C++ スニペット

xryuseix

2021 年 11 月 2 日

目次

ソースコード 1　AffineMap

```
1
2  template <class T>
3  class AffineMap {
4  public:
5    typedef vector<vector<T>> Matrix;
6    Matrix matrix; // 単位行列から変換する累積行列積
7    vector<Matrix> affine_log; // アフィン変換のログ
8    AffineMap() {
9      matrix = vector<vector<T>>{{1, 0, 0}, {0, 1, 0}, {0, 0, 1}};
10     affine_log.push_back(matrix);
11   }
12
13   // theta 度,反時計回りで回転
14   Matrix AffineMatrix_Rotation(T theta) {
15     double theta_rad = theta * M_PI / 180;
16     return Matrix{
17       {static_cast<T>(cos(theta_rad)), static_cast<T>(-sin(theta_rad)),
18       0},
19       {static_cast<T>(sin(theta_rad)), static_cast<T>(cos(theta_rad)), 0},
20       {0, 0, 1}};
21   };
22   // X==p で反転
23   Matrix AffineMatrix_ReverseX(T p) {
24     return Matrix{{-1, 0, 2 * p}, {0, 1, 0}, {0, 0, 1}};
25   };
26   // Y==p で反転
27   Matrix AffineMatrix_ReverseY(T p) {
28     return Matrix{{1, 0, 0}, {0, -1, 2 * p}, {0, 0, 1}};
29   };
30   // +x, +y へ平行移動
31   Matrix AffineMatrix_Translation(T x, T y) {
32     return Matrix{{1, 0, x}, {0, 1, y}, {0, 0, 1}};
33   };
34
35   // 回転
36   Matrix rotation(T theta) {
37     matrix = matrix_product_33_33(AffineMatrix_Rotation(theta), matrix);
38     affine_log.push_back(matrix);
39     return matrix;
40   }
41   // X==p で反転
```

```
42    Matrix reverseX(T p) {
43      matrix = matrix_product_33_33(AffineMatrix_ReverseX(p), matrix);
44      affine_log.push_back(matrix);
45      return matrix;
46    }
47    // Y==p で反転
48    Matrix reverseY(T p) {
49      matrix = matrix_product_33_33(AffineMatrix_ReverseY(p), matrix);
50      affine_log.push_back(matrix);
51      return matrix;
52    }
53    // +x, +y へ平行移動
54    Matrix translation(T x, T y, Matrix coord = Matrix(0)) {
55      if (coord == Matrix(0)) {
56        coord = matrix;
57      }
58      matrix = matrix_product_33_33(AffineMatrix_Translation(x, y), matrix);
59      affine_log.push_back(matrix);
60      return matrix;
61    }
62
63    // 3*3 * 3*1 の行列積
64    Matrix matrix_product_33_31(Matrix a, Matrix b) {
65      // a が 3*3, b が 3*1 の行列か確認
66      assert((int)a.size() == 3 && (int)a[0].size() == 3 &&
67        (int)a[1].size() == 3 && (int)a[2].size() == 3);
68      assert((int)b.size() == 3 && (int)b[0].size() == 1 &&
69        (int)b[1].size() == 1 && (int)b[2].size() == 1);
70      return Matrix{
71        {a[0][0] * b[0][0] + a[0][1] * b[1][0] + a[0][2] * b[2][0]},
72        {a[1][0] * b[0][0] + a[1][1] * b[1][0] + a[1][2] * b[2][0]},
73        {a[2][0] * b[0][0] + a[2][1] * b[1][0] + a[2][2] * b[2][0]}};
74    }
75    Matrix matrix_product_33_33(Matrix a, Matrix b) {
76      // a が 3*3, b が 3*1 の行列か確認
77      assert((int)a.size() == 3 && (int)a[0].size() == 3 &&
78        (int)a[1].size() == 3 && (int)a[2].size() == 3);
79      assert((int)b.size() == 3 && (int)b[0].size() == 3 &&
80        (int)b[1].size() == 3 && (int)b[2].size() == 3);
81      Matrix matrix_tmp(3, vector<T>(3, 0));
82      rep(i, 3) {
83        rep(j, 3) {
84          rep(k, 3) { matrix_tmp[i][j] += a[i][k] * b[k][j]; }
85        }
86      }
87      return matrix_tmp;
```

```
88    }
89  };
```

ソースコード 2　BoyerMoore

```
1
2   class BoyerMoore {
3     public:
4     string text;
5     string pattern;
6     int n;
7     int m;
8     map<char, int> lambda;
9     BoyerMoore(string text_, string pattern_) :
10      text(text_), pattern(pattern_), n(text_.size()), m(pattern_.size()) {
11      compute_lambda();
12    };
13    void compute_lambda(void) {
14      for(int j = 1; j <= m; j++) {
15        lambda[pattern.at(j - 1)] = j;
16      }
17    };
18    int get_lambda(const char& c) {
19      if (lambda.find(c) != lambda.end()) {
20        return lambda[c];
21      } else {
22        return 0;
23      }
24    };
25    bool match(void) {
26      int s = 0;
27      while(s <= n - m) {
28        int j = m;
29        while(j > 0 && pattern.at(j - 1) == text.at(s + j - 1)) {
30          j--;
31        }
32        if(j == 0) {
33          return true;//ここを消すと s が文字列の位置を示す
34          s++;
35        } else {
36          s += std::max(1, j - get_lambda(text.at(s + j - 1)));
37        }
38      }
39      return false;
```

```
  };
};
```

ソースコード 3　BridgeArticulation

```
class BridgeArticulation {
  int N, num = 0;
  vvi G;
  vi pre, low;
  vb isPassed;

  int culcLow(const int v, const int bef) {
    int nowLow = num;
    low[v] = pre[v] = nowLow;
    for(auto ne : G[v]) {
      if(ne == bef) continue;
      if(ne == 0) {
        low[0] = -1;
      }
      if(pre[ne] == -1) {
        num++;
        culcLow(ne, v);
      }
      chmin(nowLow, low[ne]);
    }
    return low[v] = nowLow;
  }

  void traceGraph(const int v, const int bef) {
    bool is_articulation = false;
    for(auto ne : G[v]) {
      if(ne == bef) continue;
      if(!isPassed[ne]) {
        if(low[ne] >= pre[v] && (bef != -1 || G[v].size() >= 2)) {
          is_articulation = true;
        }
        if(low[ne] == pre[ne]) {
          bridges.emplace_back(min(v, ne), max(v, ne));
        }
        isPassed[ne] = true;
        traceGraph(ne, v);
      }
    }
```

```
40      if(is_articulation) {
41        articulation.push_back(v);
42      }
43    }
44
45  public:
46    vpii bridges; // 橋
47    vi articulation; // 関節点
48
49    BridgeArticulation(const int _n, const vvi _G) : N(_n), G(_G) {
50      pre = vi(N, -1);
51      low = vi(N, INF);
52      isPassed = vb(N, false);
53      isPassed[0] = true;
54    }
55
56    void findBridges() {
57      culcLow(0, -1);
58      traceGraph(0, -1);
59      Sort(bridges);
60    }
61
62    void show() {
63      for(auto p : bridges) {
64        cout << p.fi << \"␣\" << p.se << endl;
65      }
66    }
67  };
```

ソースコード 4　CoordinateCompression

```
1
2   class Compress {
3   public:
4     int before_W, before_H, N;
5     vi before_X1, before_X2, before_Y1, before_Y2;
6     int after_W, after_H;
7     vi after_X1, after_X2, after_Y1, after_Y2;
8
9     // (x1,y1) -> (x2, y2) の直線上のマスが塗られているとする
10    // 点の場合は (x1,y1) == (x2, y2) とする
11    // 四角形の場合は直線の集合とする
12    // (バグってるよこれ)
13    Compress(int max_h, int max_w, int n, vi x1, vi x2, vi y1, vi y2) {
```

```
14      before_H = max_h;
15      before_W = max_w;
16      N = n;
17      before_X1 = x1;
18      before_X2 = x2;
19      before_Y1 = y1;
20      before_Y2 = y2;
21      after_X1 = vi(max_w);
22      after_X2 = vi(max_w);
23      after_Y1 = vi(max_h);
24      after_Y2 = vi(max_h);
25    }
26
27    void compress(void) {
28      after_W = exec_compress(before_X1, before_X2, after_X1, after_X2, before_W, \"
            width\");
29      after_H = exec_compress(before_Y1, before_Y2, after_Y1, after_Y2, before_H, \"
            height\");
30    }
31
32    void before_show(void) {
33      vvc v(before_H, vc(before_W, '_'));
34      cout << \"H␣=␣\" << before_H << \"␣W␣=␣\" << before_W << endl;
35      for(int i = 0; i < N; i++) {
36        for(int y = before_Y1[i]; y <= before_Y2[i]; y++) {
37          for(int x = before_X1[i]; x <= before_X2[i]; x++) {
38            v[y][x] = '#';
39          }
40        }
41      }
42      rep(i, before_H){
43        rep(j, before_W){
44          cout << v[i][j];
45        }
46        cout<<endl;
47      }
48      cout << endl;
49    }
50
51    void after_show(void) {
52      vvc v(after_H, vc(after_W, '_'));
53      cout << \"H␣=␣\" << after_H << \"␣W␣=␣\" << after_W << endl;
54      for(int i = 0; i < N; i++) {
55        for(int y = after_Y1[i]; y <= after_Y2[i]; y++) {
56          for(int x = after_X1[i]; x <= after_X2[i]; x++) {
57            v[y][x] = '#';
```

```
58          }
59        }
60      }
61      rep(i, after_H){
62        rep(j, after_W){
63          cout << v[i][j];
64        }
65        cout<<endl;
66      }
67      cout << endl;
68    }
69
70  private:
71    int exec_compress(vi &z1, vi &z2, vi &aft_z1, vi &aft_z2, int max_len, string mode)
          {
72      vector<int> zs;
73      for(int i = 0; i < N; i++) {
74        if(z1[i] > z2[i]) swap(z1[i], z2[i]);
75
76        zs.push_back(z1[i]);
77        zs.push_back(z2[i]);
78
79        if(mode == \"width\") {
80          if(z2[i] + 1 <= max_len) zs.push_back(z2[i] + 1);
81        } else if(mode == \"height\") {
82          if(0 < z1[i] - 1) zs.push_back(z1[i] - 1);
83        }
84      }
85      zs.push_back(1);
86      zs.push_back(max_len);
87
88      sort(zs.begin(), zs.end());
89      zs.erase(unique(zs.begin(), zs.end()), zs.end());
90
91      for(int i = 0; i < N; i++) {
92        aft_z1[i] = find(zs.begin(), zs.end(), z1[i]) - zs.begin() + 1;
93        aft_z2[i] = find(zs.begin(), zs.end(), z2[i]) - zs.begin() + 1;
94      }
95      return zs.size();
96    }
97  };
```

ソースコード 5　Counting

```
1
2  // modint 呼ぶ！！！
3
4  class Counting {
5    struct pre_calculation {
6      bool factorial;
7      bool partition;
8      bool combination;
9      bool bell;
10   };
11   pre_calculation pre_calc;
12
13   vector<mint> Factorial;
14   void factorial_precalc(const int MAX = 301010) {
15     Factorial = vector<mint>(MAX, 1);
16     for (int i = 1; i < MAX; i++) {
17       Factorial[i] = Factorial[i - 1] * i;
18     }
19   }
20
21   vector<mint> FactorialInv;
22   void nCk_precalc(const int MAX = 301010) {
23     if (!pre_calc.factorial) {
24       factorial_precalc();
25     }
26     FactorialInv = vector<mint>(MAX, 1);
27     for (int i = 1; i < MAX; i++) {
28       FactorialInv[i] = mint(1).pow(Factorial[i], MOD - 2);
29     }
30   }
31
32   vector<mint> Bell;
33   void bell_precalc(const int MAX = 2020) {
34     if (!pre_calc.factorial) {
35       factorial_precalc();
36     }
37     Bell = vector<mint>(MAX, 1);
38     for (int j = 1; j < MAX; j++) {
39       Bell[j] = Bell[j - 1] + mint(0).pow(-1, j) / Factorial[j];
40     }
41   }
42
43   vector<vector<mint>> Partition;
44   void partition_precalc(const int MAX = 2020) {
45     Partition = vector<vector<mint>>(MAX, vector<mint>(MAX, 0));
```

```
    for (int k = 0; k < MAX; k++) {
      Partition[0][k] = 1;
    }
    for (int n = 1; n < MAX; n++) {
      for (int k = 1; k < MAX; k++) {
        Partition[n][k] = Partition[n][k - 1];
        if (n - k >= 0) {
          Partition[n][k] += Partition[n - k][k];
        }
      }
    }
  }

public:
  Counting(void) {
    pre_calc.factorial = false;
    pre_calc.partition = false;
    pre_calc.combination = false;
    pre_calc.bell = false;
  };

  // 階乗 n! を計算
  mint factorial(const mint n) {
    if (!pre_calc.factorial) {
      factorial_precalc();
      pre_calc.factorial = true;
    }
    return Factorial[n.x];
  }

  // 二項係数 nCk を計算
  mint nCk(const mint n, const mint k) {
    if (n < k) {
      return 0;
    }
    if (!pre_calc.combination) {
      nCk_precalc();
      pre_calc.combination = true;
    }
    mint nl = Factorial[n.x];// n!
    mint nkl = FactorialInv[n.x - k.x]; // (n-k)!
    mint kl = FactorialInv[k.x]; // k!
    mint nkk = (nkl.x * kl.x);
    return nl * nkk;
  }

```

```
92  // 順列 nPk を計算
93  mint nPk(mint n, mint k) {
94    if (n < k) {
95      return 0;
96    }
97    if (!pre_calc.factorial) {
98      factorial_precalc();
99      pre_calc.factorial = true;
100   }
101   return Factorial[n.x] / Factorial[n.x - k.x];
102 }
103
104 // 包除原理 \\sum_{i=0}^k (-1)^{k-i} kCi i^n
105 mint inclusion_exclusion(const mint n, const mint k) {
106   if (!pre_calc.combination) {
107     nCk_precalc();
108     pre_calc.combination = true;
109   }
110   mint ans = 0;
111   for (int i = 0; i <= k.x; i++) {
112     ans += mint(0).pow(-1, k - i) * nCk(k, i) * mint(0).pow(i, n);
113   }
114   return ans;
115 }
116
117 // スターリング数 S(n, k) を計算
118 mint stirling(const mint n, const mint k) {
119   mint res = inclusion_exclusion(n, k);
120   res /= Factorial[k.x];
121   return res;
122 }
123
124 // ベル数 B(n, k) を計算
125 mint bell(const mint n, const mint k) {
126   if (!pre_calc.bell) {
127     bell_precalc();
128     pre_calc.bell = true;
129   }
130   mint ans = 0;
131   for (int i = 0; i <= k.x; i++) {
132     ans += mint(0).pow(i, n) / Factorial[i] * Bell[k.x - i];
133   }
134   return ans;
135 }
136
137 // 分割数 P(n, k) を計算
```

```
  // n - k < 0 場合は別途,場合分けを行う
  mint partition(const mint n, const mint k) {
    if (n.x < 0) {
      return 0;
    }
    if (!pre_calc.partition) {
      partition_precalc();
      pre_calc.partition = true;
    }
    return Partition[n.x][k.x];
  }
};
```

ソースコード 6　LazySegmentTree

```
1
2   template <class X, class MONOID, class M, class MANAGE>
3   class LazySegmentTree {
4   public:
5     // セグメント木の葉の要素数
6     int n;
7
8     // セグメント木
9     vector<X> tree;
10    vector<M> lazy;
11
12    // モノイド
13    MONOID mono;
14    MANAGE manage;
15
16    LazySegmentTree(const vector<X>& v) : n(1 << (int)ceil(log2(v.size()))) {
17      tree = vector<X>(n << 1, mono.unit);
18      lazy = vector<M>(n << 1, manage.unit);
19      for (int i = 0; i < v.size(); ++i) {
20        tree[i + n] = v[i];
21      }
22      for (int i = n - 1; i > 0; --i) {
23        tree[i] = mono.calc(tree[i << 1 | 0], tree[i << 1 | 1]);
24      }
25    }
26
27    LazySegmentTree(const int _n) : n(1 << (int)ceil(log2(_n))) {
28      tree = vector<X>(n << 1, mono.unit);
29      lazy = vector<M>(n << 1, manage.unit);
```

```
30    }
31
32    // k 番目のノードの遅延評価を行う
33    void eval(const int k, int l, int r) {
34      // 遅延評価配列が空でない時,値を伝播する
35      if (lazy[k] == manage.unit) {
36        return;
37      }
38      if (k < n) {
39        lazy[k << 1 | 0] = manage.fm(lazy[k << 1 | 0], lazy[k]);
40        lazy[k << 1 | 1] = manage.fm(lazy[k << 1 | 1], lazy[k]);
41      }
42      tree[k] = manage.fa(tree[k], lazy[k], r - l);
43      // 伝播が終わったので自ノードの遅延配列を空にする
44      lazy[k] = manage.unit;
45    }
46
47    // 区間 [l, r) の値を全て更新する (遅延評価)
48    void updateRange(const int l, const int r, const X x) {
49      updateRange(l, r, x, 1, 0, n);
50    }
51
52    void updateRange(const int a, const int b, const X x, const int k,
53            const int l, const int r) {
54      // k 番目のノードに対して遅延評価を行う
55      eval(k, l, r);
56
57      // 完全に被覆しているならば､遅延配列に値を入れた後に評価
58      if (a <= l && r <= b) {
59        lazy[k] = manage.fm(lazy[k], x);
60        eval(k, l, r);
61      } else if (a < r && l < b) {
62        updateRange(a, b, x, k << 1 | 0, l, (l + r) >> 1);
63        updateRange(a, b, x, k << 1 | 1, (l + r) >> 1, r);
64        tree[k] = mono.calc(tree[k << 1 | 0], tree[k << 1 | 1]);
65      }
66    }
67
68    // 区間 [l, r) の merge 結果を取得する
69    X getRange(const int l, const int r) { return getRange(l, r, 1, 0, n); }
70
71    X getRange(const int a, const int b, const int k, const int l,
72        const int r) {
73      // 関数が呼び出されたら評価！
74      eval(k, l, r);
75      if (r <= a || b <= l) { // 完全に外側の時
```

```
76        return mono.unit;
77      } else if (a <= l && r <= b) { // 完全に内側の時
78        return tree[k];
79      } else { // 一部区間が被る時
80        X vl = getRange(a, b, k << 1 | 0, l, (l + r) >> 1);
81        X vr = getRange(a, b, k << 1 | 1, (l + r) >> 1, r);
82        return mono.calc(vl, vr);
83      }
84    }
85
86    X operator[](const int k) {
87      // st[i] で添字 i の要素の値を返す
88      if (k - n >= 0 || k < 0) {
89        return -INF;
90      }
91      return tree[tree.size() - n + k];
92    }
93
94    void show(void) {
95      this->showTree(tree);
96      this->showLazy(lazy);
97    }
98
99    template <class T>
100   void showTree(vector<T>& tree) {
101     int ret = 2;
102     for (int i = 1; i < 2 * n; ++i) {
103       if (tree[i] == mono.unit) {
104         cout << \"UNIT␣\";
105       } else {
106         cout << tree[i] << \"␣\";
107       }
108       if (i == ret - 1) {
109         cout << endl;
110         ret <<= 1;
111       }
112     }
113     cout << endl;
114   }
115 };
116
117 using X = ll;
118 using M = ll;
119
120 struct Monoid {
121   Monoid(void) {}
```

```
122
123     struct Sum {
124       const X unit = 0;
125       static X calc(X a, X b) { return a + b; }
126     };
127     struct Min {
128       const X unit = INF;
129       static X calc(X a, X b) { return min(a, b); }
130     };
131
132     // モノイド
133     Sum monoid;
134
135     // 単位元
136     X unit = monoid.unit;
137
138     // 演算関数
139     function<X(X, X)> calc = monoid.calc;
140   };
141
142   struct Manage {
143     Manage(void) {}
144
145     struct Add {
146       const M unit = 0;
147       static M fa(X x, M m, size_t size) { return x + m * size; } // RSQ+RAQ
148       static M fa2(X x, M m, size_t size) { return x + m; }// RMQ+RAQ
149       static M fm(M m1, M m2) { return m1 + m2; }
150     };
151     struct Update {
152       const M unit = INF;
153       static M fa(X x, M m, size_t size) { return m * size; } // RSQ+RUQ
154       static M fa2(X x, M m, size_t size) { return m; }// RMQ+RUQ
155       static M fm(M m1, M m2) { return m2; }
156     };
157
158     // 作用構造体
159     Update manage;
160
161     // 単位元
162     M unit = manage.unit;
163
164     // 演算関数
165     function<M(X, M, size_t)> fa = manage.fa;
166     function<M(M, M)> fm = manage.fm;
167   };
```

ソースコード 7　MaximumFlow

```
1
2  class MaximumFlow {
3
4    int v;
5
6    // 辺を表す構造体 (行き先,容量,逆辺のインデックス)
7    struct edge {
8      int to;
9      int cap;
10     int rev;
11   };
12
13   vector<vector<edge>> G; // グラフの隣接リスト表現
14   vector<bool> used; // DFS ですでに調べたかのフラグ
15
16   // 増加パスをDFS で探す(今いる頂点，ゴールの頂点，今の頂点以降のフローの最小値)
17   int dfs(int v, int t, int f) {
18     if (v == t) return f;
19     used[v] = true;
20     for (int i = 0; i < G[v].size(); i++) {
21       // v から行ける&&cap>0の頂点を全てたどる
22       edge& e = G[v][i];
23       if (!used[e.to] && e.cap > 0) {
24         // 次の頂点 (e.to)以降で
               t まで行けるパスを探索し，その時のフローの最小値を d とする
25         int d = dfs(e.to, t, min(f, e.cap));
26         if (d > 0) {
27           e.cap -= d;
28           G[e.to][e.rev].cap += d;
29           return d;
30         }
31       }
32     }
33     return 0;
34   }
35
36 public:
37   MaximumFlow(int _v) : v(_v) {
38     used = vector<bool>(v);
39     G = vector<vector<edge>>(v);
40   }
```

```
41
42    // from から to へ向かう容景 cap の辺をグラフに追加する
43    void add(int from, int to, int cap) {
44      G[from].push_back((edge){to, cap, (int)G[to].size()});
45      G[to].push_back((edge){from, 0, (int)G[from].size() - 1});
46    }
47
48    // s から t への最大流を求める
49    int maxFlow(int s, int t) {
50      int flow = 0;
51      while (true) {
52        used = vector<bool>(v);
53        int f = dfs(s, t, INF);
54        if (f == 0) {
55          return flow;
56        }
57        flow += f;
58      }
59    }
60  };
```

ソースコード 8　MinCostFlow

```
1
2   class MinCostFlow {
3
4     int V; // 頂点数
5
6     // 辺を表す構造体 (行き先,容量,逆辺のインデックス)
7     struct edge {
8       int to;
9       int cap;
10      int cost;
11      int rev;
12    };
13
14    vector<vector<edge>> G; // グラフの隣接リスト表現
15    vector<int> h; // ポテンシャル
16    vector<int> prevV; // 直前の頂点
17    vector<int> prevE; // 直前の辺
18    vector<int> dist; // 最短距離
19    typedef pair<int, int> PI;
20
21  public:
```

```
22    MinCostFlow(int _v) : V(_v) {
23      G = vector<vector<edge>>(V);
24      h = vector<int>(V);
25      prevV = vector<int>(V);
26      prevE = vector<int>(V);
27    }
28
29    // from から to へ向かう容景 cap の辺をグラフに追加する
30    void add(int from, int to, int cap, int cost) {
31      G[from].push_back((edge){to, cap, cost, (int)G[to].size()});
32      G[to].push_back((edge){from, 0, -cost, (int)G[from].size() - 1});
33    }
34
35    // s から t への最大流を求める
36    int minCostFlow(int s, int t, int f) {
37      int res = 0;
38      while (f > 0) {
39        // ダイクストラ法を用いてh を更新する
40        priority_queue<PI, vector<PI>, greater<PI>> que;
41        dist = vector<int>(V, INF);
42        dist[s] = 0;
43        que.push(PI(0, s));
44        while(!que.empty()) {
45          PI p = que.top();
46          que.pop();
47          int v = p.second;
48          if(dist[v] < p.first) continue;
49          for (int i = 0; i < G[v].size(); i++) {
50            edge e = G[v][i];
51            if(e.cap > 0 && dist[e.to] > dist[v] + e.cost + h[v] - h[e.to]) {
52              dist[e.to] = dist[v] + e.cost + h[v] - h[e.to];
53              prevV[e.to] = v;
54              prevE[e.to] = i;
55              que.push(PI(dist[e.to], e.to));
56            }
57          }
58        }
59        if(dist[t] == INF) {
60          // これ以上流せない
61          return -1;
62        }
63        for(int v = 0; v < V; v++) {
64          h[v] += dist[v];
65        }
66        // s-t 問最短路に沿って目一杯流す
67        int d = f;
```

```
68        for(int v = t; v != s; v = prevV[v] ) {
69          d = min(d, G[prevV[v]][prevE[v]].cap);
70        }
71        f -= d;
72        res += d*h[t];
73        for(int v = t; v != s; v = prevV[v]) {
74          edge& e = G[prevV[v]][prevE[v]];
75          e.cap -= d;
76          G[v][e.rev].cap += d;
77        }
78      }
79      return res;
80    }
81  };
```

ソースコード 9　RollingHash

```
1
2   std::mt19937 mt{ std::random_device{}() };
3   std::uniform_int_distribution<int> dist(129, INF);
4   const int BASE = dist(mt);
5
6   class RollingHash {
7   public:
8     string str;
9     vector<ull> powBase, csumHash;
10    const ull ROLMOD = (1LL << 61) - 1;
11    const ull MASK30 = (1LL << 30) - 1;
12    const ull MASK31 = (1LL << 31) - 1;
13    const ull LLMAX = ROLMOD*((1LL << 3) - 1);
14
15    RollingHash(const string s) : str(s) {
16      powBase.resize(s.size() + 1);
17      csumHash.resize(s.size() + 1);
18      powBase[0] = 1;
19      for(int i = 0; i < s.size(); i++) {
20        powBase[i + 1] = calcMod(multiple(powBase[i], BASE));
21      }
22    }
23
24    void rollingHash() {
25      csumHash[0] = 0;
26      for(int i = 0; i < str.size(); ++i) {
27        csumHash[i + 1] = calcMod(multiple(csumHash[i], BASE) + str[i]);
```

```
28      }
29    }
30
31    ull getHash(const int begin, const int length) {
32      return calcMod(csumHash[begin + length] + LLMAX - multiple(csumHash[begin],
            powBase[length]));
33    }
34
35    string substr(const int begin) {
36      return str.substr(begin);
37    }
38
39    string substr(const int begin, const int length) {
40      if(length < 0) {
41        return str.substr(begin, str.size() + length - begin + 1);
42      } else {
43        return str.substr(begin, length);
44      }
45    }
46
47  private:
48    ull calcMod(const ull num) {
49      const ull modNum = (num & ROLMOD) + (num >> 61);
50      return (modNum >= ROLMOD) ? modNum - ROLMOD : modNum;
51    }
52
53    ull multiple(const ull leftNum, const ull rightNum) {
54      ull lu = leftNum >> 31;
55      ull ld = leftNum & MASK31;
56      ull ru = rightNum >> 31;
57      ull rd = rightNum & MASK31;
58      ull middleBit = ld * ru + lu * rd;
59      return ((lu * ru) << 1) + ld * rd + ((middleBit & MASK30) << 31) + (middleBit
            >> 30);
60    }
61
62  };
```

ソースコード 10　SegmentTree

```
1
2   template <typename T>
3   class Sum {
4   public:
```

```
5     // 単位元
6     T unit;
7
8     Sum(void) {
9       // 単位元
10      unit = 0;
11    }
12
13    // 演算関数
14    T calc(T a, T b) { return a + b; }
15  };
16
17  template <typename T>
18  struct Min {
19  public:
20    // 単位元
21    T unit;
22
23    Min(void) {
24      // 単位元
25      unit = INF;
26    }
27
28    // 演算関数
29    T calc(T a, T b) { return min(a, b); }
30  };
31
32  template <typename T, class MONOID>
33  class SegmentTree {
34  public:
35    // セグメント木の葉の要素数
36    int n;
37
38    // セグメント木
39    vector<T> tree;
40
41    // モノイド
42    MONOID mono;
43
44    SegmentTree(vector<T>& v) : n(1 << (int)ceil(log2(v.size()))) {
45      tree = vector<T>(n << 1, mono.unit);
46      for (int i = 0; i < v.size(); ++i) {
47        tree[i + n] = v[i];
48      }
49      for (int i = n - 1; i > 0; --i) {
50        tree[i] = mono.calc(tree[i << 1 | 0], tree[i << 1 | 1]);
```

```
51      }
52    }
53
54    SegmentTree(int _n) : n(1 << (int)ceil(log2(_n))) {
55      tree = vector<T>(n << 1, mono.unit);
56    }
57
58    // k 番目の値(0-indexed)をx に変更
59    void update(int k, T x) {
60      k += n;
61      tree[k] = x;
62      for (k = k >> 1; k > 0; k >>= 1) {
63        tree[k] = mono.calc(tree[k << 1 | 0], tree[k << 1 | 1]);
64      }
65    }
66
67    // k 番目の値(0-indexed)をx を加算
68    void add(int k, T x) {
69      k += n;
70      tree[k] += x;
71      for (k = k >> 1; k > 0; k >>= 1) {
72        tree[k] = mono.calc(tree[k << 1 | 0], tree[k << 1 | 1]);
73      }
74    }
75
76    // [l, r)の最小値 (0-indexed)を求める.
77    T query(int l, int r) {
78      T res = mono.unit;
79      l += n;
80      r += n;
81      while (l < r) {
82        if (l & 1) {
83          res = mono.calc(res, tree[l++]);
84        }
85        if (r & 1) {
86          res = mono.calc(res, tree[--r]);
87        }
88        l >>= 1;
89        r >>= 1;
90      }
91      return res;
92    }
93
94    T operator[](int k) {
95      // st[i]で添字i の要素の値を返す
96      if (k - n >= 0 || k < 0) {
```

```
97        return -INF;
98      }
99      return tree[tree.size() - n + k];
100   }
101
102   void show(int elements = 31) {
103     int ret = 2;
104     for (int i = 1; i < min(2 * n, elements); ++i) {
105       if (tree[i] == mono.unit)
106         cout << \"UNIT␣\";
107       else
108         cout << tree[i] << \"␣\";
109       if (i == ret - 1) {
110         cout << endl;
111         ret <<= 1;
112       }
113     }
114     cout << endl;
115   }
116 };
```

ソースコード 11　StronglyConnectedComponent

```
1
2  class StronglyConnectedComponent {
3  public:
4    int V; // 頂点数
5    int SubGraph; // 強連結成分の数
6    vvi Graph; // グラフの隣接リスト表現
7    vvi revGraph; // 辺の向きを逆にしたグラフ
8    vvi SmallGraph; // 強連結成分分解によって縮めたグラフ
9    vi dfsline; // 帰りがけ順の並び
10   vi compo; // cmp[i]で頂点i の属するグループ
11   vb used;// すでに調べたか
12
13   StronglyConnectedComponent(int v) {
14     V = v;
15     Graph = vvi(v);
16     revGraph = vvi(v);
17     used = vb(v);
18     compo = vi(v);
19   }
20
21   int operator[](int k) {
```

```
22      // scc[i]でi 番目の頂点のグループ番号を返す
23      return compo[k];
24    }
25
26    void add_edge(int from, int to) {
27      Graph[from].push_back(to);
28      revGraph[to].push_back(from);
29    }
30
31    void dfs(int v) {
32      used[v] = true;
33      for(int i = 0; i < Graph[v].size(); i++) {
34        if(!used[Graph[v][i]]) dfs(Graph[v][i]);
35      }
36      dfsline.push_back(v);
37    }
38
39    void revdfs(int v, int k) {
40      used[v] = true;
41      compo[v] = k;
42      for(int i = 0; i < revGraph[v].size(); i++) {
43        if(!used[revGraph[v][i]]) revdfs(revGraph[v][i], k);
44      }
45    }
46
47    int scc(void) {
48      used = vb((int)used.size(), false);
49      dfsline.clear();
50      for(int v = 0; v < V; v++) {
51        if(!used[v]) dfs(v);
52      }
53      used = vb(used.size(), false);
54      SubGraph = 0;
55      for(int i = dfsline.size() - 1; i >= 0; i--) {
56        if(!used[dfsline[i]]) revdfs(dfsline[i], SubGraph++);
57      }
58      for(int i = 0; i < compo.size(); i++) {
59        compo[i] = SubGraph - compo[i] - 1;
60      }
61      return SubGraph;
62    }
63
64    void build(void) {
65      // 縮めたグラフを構築する
66      SmallGraph = vvi(SubGraph);
67      for (int i = 0; i < Graph.size(); i++) {
```

```
68        for(int j = 0; j < Graph[i].size(); j++) {
69          int to = Graph[i][j];
70          int s = compo[i], t = compo[to];
71          if (s != t){
72            SmallGraph[s].push_back(t);
73          }
74        }
75      }
76      for(int i = 0; i < SmallGraph.size(); i++) {
77        // 被った辺を削除
78        SmallGraph[i].erase(unique(SmallGraph[i].begin(), SmallGraph[i].end()),
              SmallGraph[i].end());
79      }
80    }
81
82    void show_set_to_edge(void) {
83      for(int i = 0; i < SmallGraph.size(); i++) {
84        cout << \"集合\" << i << \"から出ている辺␣:␣\";
85        for(int j = 0; j < SmallGraph[i].size(); j++) {
86          cout << SmallGraph[i][j] << '␣';
87        }
88        cout << endl;
89      }
90      cout << endl;
91    }
92
93    void show_group_of_node(void) {
94      for(int i = 0; i < V; i++) {
95        cout << \"頂点\" << i << \"の属するグループ␣:␣\" << compo[i] << endl;
96      }
97      cout << endl;
98    }
99
100   void show_node_in_group(void) {
101     vvi group(SubGraph);
102     for(int i = 0; i < compo.size(); i++) {
103       group[compo[i]].push_back(i);
104     }
105     for(int i = 0; i < SmallGraph.size(); i++) {
106       cout << \"グループ\" << i << \"に属する頂点␣:␣\";
107       for(int j = 0; j < group[i].size(); j++) {
108         cout << group[i][j] << '␣';
109       }
110       cout << endl;
111     }
112     cout << endl;
```

```
113   }
114 };
```

ソースコード 12　Trie

```
1
2   template <int char_size, int base>
3   class Trie {
4   public:
5     struct Node {
6       vector<int> next;// 子の頂点番号を格納。存在しなければ-1
7       vector<int> accept; // 末端がこの頂点になる文字列 (受理状態)のID
8       int c; // 文字,base からの距離
9       int common; // いくつの文字列がこの頂点を共有しているか
10      Node(int c_) : c(c_), common(0) { next = vector<int>(char_size, -1); }
11    };
12    vector<Node> nodes; // trie 木本体
13    Trie() { nodes.push_back(Node(0)); }
14
15    /*
16    単語の検索
17    prefix ... 先頭のword が一致していればいいかどうか
18    */
19    bool search(const string word, const bool prefix = false) {
20      int node_id = 0;
21      for (auto w : word) {
22        int c = w - base;
23        int next_id = nodes[node_id].next[c];
24        // printf(\"%c %d %d\n\"w,c,next_id);
25        if (next_id == -1) {
26          return false;
27        }
28        node_id = next_id;
29      }
30      return (prefix) ? true : nodes[node_id].accept.size();
31    }
32
33    /*
34    単語の挿入
35    word_id ... 何番目に追加する単語か
36    */
37    void insert(const string &word) {
38      int node_id = 0;
39      for (auto w : word) {
```

```
40        int c = w - base;
41        int next_id = nodes[node_id].next[c];
42        if (next_id == -1) { // 次の頂点が存在しなければ追加
43          next_id = nodes.size();
44          nodes.push_back(Node(c));
45          nodes[node_id].next[c] = next_id;
46        }
47        ++nodes[node_id].common;
48        node_id = next_id;
49      }
50      ++nodes[node_id].common;
51      nodes[node_id].accept.push_back(nodes[0].common - 1); // 単語の終端なので、頂点に番
            号を入れておく
52    }
53
54    /*
55    prefix の検索
56    */
57    bool start_with(const string& prefix) { return search(prefix, true); }
58
59    /*
60    挿入した単語数
61    */
62    int count() const { return nodes[0].common; }
63
64    /*
65    頂点数
66    */
67    int size() const { return nodes.size(); }
68  };
```

ソースコード 13　bellmanford

```
1
2   // 頂点from から頂点 to へのコスト cost の辺
3   struct bf_edge {
4     int from;
5     int to;
6     int cost;
7   };
8
9   class Bellman_Ford{
10  public:
11    vector<bf_edge> es; // 辺
```

```
12    vector<int> d; // d[i]...頂点s から頂点 i までの最短距離
13    int V, E; // V は頂点数、E は辺数
14
15    Bellman_Ford(int v, int e) {
16      V = v;
17      E = e;
18      d = vector<int>(v);
19    }
20
21    void add(int from, int to, int cost) {
22      bf_edge ed = {from, to, cost};
23      es.push_back(ed);
24    }
25
26    // s 番目の頂点から各頂点への最短距離を求める
27    // true なら負の閉路が存在する
28    bool shortest_path(int s) {
29      for(int i = 0; i < V; i++) {
30        d[i] = 0;
31      }
32      for (int i = 0; i < 3*V; i++) {
33        for(int j = 0; j < E; j++) {
34          bf_edge e = es[j];
35          if(d[e.to] > d[e.from] + e.cost) {
36            d[e.to] = d[e.from] + e.cost;
37
38            // 3n 回目にも更新があるなら負の閉路が存在する
39            if(i == V - 1)return true;
40          }
41        }
42      }
43      return false;
44    }
45  };
```

ソースコード 14　bfs

```
1
2   // 各座標に格納したい情報を構造体にする
3   // 今回はX 座標,Y 座標,深さ (距離)を記述している
4   struct Corr {
5     int x;
6     int y;
7     int depth;
```

```
};
queue<Corr> q;
int bfs(vector<vector<int>> grid) {
  // 既に探索の場所を 1,探索していなかったら 0を格納する配列
  vector<vector<int>> ispassed(grid.size(), vector<int>(grid[0].size(), false));
  // このような記述をしておくと,この後の for 文が綺麗にかける
  int dx[8] = {1, 0, -1, 0};
  int dy[8] = {0, 1, 0, -1};
  while(!q.empty()) {
    Corr now = q.front();q.pop();
    /*
      今いる座標は (x,y)=(now.x, now.y)で,深さ (距離)はnow.depth である
      ここで,今いる座標がゴール (探索対象)なのか判定する
    */
    for(int i = 0; i < 4; i++) {
      int nextx = now.x + dx[i];
      int nexty = now.y + dy[i];

      // 次に探索する場所の X 座標がはみ出した時
      if(nextx >= grid[0].size()) continue;
      if(nextx < 0) continue;

      // 次に探索する場所の Y 座標がはみ出した時
      if(nexty >= grid.size()) continue;
      if(nexty < 0) continue;

      // 次に探索する場所が既に探索済みの場合
      if(ispassed[nexty][nextx]) continue;

      ispassed[nexty][nextx] = true;
      Corr next = {nextx, nexty, now.depth+1};
      q.push(next);
    }
  }
}
```

ソースコード 15　binarysearch

```

// vector v の中の n 以下の数で最大のものを返す
int bs(vector<ll> &v, ll x){
  int ok = -1; //これが x 以下
  int ng = v.size(); //x 以上
  // 問題によって ok と ng を入れ替える
```

```
  while(abs(ok - ng) > 1){
    int mid = (ok + ng)/2;
    if(v[mid] <= x) ok = mid;
    else ng = mid;
  }
  return ok;
}
```

ソースコード 16　bit

```
template <typename T>
class BIT {
  int N;
  vector<T> tree;

public:
  BIT(vector<T>& v) : N(v.size()), tree(vector<T>(v.size() + 1)) {
    rep(i, v.size()) { add(i, v[i]); }
  }
  BIT(int n) : N(n), tree(vector<T>(n + 1)) {}

  void add(int i, T x = 1) {
    for (++i; i <= N; i += i & -i) {
      tree[i] += x;
    }
  }

  T sum(int l) { // [0, l)の和
    T x = 0;
    for (; l; l -= l & -l) {
      x += tree[l];
    }
    return x;
  }

  T sum(int l, int r) { // [l, r)の和
    return sum(r) - sum(l);
  }

  T sum_back(int l) { // [l, N)の和
    return sum(N) - sum(l);
  }
};
```

ソースコード 17　boostlibrary

```

#include <boost/multiprecision/cpp_dec_float.hpp>
#include <boost/multiprecision/cpp_int.hpp>
namespace mp = boost::multiprecision;
// 任意長整数型
using Bint = mp::cpp_int;
// 仮数部が 10進数で 1024桁の浮動小数点数型 (TLE したら小さくする)
using Real = mp::number<mp::cpp_dec_float<1024>>;
```

ソースコード 18　codeforces

```
int Q; cin >> Q; while(Q--) {
```

ソースコード 19　combination

```

#define MAX_NCK 201010
ll f[MAX_NCK], rf[MAX_NCK];

// modinv も呼ぶ！！

bool isinit = false;

void init(void) {
  f[0] = 1;
  rf[0] = modinv(1);
  for(int i = 1; i < MAX_NCK; i++) {
    f[i] = (f[i - 1] * i) % MOD;
    rf[i] = modinv(f[i]);
  }
}

ll nCk(int n, int k) {
  if(!isinit) {
```

```
    init();
    isinit = 1;
  }
  if(n < k) return 0;
  ll nl = f[n]; // n!
  ll nkl = rf[n - k]; // (n-k)!
  ll kl = rf[k]; // k!
  ll nkk = (nkl * kl) % MOD;

  return (nl * nkk) % MOD;
}
```

ソースコード 20　combination2

```
// 逆元を使わないnCk
// なんでこうなるかわよくわからん
const int NCKMAX = 3;
int C[NCKMAX][NCKMAX];
bool isinit = false;
void init() {
  rep(i, NCKMAX) { C[i][0] = C[i][i] = 1; }
  for (int i = 1; i < NCKMAX; i++) {
    for (int j = 1; j < i; j++) {
      C[i][j] = (C[i - 1][j - 1] + C[i - 1][j]) % NCKMAX;
    }
  }
}

int nCk(int n, int k) {
  if (!isinit) {
    init();
    isinit = 1;
  }
  if (k < 0 || k > n) return 0;
  int ans = 1;
  while (n > 0) {
    int x = n % NCKMAX;
    int y = k % NCKMAX;
    n /= NCKMAX;
    k /= NCKMAX;
    ans = (ans * C[x][y]) % NCKMAX;
  }
  return ans;
```

```
}
```

ソースコード 21　conlis

```
int conlis(vector<int>& v) {
  vi dp(v.size() + 1, 0);
  int ans = 0;
  for(int i = 0; i < v.size(); i++) {
    dp[v[i]] = dp[v[i] - 1] + 1;
    ans = max(ans, dp[v[i]]);
  }
  return ans;
}
```

ソースコード 22　digitsum

```
int digsum(int n) {
  int res = 0;
  while(n > 0) {
    res += n%10;
    n /= 10;
  }
  return res;
}
```

ソースコード 23　dijkstra

```
class DIJKSTRA {
public:
  int V;

  struct dk_edge {
    int to;
    ll cost;
  };
```

```
10
11    typedef pair<ll, int> PI; // first は最短距離、second は頂点の番号
12    vector<vector<dk_edge> > G;
13    vector<ll> d; //これ答え。d[i]:=V[i]までの最短距離
14    vector<int> prev; //経路復元
15
16    DIJKSTRA(int size) {
17      V = size;
18      G = vector<vector<dk_edge> >(V);
19      prev = vector<int>(V, -1);
20    }
21
22    void add(int from, int to, ll cost) {
23      dk_edge e = {to, cost};
24      G[from].push_back(e);
25    }
26
27    void dijkstra(int s) {
28      // greater<P>を指定することでfirst が小さい順に取り出せるようにする
29      priority_queue<PI, vector<PI>, greater<PI> > que;
30      d = vector<ll>(V, LLINF);
31      d[s] = 0;
32      que.push(PI(0, s));
33
34      while (!que.empty()) {
35        PI p = que.top();
36        que.pop();
37        int v = p.second;
38        if (d[v] < p.first) continue;
39        for (int i = 0; i < G[v].size(); i++) {
40          dk_edge e = G[v][i];
41          if (d[e.to] > d[v] + e.cost) {
42            d[e.to] = d[v] + e.cost;
43            prev[e.to] = v;
44            que.push(PI(d[e.to], e.to));
45          }
46        }
47      }
48    }
49    vector<int> get_path(int t) {
50      vector<int> path;
51      for (; t != -1; t = prev[t]) {
52        // t が s になるまで prev[t]をたどっていく
53        path.push_back(t);
54      }
55      //このままだと t->s の順になっているので逆順にする
```

```
    reverse(path.begin(), path.end());
    return path;
  }
  void show(void) {
    for (int i = 0; i < d.size() - 1; i++) {
      cout << d[i] << \"␣\";
    }
    cout << d[d.size() - 1] << endl;
  }
};
```

ソースコード 24　doubleSort

```

template <class T, class U>
void doubleSort(vector<T>& v, vector<U>& w) {
  assert(v.size() == w.size());
  vector<pair<T, U>> p(v.size());
  rep(i, v.size()) { p[i] = mp(v[i], w[i]); }
  Sort(p);
  rep(i, p.size()) {
    v[i] = p[i].first;
    w[i] = p[i].second;
  }
}
```

ソースコード 25　eratosthenes

```

class Sieve{
  int N;
  void eratosmake(void) {
    // i を残して i の倍数を消していく
    for(int i = 2; i < N; i++) {
      if(nums[i] == 1) {
        for(int j = i + i; j < N; j += i){
          nums[j] = i;
        }
      }
    }
  }
```

```

  public:
    vector<int> nums;
    Sieve(int n):N(n){
      nums = vi(n+1, 1);
      eratosmake();
    }
    bool isPrime(int n) {
      return nums[n] == 1;
    }
    int minPrimeFactor(int n) {
      return nums[n];
    }
  };
```

ソースコード 26　extgcd

```
  // x,y に ax + by = gcd(a, b) を満たす値が格納される
  // 返り値は<gcd(a, b), x, y>
  // auto [g, x, y] = extgcd(a, b);のように呼び出す
  tuple<ll, ll, ll> extgcd(ll a, ll b) {
    if (b == 0) {
      return {a, 1, 0};
    }
    auto [g, x, y] = extgcd(b, a % b);
    return {g, y, x - a / b * y};
  }
```

ソースコード 27　gcd

```
ll gcd(ll a, ll b) { return b ? gcd(b, a%b) : a;
```

ソースコード 28　getline

```

  string getline() {
    string s;
```

```
4     fflush(stdin);
5     getline(cin, s);
6     return s;
7   }
```

ソースコード 29　gpriorityqueue

```
1 priority_queue<int, vector<int>, greater<int> > queue
```

ソースコード 30　hakidashi

```
1
2   #define RANK 20 // 20元連立方程式まで解ける
3   /*
4   使用方法
5     double a[RANK][RANK+1];
6     int i, n;
7     a[0][0] = 2; a[0][1] = 3; a[0][2] = 1; a[0][3] = 4;
8     a[1][0] = 4; a[1][1] = 1; a[1][2] = -3 ; a[1][3] = -2;
9     a[2][0] = -1; a[2][1] = 2; a[2][2] = 2; a[2][3] = 2;
10    n = 3;
11    hakidashi(a,n);
12  */
13  void hakidashi(double a[][RANK+1], int n) {
14    double piv, t;
15    int i, j, k;
16    for (k = 0; k < n; k++) {
17      piv = a[k][k];
18      for (j = k; j < n + 1; j++) {
19        a[k][j] = a[k][j]/piv;
20      }
21      for (i = 0; i < n; i++) {
22        if (i != k) {
23          t = a[i][k];
24          for (j = k; j < n+1; j++) {
25            a[i][j] = a[i][j] - t*a[k][j];
26          }
27        }
28      }
29    }
30  }
```

ソースコード 31　hutei

```
1
2  void hutei(int a, int b, int c, bool minus) {
3    vector<int> arr;
4
5    // A / B = div...mod
6    int A = max(a, b);
7    int B = min(a, b);
8    int div, mod;
9
10   while(1) {
11     div = A/B;
12     mod = A%B;
13     arr.push_back(div);
14
15     A = B;
16     B = mod;
17
18     if(mod == 1) {
19       break;
20     }
21   }
22
23   vector<vector<int> > calc(2, vector<int> (arr.size() + 1, INF));
24
25   for(int i = 0; i < arr.size() - 1; i++) {
26     calc[0][i] = -arr[i];
27   }
28   calc[1][arr.size() - 1] = -arr[arr.size() - 1];
29   calc[1][arr.size()] = 1;
30
31   for(int i = arr.size()-2; i >= 0; i--) {
32     calc[1][i] = calc[0][i]*calc[1][i + 1] + calc[1][i + 2];
33   }
34
35   int x = calc[1][0]*c;
36   int y = calc[1][1]*c;
37
38   if(minus) {
39     y *= -1;
40   }
41   cout << a << \"(\" << b << \"m␣+␣\" << x << \")\";
```

```
42    if(minus) {
43      cout << \"␣-␣\";
44    } else {
45      cout << \"␣+␣\";
46    }
47    cout << b << \"(\" << a << \"m␣+␣\" << y << \")\" << \"␣=␣\" << c << endl;
48  }
```

ソースコード 32　icpctemplate

```
1
2   #include <iostream>
3   #include <algorithm>
4   #include <string>
5   #include <vector>
6   using namespace std;
7   typedef long long int ll;
8   typedef vector<int> vi;
9   #define rep(i,n) for(int i = 0; i < (n); ++i)
10  int main(void){}
```

ソースコード 33　indexdistance

```
1
2   int indexdistance(vector<int> distance_array, char c) {
3     return static_cast<int>(std::distance(std::begin(distance_array), std::find(std::
          begin(distance_array), std::end(distance_array), c)));
4   }
```

ソースコード 34　intersect

```
1
2   void intersect(set<int> &Set_A, set<int> &Set_B, set<int> &Set_res) {
3     set_intersection(Set_A.begin(), Set_A.end(), Set_B.begin(), Set_B.end(), inserter(
          Set_res, Set_res.end()));
4   }
```

ソースコード 35　isPrime

```
1
2  bool isPrime(ll x){
3    if(x < 2)return 0;
4    else if(x == 2) return 1;
5    if(x%2 == 0) return 0;
6    for(ll i = 3; i*i <= x; i += 2) if(x%i == 0) return 0;
7    return 1;
8  }
```

ソースコード 36　kika

```
1
2  /* ==== 幾何ライブラリ ==== */
3  /* 点 */
4  struct Point {
5    double x;
6    double y;
7    Point(double x = 0.0, double y = 0.0) : x(x), y(y) {}
8
9    // === 四則演算の定義 ===
10   friend inline Point operator + (const Point &p, const Point &q) {return Point(p.x +
         q.x, p.y + q.y);}
11   friend inline Point operator - (const Point &p, const Point &q) {return Point(p.x -
         q.x, p.y - q.y);}
12   friend inline Point operator * (const Point &p, const double a) {return Point(p.x *
         a, p.y * a);}
13   friend inline Point operator * (const double a, const Point &p) {return Point(a * p
        .x, a * p.y);}
14   friend inline Point operator * (const Point &p, const Point &q) {return Point(p.x *
         q.x - p.y * q.y, p.x * q.y + p.y * q.x);}
15   friend inline Point operator / (const Point &p, const double a) {return Point(p.x /
         a, p.y / a);}
16
17   // === その他の演算 ===
18   // 反時計回りに 90度回転
19   friend Point rot90(const Point &p) {return Point(-p.y, p.x);}
20
21   // 直線b,c からみて，a がどちら側にいるか判定
22   // 1: b を上 c を下とした時に a が右側にある，-1: a が左側にある，0: a は直線 bc 上
```

```
23    friend int simple_ccw(const Point &a, const Point &b, const Point &c) {
24      if(OuterProduct(b-a, c-a) > EPS) return 1;
25      if(OuterProduct(b-a, c-a) < -EPS) return -1;
26      return 0;
27    }
28
29    // 内積
30    friend inline double InnerProduct(const Point &p, const Point &q) {return p.x * q.x
          + p.y * q.y;}
31    // 外積
32    friend inline double OuterProduct(const Point &p, const Point &q) {return p.x * q.y
          - p.y * q.x;}
33
34    // 二次元のノーム (ユークリッド距離)を計算
35    friend inline double norm2(const Point &p) {return sqrt(InnerProduct(p, p));}
36
37    // === 出力 ===
38    friend ostream& operator << (ostream &s, const Point &p) {return s << '(' << p.x
          << \"␣\" << p.y << ')';}
39  };
40
41  /* 線 */
42  struct Line {
43    vector<Point> line;
44
45    Line(void) {}
46    // 線分の時
47    Line(Point a, Point b = Point(0.0, 0.0)) {
48      // x 座標が小さい方->y 座標が小さい順にしておく
49      if(a.x > b.x) {
50        swap(a, b);
51      } else if(a.x == b.x && a.y > b.y) {
52        swap(a, b);
53      }
54      line.push_back(a);
55      line.push_back(b);
56    }
57    // 多角形などの時
58    Line(vector<Point> L) {
59      /*
60      // 基本はソートするとバグるのでしないこと
61      sort(L.begin(), L.end(), [](Point const& lhs, Point const& rhs) {
62        if(lhs.x != rhs.x) return lhs.x < rhs.x;
63        else if(lhs.y != rhs.y) return lhs.y < rhs.y;
64        return true;
65      });
```

```
66      */
67      line = L;
68    }
69
70    // === 出力 ===
71    friend ostream& operator << (ostream &s, const Line &l) {
72      s << '{';
73      rep(i, l.line.size()) {
74        if(i) {
75          s << \"␣\";
76        }
77        s << l.line[i];
78      }
79      s << '}';
80      return s;
81    }
82  };
83
84  /* 単位変換 */
85  double torad(int deg) {return (double)(deg) * MATHPI / 180;}
86  double todeg(double ang) {return ang * 180 / MATHPI;}
87
88  /* 直線や多角形の交点 */
89  Line crosspoint(const Line &L, const Line &M) {
90    Line res;
91    Line l = L;
92    Line m = M;
93    l.line.push_back(l.line[0]);
94    m.line.push_back(m.line[0]);
95    for(int i = 0; i < l.line.size() - 1; i++) {
96      for(int j = 0; j < m.line.size() - 1; j++) {
97        double d = OuterProduct(m.line[j + 1] - m.line[j], l.line[i + 1] - l.line[i]);
98        if(abs(d) < EPS) continue;
99        res.line.push_back(l.line[i] + (l.line[i + 1] - l.line[i]) * OuterProduct(m.
              line[j + 1] - m.line[j], m.line[j + 1] - l.line[i]) / d);
100     }
101   }
102   return res;
103 }
104
105 /* 外心 */
106 Point gaisin(const Point a, const Point b, const Point c) {
107   // 外心は三角形の二つの辺の垂直二等分線の交点
108   Line ab( (a + b)/2, (a + b)/2 + rot90(a - b) );
109   Line bc( (b + c)/2, (b + c)/2 + rot90(b - c) );
110   return crosspoint(ab, bc).line[0];
```

```
111 }
112
113 /* 最小包含円 */
114 double SmallestEnclosingCircle(const vector<Point> &V) {
115   int N = V.size();
116   if(N <= 1) return 0;
117
118   // 最小包含円の中心の候補
119   vector<Point> CenterCandidate;
120   for(int i = 0; i < N; i++) {
121     for(int j = i + 1; j < N; j++) {
122       // 最小包含円の円弧上に点が２つしかないの時
123       CenterCandidate.push_back( (V[i] + V[j]) / 2 );
124       for(int k = j + 1; k < N; k++) {
125         if(!simple_ccw(V[i], V[j], V[k])) {
126           // 三点が一直線上にある
127           continue;
128         }
129         // 三点の外心が円の中心
130         Point r = gaisin(V[i], V[j], V[k]);
131         CenterCandidate.push_back(r);
132       }
133     }
134   }
135
136   double res = INF;
137   rep(c, CenterCandidate.size()) {
138     double tmp = 0.0;
139     rep(v, V.size()) {
140       // 中心からの距離が最大の点との距離が,包含円の半径になる
141       chmax(tmp, norm2(V[v] - CenterCandidate[c]));
142     }
143     // 候補の包含円の中で,半径が最小の包含円が最小包含円になる.
144     chmin(res, tmp);
145   }
146   return res;
147 }
148
149 // 直線a-b と点 p の距離
150 long double distance(const Point& p, const Point& a, const Point& b) {
151   long double A = b.y - a.y;
152   long double B = a.x - b.x;
153   long double C = a.y * b.x - b.y * a.x;
154   return abs(A * p.x + B * p.y + C) / sqrt(A * A + B * B);
155 }
```

ソースコード 37　knapsack

```
1
2  int knapsack(int n, int W, vi w, vi v) {
3    vvi dp(n + 1, vi (W + 1, 0));
4    for(int i = 1; i <= n; i++) {
5      for(int j = 1; j <= W; j++) {
6        if(j - w[i] >= 0) {
7          chmax(dp[i][j], dp[i - 1][j - w[i]] + v[i]);
8        }
9        chmax(dp[i][j], dp[i - 1][j]);
10     }
11   }
12   return dp[n][W];
13 }
```

ソースコード 38　kruskal

```
1
2  // Union-Find も呼んで！！
3  struct kr_edge {
4    int u; // 辺の片方,from ではないので二回辺を張る必要はない
5    int v; // 辺のもう片方
6    int cost;
7
8    // コストの大小で順序定義
9    bool operator<(const kr_edge& e) const {
10     return cost < e.cost;
11   }
12 };
13 class Kruskal{
14   public:
15
16   bool comp(const kr_edge& e1, const kr_edge& e2) { // sort 関数の比較関数
17     return e1.cost < e2.cost;
18   }
19
20   vector<kr_edge> es; // 辺の集合
21   vector<kr_edge> minst; // 最小全域木に用いられる辺の集合
22   int V, E; // 頂点数と辺数
23
```

```
24    Kruskal(int v) {
25      V = v;
26    }
27
28    void add(int v, int u, int cost){
29      kr_edge e = {v, u, cost};
30      es.push_back(e);
31    }
32
33    int kruskal(void) {
34      sort(es.begin(), es.end()); // kr_edge.cost が小さい順にソートされる
35      UnionFind uni(V); //union-find の初期化
36      int res = 0;
37      for(int i = 0; i < es.size(); i++) {
38        kr_edge e = es[i];
39        if(uni.root(e.u) != uni.root(e.v)) {
40          uni.connect(e.u, e.v);
41          res += e.cost;
42          minst.push_back(e);
43        }
44      }
45      return res;
46    }
47
48    void show(void) {
49      vvi v(V, vi(V, -1));
50      for(int i = 0; i < minst.size(); i++) {
51        v[minst[i].u][minst[i].v] = minst[i].cost;
52        v[minst[i].v][minst[i].u] = minst[i].cost;
53      }
54      for(int i = 0; i < V; i++) {
55        for(int j = 0; j < V; j++) {
56          if(v[i][j] == -1) {
57            printf(\"␣␣___␣\");
58          } else {
59            printf(\"%4d␣\" v[i][j]);
60          }
61        }
62        cout << endl;
63      }
64    }
65  };
```

ソースコード 39　lambdaSort

```
1
2  template<class T>
3  void lambdaSort(vector<T> &v) {
4    sort(all(v), [](auto const& l, auto const& r) {
5      return l.fi > r.fi; // このbool式が成り立つ時入れ替える
6    });
7  }
```

ソースコード 40　lca

```
1
2  class LCA {
3    int n;
4    int MAX;
5    vvi doubling, v;
6
7    void init() {
8      rep(i, n) {
9        for(auto j : v[i]) {
10         doubling[0][j] = i;
11       }
12     }
13     for(int i = 1; i < MAX; i++) {
14       rep(j, n) {
15         doubling[i][j] = doubling[i - 1][doubling[i - 1][j]];
16       }
17     }
18     depth[0] = 0;
19     dfs(0, -1);
20   }
21
22   void dfs(const int crrPos, const int befPos) {
23     for(auto i : v[crrPos]) {
24       if(i == befPos || depth[i] != -1) {
25         continue;
26       }
27       depth[i] = depth[crrPos] + 1;
28       dfs(i, crrPos);
29     }
30   }
31
32 public:
```

```
33    vi depth;
34
35    // v は 0 が根の bfs 木にする．親->子のように辺を張る．
36    LCA(vvi &_v) : v(_v), n(_v.size()) {
37      MAX = ceil(log2(n));
38      doubling = vvi(MAX, vi(n, 0));
39      depth = vi(n, -1);
40      init();
41    }
42
43    void show(const int height = 0) {
44      rep(i, ((!height)?MAX:height)) {
45        dump(doubling[i]);
46      }
47      dump(depth);
48    }
49
50    // ダブリングでV の num 個親の祖先を調べる
51    int doublingNode(int V, const int num) {
52      rep(i, MAX) {
53        if((1LL<<i) & num) {
54          V = doubling[i][V];
55        }
56      }
57      return V;
58    }
59
60    int lca(int A, int B) {
61      // A のが深い位置にあるようにする
62      if(depth[A] < depth[B]) {
63        swap(A, B);
64      }
65      A = doublingNode(A, depth[A] - depth[B]);
66      if(A == B) {
67        return A;
68      }
69
70      for (int k = MAX - 1; k >= 0; k--) {
71        if (doubling[k][A] != doubling[k][B]) {
72          A = doubling[k][A];
73          B = doubling[k][B];
74        }
75      }
76      return doubling[0][A];
77    }
78  };
```

ソースコード 41　lcm

```
1
2  // gcdも呼ぶ！！！
3  ll lcm(ll a, ll b) { return a / gcd(a, b) * b;}
```

ソースコード 42　lcs

```
1
2  string lcs(string s, string t) {
3    vvi dp(s.size() + 1, vi(t.size() + 1));
4
5    for(int i = 0; i < s.size(); i++) {//LCS
6      for(int j = 0; j < t.size(); j++) {
7        if(s[i] == t[j]) {
8          dp[i + 1][j + 1] = dp[i][j] + 1;
9        }
10       else{
11         dp[i + 1][j + 1] = max(dp[i][j + 1], dp[i + 1][j]);
12       }
13     }
14   }
15   // 復元
16   string ans = \"\";
17   int i = (int)s.size(), j = (int)t.size();
18   while (i > 0 && j > 0){
19     // (i-1, j) -> (i, j) と更新されていた場合
20     if (dp[i][j] == dp[i-1][j]) {
21       --i; // DP の遷移を遡る
22     }
23     // (i, j-1) -> (i, j) と更新されていた場合
24     else if (dp[i][j] == dp[i][j-1]) {
25       --j; // DP の遷移を遡る
26     }
27     // (i-1, j-1) -> (i, j) と更新されていた場合
28     else {
29       ans = s[i-1] + ans;
30       --i, --j; // DP の遷移を遡る
31     }
32   }
33   return ans;
```

```
}
```

ソースコード 43　lis

```
int lis(vi& v) {
  vi dp(1, v[0]);
  for(int i = 1; i < v.size(); i++) {
    if(v[i] > dp[dp.size() - 1]) {
      dp.push_back(v[i]);
    } else {
      int pos = distance(dp.begin(), lower_bound(dp.begin(), dp.end(), v[i]));
      dp[pos] = v[i];
    }
  }
  return dp.size();
}
```

ソースコード 44　millerrabin

```
// x の n 乗%mod を計算
ll pow_mod_i128(__int128_t x, ll n, ll mod = MOD) {
  ll res = 1;
  x %= MOD;
  while (n) {
    if (n & 1) res = res * x % mod;
    x = x * x % mod;
    n >>= 1;
  }
  return res;
}

// 素数判定
bool miller_rabin(ll x) {
  if (x <= 2) {
    return x == 2;
  }
  if (x % 2 == 0) {
    return false;
  }
```

```
  ll d = x - 1;
  while (d % 2 == 0) d /= 2;
  for (ll a : {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37}) {
    if (x <= a) {
      return true;
    }
    ll t = d;
    ll y = pow_mod_i128(a, t, x);
    while (t != x - 1 && y != 1 && y != x - 1) {
      y = __int128_t(y) * y % x;
      t <<= 1;
    }
    if (y != x - 1 && t % 2 == 0) {
      return false;
    }
  }
  return true;
}
```

ソースコード 45　mintcombination

```
#define MAX_MINT_NCK 201010
mint f[MAX_MINT_NCK], rf[MAX_MINT_NCK];

bool isinit = false;

void init() {
  f[0] = 1;
  rf[0] = 1;
  for(int i = 1; i < MAX_MINT_NCK; i++) {
    f[i] = f[i - 1] * i;
    // rf[i] = modinv(f[i].x);
    rf[i] = f[i].pow(f[i], MOD - 2);
  }
}

mint nCk(mint n, mint k) {
  if(n < k) return 0;
  if(!isinit) {
    init();
    isinit = 1;
  }
  mint nl = f[n.x]; // n!
```

```
    mint nkl = rf[n.x - k.x]; // (n-k)!
    mint kl = rf[k.x]; // k!
    mint nkk = (nkl.x * kl.x);

    return nl * nkk;
  }
```

ソースコード 46　modint

```

struct mint {
  ll x;
  mint(ll _x = 0) : x((_x % MOD + MOD) % MOD) {}

  /* 初期化子 */
  mint operator+() const { return mint(x); }
  mint operator-() const { return mint(-x); }

  /* 代入演算子 */
  mint& operator+=(const mint a) {
    if ((x += a.x) >= MOD) x -= MOD;
    return *this;
  }
  mint& operator-=(const mint a) {
    if ((x += MOD - a.x) >= MOD) x -= MOD;
    return *this;
  }
  mint& operator*=(const mint a) {
    (x *= a.x) %= MOD;
    return *this;
  }
  mint& operator/=(const mint a) {
    x *= modinv(a).x;
    x %= MOD;
    return (*this);
  }
  mint& operator%=(const mint a) {
    x %= a.x;
    return (*this);
  }
  mint& operator++() {
    ++x;
    if (x >= MOD) x -= MOD;
    return *this;
```

```
36  }
37  mint& operator--() {
38    if (!x) x += MOD;
39    --x;
40    return *this;
41  }
42  mint& operator&=(const mint a) {
43    x &= a.x;
44    return (*this);
45  }
46  mint& operator|=(const mint a) {
47    x |= a.x;
48    return (*this);
49  }
50  mint& operator^=(const mint a) {
51    x ^= a.x;
52    return (*this);
53  }
54  mint& operator<<=(const mint a) {
55    x *= pow(2, a).x;
56    return (*this);
57  }
58  mint& operator>>=(const mint a) {
59    x /= pow(2, a).x;
60    return (*this);
61  }
62
63  /* 算術演算子 */
64  mint operator+(const mint a) const {
65    mint res(*this);
66    return res += a;
67  }
68  mint operator-(const mint a) const {
69    mint res(*this);
70    return res -= a;
71  }
72  mint operator*(const mint a) const {
73    mint res(*this);
74    return res *= a;
75  }
76  mint operator/(const mint a) const {
77    mint res(*this);
78    return res /= a;
79  }
80  mint operator%(const mint a) const {
81    mint res(*this);
```

```
82    return res %= a;
83  }
84  mint operator&(const mint a) const {
85    mint res(*this);
86    return res &= a;
87  }
88  mint operator|(const mint a) const {
89    mint res(*this);
90    return res |= a;
91  }
92  mint operator^(const mint a) const {
93    mint res(*this);
94    return res ^= a;
95  }
96  mint operator<<(const mint a) const {
97    mint res(*this);
98    return res <<= a;
99  }
100 mint operator>>(const mint a) const {
101   mint res(*this);
102   return res >>= a;
103 }
104
105 /* 条件演算子 */
106 bool operator==(const mint a) const noexcept { return x == a.x; }
107 bool operator!=(const mint a) const noexcept { return x != a.x; }
108 bool operator<(const mint a) const noexcept { return x < a.x; }
109 bool operator>(const mint a) const noexcept { return x > a.x; }
110 bool operator<=(const mint a) const noexcept { return x <= a.x; }
111 bool operator>=(const mint a) const noexcept { return x >= a.x; }
112
113 /* ストリーム挿入演算子 */
114 friend istream& operator>>(istream& is, mint& m) {
115   is >> m.x;
116   m.x = (m.x % MOD + MOD) % MOD;
117   return is;
118 }
119 friend ostream& operator<<(ostream& os, const mint& m) {
120   os << m.x;
121   return os;
122 }
123
124 /* その他の関数 */
125 mint modinv(mint a) { return pow(a, MOD - 2); }
126 mint pow(mint x, mint n) {
127   mint res = 1;
```

```
    while (n.x > 0) {
      if ((n % 2).x) res *= x;
      x *= x;
      n.x /= 2;
    }
    return res;
  }
  mint powll(mint x, ll n) {
    mint res = 1;
    while (n > 0) {
      if (n % 2) res *= x;
      x *= x;
      n /= 2;
    }
    return res;
  }
};
```

ソースコード 47　modinv

```
1
2   // (a/b)%P の場合は，(a%P)*modinv(b)%P とする
3   ll modinv(ll a) {
4     ll b = MOD, u = 1, v = 0;
5     while (b) {
6       ll t = a / b;
7       a -= t * b; swap(a, b);
8       u -= t * v; swap(u, v);
9     }
10    u %= MOD;
11    if (u < 0) u += MOD;
12    return u;
13  }
```

ソースコード 48　ngcd

```
1
2   // gcd も呼ぶ！！！
3   ll ngcd(vector<ll> a){
4     ll d;
5     d = a[0];
```

```
6     for(int i = 1; i < a.size() && d != 1; i++) d = gcd(a[i], d);
7     return d;
8   }
```

ソースコード 49　nijihouteishiki

```
1
2   /*
3     aX^2+bX+c=0の解を求める
4     出力はこんな感じ
5     if(x1 == DBL_MIN)cout<<\"解なし\"<<endl;
6     else if(x1==DBL_MAX)cout<<\"不定\"<<endl;
7     else if(!i)cout<<x1<<\" , \"<<x2<<endl;
8     else cout<<x1<<\" +- \"<<x2<<\"i\"<<endl;
9   */
10  double x1, x2;
11  bool i = false;
12  void quadeq(double a, double b, double c){
13    double d, x;
14    if(a != 0){
15      b /= a; c /= a;
16      if(c != 0){
17        b /= 2;
18        d = b*b - c;
19        if(d > 0){
20          if(b > 0) x = -b - sqrt(d);
21          else x = -b + sqrt(d);
22          x1 = x; x2 = c/x;
23        }else if(d < 0){
24          x1 = -b; x2 = sqrt(-d); i = true;
25        }else{
26          x1 = x2 = -b;
27        }
28      }else{
29        x1 = -b; x2 = 0;
30      }
31    }else if(b != 0){
32      x1 = x2 = -c/b;
33    }
34    else if(c != 0) x1 = x2 = DBL_MIN;
35    else x1 = x2 = DBL_MAX;
36  }
```

ソースコード 50　nlcm

```
1
2  // gcdも呼ぶ！！！
3  // lcmも呼ぶ！！！
4  ll nlcm(vector<ll> numbers) {
5    ll l = numbers[0];
6    for (int i = 1; i < numbers.size(); i++) {
7      l = lcm(l, numbers[i]);
8    }
9    return l;
10 }
```

ソースコード 51　pollardrho

```
1
2  // miller_rabinを呼ぶ！！
3  ll gcd_norecursive(ll _a, ll _b) {
4    ull a = abs(_a), b = abs(_b);
5    if (a == 0) return b;
6    if (b == 0) return a;
7    int shift = __builtin_ctz(a | b);
8    a >>= __builtin_ctz(a);
9    do {
10     b >>= __builtin_ctz(b);
11     if (a > b) swap(a, b);
12     b -= a;
13   } while (b);
14   return (a << shift);
15 }
16
17 ll pollard_single(ll n) {
18   auto rand = [&](ll x) -> ll { return (__int128_t(x) % n * x % n + 1) % n; };
19   if (miller_rabin(n)) {
20     return n;
21   }
22   if (n % 2 == 0) {
23     return 2;
24   }
25   ll st = 0;
26   while (true) {
```

```
    st++;
    ll x = st, y = rand(x), d = 1;
    while (d == 1) {
      d = gcd_norecursive(y - x + n, n);
      if (d == n || d == 0) {
        break;
      } else if (d != 1) {
        return d;
      }
      x = rand(x);
      y = rand(rand(y));
    }
  }
};

// 受け取るときにソートすること
vll pollard_rho(ll n) {
  if (n == 1) {
    return {};
  }
  ll x = pollard_single(n);
  if (x == n) return {x};
  vll primes = pollard_rho(x);
  vll primes_inverse = pollard_rho(n / x);
  primes.insert(primes.end(), primes_inverse.begin(), primes_inverse.end());
  return primes;
}
```

ソースコード 52　powmod

```

// x の n 乗%mod を計算
ll pow_mod(ll x, ll n, ll mod = MOD) {
  ll res = 1;
  x %= MOD;
  while(n > 0) {
    if(n & 1) res = res*x%mod;
    x = x*x%mod;
    n >>= 1;
  }
  return res;
}
```

ソースコード 53　primedissasembly

```
1
2  map<ll, ll> prime;
3  void factorize(ll n) {
4    for(ll i = 2; i * i <= n; i++) {
5      while(n % i == 0) {
6        prime[i]++;
7        n /= i;
8      }
9    }
10   if(n != 1) {
11     prime[n] = 1;
12   }
13 }
```

ソースコード 54　printvector

```
1
2  template<class T>
3  void print_vector(vector<T> &v) {
4    rep(i, v.size()) {
5      if(!i) cout << v[i];
6      else cout << \"␣\" << v[i];
7    }
8    cout << endl;
9  }
```

ソースコード 55　sanjihouteishiki

```
1
2  // 三次方程式 ax^3+bx^2+cx+d=0を解く
3  double ans1=0, ans2=0, ans3=0;
4  void cardano(double a, double b, double c, double d){
5    double p, q, t, a3, b3, x1, x2, x3;
6    b /= (3*a); c /= a; d /= a;
7    p = b*b - c/3;
8    q = (b*(c - 2*b*b) - d)/2;
```

```
    a = q*q - p*p*p;
    if(a == 0){
      q = cbrt(q); x1 = 2*q - b; x2 = -q - b;
      cout << \"x=\" << x1 << \"␣\" << x2 << \"(重解)\" << endl;
      ans1 = x1; ans2 = x2;
    }else if(a > 0){
      if(q > 0) a3 = cbrt(q + sqrt(a));
      else a3 = cbrt(q - sqrt(a));
      b3 = p/a3;
      x1 = a3 + b3 - b; x2 = -0.5 + (a3 + b3) - b;
      x3 = fabs(a3 - b3)*sqrt(3.0)/2;
      cout << \"x=\" << x1 << \";␣\" << x2 << \"+-␣\" << x3 << \"i\" << endl;
      ans1 = x1; ans2 = x2; ans3 = x3;
    }else{
      a = sqrt(p); t = acos(q/(p*a)); a *= 2;
      x1 = a*cos(t/3) - b;
      x2 = a*cos((t+2*M_PI)/3) - b;
      x3 = a*cos((t+4*M_PI)/3) - b;
      cout << \"x=\" << x1 << \"␣\" << x2 << \"␣\" << x3 << endl;
      ans1 = x1; ans2 = x2; ans3 = x3;
    }
  }
```

ソースコード 56　stringcount

```
1
2  int stringcount(string s, char c) {
3    return count(s.cbegin(), s.cend(), c);
4  }
```

ソースコード 57　templete

```
1
2  #include <algorithm>
3  #include <bitset>
4  #include <cassert>
5  #include <cctype>
6  #include <cfloat>
7  #include <climits>
8  #include <cmath>
9  #include <cstdio>
```

```
#include <deque>
#include <iomanip>
#include <iostream>
#include <list>
#include <map>
#include <queue>
#include <random>
#include <set>
#include <stack>
#include <string>
#include <unordered_set>
#include <vector>
#pragma GCC target(\"avx2\")
#pragma GCC optimize(\"O3\")
#pragma GCC optimize(\"unroll-loops\")
using namespace std;
typedef long double ld;
typedef long long int ll;
typedef unsigned long long int ull;
typedef vector<int> vi;
typedef vector<char> vc;
typedef vector<bool> vb;
typedef vector<double> vd;
typedef vector<string> vs;
typedef vector<ll> vll;
typedef vector<pair<int, int>> vpii;
typedef vector<pair<ll, ll>> vpll;
typedef vector<vi> vvi;
typedef vector<vvi> vvvi;
typedef vector<vc> vvc;
typedef vector<vs> vvs;
typedef vector<vll> vvll;
typedef pair<int, int> P;
typedef map<int, int> mii;
typedef set<int> si;
#define rep(i, n) for (ll i = 0; i < (n); ++i)
#define rrep(i, n) for (int i = 1; i <= (n); ++i)
#define irep(it, stl) for (auto it = stl.begin(); it != stl.end(); it++)
#define drep(i, n) for (int i = (n)-1; i >= 0; --i)
#define fin(ans) cout << (ans) << '\n'
#define STLL(s) strtoll(s.c_str(), NULL, 10)
#define mp(p, q) make_pair(p, q)
#define pb(n) push_back(n)
#define all(a) a.begin(), a.end()
#define Sort(a) sort(a.begin(), a.end())
#define Rort(a) sort(a.rbegin(), a.rend())
```

```
#define fi first
#define se second
// #include <atcoder/all>
// using namespace atcoder;
constexpr int dx[8] = {1, 0, -1, 0, 1, -1, -1, 1};
constexpr int dy[8] = {0, 1, 0, -1, 1, 1, -1, -1};
template <class T, class U>
inline bool chmax(T& a, U b) {
  if (a < b) {
    a = b;
    return 1;
  }
  return 0;
}
template <class T, class U>
inline bool chmin(T& a, U b) {
  if (a > b) {
    a = b;
    return 1;
  }
  return 0;
}
template <class T, class U>
ostream& operator<<(ostream& os, pair<T, U>& p) {
  cout << \"(\" << p.first << \"␣\" << p.second << \")\";
  return os;
}
template <class T>
inline void dump(T& v) {
  // if (v.size() > 100) {
  // cout << \"ARRAY IS TOO LARGE!!!\" << endl;
  // } else {
  irep(i, v) { cout << (*i) << ((i == --v.end()) ? '\n' : '␣'); }
  // }
}
template <class T, class U>
inline void dump(map<T, U>& v) {
  if (v.size() > 100) {
    cout << \"ARRAY␣IS␣TOO␣LARGE!!!\" << endl;
  } else {
    irep(i, v) { cout << i->first << \"␣\" << i->second << '\n'; }
  }
}
template <class T, class U>
inline void dump(pair<T, U>& p) {
  cout << p.first << \"␣\" << p.second << '\n';
```

```
}
inline void yn(const bool b) { b ? fin(\"yes\") : fin(\"no\"); }
inline void Yn(const bool b) { b ? fin(\"Yes\") : fin(\"No\"); }
inline void YN(const bool b) { b ? fin(\"YES\") : fin(\"NO\"); }
const int INF = INT_MAX;
constexpr ll LLINF = 1LL << 61;
// constexpr ll MOD = 1000000007; // 998244353; //
constexpr ld EPS = 1e-11;
void Case(int i) { printf(\"Case␣#%d:␣\" i); }
/* -------------------- ここまでテンプレ -------------------- */

int main() {

}
```

ソースコード 58　topologicalsort

```

vvi G(1000); // グラフ (リスト)

// トポロジカルソート
void rec(int v, vector<bool> &seen, vector<int> &order) {
  seen[v] = true;
  for (int i= 0; i < G[v].size(); i++) {
    int next = G[v][i];
    if (seen[next]) continue; // 既に訪問済みなら探索しない
    rec(next, seen, order);
  }
  order.push_back(v);
}

vector<int> topo(int N) { // N はノード数
  // 探索
  vector<bool> seen(N, 0); // 初期状態では全ノードが未訪問
  vector<int> order; // トポロジカルソート順
  for (int v = 0; v < N; ++v) {
    if (seen[v]) continue; // 既に訪問済みなら探索しない
    rec(v, seen, order);
  }
  reverse(order.begin(), order.end());
  return order;
}
```

ソースコード 59　unionfind

```
1
2  class UnionFind {
3  public:
4    // 親の番号を格納.親だった場合は-(その集合のサイズ)
5    vector<int> Parent;
6    // 重さの差を格納
7    vector<ll> diffWeight;
8
9    UnionFind(const int N) {
10     Parent = vector<int>(N, -1);
11     diffWeight = vector<ll>(N, 0);
12   }
13
14   // A がどのグループに属しているか調べる
15   int root(const int A) {
16     if (Parent[A] < 0) return A;
17     int Root = root(Parent[A]);
18     diffWeight[A] += diffWeight[Parent[A]];
19     return Parent[A] = Root;
20   }
21
22   // 自分のいるグループの頂点数を調べる
23   int size(const int A) {
24     return -Parent[root(A)];
25   }
26
27   // 自分の重さを調べる
28   ll weight(const int A) {
29     root(A); // 経路圧縮
30     return diffWeight[A];
31   }
32
33   // 重さの差を計算する
34   ll diff(const int A, const int B) {
35     return weight(B) - weight(A);
36   }
37
38   // A と B をくっ付ける
39   bool connect(int A, int B, ll W = 0) {
40     // W を root との重み差分に変更
41     W += weight(A);
```

```
42      W -= weight(B);
43
44      // A と B を直接つなぐのではなく、root(A)にroot(B)をくっつける
45      A = root(A);
46      B = root(B);
47
48      if (A == B) {
49        //すでにくっついてるからくっ付けない
50        return false;
51      }
52
53      // 大きい方 (A)に小さいほう (B)をくっ付ける
54      // 大小が逆だったらひっくり返す
55      if (size(A) < size(B)) {
56        swap(A, B);
57        W = -W;
58      }
59
60      // A のサイズを更新する
61      Parent[A] += Parent[B];
62      // B の親を A に変更する
63      Parent[B] = A;
64
65      // A は B の親であることが確定しているので B に W の重みを充てる
66      diffWeight[B] = W;
67
68      return true;
69    }
70  };
```

ソースコード 60　warshallfloyd

```
1
2   template <typename T>
3   class WAR_FLY {
4   public:
5     vector<vector<T>> d; // 辺の数
6     int V; // 頂点の数
7
8     WAR_FLY(int n) {
9       V = n;
10      d = vector<vector<T> > (n,vector<T>(n));
11      for(int i = 0; i < V; i++) {
12        for(int j = 0; j < V; j++) {
```

```
13          if(i == j) {
14            d[i][j] = 0;
15          }
16          else {
17            d[i][j] = LLINF;
18          }
19        }
20      }
21    }
22
23    void warshall_floyd(void) {
24      for(int k = 0; k < V; k++) {
25        for(int i = 0; i < V; i++) {
26          for(int j = 0; j < V; j++) {
27            d[i][j] = min((ll)d[i][j], (ll)d[i][k] + (ll)d[k][j]);
28          }
29        }
30      }
31    }
32
33    void add(int from, int to, T cost) {
34      d[from][to] = cost;
35    }
36
37    bool is_negative_loop(void) {
38      for(int i = 0; i < V; i++) {
39        if(d[i][i] < 0) return true;
40      }
41      return false;
42    }
43
44    void show(void) {
45      for(int i = 0; i < V; i++) {
46        for(int j = 0; j < V; j++) {
47          cout << d[i][j] << \"␣\";
48        }
49        cout << endl;
50      }
51    }
52  };
```

ソースコード 61　yakusuenum

```
1
```

```
// 約数を全て出力する
// 1 の場合は 1 しか出力しない
// N 自身は出力しない (15=1*3*5)
vector<ll> enum_div(ll n) {
  vector<ll> ret;
  for (ll i = 1 ; i * i <= n ; ++i) {
    if (n % i == 0) {
      ret.push_back(i);
      if (i != 1 && i * i != n) {
        ret.push_back(n / i);
      }
    }
  }
  return ret;
}
```

ソースコード 62　zip

```

map<ll, ll> zip;
int compress(vector<ll> x) {
  int unzip[x.size()];
  sort(x.begin(), x.end());
  x.erase(unique(x.begin(),x.end()),x.end());
  for(int i = 0; i < x.size(); i++){
    zip[x[i]] = i;
    unzip[i] = x[i];
  }
  return x.size();
}
```