Contents

一切的开始

宏定义

- 需要 C++11

```cpp
#include <bits/stdc++.h>
using namespace std;
using LL = long long;
#define FOR(i, x, y) for (decay<decltype(y)>::type i = (x), _##i = (y); i < _##i; ++i)
#define FORD(i, x, y) for (decay<decltype(x)>::type i = (x), _##i = (y); i > _##i; --i)
#ifdef zerol
#define dbg(args...) do { cout << "\033[32;1m" << #args << " -> "; err(args); } while (0)
#else
#define dbg(...)
#endif
void err() { cout << "\033[39;0m" << endl; }
template<template<typename...> class T, typename t, typename... Args>
void err(T<t> a, Args... args) { for (auto x: a) cout << x << ' '; err(args...); }
template<typename T, typename... Args>
void err(T a, Args... args) { cout << a << ' '; err(args...); }
// --------------------------------------------------------------------------
```

- 更多配色:
  - 33 黄色
  - 34 蓝色
  - 31 橙色
- POJ/BZOJ version

```cpp
#include <cstdio>
#include <iostream>
#include <algorithm>
#include <cmath>
#include <string>
#include <vector>
#include <set>
#include <queue>
#include <cstring>
#include <cassert>
using namespace std;
typedef long long LL;
#define FOR(i, x, y) for (LL i = (x), _##i = (y); i < _##i; ++i)
#define FORD(i, x, y) for (LL i = (x), _##i = (y); i > _##i; --i)
#ifdef zerol
#define dbg(args...) do { cout << "\033[32;1m" << #args<< " -> "; err(args); } while (0)
#else
#define dbg(...)
#endif
void err() { cout << "\033[39;0m" << endl; }
template<typename T, typename... Args>
void err(T a, Args... args) {
    cout << a << ' '; err(args...);
}
// --------------------------------------------------------------------------
```

- HDU Assert Patch

```cpp
#ifdef ONLINE_JUDGE
#define assert(condition) if (!(condition)) { int x = 1, y = 0; cout << x / y << endl; }
#endif
```

快速读

```cpp
inline char next_char() {
    static char buf[100000], *p1 = buf, *p2 = buf;
    return p1 == p2 && (p2 = (p1 = buf) + fread(buf, 1, 100000, stdin), p1 == p2) ? EOF : *p1++;
}
inline bool maybe_digit(char c) {
```

```
 6        return c >= '0' && c <= '9';
 7    }
 8    template <typename T>
 9    void rn(T& _v) {
10        static char ch;
11        static bool negative = false;
12        _v = 0;
13        while (!maybe_digit(ch)) {
14            negative = ch == '-';
15            ch = next_char();
16        }
17        do _v = (_v << 1) + (_v << 3) + ch - '0';
18        while (maybe_digit(ch = next_char()));
19        if (negative) _v = -_v;
20    }
21
22    template <typename T>
23    void o(T p) {
24        static int stk[70], tp;
25        if (p == 0) {
26            putchar('0');
27            return;
28        }
29        if (p < 0) { p = -p; putchar('-'); }
30        while (p) stk[++tp] = p % 10, p /= 10;
31        while (tp) putchar(stk[tp--] + '0');
32    }
```

- 需要初始化
- 需要一次读入
- 不支持负数

```
 1    const int MAXS = 100 * 1024 * 1024;
 2    char buf[MAXS];
 3    template<typename T>
 4    inline bool read(T& x) {
 5        static char* p = buf;
 6        x = 0;
 7        while (*p && !isdigit(*p)) ++p;
 8        if (!*p) return false;
 9        while (isdigit(*p)) x = x * 10 + *p++ - 48;
10        return true;
11    }
12
13    fread(buf, 1, MAXS, stdin);
```

对拍

```
 1    #!/usr/bin/env bash
 2    g++ -o r main.cpp -O2 -std=c++11
 3    g++ -o std std.cpp -O2 -std=c++11
 4    while true; do
 5        python gen.py > in
 6        ./std < in > stdout
 7        ./r < in > out
 8        if test $? -ne 0; then
 9            exit 0
10        fi
11        if diff stdout out; then
12            printf "AC\n"
13        else
14            printf "GG\n"
15            exit 0
16        fi
17    done
```

为什么 C++ 不自带这个?

```cpp
LL bin(LL x, LL n, LL MOD) {
    LL ret = MOD != 1;
    for (x %= MOD; n; n >>= 1, x = x * x % MOD)
        if (n & 1) ret = ret * x % MOD;
    return ret;
}
inline LL get_inv(LL x, LL p) { return bin(x, p - 2, p); }
```

数据结构

ST 表

- 二维

```cpp
int f[maxn][maxn][10][10];
inline int highbit(int x) { return 31 - __builtin_clz(x); }
inline int calc(int x, int y, int xx, int yy, int p, int q) {
    return max(
        max(f[x][y][p][q], f[xx - (1 << p) + 1][yy - (1 << q) + 1][p][q]),
        max(f[xx - (1 << p) + 1][y][p][q], f[x][yy - (1 << q) + 1][p][q])
    );
}
void init() {
    FOR (x, 0, highbit(n) + 1)
    FOR (y, 0, highbit(m) + 1)
        FOR (i, 0, n - (1 << x) + 1)
        FOR (j, 0, m - (1 << y) + 1) {
            if (!x && !y) { f[i][j][x][y] = a[i][j]; continue; }
            f[i][j][x][y] = calc(
                i, j,
                i + (1 << x) - 1, j + (1 << y) - 1,
                max(x - 1, 0), max(y - 1, 0)
            );
        }
}
inline int get_max(int x, int y, int xx, int yy) {
    return calc(x, y, xx, yy, highbit(xx - x + 1), highbit(yy - y + 1));
}
```

- 一维

```cpp
struct RMQ {
    int f[22][M];
    inline int highbit(int x) { return 31 - __builtin_clz(x); }
    void init(int* v, int n) {
        FOR (i, 0, n) f[0][i] = v[i];
        FOR (x, 1, highbit(n) + 1)
            FOR (i, 0, n - (1 << x) + 1)
                f[x][i] = min(f[x - 1][i], f[x - 1][i + (1 << (x - 1))]);
    }
    int get_min(int l, int r) {
        assert(l <= r);
        int t = highbit(r - l + 1);
        return min(f[t][l], f[t][r - (1 << t) + 1]);
    }
} rmq;
```

线段树

- 普适

```cpp
namespace sg {
    struct Q {
        LL setv;
```

```
4          explicit Q(LL setv = -1): setv(setv) {}
5          void operator += (const Q& q) { if (q.setv != -1) setv = q.setv; }
6      };
7      struct P {
8          LL min;
9          explicit P(LL min = INF): min(min) {}
10          void up(Q& q) { if (q.setv != -1) min = q.setv; }
11      };
12      template<typename T>
13      P operator & (T&& a, T&& b) {
14          return P(min(a.min, b.min));
15      }
16      P p[maxn << 2];
17      Q q[maxn << 2];
18  #define lson o * 2, l, (l + r) / 2
19  #define rson o * 2 + 1, (l + r) / 2 + 1, r
20      void up(int o, int l, int r) {
21          if (l == r) p[o] = P();
22          else p[o] = p[o * 2] & p[o * 2 + 1];
23          p[o].up(q[o]);
24      }
25      void down(int o, int l, int r) {
26          q[o * 2] += q[o]; q[o * 2 + 1] += q[o];
27          q[o] = Q();
28          up(lson); up(rson);
29      }
30      template<typename T>
31      void build(T&& f, int o = 1, int l = 1, int r = n) {
32          if (l == r) q[o] = f(l);
33          else { build(f, lson); build(f, rson); q[o] = Q(); }
34          up(o, l, r);
35      }
36      P query(int ql, int qr, int o = 1, int l = 1, int r = n) {
37          if (ql > r || l > qr) return P();
38          if (ql <= l && r <= qr) return p[o];
39          down(o, l, r);
40          return query(ql, qr, lson) & query(ql, qr, rson);
41      }
42      void update(int ql, int qr, const Q& v, int o = 1, int l = 1, int r = n) {
43          if (ql > r || l > qr) return;
44          if (ql <= l && r <= qr) q[o] += v;
45          else {
46              down(o, l, r);
47              update(ql, qr, v, lson); update(ql, qr, v, rson);
48          }
49          up(o, l, r);
50      }
51  }
```

- SET + ADD

```
1   struct IntervalTree {
2   #define ls o * 2, l, m
3   #define rs o * 2 + 1, m + 1, r
4       static const LL M = maxn * 4, RS = 1E18 - 1;
5       LL addv[M], setv[M], minv[M], maxv[M], sumv[M];
6       void init() {
7           memset(addv, 0, sizeof addv);
8           fill(setv, setv + M, RS);
9           memset(minv, 0, sizeof minv);
10          memset(maxv, 0, sizeof maxv);
11          memset(sumv, 0, sizeof sumv);
12      }
13      void maintain(LL o, LL l, LL r) {
14          if (l < r) {
15              LL lc = o * 2, rc = o * 2 + 1;
16              sumv[o] = sumv[lc] + sumv[rc];
17              minv[o] = min(minv[lc], minv[rc]);
18              maxv[o] = max(maxv[lc], maxv[rc]);
19          } else sumv[o] = minv[o] = maxv[o] = 0;
20          if (setv[o] != RS) { minv[o] = maxv[o] = setv[o]; sumv[o] = setv[o] * (r - l + 1); }
21          if (addv[o]) { minv[o] += addv[o]; maxv[o] += addv[o]; sumv[o] += addv[o] * (r - l + 1); }
```

```
22        }
23      void build(LL o, LL l, LL r) {
24          if (l == r) addv[o] = a[l];
25          else {
26              LL m = (l + r) / 2;
27              build(ls); build(rs);
28          }
29          maintain(o, l, r);
30      }
31      void pushdown(LL o) {
32          LL lc = o * 2, rc = o * 2 + 1;
33          if (setv[o] != RS) {
34              setv[lc] = setv[rc] = setv[o];
35              addv[lc] = addv[rc] = 0;
36              setv[o] = RS;
37          }
38          if (addv[o]) {
39              addv[lc] += addv[o]; addv[rc] += addv[o];
40              addv[o] = 0;
41          }
42      }
43      void update(LL p, LL q, LL o, LL l, LL r, LL v, LL op) {
44          if (p <= r && l <= q)
45          if (p <= l && r <= q) {
46              if (op == 2) { setv[o] = v; addv[o] = 0; }
47              else addv[o] += v;
48          } else {
49              pushdown(o);
50              LL m = (l + r) / 2;
51              update(p, q, ls, v, op); update(p, q, rs, v, op);
52          }
53          maintain(o, l, r);
54      }
55      void query(LL p, LL q, LL o, LL l, LL r, LL add, LL& ssum, LL& smin, LL& smax) {
56          if (p > r || l > q) return;
57          if (setv[o] != RS) {
58              LL v = setv[o] + add + addv[o];
59              ssum += v * (min(r, q) - max(l, p) + 1);
60              smin = min(smin, v);
61              smax = max(smax, v);
62          } else if (p <= l && r <= q) {
63              ssum += sumv[o] + add * (r - l + 1);
64              smin = min(smin, minv[o] + add);
65              smax = max(smax, maxv[o] + add);
66          } else {
67              LL m = (l + r) / 2;
68              query(p, q, ls, add + addv[o], ssum, smin, smax);
69              query(p, q, rs, add + addv[o], ssum, smin, smax);
70          }
71      }
72  } IT;
```

均摊复杂度线段树

- 区间取 max，区间求和。

```
1   namespace R {
2   #define lson o * 2, l, (l + r) / 2
3   #define rson o * 2 + 1, (l + r) / 2 + 1, r
4       int m1[N], m2[N], cm1[N];
5       LL sum[N];
6       void up(int o) {
7           int lc = o * 2, rc = lc + 1;
8           m1[o] = max(m1[lc], m1[rc]);
9           sum[o] = sum[lc] + sum[rc];
10          if (m1[lc] == m1[rc]) {
11              cm1[o] = cm1[lc] + cm1[rc];
12              m2[o] = max(m2[lc], m2[rc]);
13          } else {
```

```
14              cm1[o] = m1[lc] > m1[rc] ? cm1[lc] : cm1[rc];
15              m2[o] = max(min(m1[lc], m1[rc]), max(m2[lc], m2[rc]));
16          }
17      }
18      void mod(int o, int x) {
19          if (x >= m1[o]) return;
20          assert(x > m2[o]);
21          sum[o] -= 1LL * (m1[o] - x) * cm1[o];
22          m1[o] = x;
23      }
24      void down(int o) {
25          int lc = o * 2, rc = lc + 1;
26          mod(lc, m1[o]); mod(rc, m1[o]);
27      }
28      void build(int o, int l, int r) {
29          if (l == r) { int t; read(t); sum[o] = m1[o] = t; m2[o] = -1; cm1[o] = 1; }
30          else { build(lson); build(rson); up(o); }
31      }
32      void update(int ql, int qr, int x, int o, int l, int r) {
33          if (r < ql || qr < l || m1[o] <= x) return;
34          if (ql <= l && r <= qr && m2[o] < x) { mod(o, x); return; }
35          down(o);
36          update(ql, qr, x, lson); update(ql, qr, x, rson);
37          up(o);
38      }
39      int qmax(int ql, int qr, int o, int l, int r) {
40          if (r < ql || qr < l) return -INF;
41          if (ql <= l && r <= qr) return m1[o];
42          down(o);
43          return max(qmax(ql, qr, lson), qmax(ql, qr, rson));
44      }
45      LL qsum(int ql, int qr, int o, int l, int r) {
46          if (r < ql || qr < l) return 0;
47          if (ql <= l && r <= qr) return sum[o];
48          down(o);
49          return qsum(ql, qr, lson) + qsum(ql, qr, rson);
50      }
51  }
```

持久化线段树

- ADD

```
1  namespace tree {
2  #define mid ((l + r) >> 1)
3  #define lson ql, qr, l, mid
4  #define rson ql, qr, mid + 1, r
5      struct P {
6          LL add, sum;
7          int ls, rs;
8      } tr[maxn * 45 * 2];
9      int sz = 1;
10     int N(LL add, int l, int r, int ls, int rs) {
11         tr[sz] = {add, tr[ls].sum + tr[rs].sum + add * (len[r] - len[l - 1]), ls, rs};
12         return sz++;
13     }
14     int update(int o, int ql, int qr, int l, int r, LL add) {
15         if (ql > r || l > qr) return o;
16         const P& t = tr[o];
17         if (ql <= l && r <= qr) return N(add + t.add, l, r, t.ls, t.rs);
18         return N(t.add, l, r, update(t.ls, lson, add), update(t.rs, rson, add));
19     }
20     LL query(int o, int ql, int qr, int l, int r, LL add = 0) {
21         if (ql > r || l > qr) return 0;
22         const P& t = tr[o];
23         if (ql <= l && r <= qr) return add * (len[r] - len[l - 1]) + t.sum;
24         return query(t.ls, lson, add + t.add) + query(t.rs, rson, add + t.add);
25     }
26 }
```

K-D Tree

最优化问题一定要用全局变量大力剪枝，而且左右儿子先递归潜力大的

- 维护信息
- 带重构（适合在线）
- 插入时左右儿子要标记为 null。

```
1  namespace kd {
2      const int K = 2, inf = 1E9, M = N;
3      const double lim = 0.7;
4      struct P {
5          int d[K], l[K], r[K], sz, val;
6          LL sum;
7          P *ls, *rs;
8          P* up() {
9              sz = ls->sz + rs->sz + 1;
10             sum = ls->sum + rs->sum + val;
11             FOR (i, 0, K) {
12                 l[i] = min(d[i], min(ls->l[i], rs->l[i]));
13                 r[i] = max(d[i], max(ls->r[i], rs->r[i]));
14             }
15             return this;
16         }
17     } pool[M], *null = new P, *pit = pool;
18     static P *tmp[M], **pt;
19     void init() {
20         null->ls = null->rs = null;
21         FOR (i, 0, K) null->l[i] = inf, null->r[i] = -inf;
22         null->sum = null->val = 0;
23         null->sz = 0;
24     }
25
26     P* build(P** l, P** r, int d = 0) { // [l, r)
27         if (d == K) d = 0;
28         if (l >= r) return null;
29         P** m = l + (r - l) / 2; assert(l <= m && m < r);
30         nth_element(l, m, r, [&](const P* a, const P* b){
31             return a->d[d] < b->d[d];
32         });
33         P* o = *m;
34         o->ls = build(l, m, d + 1); o->rs = build(m + 1, r, d + 1);
35         return o->up();
36     }
37     P* Build() {
38         pt = tmp; FOR (it, pool, pit) *pt++ = it;
39         return build(tmp, pt);
40     }
41     inline bool inside(int p[], int q[], int l[], int r[]) {
42         FOR (i, 0, K) if (r[i] < q[i] || p[i] < l[i]) return false;
43         return true;
44     }
45     LL query(P* o, int l[], int r[]) {
46         if (o == null) return 0;
47         FOR (i, 0, K) if (o->r[i] < l[i] || r[i] < o->l[i]) return 0;
48         if (inside(o->l, o->r, l, r)) return o->sum;
49         return query(o->ls, l, r) + query(o->rs, l, r) +
50                 (inside(o->d, o->d, l, r) ? o->val : 0);
51     }
52     void dfs(P* o) {
53         if (o == null) return;
54         *pt++ = o; dfs(o->ls); dfs(o->rs);
55     }
56     P* ins(P* o, P* x, int d = 0) {
57         if (d == K) d = 0;
58         if (o == null) return x->up();
59         P*& oo = x->d[d] <= o->d[d] ? o->ls : o->rs;
60         if (oo->sz > o->sz * lim) {
61             pt = tmp; dfs(o); *pt++ = x;
62             return build(tmp, pt, d);
63         }
```

```
64          oo = ins(oo, x, d + 1);
65          return o->up();
66      }
67  }
```

- 维护信息
- 带修改（适合离线）

```
1   namespace kd {
2       const int K = 3, inf = 1E9, M = N << 3;
3       extern struct P* null;
4       struct P {
5           int d[K], l[K], r[K], val;
6           int Max;
7           P *ls, *rs, *fa;
8           P* up() {
9               Max = max(val, max(ls->Max, rs->Max));
10              FOR (i, 0, K) {
11                  l[i] = min(d[i], min(ls->l[i], rs->l[i]));
12                  r[i] = max(d[i], max(ls->r[i], rs->r[i]));
13              }
14              return ls->fa = rs->fa = this;
15          }
16      } pool[M], *null = new P, *pit = pool;
17      void upd(P* o, int val) {
18          o->val = val;
19          for (; o != null; o = o->fa)
20              o->Max = max(o->Max, val);
21      }
22      static P *tmp[M], **pt;
23      void init() {
24          null->ls = null->rs = null;
25          FOR (i, 0, K) null->l[i] = inf, null->r[i] = -inf;
26          null->Max = null->val = 0;
27      }
28      P* build(P** l, P** r, int d = 0) { // [l, r)
29          if (d == K) d = 0;
30          if (l >= r) return null;
31          P** m = l + (r - l) / 2; assert(l <= m && m < r);
32          nth_element(l, m, r, [&](const P* a, const P* b){
33              return a->d[d] < b->d[d];
34          });
35          P* o = *m;
36          o->ls = build(l, m, d + 1); o->rs = build(m + 1, r, d + 1);
37          return o->up();
38      }
39      P* Build() {
40          pt = tmp; FOR (it, pool, pit) *pt++ = it;
41          P* ret = build(tmp, pt); ret->fa = null;
42          return ret;
43      }
44      inline bool inside(int p[], int q[], int l[], int r[]) {
45          FOR (i, 0, K) if (r[i] < q[i] || p[i] < l[i]) return false;
46          return true;
47      }
48      int query(P* o, int l[], int r[]) {
49          if (o == null) return 0;
50          FOR (i, 0, K) if (o->r[i] < l[i] || r[i] < o->l[i]) return 0;
51          if (inside(o->l, o->r, l, r)) return o->Max;
52          int ret = 0;
53          if (o->val > ret && inside(o->d, o->d, l, r)) ret = max(ret, o->val);
54          if (o->ls->Max > ret) ret = max(ret, query(o->ls, l, r));
55          if (o->rs->Max > ret) ret = max(ret, query(o->rs, l, r));
56          return ret;
57      }
58  }
```

- 最近点对
- 要用全局变量大力剪枝

```
1   namespace kd {
2       const int K = 3;
```

```cpp
        const int M = N;
        const int inf = 1E9 + 100;
        struct P {
            int d[K];
            int l[K], r[K];
            P *ls, *rs;
            P* up() {
                FOR (i, 0, K) {
                    l[i] = min(d[i], min(ls->l[i], rs->l[i]));
                    r[i] = max(d[i], max(ls->r[i], rs->r[i]));
                }
                return this;
            }
        } pool[M], *null = new P, *pit = pool;
        static P *tmp[M], **pt;
        void init() {
            null->ls = null->rs = null;
            FOR (i, 0, K) null->l[i] = inf, null->r[i] = -inf;
        }
        P* build(P** l, P** r, int d = 0) { // [l, r)
            if (d == K) d = 0;
            if (l >= r) return null;
            P** m = l + (r - l) / 2;
            nth_element(l, m, r, [&](const P* a, const P* b){
                return a->d[d] < b->d[d];
            });
            P* o = *m;
            o->ls = build(l, m, d + 1); o->rs = build(m + 1, r, d + 1);
            return o->up();
        }
        LL eval(P* o, int d[]) {
            // ...
        }
        LL dist(int d1[], int d2[]) {
            // ...
        }
        LL S;
        LL query(P* o, int d[]) {
            if (o == null) return 0;
            S = max(S, dist(o->d, d));
            LL mdl = eval(o->ls, d), mdr = eval(o->rs, d);
            if (mdl < mdr) {
                if (S > mdl) S = max(S, query(o->ls, d));
                if (S > mdr) S = max(S, query(o->rs, d));
            } else {
                if (S > mdr) S = max(S, query(o->rs, d));
                if (S > mdl) S = max(S, query(o->ls, d));
            }
            return S;
        }
        P* Build() {
            pt = tmp; FOR (it, pool, pit) *pt++ = it;
            return build(tmp, pt);
        }
    }
```

树状数组

- 注意：0 是无效下标

```cpp
namespace bit {
    LL c[M];
    inline int lowbit(int x) { return x & -x; }
    void add(int x, LL v) {
        for (; x < M; x += lowbit(x))
            c[x] += v;
    }
    LL sum(int x) {
        LL ret = 0;
```

```
10          for (; x > 0; x -= lowbit(x))
11              ret += c[x];
12          return ret;
13      }
14      int kth(LL k) {
15          int ret = 0;
16          LL cnt = 0;
17          FORD (i, 20, -1) {
18              ret += 1 << i;
19              if (ret >= M || cnt + c[ret] >= k)
20                  ret -= 1 << i;
21              else cnt += c[ret];
22          }
23          return ret + 1;
24      }
25  }
```

- 区间修改 & 区间查询（单点修改，查询前缀和的前缀和）

```
1   namespace bit {
2       int c[maxn], cc[maxn];
3       inline int lowbit(int x) { return x & -x; }
4       void add(int x, int v) {
5           for (int i = x; i <= n; i += lowbit(i)) {
6               c[i] += v; cc[i] += x * v;
7           }
8       }
9       void add(int l, int r, int v) { add(l, v); add(r + 1, -v); }
10      int sum(int x) {
11          int ret = 0;
12          for (int i = x; i > 0; i -= lowbit(i))
13              ret += (x + 1) * c[i] - cc[i];
14          return ret;
15      }
16      int sum(int l, int r) { return sum(r) - sum(l - 1); }
17  }
```

- 单点修改，查询前缀和的前缀和的前缀和（有用才怪）

```
1   namespace bit {
2       LL c[N], cc[N], ccc[N];
3       inline LL lowbit(LL x) { return x & -x; }
4       void add(LL x, LL v) {
5           for (LL i = x; i < N; i += lowbit(i)) {
6               c[i] = (c[i] + v) % MOD;
7               cc[i] = (cc[i] + x * v) % MOD;
8               ccc[i] = (ccc[i] + x * x % MOD * v) % MOD;
9           }
10      }
11      void add(LL l, LL r, LL v) { add(l, v); add(r + 1, -v); }
12      LL sum(LL x) {
13          static LL INV2 = (MOD + 1) / 2;
14          LL ret = 0;
15          for (LL i = x; i > 0; i -= lowbit(i))
16              ret += (x + 1) * (x + 2) % MOD * c[i] % MOD
17                     - (2 * x + 3) * cc[i] % MOD
18                     + ccc[i];
19          return ret % MOD * INV2 % MOD;
20      }
21      LL sum(LL l, LL r) { return sum(r) - sum(l - 1); }
22  }
```

- 三维

```
1   inline int lowbit(int x) { return x & -x; }
2   void update(int x, int y, int z, int d) {
3       for (int i = x; i <= n; i += lowbit(i))
4           for (int j = y; j <= n; j += lowbit(j))
5               for (int k = z; k <= n; k += lowbit(k))
6                   c[i][j][k] += d;
7   }
8   LL query(int x, int y, int z) {
9       LL ret = 0;
```

```cpp
10      for (int i = x; i > 0; i -= lowbit(i))
11          for (int j = y; j > 0; j -= lowbit(j))
12              for (int k = z; k > 0; k -= lowbit(k))
13                  ret += c[i][j][k];
14      return ret;
15  }
16  LL solve(int x, int y, int z, int xx, int yy, int zz) {
17      return   query(xx, yy, zz)
18             - query(xx, yy, z - 1)
19             - query(xx, y - 1, zz)
20             - query(x - 1, yy, zz)
21             + query(xx, y - 1, z - 1)
22             + query(x - 1, yy, z - 1)
23             + query(x - 1, y - 1, zz)
24             - query(x - 1, y - 1, z - 1);
```

主席树

- 正常主席树

```cpp
1   namespace tree {
2   #define mid ((l + r) >> 1)
3   #define lson l, mid
4   #define rson mid + 1, r
5       const int MAGIC = M * 30;
6       struct P {
7           int sum, ls, rs;
8       } tr[MAGIC] = {{0, 0, 0}};
9       int sz = 1;
10      int N(int sum, int ls, int rs) {
11          if (sz == MAGIC) assert(0);
12          tr[sz] = {sum, ls, rs};
13          return sz++;
14      }
15      int ins(int o, int x, int v, int l = 1, int r = ls) {
16          if (x < l || x > r) return o;
17          const P& t = tr[o];
18          if (l == r) return N(t.sum + v, 0, 0);
19          return N(t.sum + v, ins(t.ls, x, v, lson), ins(t.rs, x, v, rson));
20      }
21      int query(int o, int ql, int qr, int l = 1, int r = ls) {
22          if (ql > r || l > qr) return 0;
23          const P& t = tr[o];
24          if (ql <= l && r <= qr) return t.sum;
25          return query(t.ls, ql, qr, lson) + query(t.rs, ql, qr, rson);
26      }
27  }
```

- 第 k 大

```cpp
1   struct TREE {
2   #define mid ((l + r) >> 1)
3   #define lson l, mid
4   #define rson mid + 1, r
5       struct P {
6           int w, ls, rs;
7       } tr[maxn * 20];
8       int sz = 1;
9       TREE() { tr[0] = {0, 0, 0}; }
10      int N(int w, int ls, int rs) {
11          tr[sz] = {w, ls, rs};
12          return sz++;
13      }
14      int ins(int tt, int l, int r, int x) {
15          if (x < l || r < x) return tt;
16          const P& t = tr[tt];
17          if (l == r) return N(t.w + 1, 0, 0);
18          return N(t.w + 1, ins(t.ls, lson, x), ins(t.rs, rson, x));
19      }
20      int query(int pp, int qq, int l, int r, int k) { // (pp, qq]
```

```cpp
            if (l == r) return l;
            const P &p = tr[pp], &q = tr[qq];
            int w = tr[q.ls].w - tr[p.ls].w;
            if (k <= w) return query(p.ls, q.ls, lson, k);
            else return query(p.rs, q.rs, rson, k - w);
        }
} tree;
```

- 树状数组套主席树

```cpp
typedef vector<int> VI;
struct TREE {
#define mid ((l + r) >> 1)
#define lson l, mid
#define rson mid + 1, r
    struct P {
        int w, ls, rs;
    } tr[maxn * 20 * 20];
    int sz = 1;
    TREE() { tr[0] = {0, 0, 0}; }
    int N(int w, int ls, int rs) {
        tr[sz] = {w, ls, rs};
        return sz++;
    }
    int add(int tt, int l, int r, int x, int d) {
        if (x < l || r < x) return tt;
        const P& t = tr[tt];
        if (l == r) return N(t.w + d, 0, 0);
        return N(t.w + d, add(t.ls, lson, x, d), add(t.rs, rson, x, d));
    }
    int ls_sum(const VI& rt) {
        int ret = 0;
        FOR (i, 0, rt.size())
            ret += tr[tr[rt[i]].ls].w;
        return ret;
    }
    inline void ls(VI& rt) { transform(rt.begin(), rt.end(), rt.begin(), [&](int x)->int{ return tr[x].ls; }); }
    inline void rs(VI& rt) { transform(rt.begin(), rt.end(), rt.begin(), [&](int x)->int{ return tr[x].rs; }); }
    int query(VI& p, VI& q, int l, int r, int k) {
        if (l == r) return l;
        int w = ls_sum(q) - ls_sum(p);
        if (k <= w) {
            ls(p); ls(q);
            return query(p, q, lson, k);
        }
        else {
            rs(p); rs(q);
            return query(p, q, rson, k - w);
        }
    }
} tree;
struct BIT {
    int root[maxn];
    void init() { memset(root, 0, sizeof root); }
    inline int lowbit(int x) { return x & -x; }
    void update(int p, int x, int d) {
        for (int i = p; i <= m; i += lowbit(i))
            root[i] = tree.add(root[i], 1, m, x, d);
    }
    int query(int l, int r, int k) {
        VI p, q;
        for (int i = l - 1; i > 0; i -= lowbit(i)) p.push_back(root[i]);
        for (int i = r; i > 0; i -= lowbit(i)) q.push_back(root[i]);
        return tree.query(p, q, 1, m, k);
    }
} bit;

void init() {
    m = 10000;
    tree.sz = 1;
    bit.init();
    FOR (i, 1, m + 1)
```

```
63        bit.update(i, a[i], 1);
64    }
```

左偏树

```
1    namespace LTree {
2        extern struct P* null, *pit;
3        queue<P*> trash;
4        const int M = 1E5 + 100;
5        struct P {
6            P *ls, *rs;
7            LL v;
8            int d;
9            void operator delete (void* ptr) {
10               trash.push((P*)ptr);
11           }
12           void* operator new(size_t size) {
13               if (trash.empty()) return pit++;
14               void* ret = trash.front(); trash.pop(); return ret;
15           }
16
17           void prt() {
18               if (this == null) return;
19               cout << v << ' ';
20               ls->prt(); rs->prt();
21           }
22       } pool[M], *pit = pool, *null = new P{0, 0, -1, -1};
23       P* N(LL v) {
24           return new P{null, null, v, 0};
25       }
26       P* merge(P* a, P* b) {
27           if (a == null) return b;
28           if (b == null) return a;
29           if (a->v > b->v) swap(a, b);
30           a->rs = merge(a->rs, b);
31           if (a->ls->d < a->rs->d) swap(a->ls, a->rs);
32           a->d = a->rs->d + 1;
33           return a;
34       }
35
36       LL pop(P*& o) {
37           LL ret = o->v;
38           P* t = o;
39           o = merge(o->ls, o->rs);
40           delete t;
41           return ret;
42       }
43   }
```

可持久化

```
1    namespace LTree {
2        extern struct P* null, *pit;
3        queue<P*> trash;
4        const int M = 1E6 + 100;
5        struct P {
6            P *ls, *rs;
7            LL v;
8            int d;
9            void operator delete (void* ptr) {
10               trash.push((P*)ptr);
11           }
12           void* operator new(size_t size) {
13               if (trash.empty()) return pit++;
14               void* ret = trash.front(); trash.pop(); return ret;
15           }
16       } pool[M], *pit = pool, *null = new P{0, 0, -1, -1};
17       P* N(LL v, P* ls = null, P* rs = null) {
18           if (ls->d < rs->d) swap(ls, rs);
19           return new P{ls, rs, v, rs->d + 1};
```

```
20          }
21      P* merge(P* a, P* b) {
22          if (a == null) return b;
23          if (b == null) return a;
24          if (a->v < b->v)
25              return N(a->v, a->ls, merge(a->rs, b));
26          else
27              return N(b->v, b->ls, merge(b->rs, a));
28      }
29
30      LL pop(P*& o) {
31          LL ret = o->v;
32          o = merge(o->ls, o->rs);
33          return ret;
34      }
35  }
```

Treap

- 非旋 Treap
- v 小根堆
- 模板题 bzoj 3224
- lower 第一个大于等于的是第几个 (0-based)
- upper 第一个大于的是第几个 (0-based)
- split 左侧分割出 rk 个元素
- 树套树略

```
1  namespace treap {
2      const int M = maxn * 17;
3      extern struct P* const null;
4      struct P {
5          P *ls, *rs;
6          int v, sz;
7          unsigned rd;
8          P(int v): ls(null), rs(null), v(v), sz(1), rd(rnd()) {}
9          P(): sz(0) {}
10
11          P* up() { sz = ls->sz + rs->sz + 1; return this; }
12          int lower(int v) {
13              if (this == null) return 0;
14              return this->v >= v ? ls->lower(v) : rs->lower(v) + ls->sz + 1;
15          }
16          int upper(int v) {
17              if (this == null) return 0;
18              return this->v > v ? ls->upper(v) : rs->upper(v) + ls->sz + 1;
19          }
20      } *const null = new P, pool[M], *pit = pool;
21
22      P* merge(P* l, P* r) {
23          if (l == null) return r; if (r == null) return l;
24          if (l->rd < r->rd) { l->rs = merge(l->rs, r); return l->up(); }
25          else { r->ls = merge(l, r->ls); return r->up(); }
26      }
27
28      void split(P* o, int rk, P*& l, P*& r) {
29          if (o == null) { l = r = null; return; }
30          if (o->ls->sz >= rk) { split(o->ls, rk, l, o->ls); r = o->up(); }
31          else { split(o->rs, rk - o->ls->sz - 1, o->rs, r); l = o->up(); }
32      }
33  }
```

- 持久化 Treap

```
1  namespace treap {
2      const int M = maxn * 17 * 12;
3      extern struct P* const null, *pit;
4      struct P {
5          P *ls, *rs;
```

```cpp
        int v, sz;
        LL sum;
        P(P* ls, P* rs, int v): ls(ls), rs(rs), v(v), sz(ls->sz + rs->sz + 1),
                                                sum(ls->sum + rs->sum + v) {}
        P() {}

        void* operator new(size_t _) { return pit++; }
        template<typename T>
        int rk(int v, T&& cmp) {
            if (this == null) return 0;
            return cmp(this->v, v) ? ls->rk(v, cmp) : rs->rk(v, cmp) + ls->sz + 1;
        }
        int lower(int v) { return rk(v, greater_equal<int>()); }
        int upper(int v) { return rk(v, greater<int>()); }
    } pool[M], *pit = pool, *const null = new P;
    P* merge(P* l, P* r) {
        if (l == null) return r; if (r == null) return l;
        if (rnd() % (l->sz + r->sz) < l->sz) return new P{l->ls, merge(l->rs, r), l->v};
        else return new P{merge(l, r->ls), r->rs, r->v};
    }
    void split(P* o, int rk, P*& l, P*& r) {
        if (o == null) { l = r = null; return; }
        if (o->ls->sz >= rk) { split(o->ls, rk, l, r); r = new P{r, o->rs, o->v}; }
        else { split(o->rs, rk - o->ls->sz - 1, l, r); l = new P{o->ls, l, o->v}; }
    }
}
```

- 带 pushdown 的持久化 Treap
- 注意任何修改操作前一定要 FIX

```cpp
int now;
namespace Treap {
    const int M = 10000000;
    extern struct P* const null, *pit;
    struct P {
        P *ls, *rs;
        int sz, time;
        LL cnt, sc, pos, add;
        bool rev;

        P* up() { sz = ls->sz + rs->sz + 1; sc = ls->sc + rs->sc + cnt; return this; } // MOD
        P* check() {
            if (time == now) return this;
            P* t = new(pit++) P; *t = *this; t->time = now; return t;
        };
        P* _do_rev() { rev ^= 1; add *= -1; pos *= -1; swap(ls, rs); return this; } // MOD
        P* _do_add(LL v) { add += v; pos += v; return this; } // MOD
        P* do_rev() { if (this == null) return this; return check()->_do_rev(); } // FIX & MOD
        P* do_add(LL v) { if (this == null) return this; return check()->_do_add(v); } // FIX & MOD
        P* _down() { // MOD
            if (rev) { ls = ls->do_rev(); rs = rs->do_rev(); rev = 0; }
            if (add) { ls = ls->do_add(add); rs = rs->do_add(add); add = 0; }
            return this;
        }
        P* down() { return check()->_down(); } // FIX & MOD
        void _split(LL p, P*& l, P*& r) { // MOD
            if (pos >= p) { ls->split(p, l, r); ls = r; r = up(); }
            else          { rs->split(p, l, r); rs = l; l = up(); }
        }
        void split(LL p, P*& l, P*& r) { // FIX & MOD
            if (this == null) l = r = null;
            else down()->_split(p, l, r);
        }
    } pool[M], *pit = pool, *const null = new P;
    P* merge(P* a, P* b) {
        if (a == null) return b; if (b == null) return a;
        if (rand() % (a->sz + b->sz) < a->sz) { a = a->down(); a->rs = merge(a->rs, b); return a->up(); }
        else                                  { b = b->down(); b->ls = merge(a, b->ls); return b->up(); }
    }
}
```

Treap-序列

- 区间 ADD, SUM

```
1   namespace treap {
2       const int M = 8E5 + 100;
3       extern struct P*const null;
4       struct P {
5           P *ls, *rs;
6           int sz, val, add, sum;
7           P(int v, P* ls = null, P* rs = null): ls(ls), rs(rs), sz(1), val(v), add(0), sum(v) {}
8           P(): sz(0), val(0), add(0), sum(0) {}
9
10          P* up() {
11              assert(this != null);
12              sz = ls->sz + rs->sz + 1;
13              sum = ls->sum + rs->sum + val + add * sz;
14              return this;
15          }
16          void upd(int v) {
17              if (this == null) return;
18              add += v;
19              sum += sz * v;
20          }
21          P* down() {
22              if (add) {
23                  ls->upd(add); rs->upd(add);
24                  val += add;
25                  add = 0;
26              }
27              return this;
28          }
29
30          P* select(int rk) {
31              if (rk == ls->sz + 1) return this;
32              return ls->sz >= rk ? ls->select(rk) : rs->select(rk - ls->sz - 1);
33          }
34      } pool[M], *pit = pool, *const null = new P, *rt = null;
35
36      P* merge(P* a, P* b) {
37          if (a == null) return b->up();
38          if (b == null) return a->up();
39          if (rand() % (a->sz + b->sz) < a->sz) {
40              a->down()->rs = merge(a->rs, b);
41              return a->up();
42          } else {
43              b->down()->ls = merge(a, b->ls);
44              return b->up();
45          }
46      }
47
48      void split(P* o, int rk, P*& l, P*& r) {
49          if (o == null) { l = r = null; return; }
50          o->down();
51          if (o->ls->sz >= rk) {
52              split(o->ls, rk, l, o->ls);
53              r = o->up();
54          } else {
55              split(o->rs, rk - o->ls->sz - 1, o->rs, r);
56              l = o->up();
57          }
58      }
59
60      inline void insert(int k, int v) {
61          P *l, *r;
62          split(rt, k - 1, l, r);
63          rt = merge(merge(l, new (pit++) P(v)), r);
64      }
65
66      inline void erase(int k) {
67          P *l, *r, *_, *t;
```

```
68          split(rt, k - 1, l, t);
69          split(t, 1, _, r);
70          rt = merge(l, r);
71      }
72
73      P* build(int l, int r, int* a) {
74          if (l > r) return null;
75          if (l == r) return new(pit++) P(a[l]);
76          int m = (l + r) / 2;
77          return (new(pit++) P(a[m], build(l, m - 1, a), build(m + 1, r, a)))->up();
78      }
79  };
```

- 区间 REVERSE, ADD, MIN

```
1   namespace treap {
2       extern struct P*const null;
3       struct P {
4           P *ls, *rs;
5           int sz, v, add, m;
6           bool flip;
7           P(int v, P* ls = null, P* rs = null): ls(ls), rs(rs), sz(1), v(v), add(0), m(v), flip(0) {}
8           P(): sz(0), v(INF), m(INF) {}
9
10          void upd(int v) {
11              if (this == null) return;
12              add += v; m += v;
13          }
14          void rev() {
15              if (this == null) return;
16              swap(ls, rs);
17              flip ^= 1;
18          }
19          P* up() {
20              assert(this != null);
21              sz = ls->sz + rs->sz + 1;
22              m = min(min(ls->m, rs->m), v) + add;
23              return this;
24          }
25          P* down() {
26              if (add) {
27                  ls->upd(add); rs->upd(add);
28                  v += add;
29                  add = 0;
30              }
31              if (flip) {
32                  ls->rev(); rs->rev();
33                  flip = 0;
34              }
35              return this;
36          }
37
38          P* select(int k) {
39              if (ls->sz + 1 == k) return this;
40              if (ls->sz >= k) return ls->select(k);
41              return rs->select(k - ls->sz - 1);
42          }
43
44      } pool[M], *const null = new P, *pit = pool, *rt = null;
45
46      P* merge(P* a, P* b) {
47          if (a == null) return b;
48          if (b == null) return a;
49          if (rnd() % (a->sz + b->sz) < a->sz) {
50              a->down()->rs = merge(a->rs, b);
51              return a->up();
52          } else {
53              b->down()->ls = merge(a, b->ls);
54              return b->up();
55          }
56      }
57
```

```
58    void split(P* o, int k, P*& l, P*& r) {
59        if (o == null) { l = r = null; return; }
60        o->down();
61        if (o->ls->sz >= k) {
62            split(o->ls, k, l, o->ls);
63            r = o->up();
64        } else {
65            split(o->rs, k - o->ls->sz - 1, o->rs, r);
66            l = o->up();
67        }
68    }
69
70    P* build(int l, int r, int* v) {
71        if (l > r) return null;
72        int m = (l + r) >> 1;
73        return (new (pit++) P(v[m], build(l, m - 1, v), build(m + 1, r, v)))->up();
74    }
75
76    void go(int x, int y, void f(P*&)) {
77        P *l, *m, *r;
78        split(rt, y, l, r);
79        split(l, x - 1, l, m);
80        f(m);
81        rt = merge(merge(l, m), r);
82    }
83 }
84 using namespace treap;
85 int a[maxn], n, x, y, Q, v, k, d;
86 char s[100];
87
88 int main() {
89     cin >> n;
90     FOR (i, 1, n + 1) scanf("%d", &a[i]);
91     rt = build(1, n, a);
92     cin >> Q;
93     while (Q--) {
94         scanf("%s", s);
95         if (s[0] == 'A') {
96             scanf("%d%d%d", &x, &y, &v);
97             go(x, y, [](P*& o){ o->upd(v); });
98         } else if (s[0] == 'R' && s[3] == 'E') {
99             scanf("%d%d", &x, &y);
100            go(x, y, [](P*& o){ o->rev(); });
101        } else if (s[0] == 'R' && s[3] == 'O') {
102            scanf("%d%d%d", &x, &y, &d);
103            d %= y - x + 1;
104            go(x, y, [](P*& o){
105                P *l, *r;
106                split(o, o->sz - d, l, r);
107                o = merge(r, l);
108            });
109        } else if (s[0] == 'I') {
110            scanf("%d%d", &k, &v);
111            go(k + 1, k, [](P*& o){ o = new (pit++) P(v); });
112        } else if (s[0] == 'D') {
113            scanf("%d", &k);
114            go(k, k, [](P*& o){ o = null; });
115        } else if (s[0] == 'M') {
116            scanf("%d%d", &x, &y);
117            go(x, y, [](P*& o) {
118                printf("%d\n", o->m);
119            });
120        }
121    }
122 }
```

● 持久化

```
1 namespace treap {
2     struct P;
3     extern P*const null;
4     P* N(P* ls, P* rs, LL v, bool fill);
```

```
5      struct P {
6          P *const ls, *const rs;
7          const int sz, v;
8          const LL sum;
9          bool fill;
10         int cnt;
11
12         void split(int k, P*& l, P*& r) {
13             if (this == null) { l = r = null; return; }
14             if (ls->sz >= k) {
15                 ls->split(k, l, r);
16                 r = N(r, rs, v, fill);
17             } else {
18                 rs->split(k - ls->sz - fill, l, r);
19                 l = N(ls, l, v, fill);
20             }
21         }
22
23
24     } *const null = new P{0, 0, 0, 0, 0, 0, 1};
25
26     P* N(P* ls, P* rs, LL v, bool fill) {
27         ls->cnt++; rs->cnt++;
28         return new P{ls, rs, ls->sz + rs->sz + fill, v, ls->sum + rs->sum + v, fill, 1};
29     }
30
31     P* merge(P* a, P* b) {
32         if (a == null) return b;
33         if (b == null) return a;
34         if (rand() % (a->sz + b->sz) < a->sz)
35             return N(a->ls, merge(a->rs, b), a->v, a->fill);
36         else
37             return N(merge(a, b->ls), b->rs, b->v, b->fill);
38     }
39
40     void go(P* o, int x, int y, P*& l, P*& m, P*& r) {
41         o->split(y, l, r);
42         l->split(x - 1, l, m);
43     }
44 }
```

可回滚并查集

- 注意这个不是可持久化并查集
- 查找时不进行路径压缩
- 复杂度靠按秩合并解决

```
1  namespace uf {
2      int fa[maxn], sz[maxn];
3      int undo[maxn], top;
4      void init() { memset(fa, -1, sizeof fa); memset(sz, 0, sizeof sz); top = 0; }
5      int findset(int x) { while (fa[x] != -1) x = fa[x]; return x; }
6      bool join(int x, int y) {
7          x = findset(x); y = findset(y);
8          if (x == y) return false;
9          if (sz[x] > sz[y]) swap(x, y);
10         undo[top++] = x;
11         fa[x] = y;
12         sz[y] += sz[x] + 1;
13         return true;
14     }
15     inline int checkpoint() { return top; }
16     void rewind(int t) {
17         while (top > t) {
18             int x = undo[--top];
19             sz[fa[x]] -= sz[x] + 1;
20             fa[x] = -1;
21         }
```

```
22            }
23      }
```

舞蹈链

- 注意 link 的 y 的范围是 [1, n]

- 注意在某些情况下替换掉 memset

- 精确覆盖

```cpp
1   struct P {
2       P *L, *R, *U, *D;
3       int x, y;
4   };
5
6   const int INF = 1E9;
7
8   struct DLX {
9   #define TR(i, D, s) for (P* i = s->D; i != s; i = i->D)
10      static const int M = 2E5;
11      P pool[M], *h[M], *r[M], *pit;
12      int sz[M];
13      bool solved;
14      stack<int> ans;
15      void init(int n) {
16          pit = pool;
17          ++n;
18          solved = false;
19          while (!ans.empty()) ans.pop();
20          memset(r, 0, sizeof r);
21          memset(sz, 0, sizeof sz);
22          FOR (i, 0, n)
23              h[i] = new (pit++) P;
24          FOR (i, 0, n) {
25              h[i]->L = h[(i + n - 1) % n];
26              h[i]->R = h[(i + 1) % n];
27              h[i]->U = h[i]->D = h[i];
28              h[i]->y = i;
29          }
30      }
31
32      void link(int x, int y) {
33          sz[y]++;
34          auto p = new (pit++) P;
35          p->x = x; p->y = y;
36          p->U = h[y]->U; p->D = h[y];
37          p->D->U = p->U->D = p;
38          if (!r[x]) r[x] = p->L = p->R = p;
39          else {
40              p->L = r[x]; p->R = r[x]->R;
41              p->L->R = p->R->L = p;
42          }
43      }
44
45      void remove(P* p) {
46          p->L->R = p->R; p->R->L = p->L;
47          TR (i, D, p)
48              TR (j, R, i) {
49                  j->D->U = j->U; j->U->D = j->D;
50                  sz[j->y]--;
51              }
52      }
53
54      void recall(P* p) {
55          p->L->R = p->R->L = p;
56          TR (i, U, p)
57              TR (j, L, i) {
58                  j->D->U = j->U->D = j;
59                  sz[j->y]++;
```

```cpp
60              }
61          }
62
63      bool dfs(int d) {
64          if (solved) return true;
65          if (h[0]->R == h[0]) return solved = true;
66          int m = INF;
67          P* c;
68          TR (i, R, h[0])
69              if (sz[i->y] < m) { m = sz[i->y]; c = i; }
70          remove(c);
71          TR (i, D, c) {
72              ans.push(i->x);
73              TR (j, R, i) remove(h[j->y]);
74              if (dfs(d + 1)) return true;
75              TR (j, L, i) recall(h[j->y]);
76              ans.pop();
77          }
78          recall(c);
79          return false;
80      }
81  } dlx;
```

- 可重复覆盖

```cpp
1  struct P {
2      P *L, *R, *U, *D;
3      int x, y;
4  };
5
6  const int INF = 1E9;
7
8  struct DLX {
9  #define TR(i, D, s) for (P* i = s->D; i != s; i = i->D)
10     static const int M = 2E5;
11     P pool[M], *h[M], *r[M], *pit;
12     int sz[M], vis[M], ans, clk;
13     void init(int n) {
14         clk = 0;
15         ans = INF;
16         pit = pool;
17         ++n;
18         memset(r, 0, sizeof r);
19         memset(sz, 0, sizeof sz);
20         memset(vis, -1, sizeof vis);
21         FOR (i, 0, n)
22             h[i] = new (pit++) P;
23         FOR (i, 0, n) {
24             h[i]->L = h[(i + n - 1) % n];
25             h[i]->R = h[(i + 1) % n];
26             h[i]->U = h[i]->D = h[i];
27             h[i]->y = i;
28         }
29     }
30
31     void link(int x, int y) {
32         sz[y]++;
33         auto p = new (pit++) P;
34         p->x = x; p->y = y;
35         p->U = h[y]->U; p->D = h[y];
36         p->D->U = p->U->D = p;
37         if (!r[x]) r[x] = p->L = p->R = p;
38         else {
39             p->L = r[x]; p->R = r[x]->R;
40             p->L->R = p->R->L = p;
41         }
42     }
43
44     void remove(P* p) {
45         TR (i, D, p) {
46             i->L->R = i->R;
47             i->R->L = i->L;
```

```
48          }
49      }
50
51      void recall(P* p) {
52          TR (i, U, p)
53              i->L->R = i->R->L = i;
54      }
55
56      int eval() {
57          ++clk;
58          int ret = 0;
59          TR (i, R, h[0])
60              if (vis[i->y] != clk) {
61                  ++ret;
62                  vis[i->y] = clk;
63                  TR (j, D, i)
64                      TR (k, R, j)
65                          vis[k->y] = clk;
66              }
67          return ret;
68      }
69
70      void dfs(int d) {
71          if (h[0]->R == h[0]) { ans = min(ans, d); return; }
72          if (eval() + d >= ans) return;
73          P* c;
74          int m = INF;
75          TR (i, R, h[0])
76              if (sz[i->y] < m) { m = sz[i->y]; c = i; }
77          TR (i, D, c) {
78              remove(i);
79              TR (j, R, i) remove(j);
80              dfs(d + 1);
81              TR (j, L, i) recall(j);
82              recall(i);
83          }
84      }
85 } dlx;
```

CDQ 分治

```
1  const int maxn = 2E5 + 100;
2  struct P {
3      int x, y;
4      int* f;
5      bool d1, d2;
6  } a[maxn], b[maxn], c[maxn];
7  int f[maxn];
8
9  void go2(int l, int r) {
10     if (l + 1 == r) return;
11     int m = (l + r) >> 1;
12     go2(l, m); go2(m, r);
13     FOR (i, l, m) b[i].d2 = 0;
14     FOR (i, m, r) b[i].d2 = 1;
15     merge(b + l, b + m, b + m, b + r, c + l, [](const P& a, const P& b)->bool {
16             if (a.y != b.y) return a.y < b.y;
17             return a.d2 > b.d2;
18         });
19     int mx = -1;
20     FOR (i, l, r) {
21         if (c[i].d1 && c[i].d2) *c[i].f = max(*c[i].f, mx + 1);
22         if (!c[i].d1 && !c[i].d2) mx = max(mx, *c[i].f);
23     }
24     FOR (i, l, r) b[i] = c[i];
25 }
26
27 void go1(int l, int r) { // [l, r)
28     if (l + 1 == r) return;
```

```cpp
    int m = (l + r) >> 1;
    go1(l, m);
    FOR (i, l, m) a[i].d1 = 0;
    FOR (i, m, r) a[i].d1 = 1;
    copy(a + l, a + r, b + l);
    sort(b + l, b + r, [](const P& a, const P& b)->bool {
            if (a.x != b.x) return a.x < b.x;
            return a.d1 > b.d1;
        });
    go2(l, r);
    go1(m, r);
}
```

- k 维 LIS

```cpp
struct P {
    int v[K];
    LL f;
    bool d[K];
} o[N << 10];
P* a[K][N << 10];
int k;
void go(int now, int l, int r) {
    if (now == 0) {
        if (l + 1 == r) return;
        int m = (l + r) / 2;
        go(now, l, m);
        FOR (i, l, m) a[now][i]->d[now] = 0;
        FOR (i, m, r) a[now][i]->d[now] = 1;
        copy(a[now] + l, a[now] + r, a[now + 1] + l);
        sort(a[now + 1] + l, a[now + 1] + r, [now](const P* a, const P* b){
            if (a->v[now] != b->v[now]) return a->v[now] < b->v[now];
            return a->d[now] > b->d[now];
        });
        go(now + 1, l, r);
        go(now, m, r);
    } else {
        if (l + 1 == r) return;
        int m = (l + r) / 2;
        go(now, l, m); go(now, m, r);
        FOR (i, l, m) a[now][i]->d[now] = 0;
        FOR (i, m, r) a[now][i]->d[now] = 1;
        merge(a[now] + l, a[now] + m, a[now] + m, a[now] + r, a[now + 1] + l, [now](const P* a, const P* b){
            if (a->v[now] != b->v[now]) return a->v[now] < b->v[now];
            return a->d[now] > b->d[now];
        });
        copy(a[now + 1] + l, a[now + 1] + r, a[now] + l);
        if (now < k - 2) {
            go(now + 1, l, r);
        } else {
            LL sum = 0;
            FOR (i, l, r) {
                dbg(a[now][i]->v[0], a[now][i]->v[1], a[now][i]->f,
                        a[now][i]->d[0], a[now][i]->d[1]);
                int cnt = 0;
                FOR (j, 0, now + 1) cnt += a[now][i]->d[j];
                if (cnt == 0) {
                    sum += a[now][i]->f;
                } else if (cnt == now + 1) {
                    a[now][i]->f = (a[now][i]->f + sum) % MOD;
                }
            }
        }
    }
}
```

哈希表

- 必须初始化

- 备选素数 1572869, 3145739, 6291469, 12582917, 25165843, 50331653

```
1   const LL HASH_MOD=1572869;
2   LL key[HASH_MOD], val[HASH_MOD];
3   int head[HASH_MOD], next[HASH_MOD];
4   struct Hash {
5       int sz;
6       void init() {
7           memset(head, -1, sizeof head);
8           sz = 0;
9       }
10      LL insert(LL x, LL y) {
11          int k = x % HASH_MOD;
12          key[sz] = x;
13          val[sz] = y;
14          next[sz] = head[k];
15          head[k] = sz++;
16      }
17      LL find(LL x) {
18          int k = x % HASH_MOD;
19          for (int i = head[k]; i != -1; i = next[i])
20              if (key[i] == x)
21                  return val[i];
22          return -1;
23      }
24  };
```

笛卡尔树

```
1   void build(const vector<int>& a) {
2       static P *stack[M], *x, *last;
3       int p = 0;
4       FOR (i, 0, a.size()) {
5           x = new P(i + 1, a[i]);
6           last = null;
7           while (p && stack[p - 1]->v > x->v) {
8               stack[p - 1]->maintain();
9               last = stack[--p];
10          }
11          if (p) stack[p - 1]->rs = x;
12          x->ls = last;
13          stack[p++] = x;
14      }
15      while (p)
16          stack[--p]->maintain();
17      rt = stack[0];
18  }
```

```
1   void build() {
2       static int s[N], last;
3       int p = 0;
4       FOR (x, 1, n + 1) {
5           last = 0;
6           while (p && val[s[p - 1]] > val[x]) last = s[--p];
7           if (p) G[s[p - 1]][1] = x;
8           if (last) G[x][0] = last;
9           s[p++] = x;
10      }
11      rt = s[0];
12  }
```

Trie

- Trie 二进制版
- M 为二进制的位数
- 使用前必须初始化

```
1   struct Trie2 {
2       int ch[N * M][2], sz;
3       void init() {
4           memset(ch, 0, sizeof ch);
5           sz = 1;
6       }
7       void insert(LL x) {
8           int u = 0;
9           FORD (i, M, -1) {
10              bool b = x & (1LL << i);
11              if (!ch[u][b])
12                  ch[u][b] = sz++;
13              u = ch[u][b];
14          }
15      }
16  } trie;
```

pb_ds

- 优先队列

- binary_heap_tag

- pairing_heap_tag 支持修改

- thin_heap_tag 如果修改只有 increase 则较快，不支持 join

```
1   #include<ext/pb_ds/priority_queue.hpp>
2   template<typename _Tv,
3       typename Cmp_Fn = std::less<_Tv>,
4       typename Tag = pairing_heap_tag,
5       typename _Alloc = std::allocator<char> >
6   class priority_queue;
```

```
1   #include<ext/pb_ds/priority_queue.hpp>
2   using namespace __gnu_pbds;
3
4   typedef __gnu_pbds::priority_queue<LL, less<LL>, pairing_heap_tag> PQ;
5   __gnu_pbds::priority_queue<int, cmp, pairing_heap_tag>::point_iterator it;
6   PQ pq, pq2;
7
8   int main() {
9       auto it = pq.push(2);
10      pq.push(3);
11      assert(pq.top() == 3);
12      pq.modify(it, 4);
13      assert(pq.top() == 4);
14      pq2.push(5);
15      pq.join(pq2);
16      assert(pq.top() == 5);
17  }
```

- 树

- ov_tree_tag

- rb_tree_tag

- splay_tree_tag

- mapped: null_type 或 null_mapped_type（旧版本）为空

- Node_Update 为 tree_order_statistics_node_update 时才可以 find_by_order & order_of_key

- find_by_order 找 order + 1 小的元素（其实都是从 0 开始计数），或者有 order 个元素比它小的 key

- order_of_key 有多少个比 r_key 小的元素

- join & split

```
1   template<typename Key, typename Mapped, typename Cmp_Fn = std::less<Key>,
2      typename Tag = rb_tree_tag,
3      template<typename Node_CItr, typename Node_Itr,
4               typename Cmp_Fn_, typename _Alloc_>
5      class Node_Update = null_node_update,
6      typename _Alloc = std::allocator<char> >
7   class tree
8
9   #include <ext/pb_ds/assoc_container.hpp>
10  using namespace __gnu_pbds;
11  using Tree = tree<int, null_type, less<int>, rb_tree_tag, tree_order_statistics_node_update>;
12  Tree t;
```

- hash table

```
1   #include<ext/pb_ds/assoc_container.hpp>
2   #include<ext/pb_ds/hash_policy.hpp>
3   using namespace __gnu_pbds;
4
5   gp_hash_table<int, int> mp;
6   cc_hash_table<int, int> mp;
```

Link-Cut Tree

- 图中相邻的结点在伸展树中不一定是父子关系
- 遇事不决 make_root
- 跑左右儿子的时候不要忘记 down

```
1   namespace lct {
2       extern struct P *const null;
3       const int M = N;
4       struct P {
5           P *fa, *ls, *rs;
6           int v, maxv;
7           bool rev;
8
9           bool has_fa() { return fa->ls == this || fa->rs == this; }
10          bool d() { return fa->ls == this; }
11          P*& c(bool x) { return x ? ls : rs; }
12          void do_rev() {
13              if (this == null) return;
14              rev ^= 1;
15              swap(ls, rs);
16          }
17          P* up() {
18              maxv = max(v, max(ls->maxv, rs->maxv));
19              return this;
20          }
21          void down() {
22              if (rev) {
23                  rev = 0;
24                  ls->do_rev(); rs->do_rev();
25              }
26          }
27          void all_down() { if (has_fa()) fa->all_down(); down(); }
28      } *const null = new P{0, 0, 0, 0, 0, 0}, pool[M], *pit = pool;
29
30      void rot(P* o) {
31          bool dd = o->d();
32          P *f = o->fa, *t = o->c(!dd);
33          if (f->has_fa()) f->fa->c(f->d()) = o; o->fa = f->fa;
34          if (t != null) t->fa = f; f->c(dd) = t;
35          o->c(!dd) = f->up(); f->fa = o;
36      }
37      void splay(P* o) {
38          o->all_down();
39          while (o->has_fa()) {
40              if (o->fa->has_fa())
41                  rot(o->d() ^ o->fa->d() ? o : o->fa);
```

```
42            rot(o);
43        }
44        o->up();
45    }
46    void access(P* u, P* v = null) {
47        if (u == null) return;
48        splay(u); u->rs = v;
49        access(u->up()->fa, u);
50    }
51    void make_root(P* o) {
52        access(o); splay(o); o->do_rev();
53    }
54    void split(P* o, P* u) {
55        make_root(o); access(u); splay(u);
56    }
57    void link(P* u, P* v) {
58        make_root(u); u->fa = v;
59    }
60    void cut(P* u, P* v) {
61        split(u, v);
62        u->fa = v->ls = null; v->up();
63    }
64    bool adj(P* u, P* v) {
65        split(u, v);
66        return v->ls == u && u->ls == null && u->rs == null;
67    }
68    bool linked(P* u, P* v) {
69        split(u, v);
70        return u == v || u->fa != null;
71    }
72    P* findrt(P* o) {
73        access(o); splay(o);
74        while (o->ls != null) o = o->ls;
75        return o;
76    }
77    P* findfa(P* rt, P* u) {
78        split(rt, u);
79        u = u->ls;
80        while (u->rs != null) {
81            u = u->rs;
82            u->down();
83        }
84        return u;
85    }
86 }
```

- 维护子树大小

```
1  P* up() {
2      sz = ls->sz + rs->sz + _sz + 1;
3      return this;
4  }
5  void access(P* u, P* v = null) {
6      if (u == null) return;
7      splay(u);
8      u->_sz += u->rs->sz - v->sz;
9      u->rs = v;
10     access(u->up()->fa, u);
11 }
12 void link(P* u, P* v) {
13     split(u, v);
14     u->fa = v; v->_sz += u->sz;
15     v->up();
16 }
```

莫队

- [l, r)

```
1  while (l > q.l) mv(--l, 1);
```

```
2    while (r < q.r) mv(r++, 1);
3    while (l < q.l) mv(l++, -1);
4    while (r > q.r) mv(--r, -1);
```

- 树上莫队
- 注意初始状态 $u = v = 1$, flip(1)

```
1    struct Q {
2        int u, v, idx;
3        bool operator < (const Q& b) const {
4            const Q& a = *this;
5            return blk[a.u] < blk[b.u] || (blk[a.u] == blk[b.u] && in[a.v] < in[b.v]);
6        }
7    };
8
9    void dfs(int u = 1, int d = 0) {
10       static int S[maxn], sz = 0, blk_cnt = 0, clk = 0;
11       in[u] = clk++;
12       dep[u] = d;
13       int btm = sz;
14       for (int v: G[u]) {
15           if (v == fa[u]) continue;
16           fa[v] = u;
17           dfs(v, d + 1);
18           if (sz - btm >= B) {
19               while (sz > btm) blk[S[--sz]] = blk_cnt;
20               ++blk_cnt;
21           }
22       }
23       S[sz++] = u;
24       if (u == 1) while (sz) blk[S[--sz]] = blk_cnt - 1;
25   }
26
27   void flip(int k) {
28       dbg(k);
29       if (vis[k]) {
30           // ...
31       } else {
32           // ...
33       }
34       vis[k] ^= 1;
35   }
36
37   void go(int& k) {
38       if (bug == -1) {
39           if (vis[k] && !vis[fa[k]]) bug = k;
40           if (!vis[k] && vis[fa[k]]) bug = fa[k];
41       }
42       flip(k);
43       k = fa[k];
44   }
45
46   void mv(int a, int b) {
47       bug = -1;
48       if (vis[b]) bug = b;
49       if (dep[a] < dep[b]) swap(a, b);
50       while (dep[a] > dep[b]) go(a);
51       while (a != b) {
52           go(a); go(b);
53       }
54       go(a); go(bug);
55   }
56
57   for (Q& q: query) {
58       mv(u, q.u); u = q.u;
59       mv(v, q.v); v = q.v;
60       ans[q.idx] = Ans;
61   }
```

数学

矩阵运算

```cpp
struct Mat {
    static const LL M = 2;
    LL v[M][M];
    Mat() { memset(v, 0, sizeof v); }
    void eye() { FOR (i, 0, M) v[i][i] = 1; }
    LL* operator [] (LL x) { return v[x]; }
    const LL* operator [] (LL x) const { return v[x]; }
    Mat operator * (const Mat& B) {
        const Mat& A = *this;
        Mat ret;
        FOR (i, 0, M)
            FOR (j, 0, M)
                FOR (k, 0, M)
                    ret[i][j] = (ret[i][j] + A[i][k] * B[k][j]) % MOD;
        return ret;
    }
    Mat pow(LL n) const {
        Mat A = *this, ret; ret.eye();
        for (; n; n >>= 1, A = A * A)
            if (n & 1) ret = ret * A;
        return ret;
    }
    Mat operator + (const Mat& B) {
        const Mat& A = *this;
        Mat ret;
        FOR (i, 0, M)
            FOR (j, 0, M)
                ret[i][j] = (A[i][j] + B[i][j]) % MOD;
        return ret;
    }
    void prt() const {
        FOR (i, 0, M)
            FOR (j, 0, M)
                printf("%lld%c", (*this)[i][j], j == M - 1 ? '\n' : ' ');
    }
};
```

筛

- 线性筛

```cpp
const LL p_max = 1E6 + 100;
LL pr[p_max], p_sz;
void get_prime() {
    static bool vis[p_max];
    FOR (i, 2, p_max) {
        if (!vis[i]) pr[p_sz++] = i;
        FOR (j, 0, p_sz) {
            if (pr[j] * i >= p_max) break;
            vis[pr[j] * i] = 1;
            if (i % pr[j] == 0) break;
        }
    }
}
```

- 线性筛 + 欧拉函数

```cpp
const LL p_max = 1E5 + 100;
LL phi[p_max] = {-1, 1};
void get_phi() {
    static bool vis[p_max];
    static LL prime[p_max], p_sz, d;
    FOR (i, 2, p_max) {
        if (!vis[i]) {
```

```
8            prime[p_sz++] = i;
9            phi[i] = i - 1;
10        }
11        for (LL j = 0; j < p_sz && (d = i * prime[j]) < p_max; ++j) {
12            vis[d] = 1;
13            if (i % prime[j] == 0) {
14                phi[d] = phi[i] * prime[j];
15                break;
16            }
17            else phi[d] = phi[i] * (prime[j] - 1);
18        }
19    }
20 }
```

- 线性筛 + 莫比乌斯函数

```
1  const LL p_max = 1E5 + 100;
2  LL mu[p_max] = {-1, 1};
3  void get_mu() {
4      static bool vis[p_max];
5      static LL prime[p_max], p_sz, d;
6      mu[1] = 1;
7      FOR (i, 2, p_max) {
8          if (!vis[i]) {
9              prime[p_sz++] = i;
10             mu[i] = -1;
11         }
12         for (LL j = 0; j < p_sz && (d = i * prime[j]) < p_max; ++j) {
13             vis[d] = 1;
14             if (i % prime[j] == 0) {
15                 mu[d] = 0;
16                 break;
17             }
18             else mu[d] = -mu[i];
19         }
20     }
21 }
```

亚线性筛

min_25

```
1  namespace min25 {
2      const int M = 1E6 + 100;
3      LL B, N;
4
5      // g(x)
6      inline LL pg(LL x) { return 1; }
7      inline LL ph(LL x) { return x % MOD; }
8      // Sum[g(i),{x,2,x}]
9      inline LL psg(LL x) { return x % MOD - 1; }
10     inline LL psh(LL x) {
11         static LL inv2 = (MOD + 1) / 2;
12         x = x % MOD;
13         return x * (x + 1) % MOD * inv2 % MOD - 1;
14     }
15     // f(pp=p^k)
16     inline LL fpk(LL p, LL e, LL pp) { return (pp - pp / p) % MOD; }
17     // f(p) = fgh(g(p), h(p))
18     inline LL fgh(LL g, LL h) { return h - g; }
19
20     LL pr[M], pc, sg[M], sh[M];
21     void get_prime(LL n) {
22         static bool vis[M]; pc = 0;
23         FOR (i, 2, n + 1) {
24             if (!vis[i]) {
25                 pr[pc++] = i;
26                 sg[pc] = (sg[pc - 1] + pg(i)) % MOD;
27                 sh[pc] = (sh[pc - 1] + ph(i)) % MOD;
```

```
28                }
29            FOR (j, 0, pc) {
30                if (pr[j] * i > n) break;
31                vis[pr[j] * i] = 1;
32                if (i % pr[j] == 0) break;
33            }
34        }
35    }
36
37    LL w[M];
38    LL id1[M], id2[M], h[M], g[M];
39    inline LL id(LL x) { return x <= B ? id1[x] : id2[N / x]; }
40
41    LL go(LL x, LL k) {
42        if (x <= 1 || (k >= 0 && pr[k] > x)) return 0;
43        LL t = id(x);
44        LL ans = fgh((g[t] - sg[k + 1]), (h[t] - sh[k + 1]));
45        FOR (i, k + 1, pc) {
46            LL p = pr[i];
47            if (p * p > x) break;
48            ans -= fgh(pg(p), ph(p));
49            for (LL pp = p, e = 1; pp <= x; ++e, pp = pp * p)
50                ans += fpk(p, e, pp) * (1 + go(x / pp, i)) % MOD;
51        }
52        return ans % MOD;
53    }
54
55    LL solve(LL _N) {
56        N = _N;
57        B = sqrt(N + 0.5);
58        get_prime(B);
59        int sz = 0;
60        for (LL l = 1, v, r; l <= N; l = r + 1) {
61            v = N / l; r = N / v;
62            w[sz] = v; g[sz] = psg(v); h[sz] = psh(v);
63            if (v <= B) id1[v] = sz; else id2[r] = sz;
64            sz++;
65        }
66        FOR (k, 0, pc) {
67            LL p = pr[k];
68            FOR (i, 0, sz) {
69                LL v = w[i]; if (p * p > v) break;
70                LL t = id(v / p);
71                g[i] = (g[i] - (g[t] - sg[k]) * pg(p)) % MOD;
72                h[i] = (h[i] - (h[t] - sh[k]) * ph(p)) % MOD;
73            }
74        }
75        return (go(N, -1) % MOD + MOD + 1) % MOD;
76    }
77 }
```

杜教筛

```
1  namespace dujiao {
2      const int M = 5E6;
3      LL f[M] = {0, 1};
4      void init() {
5          static bool vis[M];
6          static LL pr[M], p_sz, d;
7          FOR (i, 2, M) {
8              if (!vis[i]) { pr[p_sz++] = i; f[i] = -1; }
9              FOR (j, 0, p_sz) {
10                 if ((d = pr[j] * i) >= M) break;
11                 vis[d] = 1;
12                 if (i % pr[j] == 0) {
13                     f[d] = 0;
14                     break;
15                 } else f[d] = -f[i];
16             }
17         }
```

```cpp
            FOR (i, 2, M) f[i] += f[i - 1];
    }
    inline LL s_fg(LL n) { return 1; }
    inline LL s_g(LL n) { return n; }

    LL N, rd[M];
    bool vis[M];
    LL go(LL n) {
        if (n < M) return f[n];
        LL id = N / n;
        if (vis[id]) return rd[id];
        vis[id] = true;
        LL& ret = rd[id] = s_fg(n);
        for (LL l = 2, v, r; l <= n; l = r + 1) {
            v = n / l; r = n / v;
            ret -= (s_g(r) - s_g(l - 1)) * go(v);
        }
        return ret;
    }
    LL solve(LL n) {
        N = n;
        memset(vis, 0, sizeof vis);
        return go(n);
    }
}
```

素数测试

- 前置：快速乘、快速幂
- int 范围内只需检查 2, 7, 61
- long long 范围 2, 325, 9375, 28178, 450775, 9780504, 1795265022
- 3E15 内 2, 2570940, 880937, 610386380, 4130785767
- 4E13 内 2, 2570940, 211991001, 3749873356
- http://miller-rabin.appspot.com/

```cpp
bool checkQ(LL a, LL n) {
    if (n == 2 || a >= n) return 1;
    if (n == 1 || !(n & 1)) return 0;
    LL d = n - 1;
    while (!(d & 1)) d >>= 1;
    LL t = bin(a, d, n);   // 不一定需要快速乘
    while (d != n - 1 && t != 1 && t != n - 1) {
        t = mul(t, t, n);
        d <<= 1;
    }
    return t == n - 1 || d & 1;
}

bool primeQ(LL n) {
    static vector<LL> t = {2, 325, 9375, 28178, 450775, 9780504, 1795265022};
    if (n <= 1) return false;
    for (LL k: t) if (!checkQ(k, n)) return false;
    return true;
}
```

线性递推

```cpp
// k 为 m 最高次数 且  a[m] == 1
namespace BerlekampMassey {
    inline void up(LL& a, LL b) { (a += b) %= MOD; }

    V mul(const V& a, const V& b, const V& m, int k) {
        V r; r.resize(2 * k - 1);
        FOR (i, 0, k)
            FOR (j, 0, k)
                up(r[i + j], a[i] * b[j]);
```

```
10          FORD (i, k - 2, - 1) {
11              FOR (j, 0, k)
12                  up(r[i + j], r[i + k] * m[j]);
13              r.pop_back();
14          }
15          return r;
16      }
17
18      V pow(LL n, const V& m) {
19          int k = (int)m.size() - 1; assert(m[k] == -1 || m[k] == MOD - 1);
20          V r(k), x(k); r[0] = x[1] = 1;
21          for (; n; n >>= 1, x = mul(x, x, m, k))
22              if (n & 1) r = mul(x, r, m, k);
23          return r;
24      }
25
26      LL go(const V& a, const V& x, LL n) {
27          // a: (-1, a1, a2, ..., ak).reverse
28          // x: x1, x2, ..., xk
29          // x[n] = sum[a[i]*x[n-i],{i,1,k}]
30          int k = (int)a.size() - 1;
31          if (n <= k) return x[n - 1];
32          V r = pow(n - 1, a);
33          LL ans = 0;
34          FOR (i, 0, k)
35              up(ans, r[i] * x[i]);
36          return ans;
37      }
38
39      V BM(const V& x) {
40          V a = {-1}, b = {233};
41          FOR (i, 1, x.size()) {
42              b.push_back(0);
43              LL d = 0, la = a.size(), lb = b.size();
44              FOR (j, 0, la) up(d, a[j] * x[i - la  + 1 + j]);
45              if (d == 0) continue;
46              V t; for (auto& v: b) t.push_back(d * v % MOD);
47              FOR (j, 0, a.size()) up(t[lb - 1 - j], a[la - 1 - j]);
48              if (lb > la) {
49                  b = a;
50                  LL inv = -get_inv(d, MOD);
51                  for (auto& v: b) v = v * inv % MOD;
52              }
53              a.swap(t);
54          }
55          for (auto& v: a) up(v, MOD);
56          return a;
57      }
58  }
```

扩展欧几里得

- 求 $ax + by = gcd(a, b)$ 的一组解
- 如果 a 和 b 互素，那么 x 是 a 在模 b 下的逆元
- 注意 x 和 y 可能是负数

```
1  LL ex_gcd(LL a, LL b, LL &x, LL &y) {
2      if (b == 0) {
3          x = 1;
4          y = 0;
5          return a;
6      }
7      LL ret = ex_gcd(b, a % b, y, x);
8      y -= a / b * x;
9      return ret;
10  }
```

类欧几里得

- $m = \lfloor \frac{an+b}{c} \rfloor$.
- $f(a,b,c,n) = \sum_{i=0}^{n} \lfloor \frac{ai+b}{c} \rfloor$: 当 $a \geq c$ or $b \geq c$ 时,$f(a,b,c,n) = (\frac{a}{c})n(n+1)/2 + (\frac{b}{c})(n+1) + f(a \bmod c, b \bmod c, c, n)$;否则 $f(a,b,c,n) = nm - f(c, c-b-1, a, m-1)$。
- $g(a,b,c,n) = \sum_{i=0}^{n} i\lfloor \frac{ai+b}{c} \rfloor$: 当 $a \geq c$ or $b \geq c$ 时,$g(a,b,c,n) = (\frac{a}{c})n(n+1)(2n+1)/6 + (\frac{b}{c})n(n+1)/2 + g(a \bmod c, b \bmod c, c, n)$;否则 $g(a,b,c,n) = \frac{1}{2}(n(n+1)m - f(c, c-b-1, a, m-1) - h(c, c-b-1, a, m-1))$。
- $h(a,b,c,n) = \sum_{i=0}^{n} \lfloor \frac{ai+b}{c} \rfloor^2$: 当 $a \geq c$ or $b \geq c$ 时,$h(a,b,c,n) = (\frac{a}{c})^2 n(n+1)(2n+1)/6 + (\frac{b}{c})^2(n+1) + (\frac{a}{c})(\frac{b}{c})n(n+1) + h(a \bmod c, b \bmod c, c, n) + 2(\frac{a}{c})g(a \bmod c, b \bmod c, c, n) + 2(\frac{b}{c})f(a \bmod c, b \bmod c, c, n)$;否则 $h(a,b,c,n) = nm(m+1) - 2g(c, c-b-1, a, m-1) - 2f(c, c-b-1, a, m-1) - f(a,b,c,n)$。

逆元

- $ax \equiv 1 \pmod{p}$

- 如果 p 不是素数,使用拓展欧几里得

- 模数是素数,求一个数的逆元

- 前置模板:快速幂

```
inline LL get_inv(LL x, LL p) { return bin(x, p - 2, p); }
```

- 预处理

$$1 - n$$

的逆元

```
LL inv[N] = {-1, 1};
void inv_init(LL n, LL p) {
    inv[1] = 1;
    FOR (i, 2, n)
        inv[i] = (p - p / i) * inv[p % i] % p;
}
```

- 预处理阶乘及其逆元

```
LL invf[M], fac[M] = {1};
void fac_inv_init(LL n, LL p) {
    FOR (i, 1, n)
        fac[i] = i * fac[i - 1] % p;
    invf[n - 1] = bin(fac[n - 1], p - 2, p);
    FORD (i, n - 2, -1)
        invf[i] = invf[i + 1] * (i + 1) % p;
}
```

组合数

- 如果数较小,模较大时使用逆元
- 前置模板:逆元-预处理阶乘及其逆元

```
inline LL C(LL n, LL m) { // n >= m >= 0
    return n < m || m < 0 ? 0 : fac[n] * invf[m] % MOD * invf[n - m] % MOD;
}
```

- 如果模数较小,数字较大,使用 Lucas 定理
- 前置模板可选 1:求组合数(如果使用阶乘逆元,需 fac_inv_init(MOD, MOD);)
- 前置模板可选 2:模数不固定下使用,无法单独使用。

```
LL C(LL n, LL m) { // m >= n >= 0
    if (m - n < n) n = m - n;
    if (n < 0) return 0;
    LL ret = 1;
    FOR (i, 1, n + 1)
        ret = ret * (m - n + i) % MOD * bin(i, MOD - 2, MOD) % MOD;
```

```
7        return ret;
8    }
```

```
1    LL Lucas(LL n, LL m) { // m >= n >= 0
2        return m ? C(n % MOD, m % MOD) * Lucas(n / MOD, m / MOD) % MOD : 1;
3    }
```

- 组合数预处理

```
1    LL C[M][M];
2    void init_C(int n) {
3        FOR (i, 0, n) {
4            C[i][0] = C[i][i] = 1;
5            FOR (j, 1, i)
6                C[i][j] = (C[i - 1][j] + C[i - 1][j - 1]) % MOD;
7        }
8    }
```

第二类斯特灵数

```
1    S[0][0] = 1;
2        FOR (i, 1, N)
3            FOR (j, 1, i + 1) S[i][j] = (S[i - 1][j - 1] + j * S[i - 1][j]) % MOD;
```

FFT & NTT & FWT

- NTT
- 前置：快速幂

```
1    LL wn[N << 2], rev[N << 2];
2    int NTT_init(int n_) {
3        int step = 0; int n = 1;
4        for ( ; n < n_; n <<= 1) ++step;
5        FOR (i, 1, n)
6            rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (step - 1));
7        int g = bin(G, (MOD - 1) / n, MOD);
8        wn[0] = 1;
9        for (int i = 1; i <= n; ++i)
10           wn[i] = wn[i - 1] * g % MOD;
11       return n;
12   }
13
14   void NTT(LL a[], int n, int f) {
15       FOR (i, 0, n) if (i < rev[i])
16           std::swap(a[i], a[rev[i]]);
17       for (int k = 1; k < n; k <<= 1) {
18           for (int i = 0; i < n; i += (k << 1)) {
19               int t = n / (k << 1);
20               FOR (j, 0, k) {
21                   LL w = f == 1 ? wn[t * j] : wn[n - t * j];
22                   LL x = a[i + j];
23                   LL y = a[i + j + k] * w % MOD;
24                   a[i + j] = (x + y) % MOD;
25                   a[i + j + k] = (x - y + MOD) % MOD;
26               }
27           }
28       }
29       if (f == -1) {
30           LL ninv = get_inv(n, MOD);
31           FOR (i, 0, n)
32               a[i] = a[i] * ninv % MOD;
33       }
34   }
```

- FFT
- n 需补成 2 的幂（n 必须超过 a 和 b 的最高指数之和）

```
1   typedef double LD;
2   const LD PI = acos(-1);
3   struct C {
4       LD r, i;
5       C(LD r = 0, LD i = 0): r(r), i(i) {}
6   };
7   C operator + (const C& a, const C& b) {
8       return C(a.r + b.r, a.i + b.i);
9   }
10  C operator - (const C& a, const C& b) {
11      return C(a.r - b.r, a.i - b.i);
12  }
13  C operator * (const C& a, const C& b) {
14      return C(a.r * b.r - a.i * b.i, a.r * b.i + a.i * b.r);
15  }
16
17  void FFT(C x[], int n, int p) {
18      for (int i = 0, t = 0; i < n; ++i) {
19          if (i > t) swap(x[i], x[t]);
20          for (int j = n >> 1; (t ^= j) < j; j >>= 1);
21      }
22      for (int h = 2; h <= n; h <<= 1) {
23          C wn(cos(p * 2 * PI / h), sin(p * 2 * PI / h));
24          for (int i = 0; i < n; i += h) {
25              C w(1, 0), u;
26              for (int j = i, k = h >> 1; j < i + k; ++j) {
27                  u = x[j + k] * w;
28                  x[j + k] = x[j] - u;
29                  x[j] = x[j] + u;
30                  w = w * wn;
31              }
32          }
33      }
34      if (p == -1)
35          FOR (i, 0, n)
36              x[i].r /= n;
37  }
38
39  void conv(C a[], C b[], int n) {
40      FFT(a, n, 1);
41      FFT(b, n, 1);
42      FOR (i, 0, n)
43          a[i] = a[i] * b[i];
44      FFT(a, n, -1);
45  }
```

- FWT

```
1   template<typename T>
2   void fwt(LL a[], int n, T f) {
3       for (int d = 1; d < n; d *= 2)
4           for (int i = 0, t = d * 2; i < n; i += t)
5               FOR (j, 0, d)
6                   f(a[i + j], a[i + j + d]);
7   }
8
9   void AND(LL& a, LL& b) { a += b; }
10  void OR(LL& a, LL& b) { b += a; }
11  void XOR (LL& a, LL& b) {
12      LL x = a, y = b;
13      a = (x + y) % MOD;
14      b = (x - y + MOD) % MOD;
15  }
```

simpson 自适应积分

```
1   LD simpson(LD l, LD r) {
2       LD c = (l + r) / 2;
3       return (f(l) + 4 * f(c) + f(r)) * (r - l) / 6;
4   }
```

```
5
6    LD asr(LD l, LD r, LD eps, LD S) {
7        LD m = (l + r) / 2;
8        LD L = simpson(l, m), R = simpson(m, r);
9        if (fabs(L + R - S) < 15 * eps) return L + R + (L + R - S) / 15;
10       return asr(l, m, eps / 2, L) + asr(m, r, eps / 2, R);
11   }
12
13   LD asr(LD l, LD r, LD eps) { return asr(l, r, eps, simpson(l, r)); }
```

- FWT

```
1    template<typename T>
2    void fwt(LL a[], int n, T f) {
3        for (int d = 1; d < n; d *= 2)
4            for (int i = 0, t = d * 2; i < n; i += t)
5                FOR (j, 0, d)
6                    f(a[i + j], a[i + j + d]);
7    }
8
9    auto f = [](LL& a, LL& b) { // xor
10           LL x = a, y = b;
11           a = (x + y) % MOD;
12           b = (x - y + MOD) % MOD;
13   };
```

快速乘

```
1    LL mul(LL a, LL b, LL m) {
2        LL ret = 0;
3        while (b) {
4            if (b & 1) {
5                ret += a;
6                if (ret >= m) ret -= m;
7            }
8            a += a;
9            if (a >= m) a -= m;
10           b >>= 1;
11       }
12       return ret;
13   }
```

- O(1)

```
1    LL mul(LL u, LL v, LL p) {
2        return (u * v - LL((long double) u * v / p) * p + p) % p;
3    }
```

快速幂

- 如果模数是素数，则可在函数体内加上 n %= MOD - 1; （费马小定理）。

```
1    LL bin(LL x, LL n, LL MOD) {
2        LL ret = MOD != 1;
3        for (x %= MOD; n; n >>= 1, x = x * x % MOD)
4            if (n & 1) ret = ret * x % MOD;
5        return ret;
6    }
```

- 防爆 LL
- 前置模板：快速乘

```
1    LL bin(LL x, LL n, LL MOD) {
2        LL ret = MOD != 1;
3        for (x %= MOD; n; n >>= 1, x = mul(x, x, MOD))
4            if (n & 1) ret = mul(ret, x, MOD);
5        return ret;
6    }
```

高斯消元

- n - 方程个数, m - 变量个数, a 是 n * (m + 1) 的增广矩阵, free 是否为自由变量

- 返回自由变量个数, -1 无解, -2 无整数解

- 浮点数版本

```cpp
typedef double LD;
const LD eps = 1E-10;
const int maxn = 2000 + 10;

int n, m;
LD a[maxn][maxn], x[maxn];
bool free_x[maxn];

inline int sgn(LD x) { return (x > eps) - (x < -eps); }


int guass(LD a[maxn][maxn], int n, int m) {
    memset(free_x, 1, sizeof free_x); memset(x, 0, sizeof x);
    int r = 0, c = 0;
    while (r < n && c < m) {
        int m_r = r;
        FOR (i, r + 1, n)
            if (fabs(a[i][c]) > fabs(a[m_r][c])) m_r = i;
        if (m_r != r)
            FOR (j, c, m + 1)
                swap(a[r][j], a[m_r][j]);
        if (!sgn(a[r][c])) {
            a[r][c] = 0;
            ++c;
            continue;
        }
        FOR (i, r + 1, n)
            if (a[i][c]) {
                LD t = a[i][c] / a[r][c];
                FOR (j, c, m + 1) a[i][j] -= a[r][j] * t;
            }
        ++r; ++c;
//        FOR (i, 0, n)
//            FOR (j, 0, m + 1)
//                printf("%.2f%c", a[i][j], j == _j - 1 ? '\n' : ' '); puts("");
    }
    FOR (i, r, n)
        if (sgn(a[i][m])) return -1;
    if (r < m) {
        FORD (i, r - 1, -1) {
            int f_cnt = 0, k = -1;
            FOR (j, 0, m)
                if (sgn(a[i][j]) && free_x[j]) {
                    ++f_cnt;
                    k = j;
                }
            if(f_cnt > 0) continue;
            LD s = a[i][m];
            FOR (j, 0, m)
                if (j != k) s -= a[i][j] * x[j];
            x[k] = s / a[i][k];
            free_x[k] = 0;
        }
        return m - r;
    }
    FORD (i, m - 1, -1) {
        LD s = a[i][m];
        FOR (j, i + 1, m)
            s -= a[i][j] * x[j];
        x[i] = s / a[i][i];
    }
    return 0;
}
```

- 数据

```
3 4
1 1 -2 2
2 -3 5 1
4 -1 1 5
5 0 -1 7
// many

3 4
1 1 -2 2
2 -3 5 1
4 -1 -1 5
5 0 -1 0 2
// no

3 4
1 1 -2 2
2 -3 5 1
4 -1 1 5
5 0 1 0 7
// one
```

质因数分解

- 前置模板：素数筛
- 带指数

```cpp
LL factor[30], f_sz, factor_exp[30];
void get_factor(LL x) {
    f_sz = 0;
    LL t = sqrt(x + 0.5);
    for (LL i = 0; pr[i] <= t; ++i)
        if (x % pr[i] == 0) {
            factor_exp[f_sz] = 0;
            while (x % pr[i] == 0) {
                x /= pr[i];
                ++factor_exp[f_sz];
            }
            factor[f_sz++] = pr[i];
        }
    if (x > 1) {
        factor_exp[f_sz] = 1;
        factor[f_sz++] = x;
    }
}
```

- 不带指数

```cpp
LL factor[30], f_sz;
void get_factor(LL x) {
    f_sz = 0;
    LL t = sqrt(x + 0.5);
    for (LL i = 0; pr[i] <= t; ++i)
        if (x % pr[i] == 0) {
            factor[f_sz++] = pr[i];
            while (x % pr[i] == 0) x /= pr[i];
        }
    if (x > 1) factor[f_sz++] = x;
}
```

原根

- 前置模板: 素数筛, 快速幂, 分解质因数
- 要求 p 为质数

```cpp
LL find_smallest_primitive_root(LL p) {
    get_factor(p - 1);
    FOR (i, 2, p) {
        bool flag = true;
        FOR (j, 0, f_sz)
            if (bin(i, (p - 1) / factor[j], p) == 1) {
                flag = false;
                break;
            }
        if (flag) return i;
    }
    assert(0); return -1;
}
```

公式

- 当 $x \geq \phi(p)$ 时有 $a^x \equiv a^{x \bmod \phi(p) + \phi(p)} \pmod{p}$

斐波那契数列性质

- $F_{a+b} = F_{a-1} \cdot F_b + F_a \cdot F_{b+1}$

常见生成函数

- $(1 + ax)^n = \sum_{k=0}^{n} \binom{n}{k} a^k x^k$
- $\dfrac{1 - x^{r+1}}{1 - x} = \sum_{k=0}^{n} x^k$
- $\dfrac{1}{1 - ax} = \sum_{k=0}^{\infty} a^k x^k$
- $\dfrac{1}{(1-x)^2} = \sum_{k=0}^{\infty} (k+1) x^k$
- $\dfrac{1}{(1-x)^n} = \sum_{k=0}^{\infty} \binom{n+k-1}{k} x^k$
- $e^x = \sum_{k=0}^{\infty} \dfrac{x^k}{k!}$
- $\ln(1 + x) = \sum_{k=0}^{\infty} \dfrac{(-1)^{k+1}}{k} x^k$

中国剩余定理

- 无解返回 -1
- 前置模板: 拓展欧几里得

```cpp
LL CRT(LL *m, LL *r, LL n) {
    if (!n) return 0;
    LL M = m[0], R = r[0], x, y, d;
    FOR (i, 1, n) {
        d = ex_gcd(M, m[i], x, y);
        if ((r[i] - R) % d) return -1;
        x = (r[i] - R) / d * x % (m[i] / d);
        R += x * M;
        M = M / d * m[i];
        R %= M;
    }
    return R >= 0 ? R : R + M;
}
```

伯努利数和等幂求和

- 预处理逆元
- 预处理组合数
- $\sum_{i=0}^{n} i^k = \frac{1}{k+1} \sum_{i=0}^{k} \binom{k+1}{i} B_{k+1-i}(n+1)^i$.
- 也可以 $\sum_{i=0}^{n} i^k = \frac{1}{k+1} \sum_{i=0}^{k} \binom{k+1}{i} B_{k+1-i}^+ n^i$。区别在于 $B_1^+ = 1/2$。(心态崩了)

```cpp
namespace Bernoulli {
    const int M = 100;
    LL inv[M] = {-1, 1};
    void inv_init(LL n, LL p) {
        FOR (i, 2, n)
            inv[i] = (p - p / i) * inv[p % i] % p;
    }

    LL C[M][M];
    void init_C(int n) {
        FOR (i, 0, n) {
            C[i][0] = C[i][i] = 1;
            FOR (j, 1, i)
                C[i][j] = (C[i - 1][j] + C[i - 1][j - 1]) % MOD;
        }
    }

    LL B[M] = {1};
    void init() {
        inv_init(M, MOD);
        init_C(M);
        FOR (i, 1, M - 1) {
            LL& s = B[i] = 0;
            FOR (j, 0, i)
                s += C[i + 1][j] * B[j] % MOD;
            s = (s % MOD * -inv[i + 1] % MOD + MOD) % MOD;
        }
    }

    LL p[M] = {1};
    LL go(LL n, LL k) {
        n %= MOD;
        if (k == 0) return n;
        FOR (i, 1, k + 2)
            p[i] = p[i - 1] * (n + 1) % MOD;
        LL ret = 0;
        FOR (i, 1, k + 2)
            ret += C[k + 1][i] * B[k + 1 - i] % MOD * p[i] % MOD;
        ret = ret % MOD * inv[k + 1] % MOD;
        return ret;
    }
}
```

单纯形

- 要求有基本解，也就是 x 为零向量可行
- v 要初始化为 0，n 表示向量长度，m 表示约束个数

```cpp
// min{ b x } / max { c x }
// A x >= c   / A x <= b
// x >= 0
namespace lp {
    int n, m;
    double a[M][N], b[M], c[N], v;

    void pivot(int l, int e) {
        b[l] /= a[l][e];
        FOR (j, 0, n) if (j != e) a[l][j] /= a[l][e];
        a[l][e] = 1 / a[l][e];

        FOR (i, 0, m)
```

```
14                if (i != l && fabs(a[i][e]) > 0) {
15                    b[i] -= a[i][e] * b[l];
16                    FOR (j, 0, n)
17                        if (j != e) a[i][j] -= a[i][e] * a[l][j];
18                    a[i][e] = -a[i][e] * a[l][e];
19                }
20            v += c[e] * b[l];
21            FOR (j, 0, n) if (j != e) c[j] -= c[e] * a[l][j];
22            c[e] = -c[e] * a[l][e];
23        }
24        double simplex() {
25            while (1) {
26                int e = -1, l = -1;
27                FOR (i, 0, n) if (c[i] > eps) { e = i; break; }
28                if (e == -1) return v;
29                double t = INF;
30                FOR (i, 0, m)
31                    if (a[i][e] > eps && t > b[i] / a[i][e]) {
32                        t = b[i] / a[i][e];
33                        l = i;
34                    }
35                if (l == -1) return INF;
36                pivot(l, e);
37            }
38        }
39    }
```

图论

LCA

- 倍增

```
1    void dfs(int u, int fa) {
2        pa[u][0] = fa; dep[u] = dep[fa] + 1;
3        FOR (i, 1, SP) pa[u][i] = pa[pa[u][i - 1]][i - 1];
4        for (int& v: G[u]) {
5            if (v == fa) continue;
6            dfs(v, u);
7        }
8    }
9
10   int lca(int u, int v) {
11       if (dep[u] < dep[v]) swap(u, v);
12       int t = dep[u] - dep[v];
13       FOR (i, 0, SP) if (t & (1 << i)) u = pa[u][i];
14       FORD (i, SP - 1, -1) {
15           int uu = pa[u][i], vv = pa[v][i];
16           if (uu != vv) { u = uu; v = vv; }
17       }
18       return u == v ? u : pa[u][0];
19   }
```

最短路

```
1    bool BF() {
2        queue<int> q;
3        FOR (i, 1, n) d[i] = INF;
4        d[0] = 0; inq[0] = true; q.push(0);
5        while (!q.empty()) {
6            int u = q.front(); q.pop();
7            inq[u] = false;
8            for (E& e: G[u]) {
9                int v = e.to;
10               if (d[u] < INF && d[v] > d[u] + e.d) {
11                   d[v] = d[u] + e.d;
```

```
12                if (!inq[v]) {
13                    q.push(v); inq[v] = true;
14                    if (++cnt[v] > n) return false;
15                }
16            }
17        }
18    }
19    return true;
20 }
```

网络流

- 最大流

```
1  struct E {
2      int to, cp;
3      E(int to, int cp): to(to), cp(cp) {}
4  };
5
6  struct Dinic {
7      static const int M = 1E5 * 5;
8      int m, s, t;
9      vector<E> edges;
10     vector<int> G[M];
11     int d[M];
12     int cur[M];
13
14     void init(int n, int s, int t) {
15         this->s = s; this->t = t;
16         for (int i = 0; i <= n; i++) G[i].clear();
17         edges.clear(); m = 0;
18     }
19
20     void addedge(int u, int v, int cap) {
21         edges.emplace_back(v, cap);
22         edges.emplace_back(u, 0);
23         G[u].push_back(m++);
24         G[v].push_back(m++);
25     }
26
27     bool BFS() {
28         memset(d, 0, sizeof d);
29         queue<int> Q;
30         Q.push(s); d[s] = 1;
31         while (!Q.empty()) {
32             int x = Q.front(); Q.pop();
33             for (int& i: G[x]) {
34                 E &e = edges[i];
35                 if (!d[e.to] && e.cp > 0) {
36                     d[e.to] = d[x] + 1;
37                     Q.push(e.to);
38                 }
39             }
40         }
41         return d[t];
42     }
43
44     int DFS(int u, int cp) {
45         if (u == t || !cp) return cp;
46         int tmp = cp, f;
47         for (int& i = cur[u]; i < G[u].size(); i++) {
48             E& e = edges[G[u][i]];
49             if (d[u] + 1 == d[e.to]) {
50                 f = DFS(e.to, min(cp, e.cp));
51                 e.cp -= f;
52                 edges[G[u][i] ^ 1].cp += f;
53                 cp -= f;
54                 if (!cp) break;
55             }
```

```
56              }
57              return tmp - cp;
58          }
59
60      int go() {
61          int flow = 0;
62          while (BFS()) {
63              memset(cur, 0, sizeof cur);
64              flow += DFS(s, INF);
65          }
66          return flow;
67      }
68  } DC;
```

● 费用流

```
1   struct E {
2       int from, to, cp, v;
3       E() {}
4       E(int f, int t, int cp, int v) : from(f), to(t), cp(cp), v(v) {}
5   };
6
7   struct MCMF {
8       int n, m, s, t;
9       vector<E> edges;
10      vector<int> G[maxn];
11      bool inq[maxn];        //是否在队列
12      int d[maxn];           //Bellman_ford 单源最短路径
13      int p[maxn];           //p[i] 表从 s 到 i 的最小费用路径上的最后一条弧编号
14      int a[maxn];           //a[i] 表示从 s 到 i 的最小残量
15
16      void init(int _n, int _s, int _t) {
17          n = _n; s = _s; t = _t;
18          FOR (i, 0, n + 1) G[i].clear();
19          edges.clear(); m = 0;
20      }
21
22      void addedge(int from, int to, int cap, int cost) {
23          edges.emplace_back(from, to, cap, cost);
24          edges.emplace_back(to, from, 0, -cost);
25          G[from].push_back(m++);
26          G[to].push_back(m++);
27      }
28
29      bool BellmanFord(int &flow, int &cost) {
30          FOR (i, 0, n + 1) d[i] = INF;
31          memset(inq, 0, sizeof inq);
32          d[s] = 0, a[s] = INF, inq[s] = true;
33          queue<int> Q; Q.push(s);
34          while (!Q.empty()) {
35              int u = Q.front(); Q.pop();
36              inq[u] = false;
37              for (int& idx: G[u]) {
38                  E &e = edges[idx];
39                  if (e.cp && d[e.to] > d[u] + e.v) {
40                      d[e.to] = d[u] + e.v;
41                      p[e.to] = idx;
42                      a[e.to] = min(a[u], e.cp);
43                      if (!inq[e.to]) {
44                          Q.push(e.to);
45                          inq[e.to] = true;
46                      }
47                  }
48              }
49          }
50          if (d[t] == INF) return false;
51          flow += a[t];
52          cost += a[t] * d[t];
53          int u = t;
54          while (u != s) {
55              edges[p[u]].cp -= a[t];
56              edges[p[u] ^ 1].cp += a[t];
```

```
57              u = edges[p[u]].from;
58          }
59          return true;
60      }
61
62      int go() {
63          int flow = 0, cost = 0;
64          while (BellmanFord(flow, cost));
65          return cost;
66      }
67  } MM;
```

- zkw 费用流（代码长度没有优势）
- 不允许有负权边

```
1   struct E {
2       int to, cp, v;
3       E() {}
4       E(int to, int cp, int v): to(to), cp(cp), v(v) {}
5   };
6
7   struct MCMF {
8       int n, m, s, t, cost, D;
9       vector<E> edges;
10      vector<int> G[maxn];
11      bool vis[maxn];
12
13      void init(int _n, int _s, int _t) {
14          n = _n; s = _s; t = _t;
15          FOR (i, 0, n + 1) G[i].clear();
16          edges.clear(); m = 0;
17      }
18
19      void addedge(int from, int to, int cap, int cost) {
20          edges.emplace_back(to, cap, cost);
21          edges.emplace_back(from, 0, -cost);
22          G[from].push_back(m++);
23          G[to].push_back(m++);
24      }
25
26      int aug(int u, int cp) {
27          if (u == t) {
28              cost += D * cp;
29              return cp;
30          }
31          vis[u] = true;
32          int tmp = cp;
33          for (int idx: G[u]) {
34              E& e = edges[idx];
35              if (e.cp && !e.v && !vis[e.to]) {
36                  int f = aug(e.to, min(cp, e.cp));
37                  e.cp -= f;
38                  edges[idx ^ 1].cp += f;
39                  cp -= f;
40                  if (!cp) break;
41              }
42          }
43          return tmp - cp;
44      }
45
46      bool modlabel() {
47          int d = INF;
48          FOR (u, 0, n + 1)
49              if (vis[u])
50                  for (int& idx: G[u]) {
51                      E& e = edges[idx];
52                      if (e.cp && !vis[e.to]) d = min(d, e.v);
53                  }
54          if (d == INF) return false;
55          FOR (u, 0, n + 1)
56              if (vis[u])
```

```
57              for (int& idx: G[u]) {
58                  edges[idx].v -= d;
59                  edges[idx ^ 1].v += d;
60              }
61          D += d;
62          return true;
63      }
64
65      int go(int k) {
66          cost = D = 0;
67          int flow = 0;
68          while (true) {
69              memset(vis, 0, sizeof vis);
70              int t = aug(s, INF);
71              if (!t && !modlabel()) break;
72              flow += t;
73          }
74          return cost;
75      }
76  } MM;
```

树上路径交

```
1   int intersection(int x, int y, int xx, int yy) {
2       int t[4] = {lca(x, xx), lca(x, yy), lca(y, xx), lca(y, yy)};
3       sort(t, t + 4);
4       int r = lca(x, y), rr = lca(xx, yy);
5       if (dep[t[0]] < min(dep[r], dep[rr]) || dep[t[2]] < max(dep[r], dep[rr]))
6           return 0;
7       int tt = lca(t[2], t[3]);
8       int ret = 1 + dep[t[2]] + dep[t[3]] - dep[tt] * 2;
9       return ret;
10  }
```

树上点分治

```
1   int get_sz(int u, int fa) {
2       int& s = sz[u] = 1;
3       for (E& e: G[u]) {
4           int v = e.to;
5           if (vis[v] || v == fa) continue;
6           s += get_sz(v, u);
7       }
8       return s;
9   }
10
11  void get_rt(int u, int fa, int s, int& m, int& rt) {
12      int t = s - sz[u];
13      for (E& e: G[u]) {
14          int v = e.to;
15          if (vis[v] || v == fa) continue;
16          get_rt(v, u, s, m, rt);
17          t = max(t, sz[v]);
18      }
19      if (t < m) { m = t; rt = u; }
20  }
21
22  void dfs(int u) {
23      int tmp = INF; get_rt(u, -1, get_sz(u, -1), tmp, u);
24      vis[u] = true;
25      get_dep(u, -1, 0);
26      // ...
27      for (E& e: G[u]) {
28          int v = e.to;
29          if (vis[v]) continue;
30          // ...
31          dfs(v);
```

```
32          }
33      }
```

● 动态点分治

```
1   const int maxn = 15E4 + 100, INF = 1E9;
2   struct E {
3       int to, d;
4   };
5   vector<E> G[maxn];
6   int n, Q, w[maxn];
7   LL A, ans;
8
9   bool vis[maxn];
10  int sz[maxn];
11
12  int get_rt(int u) {
13  //    dbg(u);
14      static int q[N], fa[N], sz[N], mx[N];
15      int p = 0, cur = -1;
16      q[p++] = u; fa[u] = -1;
17      while (++cur < p) {
18          u = q[cur]; mx[u] = 0; sz[u] = 1;
19          for (int& v: G[u])
20              if (!vis[v] && v != fa[u]) fa[q[p++] = v] = u;
21      }
22      FORD (i, p - 1, -1) {
23          u = q[i];
24          mx[u] = max(mx[u], p - sz[u]);
25          if (mx[u] * 2 <= p) return u;
26          sz[fa[u]] += sz[u];
27          mx[fa[u]] = max(mx[fa[u]], sz[u]);
28      }
29      assert(0);
30  }
31
32  int get_sz(int u, int fa) {
33      int& s = sz[u] = 1;
34      for (E& e: G[u]) {
35          int v = e.to;
36          if (vis[v] || v == fa) continue;
37          s += get_sz(v, u);
38      }
39      return s;
40  }
41
42  void get_rt(int u, int fa, int s, int& m, int& rt) {
43      int t = s - sz[u];
44      for (E& e: G[u]) {
45          int v = e.to;
46          if (vis[v] || v == fa) continue;
47          get_rt(v, u, s, m, rt);
48          t = max(t, sz[v]);
49      }
50      if (t < m) { m = t; rt = u; }
51  }
52
53  int dep[maxn], md[maxn];
54  void get_dep(int u, int fa, int d) {
55      dep[u] = d; md[u] = 0;
56      for (E& e: G[u]) {
57          int v = e.to;
58          if (vis[v] || v == fa) continue;
59          get_dep(v, u, d + e.d);
60          md[u] = max(md[u], md[v] + 1);
61      }
62  }
63
64  struct P {
65      int w;
66      LL s;
67  };
```

49

```cpp
using VP = vector<P>;
struct R {
    VP *rt, *rt2;
    int dep;
};
VP pool[maxn << 1], *pit = pool;
vector<R> tr[maxn];

void go(int u, int fa, VP* rt, VP* rt2) {
    tr[u].push_back({rt, rt2, dep[u]});
    for (E& e: G[u]) {
        int v = e.to;
        if (v == fa || vis[v]) continue;
        go(v, u, rt, rt2);
    }
}

void dfs(int u) {
    int tmp = INF; get_rt(u, -1, get_sz(u, -1), tmp, u);
    vis[u] = true;
    get_dep(u, -1, 0);
    VP* rt = pit++; tr[u].push_back({rt, nullptr, 0});
    for (E& e: G[u]) {
        int v = e.to;
        if (vis[v]) continue;
        go(v, u, rt, pit++);
        dfs(v);
    }
}

bool cmp(const P& a, const P& b) { return a.w < b.w; }

LL query(VP& p, int d, int l, int r) {
    l = lower_bound(p.begin(), p.end(), P{l, -1}, cmp) - p.begin();
    r = upper_bound(p.begin(), p.end(), P{r, -1}, cmp) - p.begin() - 1;
    return p[r].s - p[l - 1].s + 1LL * (r - l + 1) * d;
}

int main() {
    cin >> n >> Q >> A;
    FOR (i, 1, n + 1) scanf("%d", &w[i]);
    FOR (_, 1, n) {
        int u, v, d; scanf("%d%d%d", &u, &v, &d);
        G[u].push_back({v, d}); G[v].push_back({u, d});
    }
    dfs(1);
    FOR (i, 1, n + 1)
        for (R& x: tr[i]) {
            x.rt->push_back({w[i], x.dep});
            if (x.rt2) x.rt2->push_back({w[i], x.dep});
        }
    FOR (it, pool, pit) {
        it->push_back({-INF, 0});
        sort(it->begin(), it->end(), cmp);
        FOR (i, 1, it->size())
            (*it)[i].s += (*it)[i - 1].s;
    }
    while (Q--) {
        int u; LL a, b; scanf("%d%lld%lld", &u, &a, &b);
        a = (a + ans) % A; b = (b + ans) % A;
        int l = min(a, b), r = max(a, b);
        ans = 0;
        for (R& x: tr[u]) {
            ans += query(*(x.rt), x.dep, l, r);
            if (x.rt2) ans -= query(*(x.rt2), x.dep, l, r);
        }
        printf("%lld\n", ans);
    }
}
```

树链剖分

```cpp
int fa[N], dep[N], idx[N], out[N], ridx[N];
namespace hld {
    int sz[N], son[N], top[N], clk;
    void predfs(int u, int d) {
        dep[u] = d; sz[u] = 1;
        int& maxs = son[u] = -1;
        for (int& v: G[u]) {
            if (v == fa[u]) continue;
            fa[v] = u;
            predfs(v, d + 1);
            sz[u] += sz[v];
            if (maxs == -1 || sz[v] > sz[maxs]) maxs = v;
        }
    }
    void dfs(int u, int tp) {
        top[u] = tp; idx[u] = ++clk; ridx[clk] = u;
        if (son[u] != -1) dfs(son[u], tp);
        for (int& v: G[u])
            if (v != fa[u] && v != son[u]) dfs(v, v);
        out[u] = clk;
    }
    template<typename T>
    int go(int u, int v, T&& f = [](int, int) {}) {
        int uu = top[u], vv = top[v];
        while (uu != vv) {
            if (dep[uu] < dep[vv]) { swap(uu, vv); swap(u, v); }
            f(idx[uu], idx[u]);
            u = fa[uu]; uu = top[u];
        }
        if (dep[u] < dep[v]) swap(u, v);
        // f(idx[v], idx[u]);
        // if (u != v) f(idx[v] + 1, idx[u]);
        return v;
    }
    int up(int u, int d) {
        while (d) {
            if (dep[u] - dep[top[u]] < d) {
                d -= dep[u] - dep[top[u]];
                u = top[u];
            } else return ridx[idx[u] - d];
            u = fa[u]; --d;
        }
        return u;
    }
}
```

- HDU 3966

```cpp
# include <bits/stdc++.h>
using namespace std;
typedef long long LL;
#define FOR(i, x, y) for (decay<decltype(y)>::type i = (x), _##i = (y); i < _##i; ++i)
#define FORD(i, x, y) for (decay<decltype(x)>::type i = (x), _##i = (y); i > _##i; --i)
#ifdef zerol
#define dbg(args...) do { cout << "\033[32;1m" << #args<< " -> "; err(args); } while (0)
#else
#define dbg(...)
#endif
void err() { cout << "\033[39;0m" << endl; }
template<typename T, typename... Args>
void err(T a, Args... args) { cout << a << ' '; err(args...); }
// -------------------------------------------------------------------
const int maxn = 5E4 + 100;
vector<int> G[maxn];
int dep[maxn], sz[maxn], son[maxn], fa[maxn], idx[maxn], top[maxn];
int clk, n, Q;

struct IntervalTree {
#define ls o * 2, l, (l + r) >> 1
```

```
22  #define rs o * 2 + 1, ((l + r) >> 1) + 1, r
23      static const int M = maxn << 2;
24      int addv[M];
25      void init() { memset(addv, 0, sizeof addv); }
26      int query(int k, int o, int l, int r, int add = 0) {
27          if (k < l || r < k) return 0;
28          if (l == r) return add + addv[o];
29          return query(k, ls, add + addv[o]) + query(k, rs, add + addv[o]);
30      }
31      void update(int p, int q, int o, int l, int r, int add) {
32          assert(l <= r && r <= n);
33          if (q < l || r < p) return;
34          if (p <= l && r <= q) addv[o] += add;
35          else { update(p, q, ls, add); update(p, q, rs, add); }
36      }
37  } IT;
38
39  void predfs(int u, int d) {
40      dep[u] = d;
41      sz[u] = 1;
42      int& maxs = son[u] = -1;
43      for (int v: G[u])
44          if (v != fa[u]) {
45              fa[v] = u;
46              predfs(v, d + 1);
47              sz[u] += sz[v];
48              if (maxs == -1 || sz[v] > sz[maxs])
49                  maxs = v;
50          }
51  }
52
53  void dfs(int u, int tp) {
54      top[u] = tp;
55      idx[u] = ++clk;
56      if (son[u] != -1) dfs(son[u], tp);
57      for (int v: G[u])
58          if (v != son[u] && v != fa[u])
59              dfs(v, v);
60  }
61
62  void update(int u, int v, int add) {
63      int uu = top[u], vv = top[v];
64      while (uu != vv) {
65          if (dep[uu] < dep[vv]) { swap(uu, vv); swap(u, v); }
66          IT.update(idx[uu], idx[u], 1, 1, n, add);
67          u = fa[uu];
68          uu = top[u];
69      }
70      if (dep[u] < dep[v]) swap(u, v);
71      dbg(u, v, idx[u], idx[v]);
72      IT.update(idx[v], idx[u], 1, 1, n, add);
73  }
74
75  int a[maxn];
76  void init();
77  int main() {
78      int u, v, l, r, k, d;
79      char s[100];
80      while (cin >> n >> Q >> Q) {
81          init();
82          FOR (i, 1, n + 1) scanf("%d", &a[i]);
83          FOR (i, 1, n) {
84              scanf("%d%d", &u, &v);
85              G[u].push_back(v);
86              G[v].push_back(u);
87          }
88          predfs(1, 1);
89          dfs(1, 1);
90          while (Q--) {
91              scanf("%s", s);
92              if (s[0] == 'I') {
```

```
93              scanf("%d%d%d", &l, &r, &d);
94              update(l, r, d);
95          } else if (s[0] == 'D') {
96              scanf("%d%d%d", &l, &r, &d);
97              update(l, r, -d);
98          } else {
99              scanf("%d", &k);
100             printf("%d\n", a[k] + IT.query(idx[k], 1, 1, n));
101         }
102     }
103   }
104 }
105
106 void init() {
107     clk = 0;
108     fa[1] = 0;
109     IT.init();
110     FOR (i, 0, n + 1) G[i].clear();
111 }
```

- SPOJ QTREE

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  typedef long long LL;
4  #define FOR(i, x, y) for (decay<decltype(y)>::type i = (x), _##i = (y); i < _##i; ++i)
5  #define FORD(i, x, y) for (decay<decltype(x)>::type i = (x), _##i = (y); i > _##i; --i)
6  #ifdef zerol
7  #define dbg(args...) do { cout << "\033[32;1m" << #args<< " -> "; err(args); } while (0)
8  #else
9  #define dbg(...)
10 #endif
11 void err() { cout << "\033[39m" << endl; }
12 template<typename T, typename... Args>
13 void err(T a, Args... args) {
14     cout << a << ' ';
15     err(args...);
16 }
17 // ----------------------------------------------------------------------------
18 const int maxn = 10000 * 2 * 4 + 100;
19 struct Edge {
20     int from, to, c;
21     Edge(int u, int v, int c): from(u), to(v), c(c) {}
22 };
23 vector<Edge> edge;
24 vector<int> G[maxn];
25 int fa[maxn], dep[maxn], sz[maxn], son[maxn], top[maxn], idx[maxn], w[maxn], val[maxn];
26 LL sum[maxn];
27 int n, clk, len;
28
29 struct IntervalTree {
30 #define lson p, q, o * 2, l, m
31 #define rson p, q, o * 2 + 1, m + 1, r
32     int maxv[maxn];
33     void init() { memset(maxv, 0, sizeof maxv); }
34     int query(int p, int q, int o, int l, int r) {
35 //      dbg(p, q);
36         assert(p <= q);
37         if (p > r || q < l) return 0;
38         if (p <= l && r <= q) return maxv[o];
39         int m = (l + r) / 2;
40         return max(query(lson), query(rson));
41     }
42     void maintain(int o, int l, int r) {
43         if (l < r)
44             maxv[o] = max(maxv[o * 2], maxv[o * 2 + 1]);
45     }
46     void update(int p, int q, int o, int l, int r, int v) {
47 //      dbg(p, q, o, l, r, v);
48         assert(p <= q);
49         if (p > r || q < l) return;
50         if (p <= l && r <= q) maxv[o] = v;
```

```
51              else {
52                  int m = (l + r) / 2;
53                  update(lson, v); update(rson, v);
54                  maintain(o, l, r);
55              }
56          }
57      } IT;
58
59      void dfs1(int u, int d) {
60          dep[u] = d;
61          sz[u] = 1;
62          FOR (i, 0, G[u].size()) {
63              Edge& e = edge[G[u][i]];
64              int v = e.to;
65              if (v == fa[u]) continue;
66              val[v] = e.c;
67  //            dbg(v, e.from, e.to, e.c);
68              fa[v] = u;
69              dfs1(v, d + 1);
70              sz[u] += sz[v];
71              if (son[u] == -1 || sz[v] > sz[son[u]])
72                  son[u] = v;
73          }
74      }
75
76      void dfs2(int u, int tp) {
77          top[u] = tp;
78          idx[u] = ++clk;
79          w[idx[u]] = tp;
80          if (son[u] == -1) return;
81          dfs2(son[u], tp);
82          FOR (i, 0, G[u].size()) {
83              int v = edge[G[u][i]].to;
84              if (v != son[u] && v != fa[u])
85                  dfs2(v, v);
86          }
87      }
88
89      int query(int u, int v) {
90          dbg(u, v);
91          int uu = top[u], vv = top[v], ret = 0;
92          while (uu != vv) {
93              if (dep[uu] < dep[vv]) { swap(u, v); swap(uu, vv); }
94  //            dbg(u, v, uu, vv, dep[uu], dep[vv], idx[uu], idx[u]);
95              ret = max(ret, IT.query(idx[uu], idx[u], 1, 1, len));
96              u = fa[uu];
97              uu = top[u];
98          }
99          if (dep[u] < dep[v]) swap(u, v);
100 //        dbg(idx[v], idx[u]);
101         if (u != v) ret = max(ret, IT.query(idx[v] + 1, idx[u], 1, 1, len));
102         return ret;
103     }
104
105     void init();
106     void add_edge(int u, int v, int c);
107
108     int main() {
109 #ifdef zerol
110         freopen("in", "r", stdin);
111 #endif
112         int T, u, v, c;
113         char s[100];
114         cin >> T;
115         while (T--) {
116             cin >> n;
117             for (len = 1; len < n; len *= 2);
118             init();
119             FOR (i, 1, n) {
120                 scanf("%d%d%d", &u, &v, &c);
121                 add_edge(u, v, c);
```

```
122                add_edge(v, u, c);
123            }
124            dfs1(1, 0);
125            dfs2(1, 1);
126    //        FOR (i, 1, n + 1) dbg(idx[i], w[i]);
127            FOR (i, 2, n + 1)
128                IT.update(idx[i], idx[i], 1, 1, len, val[i]);
129            while (scanf("%s", s) && s[0] != 'D') {
130                scanf("%d%d", &u, &v);
131                if (s[0] == 'C') {
132                    Edge& e = edge[u * 2 - 1];
133                    dbg(u, e.from, e.to);
134                    int t = max(idx[e.from], idx[e.to]);
135                    IT.update(t, t, 1, 1, len, v);
136                    dbg("upd", t, v);
137                }
138                if (s[0] == 'Q') printf("%d\n", query(u, v));
139            }
140            FOR (i, 1, n + 1) if (idx[i] == 2) dbg(i, idx[i]);
141            dbg(IT.query(idx[2], idx[2], 1, 1, len));
142            dbg(IT.query(idx[6], idx[6], 1, 1, len));
143        }
144    }
145
146    void init() {
147        edge.clear();
148        memset(son, -1, sizeof son);
149        memset(sum, 0, sizeof sum);
150        IT.init();
151        FOR (i, 0, n + 1) G[i].clear();
152        clk = 0;
153        fa[1] = 0;
154        sum[0] = sum[1] = 0;
155    }
156
157    void add_edge(int u, int v, int c) {
158        edge.emplace_back(u, v, c);
159        G[u].push_back(edge.size() - 1);
160    }
```

二分图匹配

- 最小覆盖数 = 最大匹配数
- 最大独立集 = 顶点数 - 二分图匹配数
- DAG 最小路径覆盖数 = 结点数 - 拆点后二分图最大匹配数

```
1    struct MaxMatch {
2        int n;
3        vector<int> G[maxn];
4        int vis[maxn], left[maxn], clk;
5
6        void init(int n) {
7            this->n = n;
8            FOR (i, 0, n + 1) G[i].clear();
9            memset(left, -1, sizeof left);
10           memset(vis, -1, sizeof vis);
11       }
12
13       bool dfs(int u) {
14           for (int v: G[u])
15               if (vis[v] != clk) {
16                   vis[v] = clk;
17                   if (left[v] == -1 || dfs(left[v])) {
18                       left[v] = u;
19                       return true;
20                   }
21               }
22           return false;
23       }
```

```
24
25    int match() {
26        int ret = 0;
27        for (clk = 0; clk <= n; ++clk)
28            if (dfs(clk)) ++ret;
29        return ret;
30    }
31 } MM;
```

- 二分图最大权完美匹配 KM

```
1  namespace R {
2      const int maxn = 300 + 10;
3      int n, m;
4      int left[maxn], L[maxn], R[maxn];
5      int w[maxn][maxn], slack[maxn];
6      bool visL[maxn], visR[maxn];
7
8      bool dfs(int u) {
9          visL[u] = true;
10         FOR (v, 0, m) {
11             if (visR[v]) continue;
12             int t = L[u] + R[v] - w[u][v];
13             if (t == 0) {
14                 visR[v] = true;
15                 if (left[v] == -1 || dfs(left[v])) {
16                     left[v] = u;
17                     return true;
18                 }
19             } else slack[v] = min(slack[v], t);
20         }
21         return false;
22     }
23
24     int go() {
25         memset(left, -1, sizeof left);
26         memset(R, 0, sizeof R);
27         memset(L, 0, sizeof L);
28         FOR (i, 0, n)
29             FOR (j, 0, m)
30                 L[i] = max(L[i], w[i][j]);
31
32         FOR (i, 0, n) {
33             memset(slack, 0x3f, sizeof slack);
34             while (1) {
35                 memset(visL, 0, sizeof visL); memset(visR, 0, sizeof visR);
36                 if (dfs(i)) break;
37                 int d = 0x3f3f3f3f;
38                 FOR (j, 0, m) if (!visR[j]) d = min(d, slack[j]);
39                 FOR (j, 0, n) if (visL[j]) L[j] -= d;
40                 FOR (j, 0, m) if (visR[j]) R[j] += d; else slack[j] -= d;
41             }
42         }
43         int ret = 0;
44         FOR (i, 0, m) if (left[i] != -1) ret += w[left[i]][i];
45         return ret;
46     }
47 }
```

虚树

```
1  void go(vector<int>& V, int& k) {
2      int u = V[k]; f[u] = 0;
3      dbg(u, k);
4      for (auto& e: G[u]) {
5          int v = e.to;
6          if (v == pa[u][0]) continue;
7          while (k + 1 < V.size()) {
8              int to = V[k + 1];
9              if (in[to] <= out[v]) {
```

```
10                go(V, ++k);
11                if (key[to]) f[u] += w[to];
12                else f[u] += min(f[to], (LL)w[to]);
13            } else break;
14        }
15    }
16    dbg(u, f[u]);
17 }
18 inline bool cmp(int a, int b) { return in[a] < in[b]; }
19 LL solve(vector<int>& V) {
20    static vector<int> a; a.clear();
21    for (int& x: V) a.push_back(x);
22    sort(a.begin(), a.end(), cmp);
23    FOR (i, 1, a.size())
24        a.push_back(lca(a[i], a[i - 1]));
25    a.push_back(1);
26    sort(a.begin(), a.end(), cmp);
27    a.erase(unique(a.begin(), a.end()), a.end());
28    dbg(a);
29    int tmp; go(a, tmp = 0);
30    return f[1];
31 }
```

欧拉路径

```
1  int S[N << 1], top;
2  Edge edges[N << 1];
3  set<int> G[N];
4
5  void DFS(int u) {
6      S[top++] = u;
7      for (int eid: G[u]) {
8          int v = edges[eid].get_other(u);
9          G[u].erase(eid);
10         G[v].erase(eid);
11         DFS(v);
12         return;
13     }
14 }
15
16 void fleury(int start) {
17     int u = start;
18     top = 0; path.clear();
19     S[top++] = u;
20     while (top) {
21         u = S[--top];
22         if (!G[u].empty())
23             DFS(u);
24         else path.push_back(u);
25     }
26 }
```

强连通分量与 2-SAT

```
1  int n, m;
2  vector<int> G[N], rG[N], vs;
3  int used[N], cmp[N];
4
5  void add_edge(int from, int to) {
6      G[from].push_back(to);
7      rG[to].push_back(from);
8  }
9
10 void dfs(int v) {
11     used[v] = true;
12     for (int u: G[v]) {
13         if (!used[u])
```

```
14              dfs(u);
15      }
16      vs.push_back(v);
17  }
18
19  void rdfs(int v, int k) {
20      used[v] = true;
21      cmp[v] = k;
22      for (int u: rG[v])
23          if (!used[u])
24              rdfs(u, k);
25  }
26
27  int scc() {
28      memset(used, 0, sizeof(used));
29      vs.clear();
30      for (int v = 0; v < n; ++v)
31          if (!used[v]) dfs(v);
32      memset(used, 0, sizeof(used));
33      int k = 0;
34      for (int i = (int) vs.size() - 1; i >= 0; --i)
35          if (!used[vs[i]]) rdfs(vs[i], k++);
36      return k;
37  }
38
39  int main() {
40      cin >> n >> m;
41      n *= 2;
42      for (int i = 0; i < m; ++i) {
43          int a, b; cin >> a >> b;
44          add_edge(a - 1, (b - 1) ^ 1);
45          add_edge(b - 1, (a - 1) ^ 1);
46      }
47      scc();
48      for (int i = 0; i < n; i += 2) {
49          if (cmp[i] == cmp[i + 1]) {
50              puts("NIE");
51              return 0;
52          }
53      }
54      for (int i = 0; i < n; i += 2) {
55          if (cmp[i] > cmp[i + 1]) printf("%d\n", i + 1);
56          else printf("%d\n", i + 2);
57      }
58  }
```

拓补排序

```
1  vector<int> toporder(int n) {
2      vector<int> orders;
3      queue<int> q;
4      for (int i = 0; i < n; i++)
5          if (!deg[i]) {
6              q.push(i);
7              orders.push_back(i);
8          }
9      while (!q.empty()) {
10         int u = q.front(); q.pop();
11         for (int v: G[u])
12             if (!--deg[v]) {
13                 q.push(v);
14                 orders.push_back(v);
15             }
16     }
17     return orders;
18 }
```

一般图匹配（没有测试过）

```
1   // maximum matching (graph not necessarily bipartite)
2   // whatever code uses this needs to set N in main()
3   // author claims N^4 running time
4
5   #define PB push_back
6   #define SZ(x) ((int)(x).size())
7   #define REP(i, n) for(int i=0; i<n; ++i)
8   #define FOR(i, b, e) for(typeof(e) i=b; i!=e; ++i)
9
10  int N; // the number of vertices in the graph
11
12  typedef vector<int> VI;
13  typpdef vector<vector<int>> VVI;
14
15  VI match;
16  VI vis;
17
18  void couple(int n, int m) {
19      match[n] = m;
20      match[m] = n;
21  }
22
23  // returns true if something interesting has been found, thus a
24  // augmenting path or a blossom (if blossom is non-empty).
25  // the dfs returns true from the moment the stem of the flower is
26  // reached and thus the base of the blossom is an unmatched node.
27  // blossom should be empty when dfs is called and
28  // contains the nodes of the blossom when a blossom is found.
29  bool dfs(int n, VVI &conn, VI &blossom) {
30      vis[n] = 0;
31      REP(i, N) if (conn[n][i]) {
32              if (vis[i] == -1) {
33                  vis[i] = 1;
34                  if (match[i] == -1 || dfs(match[i], conn, blossom)) {
35                      couple(n, i);
36                      return true;
37                  }
38              }
39              if (vis[i] == 0 || SZ(blossom)) {  // found flower
40                  blossom.PB(i);
41                  blossom.PB(n);
42                  if (n == blossom[0]) {
43                      match[n] = -1;
44                      return true;
45                  }
46                  return false;
47              }
48          }
49      return false;
50  }
51
52  // search for an augmenting path.
53  // if a blossom is found build a new graph (newconn) where the
54  // (free) blossom is shrunken to a single node and recurse.
55  // if a augmenting path is found it has already been augmented
56  // except if the augmented path ended on the shrunken blossom.
57  // in this case the matching should be updated along the appropriate
58  // direction of the blossom.
59  bool augment(VVI & conn) {
60      REP(m, N) if (match[m] == -1) {
61              VI blossom;
62              vis = VI(N, -1);
63              if (!dfs(m, conn, blossom)) continue;
64              if (SZ(blossom) == 0) return true; // augmenting path found
65
66              // blossom is found so build shrunken graph
67              int base = blossom[0], S = SZ(blossom);
68              VVI newconn = conn;
69              FOR(i, 1, S - 1) REP(j, N)
```

```
70              newconn[base][j] = newconn[j][base] |= conn[blossom[i]][j];
71          FOR(i, 1, S - 1) REP(j, N)
72              newconn[blossom[i]][j] = newconn[j][blossom[i]] = 0;
73          newconn[base][base] = 0; // is now the new graph
74          if (!augment(newconn)) return false;
75          int n = match[base];
76
77          // if n!=-1 the augmenting path ended on this blossom
78          if (n != -1)
79              REP(i, S) if (conn[blossom[i]][n]) {
80                  couple(blossom[i], n);
81                  if (i & 1) for (int j = i + 1; j < S; j += 2)
82                      couple(blossom[j], blossom[j + 1]);
83                  else for (int j = 0; j < i; j += 2)
84                      couple(blossom[j], blossom[j + 1]);
85                  break;
86              }
87          return true;
88      }
89      return false;
90  }
91
92  // conn should have N VI's, each of length N.  conn[i][j] = 1 if (i,j) is an
93  // edge, and conn[i][j] = 0 otherwise.  returns the number of edges in a max
94  // matching.
95  int edmonds(VVI & conn) {
96      int res = 0;
97      match = VI(N, -1);
98      while (augment(conn)) res++;
99      return res;
100 }
```

计算几何

圆的反演

```
1   typedef double LD;
2   const LD PI = 3.14159265358979323846;
3   const LD eps = 1E-10;
4   const LD R2 = 1.0;
5   int sgn(LD x) { return fabs(x) < eps ? 0 : (x > 0 ? 1 : -1); }
6   struct P {
7       LD x, y;
8       P(LD x = 0, LD y = 0): x(x), y(y) {}
9       P operator * (LD k) { return P(x * k, y * k); }
10      P operator / (LD k) { return P(x / k, y / k); }
11      string prt() const {
12          char s[100];
13          sprintf(s, "(%.2f, %.2f)", x, y);
14          return string(s);
15      }
16  };
17  typedef P V;
18  P operator - (const P& a, const P& b) { return P(a.x - b.x, a.y - b.y); }
19  P operator + (const P& a, const P& b) { return P(a.x + b.x, a.y + b.y); }
20  struct C {
21      P p;
22      LD r;
23      C(LD x = 0, LD y = 0, LD r = 0): p(x, y), r(r) {}
24  };
25  LD dist(V v) { return sqrt(v.x * v.x + v.y * v.y); }
26
27  C inv(C c, const P& o) {
28      LD d = dist(c.p - o);
29      assert(sgn(d) != 0);
30      LD a = 1 / (d - c.r);
31      LD b = 1 / (d + c.r);
32      c.r = (a - b) / 2 * R2;
```

```
33        c.p = o + (c.p - o) * ((a + b) * R2 / 2 / d);
34        return c;
35    }
```

二维

- nxt 宏要求多边形变量名为 s
- L 可隐式转换为 V(P)
- 可以自定义结构体 PP，可隐式转换为 P

```
1   #define y1 yy1
2   #define nxt(i) ((i + 1) % s.size())
3   typedef double LD;
4   const LD PI = 3.14159265358979323846;
5   const LD eps = 1E-10;
6   int sgn(LD x) { return fabs(x) < eps ? 0 : (x > 0 ? 1 : -1); }
7   struct L;
8   struct P;
9   //struct PP;
10  typedef P V;
11  struct P {
12      LD x, y;
13      explicit P(LD x = 0, LD y = 0): x(x), y(y) {}
14      P(const L& l);
15  //    P(const PP& pp);
16  };
17  struct L {
18      P s, t;
19      L() {}
20      L(P s, P t): s(s), t(t) {}
21  };
22
23  P operator + (const P& a, const P& b) { return P(a.x + b.x, a.y + b.y); }
24  P operator - (const P& a, const P& b) { return P(a.x - b.x, a.y - b.y); }
25  P operator * (const P& a, LD k) { return P(a.x * k, a.y * k); }
26  P operator / (const P& a, LD k) { return P(a.x / k, a.y / k); }
27  inline int operator < (const P& a, const P& b) {
28      return sgn(a.x - b.x) < 0 || (sgn(a.x - b.x) == 0 && sgn(a.y - b.y) < 0);
29  }
30  bool operator == (const P& a, const P& b) { return !sgn(a.x - b.x) && !sgn(a.y - b.y); }
31  P::P(const L& l) { *this = l.t - l.s; }
32  ostream &operator << (ostream &os, const P &p) {
33      return (os << "(" << p.x << "," << p.y << ")");
34  }
35  istream &operator >> (istream &is, P &p) {
36      return (is >> p.x >> p.y);
37  }
38
39  // ----------------------------------------
40
41  //struct PP {
42  //    P p;
43  //    LD v, l;
44  //};
45  //P::P(const PP& pp) { *this = pp.p; }
46  typedef P PP;
47
48  typedef vector<PP> S;
49
50  // ----------------------------------------
51  LD dist(const P& p) { return sqrt(p.x * p.x + p.y * p.y); }
52  LD dot(const V& a, const V& b) { return a.x * b.x + a.y * b.y; }
53  LD det(const V& a, const V& b) { return a.x * b.y - a.y * b.x; }
54  LD cross(const P& s, const P& t, const P& o = P()) { return det(s - o, t - o); }
55
56  // 如需支持 unique, 需要加 eps
57  bool cmp_xy(const P& a, const P& b) { return a.x < b.x || a.x == b.x && a.y < b.y; }
58
59  // 象限
```

```
60    int quad(P p) {
61        int x = sgn(p.x), y = sgn(p.y);
62        if (x > 0 && y >= 0) return 1;
63        if (x <= 0 && y > 0) return 2;
64        if (x < 0 && y <= 0) return 3;
65        if (x >= 0 && y < 0) return 4;
66        assert(0);
67    }
68
69    // 仅适用于参照点在所有点一侧的情况
70    struct cmp_angle {
71        P p;
72        bool operator () (const P& a, const P& b) {
73    //        int qa = quad(a), qb = quad(b);
74    //        if (qa != qb) return qa < qb;
75            int d = sgn(cross(a, b, p));
76            if (d) return d > 0;
77            return dist(a - p) < dist(b - p);
78        }
79    };
80
81    // ---------------线----------------
82
83    // 是否平行
84    bool parallel(const L& a, const L& b) {
85        return !sgn(det(a, b));
86    }
87    // 直线是否相等
88    bool l_eq(const L& a, const L& b) {
89        return parallel(a, b) && parallel(L(a.s, b.t), L(b.s, a.t));
90    }
91    // 逆时针旋转 r 弧度
92    P rotation(const P& p, const LD& r) { return P(p.x * cos(r) - p.y * sin(r), p.x * sin(r) + p.y * cos(r)); }
93    P RotateCCW90(const P& p) { return P(-p.y, p.x); }
94    P RotateCW90(const P& p) { return P(p.y, -p.x); }
95    // 单位法向量
96    V normal(const V& v) { return V(-v.y, v.x) / dist(v); }
97
98
99    // ---------------点和线----------------
100
101   // 点在线段上  <= 0 包含端点 < 0 则不包含
102   bool p_on_seg(const P& p, const L& seg) {
103       P a = seg.s, b = seg.t;
104       return !sgn(det(p - a, b - a)) && sgn(dot(p - a, p - b)) <= 0;
105   }
106   // 点到直线距离
107   LD dist_to_line(const P& p, const L& l) {
108       return fabs(cross(l.s, l.t, p)) / dist(l);
109   }
110   // 点到线段距离
111   LD dist_to_seg(const P& p, const L& l) {
112       if (l.s == l.t) return dist(p - l);
113       V vs = p - l.s, vt = p - l.t;
114       if (sgn(dot(l, vs)) < 0) return dist(vs);
115       else if (sgn(dot(l, vt)) > 0) return dist(vt);
116       else return dist_to_line(p, l);
117   }
118
119
120   // ---------------线和线----------------
121
122   // 求直线交 需要事先保证有界
123   P l_intersection(const L& a, const L& b) {
124       LD s1 = det(a, b.s - a.s), s2 = det(a, b.t - a.s);
125       return (b.s * s2 - b.t * s1) / (s2 - s1);
126   }
127   // 向量夹角的弧度
128   LD angle(const V& a, const V& b) {
129       LD r = asin(fabs(det(a, b)) / dist(a) / dist(b));
130       if (sgn(dot(a, b)) < 0) r = PI - r;
```

```
131        return r;
132    }
133    // 线段和直线是否有交    1 = 规范, 2 = 不规范
134    int s_l_cross(const L& seg, const L& line) {
135        int d1 = sgn(cross(line.s, line.t, seg.s));
136        int d2 = sgn(cross(line.s, line.t, seg.t));
137        if ((d1 ^ d2) == -2) return 1; // proper
138        if (d1 == 0 || d2 == 0) return 2;
139        return 0;
140    }
141    // 线段的交    1 = 规范, 2 = 不规范
142    int s_cross(const L& a, const L& b, P& p) {
143        int d1 = sgn(cross(a.t, b.s, a.s)), d2 = sgn(cross(a.t, b.t, a.s));
144        int d3 = sgn(cross(b.t, a.s, b.s)), d4 = sgn(cross(b.t, a.t, b.s));
145        if ((d1 ^ d2) == -2 && (d3 ^ d4) == -2) { p = l_intersection(a, b); return 1; }
146        if (!d1 && p_on_seg(b.s, a)) { p = b.s; return 2; }
147        if (!d2 && p_on_seg(b.t, a)) { p = b.t; return 2; }
148        if (!d3 && p_on_seg(a.s, b)) { p = a.s; return 2; }
149        if (!d4 && p_on_seg(a.t, b)) { p = a.t; return 2; }
150        return 0;
151    }
152
153
154    // ---------------多边形---------------
155
156    // 点是否在多边形中 0 = 在外部 1 = 在内部 -1 = 在边界上
157    int inside(const S& s, const P& p) {
158        int cnt = 0;
159        FOR (i, 0, s.size()) {
160            P a = s[i], b = s[nxt(i)];
161            if (p_on_seg(p, L(a, b))) return -1;
162            if (sgn(a.y - b.y) <= 0) swap(a, b);
163            if (sgn(p.y - a.y) > 0) continue;
164            if (sgn(p.y - b.y) <= 0) continue;
165            cnt += sgn(cross(b, a, p)) > 0;
166        }
167        return bool(cnt & 1);
168    }
169    // 多边形面积, 有向面积可能为负
170    LD polygon_area(const S& s) {
171        LD ret = 0;
172        FOR (i, 1, (LL)s.size() - 1)
173            ret += cross(s[i], s[i + 1], s[0]);
174        return ret / 2;
175    }
176    // 构建凸包 点不可以重复 < 0 边上可以有点, <= 0 则不能
177    // 会改变输入点的顺序
178    const int MAX_N = 1000;
179    S convex_hull(S& s) {
180    //    assert(s.size() >= 3);
181        sort(s.begin(), s.end(), cmp_xy);
182        S ret(MAX_N * 2);
183        int sz = 0;
184        FOR (i, 0, s.size()) {
185            while (sz > 1 && sgn(cross(ret[sz - 1], s[i], ret[sz - 2])) < 0) --sz;
186            ret[sz++] = s[i];
187        }
188        int k = sz;
189        FORD (i, (LL)s.size() - 2, -1) {
190            while (sz > k && sgn(cross(ret[sz - 1], s[i], ret[sz - 2])) < 0) --sz;
191            ret[sz++] = s[i];
192        }
193        ret.resize(sz - (s.size() > 1));
194        return ret;
195    }
196
197    P ComputeCentroid(const vector<P> &p) {
198        P c(0, 0);
199        LD scale = 6.0 * polygon_area(p);
200        for (unsigned i = 0; i < p.size(); i++) {
201            unsigned j = (i + 1) % p.size();
```

```
202        c = c + (p[i] + p[j]) * (p[i].x * p[j].y - p[j].x * p[i].y);
203    }
204    return c / scale;
205 }
206
207 // ------------------圆-------------------
208
209 P ComputeCircleCenter(P a, P b, P c) {
210    b = (a + b) / 2;
211    c = (a + c) / 2;
212    return l_intersection({b, b + RotateCW90(a - b)}, {c , c + RotateCW90(a - c)});
213 }
214 vector<P> CircleLineIntersection(P a, P b, P c, LD r) {
215    vector<P> ret;
216    b = b - a;
217    a = a - c;
218    LD A = dot(b, b), B = dot(a, b), C = dot(a, a) - r * r;
219    LD D = B * B - A * C;
220    if (sgn(D) < 0) return ret;
221    ret.push_back(c + a + b * (-B + sqrt(D + eps)) / A);
222    if (sgn(D) > 0) ret.push_back(c + a + b * (-B - sqrt(D)) / A);
223    return ret;
224 }
225 vector<P> CircleCircleIntersection(P a, P b, LD r, LD R) {
226    vector<P> ret;
227    LD d = dist(a - b);
228    if (sgn(d) == 0 || sgn(d - (r + R)) > 0 || sgn(d + min(r, R) - max(r, R)) < 0) return ret;
229    LD x = (d * d - R * R + r * r) / (2 * d);
230    LD y = sqrt(r * r - x * x);
231    P v = (b - a) / d;
232    ret.push_back(a + v * x + RotateCCW90(v) * y);
233    if (sgn(y) > 0) ret.push_back(a + v * x - RotateCCW90(v) * y);
234    return ret;
235 }
236
237 // ----------------模板结束----------------
```

旋转卡壳

```
1  LD rotatingCalipers(S& qs) {
2      int n = qs.size();
3      if (n == 2)
4          return dist(qs[0] - qs[1]);
5      int i = 0, j = 0;
6      FOR (k, 0, n) {
7          if (!(qs[i] < qs[k])) i = k;
8          if (qs[j] < qs[k]) j = k;
9      }
10     LD res = 0;
11     int si = i, sj = j;
12     while (i != sj || j != si) {
13         res = max(res, dist(qs[i] - qs[j]));
14         if (sgn(cross(qs[(i+1)%n] - qs[i], qs[(j+1)%n] - qs[j])) < 0)
15             i = (i + 1) % n;
16         else j = (j + 1) % n;
17     }
18     return res;
19 }
20
21 int main() {
22     int n;
23     while (cin >> n) {
24         S v(n);
25         FOR (i, 0, n) cin >> v[i].x >> v[i].y;
26         convex_hull(v);
27         printf("%.0f\n", rotatingCalipers(v));
28     }
29 }
```

没有测试过的

```
int relation(Point p, Circle a) {//点和圆的关系
    //0: 圆外 1: 圆上 2: 圆内
    double d = dis(p, a.p);
    if (dcmp(d - a.r) == 0) return 1;
    return (dcmp(d - a.r) < 0 ? 2 : 0);
}

int relation(Line a, Circle b) {//直线和圆的关系
    //0: 相离 1: 相切 2: 相交
    double p = point_to_line(b.p, a);
    if (dcmp(p - b.r) == 0) return 1;
    return (dcmp(p - b.r) < 0 ? 2 : 0);
}

int relation(Circle a, Circle v) {//两圆的位置关系
    //1: 内含 2: 内切 3: 相交 4: 外切 5: 相离
    double d = dis(a.p, v.p);
    if (dcmp(d - a.r - v.r) > 0) return 5;
    if (dcmp(d - a.r - v.r) == 0) return 4;
    double l = fabs(a.r - v.r);
    if (dcmp(d - a.r - v.r) < 0 && dcmp(d - l) > 0) return 3;
    if (dcmp(d - l) == 0) return 2;
    if (dcmp(d - l) < 0) return 1;
    assert (0);
}

double circle_traingle_area(Point a, Point b, Circle c) {//圆心三角形的面积
    //a.output (), b.output (), c.output ();
    Point p = c.p;
    double r = c.r; //cout << cross (p-a, p-b) << endl;
    if (dcmp(cross(p - a, p - b)) == 0) return 0;
    Point q[5];
    int len = 0;
    q[len++] = a;
    Line l(a, b);
    Point p1, p2;
    if (line_cirlce_intersection(l, c, q[1], q[2]) == 2) {
        if (dcmp(dot(a - q[1], b - q[1])) < 0) q[len++] = q[1];
        if (dcmp(dot(a - q[2], b - q[2])) < 0) q[len++] = q[2];
    }
    q[len++] = b;
    if (len == 4 && dcmp(dot(q[0] - q[1], q[2] - q[1])) > 0)
        swap(q[1], q[2]);
    double res = 0;
    for (int i = 0; i < len - 1; i++) {
        if (relation(q[i], c) == 0 || relation(q[i + 1], c) == 0) {
            double arg = rad(q[i] - p, q[i + 1] - p);
            res += r * r * arg / 2.0;
        } else {
            res += fabs(cross(q[i] - p, q[i + 1] - p)) / 2;
        }
    }
    return res;
}
```

半平面交

```
struct Line {
    PT p, v;
    double ang;

    Line() {}
    Line(PT from, PT to) : p(from), v(to - from) { ang = atan2(v.y, v.x); }
    friend bool operator<(Line a, Line b) {
        return a.ang < b.ang;
    }
};
```

```
11
12   bool OnLeft(Line L, PT p) {
13       return dcmp(cross(L.v, p - L.p)) >= 0;
14   }
15
16   PT GetIntersection(Line a, Line b) {
17       PT u = a.p - b.p;
18       ld t = cross(b.v, u) / cross(a.v, b.v);
19       return a.p + a.v * t;
20   }
21
22   vector<PT> HalfplaneIntersection(vector<Line>& L) {
23       int n = L.size();
24       sort(L.begin(), L.end());
25
26       int first, last;
27       vector<PT> p(n);
28       vector<Line> q(n);
29       q[first = last = 0] = L[0];
30       for (int i = 1; i < n; i++) {
31           while (first < last && !OnLeft(L[i], p[last - 1])) last--;
32           while (first < last && !OnLeft(L[i], p[first])) first++;
33           q[++last] = L[i];
34           if (dcmp(cross(q[last].v, q[last - 1].v)) == 0) {
35               last--;
36               if (OnLeft(q[last], L[i].p)) q[last] = L[i];
37           }
38           if (first < last) p[last - 1] = GetIntersection(q[last - 1], q[last]);
39       }
40       while (first < last && !OnLeft(q[first], p[last - 1])) last--;
41       if (last - first <= 1) return vector<PT>();
42       p[last] = GetIntersection(q[last], q[first]);
43
44       return vector<PT>(p.begin() + first, p.begin() + last + 1);
45   }
46
47   vector<PT> convexIntersection(const vector<PT> &v1, const vector<PT> &v2) {
48       vector<Line> h;
49       int n = v1.size(), m = v2.size();
50       for (int i = 0; i < n; ++i)
51           h.push_back(Line(v1[i], v1[(i + 1) % n]));
52       for (int i = 0; i < m; ++i)
53           h.push_back(Line(v2[i], v2[(i + 1) % m]));
54       return HalfplaneIntersection(h);
55   }
56
57   ld ComputeSignedArea(const vector<PT> &p) {
58       ld area = 0;
59       for (unsigned i = 0; i < p.size(); i++) {
60           unsigned j = (i + 1) % p.size();
61           area += p[i].x * p[j].y - p[j].x * p[i].y;
62       }
63       return area / 2.0;
64   }
65
66   ld ComputeArea(const vector<PT> &p) {
67       return fabs(ComputeSignedArea(p));
68   }
```

字符串

后缀自动机

- 广义后缀自动机如果直接使用以下代码的话会产生一些冗余状态（置 last 为 1），所以要用拓扑排序。用 len 基数排序不能。
- 字符集大的话要使用 map。
- 树上 dp 时注意边界（root 和 null）。
- rsort 需要初始化

```
1   namespace sam {
2       const int M = N << 1;
3       int t[M][26], len[M] = {-1}, fa[M], sz = 2, last = 1;
4       void ins(int ch) {
5           int p = last, np = last = sz++;
6           len[np] = len[p] + 1;
7           for (; p && !t[p][ch]; p = fa[p]) t[p][ch] = np;
8           if (!p) { fa[np] = 1; return; }
9           int q = t[p][ch];
10          if (len[p] + 1 == len[q]) fa[np] = q;
11          else {
12              int nq = sz++; len[nq] = len[p] + 1;
13              memcpy(t[nq], t[q], sizeof t[0]);
14              fa[nq] = fa[q];
15              fa[np] = fa[q] = nq;
16              for (; t[p][ch] == q; p = fa[p]) t[p][ch] = nq;
17          }
18      }
19
20      int c[M] = {1}, a[M];
21      void rsort() {
22          FOR (i, 1, sz) c[i] = 0;
23          FOR (i, 1, sz) c[len[i]]++;
24          FOR (i, 1, sz) c[i] += c[i - 1];
25          FOR (i, 1, sz) a[--c[len[i]]] = i;
26      }
27  }
```

- 真·广义后缀自动机

```
1   int t[M][26], len[M] = {-1}, fa[M], sz = 2, last = 1;
2   LL cnt[M][2];
3   void ins(int ch, int id) {
4       int p = last, np = 0, nq = 0, q = -1;
5       if (!t[p][ch]) {
6           np = sz++;
7           len[np] = len[p] + 1;
8           for (; p && !t[p][ch]; p = fa[p]) t[p][ch] = np;
9       }
10      if (!p) fa[np] = 1;
11      else {
12          q = t[p][ch];
13          if (len[p] + 1 == len[q]) fa[np] = q;
14          else {
15              nq = sz++; len[nq] = len[p] + 1;
16              memcpy(t[nq], t[q], sizeof t[0]);
17              fa[nq] = fa[q];
18              fa[np] = fa[q] = nq;
19              for (; t[p][ch] == q; p = fa[p]) t[p][ch] = nq;
20          }
21      }
22      last = np ? np : nq ? nq : q;
23      cnt[last][id] = 1;
24  }
```

- 按字典序建立后缀树注意逆序插入

```
1   void ins(int ch, int pp) {
2       int p = last, np = last = sz++;
3       len[np] = len[p] + 1; one[np] = pos[np] = pp;
4       for (; p && !t[p][ch]; p = fa[p]) t[p][ch] = np;
5       if (!p) { fa[np] = 1; return; }
6       int q = t[p][ch];
7       if (len[q] == len[p] + 1) fa[np] = q;
8       else {
9           int nq = sz++; len[nq] = len[p] + 1; one[nq] = one[q];
10          memcpy(t[nq], t[q], sizeof t[0]);
11          fa[nq] = fa[q];
12          fa[q] = fa[np] = nq;
13          for (; p && t[p][ch] == q; p = fa[p]) t[p][ch] = nq;
14      }
15  }
```

```
16
17  int up[M], c[256] = {2}, a[M];
18  void rsort2() {
19      FOR (i, 1, 256) c[i] = 0;
20      FOR (i, 2, sz) up[i] = s[one[i] + len[fa[i]]];
21      FOR (i, 2, sz) c[up[i]]++;
22      FOR (i, 1, 256) c[i] += c[i - 1];
23      FOR (i, 2, sz) a[--c[up[i]]] = i;
24      FOR (i, 2, sz) G[fa[a[i]]].push_back(a[i]);
25  }
```

- 广义后缀自动机建后缀树，必须反向插入

```
1   int t[M][26], len[M] = {0}, fa[M], sz = 2, last = 1;
2   char* one[M];
3   void ins(int ch, char* pp) {
4       int p = last, np = 0, nq = 0, q = -1;
5       if (!t[p][ch]) {
6           np = sz++; one[np] = pp;
7           len[np] = len[p] + 1;
8           for (; p && !t[p][ch]; p = fa[p]) t[p][ch] = np;
9       }
10      if (!p) fa[np] = 1;
11      else {
12          q = t[p][ch];
13          if (len[p] + 1 == len[q]) fa[np] = q;
14          else {
15              nq = sz++; len[nq] = len[p] + 1; one[nq] = one[q];
16              memcpy(t[nq], t[q], sizeof t[0]);
17              fa[nq] = fa[q];
18              fa[np] = fa[q] = nq;
19              for (; t[p][ch] == q; p = fa[p]) t[p][ch] = nq;
20          }
21      }
22      last = np ? np : nq ? nq : q;
23  }
24  int up[M], c[256] = {2}, aa[M];
25  vector<int> G[M];
26  void rsort() {
27      FOR (i, 1, 256) c[i] = 0;
28      FOR (i, 2, sz) up[i] = *(one[i] + len[fa[i]]);
29      FOR (i, 2, sz) c[up[i]]++;
30      FOR (i, 1, 256) c[i] += c[i - 1];
31      FOR (i, 2, sz) aa[--c[up[i]]] = i;
32      FOR (i, 2, sz) G[fa[aa[i]]].push_back(aa[i]);
33  }
```

- 匹配

```
1   int u = 1, l = 0;
2   FOR (i, 0, strlen(s)) {
3       int ch = s[i] - 'a';
4       while (u && !t[u][ch]) { u = fa[u]; l = len[u]; }
5       ++l; u = t[u][ch];
6       if (!u) u = 1;
7       // do something...
8   }
```

- 获取子串状态

```
1   int get_state(int l, int r) {
2       int u = rpos[r], s = r - l + 1;
3       FORD (i, SP - 1, -1) if (len[pa[u][i]] >= s) u = pa[u][i];
4       return u;
5   }
```

- 配合 LCT

```
1   namespace lct_sam {
2       extern struct P *const null;
3       const int M = N;
4       struct P {
5           P *fa, *ls, *rs;
```

```
6          int last;
7
8          bool has_fa() { return fa->ls == this || fa->rs == this; }
9          bool d() { return fa->ls == this; }
10         P*& c(bool x) { return x ? ls : rs; }
11         P* up() { return this; }
12         void down() {
13             if (ls != null) ls->last = last;
14             if (rs != null) rs->last = last;
15         }
16         void all_down() { if (has_fa()) fa->all_down(); down(); }
17     } *const null = new P{0, 0, 0, 0}, pool[M], *pit = pool;
18     P* G[N];
19     int t[M][26], len[M] = {-1}, fa[M], sz = 2, last = 1;
20
21     void rot(P* o) {
22         bool dd = o->d();
23         P *f = o->fa, *t = o->c(!dd);
24         if (f->has_fa()) f->fa->c(f->d()) = o; o->fa = f->fa;
25         if (t != null) t->fa = f; f->c(dd) = t;
26         o->c(!dd) = f->up(); f->fa = o;
27     }
28     void splay(P* o) {
29         o->all_down();
30         while (o->has_fa()) {
31             if (o->fa->has_fa())
32                 rot(o->d() ^ o->fa->d() ? o : o->fa);
33             rot(o);
34         }
35         o->up();
36     }
37     void access(int last, P* u, P* v = null) {
38         if (u == null) { v->last = last; return; }
39         splay(u);
40         P *t = u;
41         while (t->ls != null) t = t->ls;
42         int L = len[fa[t - pool]] + 1, R = len[u - pool];
43
44         if (u->last) bit::add(u->last - R + 2, u->last - L + 2, 1);
45         else bit::add(1, 1, R - L + 1);
46         bit::add(last - R + 2, last - L + 2, -1);
47
48         u->rs = v;
49         access(last, u->up()->fa, u);
50     }
51     void insert(P* u, P* v, P* t) {
52         if (v != null) { splay(v); v->rs = null; }
53         splay(u);
54         u->fa = t; t->fa = v;
55     }
56
57     void ins(int ch, int pp) {
58         int p = last, np = last = sz++;
59         len[np] = len[p] + 1;
60         for (; p && !t[p][ch]; p = fa[p]) t[p][ch] = np;
61         if (!p) fa[np] = 1;
62         else {
63             int q = t[p][ch];
64             if (len[p] + 1 == len[q]) { fa[np] = q; G[np]->fa = G[q]; }
65             else {
66                 int nq = sz++; len[nq] = len[p] + 1;
67                 memcpy(t[nq], t[q], sizeof t[0]);
68                 insert(G[q], G[fa[q]], G[nq]);
69                 G[nq]->last = G[q]->last;
70                 fa[nq] = fa[q];
71                 fa[np] = fa[q] = nq;
72                 G[np]->fa = G[nq];
73                 for (; t[p][ch] == q; p = fa[p]) t[p][ch] = nq;
74             }
75         }
76         access(pp + 1, G[np]);
```

```
77            }
78
79        void init() {
80            ++pit;
81            FOR (i, 1, N) {
82                G[i] = pit++;
83                G[i]->ls = G[i]->rs = G[i]->fa = null;
84            }
85            G[1] = null;
86        }
87    }
```

回文自动机

```
1    namespace pam {
2        int t[N][26], fa[N], len[N], rs[N], cnt[N];
3        int sz, n, last;
4        int _new(int l) {
5            memset(t[sz], 0, sizeof t[0]);
6            len[sz] = l; cnt[sz] = 0;
7            return sz++;
8        }
9        void init() {
10            rs[n = sz = 0] = -1;
11            last = _new(0);
12            fa[last] = _new(-1);
13        }
14        int get_fa(int x) {
15            while (rs[n - 1 - len[x]] != rs[n]) x = fa[x];
16            return x;
17        }
18        void ins(int ch) {
19            rs[++n] = ch;
20            int p = get_fa(last);
21            if (!t[p][ch]) {
22                int np = _new(len[p] + 2);
23                fa[np] = t[get_fa(fa[p])][ch];
24                t[p][ch] = np;
25            }
26            ++cnt[last = t[p][ch]];
27        }
28    }
```

manacher

```
1    int RL[N];
2    void manacher(int* a, int n) { // "abc" => "#a#b#a#"
3        int r = 0, p = 0;
4        FOR (i, 0, n) {
5            if (i < r) RL[i] = min(RL[2 * p - i], r - i);
6            else RL[i] = 1;
7            while (i - RL[i] >= 0 && i + RL[i] < n && a[i - RL[i]] == a[i + RL[i]])
8                RL[i]++;
9            if (RL[i] + i - 1 > r) { r = RL[i] + i - 1; p = i; }
10        }
11        FOR (i, 0, n) --RL[i];
12    }
```

哈希

内置了自动双哈希开关（小心 TLE）。

```
1    #include <bits/stdc++.h>
2    using namespace std;
3
4    #define ENABLE_DOUBLE_HASH
```

```
5
6    typedef long long LL;
7    typedef unsigned long long ULL;
8
9    const int x = 135;
10   const int N = 4e5 + 10;
11   const int p1 = 1e9 + 7, p2 = 1e9 + 9;
12   ULL xp1[N], xp2[N], xp[N];
13
14   void init_xp() {
15       xp1[0] = xp2[0] = xp[0] = 1;
16       for (int i = 1; i < N; ++i) {
17           xp1[i] = xp1[i - 1] * x % p1;
18           xp2[i] = xp2[i - 1] * x % p2;
19           xp[i] = xp[i - 1] * x;
20       }
21   }
22
23   struct String {
24       char s[N];
25       int length, subsize;
26       bool sorted;
27       ULL h[N], hl[N];
28
29       ULL hash() {
30           length = strlen(s);
31           ULL res1 = 0, res2 = 0;
32           h[length] = 0;  // ATTENTION!
33           for (int j = length - 1; j >= 0; --j) {
34           #ifdef ENABLE_DOUBLE_HASH
35               res1 = (res1 * x + s[j]) % p1;
36               res2 = (res2 * x + s[j]) % p2;
37               h[j] = (res1 << 32) | res2;
38           #else
39               res1 = res1 * x + s[j];
40               h[j] = res1;
41           #endif
42               // printf("%llu\n", h[j]);
43           }
44           return h[0];
45       }
46
47       // 获取子串哈希, 左闭右开区间
48       ULL get_substring_hash(int left, int right) const {
49           int len = right - left;
50       #ifdef ENABLE_DOUBLE_HASH
51           // get hash of s[left...right-1]
52           unsigned int mask32 = ~(0u);
53           ULL left1 = h[left] >> 32, right1 = h[right] >> 32;
54           ULL left2 = h[left] & mask32, right2 = h[right] & mask32;
55           return ((((left1 - right1 * xp1[len] % p1 + p1) % p1) << 32) |
56                   (((left2 - right2 * xp2[len] % p2 + p2) % p2)));
57       #else
58           return h[left] - h[right] * xp[len];
59       #endif
60       }
61
62       void get_all_subs_hash(int sublen) {
63           subsize = length - sublen + 1;
64           for (int i = 0; i < subsize; ++i)
65               hl[i] = get_substring_hash(i, i + sublen);
66           sorted = 0;
67       }
68
69       void sort_substring_hash() {
70           sort(hl, hl + subsize);
71           sorted = 1;
72       }
73
74       bool match(ULL key) const {
75           if (!sorted) assert (0);
```

```
76          if (!subsize) return false;
77          return binary_search(hl, hl + subsize, key);
78      }
79
80      void init(const char *t) {
81          length = strlen(t);
82          strcpy(s, t);
83      }
84  };
85
86  int LCP(const String &a, const String &b, int ai, int bi) {
87      // Find LCP of a[ai...] and b[bi...]
88      int l = 0, r = min(a.length - ai, b.length - bi);
89      while (l < r) {
90          int mid = (l + r + 1) / 2;
91          if (a.get_substring_hash(ai, ai + mid) == b.get_substring_hash(bi, bi + mid))
92              l = mid;
93          else r = mid - 1;
94      }
95      return l;
96  }
97
98  int check(int ans) {
99      if (T.length < ans) return 1;
100     T.get_all_subs_hash(ans); T.sort_substring_hash();
101     for (int i = 0; i < S.length - ans + 1; ++i)
102         if (!T.match(S.get_substring_hash(i, i + ans)))
103             return 1;
104     return 0;
105 }
106
107 int main() {
108     init_xp();  // DON'T FORGET TO DO THIS!
109
110     for (int tt = 1; tt <= kases; ++tt) {
111         scanf("%d", &n); scanf("%s", str);
112         S.init(str);
113         S.hash(); T.hash();
114     }
115 }
```

后缀数组

构造时间: $O(L \log L)$; 查询时间 $O(\log L)$。suffix 数组是排好序的后缀下标, suffix 的反数组是后缀数组。

```
1   #include <bits/stdc++.h>
2   using namespace std;
3
4   const int N = 2e5 + 10;
5   const int Nlog = 18;
6
7   struct SuffixArray {
8       const int L;
9       vector<vector<int> > P;
10      vector<pair<pair<int, int>, int> > M;
11      int s[N], sa[N], rank[N], height[N];
12      // s: raw string
13      // sa[i]=k: s[k...L-1] ranks i (0 based)
14      // rank[i]=k: the rank of s[i...L-1] is k (0 based)
15      // height[i] = lcp(sa[i-1], sa[i])
16
17      SuffixArray(const string &raw_s) : L(raw_s.length()), P(1, vector<int>(L, 0)), M(L) {
18          for (int i = 0; i < L; i++)
19              P[0][i] = this->s[i] = int(raw_s[i]);
20          for (int skip = 1, level = 1; skip < L; skip *= 2, level++) {
21              P.push_back(vector<int>(L, 0));
22              for (int i = 0; i < L; i++)
23                  M[i] = make_pair(make_pair(P[level - 1][i], i + skip < L ? P[level - 1][i + skip] : -1000), i);
24              sort(M.begin(), M.end());
```

```
25          for (int i = 0; i < L; i++)
26              P[level][M[i].second] = (i > 0 && M[i].first == M[i - 1].first) ? P[level][M[i - 1].second] : i;
27          }
28          for (unsigned i = 0; i < P.back().size(); ++i) {
29              rank[i] = P.back()[i];
30              sa[rank[i]] = i;
31          }
32      }
33
34      // This is a traditional way to calculate LCP
35      void getHeight() {
36          memset(height, 0, sizeof height);
37          int k = 0;
38          for (int i = 0; i < L; ++i) {
39              if (rank[i] == 0) continue;
40              if (k) k--;
41              int j = sa[rank[i] - 1];
42              while (i + k < L && j + k < L && s[i + k] == s[j + k]) ++k;
43              height[rank[i]] = k;
44          }
45          rmq_init(height, L);
46      }
47
48      int f[N][Nlog];
49      inline int highbit(int x) {
50          return 31 - __builtin_clz(x);
51      }
52
53      int rmq_query(int x, int y) {
54          int p = highbit(y - x + 1);
55          return min(f[x][p], f[y - (1 << p) + 1][p]);
56      }
57
58      // arr has to be 0 based
59      void rmq_init(int *arr, int length) {
60          for (int x = 0; x <= highbit(length); ++x)
61              for (int i = 0; i <= length - (1 << x); ++i) {
62                  if (!x) f[i][x] = arr[i];
63                  else f[i][x] = min(f[i][x - 1], f[i + (1 << (x - 1))][x - 1]);
64              }
65      }
66
67      #ifdef NEW
68      // returns the length of the longest common prefix of s[i...L-1] and s[j...L-1]
69      int LongestCommonPrefix(int i, int j) {
70          int len = 0;
71          if (i == j) return L - i;
72          for (int k = (int) P.size() - 1; k >= 0 && i < L && j < L; k--) {
73              if (P[k][i] == P[k][j]) {
74                  i += 1 << k;
75                  j += 1 << k;
76                  len += 1 << k;
77              }
78          }
79          return len;
80      }
81      #else
82      int LongestCommonPrefix(int i, int j) {
83          // getHeight() must be called first
84          if (i == j) return L - i;
85          if (i > j) swap(i, j);
86          return rmq_query(i + 1, j);
87      }
88      #endif
89
90      int checkNonOverlappingSubstring(int K) {
91          // check if there is two non-overlapping identical substring of length K
92          int minsa = 0, maxsa = 0;
93          for (int i = 0; i < L; ++i) {
94              if (height[i] < K) {
95                  minsa = sa[i]; maxsa = sa[i];
```

```cpp
            } else {
                minsa = min(minsa, sa[i]);
                maxsa = max(maxsa, sa[i]);
                if (maxsa - minsa >= K) return 1;
            }
        }
        return 0;
    }

    int checkBelongToDifferentSubstring(int K, int split) {
        int minsa = 0, maxsa = 0;
        for (int i = 0; i < L; ++i) {
            if (height[i] < K) {
                minsa = sa[i]; maxsa = sa[i];
            } else {
                minsa = min(minsa, sa[i]);
                maxsa = max(maxsa, sa[i]);
                if (maxsa > split && minsa < split) return 1;
            }
        }
        return 0;
    }

} *S;

int main() {
    string s, t;
    cin >> s >> t;
    int sp = s.length();
    s += "*" + t;
    S = new SuffixArray(s);
    S->getHeight();
    int left = 0, right = sp;
    while (left < right) {
        int mid = (left + right + 1) / 2;
        if (S->checkBelongToDifferentSubstring(mid, sp))
            left = mid;
        else right = mid - 1;
    }
    printf("%d\n", left);
}
```

- SA-IS
- 仅在后缀自动机被卡内存或者卡常且需要 O(1) LCA 的情况下使用（比赛中敲这个我觉得不行）

```cpp
//  rk [0..len-1] -> [1..len], sa/ht [1..len]
// s[i] > 0 && s[len] = 0
template<size_t size>
struct SuffixArray {
    bool type[size << 1];
    int bucket[size], bucket1[size];
    int sa[size], rk[size], ht[size];
    inline bool isLMS(const int i, const bool *type) { return i > 0 && type[i] && !type[i - 1]; }
    template<class T>
    inline void inducedSort(T s, int *sa, const int len, const int sigma, const int bucketSize, bool *type, int
        *bucket, int *cntbuf, int *p) {
        memset(bucket, 0, sizeof(int) * sigma);
        memset(sa, -1, sizeof(int) * len);
        for (int i = 0; i < len; i++) bucket[s[i]]++;
        cntbuf[0] = bucket[0];
        for (int i = 1; i < sigma; i++) cntbuf[i] = cntbuf[i - 1] + bucket[i];
        for (int i = bucketSize - 1; i >= 0; i--) sa[--cntbuf[s[p[i]]]] = p[i];
        for (int i = 1; i < sigma; i++) cntbuf[i] = cntbuf[i - 1] + bucket[i - 1];
        for (int i = 0; i < len; i++) if (sa[i] > 0 && !type[sa[i] - 1]) sa[cntbuf[s[sa[i] - 1]]++] = sa[i] - 1;
        cntbuf[0] = bucket[0];
        for (int i = 1; i < sigma; i++) cntbuf[i] = cntbuf[i - 1] + bucket[i];
        for (int i = len - 1; i >= 0; i--) if (sa[i] > 0 && type[sa[i] - 1]) sa[--cntbuf[s[sa[i] - 1]]] = sa[i] - 1;
    }
    template<typename T>
    inline void sais(T s, int *sa, int len, bool *type, int *bucket, int *bucket1, int sigma) {
        int i, j, bucketSize = 0, cnt = 0, p = -1, x, *cntbuf = bucket + sigma;
```

```
26          type[len - 1] = 1;
27          for (i = len - 2; i >= 0; i--) type[i] = s[i] < s[i + 1] || (s[i] == s[i + 1] && type[i + 1]);
28          for (i = 1; i < len; i++) if (type[i] && !type[i - 1]) bucket1[bucketSize++] = i;
29          inducedSort(s, sa, len, sigma, bucketSize, type, bucket, cntbuf, bucket1);
30          for (i = bucketSize = 0; i < len; i++) if (isLMS(sa[i], type)) sa[bucketSize++] = sa[i];
31          for (i = bucketSize; i < len; i++) sa[i] = -1;
32          for (i = 0; i < bucketSize; i++) {
33              x = sa[i];
34              for (j = 0; j < len; j++) {
35                  if (p == -1 || s[x + j] != s[p + j] || type[x + j] != type[p + j]) { cnt++, p = x; break; }
36                  else if (j > 0 && (isLMS(x + j, type) || isLMS(p + j, type))) break;
37              }
38              x = (~x & 1 ? x >> 1 : x - 1 >> 1), sa[bucketSize + x] = cnt - 1;
39          }
40          for (i = j = len - 1; i >= bucketSize; i--) if (sa[i] >= 0) sa[j--] = sa[i];
41          int *s1 = sa + len - bucketSize, *bucket2 = bucket1 + bucketSize;
42          if (cnt < bucketSize) sais(s1, sa, bucketSize, type + len, bucket, bucket1 + bucketSize, cnt);
43          else for (i = 0; i < bucketSize; i++) sa[s1[i]] = i;
44          for (i = 0; i < bucketSize; i++) bucket2[i] = bucket1[sa[i]];
45          inducedSort(s, sa, len, sigma, bucketSize, type, bucket, cntbuf, bucket2);
46      }
47      template<typename T>
48      inline void getHeight(T s, const int len, const int *sa) {
49          for (int i = 0, k = 0; i < len; i++) {
50              if (rk[i] == 0) k = 0;
51              else {
52                  if (k > 0) k--;
53                  int j = sa[rk[i] - 1];
54                  while (i + k < len && j + k < len && s[i + k] == s[j + k]) k++;
55              }
56              ht[rk[i]] = k;
57          }
58      }
59      template<class T>
60      inline void init(T s, int len, int sigma) {
61          sais(s, sa, ++len, type, bucket, bucket1, sigma);
62          for (int i = 1; i < len; i++) rk[sa[i]] = i;
63          getHeight(s, len, sa);
64      }
65  };
```

KMP 自动机

```
1   int m; int pat[N];
2   namespace kmp {
3       int f[N];   // f[i] 表示已匹配成功 i 个, 失配要去哪里
4
5       template<typename T>
6       int go(int stat, T c, bool& acc) {
7           // stat 是当前态 (表示已经匹配了 stat 个字符), c 是要走的边
8           while (stat && c != pat[stat]) stat = f[stat];
9           if (c == pat[stat]) stat++;
10          if (stat == m) acc = true;
11          return stat;
12      }
13
14      void getFail() {
15          static int f2[N];
16          f[0] = f[1] = 0;
17          f2[0] = f2[1] = 0;
18          FOR (i, 1, m) {
19              int j = f2[i];
20              while (j && pat[i] != pat[j]) j = f2[j];
21              f2[i+1] = f[i+1] = (pat[i] == pat[j]) ? j+1 : 0;
22              if (f[i+1] == j+1 && pat[i+1] == pat[j+1]) f[i+1] = f[j+1];
23          }
24          FOR (i, 0, m) dbg(i, f[i]);
25      }
26  }
```

- 拓展 KMP

```cpp
#include <bits/stdc++.h>
using namespace std;

/*
 Define template S, pattern T, len(S)=n, len(T)=m
 Find the longest common prefix of T and every suffix of S
 ex[i]: the LCP between T and S[i..n-1]
 */

const int maxn = 1e6 + 10;
int nt[maxn], ex[maxn];
char s[maxn], t[maxn];

void get_next(char *str) {
    int i = 0, j, po, len = strlen(str);
    nt[0] = len;
    while (str[i] == str[i + 1] && i + 1 < len)
        i++;
    nt[1] = i;
    po = 1;
    for (i = 2; i < len; i++) {
        if (nt[i - po] + i < nt[po] + po)
            nt[i] = nt[i - po];
        else {
            j = nt[po] + po - i;
            if (j < 0) j = 0;
            while (i + j < len && str[j] == str[j + i])
                j++;
            nt[i] = j;
            po = i;
        }
    }
}

void exkmp(char *s1, char *s2) {
    int i = 0, j, po, len = strlen(s1), l2 = strlen(s2);
    get_next(s2);
    while (s1[i] == s2[i] && i < l2 && i < len)
        i++;
    ex[0] = i;
    po = 0;
    for (i = 1; i < len; i++) {
        if (nt[i - po] + i < ex[po] + po)
            ex[i] = nt[i - po];
        else {
            j = ex[po] + po - i;
            if (j < 0) j = 0;
            while (i + j < len && j < l2 && s1[j + i] == s2[j])
                j++;
            ex[i] = j;
            po = i;
        }
    }
}

int main() {
    const int modn = 1e9 + 7;
    int T; scanf("%d", &T);
    while (T--) {
        memset(nt, 0, sizeof nt);
        memset(ex, 0, sizeof ex);
        scanf("%s", s); scanf("%s", t);
        int slen = strlen(s), tlen = strlen(t);
        reverse(s, s + slen);
        reverse(t, t + tlen);
        exkmp(s, t);
        int ans = 0;
        for (int i = 0; i < slen; ++i)
            ans = (ans + 1LL * ex[i] * (ex[i] + 1) / 2) % modn;
        printf("%d\n", ans);
```

```
71          }
72      }
```

Trie

```
1  namespace trie {
2      int t[N][26], sz, ed[N];
3      void init() { sz = 2; memset(ed, 0, sizeof ed); }
4      int _new() { memset(t[sz], 0, sizeof t[sz]); return sz++; }
5      void ins(char* s, int p) {
6          int u = 1;
7          FOR (i, 0, strlen(s)) {
8              int c = s[i] - 'a';
9              if (!t[u][c]) t[u][c] = _new();
10             u = t[u][c];
11         }
12         ed[u] = p;
13     }
14 }
```

AC 自动机

```
1  const int N = 1e6 + 100, M = 26;
2
3  int mp(char ch) { return ch - 'a'; }
4
5  struct ACA {
6      int ch[N][M], danger[N], fail[N];
7      int sz;
8      void init() {
9          sz = 1;
10         memset(ch[0], 0, sizeof ch[0]);
11         memset(danger, 0, sizeof danger);
12     }
13     void insert(const string &s, int m) {
14         int n = s.size(); int u = 0, c;
15         FOR (i, 0, n) {
16             c = mp(s[i]);
17             if (!ch[u][c]) {
18                 memset(ch[sz], 0, sizeof ch[sz]);
19                 danger[sz] = 0; ch[u][c] = sz++;
20             }
21             u = ch[u][c];
22         }
23         danger[u] |= 1 << m;
24     }
25     void build() {
26         queue<int> Q;
27         fail[0] = 0;
28         for (int c = 0, u; c < M; c++) {
29             u = ch[0][c];
30             if (u) { Q.push(u); fail[u] = 0; }
31         }
32         while (!Q.empty()) {
33             int r = Q.front(); Q.pop();
34             danger[r] |= danger[fail[r]];
35             for (int c = 0, u; c < M; c++) {
36                 u = ch[r][c];
37                 if (!u) {
38                     ch[r][c] = ch[fail[r]][c];
39                     continue;
40                 }
41                 fail[u] = ch[fail[r]][c];
42                 Q.push(u);
43             }
44         }
45     }
```

```
46    } ac;
47
48    char s[N];
49
50    int main() {
51        int n; scanf("%d", &n);
52        ac.init();
53        while (n--) {
54            scanf("%s", s);
55            ac.insert(s, 0);
56        }
57        ac.build();
58
59        scanf("%s", s);
60        int u = 0; n = strlen(s);
61        FOR (i, 0, n) {
62            u = ac.ch[u][mp(s[i])];
63            if (ac.danger[u]) {
64                puts("YES");
65                return 0;
66            }
67        }
68        puts("NO");
69        return 0;
70    }
```

杂项

STL

- copy

```
1    template <class InputIterator, class OutputIterator>
2      OutputIterator copy (InputIterator first, InputIterator last, OutputIterator result);
```

- merge (如果相等，第一个优先)

```
1    template <class InputIterator1, class InputIterator2,
2            class OutputIterator, class Compare>
3      OutputIterator merge (InputIterator1 first1, InputIterator1 last1,
4                        InputIterator2 first2, InputIterator2 last2,
5                        OutputIterator result, Compare comp);
```

- for_each

```
1    template <class InputIterator, class Function>
2      Function for_each (InputIterator first, InputIterator last, Function fn);
```

- transform

```
1    template <class InputIterator, class OutputIterator, class UnaryOperation>
2      OutputIterator transform (InputIterator first1, InputIterator last1,
3                            OutputIterator result, UnaryOperation op);
```

- numeric_limits

```
1    template <class T> numeric_limits;
```

- iota

```
1    template< class ForwardIterator, class T >
2    void iota( ForwardIterator first, ForwardIterator last, T value );
```

日期

```
1    // Routines for performing computations on dates.  In these routines,
2    // months are exprsesed as integers from 1 to 12, days are expressed
3    // as integers from 1 to 31, and years are expressed as 4-digit
```

```
4   // integers.
5
6   string dayOfWeek[] = {"Mo", "Tu", "We", "Th", "Fr", "Sa", "Su"};
7
8   // converts Gregorian date to integer (Julian day number)
9
10  int DateToInt (int m, int d, int y){
11    return
12      1461 * (y + 4800 + (m - 14) / 12) / 4 +
13      367 * (m - 2 - (m - 14) / 12 * 12) / 12 -
14      3 * ((y + 4900 + (m - 14) / 12) / 100) / 4 +
15      d - 32075;
16  }
17
18  // converts integer (Julian day number) to Gregorian date: month/day/year
19
20  void IntToDate (int jd, int &m, int &d, int &y){
21    int x, n, i, j;
22
23    x = jd + 68569;
24    n = 4 * x / 146097;
25    x -= (146097 * n + 3) / 4;
26    i = (4000 * (x + 1)) / 1461001;
27    x -= 1461 * i / 4 - 31;
28    j = 80 * x / 2447;
29    d = x - 2447 * j / 80;
30    x = j / 11;
31    m = j + 2 - 12 * x;
32    y = 100 * (n - 49) + i + x;
33  }
34
35  // converts integer (Julian day number) to day of week
36
37  string IntToDay (int jd){
38    return dayOfWeek[jd % 7];
39  }
```

子集枚举

● 枚举真子集

```
1   for (int s = (S - 1) & S; s; s = (s - 1) & S)
```

● 枚举大小为 k 的子集

```
1   template<typename T>
2   void subset(int k, int n, T&& f) {
3       int t = (1 << k) - 1;
4       while (t < 1 << n) {
5           f(t);
6           int x = t & -t, y = t + x;
7           t = ((t & ~y) / x >> 1) | y;
8       }
9   }
```

权值最大上升子序列

```
1   const LL maxn = 1E5 + 10;
2   const LL INF = 1E10;
3   struct P {
4       LL k, v;
5       bool operator < (const P& rhs) const {
6           return k < rhs.k || (k == rhs.k && v < rhs.v);
7       }
8   };
9   LL k[maxn], v[maxn], n, T;
10  set<P> s;
11
```

```cpp
int main() {
    cin >> T;
    while (T--) {
        s.clear();
        s.insert({-INF, 0});
        cin >> n;
        FOR (i, 0, n) scanf("%lld", &k[i]);
        FOR (i, 0, n) scanf("%lld", &v[i]);
        FOR (i, 0, n) {
            auto it = s.lower_bound({k[i], INF});
            LL vv = (--it)->v + v[i];
            ++it;
            while (it != s.end() && it->v <= vv)
                it = s.erase(it);
            if (it == s.end() || it->k != k[i]) s.insert({k[i], vv});
        }
        cout << s.rbegin()->v << endl;
    }
}
```

数位 DP

```cpp
LL dfs(LL base, LL pos, LL len, LL s, bool limit) {
    if (pos == -1) return s ? base : 1;
    if (!limit && dp[base][pos][len][s] != -1) return dp[base][pos][len][s];
    LL ret = 0;
    LL ed = limit ? a[pos] : base - 1;
    FOR (i, 0, ed + 1) {
        tmp[pos] = i;
        if (len == pos)
            ret += dfs(base, pos - 1, len - (i == 0), s, limit && i == a[pos]);
        else if (s &&pos < (len + 1) / 2)
            ret += dfs(base, pos - 1, len, tmp[len - pos] == i, limit && i == a[pos]);
        else
            ret += dfs(base, pos - 1, len, s, limit && i == a[pos]);
    }
    if (!limit) dp[base][pos][len][s] = ret;
    return ret;
}

LL solve(LL x, LL base) {
    LL sz = 0;
    while (x) {
        a[sz++] = x % base;
        x /= base;
    }
    return dfs(base, sz - 1, sz - 1, 1, true);
}
```

土制 bitset

```cpp
// M 要开大至少 1 个 64
const int M = (1E4 + 200) / 64;
typedef unsigned long long ULL;
const ULL ONE = 1;

struct Bitset {
    ULL a[M];
    void go(int x) {
        int offset = x / 64; x %= 64;
        for (int i = offset, j = 0; i + 1 < M; ++i, ++j) {
            a[j] |= a[i] >> x;
            if (x) a[j] |= a[i + 1] << (64 - x); // 不能左移 64 位
        }
    }
    void init() { memset(a, 0, sizeof a); }
    void set(int x) {
```

```
17          int offset = x / 64; x %= 64;
18          a[offset] |= (ONE << x);
19      }
20      void prt() {
21          FOR (i, 0, M) FOR (j, 0, 64) putchar((a[i] & (ONE << j)) ? '1' : '0');
22          puts("");
23      }
24      int lowbit() {
25          FOR (i, 0, M) if (a[i]) return i * 64 + __builtin_ctzll(a[i]);
26          assert (0);
27      }
28      int highbit(int x) {
29          // [0,x) 的最高位
30          int offset = x / 64; x %= 64;
31          FORD (i, offset, -1) {
32              if (!a[i]) continue;
33              if (i == offset) {
34                  FORD (j, x - 1, -1) if ((ONE << j) & a[i]) { return i * 64 + j; }
35              } else return i * 64 + 63 - __builtin_clzll(a[i]);
36          }
37          assert (0);
38      }
39  };
```

随机

- 不要使用 rand()。
- chrono::steady_clock::now().time_since_epoch().count() 可用于计时。
- 64 位可以使用 mt19937_64。

```
1   int main() {
2       mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());
3       vector<int> permutation(N);
4
5       for (int i = 0; i < N; i++)
6           permutation[i] = i;
7       shuffle(permutation.begin(), permutation.end(), rng);
8
9       for (int i = 0; i < N; i++)
10          permutation[i] = i;
11      for (int i = 1; i < N; i++)
12          swap(permutation[i], permutation[uniform_int_distribution<int>(0, i)(rng)]);
```

伪随机数

```
1   unsigned rnd() {
2       static unsigned A = 1 << 16 | 3, B = 33333331, C = 2341;
3       return C = A * C + B;
4   }
```

Java

Regex

```
1   // Code which demonstrates the use of Java's regular expression libraries.
2   // This is a solution for
3   //
4   //   Loglan: a logical language
5   //   http://acm.uva.es/p/v1/134.html
6
7   import java.util.*;
8   import java.util.regex.*;
9
10  public class LogLan {
11
```

```java
public static void main(String args[]) {

    String regex = BuildRegex();
    Pattern pattern = Pattern.compile(regex);

    Scanner s = new Scanner(System.in);
    while (true) {

        // In this problem, each sentence consists of multiple lines, where the last
        // line is terminated by a period.  The code below reads lines until
        // encountering a line whose final character is a '.'.  Note the use of
        //
        //    s.length() to get length of string
        //    s.charAt() to extract characters from a Java string
        //    s.trim() to remove whitespace from the beginning and end of Java string
        //
        // Other useful String manipulation methods include
        //
        //    s.compareTo(t) < 0 if s < t, lexicographically
        //    s.indexOf("apple") returns index of first occurrence of "apple" in s
        //    s.lastIndexOf("apple") returns index of last occurrence of "apple" in s
        //    s.replace(c,d) replaces occurrences of character c with d
        //    s.startsWith("apple) returns (s.indexOf("apple") == 0)
        //    s.toLowerCase() / s.toUpperCase() returns a new lower/uppercased string
        //
        //    Integer.parseInt(s) converts s to an integer (32-bit)
        //    Long.parseLong(s) converts s to a long (64-bit)
        //    Double.parseDouble(s) converts s to a double

        String sentence = "";
        while (true) {
            sentence = (sentence + " " + s.nextLine()).trim();
            if (sentence.equals("#")) return;
            if (sentence.charAt(sentence.length() - 1) == '.') break;
        }

        // now, we remove the period, and match the regular expression

        String removed_period = sentence.substring(0, sentence.length() - 1).trim();
        if (pattern.matcher(removed_period).find()) {
            System.out.println("Good");
        } else {
            System.out.println("Bad!");
        }
    }
}
}
```

Decimal Format

```java
// examples for printing floating point numbers

import java.util.*;
import java.io.*;
import java.text.DecimalFormat;

public class DecFormat {
    public static void main(String[] args) {
        DecimalFormat fmt;

        // round to at most 2 digits, leave of digits if not needed
        fmt = new DecimalFormat("#.##");
        System.out.println(fmt.format(12345.6789)); // produces 12345.68
        System.out.println(fmt.format(12345.0)); // produces 12345
        System.out.println(fmt.format(0.0)); // produces 0
        System.out.println(fmt.format(0.01)); // produces .1

        // round to precisely 2 digits
        fmt = new DecimalFormat("#.00");
        System.out.println(fmt.format(12345.6789)); // produces 12345.68
```

```java
21          System.out.println(fmt.format(12345.0)); // produces 12345.00
22          System.out.println(fmt.format(0.0)); // produces .00
23
24          // round to precisely 2 digits, force leading zero
25          fmt = new DecimalFormat("0.00");
26          System.out.println(fmt.format(12345.6789)); // produces 12345.68
27          System.out.println(fmt.format(12345.0)); // produces 12345.00
28          System.out.println(fmt.format(0.0)); // produces 0.00
29
30          // round to precisely 2 digits, force leading zeros
31          fmt = new DecimalFormat("000000000.00");
32          System.out.println(fmt.format(12345.6789)); // produces 000012345.68
33          System.out.println(fmt.format(12345.0)); // produces 000012345.00
34          System.out.println(fmt.format(0.0)); // produces 000000000.00
35
36          // force leading '+'
37          fmt = new DecimalFormat("+0;-0");
38          System.out.println(fmt.format(12345.6789)); // produces +12346
39          System.out.println(fmt.format(-12345.6789)); // produces -12346
40          System.out.println(fmt.format(0)); // produces +0
41
42          // force leading positive/negative, pad to 2
43          fmt = new DecimalFormat("positive 00;negative 0");
44          System.out.println(fmt.format(1)); // produces "positive 01"
45          System.out.println(fmt.format(-1)); // produces "negative 01"
46
47          // qoute special chars (#)
48          fmt = new DecimalFormat("text with '#' followed by #");
49          System.out.println(fmt.format(12.34)); // produces "text with # followed by 12"
50
51          // always show "."
52          fmt = new DecimalFormat("#.#");
53          fmt.setDecimalSeparatorAlwaysShown(true);
54          System.out.println(fmt.format(12.34)); // produces "12.3"
55          System.out.println(fmt.format(12)); // produces "12."
56          System.out.println(fmt.format(0.34)); // produces "0.3"
57
58          // different grouping distances:
59          fmt = new DecimalFormat("#,####.###");
60          System.out.println(fmt.format(123456789.123)); // produces "1,2345,6789.123"
61
62          // scientific:
63          fmt = new DecimalFormat("0.000E00");
64          System.out.println(fmt.format(123456789.123)); // produces "1.235E08"
65          System.out.println(fmt.format(-0.000234)); // produces "-2.34E-04"
66
67          // using variable number of digits:
68          fmt = new DecimalFormat("0");
69          System.out.println(fmt.format(123.123)); // produces "123"
70          fmt.setMinimumFractionDigits(8);
71          System.out.println(fmt.format(123.123)); // produces "123.12300000"
72          fmt.setMaximumFractionDigits(0);
73          System.out.println(fmt.format(123.123)); // produces "123"
74
75          // note: to pad with spaces, you need to do it yourself:
76          // String out = fmt.format(...)
77          // while (out.length() < targlength) out = " "+out;
78      }
79  }
```

Sort

```java
1  import java.util.ArrayList;
2  import java.util.Collections;
3  import java.util.List;
4
5  public class Employee implements Comparable<Employee> {
6      private int id;
7      private String name;
8      private int age;
```

```java
9
10      public Employee(int id, String name, int age) {
11          this.id = id;
12          this.name = name;
13          this.age = age;
14      }
15
16      @Override
17      public int compareTo(Employee o) {
18          if (id > o.id) {
19              return 1;
20          } else if (id < o.id) {
21              return -1;
22          }
23          return 0;
24      }
25
26      public static void main(String[] args) {
27          List<Employee> list = new ArrayList<Employee>();
28          list.add(new Employee(2, "Java", 20));
29          list.add(new Employee(1, "C", 30));
30          list.add(new Employee(3, "C#", 10));
31          Collections.sort(list);
32      }
33  }
```

扩栈（本地使用）

```c
1   #include <sys/resource.h>
2   void init_stack(){
3       const rlim_t kStackSize = 512 * 1024 * 1024;
4       struct rlimit rl;
5       int result;
6       result = getrlimit(RLIMIT_STACK, &rl);
7       if (result == 0) {
8           if (rl.rlim_cur < kStackSize) {
9               rl.rlim_cur = kStackSize;
10              result = setrlimit(RLIMIT_STACK, &rl);
11              if (result != 0) {
12                  fprintf(stderr, "setrlimit returned result = %d\n", result);
13              }
14          }
15      }
16  }
```

心态崩了

- (int)v.size()
- 1LL << k
- 递归函数用全局或者 static 变量要小心
- 预处理组合数注意上限
- 想清楚到底是要 multiset 还是 set
- 提交之前看一下数据范围，测一下边界
- 数据结构注意数组大小（2 倍，4 倍）
- 字符串注意数据集
- 如果函数中使用了默认参数的话，注意调用时的参数个数。
- 注意要读完
- 构造参数无法使用自己
- 树链剖分/dfs 序，初始化或者询问不要忘记 idx, ridx
- 排序时注意结构体的所有属性是不是考虑了
- 不要把 while 写成 if
- 不要把 int 开成 char
- 清零的时候全部用 0~n+1。

- 模意义下不要用除法
- 哈希不要自然溢出
- 最短路不要 SPFA，乖乖写 Dijkstra
- 上取整以及 GCD 小心负数
- mid 用 `l + (r - l) / 2` 可以避免溢出和负数的问题