Contents

1 Basic

```
se rnu nu ai ts=4 sw=4 mouse=vi incsearch
sy on
ino ( ()<Esc>i
ino ' ''<Esc>i
ino " ""<Esc>i
ino [ []<Esc>i
ino { {}<Esc>i
ino {<CR> {<CR><Tab><CR>}<Esc>hxka
ino jj <Esc>
ino jk <Esc>
ino kk <Esc>
map <F5> :w<CR>:!
    g++ -g -fsanitize=undefined,address  -Wall -Wextra
     -Wshadow %:r.cpp &&echo "Compiled" && ./a.out<CR>
```

1.1 Default [93d400]

```cpp
#include <bits/stdc++.h>
using namespace std;
#define pb emplace_back
#define iter(x) (x).begin(), (x).end()
#define size(x) (int)(x).size()
typedef long long ll;

signed main() {
    cin.tie(0)->sync_with_stdio(0);
}
```

```
setxkbmap -option caps:swapescape #caps to esc
cpp file.cpp -dD -P -fpreprocessed |
    tr -d '[:space:]' | md5sum | cut -c-6 #hash command
```

```bash
#!/usr/bin/bash
for ((i=0;;i++))
do
python gen.py > case.in
./A < case.in > aout
./B < case.in > bout

if ! (cmp -s aout bout);
then
cat case.in
fi

done
```

1.2 Default-extension [7a1c54]

```cpp
#define INF 0x3f3f3f3f
#define lowbit(x) ((x) & -(x))
#define tmin(a, b) (a) = min((a), (b))
#define tmax(a, b) (a) = max((a), (b))
typedef pair<int, int> pii;

void db() { cerr << endl; }
template <class T, class... U>
void db(T a, U... b) { cerr << a << " ", db(b...); }
```

1.3 Pragma [0ca2dd]

```cpp
#pragma GCC optimize("Ofast,no-stack-protector")
#pragma GCC optimize("no-math-errno,unroll-loops")
#pragma GCC target("sse,sse2,sse3,ssse3,sse4")
#pragma GCC target("popcnt,abm,mmx,avx,arch=skylake")
__builtin_ia32_ldmxcsr(__builtin_ia32_stmxcsr()|0x8040)
```

1.4 FastIO [11a75d]

```cpp
inline char gc() {
    const static int SZ = 1 << 16;
    static char buf[SZ], *p1, *p2;
    if (p1 == p2 && (p2 =
        buf + fread(p1 = buf, 1, SZ, stdin), p1 == p2))
        return -1;
    return *p1++;
}
void rd() {}
template <typename T, typename... U>
void rd(T &x, U &...y) {
    x = 0;
    bool f = 0;
    char c = gc();
    while (!isdigit(c))
        f ^= !(c ^ 45), c = gc();
    while (isdigit(c))
        x = (x << 1) + (x << 3) + (c ^ 48), c = gc();
    f && (x = -x), rd(y...);
}

template <typename T>
void prt(T x) {
    if (x < 0)
        putchar('-'), x = -x;
    if (x > 9)
        prt(x / 10);
    putchar((x % 10) ^ 48);
}
```

2 Geometry

2.1 Template [47e2bd]

```cpp
struct P {
#define eps 1e-9
    double x, y;
    P() { x = y = 0; }
    P(double x, double y) : x(x), y(y) {}
    friend bool operator<(const P &a, const P &b)
        { return a.x == b.x ? a.y < b.y : a.x < b.x; }
    friend bool operator<=(const P &a, const P &b) {
        return a.x == b.x ? a.y <= b.y : a.x <= b.x; }
    friend bool operator==(const P &a, const
        P &b) { return a.x == b.x && a.y == b.y; }
    friend bool operator!=(const P &a, const
        P &b) { return a.x != b.x || a.y != b.y; }
    P operator+(const
        P &b) const { return P(x + b.x, y + b.y); }
    void operator+=(const P &b) { x += b.x, y += b.y; }
    P operator-(const
        P &b) const { return P(x - b.x, y - b.y); }
    void operator-=(const P &b) { x -= b.x, y -= b.y; }
    P operator
        *(double b) const { return P(x * b, y * b); }
    void operator*=(double b) { x *= b, y *= b; }
    P operator
        /(double b) const { return P(x / b, y / b); }
    void operator/=(double b) { x /= b, y /= b; }
    double operator*(
        const P &b) const { return x * b.x + y * b.y; }
    double operator^(
        const P &b) const { return x * b.y - y * b.x; }
    double lth() const { return sqrt(x * x + y * y); }
    double lth2() const { return x * x + y * y; }
    inline void print
        () { cout << '(' << x << ' ' << y << ')'; }
    friend istream &operator>>(
        istream &is, P &a) { return is >> a.x >> a.y; }
    friend ostream &operator<<(ostream &os,
        const P &a) { return os << a.x << ' ' << a.y; }
};
int ori(const P &a, const P &b, const P &c) {
    double k = (b - a) ^ (c - a);
    if (fabs(k) < eps)
        return 0;
    return k > 0 ? 1 : -1;
}
inline
     bool within(const P &a, const P &b, const P &c) {
    return (b - a) * (c - a) < eps;
}
bool intersects
    (const P &a, const P &b, const P &c, const P &d) {
    int abc = ori(a, b, c);
    int abd = ori(a, b, d);
    int cda = ori(c, d, a);
    int cdb = ori(c, d, b);
    if (!abc && !abd)
        return within(a, c, d) || within(b, c, d) ||
               within(c, a, b) || within(d, a, b);
    return abc * abd <= 0 && cda * cdb <= 0;
}
P intersection
    (const P &a, const P &b, const P &c, const P &d) {
    double abc = (b - a) ^ (c - a);
    double abd = (b - a) ^ (d - a);
    return (d * abc - c * abd) / (abc - abd);
}
P BASE(0, 0);
inline bool updown(const P &a) {
    if (fabs(a.y) < eps)
        return a.x > eps;
    return a.y > eps;
}
bool cmp(const P &a, const P &b) {
    bool ba = updown(a - BASE), bb = updown(b - BASE);
    if (ba ^ bb)
        return ba;
    return ori(BASE, a, b) > 0;
}
void fdhl(vector<P> &ar, vector<P> &hl) {
    sort(iter(ar));
    for (int i = 0; i < 2; i++) {
        int prln = size(hl);
        for (int j = 0; j < size(ar); j++) {
            while
                (size(hl) - prln > 1 && ori(hl[size(hl
               ) - 1], hl[size(hl) - 2], ar[j]) >= 0)
               hl.pop_back();
            hl.push_back(ar[j]);
        }
        if (size(hl) > 1)
            hl.pop_back();
        reverse(ar.begin(), ar.end());
    }
    if (size(hl) > 1 && hl.front() == hl.back())
        hl.pop_back();
}
bool in(const P &a, const vector<P> &hl) {
    int ln = size(hl);
    if (ln == 1)
        return a == hl[0];
    if (ln == 2) {
        return !ori(hl
            [0], hl[1], a) && within(a, hl[0], hl[1]);
    }
    int l = 1, r = ln - 1, m;
    while (l < r - 1) {
        m = (l + r) >> 1;
        if (ori(hl[0], a, hl[m]) < 0)
            l = m;
        else
            r = m;
    }
    return ori(hl[0], hl[l], a) >= 0 && ori(hl[l
        ], hl[r], a) >= 0 && ori(hl[r], hl[0], a) >= 0;
}
```

2.2 Minimum Enclosing Circle [1f681e]

```
P perp(const P &a) {
    return P(-a.y, a.x);
}
struct Circle {
    P o;
    double r;
    inline bool in(const P
         &a) const { return (a - o).lth() <= r + eps; }
};
Circle getcircle(const P &a, const P &b) {
    return Circle{(a + b) / 2, (a - b).lth() / 2};
}
Circle getcircle(const P &a, const P &b, const P &c) {
    const P p1 = (a + b) / 2, p2 = (a + c) / 2;
    Circle res;
    res.o = intersection
        (p1, p1 + perp(a - b), p2, p2 + perp(a - c));
    res.r = (res.o - a).lth();
    return res;
}
Circle findcircle(vector<P> &ar) {
    int n = size(ar);
    shuffle(iter(ar), mt19937(time(NULL)));
    Circle res = {ar[0], 0};
    for (int i = 0; i < n; i++) {
        if (res.in(ar[i]))
            continue;
        res = {ar[i], 0};
        for (int j = 0; j < i; j++) {
            if (res.in(ar[j]))
                continue;
            res = getcircle(ar[i], ar[j]);
            for (int k = 0; k < j; k++) {
                if (res.in(ar[k]))
                    continue;
                res = getcircle(ar[i], ar[j], ar[k]);
            }
        }
    }
    return res;
}
```

2.3 Convex Hull [61610f]

```
void fdhl(vector<P> &ar, vector<P> &hl) {
    sort(iter(ar));
    for (int i = 0; i < 2; i++) {
        int prln = size(hl);
        for (int j = 0; j < size(ar); j++) {
            while
                (size(hl) - prln > 1 && ori(hl[size(hl
               ) - 1], hl[size(hl) - 2], ar[j]) >= 0)
               hl.pop_back();
            hl.push_back(ar[j]);
        }
        if (size(hl) > 1)
```

```
            hl.pop_back();
        reverse(ar.begin(), ar.end());
    }
    if (size(hl) > 1 && hl.front() == hl.back())
        hl.pop_back();
}
```

2.4 Polar Angle Vector [56401e]

```
P BASE(0, 0), BASEV(1, 0);
inline bool updown(const P &a) {
    int tmp = ori(BASE, BASE + BASEV, a);
    if (!tmp)
        return BASEV * (a - BASE) > 0;
    return tmp > 0;
}
bool cmp(const P &a, const P &b) {
    bool ba = updown(a), bb = updown(b);
    if (ba ^ bb)
        return ba;
    return ori(BASE, a, b) > 0;
}
```

2.5 In Convex Hull [513844]

```
bool in(const P &a, const vector<P> &hl) {
    int ln = size(hl);
    if (ln == 1)
        return a == hl[0];
    if (ln == 2) {
        return !ori(hl
            [0], hl[1], a) && within(a, hl[0], hl[1]);
    }
    int l = 1, r = ln - 1, m;
    while (l < r - 1) {
        m = (l + r) >> 1;
        if (ori(hl[0], a, hl[m]) < 0)
            l = m;
        else
            r = m;
    }
    return ori(hl[0], hl[l], a) >= 0 && ori(hl[l
        ], hl[r], a) >= 0 && ori(hl[r], hl[0], a) >= 0;
}
```

2.6 Intersectsion [c5adb9]

```
inline
     bool within(const P &a, const P &b, const P &c) {
    return (b - a) * (c - a) < eps;
}
bool intersects
    (const P &a, const P &b, const P &c, const P &d) {
    int abc = ori(a, b, c);
    int abd = ori(a, b, d);
    int cda = ori(c, d, a);
    int cdb = ori(c, d, b);
    if (!abc && !abd)
        return within(a, c, d) || within(b, c, d) ||
               within(c, a, b) || within(d, a, b);
    return abc * abd <= 0 && cda * cdb <= 0;
}
P intersection
    (const P &a, const P &b, const P &c, const P &d) {
    double abc = (b - a) ^ (c - a);
    double abd = (b - a) ^ (d - a);
    return (d * abc - c * abd) / (abc - abd);
}
```

2.7 Minkowski [542fed]

```
//need Geometry template
void reorder_polygon(vector<P> & pt){
    int pos = 0;
    for(int i = 1; i < size(pt); i++){
        if(pt[i].y < pt[pos].y || (pt
            [i].y == pt[pos].y && pt[i].x < pt[pos].x))
            pos = i;
    }
    rotate(pt.begin(), pt.begin() + pos, pt.end());
}
vector<P> minkowski(vector<P> A, vector<P> B){
    reorder_polygon(A);
    reorder_polygon(B);
    A.push_back(A[0]);A.push_back(A[1]);
    B.push_back(B[0]);B.push_back(B[1]);
    vector<P> result;
    int i = 0, j = 0;
    while(i < size(A) - 2 || j < size(B) - 2){
        result.push_back(A[i] + B[j]);
        auto cross
            = (A[i + 1] - A[i])^(B[j + 1] - B[j]);
        if(cross >= 0 && i < size(A) - 2)
            ++i;
        if(cross <= 0 && j < size(B) - 2)
            ++j;
    }
    return result;
}
```

2.8 Polar Angle Sort [05fa80]

```
P BASE(0, 0);
inline bool updown(const P &a) {
    if (fabs(a.y) < eps)
        return a.x > eps;
    return a.y > eps;
}
bool cmp(const P &a, const P &b) {
    bool ba = updown(a - BASE), bb = updown(b - BASE);
    if (ba ^ bb)
        return ba;
    return ori(BASE, a, b) > 0;
}
```

2.9 Rotation Sweep Line [9fd768]

```
// int cmp1(const P &a, const P &b) {
//
     bool ba = updown(a - BASE), bb = updown(b - BASE);
//     if (ba ^ bb)
//         return ba - !ba;
//     return ori(BASE, a, b);
// }
// Collinear
void sweep(int n, vector<P> ar) {
    static const int N = 2005;
    static int id[N], po[N];
    static pii lr[N * N];
    int m = 0;
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            if (i != j)
                lr[m++] = pii(i, j);
    sort(lr, lr + m, [&](const pii
         &a, const pii &b) { return cmp(ar[a.first] -
         ar[a.second], ar[b.first] - ar[b.second]); });
    // sort
        (lr, lr + m, [&](const pii &a, const pii &b) {
    //     int tmp = cmp1(ar[a.first
        ] - ar[a.second], ar[b.first] - ar[b.second]);
    //     if (tmp)
    //         return tmp > 0;
    //     if (ar[a.first] != ar[b.first])
    //         return ar[a.first] < ar[b.first];
    //     return ar[a.second] < ar[b.second];
    // });
    iota(id, id + n, 0);
    sort(id, id +
        n, [&](int a, int b) { return ar[a].y == ar[b].
        y ? ar[a].x < ar[b].x : ar[a].y < ar[b].y; });
    for (int i = 0; i < n; i++)
        po[id[i]] = i;
    for (int i = 0; i < m; i++) {
        swap(id[po[lr[i].first]], id[po[lr[i].second]])
            , swap(po[lr[i].first], po[lr[i].second]);
    }
}
```

2.10 Half Plane Intersection [c20faa]

```
struct Line{ P X,Y; };
pair<ll,ll> area_pair(Line a, Line b)
{ return pair<ll,ll>((
    a.Y - a.X)^(b.X - a.X), (a.Y - a.X)^(b.Y - a.X)); }
bool isin(Line l0, Line l1, Line l2) {
  // Check inter(l1, l2) strictly in l0
  auto [a02X, a02Y] = area_pair(l0, l2);
  auto [a12X, a12Y] = area_pair(l1, l2);
  if (a12X - a12Y < 0) a12X *= -1, a12Y *= -1;
  return (__int128)
       a02Y * a12X - (__int128) a02X * a12Y > 0; // C^4
}
/* Having solution, check size > 2 */
/* --^-- Line.X --^-- Line.Y --^-- */
/* add polar angle sort below:*/
/* add polar angle sort above:*/
vector<Line> halfPlaneInter(vector<Line> arr) {
  sort(iter(arr), [&](Line a, Line b) -> int {
    return cmp(a.Y - a.X, b.Y - b.X);
  });
  vector<Line> filtered;
  for (int i = 0; i < size(arr); i++) {
    if (i > 0) {
      auto cross = (arr[
        i].Y - arr[i].X) ^ (arr[i-1].Y - arr[i-1].X);
      if (abs(cross) < eps) {
        if (ori
            (arr[i-1].X, arr[i-1].Y, arr[i].X) <= 0) {
          continue;
        } else {
          filtered.pop_back();
        }
      }
    }
    filtered.pb(arr[i]);
  }
  if (filtered.empty()) return {};
  deque<Line> dq(1, filtered[0]);
  for (int i = 1; i < size(filtered); i++) {
    auto p = filtered[i];
    while (size(dq) >= 2 && !isin(
        p, dq[size(dq) - 2], dq.back())) dq.pop_back();
    while (size(dq)
        >= 2 && !isin(p, dq[0], dq[1])) dq.pop_front();
    dq.pb(p);
  }
  while (size(dq) >=
       3 && !isin(dq[0], dq[size(dq) - 2], dq.back()))
    dq.pop_back();
  while
      (size(dq) >= 3 && !isin(dq.back(), dq[0], dq[1]))
    dq.pop_front();
  if (size(dq) < 3) return {};
  return vector<Line>(iter(dq));
}
```

2.11 Insection Line-Convex [d9d769]

```
int TangentDir(vector<P> &C, P dir) {
  return cyc_tsearch(size(C), [&](int a, int b) {
      return (dir ^ C[a]) > (dir ^ C[b]);
      });
}
#define cmpL(i) ori(a, C[i], b)
pii lineHull(P a, P b, vector<P> &C) {
  int A = TangentDir(C, a - b);
  int B = TangentDir(C, b - a);
  int n = size(C);
  if (cmpL(A) < 0 || cmpL(B) > 0)
    return pii(-1, -1); // no collision
  auto gao = [&](int l, int r) {
    for (int t = l; (l + 1) % n != r; ) {
      int m = ((l + r + (l < r ? 0 : n)) / 2) % n;
      (cmpL(m) == cmpL(t) ? l : r) = m;
    }
    return (l + !cmpL(r)) % n;
  };
  pii res = pii(gao(B, A), gao(A, B)); // (i, j)
  if (res.first == res.second) // touching the corner i
    return pii(res.first, -1);
  if (!cmpL(res.
      first) && !cmpL(res.second)) // along side i, i+1
    switch ((res.first - res.second + n + 1) % n) {
      case 0: return pii(res.first, res.second);
      case 2: return pii(res.second, res.second);
    }
  /* crossing sides (i, i+1) and (j, j+1)
     crossing corner i is treated as side (i, i+1)
     returned in
          the same order as the line hits the convex */
  return res;
} // convex cut: (r, l]
```

2.12 Tagent ConvexHull [c3a3b8]

```
/* The point should be strictly out of hull
   return arbitrary point on the tangent line */
pii get_tangent(vector<P> &C, P p) {
  auto gao = [&](int s) {
    return cyc_tsearch(size(C), [&](int x, int y)
        { return ori(p, C[x], C[y]) == s; });
  };
  return pii(gao(1), gao(-1));
} // return (a, b), ori(p, C[a], C[b]) >= 0
```

3 Graph

3.1 BellmonFord [8c4cd9]

```
struct BellmonFord {
    struct E {
        int u, v;
        ll w;
        int s() { return u < v ? u : -u; }
    };
    int n;
    ll inf = LONG_LONG_MAX;
    vector<ll> d;
    vector<E> e;
    BellmonFord(int _n) : n(_n), e(0) {}
    void add_edge
        (int u, int v, ll w) { e.pb(E{u, v, w}); }
    void go(int s) {
        d.assign(n, inf);
        vector<int> p(n, -1);
        d[s] = 0;
        sort(iter(e
            ), [](E a, E b) { return a.s() < b.s(); });
        int lim = n / 2 + 2;
        for (int i = 0; i < lim; ++i)
            for (auto [u, v, w] : e) {
                if (abs(d[u]) == inf)
                    continue;
                if (d[v] > d[u] + w) {
                    p[v] = u;
                    d[v] = (
                        i < lim - 1 ? d[u] + w : -inf);
                }
            }
        for (int i = 0; i < lim; ++i)
            for (auto [u, v, w] : e)
                if (d[u] == -inf)
                    d[v] = -inf;
    }
};
```

3.2 Block Cut [ac5ea2]

```
struct BCT {
    static const int N = 1e6 + 5; // change
    vector<int> e1[N], e2[N], sk; // e2 is the new tree
    int low[N], dfn[N], ctd = 0, ctn = 0;
    bool vs[N];
    void dfs(int u, int fa) {
        dfn[u] = low[u] = ++ctd;
        sk.pb(u), vs[u] = 1;
        for (int v : e1[u]) {
            if (v == fa)
                continue;
            if (dfn[v]) {
                if (vs[v])
                    tmin(low[u], dfn[v]);
                continue;
            }
            dfs(v, u), tmin(low[u], low[v]);
            if (low[v] >= dfn[u]) {
                e2[u].pb(++ctn);
                int x;
                do {
                    x = sk.back();
                    sk.pop_back();
                    e2[ctn].pb(x);
                } while (x != v);
            }
        }
        vs[u] = 0;
    }
    inline void addedge
        (int x, int y) { e1[x].pb(y), e1[y].pb(x); }
    inline void ini(int n, int rt) {
        for (int i = 1; i <= ctn; i++)
            e1[i].clear(), e2[i].clear();
        ctn = n, ctd = 0;
        memset(dfn + 1, 0, n << 2);
        memset(vs + 1, 0, n);
        sk.clear();
        dfs(rt, -1);
    }
};
```

3.3 Vertex BCC [3faed3]

```
struct VertexBCC { // !simple affect BCCofE
  int n, m, dft, nbcc;
  vector<int> low, dfn, bln, is_ap, st1, st2, BCCofE;
  vector<vector<pii>> G;
  vector<vector<int>> bcc, nG;
  void dfs(int u, int f) {
    int child = 0;
    low[u] = dfn[u] = ++dft, st1.pb(u);
    for (auto [v, id] : G[u])
      if (!dfn[v]) {
        st2.pb(id);
        dfs(v, u), ++child;
        tmin(low[u], low[v]);
        if (dfn[u] <= low[v]) {
          is_ap[u] = 1, bln[u] = bln[v] = nbcc++;
          bcc.pb(vector<int>(1, u)), bcc.back().pb(v);
          for (; st1.back() != v; st1.pop_back())
            bcc[bln[u]].pb
                (st1.back()), bln[st1.back()] = bln[u];
          st1.pop_back();
          while (st2.back() != id)
            BCCofE
                [st2.back()] = bln[u], st2.pop_back();
          BCCofE[id] = bln[u], st2.pop_back();
        }
      }
      else if (v != f) {
        tmin(low[u], dfn[v]);
        if (dfn[v] < dfn[u])
          st2.pb(id);
      }
    if (f == -1 && child < 2)
      is_ap[u] = 0;
    if (f == -1 && child == 0)
      bcc.pb(vector<int>(1, u)), bln[u] = nbcc++;
  }
  VertexBCC
      (int _n) : n(_n), m(0), low(n), bln(n), G(n) {}
  void add_edge(int u, int v) {
    G[u].pb(v, m), G[v].pb(u, m++);
  }
  void slv() {
    is_ap.assign(n, 0), dfn = is_ap, dft = nbcc = 0;
    st1.clear(), st2.clear(), BCCofE.assign(m, -1);
    for (int i = 0; i < n; ++i)
      if (!dfn[i])
        dfs(i, -1);
  }
  void block_cut_tree() {
    int tmp = nbcc;
    for (int i = 0; i < n; ++i)
      if (is_ap[i])
        bln[i] = tmp++;
    nG.assign(tmp, vector<int>(0));
    for (int i = 0; i < nbcc; ++i)
      for (int j : bcc[i])
        if (is_ap[j])
          nG[i].pb(bln[j]), nG[bln[j]].pb(i);
  } // up to 2 * n - 2 nodes!! bln[i] for id
};
```

3.4 Edge BCC [353c57]

```
struct EdgeBCC {
  int n, m, dft, necc;
  vector<int> low, dfn, bln, is_bridge, stk;
  vector<vector<pii>> G;
  void dfs(int u, int f) {
    dfn[u] = low[u] = ++dft, stk.pb(u);
    for (auto [v, id] : G[u])
      if (!dfn[v])
        dfs(v, id), tmin(low[u], low[v]);
      else if (id != f)
        tmin(low[u], dfn[v]);
    if (low[u] == dfn[u]) {
      if (f != -1) is_bridge[f] = 1;
      for (; stk.back() != u; stk.pop_back())
        bln[stk.back()] = necc;
      bln[u] = necc++, stk.pop_back();
    }
  }
  EdgeBCC(int _n): n(_n), m(0), low(n), bln(n), G(n) {}
  void add_edge(int u, int v) {
    G[u].pb(v, m), G[v].pb(u, m++);
  }
  void slv() {
    is_bridge.assign(m, 0), stk.clear();
    dfn.assign(n, 0), dft = necc = 0;
    for (int i = 0; i < n; ++i)
      if (!dfn[i]) dfs(i, -1);
  }
}; // ecc_id(i): bln[i]
```

3.5 Centroid Decomp [314317]

```
struct CentroidDecomposition {
    vector<vector<int> > g;
    vector<int> sub;
    vector<bool> v;
    vector<vector<int>> tree;
    int root;
    void add_edge(int a, int b) {
        g[a].push_back(b);
        g[b].push_back(a);
    }
    CentroidDecomposition(const vector<vector
        <int> > &g_, int isbuild = true) : g(g_) {
        sub.resize(size(g), 0);
        v.resize(size(g), false);
        if (isbuild) build();
    }

    void build() {
        tree.resize(size(g));
        root = build_dfs(0);
    }

    int get_size(int cur, int par) {
        sub[cur] = 1;
        for (auto &dst : g[cur]) {
            if (dst == par || v[dst]) continue;
            sub[cur] += get_size(dst, cur);
        }
        return sub[cur];
    }

    int get_centroid(int cur, int par, int mid) {
        for (auto &dst : g[cur]) {
            if (dst == par || v[dst]) continue;
            if (sub[dst] > mid
                ) return get_centroid(dst, cur, mid);
        }
        return cur;
    }

    int build_dfs(int cur) {
        int centroid = get_centroid
            (cur, -1, get_size(cur, -1) / 2);
        v[centroid] = true;
        for (auto &dst : g[centroid]) {
            if (!v[dst]) {
                int nxt = build_dfs(dst);
                if (centroid != nxt
                    ) tree[centroid].emplace_back(nxt);
            }
        }
        v[centroid] = false;
        return centroid;
    }
};
```

3.6 C3 [e348b4]

```
namespace C3 {
    const int N = 1e5 + 5; // change
    vector<int> eg[N], to[N];
    int pl[N], ln, pd[N], id[N], cnt[N];
    ll ans;
    inline void addedge(int x, int y) {
        eg[x].pb(y), eg[y].pb(x);
    }
    ll blt(int n) {
        priority_queue<pii> pq;
        for (int i = 1; i <= n; i++)
            pq.push(pii(pd[i] = -size(eg[i]), i));
        while (!pq.empty()) {
            pii tp = pq.top();
            pq.pop();
            if (id[tp.second])
                continue;
            pl[++ln] = tp.second;
            id[tp.second] = ln;
            for (int v : eg[tp.second])
                if (!id[v])
                    to[v].pb(tp.second
                        ), pq.push(pii(++pd[v], v));
        }
        ans = 0;
        for (int p = 1; p <= n; p++) {
            for (int u : to[pl[p]])
                cnt[u]++;
            for (int u : to[pl[p]])
                for (int v : to[u])
                    ans += cnt[v];
            for (int u : to[pl[p]])
                cnt[u] = 0;
        }
        return ans;
    }
}
```

3.7 C4 [f966c6]

```
namespace C4 {
    const int N = 1e5 + 5; // change
    vector<int> eg[N], to[N];
    int pl[N], ln, pd[N], id[N], cnt[N];
    ll ans;
    inline void addedge(int x, int y) {
        eg[x].pb(y), eg[y].pb(x);
    }
    ll blt(int n) {
        priority_queue<pii> pq;
        for (int i = 1; i <= n; i++)
            pq.push(pii(pd[i] = -size(eg[i]), i));
        while (!pq.empty()) {
            pii tp = pq.top();
            pq.pop();
            if (id[tp.second])
                continue;
            pl[++ln] = tp.second;
            id[tp.second] = ln;
            for (int v : eg[tp.second])
                if (!id[v])
                    to[v].pb(tp.second
                        ), pq.push(pii(++pd[v], v));
        }
        ans = 0;
        for (int p = 1; p <= n; p++) {
            for (int u : to[pl[p]])
                for (int v : eg[u])
                    if (id[v] < p)
                        ans += cnt[v]++;
            for (int u : to[pl[p]])
                for (int v : eg[u])
                    cnt[v] = 0;
        }
        return ans;
    }
}
```

3.8 Directed MST [de79a2]

```
struct DirectedMST { // O(m+nlgn)
    struct edge {
        int u, v;
        ll w;
    };
    int n;
    vector<edge> e;
    DirectedMST(int _n) : n(_n), e(0) {}
    void add_edge
        (int u, int v, ll w) { e.pb(edge{u, v, w}); }
    vector<int>
        slv(int root) { // 0-based, return idx of edges
        using T = pair<ll, int>;
        using PQ = pair<priority_queue
            <T, vector<T>, greater<T>>, ll>;
        auto push = [](PQ &pq, T v) {
            pq.first.
                emplace(v.first - pq.second, v.second);
        };
        auto top = [](const PQ &pq) -> T {
            auto r = pq.first.top();
            return {r.first + pq.second, r.second};
        };
        auto join = [&push, &top](PQ &a, PQ &b) {
            if (size(a.first) < size(b.first))
                swap(a, b);
            while (!b.first.empty())
                push(a, top(b)), b.first.pop();
        };
        vector<PQ> h(n * 2);
        for (int i = 0; i < size(e); ++i)
            push(h[e[i].v], {e[i].w, i});
        vector<int> a(n *
             2), v(n * 2, -1), pa(n * 2, -1), r(n * 2);
        iota(iter(a), 0);
        auto o = [&](int x) {
            int y;
            for (y = x; a[y] != y; y = a[y])
                ;
            for (int ox = x; x != y; ox = x)
                x = a[x], a[ox] = y;
            return y;
        };
        v[root] = n + 1;
        int pc = n;
        for (int i = 0; i < n; ++i)
            if (v[i] == -1) {
                for (int p = i; v[p] == -1
                     || v[p] == i; p = o(e[r[p]].u)) {
                    if (v[p] == i) {
                        int q = p;
                        p = pc++;
                        do {
                            h[q].second = -
                                h[q].first.top().first;
                            join(h[pa
                                [q] = a[q] = p], h[q]);
                        } while
                            ((q = o(e[r[q]].u)) != p);
                    }
                    v[p] = i;
                    while (!h[p].first.empty() &&
                         o(e[top(h[p]).second].u) == p)
                        h[p].first.pop();
                    r[p] = top(h[p]).second;
                }
            }
        vector<int> ans;
        for (int i = pc - 1; i >= 0; i--)
            if (i != root && v[i] != n) {
                for (int f = e[r[i]].
                    v; f != -1 && v[f] != n; f = pa[f])
                    v[f] = n;
                ans.pb(r[i]);
            }
        return ans;
    }
};
```

3.9 Dominator Tree [602345]

```
struct DOT {
    static const int N = 2e5 + 5; // change
    int dfn
        [N], id[N], dfc, fa[N], idm[N], sdm[N], bst[N];
    vector<int> G[N], rG[N];
    void ini(int n) { // remember to initialize
        for (int i = 1; i <= n; i++)
            G[i].clear(), rG[i].clear();
        fill(dfn, dfn + n + 1, 0);
    }
    inline void addedge
        (int u, int v) { G[u].pb(v), rG[v].pb(u); }
    int f(int x, int lm) {
        if (x <= lm)
            return x;
        int cr = f(fa[x], lm);
        if (sdm[bst[fa[x]]] < sdm[bst[x]])
            bst[x] = bst[fa[x]];
        return fa[x] = cr;
    }
    void dfs(int u) {
        id[dfn[u] = ++dfc] = u;
        for (int v : G[u])
            if (!dfn[v])
                dfs(v), fa[dfn[v]] = dfn[u];
    }
    void tar(vector<int> *eg, int rt) {
        dfc = 0, dfs(rt);
        for (int i = 1; i <= dfc; i++)
            sdm[i] = bst[i] = i;
        for (int i = dfc; i > 1; i--) {
            int u = id[i];
            for (int v : rG[u])
                if ((v = dfn[v]))
                    f(v, i), tmin(sdm[i], sdm[bst[v]]);
            eg[sdm[i]].pb(i), u = fa[i];
            for (int v : eg[u])
                f(v, u), idm[v]
                    = (sdm[bst[v]] == u ? u : bst[v]);
            eg[u].clear();
        }
        for (int i = 2; i <= dfc; i++) {
            if (sdm[i] != idm[i])
                idm[i] = idm[idm[i]];
            eg[id[idm[i]]].pb(id[i]);
        }
    }
};
```

3.10 Eulerian Path [1628e1]

```
struct EulerianPath {
  int n, m;
  vector<vector<pii>> adj;
  vector<int> cur, vst;
  vector<pii> ans;
  EulerianPath
      (int _n) : n(_n), m(0), adj(n, vector<pii>(0)) {}
  void add_edge(int a, int b) { adj[a].pb(b, m++); }
  void dfs(int u) {
    for (; cur[u] < size(adj[u]); ++cur[u]) {
      auto [v, id] = adj[u][cur[u]];
      if (vst[id]) continue;
      vst[id] = 1, dfs(v);
      ans.pb(u, v);
    }
  }
  bool go() {
    vector<int> in(n, 0), out(n, 0);
    for (int i = 0; i < n; ++i) {
      out[i] = size(adj[i]);
      for (auto [j, id] : adj[i])
        ++in[j];
    }
    int s = -1, t = -1, ss = -1;
    for (int i = 0; i < n; ++i) {
      if (out[i]) ss = i;
      if (in[i] == out[i]) continue;
      if (in[i] + 1 == out[i]) {
        if (s != -1) return 0;
        s = i;
        continue;
      }
      if (in[i] - 1 == out[i]) {
        if (t != -1) return 0;
        t = i;
        continue;
      }
      return 0;
    }
    if (s == -1) s = ss;
    cur.assign(n, 0), vst.assign(m, 0), ans.clear();
    dfs(s), reverse(iter(ans));
    return accumulate(iter(vst), 0) == m;
  }
};
```

3.11 Gomory-Hu Tree [ba0b07]

```
struct edge{
  int u,v,w;
};
vector<edge> result;
vector<edge> ed;
void GomoryHu(int N){
  vector<int> par(N,0);
  for(int i=1;i<N;i++){
    FLOW din;
    for(const auto &[u,v,c]:ed){
      din.addedge(u,v,c,c);
    }
    din.s = i;
    din.t = par[i];
    result.push_back({i,par[i],din.getflow()});
    for(int j=i+1;j<N;++j){
      if(par[j]==par[i] && din.dp[j]<INF) par[j]=i;
    }
  }
}
```

3.12 HVL Decomp [5f618c]

```
namespace HLD {
    int in[N], ou[N], dfc, top[N], sz[N], fa[N], ch[N];
    vector<int> eg[N];
    inline void addedge(int u, int v) {
        eg[u].pb(v), eg[v].pb(u);
    }
    inline void ini(int n) { // after using
        for (int i = 1; i <= n; i++)
            eg[i].clear();
        memset(ch + 1, 0, n << 2), dfc = 0;
    }
    void dsz(int u) {
        sz[u] = 1;
        for (int v : eg[u]) {
            if (v == fa[u])
                continue;
            fa[v] = u, dsz(v), sz[u] += sz[v];
            if (sz[v] > sz[ch[u]])
                ch[u] = v;
        }
    }
    void dfs(int u) {
        in[u] = ++dfc;
        if (ch[u])
            top[ch[u]] = top[u], dfs(ch[u]);
        for (int v : eg[u])
            if (v != fa[u] && v != ch[u])
                top[v] = v, dfs(v);
        ou[u] = dfc;
    }
    inline void blt(int rt) {
        fa[rt] = 0, dsz(rt), top[rt] = rt, dfs(rt);
    }
    void slv(int u, int v) {
        while (top[u] != top[v]) {
            if (in[top[u]] < in[top[v]])
                swap(u, v);
            // [ in[top[u]], in[u] ];
            u = fa[top[u]];
        }
        if (in[u] > in[v])
            swap(u, v);
        // [ in[u], in[v] ];
    }
}
```

3.13 Hopcroft-Karp [f6d745]

```
struct HK {
    static const
         int N = 1e5 + 5, M = N * 2; // change, 1-base
    int fr[N], pr[N], hd[N], m1
        [N], m2[N], to[M], nxt[M], ct = 1, n1, n2, ans;
    inline void ini(int _n1, int _n2) {
        n1 = _n1, n2 = _n2, ct = 1;
        memset(m1 + 1, 0, n1 << 2);
        memset(m2 + 1, 0, n2 << 2);
        memset(hd + 1, 0, n1 << 2);
    }
    inline void addedge(int u, int v) {
        to[ct] = v, nxt[ct] = hd[u], hd[u] = ct++;
    }
    int getmatch() {
        ans = 0;
        for (bool ok = 1; ok;) {
            ok = 0, memset(fr + 1, 0, n1 << 2);
            queue<int> q;
            for (int i = 1; i <= n1; i++)
                if (!m1[i])
                    q.push(fr[i] = i);
            while (!q.empty()) {
                int u = q.front();
                q.pop();
                if (m1[fr[u]])
                    continue;
                for (int i = hd[u]; i; i = nxt[i]) {
                    int v = to[i];
                    if (!m2[v]) {
                        while (v)
                            m2[v] = u, swap
                                (m1[u], v), u = pr[u];
                        ok = 1, ans++;
                        break;
                    } else if (!fr[m2[v]])
                        q.push(v = m2[v])
                            , fr[v] = fr[u], pr[v] = u;
                }
            }
        }
        return ans;
    }
};
```

3.14 KM [6aefd7]

```
struct KM {
    static const int N = 505;
    int pr[N], eg[N][N], m1[N], m2[N], n;
    ll lx[N], ly[N], sl[N];
    bool vy[N];
    ll ans;
    inline void ini(int _n) {
        n = _n;
        memset(lx + 1, ~INF, n << 3);
        memset(ly + 1, 0, n << 3);
        memset(m1 + 1, 0, n << 2);
        memset(m2 + 1, 0, n << 2);
        for (int i = 1; i <= n; i++)
            memset(eg[i] + 1, ~INF, n << 2);
    }
    inline void addedge(int u, int v, int w) {
        tmax(eg[u][v], w), tmax(lx[u], (ll)w);
    }
    ll getmatch() {
        ans = 0;
        for (int p = 1, u, v; p <= n; p++) {
            memset(sl + 1, INF, n << 3);
            memset(vy + 1, 0, n);
            for (u = p;; u = m2[v], vy[v] = 1) {
                ll mn = 1ll << 60;
                for (int i = 1; i <= n; i++) {
                    if (vy[i])
                        continue;
                    if (eg[u][i] != ~INF && lx
                        [u] + ly[i] - eg[u][i] < sl[i])
                        sl[i] = lx[u] + ly
                            [i] - eg[u][i], pr[i] = u;
                    if (sl[i] < mn)
                        mn = sl[i], v = i;
                }
                for (int i = 1; i <= n; i++) {
                    if (vy[i])
                        ly[i] += mn, lx[m2[i]] -= mn;
                    else
                        sl[i] -= mn;
                }
                lx[p] -= mn;
                if (!m2[v])
                    break;
            }
            while (v)
                u = pr[v], m2[v] = u, swap(v, m1[u]);
        }
        for (int i = 1; i <= n; i++)
            ans += eg[i][m1[i]];
        return ans;
    }
};
```

3.15 Matching [becd87]

```
struct Matching {
    static const int maxn
         = 505, p = (int)1e9 + 7;// change this, 1-base
    int sizen = 0;
    int sub_n=0;
    int id[maxn], vertices[maxn], matches[maxn];
    bool row_marked
        [maxn] = {false}, col_marked[maxn] = {false};
    int A[maxn][maxn], B[maxn][maxn], t[maxn][maxn];
    vector<pair<int,int> > sidearr;
    void init(int _n) {
        sizen = _n;
        sub_n = 0;
        fill(id,id+_n+1,0);
        fill(vertices,vertices+_n+1,0);
        fill(matches,matches+_n+1,0);
        fill(row_marked,row_marked+_n+1,0);
        fill(col_marked,col_marked+_n+1,0);
        for(int i=0; i<=_n; i++) {
            fill(A[i],A[i]+_n+1,0);
            fill(B[i],B[i]+_n+1,0);
            fill(t[i],t[i]+_n+1,0);
        }
        sidearr.clear();
    }
    Matching(int _n) {
        init(_n);
    }
    int qpow(int a, int b) {
        int ans = 1;
        while (b) {
            if (b & 1) ans = (long long)ans * a % p;
            a = (long long)a * a % p;
            b >>= 1;
        }
        return ans;
    }
    void Gauss(int A[][maxn], int B[][maxn], int n) {
        if (B) {
            memset(B, 0, sizeof(t));
            for (int i = 1; i <= n; i++) B[i][i] = 1;
        }
        for (int i = 1; i <= n; i++) {
            if (!A[i][i]) {
                for (int j = i + 1; j <= n; j++) {
                    if (A[j][i]) {
                        swap(id[i], id[j]);
                        for (int k = i; k <= n; k
                            ++) swap(A[i][k], A[j][k]);
                        if (B
                            ) for (int k = 1; k <= n; k
                            ++) swap(B[i][k], B[j][k]);
                        break;
                    }
                }
                if (!A[i][i]) continue;
            }
            int inv = qpow(A[i][i], p - 2);
            for (int j = 1; j <= n; j++) {
                if (i != j && A[j][i]) {
                    int t
                        = (long long)A[j][i] * inv % p;
                    for (int k = i; k <= n
                        ; k++) if (A[i][k]) A[j][k] = (
                        A[j][k] - (ll)t * A[i][k]) % p;
                    if (B) {
                        for (int k = 1; k <= n; k++) if
                             (B[i][k]) B[j][k] = (B[j][
                                    k] - (ll)t * B[i][k]) % p;
                    }
                }
            }
        }
        if (B) {
            for (int i = 1; i <= n; i++) {
                int inv = qpow(A[i][i], p - 2);
                for (int j = 1; j <= n; j++) {
                    if (B[i][j]) B[i][j]
                        = (long long)B[i][j] * inv % p;
                }
            }
        }
    }
    void eliminate(int r, int c) {
        row_marked[r] = col_marked[c] = true;
        int inv = qpow(B[r][c], p - 2);
        for (int i = 1; i <= sub_n; i++) {
            if (!row_marked[i] && B[i][c]) {
                int t = (long long)B[i][c] * inv % p;

                for (int j = 1; j <= sub_n; j++)
                    if (!col_marked[j] && B[r][j])
                        B[i][j] = (B[i][j] - (
                            long long)t * B[r][j]) % p;
            }
        }
    }
    void add_edge(int a,int b) {
        sidearr.pb(min(a,b),max(a,b));
    }
    void build_matching() {
        auto rng = mt19937(chrono::steady_clock
            ::now().time_since_epoch().count());
        for(auto e : sidearr) {
            int x = e.first;
            int y = e.second;
            A[x][y] = rng() % p;
            A[y][x] = -A[x][y];
        }
        for (int i = 1; i <= sizen; i++) id[i] = i;
        memcpy(t, A, sizeof(t));
        Gauss(A, nullptr, sizen);
        for (int i = 1; i <= sizen; i++) {
            if (A[id[i]][id[i]]) vertices[++sub_n] = i;
        }
        for (int i = 1; i <= sub_n; i++) {
            for (int j = 1; j <= sub_n; j++)
                A[i][j] = t[vertices[i]][vertices[j]];
        }
        Gauss(A, B, sub_n);
        for (int i = 1; i <= sub_n; i++) {
            if (!matches[vertices[i]]) {
                for (int j = i + 1; j <= sub_n; j++) {
                    if (!matches
                        [vertices[j]] && t[vertices
                        [i]][vertices[j]] && B[j][i]) {
                        matches[
                            vertices[i]] = vertices[j];
                        matches[
                            vertices[j]] = vertices[i];
                        eliminate(i, j);
                        eliminate(j, i);
                        break;
                    }
                }
            }
        }
    }
    int matched(int x) {
        return matches[x];
    }
};
```

3.16 Max Clique [2b1496]

```
struct MaxClique { // fast
     when N <= 100, 0-base,output sol[N] to get node;
  static const int N = 105;
  bitset<N> G[N], cs[N];
  int ans, sol[N], q, cur[N], d[N], n;
  MaxClique(int _n) {
    n = _n;
    for (int i = 0; i < n; ++i) G[i].reset();
  }
  void add_edge(int u, int v) {
    G[u][v] = G[v][u] = 1;
  }
  void pre_dfs(vector<int> &r, int l, bitset<N> mask) {
    if (l < 4) {
      for (int i : r) d[i] = (G[i] & mask).count();
      sort(r.begin(),r.end()
          , [&](int x, int y) { return d[x] > d[y]; });
    }
    vector<int> c(size(r));
    int lft = max(ans - q + 1, 1), rgt = 1, tp = 0;
    cs[1].reset(), cs[2].reset();
    for (int p : r) {
      int k = 1;
      while ((cs[k] & G[p]).any()) ++k;
      if (k > rgt) cs[++rgt + 1].reset();
      cs[k][p] = 1;
      if (k < lft) r[tp++] = p;
    }
    for (int k = lft; k <= rgt; ++k)
      for (int p = cs[k]._Find_first
          (); p < N; p = cs[k]._Find_next(p))
        r[tp] = p, c[tp] = k, ++tp;
    dfs(r, c, l + 1, mask);
  }
  void dfs(vector<
      int> &r, vector<int> &c, int l, bitset<N> mask) {
    while (!r.empty()) {
      int p = r.back();
      r.pop_back(), mask[p] = 0;
      if (q + c.back() <= ans) return;
      cur[q++] = p;
      vector<int> nr;
      for (int i : r) if (G[p][i]) nr.pb(i);
      if (!nr.empty()) pre_dfs(nr, l, mask & G[p]);
      else if (q > ans) ans = q, copy_n(cur, q, sol);
      c.pop_back(), --q;
    }
  }
  int solve() {
    vector<int> r(n);
    ans = q = 0, iota(r.begin(),r.end(), 0);
    pre_dfs(r, 0, bitset<N>(string(n, '1')));
    return ans;
  }
};
```

3.17 Max Flow [b0a4f0]

```
struct FLOW {
    static const int N = 1e3 + 5, M = N * 10; // change
    int dp[N
        ], cr[N], hd[N], ct = 2, s = 0, t = 1, n, flow;
  FLOW(int _n):n(_n){}
    struct E {
        int to, cap, nxt;
    } eg[M];
    inline void addedge(int u, int v, int w,int d=0) {
        eg[ct] = {v, w, hd[u]};
        hd[u] = ct++;
        eg[ct] = {u, d, hd[v]};
        hd[v] = ct++;
    }
    bool bfs() {
        memset(dp, INF, (n + 1) << 2);
        queue<int> q;
        q.push(s), dp[s] = 0;
        while (!q.empty()) {
            int u = q.front();
            q.pop();
            for (int i = hd[u]; i; i = eg[i].nxt) {
                const int v = eg[i].to;
                if (!eg[i].cap || dp[u] + 1 >= dp[v])
                    continue;
                dp[v] = dp[u] + 1, q.push(v);
            }
        }
        return dp[t] != INF;
    }
    int dfs(int u, int fl) {
        if (u == t)
            return fl;
        int sm = 0;
        for (int &i = cr[u]; i; i = eg[i].nxt) {
            int v = eg[i].to, &w = eg[i].cap;
            if (!w || dp[u] + 1 != dp[v])
                continue;
            int tp = dfs(v, min(w, fl - sm));
            w -= tp, sm += tp, eg[i ^ 1].cap += tp;
            if (sm == fl)
                return fl;
        }
        return sm;
    }
    int getflow() {
        flow = 0;
        while (bfs())
            memcpy(cr,
                hd, (n + 1) << 2), flow += dfs(s, INF);
        return flow;
    }
};
```

3.18 Flow With Demand [160a76]

```
struct DMFLOW {
    struct FLOW {
        static const
             int N = 305, M = N * N * 12; // change
        int dp[N], cr
            [N], hd[N], ct = 2, s = 0, t = 1, n, flow;
        inline void ini(int _n) { n = _n; }
        struct E {
            int to, cap, nxt;
        } eg[M];
        inline void addedge(int u, int v, int w) {
            eg[ct] = {v, w, hd[u]};
            hd[u] = ct++;
            eg[ct] = {u, 0, hd[v]};
            hd[v] = ct++;
        }
        bool bfs() {
            memset(dp, INF, (n + 1) << 2);
            queue<int> q;
            q.push(s), dp[s] = 0;
            while (!q.empty()) {
                int u = q.front();
                q.pop();
                for (int i = hd[u]; i; i = eg[i].nxt) {
                    const int v = eg[i].to;
                    if (!eg
                        [i].cap || dp[u] + 1 >= dp[v])
                        continue;
                    dp[v] = dp[u] + 1, q.push(v);
                }
            }
            return dp[t] != INF;
        }
        int dfs(int u, int fl) {
            if (u == t)
                return fl;
            int sm = 0;
            for (int &i = cr[u]; i; i = eg[i].nxt) {
                int v = eg[i].to, &w = eg[i].cap;
                if (!w || dp[u] + 1 != dp[v])
                    continue;
                int tp = dfs(v, min(w, fl - sm));
                w -= tp, sm += tp, eg[i ^ 1].cap += tp;
                if (sm == fl)
                    return fl;
            }
            return sm;
        }
        int getflow() {
            flow = 0;
            while (bfs())
                memcpy(cr, hd,
                    (n + 1) << 2), flow += dfs(s, INF);
            return flow;
        }
    } fw;
    int n, sum, s, t;
    inline void ini(int _n) {
        n = _n, fw.ini(fw.t = n + 1), sum = fw.s = 0; }
    inline void addlr(int u, int v, int l, int r) {
        if (l) {
            fw.addedge(u, fw.t, l);
            fw.addedge(fw.s, v, l);
            sum += l;
        }
        if (r - l)
            fw.addedge(u, v, r - l);
    }
    int slv() {
        fw.addedge(t, s, INF);
        int res = fw.getflow();
        if (res != sum)
            return -1;
        fw.s = s, fw.t = t;
        res = fw.getflow();
        // maximum
        // fw.s = t, fw.t = s;
        // res = INF - fw.getflow();
        // minimum
        return sum = res;
    }
} dmfw;
```

3.19 MCMF [5190af]

```
struct MCMF {
    static const int N = 5005, M = N * 20;
    struct E {
        int to, cap, co, nxt;
    } eg[M];
    int dp[N], hd[
        N], cr[N], ct = 2, s = 0, t = 1, n, flow, cost;
    bool vd[N];
    inline void ini(int _n) { n = _n; }
    inline void addedge(int u, int v, int w, int c) {
        eg[ct] = {v, w, c, hd[u]};
        hd[u] = ct++;
        eg[ct] = {u, 0, -c, hd[v]};
        hd[v] = ct++;
    }
    bool spfa() {
        queue<int> q;
        memset(dp, INF, (n + 1) << 2);
        q.push(s), vd[s] = 1, dp[s] = 0;
        while (!q.empty()) {
            int u = q.front();
            q.pop(), vd[u] = 0;
            for (int i = hd[u]; i; i = eg[i].nxt) {
                const int v = eg[i].to, c = eg[i].co;
                if (!eg[i].cap || dp[u] + c >= dp[v])
                    continue;
                dp[v] = dp[u] + c;
                if (!vd[v])
                    vd[v] = 1, q.push(v);
            }
        }
        return dp[t] != INF;
    }
    int dfs(int u, int fl) {
        if (u == t)
            return fl;
        int sm = 0;
        vd[u] = 1;
        for (int &i = cr[u]; i; i = eg[i].nxt) {
            int &w =
                 eg[i].cap, v = eg[i].to, c = eg[i].co;
            if (!w || vd[v] || dp[u] + c != dp[v])
                continue;
            int tp = dfs(v, min(fl - sm, w));
            w -= tp, eg[i ^
                1].cap += tp, sm += tp, cost += tp * c;
        }
        vd[u] = 0;
        return sm;
    }
    pii getflow() {
        flow = cost = 0;
        while (spfa())
            memcpy(cr,
                hd, (n + 1) << 2), flow += dfs(s, INF);
        return pii(flow, cost);
    }
};
```

3.20 MCMF With Demand [4c3d71]

```
struct DMMCMF {
    struct MCMF {
        static const int N = 315, M = N * N;
        struct E {
            int to, cap, co, nxt;
        } eg[M];
        int dp[N], hd[N],
            cr[N], ct = 2, s = 0, t = 1, n, flow, cost;
        bool vd[N];
        inline void ini(int _n) { n = _n; }
        inline
             void addedge(int u, int v, int w, int c) {
            eg[ct] = {v, w, c, hd[u]};
            hd[u] = ct++;
            eg[ct] = {u, 0, -c, hd[v]};
            hd[v] = ct++;
        }
        bool spfa() {
            queue<int> q;
            memset(dp, INF, (n + 1) << 2);
            q.push(s), vd[s] = 1, dp[s] = 0;
            while (!q.empty()) {
                int u = q.front();
                q.pop(), vd[u] = 0;
                for (int i = hd[u]; i; i = eg[i].nxt) {
                    const
                        int v = eg[i].to, c = eg[i].co;
                    if (!eg
                        [i].cap || dp[u] + c >= dp[v])
                        continue;
                    dp[v] = dp[u] + c;
                    if (!vd[v])
                        vd[v] = 1, q.push(v);
                }
            }
            return dp[t] != INF;
        }
        int dfs(int u, int fl) {
            if (u == t)
                return fl;
            int sm = 0;
            vd[u] = 1;
            for (int &i = cr[u]; i; i = eg[i].nxt) {
                int &w = eg[
                    i].cap, v = eg[i].to, c = eg[i].co;
                if (!w || vd[v] || dp[u] + c != dp[v])
                    continue;
                int tp = dfs(v, min(fl - sm, w));
                w -= tp, eg[i ^ 1].cap
                     += tp, sm += tp, cost += tp * c;
            }
            vd[u] = 0;
            return sm;
        }
        pii getflow() {
            flow = cost = 0;
            while (spfa())
                memcpy(cr, hd,
                    (n + 1) << 2), flow += dfs(s, INF);
            return pii(flow, cost);
        }
    } fw;
    int n, sfl, sco, s, t;
    void ini(int _n) { n = _n
        , fw.ini(fw.t = n + 1), fw.s = sfl = sco = 0; }
    inline
        void addlr(int u, int v, int l, int r, int c) {
        if (l) {
            fw.addedge(u, fw.t, l, 0);
            fw.addedge(fw.s, v, l, 0);
            sco += l * c, sfl += l;
        }
        if (r - l)
            fw.addedge(u, v, r - l, c);
    }
    int slv() {
        fw.addedge(t, s, INF, 0);
        if (sfl != fw.getflow().first)
            return -1;
        return sco += fw.cost;
    }
} dmfw;
```

3.21 SCC [ff24f9]

```
struct SCC {
  int n, dft, nscc;
  vector<int> low, dfn, bln, instk, stk;
  vector<vector<int>> G;
  void dfs(int u) {
    low[u] = dfn[u] = ++dft;
    instk[u] = 1, stk.pb(u);
    for (int v : G[u])
      if (!dfn[v])
        dfs(v), tmin(low[u], low[v]);
      else if (instk[v] && dfn[v] < dfn[u])
        tmin(low[u], dfn[v]);
    if (low[u] == dfn[u]) {
      for (; stk.back() != u; stk.pop_back())
        bln[stk.back()] = nscc, instk[stk.back()] = 0;
      instk[u] = 0, bln[u] = nscc++, stk.pop_back();
    }
  }
  SCC(int _n): n(_n), low(n), bln(n), G(n) {}
  void add_edge(int u, int v) { G[u].pb(v); }
  void slv() {
    dfn.assign(n, 0), instk = dfn, dft = nscc = 0;
    for (int i = 0; i < n; ++i)
      if (!dfn[i]) dfs(i);
  }
}; // scc_id(i): bln[i]
```

3.22 SPFA [0ad2fd]

```
struct SPFA {
    int n;
    vector<ll> d;
    vector<vector<pair<int, ll>>> adj;
    SPFA(int _n
        ) : n(_n), adj(_n, vector<pair<int, ll>>(0)) {}
    void add_edge
        (int u, int v, ll w) { adj[u].pb(v, w); }
    bool go(int s) {
        vector<int> cnt(n, 0), inq(n, 0);
        queue<int> q;
        d.assign(n, LONG_LONG_MAX);
        d[s] = 0, inq[s] = 1, q.emplace(s);
        while (!q.empty()) {
            int u = q.front();
            q.pop(), inq[u] = 0;
            for (auto [v, w] : adj[u]) {
                if (d[v] > d[u] + w) {
                    if (++cnt[v] >= n)
                        return 0; // negative cycle
                    d[v] = d[u] + w;
                    if (!inq[v])
                        q.emplace(v), inq[v] = 1;
                }
            }
        }
        return 1;
    }
};
```

3.23 2-SAT [710518]

```
struct TwoSAT { // SCC needed
  int n;
  vector<bool> istrue;
  SCC scc;
  TwoSAT(int _n) : n(_n), scc(n + n) { }
  int rv(int a) { return a >= n ? a - n : a + n; }
  void add_clause(int a, int b) {
    scc.add_edge(rv(a), b);
    scc.add_edge(rv(b), a);
  }
  bool slv() {
    scc.slv(), istrue.assign(n + n, 0);
    for (int i = 0; i < n; ++i) {
      if (scc.bln[i] == scc.bln[i + n]) return 0;
      istrue[i] = scc.bln[i] < scc.bln[i + n];
      istrue[i + n] = !istrue[i];
    }
    return 1;
  }
};
```

4 Math

4.1 Big Interger [4347d6]

```
#undef size
template<typename T>
inline string to_string(const T& x){
  stringstream ss;
  return ss<<x,ss.str();
}
struct bigN:vector<ll>{
  const static int base=1000000000,width=log10(base);
  bool negative;
  bigN(const_iterator
       a,const_iterator b):vector<ll>(a,b){}
  bigN(string s){
    if(s.empty())return;
    if(s[0]=='-')negative=1,s=s.substr(1);
    else negative=0;
    for(int i=int(s.size())-1;i>=0;i-=width){
      ll t=0;
      for(int j=max(0,i-width+1);j<=i;++j)
        t=t*10+s[j]-'0';
      push_back(t);
    }
    trim();
  }
  template<typename T>
    bigN(const T &x):bigN(to_string(x)){}
  bigN():negative(0){}
  void trim(){
    while(size()&&!back())pop_back();
    if(empty())negative=0;
  }
  void carry(int _base=base){
    for(size_t i=0;i<size();++i){
      if(at(i)>=0&&at(i)<_base)continue;
      if(i+1u==size())push_back(0);
      int r=at(i)%_base;
      if(r<0)r+=_base;
      at(i+1)+=(at(i)-r)/_base,at(i)=r;
    }
  }
  int abscmp(const bigN &b)const{
    if(size()>b.size())return 1;
    if(size()<b.size())return -1;
    for(int i=int(size())-1;i>=0;--i){
      if(at(i)>b[i])return 1;
      if(at(i)<b[i])return -1;
    }
    return 0;
  }
  int cmp(const bigN &b)const{
    if(negative!=b.negative)return negative?-1:1;
    return negative?-abscmp(b):abscmp(b);
  }
  bool operator<(const bigN&b)const{return cmp(b)<0;}
  bool operator>(const bigN&b)const{return cmp(b)>0;}
  bool operator<=(const bigN&b)const{return cmp(b)<=0;}
  bool operator>=(const bigN&b)const{return cmp(b)>=0;}
  bool operator==(const bigN&b)const{return !cmp(b);}
  bool operator!=(const bigN&b)const{return cmp(b)!=0;}
  bigN abs()const{
    bigN res=*this;
    return res.negative=0, res;
  }
  bigN operator-()const{
    bigN res=*this;
    return res.negative=!negative,res.trim(),res;
  }
  bigN operator+(const bigN &b)const{
    if(negative)return -(-(*this)+(-b));
    if(b.negative)return *this-(-b);
    bigN res=*this;
    if(b.size()>size())res.resize(b.size());
    for(size_t i=0;i<b.size();++i)res[i]+=b[i];
    return res.carry(),res.trim(),res;
  }
  bigN operator-(const bigN &b)const{
    if(negative)return -(-(*this)-(-b));
    if(b.negative)return *this+(-b);
    if(abscmp(b)<0)return -(b-(*this));
    bigN res=*this;
    if(b.size()>size())res.resize(b.size());
    for(size_t i=0;i<b.size();++i)res[i]-=b[i];
    return res.carry(),res.trim(),res;
  }
  bigN operator*(const bigN &b)const{
    bigN res;
    res.negative=negative!=b.negative;
    res.resize(size()+b.size());
    for(size_t i=0;i<size();++i)
      for(size_t j=0;j<b.size();++j)
        if((res[i+j]+=at(i)*b[j])>=base){
          res[i+j+1]+=res[i+j]/base;
          res[i+j]%=base;
        }
    return res.trim(),res;
  }
  bigN operator/(const bigN &b)const{
    int norm=base/(b.back()+1);
    bigN x=abs()*norm;
    bigN y=b.abs()*norm;
    bigN q,r;
    q.resize(x.size());
    for(int i=int(x.size())-1;i>=0;--i){
      r=r*base+x[i];
      int s1=r.size()<=y.size()?0:r[y.size()];
      int s2=r.size()<y.size()?0:r[y.size()-1];
      int d=(ll(base)*s1+s2)/y.back();
      r=r-y*d;
      while(r.negative)r=r+y,--d;
      q[i]=d;
    }
    q.negative=negative!=b.negative;
    return q.trim(),q;
  }
  bigN operator%(const bigN &b)const{
    return *this-(*this/b)*b;
  }
  friend istream& operator>>(istream &ss,bigN &b){
    string s;
    return ss>>s, b=s, ss;
  }
  friend
       ostream& operator<<(ostream &ss,const bigN &b){
    if(b.negative)ss<<'-';
    ss<<(b.empty()?0:b.back());
    for(int i=int(b.size())-2;i>=0;--i)
      ss<<setw(width)<<setfill('0')<<b[i];
    return ss;
  }
  template<typename T>
    operator T(){
      stringstream ss;
      ss<<*this;
      T res;
      return ss>>res,res;
    }
};
#define size(x) (int)(x).size()
```

4.2 Mod Int [c34f76]

```
template <unsigned P>
struct mint { // P not prime break /=
    unsigned v;
     mint(ll v = 0) : v((v%P+P) % P) {}
     mint &operator+=(mint const &o) {
        v = (v += o.v) >= P ? v - P : v;
        return *this;
    }
     mint &operator-=(mint const &o) {
        v = (v < o.v) ? v + P - o.v : v - o.v;
        return *this;
    }
     mint &operator*=(mint const &o) {
        v = 1ll * v * o.v % P;
        return *this;
    }
     friend
         mint operator+(mint const &a, mint const &b) {
        return mint(a) += b;
    }
     friend
         mint operator-(mint const &a, mint const &b) {
        return mint(a) -= b;
    }
     friend
         mint operator*(mint const &a, mint const &b) {
        return mint(a) *= b;
    }
     inline mint pow(ll n) const {
        mint r(1);
        mint a = v;
        for (; n; a *= a, n >>= 1)
            r *= (n & 1) ? (a) : (mint(1));
        return r;
    }
     mint &operator/=(mint const &o) {
        *this *= o.pow(P - 2);
        return *this;
    }
     friend
         mint operator/(mint const &a, mint const &b) {
        return mint(a) /= b;
    }
     friend ostream
          &operator<<(ostream &os, mint const &m) {
```

```
        return os << m.v;
    }
     friend istream &operator>>(istream &is, mint &m) {
        return is >> m.v;
    }
};
```

4.3 Fraction [9c92bf]

```
struct frac {
    ll n, d;
    frac(const
         ll &_n = 0, const ll &_d = 1) : n(_n), d(_d) {
        ll t = __gcd(n, d);
        n /= t, d /= t;
        if (d < 0)
            n = -n, d = -d;
    }
    frac operator-() const {
        return frac(-n, d);
    }
    frac operator+(const frac &b) const {
        return frac(n * b.d + b.n * d, d * b.d);
    }
    void operator+=(const frac &b) {
        *this = frac(n * b.d + b.n * d, d * b.d);
    }
    frac operator-(const frac &b) const {
        return frac(n * b.d - b.n * d, d * b.d);
    }
    void operator-=(const frac &b) {
        *this = frac(n * b.d - b.n * d, d * b.d);
    }
    frac operator*(const frac &b) const {
        return frac(n * b.n, d * b.d);
    }
    void operator*=(const frac &b) {
        *this = frac(n * b.n, d * b.d);
    }
    frac operator/(const frac &b) const {
        return frac(n * b.d, d * b.n);
    }
    void operator/=(const frac &b) {
        *this = frac(n * b.d, d * b.n);
    }
    friend ostream
         &operator<<(ostream &os, frac const &f) {
        if (f.d == 1)
            return os << f.n;
        return os << f.n << '/' << f.d;
    }
    friend istream &operator>>(istream &is, frac &f) {
        istream &tp = is >> f.n >> f.d;
        f = frac(f.n, f.d);
        return tp;
    }
};
```

4.4 FWT [02d887]

```
struct Fast_Walsh_Transform { // Modint needed
    string op; // and, or, xor
    void fwt(vector<mint> &v, bool ifwt) {
        int n = __lg(size(v));
        mint iv2 = mint(1) / 2;
        for (int i = 0; i < n; ++i)
            for (int j = 0; j < 1 << n; ++j)
                if (op == "and" && (~j >> i &
                    1) || op == "or" && (j >> i & 1)) {
                    if (!ifwt)
                        v[j] += v[j ^ (1 << i)];
                    else
                        v[j] -= v[j ^ (1 << i)];
                } else
                     if (op == "xor" && (j >> i & 1)) {
                    mint x = v[j ^ (1 << i)], y = v[j];
                    if (!ifwt)
                        v[j ^ (1 <<
                             i)] = x + y, v[j] = x - y;
                    else
                        v[j ^ (1 << i)] = (x + y) *
                             iv2, v[j] = (x - y) * iv2;
                }
    }
    vector<mint> v1, v2; // size(v1) = size(v2) = 2^k
    Fast_Walsh_Transform(const vector
        <mint> &_v1, const vector<mint> &_v2, const
         string &_op) : v1(_v1), v2(_v2), op(_op) {}
    vector<mint> solve
        () { // ans_k = \sum_{i op j = k} a_i * b_j
        fwt(v1, 0), fwt(v2, 0);
        for (int i = 0; i < size(v1); ++i)
            v1[i] *= v2[i];
        fwt(v1, 1);
        return v1;
    }
};
```

4.5 FFT [c93fdc]

```
typedef complex<double> cd;
struct FFT {
#define M_PI 3.14159265358979323846264338327950288
    static const int K = 19, N = 1 << K; // change
    cd pl[N];
    int rv[N];
    void dft(vector<cd> &ar) {
        int n = size(ar), k = log2(n);
        if (n <= 1)
            return;
        for (int i = 1; i < n; i++)
            if (i < rv[i] >> (K - k))
                swap(ar[i], ar[rv[i] >> (K - k)]);
        cd a, b;
        for (int l = 1; l < n; l <<= 1) {
            for (int i = 0; i < n; i += l << 1) {
                for (int j = 0; j < l; j++) {
                    a = ar[i + j],
                         b = ar[i + j + l] * pl[j + l];
                    ar[i + j] = a + b;
                    ar[i + j + l] = a - b;
                }
            }
        }
    }
    void idft(vector<cd> &ar) {
        double n = size(ar);
        reverse(ar.begin() + 1, ar.end());
        for (cd &i : ar)
            i /= n;
        dft(ar);
    }
    vector<cd> pmul(vector<cd> a, vector<cd> b) {
        int n = size(a) + size(b) - 1;
        while (n & (n - 1))
            n += lowbit(n);
        a.resize(n), b.resize(n);
        dft(a), dft(b);
        for (int i = 0; i < n; i++)
            a[i] *= b[i];
        idft(a), a.resize(n);
        return a;
    }
    FFT() {
        pl[1] = polar(1.0, 0.0);
        for (int k = 2; k < N; k <<= 1)
            for (int i = k; i < k << 1; i++)
                pl[i] = polar(1.0,
                     2.0 * M_PI * (i - k) / (k * 2.0));
        for (int i = 1, hb = -1; i < N; i++) {
            if (!(i & (i - 1)))
                hb++;
            rv[i] =
                 rv[i ^ (1 << hb)] | 1 << (K - 1 - hb);
        }
    }
};
```

4.6 NTT [5af390]

```
const int M = 998244353;
typedef mint<M> mi;
struct NTT {
    static const int K = 20, N = 1 << K;
    mi pl[N];
    int rv[N];
    void dft(vector<mi> &ar) {
        static int n, k;
        n = size(ar), k = log2(n);
        if (n <= 1)
            return;
        for (int i = 1; i < n; i++)
            if (i < rv[i] >> (K - k))
                swap(ar[i], ar[rv[i] >> (K - k)]);
        static mi a, b;
        for (int l = 1; l < n; l <<= 1) {
            for (int i = 0; i < n; i += l << 1) {
                for (int j = 0; j < l; j++) {
                    a = ar[i + j],
                         b = ar[i + j + l] * pl[j + l];
                    ar[i + j] = a + b;
                    ar[i + j + l] = a - b;
                }
            }
        }
    }
    void idft(vector<mi> &ar) {
        static mi ivn;
        ivn.v = 1, ivn /= size(ar);
        reverse(ar.begin() + 1, ar.end());
        for (mi &i : ar)
            i *= ivn;
        dft(ar);
    }
  template<typename T>
    T pmul(T a, T b) {
        static int n;
        n = size(a) + size(b) - 1;
        while (n & (n - 1))
            n += lowbit(n);
        a.resize(n), b.resize(n);
        dft(a), dft(b);
        for (int i = 0; i < n; i++)
            a[i] *= b[i];
        idft(a), a.resize(n);
        return a;
    }
    NTT() {
        pl[1] = 1;
        for (int k = 1; k < K; k++) {
            mi omega = mi(3).pow((M - 1) >> (k + 1));
            for (int i = 1 << (k - 1); i < 1 << k; i++)
                pl[i * 2] = pl
                    [i], pl[i * 2 + 1] = pl[i] * omega;
        }
        for (int i = 1, hb = -1; i < N; i++) {
            if (!(i & (i - 1)))
                hb++;
            rv[i] =
                 rv[i ^ (1 << hb)] | 1 << (K - 1 - hb);
        }
    }
};
```

4.7 Polynomial [048b32]

```
template<typename T>
struct Poly:vector<T>{
  using vector<T>::vector;
  static NTT  con;
  Poly(const Poly &p,int m): vector<T>(m){
    copy_n(p.data(),min(size(p),m),this->data());
  }
  Poly& isz(int m) { return this->resize(m), *this; }
  Poly operator+(T const &b){
    Poly ret = Poly(*this,size(*this));
    ret[0]+=b;
    return ret;
  }
  Poly operator*(T const &b){
    Poly ret = Poly(*this,size(*this));
    for(int i=0;i<size(ret);i++) ret[i]*=b;
    return ret;
  }
  Poly operator+(Poly &b){
    Poly ret = Poly(*this,max(size(b),size(*this)));
    for(int i=0;i<size(ret);i++) ret[i]+=b[i];
    return ret;
  }
  Poly operator*(Poly b){
    return con.pmul(*this,b);
  }
  Poly dx() {
    Poly ret(size(*this)-1);
    for(int i
        =0;i<size(ret);i++) ret[i]=T(i+1)*(*this)[i+1];
    ret.resize(max(1,size(ret)));
    return ret;
  }
  Poly ix() {
    Poly ret(size(*this) + 1);
    for(int i=1;i<size
        (ret);i++) ret[i] = T(1)/T(i) * (*this)[i-1];
    return ret;
  }
  Poly inv() {
    int n = size(*this);
    if(n==1) return {T(1)/(*this)[0]};
    int m = n<<1;
    while(m^lowbit(m)) m+=lowbit(m);
    Poly xi = Poly(*this,(n+1)/2).inv().isz(m);
    Poly yi(*this,m);
    con.dft(xi);con.dft(yi);
    for(int i=0;i<m;i++){
      xi[i] = xi[i]*(mi(2)-yi[i]*xi[i]);
    }
    con.idft(xi);
    return xi.isz(n);
  }
  Poly ln(){
    int n = size(*this);
    Poly ret = (*this).dx();
    Poly inv = (*this).inv();
    ret = ret*inv;
    ret.resize(n);
    return ret.ix().isz(n);
  }
  Poly exp(){
    int n = size(*this);
    if(n==1) return {T(1)};
    Poly xi = Poly(*this,(n+1)/2).exp().isz(n);
    Poly yi = xi.ln();yi[0]=T(-1);
    for(int i=0;i<n;i++) yi[i] = (*this)[i]-yi[i];
    return (xi*yi).isz(n);
  }
  Poly pow(T k){
    Poly d = (*this).ln();
    d = d*k;
    d = d.exp();
    return d;
  }
};
template<typename T>
NTT Poly<T>::con;
```

4.8 Bit Conv [f7735d]

```
template <typename T>
```

```
inline void tozero(vector<T> &a, int n, int d) {
    d = 1 << d;
    for (int i = 0; i < 1 << n; i += d << 1) {
        for (int j = 0; j < d; j++) {
            a[i + j] += a[i + j + d];
            a[i + j + d] = 0;
        }
    }
}
template <typename T>
inline void flip(vector<T> &a, int n, int d) {
    d = 1 << d;
    for (int i = 0; i < 1 << n; i += d << 1) {
        for (int j = 0; j < d; j++) {
            swap(a[i + j], a[i + j + d]);
        }
    }
}

template <typename T>
vector<T> GCONV
    (vector<T> ar, vector<T> br, vector<string> ops) {
    int n = size(ops);
    ar.resize(1 << n), br.resize(1 << n);
    vector<int> op(n, 0);
    int fg = 0;
    for (int d = 0; d < n; d++) {
        if (ops[d][0] == '1') {
            fg ^= (1 << d);
            for (int i = 0; i < 4; i++)
                ops[d][i] = '0' + '1' - ops[d][i];
        }
        if (ops[d] == "0000")
            tozero(
                ar, n, d), tozero(br, n, d), op[d] = 0;
        else if (ops[d] == "0001")
            op[d] = 1;
        else if (ops[d] == "0010")
            flip(br, n, d), op[d] = 1;
        else if (ops[d] == "0011")
            tozero(br, n, d), op[d] = 2;
        else if (ops[d] == "0100")
            flip(ar, n, d), op[d] = 1;
        else if (ops[d] == "0101")
            tozero(ar, n, d), op[d] = 2;
        else if (ops[d] == "0110")
            op[d] = 3;
        else
            op[d] = 2;
        FWT(ar
            , n, d, op[d], 0), FWT(br, n, d, op[d], 0);
    }
    for (int i = 0; i < 1 << n; i++)
        ar[i] *= br[i];
    for (int d = n - 1; d >= 0; d--)
        FWT(ar, n, d, op[d], 1);
    for (int i = 0; i < 1 << n; i++)
        if (i < (i ^ fg))
            swap(ar[i], ar[i ^ fg]);
    return ar;
}
```

4.9 Miller Rabin [67a711]

```
bool isPrime(const uint64_t n) {
  if (n < 2 || n % 6 % 4 != 1) return (n | 1) == 3;
  uint64_t A[] = {2,
     325, 9375, 28178, 450775, 9780504, 1795265022},
  s = __builtin_ctzll(n-1), D = n >> s;
  for (auto a : A) {
    uint64_t p =1,g=a%n,i=s,d=D;
    for(;d;g=__int128
      (g)*g%n,d/=2) if(d&1) p = __int128(p)*g%n;
    while (p != 1 && p
       != n - 1 && a % n && i--) p = __int128(p)*p%n;
    if (p != n-1 && i != s) return 0;
  }
  return 1;
}
```

4.10 Pollard' s Rho [938b8b]

```
ll PollardRho(
    ll x) {// get a factor of x(not prime) in O(x^0.25)
  if(x%2==0) return 2;
  if(x%3==0) return 3;
    ll s = 0, t = 0;
    ll c = (ll)rand() % (x - 1) + 1;
    int step = 0, g = 1;
    ll val = 1;
    for (g = 1;; g <<= 1, s = t, val = 1) {
        for (step = 1; step <= g; ++step) {
            t =  (__int128(t)*t+c)%x;
            val = __int128(val) * abs(t - s) % x;
            if ((step % 127) == 0) {
                ll d = __gcd(val, x);
                if (d > 1) return d;
            }
        }
        ll d = __gcd(val, x);
        if (d > 1) return d;
    }
}
```

4.11 Discrete Log [78c47f]

```
int mod_log(int a, int b, int m) {
  if(b==1%m) return 0;
    int n = int(ceil(sqrt(m)));
    int e = 0;
    int k = 1;
    for (int g; (g = gcd(a, m)) != 1;) {
        if (b % g != 0) return -1;
        b /= g; m /= g;
        e++;
    k = (1LL*k*(a/g))%m;
        if (b == k) return e;
    }
    int x = 1;
    unordered_map<int, int> map_;
    for (int q = 0; q < n; q++, x = (1LL*x*a)%m) {
        map_[(1LL*x*b)%m] = q;
    }
    int y = x;
    for (int p = 1; p <= n; p++, y = (1LL*y*x)%m) {
        if (auto it =
             map_.find((1LL*y*k)%m); it != end(map_)) {
            return int(p) * n - it->second + e;
        }
    }
    return -1;
}
```

4.12 Exgcd/CRT [f45f4d]

```
// find x,y such that ax+by=gcd(a,b)
ll exgcd(ll a, ll b, ll &x, ll &y) {
  if (!b) return x = 1, y = 0, a;
  ll d = exgcd(b, a % b, y, x);
  return y -= a/b * x, d;
}
ll CRT(int k, ll* a, ll* r) {
  ll n = 1, ans = 0;
  for (int i = 1; i <= k; i++) n = n * r[i];
  for (int i = 1; i <= k; i++) {
    ll m = n / r[i], b, y;
    exgcd(m, r[i], b, y);  // b * m mod r[i] = 1
    ans = (ans + a[i] * m * b % n) % n;
  }
  return (ans % n + n) % n;
}
// not coprime
    : x = m1p+a1 = m2q+a2 => m1p-m2q = a2-a1, use exgcd
```

4.13 Semi-Euclid [e7979f]

```
//0~(n-1)!!!! sum floor((ax+b)/m)
ll semiEuclid(ll a,ll b,ll m,ll n){
  if(a==0) return (b/m)*(n);
  if(a >= m or b >= m) return n*(
      n-1)/2*(a/m) + n*(b/m) + semiEuclid(a%m,b%m,m,n);
  ll l = (a*(n-1)+b)/m;
  return l*(n-1) - semiEuclid(m,m-b-1,a,l);
}
```

4.14 LP

Primal	**Dual**
Maximize $c^\intercal x$ s.t. $Ax \le b, x \ge 0$	Minimize $b^\intercal y$ s.t. $A^\intercal y \ge c, y \ge 0$
Maximize $c^\intercal x$ s.t. $Ax \le b$	Minimize $b^\intercal y$ s.t. $A^\intercal y = c, y \ge 0$
Maximize $c^\intercal x$ s.t. $Ax = b, x \ge 0$	Minimize $b^\intercal y$ s.t. $A^\intercal y \ge c$

$\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$ are optimal if and only if for all $i \in [1,n]$, either $\bar{x}_i = 0$ or $\sum_{j=1}^{m} A_{ji}\bar{y}_j = c_i$ holds and for all $i \in [1,m]$ either $\bar{y}_i = 0$ or $\sum_{j=1}^{n} A_{ij}\bar{x}_j = b_j$ holds.

1. In case of minimization, let $c'_i = -c_i$
2. $\sum_{1\le i\le n} A_{ji}x_i \ge b_j \to \sum_{1\le i\le n} -A_{ji}x_i \le -b_j$
3. $\sum_{1\le i\le n} A_{ji}x_i = b_j$
   - $\sum_{1\le i\le n} A_{ji}x_i \le b_j$
   - $\sum_{1\le i\le n} A_{ji}x_i \ge b_j$
4. If x_i has no lower bound, replace x_i with $x_i - x'_i$

4.15 Simplex [6b4566]

```
const int MAXN = 11000, MAXM = 405;
const double eps = 1E-10;
double a[MAXN][MAXM], b[MAXN], c[MAXM];
double d[MAXN][MAXM], x[MAXM];
int ix[MAXN + MAXM]; // !!! array all indexed from 0
// max{cx} subject to {Ax<=b,x>=0}
// n: constraints, m: vars !!!
// x[] is the optimal solution vector
// usage :
// value = simplex(a, b, c, N, M);
double simplex(int n, int m){
  ++m;
  fill_n(d[n], m + 1, 0);
  fill_n(d[n + 1], m + 1, 0);
  iota(ix, ix + n + m, 0);
  int r = n, s = m - 1;
  for (int i = 0; i < n; ++i) {
    for (int j = 0; j < m - 1; ++j) d[i][j] = -a[i][j];
    d[i][m - 1] = 1;
    d[i][m] = b[i];
    if (d[r][m] > d[i][m]) r = i;
  }
  copy_n(c, m - 1, d[n]);
  d[n + 1][m - 1] = -1;
  for (double dd;; ) {
    if (r < n) {
      swap(ix[s], ix[r + m]);
      d[r][s] = 1.0 / d[r][s];
      for (int j = 0; j <= m; ++j)
        if (j != s) d[r][j] *= -d[r][s];
      for (int i = 0; i <= n + 1; ++i) if (i != r) {
        for (int j = 0; j <= m; ++j) if (j != s)
          d[i][j] += d[r][j] * d[i][s];
        d[i][s] *= d[r][s];
      }
    }
    r = s = -1;
    for (int j = 0; j < m; ++j)
      if (s < 0 || ix[s] > ix[j]) {
        if (d[n + 1][j] > eps ||
            (d[n + 1][j] > -eps && d[n][j] > eps))
          s = j;
      }
    if (s < 0) break;
    for (int i = 0; i < n; ++i) if (d[i][s] < -eps) {
      if (r < 0 ||
          (dd = d[r][m]
                / d[r][s] - d[i][m] / d[i][s]) < -eps ||
          (dd < eps && ix[r + m] > ix[i + m]))
        r = i;
    }
    if (r < 0) return -1; // not bounded
  }
  if (d[n + 1][m] < -eps) return -1; // not executable
  double ans = 0;
  fill_n(x, m, 0);
  for (int i = m; i <
      n + m; ++i) { // the missing enumerated x[i] = 0
    if (ix[i] < m - 1){
      ans += d[i - m][m] * c[ix[i]];
      x[ix[i]] = d[i-m][m];
    }
  }
  return ans;
}
```

4.16 Theorem

- LTE Lemma $\nu_p(x^n - y^n) = \nu_p(x-y) + \nu_p(n)$, if $p|x-y$. $\nu_2(x^n - y^n) = \nu_2(x-y) + \nu_2(n)$, if $4|x-y$. $\nu_2(x^n - y^n) = \nu_2(x-y) + \nu_2(x+y) + \nu_2(n) - 1$, if $2|x-y$ and n is even. $\nu_p(x^n + y^n) = \nu_p(x+y) + \nu_p(n)$, if $p|x+y$ and n is odd. $\nu_2(x^n + y^n) = 1$, if $2|x+y$ and n is even. $\nu_2(x^n + y^n) = \nu(x+y)$ if $2|x+y$ and n is odd.
- Cramer' s rule

$$\begin{array}{c} ax+by=e \\ cx+dy=f \end{array} \Rightarrow \begin{array}{c} x=\dfrac{ed-bf}{ad-bc} \\ y=\dfrac{af-ec}{ad-bc} \end{array}$$

- Vandermonde' s Identity

$$C(n+m,k)=\sum_{i=0}^{k} C(n,i)C(m,k-i)$$

- Kirchhoff' s Theorem
  Denote L be a $n \times n$ matrix as the Laplacian matrix of graph G, where $L_{ii} = d(i)$, $L_{ij} = -c$ where c is the number of edge (i,j) in G.
  - The number of undirected spanning in G is $|\det(\tilde{L}_{11})|$.
  - The number of directed spanning tree rooted at r in G is $|\det(\tilde{L}_{rr})|$.
- Tutte' s Matrix
  Let D be a $n \times n$ matrix, where $d_{ij} = x_{ij}$ (x_{ij} is chosen uniformly at random) if $i < j$ and $(i,j) \in E$, otherwise $d_{ij} = -d_{ji}$. $\frac{rank(D)}{2}$ is the maximum matching on G.
- Cayley' s Formula
  - Given a degree sequence $d_1, d_2, \ldots, d_n$ for each labeled vertices, there are $\frac{(n-2)!}{(d_1-1)!(d_2-1)!\cdots(d_n-1)!}$ spanning trees.
  - Let $T_{n,k}$ be the number of labeled forests on n vertices with k components, such that vertex $1,2,\ldots,k$ belong to different components. Then $T_{n,k} = kn^{n-k-1}$.
- Erd⊠s–Gallai theorem
  A sequence of nonnegative integers $d_1 \ge \cdots \ge d_n$ can be represented as the degree sequence of a finite simple graph on n vertices if and only if $d_1 + \cdots + d_n$ is even and $\displaystyle\sum_{i-1}^{k} d_i \le k(k-1) + \sum_{i=k+1}^{n} \min(d_i,k)$ holds for every $1 \le k \le n$.
- Gale–Ryser theorem
  A pair of sequences of nonnegative integers $a_1 \ge \cdots \ge a_n$ and $b_1,\ldots,b_n$ is bigraphic if and only if $\displaystyle\sum_{i=1}^{n} a_i = \sum_{i=1}^{n} b_i$ and $\displaystyle\sum_{i=1}^{k} a_i \le \sum_{i=1}^{n} \min(b_i,k)$ holds for every $1 \le k \le n$.
- Fulkerson–Chen–Anstee theorem
  A sequence $(a_1, b_1), \ldots, (a_n, b_n)$ of nonnegative integer pairs with $a_1 \ge \cdots \ge a_n$ is digraphic if and only if $\displaystyle\sum_{i=1}^{n} a_i = \sum_{i=1}^{n} b_i$ and $\displaystyle\sum_{i=1}^{k} a_i \le \sum_{i=1}^{k} \min(b_i,k-1) + \sum_{i=k+1}^{n} \min(b_i,k)$ holds for every $1 \le k \le n$.
- Pick' s theorem
  For simple polygon, when points are all integer, we have $A = \#\{\text{lattice points in the interior}\} + \frac{\#\{\text{lattice points on the boundary}\}}{2} - 1$.
- Möbius inversion formula
  - $f(n) = \sum_{d|n} g(d) \Leftrightarrow g(n) = \sum_{d|n} \mu(d) f(\frac{n}{d})$
  - $f(n) = \sum_{n|d} g(d) \Leftrightarrow g(n) = \sum_{n|d} \mu(\frac{d}{n}) f(d)$
- Spherical cap
  - A portion of a sphere cut off by a plane.
  - r: sphere radius, a: radius of the base of the cap, h: height of the cap, θ: $\arcsin(a/r)$.
  - Volume $= \pi h^2(3r-h)/3 = \pi h(3a^2+h^2)/6 = \pi r^3(2+\cos\theta)(1-\cos\theta)^2/3$.

  - Area $=2\pi rh=\pi(a^2+h^2)=2\pi r^2(1-\cos\theta)$.
- Lagrange multiplier
  - Optimize $f(x_1,\ldots,x_n)$ when k constraints $g_i(x_1,\ldots,x_n)=0$.
  - Lagrangian function $\mathcal{L}(x_1,\ldots,x_n,\lambda_1,\ldots,\lambda_k)=f(x_1,\ldots,x_n)-\sum_{i=1}^{k}\lambda_i g_i(x_1,\ldots,x_n)$.
  - The solution corresponding to the original constrained optimization is always a saddle point of the Lagrangian function.
- Nearest points of two skew lines
  - Line 1: $\boldsymbol{v}_1=\boldsymbol{p}_1+t_1\boldsymbol{d}_1$
  - Line 2: $\boldsymbol{v}_2=\boldsymbol{p}_2+t_2\boldsymbol{d}_2$
  - $\boldsymbol{n}=\boldsymbol{d}_1\times\boldsymbol{d}_2$
  - $\boldsymbol{n}_1=\boldsymbol{d}_1\times\boldsymbol{n}$
  - $\boldsymbol{n}_2=\boldsymbol{d}_2\times\boldsymbol{n}$
  - $\boldsymbol{c}_1=\boldsymbol{p}_1+\frac{(\boldsymbol{p}_2-\boldsymbol{p}_1)\cdot\boldsymbol{n}_2}{\boldsymbol{d}_1\cdot\boldsymbol{n}_2}\boldsymbol{d}_1$
  - $\boldsymbol{c}_2=\boldsymbol{p}_2+\frac{(\boldsymbol{p}_1-\boldsymbol{p}_2)\cdot\boldsymbol{n}_1}{\boldsymbol{d}_2\cdot\boldsymbol{n}_1}\boldsymbol{d}_2$
- Derivatives/Integrals

  Integration by parts: $\int_a^b f(x)g(x)dx=[F(x)g(x)]_a^b-\int_a^b F(x)g'(x)dx$

$$\begin{array}{|c|c|c|}
\frac{d}{dx}\sin^{-1}x=\frac{1}{\sqrt{1-x^2}} & \frac{d}{dx}\cos^{-1}x=-\frac{1}{\sqrt{1-x^2}} & \frac{d}{dx}\tan^{-1}x=\frac{1}{1+x^2} \\
\frac{d}{dx}\tan x=1+\tan^2 x & \int\tan ax=-\frac{\ln|\cos ax|}{a} & \\
\int e^{-x^2}=\frac{\sqrt{\pi}}{2}\operatorname{erf}(x) & \int xe^{ax}dx=\frac{e^{ax}}{a^2}(ax-1) &
\end{array}$$

$$\int\sqrt{a^2+x^2}=\frac{1}{2}\left(x\sqrt{a^2+x^2}+a^2\operatorname{asinh}(x/a)\right)$$

- Spherical Coordinate

$$(x,y,z)=(r\sin\theta\cos\phi,r\sin\theta\sin\phi,r\cos\theta)$$

$$(r,\theta,\phi)=(\sqrt{x^2+y^2+z^2},\operatorname{acos}(z/\sqrt{x^2+y^2+z^2}),\operatorname{atan2}(y,x))$$

- Rotation Matrix

$$M(\theta)=\begin{bmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{bmatrix},R_x(\theta_x)=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\theta_x & -\sin\theta_x\\ 0 & \sin\theta & \cos\theta\end{bmatrix}$$

- Pell’ s equation $x^2-ny^2=1\Rightarrow$ let x_1,y_1 be the smallest solution: the other solution has the form $x_i+y_i\sqrt{n}=(x_1+y_1\sqrt{n})^i$ (x_1,y_1 can be found by continuous fraction expansion of $\sqrt{n}$)
- Lucas’ Theorem

$$\binom{n}{i}\equiv\prod_{j=0}^{m}\binom{n_j}{i_j}\pmod{p}$$

  where a_j is a’ s representation in p-base
- Cipolla’ s algorithm find $a\Rightarrow(\frac{a^2-n}{p})=-1$(legendre symbol) and calculate $(a+\sqrt{a^2-n})^{(p+1)/2}$ in the field $(F)_p(\sqrt{a^2-n})$, that will be a solution.(note: p=2 need to be seperate out)

5 String

5.1 AC auto [0852a8]

```cpp
struct ACauto {
    const static int N = 2e5 + 5; // change
    int tr
        [26][N], fail[N], ctn = 1, cnt[N], endat[N], n;
    vector<int> top; // fail tree topological order
    inline void clr(int p) {
        fail[p] = cnt[p] = 0;
        for (int i = 0; i < 26; i++)
            tr[i][p] = 0;
    }
    inline int add(const string &s) {
        int cr = 1;
        for (int c : s) {
            c -= 'a';
            if (!tr[c][cr])
                clr(tr[c][cr] = ++ctn);
            cr = tr[c][cr];
        }
        return cr;
    }
    void blt(const vector<string> &ar) {
        for (int i = 0; i < 26; i++)
            tr[i][0] = 1;
        clr(ctn = 1), n = size(ar);
        for (int i = 0; i < n; i++)
            endat[i] = add(ar[i]);
        queue<int> q;
        q.push(1);
        while (!q.empty()) {
            int pr = q.front();
            q.pop();
            top.pb(pr);
            for (int i = 0; i < 26; i++) {
                int &cr = tr[i][pr];
                if (cr)
                    fail[cr]
                        = tr[i][fail[pr]], q.push(cr);
                else
                    cr = tr[i][fail[pr]];
            }
        }
        reverse(iter(top));
    }
    void qry(const string &s) {
        int cr = 1;
        for (char c : s) // ways to walk
            cr = tr[c - 'a'][cr], cnt[cr]++;
        for (int i : top)
            cnt[fail[i]] += cnt[i];
    }
};
```

5.2 SA [e051ae]

```cpp
struct SA {
    static const int N = 5e5 + 5; // change
    int sa[N], rk[N], cnt[N], lcp[N], tmp[N], n;
    void blt(const string &s) {
        n = s.length();
        int m = 128;
        memset(cnt + 1, 0, m << 2);
        for (int i = 0; i < n; i++)
            cnt[rk[i] = s[i]]++;
        for (int i = 1; i <= m; i++)
            cnt[i] += cnt[i - 1];
        for (int i = n - 1; i >= 0; i--)
            sa[--cnt[rk[i]]] = i;
        for (int k = 1;; k <<= 1) {
            int ln = 0;
            for (int i = n - k; i < n; i++)
                tmp[ln++] = i;
            for (int i = 0; i < n; i++)
                if (sa[i] >= k)
                    tmp[ln++] = sa[i] - k;
            memset(cnt + 1, 0, m << 2);
            for (int i = 0; i < n; i++)
                cnt[rk[i]]++;
            for (int i = 1; i <= m; i++)
                cnt[i] += cnt[i - 1];
            for (int i = n - 1; i >= 0; i--)
                sa[--cnt[rk[tmp[i]]]] = tmp[i];
            memcpy(tmp, rk, n << 2), rk[sa[0]] = m = 1;
            for (int i = 1; i < n; i++) {
                if (tmp[sa[i]] != tmp[sa[i - 1]] ||
                    sa[i - 1] + k >= n ||
                    tmp[sa
                        [i] + k] != tmp[sa[i - 1] + k])
                    m++;
                rk[sa[i]] = m;
            }
            if (m == n)
                break;
        }
        for (int i = 0, k = 0; i < n; i++, k -= !!k) {
            if (rk[i] == n)
                continue;
            int j = sa[rk[i]];
            while (i + k <
                n && j + k < n && s[i + k] == s[j + k])
                k++;
            lcp[rk[i]] = k;
        } // lcp[1~n-1],sa[0~n-1]
    }
};
```

5.3 KMP [2b5dca]

```
vector<int> kmp(string &s) {
    int n = size(s);
    vector<int> pi(n, 0);
    for (int i = 1; i < n; i++) {
        int j = pi[i - 1];
        while (j > 0 && s[i] != s[j])
            j = pi[j - 1];
        if (s[i] == s[j])
            j++;
        pi[i] = j;
    }
    return pi;
}
```

5.4 Z [29dad4]

```
vector<int> z_algo(string &s) { // 0-base
    int n = size(s);
    vector<int> z(n, 0);
    for (int i = 1, l = 0, r = 0; i < n; i++) {
        if (i <= r)
            z[i] = min(z[i - l], r - i + 1);
        while (i + z[i] < n && s[z[i]] == s[i + z[i]])
            z[i]++;
        if (i + z[i] - 1 > r)
            l = i, r = i + z[i] - 1;
    }
    z[0] = n;
    return z;
}
```

5.5 Manacher [947d5b]

```
vector<int> manacher(string
    &s) {//pad s with $->$a$b$c....z$, realans = ans-1.
  string t(size(s) * 2 + 1, '$');
  for(int i=0;i<size(s);i++) t[i * 2 + 1] = s[i];
  int n = size(t), m = 0, r = 0;
  vector<int> p(n);
  for(int i=0;i<n;i++) {
    p[i] = (r > i ? min(r - i, p[m - (i - m)]) : 1);
    while (i-p[i]>=0 && i + p[i] < n &&
      t[i - p[i]] == t[i + p[i]]) ++p[i];
    if (i + p[i] > r) m = i, r = i + p[i];
  }
  return p;
}
```

5.6 SAM [96a460]

```
struct SAM {
    static
        const int N = 2e5 + 6; // node < 2n ; edge < 3n
    int tr[26][
        N], len[N], lnk[N], tmp[N], pl[N], ctn, lst;
    inline void ini(int x) {
        for (int i = 0; i < 26; i++)
            tr[i][x] = 0;
    }
    inline void ini() {
        ini(ctn = lst = len[0] = 0), lnk[0] = -1;
        memset(tmp, 0, sizeof(tmp));
    }
    inline void cp(int x, int y) {
        lnk[x] = lnk[y];
        for (int i = 0; i < 26; i++)
            tr[i][x] = tr[i][y];
    }
    void ext(const char cc) {
        static int c, cr, p, q;
        c = cc - 'a', p = lst;
        ini(cr = ++ctn), len[cr] = len[p] + 1;
        while (~p && !tr[c][p])
            tr[c][p] = cr, p = lnk[p];
        if (~p) {
            q = tr[c][p];
            if (len[q] == len[p] + 1)
                lnk[cr] = q;
            else {
                cp(++ctn, q), len[ctn] = len[p] + 1;
                while (~p && tr[c][p] == q)
                    tr[c][p] = ctn, p = lnk[p];
                lnk[cr] = lnk[q] = ctn;
            }
        } else
            lnk[cr] = 0;
        lst = cr;
    }
    void blt(const string &s) {
        ini();
        for (const char &c : s)
            ext(c);
        for (int i = 0; i <= ctn; i++)
            tmp[len[i]]++;
        for (int i = 1; i <= ctn; i++)
            tmp[i] += tmp[i - 1];
        for (int i = ctn; i >= 0; i--)
            pl[--tmp[len[i]]] = i;
    }
};
```

5.7 exSAM [98e565]

```
struct exSAM {
    static const int N = 2e6 + 6;
    int tr[26][
        N], len[N], lnk[N], tmp[N], pl[N], ctn, lst;
    inline void ini(int x) {
        for (int i = 0; i < 26; i++)
            tr[i][x] = 0;
    }
    inline void ini() {
        ini(ctn = lst = len[0] = 0), lnk[0] = -1;
    memset(tmp,0,sizeof(tmp));
    }
    inline void cp(int x, int y) {
        lnk[x] = lnk[y];
        for (int i = 0; i < 26; i++)
            tr[i][x] = tr[i][y];
    }
    void ext(const char cc) {
        static int c, cr, p, q,*x;
        c = cc - 'a', p = lst;
    if (!tr[c][p]){
      ini(cr = ++ctn), len[cr] = len[p] + 1;
        while (~p && !tr[c][p])
          tr[c][p] = cr, p = lnk[p];
      lst=cr,x=&lnk[cr];
    }
    else x=&lst;
        if (~p) {
            q = tr[c][p];
            if (len[q] == len[p] + 1)
                *x = q;
            else {
                cp(++ctn, q), len[ctn] = len[p] + 1;
                while (~p && tr[c][p] == q)
                    tr[c][p] = ctn, p = lnk[p];
                *x = lnk[q] = ctn;
            }
        } else
            lnk[cr] = 0;
    }
    void blt(const string &s) {
        lst=0;
        for (const char &c : s)
            ext(c);
    }
};
```

6 Data Sructure

6.1 Li-Chao [f2885c]

```
struct LiChaoMin {
    struct line {
        ll m, k, id;
        line(ll _m = 0, ll _k
            = 0, ll _id = 0) : m(_m), k(_k), id(_id) {}
        ll at(ll x) { return m * x + k; }
    };
    struct node {
        node *l, *r;
        line f;
        node(line v) : f(v), l(NULL), r(NULL) {}
    };
    node *root;
    int sz;
    void insert(node *&x, int l, int r, line &ln) {
        if (!x) {
            x = new node(ln);
            return;
        }
        ll trl = x->f.at(l), trr = x->f.at(r);
        ll vl = ln.at(l), vr = ln.at(r);
        if (trl <= vl && trr <= vr)
            return;
        if (trl > vl && trr > vr) {
            x->f = ln;
            return;
        }
        if (trl > vl)
            swap(x->f, ln);
        int mid = (l + r) >> 1;
        if (x->f.at(mid) < ln.at(mid))
            insert(x->r, mid + 1, r, ln);
        else
            swap(x->f, ln), insert(x->l, l, mid, ln);
    }
    ll query(node *&x, int l, int r, ll idx) {
        if (!x)
            return LONG_LONG_MAX;
        if (l == r)
            return x->f.at(idx);
        int mid = (l + r) >> 1;
        if (mid >= idx)
            return min(x
                ->f.at(idx), query(x->l, l, mid, idx));
        return min(x
            ->f.at(idx), query(x->r, mid + 1, r, idx));
    }
    LiChaoMin(int _sz) : sz(_sz + 1), root(NULL) {}
    void add_line(ll m, ll k, ll id = 0) {
        auto ln = line(m, k, id);
        insert(root, -sz, sz, ln);
    } // -sz <= query_x <= sz
    ll query
        (ll idx) { return query(root, -sz, sz, idx); }
};
```

6.2 Treap [d7ba9f]

```
namespace Treap {
    const int N = 2e5 + 5;
    struct node {
        int ky, sz, mn, mx, ln, rn;
        ll sum;
        bool tg;
        static node *pl;
        node() { sum = sz = 0, mx = ~INF, mn = INF; }
        inline void ini(int v) { sum = ky
             = mx = mn = v, sz = 1, ln = rn = tg = 0; }
        inline void upd(int v) {
            ky = mx = mn = v;
            sum = 1ll * v * sz;
            tg = 1;
        }
        inline void up() {
            sz = 1 + pl[ln].sz + pl[rn].sz;
            sum = mx = mn = ky;
            sum += pl[ln].sum + pl[rn].sum;
            tmax(mx, max(pl[ln].mx, pl[rn].mx));
            tmin(mn, min(pl[ln].mn, pl[rn].mn));
        }
        inline
             void down() { tg && (ln && (pl[ln].upd(ky)
            , 0), rn && (pl[rn].upd(ky), 0)), tg = 0; }
    } pool[N];
    mt19937 rnd(time(0));
    node *node::pl = pool, *pl = pool;
    int ctp = 0;
    inline int nwnd(int v) {
        pl[++ctp].ini(v);
        return ctp;
    }
    int mg(int a, int b) {
        if (!a || !b)
            return a ? a : b;
        if ((int
            )rnd() % (pl[a].sz + pl[b].sz) < pl[a].sz)
            return pl[a].down(), pl[
                a].rn = mg(pl[a].rn, b), pl[a].up(), a;
        else
            return pl[b].down(), pl[
                b].ln = mg(a, pl[b].ln), pl[b].up(), b;
    }
    void splsz(int rt, int &a, int &b, int k) {
        if (!rt)
            return a = b = 0, void();
        pl[rt].down();
        if (pl[pl[rt].ln].sz < k)
            a = rt, splsz(pl[rt].rn,
                pl[a].rn, b, k - pl[pl[rt].ln].sz - 1);
        else
            b = rt, splsz(pl[rt].ln, a, pl[b].ln, k);
        pl[rt].up();
    }
    void splky(int rt, int &a, int &b, int v) {
        if (!rt)
            return a = b = 0, void();
        pl[rt].down();
        if (pl[rt].ky <= v)
            a = rt, splky(pl[rt].rn, pl[a].rn, b, v);
        else
            b = rt, splky(pl[rt].ln, a, pl[b].ln, v);
        pl[rt].up();
    }
}
```

6.3 Link Cut Tree [49d7e6]

```
struct LCT {
    static const int N = 4e5 + 5; // change
    int fa[N], ch[2][N], sz[N], sv[N];
#define gch(x) ((x) == ch[1][fa[x]])
#define nrt(x) ((x) == ch[gch(x)][fa[x]])
#define up
    (x) sz[x] = sz[ch[0][x]] + sz[ch[1][x]] + sv[x] + 1
    inline void rota(int x) {
        int f = fa[x], ff = fa[f], k = gch(x);
        if (nrt(f))
            ch[gch(f)][ff] = x;
        fa[x] = ff;
        if (ch[!k][x])
            fa[ch[!k][x]] = f;
        ch[k][f] = ch[!k][x];
        ch[!k][x] = f, fa[f] = x;
        up(f), up(x);
    }
    inline void splay(int x) {
        for (int f = fa[x]; nrt(x); rota(x), f = fa[x])
            if (nrt(f))
                rota(gch(x) ^ gch(f) ? x : f);
    }
    inline int acc(int x) {
        int p;
        for (p = 0; x; p = x, x = fa[x])
            splay(x), sv[x] += sz[
                ch[1][x]] - sz[p], ch[1][x] = p, up(x);
        return p;
    }
    inline int findroot(int x) {
        int cr = acc(x);
        while (ch[0][cr])
            cr = ch[0][cr];
        splay(cr);
        return cr;
    }
    void link(int x, int y) {
        int rt = findroot(y);
        acc(x), acc(rt);
        acc(y), splay(y);
        sv[y] += sz[x];
        fa[x] = y, up(y);
        acc(rt);
    }
    void cut(int x, int y) {
        int rt = findroot(y);
        acc(rt);
        acc(y), splay(y), splay(x);
        sv[y] -= sz[x];
        fa[x] = 0, up(y);
        acc(x), acc(rt);
    }
};
```

6.4 Ultimate Segment Tree [6e7e86]

```
struct SegBeat {
    int n,n0;
    vector<ll> max_v,smax_v
        ,max_c,min_v,smin_v,min_c,sum,len,ladd,lval;
    void update_node_max(int k, ll x) {
        sum[k] += (x - max_v[k]) * max_c[k];
        if(max_v
            [k] == min_v[k]) max_v[k] = min_v[k] = x;
        else if(max_v
            [k] == smin_v[k]) max_v[k] = smin_v[k] = x;
        else max_v[k] = x;
        if(lval[k] != 1e18 && x < lval[k]) lval[k] = x;
    }
    void update_node_min(int k, ll x) {
        sum[k] += (x - min_v[k]) * min_c[k];
        if(max_v
            [k] == min_v[k]) max_v[k] = min_v[k] = x;
        else if(smax_v
            [k] == min_v[k]) min_v[k] = smax_v[k] = x;
        else min_v[k] = x;
        if(lval[k] != 1e18 && lval[k] < x) lval[k] = x;
    }
    void push(int k) {
        if(n0-1 <= k) return;
        if(lval[k] != 1e18) {
            updateall(2*k+1, lval[k]);
            updateall(2*k+2, lval[k]);
            lval[k] = 1e18;
            return;
        }
        if(ladd[k] != 0) {
            addall(2*k+1, ladd[k]);
            addall(2*k+2, ladd[k]);
            ladd[k] = 0;
        }
        if(max_v[k] < max_v
            [2*k+1]) update_node_max(2*k+1, max_v[k]);
        if(min_v[2*k+1] <
            min_v[k]) update_node_min(2*k+1, min_v[k]);
        if(max_v[k] < max_v
            [2*k+2]) update_node_max(2*k+2, max_v[k]);
        if(min_v[2*k+2] <
            min_v[k]) update_node_min(2*k+2, min_v[k]);
    }
    void update(int k) {
        sum[k] = sum[2*k+1] + sum[2*k+2];
        if(max_v[2*k+1] < max_v[2*k+2]) {
            max_v[k] = max_v[2*k+2];
            max_c[k] = max_c[2*k+2];
            smax_v
                [k] = max(max_v[2*k+1], smax_v[2*k+2]);
        } else if(max_v[2*k+1] > max_v[2*k+2]) {
            max_v[k] = max_v[2*k+1];
            max_c[k] = max_c[2*k+1];
            smax_v
                [k] = max(smax_v[2*k+1], max_v[2*k+2]);
        } else {
            max_v[k] = max_v[2*k+1];
            max_c[k] = max_c[2*k+1] + max_c[2*k+2];
            smax_v[
                k] = max(smax_v[2*k+1], smax_v[2*k+2]);
        }
        if(min_v[2*k+1] < min_v[2*k+2]) {
            min_v[k] = min_v[2*k+1];
            min_c[k] = min_c[2*k+1];
            smin_v
                [k] = min(smin_v[2*k+1], min_v[2*k+2]);
        } else if(min_v[2*k+1] > min_v[2*k+2]) {
            min_v[k] = min_v[2*k+2];
            min_c[k] = min_c[2*k+2];
            smin_v
                [k] = min(min_v[2*k+1], smin_v[2*k+2]);
        } else {
            min_v[k] = min_v[2*k+1];
            min_c[k] = min_c[2*k+1] + min_c[2*k+2];
            smin_v[
                k] = min(smin_v[2*k+1], smin_v[2*k+2]);
        }
    }
    void _chmin
        (ll x, int a, int b, int k, int l, int r) {
        if(b <= l || r <= a || max_v[k] <= x) return;
        if(a <= l && r <= b && smax_v[k] < x) {
            update_node_max(k, x);
            return;
        }
        push(k);
        _chmin(x, a, b, 2*k+1, l, (l+r)/2);
        _chmin(x, a, b, 2*k+2, (l+r)/2, r);
        update(k);
    }
    void _chmax
        (ll x, int a, int b, int k, int l, int r) {
        if(b <= l || r <= a || x <= min_v[k]) return;
        if(a <= l && r <= b && x < smin_v[k]) {
            update_node_min(k, x);
            return;
        }
        push(k);
        _chmax(x, a, b, 2*k+1, l, (l+r)/2);
        _chmax(x, a, b, 2*k+2, (l+r)/2, r);
        update(k);
    }
    void addall(int k, ll x) {
        max_v[k] += x;
        if(smax_v[k] != -1e18) smax_v[k] += x;
        min_v[k] += x;
        if(smin_v[k] != 1e18) smin_v[k] += x;
        sum[k] += len[k] * x;
        if(lval[k] != 1e18) lval[k] += x;
        else ladd[k] += x;
    }
    void updateall(int k, ll x) {
        max_v[k] = x;
        smax_v[k] = -1e18;
        min_v[k] = x;
        smin_v[k] = 1e18;
        max_c[k] = min_c[k] = len[k];
        sum[k] = x * len[k];
        lval[k] = x;
        ladd[k] = 0;
    }
    void _add_val
        (ll x, int a, int b, int k, int l, int r) {
```

```
        if(b <= l || r <= a) return;
        if(a <= l && r <= b) {
            addall(k, x);
            return;
        }
        push(k);
        _add_val(x, a, b, 2*k+1, l, (l+r)/2);
        _add_val(x, a, b, 2*k+2, (l+r)/2, r);
        update(k);
    }
    void _update_val
        (ll x, int a, int b, int k, int l, int r) {
        if(b <= l || r <= a) return;
        if(a <= l && r <= b) {
            updateall(k, x);
            return;
        }
        push(k);
        _update_val(x, a, b, 2*k+1, l, (l+r)/2);
        _update_val(x, a, b, 2*k+2, (l+r)/2, r);
        update(k);
    }
    ll _query_max(int a, int b, int k, int l, int r) {
        if(b <= l || r <= a) return -1e18;
        if(a <= l && r <= b) return max_v[k];
        push(k);
        ll lv = _query_max(a, b, 2*k+1, l, (l+r)/2);
        ll rv = _query_max(a, b, 2*k+2, (l+r)/2, r);
        return max(lv, rv);
    }
    ll _query_min(int a, int b, int k, int l, int r) {
        if(b <= l || r <= a) return 1e18;
        if(a <= l && r <= b) return min_v[k];
        push(k);
        ll lv = _query_min(a, b, 2*k+1, l, (l+r)/2);
        ll rv = _query_min(a, b, 2*k+2, (l+r)/2, r);
        return min(lv, rv);
    }
    ll _query_sum(int a, int b, int k, int l, int r) {
        if(b <= l || r <= a) return 0;
        if(a <= l && r <= b) return sum[k];
        push(k);
        ll lv = _query_sum(a, b, 2*k+1, l, (l+r)/2);
        ll rv = _query_sum(a, b, 2*k+2, (l+r)/2, r);
        return lv + rv;
    }
    SegBeat(int _n) : n(_n) {
        max_v.resize(4*_n+4,0);smax_v.resize
            (4*_n+4,-1e18);max_c.resize(4*_n+4,1);
        min_v.resize(4*_n+4,0);smin_v.
            resize(4*_n+4,1e18);min_c.resize(4*_n+4,1);
        sum.resize(4*_n+4,0);len.resize(4*_n+4,0);
        ladd.resize(4*_n+4,0);lval.resize(4*_n+4,1e18);
        n0 = 1;
        while(n0 < n) n0 <<= 1;
        len[0] = n0;
        for(int i=0; i<n0-1; ++
            i) len[2*i+1] = len[2*i+2] = (len[i] >> 1);
        for(int i=n; i<n0; ++i) {
            max_v[n0-1+i] = smax_v[n0-1+i] = -1e18;
            min_v[n0-1+i] = smin_v[n0-1+i] = 1e18;
            max_c[n0-1+i] = min_c[n0-1+i] = 0;
        }
        for(int i=n0-2; i>=0; i--) update(i);
    }
    void chmin(int
         a, int b, ll x) {_chmin(x, a-1, b, 0, 0, n0);}
    void chmax(int
         a, int b, ll x) {_chmax(x, a-1, b, 0, 0, n0);}
    void add_val(int a
        , int b, ll x) {_add_val(x, a-1, b, 0, 0, n0);}
    void update_val(int a, int
         b, ll x) {_update_val(x, a-1, b, 0, 0, n0);}
    ll query_max(int a
        , int b) {return _query_max(a-1, b, 0, 0, n0);}
    ll query_min(int a
        , int b) {return _query_min(a-1, b, 0, 0, n0);}
    ll query_sum(int a
        , int b) {return _query_sum(a-1, b, 0, 0, n0);}
};
```

7 Misc

7.1 Total Binary Search [ac23f3]

```
struct TotalBS {
    int Q;
    vector<int> ans;
    TotalBS(int _Q) {
        Q=_Q;
        ans.resize(Q);
    }
    void split(vector<
        int> &qrys,vecotr<int> &ok,vector<int> &fail) {
        for(auto i :qrys) {
        }
        vector<int>.swap(qrys);
        return;
    }
    void do_things(int l,int mid) {
        return;
    }
    void undo_things(int l,int mid) {
        return;
    }
    void total_BS(int l, int r, vector<int> &qrys) {
        if (l == r) {
            for(auto i : qrys) {
                ans[i] = l;
            }
        }
        int mid = (l + r) / 2;
        do_things(l, mid);
        vector<int> lft,rgt;
        split(qrys,lft,rgt);
        total_BS(mid + 1, r, rgt);
        undo_things(l, mid);
        total_BS(l, mid, lft);
    }
};
```

7.2 Cyclic Taranry Search [9017cc]

```
/* bool pred(int a, int b);
   f(0) ~ f(n - 1) is a cyclic-shift U-function
   return idx s.t. pred(x, idx) is false forall x*/
int cyc_tsearch(int n, auto pred) {
  if (n == 1) return 0;
  int l = 0, r = n; bool rv = pred(1, 0);
  while (r - l > 1) {
    int m = (l + r) / 2;
    if (pred(0, m) ? rv: pred(m, (m + 1) % n)) r = m;
    else l = m;
  }
  return pred(l, r % n) ? l : r % n;
}
```

7.3 CDQ [11f96f]

```
void CDQ(int l, int r) { // 三維偏序
    if (l == r)
        return;
    int mid = (l + r) / 2;
    CDQ(l, mid);
    CDQ(mid + 1, r);
    sort(arr + l, arr + mid + 1, cmpB);
    sort(arr + mid + 1, arr + r + 1, cmpB);
    int i = l;
    int j = mid + 1;
    while (j <= r) {
        while (i <= mid && ue[i].b <= ue[j].b) {
            BIT.Add(arr[i].c, arr[i].cnt);
            i++;
        }
        ue[j].res += BIT.Ask(arr[j].c);
        j++;
    }
    for (int k = l; k < i; k++)
        BIT.Add(arr[k].c, -arr[k].cnt);
    return;
}
```