幾何

```
const double EPS = 1e−8;
const double INF = 1e12;

//point
typedef complex<double> P;
namespace std {
	bool operator < (const P& a, const P& b) {
		return real(a) != real(b) ? real(a) < real(b) : imag(a) < imag(b);
	}
	bool cmp_y(const P &a, const P &b){
		return a.imag() != b.imag() ? a.imag() < b.imag() : a.real() < b.real();
	}
}
double cross(const P& a, const P& b) {
	return imag(conj(a)∗b);
}
double dot(const P& a, const P& b) {
	return real(conj(a)∗b);
}

// line
struct L : public vector<P> {
	L(const P& a, const P& b) {
		push_back(a); push_back(b);
	}
};

// polygon
typedef vector<P> G;

// circle
struct C {
	P p; double r;
	C(const P& p, double r) : p(p), r(r) {}
};
```

点と線分

```
//point
typedef complex<double> P;
namespace std {
	bool operator < (const P& a, const P& b) {
		return real(a) != real(b) ? real(a) < real(b) : imag(a) < imag(b);
	}
}
double cross(const P& a, const P& b) {
	return imag(conj(a)∗b);
}
double dot(const P& a, const P& b) {
	return real(conj(a)∗b);
}
// line
```

```
15  struct L : public vector<P> {
16          L(const P& a, const P& b) {
17                  push_back(a); push_back(b);
18          }
19  };
20
21  int ccw(P a, P b, P c) {
22    b -= a; c -= a;
23    if (cross(b, c) > 0) return +1; // counter clockwise
24    if (cross(b, c) < 0) return -1; // clockwise
25    if (dot(b, c) < 0) return +2; // c--a--b on line
26    if (norm(b) < norm(c)) return -2; // a--b--c on line
27    return 0;         // a--c--b on line
28  }
29
30  //L が直線,S が線分
31  bool intersectLL(const L &l, const L &m) {
32    return abs(cross(l[1]-l[0], m[1]-m[0])) > EPS || // non-parallel
33           abs(cross(l[1]-l[0], m[0]-l[0])) < EPS; // same line
34  }
35  bool intersectLS(const L &l, const L &s) {
36    return cross(l[1]-l[0], s[0]-l[0])* // s[0] is left of l
37           cross(l[1]-l[0], s[1]-l[0]) < EPS; // s[1] is right of l
38  }
39  bool intersectLP(const L &l, const P &p) {
40    return abs(cross(l[1]-p, l[0]-p)) < EPS;
41  }
42  bool intersectSS(const L &s, const L &t) {
43    return ccw(s[0],s[1],t[0])*ccw(s[0],s[1],t[1]) <= 0 &&
44           ccw(t[0],t[1],s[0])*ccw(t[0],t[1],s[1]) <= 0;
45  }
46  bool intersectSP(const L &s, const P &p) {
47    return abs(s[0]-p)+abs(s[1]-p)-abs(s[1]-s[0]) < EPS; // triangle inequality
48  }
49
50  P projection(const L &l, const P &p) {
51          double t = dot(p-l[0], l[0]-l[1]) / norm(l[0]-l[1]);
52          return l[0] + t*(l[0]-l[1]);
53  }
54  P reflection(const L &l, const P &p) {
55          return p + (double)2 * (projection(l, p) - p);
56  }
57  double distanceLP(const L &l, const P &p) {
58          return abs(p - projection(l, p));
59  }
60  double distanceLL(const L &l, const L &m) {
61          return intersectLL(l, m) ? 0 : distanceLP(l, m[0]);
62  }
63  double distanceLS(const L &l, const L &s) {
64          if (intersectLS(l, s)) return 0;
65          return min(distanceLP(l, s[0]), distanceLP(l, s[1]));
66  }
67  double distanceSP(const L &s, const P &p) {
```

```
68          const P r = projection(s, p);
69          if (intersectSP(s, r)) return abs(r − p);
70          return min(abs(s[0] − p), abs(s[1] − p));
71      }
72      double distanceSS(const L &s, const L &t) {
73          if (intersectSS(s, t)) return 0;
74                  return min({distanceSP(s, t[0]), distanceSP(s, t[1]), distanceSP(t, s[0]),
                        distanceSP(t, s[1])});
75      }
76      P crosspoint(const L &l, const L &m) {
77        double A = cross(l[1] − l[0], m[1] − m[0]);
78        double B = cross(l[1] − l[0], l[1] − m[0]);
79        if (abs(A) < EPS && abs(B) < EPS) return m[0]; // same line
80        if (abs(A) < EPS) assert(false); // !!!PRECONDITION NOT SATISFIED!!!
81        return m[0] + B / A ∗ (m[1] − m[0]);
82      }
```

多角形

```
1       // polygon
2       typedef vector<P> G;
3
4       P extreme(const vector<P> &po, const L &l) {
5         int k = 0;
6         for (int i = 1; i < po.size(); ++i)
7           if (dot(po[i], l[1]−l[0]) > dot(po[k], l[1]−l[0])) k = i;
8         return po[k];
9       }
10
11      enum { OUT, ON, IN };
12      int contains(const G& po, const P& p) {
13        bool in = false;
14        for (int i = 0; i < po.size(); ++i) {
15            P a = po[i] − p, b = po[(i+1)%po.size()] − p;
16          if (imag(a) > imag(b)) swap(a, b);
17          if (imag(a) <= 0 && 0 < imag(b))
18            if (cross(a, b) < 0) in = !in;
19          if (cross(a, b) == 0 && dot(a, b) <= 0) return ON;
20        }
21        return in ? IN : OUT;
22      }
23
24      double area2(const G& po) {
25            double A = 0;
26            for (int i = 0; i < po.size(); ++i)
27            A += cross(po[i], po[(i+1)%po.size()]);
28        //最後にまとめて割る 2
29        return A/2;
30      }
31
32      bool isconvex(const G &p) {
33            int n = p.size();
34            if(cross(p[0]−p[n−1],p[n−2]−p[n−1]) < 0) return false;
```

```
35          for(int i = 1; i < n−1; ++i) {
36                  if(cross(p[i+1]−p[i],p[i−1]−p[i]) < 0) return false;
37          }
38          return true;
39  }
40
41  G convex_hull(G ps) {
42    int n = ps.size(), k = 0;
43    sort(ps.begin(), ps.end(), cmp_y);
44    G r(2∗n);
45    for(int i = 0; i < n; i++){
46      while(k>1 && cross(r[k−1]−r[k−2],ps[i]−r[k−2]) < −EPS)k−−;
47      r[k++] = ps[i];
48    }
49    for(int i = n−2, t = k; i >= 0; i−−){
50      while(k>t && cross(r[k−1]−r[k−2],ps[i]−r[k−2]) < −EPS)k−−;
51      r[k++] = ps[i];
52    }
53    r.resize(k−1);
54    return r;
55  }
56
57  // caliper
58  double convex_diameter(const G &pt) {
59          const int n = pt.size();
60          if(n <= 1) return 0;
61          if(n == 2) return abs(pt[0]−pt[1]);
62
63          int i = 0, j = 0;
64          for(int k = 0; k < n; ++k){
65                  if(!(pt[i] < pt[k])) i = k;
66                  if(pt[j] < pt[k]) j = k;
67          }
68
69          double res = 0;
70          int si = i, sj = j;
71          while(i != sj || j != si) {
72                  res = max(res, abs(pt[i]−pt[j]));
73                  if(cross(pt[(i+1)%n]−pt[i],pt[(j+1)%n]−pt[j]) < 0) i = (i+1)%n;
74                  else j = (j+1)%n;
75          }
76          return res;
77  }
78
79  // 凸多角形po を l で切った左の多角形を返す
80  G convex_cut(const G& po, const L& l) {
81    G Q;
82    for (int i = 0; i < po.size(); ++i) {
83        P A = po[i], B = po[(i+1)%po.size()];
84      if (ccw(l[0], l[1], A) != −1) Q.push_back(A);
85        if (ccw(l[0], l[1], A)∗ccw(l[0], l[1], B) < 0) {
86                  Q.push_back(crosspoint(L(A, B), l));
87          }
```

```
88      }
89      return Q;
90  }
91
92  //最近点対 嘘
93  double closestPair(G p, int flag=1) {
94          if(flag) sort(p.begin(), p.end());
95          int n = p.size(), s = 0, m=n/2;
96          if(n<=1) return INF;
97          G b(begin(p), begin(p)+m), c(begin(p)+m, end(p)), e;
98          double x = p[m].real(), d=min(closestPair(b, 0), closestPair(c, 0));
99          sort(p.begin(), p.end(), cmp_y);
100         for(int i=0; i<n; ++i) {
101                 if(abs(real(p[i])−x) >= d) continue;
102                 for(int j=0; j<e.size(); ++j) {
103                         if(imag(p[i]−e[e.size()−1−j]) >= d) break;
104                         d = min(d, abs(p[i]−e[e.size()−1−j]));
105                 }
106                 e.push_back(p[i]);
107         }
108         return d;
109 }
```

円

```
1   // circle
2   struct C {
3           P p; double r;
4           C(const P& p, double r) : p(p), r(r) {}
5   };
6
7   int intersectCC(const C& a, const C& b) {
8           double dist = sqrt(norm(a.p−b.p)), r1 = a.r + b.r, r2 = abs(a.r − b.r);
9           if(r1 < dist) return 4;  //外
10          if(dist == r1) return 3; //外接
11          if(r2 < dist && dist < r1) return 2; //2点で交わる
12          if(dist == r2) return 1; //内接
13          return 0; //内包
14  }
15
16  vector<P> crossPointCL(C c, L l) {
17          double d = distanceLP(l, c.p), r = c.r;
18          P m = projection(l, c.p);
19          P x = sqrt(r∗r−d∗d)/abs(l[1]−l[0])∗(l[1]−l[0]);
20          vector<P> ret(2,m);
21          ret[0] −= x;
22          ret[1] += x;
23          sort(ret.begin(), ret.end()); //!!!
24          return ret;
25  }
26
27  vector<P> crossPointCC(C a, C b) {
28          double d = abs(a.p−b.p);
```

```
29          double t = (a.r*a.r-b.r*b.r+d*d)/2/d, h = sqrt(a.r*a.r-t*t);
30          P m = t/abs(b.p-a.p)*(b.p-a.p)+a.p;
31          P n = n_vector(a.p-b.p);
32          vector<P> ret(2, m);
33          ret[0] -= h*n;
34          ret[1] += h*n;
35          sort(ret.begin(), ret.end()); //!!!
36          return ret;
37      }
38
39      //接線
40      vector<P> tangentLC(P p, C c) {
41          C c2 = C((p+c.p)/2.0, abs(p-c.p)/2.0);
42          return crossPointCC(c, c2);
43      }
44
45      //共通接線
46      vector<P> commonTangent(C a, C b) {
47          vector<P> ret, cp;
48          if(a.r == b.r) {
49                  P n = n_vector(b.p-a.p);
50                  ret.push_back(P{a.p+a.r*n});
51                  ret.push_back(P{a.p-a.r*n});
52          }
53          P i = (a.p*b.r+b.p*a.r)/(a.r+b.r);
54          if(abs(a.p-b.p) > a.r+b.r){
55          cp = tangentLC(i,a);
56          for(int i = 0; i < 2; ++i) ret.push_back(cp[i]);
57          if(a.r != b.r){
58                          P e = (a.p*b.r-b.p*a.r)/(b.r-a.r);
59                          cp = tangentLC(e,a);
60                          for(int i = 0; i < 2; ++i) ret.push_back(cp[i]);
61          }
62          }else if(abs(a.p-b.p) == a.r+b.r){
63          cp = tangentLC(i,a);
64          ret.push_back(cp[0]);
65          if(a.r != b.r){
66                  P e = (a.p*b.r-b.p*a.r)/(b.r-a.r);
67                          cp = tangentLC(e,a);
68                          for(int i = 0; i < 2; ++i) ret.push_back(cp[i]);
69          }
70          }else if(abs(a.p-b.p) > abs(a.r-b.r)){
71          if(a.r != b.r){
72                  P e = (a.p*b.r-b.p*a.r)/(b.r-a.r);
73                          cp = tangentLC(e,a);
74                          for(int i = 0; i < 2; ++i) ret.push_back(cp[i]);
75          }
76          }else if(abs(a.p-b.p) == abs(a.r-b.r)){
77          P e = (a.p*b.r-b.p*a.r)/(b.r-a.r);
78          cp = tangentLC(e,a);
79          ret.push_back(cp[0]);
80          }
81          sort(ret.begin(), ret.end());
```

```
82          return ret;
83      }
84
85      /*
86      3点が与えられたときに円を求める
87      返り値は {中心のx座標、y座標、半径}
88      3点が直線上に並んでいるときは {0, 0, −1}を返す
89      */
90      C calcCircle(int x1, int y1, int x2, int y2, int x3, int y3) {
91        long ox, oy, a, b, c, d;
92        long r1, r2, r3;
93
94        a = x2 − x1;
95        b = y2 − y1;
96        c = x3 − x1;
97        d = y3 − y1;
98
99        int cx, cy, r;
100       if ((a && d) || (b && c)) {
101         ox = x1 + (d * (a * a + b * b) − b * (c * c + d * d)) / (a * d − b * c) / 2;
102         if (b) {
103           oy = (a * (x1 + x2 − ox − ox) + b * (y1 + y2)) / b / 2;
104         } else {
105           oy = (c * (x1 + x3 − ox − ox) + d * (y1 + y3)) / d / 2;
106         }
107         r1 = sqrt((ox − x1) * (ox − x1) + (oy − y1) * (oy − y1));
108         r2 = sqrt((ox − x2) * (ox − x2) + (oy − y2) * (oy − y2));
109         r3 = sqrt((ox − x3) * (ox − x3) + (oy − y3) * (oy − y3));
110         cx = ox;
111         cy = oy;
112         r = (r1 + r2 + r3) / 3;
113         return {P{cx, cy}, r};
114       }
115
116       return {P{0, 0}, −1};
117     }
```

```
1     // Binary Indexed Tree max
2     #define MAX_N 100000
3     int n;
4     int bit[MAX_N];
5
6     // a−1の位置にw を更新
7     void add(int a, int w) {
8       for (int x = a; x < n; x |= x + 1) {
9         if (bit[x] < w)
10          bit[x] = w;
11      }
12    }
13
14    // 0 から a−1 までの最大値を求める
15    int maximum(int a) {
16      int ret = −INF;
```

```
17    for (int x = a − 1; x >= 0; x = (x & (x + 1)) − 1)
18      ret = max(ret, bit[x]);
19    return ret;
20  }
```

```
1   bool prime[1000000];
2   memset(prime, true, sizeof(prime));
3   prime[0] = prime[1] = false;
4   for (int i = 2; i ∗ i <= 1000000; i++) {
5         if (prime[i]) {
6               for (int j = 2 ∗ i; j <= 1000000; j += i) {
7                     prime[j] = false;
8               }
9         }
10  }
```

lca

```
1   VI G[MAX_N]; //グラフの隣接リスト
2   int root = 0; //根のノード
3
4   int parent[MAX_LOG_N][MAX_N];
5   int depth[MAX_N];
6
7   void dfs(int v, int p, int d) {
8     parent[0][v] = p;
9     depth[v] = d;
10    REP(i, G[v].size()) if(G[v][i] != p) dfs(G[v][i], v, d+1);
11  }
12
13  //初期化 O(logn)
14  void init(int n) {
15    dfs(root, −1, 0);
16    REP(k, MAX_LOG_N−1) REP(v, n) {
17      if(parent[k][v] < 0) parent[k+1][v] = −1;
18      else parent[k+1][v] = parent[k][parent[k][v]];
19    }
20  }
21
22  // u と v の lca を求める
23  int lca(int u, int v) {
24    if(depth[u] > depth[v]) swap(u, v);
25    REP(k, MAX_LOG_N) {
26      if((depth[v]−depth[u]) >> k & 1) v = parent[k][v];
27    }
28    if(u == v) return u;
29    for(int k = MAX_LOG_N−1; k>=0; k−−) {
30      if(parent[k][u] != parent[k][v]) {
31        u = parent[k][u];
32        v = parent[k][v];
33      }
34    }
35    return parent[0][u];
```

```
}
```

binary pow

```
//二分累乗法 x の e 乗
ll binpow(ll x, ll e) {
  ll a = 1, p = x;
  while(e > 0) {
    if(e%2 == 0) {p = (p*p) % MOD; e /= 2;}
    else {a = (a*p) % MOD; e--;}
  }
  return a % MOD;
}
```

素因数分解

```
// first: 素因数 second: first が何個あるか O(nsqrt(n))
map<ll, ll> v;
ll a = 2;
while(x >= a*a) {
  if(x % a == 0) {
    v[a]++;
    x /= a;
  } else {
    a++;
  }
}
v[x]++;
```

string split

```
// sep で区切る
vector<string> split(const string &str, char sep)
{
  vector<string> v;
  auto first = str.begin();
  while( first != str.end() ) {
    auto last = first;
    while( last != str.end() && *last != sep ) ++last;
    v.push_back(std::string(first, last));
    if( last != str.end() ) ++last;
    first = last;
  }
  return v;
}
```

トポロジカルソート

```
//グラフの隣接リスト
VI g[100010];
//頂点の入次数を管理
int h[100010];
signed main(void)
{
```

```
7     //頂点数 v、辺の数 e
8     int v, e;
9     cin >> v >> e;
10    REP(i, e) {
11      int s, t;
12      cin >> s >> t;
13      //頂点 s から頂点 t への有向辺
14      g[s].push_back(t);
15      h[t]++;
16    }
17
18    //入次数が 0の頂点の集合
19    stack<int> st;
20
21    //入次数が 0の頂点であればst に追加
22    REP(i, v) if(h[i] == 0) st.push(i);
23
24    //ソートされた後のグラフ
25    VI ans;
26    //st がなくなるまでループ
27    while(st.size()) {
28      //st の集合のから一つ取り出す
29      int i = st.top(); st.pop();
30      ans.push_back(i);
31      for(auto& j: g[i]) {
32        //隣接する頂点の入次数をマイナス 1
33        h[j]--;
34        //これによって入次数が 0になればst に追加
35        if(h[j] == 0) st.push(j);
36      }
37    }
38
39    //ans を順に出力
40    for(int i: ans) cout << i << endl;
41
42    return 0;
43  }
```

構文解析

```
1   typedef string::const_iterator State;
2   class ParseError {};
3   int expression(State&);
4   int term(State&);
5   int number(State&);
6   int factor(State&);
7
8   int expression(State &begin) {
9         int ret = term(begin);
10        while(1) {
11                if(*begin == '+') {
12                        begin++;
13                        ret += term(begin);
```

```
14                  } else if(*begin == '-') {
15                          begin++;
16                          ret -= term(begin);
17                  } else {
18                          break;
19                  }
20          }
21          return ret;
22  }
23
24  int term(State &begin) {
25          int ret = factor(begin);
26          while(1) {
27                  if(*begin == '*') {
28                          begin++;
29                          ret *= factor(begin);
30                  } else if(*begin == '/') {
31                          begin++;
32                          ret /= factor(begin);
33                  } else {
34                          break;
35                  }
36          }
37          return ret;
38  }
39
40  int factor(State &begin) {
41          if(*begin == '(') {
42                  begin++;
43                  int ret = expression(begin);
44                  begin++;
45          } else {
46                  return number(begin);
47          }
48  }
49
50  int number(State &begin) {
51          int ret = 0;
52          while(isdigit(*begin)) {
53                  ret *= 10;
54                  ret += *begin - '0';
55                  begin++;
56          }
57          return ret;
58  }
59
60  signed main(void)
61  {
62          string s;
63          getline(cin, s);
64
65          //REP(i, n) REP(j, n)
66          State begin = s.begin();
```

```
67          int ans = expression(begin);
68          cout << ans << endl;
69
70          return 0;
71      }
```

座圧

```
1   int b[100010];
2   map<ll, ll> mp;
3   vector<ll> c(n), d(n);
4   REP(i, n) {
5     mp[b[i]] = 0;
6     d[i] = b[i];
7   }
8   int rank = 0, cnt = 0;
9   for(auto& i: mp) i.second = rank++;
10  for(auto& i: d) {
11    c[cnt] = mp[i];
12    cnt++;
13  }
```

平衡二分探索木

```
1   unsigned xorShift() {
2         static unsigned z = time(NULL);
3         z ^= z << 13; z ^= z >> 17; z ^= z << 5;
4         return z;
5   }
6
7   struct node {
8     int val;
9     node *ch[2];
10    int pri;
11    int cnt; //部分木のサイズ
12    int sum; //部分木の値の和
13
14    node(int v, double p): val(v),pri(p),cnt(1),sum(v) {
15      ch[0] = ch[1] = nullptr;
16    }
17  };
18
19  int count(node *t) {return t == nullptr ? 0: t->cnt;}
20  int sum(node *t) {return t == nullptr ? 0: t->sum;}
21
22  node *update(node *t) {
23    t->cnt = count(t->ch[0]) + count(t->ch[1]) + 1;
24    t->sum = sum(t->ch[0]) + sum(t->ch[1]) + t->val;
25    return t;
26  }
27
28  // b=0で左回転、b=1で右回転
29  node *rotate(node *t, int b) {
```

```
30    node *s = t->ch[1-b];
31    t->ch[1-b] = s->ch[b];
32    s->ch[b] = t;
33    update(t);
34    update(s);
35    return s;
36  }
37
38  node *insert(node *t, int val, int pri) {
39    if(t == nullptr) return new node(val, pri);
40    else if(val == t->val) return t;
41    else if(val < t->val) {
42      t->ch[0] = insert(t->ch[0], val, pri);
43      if(t->pri > t->ch[0]->pri) {
44        t = rotate(t, 1);
45      }
46    } else{
47      t->ch[1] = insert(t->ch[1], val, pri);
48      if(t->pri > t->ch[1]->pri) {
49        t = rotate(t, 0);
50      }
51    }
52    return update(t);
53  }
54
55  node *erase(node *t, int x) {
56    if (t->val == x) {
57      if (t->ch[0] && t->ch[1]) {
58        if (t->ch[0]->pri < t->ch[1]->pri) {
59          t = rotate(t, 1);
60          t->ch[1] = erase(t->ch[1], x);
61          return update(t);
62        } else {
63          t = rotate(t, 0);
64          t->ch[0] = erase(t->ch[0], x);
65          return update(t);
66        }
67      } else {
68        return t->ch[0] ? t->ch[0] : t->ch[1];
69      }
70    } else if (x < t->val) {
71      t->ch[0] = erase(t->ch[0], x);
72    } else {
73      t->ch[1] = erase(t->ch[1], x);
74    }
75    return update(t);
76  }
77
78  int level(node *t, int k) {
79    if(k < count(t->ch[0])) return level(t->ch[0], k);
80    if(k == count(t->ch[0])) return t->val;
81    return level(t->ch[1], k-count(t->ch[0])-1);
82  }
```

サイコロ

```
1   struct Dice{
2     //top, front, right, left, back, bottom
3     int side[6];
4     Dice(){}
5     Dice(int s[]){
6       for(int i=0; i<6; ++i) side[i] = s[i];
7     }
8
9     void rotate(int op){
10      int tmp = '␣';
11      //右に倒す
12      if(op==0){
13        tmp = side[0];
14        side[0] = side[3];
15        side[3] = side[5];
16        side[5] = side[2];
17        side[2] = tmp;
18      }
19
20      //前に倒す
21      if(op==1){
22        tmp = side[0];
23        side[0] = side[4];
24        side[4] = side[5];
25        side[5] = side[1];
26        side[1] = tmp;
27      }
28
29      //左に倒す
30      if(op==2){
31        tmp = side[0];
32        side[0] = side[2];
33        side[2] = side[5];
34        side[5] = side[3];
35        side[3] = tmp;
36      }
37
38      //後ろに倒す
39      if(op==3){
40        tmp = side[0];
41        side[0] = side[1];
42        side[1] = side[5];
43        side[5] = side[4];
44        side[4] = tmp;
45      }
46
47      //top と bottom を軸に右回転
48      if(op==4){
49        tmp = side[1];
50        side[1] = side[2];
```

```
51          side[2] = side[4];
52          side[4] = side[3];
53          side[3] = tmp;
54        }
55
56        //top と bottom を軸に左回転
57        if(op==5){
58          tmp = side[1];
59          side[1] = side[3];
60          side[3] = side[4];
61          side[4] = side[2];
62          side[2] = tmp;
63        }
64      }
65    };
66
67    //24通りのサイコロを生成する
68    Dice initDice, dice[24];
69    void makeDice(){
70      int tmpNum[] = {1,2,3,4,5,6};
71      initDice = Dice(tmpNum);
72
73      for(int i=0; i<24; ++i){
74        if(i==4) initDice.rotate(1);
75        if(i==8) initDice.rotate(1);
76        if(i==12)initDice.rotate(1);
77        if(i==16){
78          initDice.rotate(1);
79          initDice.rotate(0);
80        }
81        if(i==20){
82          initDice.rotate(2);
83          initDice.rotate(2);
84        }
85        initDice.rotate(4);
86        dice[i] = initDice;
87      }
88    }
89
90    // d[top][front] front, right, back, left
91    // 問題のdice の設定とあっているか確認すること！！！！！
92    int d[6][6][4] = {
93      {{−1, −1, −1, −1}, {2, 4, 5, 3}, {3, 2, 4, 5}, {4, 5, 3, 2}, {5, 3, 2, 4}, {−1, −1, −1, −1}}
94      ,{{1, 3, 6, 4}, {−1, −1, −1, −1}, {3, 6, 4, 1}, {4, 1, 3, 6}, {−1, −1, −1, −1}, {6, 4, 1,
            3}}
95      ,{{1, 5, 6, 2}, {2, 1, 5, 6}, {−1,−1,−1,−1}, {−1,−1,−1,−1}, {5, 6, 2, 1}, {6, 2, 1, 5}}
96      ,{{1, 2, 6, 5}, {2, 6, 5, 1}, {−1,−1,−1,−1}, {−1,−1,−1,−1}, {5, 1, 2, 6}, {6, 5, 1, 2}}
97      ,{{1, 4, 6, 3}, {−1,−1,−1,−1}, {3, 1, 4, 6}, {4, 6, 3, 1}, {−1,−1,−1,−1}, {6, 3, 1, 4}}
98      ,{{−1,−1,−1,−1}, {2, 3, 5, 4}, {3, 5, 4, 2}, {4, 2, 3, 5}, {5, 4, 2, 3}, {−1,−1,−1,−1}}};
```

中国人配達問題

```
1   int d[17][17], f[17];
```

```
signed main(void)
{
  int v, e, tot = 0;
  cin >> v >> e;
  REP(i, v) REP(j, v) d[i][j] = INF;
  REP(i, e) {
    int a, b, c;
    cin >> a >> b >> c;
    d[a][b] = min(d[a][b], c); d[b][a] = min(d[b][a], c);
    f[a]++; f[b]++;
    tot += c;
  }
  VI o;
  REP(i, v) if(f[i]%2) o.push_back(i);
  REP(k, v) REP(i, v) REP(j, v) d[i][j] = min(d[i][j], d[i][k] + d[k][j]);
  int dp[1<<16], os = o.size();
  fill(dp, dp+(1<<os), INF);
  dp[0] = 0;
  REP(s, 1<<os) REP(i, os) if(~s>>i&1) REP(j, i) if(~s>>j&1) {
    dp[s|1<<i|1<<j] = min(dp[s|1<<i|1<<j], dp[s] + d[o[i]][o[j]]);
  }
  cout << tot + dp[(1<<os)-1] << endl;
  return 0;
}
```

編集距離

```
int dp[1010][1010];
signed main(void)
{
  string s, t;
  cin >> t >> s;
  const int n = s.size(), m = t.size();
  REP(i, n+1) REP(j, m+1) {
    if(i == 0) dp[i][j] = j;
    else if(j == 0) dp[i][j] = i;
    else {
      if(s[i-1] == t[j-1]) {
        dp[i][j] = min({dp[i-1][j-1], dp[i-1][j]+1, dp[i][j-1]+1});
      } else {
        dp[i][j] = min({dp[i-1][j] + 1, dp[i][j-1] + 1, dp[i-1][j-1]+1});
      }
    }
  }
  cout << dp[n][m] << endl;
  return 0;
}
```

```
int n, h[100010], l[100010], r[100010], st[100010];
signed main(void)
{
  // i 番目の高さが h[i]のヒストグラム中で最大の長方形の面積
  cin >> n;
```

```
    REP(i, n) cin >> h[i];
    int t = 0;
    REP(i, n) {
      while(t>0 && h[st[t-1]] >= h[i]) t--;
      l[i] = t == 0 ? 0 : (st[t-1]+1);
      st[t++] = i;
    }
    t = 0;
    for(int i=n-1; i>=0; --i) {
      while(t > 0 && h[st[t-1]] >= h[i]) t--;
      r[i] = t == 0 ? n : st[t-1];
      st[t++] = i;
    }
    ll ret = 0;
    REP(i, n) ret = max(ret, (ll)h[i]*(r[i]-l[i]));
    cout << ret << endl;
    return 0;
}
```

```
int n, c[1500][1500], dp[1500][1500], l[1500], r[1500], st[1500];
signed main(void)
{
    // h * w の 0,1からなる行列の中で、0のみを使ってできる最大の長方形の面積
    // O(HW)
    int H, W;
    cin >> H >> W;
    REP(i, H) REP(j, W) cin >> c[i][j];

    REP(i, W) {
      int cnt = 1;
      REP(j, H) {
        if(!c[j][i]) {
          dp[j][i] = cnt;
          cnt++;
        } else {
          dp[j][i] = 0;
          cnt = 1;
        }
      }
    }
    ll ret = 0;
    REP(j, H) {
      int t = 0;
      REP(i, W) {
        while(t>0 && dp[j][st[t-1]] >= dp[j][i]) t--;
        l[i] = t == 0 ? 0 : (st[t-1]+1);
        st[t++] = i;
      }
      t = 0;
      for(int i=W-1; i>=0; --i) {
        while(t > 0 && dp[j][st[t-1]] >= dp[j][i]) t--;
        r[i] = t == 0 ? n : st[t-1];
        st[t++] = i;
```

```
35        }
36        REP(i, W) ret = max(ret, (ll)dp[j][i]*(r[i]-l[i]));
37      }
38      cout << ret << endl;
39      return 0;
40    }
```

個数制限付きナップザック

```
1   int dp[10010];
2   int v[105], w[105], m[105];
3   signed main(void)
4   {
5     int n, W;
6     cin >> n >> W;
7     REP(i, n) cin >> v[i] >> w[i] >> m[i];
8     REP(i, n) {
9       int num = m[i];
10      for(int k=1; num > 0; k <<= 1) {
11        int mu = min(k, num);
12        //ダブリング
13        for(int j=W; j>=w[i]*mu; --j) {
14          dp[j] = max(dp[j], dp[j-w[i]*mu]+v[i]*mu);
15        }
16        num -= mu;
17      }
18    }
19    cout << dp[W] << endl;
20    return 0;
21  }
```