Содержание

final/template/vimrc.txt

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

final/template/template.cpp

```cpp
1  // team : SPb ITMO University 1
2  #include <bits/stdc++.h>
3
4  #define F first
5  #define S second
6  #define pb push_back
7  #define forn(i, n) for(int i = 0 ; (i) < (n) ; ++i)
8  #define eprintf(...) fprintf(stderr, __VA_ARGS__),↵
       fflush(stderr)
9  #define sz(a) ((int)(a).size())
10 #define all(a) (a).begin(),a.end()
11 #define pw(x) (1LL<<(x))
12
13 using namespace std;
14
15 typedef long long ll;
16 typedef double dbl;
17 typedef vector<int> vi;
18 typedef pair<int, int> pi;
19
20 const int INF = 1.01e9;
21 const dbl eps = 1e-9;
22
23 /* --- main part --- */
24
25
26
27
28
29
30
31 int main()
32 {
33 #define TASK ""
34 #ifdef home
35    assert(freopen(TASK".in", "r", stdin));
36    //assert(freopen(TASK".out", "w", stdout));
37 #endif
38
39
40
41
42
43 #ifdef home
44    eprintf("time = %d ms\n", (int)(clock() * 1000. / ↵
       CLOCKS_PER_SEC));
45 #endif
46    return 0;
47 }
```

final/template/fastIO.cpp

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

/** Interface */

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

/** Read */

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

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

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

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

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

inline int readInt() {
  int s = 1, c = readChar();
  int x = 0;
  if (c == '-') s = -1, c = getChar();
  while ('0' <= c && c <= '9') x = x * 10 + c - '0',
    c = getChar();
  return s == 1 ? x : -x;
}


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

final/template/hashTable.cpp

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

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

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

```cpp
    return f[i];
  }
};

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

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

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

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

final/template/optimizations.cpp

```cpp
// from anta code http://codeforces.com/contest/755/
//    submission/23864531

#pragma GCC optimize ("O3")
#pragma GCC target ("sse4")
inline void fasterLLDivMod(unsigned long long x,
    unsigned y, unsigned &out_d, unsigned &out_m) {
  unsigned xh = (unsigned)(x >> 32), xl = (unsigned)
    x, d, m;
#ifdef __GNUC__
  asm(
    "divl %4; \n\t"
    : "=a" (d), "=d" (m)
    : "d" (xh), "a" (xl), "r" (y)
  );
#else
  __asm {
    mov edx, dword ptr[xh];
    mov eax, dword ptr[xl];
    div dword ptr[y];
    mov dword ptr[d], eax;
    mov dword ptr[m], edx;
  };
#endif
  out_d = d; out_m = m;
}


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

final/template/Template.java

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

public class Template {
  FastScanner in;
  PrintWriter out;

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

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

      solve();

      out.close();
```

```java
21        } catch (IOException e) {
22          e.printStackTrace();
23        }
24    }
25
26    class FastScanner {
27      BufferedReader br;
28      StringTokenizer st;
29
30      FastScanner() {
31        br = new BufferedReader(new InputStreamReader(↩
   System.in));
32      }
33
34      String next() {
35        while (st == null || !st.hasMoreTokens()) {
36          try {
37            st = new StringTokenizer(br.readLine());
38          } catch (IOException e) {
39            e.printStackTrace();
40          }
41        }
42        return st.nextToken();
43      }
44
45      int nextInt() {
46        return Integer.parseInt(next());
47      }
48    }
49
50    public static void main(String[] arg) {
51      new Template().run();
52    }
53 }
```

final/numeric/fft.cpp

```cpp
1  namespace fft
2  {
3    const int maxBase = 21;
4    const int maxN = 1 << maxBase;
5
6    struct num
7    {
8      dbl x, y;
9      num(){}
10     num(dbl xx, dbl yy): x(xx), y(yy) {}
11     num(dbl alp): x(cos(alp)), y(sin(alp)) {}
12   };
13
14   inline num operator + (num a, num b) { return num(↩
   a.x + b.x, a.y + b.y); }
15   inline num operator - (num a, num b) { return num(↩
   a.x - b.x, a.y - b.y); }
16   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); ↩
   }
17   inline num conj(num a) { return num(a.x, -a.y); }
18
19   const dbl PI = acos(-1);
20
21   num root[maxN];
22   int rev[maxN];
23   bool rootsPrepared = false;
24
25   void prepRoots()
26   {
27     if (rootsPrepared) return;
28     rootsPrepared = true;
29     root[1] = num(1, 0);
30     for (int k = 1; k < maxBase; ++k)
31     {
32       num x(2 * PI / pw(k + 1));
33       for (int i = pw(k - 1); i < pw(k); ++i)
34       {
35         root[2 * i] = root[i];
36         root[2 * i + 1] = root[i] * x;
37       }
38     }
39   }
40
41   int base, N;
42
43   int lastRevN = -1;
44   void prepRev()
45   {
46     if (lastRevN == N) return;
47     lastRevN = N;
48     forn(i, N) rev[i] = (rev[i >> 1] >> 1) + ((i & ↩
   1) << (base - 1));
49   }
50
51   void fft(num *a, num *f)
52   {
53     forn(i, N) f[i] = a[rev[i]];
54     for (int k = 1; k < N; k <<= 1) for (int i = 0; ↩
   i < N; i += 2 * k) forn(j, k)
55     {
56       num z = f[i + j + k] * root[j + k];
57       f[i + j + k] = f[i + j] - z;
58       f[i + j] = f[i + j] + z;
59     }
60   }
61
62   num a[maxN], b[maxN], f[maxN], g[maxN];
63   ll A[maxN], B[maxN], C[maxN];
64
65   void _multMod(int mod)
66   {
67     forn(i, N)
68     {
69       int x = A[i] % mod;
70       a[i] = num(x & (pw(15) - 1), x >> 15);
71     }
72     forn(i, N)
73     {
74       int x = B[i] % mod;
75       b[i] = num(x & (pw(15) - 1), x >> 15);
76     }
77     fft(a, f);
78     fft(b, g);
79
80     forn(i, N)
81     {
82       int j = (N - i) & (N - 1);
```

```
 83       num a1 = (f[i] + conj(f[j])) * num(0.5, 0);
 84       num a2 = (f[i] - conj(f[j])) * num(0, -0.5);
 85       num b1 = (g[i] + conj(g[j])) * num(0.5 / N, 0)↩
     ;
 86       num b2 = (g[i] - conj(g[j])) * num(0, -0.5 / N↩
     );
 87       a[j] = a1 * b1 + a2 * b2 * num(0, 1);
 88       b[j] = a1 * b2 + a2 * b1;
 89     }
 90
 91     fft(a, f);
 92     fft(b, g);
 93
 94     forn(i, N)
 95     {
 96       ll aa = f[i].x + 0.5;
 97       ll bb = g[i].x + 0.5;
 98       ll cc = f[i].y + 0.5;
 99       C[i] = (aa + bb % mod * pw(15) + cc % mod * pw↩
     (30)) % mod;
100     }
101   }
102
103   void prepAB(int n1, int n2)
104   {
105     base = 1;
106     N = 2;
107     while (N < n1 + n2) base++, N <<= 1;
108
109     for (int i = n1; i < N; ++i) A[i] = 0;
110     for (int i = n2; i < N; ++i) B[i] = 0;
111
112     prepRoots();
113     prepRev();
114   }
115
116   void mult(int n1, int n2)
117   {
118     prepAB(n1, n2);
119     forn(i, N) a[i] = num(A[i], B[i]);
120     fft(a, f);
121     forn(i, N)
122     {
123       int j = (N - i) & (N - 1);
124       a[i] = (f[j] * f[j] - conj(f[i] * f[i])) * num↩
     (0, -0.25 / N);
125     }
126     fft(a, f);
127     forn(i, N) C[i] = (ll)round(f[i].x);
128   }
129
130
131   void multMod(int n1, int n2, int mod)
132   {
133     prepAB(n1, n2);
134     _multMod(mod);
135   }
136
137   int D[maxN];
138
139   void multLL(int n1, int n2)
140   {
141     prepAB(n1, n2);
142
143     int mod1 = 1.5e9;
144     int mod2 = mod1 + 1;
145
146     _multMod(mod1);
147
148     forn(i, N) D[i] = C[i];
149
150     _multMod(mod2);
151
152     forn(i, N)
153     {
154       C[i] = D[i] + (C[i] - D[i] + (ll)mod2) * (ll)↩
     mod1 % mod2 * mod1;
155     }
156   }
157   // HOW TO USE ::
158   // -- set correct maxBase
159   // -- use mult(n1, n2), multMod(n1, n2, mod) and ↩
     multLL(n1, n2)
160   // -- input : A[], B[]
161   // -- output : C[]
162 }
```

final/numeric/fftint.cpp

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

final/numeric/blackbox.cpp

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

```
20          z <<= 1;
21       }
22       return C[k];
23    }
24    // A — constant array
25    // magic(k, x):: B[k] = x, returns C[k]
26    // !! WARNING !! better to set N twice the size ↩
          needed
27 }
```

final/numeric/crt.cpp

```
1  int rev(int x, int m)
2  {
3     if (x == 1) return 1;
4     return (1 - rev(m % x, x) * (ll)m) / x + m;
5  }
6
7  int CRT(int a1, int m1, int a2, int m2)
8  {
9     return (a1 - a2 % m1 + m1) * (ll)rev(m2, m1) % m1 ↩
          * m2 + a2;
10 }
```

```
56
57       ll n = N;
58
59       for (int i = 0; i < math::pc && p[i] < MX; ++i) ↩
          if (n % p[i] == 0)
60       {
61          primes.pb(p[i]);
62          while (n % p[i] == 0) n /= p[i];
63       }
64
65       go(n);
66
67       sort(primes.begin(), primes.end());
68
69       vector<pair<ll, int>> res;
70       for (ll x : primes)
71       {
72          int cnt = 0;
73          while (N % x == 0)
74          {
75             cnt++;
76             N /= x;
77          }
78          res.push_back({x, cnt});
79       }
80       return res;
81    }
82 }
```

final/numeric/pollard.cpp

final/numeric/poly.cpp

```
1  namespace pollard
2  {
3     using math::p;
4
5     vector<pair<ll, int>> getFactors(ll N)
6     {
7        vector<ll> primes;
8
9        const int MX = 1e5;
10       const ll MX2 = MX * (ll)MX;
11
12       assert(MX <= math::maxP && math::pc > 0);
13
14       function<void(ll)> go = [&go, &primes](ll n)
15       {
16          for (ll x : primes) while (n % x == 0) n /= x;
17          if (n == 1) return;
18          if (n > MX2)
19          {
20             auto F = [&](ll x) {
21                ll k = ((long double)x * x) / n;
22                ll r = (x * x - k * n + 3) % n;
23                return r < 0 ? r + n : r;
24             };
25             ll x = mt19937_64()() % n, y = x;
26             const int C = 3 * pow(n, 0.25);
27
28             ll val = 1;
29             forn(it, C)
30             {
31                x = F(x), y = F(F(y));
32                if (x == y) continue;
33                ll delta = abs(x - y);
34                ll k = ((long double)val * delta) / n;
35                val = (val * delta - k * n) % n;
36                if (val < 0) val += n;
37                if (val == 0)
38                {
39                   ll g = __gcd(delta, n);
40                   go(g), go(n / g);
41                   return;
42                }
43                if ((it & 255) == 0)
44                {
45                   ll g = __gcd(val, n);
46                   if (g != 1)
47                   {
48                      go(g), go(n / g);
49                      return;
50                   }
51                }
52             }
53          }
54          primes.pb(n);
55       };
```

```
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 }
```

final/numeric/simplex.cpp

```
1
2   namespace simplex {
3     const int MAX_N = -1; // number of variables
4     const int MAX_M = -1; // number of inequalities
5     dbl a[MAX_M][MAX_N];
6     dbl b[MAX_M];
7     dbl c[MAX_N];
8     dbl v;
9     ll n, m;
10    int left[MAX_M];
11    int up[MAX_N];
12    int pos[MAX_N];
13    dbl res[MAX_N];
14
15    void init(int nn, int mm) {
16      n = nn;
17      m = mm;
18      v = 0;
19      for (int i = 0; i < m; i++)
20        for (int j = 0; j < n; j++)
21          a[i][j] = 0;
22      for (int i = 0; i < m; i++)
23        b[i] = 0;
24      for (int i = 0; i < n; i++)
25        c[i] = 0;
26    }
27
28    void pivot(int x, int y) {
29      swap(left[x], up[y]);
30      dbl k = a[x][y];
31      a[x][y] = 1;
32      b[x] /= k;
33      int cur = 0;
34      for (int i = 0; i < n; i++) {
35        a[x][i] = a[x][i] / k;
36        if (!eq(a[x][i], 0))
37          pos[cur++] = i;
38      }
39
40      for (int i = 0; i < m; i++) {
41        if (i == x || eq(a[i][y], 0)) continue;
42        dbl cof = a[i][y];
43        b[i] -= cof * b[x];
44        a[i][y] = 0;
45        for (int j = 0; j < cur; j++)
46          a[i][pos[j]] -= cof * a[x][pos[j]];
47      }
48      dbl cof = c[y];
49      v += cof * b[x];
50      c[y] = 0;
51      for (int i = 0; i < cur; i++) {
52        c[pos[i]] -= cof * a[x][pos[i]];
53      }
54    }
55
56    void solve() {
57      for (int i = 0; i < n; i++)
58        up[i] = i;
59      for (int i = 0; i < m; i++)
60        left[i] = i + n;
61
62      while (1) {
63        int x = -1;
64        for (int i = 0; i < m; i++)
65          if (ls(b[i], 0) && (x == -1 || b[i] < b[x]))↩
        {
66            x = i;
67          }
68        if (x == -1) break;
69        int y = -1;
70        for (int j = 0; j < n; j++)
71          if (ls(a[x][j], 0)) {
72            y = j;
73            break;
74          }
75        if (y == -1) {
76          assert(false); // no solution
77        }
78        pivot(x, y);
79      }
80      while (1) {
81        int y = -1;
82        for (int i = 0; i < n; i++)
83          if (ls(0, c[i]) && (y == -1 || (c[i] > c[y])↩
        )) {
84            y = i;
85          }
86        if (y == -1) break;
```

```
 87          int x = -1;
 88          for (int i = 0; i < m; i++) {
 89              if (ls(0, a[i][y])) {
 90                  if (x == -1 || (b[i] / a[i][y] < b[x] / a[↩
                    x][y])) {
 91                      x = i;
 92                  }
 93              }
 94          }
 95          if (y == -1) {
 96              assert(false); // infinite solution
 97          }
 98          pivot(x, y);
 99      }
100
101      memset(res, 0, sizeof(res));
102
103      for (int i = 0; i < m; i++) {
104          if (left[i] < n) {
105              res[left[i]] = b[i];
106          }
107      }
108  }
109  // HOW TO USE ::
110  // -- call init(n, m)
111  // -- call solve()
112  // -- variables in "up" equals to zero
113  // -- variables in "left" equals to b
114  // -- max: c * x
115  // -- b[i] >= a[i] * x
116  // -- answer in "v"
117  // -- sertificate in "res"
118 };
119
```

final/geom/commonTangents.cpp

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

final/geom/halfplaneIntersection.cpp

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

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

final/geom/minDisc.cpp

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

final/geom/convexHull3D-N2.cpp

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

final/geom/polygonArcCut.cpp

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

```
19          else
20            res.pb(p[i]);
21        }
22        if (p[i].S.type == 0) {
23          if (sign(l.v * (A - 1.0)) * sign(l.v * (B - 1.↵
    0)) == -1) {
24            pt FF = Line(A, B) * l;
25            res.pb(make_pair(FF, SEG));
26          }
27        }
28        else {
29          pt E, F;
30          if (intCL(p[i].S.O, p[i].S.R, l, E, F)) {
31            if (onArc(p[i].S.O, A, E, B))
32              res.pb({E, SEG});
33            if (onArc(p[i].S.O, A, F, B))
34              res.pb({F, p[i].S});
35          }
36        }
37      }
38      return res;
39  }
```

final/strings/eertree.cpp

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

```
10     memset(nxt, -1, sizeof(nxt));
11     memset(link, -1, sizeof(link));
12     memset(len, 0, sizeof(len));
13     last = 0;
14     sz = 1;
15   }
16
17   void add(int c) {
18     int cur = sz++;
19     len[cur] = len[last] + 1;
20     pos[cur] = len[cur];
21     int p = last;
22     last = cur;
23     for (; p != -1 && nxt[p][c] == -1; p = link[p]) ↵
    nxt[p][c] = cur;
24     if (p == -1) {
25       link[cur] = 0;
26       return;
27     }
28     int q = nxt[p][c];
29     if (len[p] + 1 == len[q]) {
30       link[cur] = q;
31       return;
32     }
33     int clone = sz++;
34     memcpy(nxt[clone], nxt[q], sizeof(nxt[q]));
35     len[clone] = len[p] + 1;
36     pos[clone] = pos[q];
37     link[clone] = link[q];
38     link[q] = link[cur] = clone;
39     for (; p != -1 && nxt[p][c] == q; p = link[p]) ↵
    nxt[p][c] = clone;
40   }
41
42   int n;
43   string s;
44   int l[MAXN], r[MAXN];
45   int e[MAXN][SIGMA];
46
47   void getSufTree(string _s) {
48     memset(e, -1, sizeof(e));
49     s = _s;
50     n = s.length();
51     reverse(s.begin(), s.end());
52     init();
53     for (int i = 0; i < n; i++) add(s[i] - 'a');
54     reverse(s.begin(), s.end());
55     for (int i = 1; i < sz; i++) {
56       int j = link[i];
57       l[i] = n - pos[i] + len[j];
58       r[i] = n - pos[i] + len[i];
59       e[j][s[l[i]] - 'a'] = i;
60     }
61   }
62 }
63
64 namespace duval {
65   string s;
66   int n = (int) s.length();
67   int i=0;
68   while (i < n) {
69     int j=i+1, k=i;
70     while (j < n && s[k] <= s[j]) {
71       if (s[k] < s[j])
72         k = i;
73       else
74         ++k;
75       ++j;
76     }
77     while (i <= k) {
78       cout << s.substr(i, j-k) << ' ';
79       i += j - k;
80     }
81   }
82 }
```

final/strings/sufAutomaton.cpp

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

final/graphs/centroid.cpp

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

final/graphs/dinica.cpp

```
1   namespace flow
2   {
3      const int maxn = 1e5 + 10;
4      const int maxe = 2 * maxn;
5
6      int head[maxn], next[maxe], to[maxe], f[maxe], ec ←
        = 1;
7      int ST, EN, N = maxn;
8
9      inline void setN(int n)
10     {
11        ST = n;
12        EN = n + 1;
13        N = n + 2;
14     }
15
16     inline void _add(int x, int y, int ff)
17     {
18        ++ec;
19        to[ec] = y;
20        next[ec] = head[x];
21        head[x] = ec;
22        f[ec] = ff;
23     }
24
25     inline int add(int x, int y, int ff)
26     {
27        _add(x, y, ff);
28        _add(y, x, 0);
29        return ec - 1;
30     }
31
32     void clear()
33     {
34        forn(i, N) head[i] = 0;
35        ec = 1;
36     }
37
38     int d[maxn];
39     int q[maxn], st = 0, en = 0;
40
41     int bfs()
42     {
43        forn(i, N) d[i] = 1e9;
44        st = 0, en = 0;
45        d[ST] = 0;
46        q[en++] = ST;
47        while (st < en)
```

```
48        {
49           int x = q[st++];
50           if (x == EN) return 1;
51           for (int e = head[x]; e; e = next[e])
52           {
53              int y = to[e];
54              if (d[y] == 1e9 && f[e])
55              {
56                 d[y] = d[x] + 1;
57                 q[en++] = y;
58              }
59           }
60        }
61        return 0;
62     }
63
64     int pushed;
65     int fst[maxn];
66
67     int dfs(int x, int flow = 1e9)
68     {
69        if (x == EN)
70        {
71           pushed = flow;
72           return 1;
73        }
74        for (; fst[x]; fst[x] = next[fst[x]])
75        {
76           int e = fst[x];
77           int y = to[e];
78           if (d[y] == d[x] + 1 && f[e] && dfs(y, min(f[e←
        ], flow)))
79           {
80              f[e] -= pushed;
81              f[e ^ 1] += pushed;
82              return 1;
83           }
84        }
85        return 0;
86     }
87
88
89     ll calcFlow()
90     {
91        ll res = 0;
92        while (bfs())
93        {
94           forn(i, N) fst[i] = head[i];
95           while (dfs(ST))
96           {
97              res += pushed;
98           }
99        }
100       return res;
101    }
102
103    // HOW TO USE ::
104    // -- set maxn and maxe (special for izban)
105    // -- add adges using add(x, y, f), call setN(n)
106    // -- run calcFlow
107 }
```

final/graphs/dominatorTree.cpp

```
1   namespace domtree {
2      const int MAXN = 300100;
3      int n;
4      vector<int> e[MAXN];
5      vector<int> g[MAXN];
6      int par[MAXN];
7      int in[MAXN], rin[MAXN], tmr;
8      int dom[MAXN], sdom[MAXN], cmn[MAXN];
9      int p[MAXN];
10     int adom[MAXN];
11     vector<int> vct[MAXN];
12
13     void init(int _n) {
14        n = _n;
15        for (int i = 0; i < n; i++) {
16           e[i].clear();
17        }
18     }
19
20     void addEdge(int from, int to) {
21        e[from].push_back(to);
22     }
23
```

```
24    void dfs(int v) {
25      in[v] = tmr++;
26      rin[in[v]] = v;
27      for (int to : e[v]) {
28        if (in[to] == -1) {
29          dfs(to);
30          par[in[to]] = in[v];
31        }
32        g[in[to]].push_back(in[v]);
33      }
34    }
35
36    int get(int u, int x = 0) {
37      if (u == p[u]) return x ? -1 : u;
38      int v = get(p[u], x + 1);
39      if (v < 0) return u;
40      if (sdom[cmn[p[u]]] < sdom[cmn[u]]) cmn[u] = cmn↩
      [p[u]];
41      p[u] = v;
42      return x ? v : cmn[u];
43    }
44
45    void uni(int u, int v) {
46      p[v] = u;
47    }
48
49    void calc() {
50      for (int i = 0; i < n; i++) {
51        in[i] = -1;
52        adom[i] = -1;
53        dom[i] = sdom[i] = p[i] = cmn[i] = i;
54        vct[i].clear();
55        g[i].clear();
56      }
57      tmr = 0;
58      dfs(0);
59      for (int i = tmr - 1; i >= 0; i--) {
60        for (int to : g[i]) sdom[i] = min(sdom[i], ↩
      sdom[get(to)]);
61        if (i > 0) vct[sdom[i]].push_back(i);
62        for (int w : vct[i]) {
63          int v = get(w);
64          if (sdom[v] == sdom[w]) dom[w] = sdom[w];
65          else dom[w] = v;
66        }
67        if (i > 0) uni(par[i], i);
68      }
69      for (int i = 1; i < tmr; i++) {
70        if (dom[i] != sdom[i]) dom[i] = dom[dom[i]];
71        adom[rin[i]] = rin[dom[i]];
72      }
73    }
74 }
```

final/graphs/fenwick-min.cpp

```
1  const int inf = 1.01e9;
2  const int maxn = 1e5;
3
4  namespace fenwik
5  {
6    const int N = maxn + 1;
7
8    int a[N], l[N], r[N];
9
10   void modify(int q, int v) {
11     q++;
12     a[q] = min(a[q], v);
13     int x = q;
14     while (x < N) {
15       l[x] = min(l[x], v);
16       x = (x | (x - 1)) + 1;
17     }
18     x = q;
19     while (x > 0) {
20       r[x] = min(r[x], v);
21       x &= x - 1;
22     }
23   }
24
25   int find_min(int ll, int rr) {
26     ll++;
27     rr++;
28     int res = inf;
29     int x = ll;
30     while ((x | (x - 1)) + 1 <= rr) {
31       res = min(res, r[x]);
```

```
32       x = (x | (x - 1)) + 1;
33     }
34     res = min(res, a[x]);
35     x = rr;
36     while ((x & (x - 1)) >= ll) {
37       res = min(res, l[x]);
38       x &= x - 1;
39     }
40     return res;
41   }
42
43   // indexes 0 .. maxn-1
44   // (!) to init fill (a, l, r) with INF
45   // (!) modify supports only decreasing of the ↩
      value
46   // find_min [l, r] (both inclusive)
47 }
```

final/graphs/generalMatching.cpp

```
1  //COPYPASTED FROM E-MAXX
2  namespace GeneralMatching {
3    const int MAXN = 256;
4    int n;
5    vector<int> g[MAXN];
6    int match[MAXN], p[MAXN], base[MAXN], q[MAXN];
7    bool used[MAXN], blossom[MAXN];
8
9    int lca (int a, int b) {
10     bool used[MAXN] = { 0 };
11     for (;;) {
12       a = base[a];
13       used[a] = true;
14       if (match[a] == -1) break;
15       a = p[match[a]];
16     }
17     for (;;) {
18       b = base[b];
19       if (used[b]) return b;
20       b = p[match[b]];
21     }
22   }
23
24   void mark_path (int v, int b, int children) {
25     while (base[v] != b) {
26       blossom[base[v]] = blossom[base[match[v]]] = ↩
        true;
27       p[v] = children;
28       children = match[v];
29       v = p[match[v]];
30     }
31   }
32
33   int find_path (int root) {
34     memset (used, 0, sizeof used);
35     memset (p, -1, sizeof p);
36     for (int i=0; i<n; ++i)
37       base[i] = i;
38
39     used[root] = true;
40     int qh=0, qt=0;
41     q[qt++] = root;
42     while (qh < qt) {
43       int v = q[qh++];
44       for (size_t i=0; i<g[v].size(); ++i) {
45         int to = g[v][i];
46         if (base[v] == base[to] || match[v] == to) ↩
          continue;
47         if (to == root || (match[to] != -1 && p[↩
          match[to]] != -1)) {
48           int curbase = lca (v, to);
49           memset (blossom, 0, sizeof blossom);
50           mark_path (v, curbase, to);
51           mark_path (to, curbase, v);
52           for (int i=0; i<n; ++i)
53             if (blossom[base[i]]) {
54               base[i] = curbase;
55               if (!used[i]) {
56                 used[i] = true;
57                 q[qt++] = i;
58               }
59             }
60         }
61         else if (p[to] == -1) {
62           p[to] = v;
63           if (match[to] == -1)
64             return to;
```

```
65              to = match[to];
66              used[to] = true;
67              q[qt++] = to;
68            }
69          }
70        }
71        return -1;
72      }
73
74      vector<pair<int, int> > solve(int _n, vector<pair<
         int, int> > edges) {
75        n = _n;
76        for (int i = 0; i < n; i++) g[i].clear();
77        for (auto o : edges) {
78          g[o.first].push_back(o.second);
79          g[o.second].push_back(o.first);
80        }
81        memset(match, -1, sizeof match);
82        for (int i=0; i<n; ++i) {
83          if (match[i] == -1) {
84            int v = find_path (i);
85            while (v != -1) {
86              int pv = p[v], ppv = match[pv];
87              match[v] = pv, match[pv] = v;
88              v = ppv;
89            }
90          }
91        }
92        vector<pair<int, int> > ans;
93        for (int i = 0; i < n; i++) {
94          if (match[i] > i) {
95            ans.push_back(make_pair(i, match[i]));
96          }
97        }
98        return ans;
99      }
100   }
```

final/graphs/heavyLight.cpp

```
1   namespace hld {
2     const int N = 1 << 17;
3     int par[N], heavy[N], h[N];
4     int root[N], pos[N];
5     int n;
6     vector<vector<int> > e;
7     segtree tree;
8
9     int dfs(int v) {
10      int sz = 1, mx = 0;
11      for (int to : e[v]) {
12        if (to == par[v]) continue;
13        par[to] = v;
14        h[to] = h[v] + 1;
15        int cur = dfs(to);
16        if (cur > mx) heavy[v] = to, mx = cur;
17        sz += cur;
18      }
19      return sz;
20    }
21
22    template <typename T>
23    void path(int u, int v, T op) {
24      for (; root[u] != root[v]; v = par[root[v]]) {
25        if (h[root[u]] > h[root[v]]) swap(u, v);
26        op(pos[root[v]], pos[v] + 1);
27      }
28      if (h[u] > h[v]) swap(u, v);
29      op(pos[u], pos[v] + 1);
30    }
31
32    void init(vector<vector<int> > _e) {
33      e = _e;
34      n = e.size();
35      tree = segtree(n);
36      memset(heavy, -1, sizeof(heavy[0]) * n);
37      par[0] = -1;
38      h[0] = 0;
39      dfs(0);
40      for (int i = 0, cpos = 0; i < n; i++) {
41        if (par[i] == -1 || heavy[par[i]] != i) {
42          for (int j = i; j != -1; j = heavy[j]) {
43            root[j] = i;
44            pos[j] = cpos++;
45          }
46        }
47      }
```

```
48    }
49
50    void add(int v, int x) {
51      tree.add(pos[v], x);
52    }
53
54    int get(int u, int v) {
55      int res = 0;
56      path(u, v, [&](int l, int r) {
57        res = max(res, tree.get(l, r));
58      });
59      return res;
60    }
61  }
```

final/graphs/hungary.cpp

```
1   namespace hungary
2   {
3     const int N = 210;
4
5     int a[N][N];
6     int ans[N];
7
8     int calc(int n, int m)
9     {
10      ++n, ++m;
11      vi u(n), v(m), p(m), prev(m);
12      for (int i = 1; i < n; ++i)
13      {
14        p[0] = i;
15        int x = 0;
16        vi mn(m, inf);
17        vi was(m, 0);
18        while (p[x])
19        {
20          was[x] = 1;
21          int ii = p[x], dd = inf, y = 0;
22          for (int j = 1; j < m; ++j) if (!was[j])
23          {
24            int cur = a[ii][j] - u[ii] - v[j];
25            if (cur < mn[j]) mn[j] = cur, prev[j] = x;
26            if (mn[j] < dd) dd = mn[j], y = j;
27          }
28          forn(j, m)
29          {
30            if (was[j]) u[p[j]] += dd, v[j] -= dd;
31            else mn[j] -= dd;
32          }
33          x = y;
34        }
35        while (x)
36        {
37          int y = prev[x];
38          p[x] = p[y];
39          x = y;
40        }
41      }
42      for (int j = 1; j < m; ++j)
43      {
44        ans[p[j]] = j;
45      }
46      return -v[0];
47    }
48    // HOW TO USE ::
49    // -- set values to a[1..n][1..m] (n <= m)
50    // -- run calc(n, m) to find MINIMUM
51    // -- to restore permutation use ans[]
52    // -- everything works on negative numbers
53    //
54    // !! i don't understand this code, it's
         copypasted from e-maxx (and rewrited by enot110
         )
55  }
```

final/graphs/max-flow-min-cost.cpp

```
1   namespace flow
2   {
3     const int maxn = 2e5 + 10;
4     const int maxe = 2 * maxn;
```

```
5
6     int head[maxn], next[maxe], to[maxe], flow[maxe], ←
        cost[maxe], ec = 1;
7     int ST, EN, N = maxn;
8
9     inline void setN(int n)
10    {
11      ST = n;
12      EN = n + 1;
13      N = n + 2;
14    }
15
16    inline void _add(int x, int y, int f, int c)
17    {
18      ++ec;
19      to[ec] = y;
20      next[ec] = head[x];
21      head[x] = ec;
22      flow[ec] = f;
23      cost[ec] = c;
24    }
25
26    inline int add(int x, int y, int f, int c)
27    {
28      _add(x, y, f, c);
29      _add(y, x, 0, -c);
30      return ec - 1;
31    }
32
33    void clear()
34    {
35      forn(i, N) head[i] = 0;
36      ec = 1;
37    }
38
39    ll d[maxn], p[maxn];
40    int last[maxn];
41    int used[maxn];
42
43    pair<ll, ll> _calc(int flag)
44    {
45      const ll INF = 1e12;
46      forn(i, N) p[i] = INF;
47      p[ST] = 0;
48      forn(_, N) forn(x, N) for (int e = head[x]; e; e←
        = next[e]) if (flow[e] > 0)
49      {
50        int y = to[e];
51        if (p[y] > p[x] + cost[e])
52        {
53          p[y] = p[x] + cost[e];
54        }
55      }
56
57      ll resFlow = 0, resCost = 0;
58      while (1)
59      {
60        forn(i, N) d[i] = INF, used[i] = 0;
61        d[ST] = 0;
62        forn(_, N)
63        {
64          int x = -1;
65          forn(i, N) if (!used[i] && (x == -1 || d[x] ←
        > d[i])) x = i;
66          used[x] = 1;
67          if (d[x] == INF) break;
68          for (int e = head[x]; e; e = next[e]) if (←
        flow[e] > 0)
69          {
70            int y = to[e];
71            ll len = cost[e] + p[x] - p[y];
72            if (d[y] > d[x] + len)
73            {
74              d[y] = d[x] + len;
75              last[y] = e;
76            }
77          }
78        }
79
80        if (d[EN] == INF) break;
81
82        ll realCost = d[EN] + p[EN] - p[ST];
83        if (flag && realCost > 0) break;
84
85        int pushed = inf;
86        int x = EN;
87        while (x != ST)
88        {
89          int e = last[x];
90          pushed = min(pushed, flow[e]);
91          x = to[e ^ 1];
92        }
93
94        resCost += realCost * pushed;
95        resFlow += pushed;
96
97        x = EN;
98        while (x != ST)
99        {
100           int e = last[x];
101           flow[e] -= pushed;
102           flow[e ^ 1] += pushed;
103           x = to[e ^ 1];
104         }
105
106         forn(i, N) p[i] += d[i];
107       }
108       return mp(resFlow, resCost);
109     }
110
111     pair<ll, ll> maxFlow()
112     {
113       return _calc(0);
114     }
115
116     pair<ll, ll> minCost()
117     {
118       return _calc(1);
119     }
120
121     // HOW TO USE::
122     // -- add adges using add(x, y, f, c), call setN(n←
        )
123     // -- run maxFlow/minCost, returns pair(flow, cost←
        )
124   }
```

final/graphs/retro.cpp

```
1     namespace retro
2     {
3       const int N = 4e5 + 10;
4
5       vi v[N];
6       vi vrev[N];
7
8       void add(int x, int y)
9       {
10        v[x].pb(y);
11        vrev[y].pb(x);
12      }
13
14      const int UD = 0;
15      const int WIN = 1;
16      const int LOSE = 2;
17
18      int res[N];
19      int moves[N];
20      int deg[N];
21      int q[N], st, en;
22
23      void calc(int n)
24      {
25        forn(i, n) deg[i] = sz(v[i]);
26        st = en = 0;
27        forn(i, n) if (!deg[i])
28        {
29          q[en++] = i;
30          res[i] = LOSE;
31        }
32        while (st < en)
33        {
34          int x = q[st++];
35          for (int y : vrev[x])
36          {
37            if (res[y] == UD && (res[x] == LOSE || (--←
        deg[y] == 0 && res[x] == WIN)))
38            {
39              res[y] = 3 - res[x];
40              moves[y] = moves[x] + 1;
41              q[en++] = y;
42            }
43          }
44        }
45      }
46    }
```

final/graphs/smith.cpp

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

struct graph
{
  int n;

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

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

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

  set<int> s[N];

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

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

final/graphs/twoChinese.cpp

```cpp
const int INF = 1e9;
struct Edge {
  int from, to, w, id;
};
namespace dmst {
  int n;
  vector<int> p;
  vector<Edge> edges;

  int get(int x) {
    if (x == p[x]) return x;
    return p[x] = get(p[x]);
  }

  void uni(int u, int v) {
    p[get(v)] = get(u);
  }

  vector<Edge> solve() {
    vector<int> id(n, -1);
    vector<int> vert;
    int cn = 0;
    for (int i = 0; i < n; i++) if (get(i) == i) {
      vert.push_back(i);
      id[i] = cn++;
    }
    if (cn == 1) return vector<Edge>();

    vector<vector<int> > e(cn);
    for (int i = 0; i < (int)edges.size(); i++) {
      if (get(edges[i].to) != get(edges[i].from)) {
        e[id[get(edges[i].to)]].push_back(i);
      }
    }

    vector<int> nxtId(cn, -1);
    for (int i = 0; i < cn; i++) {
      int mn = INF;
      for (int id : e[i]) mn = min(mn, edges[id].w);
      for (int id : e[i]) {
        edges[id].w -= mn;
        if (edges[id].w == 0) nxtId[i] = id;
      }
    }

    vector<char> vis(cn);
    vis[0] = 1;
    int cur = 1;
    while (!vis[cur]) {
      vis[cur] = 1;
      cur = id[get(edges[nxtId[cur]].from)];
    }
    vector<Edge> ans;
    if (cur == 0) {
      for (int i = 0; i < cn; i++) {
        if (vis[i] && i != 0) {
          ans.push_back(edges[nxtId[i]]);
          uni(0, vert[i]);
        }
      }
      auto nans = solve();
      for (auto ee : nans) ans.push_back(ee);
      return ans;
    }
    vector<int> cp = p;
    int o = cur;
    while (1) {
      uni(vert[o], vert[cur]);
      ans.push_back(edges[nxtId[cur]]);
      int to = id[get(edges[nxtId[cur]].from)];
      if (to == o) break;
      cur = to;
    }
    vector<Edge> nedges = solve();
```

```
75       p = cp;
76       vector<char> covered(cn);
77       for (auto ee : nedges) covered[id[get(ee.to)]] =↵
          1;
78       for (auto ee : ans) if (!covered[id[get(ee.to)↵
         ]]) nedges.push_back(ee);
79       return nedges;
80     }
81
82     // root is 0
83     vector<Edge> getMst(int _n, vector<Edge> _edges) {
84       n = _n;
85       edges = _edges;
86       p.resize(n);
87       for (int i = 0; i < n; i++) p[i] = i;
88
89       return solve();
90     }
91 }
```

final/graphs/linkcut.cpp

```
 1 #include <iostream>
 2 #include <cstdio>
 3 #include <cassert>
 4
 5 using namespace std;
 6
 7 // BEGIN ALGO
 8
 9 const int MAXN = 110000;
10
11 typedef struct _node{
12  _node *l, *r, *p, *pp;
13  int size; bool rev;
14  _node();
15  explicit _node(nullptr_t){
16   l = r = p = pp = this;
17   size = rev = 0;
18  }
19  void push(){
20   if (rev){
21    l->rev ^= 1; r->rev ^= 1;
22    rev = 0; swap(l,r);
23   }
24  }
25  void update();
26 }* node;
27 node None = new _node(nullptr);
28 node v2n[MAXN];
29 _node::_node(){
30  l = r = p = pp = None;
31  size = 1; rev = false;
32 }
33 void _node::update(){
34  size = (this != None) + l->size + r->size;
35  l->p = r->p = this;
36 }
37 void rotate(node v){
38  assert(v != None && v->p != None);
39  assert(!v->rev); assert(!v->p->rev);
40  node u = v->p;
41  if (v == u->l)
42   u->l = v->r, v->r = u;
43  else
44   u->r = v->l, v->l = u;
45  swap(u->p,v->p); swap(v->pp,u->pp);
46  if (v->p != None){
47   assert(v->p->l == u || v->p->r == u);
48   if (v->p->r == u) v->p->r = v;
49   else v->p->l = v;
50  }
51  u->update(); v->update();
52 }
53 void bigRotate(node v){
54  assert(v->p != None);
55  v->p->p->push();
56  v->p->push();
57  v->push();
58  if (v->p->p != None){
59   if ((v->p->l == v) ^ (v->p->p->r == v->p))
60    rotate(v->p);
61   else
62    rotate(v);
63  }
64  rotate(v);
65 }
66 inline void Splay(node v){
67  while (v->p != None) bigRotate(v);
68 }
69 inline void splitAfter(node v){
70  v->push();
71  Splay(v);
72  v->r->p = None;
73  v->r->pp = v;
74  v->r = None;
75  v->update();
76 }
77 void expose(int x){
78  node v = v2n[x];
79  splitAfter(v);
80  while (v->pp != None){
81   assert(v->p == None);
82   splitAfter(v->pp);
83   assert(v->pp->r == None);
84   assert(v->pp->p == None);
85   assert(!v->pp->rev);
86   v->pp->r = v;
87   v->pp->update();
88   v = v->pp;
89   v->r->pp = None;
90  }
91  assert(v->p == None);
92  Splay(v2n[x]);
93 }
94 inline void makeRoot(int x){
95  expose(x);
96  assert(v2n[x]->p == None);
97  assert(v2n[x]->pp == None);
98  assert(v2n[x]->r == None);
99  v2n[x]->rev ^= 1;
100 }
101 inline void link(int x,int y){
102  makeRoot(x); v2n[x]->pp = v2n[y];
103 }
104 inline void cut(int x,int y){
105  expose(x);
106  Splay(v2n[y]);
107  if (v2n[y]->pp != v2n[x]){
108   swap(x,y);
109   expose(x);
110   Splay(v2n[y]);
111   assert(v2n[y]->pp == v2n[x]);
112  }
113  v2n[y]->pp = None;
114 }
115 inline int get(int x,int y){
116  if (x == y) return 0;
117  makeRoot(x);
118  expose(y); expose(x);
119  Splay(v2n[y]);
120  if (v2n[y]->pp != v2n[x]) return -1;
121  return v2n[y]->size;
122 }
123 // END ALGO
124
125 _node mem[MAXN];
126
127
128 int main(){
129  freopen("linkcut.in","r",stdin);
130  freopen("linkcut.out","w",stdout);
131
132  int n,m;
133  scanf("%d %d",&n,&m);
134
135  for (int i = 0; i < n; i++)
136   v2n[i] = &mem[i];
137
138  for (int i = 0; i < m; i++){
139   int a,b;
140   if (scanf(" link %d %d",&a,&b) == 2)
141    link(a-1,b-1);
142   else if (scanf(" cut %d %d",&a,&b) == 2)
143    cut(a-1,b-1);
144   else if (scanf(" get %d %d",&a,&b) == 2)
145    printf("%d\n",get(a-1,b-1));
146   else
147    assert(false);
148  }
149  return 0;
150 }
```

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

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

—

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

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

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

Методы Адамса-Башфорта

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

—

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

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

—

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

Не заходит FFT по TL-ю − чекнуть что стоит double, а не long double

mt19937 генерит случайный unsigned int, если хочется больше есть mt19937_64

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

Перед сабмитом чекнуть что все выводится в printf, а не eprintf!!!

—

Лемма Бернсайда:

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

—

Число простых быстрее O(n):

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

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

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

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

—

Числелки:

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

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

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

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

Table of Integrals*

Basic Forms

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

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

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

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

Integrals of Rational Functions

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

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

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

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

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

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

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

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

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

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

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

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

Integrals with Roots

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Integrals with Logarithms

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

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

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

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

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

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

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

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

Integrals with Exponentials

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

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

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

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

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

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

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

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

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

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

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

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

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

Integrals with Trigonometric Functions

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Products of Trigonometric Functions and Monomials

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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