 };
```

24   final/numeric/rootsPolynom.cpp

```cpp
1  const double EPS = 1e−9;
2  double cal(const vector<double> &coef, double x) {
3    double e = 1, s = 0;
4    for (double i : coef) s += i * e, e *= x;
5    return s;
6  }
7
8  int dblcmp(double x) {
9    if (x < −EPS) return −1;
10   if (x > EPS) return 1;
11   return 0;
12 }
13
14 double find(const vector<double> &coef, double l, ↩
       double r) {
15   int sl = dblcmp(cal(coef, l)), sr = dblcmp(cal(↩
       coef, r));
16   if (sl == 0) return l;
17   if (sr == 0) return r;
18   for (int tt = 0; tt < 100 && r − l > EPS; ++tt) {
19     double mid = (l + r) / 2;
20     int smid = dblcmp(cal(coef, mid));
21     if (smid == 0) return mid;
22     if (sl * smid < 0) r = mid;
23     else l = mid;
24   }
25   return (l + r) / 2;
26 }
27
28 vector<double> rec(const vector<double> &coef, int n↩
       ) {
29   vector<double> ret; // c[0]+c[1]*x+c[2]*x^2+...+c[↩
       n]*x^n, c[n]==1
```

```cpp
30      if (n == 1) {
31        ret.push_back(-coef[0]);
32        return ret;
33      }
34      vector<double> dcoef(n);
35      for (int i = 0; i < n; ++i) dcoef[i] = coef[i + 1]↩
          * (i + 1) / n;
36      double b = 2; // fujiwara bound
37      for (int i = 0; i <= n; ++i) b = max(b, 2 * pow(↩
          fabs(coef[i]), 1.0 / (n - i)));
38      vector<double> droot = rec(dcoef, n - 1);
39      droot.insert(droot.begin(), -b);
40      droot.push_back(b);
41      for (int i = 0; i + 1 < droot.size(); ++i) {
42        int sl = dblcmp(cal(coef, droot[i])), sr = ↩
          dblcmp(cal(coef, droot[i + 1]));
43        if (sl * sr > 0) continue;
44        ret.push_back(find(coef, droot[i], droot[i + 1])↩
          );
45      }
46      return ret;
47    }
48
49    vector<double> solve(vector<double> coef) {
50      int n = coef.size() - 1;
51      while (coef.back() == 0) coef.pop_back(), --n;
52      for (int i = 0; i <= n; ++i) coef[i] /= coef[n];
53      return rec(coef, n);
54    }
```

25　final/numeric/phiFunction.cpp

```cpp
1    void totient(){
2     for(int i = 0; i < MAX; i++){
3      phi[i] = i;
4      pr[i] = true;
5     }
6     for(int i = 2; i < MAX; i++)
7      if(pr[i]){
8        for(int j = i; j < MAX; j+=i){
9         pr[j] = false;
10        phi[j] = phi[j] - (phi[j] / i);
11       }
12       pr[i] = true;
13      }
14    }
```

26　final/numeric/partition.cpp

```cpp
1    // number of ways to divide n to integers(unordered)↩
        , O(n^(3/2))
2    int partition(int n) {
3      int dp[n + 1];
4      dp[0] = 1;
5      for (int i = 1; i <= n; i++) {
6        dp[i] = 0;
7        for (int j = 1, r = 1; i - (3 * j * j - j) / 2 ↩
            >= 0; ++j, r *= -1) {
8          dp[i] += dp[i - (3 * j * j - j) / 2] * r;
9          if (i - (3 * j * j + j) / 2 >= 0) dp[i] += dp[↩
              i - (3 * j * j + j) / 2] * r;
10       }
11     }
12     return dp[n];
13   }
```

27　final/geom/commonTangents.cpp

```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();
```

```cpp
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(0.01, 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}
```

28　final/geom/halfplaneIntersection.cpp

```cpp
1    int getPart(pt v) {
2      return ls(v.y, 0) || (eq(0, v.y) && ls(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 (eq(0, a * b)) return 0;
11     if (0 < a * b) return -1;
12     return 1;
13   }
14
15   double planeInt(vector<Line> l) {
16     sort(all(l), [](Line a, Line b) {
17         int r = cmpV(a.v, b.v);
18         if (r != 0) return r < 0;
19         return a.O % a.v.rotate() > b.O % a.v.rotate()↩
           ;
20       });
21
22     l.resize(unique(all(l), [](Line A, Line B) { ↩
         return cmpV(A.v, B.v) == 0; }) - l.begin());
23     for (int i = 0; i < sz(l); i++)
24       l[i].id = i;
25
26     // if an infinite answer is possible
27     int flagUp = 0;
28     int flagDown = 0;
29     for (int i = 0; i < sz(l); i++) {
30       int part = getPart(l[i].v);
31       if (part == 1) flagUp = 1;
32       if (part == 0) flagDown = 1;
33     }
34     if (!flagUp || !flagDown) return -1;
35
36     for (int i = 0; i < sz(l); i++) {
37       pt v = l[i].v;
38       pt u = l[(i + 1) % sz(l)].v;
39       if (eq(0, v * u) && ls(v % u, 0)) {
40         pt dir = l[i].v.rotate();
41         if (le(l[(i + 1) % sz(l)].O % dir, l[i].O % ↩
           dir)) return 0;
42         return -1;
43       }
44       if (ls(v * u, 0))
45         return -1;
46     }
47     // main part
48     vector<Line> st;
49     for (int tt = 0; tt < 2; tt++) {
50       for (auto L: l) {
51         for (; sz(st) >= 2 && le(st[sz(st) - 2].v * (↩
           st.back() * L - st[sz(st) - 2].O), 0); st.↩
           pop_back());
52         st.pb(L);
```

```
53        if (sz(st) >= 2 && le(st[sz(st) - 2].v * st.←
          back().v, 0)) return 0; // useless line
54      }
55    }
56    vector<int> use(sz(l), -1);
57    int left = -1, right = -1;
58    for (int i = 0; i < sz(st); i++) {
59      if (use[st[i].id] == -1) {
60        use[st[i].id] = i;
61      }
62      else {
63        left = use[st[i].id];
64        right = i;
65        break;
66      }
67    }
68    vector<Line> tmp;
69    for (int i = left; i < right; i++)
70      tmp.pb(st[i]);
71    vector<pt> res;
72    for (int i = 0; i < (int)tmp.size(); i++)
73      res.pb(tmp[i] * tmp[(i + 1) % tmp.size()]);
74    double area = 0;
75    for (int i = 0; i < (int)res.size(); i++)
76      area += res[i] * res[(i + 1) % res.size()];
77    return area / 2;
78 }
```

29 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 }
```

30 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;
```

31 final/geom/convexDynamic.cpp

```
1  struct convex {
2    map<ll, ll> M;
3    bool get(int x, int y) {
4      if (M.size() == 0)
5        return false;
6      if (M.count(x))
7        return M[x] >= y;
8      if (x < M.begin()->first || x > M.rbegin()->←
    first)
9        return false;
10
11     auto it1 = M.lower_bound(x), it2 = it1;
12     it1--;
13
14     return pt(pt(*it1), pt(x, y)) % pt(pt(*it1), pt←
    (*it2)) >= 0;
15   }
16   void add(int x, int y) {
17     if (get(x, y)) return;
18
19     pt P(x, y);
20     M[x] = y;
21
22     auto it = M.lower_bound(x), it1 = it;
23     it1--;
24     auto it2 = it1;
25     it2--;
26
27     if (it != M.begin() && it1 != M.begin()) {
28       while (it1 != M.begin() && (pt(pt(*it2), pt(*←
    it1)) % pt(pt(*it1), P)) >= 0) {
29         M.erase(it1);
30         it1 = it2;
31         it2--;
32       }
33     }
34     it1 = it, it1++;
35     if (it1 == M.end()) return;
36     it2 = it1, it2++;
37
38     if (it1 != M.end() && it2 != M.end()) {
```

```
39        while (it2 != M.end() && (pt(P, pt(*it1)) % pt↩
          (pt(*it1), pt(*it2))) >= 0) {
40          M.erase(it1);
41          it1 = it2;
42          it2++;
43        }
44      }
45    }
46  } H, J;
47
48  int solve() {
49    int q;
50    cin >> q;
51    while (q--) {
52      int t, x, y;
53      cin >> t >> x >> y;
54      if (t == 1) {
55        H.add(x, y);
56        J.add(x, -y);
57      }
58      else {
59        if (H.get(x, y) && J.get(x, -y))
60          puts("YES");
61        else
62          puts("NO");
63      }
64    }
65    return 0;
66  }
```

32   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  }
```

33   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  }
```

34   final/geom/checkPlaneInt.cpp

```
1   bool eq(dbl A, dbl B) { return abs(A - B) < 1e-9; }
2
3   bool ls(dbl A, dbl B) { return A < B && !eq(A, B); }
4
5   bool le(dbl A, dbl B) { return A < B || eq(A, B); }
6   struct pt {
7     double x, y;
8     pt(double x, double y) : x(x), y(y) {}
9     pt() : pt(0, 0) {}
10    double operator%(pt b) const { return x * b.x + y ↩
        * b.y; }
11    // Orintation of cross product and rotation DO ↩
        matter in some algorithms
12    double operator*(pt b) const { return x * b.y - y ↩
        * b.x; }
13    pt rotate() { return {y, -x}; }
14    pt operator-(pt b) const { return {x - b.x, y - b.↩
        y}; }
15    pt operator*(double t) const { return {x * t, y * ↩
        t}; }
16    pt operator+(pt b) const { return {x + b.x, y + b.↩
        y}; }
17  };
18
19  // Also this is half-plane struct
20  struct Line {
21    pt O, v;
22
23    // Ax + By + C <= 0
24    Line(double A, double B, double C) {
25      double l = sqrt(A * A + B * B);
26      A /= l, B /= l, C /= l;
27      O = pt(-A * C, -B * C);
28      v = pt(-B, A);
29    }
30    //intersection with l
31    pt operator*(Line l) {
32      pt u = l.v.rotate();
33      dbl t = (l.O - O) % u / (v % u);
34      return O + v * t;
35    }
36    // Half-plane with point O on the border, ↩
        everything to the LEFT of direction vector v is↩
         inside
37    Line(pt O, pt v) : O(O), v(v) {}
38  };
39
40  const double EPS = 1e-14;
41  double INF = 1e50;
42
43  // vector<Line> lines{
44  //    Line(pt(0, 0), pt(0, -1)),
45  //    Line(pt(0, 0), pt(-1, 0)),
46  //    Line(pt(1, 1), pt(0, 1)),
47  // };
48  // checkPoint(lines, p) == true
49  // Intersection of lines is rectangle of set o
50  // Time complexity is O(n)
51  bool checkPoint(vector<Line> &l, pt &ret) {
52    random_shuffle(l.begin(), l.end());
53    pt A = l[0].O;
54    for (int i = 1; i < l.size(); i++) {
55      if (l[i].v * (A - l[i].O) < -EPS) {
56        double mn = -INF;
57        double mx = INF;
58        for (int j = 0; j < i; j++) {
59          if (abs(l[j].v * l[i].v) < EPS) {
60            if (l[j].v % l[i].v < 0 && (l[j].O - l[i].↩
        O) % l[i].v.rotate() < EPS) {
61              return false;
62            }
63          } else {
```

```
64            pt u = l[j].v.rotate();
65            double proj = (l[j].O - l[i].O) % u / (l[i←
      ].v % u);
66            if (l[i].v * l[j].v > 0) {
67              mx = min(mx, proj);
68            } else {
69              mn = max(mn, proj);
70            }
71          }
72        }
73        if (mn <= mx) {
74          A = l[i].O + l[i].v * mn;
75        } else {
76          return false;
77        }
78      }
79    }
80    ret = A;
81    return true;
82  }
```

35   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 }
```

36   final/geom/chtDynamic.cpp

```
1
2  const ll is_query = -(1LL << 62);
3
4  struct Line {
5    ll m, b;
6    mutable function<const Line *()> succ;
7
8    bool operator<(const Line &rhs) const {
9      if (rhs.b != is_query) return m < rhs.m;
10     const Line *s = succ();
11     if (!s) return 0;
12     ll x = rhs.m;
13     return b - s->b < (s->m - m) * x;
14   }
15 };
16
17 struct HullDynamic : public multiset<Line> {
18   bool bad(iterator y) {
19     auto z = next(y);
20     if (y == begin()) {
21       if (z == end()) return 0;
22       return y->m == z->m && y->b <= z->b;
23     }
24     auto x = prev(y);
25     if (z == end()) return y->m == x->m && y->b <= x←
      ->b;
26     return (x->b - y->b) * (z->m - y->m) >= (y->b - ←
      z->b) * (y->m - x->m);
27   }
28
29   void insert_line(ll m, ll b) {
30     auto y = insert({m, b});
31     y->succ = [=] { return next(y) == end() ? 0 : &*←
      next(y); };
32     if (bad(y)) {
33       erase(y);
34       return;
35     }
36     while (next(y) != end() && bad(next(y))) erase(←
      next(y));
37     while (y != begin() && bad(prev(y))) erase(prev(←
      y));
38   }
39
```

```
40   ll eval(ll x) {
41     auto l = *lower_bound((Line) {x, is_query});
42     return l.m * x + l.b;
43   }
44 };
```

37   final/geom/rotate3D.cpp

```
1  // Rotate 3d point along axis on angle
2  /*
3   * 2D
4   * x' = x cos a - y sin a
5   * y' = x sin a + y cos a
6   */
7  struct quater {
8    double w, x, y, z; // w + xi + yj + zk
9    quater(double tw, const pt3 &v) : w(tw), x(v.x), y←
      (v.y), z(v.z) { }
10   quater(double tw, double tx, double ty, double tz)←
      : w(tw), x(tx), y(ty), z(tz) { }
11   pt3 vector() const {
12     return {x, y, z};
13   }
14   quater conjugate() const {
15     return {w, -x, -y, -z};
16   }
17   quater operator*(const quater &q2) {
18     return {w * q2.w - x * q2.x - y * q2.y - z * q2.←
      z, w * q2.x + x * q2.w + y * q2.z - z * q2.y, w←
      * q2.y - x * q2.z + y * q2.w + z * q2.x, w * ←
      q2.z + x * q2.y - y * q2.x + z * q2.w};
19   }
20 };
21
22 pt3 rotate(pt3 axis, pt3 p, double angle) {
23   quater q = quater(cos(angle / 2), axis * sin(angle←
      / 2));
24   return (q * quater(0, p) * q.conjugate()).vector()←
      ;
25 }
```

38   final/geom/circleInter.cpp

```
1  pair<pt, pt> inter(pt O1, double r1, pt O2, double ←
      r2) {
2    line m(O1, O2);
3    double a = (r2 * r2 - r1 * r1 + d * d) / (2 * d);
4    pt HO2(-m.b, m.a);
5    HO2.to_dist(-a);
6    point H(O2.x + HO2.x, O2.y + HO2.y);
7    double h = sqrt(r2 * r2 - a * a);
8    pt HP1(m.a, m.b);
9    HP1.to_dist(h);
10   pt P1(H.x + HP1.x, H.y + HP1.y);
11   pt P2(H.x - HP1.x, H.y - HP1.y);
12   return { P1, P2 };
13 }
```

39   final/geom/sphericalDistance.cpp

```
1  double sphericalDistance(double f1, double t1,
2      double f2, double t2, double radius) {
3    double dx = sin(t2)*cos(f2) - sin(t1)*cos(f1);
4    double dy = sin(t2)*sin(f2) - sin(t1)*sin(f1);
5    double dz = cos(t2) - cos(t1);
6    double d = sqrt(dx*dx + dy*dy + dz*dz);
7    return radius*2*asin(d/2);
8  }
```

40   final/strings/eertree.cpp

```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;
  }
}
```

41   final/strings/manacher.cpp

```cpp
vector<int> Pal1(string s) {
  int n = (int)s.size();
  vector<int> d1(n);
  int l = 0, r = -1;
  for (int i = 0, k; i < n; i++) {
    if (i > r) k = 1;
    else k = min(d1[l + r - i], r - i);
    while (0 <= i - k && i + k < n && s[i - k] == s[←
    i + k]) k++;
    d1[i] = k;
    if (i + k - 1 > r) r = i + k - 1, l = i - k + 1;
  }
  return d1;
}

vector<int> Pal2(string s) {
  int n = (int)s.size();
  vector<int> d2(n);
  int l = 0, r = -1;
  for (int i = 0, k; i < n; i++) {
    if (i > r) k = 0;
    else k = min(d2[l + r - i + 1], r - i + 1);
    while (i + k < n && i - k - 1 >= 0 && s[i + k] ←
    == s[i - k - 1]) k++;
    d2[i] = k;
    if (i + k - 1 > r) l = i - k, r = i + k - 1;
  }
  return d2;
}
```

42   final/strings/sufAutomaton.cpp

```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;
    }
  }
}
```

43   final/strings/sufTree.cpp

```cpp
const int N = 1e5, VN = 2 * N;

map<char,int> t[VN];
int l[VN], r[VN], p[VN], term[VN]; // ребро p[v] -> ←
 v этоотрезок  [l[v], r[v]) исходнойстроки
int cc, suf[VN], vn = 2, v = 1, pos; // ←
    идёмпоребруиз   p[v] в v, сейчасстоимв   pos

void init() {
  for (int i = 0; i < 127; i++) t[0][i] = 1; // 0 = ←
    фиктивная, 1 = корень
  l[1] = -1;
}

void add(char c, int i, const string &s) {
```

```
13    auto new_leaf = [&](int v) {
14      p[vn] = v, l[vn] = i, r[vn] = N, t[v][c] = vn++;
15    };
16    go:;
17    if (r[v] <= pos) {
18      if (!t[v].count(c)) {
19        new_leaf(v), v = suf[v], pos = r[v];
20        goto go;
21      }
22      v = t[v][c], pos = l[v] + 1;
23    } else if (c == s[pos]) {
24      pos++;
25    } else {
26      int x = vn++;
27      l[x] = l[v], r[x] = pos, l[v] = pos;
28      p[x] = p[v], p[v] = x;
29      t[p[x]][s[l[x]]] = x, t[x][s[pos]] = v;
30      new_leaf(x);
31      v = suf[p[x]], pos = l[x];
32      while (pos < r[x])
33        v = t[v][s[pos]], pos += r[v] - l[v];
34      suf[x] = (pos == r[x] ? v : vn);
35      pos = r[v] - (pos - r[x]);
36      goto go;
37    }
38  }
39
40  int main() {
41    init();
42    string s; cin >> s;
43    s += (char)0; // term
44    for (int i = 0; i < (int)s.size(); i++) {
45      add(s[i], i, s);
46    }
47    for (int i = 1; i < vn; i++) r[i] = min(r[i], (int)s.size());
48    for (int i = 1; i < vn; i++) {
49      for (auto c : t[i]) err("%d [%d, %d) %d\n", i, l[c.second], r[c.second], c.second);
50    }
51  }
```

44   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;
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 }
```

45   final/strings/sufArrayLinear.cpp

```
1  const int dd = (int)2e6 + 3;
2
3  ll cnt2[dd];
4  int AN;
5  int A[3 * dd + 100];
6  int cnt[dd + 1]; // Should be >= 256
7  int SA[dd + 1];
8
9  /* Used by suffix_array. */
10 void radix_pass(int* A, int AN, int* R, int RN, int* D) {
11   memset(cnt, 0, sizeof(int) * (AN + 1));
12   int* C = cnt + 1;
13   for(int i = 0; i < RN; i++) ++C[A[R[i]]];
14   for(int i = -1, v = 0; i <= AN && v < RN; v += C[i++]) swap(v, C[i]);
15   for(int i = 0; i < RN; i++) D[C[A[R[i]]]++] = R[i];
16 }
17
18 /* DC3 in O(N) using 20N bytes of memory. Stores the
19  * suffix array of the string
20  * [A,A+AN) into SA where SA[i] (0<=i<=AN) gives the
21  *   starting position of the
22  * i-th least suffix of A (including the empty
23  *   suffix).
24  */
25 void suffix_array(int* A, int AN) {
26   // Base case... length 1 string.
27   if(!AN) {
28     SA[0] = 0;
29   } else if(AN == 1) {
30     SA[0] = 1; SA[1] = 0;
31     return;
32   }
33
34   // Sort all strings of length 3 starting at non-
35   //   multiples of 3 into R.
36   int RN = 0;
37   int* SUBA = A + AN + 2;
38   int* R = SUBA + AN + 2;
39   for(int i = 1; i < AN; i += 3) SUBA[RN++] = i;
40   for(int i = 2; i < AN; i += 3) SUBA[RN++] = i;
41   A[AN + 1] = A[AN] = -1;
42   radix_pass(A + 2, AN - 2, SUBA, RN, R);
43   radix_pass(A + 1, AN - 1, R, RN, SUBA);
44   radix_pass(A + 0, AN - 0, SUBA, RN, R);
45
46   // Compute the relabel array if we need to
47   //   recursively solve for the
48   // non-multiples.
49   int resfix, resmul, v;
50   if(AN % 3 == 1) {
51     resfix = 1; resmul = RN >> 1;
52   } else {
53     resfix = 2; resmul = RN + 1 >> 1;
54   }
55   for(int i = v = 0; i < RN; i++) {
56     v += i && (A[R[i - 1] + 0] != A[R[i] + 0] ||
57       A[R[i - 1] + 1] != A[R[i] + 1] ||
58       A[R[i - 1] + 2] != A[R[i] + 2]);
59     SUBA[R[i] / 3 + (R[i] % 3 == resfix) * resmul] = v;
60   }
61
62   // Recursively solve if needed to compute relative
63   //   ranks in the final suffix
64   // array of all non-multiples.
65   if(v + 1 != RN) {
66     suffix_array(SUBA, RN);
67     SA[0] = AN;
68     for(int i = 1; i <= RN; i++) {
69       SA[i] = SA[i] < resmul ? 3 * SA[i] + (resfix==1?2:1) :
70         3 * (SA[i] - resmul) + resfix;
```

```
65    }
66   } else {
67     SA[0] = AN;
68     memcpy(SA + 1, R, sizeof(int) * RN);
69   }
70
71   // Compute the relative ordering of the multiples.
72   int NMN = RN;
73   for(int i = RN = 0; i <= NMN; i++) {
74     if(SA[i] % 3 == 1) {
75       SUBA[RN++] = SA[i] - 1;
76     }
77   }
78   radix_pass(A, AN, SUBA, RN, R);
79
80   // Compute the reverse SA for what we know so far.
81   for(int i = 0; i <= NMN; i++) {
82     SUBA[SA[i]] = i;
83   }
84
85   // Merge the orderings.
86   int ii = RN - 1;
87   int jj = NMN;
88   int pos;
89   for(pos = AN; ii >= 0; pos--) {
90     int i = R[ii];
91     int j = SA[jj];
92     int v = A[i] - A[j];
93     if(!v) {
94       if(j % 3 == 1) {
95         v = SUBA[i + 1] - SUBA[j + 1];
96       } else {
97         v = A[i + 1] - A[j + 1];
98         if(!v) v = SUBA[i + 2] - SUBA[j + 2];
99       }
100    }
101    SA[pos] = v < 0 ? SA[jj--] : R[ii--];
102  }
103 }
104
105 char s[dd + 1];
106
107 /* Copies the string in s into A and reduces the ↩
         characters as needed. */
108 void prep_string() {
109   int v = AN = 0;
110   memset(cnt, 0, 256 * sizeof(int));
111   for(char* ss = s; *ss; ++ss, ++AN) cnt[*ss]++;
112   for(int i = 0; i < AN; i++) cnt[s[i]]++;
113   for(int i = 0; i < 256; i++) cnt[i] = cnt[i] ? v++ ↩
          : -1;
114   for(int i = 0; i < AN; i++) A[i] = cnt[s[i]];
115 }
116
117 /* Computes the reverse SA index. REVSA[i] gives the↩
          index of the suffix
118  * starting a i in the SA array. In other words, ↩
          REVSA[i] gives the number of
119  * suffixes before the suffix starting at i. This ↩
          can be useful in itself but
120  * is also used for compute_lcp().
121  */
122 int REVSA[dd + 1];
123 void compute_reverse_sa() {
124   for(int i = 0; i <= AN; i++) {
125     REVSA[SA[i]] = i;
126   }
127 }
128
129 /* Computes the longest common prefix between ↩
          adjacent suffixes. LCP[i] gives
130  * the longest common suffix between the suffix ↩
          starting at i and the next
131  * smallest suffix. Runs in O(N) time.
132  */
133 int LCP[dd + 1];
134 void compute_lcp() {
135   int len = 0;
136   for(int i = 0; i < AN; i++, len = max(0, len - 1)) ↩
          {
137     int s = REVSA[i];
138     int j = SA[s - 1];
139     for(; i + len < AN && j + len < AN && A[i + len] ↩
          == A[j + len]; len++);
140     LCP[s] = len;
141   }
142 }
```

46   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 }
```

47 final/graphs/alphaBetta.cpp

```
1  int alphabeta(state s, int alpha, int beta) {
2    if (s.finished()) return s.score();
3    for (state t : s.next()) {
4      alpha = max(alpha, -alphabeta(t, -beta, -alpha))↵
         ;
5      if (alpha >= beta) break;
6    }
7    return alpha;
8  }
```

48 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  }
```

49 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  }
```

50 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  }
```

51 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  }
```

52 final/graphs/minCost.cpp

```
1   ll findflow(int s, int t) {
2     ll cost = 0;
```

```
  3      ll flow = 0;
  4
  5      forn(i, N) G[i] = inf;
  6
  7      queue<int> q;
  8
  9      q.push(s);
 10      used[s] = true;
 11      G[s] = 0;
 12
 13      while (q.size()) {
 14        int v = q.front();
 15        used[v] = false;
 16        q.pop();
 17
 18        forn(i, E[v].size()) {
 19          auto &e = E[v][i];
 20          if (e.f < e.c && G[e.to] > G[v] + e.w) {
 21            G[e.to] = G[v] + e.w;
 22            if (!used[e.to]) {
 23              q.push(e.to);
 24              used[e.to] = true;
 25            }
 26          }
 27        }
 28      }
 29
 30      while (1) {
 31        forn(i, N)
 32          d[i] = inf, p[i] = { -1, -1 }, used[i] = 0;
 33
 34        d[s] = 0;
 35        while (1) {
 36          int v = -1;
 37          forn(i, N) {
 38            if (!used[i] && d[i] != inf && (v == -1 || d
                   [i] < d[v]))
 39              v = i;
 40          }
 41          if (v == -1)
 42            break;
 43          used[v] = 1;
 44
 45          forn(i, E[v].size()) {
 46
 47            auto &e = E[v][i];
 48            if (e.f < e.c && d[e.to] > d[v] + e.w + G[v]
                   - G[e.to]) {
 49              p[e.to] = mp(v, i);
 50              d[e.to] = d[v] + e.w + G[v] - G[e.to];
 51            }
 52          }
 53
 54        }
 55        if (p[t].first == -1) {
 56          break;
 57        }
 58        int add = inf;
 59        for (int i = t; p[i].first != -1; i = p[i].first
                ) {
 60          add = min(add, E[p[i].first][p[i].second].c -
                E[p[i].first][p[i].second].f);
 61        }
 62        for (int i = t; p[i].first != -1; i = p[i].first
                ) {
 63          auto &e = E[p[i].first][p[i].second];
 64          cost += 1ll * add * e.w;
 65          e.f += add;
 66          E[e.to][e.back].f -= add;
 67        }
 68        flow += add;
 69        if (add == 0)
 70          break;
 71        forn(i, N)
 72          G[i] += d[i];
 73      }
 74      return cost;
 75    }
```

53 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        int v = bestU;
 93        int it = iters;
 94        vector<int> was(n, -1);
 95        while (was[v] == -1) {
 96          was[v] = it;
 97          v = edges[p[it][v]].from;
 98          it--;
 99        }
100        int vv = v;
101        it = was[v];
102        double sum = 0;
103        do {
104          edges[p[it][v]].flow++;
105          sum += edges[p[it][v]].cost;
106          edges[p[it][v] ^ 1].flow--;
107          v = edges[p[it][v]].from;
108          it--;
109        } while (v != vv);
110        return 1;
111      }
112    };
113
```

54   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  }
```

55   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  }
```

56   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];
```

```cpp
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 }
```

57   final/graphs/linkcut.cpp

```cpp
1 #include <iostream>
```

```cpp
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  }
```

58   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  }
```

59   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  }
```

60   final/graphs/matroidIntersection.cpp

```cpp
struct Graph {
  vector<vector<int>> G;

  Graph(int n = 0) {
    G.resize(n);
  }

  void add_edge(int v, int u) {
    G[v].push_back(u);
  }

  vector<int> get_path(vector<int> &s, vector<int> &
    t) {
    int n = G.size();
    vector<int> dist(n, inf), pr(n, -1);
    queue<int> Q;
    for (int i : s) {
      dist[i] = 0;
      Q.push(i);
    }
    while (!Q.empty()) {
      int v = Q.front();
      Q.pop();
      for (int to : G[v]) if (dist[to] > dist[v] +
    1) {
        dist[to] = dist[v] + 1;
        pr[to] = v;
        Q.push(to);
      }
    }
    int V = -1;
    for (int i : t) if (V == -1 || dist[i] < dist[V
    ]) {
      V = i;
    }
    if (V == -1 || dist[V] == inf) return {};
    vector<int> path;
    while (V != -1) {
      path.push_back(V);
      V = pr[V];
    }
    return path;
  }
};

void get_ans(vector<int> &used, int m) {
  Graph G(m);
  for (int i = 0; i < m; ++i) if (used[i]) {
    Gauss gauss;
    vector<int> color(130, 0);
    for (int j = 0; j < m; ++j) if (used[j] && j !=
    i) {
      gauss.add(a[j]);
      color[c[j]] = 1;
    }
    for (int j = 0; j < m; ++j) if (!used[j]) {
      if (gauss.check(a[j])) {
        G.add_edge(i, j);
      }
      if (!color[c[j]]) {
        G.add_edge(j, i);
      }
    }
  }

  Gauss gauss;
  vector<int> color(130, 0);
  for (int i = 0; i < m; ++i) if (used[i]) {
    gauss.add(a[i]);
    color[c[i]] = 1;
  }
  vector<int> x1, x2;
  for (int i = 0; i < m; ++i) if (!used[i]) {
    if (gauss.check(a[i])) {
      x1.push_back(i);
    }
    if (!color[c[i]]) {
      x2.push_back(i);
    }
  }
  vector<int> path = G.get_path(x1, x2);
  if (!path.size()) return;
  for (int i : path) used[i] ^= 1;
  get_ans(used, m);
}
```

61   final/graphs/compressTree.cpp

```cpp
vector<pair<int, int>> compressTree(LCA& lca, const
    vi& subset) {
  static vector<int> rev; rev.resize(sz(lca.dist));
  vi li = subset, &T = lca.time;
  auto cmp = [&](int a, int b) { return T[a] < T[b];
    };
  sort(all(li), cmp);
  int m = sz(li)-1;
  rep(i,0,m) {
    int a = li[i], b = li[i+1];
    li.push_back(lca.query(a, b));
  }
  sort(all(li), cmp);
  li.erase(unique(all(li)), li.end());
  rep(i,0,sz(li)) rev[li[i]] = i;
  vpi ret = {pii(0, li[0])};
  rep(i,0,sz(li)-1) {
    int a = li[i], b = li[i+1];
    ret.emplace_back(rev[lca.query(a, b)], b);
  }
  return ret;
}
```

```
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 $\mathcal{O}(h^4)$
$y' = f(x, y)$
$x_{n+1} = x_n + h, y_{n+1} = y_n + (k1 + 2 \cdot k2 + 2 \cdot k3 + k4) \cdot h/6$
$k1 = f(xn, yn)$
$k2 = f(xn + h/2, yn + h/2 \cdot k1)$
$k3 = f(xn + h/2, yn + h/2 \cdot k2)$
$k4 = f(xn + h, yn + h \cdot k3)$

—

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

1. Если $a^{\frac{p-1}{2}} \neq 1$, return -1

2. Выбрать случайный $1 \leq i < p$

3. $T(x) = (x + i)^{(p-1)/2} mod(x^2 - a) = bx + c$

4. Если $b \neq 0$ то вернуть $\frac{c}{b}$, иначе к шагу 2)

—

Чтобы посчитать количество остовных деревьев в неориентированном графе G:
создать матрицу $N \times N$ mat, для каждого ребра (a, b):
`mat[a][a]++`, `mat[b][b]++`, `mat[a][b]-`,
`mat[b][a]-`.

Удалить последнюю строку и столбец, взять дискриминант.

—

Лемма Бернсайда:
Группа G действует на множество X Тогда число классов эквивалентности $= \frac{\sum_{g \in G} |f(g)|}{|G|}$, где $f(g)$ = число x (из X) : $g(x) == x$

—

Число простых быстрее $\mathcal{O}(n)$:
$dp(n, k)$ – число чисел от 1 до n в которых все простые $\geq p[k]$ $dp(n, 1) = n$, $dp(n, j) = dp(n, j+1) + dp(n/p[j], j)$, $\Rightarrow dp(n, j+1) = dp(n, j) - dp(n/p[j], j)$
Если $p[j]$, $p[k] > \sqrt{n}$, то $dp(n, j) + j == dp(n, k) + k$

Делаешь все оптимайзы сверху, но не считаешь глубже dp(n, k), n < K Потом фенвиком+сортировкой подсчитываешь за (K+Q)log все эти запросы Делаешь во второй раз, но на этот раз берешь прекальканные значения Если $\sqrt{n} < p[k] < n$, то (число простых до n) = $dp(n, k) + k - 1$

—

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

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

Numoceлки:

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: $B(p^m + n) = mB(n) + B(n+1)(mod p)$

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: $C_n = \binom{2n}{n}/(n+1) = \binom{2n+1}{n}/(2n+1) = \binom{2n}{n} - \binom{2n}{n-1}$

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: see partition.cpp

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

Stirling numbers of the second kind

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

$$prod(k = 1.. + inf)(1 - x^k) = \sum_{q=-inf}^{+inf}(-1)^q x^{(3q^2-q)/2}$$

$$\sum_{k=0}^{n} k\binom{n}{k} = n2^{n-1}$$

$$\sum_{j=0}^{k}\binom{m}{j}\binom{n-m}{k-j} = \binom{n}{k}$$

$$\sum_{j=0}^{m}\binom{m}{j}^2 = \binom{2m}{m}$$

$$\sum_{m=0}^{n}\binom{m}{j}\binom{n-m}{k-j} = \binom{n+1}{k+1}$$

$$\sum_{m=k}^{n}\binom{m}{k} = \binom{n+1}{k+1}$$

$$\sum_{k=0}^{\lfloor n/2 \rfloor}\binom{n-k}{k} = F(n+1)$$

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

$$\sum_{k=q}^{n}\binom{n}{k}\binom{k}{q} = 2^{n-q}\binom{n}{q}$$

$$\sum_{k=-a}^{a}(-1)^k\binom{a+b}{a+k}\binom{b+c}{b+k}\binom{c+a}{c+k} = \frac{(a+b+c)!}{a!b!c!}$$

Формулы:

$F(n, r) = rn^{n-1-r}$ — число лесов, у которых n вершин, r компонент и каждая компонента содержит свою вершину $i \in 1, 2, \ldots, r$.

$U_n = \sum_{k=3}^{n}\binom{n}{r}\frac{(r-1)!}{2} \cdot F(n, r)$ — число унициклов

$M_n = M_{n-1} + \sum_{i=0}^{n-2}M_i M_{n-2-i} = \frac{2n+1}{n+2}M_{n-1} + \frac{3n-3}{n+2}M_{n-2}$ — количество способов провести непересекающиеся диагонали среди n точек на круге.

$nD(n) = 3(2n-1)D(n-1) - (n-1)D(n-2)$

$D(m, n) = \sum_{k=0}^{\min(m,n)}\binom{m}{k}\binom{n}{k}2^k$ — количество путей черепашки с возможностью ходить по диагонали.

23

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Integrals with Logarithms

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

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

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

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

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

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

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

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

Integrals with Exponentials

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

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

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

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

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

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

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

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

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

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

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

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

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

Integrals with Trigonometric Functions

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

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

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

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

$$\int \cos ax\,dx = \frac{1}{a}\sin ax \qquad (67)$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\int \sec x \tan x\,dx = \sec x \qquad (85)$$

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

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

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

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

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

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

$$\int \sec x \csc x\,dx = \ln|\tan x| \qquad (92)$$

Products of Trigonometric Functions and Monomials

$$\int x \cos x\,dx = \cos x + x\sin x \qquad (93)$$

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

$$\int x^2 \cos x\,dx = 2x\cos x + (x^2 - 2)\sin x \qquad (95)$$

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

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

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

$$\int x \sin x\,dx = -x\cos x + \sin x \qquad (99)$$

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

$$\int x^2 \sin x\,dx = (2 - x^2)\cos x + 2x\sin x \qquad (101)$$

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

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

Products of Trigonometric Functions and Exponentials

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

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

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

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

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

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

Integrals of Hyperbolic Functions

$$\int \cosh ax\,dx = \frac{1}{a}\sinh ax \qquad (110)$$

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

$$\int \sinh ax\,dx = \frac{1}{a}\cosh ax \qquad (112)$$

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

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

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

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

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

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

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

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

$$\int \sinh ax \cosh bx\,dx = \frac{1}{b^2 - a^2}\left[b\cosh bx \sinh ax\right.$$
$$\left.-a\cosh ax \sinh bx\right] \qquad (121)$$