## 1. Неземні огірки

Позначимо через $c_j(k)$ кількість рослин на $j$-му році життя через $k$ років після виверження. Маємо:

(1) $c_1(1) = 1$;
(2) $c_2(1) = 0$ при $j = 2, 3, \ldots, n$;
(3) $c_1(k+1) = a_1 c_1(k) + a_2 c_2(k) + \ldots + a_n c_n(k)$;
(4) $c_j(k+1) = c_{j-1}(k)$.

Запровадимо позначення:

- $c(k)$ для вектора-*стовпчика* $(c_1(k), c_2(k), \ldots, c_n(k))^t$. Тут літера $t$ позначає операцію транспонування, що перетворює рядок у стовпчик;
- $A$ для матриці (прямокутної таблиці чисел) такого вигляду:

| $a_1$ | $a_2$ | $a_3$ | $a_4$ | … | $a_{n-1}$ | $a_n$ |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | … | 0 | 0 |
| 0 | 1 | 0 | 0 | … | 0 | 0 |
| 0 | 0 | 1 | 0 | … | 0 | 0 |
| 0 | 0 | 0 | 1 | … | 0 | 0 |
| ..... | | | | | | |
| 0 | 0 | 0 | 0 | … | 1 | 0 |

Нагадаємо означення *множення* матриць. Використаємо позначення $x_{jk}$, $y_{jk}$ та $z_{jk}$ для елементів відповідно у рядку $j$ та у стовпчику $k$ матриць $X$, $Y$ та $Z$, пов'язаних рівністю $X=Y\cdot Z$. Маємо:

$$x_{jk} = y_{j1} z_{1k} + y_{j2} z_{2k} + y_{j3} z_{3k} + \ldots + y_{jn} z_{nk},$$

де $n$ — кількість стовпчиків матриці $Y$, що збігається з кількістю рядків матриці $Z$ (інакше дію множення матриць невизначено).

З означення дії множення матриць випливає асоціативний (сполучний) закон: для довільних матриць $X$, $Y$, $Z$, які можна послідовно перемножити, справджується рівність: $(X\cdot Y)\cdot Z = X\cdot(Y\cdot Z)$.

Рекурентні співвідношення (3–4) можна записати у матричному вигляді:

(5) $c(k+1) = A\cdot c(k) = A^2\cdot c(k-1) = \ldots = A^k \cdot c(1)$.

Безпосереднє використання рекурентних співвідношень (3–4) породжує алгоритм з ефективністю $O(m)$. Спробу його реалізації очікувана для більшості учасників олімпіади. Але від переможців можна сподіватися використання ефективного алгоритму піднесення до степеня матриці (5) з ефективністю $O(\log_2 m)$:

- знайти степені матриці $A$, що є степенями 2: $A^2 = A\cdot A$; $A^4 = A^2\cdot A^2$; $A^8 = A^4\cdot A^4$, …;
- знайти цифри двійкового запису числа $m-1 = \alpha_0 + 2\alpha_1 + 2^2\alpha_2 + 2^3\alpha_3 + \ldots$, де $\alpha_0$, $\alpha_1$, $\alpha_2$, $\alpha_3$, … набувають значень 0 або 1;
- обчислити $A^{(m-1)} = (A)^{\alpha_0} \cdot (A^2)^{\alpha_1} \cdot (A^4)^{\alpha_2} \cdot (A^8)^{\alpha_3} \cdot \ldots$ . Тут множення на нульовий степінь означає відсутність множення або (що еквівалентно) множенню на матрицю, всі елементи якої дорівнюють 0 крім, тих, які розташовані на головній діагоналі (номер рядка дорівнює номеру стовпчика) і які дорівнюють 1.

У всіх перелічених кроках відповідні суму й добутки обривають на найстаршому розряді двійкового запису числа $(m-1)$.

Щоб уникнути виходу за межі базового типу усі арифметичні дії потрібно виконувати над лишками від ділення на найменше спільне кратне ємностей контейнерів, а у вихідний файл записати лишки від ділення координат вектора $c(m)$ на відповідні ємності контейнерів.

Для того, щоб була можливість перевірити коректність 48 тестів з 50 без створення розв'язання, при створенні 48 тестів використано послідовність чисел Фібоначчі та геометричну прогресію.

## 2. Дільники

Існує очевидне *часткове* розв'язання: перебираючи у певному діапазоні у порядку зростання натуральні числа $j$, кратні $m$, робити таке:

- **встановити, чи обране кратне має лише такі прості дільники, що має $m$.** Для цього повторити заміну значення $k$ на $k$/НСД($k$,$m$) до тих пір, поки не буде справджуватися хоча б одна з рівностей: $k = 1$ або НСД($k$,$m$) = 1. Тут НСД($k$,$m$) — найбільший спільний дільник чисел $k$ і $m$, який можна знайти, використавши алгоритм Евкліда. Справдження рівності: $k = 1$ (після згаданих перетворень) еквівалентно відсутності «сторонніх» простих дільників;
- **у разі ствердної відповіді щодо відсутності «сторонніх» простих дільників знайти кількість всіх натуральних дільників,** обчисливши лишки від ділення на всі натуральні числа, що менші від розглядуваного числа, кратного $m$.

Зауважимо, що порядок перевірки умов істотно впливає на кількість виконуваних дій (вище подано оптимальний варіант). Так само, як і діапазон, у якому здійснюють пошук.

*Авторське* розв'язання використовує розклад чисел $m$ і $n$ на прості множники й формулу для кількості всіх натуральних дільників натурального числа з відомим розкладом на прості множники. У 2017 році на III (міському) етапі олімпіади з інформатики було запропоновано задачу 4 «Дільники» (істотно простішу за дану), повне розв'язання якої також вимагає використання поданої нижче формули (2).

Позначимо через:

- $l_2$, $l_3$, $l_5$, … — степені простих чисел 2, 3, 5, … у канонічному розкладі на прості множники числа $m$ (рівність $l_j = 0$ означає, що $m$ не кратне простому числу $j$);
- $k_2$, $k_3$, $k_5$, … — степені простих чисел 2, 3, 5, … у канонічному розкладі на прості множники шуканого числа (рівність $k_j = 0$ означає, що шукане число не кратне $j$).

Згідно з умовою задачі маємо:

(1) $0 \le l_j \le k_j$ при $j$ =2, 3, 5, … , і лише у скінченій кількості нерівностей $0 < l_j$;
(2) $n = (k_2 + 1)(k_3 + 1)(k_5 + 1) \ldots$ , і лише у скінченій кількості множників $0 < k_j$.

Позначимо через:

- $c(x)$ — кількість *різних простих* дільників натурального числа $x$,
- $q(x)$ — кількість *простих* дільників натурального числа $x$ *з урахуванням кратності* входження у канонічний розклад на прості множники. Наприклад, $c(12) = 2$, $q(12) = 3$.

Маємо:

- **при $q(n) < c(m)$** розв'язків немає, бо неможливо розподілити $q(n)$ додатних степенів серед $c(m)$ множників. Цей випадок проілюстровано першим рядком прикладу вхідних даних в умові;
- **при $q(n) = c(m)$** кількість розв'язків не перевищує $c(m)!$. У цьому випадку шукані числа відрізняються один від одного перестановкою степенів при простих числах —

дільниках $m$ — у розкладі на прості множники з урахуванням нерівностей (1). Цей випадок проілюстровано другим рядком прикладу вхідних даних в умові завдання: $q(n) = c(m) = 2$, $l_3 = l_5 = 1$, а решта значень $l_j$ дорівнюють нулю;

- **при $q(n) > c(m)$** шукані числа можна розбити на множини, елементи кожної з яких відрізняються один від одного перестановкою степенів при простих числах — дільниках $m$ — у розкладі на прості множники з урахуванням нерівностей (1). При цьому добуток усіх степенів, *збільшених на* 1, дорівнює $n$. Цей випадок проілюстровано третім рядком у прикладі вхідних даних в умові завдання: $q(n) = 3$, $c(m) = 2$, $l_3 = l_5 = 1$, а решта значень $l_j$ дорівнюють нулю.

**Схема перебору** для знаходження лінійного масиву шуканих чисел така: розглянути у порядку зростання прості дільники $m$ і для кожного з них у порядку спадання розглянути степені простих дільників $n$ у розкладі степеня простого дільника $m$. Така оптимізована схема перебору передбачає обчислення й використання таких масивів (лінійних крім останнього у переліку):

- прості дільники $m$ і $n$;
- степені простих дільників $m$ і $n$;
- степені простих дільників $n$, ще нерозподілені при переборі;
- часткові добутки числа відповіді;
- степені простих дільників $m$ відповіді;
- степені простих дільників $n$ у розкладі на прості множники степенів простих дільників $m$, збільшених на 1 (двовимірний масив).

$2^{64}$ — верхня межа базового типу для невід'ємного цілого числа з найбільшою кількістю цифр. Тому згідно з вимогами до вхідних та вихідних файлів такі масиви не можуть містити більше ніж 64 елементи, а степені простих дільників у розкладах не можуть перевищувати 64.

При визначенні степенів простих дільників $n$ у розкладі на прості множники степеня найбільшого простого дільника $m$, збільшеного на 1 не потрібно влаштовувати перебір значень, як для менших простих дільників $m$: потрібно запозичити значення з масиву степенів простих дільників $n$, ще нерозподілених при переборі. Зміст цього зауваження стане повністю зрозумілим після розгляду програми програмного втілення — авторського розв'язання.

Схема перебору вимагає *втілення однією групою операторів довільної наперед невідомої кількості повторень*. Це можна програмно втілити або використавши рекурсивну процедуру (функцію), або використавши мітки. Невелика розмірність масивів означає, що переваги від використання міток будуть непомітні. Тому авторське розв'язання (з педагогічних міркувань) використовує рекурсивну процедуру step($j$, $k$), де:
$j$ — номер простого дільника $m$;
$k$ — номер простого дільника $n$.

Після заповнення лінійного масиву потрібними числами його потрібно упорядкувати за зростанням (наприклад, швидким методом Хоара) і вивести значення його елементів в окремий рядок вихідного файлу.

## 3. Інкасація

Позначимо через $f(n,k)$ найменший необхідний час (у секундах) при наявності $n$ банкнот, можливості здати фальшиву банкноту *ще* $k$ разів, часі успішної інкасації — $A$, часі неуспішної інкасації — B (секунд). Маємо:

$$f(1, k) = 0,$$
$$f(n, 1) = (n - 2)\cdot A + B \text{ при } n > 1.$$

Що трапиться, якщо ми віднесемо $m$ банкнот ($1 \le m < n$) у банк? Можливо такі 2 випадки:

1. Банк прийняв банкноти. Тоді:
   - витрачено A секунд;
   - фальшива банкнота серед решти ($n - m$) банкнот;
   - залишилося ще $k$ спроб здати фальшиву банкноту.

   У цьому випадку щоб знайти фальшивку серед решти ($n - m$) банкнот знадобиться ще $f(n - m, k)$ секунд. Загальний час (у секундах) у цьому випадку складе:

$$g(m) = A + f(n - m, k).$$

2. Банк не прийняв банкноти. Тоді
   - витрачено $B$ секунд;
   - встановлено, що фальшива банкнота — серед повернутих $m$ банкнот.
   - Залишилося ще ($k - 1$) спроба здати фальшиву банкноту.

   У цьому випадку щоб знайти фальшивку знадобиться ще $f(m, k - 1)$ секунд. Значить загальний час (у секундах) в цьому випадку складе:

$$h(m) = B + f(m, k - 1).$$

У найгіршому випадку потрібно буде $\max(g(m), h(m))$ секунд.

Як знайти $m$? Можна перебрати всі можливі значення $m = 1, 2, \ldots, n - 1$. Таку розв'язання має оцінкою часу $O(kn^2)$. Це занадто багато.

Зауважимо: $f(n, k)$ не спадає при зростанні $n$. Це інтуїтивно зрозуміло — для виявлення фальшивки серед більшої кількості банкнот часу знадобиться не менше. Отже, значення $h(m)$ не спадає, а значення $g(m)$ не зростає при зростанні $m$.

Нам треба знайти таке натуральне $m$, що мінімізує $\max(g(m), h(m))$ при $1 \le m < n$. Нагадаємо: $h$ — не спадає, $g$ — не зростає. Тому шукане значення $m$ можна знайти двійковим поділом діапазону значень $m$. Загалом це потребуватиме $O(kn \log n)$ часу.

Розглянемо детальніше такі 3 випадки:

1. $h(1) \ge g(1)$ — графік $h$ розташовано вище графіка $g$. Внаслідок монотонності функцій маємо: $h(m) \ge g(m)$ при усіх можливих $m$. У цьому випадку $\max(g(m), h(m)) = h(m)$, а оптимальним буде таке $m$, що мінімізує $h(m)$. Функція $h$ — не спадає, тому нам потрібно мінімально можливе значення $m$, тобто 1.
2. $h(n - 1) \le g(n - 1)$ — графік $g$ розташовано вище графіка $h$. Внаслідок монотонності функцій маємо: $h(m) \le g(m)$ для всіх можливих $m$. Це означає, що $\max(g(m), h(m)) = g(m)$, а оптимальним буде таке $m$, що мінімізує $g(m)$. Функція $g$ — не зростає, тому нам потрібно максимальне можливе значення $m$, тобто $n - 1$.
3. $h(1) < g(1)$ і $g(n - 1) < h(n - 1)$ — графік $h$ «перетинає» графік $g$. Розглянемо функцію $h(m) - g(m)$. Вона не спадає, при $m = 1$ вона менша від 0, при $m = n - 1$ вона більша 0. Отже, її графік десь «перетинає» рівень 0. Двійковим пошуком (поділом діапазону значень навпіл) ми можемо знайти, де саме це відбувається. І «поруч» з цим значенням і буде шукане оптимальне значення $m$.

**Примітка**. Тут «перетинає» записано в лапках, бо $g$ і $h$ — функції *цілого*, а не *дійсного* аргументу. Під перетином рівня 0 у точці $x$ розуміємо, що або при значенні аргументу $x$ функція набуває значення 0, або що в точці $x$ вона буде від'ємна, а в $x + 1$ додатна. В

останньому випадку шуканим значенням буде або $x$, або $x + 1$. А саме те, яке мінімізує значення $\max(g(m), h(m))$.