Contents

数学

Fermat 小定理 给定质数 p，对于任意整数 a，有

$$a^p \equiv a \pmod p$$

扩展 Euler 定理 对于任意整数 a，正整数 m，和非负整数 n，有

$$a^n \equiv \begin{cases} a^{n \bmod \varphi(m)} & \text{if} \gcd(a,m) = 1 \\ a^m & \text{if} \gcd(a,m) \neq 1, n < \varphi(m) \\ a^{n \bmod \varphi(m) + \varphi(m)} & \text{if} \gcd(a,m) \neq 1, n \geq \varphi(m) \end{cases} \pmod m$$

Lucas 定理 对于素数 p，有

$$\binom{n}{m} \equiv \binom{\lfloor n/p \rfloor}{\lfloor m/p \rfloor} \binom{n \bbmod p}{m \bmod p} \pmod p$$

```cpp
int lucas(ll n, ll m, int p) {
  if (m == 0) return 1;
  return ll(binom(n % p, m % p, p)) * lucas(n / p, m / p, p) % p;
}
```

中国剩余定理 给定若干线性同余方程 $x \equiv a_i \pmod{m_i}$，如果模数两两互质，那么令 $m = \prod_{i=1}^{n} m_i, M_i = m/m_i$，并且 M_i^{-1} 是 M_i 模 m_i 意义下的逆元，那么方程组有整数解，解为 $x \equiv \sum_{i=1}^{n} a_i M_i M_i^{-1} \pmod{m}$.

```cpp
pll crt(const vector<int>& a, const vector<int>& p) {
  assert(a.size() == p.size());
  ll x = 0, m = 1;
  for (int x : p) m *= x;
  for (int i = 0; i < int(a.size()); i++) {
    ll M = m / p[i], inv = getinv(M, p[i]);
    x = (x + i128(a[i]) * M * inv % m) % m;
  }
  return {x, m};
}
```

扩展中国剩余定理 考虑两个模数不互质的线性同余方程．第一个方程的解形如 $x = a_1 + k_1 m_1$，那么第二个方程要存在一个整数 t，满足 $a_1 + t m_1 \equiv a_2 \pmod{m_2}$，等价于 $t m_1 \equiv a_2 - a_1 \pmod{m_2}$，这是一个关于变量 t 的线性同余方程．

如果 $a_2 - a_1$ 不能被 $\gcd(m_1, m_2)$ 整除，那么方程无解．

否则解出一个 t，两个方程的解为 $x^* \equiv t m_1 + a_1 \pmod{\operatorname{lcm}(m_1, m_2)}$.

对于多个模数不互质的线性同余方程，可以利用上面的方法合并前 k 个方程组与第 $k+1$ 个方程．

```cpp
pll excrt(const vector<int>& a, const vector<int>& m) {
  assert(a.size() == m.size());
  ll x = a[0] % m[0], M = m[0];
  for (int i = 1; i < int(a.size()); i++) {
    ll r1, r2, g = exgcd(M, m[i], r1, r2);
    if ((a[i] - x) % g != 0) return {-1, -1};
    i128 k1 = i128(a[i] - x) / g * r1;
    i128 tmp = x + k1 * M;
    M = lcm(M, m[i]);
    x = (tmp % M + M) % M;
  }
  return {x, M};
}
```

组合数的性质

$$\binom{n}{m} = \binom{n-1}{m} + \binom{n-1}{m-1}$$

$$\binom{n}{m} = \frac{n}{m}\binom{n-1}{m-1}$$

$$\binom{n}{k}\binom{k}{m} = \binom{n}{m}\binom{n-m}{k-m}$$

带权求和：

$$\sum_{i=0}^{n} i\binom{n}{i} = n2^{n-1}$$

$$\sum_{i=0}^{n} i^2\binom{n}{i} = n(n+1)2^{n-2}$$

Vandermonde 恒等式：

$$\sum_{i=0}^{k}\binom{n}{i}\binom{m}{k-i} = \binom{n+m}{k}$$

朱世杰恒等式：

$$\sum_{i=0}^{n}\binom{i}{m} = \binom{n+1}{m+1}$$

多重集的组合数 对于多重集 $S = \{n_1 \cdot a_1, n_2 \cdot a_2, ..., n_k \cdot a_k\}$，记从 S 中选出 $r \in \mathbb{N}^*$ 个元素组成一个多重集的方案数为 C.

如果 $r < n_i$ 恒成立，使用插板法可知

$$C = \binom{r+k-1}{k-1}$$

否则需要使用容斥原理，全集是 $\sum_{i=1}^{k} x_i = r$ 的非负整数解，S_i 是满足 $x_i \le n$ 的解的集合，那么有：

$$\left|\bigcup_{i=1}^{k}\overline{S_i}\right| = \sum_{i=1}^{k}\left|\overline{S_i}\right| - \sum_{1\le i_1<i_2\le k}\left|\overline{S_{i_1}}\cap\overline{S_{i_2}}\right| + \sum_{1\le i_1<i_2<i_3\le k}\left|\overline{S_{i_1}}\cap\overline{S_{i_2}}\cap\overline{S_{i_3}}\right| - \cdots + (-1)^{k-1}\left|\bigcap_{i=1}^{k}\overline{S_i}\right|$$

$$= \sum_{i=1}^{k}\binom{k+r-n_i-2}{k-1} - \sum_{1\le i_1<i_2\le k}\binom{k+r-n_{i_1}-n_{i_2}-3}{k-1}$$

$$+ \sum_{1\le i_1<i_2<i_3\le k}\binom{k+r-n_{i_1}-n_{i_2}-n_{i_3}-4}{k-1} - \cdots + (-1)^{k-1}\binom{k+r-\sum_{i=1}^{k}n_i-k-1}{k-1}$$

最后答案为

$$C = |U| - \left|\bigcup_{i=1}^{k} \overline{S_i}\right| = \sum_{p=0}^{k} (-1)^p \sum_{|A|=p} \binom{k + r - \sum_{i \in A} n_i - (p+1)}{k-1}$$

实现就是二进制枚举子集 A.

Catlan 数 $C_0 = C_1 = 1$，当 $n \geq 2$ 时，有

$$\begin{aligned} C_n &= \frac{1}{n+1}\binom{2n}{n} = \prod_{i=1}^{n} C_{i-1} C_{n-i} \\ &= \frac{4n-2}{n+1} C_{n-1} = \binom{2n}{n} - \binom{2n}{n-1} \end{aligned}$$

常见组合意义：由 n 个 $+1$ 与 n 个 -1 组成的数列 $a_1, a_2, \ldots, a_{2n}$，其部分和总是满足 $a_1 + a_2 + \cdots + a_k \geq 0 \quad (k = 1, 2, 3, \ldots, 2n)$ 的方案数为 C_n.

数论分块 对于形如

$$\sum_{i=1}^{n} f(i) g\left(\left\lfloor \frac{n}{i} \right\rfloor\right)$$

的和式，如果可以 $\mathcal{O}(1)$ 计算出 $f(i)$ 的区间和，那么可以利用数论分块在 $\mathcal{O}(\sqrt{n})$ 的时间内完成计算.

核心思想是给定正整数 n, l，如果令 $r = \left\lfloor \frac{n}{\lfloor n/l \rfloor} \right\rfloor$，那么 $\forall x \in [l, r]$，有 $\left\lfloor \frac{n}{x} \right\rfloor$ 为定值.

下面是 $f(x) = 1, g(x) = x$ 时的特例.

```C++
i128 floor_sum(ll n) {
  i128 ret = 0;
  for (ll l = 1, r; l <= n; l = r) {
    r = n / (n / l) + 1;
    ret += (r - l) * (n / l);
  }
  return ret;
}
```

大步小步算法 对于高次同余方程 $a^x \equiv b \pmod{p}$，如果 a, p 互质，令 $t = \lceil \sqrt{p} \rceil$，可以把 x 改写为 $i \times t - j$，其中 $0 \leq i \leq t, 0 \leq j < t$. 方程变为 $(a^t)^i \equiv b \times a^j \pmod{p}$，先枚举 j 把右边结果插入哈希表，再枚举 i 寻找是否有对应元素.

生成函数 普通生成函数 OGF 的形式为 $F(x) = \sum_{n=0}^{\infty} a_n x^n$，指数生成函数 EGF 的形式为 $F(x) = \sum_{n=0}^{\infty} \frac{a_n}{x_n} x^n$.

应用：

从 n 个球中选出若干个放入袋子 $1, 2, 3$，要求袋子 1 放入 a 倍数个，袋子 2 放入 b 倍数个，袋子 3 放入 c 倍数个.

如果这 n 个球全部相同，那么方案数为多项式

$$G(x) = \left(\sum_{n=0}^{\infty} x^{na}\right) \times \left(\sum_{n=0}^{\infty} x^{nb}\right) \times \left(\sum_{n=0}^{\infty} x^{nc}\right)$$

的第 n 项.

如果这 n 个球两两不同，那么方案数为多项式

$$G(x) = \left(\sum_{n=0}^{\infty} \frac{1}{(na)!} x^{na}\right) \times \left(\sum_{n=0}^{\infty} \frac{1}{(nb)!} x^{nb}\right) \times \left(\sum_{n=0}^{\infty} \frac{1}{(nc)!} x^{nc}\right)$$

的第 n 项.

Z 变换 设全集 $X = \{1, 2, \ldots, n\}$，对每个子集 $S \subseteq X$，有一个值 $f(S)$. 如果 $g(S)$ 是对所有子集求和的结果，$h(S)$ 是对所有超集求和的结果. 利用 $f(S)$ 可以 Z 变换得到 $g(S)$ 或者 $h(S)$，也可以从 $g(S)$ 或者 $h(S)$ 逆变换回 $f(S)$.（逆变换也称为集合上的 Mobius 变换）

$$g(S) = \sum_{T \subseteq S} f(T) \Leftrightarrow f(S) = \sum_{T \subseteq S} (-1)^{|S|-|T|} g(T)$$

$$h(S) = \sum_{T \supseteq S} f(T) \Leftrightarrow f(S) = \sum_{T \supseteq S} (-1)^{|T|-|S|} h(T)$$

应用：

有 n 个不同的约束条件对应全集，记 $f(S)$ 为恰好只满足 $i \in S$ 的条件的方案数. 那么 $g(S)$ 相当于只可能满足 $i \in S$ 的条件的方案数；$h(S)$ 相当于至少满足 $i \in S$ 的条件的方案数.

图论

基环树

```C++
//  遍历基环树森林中的一棵树，并标记出环.
//  - 时间复杂度 $\mathcal{O}(n)$.
//  - FIXME: 如果有自环，那么这个环不会被标记 `vis[v] = 3`.
void cyctree() {
  int n;                      // Input.
  vector<vector<pii>> G(n);   // Input. First: to, second: edge id.
  vector<int> vis(n);         // Input.
```

```
  auto assign_vis = [&](auto&& self, int u, int fid) -> void {
    vis[u] = 1;
    for (auto [v, id] : G[u]) {
      if (id == fid) continue;
      if (vis[v] == 1) {
        vis[v] = 3;
        vis[u] = 2;
        continue;
      } else if (vis[v] == 2) {
        continue;
      }
      self(self, v, id);
      if (vis[v] == 2) {
        if (vis[v] != 3) vis[v] = 2;
      }
    }
  };
}
```

差分约束系统 对于约束条件 $x_i \le x_j + w$，从 j 到 i 连权值为 w 的边，还需要建立超级源想每个点连权值为 n 的边，跑最短路，存在负环则无解.

杂项

树上背包 有 n 门课程，第 i 门课程的学分为 a_i，每门课程有零门或一门先修课，有先修课的课程需要先学完其先修课，才能学习该课程．一位学生要学习 m 门课程，求能获得的最大学分数．这里定义 $f[u][i]$ 为从 u 为根的树中选出 i 门课程的最大得分，时间复杂度 $\mathcal{O}(nm)$.

C++
```
#include <bits/stdc++.h>
#define PUSHB push_back
using namespace std;

int main() {
  cin.tie(nullptr)->sync_with_stdio(false);
  int n, m;
  cin >> n >> m;
  m++;
  vector<int> a(n + 1);
  vector<vector<int>> G(n + 1);
  for (int i = 0; i < n; i++) {
    int fa;
    cin >> fa >> a[i];
    fa--;
    if (fa == -1) {
      fa = n;
    }
    G[fa].PUSHB(i);
  }

  vector<int> siz(n + 1);
  vector<vector<int>> f(n + 1, vector<int>(m + 1));
  auto dfs = [&](auto&& self, int u) -> void {
    siz[u] = 1;
    f[u][1] = a[u];
    for (int v : G[u]) {
      self(self, v);
      for (int j = min(m, siz[u] + siz[v]); j >= 1; j--) {
        for (int k = max(1, j - siz[u]); k <= min(j - 1, siz[v]); k++) {
          f[u][j] = max(f[u][j], f[u][j - k] + f[v][k]);
        }
      }
      siz[u] += siz[v];
    }
  };
  dfs(dfs, n);
  cout << f[n][m] << "\n";
  return 0;
}
```

主定理 如果有递推关系式

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

其中 $a \ge 1, b > 1, n \in \mathbb{N}$，那么

$$T(n) = \begin{cases} \Theta(n^{\log_b a}) & \text{if } f(n) = \mathcal{O}(n^{\log_b a - \varepsilon}) \text{ for some } \varepsilon > 0 \\ \Theta(n^{\log_b a} \log^{k+1} n) & \text{if } f(n) = \Theta(n^{\log_b a} \log^k n) \text{ for some } k \ge 0 \\ \Theta(f(n)) & \text{if } f(n) = \Omega(n^{\log_b a + \varepsilon}) \text{ for some } \varepsilon > 0 \end{cases}$$

其中最后一条还要求对于某个常数 $c < 1$ 以及所有充分大的 n，有 $af(n/b) \le cf(n)$.

环境配置

Vim
```
" .vimrc
```

```
colo slate
syn on
filet plugin indent on
se nocp nu hls is et sr sw=2 sts=2 cino+=:0,l1,L0 udf spr bo=all
se mouse=a cb=unnamedplus
au TerminalOpen * se nonu
nn <C-s> :w<CR>
au BufNewFile *.cpp sil! 0r _vim_template.cpp | $d _
nn <Esc><C-n> :vert ter ./_vim_run %:r<CR>
nn <Esc><CR> :vs ./cache/main.in<CR>
nn <F5> :vert ter gdb ./cache/%:r<CR>

" Some platform may need those.
se t_Co=256 bs=indent,col,start
lan en_US
```

gdb

```
# .gdbinit
set debuginfo enabled off
```

C++

```cpp
// _vim_template.cpp
#include <bits/stdc++.h>
#define ALL(a) (a).begin(), (a).end()
#define FOR(i, a, b) for (int i = (a); i < (b); i++)
#define PUSHB push_back
using namespace std;
using ll = long long;
using ull = unsigned long long;
using i128 = __int128_t;
using pii = pair<int, int>;
using pll = pair<ll, ll>;
using pli = pair<ll, int>;

void solve() {
}

int main() {
  cin.tie(nullptr)->sync_with_stdio(false);
#ifdef DEBUG
  freopen("cache/main.in", "r", stdin);
  freopen("cache/main.out", "w", stdout);
#endif
  int tt = 1;
  cin >> tt;
  while (tt--) {
    solve();
  }
  return 0;
}
```

Shell

```
#!/usr/bin/env bash
# _vim_run
set -e
ulimit -s 1048576
g++-14 -DDEBUG -std=c++17 -g "$1.cpp" -o "./cache/$1" \
  -fsanitize=address,undefined -Wall -Wpedantic -Wextra
"./cache/$1" && cat ./cache/main.out
```

Shell

```
#!/usr/bin/env bash
# _check
g++ _generator.cpp -O2 -std=c++17 -o ./cache/_generator
g++ main.cpp -O2 -std=c++17 -o ./cache/main
g++ std.cpp -O2 -std=c++17 -o ./cache/std
cnt=0
while true; do
  ./cache/_generator >./cache/main.in
  ./cache/main <./cache/main.in >./cache/main.out
  ./cache/std <./cache/main.in >./cache/std.out
  # diff <(head -n 1 ./cache/main.out) <(head -n 1 ./cache/std.out)
  diff ./cache/main.out ./cache/std.out
  if [ $? -ne 0 ]; then break; fi
  cnt=$((cnt + 1))
  if [ $((cnt % 100)) -eq 0 ]; then
    cnt=0
    echo "ok 100 times"
  fi
done
```

Shell

```
#!/usr/bin/env bash
# _check_spj
g++ _generator.cpp -DSPJ -O2 -std=c++17 -o ./cache/_generator
g++ main.cpp -DSPJ -O2 -std=c++17 -o ./cache/main
g++ _spj.cpp -DSPJ -O2 -std=c++17 -o ./cache/_spj
cnt=0
while true; do
```

```
  ./cache/_generator >./cache/main.in
  ./cache/main <./cache/main.in >./cache/main.out
  ./cache/_spj <./cache/main.out || {
    echo "spj failed"
    break
  }
  cnt=$((cnt + 1))
  if [ $((cnt % 100)) -eq 0 ]; then
    cnt=0
    echo "ok 100 times"
  fi
done
```

C++

- std::cbrt(): 传入 double x, 计算 $\sqrt[3]{x}$. (更高精度有 std::cbrtl())
- std::iota(): iota(arr.begin(), arr.end(), 0), 从 0 开始填充整个 arr.
- std::string:
  - std::to_string(): 传入 long long, long double 之类的, 返回 std::string().
  - atoi(), std::stoi(): 字符串转为整数, 前者是 const char* 后者是 std::string. 更高精度有 std::stoll().
  - sscanf(), sprintf(): 类似与 scanf, printf, 不过第一个参数是 const char*.
  - std::getline(std::cin, buf): 从 std::cin 中读入一行内容放入 buf.
  - .find(): 查找某个 char 或者 std::string, 返回 size_t, 如果没找到返回 string::npos.
- std::cout 格式化输出
  - std::fixed 固定小数点, 不使用科学记数法.
  - std::setprecision(n): 保留小数点后 n 位.
  - std::setw(n) 指定最小宽度为 n, 默认右对齐, 填充空格.
  - std::setfill(ch): 指定对齐时填充的字符.
  - std::left, std::right: 指定左对齐或右对齐.
  - std::dec, std::hex, std::oct: 以 10/16/8 进制 输出.
- std::bitset 的操作
  - .count(): 返回 true 的数量.
  - .any(); .none(); .all(): 是否存在某一个 true; 不存在 true; 全部是 true.
  - .set(); .reset(): 全部设为 true; 全部设为 false.
  - .flip(): 翻转每一位.
  - .to_string(); .to_ulong(); .to_ullong(): 类型转换.
  - ._Find_first(); ._Find_next(pos): 返回第一个 true 的下标 (不存在返回 size()); 返回下标严格大于 pos 的第一个 true 的下标 (不存在返回 size()).
- sort(), unique(), erase(), next_permutation(), lower_bound(), upper_bound(): 用法略.

MATHEMATICS

comb

C++

```cpp
#include "modint.hpp"
vector<mint> fac(1, 1), ifac(1, 1);
void ensure_binom(int m) {
  int n = fac.size();
  if (m < n) return;
  fac.resize(m + 1), ifac.resize(m + 1);
  for (int i = n; i <= m; i++) fac[i] = fac[i - 1] * i;
  ifac[m] = fac[m].inv();
  for (int i = m; i > n; i--) ifac[i - 1] = ifac[i] * i;
}
mint binom(ll n, ll m) {
  if (n < m || m < 0) return 0;
  ensure_binom(n);
  return fac[n] * ifac[m] * ifac[n - m];
}
vector<vector<mint>> init_binom_table(int n) {
  vector<vector<mint>> binom(n + 1, vector<mint>(n + 1, 0));
  binom[0][0] = 1;
  for (int i = 1; i <= n; i++) {
    binom[i][0] = 1;
    for (int j = 1; j <= i; j++) {
      binom[i][j] = binom[i - 1][j - 1] + binom[i - 1][j];
    }
  }
  return binom;
}
```

gaussfp 求解线性方程组 $Ax = b$. 时间复杂度 $\mathcal{O}(n \times m \times min(n, m))$.

- 返回 0 说明有唯一解, 存储在 x 中.
- 返回正数 n 说明有无穷多解, n 是自由变量的个数, x 存储了一个解, 自由变量设为 0.
- 返回 -1 说明无解.

```cpp
template <typename fp>
int gauss_fp(const vector<vector<fp>>& A, vector<fp>& x, const vector<fp>& b) {
  const fp EPS = numeric_limits<fp>::epsilon() * fp(1e2);
  int n = A.size(), m = int(A[0].size());
  assert(int(b.size()) == n);
  vector<vector<fp>> aug(n, vector<fp>(m + 1));
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < m; j++) {
      aug[i][j] = A[i][j];
    }
    aug[i][m] = b[i];
  }
  vector<int> where(m, -1);
  int row = 0;
  for (int col = 0; col < m && row < n; col++) {
    int sel = row;
    for (int i = row; i < n; i++) {
      if (abs(aug[i][col]) > abs(aug[sel][col])) {
        sel = i;
      }
    }
    if (abs(aug[sel][col]) <= EPS) continue;
    swap(aug[sel], aug[row]);
    where[col] = row;
    for (int j = m; j >= col; j--) {
      for (int i = row + 1; i < n; i++) {
        aug[i][j] -= aug[i][col] / aug[row][col] * aug[row][j];
      }
    }
    row++;
  }
  for (int i = row; i < n; i++) {
    bool all_zero = true;
    for (int j = 0; j < m; j++) {
      if (abs(aug[i][j]) > EPS) {
        all_zero = false;
        break;
      }
    }
    if (all_zero && abs(aug[i][m]) > EPS) return -1;
  }
  x.assign(m, fp(0));
  for (int j = m - 1; j >= 0; j--) {
    if (where[j] == -1) continue;
    int r = where[j];
    fp sum = aug[r][m];
    for (int k = j + 1; k < m; k++) {
      sum -= aug[r][k] * x[k];
    }
    x[j] = sum / aug[r][j];
  }
  int free_vars = 0;
  for (int j = 0; j < m; j++) {
    if (where[j] == -1) {
      free_vars++;
    }
  }
  return free_vars;
};
```

i128gcd

```cpp
i128 gcd(i128 a, i128 b) {
  while (b != 0) {
    i128 t = b;
    b = a % b;
    a = t;
  }
  return a;
}
i128 lcm(i128 a, i128 b) { return a / gcd(a, b) * b; }
```

millerrabin 时间复杂度：$\mathcal{O}(k \log n)$，目前 $k = 9$ 足以判定 64 位整数.

```cpp
#include "pow_mod_ll.hpp"
bool isprime(ll n) {
  if (n < 2) return 0;
  int s = __builtin_ctzll(n - 1);
  ll d = (n - 1) >> s;
  for (int a : {2, 3, 5, 7, 11, 13, 17, 19, 23}) {
    if (a == n) return 1;
    ll x = pow_mod_ll(a, d, n);
    if (x == 1 || x == n - 1) continue;
```

```
    bool ok = 0;
    for (int i = 0; i + 1 < s; i++) {
      x = i128(x) * x % n;
      if (x == n - 1) {
        ok = 1;
        break;
      }
    }
    if (!ok) return 0;
  }
  return 1;
}
```

modint

C++

```
template <unsigned MOD>
struct ModInt {
  unsigned data;
  ModInt(ll v = 0) : data(norm(v % MOD)) {}
  ModInt operator-() const { return MOD - data; }
  ModInt& operator+=(ModInt rhs) { return data = norm(data + rhs.data), *this; }
  ModInt& operator-=(ModInt rhs) { return data = norm(data - rhs.data), *this; }
  ModInt& operator*=(ModInt rhs) {
    return data = ull(data) * rhs.data % MOD, *this;
  }
  ModInt& operator/=(ModInt rhs) {
    return data = ull(data) * rhs.inv() % MOD, *this;
  }
  friend ModInt operator+(ModInt lhs, ModInt rhs) { return lhs += rhs; }
  friend ModInt operator-(ModInt lhs, ModInt rhs) { return lhs -= rhs; }
  friend ModInt operator*(ModInt lhs, ModInt rhs) { return lhs *= rhs; }
  friend ModInt operator/(ModInt lhs, ModInt rhs) { return lhs /= rhs; }
  unsigned inv() const {
    ll x, y;  // Inverse does not exist if gcd(data, MOD) != 1.
    assert(exgcd(data, MOD, x, y) == 1);
    return norm(x);
  }
  ModInt pow(ull n) const { return pow_mod(data, n, MOD); }
  static ll exgcd(ll a, ll b, ll& x, ll& y) {
    x = 1, y = 0;
    ll x1 = 0, y1 = 1;
    while (b) {
      ll q = a / b;
      swap(a -= q * b, b);
      swap(x -= q * x1, x1);
      swap(y -= q * y1, y1);
    }
    return a;
  }
  static unsigned pow_mod(unsigned a, ull n, unsigned p) {
    unsigned ret = 1;
    for (; n; n /= 2) {
      if (n & 1) ret = ull(ret) * a % p;
      a = ull(a) * a % p;
    }
    return ret;
  }

 private:
  static unsigned norm(unsigned x) {
    if ((x >> (8 * sizeof(unsigned) - 1)) & 1) x += MOD;
    return x >= MOD ? x -= MOD : x;
  }
};
constexpr unsigned MOD = 998244353;
using mint = ModInt<MOD>;
```

ntt

C++

```
#include "modint.hpp"
template <int MOD = 998244353, int G = 114514, int GI = 137043501>
struct Poly {
  vector<mint> data;
  friend Poly operator*(const Poly& a, const Poly& b) {
    int n = 1, new_sz = a.data.size() + b.data.size() - 1;
    while (n < int(a.data.size() + b.data.size())) n *= 2;
    vector<mint> A = a.data, B = b.data;
    A.resize(n), B.resize(n);
    ntt(A, 1), ntt(B, 1);
    for (int i = 0; i < n; i++) A[i] *= B[i];
    ntt(A, -1), A.resize(new_sz);
    return {A};
  }
```

```
 private:
  static void ntt(vector<mint>& a, int type) {
    int len = a.size();
    vector<int> rev(len);
    for (int i = 0; i < len; i++) {
      rev[i] = rev[i >> 1] >> 1;
      if (i & 1) rev[i] |= len >> 1;
    }
    for (int i = 0; i < len; i++)
      if (i < rev[i]) swap(a[i], a[rev[i]]);
    for (int n = 2, m = 1; n <= len; n *= 2, m *= 2) {
      mint step = mint(type == 1 ? G : GI).pow((MOD - 1) / n);
      for (int j = 0; j < len; j += n) {
        mint w = 1;
        for (int k = j; k < j + m; k++) {
          mint u = a[k], v = w * a[k + m];
          a[k] = u + v, a[k + m] = u - v, w *= step;
        }
      }
    }
    if (type == -1) {
      mint inv = mint(len).pow(MOD - 2);
      for (int i = 0; i < len; i++) a[i] *= inv;
    }
  }
};
```

pollardrho 时间复杂度：期望 $\mathcal{O}(n^{1/4})$

C++

```
#include "millerrabin.hpp"
ll pollard_single(ll n) {
  if (n % 2 == 0) return 2;
  if (isprime(n)) return n;
  ll st = 0;
  auto f = [&](ll x) -> ll { return (i128(x) * x % n + st) % n; };
  while (true) {
    st++;
    ll x = st, y = f(x);
    while (true) {
      ll p = gcd(y - x + n, n);
      if (p == 0 || p == n) break;
      if (p != 1) return p;
      x = f(x), y = f(f(y));
    }
  }
}
vector<ll> breakdown(ll n) {
  if (n == 1) return {};
  ll x = pollard_single(n);
  if (x == n) return {n};
  auto l = breakdown(x), r = breakdown(n / x);
  l.insert(l.end(), ALL(r)), sort(ALL(l));
  return l;
}
```

pow_mod_ll

C++

```
ll pow_mod_ll(ll a, ull n, ll p) {
  ll ret = 1;
  for (; n; n /= 2) {
    if (n & 1) ret = i128(ret) * a % p;
    a = i128(a) * a % p;
  }
  return ret;
}
```

prime

C++

```
vector<int> primes;
bitset<MAXN> isprime;
void init_prime_table() {
  isprime.set();
  isprime[0] = isprime[1] = false;
  for (int i = 2; i < MAXN; i++) {
    if (isprime[i]) primes.PUSHB(i);
    for (int p : primes) {
      if (i * p >= MAXN) break;
      isprime[i * p] = false;
      if (i % p == 0) break;
    }
  }
}
```

proot 时间复杂度瓶颈在质因数分解.

```cpp
#include "pollardrho.hpp"
ll proot(ll p) {
  if (p == 2) return 1;
  ll phi = p - 1;
  auto factors = breakdown(phi);
  factors.erase(unique(ALL(factors)), factors.end());
  for (ll ret = 2; ret <= p; ret++) {
    bool ok = 1;
    for (int i = 0; i < int(factors.size()) && ok; i++) {
      if (pow_mod_ll(ret, phi / factors[i], p) == 1) {
        ok = 0;
        break;
      }
    }
    if (ok) return ret;
  }
  return -1;
}
```

vec2

```cpp
constexpr double EPS = 1e-10;
struct Vec2 {
  double x, y;
  Vec2(double x = 0, double y = 0) : x(x), y(y) {}

  Vec2 operator-() const { return {-x, -y}; }
  Vec2& operator+=(const Vec2& rhs) { return x += rhs.x, y += rhs.y, *this; }
  Vec2& operator-=(const Vec2& rhs) { return x -= rhs.x, y -= rhs.y, *this; }
  Vec2& operator*=(double k) { return x *= k, y *= k, *this; }
  Vec2& operator/=(double k) { return x /= k, y /= k, *this; }
  friend Vec2 operator+(Vec2 lhs, const Vec2& rhs) { return lhs += rhs; }
  friend Vec2 operator-(Vec2 lhs, const Vec2& rhs) { return lhs -= rhs; }
  friend Vec2 operator*(Vec2 lhs, double k) { return lhs *= k; }
  friend Vec2 operator*(double k, Vec2 rhs) { return rhs *= k; }
  friend Vec2 operator/(Vec2 lhs, double k) { return lhs /= k; }
  friend double dot(const Vec2& lhs, const Vec2& rhs) {
    return lhs.x * rhs.x + lhs.y * rhs.y;
  }
  friend double cross(const Vec2& lhs, const Vec2& rhs) {
    return lhs.x * rhs.y - lhs.y * rhs.x;
  }
  static int dcmp(double x) {
    if (abs(x) < EPS) return 0;
    return x < 0 ? -1 : 1;
  }
  friend bool operator==(const Vec2& lhs, const Vec2& rhs) {
    return dcmp(lhs.x - rhs.x) == 0 && dcmp(lhs.y - rhs.y) == 0;
  }
  friend bool operator<(const Vec2& lhs, const Vec2& rhs) {
    return tie(lhs.x, lhs.y) < tie(rhs.x, rhs.y);
  }
  double len_sq() const { return x * x + y * y; }
  double len() const { return sqrt(len_sq()); }
  Vec2 norm() const { return *this / len(); }
};
```

xorbasis 插入元素时间复杂度 $\mathcal{O}(L)$. 基底各数字二进制最高位不同.

```cpp
template <int L>
struct XorBasis {
  array<bitset<L>, L> data;
  bool insert(bitset<L> x) {
    for (int i = L - 1; i >= 0; i--) {
      if (!x[i]) continue;
      if (data[i].none()) return data[i] = x, true;
      x ^= data[i];
    }
    return false;
  }
  friend XorBasis operator&(XorBasis lhs, const XorBasis& rhs) {
    XorBasis ret;
    array<bitset<L>, L> buf{};
    for (int i = L - 1; i >= 0; i--) {
      auto x = rhs.data[i], y = x;
      for (int k = i; k >= 0 && x.any(); k--) {
        if (!x[k]) continue;
        if (lhs.data[k].none()) lhs.data[k] = x, buf[k] = y;
        x ^= lhs.data[k], y ^= buf[k];
      }
      ret.insert(y);
    }
    return ret;
  }
```

```
};
```

DATA STRUCTURE

dsu fasiz[i] 非负时表示 i 的父节点，否则表示 i 所在集合的大小的负值.

```C++
struct DSU {
  vector<int> fasiz;
  DSU(int n) : fasiz(n, -1) {}
  int find(int x) { return fasiz[x] < 0 ? x : fasiz[x] = find(fasiz[x]); }
  bool merge(int x, int y) {
    x = find(x), y = find(y);
    if (x == y) return false;
    if (-fasiz[x] < -fasiz[y]) swap(x, y);
    fasiz[x] += fasiz[y];
    fasiz[y] = x;
    return true;
  }
  bool same(int x, int y) { return find(x) == find(y); }
  int size(int x) { return -fasiz[find(x)]; }
};
```

dsu_short

```C++
struct DSU {
  vector<int> fa;
  DSU(int n) : fa(n) { iota(ALL(fa), 0); }
  int find(int x) { return fa[x] == x ? x : fa[x] = find(fa[x]); }
};
```

dynlzsgt

```C++
template <typename Info, typename Tag, void (*fn)(Info&, const Tag&)>
struct SegTree {
  struct Node {
    Info info = {};
    Tag tag = {};
    int ls = -1, rs = -1;
  };
  int n, _pos, _l, _r, root;
  Info _v;
  Tag _f;
  vector<Node> tree;
  SegTree(int n) : n(n) { root = alloc({}); }
  void set(int pos, Info v) { _pos = pos, _v = v, set(root, 0, n); }
  void apply(int l, int r, Tag f) { _l = l, _r = r, _f = f, apply(root, 0, n); }
  Info sum(int l, int r) { return _l = l, _r = r, sum(root, 0, n); }

 private:
  int alloc(Node node) { return tree.PUSHB(move(node)), int(tree.size()) - 1; }
  void update(int p, const Tag& f) { fn(tree[p].info, f), tree[p].tag += f; }
  void pushdown(int p) {
    update(tree[p].ls, tree[p].tag), update(tree[p].rs, tree[p].tag);
    tree[p].tag = {};
  }
  void set(int p, int pl, int pr) {
    if (pr - pl == 1) return tree[p].info = _v, tree[p].tag = {}, void();
    int ls = tree[p].ls, rs = tree[p].rs, mid = pl + (pr - pl) / 2;
    if (ls == -1) ls = tree[p].ls = alloc({});
    if (rs == -1) rs = tree[p].rs = alloc({});
    pushdown(p), _pos < mid ? set(ls, pl, mid) : set(rs, mid, pr);
    tree[p].info = tree[ls].info + tree[rs].info;
  }
  void apply(int p, int pl, int pr) {
    if (_r <= pl || pr <= _l || p == -1) return;
    if (_l <= pl && pr <= _r) return update(p, _f);
    int ls = tree[p].ls, rs = tree[p].rs, mid = pl + (pr - pl) / 2;
    if (ls == -1) ls = tree[p].ls = alloc({});
    if (rs == -1) rs = tree[p].rs = alloc({});
    pushdown(p), apply(ls, pl, mid), apply(rs, mid, pr);
    tree[p].info = tree[ls].info + tree[rs].info;
  }
  Info sum(int p, int pl, int pr) {
    if (_r <= pl || pr <= _l || p == -1) return {};
    if (_l <= pl && pr <= _r) return tree[p].info;
    int ls = tree[p].ls, rs = tree[p].rs, mid = pl + (pr - pl) / 2;
    if (ls == -1) ls = tree[p].ls = alloc({});
    if (rs == -1) rs = tree[p].rs = alloc({});
    return pushdown(p), sum(ls, pl, mid) + sum(rs, mid, pr);
  }
};
struct Info {
  ll sum = 0;
  int len = 0;
```

```
  friend Info operator+(const Info& lhs, const Info& rhs) {
    return {lhs.sum + rhs.sum, lhs.len + rhs.len};
  }
};
struct Tag {
  ll add = 0;
  Tag operator+=(const Tag& rhs) {
    add += rhs.add;
    return *this;
  }
};
void fn(Info& x, const Tag& f) { x.sum += x.len * f.add; }
```

dynsgt

```
template <typename Info>                                                    C++
struct SegTree {
  struct Node {
    Info info = {};
    int ls = -1, rs = -1;
  };
  int n, _pos, _l, _r, root;
  Info _v;
  vector<Node> tree;
  SegTree(int n) : n(n) { root = alloc({}); }
  void set(int pos, Info v) { _pos = pos, _v = v, set(root, 0, n); }
  Info sum(int l, int r) { return _l = l, _r = r, sum(root, 0, n); }

 private:
  int alloc(Node node) { return tree.PUSHB(move(node)), int(tree.size()) - 1; }
  void set(int p, int pl, int pr) {
    if (pr - pl == 1) return tree[p].info = _v, void();
    int ls = tree[p].ls, rs = tree[p].rs, mid = pl + (pr - pl) / 2;
    if (ls == -1) ls = tree[p].ls = alloc({});
    if (rs == -1) rs = tree[p].rs = alloc({});
    _pos < mid ? set(ls, pl, mid) : set(rs, mid, pr);
    tree[p].info = tree[ls].info + tree[rs].info;
  }
  Info sum(int p, int pl, int pr) {
    if (_r <= pl || pr <= _l || p == -1) return {};
    if (_l <= pl && pr <= _r) return tree[p].info;
    int ls = tree[p].ls, rs = tree[p].rs, mid = pl + (pr - pl) / 2;
    return sum(ls, pl, mid) + sum(rs, mid, pr);
  }
};
struct Info {
  int data = 2e9;
  friend Info operator+(const Info& lhs, const Info& rhs) {
    return {min(lhs.data, rhs.data)};
  }
};
```

ft

```
template <typename T>                                                       C++
struct FenwickTree {
  int n;
  vector<T> c;
  FenwickTree(int n) : n(n), c(n) {}
  void add(int pos, T x) {
    for (int i = pos + 1; i <= n; i += i & -i) c[i - 1] += x;
  }
  T pref(int r) const {
    T ret = 0;
    for (int i = r; i > 0; i -= i & -i) ret += c[i - 1];
    return ret;
  }
  T sum(int l, int r) const { return pref(r) - pref(l); }
};
```

lzsgt++ max_right 可能不可靠，目前验证方式有

- https://atcoder.jp/contests/abc426/tasks/abc426_f
- https://atcoder.jp/contests/abc389/tasks/abc389_f

```
template <typename Info, typename Tag, void (*fn)(Info&, const Tag&)>       C++
struct SegTree {
  int n, _l, _r;
  Info _acc;
  Tag _f;
  vector<Tag> tags;
  vector<Info> tree;
  SegTree(int n) : n(n), tags(4 * n), tree(4 * n) {}
  SegTree(const vector<Info>& a) : SegTree(a.size()) { build(1, 0, n, a); }
  void apply(int l, int r, Tag f) { _l = l, _r = r, _f = f, apply(1, 0, n); }
```

```
  Info sum(int l, int r) { return _l = l, _r = r, sum(1, 0, n); }
  template <typename P>
  int bsearch(int l, P pred) {
    return _acc = {}, _l = l, bsearch(1, 0, n, pred);
  }

 private:
  void update(int p, const Tag& f) { fn(tree[p], f), tags[p] += f; }
  void pushdown(int p) {
    int ls = p * 2, rs = p * 2 + 1;
    update(ls, tags[p]), update(rs, tags[p]), tags[p] = {};
  }
  void build(int p, int pl, int pr, const vector<Info>& a) {
    if (pr - pl == 1) return tree[p] = a[pl], void();
    int ls = p * 2, rs = p * 2 + 1, mid = pl + (pr - pl) / 2;
    build(ls, pl, mid, a), build(rs, mid, pr, a);
    tree[p] = tree[ls] + tree[rs];
  }
  void apply(int p, int pl, int pr) {
    if (_r <= pl || pr <= _l) return;
    if (_l <= pl && pr <= _r) return update(p, _f);
    int ls = p * 2, rs = p * 2 + 1, mid = pl + (pr - pl) / 2;
    pushdown(p), apply(ls, pl, mid), apply(rs, mid, pr);
    tree[p] = tree[ls] + tree[rs];
  }
  Info sum(int p, int pl, int pr) {
    if (_r <= pl || pr <= _l) return {};
    if (_l <= pl && pr <= _r) return tree[p];
    int ls = p * 2, rs = p * 2 + 1, mid = pl + (pr - pl) / 2;
    return pushdown(p), sum(ls, pl, mid) + sum(rs, mid, pr);
  }
  template <typename P>
  int bsearch(int p, int pl, int pr, P pred) {
    if (_l >= pr) return pr;
    if (_l <= pl) {
      if (pred(_acc + tree[p])) return _acc = _acc + tree[p], pr;
      if (pr - pl == 1) return pl;
    }
    int ls = p * 2, rs = p * 2 + 1, mid = pl + (pr - pl) / 2;
    pushdown(p);
    int qry = bsearch(ls, pl, mid, pred);
    return qry < mid ? qry : bsearch(rs, mid, pr, pred);
  }
};

struct Info {
  ll sum = 0;
  int len = 0;
  friend Info operator+(const Info& lhs, const Info& rhs) {
    return {lhs.sum + rhs.sum, lhs.len + rhs.len};
  }
};
struct Tag {
  ll add = 0;
  Tag operator+=(const Tag& rhs) {
    add += rhs.add;
    return *this;
  }
};
void fn(Info& x, const Tag& f) { x.sum += x.len * f.add; }
```

lzsgt

```
template <typename Info, typename Tag, void (*fn)(Info&, const Tag&)>
struct SegTree {
  int n, _l, _r;
  Info _acc;
  Tag _f;
  vector<Tag> tags;
  vector<Info> tree;
  SegTree(int n) : n(n), tags(4 * n), tree(4 * n) {}
  SegTree(const vector<Info>& a) : SegTree(a.size()) { build(1, 0, n, a); }
  void apply(int l, int r, Tag f) { _l = l, _r = r, _f = f, apply(1, 0, n); }
  Info sum(int l, int r) { return _l = l, _r = r, sum(1, 0, n); }

 private:
  void update(int p, const Tag& f) { fn(tree[p], f), tags[p] += f; }
  void pushdown(int p) {
    int ls = p * 2, rs = p * 2 + 1;
    update(ls, tags[p]), update(rs, tags[p]), tags[p] = {};
  }
  void build(int p, int pl, int pr, const vector<Info>& a) {
    if (pr - pl == 1) return tree[p] = a[pl], void();
```

C++

```
    int ls = p * 2, rs = p * 2 + 1, mid = pl + (pr - pl) / 2;
    build(ls, pl, mid, a), build(rs, mid, pr, a);
    tree[p] = tree[ls] + tree[rs];
  }
  void apply(int p, int pl, int pr) {
    if (_r <= pl || pr <= _l) return;
    if (_l <= pl && pr <= _r) return update(p, _f);
    int ls = p * 2, rs = p * 2 + 1, mid = pl + (pr - pl) / 2;
    pushdown(p), apply(ls, pl, mid), apply(rs, mid, pr);
    tree[p] = tree[ls] + tree[rs];
  }
  Info sum(int p, int pl, int pr) {
    if (_r <= pl || pr <= _l) return {};
    if (_l <= pl && pr <= _r) return tree[p];
    int ls = p * 2, rs = p * 2 + 1, mid = pl + (pr - pl) / 2;
    return pushdown(p), sum(ls, pl, mid) + sum(rs, mid, pr);
  }
};
struct Info {
  ll sum = 0;
  int len = 0;
  friend Info operator+(const Info& lhs, const Info& rhs) {
    return {lhs.sum + rhs.sum, lhs.len + rhs.len};
  }
};
struct Tag {
  ll add = 0;
  Tag operator+=(const Tag& rhs) {
    add += rhs.add;
    return *this;
  }
};
void fn(Info& x, const Tag& f) { x.sum += x.len * f.add; }
```

lzsgt4sl 针对扫描线特化的线段树，支持区间覆盖，撤销覆盖，查询被覆盖的总长度.

```
struct SegTree {
  int n, _l, _r, _f;
  const vector<int>& _a;
  vector<int> len, sum, tag;
  SegTree(const vector<int>& a)
      : n(a.size()), _a(a), len(4 * n), sum(4 * n), tag(4 * n) {
    build(1, 0, n);
  }
  void apply(int l, int r, int f) { _l = l, _r = r, _f = f, apply_(1, 0, n); }

 private:
  void build(int p, int pl, int pr) {
    if (pr - pl == 1) return len[p] = _a[pl], void();
    int ls = p * 2, rs = p * 2 + 1, mid = pl + (pr - pl) / 2;
    build(ls, pl, mid), build(rs, mid, pr);
    len[p] = len[ls] + len[rs];
  }
  void apply_(int p, int pl, int pr) {
    if (_r <= pl || pr <= _l) return;
    int ls = p * 2, rs = p * 2 + 1, mid = pl + (pr - pl) / 2;
    if (_l <= pl && pr <= _r) {
      sum[p] = (tag[p] += _f) ? len[p] : (pr - pl == 1 ? 0 : sum[ls] + sum[rs]);
      return;
    }
    apply_(ls, pl, mid), apply_(rs, mid, pr);
    sum[p] = tag[p] ? len[p] : sum[ls] + sum[rs];
  }
};
```

pssgt

```
template <typename Info>
struct SegTree {
  struct Node {
    Info info = {};
    int ls = -1, rs = -1;
  };
  int n, _pos, _l, _r;
  Info _v, _acc;
  vector<int> root;
  vector<Node> tree;
  SegTree(int n) : SegTree(vector<Info>(n)) {}
  SegTree(const vector<Info>& a) : n(a.size()) { root.PUSHB(build(0, n, a)); }
  void set(int p, int pos, Info f) {
    _pos = pos, _v = f, root.PUSHB(set_(p, 0, n));
  }
  Info sum(int p1, int p2, int l, int r) {
    return _l = l, _r = r, sum_(p1, p2, 0, n);
```

```
  }
  template <typename P>
  int bsearch(int p1, int p2, P pred) {
    return _acc = {}, bsearch_(p1, p2, 0, n, pred);
  }

 private:
#define MID (pl + (pr - pl) / 2)
  int new_(Node node) { return tree.PUSHB(move(node)), tree.size() - 1; }
  int pushup(int ls, int rs) {
    return new_(Node{tree[ls].info + tree[rs].info, ls, rs});
  }
  int build(int pl, int pr, const vector<Info>& a) {
    if (pr - pl == 1) return new_({a[pl], -1, -1});
    return pushup(build(pl, MID, a), build(MID, pr, a));
  }
  int set_(int p, int pl, int pr) {
    if (_pos < pl || pr <= _pos) return p;
    if (pr - pl == 1) return new_(Node{_v, -1, -1});
    return pushup(set_(tree[p].ls, pl, MID), set_(tree[p].rs, MID, pr));
  }
  Info sum_(int p1, int p2, int pl, int pr) {
    if (_r <= pl || pr <= _l) return {};
    if (_l <= pl && pr <= _r) return tree[p2].info - tree[p1].info;
    return sum_(tree[p1].ls, tree[p2].ls, pl, MID) +
           sum_(tree[p1].rs, tree[p2].rs, MID, pr);
  }
  template <typename P>
  int bsearch_(int p1, int p2, int pl, int pr, P pred) {
    if (pr - pl == 1)
      return pred(_acc + (tree[p2].info - tree[p1].info)) ? pr : pl;
    Info diff = tree[tree[p2].ls].info - tree[tree[p1].ls].info;
    if (pred(_acc + diff)) {
      _acc = _acc + diff;
      return bsearch_(tree[p1].rs, tree[p2].rs, MID, pr, pred);
    } else {
      return bsearch_(tree[p1].ls, tree[p2].ls, pl, MID, pred);
    }
  }
#undef MID
};
```

```
struct Info {
  int data = 0;
  friend Info operator+(const Info& lhs, const Info& rhs) {
    return {lhs.data + rhs.data};
  }
  friend Info operator-(const Info& lhs, const Info& rhs) {
    return {lhs.data - rhs.data};
  }
};
```

sgt

C++

```
template <typename Info>
struct SegTree {
  int n, _pos, _l, _r;
  Info _node;
  vector<Info> tree;
  SegTree(int n) : n(n), tree(4 * n) {}
  SegTree(const vector<Info>& a) : SegTree(a.size()) { build(1, 0, n, a); }
  void set(int pos, Info node) { _pos = pos, _node = node, set(1, 0, n); }
  Info sum(int l, int r) { return _l = l, _r = r, sum(1, 0, n); }

 private:
  void build(int p, int pl, int pr, const vector<Info>& a) {
    if (pr - pl == 1) return tree[p] = a[pl], void();
    int ls = p * 2, rs = p * 2 + 1, mid = pl + (pr - pl) / 2;
    build(ls, pl, mid, a), build(rs, mid, pr, a);
    tree[p] = tree[ls] + tree[rs];
  }
  void set(int p, int pl, int pr) {
    if (pr - pl == 1) return tree[p] = _node, void();
    int ls = p * 2, rs = p * 2 + 1, mid = pl + (pr - pl) / 2;
    _pos < mid ? set(ls, pl, mid) : set(rs, mid, pr);
    tree[p] = tree[ls] + tree[rs];
  }
  Info sum(int p, int pl, int pr) {
    if (_r <= pl || pr <= _l) return {};
    if (_l <= pl && pr <= _r) return tree[p];
    int ls = p * 2, rs = p * 2 + 1, mid = pl + (pr - pl) / 2;
    return sum(ls, pl, mid) + sum(rs, mid, pr);
  }
};
```

```cpp
struct Info {
  int data = 2e9;
  friend Info operator+(const Info& lhs, const Info& rhs) {
    return {min(lhs.data, rhs.data)};
  }
};
```

st

```cpp
template <typename T, typename Op>
struct SparseTable {
  int n, m;
  Op op;
  vector<vector<T>> data;
  SparseTable(vector<T> vec, Op op)
      : n(vec.size()), m(32 - __builtin_clz(n)), op(op), data(m, vector<T>(n)) {
    data[0] = move(vec);
    for (int j = 1; j < m; j++)
      for (int i = 0; i + (1 << j) <= n; i++)
        data[j][i] = op(data[j - 1][i], data[j - 1][i + (1 << (j - 1))]);
  }
  T query(int l, int r) const {
    int q = __lg(r - l);
    return op(data[q][l], data[q][r - (1 << q)]);
  }
};
```

GRAPH THEORY

dinic 时间复杂度：一般图 $\mathcal{O}(n^2m)$，二分图匹配 $\mathcal{O}(m\sqrt{n})$.

```cpp
template <typename T>
struct Dinic {
  struct Edge {
    int u, v;
    T cap, flow;
  };
  static constexpr T INF = numeric_limits<T>::max();
  int _s, _t;
  vector<int> cur, h;
  vector<vector<int>> G;
  vector<Edge> E;
  Dinic(int n) : cur(n), h(n), G(n) {}
  void adde(int u, int v, T cap) {
    G[u].PUSHB(E.size()), E.PUSHB({u, v, cap, 0});
    G[v].PUSHB(E.size()), E.PUSHB({v, u, 0, 0});
  }
  T flow(int s, int t) {
    _s = s, _t = t;
    T ret = 0;
    while (bfs()) {
      fill(ALL(cur), 0);
      ret += dfs(s, INF);
    }
    return ret;
  }

public:
  bool bfs() {
    fill(ALL(h), -1);
    queue<int> que;
    h[_s] = 0;
    que.push(_s);
    while (!que.empty()) {
      int u = que.front();
      que.pop();
      for (int i : G[u]) {
        const auto& [_, v, cap, flow] = E[i];
        if (h[v] != -1 || cap <= flow) continue;
        h[v] = h[u] + 1;
        que.push(v);
        if (v == _t) return true;
      }
    }
    return false;
  }
  T dfs(int u, T up) {
    if (u == _t) return up;
    T rest = up;
    for (int& i = cur[u]; i < int(G[u].size()); i++) {
      auto& [_, v, cap, flow] = E[G[u][i]];
      if (h[v] != h[u] + 1 || cap <= flow) continue;
      T f = dfs(v, min(rest, cap - flow));
      if (f <= 0) continue;
```

```
      flow += f;
      E[G[u][i] ^ 1].flow -= f;
      rest -= f;
      if (rest == 0) break;
    }
    return up - rest;
  }
};
```

ebcc 无向图的边双连通分量. 返回双联通分量个数，点对应的连通分量编号在 ebccid 中.

```
struct EBCC {
  int n, m = 0, cur_dfn, cur_ebcc;
  vector<char> isbridge;
  vector<int> dfn, low, ebccid;
  vector<vector<pii>> G;
  EBCC(int n) : n(n), dfn(n, -1), low(n), ebccid(n, -1), G(n) {}
  void adde(int u, int v) { G[u].PUSHB({v, m}), G[v].PUSHB({u, m++}); }
  int work() {
    isbridge.assign(m, 0);
    for (int i = 0; i < n; i++)
      if (dfn[i] == -1) tarjan(i, -1);
    for (int i = 0; i < n; i++)
      if (ebccid[i] == -1) ebccid[i] = cur_ebcc++, dfs(i);
    return cur_ebcc;
  }

 private:
  void tarjan(int u, int fa_e) {
    dfn[u] = low[u] = cur_dfn++;
    for (const auto& [v, e] : G[u]) {
      if (dfn[v] == -1) {
        tarjan(v, e), low[u] = min(low[u], low[v]);
        if (low[v] > low[u]) isbridge[e] = 1;
      } else if (e != fa_e) {
        low[u] = min(low[u], dfn[v]);
      }
    }
  }
  void dfs(int u) {
    for (const auto& [v, e] : G[u]) {
      if (ebccid[v] != -1 || isbridge[e]) continue;
      ebccid[v] = ebccid[u], dfs(v);
    }
  }
};
```

euler 无向图欧拉回路或路径，时间复杂度 $\mathcal{O}(n+m)$.

```
struct Euler {
  int n, m = 0;
  vector<char> used;
  vector<int> cur_e, path, epath;
  vector<vector<pii>> G;
  Euler(int n) : n(n), cur_e(n), G(n) {}
  void adde(int u, int v) { G[u].PUSHB({v, m}), G[v].PUSHB({u, m++}); }
  int work() {
    if (m == 0) return path.PUSHB(0), 0;
    used.assign(m, 0);
    vector<int> odd, even;
    for (int i = 0; i < n; i++)
      if (!G[i].empty()) G[i].size() & 1 ? odd.PUSHB(i) : even.PUSHB(i);
    if (odd.size() == 1 || odd.size() > 2) return -1;
    odd.empty() ? dfs(even[0]) : dfs(odd[0]);
    reverse(ALL(path)), reverse(ALL(epath));
    return int(path.size()) == m + 1 ? 0 : -1;
  }

 private:
  void dfs(int u) {
    for (int& i = cur_e[u]; i < int(G[u].size());) {
      auto [v, e] = G[u][i++];
      if (used[e]) continue;
      used[e] = 1, dfs(v), epath.PUSHB(e);
    }
    path.PUSHB(u);
  }
};
```

hld

```
#include "../ds/sgt.hpp"
struct HLD {
```

```cpp
struct Info {
  ll data = 0;
  friend Info operator+(Info lhs, Info rhs) { return {lhs.data + rhs.data}; }
};
SegTree<Info> sgt = 0;
int n, cur_dfn;
vector<int> fa, dep, siz, hson;
vector<int> dfn, rdfn, top;
vector<vector<int>> G;
HLD(int n)
    : n(n), fa(n, -1), dep(n), siz(n), hson(n, -1), dfn(n), rdfn(n), top(n),
      G(n) {}
void adde(int u, int v) { G[u].PUSHB(v), G[v].PUSHB(u); }
void init_dfn(int root) { cur_dfn = 0, dfs1(root, -1), dfs2(root, root); }
void init_sgt(const vector<int>& a) {
  vector<Info> b(n);
  for (int i = 0; i < n; i++) b[dfn[i]].data = a[i];
  sgt = SegTree(b);
}
int lca(int u, int v) const {
  while (top[u] != top[v]) {
    if (dep[top[u]] < dep[top[v]]) swap(u, v);
    u = fa[top[u]];
  }
  return dep[u] < dep[v] ? u : v;
}
void add(int p, int x) {
  Info qry = sgt.sum(dfn[p], dfn[p] + 1);
  qry.data += x;
  sgt.set(dfn[p], qry);
}
ll query(int u, int v) {
  Info ret;
  while (top[u] != top[v]) {
    if (dep[top[u]] < dep[top[v]]) swap(u, v);
    ret = ret + sgt.sum(dfn[top[u]], dfn[u] + 1);
    u = fa[top[u]];
  }
  if (dfn[u] > dfn[v]) swap(u, v);
  ret = ret + sgt.sum(dfn[u], dfn[v] + 1);
  return ret.data;
  }

 private:
  void dfs1(int u, int fa_) {
    fa[u] = fa_;
    siz[u] = 1;
    for (int v : G[u]) {
      if (v == fa_) continue;
      dep[v] = dep[u] + 1;
      dfs1(v, u);
      siz[u] += siz[v];
      if (hson[u] == -1 || siz[v] > siz[hson[u]]) hson[u] = v;
    }
  }
  void dfs2(int u, int top_) {
    dfn[u] = cur_dfn;
    rdfn[cur_dfn++] = u;
    top[u] = top_;
    if (hson[u] != -1) dfs2(hson[u], top_);
    for (int v : G[u])
      if (v != fa[u] && v != hson[u]) dfs2(v, v);
  }
};
```

lca 时间复杂度：预处理 $\mathcal{O}(n \log n)$；查询 $\mathcal{O}(1)$

C++

```cpp
struct LCA {
  int n, m, cur_dfn = 0;
  vector<int> dfn, dep;
  vector<vector<int>> G, st;
  LCA(int n)
      : n(n), m(32 - __builtin_clz(n)), dfn(n, -1), dep(n), G(n),
        st(m, vector<int>(n)) {}
  void adde(int u, int v) { G[u].PUSHB(v), G[v].PUSHB(u); }
  void work(int root) {
    dfs(root, root);
    for (int j = 1; j < m; j++)
      for (int i = 0; i + (1 << j) <= n; i++)
        st[j][i] = op4st(st[j - 1][i], st[j - 1][i + (1 << (j - 1))]);
  }
  int query(int u, int v) const {
    if (u == v) return u;
```

```
    if ((u = dfn[u]) > (v = dfn[v])) swap(u, v);
    int q = __lg(v - u);
    return op4st(st[q][u + 1], st[q][v + 1 - (1 << q)]);
  }

 private:
  int op4st(int u, int v) const { return dfn[u] < dfn[v] ? u : v; }
  void dfs(int u, int fa_) {
    st[0][dfn[u] = cur_dfn++] = fa_;
    for (int v : G[u])
      if (v != fa_) dep[v] = dep[u] + 1, dfs(v, u);
  }
};
```

primaldual 时间复杂度：$\mathcal{O}(mf \log n)$.

C++

```
template <typename T>
struct PrimalDual {
  struct Edge {
    int u, v;
    T cap, cost, flow;
  };
  static constexpr T INF = numeric_limits<T>::max();
  int n, _s, _t;
  vector<int> prev;
  vector<T> h, dist;
  vector<vector<int>> G;
  vector<Edge> E;
  PrimalDual(int n) : n(n), prev(n), h(n), dist(n), G(n) {}
  void adde(int u, int v, T cap, T cost) {
    G[u].PUSHB(E.size()), E.PUSHB({u, v, cap, cost, 0});
    G[v].PUSHB(E.size()), E.PUSHB({v, u, 0, -cost, 0});
  }
  pair<T, T> flow(int s, int t, T maxf = INF) {
    _s = s, _t = t;
    bellman_ford();
    T maxflow = 0, mincost = 0;
    while (maxflow < maxf && dijkstra()) {
      T aug = maxf - maxflow;
      for (int i = 0; i < n; i++)
        if (dist[i] != INF) h[i] += dist[i];
      for (int p = _t; p != _s; p = E[prev[p] ^ 1].v)
        aug = min(aug, E[prev[p]].cap - E[prev[p]].flow);
      for (int p = _t; p != _s; p = E[prev[p] ^ 1].v)
        E[prev[p]].flow += aug, E[prev[p] ^ 1].flow -= aug;
      maxflow += aug, mincost += aug * h[_t];
    }
    return {maxflow, mincost};
  }

private:
 void bellman_ford() {
   queue<int> que;
   vector<char> inque(n);
   fill(ALL(h), INF);
   h[_s] = 0;
   inque[_s] = 1;
   que.push(_s);
   while (!que.empty()) {
     int u = que.front();
     que.pop();
     inque[u] = 0;
     for (int i : G[u]) {
       const auto& [_, v, cap, cost, flow] = E[i];
       if (cap <= flow || h[v] <= h[u] + cost) continue;
       h[v] = h[u] + cost;
       if (!inque[v]) {
         inque[v] = 1;
         que.push(v);
       }
     }
   }
 }
 bool dijkstra() {
   priority_queue<pair<T, int>, vector<pair<T, int>>, greater<>> pq;
   fill(ALL(dist), INF);
   dist[_s] = 0;
   pq.push({0, _s});
   while (!pq.empty()) {
     auto [d, u] = pq.top();
     pq.pop();
     if (dist[u] != d) continue;
     for (int i : G[u]) {
```

```
        auto& [_, v, cap, cost, flow] = E[i];
        if (cap <= flow || dist[v] <= d + cost + h[u] - h[v]) continue;
        dist[v] = d + cost + h[u] - h[v];
        prev[v] = i;
        pq.push({dist[v], v});
      }
    }
    return dist[_t] != INF;
  }
};
```

scc 有向图的强连通分量．返回强联通分量个数，点对应的强连通分量编号在 sccid 中．

C++
```
struct SCC {
  int n, cur_dfn, cur_scc;
  stack<int> stk;
  vector<int> dfn, low, sccid;
  vector<vector<int>> G;
  SCC(int n) : n(n), dfn(n, -1), low(n), sccid(n, -1), G(n) {}
  void adde(int u, int v) { G[u].PUSHB(v); }
  int work() {
    cur_dfn = cur_scc = 0;
    for (int i = 0; i < n; i++)
      if (dfn[i] == -1) tarjan(i);
    assert(stk.empty());
    return cur_scc;
  }

 private:
  void tarjan(int u) {
    dfn[u] = low[u] = cur_dfn++, stk.push(u);
    for (int v : G[u]) {
      if (dfn[v] == -1) {
        tarjan(v), low[u] = min(low[u], low[v]);
      } else if (sccid[v] == -1) {
        low[u] = min(low[u], dfn[v]);
      }
    }
    if (dfn[u] == low[u]) {
      int v;
      do {
        v = stk.top(), stk.pop(), sccid[v] = cur_scc;
      } while (v != u);
      cur_scc++;
    }
  }
};
```

twosat 2-SAT 问题：给定 n 个布尔变量，与 m 个形式为 $(x_i = f) \vee (x_j = g)$ 的条件．如果有解答案构造在 ans 中．

C++
```
#include "scc.hpp"
struct TwoSat {
  int n;
  vector<char> ans;
  SCC scc;
  TwoSat(int n) : n(n), ans(n), scc(2 * n) {}
  void adde(int i, bool f, int j, bool g) {
    scc.adde(2 * i + !f, 2 * j + g), scc.adde(2 * j + !g, 2 * i + f);
  }
  bool work() {
    scc.work();
    const auto& sccid = scc.sccid;
    for (int i = 0; i < n; i++) {
      if (sccid[2 * i] == sccid[2 * i + 1]) return false;
      ans[i] = sccid[2 * i] > sccid[2 * i + 1];
    }
    return true;
  }
};
```

vbcc 无向图的点双连通分量．返回双联通分量个数，点对应的连通分量在 vbccs 中．

C++
```
struct VBCC {
  int n, m = 0, cur_dfn;
  vector<char> iscut;
  stack<int> stk;
  vector<int> dfn, low;
  vector<vector<int>> vbccs;
  vector<vector<pii>> G;
  VBCC(int n) : n(n), iscut(n), dfn(n, -1), low(n), G(n) {}
  void adde(int u, int v) { G[u].PUSHB({v, m}), G[v].PUSHB({u, m++}); }
  int work() {
    for (int i = 0; i < n; i++)
```

```
      if (dfn[i] == -1) tarjan(i, -1), stk.pop();
    assert(stk.empty());
    return vbccs.size();
  }

 private:
  void tarjan(int u, int fa_e) {
    dfn[u] = low[u] = cur_dfn++, stk.push(u);
    int sons = 0, flag = 0, tmp;
    for (const auto& [v, e] : G[u]) {
      if (dfn[v] == -1) {
        tarjan(v, e), sons++, low[u] = min(low[u], low[v]);
        if (low[v] >= dfn[u]) {
          flag++, vbccs.PUSHB({});
          if (fa_e != -1 || flag > 1) iscut[u] = 1;
          do {
            tmp = stk.top(), stk.pop(), vbccs.rbegin()->PUSHB(tmp);
          } while (tmp != v);
          vbccs.rbegin()->PUSHB(u);
        }
      } else if (e != fa_e) {
        low[u] = min(low[u], dfn[v]);
      }
    }
    if (fa_e == -1 && sons == 0) vbccs.PUSHB({u});
  }
};
```

MISCELLANEOUS

i128out

```
ostream& operator<<(ostream& out, i128 x) {
  if (x == 0) return out.put('0');
  if (x < 0) out.put('-'), x = -x;
  static char buf[40];
  int ptr = 40;
  while (x) buf[--ptr] = '0' + x % 10, x /= 10;
  return out.write(buf + ptr, 40 - ptr);
}
```

inthash

```
struct IntHash {
  // http://xorshift.di.unimi.it/splitmix64.c
  static uint64_t hash(uint64_t x) {
    x += 0x9e3779b97f4a7c15;
    x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
    x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
    return x ^ (x >> 31);
  }
  inline static const uint64_t SEED =
      chrono::steady_clock::now().time_since_epoch().count();
  size_t operator()(uint64_t x) const { return hash(x + SEED); }
  size_t operator()(pair<uint64_t, uint64_t> x) const {
    return hash(x.first + SEED) ^ (hash(x.second + SEED) >> 1);
  }
};
```

read

```
#define GC ch = getchar_unlocked()
ll read() {
  ll x = 0, f = 1, GC;
  while (ch < '0' || ch > '9') ch == '-' ? f = -1, GC : GC;
  while (ch >= '0' && ch <= '9') x = x * 10 + ch - '0', GC;
  return x * f;
}
#undef GC
```