

Содержание

	45	final/graphs/chordaltree.cpp	18
	46	final/graphs/minimization.cpp	18
	47	final/graphs/matroidIntersection.cpp	18
1	2	final/template/vimrc.txt	
2	2	final/template/template.cpp	
3	2	Practice round	
4	2	final/template/fastIO.cpp	
5	2	final/template/hashTable.cpp	
6	3	final/template/optimizations.cpp	
7	3	final/template/useful.cpp	
8	3	final/template/Template.java	
9	3	final/template/bitset.cpp	
10	4	final/numeric/fft.cpp	
11	5	final/numeric/fftint.cpp	
12	5	final/numeric/blackbox.cpp	
13	6	final/numeric/crt.cpp	
14	6	final/numeric/mulMod.cpp	
15	6	final/numeric/modReverse.cpp	
16	6	final/numeric/pollard.cpp	
17	6	final/numeric/poly.cpp	
18	7	final/numeric/simplex.cpp	
19	8	final/numeric/sumLine.cpp	
20	8	final/numeric/berlekamp.cpp	
21	8	final/numeric/integrate.cpp	
22	9	final/geom/commonTangents.cpp	
23	9	final/geom/halfplaneIntersection.cpp	
24	9	final/geom/minDisc.cpp	
25	10	final/geom/convexHull3D-N2.cpp	
26	10	final/geom/polygonArcCut.cpp	
27	10	final/geom/polygonTangent.cpp	
28	10	final/geom/checkPlaneInt.cpp	
29	11	final/geom/furthestPoints.cpp	
30	11	final/strings/eertree.cpp	
31	11	final/strings/sufAutomaton.cpp	
32	11	final/strings/sufArray.cpp	
33	12	final/strings/duval.cpp	
34	12	final/graphs/centroid.cpp	
35	12	final/graphs/dominatorTree.cpp	
36	13	final/graphs/generalMatching.cpp	
37	13	final/graphs/heavyLight.cpp	
38	14	final/graphs/hungary.cpp	
39	14	final/graphs/minCostNegCycle.cpp	
40	15	final/graphs/retro.cpp	
41	15	final/graphs/smith.cpp	
42	16	final/graphs/mincut.cpp	
43	16	final/graphs/twoChineseFast.cpp	
44	17	final/graphs/linkcut.cpp	

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 -<
02 <CR>
3 map <F8> :wall! <CR> :!ulimit -s 500000 && ./%:r <CR>
>
4 map <F10> :wall! <CR> :!g++ -Wall -Wextra -Wshadow -<
Wno-unused-result -o %:r % -std=c++14 -DHOME -<
D_GLIBCXX_DEBUG -fsanitize=address -g && gdb -<
./%:r <CR>
5
6 inoremap {<CR> {<CR>}<ESC>>O
7 map <c-a> ggVG
8
9 set nu
10 set rnu
11 syntax on
12
13 map <c-t> :tabnew <CR>
14 map <c-l> :tabn <CR>
15 map <c-h> :tabp <CR>
16
17 set cin
18 set sw=4
19 set so=99
20 set bs=2
21 set et
22 set sts=4

```

final/template/template.cpp

```

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

```

Practice round

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

final/template/fastIO.cpp

```

1 #include <cstdio>
2 #include <algorithm>
3
4 /** Interface */
5
6 inline int readInt();
7 inline int readUInt();
8 inline bool isEof();
9
10 /** Read */
11
12 static const int buf_size = 100000;
13 static char buf[buf_size];
14 static int buf_len = 0, pos = 0;
15
16 inline bool isEof() {
17     if (pos == buf_len) {
18         pos = 0, buf_len = fread(buf, 1, buf_size, stdin);
19         if (pos == buf_len) return 1;
20     }
21     return 0;
22 }
23
24 inline int getChar() { return isEof() ? -1 : buf[pos++]; }
25
26 inline int readChar() {
27     int c = getChar();
28     while (c != -1 && c <= 32) c = getChar();
29     return c;
30 }
31
32 inline int readUInt() {
33     int c = readChar(), x = 0;
34     while ('0' <= c && c <= '9') x = x * 10 + c - '0', c =
getChar();
35     return x;
36 }
37
38 inline int readInt() {
39     int s = 1, c = readChar();
40     int x = 0;
41     if (c == '-') s = -1, c = getChar();
42     while ('0' <= c && c <= '9') x = x * 10 + c - '0', c =
getChar();
43     return s == 1 ? x : -x;
44 }
45
46
47 // 10M int [0..1e9)
48 // cin 3.02
49 // scanf 1.2
50 // cin_sync_with_stdio(false) 0.71
51 // fastRead getchar 0.53
52 // fastRead fread 0.15

```

final/template/hashTable.cpp

```

1  template <const int max_size, class HashType, class ↵
      Data, const Data default_value>
2  struct hashTable {
3      HashType hash[max_size];
4      Data f[max_size];
5      int size;
6
7      int position(HashType H) const {
8          int i = H % max_size;
9          while (hash[i] && hash[i] != H)
10             if (++i == max_size)
11                 i = 0;
12             return i;
13     }
14
15     Data & operator [] (HashType H) {
16         assert(H != 0);
17         int i = position(H);
18         if (!hash[i]) {
19             hash[i] = H;
20             f[i] = default_value;
21             size++;
22         }
23         return f[i];
24     }
25 };
26
27 hashTable<13, int, int, 0> h;

```

final/template/optimizations.cpp

```

1  inline void fasterLLDivMod(unsigned long long x, ↵
      unsigned y, unsigned &out_d, unsigned &out_m) {
2      unsigned xh = (unsigned)(x >> 32), xl = (unsigned)↵
      x, d, m;
3  #ifdef __GNUC__
4      asm(
5          "divl %4; \n\t"
6          : "=a" (d), "=d" (m)
7          : "d" (xh), "a" (xl), "r" (y)
8      );
9  #else
10     __asm {
11         mov edx, dword ptr[xh];
12         mov eax, dword ptr[xl];
13         div dword ptr[y];
14         mov dword ptr[d], eax;
15         mov dword ptr[m], edx;
16     };
17 #endif
18     out_d = d; out_m = m;
19 }
20
21 // have no idea what sse flags are really cool; list ↵
22 // — very good with bitsets
23 #pragma GCC optimize("O3")
24 #pragma GCC target("sse,sse2,sse3,ssse3,sse4,popcnt,↵
      abm,mmx")

```

final/template/useful.cpp

```

1  #include "ext/pb_ds/assoc_container.hpp"
2  using namespace __gnu_pbds;
3
4  template <typename T> using ordered_set = tree<T, ↵
      null_type, less<T>, rb_tree_tag, ↵
      tree_order_statistics_node_update>;
5  template <typename K, typename V> using ordered_map ↵
      = tree<K, V, less<K>, rb_tree_tag, ↵
      tree_order_statistics_node_update>;
6
7  // HOW TO USE ::
8  // — order_of_key(10) returns the number of ↵
9  // — *find_by_order(10) returns 10-th smallest ↵
10 // — element in set/map (0-based)
11
12 bitset<N> a;
13 for (int i = a._Find_first(); i != a.size(); i = a.↵
      _Find_next(i)) {

```

```

13     cout << i << endl;
14 }

```

final/template/Template.java

```

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

```

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, ←
37         0); i++) {
38         res.data[i + s] |= (data[i] & mask) << rem;
39     }
40     if (rem != 0) {
41         for (int i = max(0, -s - 1); i < sz(data) - ←
42             max(s + 1, 0); i++) {
43             res.data[i + s + 1] |= (data[i] >> p1) & (pw←
44                 (rem) - 1);
45         }
46     }
47     int cc = data.size() * BASE - n;
48     res.data.back() <<= cc;
49     res.data.back() >>= cc;
50     return res;
51 }

```

final/numeric/fft.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(←
15        a.x + b.x, a.y + b.y); }
16    inline num operator - (num a, num b) { return num(←
17        a.x - b.x, a.y - b.y); }
18    inline num operator * (num a, num b) { return num(←
19        a.x * b.x - a.y * b.y, a.x * b.y + a.y * b.x); ←
20    };
21    inline num conj(num a) { return num(a.x, -a.y); }
22
23    const dbl PI = acos(-1);
24
25    num root[maxN];
26    int rev[maxN];
27    bool rootsPrepared = false;
28
29    void prepRoots()
30    {
31        if (rootsPrepared) return;
32        rootsPrepared = true;
33        root[1] = num(1, 0);
34        for (int k = 1; k < maxBase; ++k)
35        {
36            num x(2 * PI / pw(k + 1));
37            for (int i = pw(k - 1); i < pw(k); ++i)
38            {
39                root[2 * i] = root[i];
40                root[2 * i + 1] = root[i] * x;
41            }
42        }
43    }
44
45    int base, N;
46
47    int lastRevN = -1;
48    void prepRev()
49    {
50        if (lastRevN == N) return;
51        lastRevN = N;
52        forn(i, N) rev[i] = (rev[i >> 1] >> 1) + ((i & ←
53            1) << (base - 1));
54    }
55
56    void fft(num *a, num *f)
57    {
58        forn(i, N) f[i] = a[rev[i]];
59        for (int k = 1; k < N; k <<= 1) for (int i = 0; ←
60            i < N; i += 2 * k) forn(j, k)
61        {
62            num z = f[i + j + k] * root[j + k];
63            f[i + j + k] = f[i + j] - z;
64            f[i + j] = f[i + j] + z;
65        }
66    }
67
68    num a[maxN], b[maxN], f[maxN], g[maxN];
69    ll A[maxN], B[maxN], C[maxN];
70
71    void _multMod(int mod)
72    {
73        forn(i, N)
74        {
75            int x = A[i] % mod;
76            a[i] = num(x & (pw(15) - 1), x >> 15);
77        }
78        forn(i, N)
79        {
80            int x = B[i] % mod;
81            b[i] = num(x & (pw(15) - 1), x >> 15);
82        }
83        fft(a, f);
84        fft(b, g);
85
86        forn(i, N)
87        {
88            int j = (N - i) & (N - 1);
89

```

```

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

```

final/numeric/blackbox.cpp

```

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

```

```

17     forn(i, z) fft::B[i] = B[k - z + 1 + i];
18     fft::multMod(z, z, mod);
19     forn(i, 2 * z - 1) C[k + 1 + i] = (C[k + 1 + i] ←
20     ] + fft::C[i]) % mod;
21     z <= 1;
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 ←
27 // needed

```

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 ←
3     * m2 + a2;

```

final/numeric/mulMod.cpp

```

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

```

final/numeric/modReverse.cpp

```

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

```

final/numeric/pollard.cpp

```

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

```

```

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        primes.pb(n);
54    };
55
56    ll n = N;
57
58    for (int i = 0; i < math::pc && p[i] < MX; ++i) ←
59    {
60        if (n % p[i] == 0)
61        {
62            primes.pb(p[i]);
63            while (n % p[i] == 0) n /= p[i];
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/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}) - ->
125             R)).cut(k);
126         R.v.resize(2 * k);
127         forn(i, k) R[i + k] = R1[i];
128     }
129     return R.cut(n).norm();
130 }
131
132 pair<poly, poly> divide(poly A, poly B)
133 {
134     if (sz(A) < sz(B)) return {poly({0}), A};
135
136     auto rev = [](poly f)
137     {
138         reverse(all(f.v));
139         return f;
140     };
141
142     poly q = rev((inv(rev(B), sz(A) - sz(B) + 1) * rev(A)
143         (A)).cut(sz(A) - sz(B) + 1));
144     poly r = A - B * q;
145     return {q, r};
146 }

```

final/numeric/simplex.cpp

```

1 vector<double> simplex(vector<vector<double>> a) {
2     int n = a.size() - 1;
3     int m = a[0].size() - 1;
4     vector<int> left(n + 1), up(m + 1);
5     iota(up.begin(), up.end(), 0);
6     iota(left.begin(), left.end(), m);
7     auto pivot = [&](int x, int y) {
8         swap(left[x], up[y]);
9         double k = a[x][y];
10        a[x][y] = 1;
11        vector<int> vct;
12        for (int j = 0; j <= m; j++) {
13            a[x][j] /= k;
14            if (!eq(a[x][j], 0)) vct.push_back(j);
15        }
16        for (int i = 0; i <= n; i++) {
17            if (eq(a[i][y], 0) || i == x) continue;
18            k = a[i][y];
19            a[i][y] = 0;
20            for (int j : vct) a[i][j] -= k * a[x][j];
21        }
22    };
23    while (1) {
24        int x = -1;
25        for (int i = 1; i <= n; i++) if (ls(a[i][0], 0) <->
26            && (x == -1 || a[i][0] < a[x][0])) x = i;
27        if (x == -1) break;
28        int y = -1;
29        for (int j = 1; j <= m; j++) if (ls(a[x][j], 0) <->
30            && (y == -1 || a[x][j] < a[x][y])) y = j;
31        if (y == -1) assert(0); // infeasible
32        pivot(x, y);
33    }
34    while (1) {
35        int y = -1;
36        for (int j = 1; j <= m; j++) if (ls(0, a[0][j]) <->
37            && (y == -1 || a[0][j] > a[0][y])) y = j;
38        if (y == -1) break;
39        int x = -1;
40        for (int i = 1; i <= n; i++) if (ls(0, a[i][y]) <->
41            && (x == -1 || a[i][0] / a[i][y] < a[x][0] / a[
42            x][y])) x = i;
43        if (x == -1) assert(0); // unbounded
44        pivot(x, y);
45    }
46    vector<double> ans(m + 1);
47    for (int i = 1; i <= n; i++) if (left[i] <= m) ans<->
48        [left[i]] = a[i][0];
49    ans[0] = -a[0][0];
50    return ans;
51 }
52
53 // j=1..m: x[j]>=0
54 // i=1..n: sum(j=1..m) A[i][j]*x[j] <= A[i][0]
55 // max sum(j=1..m) A[0][j]*x[j]
56 // res[0] is answer
57 // res[1..m] is certificate

```

final/numeric/sumLine.cpp

```

1 // sum(i=0..n-1) (a+b*i) div m
2 ll solve(ll n, ll a, ll b, ll m) {
3     if (b == 0) return n * (a / m);
4     if (a >= m) return n * (a / m) + solve(n, a % m, b ←
5         , m);
6     if (b >= m) return n * (n - 1) / 2 * (b / m) + ←
7         solve(n, a, b % m, m);
8     return solve((a + b * n) / m, (a + b * n) % m, m, ←
9         b);
10 }

```

```

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 ←
78         + 1LL * o[i] * t[i]) % MOD;
79     return res;
80 }

```

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]) % ←
9                 MOD;
10        }
11        b.insert(b.begin(), 0);
12        if (delta != 0) {
13            vector<int> t(max(la.size(), b.size()));
14            for (int i = 0; i < (int)t.size(); i++) {
15                if (i < (int)la.size()) t[i] = (t[i] + la[i] ←
16                    ) % MOD;
17                if (i < (int)b.size()) t[i] = (t[i] - 1LL * ←
18                    delta * b[i] % MOD + MOD) % MOD;
19            }
20            if (2 * l <= r - 1) {
21                b = la;
22                int od = inv(delta);
23                for (int &x : b) x = 1LL * x * od % MOD;
24                l = r - 1;
25            }
26            la = t;
27        }
28        assert((int)la.size() == l + 1);
29        assert(l * 2 + 30 < (int)s.size());
30        reverse(la.begin(), la.end());
31        return la;
32    }
33
34    vector<int> mul(vector<int> a, vector<int> b) {
35        vector<int> c(a.size() + b.size() - 1);
36        for (int i = 0; i < (int)a.size(); i++) {
37            for (int j = 0; j < (int)b.size(); j++) {
38                c[i + j] = (c[i + j] + 1LL * a[i] * b[j]) % ←
39                    MOD;
40            }
41        }
42        vector<int> res(c.size());
43        for (int i = 0; i < (int)res.size(); i++) res[i] = ←
44            c[i] % MOD;
45        return res;
46    }
47
48    vector<int> mod(vector<int> a, vector<int> b) {
49        if (a.size() < b.size()) a.resize(b.size() - 1);
50
51        int o = inv(b.back());
52        for (int i = (int)a.size() - 1; i >= (int)b.size() ←
53            - 1; i--) {
54            if (a[i] == 0) continue;
55            int coef = 1LL * o * (MOD - a[i]) % MOD;
56            for (int j = 0; j < (int)b.size(); j++) {
57                a[i - (int)b.size() + 1 + j] = (a[i - (int)b ←
58                    .size() + 1 + j] + 1LL * coef * b[j]) % MOD;
59            }
60        }
61        while (a.size() >= b.size()) {
62            assert(a.back() == 0);
63            a.pop_back();
64        }
65        return a;
66    }
67
68    vector<int> bin(int n, vector<int> p) {

```

final/numeric/integrate.cpp

```

1 function<dbl>(dbl, dbl, function<dbl>(dbl)>> f = [&](←
2     dbl L, dbl R, function<dbl>(dbl)> g) {
3     const int ITTERS = 1000000;
4     dbl ans = 0;
5     dbl step = (R - L) * 1.0 / ITTERS;
6     for (int it = 0; it < ITTERS; it++) {
7         double x1 = L + step * it;
8         double xr = L + step * (it + 1);
9         dbl x1 = (x1 + xr) / 2;
10        dbl x0 = x1 - (x1 - x1) * sqrt(3.0 / 5);
11        dbl x2 = x1 + (x1 - x1) * sqrt(3.0 / 5);
12        ans += (5 * g(x0) + 8 * g(x1) + 5 * g(x2)) / 18 ←
13            * step;
14    }
15    return ans;
16 }

```


final/geom/commonTangents.cpp

```

1  vector<Line> commonTangents(pt A, dbl rA, pt B, dbl rB) {
2
3      vector<Line> res;
4      pt C = B - A;
5      dbl z = C.len2();
6      for (int i = -1; i <= 1; i += 2) {
7          for (int j = -1; j <= 1; j += 2) {
8              dbl r = rB * j - rA * i;
9              dbl d = z - r * r;
10             if (ls(d, 0)) continue;
11             d = sqrt(max(0.01, d));
12             pt magic = pt(r, d) / z;
13             pt v(magic % C, magic * C);
14             dbl CC = (rA * i - v % A) / v.len2();
15             pt O = v * -CC;
16             res.pb(Line(O, 0 + v.rotate()));
17         }
18     }
19     return res;
20 }
21
22 // HOW TO USE ::
23 // --- *D*-----*F*
24 // --- *...* - - - *...*
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; i++) {
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))
49                 return 0;
50             return -1;
51         }
52         if (less(v * u, 0))
53             return -1;
54     }
55
56     cur = 0;
57     vector<Line> st(n * 2);
58     for (int tt = 0; tt < 2; tt++) {
59         for (int i = 0; i < n; i++) {
60             for (; cur >= 2; cur--) {
61                 pt G = st[cur - 1] * l[i];
62                 if (!lessE(st[cur - 2].v * (G - st[cur - 2].v), 0))
63                     break;
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  pair<pt, dbl> minDisc(vector<pt> p) {
2      int n = p.size();
3      pt O = pt(0, 0);
4      dbl R = 0;
5      random_shuffle(all(p));
6      for (int i = 0; i < n; i++) {
7          if (ls(R, (O - p[i]).len())) {
8              O = p[i];
9              R = 0;
10             for (int j = 0; j < i; j++) {
11                 if (ls(R, (O - p[j]).len())) {
12                     O = (p[i] + p[j]) / 2;
13                     R = (p[i] - p[j]).len() / 2;
14                     for (int k = 0; k < j; k++) {
15                         if (ls(R, (O - p[k]).len())) {
16                             Line l1((p[i] + p[j]) / 2, (p[i] + p[j] -
17                                 p[k]).rotate());
18                             Line l2((p[k] + p[j]) / 2, (p[k] + p[j] -
19                                 p[i]).rotate());
20                             O = l1 * l2;
21                             R = (p[i] - O).len();
22                         }
23                     }
24                 }
25             }
26         }
27         return {O, R};
28     }

```

final/geom/convexHull3D-N2.cpp

```

1
2 struct Plane {
3     pt 0, 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().0) % 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].0) % 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↵
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 0 = A - m * t;

```

final/geom/polygonArcCut.cpp

```

1
2 struct Meta {
3     int type; // 0 - seg, 1 - circle
4     pt 0;
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, ↵
11     Line l) {
12     vector<pair<pt, Meta>> res;
13     int n = p.size();
14     for (int i = 0; i < n; i++) {
15         pt A = p[i].F;
16         pt B = p[(i + 1) % n].F;
17         if (le(0, l.v * (A - l.0))) {
18             if (eq(0, l.v * (A - l.0))) && p[i].S.type == 1↵
19                 && ls(0, l.v % (p[i].S.0 - A)))
20                 res.pb({A, SEG});
21         }
22     }
23 }

```

```

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

```

final/geom/polygonTangent.cpp

```

1 pt tangent(vector<pt>& p, pt 0, 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] - 0) * (p[best] - 0) * cof > 0)
11                best = id;
12        }
13        pos = best;
14    }
15    return p[pos];
16 }

```

final/geom/checkPlaneInt.cpp

```

1
2 bool checkPoint(vector<Line> l, pt& ret) {
3     random_shuffle(all(l));
4     pt A = l[0].0;
5     for (int i = 1; i < sz(l); i++) {
6         if (!le(0, l[i].v * (A - l[i].0))) {
7             dbl mn = -INF;
8             dbl mx = INF;
9             for (int j = 0; j < i; j++) {
10                if (eq(l[j].v * l[i].v, 0)) {
11                    if (l[j].v % l[i].v < 0 && (l[j].0 - l[i].↵
12                        0) % l[i].v.rotate() <= 0) {
13                        return false;
14                    }
15                }
16                else {
17                    pt u = l[j].v.rotate();
18                    dbl proj = (l[j].0 - l[i].0) % u / (l[i].v↵
19                        % u);
20                    if (l[i].v * l[j].v > 0) {
21                        mx = min(mx, proj);
22                    }
23                    else {
24                        mn = max(mn, proj);
25                    }
26                }
27            }
28            if (mn <= mx) {
29                A = l[i].0 + l[i].v * mn;
30            }
31            else {
32                return false;
33            }
34        }
35    }
36    ret = A;
37    return true;
38 }

```

final/geom/furthestPoints.cpp

```

36 }
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

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[u][c]) {
25             res = 1;
26             to[u][c] = sz;
27             len[sz] = len[u] + 2;
28             suf[sz] = to[u][c];
29             sz++;
30         }
31         last = to[u][c];
32         return res;
33     }
34
35     void init() {
36         to[blank][0] = to[blank][1] = even;
37         len[blank] = suf[blank] = INF;
38         len[even] = 0, suf[even] = odd;
39         len[odd] = -1, suf[odd] = blank;
40         last = even;
41         sz = 4;
42     }
43 }

```

final/strings/sufAutomaton.cpp

```

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

```

```

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

```

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 < 0 ? 0 : p[i] - len);
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 }

```

final/strings/duval.cpp

```

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

```

final/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(
13            y, x, n);
14    }
15    if (pivot == -1 && (size * 2 >= n || xx == -1)) ←
16        pivot = x;
17    return size;
18 }
19
20 void build( int x, int xx, int dep, int size ) {
21     assert(dep < K);
22     pivot = -1;
23     get_pivot(x, -1, size);
24     x = pivot;
25     level[x] = dep, parent[x] = xx;
26     for (int y : v[x]) if (level[y] == -1)
27     {
28         build(y, x, dep + 1, size / 2);
29     }
30 }

```

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]) & ←
40             (1 << i)) u = p[u][i];
41         if (u == v) return u;
42         for (int i = K - 1; i >= 0; i--) {
43             if (p[u][i] != p[v][i]) {
44                 u = p[u][i];
45                 v = p[v][i];
46             }
47         }
48         return p[u][0];
49     }
50 }

```

```

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

```

final/graphs/generalMatching.cpp

```

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

```

```

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

```

final/graphs/heavyLight.cpp

```

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

```

```

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

```

final/graphs/hungary.cpp

```

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

```

```

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

```

final/graphs/minCostNegCycle.cpp

```

1 struct Edge {
2     int from, to, cap, flow;
3     double cost;
4 };
5
6 struct Graph {
7     int n;
8     vector<Edge> edges;
9     vector<vector<int>> & e;
10
11     Graph(int _n) {
12         n = _n;
13         e.resize(n);
14     }
15
16     void addEdge(int from, int to, int cap, double ←
17         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←
37                         .cap) {
38                         d[cur.to] = d[v] + 1;
39                         pr[cur.to] = e[v][i];
40                         q.push(cur.to);
41                     }
42                 }
43             }
44             if (d[n - 1] == INF) break;
45             int v = n - 1;
46             while (v) {
47                 edges[pr[v]].flow++;
48                 edges[pr[v] ^ 1].flow--;
49                 v = edges[pr[v]].from;
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
59             vector<vector<double>> d(iters + 1, vector<←
60                 double>(n, INF));
61             vector<vector<int>> p(iters + 1, vector<int>(n, ←
62                 -1));
63             d[0].assign(n, 0);
64             for (int it = 0; it < iters; it++) {
65                 d[it + 1] = d[it];
66                 vector<int> nchanged(n, 0);
67                 for (int v : changed) {
68                     for (int id : e[v]) {
69                         Edge cur = edges[id];
70                         if (d[it + 1][cur.to] > d[it][v] + cur.←
71                             cost && cur.flow < cur.cap) {

```

```

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

```

final/graphs/retro.cpp

```

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

```

```

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

```

final/graphs/smith.cpp

```

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

```

```

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

```

final/graphs/mincut.cpp

```

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

```

final/graphs/twoChineseFast.cpp

```

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

```

```

void add(ll a) { x += a; xadd += a; }
void push() {
    if (l != null) l->add(xadd);
    if (r != null) r->add(xadd);
    xadd = 0;
}
};
Heap *Heap::null = new Heap("wqeqw");
Heap* merge(Heap *l, Heap *r) {
    if (l == Heap::null) return r;
    if (r == Heap::null) return l;
    l->push(); r->push();
    if (l->x > r->x)
        swap(l, r);
    l->r = merge(l->r, r);
    if (l->l->h < l->r->h)
        swap(l->l, l->r);
    l->h = l->r->h + 1;
    return l;
}
Heap *pop(Heap *h) {
    h->push();
    return merge(h->l, h->r);
}
const int N = 666666;
struct DSU {
    int p[N];
    void init(int nn) { iota(p, p + nn, 0); }
    int get(int x) { return p[x] == x ? x : p[x] = get(p[x]); }
    void merge(int x, int y) { p[get(y)] = get(x); }
} dsu;
Heap *eb[N];
int n;
/* ANS */ struct Edge {
/* ANS */ int x, y;
/* ANS */ ll c;
/* ANS */ };
/* ANS */ vector<Edge> edges;
/* ANS */ int answer[N];
void init(int nn) {
    n = nn;
    dsu.init(n);
    fill(eb, eb + n, Heap::null);
    edges.clear();
}
void addEdge(int x, int y, ll c) {
    Heap *h = new Heap(c, x);
    /* ANS */ h->ei = sz(edges);
    /* ANS */ edges.push_back({x, y, c});
    eb[y] = merge(eb[y], h);
}
ll solve(int root = 0) {
    ll ans = 0;
    static int done[N], pv[N];
    memset(done, 0, sizeof(int) * n);
    done[root] = 1;
    int tt = 1;
    /* ANS */ int cnum = 0;
    /* ANS */ static vector<ipair> eout[N];
    /* ANS */ for (int i = 0; i < n; ++i) eout[i].clear();
    for (int i = 0; i < n; ++i) {
        int v = dsu.get(i);
        if (done[v])
            continue;
        ++tt;
        while (true) {
            done[v] = tt;
            int nv = -1;
            while (eb[v] != Heap::null) {
                nv = dsu.get(eb[v]->ver);
                if (nv == v) {
                    eb[v] = pop(eb[v]);
                    continue;
                }
                break;
            }
            if (nv == -1)
                return LINF;
            ans += eb[v]->x;
            eb[v]->add(-eb[v]->x);
            /* ANS */ int ei = eb[v]->ei;
            /* ANS */ eout[edges[ei].x].push_back({++cnum, ei});
            if (!done[nv]) {
                pv[v] = nv;
                v = nv;
                continue;
            }
            if (done[nv] != tt)
                break;
            int v1 = nv;

```



```

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         continue;
117     /* ANS */ answer[nv] = ei;
118     /* ANS */ es.insert(all(eout[nv]));
119 }
120 /* ANS */ answer[root] = -1;
121 return ans;
122 }
123
124 /* Usage: twoc::init(vertex_count);
125 * twoc::addEdge(v1, v2, cost);
126 * twoc::solve(root); - returns cost or LINF
127 * twoc::answer contains index of ingoing edge for<
128 each vertex
129 */

```

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 int main(){
128     freopen("linkcut.in", "r", stdin);
129     freopen("linkcut.out", "w", stdout);
130
131     int n, m;
132     scanf("%d %d", &n, &m);
133
134     for (int i = 0; i < n; i++)
135         v2n[i] = &mem[i];
136
137     for (int i = 0; i < m; i++){
138         int a, b;
139         if (scanf(" link %d %d", &a, &b) == 2)
140             link(a-1, b-1);
141         else if (scanf(" cut %d %d", &a, &b) == 2)
142             cut(a-1, b-1);
143         else if (scanf(" get %d %d", &a, &b) == 2)
144             printf("%d\n", get(a-1, b-1));
145         else
146             assert(false);
147     }

```

```

148 }
149 return 0;
150 }

```

final/graphs/chordaltree.cpp

```

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

```

final/graphs/minimization.cpp

```

1 namespace mimimi /* ^ ^ */ {
2     const int N = 100555;
3     const int S = 3;
4     int e[N][S];
5     int label[N];
6     vector<int> eb[N][S];
7     int ans[N];
8     void solve(int n) {
9         for (int i = 0; i < n; ++i)
10             for (int j = 0; j < S; ++j)
11                 eb[i][j].clear();
12         for (int i = 0; i < n; ++i)
13             for (int j = 0; j < S; ++j)
14                 eb[e[i][j]][j].push_back(i);
15         vector<unordered_set<int>> classes(*max_element(
16             label, label + n) + 1);
17         for (int i = 0; i < n; ++i)
18             classes[label[i]].insert(i);
19         for (int i = 0; i < sz(classes); ++i)
20             if (classes[i].empty()) {
21                 classes[i].swap(classes.back());
22                 classes.pop_back();
23                 --i;
24             }
25     }

```

```

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

```

final/graphs/matroidIntersection.cpp

```

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

```

```

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

```

```

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

```

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

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

$$y' = f(x, y) \quad y_{n+1} = y_n + (k_1 + 2 * k_2 + 2 * k_3 + k_4) * h / 6$$

$$k_1 = f(x_n, y_n) \quad k_2 = f(x_n + h/2, y_n + h/2 * k_1) \quad k_3 = f(x_n + h/2, y_n + h/2 * k_2) \quad k_4 = f(x_n + h, y_n + h * k_3)$$

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) $3 \leq p$, $1 \leq a < p$, найти $x^2 = a$

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

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

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

Лемма Бернсайда:

Группа G действует на множество X Тогда число классов эквивалентности $= (\sum |f(g)| \text{ for } g \text{ in } G) / |G|$ 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 $>= p[k]$
 $dp(n, 1) = n$
 $dp(n, j) = dp(n, j+1) + dp(n / p[j], j)$

$j)$, т. е. $dp(n, j+1) = dp(n, j) - dp(n / p[j], j)$

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

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

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

—

$$\sum_{k=1..n} k^2 = n(n+1)(2n+1)/6$$

$$\sum_{k=1..n} k^3 = n^2(n+1)^2/4$$

Чиселки:

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

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

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

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

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

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

Table of Integrals*

Basic Forms

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

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

$$\int u dv = uv - \int v du \quad (3)$$

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

Integrals of Rational Functions

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

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

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

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

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

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

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

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

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

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

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

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

Integrals with Roots

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\int \sqrt{ax^2+bx+cdx} = \frac{b+2ax}{4a} \sqrt{ax^2+bx+c} + \frac{4ac-b^2}{8a^{3/2}} \ln |2ax+b+2\sqrt{a(ax^2+bx+c)}| \quad (37)$$

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

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

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

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

Integrals with Logarithms

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

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

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

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

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

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

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

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

Integrals with Exponentials

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

$$\int \sqrt{x} e^{ax} dx = \frac{1}{a} \sqrt{x} e^{ax} + \frac{i\sqrt{\pi}}{2a^{3/2}} \operatorname{erf}(i\sqrt{ax}),$$

where $\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$ (51)

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

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

$$\int x^2 e^x dx = (x^2 - 2x + 2) e^x \quad (54)$$

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

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

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

$$\int x^n e^{ax} dx = \frac{(-1)^n}{a^{n+1}} \Gamma[1+n, -ax],$$

where $\Gamma(a, x) = \int_x^\infty t^{a-1} e^{-t} dt$ (58)

$$\int e^{ax^2} dx = -\frac{i\sqrt{\pi}}{2\sqrt{a}} \operatorname{erf}(i\sqrt{a}x) \quad (59)$$

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

$$\int x e^{-ax^2} dx = -\frac{1}{2a} e^{-ax^2} \quad (61)$$

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

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

Integrals with Trigonometric Functions

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\int \sec x \tan x dx = \sec x \quad (85)$$

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

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

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

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

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

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

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

Products of Trigonometric Functions and Monomials

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

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

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

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

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

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

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

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

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

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

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

Products of Trigonometric Functions and Exponentials

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

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

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

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

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

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

Integrals of Hyperbolic Functions

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

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

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

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

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

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

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

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

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

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

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

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

